

Android



开发从入门到精通

扶松柏 编著



- 由多年从事Android平台应用软件分析、设计、编码和测试的一线工程师编著
- 从理论知识着手，引领读者搭建开发环境，逐步深入项目开发实战，最终独立完成软件开发、测试、发布工作
- 涵盖界面布局设计、基本控件设计、数据存储设计、通信设计、自动服务设计、互联网应用设计、多媒体设计、Google地图开发、Google API应用、游戏项目实战案例等Android开发最重要的内容



1 CD

随书赠送1CD，内含本书所有程序的源代码，读者在学习过程中可以随时调用、运行，也可以根据实际需要稍加改动，应用到自己的实际项目中。

兵器工业出版社



北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

- android与iphone及ipad开发书籍 持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等 持续不断更新中.....
- delphi《书籍》及《视频》教程 持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修炼班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程（个人博客系统49课全），外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班，php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库 ←

文件名 <ul style="list-style-type: none"> <input type="checkbox"/> 办公类软件使用指南 <input type="checkbox"/> 医学 <input type="checkbox"/> 历史人物传记 <input type="checkbox"/> 哲学宗教 <input type="checkbox"/> 外语资料（除英语外）（除英语外） <input type="checkbox"/> 官场类小说 <input type="checkbox"/> 建筑工程类 <input type="checkbox"/> 情感生活类小说 <input type="checkbox"/> 政治军事 <input type="checkbox"/> 教育学习科普大全 网址：WLSAM168.400GB.COM <input type="checkbox"/> 文学理论 <input type="checkbox"/> 智力开发、增强记忆、快速阅读技巧大全 <input type="checkbox"/> 社会生活 <input type="checkbox"/> 科学技术 <input type="checkbox"/> 程序编程类 <input type="checkbox"/> 经济管理 <input type="checkbox"/> 网络安全及管理 <input type="checkbox"/> 网赚系列 <input type="checkbox"/> 美食小吃烹饪煲汤大全 <input type="checkbox"/> 课外读物 	P D F 电子书专题库，内容详尽，每天不断更新！！ <p>本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习，如用于商业或非法用途的后果自负！</p>
--	--

OE Foxit PDF Editor ±à¼-ºæ·É·ÉùÓÐ (c) by Foxit Software Company, 2004 VIP培训教程，易语言黑月VIP视频教程，网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)

- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等 持续不断更新中....
- 网建设相关的pdf书籍及各种vip视频教程 持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价营销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等 持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。
- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 >

总策划：北京希望电子出版社
责任编辑：刘立 李萌
封面设计：深度文化



Android 开发从入门到精通

本书通过具体实例，详细地讲解了Android技术的具体应用和实现过程。本书内容新颖、知识全面、讲解详细，全书分为12章。第1章走进Android世界，简要讲解了理论知识和搭建开发环境；第2章界面布局实战演练，讲解了实现界面布局的典型实例的实现过程；第3章基本控件实战演练，讲解了Android基本控件典型实例的实现过程；第4章数据存储实战演练，讲解了数据存储方面典型实例的实现过程；第5章通信领域实战演练，讲解了通信领域典型实例的实现过程；第6章自动服务实战演练，讲解了自动服务方面典型实例的实现过程；第7章互联网实战演练，讲解了互联网领域典型实例的实现过程；第8章多媒体实战演练，讲解了多媒体典型实例的实现过程；第9章Google地图实战演练，讲解了Google地图典型实例的实现过程；第10章Google API实战演练，讲解了主流Google API典型实例的实现过程；第11章游戏实战演练，讲解了Android在游戏领域典型实例的实现过程；第12章优化和发布项目，讲解了优化和发布Android项目的实现过程。

本书定位于Android的中高级用户，还可以作为向此领域发展的程序员的参考书。

需要本书或技术支持的读者，请与北京清河6号信箱（邮编100085）发行部联系

发行部电话：010-62978181(总机)转发行部，010-82702675(邮购)
技术支持电话：010-62978181转535
010-62978181转522(数字生活)

北京希望电子出版社网址：www.bhp.com.cn
传 真：010-82702698
E-mail：tbd@bhp.com.cn
投 稿：wuty@bhp.com.cn

分类建议：
计算机/程序设计/Android

ISBN 978-7-80248-694-2



9 787802 486942 >

定价：69.00元 (配1张CD光盘)

Android 开发从入门到精通

扶松柏 编著

兵器工业出版社



北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

PDG

内 容 简 介

本书详细地讲解了 Android 技术的具体应用和实现过程。本书内容新颖、知识全面、讲解详细。全书分为 12 章，第 1 章走进 Android 世界，简要讲解了理论知识和搭建开发环境；第 2 章界面布局实战演练，讲解了实现界面布局的典型实例的实现过程；第 3 章基本控件实战演练，讲解了 Android 基本控件典型实例的实现过程；第 4 章数据存储实战演练，讲解了数据存储方面典型实例的实现过程；第 5 章通信领域实战演练，讲解了通信领域典型实例的实现过程；第 6 章自动服务实战演练，讲解了自动服务方面典型实例的实现过程；第 7 章互联网实战演练，讲解了互联网领域典型实例的实现过程；第 8 章多媒体实战演练，讲解了多媒体典型实例的实现过程；第 9 章 Google 地图实战演练，讲解了 Google 地图典型实例的实现过程；第 10 章 Google API 实战演练，讲解了主流 Google API 典型实例的实现过程；第 11 章游戏实战演练，讲解了 Android 在游戏领域典型实例的实现过程；第 12 章优化和发布项目，讲解了优化 Android 项目和发布 Android 作品典型实例的实现过程。本书光盘中提供了部分程序的源代码。

本书定位于 Android 的中高级用户，还可以作为向此领域发展的程序员的参考书。

图书在版编目 (CIP) 数据

Android 开发从入门到精通 / 扶松柏编著. -- 北京：
兵器工业出版社, 2012.1
ISBN 978-7-80248-694-2
I. ①A… II. ①扶… 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(2011)第 242348 号

出版发行：兵器工业出版社 北京希望电子出版社
邮编社址：100089 北京市海淀区车道沟 10 号
100085 北京市海淀区上地 3 街 9 号
金隅嘉华大厦 C 座 611
电 话：010-62978181（总机）转发行部
010-82702675（邮购）010-82702698（传真）
经 销：各地新华书店 软件连锁店
印 刷：北京瑞富峪印务有限公司
版 次：2012 年 1 月第 1 版第 1 次印刷

（版权所有 翻印必究 印装有误 负责调换）

封面设计：深度文化
责任编辑：刘立
李萌
责任校对：全 卫
开 本：787mm×1092 mm 1/16
印 张：34.5
印 数：1-3 000
字 数：846 千字
定 价：69.00 元（含 1CD）

前言



进入21世纪以来，整个社会已经逐渐变得陌生了！生活和工作的快节奏令我们目不暇接，各种各样的信息充斥着我们的视野、撞击着我们的思维。追忆过去，Windows操作系统的诞生成就了微软的霸主地位，也造就了PC时代的繁荣。然而，以Android和iPhone手机为代表的智能移动设备的发明却敲响了PC时代的丧钟！移动互联网时代(3G时代)已经来临，谁会成为这些移动设备上的主宰？毫无疑问，它就是Android！

看3G的璀璨绚丽

随着3G的到来，无线宽带的应用越来越广泛，使得更多内容丰富的应用程序布置在手机上成为可能，如视频通话、视频点播、移动互联网冲浪、在线视听、内容分享等。为了承载这些数据应用及快速部署，手机功能将会越来越智能，越来越开放，为了实现这些需求，必须有一个好的开发平台来支持，在此由Google公司发起的OHA联盟走在了业界的前列，2007年11月推出了开放的Android平台，任何公司及个人都可以免费获取到源代码及开发SDK。三星、摩托罗拉、索爱、LG、华为等公司都已推出Android平台的手机，中国移动也联合各手机厂商共同推出了基于Android平台的OPhone。按目前的发展态势，我们有理由相信，Android平台能够在短时间内跻身智能手机开发平台的前列。

Android来袭

2009年3G牌照在中国国内发放后，3G、Andriod、iPhone、Google、苹果、手机软件、移动开发等词变得耳熟能详。随着3G网络的大规模建设和智能手机的迅速普及，移动互联网时代已经微笑着迎面而来。

以创新的搜索引擎技术而一跃成为互联网巨头的Google，无线搜索成为其进军移动互联网的一块基石。早在2007年，Google中国就把无线搜索当作战略重心，不断推出新产品，尝试通过户外媒体推广移动搜索产品，并积极与运营商、终端厂商、浏览器厂商等达成战略合作。

Android操作系统是Google最具杀伤力的武器之一。苹果以其天才的创新，使得iPhone在全球迅速拥有了数千万忠实“粉丝”，而Android作为第一个完整、开放、免费的手机平台，使开发者在为其开发程序时拥有更大的自由。与 Windows Mobile、Symbian等厂商不同的是，Android操作系统免费向开发人员提供，这样可节省近三成成本，得到了众多厂商与开发者的拥护。最早进入Andriod市场的宏达电已经陆续推出了G1、Magic、Hero、Tattoo4款手机，三星也在近期推出Galaxy i7500平板电脑，连摩托罗拉也推出了新款Andorid手机Cliq，中国移动也以Android为基础开发了OPhone平台。这些发展证明Android已经成为智能手机市场的重要发展趋势。

巨大的优势

从技术角度而言，Android与iPhone相似，采用WebKit浏览器引擎，具备触摸屏、高级图形显示和上网功能，用户能够在手机上查收电子邮件、搜索网址和观看视频节目等。Android手机比iPhone等其他手机更强调搜索功能，界面更强大，可以说是一种融入了全部Web应用的平台。Android的版本包括Android1.1、Android1.5、Android1.6、Android2.0、Android2.1，



Android2.3, Android 3.0, Android 3.1, Android3.2。相信在本书出版时,很多读者已经在使用Android4.0的系统了。随着版本的更新,从最初的触屏到现在的多点触摸,从普通的联系人到现在的数据同步,从简单的GoogleMap到现在的导航系统,从基本的网页浏览到现在的HTML 5,这都说明Android已经逐渐稳定,而且功能越来越强大。此外,Android平台不仅支持 Java、C、C++等主流的编程语言,还支持Ruby、Python等脚本语言,甚至Google专为Android的应用开发推出了Simple语言,这使得Android有着非常广泛的开发群体。

无论是产品还是技术,商业应用是它最大的发展力。Android如此受厂商与开发者的青睐,它的前景一片光明。伴随着装有Android操作系统的移动设备的增加,基于Android的应用需求势必也会加。

本书的内容

由于Android平台被推出的时间才4年,了解Android平台软件开发技术的程序员还不多,如何迅速地推广和普及Android平台软件开发技术,让越来越多的人参与到Android应用的开发中,是整个产业链都在关注的一个话题。笔者较早从事了和Android相关的研究与开发工作,为了帮助开发者更快地进入Android开发行列,特意精心编写了这本Android图书。本书系统讲解了Android软件开发的基础知识,图文并茂地帮助读者学习和掌握SDK、开发流程以及常用的API等。书中以讲述实战实例为导向,用一个个典型应用生动地引领读者进行项目开发实践。本书是一本内容翔实,理论实践紧密结合的教程。

科学的学习方法

中国有句古话:“授人以鱼,不如授人以渔。”传授给人既有知识,不如传授给人学习知识的方法。通过本书,我们将告诉读者学习的方法,并介绍一条比较清晰的学习之路。

(1) 积极的心态

无论是知识还是技能,智者之所以能够更好更快地掌握这些知识和技能,很大程度上得益于良好的学习方法。人们常说:兴趣是最好的老师,压力是前进的动力,要想获得一个积极的心态,最好能对学习对象保持浓厚的兴趣。如果暂时提不起兴趣,那么就重视来自工作或生活的压力,把它们转化为学习的动力。

(2) 注重实践

实践是检验技术的真理,本着这一思想,本书罗列了大量的实例,这些实例是Android所能实现的方方面面,甚至是各个领域。实例教学的高效率将在本书身上展现得一览无余。

(3) 善用资源,学以致用

对于Android开发技术,除了少部分专业人士外,大部分人学习的目的是为了应用,开发手机应用程序。“解决问题”常常是促使人学习的一大动机,带着问题学习,不但进步快,而且很容易对网络产生更大的兴趣,从而获得持续的进步。

本书特色

本书内容丰富,实例内容覆盖全面,涵盖Android技术人员成长道路上的众多知识。我们的目标是通过一本图书,实现多本图书的价值,读者可以根据自己的需要有选择地阅读,以完善本人的知识和技能结构。在内容的编写上,本书具有以下特色:

(1) 结构合理

从读者的实际需要出发,科学安排知识结构,内容由浅入深,具有很强的知识性和实用性,反映了当前Android技术的发展和应用水平。同时全书精心筛选的最具代表性、读者最关心的知识点,几乎包括Android技术的各个方面。

(2) 易学易懂

本书条理清晰、语言简洁，可帮助读者快速掌握每个知识点；每个部分既相互连贯又自成体系，使读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行针对性的学习。

(3) 实用性强

本书彻底摒弃枯燥的理论和简单的操作，注重实用性和可操作性，讲解了各个实例的具体实现过程，使用户在掌握相关操作技能的同时，还能学习到相应的基础知识。

(4) 实例全面

书中的开发实例典型并具有创意，涵盖了Android所能触及的所有领域，每个实例都体现了移动互联网应用所需的创新精神及良好的用户体验理念，这个设计思路很值得大家去思考和学习。

读者对象

初学编程的自学者

编程爱好者

大中专院校的老师和学生

相关培训机构的老师和学员

毕业设计的学生

初中级程序开发人员

程序测试及维护人员

参加实习的初级程序员

在职程序员

资深程序员

本书主要由扶松柏编写，参与本书编写的还有李天祥、周锐、于秀青、周秀、吴善才、邢辉、孙跃杰、刘佳丽、万泉、王永忠、邓才兵、钟世礼、谭贞军、罗红仙、张加春、王东华、王振丽、熊斌、王教明、万春潮、郭慧玲、侯恩静、张玲玲、程娟、王文忠、陈强、何子夜、黄斌、吴艳臣、周中一等。书中如有纰漏和不尽如人意之处，诚请读者提出意见或建议，以便修订并使之更臻完善。

编著者



目 录

第1章 走进Android世界

1.1 智能手机飞速发展.....	2	1.3.5 创建Android虚拟设备(AVD).....	18
1.1.1 主流手机系统介绍	2	1.3.6 常见的几个问题	20
1.1.2 Android横空出世.....	3	1.3.7 SDK工具集	24
1.2 Android何以风靡世界.....	4	1.4 Android模拟器	26
1.2.1 丰厚的奖励机制	4	1.4.1 Android模拟器简介	26
1.2.2 商家的支持	4	1.4.2 模拟器和真机究竟有何区别	26
1.2.3 光明的前景	5	1.4.3 模拟器简单总结	27
1.3 搭建Android应用开发环境	5	1.5 纵览Android体系	29
1.3.1 安装Android SDK的系统要求.....	5	1.5.1 简析Android安装文件	29
1.3.2 安装JDK、Eclipse、Android SDK.....	6	1.5.2 Android体系结构介绍	31
1.3.3 设定Android SDK Home	16	1.5.3 Android应用工程文件组成	34
1.3.4 验证开发环境	17	1.5.4 应用程序的生命周期	37

第2章 界面布局实战演练

2.1 使用线性布局 (LinearLayout)	42	2.7.2 在屏幕中显示一段文字	53
2.2 使用相对布局 (RelativeLayout)	43	2.7.3 设置手机屏幕中的字体	56
2.3 使用表格布局 (TableLayout)	44	2.7.4 在屏幕中显示编辑框	59
2.4 使用绝对布局 (AbsoluteLayout) ...	46	2.7.5 在屏幕中显示复选框	61
2.5 使用标签布局 (TabLayout)	47	2.7.6 在屏幕中显示单选框	63
2.6 使用层布局 (FrameLayout)	49	2.7.7 在屏幕中显示下拉列表框	65
2.7 使用桌面组件Widget来布局	50	2.7.8 在屏幕中实现自动输入文本	68
2.7.1 在屏幕中实现一个按钮效果	51		



第3章 基本控件实战演练

3.1 使用RadioGroup控件实现选择处理...	72	3.10 在收集屏幕中显示磁盘中的图片 ..	102
3.2 使用屏幕中实现一个购物清单.....	74	3.11 触动Menu菜单控件	104
3.3 在手机屏幕中实现相框效果	78	3.12 使用SimpleAdapter实现ListView 组件的效果.....	107
3.4 在屏幕中实现选择处理	81	3.13 在屏幕中实现抽屉样式效果	110
3.5 在屏幕中实现一个相簿功能	84	3.14 使用Toast和Notification实现提醒 效果	117
3.6 开发一个文件搜索程序	88	3.15 添加/删除Spinner的菜单	125
3.7 模拟实现一个时钟效果	91		
3.8 在手机屏幕中实现进度条效果.....	94		
3.9 开发一个自动选择日期和时间的程序 ...	98		

第4章 数据存储实战演练

4.1 使用SharedPreferences存储	132	4.4 开发一个日记簿项目	142
4.2 使用SQLite存储	134	4.5 升级日记簿功能	149
4.3 使用ContentProvider存储	139		

第5章 通信领域实战演练

5.1 拨号、邮件和网址处理	164	5.6 搜索通讯录.....	190
5.2 拨打电话	167	5.7 使用Wi-Fi	196
5.3 发送短信交互	172	5.8 触摸拨号	206
5.4 发送邮件	179	5.9 获取设备运营商信息	207
5.5 实现震动效果	184		

第6章 自动服务实战演练

6.1 来短信自动提醒	218	6.5.1 一些基本知识	236
6.2 自动显示剩余电量	222	6.5.2 具体实现	237
6.3 来短信E-mail通知	226	6.6 闹钟到时响	241
6.4 来电后显示提示信息	231	6.7 黑名单来电自动静音	251
6.5 获取手机存储卡的容量	235	6.8 监听发送的短信是否成功	256

第7章 互联网实战演练

7.1 浏览指定的网页	264	7.5 播放在线音乐	278
7.2 加载显示HTML程序	267	7.6 下载在线手机铃声	288
7.3 使用浏览器打开网页	269	7.7 开发一个简易RSS系统	297
7.4 显示网络中的照片	273		

第8章 多媒体实战演练

8.1 获取图片的宽和高	314	8.6 播放MP3音乐	350
8.2 绘制各种几何图形	318	8.7 开发一个录音机程序	356
8.3 开发一个手机屏保程序	323	8.8 开发一个拍照程序	365
8.4 在屏幕上触摸移动照片	337	8.9 开发一个视频播放器	376
8.5 调节音量	342		



第9章 Google地图实战演练

9.1 获取当前位置的坐标	382	9.5 实现路径导航	403
9.2 在手机中使用谷歌地图	385	9.6 移动手机时自动更新位置	412
9.3 输入坐标后在地图中实现定位	392	9.7 在地图中绘制线路并计算距离	418
9.4 在手机中实现地址查询	397	9.8 在谷歌地图中显示指定的位置	428

第10章 Google API实战演练

10.1 模拟验证官方账号	432	10.3 在手机中生成二维条码	448
10.2 实现Google搜索	442	10.4 手机翻译	454

第11章 游戏实战演练

11.1 益智类游戏——魔塔	460	11.2 竞技类游戏——中国象棋	484
----------------------	-----	------------------------	-----

第12章 优化和发布项目

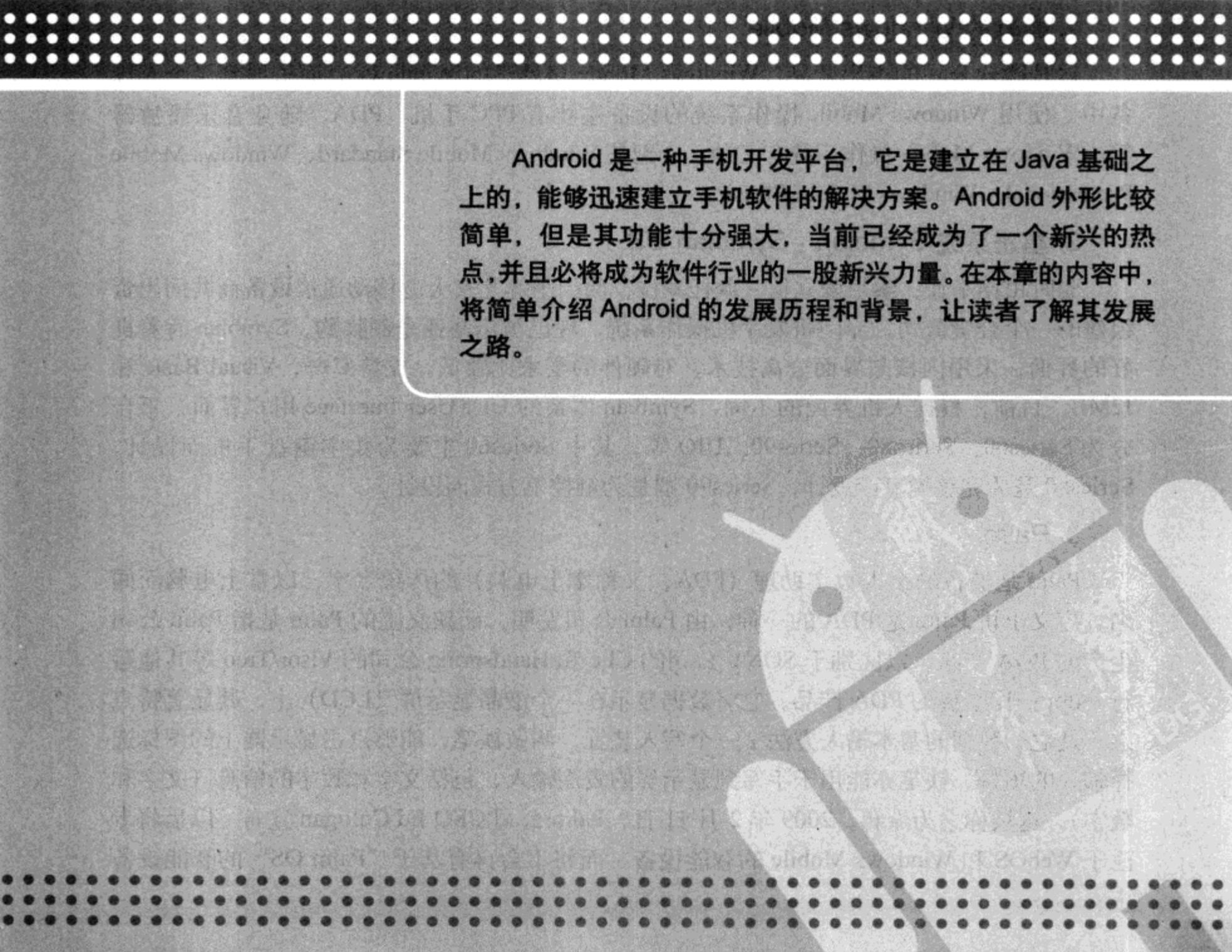
12.1 UI界面中优化之<merge />标签	516	12.6 发布自己的作品来盈利	530
12.2 测试计算机的性能	519	12.6.1 申请会员	530
12.3 测试内存性能	522	12.6.2 生成签名文件	533
12.4 Android Layout优化	525	12.6.3 使用签名文件	539
12.5 优化模拟器	527	12.6.4 发布	541



第1章

走进Android世界

Android 是一种手机开发平台，它是建立在 Java 基础之上的，能够迅速建立手机软件的解决方案。Android 外形比较简单，但是其功能十分强大，当前已经成为了一个新兴的热点，并且必将成为软件行业的一股新兴力量。在本章的内容中，将简单介绍 Android 的发展历程和背景，让读者了解其发展之路。



1.1 智能手机飞速发展

在 Android 诞生之前，智能手机这个新鲜事物大大丰富了人们的生活，得到了广大消费者的青睐。顿时间因为利益的驱动，各种智能手机操作系统纷纷建立，并且大肆招兵买马来抢夺市场。

1.1.1 主流手机系统介绍

所谓智能手机（Smartphone），是指能够具有像个人电脑那样的强大功能，拥有独立的操作系统，用户可以自行安装第三方软件和游戏等第三方服务商提供的程序，并且可以通过移动通信网络来实现无线网络接入。其实在当前市面中已经有很多优秀的智能手机产品，比如 Symbian（塞班）操作系统的 S 系列、微软的 Windows Mobile 系列。

在当今市面中最主流的智能手机系统有塞班、微软、苹果、黑莓、Palm 和本书的主角 Android。

1. 微软的 Windows Mobile

这是微软公司的杰出产品，Windows Mobile 将熟悉的 Windows 桌面扩展到了个人设备中。使用 Windows Mobile 操作系统的设备主要有 PPC 手机、PDA、随身音乐播放器等。Windows Mobile 操作系统有三种，分别是 Windows Mobile Standard、Windows Mobile Professional、Windows Mobile Classic。

2. 塞班系统 Symbian : Symbian OS

当时由诺基亚、索尼爱立信、摩托罗拉、西门子等几家大型移动通信设备商共同出资组建的一个合资公司，专门研发手机操作系统，现已被诺基亚全额收购。Symbian 有着良好的界面，采用内核与界面分离技术，对硬件的要求比较低，支持 C++、Visual Basic 和 J2ME。目前，根据人机界面的不同，Symbian 体系的 UI（User Interface 用户界面）平台分为 Series60、Series80、Series90、UIQ 等。其中 Series60 主要为数字键盘手机而设计，Series80 是为完整键盘而设计，Series90 则是为触控笔方式而设计。

3. Palm

Palm 是流行的个人数字助理（PDA，又称掌上电脑）的传统名字，以掌上电脑而闻名。广义上讲 Palm 是 PDA 的一种，由 Palm 公司发明。而狭义上的 Palm 是指 Palm 公司生产的 PDA 产品，以区别于 SONY 公司的 Clie 和 Handspring 公司的 Visor/Treo 等其他运行 Palm 操作系统的 PDA 产品。它将数据显示在一个液晶显示屏（LCD）上。其显著特点之一是它的数据的基本输入方法：一个写入装置，叫做铁笔，能够点击显示器上的图标选择输入的项目。铁笔亦能用于手写到显示屏的表面输入，包括文字和数字的信息（文字和数字），这被称之为涂鸦。2009 年 2 月 11 日，Palm 公司 CEO Ed Colligan 宣布：以后将专注于 WebOS 和 Windows Mobile 的智能设备，而将不会再有基于“Palm OS”的智能设备。



推出，除了 Palm Centro 会在以后和其他运营商合作时继续推出。

4. 黑莓 BlackBerry

BlackBerry 是加拿大 RIM 公司推出的一种移动电子邮件系统终端，其特色是支持推送式电子邮件、手提电话、文字短信、互联网传真、网页浏览及其他无线信息服务，其最大优势在于收发邮件。

5. iPhone

iPhone 由苹果公司 (Apple, Inc.) 首席执行官史蒂夫·乔布斯在 2007 年 1 月 9 日举行的 Macworld 宣布推出，2007 年 6 月 29 日在美国上市。它将创新的移动电话、可触摸宽屏 iPod 以及具有桌面级电子邮件、网页浏览、搜索和地图功能的突破性因特网通信设备这三种产品完美地融为一体。外人对苹果的评价颇高，说“iPhone 是一款革命性的，不可思议的产品，比市场上的其他任何移动电话整整领先了五年。”

6. Android

Android 是本书的主角，于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称。该平台由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放和完整的移动软件。

1.1.2 Android 横空出世

在 2007 年以前，智能手机系统领域形成了塞班、苹果和微软的三足鼎立之势。在 2007 年下半年 Android 突然神秘崛起，以完全免费为杀手锏，刚刚推出时就有颠覆三足鼎立之势。Android 即安卓，英文原义是“机器人”。虽然崛起较晚，但是系出 Linux 家族。Android 采用了 WebKit 浏览器引擎，具备触摸屏、高级图形显示和上网功能，用户能够在手机上查看电子邮件、搜索网址和观看视频节目等，同时 Android 还具有比 iPhone 等其他手机更强的搜索功能，可以说是一种融入全部 Web 应用的平台。

正是因为安卓特有的巨大优势，并且凭借 Android SDK，2010 年下半年，Android 规模便超越了苹果和塞班，成为市场占有率最高的智能手机系统。

1. 系出名门

Android 出身于 Linux 世家，是一款开源的手机操作系统。Android 功成名就之后，各大手机联盟纷纷加入，这个联盟由包括中国移动、摩托罗拉、高通、宏达电和 T-Mobile 在内的 30 多家技术和无线应用的领军企业组成。通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，希望借助建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。

2. 开发团队

Android 的研发队伍阵容强大，包括摩托罗拉、Google、HTC (宏达电)、PHILIPS、T-Mobile、高通、魅族、三星、LG 以及中国移动在内的 34 家企业，这都是在江湖享誉盛名的大佬。它们都将基于该平台开发手机的新型业务，应用之间的通用性和互联性将在最大程度上得到保持。

1.2 Android何以风靡世界

既然安卓来势汹汹，并且发展迅速，短短几年间成为了市场占有率最高的智能手机系统，究竟是什么原因让 Android 取得如此辉煌成果呢？这是因为 Android 任重而道远，为了吸引更多用户而推出了很多有“意义”的事情。

1.2.1 丰厚的奖励机制

安卓为了提高程序员的开发积极性，不但为它们提供了一流硬件的设置，还提供了一流的软件服务。并且采取了振奋人心的奖励机制，定期召开内部武林大会，夺魁者将得到重奖。

1. 开发 Android 平台的应用

口号是——只要我们的创意存在，我们将站在 Android 的最前沿浪尖。在 Android 平台上，程序员可以开发出各式各样的应用。Android 是通过 Java 语言开发的，只要具备 Java 开发基础，就能很快地上手并掌握。作为单独的 Android 开发，以 Java 作为编程门槛并不高，即使没有编程经验的门外汉，也可以在突击学习 Java 之后而学习 Android。另外，Android 完全支持 2D、3D 和数据库，并且和浏览器实现了集成。通过 Android 平台，程序员可以迅速、高效地开发出绚丽多彩的应用，例如常见的工具、管理、互联网和游戏等。

2. 奖金丰厚的 Android 大赛

为了吸引更多的用户使用 Android 开发，已经成功举办了奖金为 1000 万美元的开发者竞赛，以鼓励开发人员创建出创意十足、十分有用的软件。这种大赛对于开发人员来说，不但能提高自己的开发水平，并且高额的奖金也是学习的动力。

3. 在 Android Market 上获取收益

为了能让 Android 平台吸引更多的关注，谷歌开发了自己的 Android 软件下载店 Android Market，允许开发人员将应用程序在上面发布，也允许 Android 用户随意下载获取自己喜欢的程序。作为开发者，需要申请开发者账号才能将自己的程序上传到 Android Market，并且可以对自己的软件进行定价。所以说，只要你的软件程序足够吸引人，你就可以获得很好的金钱回报，从而达到学习、赚钱两不误。



注意 Android Market 地址是 <http://www.Android.com/market/>，感兴趣的读者可以去浏览一番。因为某些原因，可能登录不上，建议使用代理服务器。

1.2.2 商家的支持

1. 手机制造商

台湾宏达电国际（HTC）（Palm 等多款智能手机的代工厂），摩托罗拉（美国最大的手机制造商），韩国三星电子（仅次于诺基亚的全球第二大手机制造商），韩国 LG 电子，

中国移动（全球最大的移动运营商），日本 KDDI（2900 万用户），日本 NTT DoCoMo（5200 万用户），美国 Sprint Nextel（美国第三大移动运营商，5400 万用户），意大利电信（Telecom Italia）（意大利主要的移动运营商，3400 万用户），西班牙 Telefónica（在欧洲和拉美有 1.5 亿用户），T-Mobile（德意志电信旗下公司，在美国和欧洲有 1.1 亿用户）。

2. 半导体公司

Audience Corp（声音处理器公司），Broadcom Corp（无线半导体主要提供商），英特尔（Intel），Marvell Technology Group，Nvidia（图形处理器公司），SiRF（GPS 技术提供商），Synaptics（手机用户界面技术），德州仪器（Texas Instruments），高通（Qualcomm），惠普 HP（Hewlett-Packard Development Company, L.P.）。

3. 软件公司

Aplix，Ascender，eBay 的 Skype，Esmertec，Living Image，NMS Communications，Noser Engineering AG，Nuance Communications，PacketVideo，SkyPop，Sonix Network，TAT-The Astonishing Tribe，Wind River Systems。

1.2.3 光明的前景

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

在市场大势上，与 iPhone 相似，Android 采用了 WebKit 浏览器引擎，具备触摸屏、高级图形显示和上网功能，用户能够在手机上查看电子邮件、搜索网址和观看视频节目等，比 iPhone 等其他手机更强调搜索功能，界面更强大，可以说是一种融入全部 Web 应用的单一平台。

1.3 搭建Android应用开发环境

作为一项新兴的计算机开发技术，在进行开发前首先要搭建一个对应的开发环境。在搭建开发环境前，需要了解安装开发工具所需要的硬件和软件配置条件。

1.3.1 安装Android SDK的系统要求

在搭建之前，一定先确定基于 Android 应用软件所需要开发环境的要求，具体如表 1-1 所示。

Android 工具是由多个开发包组成的，具体说明如下：

- ◆ JDK：可以到网址 <http://java.sun.com/javase/downloads/index.jsp> 下载。
- ◆ Eclipse（Europa）：可以到网址 <http://www.eclipse.org/downloads/> 下载 Eclipse IDE for Java Developers。

- ◆ Android SDK：可以到网址 <http://developer.android.com> 下载。
- ◆ 还有对应的开发插件。

表1-1 开发系统所需求参数

项目	版本要求	说明	备注
操作系统	Windows XP 或 Vista, Mac OS X 10.4.8+Linux Ubuntu Drapper	根据自己的电脑自行选择	选择自己最熟悉的操作系统
软件开发包	Android SDK	选择最新版本的SDK	截止到目前，最新手机版本是2.3
IDE	Eclipse IDE+ADT	Eclipse3.3 (Europa), 3.4 (Ganymede) ADT(Android Development Tools)开发插件	选择“for Java Developer”
其他	JDK Apache Ant	Java SE Development Kit 5 或 6 Linux 和 Mac 上使用 Apache Ant 1.6.5+, Windows 上使用 1.7+ 版本	(单独的JRE不可以的，必须要有 JDK)，不兼容 Gnu Java 编译器 (gcj)

1.3.2 安装JDK、Eclipse、Android SDK

本书讲的安装是以 Windows XP SP2 为平台，安装的软件为 JDK 1.6、Eclipse 3.3、ADT1.5、Android SDK 2.3。下面具体介绍各自的安装步骤。

1. 安装 JDK

安装 Eclipse 的开发环境需要 JRE 的支持，在 Windows 上安装 JRE/JDK 非常简单，看下面的流程。

step 01 在 Sun 官方网站下载，网址为 <http://developers.sun.com/downloads/>，如图 1-1 所示。

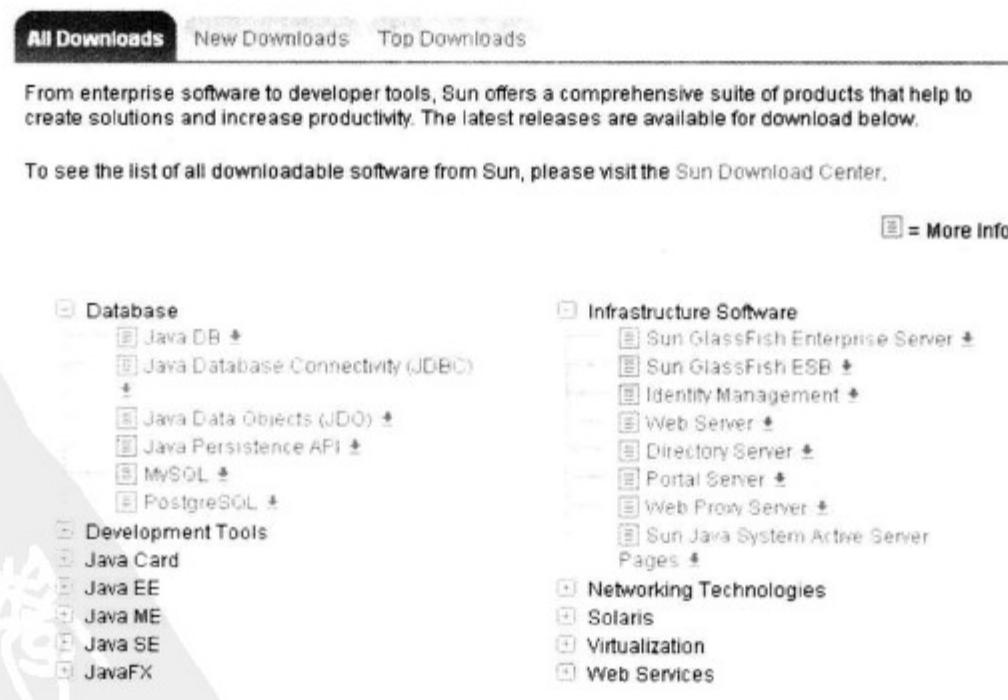


图1-1 Sun官方下载页面

step 02 在图 1-1 中可以看到有很多版本，运行 Eclipse 时虽然只需要 JRE 就可以了，但是在开发 Andriod 应用程序的时候，是需要完整的 JDK（JDK 已经包含了 JRE），且要求其版本在 1.5+ 以上，这里选择 Java SE (JDK) 6，其下载页面如图 1-2 所示。

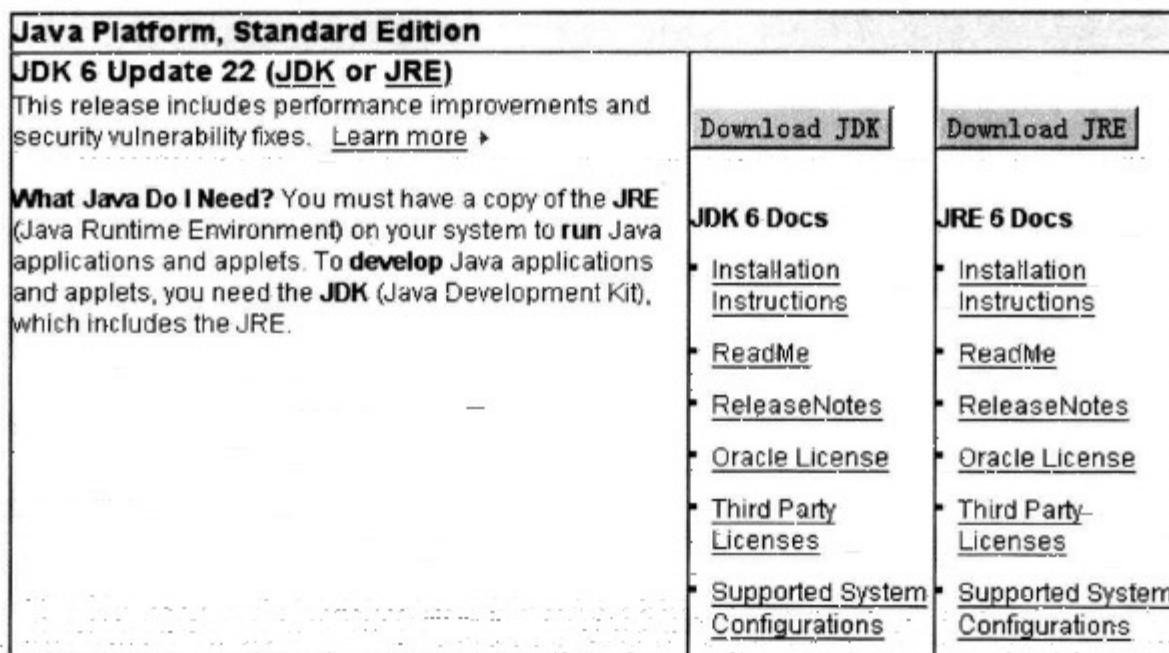


图1-2 JDK下载页面

step 03 在图 1-2 中找到“JDK 6 Update 22”，单击其右侧的【Download】按钮后弹出“填写登录信息”界面，在此输入用户的账号信息，如果没有账号可以免费注册一个。然后单击【Continue】按钮，如图 1-3 所示。

There is more information on the available files for download on the [Supported System Configurations](#) page.

Select Platform and Language for your download:

Platform: Language: Multi-language

By selecting 'Continue' below, you hereby accept the terms and conditions of the [Java SE Development Kit 6u22 License Agreement](#).

Optional: Please Log In or Register for additional functionality and [benefits](#). Or, click "Continue" now to proceed without Log In or Registration.

User Name:
Example: jim23 or jim@company.com
Password:
* [Register Now](#)
* [Why Register?](#)
* [Forgot User Name or Password ?](#)

图1-3 输入账号信息

step 04 进入“选择操作系统和语言”界面，在此首先选择“Windows”，然后单击【Download】按钮，如图 1-4 所示。

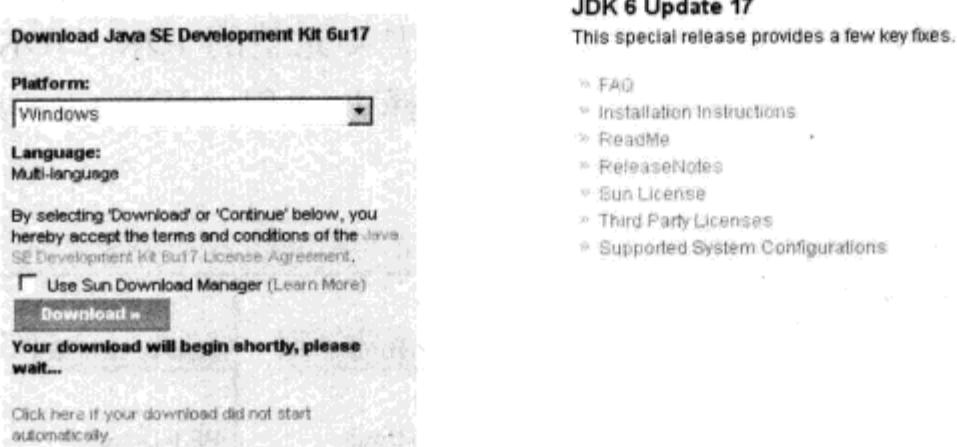


图1-4 选择“Windows”

经过上述操作后，开始下载安装文件“jdk-6u22-windows-i586.exe”。

step 05 下载完成后双击“jdk-6u22-windows-i586.exe”开始进行安装，将弹出“安装向导”对话框，在此单击【下一步】按钮，如图 1-5 所示。

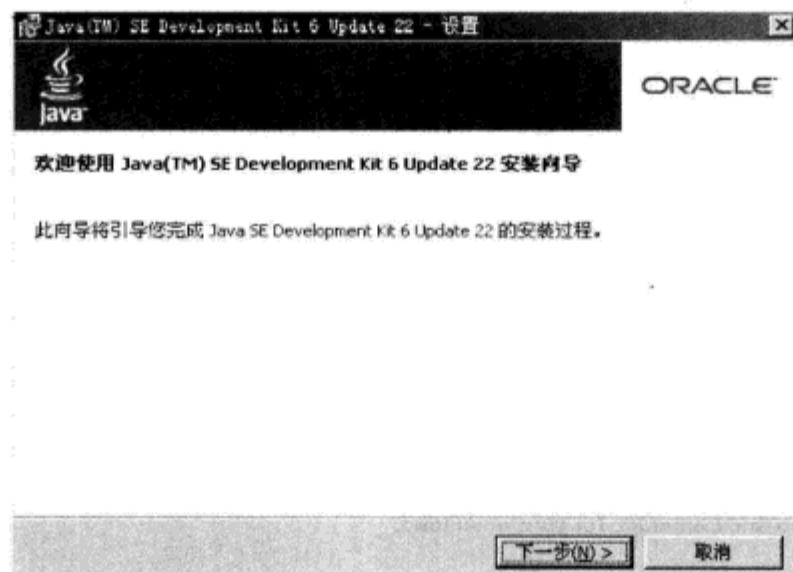


图1-5 “安装向导”对话框

step 06 弹出“自定义安装”对话框，在此选择文件的安装路径，如图 1-6 所示。

step 07 单击【下一步】按钮，开始进行安装，如图 1-7 所示。

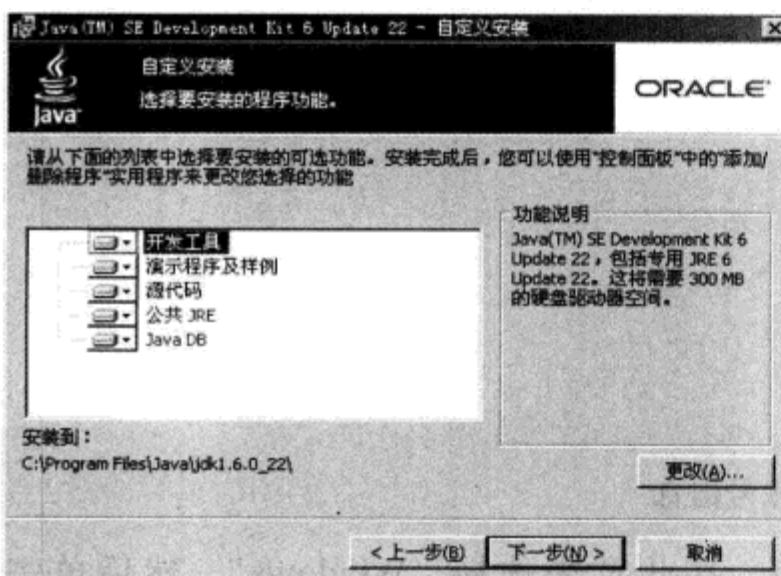


图1-6 “自定义安装”对话框

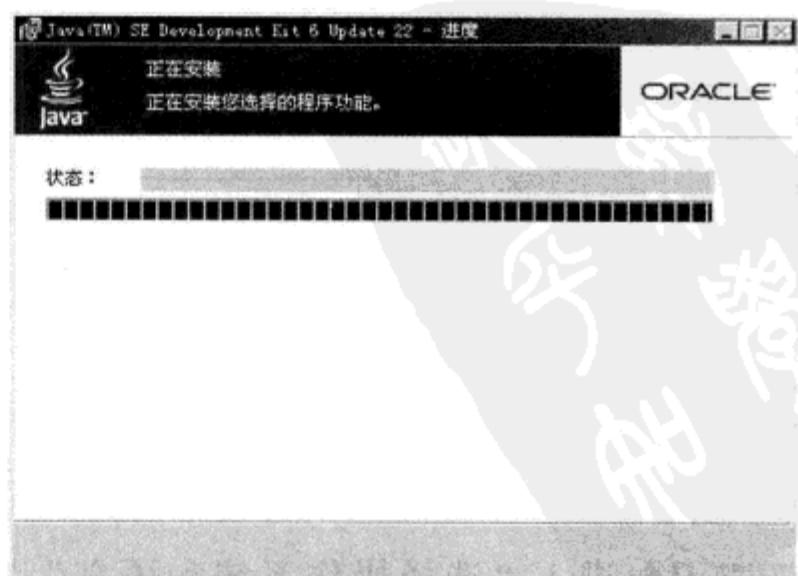
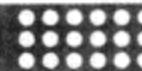


图1-7 开始安装



step 08 完成后弹出“目标文件夹”对话框，在此选择要安装的位置，如图 1-8 所示。

step 09 单击【下一步】按钮后继续开始安装，如图 1-9 所示。

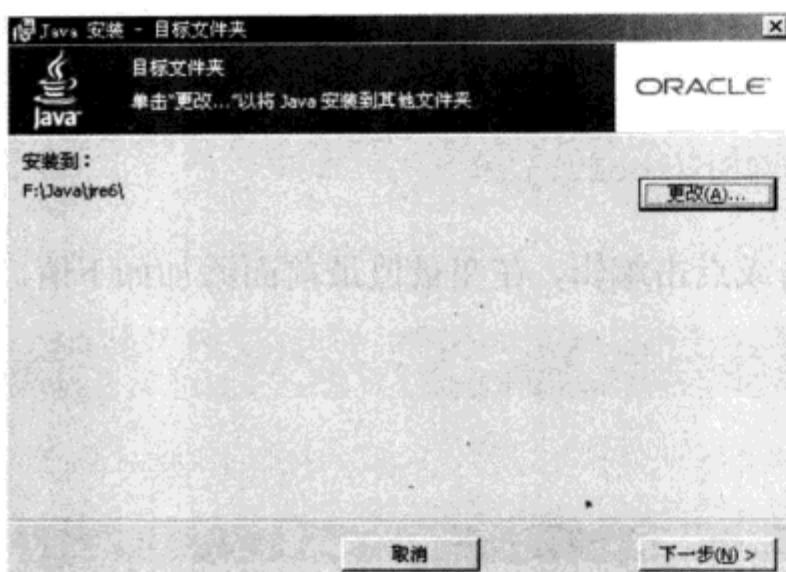


图1-8 “目标文件夹”对话框

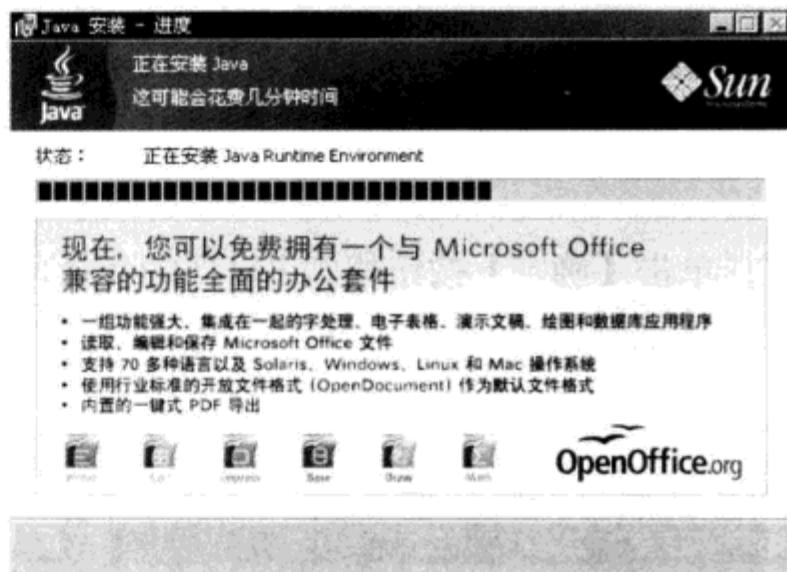


图1-9 继续安装

step 10 完成后弹出“完成”对话框，单击【完成】按钮后完成整个安装过程，如图 1-10 所示。

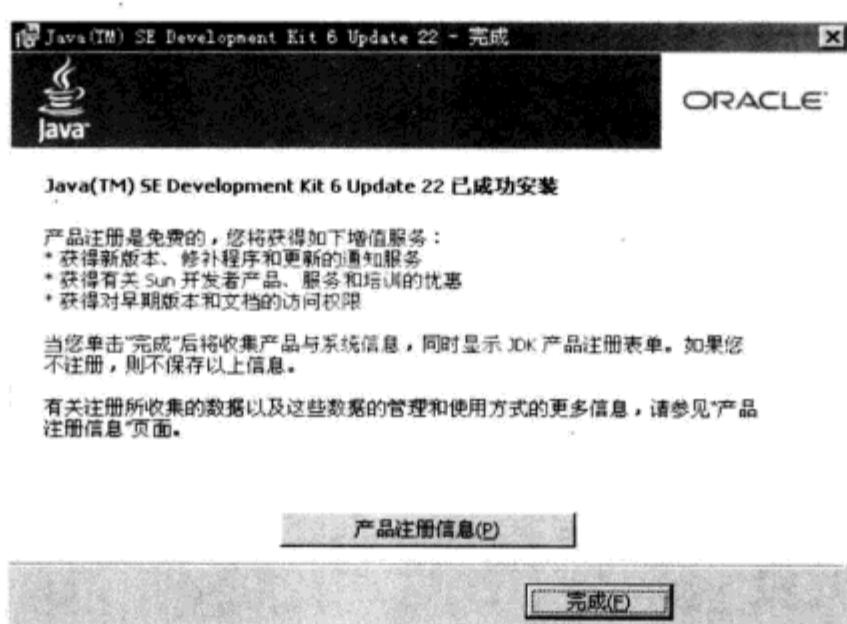


图1-10 完成安装

完成安装后可以检测是否安装成功，方法是依次单击【开始】 | 【运行】，在运行框中输入“cmd”并按下 Enter 键，在打开的 CMD 窗口中输入 java -version，如果显示如图 1-11 所示的提示信息，则说明安装成功。

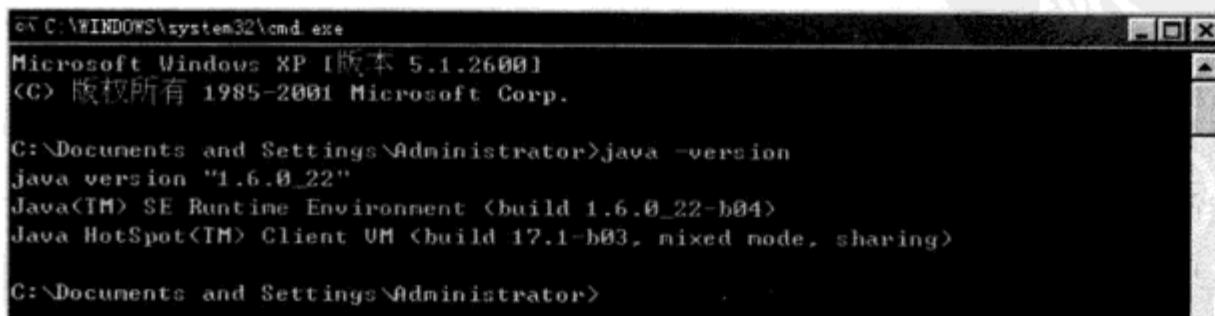


图1-11 CMD窗口

如果检测没有安装成功，需要将其目录的绝对路径添加到系统的 PATH 中。具体做法如下所示。

step 01 鼠标右键依次单击【我的电脑】|【属性】|【高级】，点击下面的“环境变量”，在下面的“系统变量”处选择新建，在变量名处输入 JAVA_HOME，变量值中输入刚才的目录，比如“F:\Java\jdk1.6.0_22”，如图 1-12 所示。

step 02 再次新建一个变量名为 classpath，其变量值如下所示。

```
.;%JAVA_HOME%/lib/rt.jar;%JAVA_HOME%/lib/tools.jar
```

单击【确定】按钮找到 PATH 的变量，双击或点击编辑，在变量值最前面添加如下值。

```
%JAVA_HOME%/bin;
```

具体如图 1-13 所示。

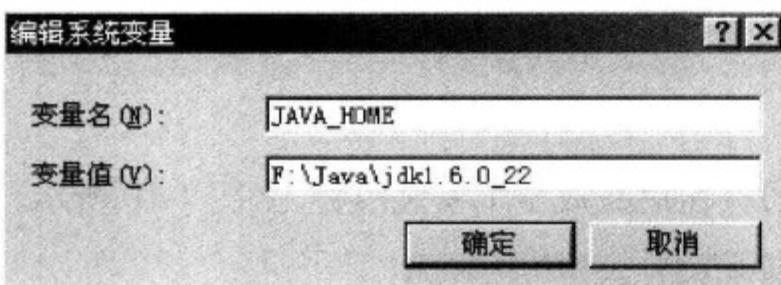


图1-12 设置系统变量

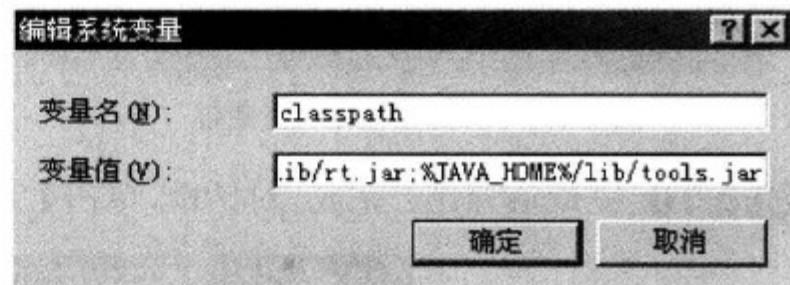


图1-13 设置系统变量

step 03 再依次单击【开始】|【运行】，在运行框中输入“cmd”并按下 Enter 键，在打开的 CMD 窗口中输入 java -version，如果显示如图 1-14 所示的提示信息，则说明安装成功。



图1-14 CMD界面



注意 上述变量设置中，安装JDK的路径是F:\Java\jdk1.6.0_22。

2. 安装 Eclipse

在安装好 JDK 后，就可以接着安装 Eclipse 了，具体步骤如下所示。

step 01 打开 Eclipse 的官方下载页面 <http://www.eclipse.org/downloads/>，如图 1-15 所示。

step 02 在如图 1-15 所示界面中选择“Eclipse IDE for Java Developers (85 MB)”，来到其下载的镜像页面，在此只需选择离用户最近的镜像即可（一般推荐的下载速度就不错），如图 1-16 所示。



The screenshot shows the Eclipse Downloads page. At the top, there are links for Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. A search bar is also present. Below the navigation, there's a section titled "Eclipse Downloads" with two tabs: "Eclipse Packages" (selected) and "Projects". Under "Eclipse Packages", there are three main sections: "Galileo Packaged (based on Eclipse 3.5 SR1)" with a "Compare Packages" link, "Eclipse IDE for Java EE Developers (189 MB)", and "Eclipse IDE for Java Developers (92 MB)". Each section includes a brief description, download count (e.g., 1,195,339, 601,023), and download links for Windows, Mac Carbon, Mac Cocoa, and Linux 32-bit and 64-bit versions. To the right, there's a sidebar with a "Looking for Older Versions or Source Code?" link and an advertisement for "CRYSTAL REPORTS FOR ECLIPSE" from SAP.

图1-15 下载页面

The screenshot shows the "Eclipse downloads - mirror selection" page. It features a sidebar with links for "Downloads Home", "Bit Torrents", "By project", "By topic", "Source code", and "More Packages". The main content area has a heading "Eclipse downloads - mirror selection" and a note about download terms. It provides a download link for "eclipse-java-galileo-SR1-win32.zip" via China Amazon AWS (http://) and indicates that a BitTorrent is available for this file. There's also a link to pick a mirror site below. On the right side, there are advertisements for "ORACLE WORLD" and "Oracle Develop", both mentioning the Premier Conference for Developers, October 11-13, 2009, San Francisco, CA, and a "Register NOW!" button.

图1-16 选择镜像

step 03 下载完成后，先找到下载的压缩包“eclipse-java-galileo-SR1-win32.zip”。



注意 ▶ 解压Eclipse下载的压缩文件后就可以使用，而无须执行安装程序，不过在使用前一定要先安装JDK。在此假设 Eclipse解压后存放的目录为 F:\eclipse。

step 04 进入解压后的目录，此时可以看到一个名为“eclipse.exe”的可执行文件，双击此文件直接运行，Eclipse能自动找到用户先期安装的JDK路径，启动界面如图 1-17 所示。

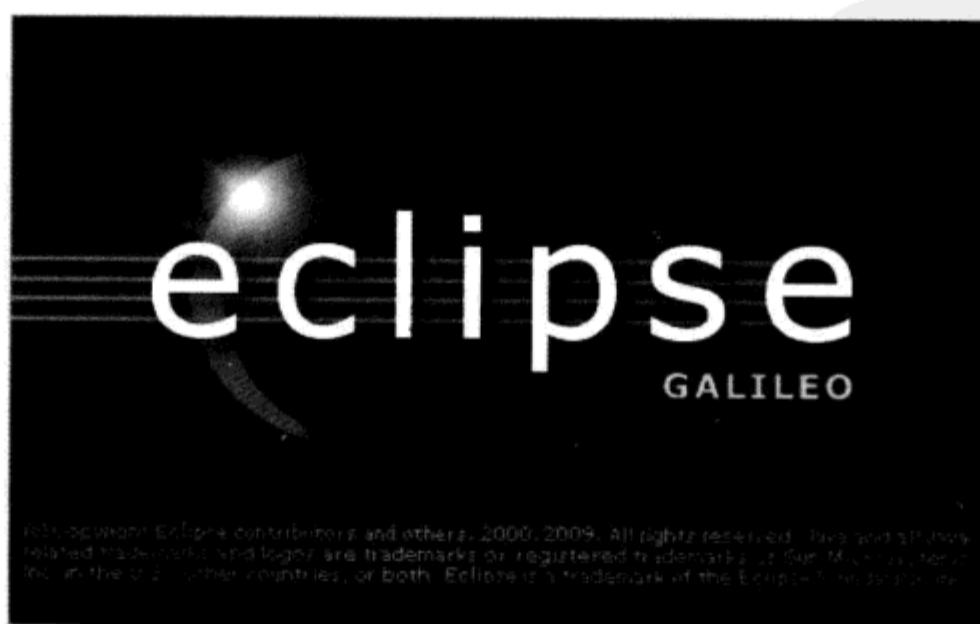


图1-17 Eclipse启动界面

step 05 因为是安装后第一次启动 Eclipse，所以会看到选择工作空间的提示，如图 1-18 所示。

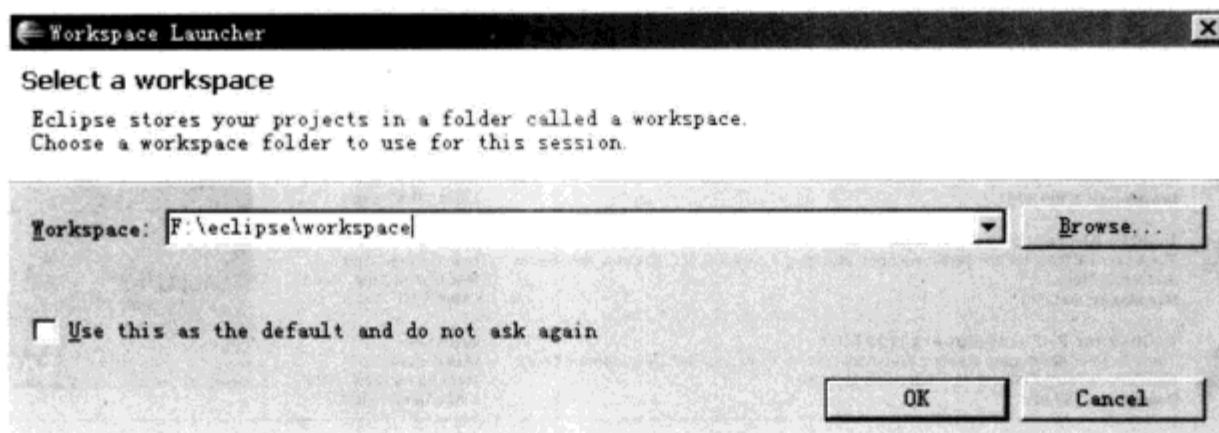


图 1-18 选择工作空间

此时单击【OK】按钮，完成 Eclipse 的安装。

3. 安装 Android SDK

完成 JDK 和 Eclipse 的安装后，接下来需要下载安装 Andriod 的 SDK，具体步骤如下：

step 01 打开 Android 开发者社区网址 <http://developer.android.com/>，然后转到 SDK 下载页面（网址是 http://developer.android.com/sdk/1.5_r1/index.html），如图 1-19 所示。



Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r04-windows.zip	23069119 bytes	c48b407de852ba483869f17337e90997
Mac OS X (intel)	android-sdk_r04-mac_86.zip	19657927 bytes	b08512765aa9b0369bb9b8fecdf763e3
Linux (i386)	android-sdk_r04-linux_86.tgz	15984887 bytes	e64b08fd9da84f4c4ae77564fe4eaee

图 1-19 SDK 下载页面

step 02 在此选择用于 Windows 平台的“android-sdk_r04-windows.zip”，下载页面如图 1-20 所示。

step 03 选中“*I agree to the terms of the Android SDK License Agreement*”选项，单击【Download】按钮开始下载。下载后解压压缩文件，假设下载后的文件解压存放 在“F:\android\”目录下，并将其 tools 目录的绝对路径添加到系统的 PATH 中，具体操作步骤如下所示。

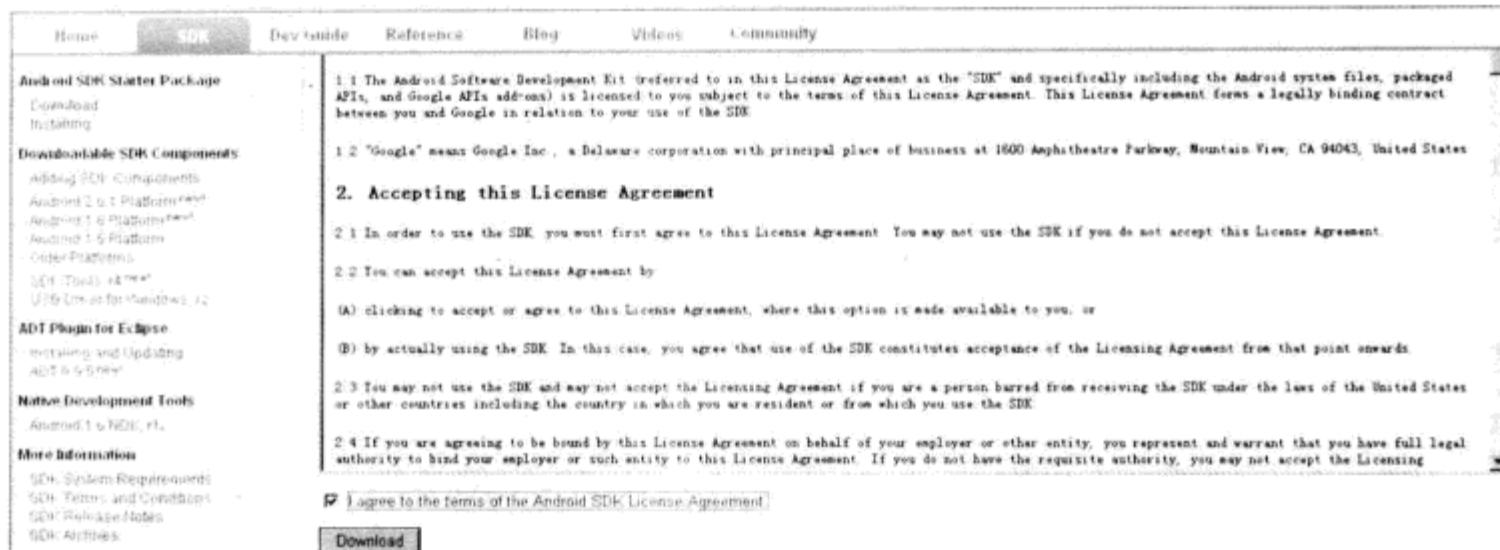


图1-20 Android SDK下载页面

① 鼠标右键依次单击【我的电脑】|【属性】|【高级】，单击下方的“环境变量”，在“系统变量”处选择新建，在变量名处输入 SDK_HOME，变量值中输入刚才的目录，比如笔者的就是 F:\android-sdk-windows，如图 1-21 所示。

② 找到 PATH 的变量，双击或点击编辑，在变量值最前面加上 %SDK_HOME%\tools;，如图 1-22 所示。

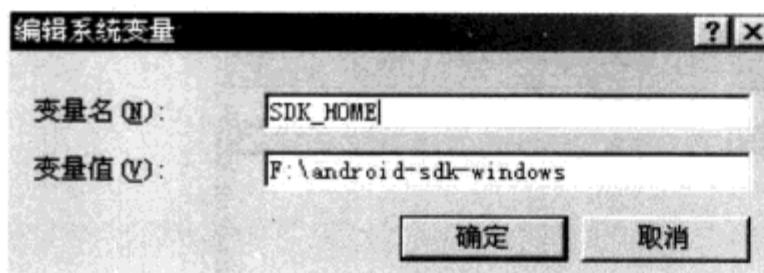


图1-21 设置系统变量

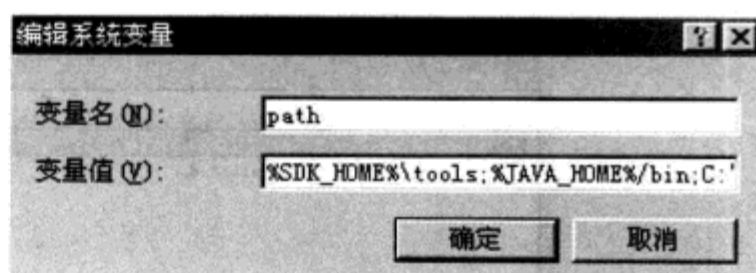


图1-22 设置系统变量

③ 再依次单击【开始】|【运行】，在运行框中输入“cmd”并按下回车键，在打开的 CMD 窗口中输入一个测试命令，例如 android -h，如果显示如图 1-23 所示的提示信息，则说明安装成功。

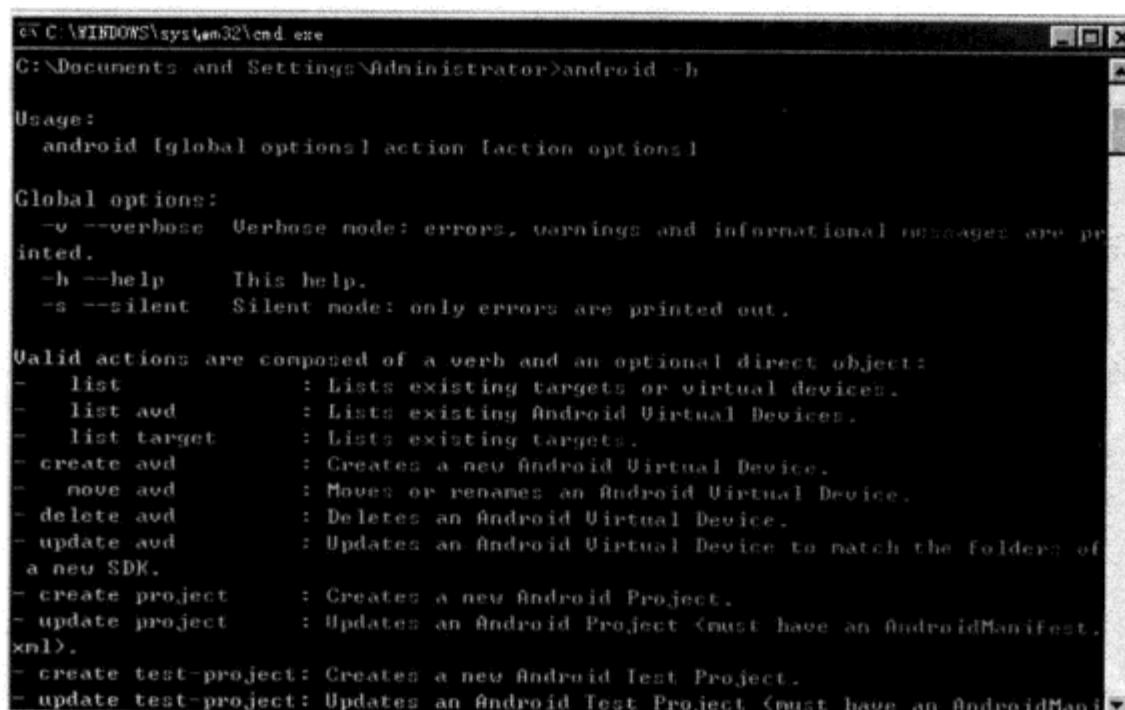


图1-23 设置系统变量

4. 安装 ADT

Android 为 Eclipse 定制了一个专用插件 Android Development Tools (ADT)，此插件为用户提供了一个强大的开发 Android 应用程序的综合环境。ADT 扩展了 Eclipse 的功能，可以让用户快速地建立 Android 项目，创建应用程序界面。要安装 Android Development Tools plug-in，需要首先打开 Eclipse IDE。然后进行如下操作。

step 01 打开 Eclipse 后，依次单击菜单栏中的【Help】|【Install New Software...】选项，如图 1-24 所示。

step 02 在弹出的对话框中单击【Add】按钮，如图 1-25 所示。

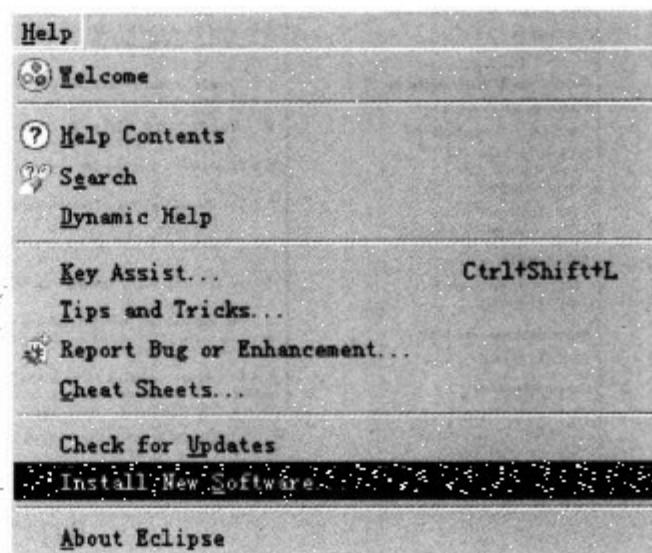


图 1-24 添加插件

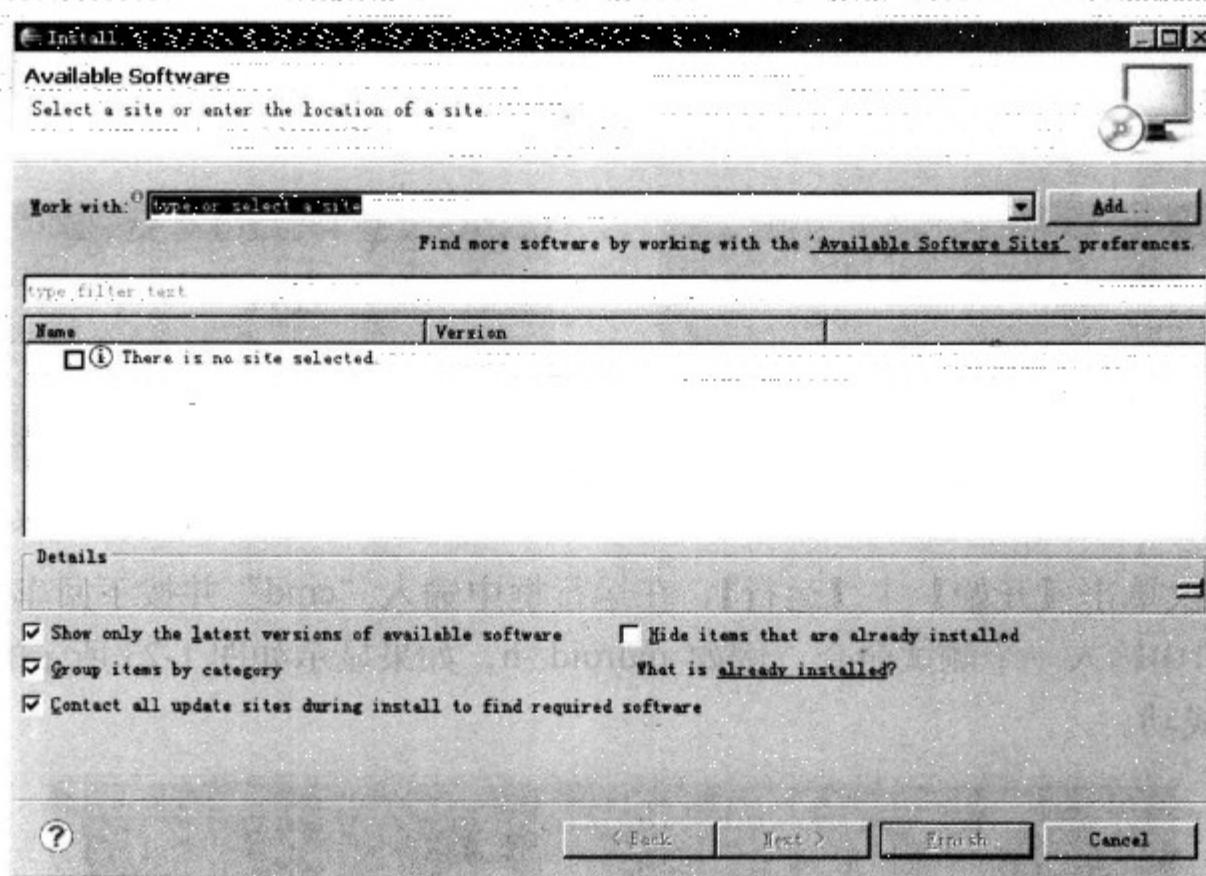


图 1-25 添加插件

step 03 在弹出的“Add Site”对话框中分别输入名字和地址，名字可以自己命名，例如“123”，但是在 Location 中必须输入插件的网络地址“<http://dl-ssl.google.com/Android/eclipse/>”，单击【OK】按钮，如图 1-26 所示。

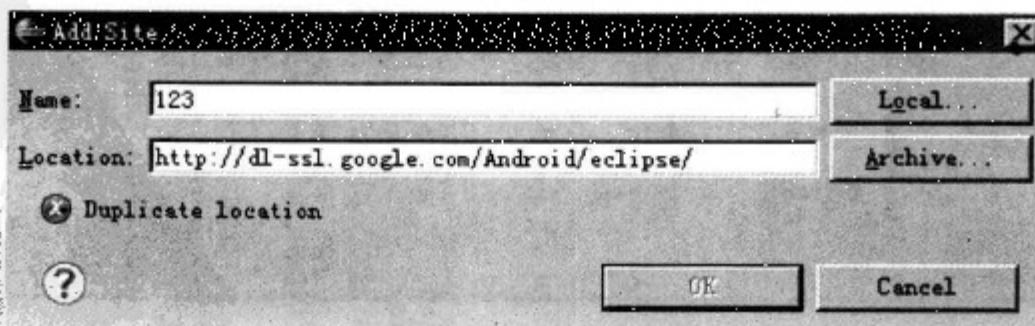


图 1-26 设置地址

step 04 单击【OK】按钮，此时在“Install”界面将会显示系统中可用的插件，如图 1-27 所示。

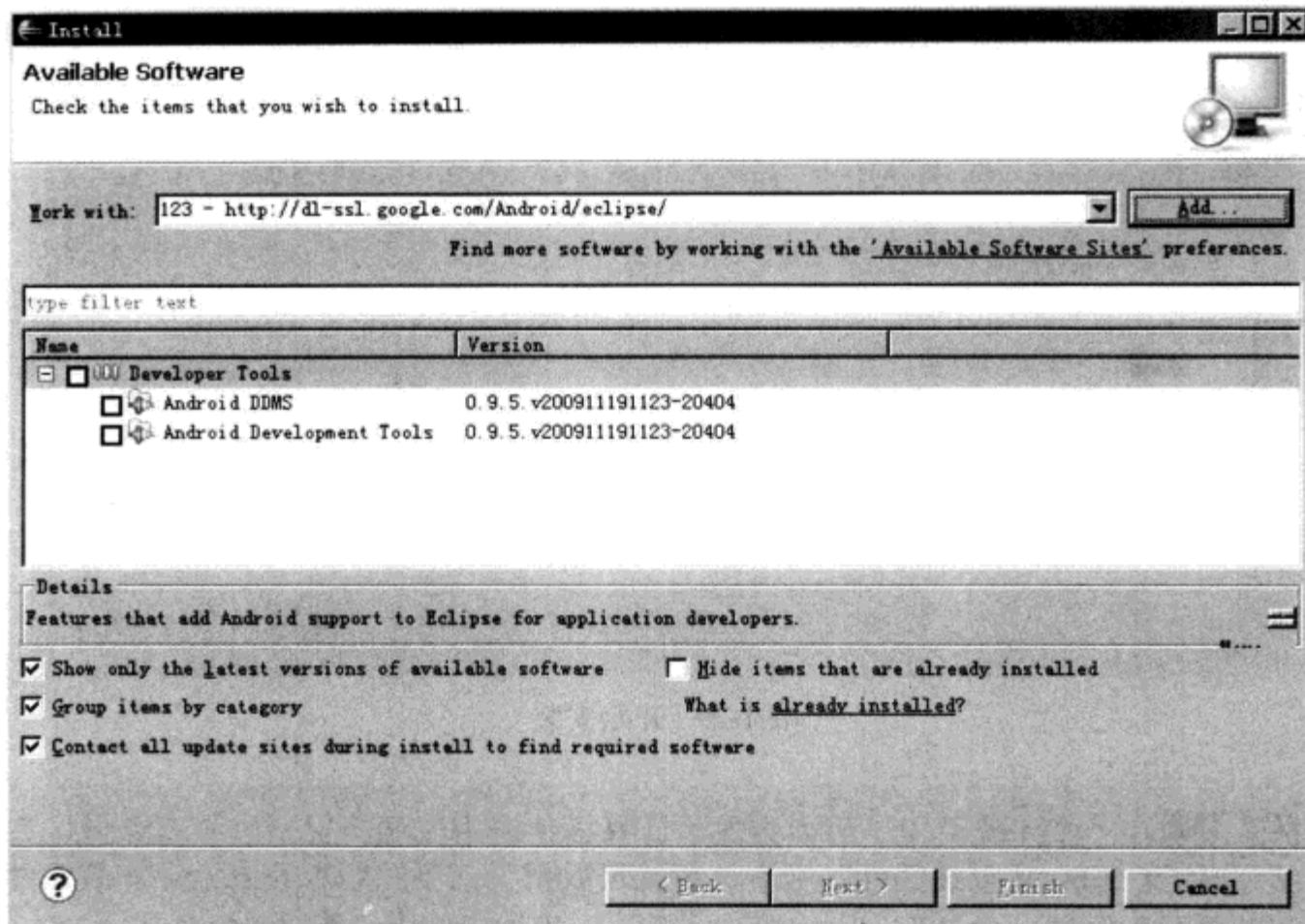


图1-27 插件列表

step 05 选中“Android DDMS”和“Android Development Tools”，然后单击【Next】按钮来到安装界面，如图 1-28 所示。

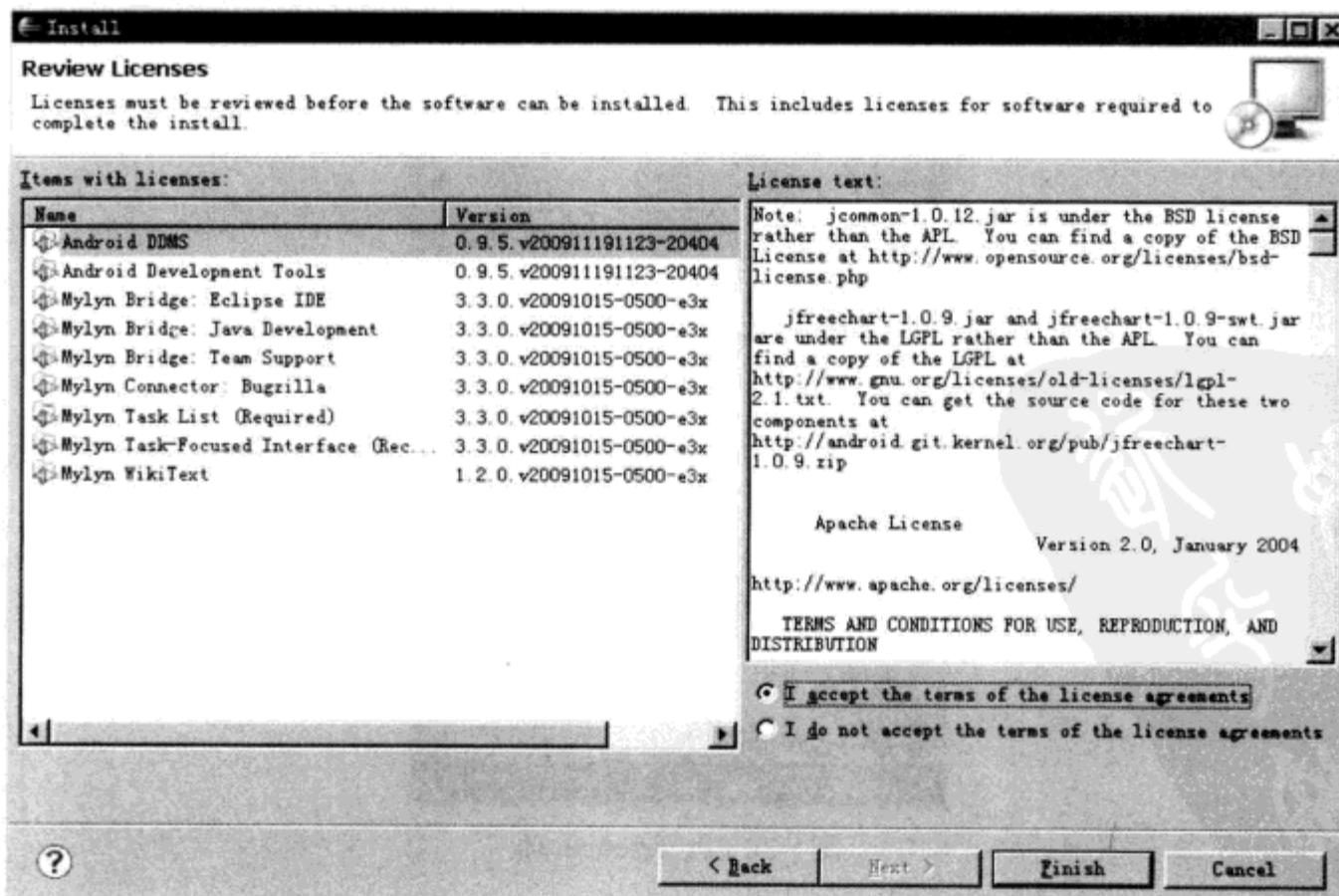


图1-28 插件安装界面

step 06 选择“*I accept*”选项，单击【Finish】按钮，开始进行安装，如图 1-29 所示。

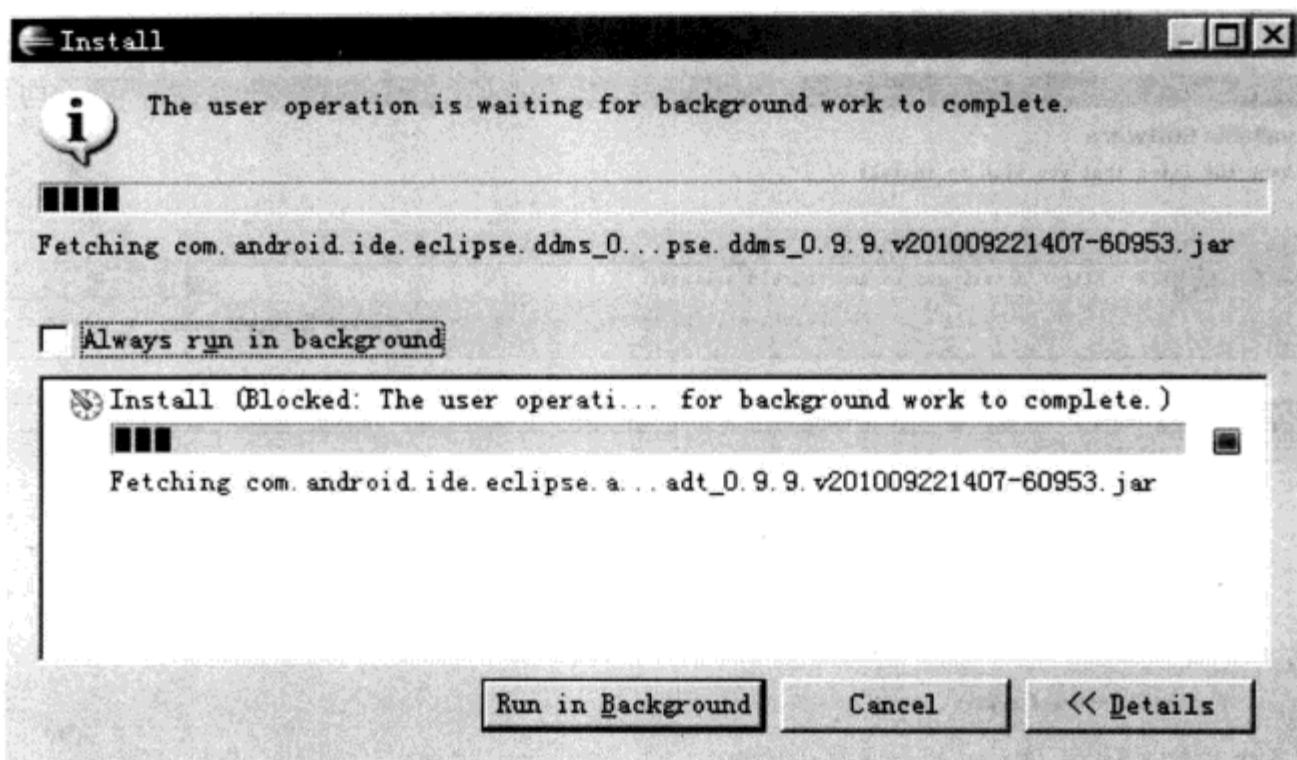


图1-29 开始安装



注意 在上个步骤中，可能会发生计算插件占用资源情况，过程有点慢。完成后会提示重启Eclipse来加载插件，等重启后就可以用了。不同版本的Eclipse安装插件的方法和步骤是不同的，但是都大同小异，读者可以根据操作提示能够自行解决。

1.3.3 设定Android SDK Home

当完成上述插件装备工作后，此时还不能使用 Eclipse 创建 Android 项目，我们还需要在 Eclipse 中设置 Android SDK 的主目录。

step 01 打开 Eclipse，在菜单中依次单击【Windows】 | 【Preferences】项，如图 1-30 所示。

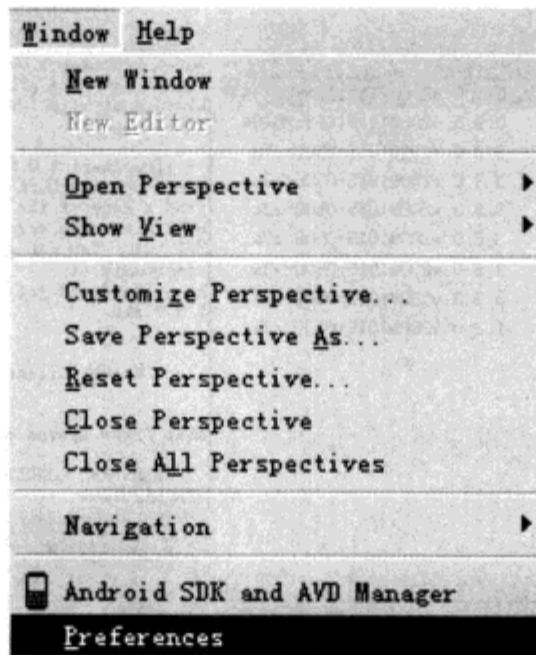


图1-30 Preferences项

step 02 在弹出的界面左侧可以看到“Android”项，选中 Android 后，在右侧设定 Android SDK 所在目录为 SDK Location，单击【OK】按钮完成设置，如图 1-31 所示。

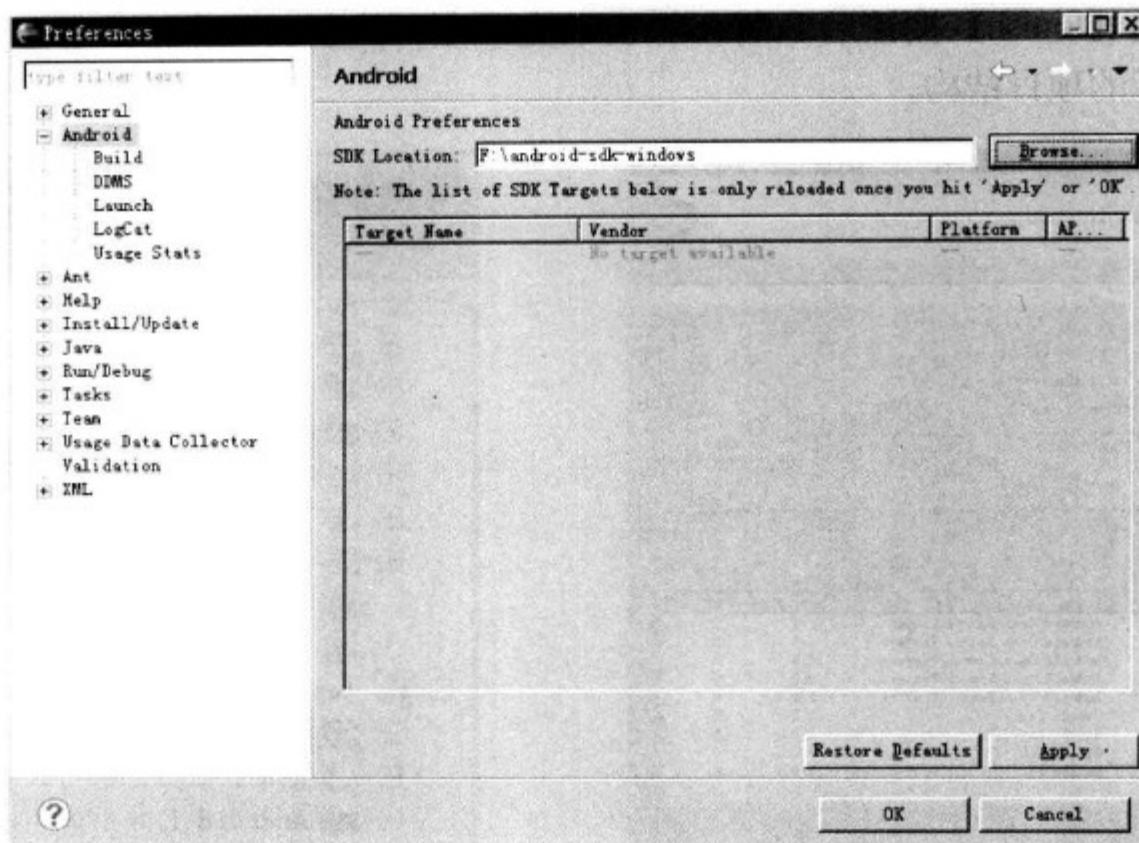


图1-31 Preferences项

1.3.4 验证开发环境

经过前面步骤的讲解，一个基本的Android开发环境算是搭建完成了。都说实践是检验真理的唯一标准，下面通过新建一个项目来验证当前的环境是否可以正常工作。

step 01 打开Eclipse，在菜单中依次选择【File】|【New】|【Project】项，在弹出的对话框上可以看到Android，如图1-32所示。

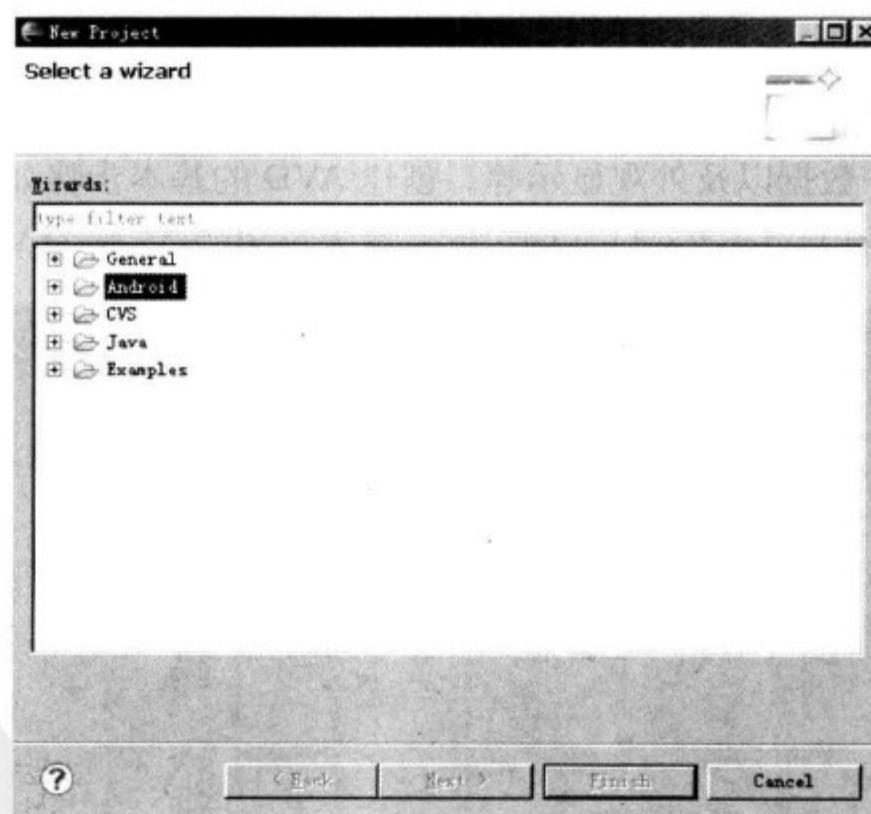


图1-32 新建项目

step 02 在图1-32上选择“Android”，单击【Next】按钮后打开“New Android Project”对话框，在对应的文本框中输入必要的信息，如图1-33所示。

step 03 单击【Finish】按钮后 Eclipse 会自动完成项目的创建工作，最后会看到如图 1-34 所示的项目结构。

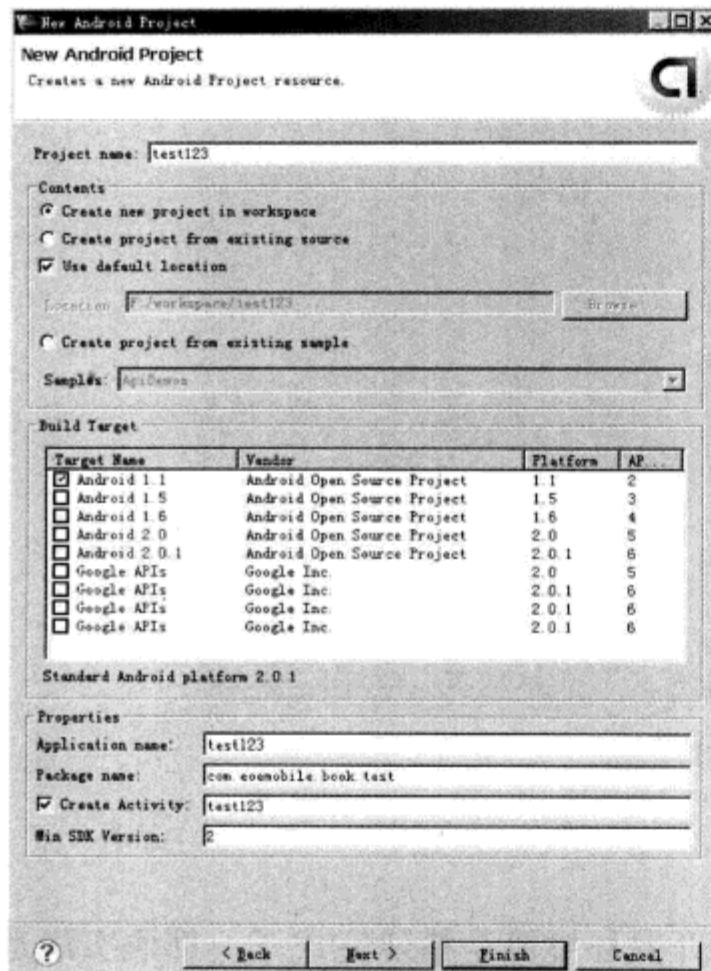


图1-33 “New Android Project”对话框

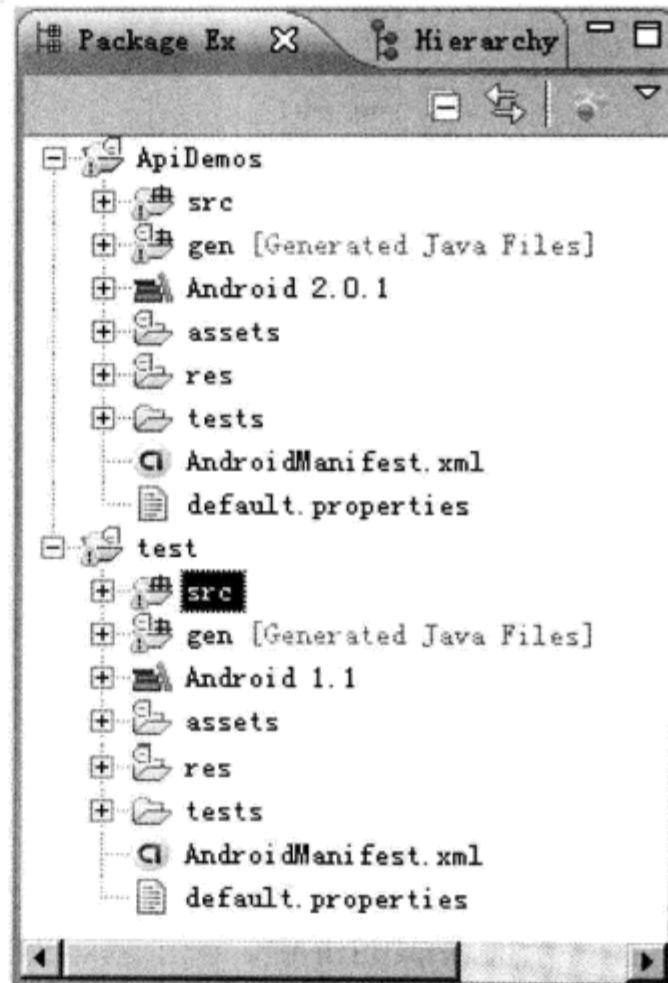


图1-34 项目结构

1.3.5 创建Android虚拟设备(AVD)

AVD 全称为 Android 虚拟设备 (Android Virtual Device)，每个 AVD 模拟了一套虚拟设备来运行 Android 平台，这个平台至少要有自己的内核，系统图像和数据分区，还可以有自己的 SD 卡和用户数据以及外观显示等。创建 AVD 的基本步骤如下所示。

step 01 在 CMD 下输入“android list targets”，查看可用的 Android 平台，如图 1-35 所示。

```
c:\> C:\WINDOWS\system32\cmd.exe  
Microsoft Windows XP [版本 5.1.2600]  
(C) 版权所有 1985-2001 Microsoft Corp.  
  
C:\Documents and Settings\Administrator>android list targets  
Available Android targets:  
id: 1 or "android-2"  
    Name: Android 1.1  
    Type: Platform  
    API level: 2  
    Revision: 1  
    Skins: HUGO (default), HUGO-L, HUGO-P, QUGO-L, QUGO-P  
id: 2 or "android-3"  
    Name: Android 1.5  
    Type: Platform  
    API level: 3  
    Revision: 1  
    Skins: HUGO (default), HUGO-L, HUGO-P, QUGO-L, QUGO-P  
id: 3 or "android-4"  
    Name: Android 1.6  
    Type: Platform  
    API level: 4  
    Revision: 1  
    Skins: HUGO (default), QUGO, QUGO800, QUGO854  
id: 4 or "android-5"  
    Name: Android 2.0
```

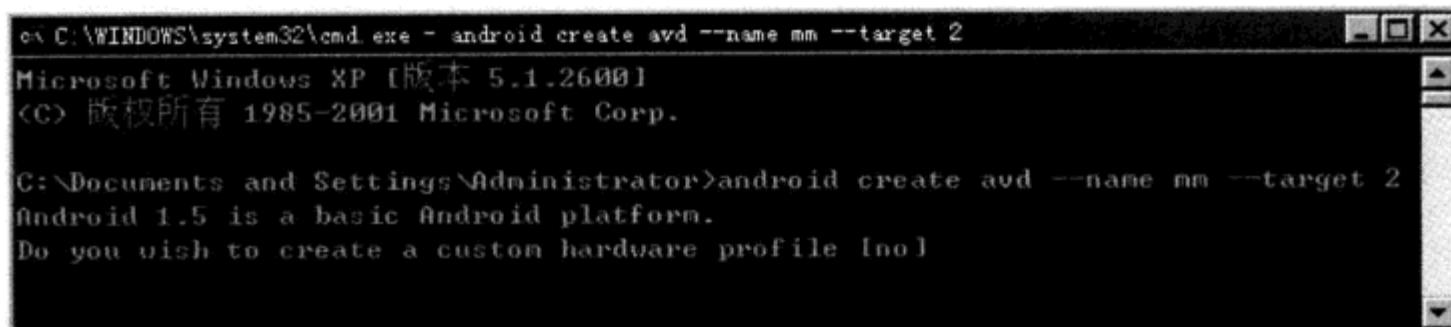
图1-35 CMD界面

在图 1-35 中显示了 4 个 targets, id 分别是 1、2、3、4。

step 02 按照如下格式创建 AVD :

```
android create avd --name <your_avd_name> --target <targetID>
```

其中 “your_avd_name” 是需要创建的 AVD 的名字, 在 CMD 窗口界面中的效果如图 1-36 所示。



```
C:\> C:\WINDOWS\system32\cmd.exe - android create avd --name mm --target 2
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\>Documents and Settings\Administrator>android create avd --name mm --target 2
Android 1.5 is a basic Android platform.
Do you wish to create a custom hardware profile [no]?
```

图1-36 CMD界面

step 03 这样就创建了一个自定义的 AVD (Android Virtual Device), 然后, 只要在 Eclipse 的 Run Configurations 里面指定一个 AVD, 即在 Target 下选中自己定义的这个 AVD, sdk_1_5_version 就可以运行了, 如图 1-37 所示。

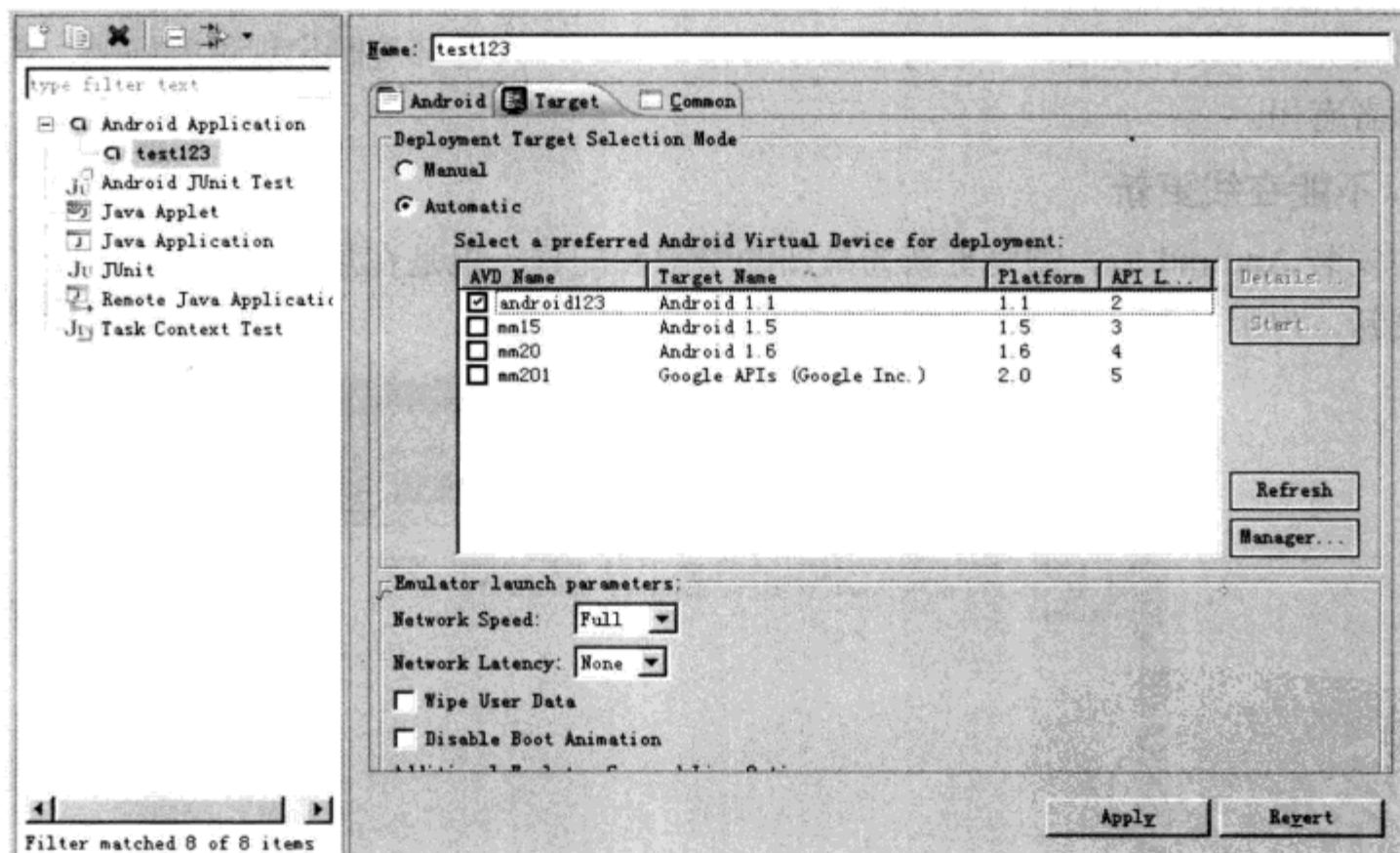


图1-37 选择AVD

step 04 如上我们选择 “sdk_1_5_version”, 单击【Apply】按钮后, 然后单击【Start】按钮弹出 “Launch” 界面, 如图 1-38 所示。

step 05 此时单击【Launch】按钮后将会运行模拟器, 如图 1-39 所示。

关于应用结构分析和讲解, 以及代码的调试部分内容会在本书后面的内容中进行详细的介绍。至此, 在 Windows 平台上的开发环境搭建完成, 安装了运行环境 JDK, 开发工具 Eclipse, Android SDK, 并安装了 ADT 并进行 SDK Home 的配置, 最后创建了一个

Android 虚拟设备 (AVD)。

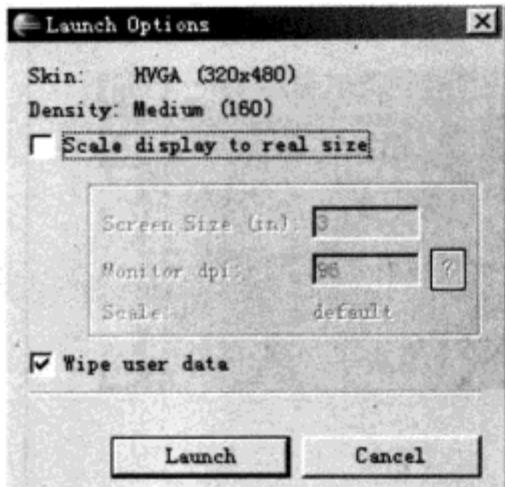


图1-38 “Launch”对话框

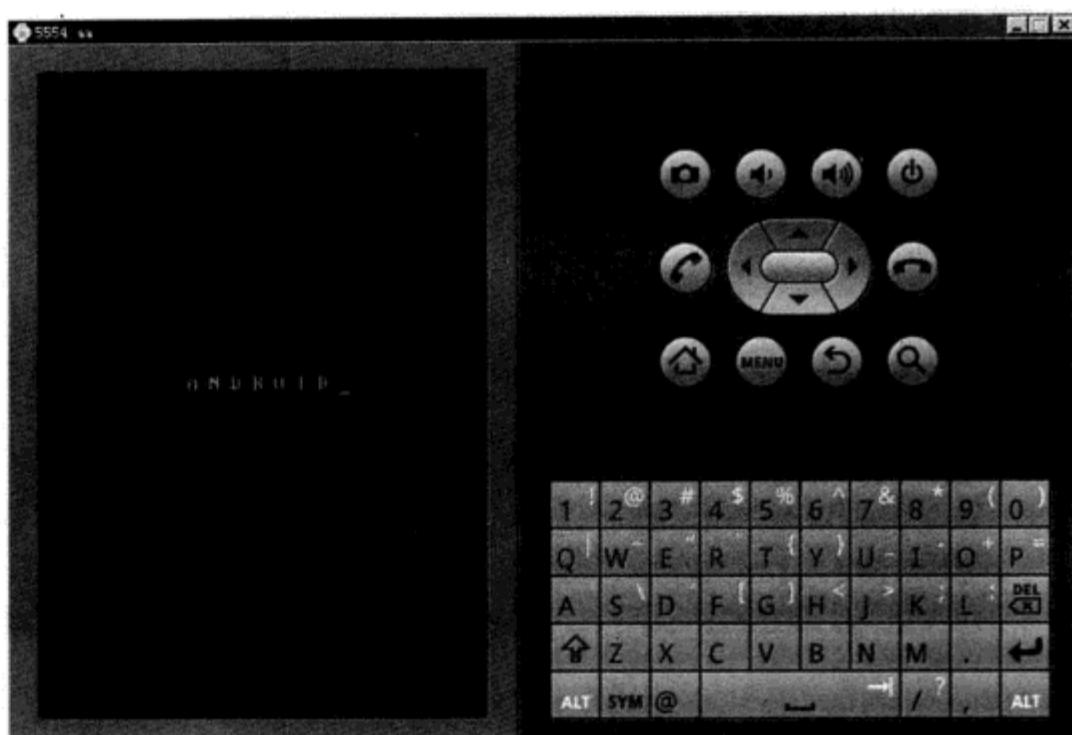


图1-39 模拟运行成功

1.3.6 常见的几个问题

搭建完成开发环境后，下面将总结在搭建 Android SDK 环境时出现过的问题，希望对广大读者有用。

1. 不能在线更新

在安装 Android 后，需要更新为最新的资源和配置。但是在启动 Android 后，经常会不能更新，弹出如图 1-40 所示的错误提示。

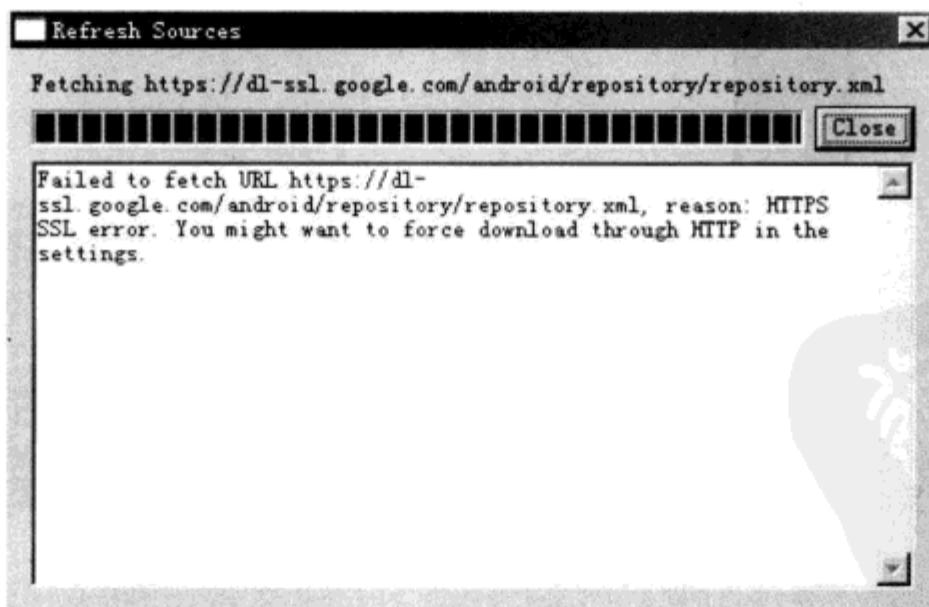


图1-40 不能更新

Android 默认的在线更新地址是 <https://dl-ssl.google.com/android/eclipse/>，但是经常会出现错误。如果此地址不能更新，可以自行设置更新地址，修改为 <http://dl-ssl.google.com/android/repository/repository.xml>。具体操作方法如下。

step 01 单击 Android 左侧的“Available Packages”选项，然后单击下面的【Add Site…】



按钮，如图 1-41 所示。

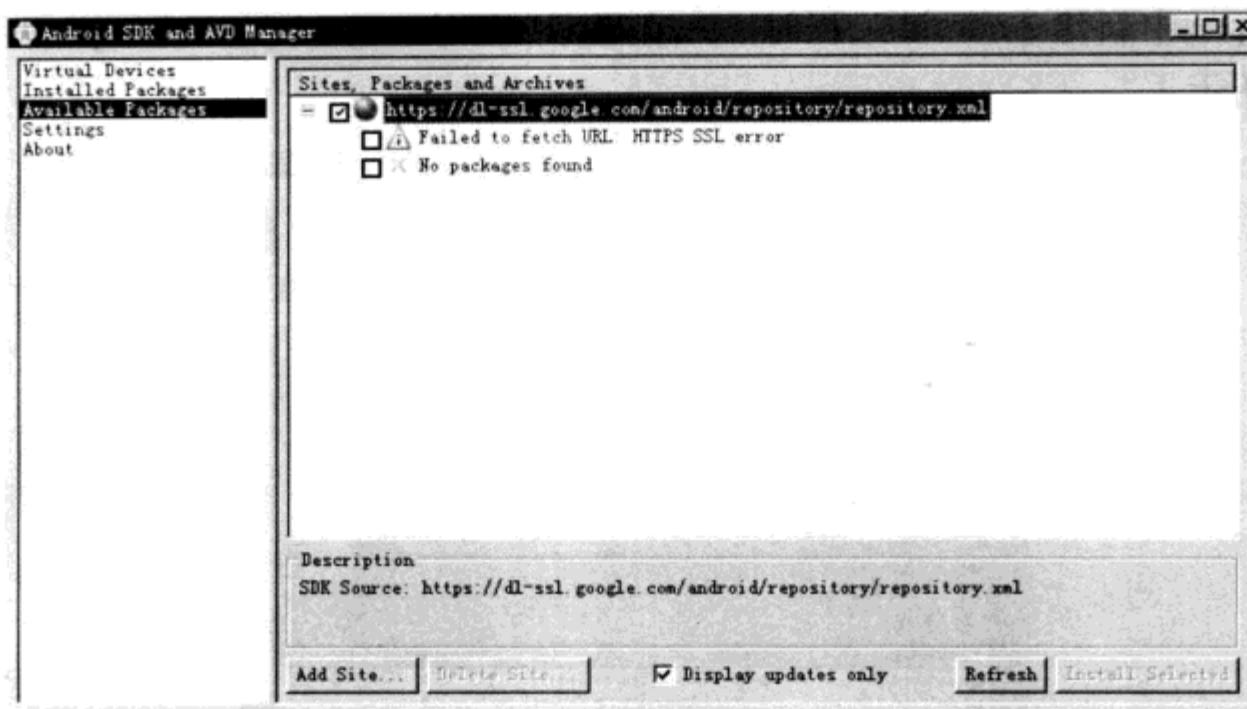


图1-41 “Available Packages”界面

step 02 在弹出的“Add Site URL”对话框中输入下面修改后的地址，如图 1-42 所示。

<http://dl-ssl.google.com/android/repository/repository.xml>

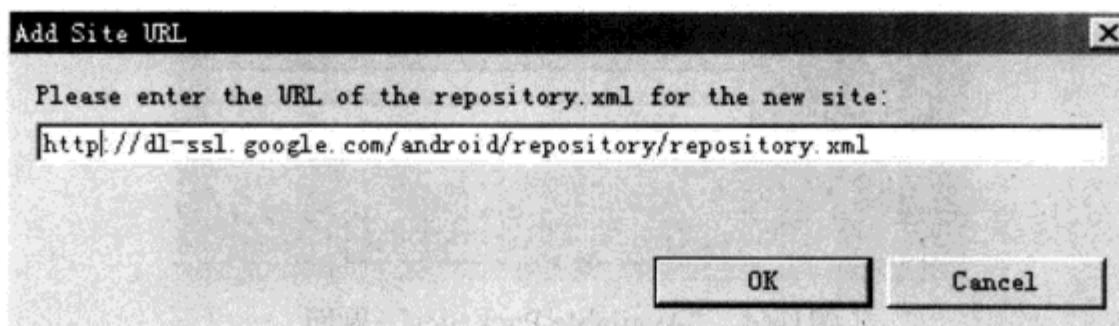


图1-42 “Add Site URL”对话框

step 03 单击【OK】按钮完成设置，此时就可以使用更新功能了，如图 1-43 所示。

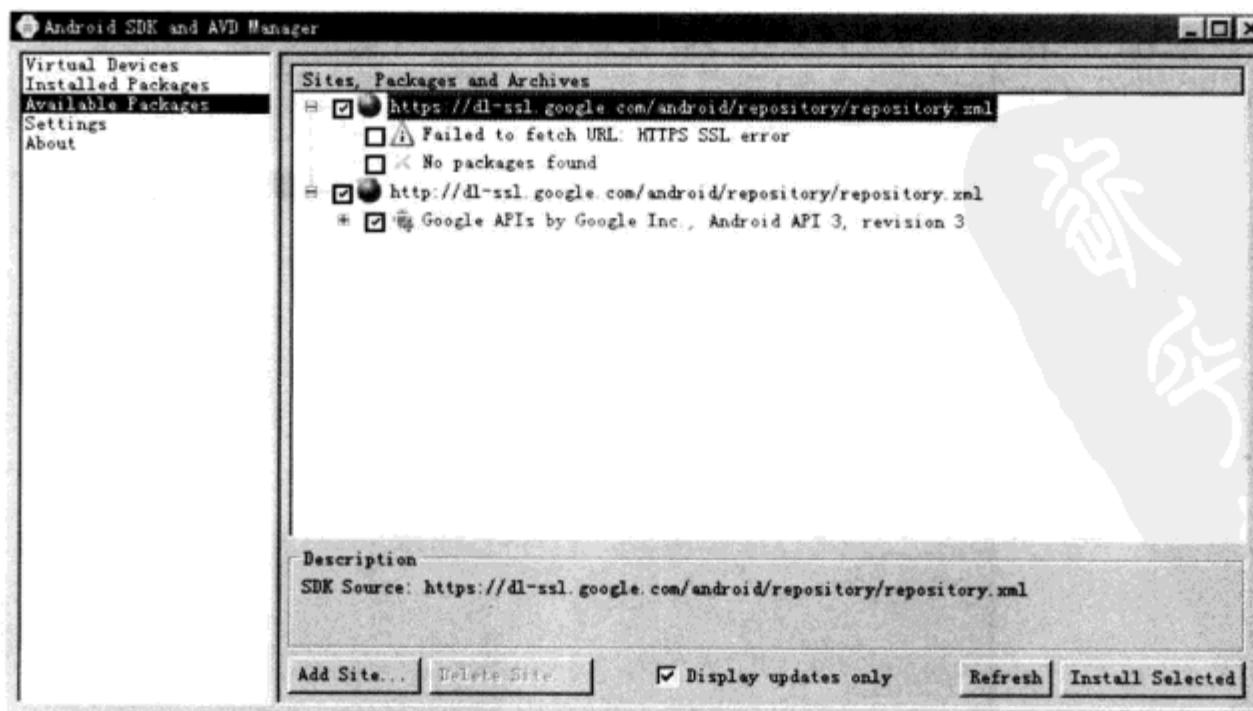


图1-43 “Available Packages”界面

2. “Project name must be specified” 提示

在 Eclipse 中新建 Android 工程时，一直显示“Project name must be specified”提示，如图 1-44 所示。

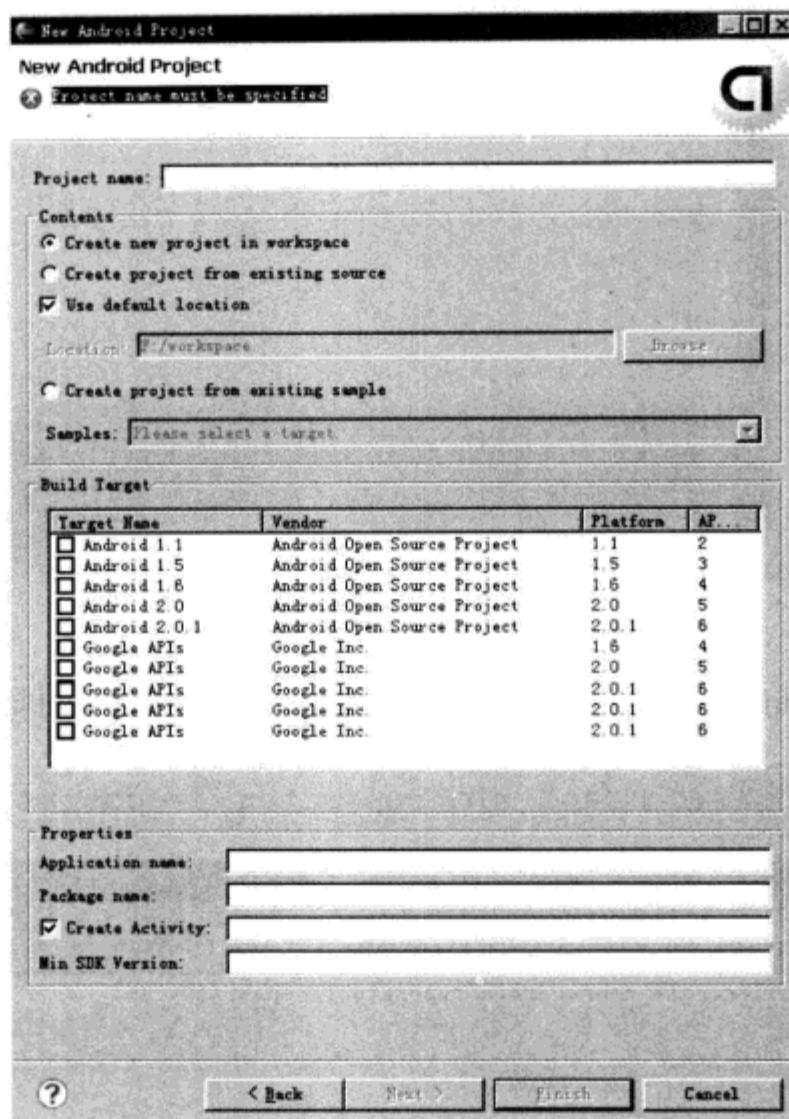


图 1-44 “Available Packages” 界面

造成上述问题的原因是 Android 没有更新完成，需要进行完全更新。具体方法如下所示。

step 01 打开 Android，选择左侧的“Installed Packages”，如图 1-45 所示。

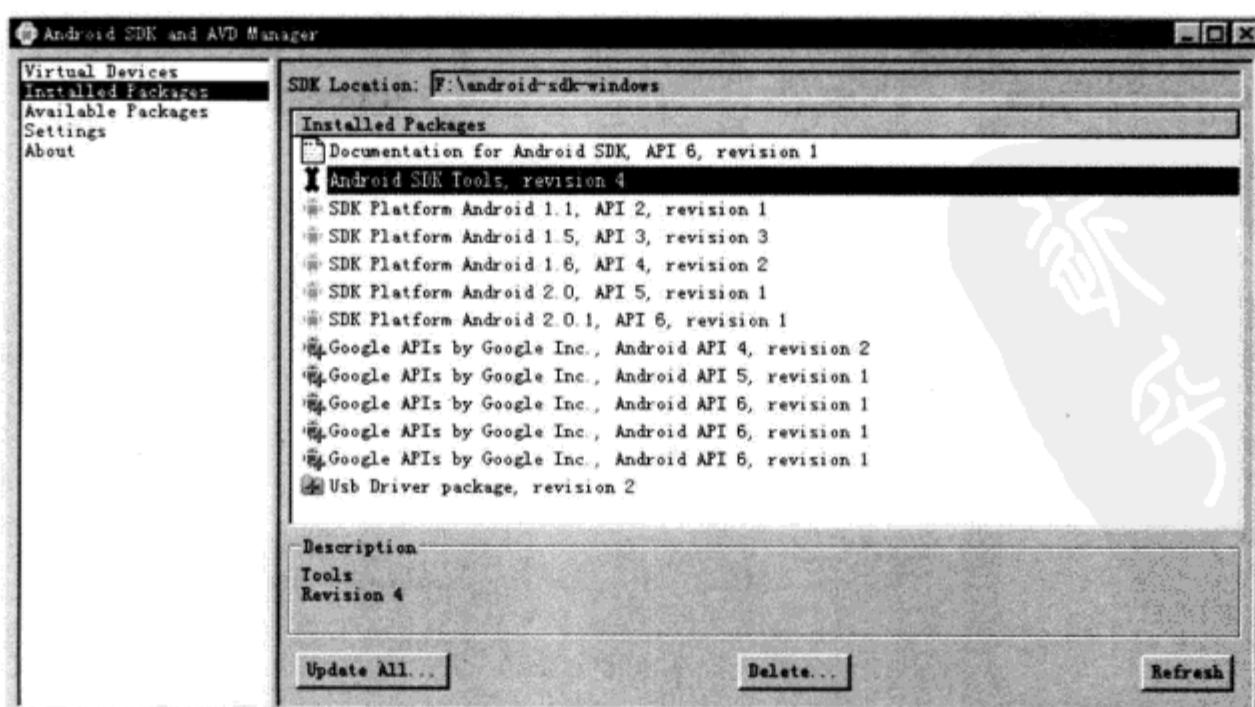


图 1-45 “Available Packages” 界面

step 02 右侧列表中选择“Android SDK Tools ,revision4”，在弹出窗口中选择“Accept”，最后单击【Install Accepted】按钮开始安装更新，如图 1-46 所示。

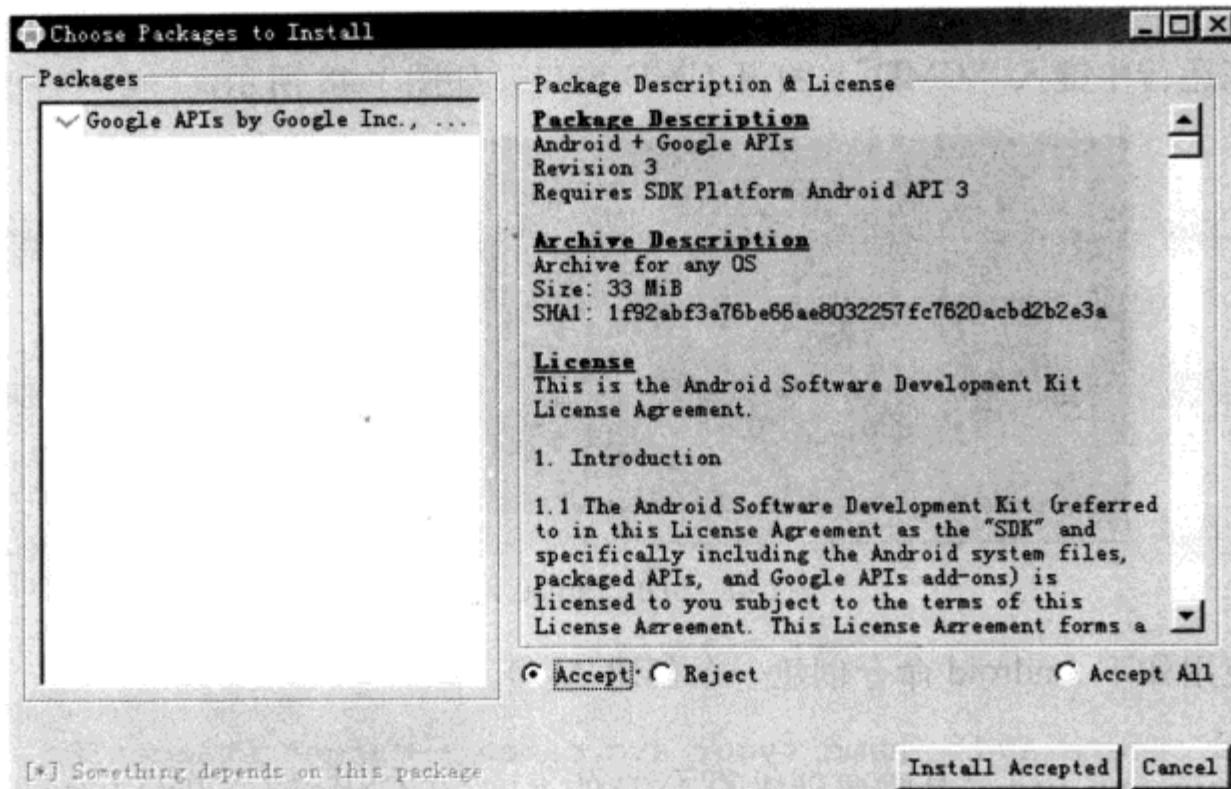


图 1-46 单击【Install Accepted】按钮

3. Target 列表中没有 Target 选项

通常来说，当 Android 开发环境搭建完毕后，在 Eclipse 工具栏中依次单击【Window】|【Preference】，单击左侧的“Android”项后会在“Preference”中显示存在的 SDK Targets，如图 1-47 所示。

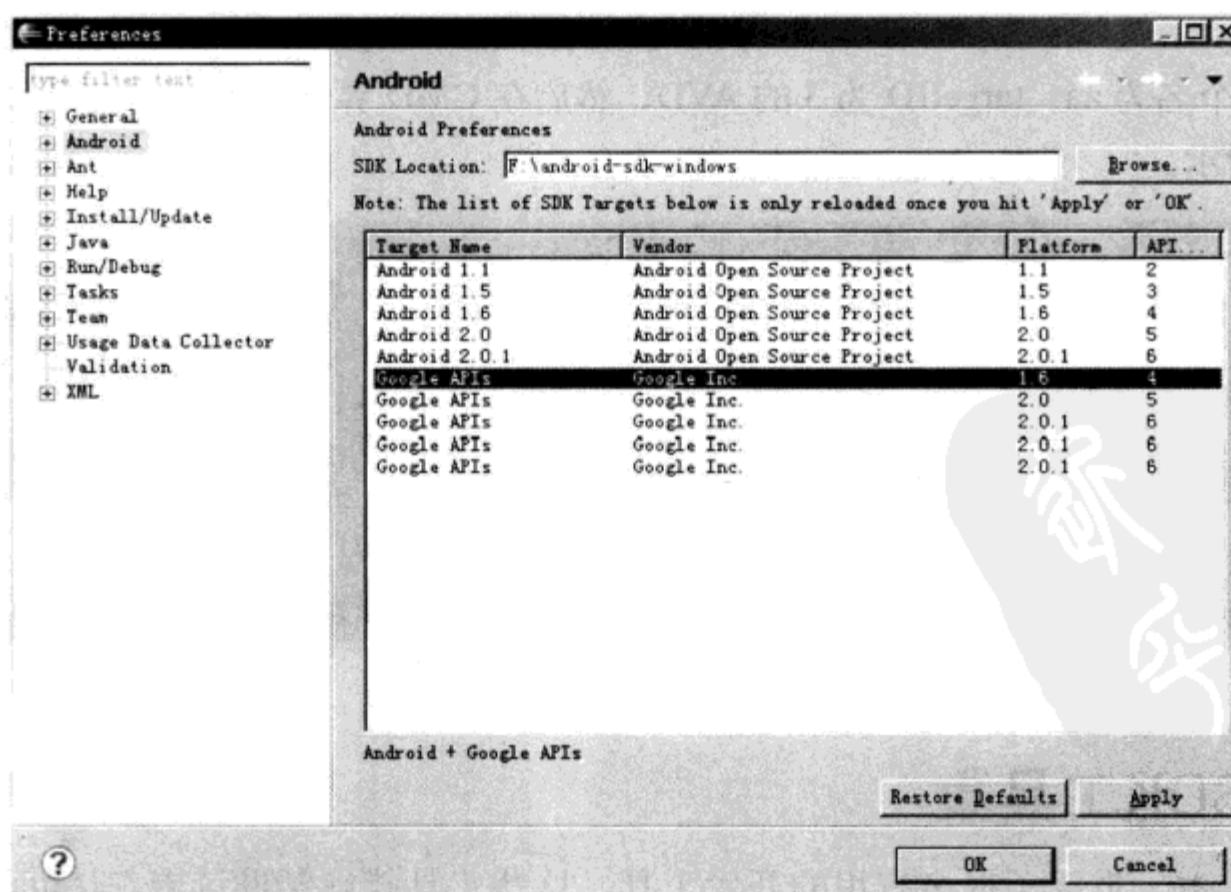


图 1-47 SDK Targets列表

但是往往由于各种原因，会不显示 SDK Targets 列表，并输出“Failed to find an AVD”

compatible with target” 错误提示。

造成上述问题的原因是没有创建 AVD 成功，此时需要我们手工安装来解决这个问题，当然前提是 Android 更新完毕。具体解决方法如下所示。

step 01 在运行中键入“CMD”，打开 CMD 窗口，如图 1-48 所示。

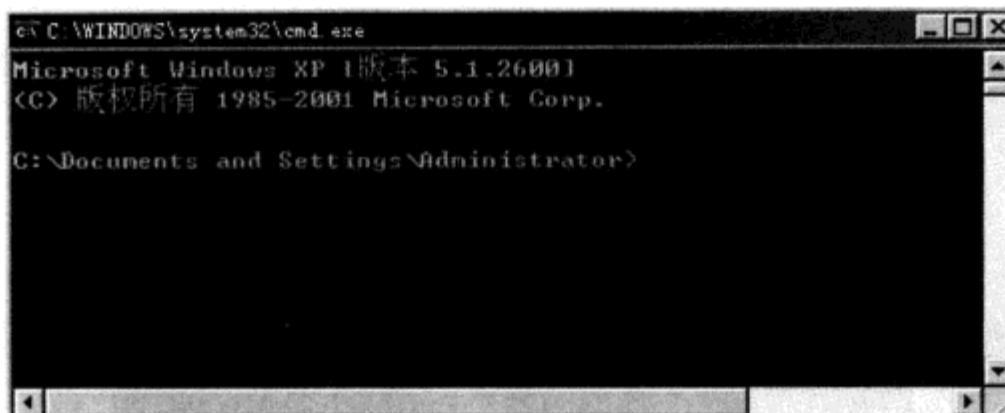


图1-48 SDK Targets列表

step 02 使用如下 Android 命令创建一个创建 AVD。

```
android create avd --name <your avd name> --target <targetID>
```

其中“your_avd_name”是需要创建的 AVD 的名字，在 CMD 窗口界面中如图 1-49 所示。

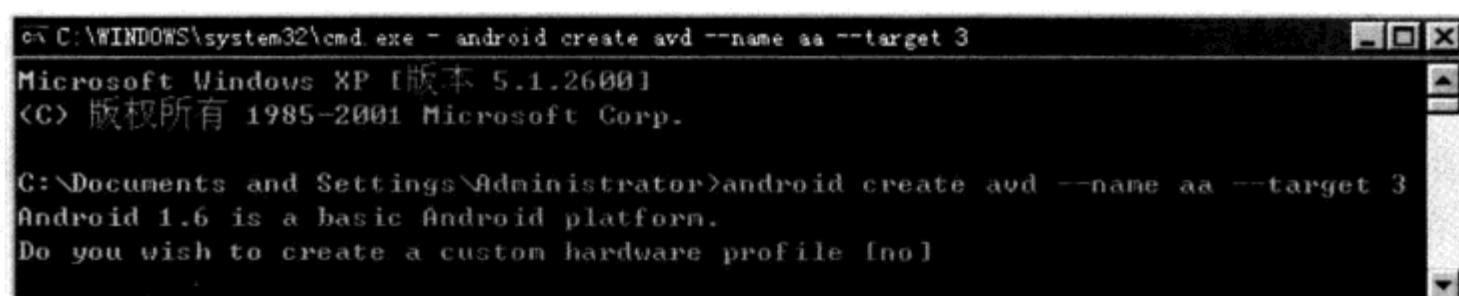


图1-49 CMD界面

创建一个名为 aa，targetID 为 3 的 AVD，然后在 CMD 界面中输入“n”，即完成操作，如图 1-50 所示。

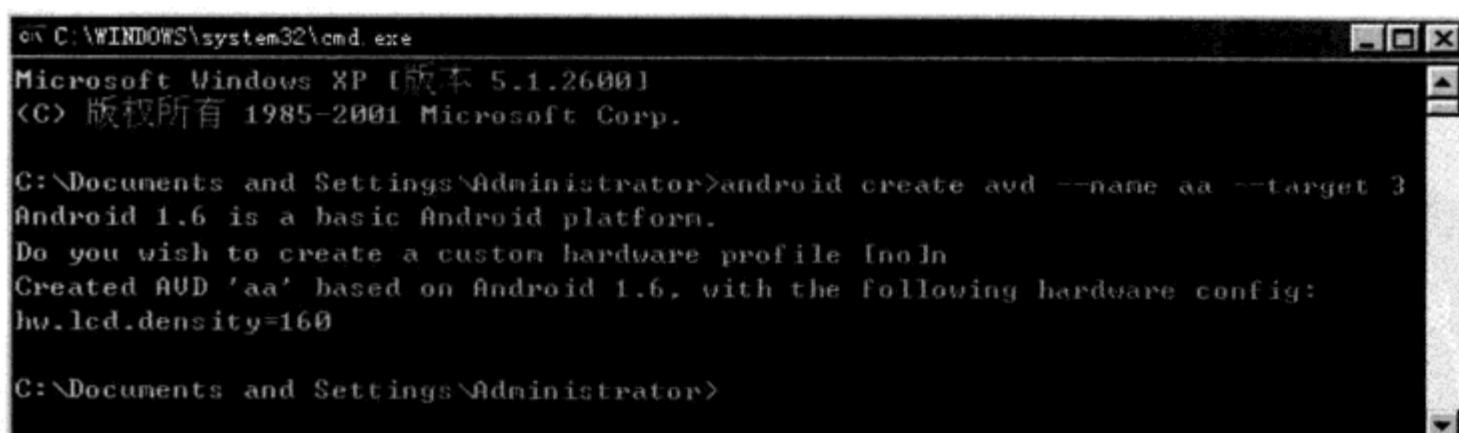


图1-50 CMD界面

1.3.7 SDK工具集

在 SDK 中，集成了很多有用的开发工具，这些工具能够帮助读者在 Android 平台上开发出有用的应用程序。在 Android SDK 中，最为有用的是 Android 模拟器和 Eclipse 的 Android 开发插件，但是 SDK 中也包含了各种在模拟器上用于调试、打包和安装的工具，



能够针对不同的场景带来很多的方便。在下面的内容中，我们将简要介绍这些工具的基本使用知识。

- ◆ **Android 模拟器**

模拟器是运行在计算机上的虚拟移动设备，有关模拟器的基本知识将在下节中进行详细介绍，在此不再讲解。

- ◆ **集成开发插件 ADT**

Android 为 Eclipse 定制了一个插件，即 Android Development Tools (ADT)，这个插件为用户提供一个强大的综合环境用于开发 Android 应用程序。ADT 扩展了 Eclipse 的功能，可以让用户快速地建立 Android 项目，创建应用程序界面，在基于 Android 框架 API 的基础上添加组件，以及用 SDK 工具集调试应用程序，甚至导出签名（或未签名）的 APKs 以便发行应用程序。

- ◆ **调试监视服务 ddms.bat**

调试监视服务 ddms.bat 集成在 Dalvik (Android 平台的虚拟机) 中，用于管理运行在模拟器或设备上的进程，并协助调试工作。它可以去除一些进程，选择一个特定的程序来调试，生成跟踪数据，查看堆和线程数据，对模拟器或设备进行屏幕快照等操作。

- ◆ **Android 调试桥 adb.exe**

Android 调试桥 (adb) 是多种用途的工具，该工具可以帮助用户管理设备或模拟器的状态。可以通过下面的几种方法加入 adb。

- (1) 在设备上运行 shell 命令；
- (2) 通过端口转发来管理模拟器或设备；
- (3) 从模拟器或设备上复制来或复制走文件。

- ◆ **Android 资源打包工具 aapt.exe**：此工具可以创建 apk 文件，在 apk 文件中包含了 Android 应用程序的二进制文件和资源文件。
- ◆ **Android 接口描述语言 aidl.exe**：用于生成进程间接口代码。
- ◆ **SQLite3 数据库 sqlite3.exe**：Android 可以创建和使用 SQLite 数据文件，开发人员和使用用户乐意方便地访问这些 SQLite 数据文件。
- ◆ **跟踪显示工具**：可以生成跟踪日志数据的图形分析视图，这些跟踪日志数据由 Android 应用程序产生。
- ◆ **创建 SD 卡工具**：用于创建磁盘镜像，此镜像可以在模拟器上模拟外部存储卡，例如常见的 SD 卡。
- ◆ **DX 工具 (dx.bat)**：将 class 字节码重写为 Android 字节码（被存储在 dex 文件中）。
- ◆ **生成 Ant 构建文件 (activitycreator.bat)**。

activitycreator.bat 是一个脚本，用于生成 Ant 构建文件。Ant 构建文件用于编译 Android 应用程序，如果在安装 ADT 插件的 Eclipse 环境下开发，则就不需要这个脚本了。

- ◆ **Android 虚拟设备**

在 Android SDK1.5 版以后的 Android 开发中，必须创建至少一个 AVD，AVD 全称为 Android 虚拟设备 (Android Virtual Device)，每个 AVD 模拟了一套虚拟设备来运行



Android 平台，这个平台至少要有自己的内核、系统图像和数据分区，还可以有自己的 SD 卡和用户数据以及外观显示等。

1.4 Android 模拟器

我们都知道程序开发需要调试，只有经过调试之后才能知道开发的程序是否正确运行。作为一款手机系统，怎么样才能在电脑平台之上调试Android程序呢？不用担心，谷歌为我们提供了模拟器来解决用户担心的问题。所谓模拟器，就是指在电脑上模拟Android系统，可以用这个模拟器来调试并运行开发的Android程序。开发人员不需要一个真实的Android手机，只通过电脑即可模拟运行一个手机，即可开发出应用在手机上面程序。

1.4.1 Android 模拟器简介

对于 Android 程序的开发者来说，模拟器的推出给开发者在开发上和测试上带来了很大的便利。无论在 Windows 下还是 Linux 下，Android 模拟器都可以顺利运行，并且官方提供了 Eclipse 插件，可将模拟器集成到 Eclipse 的 IDE 环境。当然，用户也可以从命令行启动 Android 模拟器。

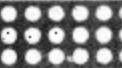
获取模拟器的方法非常简单，用户既可以从官方站点 (<http://developer.Android.com/>) 免费下载单独的模拟器，也可以先下载 Android SDK 后，解压后在其 SDK 的根目录下有一个名为“tools”文件夹，此文件夹下包含了完整的模拟器和一些非常有用的工具。

Android SDK 中包含的模拟器的功能非常齐全，电话本、通话等功能都可正常使用（当然用户没办法真的从这里打电话）。甚至其内置的浏览器和 Maps 都可以联网。用户可以使用键盘输入，鼠标点击模拟器按键输入，甚至还可以使用鼠标点击、拖动屏幕进行操纵。

1.4.2 模拟器和真机究竟有何区别

当然 Android 模拟器不能完全替代真机，具体来说有如下差异：

- ◆ 模拟器不支持呼叫和接听实际来电；但可以通过控制台模拟电话呼叫（呼入和呼出）；
- ◆ 模拟器不支持 USB 连接；
- ◆ 模拟器不支持相机 / 视频捕捉；
- ◆ 模拟器不支持音频输入（捕捉），但支持输出（重放）；
- ◆ 模拟器不支持扩展耳机；
- ◆ 模拟器不能确定连接状态；
- ◆ 模拟器不能确定电池电量水平和交流充电状态；
- ◆ 模拟器不能确定 SD 卡的插入 / 弹出；
- ◆ 模拟器不支持蓝牙。



1.4.3 模拟器简单总结

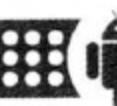
要正确地启动Android模拟器，必须先要创建一个AVD（Android Virtual Device，虚拟设备），读者可以利用AVD创建基于不同版本的模拟器。在此对Android模拟器的参数进行简单总结，其参数格式如下：

```
emulator [option] [-qemu args]
```

其中，option选项的具体说明如表1-2所示。

表1-2 模拟器选项

选项	功能描述
-sysdir <dir>	为模拟器在<dir>目录中搜索系统硬盘镜像
-system <file>	为模拟器从<file>文件中读取初始化系统镜像
-datadir <dir>	设置用户数据写入的目录
-kernel <file>	为模拟器设置使用指定的模拟器内核
-ramdisk <file>	设置内存RAM镜像文件（默认为<system>/ramdisk.img）
-image <file>	废弃，使用 -system <file> 替代
-init-data <file>	设置初始化数据镜像（默认为 <system>/userdata.img）
-initdata <file>	和“-init-data <file>”使用方法一致
-data <file>	设置数据镜像（默认为 <datadir>/userdata-qemu.img）
-partition-size <size>	system/data 分区容量大小（MB）
-cache <file>	设置模拟器缓存分区镜像（默认为临时文件）
-no-cache	禁用缓存分区
-nocache	与“-no-cache”使用方法相同
-sdcard <file>	指定模拟器SDCard镜像文件（默认为 <system>/sdcard.img）
-wipe-data	清除并重置用户数据镜像（从initdata复制）
-avd <name>	指定模拟器使用Android 虚拟设备
-skindir <dir>	设置模拟器皮肤，在<dir>目录中搜索皮肤（默认为 <system>/skins 目录）
-skin <name>	选择使用给定的皮肤
-no-skin	不适用任何模拟器皮肤
-noskin	使用方法与“-no-skin”相同
-memory <size>	物理RAM内存大小（MB）
-netspeed <speed>	设置最大网络下载、上传速度
-netdelay <delay>	网络时延模拟
-netfast	禁用网络形态



续表

选项	功能描述
-tarce <name>	代码配置可用
-show-kernel	显示内核信息
-shell	在当前终端中使用根Shell命令
-no-jni	Dalvik运行时禁用JNI检测
-nojni	使用方法与” -no-jni” 相同
-logcat <tag>	输出给定tag的Logcat信息
-no-audio	禁用音频支持
-noaudio	与 “-no-audio” 用法相同
-audio <backend>	使用指定的音频backend
-audio-in <backend>	使用指定的输入音频backend
-audoi-out <backend>	使用指定的输出音频backend
-raw-keys	禁用Unicode键盘翻转图
-radio	重定向无线模式接口到个性化设备
-port <port>	设置控制台使用的TCP 端口
-ports <consoleport>,<adbport>	设置控制台使用的TCP端口和ADB调试桥使用的TCP端口
-onion <image>	在屏幕上层使用覆盖PNG图片
-onion-alpha <%age>	指定上层皮肤半透明度
-onion-rotation 0 1 2 3	指定上层皮肤旋转
-scale <scale>	调节模拟器窗口尺寸 (三种: 1.0-3.0、dpi、auto)
-dpi-device <dpi>	设置设备的resolution (dpi单位) (默认165)
-http-proxy <proxy>	通过一个HTTP或HTTPS代理来创建TCP连接
-timezone <timezone>	使用给定的时区, 而不是主机默认的
-dns-server <server>	在模拟系统上使用给定的DNS 服务
-cpu-delay <cpudelay>	调节CPU模拟
-no-boot-anim	禁用动画来快速启动
-no-window	禁用图形化窗口显示
-version	显示模拟器版本号
-report-console <socket>	向远程socket报告控制台端口
-gps <device>	重定向GPS导航到个性化设备
-keyset <name>	指定按键设置文件名
-shell-serial <device>	根 shell的个性化设备
-old-system	支持旧版本 (pre 1.4) 系统镜像
-tcpdump <file>	把网络数据包捕获到文件中
-bootchart <timeout>	bootcharting可用



续表

选项	功能描述
-qemu args....	向qemu传递参数
-qemu -h	显示qemu帮助
-verbose	和“-debug-init”相同
-debug <tags>	可用、禁用调试信息
-debug-<tag>	使指定的调试信息可用
-debug-no-<tag>	禁用指定的调试信息
-help	打印出该帮助文档
-help-<option>	打印出指定option的帮助文档
-help-disk-images	关于硬盘镜像帮助
-help-keys	支持按钮捆绑（手机快捷键）
-help-debug-tags	显示出-debug <tag>命令中的tag可选值
-help-char-devices	个性化设备说明
-help-environment	环境变量
-help-keyset-file	指定按键绑定设置文件
-help-virtula-device	虚拟设备管理
-help-sdk-images	当使用SDK时关于硬盘镜像的信息
-help-build-images	当构建Android时，关于硬盘镜像的信息
-help-all	打印出所有帮助

1.5 纵览Android体系

本节将简要讲解 Android 体系的组成，介绍 Android 应用框架的基础性知识，为后面高级知识的学习打下基础。

1.5.1 简析Android安装文件

当我们下载并安装 Android 后，会在其安装目录中看到一些安装文件。安装 Android SDK 后，其安装目录的具体结构如图 1-51 所示。

- ◆ add-ons：里面包含了官方提供的 API 包，最为主要的是 Map 的 API。
- ◆ docs：里面包含了文档，即帮助文档和说明文档。
- ◆ platforms：针对每个版本的 SDK 版本提供了与其对应的 API 包以及一些示例文件，其中包含了各个版本的 Android，如图 1-52 所示。

- ◆ temp：里面包含了一些常用的文件模板。
- ◆ tools：包含了一些通用的工具文件。
- ◆ usb_driver：包含了 AMD64 和 X86 下的驱动文件。
- ◆ SDK Setup.exe：Android 的启动文件。

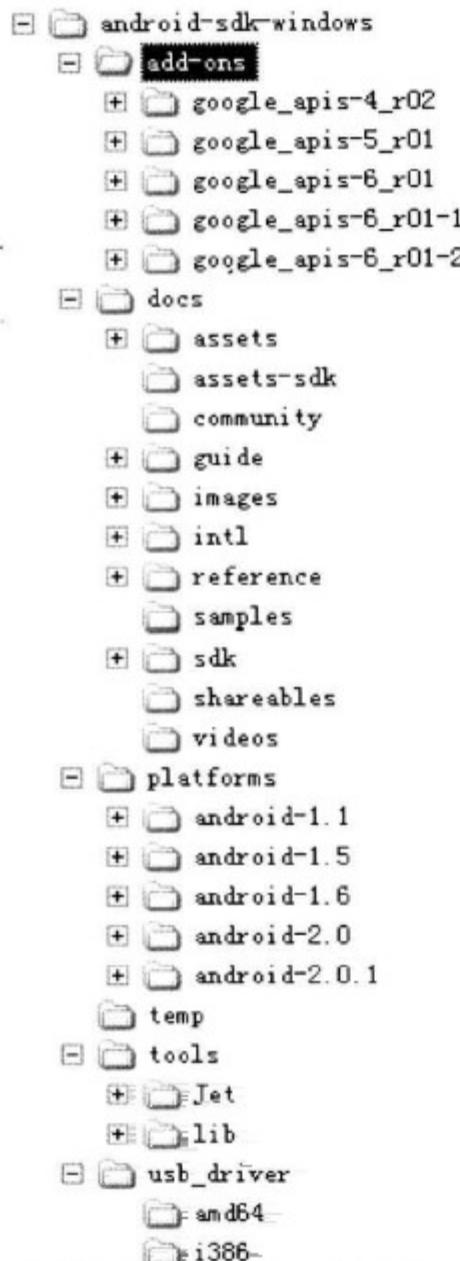


图 1-51 Android SDK 安装后的目录结构

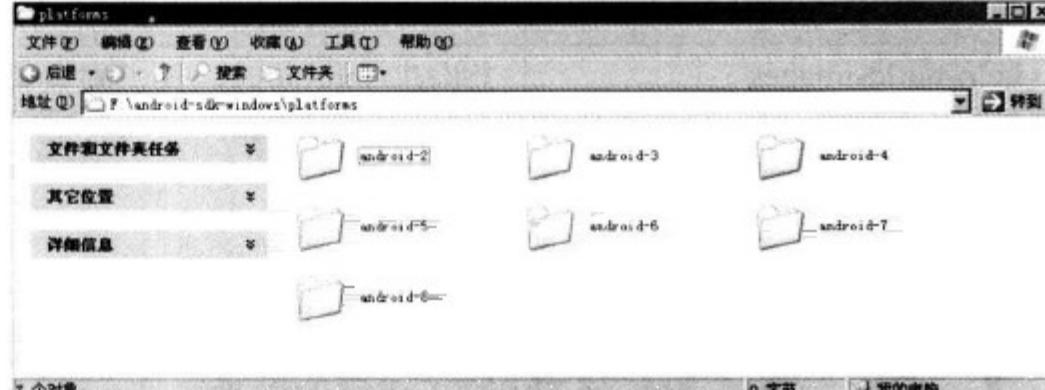


图 1-52 platforms 目录项

通过解压缩 android.jar 后，了解了其内部 API 的包结构和组织方式，如果要深入理解各个文件包内包含的 API 和 API 的具体用法，就必须学会阅读、查找 SDK 文档。可以使用浏览器打开“docs”目录下的文件 index.html，如图 1-53 所示。

在图 1-53 所示的主页中，介绍了 Android 基本概念和当前常用版本，在右侧和顶端导航中列出了一些常用的链接。此 SDK 文件对于初学者来说十分重要，可以帮助读者解决很多常见的问题，是一个很好的学习文档和帮助文档。

单击导航中的【Dev Guide】按钮后打开如图 1-54 所示的界面。

在图 1-54 所示页面中，左侧是目录索引链接，单击某个链接后，可以在右侧界面中显示对应的说明信息。如果要想迅速地理解一个问题或知识点，可以在搜索对话框中对 SDK 进行检索，搜索到自己需要的内容。当然，很多热心的程序员和学者对 SDK 进行了翻译，网络上有很多 SDK 中文版，感兴趣的读者可以从网络中获取。

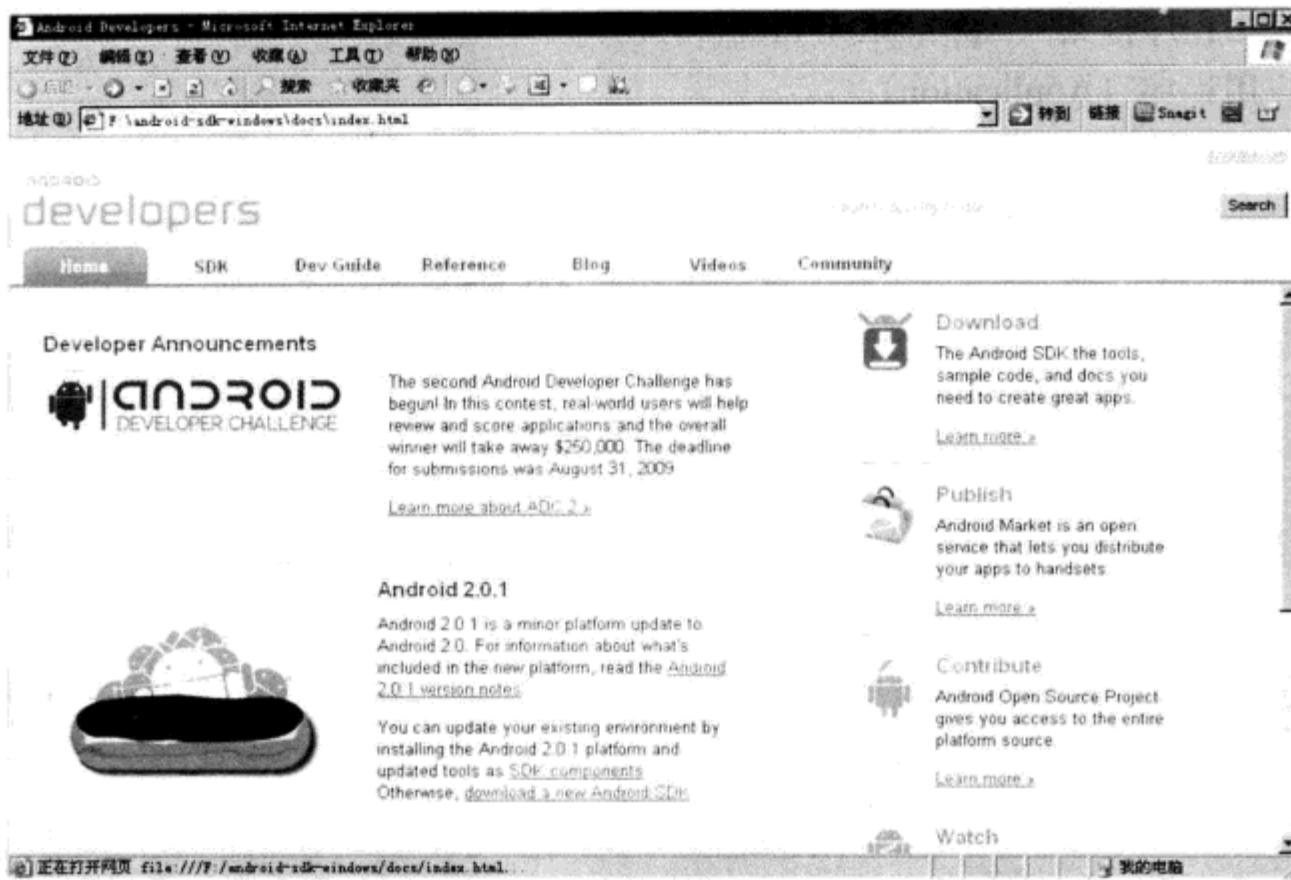


图1-53 SDK文档主页

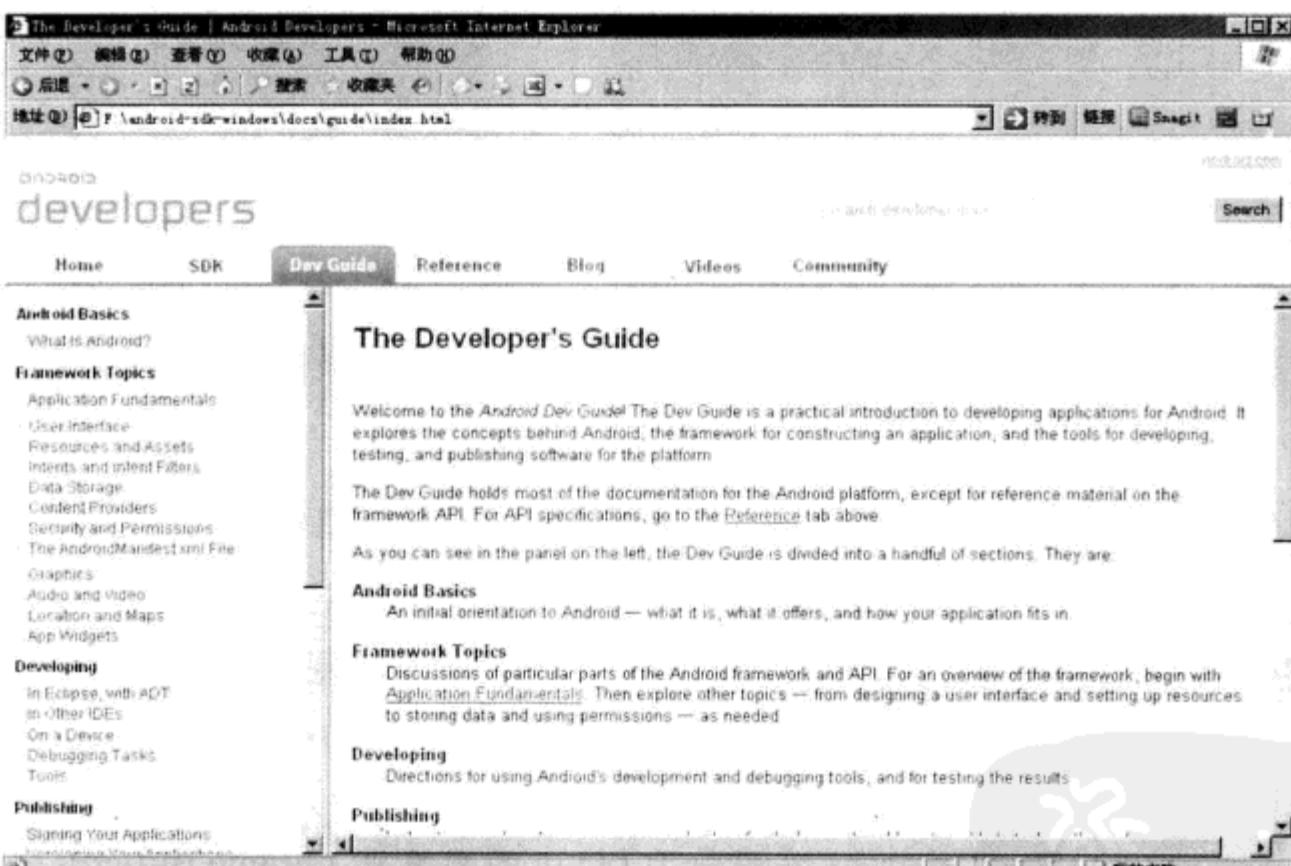


图1-54 SDK文档索引

1.5.2 Android体系结构介绍

Android作为一个移动设备的平台，其软件层次结构包括操作系统（OS）、中间件（MiddleWare）和应用程序（Application）。根据Android的软件框图，其软件层次结构自下而上分为以下4层。

- (1) 操作系统层（OS）。
- (2) 各种库（Libraries）和Android运行环境（RunTime）。

(3) 应用程序框架 (Application Framework)。

(4) 应用程序 (Application)。

上述各个层的具体结构如图 1-55 所示。

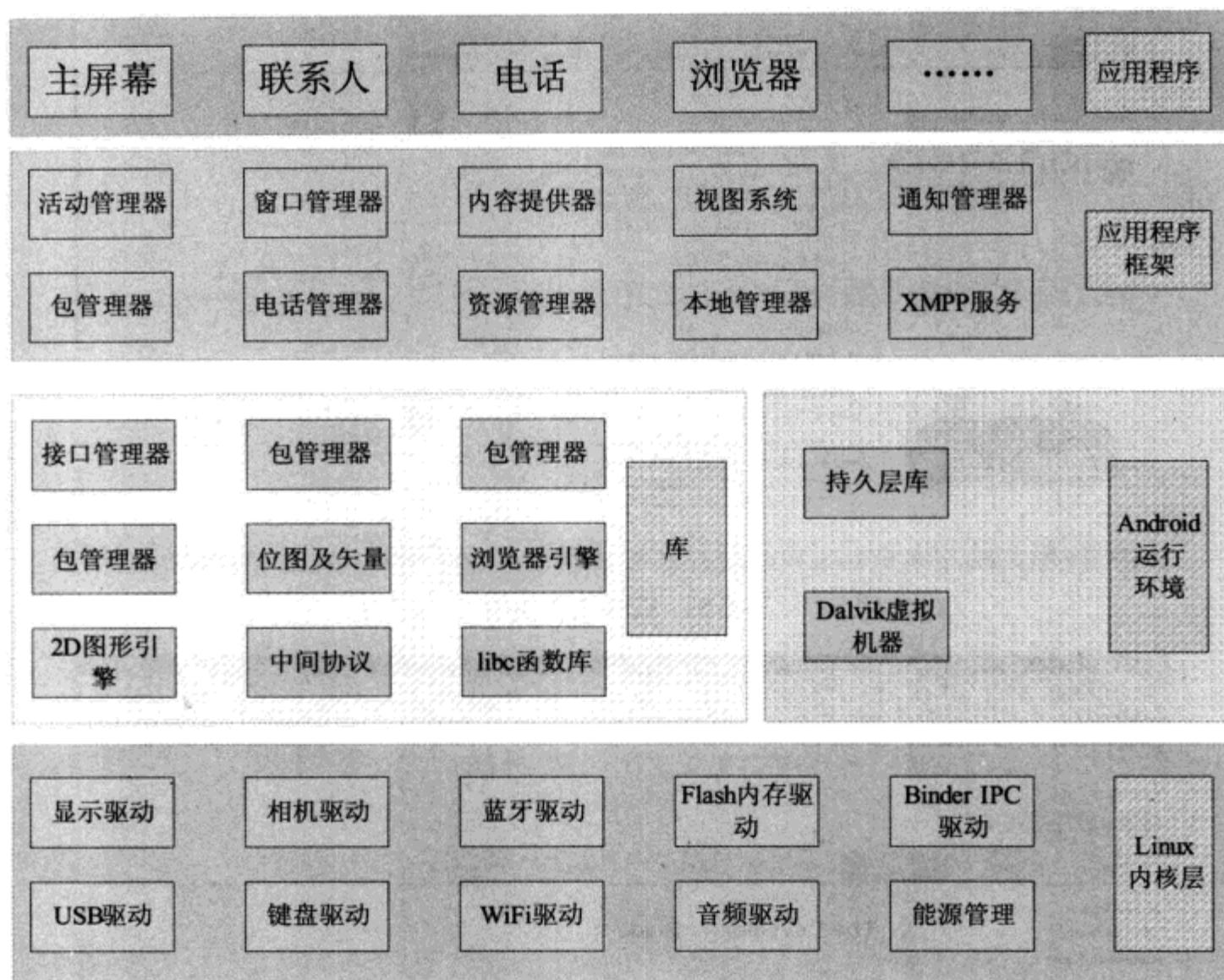
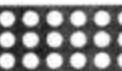


图 1-55 Android 操作系统的组件结构图

1. 操作系统层 (OS)

Android 使用 Linux 2.6 作为操作系统，Linux 2.6 是一种标准的技术，Linux 也是一个开放的操作系统。Android 对操作系统的使用包括核心和驱动程序两部分，Android 的 Linux 核心为标准的 Linux 2.6 内核，Android 更多的是需要一些与移动设备相关的驱动程序。主要的驱动如下所示：

- ◆ 显示驱动 (Display Driver)：常用基于 Linux 的帧缓冲 (Frame Buffer) 驱动；
- ◆ Flash 内存驱动 (Flash Memory Driver)：是基于 MTD 的 Flash 驱动程序；
- ◆ 照相机驱动 (Camera Driver)：常用基于 Linux 的 v4l (Video for) 驱动；
- ◆ 音频驱动 (Audio Driver)：常用基于 ALSA (Advanced Linux Sound Architecture, 高级 Linux 声音体系) 驱动；
- ◆ WiFi 驱动 (Camera Driver)：基于 IEEE 802.11 标准的驱动程序；
- ◆ 键盘驱动 (KeyBoard Driver)：作为输入设备的键盘驱动；
- ◆ 蓝牙驱动 (Bluetooth Driver)：基于 IEEE 802.15.1 标准的无线传输技术；
- ◆ Binder IPC 驱动：Android 一个特殊的驱动程序，具有单独的设备节点，提供进程



间通信的功能；

- ◆ Power Management（能源管理）：管理电池电量等信息。

2. 各种库（Libraries）和 Android 运行环境（RunTime）

本层次对应一般嵌入式系统，相当于中间件层次。Android 的本层次分成两个部分，一个是各种库，另一个是 Android 运行环境。其中包含的各种库如下所示。

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

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

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

3. 应用程序（Application）

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

4. 应用程序框架（Application Framework）

Android 的应用程序框架为应用程序层的开发者提供 APIs，它实际上是一个应用程序的框架。由于上层的应用程序是以 Java 构建的，因此本层次首先提供了包含了 UI 程序中所需要的各种控件，例如：Views（视图组件），其中又包括了 List（列表）、Grid（栅格）、



Text Box（文本框）、Button（按钮）等，甚至一个嵌入式的 Web 浏览器。

一个基本的 Android 应用程序可以利用应用程序框架中的以下五个部分。

- ◆ Activity（活动）；
- ◆ Broadcast Intent Receiver（广播意图接收者）；
- ◆ Service（服务）；
- ◆ Content Provider（内容提供者）；
- ◆ Intent and Intent Filter（意图和意图过滤器）。

1.5.3 Android 应用工程文件组成

Android 的应用工程文件主要由以下部分组成。

- ◆ src 文件：项目源文件都保存在这个目录里面。
- ◆ R.java 文件：这个文件是 Eclipse 自动生成的，应用开发者不需要去修改里边的内容。
- ◆ Android Library：这是应用运行的 Android 库。
- ◆ assets 目录：里面主要放置多媒体等一些文件。
- ◆ res 目录：里面主要放置应用会用到的资源文件。
- ◆ drawable 目录：主要放置应用会用到的图片资源。
- ◆ layout 目录：主要放置用到的布局文件。这些布局文件都是 XML 文件。
- ◆ values 目录：主要放置字符串（strings.xml）、颜色（colors.xml）、数组（arrays.xml）。
- ◆ AndroidManifest.xml：相当于应用的配置文件。在这个文件里边，必须声明应用的名称，应用所用到得 Activity、Service 以及 receiver 等。

在 Eclipse 中，一个基本的 Android 项目的目录结构如图 1-56 所示。

1. src 目录

与一般的 Java 项目一样，“src” 目录下保存的是项目的所有包及源文件（.java），“res” 目录下包含了项目中的所有资源。例如，程序图标（drawable）、布局文件（layout）和常量（values）等。不同的是，在 Java 项目中没有“gen” 目录，也没有每个 Android 项目都必须有的 AndroidManifest.xml 文件。

“.java” 格式文件是在建立项目时自动生成的，这个文件是只读模式，不能更改。R.java 文件是定义该项目所有资源的索引文件。先来看看 HelloAndroid 项目的 R.java 文件，例如下面的代码：

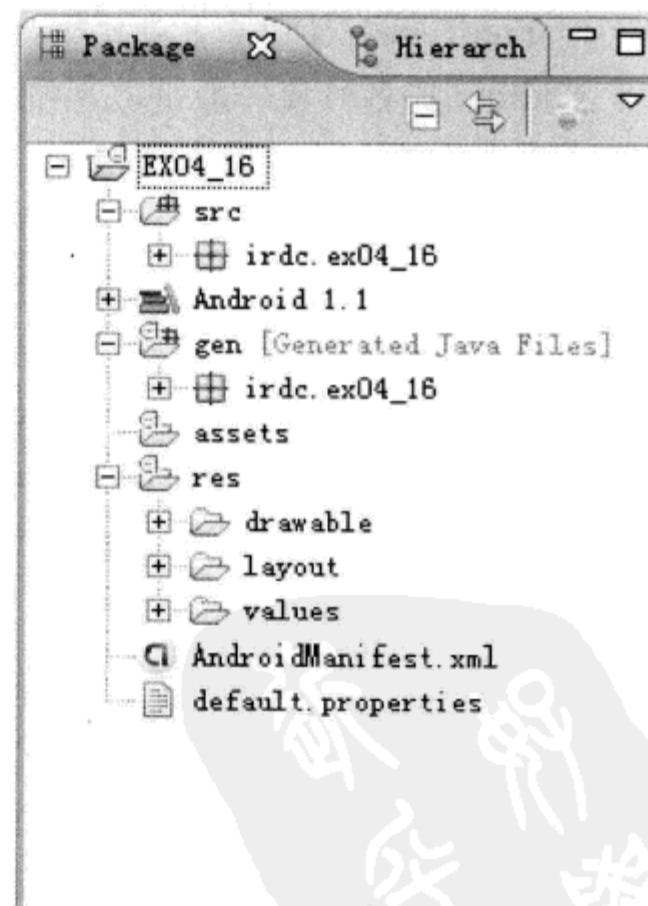


图 1-56 Android 应用工程文件组成

```
package com.yarin.Android.HelloAndroid;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

从上述代码中，可以看到定义了很多常量，并且会发现这些常量的名字都与 res 文件夹中的文件名相同，这再次证明 .java 文件中所存储的是该项目所有资源的索引。有了这个文件，在程序中使用资源将变得更加方便，可以很快地找到要使用的资源，由于这个文件不能被手动编辑，所以当用户在项目中加入了新的资源时，只需要刷新一下该项目，.java 文件便自动生成了所有资源的索引。

2. AndroidManifest.xml 文件

在文件 AndroidManifest.xml 中包含了该项目中所使用的 Activity、Service、Receiver，看下面“HelloAndroid”项目中的 AndroidManifest.xml 文件。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.yarin.Android.HelloAndroid"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="5" />
</manifest>
```



在上述代码中，intent-filters 描述了 Activity 启动的位置和时间。每当一个 Activity（或者操作系统）要执行一个操作时，它将创建出一个 Intent 的对象，这个 Intent 对象能承载的信息可描述用户想做什么，用户想处理什么数据，数据的类型，以及一些其他信息。而 Android 则会和每个 Application 所暴露的 intent-filter 的数据进行比较，找到最合适 Activity 来处理调用者所指定的数据和操作。下面来仔细分析 AndroidManifest.xml 文件，如表 1-3 所示。

表1-3 AndroidManifest.xml分析

参数	说明
manifest	根节点，描述了 package 中所有的内容
xmlns:android	包含命名空间的声明。 xmlns:android=http://schemas.android.com/apk/res/android，使得 Android 中各种标准属性能在文件中使用，提供了大部分元素中的数据
Package	声明应用程序包
application	包含 package 中 application 级别组件声明的根节点。此元素也可包含 application 的一些全局和默认的属性，如标签、icon、主题、必要的权限等。一个 manifest 能包含零个或一个此元素（不能多余一个）
android:icon	应用程序图标
android:label	应用程序名字
Activity	用来与用户交互的主要工具。Activity 是用户打开一个应用程序的初始页面，大部分被使用到的其他页面也由不同的 activity 所实现，并声明在另外的 activity 标记中。注意，每一个 activity 必须有一个<activity>标记对应，无论它给外部使用或是只用于自己的 package 中。如果一个 activity 没有对应的标记，用户将不能运行它。另外，为了支持运行时查找 Activity，可包含一个或多个<intent-filter>元素来描述 activity 所支持的操作
android:name	应用程序默认启动的 activity
intent-filter	声明了指定的一组组件支持的 Intent 值，从而形成了 Intent Filter。除了能在此元素下指定不同类型的值，属性也能放在这里来描述一个操作所需的唯一的标签、icon 和其他信息
action	组件支持的 Intent action
category	组件支持的 Intent Category。这里指定了应用程序默认启动的 activity
uses-sdk	该应用程序所使用的 sdk 版本相关

3. 常量的定义文件

下面我们看看资源文件中一些常量的定义，如 String.xml，例如下面的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloAndroid!</string>
    <string name="app_name">HelloAndroid</string>
</resources>
```



上述代码很简单，就定义了两个字符串资源。

接下来分析 HelloAndroid 项目的布局文件（layout），首先打开文件“res\layout\main.xml”，其代码如下所示。

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

在上述代码中，有以下几个布局和参数。

- ◆ < LinearLayout></LinearLayout>：线性版面配置，在这个标签中，所有元件都是按由上到下的排队排成的。
- ◆ android:orientation：表示这个介质的版面配置方式是从上到下垂直地排列其内部的视图。
- ◆ android:layout_width：定义当前视图在屏幕上所占的宽度，fill_parent 即填充整个屏幕。
- ◆ android:layout_height：定义当前视图在屏幕上所占的高度，fill_parent 即填充整个屏幕。
- ◆ wrap_content：随着文字栏位的不同而改变这个视图的宽度或高度。

在上述布局代码中，使用了一个 TextView 来配置文本标签 Widget（构件），其中设置的属性 android:layout_width 为整个屏幕的宽度，android:layout_height 可以根据文字来改变高度，而 android:text 则设置了这个 TextView 要显示的文字内容，这里引用了 @string 中的 hello 字符串，即 String.xml 文件中的 hello 所代表的字符串资源。hello 字符串的内容 "Hello World, HelloAndroid!" 这就是用户在 HelloAndroid 项目运行时看到的字符串。



注意 上面介绍的文件只是主要文件，在项目中需要用户自行编写。在项目中还有很多其他的文件，那些文件很少需要用户编写的，所以在此就不进行讲解了。

1.5.4 应用程序的生命周期

程序也如同自然界的生物一样，有自己的生命周期。应用程序的生命周期即程序的存活时间，即在啥时间内有效。Android 是一构建在 Linux 之上的开源移动开发平台，在



Android 中，多数情况下每个程序都是在各自独立的 Linux 进程中运行的。当一个程序或其某些部分被请求时，它的进程就“出生”了；当这个程序没有必要再运行下去且系统需要回收这个进程的内存用于其他程序时，这个进程就“死亡”了。可以看出，Android 程序的生命周期是由系统控制而非程序自身直接控制。这和我们编写桌面应用程序时的思维有一些不同，一个桌面应用程序的进程也是在其他进程或用户请求时被创建，但是往往是在程序自身收到关闭请求后执行一个特定的动作（比如从 main 函数中返回）而导致进程结束的。要想做好某种类型的程序或者某种平台下的程序的开发，最关键的就是要弄清楚这种类型的程序或整个平台下的程序的一般工作模式并熟记在心。在 Android 中，程序的生命周期控制就是属于这个范畴。

开发者必须理解不同的应用程序组件，尤其是 Activity、Service 和 Intent Receiver。了解这些组件是如何影响应用程序的生命周期的，这非常重要。如果不正确地使用这些组件，可能会导致系统终止正在执行重要任务的应用程序进程。

一个常见的进程生命周期漏洞的例子是 Intent Receiver（意图接收器），当 Intent Receiver 在 onReceive 方法中接收到一个 Intent（意图）时，它会启动一个线程，然后返回。一旦返回，系统将认为 Intent Receiver 不再处于活动状态；因而 Intent Receiver 所在的进程也就不再有用了（除非该进程中还有其他的组件处于活动状态）。因此，系统可能会在任意时刻终止该进程以回收占用的内存。这样进程中创建出的那个线程也将被终止。解决这个问题的方法是从 Intent Receiver 中启动一个服务，让系统知道进程中还有处于活动状态的工作。为了使系统能够正确决定在内存不足时应该终止哪个进程，Android 根据每个进程中运行的组件及组件的状态把进程放入一个“Importance Hierarchy（重要性分级）”中。进程的类型按重要程度排序包括。

1. 前台进程 (Foreground)

与用户当前正在做的事情密切相关。不同的应用程序组件能够通过不同的方法将它的宿主进程移到前台。在如下的任何一个条件下：进程正在屏幕的最前端运行一个与用户交互的活动 (Activity)，它的 onResume 方法被调用；或进程有一正在运行的 Intent Receiver(它的 IntentReceiver.onReceive 方法正在执行)；或进程有一个服务 (Service)，并且在服务的某个回调函数 (Service.onCreate、Service.onStart 或 Service.onDestroy) 内有正在执行的代码，系统将把进程移动到前台。

2. 可见进程 (Visible)

它有一个可以被用户从屏幕上看到的活动，但不在前台（它的 onPause 方法被调用）。例如，如果前台的活动是一个对话框，以前的活动就隐藏在对话框之后，就会出现这种进程。可见进程非常重要，一般不允许被终止，除非是为了保证前台进程的运行而不得不终止它。

3. 服务进程 (Service)

拥有一个已经用 startService 方法启动的服务。虽然用户无法直接看到这些进程，但它们做的事情却是用户所关心的（如后台 MP3 回放或后台网络数据的上传、下载）。因此，系统将一直运行这些进程，除非内存不足以维持所有的前台进程和可见进程。



4. 后台进程 (Background)

拥有一个当前用户看不到的活动（它的onStop方法被调用）。这些进程对用户体验没有直接的影响。如果它们正确执行了活动生命周期，系统可以在任意时刻终止该进程以回收内存，并提供给前面三种类型的进程使用。系统中通常有很多这样的进程在运行，因此，要将这些进程保存在LRU列表中，以确保当内存不足时用户最近看到的进程最后一个被终止。

5. 空进程 (Empty)

不拥有任何活动的应用程序组件的进程。保留这种进程的唯一原因是在下次应用程序的某个组件需要运行时，不需要重新创建进程，这样可以提高启动速度。

系统将以进程中当前处于活动状态组件的重要程度为基础对进程进行分类。进程的优先级可能也会根据该进程与其他进程的依赖关系而增长。例如，如果进程 A 通过在进程 B 中设置 Context.BIND_AUTO_CREATE 标记或使用 ContentProvider 被绑定到一个服务（Service），那么进程 B 在分类时至少要被看成与进程 A 同等重要。

例如，Activity 的状态转换图如图 1-57 所示。

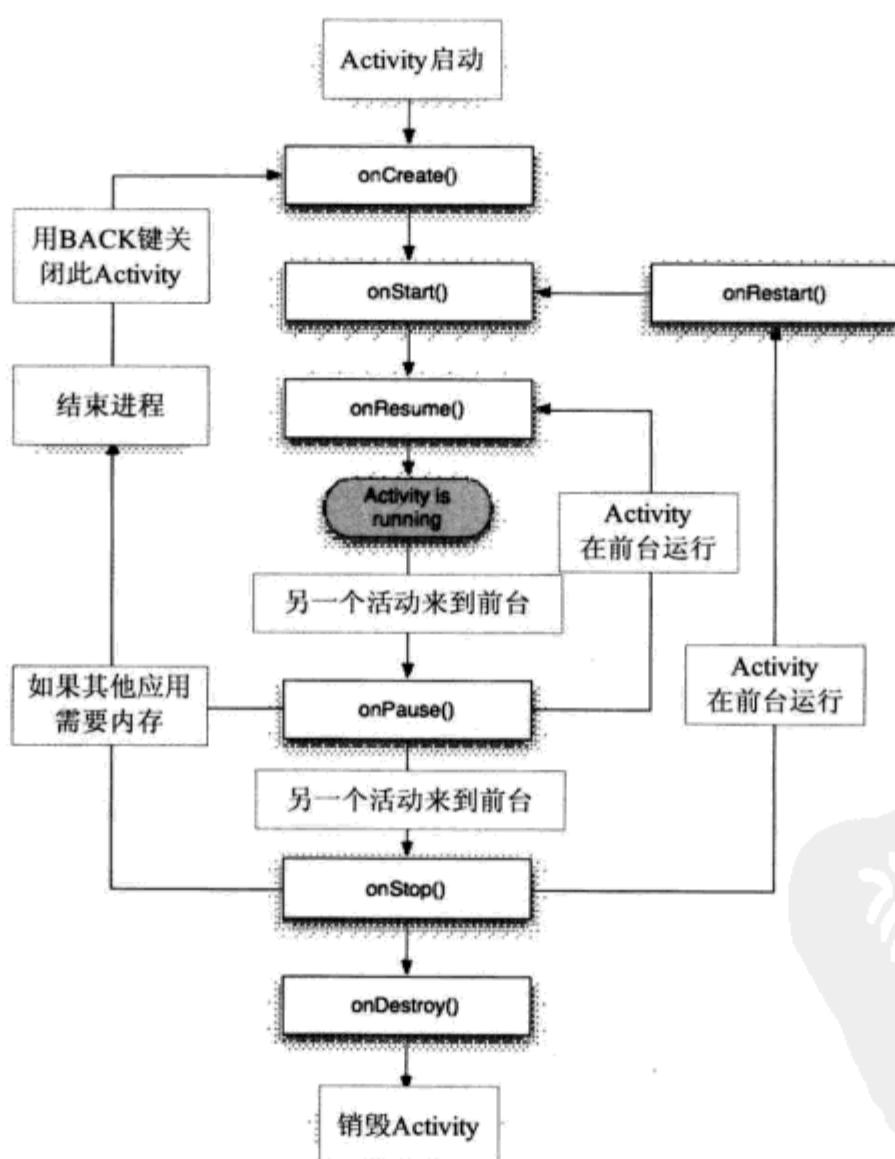


图1-57 Activity状态转换图

图1-57所示的状态的变化是由Android内存管理器决定的，Android会首先关闭那些包含Inactive Activity的应用程序，然后关闭Stopped状态的程序。在极端情况下，会移除Paused状态的程序。



第 2 章

界面布局实战演练

对于网站开发人员来说，网站结构和界面设计是影响浏览用户第一视觉的关键。而对于 Android 应用开发来说，除了功能强大的应用程序外，屏幕界面效果也是影响程序质量的重要元素。因为消费者永远喜欢的是既界面美观，而又功能强大的软件产品。在设计优美界面之前，一定要先对屏幕进行布局。在本章的内容中，将以具体实例来介绍布局 Android 屏幕的知识，为本书后面知识的学习打下基础。



2.1 使用线性布局（LinearLayout）

一个 Android 程序是由一个或多个 Activity 组成的，Activity 是一个 UI 容器，它本身不在用户界面中显示出来。其中类 View 起了一个重要的作用，View 是一个最基本的 UI 类，几乎所有的 UI 组件都是继承于 View 而实现的。

线性布局即 LinearLayout 布局，是 Android 屏幕中常用的布局方式，是一个 ViewGroup 以线性方向显示它的子视图（view）元素，即垂直或水平。

本实例的源码保存在【光盘\daima\第 2 章\xian】，实现流程如下所示。

- (1) 使用 Eclipse 创建一个名为“xian”的 Android 工程。
- (2) 编写布局文件“res/layout/main.xml”，代码如下所示。

```
<?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:orientation="horizontal"><!-- have an eye on !
-->
    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Button1"
        android:layout_weight="1"
        />
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="H Button2"
        android:layout_weight="1"
        />
    <Button android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Button3"
        android:layout_weight="1"
        />
    <Button android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Button4"
        android:layout_weight="1"
        />
```

```

    />
<Button android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button5"
        android:layout_weight="1"
    />
</LinearLayout>

```

在上述代码中，在根 LinearLayout 视图组（ViewGroup）中包含了 5 个 Button，它的子元素是以线性方式（horizontal，水平的）布局，运行效果如图 2-1 所示。



图2-1 线型布局

2.2 使用相对布局（RelativeLayout）

相对布局是指一个 ViewGroup 以相对位置显示它的子视图（View）元素，一个视图可以指定相对于它的兄弟视图的位置（例如在给定视图的左边或者下面）或相对于 RelativeLayout 的特定区域的位置。例如底部对齐，或中间偏左。

相对布局是设计用户界面的有力工具，因为它消除了嵌套视图组。如果用户发现使用了多个嵌套的 LinearLayout 视图组，可以考虑使用一个 RelativeLayout 视图组了。

本实例的源码保存在【光盘\daima\第 2 章\xiang】，实现流程如下所示。

- (1) 使用 Eclipse 创建一个名为“xiang”的 Android 工程。
- (2) 编写布局文件“res/layout/main.xml”，代码如下所示。

```

<?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">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"

```



```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_background"
    android:layout_below="@+id/label"/>
<Button
    android:id="@+id/ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/entry"
    android:layout_alignParentRight="true"
    android:layout_marginLeft="10dip"
    android:text="确定" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toLeftOf="@+id/ok"
    android:layout_alignTop="@+id/ok"
    android:text="取消" />
</RelativeLayout>
```

执行之后的效果如图 2-2 所示。



图2-2 执行效果

2.3 使用表格布局（TableLayout）

表格布局（TableLayout）是一个 ViewGroup 以表格显示它的子视图（View）元素，即行和列标识一个视图的位置。其实 Android 的表格布局跟 HTML 中的表格布局非常类似，TableRow 就像 HTML 表格的 <tr> 标记。表格布局通常用于把子元素放入到行与列中，不显示行、列或是单元格边界线，但是单元格不能横跨行，如 HTML 中一样。效果如图 2-3 所示。

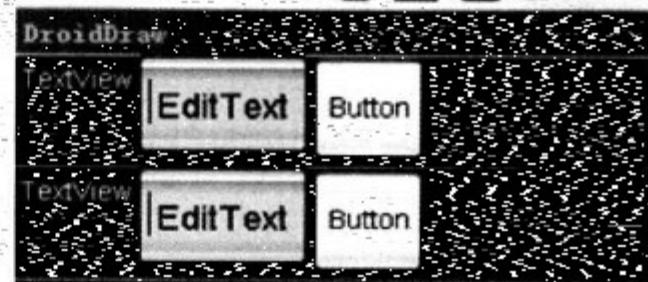


图2-3 表格布局

本实例的源码保存在【光盘\daima\第2章\biaoge】，实现流程如下所示。

step 01 使用 Eclipse 创建一个名为“biaoge”的 Android 工程。

step 02 编写布局文件“res/layout/main.xml”，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:shrinkColumns="0,1,2"><!-- have an eye on it -->
<TableRow><!-- row1 -->
    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:layout_column="0"
    />
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:layout_column="1"
    />
</TableRow>
<TableRow><!-- row2 -->
    <Button android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button3"
        android:layout_column="1"
    />
    <Button android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button4"
        android:layout_column="1"
    />
</TableRow>
<TableRow>
    <Button android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button5"
        android:layout_column="2"
    />
```



```
</TableRow>
</TableLayout>
```

执行后的效果如图 2-4 所示。

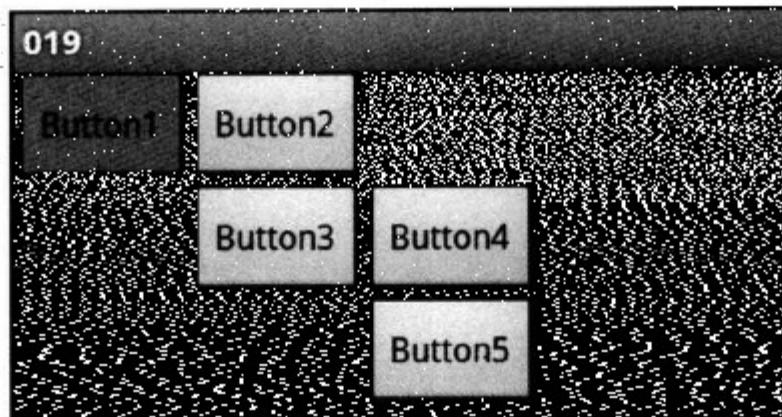


图2-4 表格布局

2.4 使用绝对布局（AbsoluteLayout）

绝对布局（AbsoluteLayout）是一个 ViewGroup 以绝对方式显示它的子视图（View）元素，即以坐标的方式来定位在屏幕上位置。这种布局方式很好理解，在布局文件或编程地设置 View 的坐标，从而绝对地定位。AbsoluteLayout 可以让子元素指定准确的 x/y 坐标值，并显示在屏幕上。(0,0) 为左上角，当向下或向右移动时，坐标值将变大。AbsoluteLayout 没有页边框，允许元素之间互相重叠（尽管不推荐）。我们通常不推荐使用 AbsoluteLayout，除非你有正当理由要使用它，因为它使界面代码太过刚性，以致在不同的设备上可能不能很好地工作，例如如图 2-5 所示的效果。

本实例的源码保存在【光盘\daima\第 2 章\juedui】，实现流程如下所示。

step 01 使用 Eclipse 创建一个名为“juedui”的 Android 工程。

step 02 编写布局文件“res/layout/main.xml”，代码如下所示。

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/AbsoluteLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView android:id="@+id/txtIntro"
        android:text="演示绝对布局"
        android:layout_width="fill_parent"
```



图2-5 AbsoluteLayout效果



```

        android:layout_height="wrap_content"
        android:layout_x="20dip"
        android:layout_y="20dip">
    </TextView>
</AbsoluteLayout>

```

执行后的效果如图 2-6 所示。

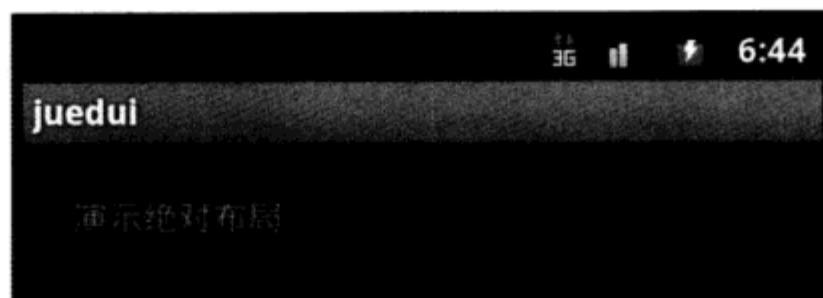


图2-6 执行效果

2.5 使用标签布局（TabLayout）

标签布局（TabLayout）是一个 ViewGroup 以标签的方式显示它的子视图（View）元素，就像在 Firefox 中的一个窗口中显示多个网页一样。为了创建一个标签 UI（tabbed UI），需要使用到 TabHost 和 TabWidget。TabHost 必须是布局的根节点，它包含为了显示标签的 TabWidget 和显示标签内容的 FrameLayout。

本实例的源码保存在【光盘\daima\第 2 章\biaoqian】，实现流程如下所示。

step 01 使用 Eclipse 创建一个名为“biaoqian”的 Android 工程。

step 02 编写布局文件“res/layout/main.xml”，代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
    <TableRow>
        <Button android:layout_width="wrap_content" android:layout_
            height="wrap_content"
            android:text=" 7 " />
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 8 "
            />
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 9 "
            />
        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" /      "
            />
    </TableRow>
</TableLayout>

```

```
    />

  </TableRow>
  <TableRow>
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 4 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 5 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 6 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" * "
            />

  </TableRow>
  <TableRow>
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 1 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 2 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 3 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" . "
            />

  </TableRow>
  <TableRow>
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" 0 "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" = "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" - "
            />
    <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text=" + "
            />
```



```
    />  
  
</TableRow>  
</TableLayout>
```

执行后将显示一个计算器的效果，如图 2-7 所示。



图2-7 计算器效果

2.6 使用层布局（FrameLayout）

层布局（FrameLayout）是最简单的一个布局对象，它被定制为用户屏幕上的一个空白备用区域，之后用户可以在其中填充一个单一对象，例如一张要发布的图片。所有的子元素将会固定在屏幕的左上角；我们不能为 FrameLayout 中的一个子元素指定一个位置。后一个子元素将会直接在前一个子元素之上进行覆盖填充，把它们部份或全部挡住（除非后一个子元素是透明的）。

本实例的源码保存在【光盘\daima\第 2 章\ceng】，实现流程如下所示。

step 01 使用 Eclipse 创建一个名为“ceng”的 Android 工程。

step 02 编写布局文件“res/layout/main.xml”，代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>  
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/  
    android"  
        android:layout_width="fill_parent"  
        android:layout_height="fill_parent">  
    <TextView  
        android:text="较大"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:textSize="50pt"/>  
    <TextView  
        android:text="一般"  
        android:layout_width="wrap_content"
```



```
    android:layout_height="wrap_content"
    android:textSize="20pt"/>
<TextView
    android:text="较小"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="10pt"/>
</FrameLayout>
```

执行后因为多层重叠而发生重影效果，如图 2-8 所示。

从图 2-8 所示效果可以看出，FrameLayout 能够将组件显示在屏幕的左上角，后面的组件覆盖前面的组件。该布局 Container 可以用来占有屏幕的某块区域来显示单一的对象，可以包

含有多个 Widgets 或者是 Container，但是所有被包含的 Widgets 或者是 Container 必须被固定到屏幕的左上角，并且一层覆盖一层，而不能通过为一个 Widgets 或者是 Container 指定一个位置。在 Container 中所包含的 Widgets 或者是 Container 的队列是采用的堆栈的结构，最后加进来的 Widgets 或者是 Container 显示在最上面。所以后一个 Widgets 或者是 Container 将会直接覆盖在前一个 Widgets 或者是 Container 之上，把它们部份或全部挡住。除非后一个 Widgets/Container 是透明的，否则必须得到 FrameLayout Container 的允许。



图2-8 计时器效果

2.7 使用桌面组件Widget来布局

Widget 组件是一个桌面组件，从名字就可以知道是实现桌面布局的。Widget 的这个“叫法”来自一个叫做 Rose 的苹果电脑工程师。在 1998 年的一天，Rose 在自己的苹果操作系统桌面玩一个可以更换皮肤的 MP3 播放器时忽发奇想：如果在我桌面上运行的所有工具都能够更换皮肤或外观，那将是一件很酷的事情，Rose 的兴奋之情溢于言表，它给这个酷酷的玩意儿起了个名字叫“Konfabulator”。

Android 本身已经自带了时钟、音乐播放器、相框和 Google 搜索 4 个 Widget 程序，不过这并不能阻止大家开发自己更加美观，功能更丰富的版本。另外，微博客、RSS 订阅、股市信息、天气预报这些 Widget 也都有流行的可能。

本实例的源码保存在【光盘\daima\第 2 章\WidgetZ】，实现流程如下所示。

step 01 使用 Eclipse 创建一个 MainActivity 作为应用程序的入口，自动生成的主文件是 MainActivity.java，其主要代码如下所示。

```
package m.usewidget;
import m.usewidget.R;
import android.app.Activity;
```

```

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}

```

在上述代码中，关联了一个模板布局文件 main.xml。这样，就可以在里面继续添加需要的控件了，例如按钮、列表框、进度条和图片等。

step 02 编写布局文件 main.xml，主要代码如下所示。

```

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

```

通过上述代码，在手机屏幕中使用了 Widget 组件。执行后不会显示任何信息，这是因为我们没有在里面添加任何元素。由此可见，Widget 组件只是起了一个“容器”的作用，我们只需要把需要显示的屏幕元素添加到这个“容器”里面即可。

在本节接下来的实例代码中，都保存在本实例项目中，即本节剩余实例的源码都保存在【光盘\daima\WidgetZ】目录中。

2.7.1 在屏幕中实现一个按钮效果

Button 控件是一个按钮控件。在项目中应用时，当单击 Button 后会触发一个事件，这个事件会实现用户需要的功能。例如用户输入一些信息，单击【确定】或【取消】按钮后会实现对应的功能。

step 01 修改布局文件 main.xml，在里面添加一个 TextView 和一个 Button。主要代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    >
</LinearLayout>

```

```
    android:layout_height="fill_parent"
    >
<TextView
    android:id="@+id/show_TextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button
    android:id="@+id/Click_Button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="点击"
    />
</LinearLayout>
```

step 02 在文件 `MainActivity.java` 中，先通过 `findViewById()` 获取 `TextView` 文本和 `Button` 按钮的资源，然后为 `Button` 按钮添加事件监听器 `Button.OnClickListener()`，最后定义处理事件处理程序。主要代码如下所示。

```
//获取TextView文本和Button按钮的资源
show= (TextView) findViewById(R.id.show_TextView);
press=(Button) findViewById(R.id.Click_Button);
//为Button按钮添加事件监听器 Button.OnClickListener()
press.setOnClickListener(new Button.OnClickListener(){
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
});
//定义处理事件处理程序
press.setOnClickListener(new Button.OnClickListener(){
    @Override
    public void onClick(View v) {
        // 点击按钮后输出一段文本
        show.setText("哎呦，button被点了一下");
    }
});
```

运行后首先显示一个“按钮 + 文本”样式的界面，当单击按钮【这是 button】后会执行单击事件，执行定义的事件处理程序，如图 2-9 所示。



图2-9 执行效果

2.7.2 在屏幕中显示一段文字

在手机屏幕中可以通过文本框控件 TextView 来显示文本。在使用 TextView 控件时通常需要遵循如下 6 个步骤。

step 01 导入 TextView 包，具体代码如下所示。

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

step 02 在文件 mainActivity.java 中声明一个 TextView，例如下面的代码。

```
private TextView mTextView01;
```

step 03 在文件 main.xml 中定义一个 TextView 对象 TextView01，例如下面的代码。

```
<TextView android:text="TextView01"
          android:id="@+id/TextView01"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_x="61px"
          android:layout_y="69px">
</TextView>
```

step 04 利用 findViewById() 方法获取 main.xml 中的 TextView，例如下面的代码。

```
mTextView01 = (TextView) findViewById(R.id.TextView01);
```

step 05 设置 TextView 标签内容，例如下面的代码。

```
String str_2 = "欢迎来到Android世界...";
mTextView01.setText(str_2);
```

step 06 设置文本超级链接，例如下面的代码。

```
<TextView
          android:id="@+id/TextView02"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:autoLink="all"
          android:text="请访问Android 开发者:
          http://developer.android.com/index.html">
</TextView>
```

本实例的实现过程比较简单，只需修改文件 mainActivity.java 即可，即在里面分别添加 12 个 TextView 对象变量，1 个 LinearLayout 对象变量、1 个 WC 整数变量、1 个 LinearLayout.LayoutParams 变量。主要代码如下所示。

```
/* 定义使用的对象 */
private LinearLayout myLayout;
```



```
private LinearLayout.LayoutParams layoutP;
private int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
private TextView black_TV, blue_TV, cyan_TV, dkgray_TV,
    gray_TV, green_TV, ltgray_TV, magenta_TV, red_TV,
    transparent_TV, white_TV, yellow_TV;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /* 实例化一个LinearLayout布局对象 */
    myLayout = new LinearLayout(this);
    /* 设置LinearLayout的布局为垂直布局 */
    myLayout.setOrientation(LinearLayout.VERTICAL);
    /* 设置LinearLayout布局背景图片 */
    myLayout.setBackgroundResource(R.drawable.back);
    /* 加载主屏布局 */
    setContentView(myLayout);
    /* 实例化一个LinearLayout布局参数，用来添加View */
    layoutP = new LinearLayout.LayoutParams(WC, WC);
    /* 构造实例化TextView对象 */
    constructTextView();
    /* 把TextView添加到LinearLayout布局中 */
    addTextView();
    /* 设置TextView文本颜色 */
    setTextViewColor();
    /* 设置TextView文本内容 */
    setTextViewText();
}

/* 设置TextView文本内容 */
public void setTextViewText() {
    black_TV.setText("黑色");
    blue_TV.setText("蓝色");
    cyan_TV.setText("青绿色");
    dkgray_TV.setText("灰黑色");
    gray_TV.setText("灰色");
    green_TV.setText("绿色");
    ltgray_TV.setText("浅灰色");
    magenta_TV.setText("红紫色");
    red_TV.setText("红色");
    transparent_TV.setText("透明");
    white_TV.setText("白色");
    yellow_TV.setText("黄色");
}

/* 设置TextView文本颜色 */
public void setTextViewColor() {
    black_TV.setTextColor(Color.BLACK);
```

```
    blue_TV.setTextColor(Color.BLUE);
    dkgray_TV.setTextColor(Color.DKGRAY);
    gray_TV.setTextColor(Color.GRAY);
    green_TV.setTextColor(Color.GREEN);
    ltgray_TV.setTextColor(Color.LTGRAY);
    magenta_TV.setTextColor(Color.MAGENTA);
    red_TV.setTextColor(Color.RED);
    transparent_TV.setTextColor(Color.TRANSPARENT);
    white_TV.setTextColor(Color.WHITE);
    yellow_TV.setTextColor(Color.YELLOW);
}
/* 构造实例化TextView对象 */
public void constructTextView() {
    black_TV = new TextView(this);
    blue_TV = new TextView(this);
    cyan_TV = new TextView(this);
    dkgray_TV = new TextView(this);
    gray_TV = new TextView(this);
    green_TV = new TextView(this);
    ltgray_TV = new TextView(this);
    magenta_TV = new TextView(this);
    red_TV = new TextView(this);
    transparent_TV = new TextView(this);
    white_TV = new TextView(this);
    yellow_TV = new TextView(this);
}
/* 把TextView添加到LinearLayout布局中 */
public void addTextView() {
    myLayout.addView(black_TV, layoutP);
    myLayout.addView(blue_TV, layoutP);
    myLayout.addView(cyan_TV, layoutP);
    myLayout.addView(dkgray_TV, layoutP);
    myLayout.addView(gray_TV, layoutP);
    myLayout.addView(green_TV, layoutP);
    myLayout.addView(ltgray_TV, layoutP);
    myLayout.addView(magenta_TV, layoutP);
    myLayout.addView(red_TV, layoutP);
    myLayout.addView(transparent_TV, layoutP);
    myLayout.addView(white_TV, layoutP);
    myLayout.addView(yellow_TV, layoutP);
}
}
```

执行后效果如图 2-10 所示。

由上述实例的实现过程可知，我们可以设置 TextView 控件的文本颜色。此功能是通



过 Color 属性实现的，各个属性值的具体说明如下所示。

- ◆ Color.BLACK：黑色
- ◆ Color.BLUE：蓝色
- ◆ Color.CYAN：青绿色
- ◆ Color.DKGRAY：灰黑色
- ◆ Color.GRAY：灰色
- ◆ Color.GREEN：绿色
- ◆ Color.LTGRAY：浅灰色
- ◆ Color.MAGENTA：红紫色
- ◆ Color.RED：红色
- ◆ Color.TRANSPARENT：透明
- ◆ Color.WHITE：白色
- ◆ Color.YELLOW：黄色

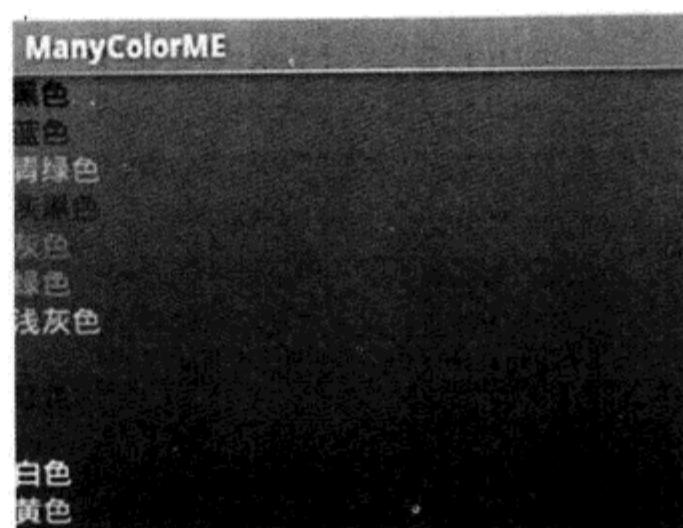


图2-10 执行效果

2.7.3 设置手机屏幕中的字体

在 Android 手机系统中，可以使用使用 TextView 来设置屏幕中静态域的字体。在计算机系统中，使用 Typeface 来表示字体的风格，具体来说有如下两种类型。

(1) Style 类型，具体说明如表 2-1 所示。

表2-1 int Style类型说明

字体	说明
BOLD	粗体
BOLD_ITALIC	粗斜体
ITALIC	斜体
NORMAL	普通字体

(2) Typeface 类型，具体说明如表 2-2 所示。

表2-2 Typeface类型说明

字体	说明
DEFAULT	默认字体
DEFAULT_BOLD	默认粗体
MONOSPACE	单间隔字体
SANS_SERIF	无衬线字体
SERIF	衬线字体

本实例的实现过程比较简单，通过修改文件 MainActivity.java 即可以实现多种字体样式的显示，主要代码如下所示。

```
public class TypefaceStudy extends Activity {  
    /*  
     * (non-Javadoc)  
     * @see android.app.Activity#onCreate(android.os.Bundle)  
     */  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

```
* android.graphics.Typeface java.lang.Object
    Typeface类指定一个字体的字体和固有风格。
* 该类用于绘制，与可选绘制设置一起使用，
    例如textSize, textSkewX, textScaleX 当绘制(测量)时来指定如何显示文本。
*/
/* 定义实例化一个 布局大小，用来添加TextView */
final int WRAP_CONTENT = ViewGroup.LayoutParams.WRAP_CONTENT;
/* 定义TextView对象 */
private TextView bold_TV, bold_italic_TV, default_TV,
                default_bold_TV, italic_TV, monospace_TV,
                normal_TV, sans_serif_TV, serif_TV;
/* 定义LinearLayout布局对象 */
private LinearLayout linearLayout;
/* 定义LinearLayout布局参数对象 */
private LinearLayout.LayoutParams linearLayouttParams;
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    /* 定义实例化一个LinearLayout对象 */
    linearLayout = new LinearLayout(this);
    /* 设置LinearLayout布局为垂直布局 */
    linearLayout.setOrientation(LinearLayout.VERTICAL);
    /* 设置布局背景图 */
    linearLayout.setBackgroundResource(R.drawable.back);
    /* 加载LinearLayout为主屏布局，显示 */
    setContentView(linearLayout);
    /* 定义实例化一个LinearLayout布局参数 */
    linearLayouttParams =
        new LinearLayout.LayoutParams(WRAP_CONTENT,WRAP_CONTENT);
    constructTextView();
    setTextSizeOf();
    setTextViewText();
    setStyleOfFont();
    setFontColor();
    toAddTextViewToLayout();
}
public void constructTextView() {
    /* 实例化TextView对象 */
    bold_TV = new TextView(this);
    bold_italic_TV = new TextView(this);
    default_TV = new TextView(this);
    default_bold_TV = new TextView(this);
    italic_TV = new TextView(this);
    monospace_TV=new TextView(this);
    normal_TV=new TextView(this);
```

```
sans_serif_TV=new TextView(this);
serif_TV=new TextView(this);
}
public void setTextSizeOf() {
    // 设置绘制的文本大小，该值必须大于0
    bold_TV.setTextSize(24.0f);
    bold_italic_TV.setTextSize(24.0f);
    default_TV.setTextSize(24.0f);
    default_bold_TV.setTextSize(24.0f);
    italic_TV.setTextSize(24.0f);
    monospace_TV.setTextSize(24.0f);
    normal_TV.setTextSize(24.0f);
    sans_serif_TV.setTextSize(24.0f);
    serif_TV.setTextSize(24.0f);
}
public void setTextViewText() {
    /* 设置文本 */
    bold_TV.setText("BOLD");
    bold_italic_TV.setText("BOLD_ITALIC");
    default_TV.setText("DEFAULT");
    default_bold_TV.setText("DEFAULT_BOLD");
    italic_TV.setText("ITALIC");
    monospace_TV.setText("MONOSPACE");
    normal_TV.setText("NORMAL");
    sans_serif_TV.setText("SANS_SERIF");
    serif_TV.setText("SERIF");
}
public void setStyleOfFont() {
    /* 设置字体风格 */
    bold_TV.setTypeface(null, Typeface.BOLD);
    bold_italic_TV.setTypeface(null, Typeface.BOLD_ITALIC);
    default_TV.setTypeface(Typeface.DEFAULT);
    default_bold_TV.setTypeface(Typeface.DEFAULT_BOLD);
    italic_TV.setTypeface(null, Typeface.ITALIC);
    monospace_TV.setTypeface(Typeface.MONOSPACE);
    normal_TV.setTypeface(null, Typeface.NORMAL);
    sans_serif_TV.setTypeface(Typeface.SANS_SERIF);
    serif_TV.setTypeface(Typeface.SERIF);
}
public void setFontColor() {
    /* 设置文本颜色 */
    bold_TV.setTextColor(Color.BLACK);
    bold_italic_TV.setTextColor(Color.CYAN);
    default_TV.setTextColor(Color.GREEN);
    default_bold_TV.setTextColor(Color.MAGENTA);
```

```

italic_TV.setTextColor(Color.RED);
monospace_TV.setTextColor(Color.WHITE);
normal_TV.setTextColor(Color.YELLOW);
sans_serif_TV.setTextColor(Color.GRAY);
serif_TV.setTextColor(Color.LTGRAY);
}

public void toAddTextViewToLayout() {
    /* 把TextView加入LinearLayout布局中 */
    linearLayout.addView(bold_TV, linearLayoutLayoutParams);
    linearLayout.addView(bold_italic_TV, linearLayoutLayoutParams);
    linearLayout.addView(default_TV, linearLayoutLayoutParams);
    linearLayout.addView(default_bold_TV, linearLayoutLayoutParams);
    linearLayout.addView(italic_TV, linearLayoutLayoutParams);
    linearLayout.addView(monospace_TV, linearLayoutLayoutParams);
    linearLayout.addView(normal_TV, linearLayoutLayoutParams);
    linearLayout.addView(sans_serif_TV, linearLayoutLayoutParams);
    linearLayout.addView(serif_TV, linearLayoutLayoutParams);
}
}

```

运行后效果如图 2-11 所示。

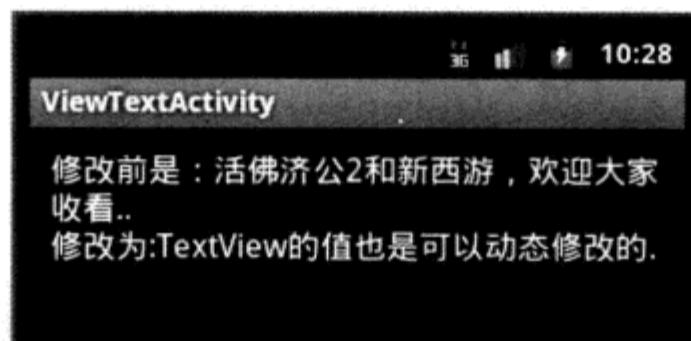


图2-11 运行效果

2.7.4 在屏幕中显示编辑框

在手机屏幕中，可以和网页一样能够显示可输入文本信息的文本框，此功能是通过编辑框控件 EditText 实现的。编辑框控件 EditText 的用法和 TextView 类似，它能生成一个可编辑的文本框。

本实例的具体实现流程如下所示。

step 01 在主窗口界面中添加一个 EditText 控件，然后设定其监听器在接收到单击事件时，程序打开 EditText 的界面。定义文件 editview.xml 来布局程序打开的 EditText 界面，具体代码如下所示。

```

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

```



```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
  >
//供用户输入值
<EditText android:id="@+id/edit_text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="这里可以输入文字" />
//用于获取输入的值
<Button android:id="@+id/get_edit_view_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="获取EditText的值" />
</LinearLayout>
```

step 02 编写事件处理文件 EditTextActivity.java，主要代码如下所示。

```
public class EditTextActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("EditTextActivity");
        setContentView(R.layout.editview);
        find_and_modify_text_view();
    }
    private void find_and_modify_text_view() {
        Button get_edit_view_button = (Button) findViewById(R.id.get_
edit_view_button);
        get_edit_view_button.setOnClickListener(get_edit_view_button_
listener);
    }

    private Button.OnClickListener get_edit_view_button_listener = new
Button.OnClickListener() {
        /**响应代码，显示EditText中的值*/
        public void onClick(View v) {
            EditText edit_text = (EditText) findViewById(R.id.edit_text);
            CharSequence edit_text_value = edit_text.getText();
            setTitle("EditText的值:"+edit_text_value);
        }
    };
}
```

执行后先显示默认的文本和输入框，如图 2-12 所示；输入一段文本并单击【获取 EditText 的值】按钮后会获取输入的文字，并显示出输入的文字，如图 2-13 所示。

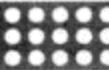


图2-12 初始效果

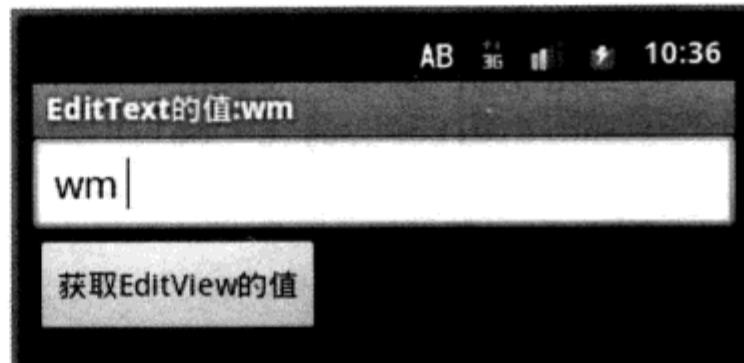


图2-13 运行效果

2.7.5 在屏幕中显示复选框

在网页中有复选框这一概念，复选框提供一个制造单一选择开关的方法；它包括一个小框和一个标签。典型的复选框有一个小的“√”（或者所设置的其他类型）或是空的，这依靠项目是否被选择来决定的。在收集屏幕上也可以实现复选框的效果，此功能是通过 CheckBox 控件实现的。CheckBox 控件能够为用户提供输入信息，用户可以一次性选择多个选项。在 Android 中，使用 CheckBox 控件也需要在 XML 布局文件中定义。

本实例的具体实现流程如下所示。

step 01 编写布局文件 check_box.xml，插入了拥有 4 个可选选项的 CheckBox 控件供用户选择，然后插入了一个 Button 控件来响应用户的单击事件。具体代码如下所示。

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

    <CheckBox android:id="@+id/plain_cb"
        android:text="Plain"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

    <CheckBox android:id="@+id/serif_cb"
        android:text="Serif"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:typeface="serif"
        />

    <CheckBox android:id="@+id/bold_cb"
        android:text="Bold"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

    <Button android:id="@+id/get_value_bt"
        android:text="获取EditView的值"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

```



```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold"
/>
<CheckBox android:id="@+id/italic_cb"
    android:text="Italic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="italic"
/>
<Button android:id="@+id/get_view_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="获取CheckBox的值"/>
</LinearLayout>
```

step 02 编写单击按钮事件的处理文件 CheckBoxActivity.java，把用户选中的选项值显示在 Title 上面。主要代码如下所示。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTitle("CheckBoxActivity");
    setContentView(R.layout.check_box);
    find_and_modify_text_view();
}

private void find_and_modify_text_view() {
    plain_cb = (CheckBox) findViewById(R.id.plain_cb);
    serif_cb = (CheckBox) findViewById(R.id.serif_cb);
    italic_cb = (CheckBox) findViewById(R.id.italic_cb);
    bold_cb = (CheckBox) findViewById(R.id.bold_cb);
    Button get_view_button = (Button) findViewById(R.id.get_view_button);
    get_view_button.setOnClickListener(get_view_button_listener);
}

private Button.OnClickListener get_view_button_listener = new
Button.OnClickListener() {
    public void onClick(View v) {
        String r = "";
        if (plain_cb.isChecked()) {
```

```

        r = r + "," + plain_cb.getText();
    }
    if (serif_cb.isChecked()) {
        r = r + "," + serif_cb.getText();
    }
    if (italic_cb.isChecked()) {
        r = r + "," + italic_cb.getText();
    }
    if (bold_cb.isChecked()) {
        r = r + "," + bold_cb.getText();
    }
    setTitle("Checked: " + r);
}
}
}

```

执行后先显示 4 个选项值供用户选择，如图 2-14 所示；用户选择某些选项并单击按钮【获取复选框的值】，将在屏幕顶端用文本提示已经选择的选项，如图 2-15 所示。



图 2-14 初始效果



图 2-15 运行效果

2.7.6 在屏幕中显示单选框

与复选框相比，单选框只能选中一项命令，是图形用户界面上的一种控件。单选框允许用户在一组选项中选择其中的一个选项。单选框的外观一般是一个空白的圆洞，而在它的旁边则通常有一个文字的标签。它的用途除了描述之外，还可用于选择该选择：当用户按下标签，所应的选择钮就会被选上。已选上的选择钮一般会在圆洞内加上一小圆点。单项选择控件 RadioGroup 是和多项选择控件 CheckBox 相对应的，但是它只能供用户选择一个选项。

本实例的具体实现流程如下所示。

step 01 编写布局文件 radio_group.xml，在里面设置 4 个选项供用户选择，具体代码如下所示。

```

<?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:orientation="vertical">
    <RadioGroup>
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:checkedButton="@+id/lunch"
        android:id="@+id/menu">
        <RadioButton
            android:text="AA"
            android:id="@+id/breakfast"/>
        <RadioButton
            android:text="BB"
            android:id="@+id/lunch" />
        <RadioButton
            android:text="CC"
            android:id="@+id/dinner" />
        <RadioButton
            android:text="DD"
            android:id="@+id/all" />
    </RadioGroup>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="清除"
        android:id="@+id/clear" />
</LinearLayout>
```

通过在上述代码，在屏幕插入了一个 RadioGroup 空间，在里面提供了 4 个选项供用户选择，然后插入了一个 Button 控件，用于清除用户选择的选项。

step 02 编写事件处理文件 RadioGroupActivity.java，主要代码如下所示。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.radio_group);
    setTitle("RadioGroupActivity");
    mRadioGroup = (RadioGroup) findViewById(R.id.menu);
    Button clearButton = (Button) findViewById(R.id.clear);
    clearButton.setOnClickListener(this);
}
```

当用户单击【清除】按钮后会使用 setTitle 修改 Title 值为“RadioGroupActivity”，然后获取 RadioGroup 对象和按钮对象。执行后先显示 4 个选项值供用户选择，如图 2-16 所示。



用户选择一个选项并单击【清除】按钮后会清除选择的选项，如图 2-17 所示。



图2-16 初始效果



图2-17 运行效果

2.7.7 在屏幕中显示下拉列表框

在 Android 手机屏幕中，可以使用下拉列表控件 Spinner 来显示一个下拉列表框效果。使用下拉列表框后，用户不需要输入的数据，只需选择一个选项后即可在框中完成数据输入工作。Spinner 位于 android.widget 包下，每次只显示用户选中的元素，当用户再次点击时，会弹出选择列表供用户选择，而选择列表中的元素同样来自适配器。Spinner 是类 View 的一个子类。

本实例的具体实现流程如下所示。

step 01 在主布局文件 main.xml 中添加按钮【Spinner】，单击此按钮后会启动新界面 SpinnerActivity。主要代码如下所示。

```
<Button android:id="@+id/spinner_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Spinner"
    />
```

step 02 在文件 MainActivity.java 中编写按钮单击事件的处理代码，具体如下所示。

```
private Button.OnClickListener spinner_button_listener = new Button.
OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setClass(MainActivity.this, SpinnerActivity.
        class);
        startActivity(intent);
    }
};
```

在上述代码中，启动了 SpinnerActivity，此 SpinnerActivity 可以展示 Spinner 组件的界面。在具体实现上，先创建了 SpinnerActivity 的 Activity，然后修改了其 onCreate 方法，设置其对应模板为 spinner.xml。在文件 SpinnerActivity.java 中的对应代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.spinner);  
    find_and_modify_view();  
}
```

step 03 编写文件下拉列表框界面的布局文件 spinner.xml，在里面添加了 2 个 TextView 控件和 2 个 Spinner 控件。主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/  
    android"  
        android:orientation="vertical"  
        android:layout_width="fill_parent"  
        android:layout_height="fill_parent"  
    >  
    <TextView  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:text="Spinner_1"  
    />  
    <Spinner android:id="@+id/spinner_1"  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:drawSelectorOnTop="false"  
    />  
</LinearLayout>
```

step 04 在设置文件 AndroidManifest.xml 中设置定义的 Spinner 组件的 ID 为 spinner_1，宽度占满了其父元素 “LinearLayout” 的宽，高度自适应。主要代码如下所示。

```
<activity android:name="SpinnerActivity"></activity>
```

经过上述操作后，就可以在屏幕中生成一个简单的单选选项界面，但是在列表中并没有选项值。如果要在下拉列表中实现可供用户选择的选项值，需要在里面填充一些数据。

step 05 载入列表数据，首先定义需要载入的数据，然后在方法 onCreate 中通过调用 find_and_modify_view() 来载入数据。在文件 SpinnerActivity.java 中实现上述功能的代码如下所示。

```
private static final String[] mCountries = { "China", "Russia",  
    "Germany",  
    "Ukraine", "Belarus", "USA" };  
private void find_and_modify_view() {  
    spinner_c = (Spinner) findViewById(R.id.spinner_1);  
    allcountries = new ArrayList<String>();  
    for (int i = 0; i < mCountries.length; i++) {  
        allcountries.add(mCountries[i]);  
    }  
}
```



```

        }
        aspnCountries = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, allcountries);
        aspnCountries
            .setDropDownViewResource(android.R.layout.simple_
        spinner_dropdown_item);
        spinner_c.setAdapter(aspnCountries);
    }
}

```

通过上述代码，将定义的 mCountries 数据载入到了 Spinner 组件中。

step 06 在文件 spinner.xml 中预定义数据，即在文件 spinner.xml 的模板中再添加一个 Spinner 组件。主要代码如下所示。

```

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Spinner_2 From arrays xml file"
/>
<Spinner android:id="@+id/spinner_2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:drawSelectorOnTop="false"
/>

```

step 07 在文件 SpinnerActivity.java 中初始化 Spinner 中的值，具体代码如下所示。

```

spinner_2 = (Spinner) findViewById(R.id.spinner_2);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
    this, R.array.countries, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
spinner_2.setAdapter(adapter);

```

在上述代码中，将 R.array.countries 对应值载入到了 spinner_2 中，在 R.array.countries 中对应的值是在文件 array.xml 中预先定义的。文件 array.xml 的主要代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Used in Spinner/spinner_2.java -->
    <string-array name="countries">
        <item>China2</item>
        <item>Russia2</item>
        <item>Germany2</item>
        <item>Ukraine2</item>
        <item>Belarus2</item>
        <item>USA2</item>
    </string-array>
</resources>

```



通过上述代码，预定义了一个名为“countries”的数组。

执行后先显示 2 个下拉列表表单，如图 2-18 所示；当单击一个下拉列表单后面的▼会弹出一个由 Spinner 组件实现的下拉选项框，如图 2-19 所示；当选择选择一个选项后，选项值会自动出现在输入表单中，如图 2-20 所示。



图2-18 初始效果



图2-19 运行效果



图2-20 选择的值自动出现在表单中

2.7.8 在屏幕中实现自动输入文本

在 Android 手机屏幕中，可以使用控件 AutoCompleteTextView 实现自动输入文本功能。此控件的主要功能是帮助用户自动输入数据，例如当用户输入一个字符后，能够根据这个字符提示显示出与之相关的数据。此应用在搜索引擎中比较常见，例如用户在百度中输入关键字“客户”后，会在下拉列表中自动显示出相关的关键词。如图 2-21 所示。

在控件 AutoCompleteTextView 中主要有如下三个方法。

- ◆ clearListSelection()：清除选中的列表项；
- ◆ dismissDropDown()：如果存在关闭下拉菜单；
- ◆ getAdapter()：获取适配器。

本实例的具体实现流程如下所示。

step 01 修改主布局文件 main.xml，在里面添加一个 TextView、一个 AutoCompleteTextView 和一个 Button。主要代码如下所示。



图2-21 百度的输入提示框

```

<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget0"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/TextView_InputShow"
    android:layout_width="228px"
    android:layout_height="47px"
    android:text="请输入"
    android:textSize="25px"
    android:layout_x="42px"
    android:layout_y="37px"
>
</TextView>
<AutoCompleteTextView
    android:id="@+id/AutoCompleteTextView_input"
    android:layout_width="275px"
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="18sp"
    android:layout_x="23px"
    android:layout_y="98px"
>
</AutoCompleteTextView>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="127dip"
    android:text="清空"
    android:id="@+id/Button_clean"
    android:layout_y="150dip">
</Button>
</AbsoluteLayout>

```

step 02 修改文件 mainActivity.java，添加自动完成功能处理事件的代码。主要代码如下所示。

```

/*定义要使用的类对象*/
private String[] normalString =
    new String[] {
        "Android", "Android Blog", "Android Market", "Android SDK",
        "Android AVD", "BlackBerry", "BlackBerry JDE", "Symbian",
        "Symbian Carbide", "Java 2ME", "Java FX", "Java 2EE",
        "Java 2SE", "Mobile", "Motorola", "Nokia", "Sun",
        "Nokia Symbian", "Nokia forum", "WindowsMobile", "Broncho",
}

```

```
"Windows XP", "Google", "Google Android ", "Google 浏览器",
"IBM", "MicroSoft", "Java", "C++", "C", "C#", "J#", "VB"
};

@SuppressWarnings("unused")
private TextView show;
private AutoCompleteTextView autoTextView;
private Button clean;
private ArrayAdapter<String> arrayAdapter;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /*装入主屏布局main.xml*/
    setContentView(R.layout.main);
    /*从XML中获取UI元素对象*/
    show = (TextView) findViewById(R.id.TextView_InputShow);
    autoTextView =
        (AutoCompleteTextView) findViewById(R.id.AutoCompleteTextView_
input);
    clean = (Button) findViewById(R.id.Button_clean);
    /*实现一个适配器对象，用来给自动完成输入框添加自动装入的内容*/
    arrayAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line,
normalString);
    /*给自动完成输入框添加内容适配器*/
    autoTextView.setAdapter(arrayAdapter);
    /*给清空按钮添加点击事件处理监听器*/
    clean.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            /*清空*/
            autoTextView.setText("");
        }
    });
}
}
```

执行后可以在文本框中输入数据，输入后会根据预先准备的数据进行提示，如图 2-22 所示。

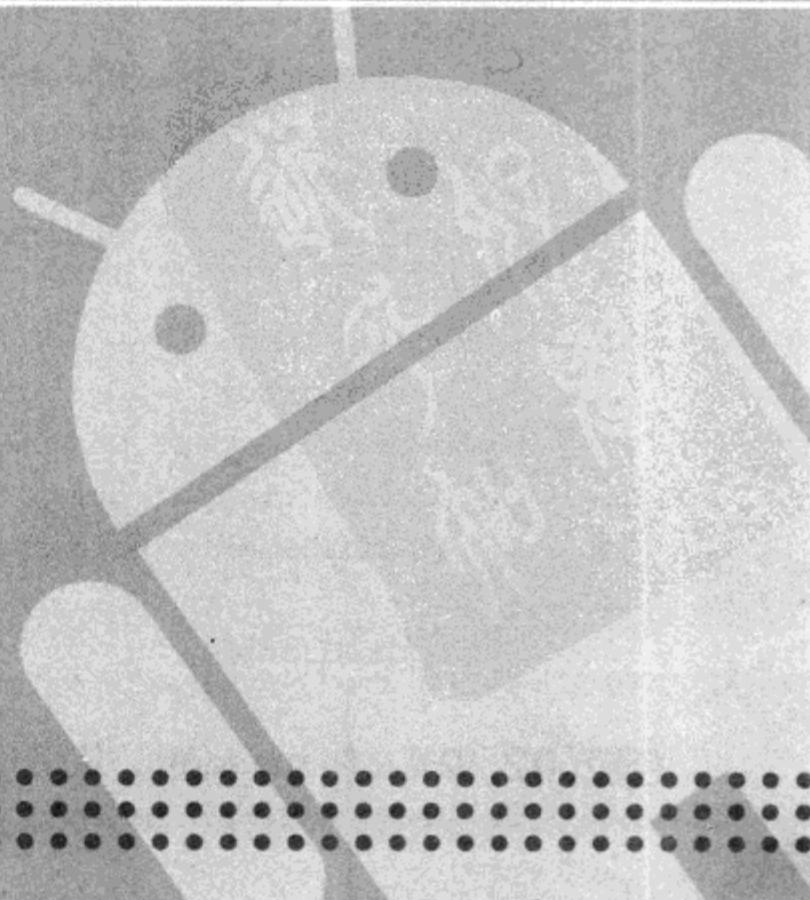


图 2-22 弹出文本输入提示框

第 3 章

基本控件实战演练

作为手机开发项目来说，几乎所有的应用功能都需要用控件来实现。控件就如同在 Web 开发中的一个模块，通过调用这些控件能够实现对应的功能效果。其实在本书第 2 章的内容中，所有的实例都是基于控件实现的，无论是 Button 还是 TextView，都是一个控件。之所以将它们作为单独的一章来讲解，是因为屏幕布局的重要性。在本章的内容中，将通过具体的实例来讲解 Android 系统中各个常用控件的基本用法。



3.1 使用RadioGroup控件实现选择处理

在本实例中布置了一个 TextView Widget 控件和一个 RadioGroup 控件，并在 RadioGroup 中放置 2 个 RadioButton 控件，默认样式为不选择。当程序运行后使用 onCheckedChanged 作为启动事件装置，当用户选择其中一个按钮时显示被选择项的内容，最后将 RadioButton 的选项文字显示于 TextView 当中。

本实例的源码保存在【光盘\daima\第 3 章\tixing】，实现流程如下所示。

step 01 文件 main.xml 中分别插入 1 个 TextView 控件、1 个 RadioGroup 控件和 2 个 RadioButton 控件，主要代码如下所示。

```
<TextView  
    android:id="@+id/myTextView"  
    android:layout_width="228px"  
    android:layout_height="49px"  
    android:text="@string/str_radio_question1"  
    android:textSize="30sp"  
    android:layout_x="37px"  
    android:layout_y="3px"  
/>  
<RadioGroup  
    android:id="@+id/myRadioGroup"  
    android:layout_width="137px"  
    android:layout_height="216px"  
    android:orientation="vertical"  
    android:layout_x="3px"  
    android:layout_y="54px"  
>>  
<RadioButton  
    android:id="@+id/myRadioButton1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/tr_radio_op1"  
/>>  
<RadioButton  
    android:id="@+id/myRadioButton2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/tr_radio_op2"  
/>>  
</RadioGroup>
```

step 02 编写文件 xuanze.java，使用 OnCheckedChangeListener 来启动 RadioGroup 的事件，启动后

将被勾选选项的文字显示在 TextView 文本框中。文件 xuanze.java 的主要代码如下所示。

```

public class xuanze extends Activity
{
    public TextView mTextView1;
    public RadioGroup mRadioGroup1;
    public RadioButton mRadio1,mRadio2;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /*取得 TextView、RadioGroup、RadioButton对象*/
        mTextView1 = (TextView) findViewById(R.id.myTextView);
        mRadioGroup1 = (RadioGroup) findViewById(R.id.myRadioGroup);
        mRadio1 = (RadioButton) findViewById(R.id.myRadioButton1);
        mRadio2 = (RadioButton) findViewById(R.id.myRadioButton2);

        /*RadioGroup用OnCheckedChangeListener来运行*/
        mRadioGroup1.setOnCheckedChangeListener(mChangeRadio);
    }

    private RadioGroup.OnCheckedChangeListener mChangeRadio = new
        RadioGroup.OnCheckedChangeListener()
    {
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId)
        {
            // TODO Auto-generated method stub
            if(checkedId==mRadio1.getId())
            {
                /*把mRadio1的内容传到mTextView1*/
                mTextView1.setText(mRadio1.getText());
            }
            else if(checkedId==mRadio2.getId())
            {
                /*把mRadio2的内容传到mTextView1*/
                mTextView1.setText(mRadio2.getText());
            }
        }
    };
}

```

实例执行后的效果如图 3-1 所示，当选择一个选项后会提示显示出选择的值，如图 3-2 所示。

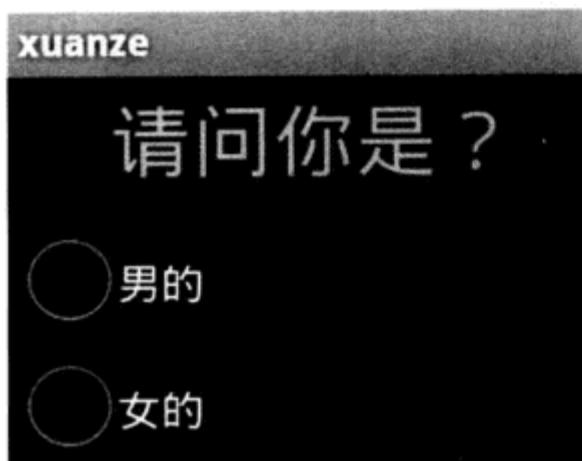


图3-1 初始效果

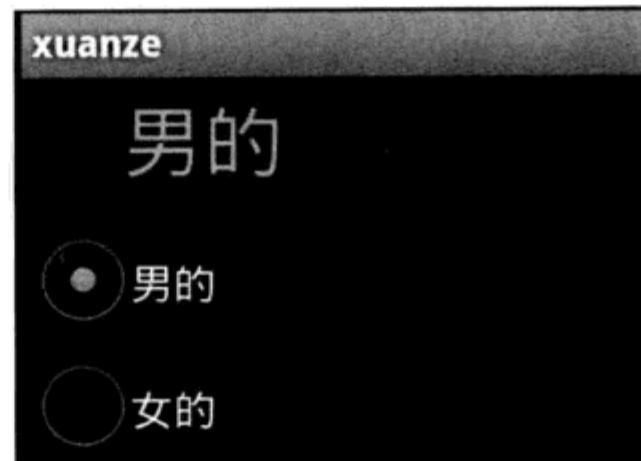


图3-2 输出提示

3.2 使用屏幕中实现一个购物清单

在 Android 系统中，控件 CheckBox 是一个可以供用户选择的复选框控件。在本实例中，通过 CheckBox.setOnCheckedChangeListener 在程序中设置了 3 个 CheckBox 复选项，分别表示 3 种物品列表，当用户选中其中的一个物品后，会在 TextView 控件中里显示所选择的物品列表。

本实例的源码保存在【光盘\daima\第 3 章\fuxuan】，实现流程如下所示。

step 01 在文件 main.xml 中分别插入了 1 个 TextView 控件和 3 个 CheckBox 控件，主要代码如下所示。

```
<TextView  
    android:id="@+id/myTitle"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/str_title"  
>  
</TextView>  
<CheckBox  
    android:id="@+id/myCheckBox1"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/str_checkbox1"  
>  
</CheckBox>  
<CheckBox  
    android:id="@+id/myCheckBox2"
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/str_checkbox2"
  >
</CheckBox>
<CheckBox
    android:id="@+id/myCheckBox3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/str_checkbox3"
  >
</CheckBox>
<TextView
    android:id="@+id/myTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
  >
</TextView>
```

step 02 在文件 fuxuan.java 中分别构建了 3 个 CheckBox 对象和 1 个 TextView 对象，然后通过响应 setOnCheckedChangeListener 事件，使用方法 onCheckedChanged() 来更新 TextView 中显示的文字。在具体实现上，提供了一个货物清单供用户选择要买的商品。当用户选择商品后，会显示对应的购买物品，如果任选两个会超过我们的购物能力，会显示超额提示。文件 fuxuan.java 的主要代码如下所示。

```
public class fuxuan extends Activity
{
    /*声明对象变量*/
    private TextView mTextView1;
    private CheckBox mCheckBox1;
    private CheckBox mCheckBox2;
    private CheckBox mCheckBox3;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /*通过findViewById取得TextView对象并调整文字内容*/
        mTextView1 = (TextView) findViewById(R.id.myTextView1);
        mTextView1.setText("你所选择的项目有： ");
```



```
/*通过findViewById取得三个CheckBox对象*/
mCheckBox1=(CheckBox)findViewById(R.id.myCheckBox1);
mCheckBox2=(CheckBox)findViewById(R.id.myCheckBox2);
mCheckBox3=(CheckBox)findViewById(R.id.myCheckBox3);

/*设置OnCheckedChangeListener给三个CheckBox对象*/
mCheckBox1.setOnCheckedChangeListener(mCheckBoxChanged);
mCheckBox2.setOnCheckedChangeListener(mCheckBoxChanged);
mCheckBox3.setOnCheckedChangeListener(mCheckBoxChanged);
}

/*声明并建构onCheckedChangeListener对象*/
private CheckBox.OnCheckedChangeListener mCheckBoxChanged
= new CheckBox.OnCheckedChangeListener()
{
    /*implement onCheckedChanged方法*/
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
                                boolean isChecked)
    {
        // TODO Auto-generated method stub
        /*通过getString()取得CheckBox的文字字符串*/
        String str0="所选的项目为：";
        String str1=getString(R.string.str_checkbox1);
        String str2=getString(R.string.str_checkbox2);
        String str3=getString(R.string.str_checkbox3);
        String plus=";";
        String result="但是超过预算啰!!";
        String result2="还可以再多买几个喔!!";

        /*任一CheckBox被勾选后，该CheckBox的文字会改变TextView的文字内容
         * 三个对象总共八种情境*/
        if(mCheckBox1.isChecked()==true & mCheckBox2.isChecked()==true
           & mCheckBox3.isChecked()==true)
        {
            mTextView1.setText(str0+str1+plus+str2+plus+str3+result);
        }
        else if(mCheckBox1.isChecked()==false & mCheckBox2.isChecked()==true
           & mCheckBox3.isChecked()==true)
        {
            mTextView1.setText(str0+str2+plus+str3+result);
        }
    }
}
```

```
        else if(mCheckBox1.isChecked() == true & mCheckBox2.
isChecked() == false
                & mCheckBox3.isChecked() == true)
        {
            mTextView1.setText(str0+str1+plus+str3+result);
        }
        else if(mCheckBox1.isChecked() == true & mCheckBox2.
isChecked() == true
                & mCheckBox3.isChecked() == false)
        {
            mTextView1.setText(str0+str1+plus+str2+result);
        }
        else if(mCheckBox1.isChecked() == false & mCheckBox2.
isChecked() == false
                & mCheckBox3.isChecked() == true)
        {
            mTextView1.setText(str0+str3+plus+result2);
        }
        else if(mCheckBox1.isChecked() == false & mCheckBox2.
isChecked() == true
                & mCheckBox3.isChecked() == false)
        {
            mTextView1.setText(str0+str2);
        }
        else if(mCheckBox1.isChecked() == true & mCheckBox2.
isChecked() == false
                & mCheckBox3.isChecked() == false)
        {
            mTextView1.setText(str0+str1);
        }
        else if(mCheckBox1.isChecked() == false & mCheckBox2.
isChecked() == false
                & mCheckBox3.isChecked() == false)
        {
            mTextView1.setText(str0);
        }
    }
}
};
```

实例执行后的效果如图 3-3 所示，当选择一种商品后会在下方显示对应的提示信息，如图 3-4 所示。如果任意选两种，则会显示超出预算的提示，如图 3-5 所示。

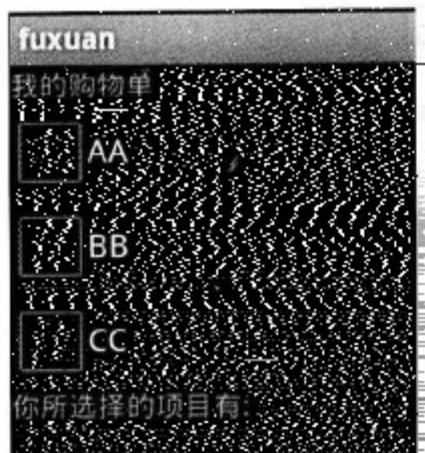


图3-3 初始效果



图3-4 显示提示效果

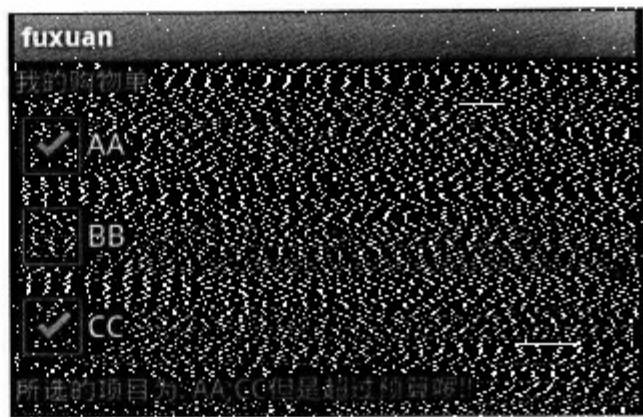


图3-5 超出预算提示效果

3.3 在手机屏幕中实现相框效果

在本实例中，我们需要预先准备了三张图片，其中两张是外框图，一张是内框图，将这三张图片放在“res/drawable”目录下面。图片是 PNG 格式的图形文件，图像大小是手机屏幕大小，读者可以依据手机的分辨率来调整 ImageView 的大小。

在 Layout 布局中创建了两个 ImageView 图像视图，并且以绝对坐标的方式布局在一起，然后在它们的下方放上两个 Button 按钮，单击按钮后能够实现切换图片功能，在此需要设置 Button 事件里处理置换图片的动作。当点击 Button1 按钮后，在 ImageView1 中会显示“right”中的图片；点击 Button2 后，在 ImageView1 中会显示“left”的图片，而 ImageView2 都是固定不动的图片。

本实例的源码保存在【光盘\daima\第3章\xiangkuang】，实现流程如下所示。

step 01 在文件 main.xml 中分别创建 2 个 ImageView 控件，其中 1 个作为外框来使用，另一个作为内框来使用。在摆放图片时，需要做一个排序堆栈顺序，将前景图放在上方（以 AbsoluteLayout），将背景图放在前景图的下方。主要代码如下所示。

```
<ImageView  
    android:id="@+id/myImageView1"  
    android:layout_width="320px"  
    android:layout_height="280px"  
    android:layout_x="0px"  
    android:layout_y="36px"  
/>  
<ImageView  
    android:id="@+id/myImageView2"  
    android:layout_width="104px"  
    android:layout_height="157px"  
    android:layout_x="101px"  
    android:layout_y="119px"  
/>
```



```
<Button  
    android:id="@+id/myButton1"  
    android:layout_width="105px"  
    android:layout_height="66px"  
    android:text="图片1"  
    android:layout_x="9px"  
    android:layout_y="356px"  
/>  
  
<Button  
    android:id="@+id/myButton2"  
    android:layout_width="105px"  
    android:layout_height="66px"  
    android:text="图片2"  
    android:layout_x="179px"  
    android:layout_y="356px"  
/>
```

step 02 编写文件 xiangkuang.java，其核心功能是通过 getResources() 方法实现的，此方法负责访问 Resource ID，无论是访问资源里的图文件、文字都要用到 getResources()；在此使用 getResources().getDrawable() 来载入 res/drawable 里的图文件，并将图片放置在 ImageView 当中。文件 xiangkuang.java 的主要代码如下所示。

```
public class xiangkuang extends Activity  
{  
    /*声明 Button、ImageView对象*/  
    private ImageView mImageView01;  
    private ImageView mImageView02;  
    private Button mButton01;  
    private Button mButton02;  
  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState)  
    {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        /*取得 Button、ImageView对象*/  
        mImageView01 = (ImageView) findViewById(R.id.myImageView1);  
        mImageView02 = (ImageView) findViewById(R.id.myImageView2);  
        mButton01 = (Button) findViewById(R.id.myButton1);  
        mButton02 = (Button) findViewById(R.id.myButton2);  
  
        /*设置ImageView背景图*/
```



```
mImageView01.setImageDrawable(getResources().  
    getDrawable(R.drawable.right));  
mImageView02.setImageDrawable(getResources().  
    getDrawable(R.drawable.aaa));  
  
/*用OnCClickListener事件来启动*/  
mButton01.setOnClickListener(new Button.OnClickListener()  
{  
    @Override  
    public void onClick(View v)  
    {  
        /*当启动后， ImageView立刻换背景图*/  
        mImageView01.setImageDrawable(getResources().  
            getDrawable(R.drawable.right));  
    }  
});  
  
mButton02.setOnClickListener(new Button.OnClickListener()  
{  
    @Override  
    public void onClick(View v)  
    {  
        mImageView01.setImageDrawable(getResources().  
            getDrawable(R.drawable.left));  
    }  
});  
  
}  
}
```

实例执行后的初始效果如图 3-6 所示，当分别单击按钮【pic1】和【pic2】后，会分别显示用“Left”和“Right”素材的相框，如图 3-7 所示。

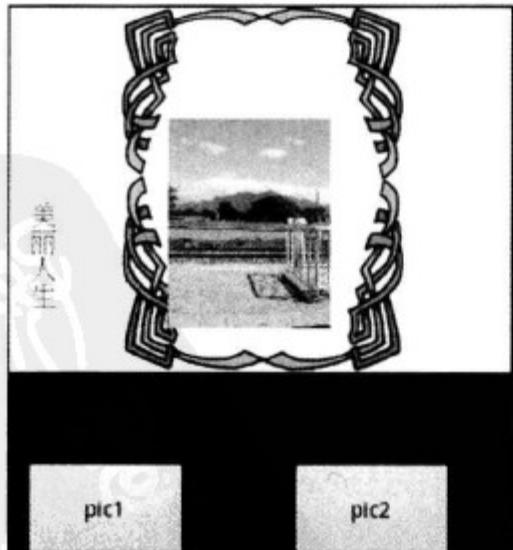


图3-6 初始效果



图3-7 不同素材相框



```
private View inflateView(int resource) {
    LayoutInflator vi = (LayoutInflator) getSystemService(Context.
LAYOUT_INFLATER_SERVICE);
    return vi.inflate(resource, null);
}

protected void showNotification() {

    NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);

    CharSequence title = "首都";
    CharSequence contents = "北京";

    PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
        new Intent(this, ActivityMain.class), 0);

    Notification notification = new Notification(R.drawable.default_icon,
        title, System.currentTimeMillis());

    notification.setLatestEventInfo(this, title, contents, contentIntent);

    // 100ms延迟后，振动250ms，停止100ms后振动500ms
    notification.vibrate = new long[] { 100, 250, 100, 500 };

    notificationManager.notify(NOTIFICATIONS_ID, notification);
}
}
```

此文件的处理流程如下所示。

- ◆ 实例化 Toast，每个 Toast 和一个 View 相关。
- ◆ 设置 Toast 的长短，使用 “showToast(Toast.LENGTH_SHORT);” 设置 Toast 短时间显示，使用 “showToast(Toast.LENGTH_LONG);” 设置 Toast 长时间显示。
- ◆ 通过函数 showToast 来显示 Toast，具体说明如下。
 - 当单击【恒久显示】按钮后，程序会长时间地显示提醒，并用 Notification 在状态栏中提示用户。
 - 当单击【短暂显示】按钮后，程序会短时间地显示提醒。

step 06 单击第二个 Button 按钮后来到对应的新界面，新界面使用文件 activity_toast.xml 实现布局，主要代码如下所示。

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

translate 和 alpha 两种。具体代码如下。

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="-100%p"
        android:duration="300"
    >
    </translate>
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:duration="300">
    </alpha>
</set>
```

step 04 编写文件 xiala.java 中，在新建 ArrayAdapter 时使用 ArrayAdapter(Context context, int textViewResourceId, T[] objects) 这个生成器，参数 textViewResourceId 使用 Android 提供的 ResourceID，参数 objects 为必须传递的字符串数组 (String Array)。文件 xiala.java 的主要代码如下所示。

```
public class xiala extends Activity
{
    private static final String[] countriesStr =
    { "AA", "BB", "CC", "DD" };
    private TextView myTextView;
    private Spinner mySpinner;
    private ArrayAdapter<String> adapter;
    Animation myAnimation;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);

        /* 以findViewById()取得对象 */
        myTextView = (TextView) findViewById(R.id.myTextView);
        mySpinner = (Spinner) findViewById(R.id.mySpinner);

        adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, countriesStr);
        /* myspinner_dropdown为自定义下拉菜单模式定义在res/layout目录下 */
```

```
adapter.setDropDownViewResource(R.layout.myspinner_dropdown);

/* 将ArrayAdapter添加Spinner对象中 */
mySpinner.setAdapter(adapter);

/* 将mySpinner添加OnItemSelectedListener */
mySpinner.setOnItemSelectedListener
(new Spinner.OnItemSelectedListener()
{
    @Override
    public void onItemSelected
        (AdapterView<?> arg0, View arg1, int arg2,
         long arg3)
    {
        /* 将所选mySpinner的值带入myTextView中 */
        myTextView.setText("您选择的是" + countriesStr[arg2]);
        /* 将mySpinner显示 */
        arg0.setVisibility(View.VISIBLE);
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0)
    {
        // TODO Auto-generated method stub
    }
});

/* 取得Animation定义在res/anim目录下 */
myAnimation = AnimationUtils.loadAnimation(this, R.anim.my_anim);

/* 将mySpinner添加OnTouchListener */
mySpinner.setOnTouchListener(new Spinner.OnTouchListener()
{
    @Override
    public boolean onTouch(View v, MotionEvent event)
    {
        /* 将mySpinner运行Animation */
        v.startAnimation(myAnimation);
        /* 将mySpinner隐藏 */
        v.setVisibility(View.INVISIBLE);
        return false;
    }
})
```



```
});  
  
    mySpinner.setOnFocusChangeListener(new Spinner.OnFocusChangeListener() {  
        @Override  
        public void onFocusChange(View v, boolean hasFocus) {  
            // TODO Auto-generated method stub  
        }  
    });  
}  
}
```

执行后的效果如图 3-8 所示，单击下拉菜单后显示悬浮选项效果界面，在里面有 4 个选项供用户选择，如图 3-9 所示。

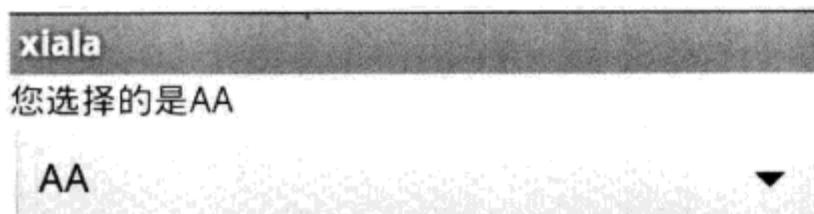


图3-8 初始效果

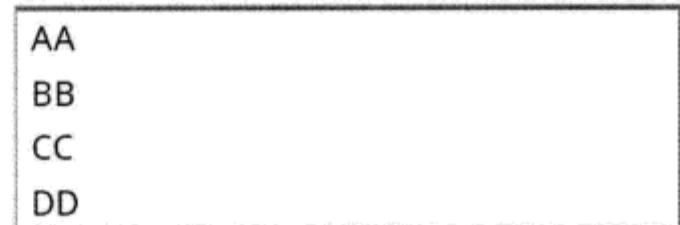


图3-9 4个选项

当选择一个选项后，会显示出对应的提示信息，例如选择了“CC”后的效果如图 3-10 所示。

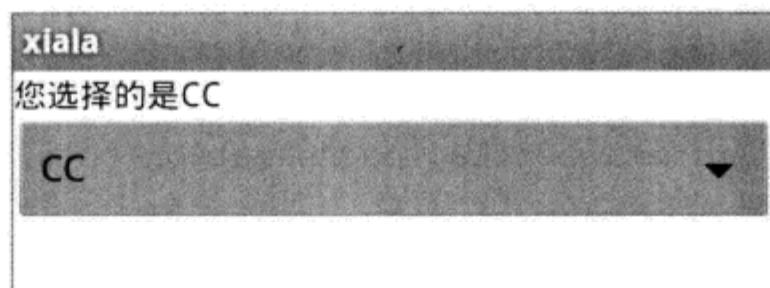


图3-10 提示信息

3.5 在屏幕中实现一个相簿功能

在本实例中，首先将 6 张 PNG 图片导入到“Drawable”目录下，在响应 onCreate 的同时将素材图片载入到 Gallery Widget 中；然后添加一个 OnItemClick 事件以取得图片的 ID 编号，这样就可以响应用户点击图片时的状态，完成 Gallery 的高级使用。

本实例的源码保存在【光盘\daima\第 3 章\xiangbu】，实现流程如下所示。

step 01 在文件 main.xml 中插入了 1 个 Gallery 控件，其主要代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<Gallery xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mygallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>

```

step 02 编写文件 attrs.xml 设置定义 layout 外部资源的样式，并且设置随着滑动而改变 layout 背景图的效果。主要代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Gallery">
        <attr name="android:galleryItemBackground"/>
    </declare-styleable>
</resources>

```

step 03 编写文件 xiangbu.java，在此文件中，ImageAdapter 继承于 BaseAdapter class 的未实现方法的重写构造，通过 Gallery 中的 OnItemClick() 方法来响应了图片滑动及 Layout 宽和高的设置。文件 xiangbu.java 的主要代码如下所示。

```

public class xiangbu extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*通过findViewById取得*/
        Gallery g = (Gallery) findViewById(R.id.mygallery);
        /* 添加一ImageAdapter并设置给Gallery对象 */
        g.setAdapter(new ImageAdapter(this));

        /* 设置一个itemclickListener并Toast被点击图片的位置 */
        g.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick
            (AdapterView<?> parent, View v, int position, long id)
            {
                Toast.makeText
                (example9.this, getString(R.string.my_gallery_text_pre)
                + position+ getString(R.string.my_gallery_text_post),
                Toast.LENGTH_SHORT).show();
            }
        });
    }
}

```

```
    }

    /* 改写BaseAdapter自定义一ImageAdapter class */
    public class ImageAdapter extends BaseAdapter
    {
        /*声明变量*/
        int mGalleryItemBackground;
        private Context mContext;

        /*ImageAdapter的构造器*/
        public ImageAdapter(Context c)
        {
            mContext = c;

            /* 使用在res/values/attrs.xml中的<declare-styleable>定义
             * 的Gallery属性.*/
            TypedArray a = obtainStyledAttributes(R.styleable.Gallery);

            /*取得Gallery属性的Index id*/
            mGalleryItemBackground = a.getResourceId
                (R.styleable.Gallery_android_galleryItemBackground, 0);

            /*让对象的styleable属性能够反复使用*/
            a.recycle();
        }

        /* 覆盖的方法getCount,返回图片数目 */
        public int getCount()
        {
            return myImageIds.length;
        }

        /* 覆盖的方法getItemId,返回图像的数组id */
        public Object getItem(int position)
        {
            return position;
        }
        public long getItemId(int position)
        {
            return position;
        }
    }
```

```
/* 覆盖的方法getView,返回一View对象 */
public View getView
(int position, View convertView, ViewGroup parent)
{
    /*产生ImageView对象*/
    ImageView i = new ImageView(mContext);
    /*设置图片给imageView对象*/
    i.setImageResource(myImageIds[position]);
    /*重新设置图片的宽高*/
    i.setScaleType(ImageView.ScaleType.FIT_XY);
    /*重新设置Layout的宽高*/
    i.setLayoutParams(new Gallery.LayoutParams(136, 88));
    /*设置Gallery背景图*/
    i.setBackgroundDrawable(mGalleryItemBackground);
    /*返回imageView对象*/
    return i;
}

/*建构一Integer array并取得预加载Drawable的图片id*/
private Integer[] myImageIds =
{
    R.drawable.photo1,
    R.drawable.photo2,
    R.drawable.photo3,
    R.drawable.photo4,
    R.drawable.photo5,
    R.drawable.photo6,
};
}
```

执行后可以通过滑动鼠标的方式来浏览每一幅图片，如图 3-11 所示。

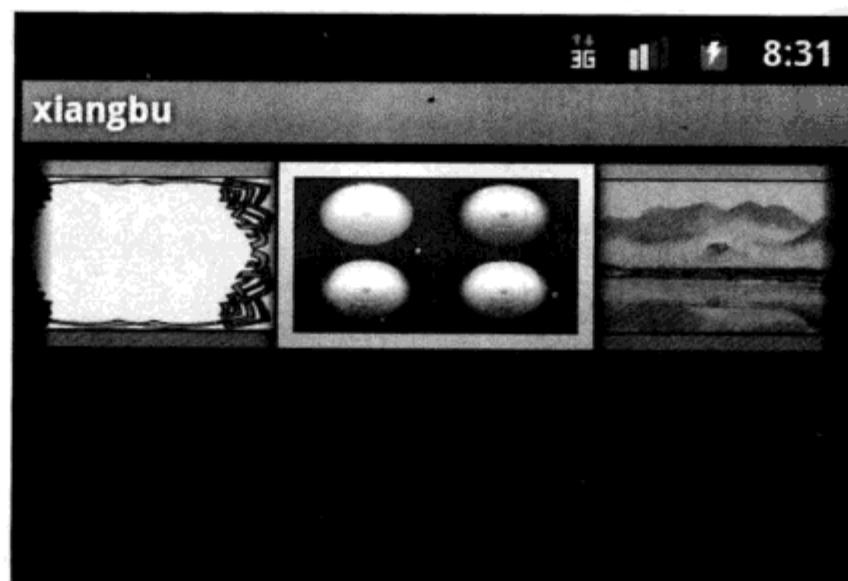


图3-11 执行效果

3.6 开发一个文件搜索程序

我们都知道网页中的搜索功能，经常用百度和谷歌来搜索所需要的学习资料。这里的文件搜索是指，通过输入关键字的方式快速检索需要的文件，如同 Windows 系统“开始”菜单中的搜索功能一样，如图 3-12 所示。



图3-12 Windows系统“开始”菜单中的搜索

在 Android 手机系统中也可以实现文件搜索的功能，此功能是通过 Java I/O 的 API 中提供的 `java.io.File` 对象实现的，只要利用 `File` 对象的方法，并结合 Android 中的对象 `EditText` 和 `TextView` 等就可以实现手机文件的搜索功能。

在本实例中，分别使用了 `EditText`、`Button` 和 `TextView` 三个对象来实现此功能，用户将要搜索的文件名称或关键字输入 `EditText` 中，单击 `Button` 后，程序会在根目录中寻找符合的文件，并将搜索结果显示于 `TextView` 中；如果找不到符合的文件，则显示找不到文件。

本实例的源码保存在【光盘\daima\第3章\sousuo】，实现流程如下所示。

step 01 在文件 `main.xml` 中分别插入 1 个 `TextView` 控件，1 个 `EditText` 控件和 1 个 `Button` 控件，主要代码如下所示。

```
<EditText  
    android:id="@+id/mKeyword"  
    android:layout_width="198px"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
    android:layout_x="110px"  
    android:layout_y="12px"  
>  
</EditText>
```

```

<Button
    android:id="@+id/mButton"
    android:layout_width="86px"
    android:layout_height="48px"
    android:text="@string/str_button"
    android:layout_x="100px"
    android:layout_y="72px"
    >
</Button>
<TextView
    android:id="@+id/mResult"
    android:layout_width="296px"
    android:layout_height="283px"
    android:layout_x="10px"
    android:layout_y="132px"
    android:textColor="@drawable/blue"
    >
</TextView>

```

step 02 编写文件 sousuo.java 中,以 java.io.File 对象来取得根目录下的文件,经过比较后,将符合结果的文件路径写入 TextView 中,若要在 TextView 中换行,需使用 “\n” 换行符实现换行处理。文件 sousuo.java 的主要代码如下所示。

```

public class sousuo extends Activity
{
    /*声明对象变量*/
    private Button mButton;
    private EditText mKeyword;
    private TextView mResult;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);

        /* 初始化对象 */
        mKeyword=(EditText)findViewById(R.id.mKeyword);
        mButton=(Button)findViewById(R.id.mButton);
        mResult=(TextView) findViewById(R.id.mResult);

        /* 将mButton添加onClickListener */
    }
}

```

```
mButton.setOnClickListener(new Button.OnClickListener()
{
    public void onClick(View v)
    {
        /*取得输入的关键字*/
        String keyword = mKeyword.getText().toString();
        if(keyword.equals(""))
        {
            mResult.setText("关键字不能为空!!!");
        }
        else
        {
            mResult.setText(searchFile(keyword));
        }
    }
});

/* 搜索文件的method */
private String searchFile(String keyword)
{
    String result="";
    File[] files=new File("/").listFiles();
    for( File f : files )
    {
        if(f.getName().indexOf(keyword)>=0)
        {
            result+=f.getPath()+"\n";
        }
    }
    if(result.equals("")) result="老大，找不到文件!!";
    return result;
}
}
```

这样，整个实例介绍完毕。执行后的效果如图 3-13 所示，输入搜索关键字，并单击【检索文件】按钮后会在按钮下方显示对应的搜索结果，如图 3-14 所示。

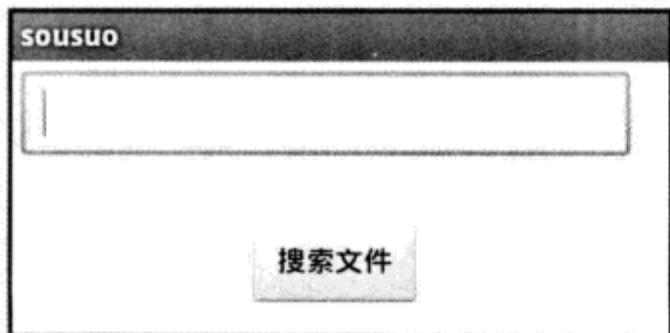


图3-13 执行效果

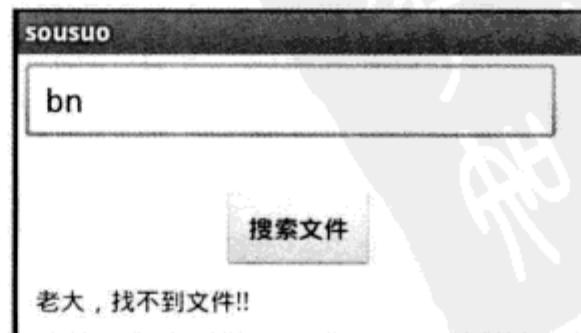


图3-14 搜索效果



3.7 模拟实现一个时钟效果

在Android系统中有一个专门的时钟对象AnalogClock Widget，通过此对象可以在屏幕上实现一个时钟样子的效果。在本实例屏幕的上方显示一个时钟效果界面，并在其下放置一个TextView以显示一个电子时钟效果。

在具体实现上，本实例需要使用如下三个对象。

- ◆ android.os.Handler：通过产生的Thread对象在进程内同步调用方法System.currentTimeMillis()，这样可以取得系统时间；
- ◆ java.lang.Thread：是联系Activity与Thread的桥梁；
- ◆ android.os.Message：使用Message对象通知Handler对象，在收到Message对象后将时间变量的值显示在TextView中，这样就实现了数字时钟功能。

本实例的源码保存在【光盘\daima\第3章\shizhong】，实现流程如下所示。

step 01 编写布局文件main.xml，在里面插入了一个AnalogClock Widget控件，实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/widget27"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
>
    <AnalogClock
        android:id="@+id/myAnalogClock"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
    >
    </AnalogClock>
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textSize="20sp"
        android:textColor="@drawable/white"
        android:layout_gravity="center_horizontal"
    >
    </TextView>
```



```
</LinearLayout>
```

step(02) 编写文件 shizhong.java，在此需要另外加载 Java 对象 Calendar 和 Thread，在响应 onCreate() 时构造这两个对象，并且实现了方法 handleMessage() 和 run()。文件 shizhong.java 的主要代码如下所示。

```
public class shizhong extends Activity
{
    /*声明一常数作为判别信息用*/
    protected static final int GUINOTIFIER = 0x1234;

    /*声明两个widget对象变量*/
    private TextView mTextView;
    public AnalogClock mAnalogClock;

    /*声明与时间相关的变量*/
    public Calendar mCalendar;
    public int mMinutes;
    public int mHour;

    /*声明关键Handler与Thread变量*/
    public Handler mHandler;
    private Thread mClockThread;

    /** Called when the activity is first created. */
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /*通过findViewById取得两个widget对象*/
        mTextView=(TextView)findViewById(R.id.myTextView);
        mAnalogClock=(AnalogClock)findViewById(R.id.myAnalogClock);

        /*通过Handler来接收运行线程所传递的信息并更新TextView*/
        mHandler = new Handler()
        {
            public void handleMessage(Message msg)
            {
                /*这里是处理信息的方法*/
                switch (msg.what)
                {
                    case example13.GUINOTIFIER:
                        /* 在这处理要TextView对象Show时间的事件 */
                }
            }
        };
    }
}
```

```

        mTextView.setText(mHour+" : "+mMinutes);
        break;
    }
    super.handleMessage(msg);
}
};

/*通过运行线程来持续取得系统时间*/
mClockThread=new LooperThread();
mClockThread.start();
}

/*改写一个Thread Class用来持续取得系统时间*/
class LooperThread extends Thread
{
    public void run()
    {
        super.run();
        try
        {
            do
            {
                /*取得系统时间*/
                long time = System.currentTimeMillis();
                /*通过Calendar对象来取得小时与分钟*/
                final Calendar mCalendar = Calendar.getInstance();
                mCalendar.setTimeInMillis(time);
                mHour = mCalendar.get(Calendar.HOUR);
                mMinutes = mCalendar.get(Calendar.MINUTE);

                /*让运行线程休息一秒*/
                Thread.sleep(1000);
                /*重要关键程序：取得时间后发出信息给Handler*/
                Message m = new Message();
                m.what = example13.GUINOTIFIER;
                example13.this.mHandler.sendMessage(m);
            }while(example13LooperThread.interrupted()==false);
            /*当系统发出中断信息时停止本循环*/
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

```
    }  
}
```

执行后会显示一个数字时钟效果，如图 3-15 所示。



图3-15 执行效果

3.8 在手机屏幕中实现进度条效果

在本实例中，通过使用控件 ProgressBar 和 Handler 联合实现了后台进度条效果。首先使用控件 ProgressBar 实现了进度条，然后使用 Handler 访问了新进程 Activity 中的 Widget，并将运行状态在屏幕中显示出来。通过 Handler 对象和 Message 对象，将进程里的状态往外传递，最后由 Activity 的 Handler 事件来取得运行状态。

本实例的源码保存在【光盘\daima\第 3 章\jindu】，实现流程如下所示。

step 01 编写布局文件 main.xml，在里面插入了 ProgressBar 进度条控件，主要代码如下所示。

```
<TextView  
    android:id="@+id/myTextView1"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:textColor="@drawable/blue"  
    android:text="@string/hello"  
/>  
<ProgressBar  
    android:id="@+id/myProgressBar1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:visibility="gone"  
/>  
<Button  
    android:id="@+id/myButton1"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button1" />
```

step 02 编写文件是 jindu.java，主要实现代码如下所示。

```
public class jindu extends Activity
{
    private TextView mTextView01;
    private Button mButton01;
    private ProgressBar mProgressBar01;
    public int intCounter=0;

    /* 自定义Handler信息代码，用以作为识别事件处理 */
    protected static final int GUI_STOP_NOTIFIER = 0x108;
    protected static final int GUI_THREADING_NOTIFIER = 0x109;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mButton01 = (Button) findViewById(R.id.myButton1);
        mTextView01 = (TextView) findViewById(R.id.myTextView1);

        /* 设置ProgressBar widget对象 */
        mProgressBar01 = (ProgressBar) findViewById(R.id.myProgressBar1);

        /* 调用setIndeterminate方法赋值indeterminate模式为false */
        mProgressBar01.setIndeterminate(false);

        /* 当点击按钮后，开始运行线程工作 */
        mButton01.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                // TODO Auto-generated method stub

                /* 点击按钮让ProgressBar显示 */
                mTextView01.setText(R.string.str_progress_start);

                /* 将隐藏的ProgressBar显示出来 */
            }
        });
    }
}
```

```
mProgressBar01.setVisibility(View.VISIBLE);

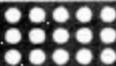
/* 指定Progress为最多100 */
mProgressBar01.setMax(100);

/* 初始Progress为0 */
mProgressBar01.setProgress(0);

/* 起始一个运行线程 */
new Thread(new Runnable()
{
    public void run()
    {
        /* 默认0至9，共运行10次的循环叙述 */
        for (int i=0;i<10;i++)
        {
            try
            {
                /* 成员变量，用以识别加载进度 */
                intCounter = (i+1)*20;
                /* 每运行一次循环，即暂停1秒 */
                Thread.sleep(1000);

                /* 当Thread运行5秒后显示运行结束 */
                if(i==4)
                {
                    /* 以Message对象，传递参数给Handler */
                    Message m = new Message();

                    /* 以what属性指定User自定义 */
                    m.what = jindu.GUI_STOP_NOTIFIER;
                    jindu.this.myMessageHandler.sendMessage(m);
                    break;
                }
                else
                {
                    Message m = new Message();
                    m.what = jindu.GUI_THREADING_NOTIFIER;
                    jindu.this.myMessageHandler.sendMessage(m);
                }
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
        }
    }
});
```



```
        }
    }
}).start();
}
);
/* Handler建构之后，会聆听传来的信息代码 */
Handler myMessageHandler = new Handler()
{
    // @Override
    public void handleMessage(Message msg)
    {
        switch (msg.what)
        {
            /* 当取得识别为 离开运行线程时所取得的信息 */
            case jindu.GUI_STOP_NOTIFIER:
                /* 显示运行终了 */
                mTextView01.setText(R.string.str_progress_done);

                /* 设置ProgressBar Widget为隐藏 */
                mProgressBar01.setVisibility(View.GONE);
                Thread.currentThread().interrupt();
                break;

            /* 当取得识别为 持续在运行线程当中时所取得的信息 */
            case jindu.GUI_THREADING_NOTIFIER:
                if(!Thread.currentThread().isInterrupted())
                {
                    mProgressBar01.setProgress(intCounter);
                    /* 将显示进度显示于TextView当中 */
                    mTextView01.setText
                    (
                        getResources().getText(R.string.str_progress_start) +
                        "("+Integer.toString(intCounter)+"%)\n" +
                        "Progress:"+
                        Integer.toString(mProgressBar01.getProgress()) +
                        "\n"+ "Indeterminate:"+
                        Boolean.toString(mProgressBar01.isIndeterminate())
                    );
                }
                break;
        }
        super.handleMessage(msg);
    }
}
```

盗版
必究



```
        }
    };
}
```

在本实例中，在屏幕中设计了一个按钮，单击此按钮后会显示 ProgressBar 进度条。在默认的布局文件 main.xml 中，并没有指定它的 indeterminate 属性，所以即使在程序中调用了 ProgressBar 的 setIndeterminate() 方法，还是无法改变 ProgressBar.getProgress 的值，这个值永远都是 0。所以在上述代码中使用了循环动画来作为运行过程中的显示素材，并使用了一个 counter 递增整数来表示运行进度的百分比。

执行后会显示一个按钮界面，用户单击【按我】按钮后会显示一个提示的进度条，如图 3-16 所示。

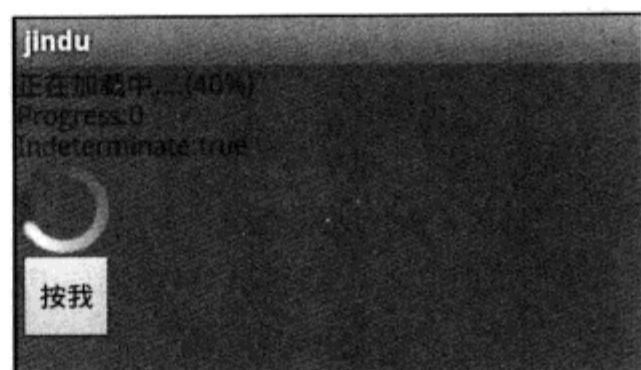


图3-16 执行效果

3.9 开发一个自动选择日期和时间的程序

在 Android 的 API 中，提供了两个十分重要的对象，分别是 DatePicker 和 TimePicker，通过它们可以实现动态输入日期和时间。在本实例中，使用了 DatePicker、TimePicker 和 TextView 三种对象，其中用 TextView 来显示日期与时间，默认带入目前系统的日期与时间，用 DatePicker 与 TimePicker 让用户动态调整日期与时间。当用户调整了 DatePicker 日期或 TimePicker 时间时，在 TextView 中显示的日期与时间也会随之改变。

本实例的源码保存在【光盘\daima\第 3 章\ri】，实现流程如下所示。

step 01 编写布局文件 main.xml，在里面分别插入了 DatePicker 日期控件和 TimePicker 时间控件，主要代码如下所示。

```
<AbsoluteLayout
    android:id="@+id/layout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <DatePicker
        android:id="@+id/dPicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="20px"
        android:layout_y="20px"
    >
```

```
</DatePicker>
<TimePicker
    android:id="@+id/tPicker"
    android:layout_width="220px"
    android:layout_height="wrap_content"
    android:layout_x="20px"
    android:layout_y="180px"
    >
</TimePicker>
<TextView
    android:id="@+id/showTime"
    android:layout_width="wrap_content"
    android:layout_height="34px"
    android:textSize="24sp"
    android:layout_x="30px"
    android:layout_y="330px"
    android:textColor="@drawable/black"
    >
</TextView>
</AbsoluteLayout>
```

step 02 编写文件是 ri.java，在文件使用 updateDisplay() 方法来设置在 TextView 中所显示的日期时间，使用 java.util.Calendar 对象来获取当前系统的时间，并将这个时间传入到 TextView 中。如果用户改变了 DatePicker 中的年、月、日时，会运行如下两类操作。

- ◆ 触发 DatePicker 的 onDateChange() 事件，并同时运行方法 updateDisplay() 来重新设置在 TextView 中显示的日期。
- ◆ 触发 TimePicker 的 onTimeChange() 事件，运行方法 updateDisplay() 来重新设置在 TextView 中显示的时间。

文件 ri.java 的主要代码如下所示。

```
public class ri extends Activity
{
    /*声明日期及时间变量*/
    private int mYear;
    private int mMonth;
    private int mDay;
    private int mHour;
    private int mMinute;
    /*声明对象变量*/
    TextView tv;
    TimePicker tp;
    DatePicker dp;
```



```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    /*取得目前日期与时间*/
    Calendar c=Calendar.getInstance();
    mYear=c.get(Calendar.YEAR);
    mMonth=c.get(Calendar.MONTH);
    mDay=c.get(Calendar.DAY_OF_MONTH);
    mHour=c.get(Calendar.HOUR_OF_DAY);
    mMinute=c.get(Calendar.MINUTE);

    super.onCreate(savedInstanceState);
    /* 载入main.xml Layout */
    setContentView(R.layout.main);

    /*取得TextView对象，并调用updateDisplay()
     来设置显示的初始日期时间*/
    tv= (TextView) findViewById(R.id.showTime);
    updateDisplay();

    /*取得DatePicker对象，以init()
     设置初始值与onDateChangeListener() */
    dp= (DatePicker) findViewById(R.id.dPicker);
    dp.init(mYear,mMonth,mDay,new DatePicker.OnDateChangedListener()
    {
        @Override
        public void onDateChanged(DatePicker view,int year,
                                  int monthOfYear,int dayOfMonth)
        {
            mYear=year;
            mMonth= monthOfYear;
            mDay=dayOfMonth;
            /*调用updateDisplay()来改变显示日期*/
            updateDisplay();
        }
    });
    /*取得TimePicker对象，并设置为24小时制显示*/
    tp= (TimePicker) findViewById(R.id.tPicker);
    tp.setIs24HourView(true);

    /*setOnTimeChangedListener，并覆盖onTimeChanged event*/
    tp.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener()
    {
```

```

@Override
public void onTimeChanged(TimePicker view,
                           int hourOfDay,
                           int minute)
{
    mHour=hourOfDay;
    mMinute=minute;
    /*调用updateDisplay()来改变显示时间*/
    updateDisplay();
}
});

/*设置显示日期时间的方法*/
private void updateDisplay()
{
    tv.setText(
        new StringBuilder().append(mYear).append("/")
            .append(format(mMonth + 1)).append("/")
            .append(format(mDay)).append(" ")
            .append(format(mHour)).append(":")
            .append(format(mMinute))
    );
}

/*日期时间显示两位数的方法*/
private String format(int x)
{
    String s="" +x;
    if(s.length()==1) s="0"+s;
    return s;
}
}

```

执行后会显示一个数字时钟效果，我们可以通过单击+和-来自动选择日期和时间。效果如图 3-17 所示。

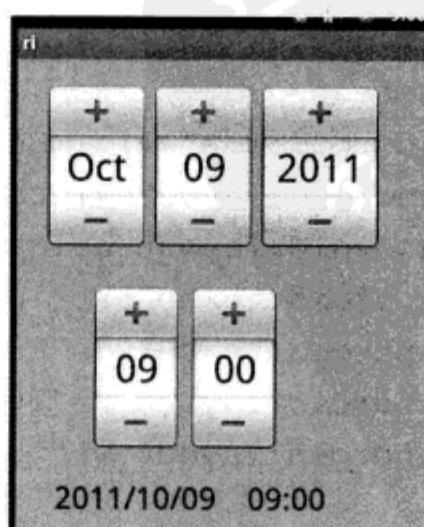


图3-17 执行效果



3.10 在收集屏幕上显示磁盘中的图片

在 Android 项目中，图片文件的默认保存目录是“res\drawable”，这样可以通过 R.drawable.id 来获取图片的 id，我们可以在程序中任意使用这幅图片。但是如果将此图片保存在“res\drawable” 目录下，例如仅仅想从存储卡中获取一副图片，此时需要 Android API 中的 Bitmap 对象来实现。

在本实例中，将使用 decodeFile 方法来加载手机磁盘中图片文件，并显示在手机屏幕中。本实例的源码保存在【光盘\daima\第 3 章\xiantu】，实现流程如下所示。

具体实现

step 01 编写文件 main.xml，在屏幕中分别插入 1 个 TextView 控件、1 个 ImageView 控件和 1 个 Button 控件。具体代码如下所示。

```
<TextView  
    android:id="@+id/mTextView"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/str_text"  
    android:layout_x="30px"  
    android:layout_y="30px"  
    android:textColor="@drawable/blue"  
>  
</TextView>  
<ImageView  
    android:id="@+id/mImageView"  
    android:layout_width="250px"  
    android:layout_height="188px"  
    android:src="@drawable/ex04_22_1"  
    android:layout_x="30px"  
    android:layout_y="62px"  
>  
</ImageView>  
<Button  
    android:id="@+id/mButton"  
    android:layout_width="155px"  
    android:layout_height="wrap_content"  
    android:text="@string/str_button"  
    android:textSize="18sp"  
    android:layout_x="80px"  
    android:layout_y="302px"  
>  
</Button>
```

step 02 编写文件xiantu.java，在文件中使用decodeFile(fileName)方法获取Bitmap对象，然后使用setImageBitmap()来设置此Bitmap对象的显示。文件xiantu.java的主要代码如下所示。

```

public class xiantu extends Activity
{
    /*声明对象变量*/
    private ImageView mImageView;
    private Button mButton;
    private TextView mTextView;
    private String fileName="/data/123.png";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);

        /* 取得Button对象，并添加onClickListener */
        mButton = (Button)findViewById(R.id.mButton);
        mButton.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                /* 取得对象 */
                mImageView = (ImageView)findViewById(R.id.mImageView);
                mTextView=(TextView)findViewById(R.id.mTextView);
                /* 检查文件是否存在 */
                File f=new File(fileName);
                if(f.exists())
                {
                    /* 产生Bitmap对象，并放入mImageView中 */
                    Bitmap bm = BitmapFactory.decodeFile(fileName);
                    mImageView.setImageBitmap(bm);
                    mTextView.setText(fileName);
                }
                else
                {
                    mTextView.setText("文件不存在");
                }
            }
        });
    }
}

```

执行后将显示加载指定的图片并显示在屏幕当中，如图 3-18 所示。

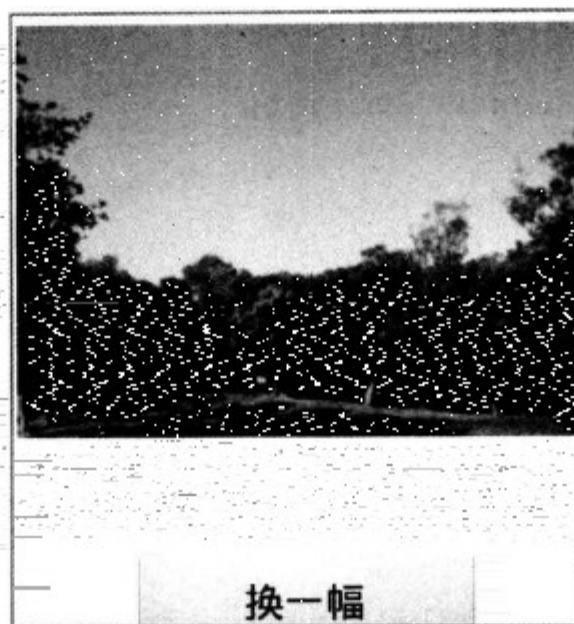


图3-18 执行效果

3.11 触动Menu菜单控件

在 Android 系统中，可以使用控件 Menu 为用户提供一个友好的界面显示效果。大部分手机应用程序包括如下两种人机互动方式。

- ◆ 第一种：直接通过 GUI 的 Views，这种可以满足大部分的交互操作；
- ◆ 第二种：应用 Menu，当按下 Menu 按钮后，会弹出与当前活动状态下的应用程序相匹配的菜单。

上述两种方式都有各自的优势，而且可以相辅相成，即便用户可以由主界面完成大部分操作，但是适当地拓展 Menu 功能可以更加完善应用程序，至少用户可以通过排列整齐的按钮清晰地了解当前模式下可以使用的功能。

本实例的源码保存在【光盘\daima\第 3 章\mm】，实现流程如下所示。

step 01 编写布局文件 main.xml 主文件，在里面分别插入了 1 个 TextView 控件和 2 个 Button 控件。其中 TextView 控件用于显示文本，并用 “layout_width” 设置了 Button 的宽度，用 “layout_height” 设置了 Button 的高度。通过符号 @ 来设置了读取变量值并进行替换。文件 main.xml 的主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/hello" />
    <Button android:id="@+id/button1"
```

```

        android:layout_width="100px"
        android:layout_height="wrap_content" android:text="@string/
button1" />
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/
button2" />
</LinearLayout>

```

- ◆ android:text=" @string/button1"：相当于 <string name="button1">button1</string>
- ◆ android:text="@string/button2"：相当于 <string name="button2">button2</string>

上面的符号 @ 十分重要，用于提示 XML 文件解析器解析 @ 后面的名字，例如上面的 "@string/button1"，解析器会从文件 "values/string.xml" 中读取 Button1 这个变量值。

文件 string.xml 中定义了 TextView 和 Button 的值，具体代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ActivityMenu</string>
    <string name="app_name">HelloMenu</string>
    <string name="button1">button1</string>
    <string name="button2">button2</string>
</resources>

```

step 02 编写文件 mm.java，首先定义函数 onCreate 来显示 main.xml 设置的布局，并设置 2 个 Button 为不可见状态；然后定义函数 onCreateOptionsMenu 来生成 menu，此函数是一个回调方法，只有当按下手机设备上的 menu 按钮后，Android 才会生成一个包含 2 个子项的菜单。在具体实现上，将首先得到 super 函数调用后的返回值，并在 onCreateOptionsMenu 的最后返回；然后调用 menu.add 给 menu 添加 1 个项；最后定义函数 onOptionsItemSelected，只有当按下手机设备上的 Menu 按钮后，Android 才会调用执行。而这个事件就是单击菜单里的某一项，即 MenuItem。

文件 Mm.java 的主要代码如下所示。

```

public class ActivityMenu extends Activity {
    public static final int ITEM0 = Menu.FIRST;
    public static final int ITEM1 = Menu.FIRST + 1;
    Button button1;
    Button button2;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button1 = (Button) findViewById(R.id.button1);

```

```
button2 = (Button) findViewById(R.id.button2);
/* 设置2个button不可见 */
button1.setVisibility(View.INVISIBLE);
button2.setVisibility(View.INVISIBLE);
}

@Override
/*
 * menu.findItem(EXIT_ID);找到特定的MenuItem
 * MenuItem.setIcon.可以设置menu按钮的背景
 */
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, ITEM0, 0, "现在显示按钮1");
    menu.add(0, ITEM1, 0, "现在显示按钮2");
    menu.findItem(ITEM1);
    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case ITEM0:
            actionClickMenuItem1();
            break;
        case ITEM1:
            actionClickMenuItem2(); break;
    }
    return super.onOptionsItemSelected(item);
}

/*
 * 点击第一个menu的第一个按钮执行的动作
 */
private void actionClickMenuItem1() {
    setTitle("button1 可见");
    button1.setVisibility(View.VISIBLE);
    button2.setVisibility(View.INVISIBLE);
}

/*
 * 点击第二个menu的第一个按钮执行的动作
 */
private void actionClickMenuItem2() {
    setTitle("可以看到button2");
    button1.setVisibility(View.INVISIBLE);
    button2.setVisibility(View.VISIBLE);
}
}
```



执行后的效果如图 3-19 所示；当单击模拟器上的“Menu”键后会触发程序，并在屏幕上显示预先设置已经隐藏的 2 个按钮，如图 3-20 所示。



图3-19 初始效果



图3-20 触发设备后的效果

3.12 使用SimpleAdapter实现ListView组件的效果

在Android系统中，组件ListView经常被用到，它可以在屏幕内实现列表显示一些信息。Android中的ListView控件可以通过Adapter适配器来构建显示功能。在Android系统中可以使用3种Adapter，分别是：ArrayAdapter、SimpleAdapter和CursorAdapter。在本实例中，将使用SimpleAdapter来实现ListView效果。本实例的源码保存在【光盘\daima\第3章\lie】，实现流程如下所示。

step 01 构建List列表，设置用Map来实现列表中的每一项。然后编码创建TestList（测试列表）类继承Activity。具体代码如下所示。

```
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayList<HashMap<String, Object>> users = new
ArrayList<HashMap<String, Object>>();
        for (int i = 0; i < 10; i++) {
            HashMap<String, Object> user = new HashMap<String,
Object>();
            user.put("img", R.drawable.user);
            user.put("username", "姓名(" + i + ")");
            user.put("age", (20 + i) + "");
        }
    }
```



```
        users.add(user);
    }

    SimpleAdapter saImageItems = new SimpleAdapter(this,
        users, // 数据来源
        R.layout.user, // 每一个user xml 相当ListView的一个组件
        new String[] { "img", "username", "age" },
        // 分别对应view 的id
        new int[] { R.id.img, R.id.name, R.id.age });
    // 获取listview
    ((ListView) findViewById(R.id.users)).setAdapter(saImageItems);
```

step 02 编写文件 main.xml 实现布局，在里面插入 3 个 TextView，其中 ListView 前面是标题行， ListView 相当于用来显示数据的容器，里面每行是一个用户信息。具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:text="工资资料" android:gravity="center"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:background="#DAA520"
        android:textColor="#000000">
    </TextView>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:text="名字"
            android:gravity="center" android:layout_width="160px"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:background="#7CFC00">
        </TextView>
        <TextView android:text="薪水"
            android:layout_width="170px" android:gravity="center"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:background="#F0E68C">
        </TextView>
    </LinearLayout>
    <ListView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/users">
    </ListView>
</LinearLayout>
```

step 03 编写文件 use.xml，用于定义用户信息布局。在文件中设置每行包含了1个 img 图片和2段 TextView 文本信息，这个文件以参数的形式通过 Adapter 在 ListView 中显示。主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:layout_width="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    >
    <TableRow >
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/img">
        </ImageView>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="150px"
            android:id="@+id/name">
        </TextView>
        <TextView
            android:layout_height="wrap_content"
            android:layout_width="170px"
            android:id="@+id/age">
        </TextView>
    </TableRow>
</TableLayout>
```

step 04 编写文件 lie.java，在文件中使用 for 循环语句设置姓名是 i 递增，薪水是 i*1000 递增，并且使用 SimpleAdapter 显示每块区域的用户信息。主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ArrayList<HashMap<String, Object>> users = new
    ArrayList<HashMap<String, Object>>();
    for (int i = 0; i < 10; i++) {
        HashMap<String, Object> user = new HashMap<String, Object>();
        user.put("img", R.drawable.user);
        user.put("username", "姓名(" + i + ")");
        user.put("age", (1000 * i) + "");
        users.add(user);
    }
    SimpleAdapter saImageItems = new SimpleAdapter(this,
```



```
users,          // 数据来源
R.layout.user,  // 每一个user xml 相当ListView的一个组件
new String[] { "img", "username", "age" },
// 分别对应view 的id
new int[] { R.id.img, R.id.name, R.id.age } );
```

执行后的效果如图 3-21 所示。

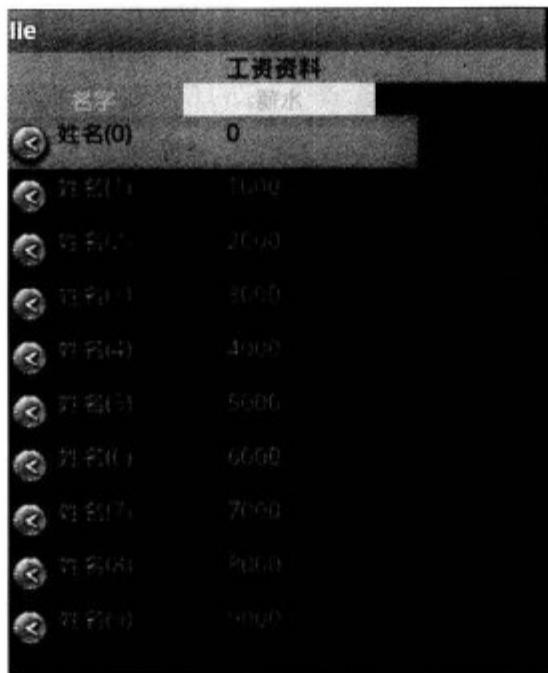


图3-21 运行效果

3.13 在屏幕中实现抽屉样式效果

抽屉效果在当前的触摸手机中大行其道，绚丽的效果和动感的视觉冲击给我们的生活带来了绚丽的色彩。在 Android 开发过程中，我们喜欢使用特效，比如抽屉效果，这样可以给人很好的体验。点击一个按钮，就像拉抽屉一样展开界面，这样的效果正是我在这里所要说明的。比如在 AVD 或真机上，我们都看过这种效果。比较常用的应用是 LAUNCH 应用，在本实例中实现了一个基本的拉抽屉效果。本实例的源代码保存在【光盘\daima\第三章\chouti】。

step 01 编写主布局文件 main.xml，主要代码如下所示。

```
<?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:id="@+id/container">
    <GridView android:id="@+id/gridview" android:layout_width="fill_parent"
        android:layout_height="fill_parent" android:numColumns="auto_fit"
        android:verticalSpacing="10dp" android:gravity="center"
        android:columnWidth="50dip" android:horizontalSpacing="10dip" />
</LinearLayout>
```



step 02 编写 GridView 的布局文件 Item.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content" android:paddingBottom="4dip"
    android:layout_width="fill_parent">
    <ImageView android:layout_height="wrap_content" android:id="@+id/
ItemImage"
        android:layout_width="wrap_content" android:layout_
        centerHorizontal="true">
        </ImageView>
        <TextView android:layout_width="wrap_content"
            android:layout_below="@+id/ItemImage" android:layout_
            height="wrap_content"
            android:text="TextView01" android:layout_centerHorizontal="true"
            android:id="@+id/ItemText">
        </TextView>
    </RelativeLayout>
```

step 03 编写核心功能文件 Panel.java，此文件是抽屉组件的源码，这个抽屉只实现了从右往左的弹出 / 从左往右的收缩，读者可以根据自己的需要修改源码来改变抽屉动作的方向。

```
public class Panel extends LinearLayout{

    public interface PanelClosedEvent {
        void onPanelClosed(View panel);
    }

    public interface PanelOpenedEvent {
        void onPanelOpened(View panel);
    }

    /**Handle的宽度，与Panel等高*/
    private final static int HANDLE_WIDTH=30;
    /**每次自动展开/收缩的范围*/
    private final static int MOVE_WIDTH=20;
    private Button btnHandle;
    private LinearLayout panelContainer;
    private int mRightMargin=0;
    private Context mContext;
    private PanelClosedEvent panelClosedEvent=null;
    private PanelOpenedEvent panelOpenedEvent=null;

    /**
     * otherView自动布局以适应Panel展开/收缩的空间变化
     * @author GV
```

```
* 
*/
public Panel(Context context, View otherView, int width, int height)
{
    super(context);
    this.mContext=context;

    //改变Panel附近组件的属性
    LayoutParams otherLP=(LayoutParams) otherView.getLayoutParams();
    otherLP.weight=1;//支持挤压
    otherView.setLayoutParams(otherLP);

    //设置Panel本身的属性
    LayoutParams lp=new LayoutParams(width, height);
    lp.rightMargin=-lp.width+HANDLE_WIDTH;//Panel的Container在屏幕不可视区域，Handle在可视区域
    mRightMargin=Math.abs(lp.rightMargin);
    this.setLayoutParams(lp);
    this.setOrientation(LinearLayout.HORIZONTAL);

    //设置Handle的属性
    btnHandle=new Button(context);
    btnHandle.setLayoutParams(new LayoutParams(HANDLE_WIDTH,height));
    btnHandle.setOnClickListener(new OnClickListener(){

        @Override
        public void onClick(View arg0) {
            LayoutParams lp = (LayoutParams) Panel.this.
getLayoutParams();
            if (lp.rightMargin < 0)// CLOSE的状态
                new AsynMove().execute(new Integer[] { MOVE_WIDTH });
            // 正数展开
            else if (lp.rightMargin >= 0)// OPEN的状态
                new AsynMove().execute(new Integer[] { -MOVE_
WIDTH });
            // 负数收缩
        }

    });

    //btnHandle.setOnTouchListener(HandleTouchEvent);
    this.addView(btnHandle);

    //设置Container的属性
    panelContainer=new LinearLayout(context);
    panelContainer.setLayoutParams(new LayoutParams(LayoutParams.
```



```
FILL_PARENT,
        LayoutParams.FILL_PARENT));
    this.addView(panelContainer);
}

/**
 * 定义收缩时的回调函数
 * @param event
 */
public void setPanelClosedEvent(PanelClosedEvent event)
{
    this.panelClosedEvent=event;
}

/**
 * 定义展开时的回调函数
 * @param event
 */
public void setPanelOpenedEvent(PanelOpenedEvent event)
{
    this.panelOpenedEvent=event;
}

/**
 * 把View放在Panel的Container
 * @param v
 */
public void fillPanelContainer(View v)
{
    panelContainer.addView(v);
}

/**
 * 异步移动Panel
 * @author hellogv
 */
class AsynMove extends AsyncTask<Integer, Integer, Void> {

    @Override
    protected Void doInBackground(Integer... params) {
        int times;
        if (mRightMargin % Math.abs(params[0]) == 0)// 整除
            times = mRightMargin / Math.abs(params[0]);
        else
            // 有余数
    }
}
```

盗
书
网
站
www.
pdfto
cn



```
times = mRightMargin / Math.abs(params[0]) + 1;

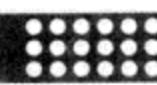
for (int i = 0; i < times; i++) {
    publishProgress(params);
    try {
        Thread.sleep(Math.abs(params[0]));
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
return null;
}

@Override
protected void onProgressUpdate(Integer... params) {
    LayoutParams lp = (LayoutParams) Panel.this.getLayoutParams();
    if (params[0] < 0)
        lp.rightMargin = Math.max(lp.rightMargin + params[0],
                                  (-mRightMargin));
    else
        lp.rightMargin = Math.min(lp.rightMargin + params[0],
                                 0);

    if(lp.rightMargin==0 && panelOpenedEvent!=null){//展开之后
        panelOpenedEvent.onPanelOpened(Panel.this); //调用OPEN
    }
    //收缩之后
    else if(lp.rightMargin==-(mRightMargin) &&
panelClosedEvent!=null){
        panelClosedEvent.onPanelClosed(Panel.this); //调用
    }
    Panel.this.setLayoutParams(lp);
}
}
```

step 04 编写文件 main.java，此文件是整个实例的主控部分，用于演示使用 Panel 实现抽屉效果。文件 main.java 的实现代码如下所示。

```
public class main extends Activity {
    public Panel panel;
    public LinearLayout container;
```



```
public GridView gridview;
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    this.setTitle("“可动态布局”的抽屉组件之构建基础-----hellologv");
    gridview = (GridView) findViewById(R.id.gridView);
    container=(LinearLayout)findViewById(R.id.container);
    panel=new Panel(this,gridview,200,LayoutParams.FILL_PARENT);
    container.addView(panel);//加入Panel控件

    //新建测试组件
    TextView tvTest=new TextView(this);
    tvTest.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));
    tvTest.setText("测试组件，红字白底");
    tvTest.setTextColor(Color.RED);
    tvTest.setBackgroundColor(Color.WHITE);
    //加入到Panel里面
    panel.fillPanelContainer(tvTest);

    panel.setPanelClosedEvent(panelClosedEvent);
    panel.setPanelOpenedEvent(panelOpenedEvent);

    //往GridView填充测试数据
    ArrayList<HashMap<String, Object>> lstImageItem = new
ArrayList<HashMap<String, Object>>();
    for (int i = 0; i < 100; i++) {
        HashMap<String, Object> map = new HashMap<String, Object>();
        map.put("ItemImage", R.drawable.icon);
        map.put("ItemText", "NO." + String.valueOf(i));
        lstImageItem.add(map);
    }

    SimpleAdapter saImageItems = new SimpleAdapter(this,
        lstImageItem,
        R.layout.item,
        new String[] { "ItemImage", "ItemText" },
        new int[] { R.id.ItemImage, R.id.ItemText });
    gridview.setAdapter(saImageItems);
    gridview.setOnItemClickListener(new ItemClickListener());
}

PanelClosedEvent panelClosedEvent =new PanelClosedEvent() {
```

```
    @Override
    public void onPanelClosed(View panel) {
        Log.e("panelClosedEvent", "panelClosedEvent");
    }

};

PanelOpenedEvent panelOpenedEvent =new PanelOpenedEvent(){
    @Override
    public void onPanelOpened(View panel) {
        Log.e("panelOpenedEvent", "panelOpenedEvent");
    }

};

class ItemClickListener implements OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> arg0,View arg1, int arg2, long arg3) {
        @SuppressWarnings("unchecked")
        HashMap<String, Object> item = (HashMap<String, Object>) arg0
            .getItemAtPosition(arg2);
        setTitle((String) item.get("ItemText"));
    }
}
```

此时就在屏幕上实现了一个简单的抽屉效果，如图 3-22 所示。

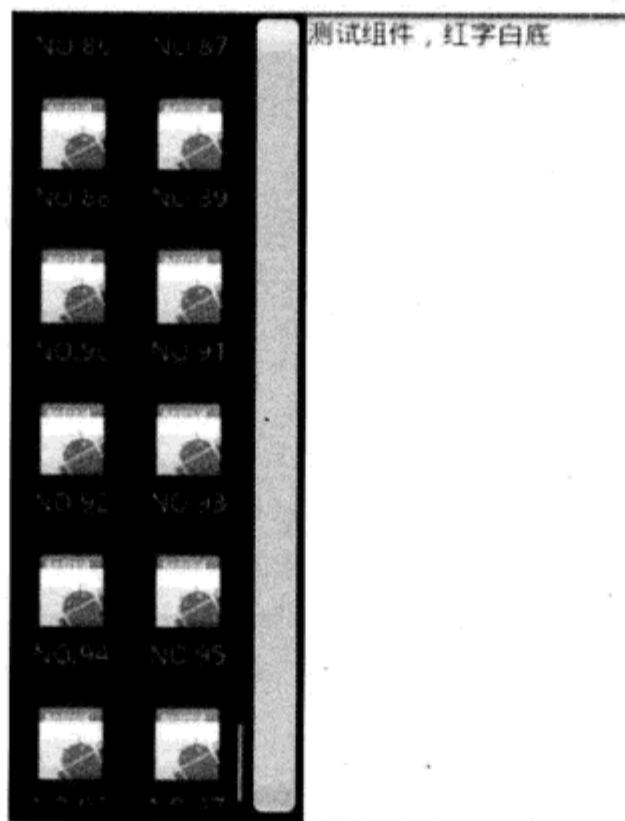


图3-22 执行效果

3.14 使用Toast和Notification实现提醒效果

我们知道在Android系统中可以使用Dialog、Toast和Notification实现提醒功能，在本实例中将演示联合使用Toast和Notification实现提醒功能的过程。本实例重点讲解Notification的用法，Notification是一个和提醒有关的应用，它通常和NotificationManager一块来设置通知。设置通知的方法比较简单，最基本的方式就是新建一个Notification对象，然后设置好通知的各项参数，最后使用系统后台运行的NotificationManager服务将通知发出来。

本实例的源码保存在【光盘\daima\第3章\lianhe】，实现流程如下所示。

step 01 编写文件 main.xml 主文件，此文件是一个布局文件，功能是在屏幕中插入2个 Button 按钮。主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="何谓
        Notification" />
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="何谓Toast" />
</LinearLayout>
```

通过上述代码，执行后效果如图 3-23 所示。

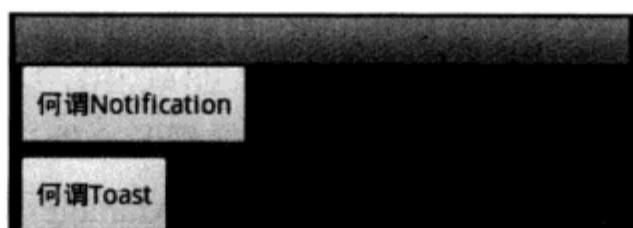


图3-23 插入2个Button

step 02 编写文件 ActivityMain.java，对两个 Button 按钮绑定了单击监听器 OnClickListener，当单击这2个 Button 按钮时，会跳转到新的 Activity 上面。主要代码如下所示。

```
public class ActivityMain extends Activity {
    OnClickListener listener1 = null;
    OnClickListener listener2 = null;
```



```
Button button1;
Button button2;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    listener1 = new OnClickListener() {
        public void onClick(View v) {
            setTitle("使用Notification");
            Intent intent = new Intent(ActivityMain.this,
                MainNotification.class);
            startActivity(intent);
        }
    };
    listener2 = new OnClickListener() {
        public void onClick(View v) {
            setTitle("使用Toast");
            Intent intent = new Intent(ActivityMain.this,
                ActivityToast.class);
            startActivity(intent);
        }
    };
    setContentView(R.layout.main);
    button1 = (Button) findViewById(R.id.button1);
    button1.setOnClickListener(listener1);
    button2 = (Button) findViewById(R.id.button2);
    button2.setOnClickListener(listener2);
}
```

step 03 编写单击第一个 Button 按钮的处理程序，即单击图 3-23 中的【何谓 Notification】按钮后，执行文件 MainNotification.java。在实现此文件过程中需要特别注意如下 4 点。

- ◆ 因为所有的 Notification 都是通过 NotificationManager 来管理的，所以需要先创建 NotificationManager 对象实例来管理这个 Activity 中信息。实现代码如下。

```
mNotificationManager=(NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

- ◆ 函数 setWeather 的功能是，实例化一个 Notification 并将结果显示出来，此函数是 MainNotification 中的重要函数之一，
- ◆ 需要注意如下代码中的三个参数。

```
Notification notification = new Notification(drawable, tickerText,
    System.currentTimeMillis());
```



- 第一个：要显示的图片的 ID。
- 第二个：显示的文本文字。
- 第三个：Notification 显示的时间，一般是立即显示，时间就是 System.currentTimeMillis()。
- ◆ 函数 setDefault 非常重要，此函数初始化了一个 Notification，在设置 Notification 时使用了如下默认值的形式。

```
notification.defaults = defaults;
```

另外，在文件 MainNotification.java 中还用到了以下几种表现形式。

- Notification.DEFAULT_VIBRATE：表示当前的 Notification 显示出来时手机会发出震动。
- Notification.DEFAULT_SOUND：表示当前的 Notification 显示出来时手机会伴随音乐。
- Notification.DEFAULT_ALL：表示当前的 Notification 显示出来时手机即会震动，也会伴随音乐。

文件 MainNotification.java 的主要代码如下所示。

```
public class MainNotification extends Activity {  
    private static int NOTIFICATIONS_ID = R.layout.activity_notification;  
    private NotificationManager mNotificationManager;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_notification);  
  
        Button button;  
  
        mNotificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);  
  
        button = (Button) findViewById(R.id.sun_1);  
        button.setOnClickListener(new Button.OnClickListener() {  
            public void onClick(View v) {  
                setWeather("很好", "天气好", "很好", R.drawable.sun);  
            }  
        });  
  
        button = (Button) findViewById(R.id.cloudy_1);  
        button.setOnClickListener(new Button.OnClickListener() {  
            public void onClick(View v) {  
                setWeather("一般", "天气", "一般", R.drawable.cloudy);  
            }  
        });  
    }  
}
```

```
button = (Button) findViewById(R.id.rain_1);
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        setWeather("不好", "天气", "不好", R.drawable.rain);
    }
});

button = (Button) findViewById(R.id.defaultSound);
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        setDefault(Notification.DEFAULT_SOUND);
    }
});

button = (Button) findViewById(R.id.defaultVibrate);
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        setDefault(Notification.DEFAULT_VIBRATE);
    }
});

button = (Button) findViewById(R.id.defaultAll);
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        setDefault(Notification.DEFAULT_ALL);
    }
});

button = (Button) findViewById(R.id.clear);
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        mNotificationManager.cancel(NOTIFICATIONS_ID);
    }
});

private void setWeather(String tickerText, String title, String content,
        int drawable) {

    Notification notification = new Notification(drawable, tickerText,
            System.currentTimeMillis());
    PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
            new Intent(this, ActivityMain.class), 0);
    notification.setLatestEventInfo(this, title, content,
            contentIntent);
    mNotificationManager.notify(NOTIFICATIONS_ID, notification);
}
```



```

    }

    private void setDefault(int defaults) {
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
            new Intent(this, ActivityMain.class), 0);
        String title = "天气";
        String content = "晴";
        final Notification notification = new Notification(R.drawable.sun,
            content, System.currentTimeMillis());
        notification.setLatestEventInfo(this, title, content,
        contentIntent);
        notification.defaults = defaults;
        mNotificationManager.notify(NOTIFICATIONS_ID, notification);
    }
}

```

step 04 单击第一个 Button 按钮后会来到一个新界面，新界面的布局文件是由 activity_notification.xml 实现的，主要代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
            <Button
                android:id="@+id/sun_1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="适合" />
            <Button
                android:id="@+id/cloudy_1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="不太适" />
            <Button
                android:id="@+id/rain_1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="一点也不适合" />
        
```

```
</LinearLayout>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dip"
    android:text="高级notification" />
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <Button
        android:id="@+id/defaultSound"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="有声音" />
    <Button
        android:id="@+id/defaultVibrate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="振动" />
    <Button
        android:id="@+id/defaultAll"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="声音+振动" />
</LinearLayout>
<Button android:id="@+id/clear"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dip"
    android:text="清除提示" />
</LinearLayout>
</ScrollView>
```

执行后新界面的效果如图 3-24 所示。

例如单击【有声音】按钮后，会发出有声音的 Notification 提示。

step 05 编写第二个Button按钮的处理代码，即单击图3-23中的【何谓Toast】按钮后，执行文件ActivityToast.java，此文件是通过Toast实现的，它不需要用NotificationManager来管理。文件ActivityToast.java的主要代码如下所示。



图3-24 运行效果

```
public class ActivityToast extends Activity {  
  
    OnClickListener listener1 = null;  
    OnClickListener listener2 = null;  
    Button button1;  
    Button button2;  
    private static int NOTIFICATIONS_ID = R.layout.activity_toast;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        listener1 = new OnClickListener() {  
            public void onClick(View v) {  
                setTitle("短暂的提醒");  
                showToast(Toast.LENGTH_SHORT);  
            }  
        };  
        listener2 = new OnClickListener() {  
            public void onClick(View v) {  
                setTitle("长久的提醒");  
                showToast(Toast.LENGTH_LONG);  
                showNotification();  
            }  
        };  
        setContentView(R.layout.activity_toast);  
        button1 = (Button) findViewById(R.id.button1);  
        button1.setOnClickListener(listener1);  
        button2 = (Button) findViewById(R.id.button2);  
        button2.setOnClickListener(listener2);  
    }  
  
    protected void showToast(int type) {  
  
        View view = inflateView(R.layout.toast);  
  
        TextView tv = (TextView) view.findViewById(R.id.content);  
        tv.setText("欢迎来到伟大的首都北京!");  
  
        Toast toast = new Toast(this);  
        toast.setView(view);  
        toast.setDuration(type);  
        toast.show();  
    }  
}
```



```
private View inflateView(int resource) {
    LayoutInflator vi = (LayoutInflator) getSystemService(Context.
LAYOUT_INFLATER_SERVICE);
    return vi.inflate(resource, null);
}

protected void showNotification() {

    NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);

    CharSequence title = "首都";
    CharSequence contents = "北京";

    PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
        new Intent(this, ActivityMain.class), 0);

    Notification notification = new Notification(R.drawable.default_icon,
        title, System.currentTimeMillis());

    notification.setLatestEventInfo(this, title, contents, contentIntent);

    // 100ms延迟后，振动250ms，停止100ms后振动500ms
    notification.vibrate = new long[] { 100, 250, 100, 500 };

    notificationManager.notify(NOTIFICATIONS_ID, notification);
}
}
```

此文件的处理流程如下所示。

- ◆ 实例化 Toast，每个 Toast 和一个 View 相关。
- ◆ 设置 Toast 的长短，使用 “showToast(Toast.LENGTH_SHORT);” 设置 Toast 短时间显示，使用 “showToast(Toast.LENGTH_LONG);” 设置 Toast 长时间显示。
- ◆ 通过函数 showToast 来显示 Toast，具体说明如下。
 - 当单击【恒久显示】按钮后，程序会长时间地显示提醒，并用 Notification 在状态栏中提示用户。
 - 当单击【短暂显示】按钮后，程序会短时间地显示提醒。

step 06 单击第二个 Button 按钮后来到对应的新界面，新界面使用文件 activity_toast.xml 实现布局，主要代码如下所示。

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



```

    android:layout_height="fill_parent">
    <Button android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="短暂的
Toast"/>
    <Button android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="长久的
Toast" />
</LinearLayout>

```

执行后新界面的效果如图 3-25 所示。

单击【短暂的 Toast】按钮后会短时间显示一个提醒，当单击【长久的 Toast】按钮后会长时间显示一个提醒，这两种方式的提醒界面都是相同的，具体效果如图 3-26 所示。

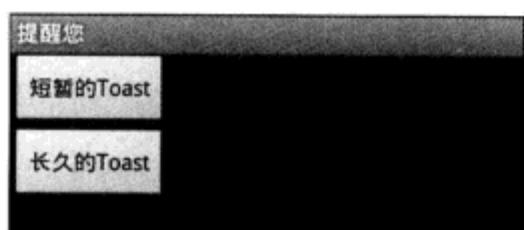


图3-25 运行效果



图3-26 运行效果

3.15 添加/删除Spinner的菜单

在本实例中，使用 Spinner 的自定义菜单实现了和事件的交互处理，利用 ArrayList（动态数组）的依赖性在动态增减 Spinner 下拉菜单中的选项。在具体实现上，将设计一个 EditText 输入框，当用户输入了文字并点击【添加】按钮的同时，就会将输入的值添加 Spinner 至下拉菜单的最后一项，接着 Spinner 会停留在刚添加好的选项上；点击【删除】按钮则会删除选择的 Spinner 选项。

本实例的源码保存在【光盘\daima\第 3 章\dong】，实现流程如下所示。

step 01 编写文件 main.xml，在里面分别插入了 1 个 TextView 文本框、1 个 Button 按钮和 1 个 EditText 编辑框。具体代码如下所示。

```

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

```

```
    android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/white"
    >
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/title"
        android:textColor="@drawable/black"
    >
    </TextView>
    <EditText
        android:id="@+id/myEditText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    >
    </EditText>
    <Button
        android:id="@+id/myButton_add"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="添加一个">
    </Button>
    <Button
        android:id="@+id/myButton_remove"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="删除一个">
    </Button>
    <Spinner
        android:id="@+id/mySpinner"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    >
    </Spinner>
</LinearLayout>
```

step 02 编写文件 dong.java，其具体实现流程如下所示。

- ◆ 定义数组 String[], 保存 4 个数据。
- ◆ 载入 main.xml 的布局，然后新建 ArrayAdapter 对象并将 allCountries 传入。
- ◆ 用 findViewById() 取得对象，并将 ArrayAdapter 添加 Spinner 对象中。
- ◆ 实现添加处理：先比较添加的值是否已存在，不存在才可添加；然后将值添加至

adapter，并取得添加的值的位置，将 Spinner 选择在添加的值的位置；最后，将 myEditText 清空。

- 通过 setOnClickListener 将 myButton_remove 添加 OnClickListener：首先，删除 mySpinner 的值；然后，将 myEditText 清空；最后，将 myTextView 清空。
- 使用 setOnItemSelectedListener 监听用户选择选项，将选择的项 mySpinner 添加 OnItemSelectedListener 中，并将所选项 mySpinner 的值传入 myTextView 中。
 - 文件 dong.java 的主要代码如下所示。

```
public class dong extends Activity
{
    private static final String[] countriesStr =
    { "张三", "李四", "王五", "赵六" };
    private TextView myTextView;
    private EditText myEditText;
    private Button myButton_add;
    private Button myButton_remove;
    private Spinner mySpinner;
    private ArrayAdapter<String> adapter;
    private List<String> allCountries;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);

        allCountries = new ArrayList<String>();
        for (int i = 0; i < countriesStr.length; i++)
        {
            allCountries.add(countriesStr[i]);
        }

        /* new ArrayAdapter对象并将allCountries传入 */
        adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, allCountries);
        adapter
            .setDropDownViewResource
            (android.R.layout.simple_spinner_dropdown_item);

        /* 以findViewById()取得对象 */
    }
}
```



```
myTextView = (TextView) findViewById(R.id.myTextView);
myEditText = (EditText) findViewById(R.id.myEditText);
myButton_add = (Button) findViewById(R.id.myButton_add);
myButton_remove = (Button) findViewById(R.id.myButton_remove);
mySpinner = (Spinner) findViewById(R.id.mySpinner);

/* 将 ArrayAdapter添加到Spinner对象中 */
mySpinner.setAdapter(adapter);

/* 将myButton_add添加OnClickListener */
myButton_add.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        String newCountry = myEditText.getText().toString();

        /* 先比较添加的值是否已存在，不存在才可添加 */
        for (int i = 0; i < adapter.getCount(); i++)
        {
            if (newCountry.equals(adapter.getItem(i)))
            {
                return;
            }
        }

        if (!newCountry.equals(""))
        {
            /* 将值添加至adapter */
            adapter.add(newCountry);
            /* 取得添加的值的位置 */
            int position = adapter.getPosition(newCountry);
            /* 将Spinner选择在添加的值的位置 */
            mySpinner.setSelection(position);
            /* 将myEditText清空 */
            myEditText.setText("");
        }
    }
});

/* 将myButton_remove添加OnClickListener */
});
```

```

myButton_remove.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        if (mySpinner.getSelectedItem() != null)
        {
            /* 删除mySpinner的值 */
            adapter.remove(mySpinner.getSelectedItem().toString());
            /* 将myEditText清空 */
            myEditText.setText("");
            if (adapter.getCount() == 0)
            {
                /* 将myTextView清空 */
                myTextView.setText("");
            }
        }
    });
}

/* 将mySpinner添加OnItemSelectedListener */
mySpinner.setOnItemSelectedListener
(new Spinner.OnItemSelectedListener()
{
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2,
                               long arg3)
    {
        /* 将所选mySpinner的值带入myTextView中 */
        myTextView.setText(arg0.getSelectedItem().toString());
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0)
    {
    }
});
}
}

```

执行后的效果如图 3-27 所示；在框中输入一个选项，如图 3-28 所示，然后单击【添加】按钮后会将输入的信息添加到下拉框中，如图 3-29 所示；当单击【删除】按钮后会删除



当前下拉框中显示的选项信息。

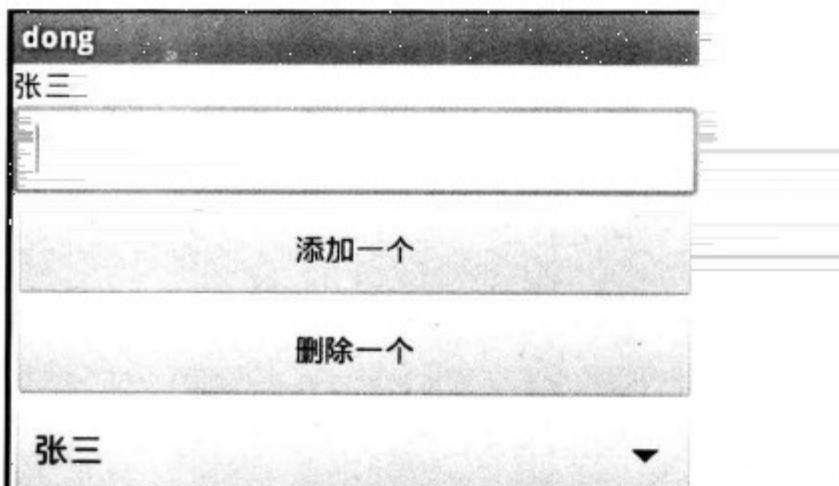


图3-27 初始效果

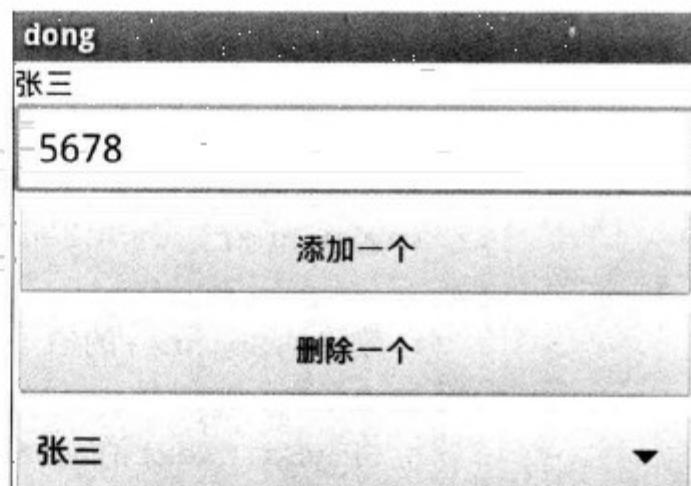


图3-28 添加一个值

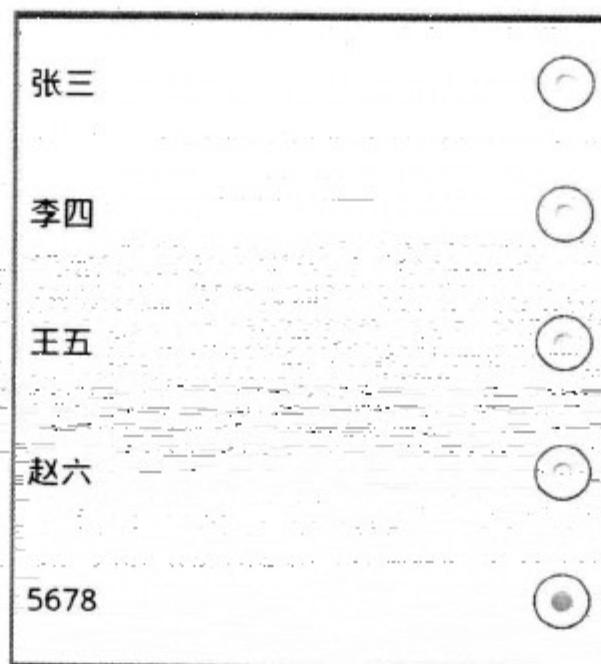
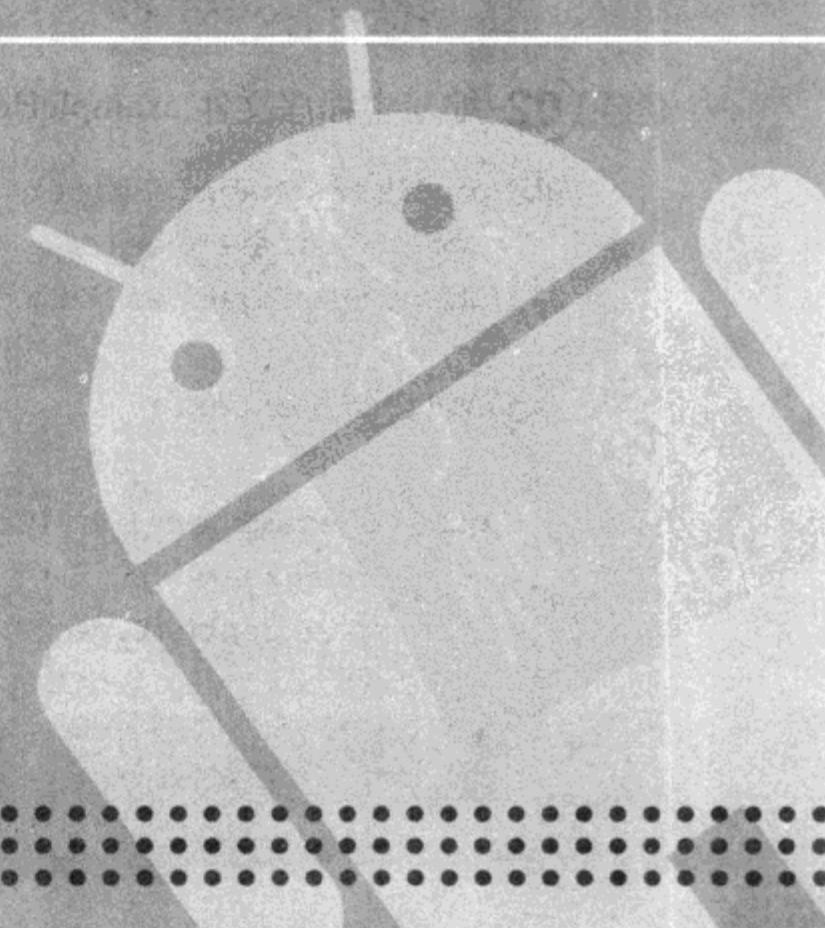


图3-29 添加的新值在列表中

第 4 章

数据存储实战演练

其实在手机中也有 Web 项目和桌面项目中的数据库工具，在手机中也需要数据存储。数据存储是手机领域中最常见应用之一，通过数据存储能够在移动设备中显示不同的信息。本章将详细讲解在 Android 系统中实现数据存储的基本流程。





4.1 使用SharedPreferences存储

SharedPreferences 是 Android 系统中一种重要的数据存储方式，经常存储一些默认欢迎语、登录用户名和密码等简单的信息。SharedPreferences 是以键值对的方式存储，这样开发人员可以很方便地实现读取和存入。

SharedPreferences 类似过去 Windows 系统上的 ini 配置文件，但是可以分为多种权限，可以全局共享访问。虽然此存储方式的整体效率不是特别的高，但是对于常规的轻量级而言比 SQLite 好很多，如果真的存储量不大可以考虑自己定义文件格式。XML 处理时 Dalvik 会通过自带底层的本地 XML Parser 解析，比如 XMLpull 方式，这样对于内存资源占用比较好。

本实例的源码保存在【光盘\daima\第 4 章\jiandan】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体代码如下所示。

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

step 02 编写主程序文件 exampleHelper.java，主要代码如下所示。

```
public class exampleHelper {
    SharedPreferences sp;
    SharedPreferences.Editor editor;

    Context context;

    public exampleHelper(Context c, String name) {
        context = c;
        sp = context.getSharedPreferences(name, 0);
        editor = sp.edit();
    }
}
```

```

public void putValue(String key, String value) {
    editor = sp.edit();
    editor.putString(key, value);
    editor.commit();
}

public String getValue(String key) {
    return sp.getString(key, null);
}
}

```

step 03 编写文件 exampleuse.java，主要代码如下所示。

```

public class exampleuse extends Activity {
    public final static String COLUMN_NAME = "name";
    public final static String COLUMN_MOBILE = "mobile";
    exampleHelper sp;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        sp = new exampleHelper(this, "contacts");

        //设置存储的信息
        sp.putValue(COLUMN_NAME, "Mr WANG");
        sp.putValue(COLUMN_MOBILE, "XXXXXXXXXX");

        //2. to fetch the value
        String name = sp.getValue(COLUMN_NAME);
        String mobile = sp.getValue(COLUMN_MOBILE);

        TextView tv = new TextView(this);
        tv.setText("NAME:" + name + "\n" + "MOBILE:" + mobile);
        setContentView(tv);
    }
}

```

执行后的效果如图 4-1 所示。

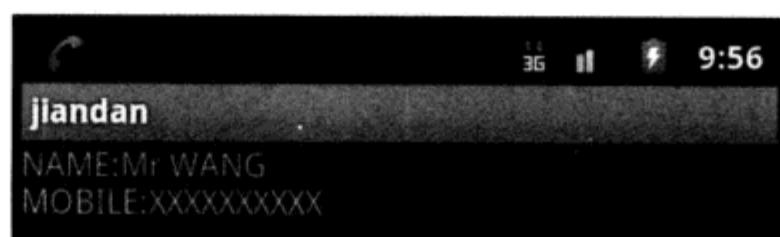


图4-1 执行效果



4.2 使用SQLite存储

在 Android 数据库编程中，和传统编程一样，也通常需要使用 SQL 语句来操作数据库中的数据，例如添加、删除、修改等操作。在 Android 开发应用中，通过 SQL 可以分别实现对这些数据的添加、删除和修改操作。在本实例中，使用 SQLite 存储方式来保存数据，并通过编程方式来操作里面的数据。

本实例的源码保存在【光盘\daima\第 4 章\SQLC】，具体实现流程如下所示。

编写文件 SQLC.java，其具体实现流程如下所示。

step 01 定义继承于 SQLiteOpenHelper 的类 DatabaseHelper，具体代码如下所示。

```
private static class DatabaseHelper extends SQLiteOpenHelper {  
    DatabaseHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        String sql = "CREATE TABLE " + TABLE_NAME + " (" + TITLE  
                + " text not null, " + BODY + " text not null " + ");";  
        Log.i("haiyang:createDB=", sql);  
        db.execSQL(sql);  
    }  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int  
newVersion) {  
    }  
}
```

在上述代码中，类 DatabaseHelper 继承了 SQLiteOpenHelper 类，并且重写了 onCreate 和 onUpgrade 方法。在 onCreate() 方法里边首先我们构造一条 SQL 语句，然后调用 db.execSQL(sql) 执行 SQL 语句。这条 SQL 语句为我们生成了一张数据库表。

此处的 SQLiteOpenHelper 是一个辅助类，功能是生成一个数据库，并管理数据库的版本。当在程序当中调用此类中的方法 getWritableDatabase 或者 getReadableDatabase 的时候，如果没有数据，Android 系统就会自动生成一个数据库。SQLiteOpenHelper 同时也是一个抽象类，其主要功能是通过如下 3 个函数实现的。

- ◆ onCreate (SQLiteDatabase)：在数据库第一次生成的时候会调用这个方法，一般用户在这个方法里边生成数据库表。
- ◆ onUpgrade (SQLiteDatabase, int, int)：当数据库需要升级的时候，Android 系统会主动地调用这个方法。一般我们在这个方法里边删除数据表，并建立新的数据表，当然是否还需要做其他的操作，完全取决于应用的需求。

- ◆ onOpen (SQLiteDatabase)：这是当打开数据库时的回调函数，一般也不会用到。

step 02 编写按钮处理事件，单击【插入两条数据】按钮后插入两条新的记录到数据库中的 diary 表中，并且在屏幕的 title（标题）区域就会有成功的提示，如图 4-2 所示。

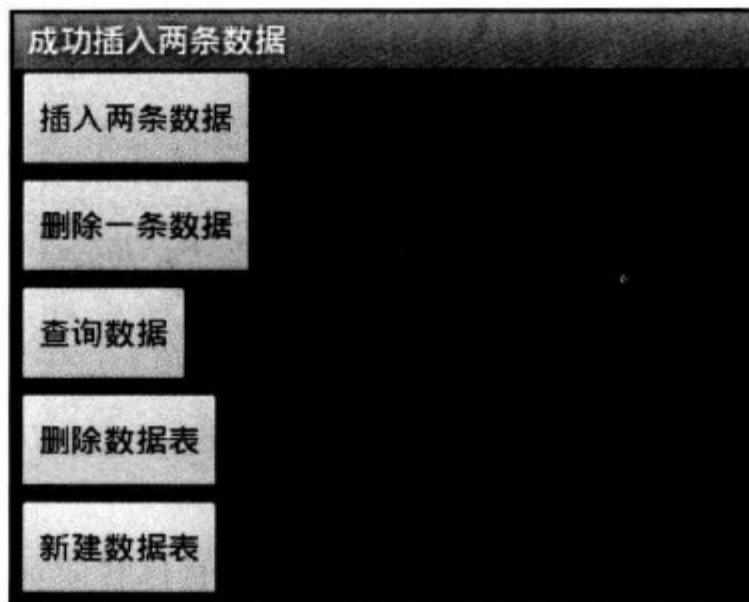


图4-2 插入成功

单击【插入两条数据】按钮后执行监听器里的 onClick 方法，并最终执行 insertItem 方法以插入两条数据。具体代码如下所示。

```
/*插入两条数据*/
private void insertItem() {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String sql1 = "insert into " + TABLE_NAME + " (" + TITLE + ", "
+ BODY
        + ") values('haiyang', 'Android很好');");
    String sql2 = "insert into " + TABLE_NAME + " (" + TITLE + ", "
" + BODY
        + ") values('icesky', 'Android很好');");
    try {
        Log.i("haiyang:sql1=", sql1);
        Log.i("haiyang:sql2=", sql2);
        db.execSQL(sql1);
        db.execSQL(sql2);
        setTitle("成功插入两条数据");
    } catch (SQLException e) {
        setTitle("插入失败");
    }
}
```

在上述代码中，sql1 和 sql2 是标准的实现插入操作的 SQL 语句，“Log.i()”会将参数内容打印到日志中，并且打印级别是 Info 级别，“db.execSQL (sql1)”的功能是执行 SQL 语句。

step 03 单击【查询数据】按钮后会在屏幕的 title 区域中显示当前数据表中的数据条数，因为刚才插入了两条数据，所以此时显示为两条，如图 4-3 所示。

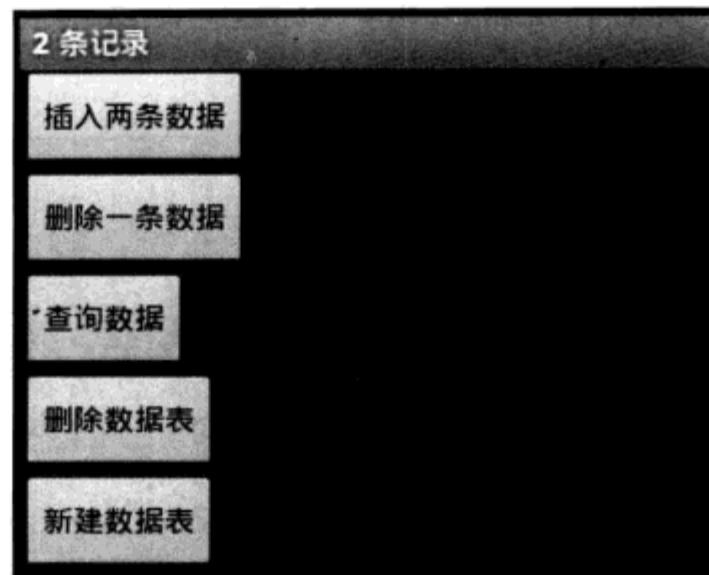


图4-3 查询数据

单击【查询数据】按钮后会执行 showItems 方法，具体代码如下所示。

```
/* 在屏幕的title区域显示当前数据表当中的数据的条数 */
private void showItems() {
    //得到一个可写的数据库/
    SQLiteDatabase db = mOpenHelper.getReadableDatabase();
    String col[] = { TITLE, BODY };
    Cursor cur = db.query(TABLE_NAME, col, null, null, null, null,
    null, null);
    //通过getCount()方法，可以得到Cursor当中数据的个数。/
    Integer num = cur.getCount();
    setTitle(Integer.toString(num) + " 条记录");
}
```

在上述代码中，通过如下代码将查询到的数据放到一个 Cursor 中。

```
Cursor cur = db.query(TABLE_NAME, col, null, null, null, null, null)
```

通过上述语句，在 Cursor 中封装了表 TABLE_NAME 中的所有数据条列。在 query() 方法中包含了 7 个参数，各个参数的具体说明如下。

- ◆ 第 1 个参数：代表数据库里边表的名字，例如在我们这个例子，表的名字就是 TABLE_NAME，也就是 "diary"。
- ◆ 第 2 个参数：代表想要返回数据包含的列的信息。在这个例子当中我们想要得到的列有 title、body。我们把这两个列的名字放到字符串数组里边来。
- ◆ 第 3 个参数 selection：相当于 SQL 语句的 where 部分，如果想返回所有的数据，那么就直接置为 null。
- ◆ 第 4 个参数 selectionArgs：在 selection 部分可能会用到 "?" 操作符，此时需要在

selectionArgs 定义的字符串代替 selection 中的“?”。

- ◆ 第 5 个参数 groupBy：定义查询出来的数据是否分组，如果为 null 则说明不用分组。
- ◆ 第 6 个参数 having：相当于 SQL 语句当中的 having 部分。
- ◆ 第 7 个参数 orderBy：用于描述我们期望的返回值是否需要排序，如果设置为 null 则说明不需要排序。

step 04 单击【删除一条数据】按钮后会删除库中的一条数据，如果成功删除，在屏幕的 title 区域会显示对应的文字提示，如图 4-4 所示。



图4-4 删除一条数据

现在我们再单击查询数据库按钮，看数据库里边的记录是不是少了一条。当单击【删除一条数据】按钮后会执行 deleteItem 方法，其代码如下所示。

```
/*
 * 删除其中的一条数据
 */
private void deleteItem() {
    try {
        SQLiteDatabase db = mOpenHelper.getWritableDatabase();
        db.delete(TABLE_NAME, " title = 'aaa'", null);
        setTitle("删除title为aaa的一条记录");
    } catch (SQLException e) {
    }
}
```

在上述代码中，通过如下语句删除了一条 title 为 'aaa' 的数据。当然如果有很多条数据 title 都为 'aaa'，那么一并删除。在方法 delete 中各个参数的具体说明如下。

- ◆ 第一个参数：表示数据库表名，在这里是 TABLE_NAME，也就是 diary。
- ◆ 第二个参数：相当于 SQL 语句中的 where 部分，也就是描述了删除的条件。如果在第二个参数当中有“？””，那么第三个参数中的字符串会依次替换在第二个参数当中出现的“？”符号。

step 05 单击【删除数据表】按钮后可以删除数据表 diary，如图 4-5 所示。

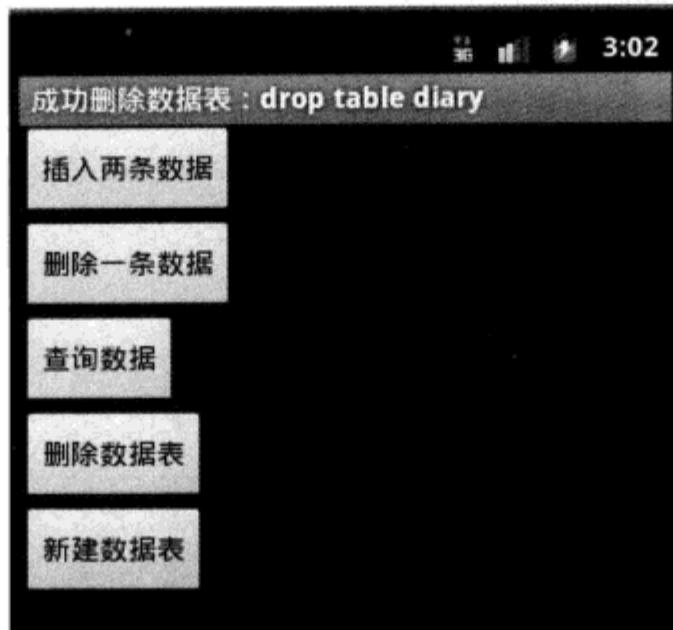


图4-5 删除表

删除数据表功能是通过方法 dropTable 实现的，在此方法中构造了一个标准的删除数据表的 SQL 语句，然后执行这条语句 db.execSQL(sql)。具体代码如下所示。

```
/*删除数据表*/  
private void dropTable() {  
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();  
    String sql = "drop table " + TABLE_NAME;  
    try {  
        db.execSQL(sql);  
        setTitle("成功删除数据表: " + sql);  
    } catch (SQLException e) {  
        setTitle("删除错误");  
    }  
}
```

step 06 此时单击其他的按钮，程序会出现异常，在此单击【新建数据表】按钮，如图 4-6 所示。

如果此时单击【查询数据】按钮，则显示有 0 条数据，如图 4-7 所示。

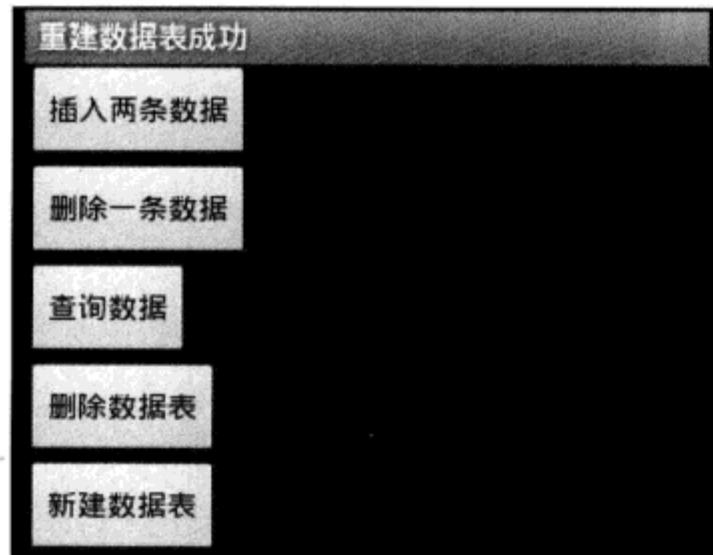


图4-6 新建表

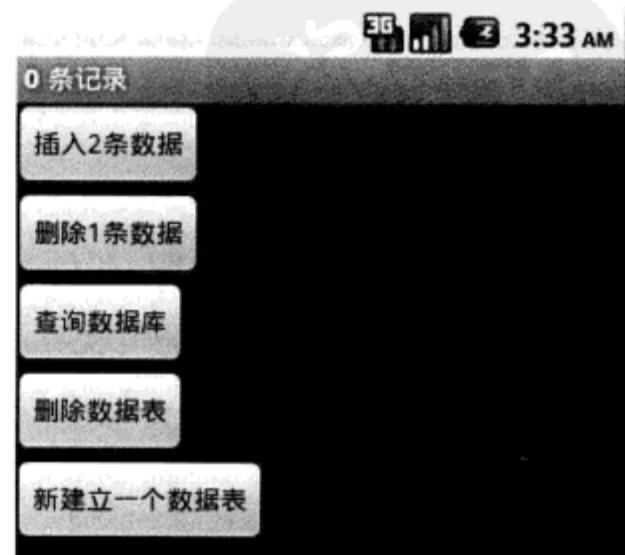
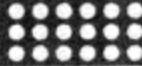


图4-7 显示0条记录



创建新表功能是通过方法 CreateTable 实现的，主要代码如下所示。

```
/*重新建立数据表*/
private void CreateTable() {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String sql = "CREATE TABLE " + TABLE_NAME + " (" + TITLE
        + " text not null, " + BODY + " text not null " + ");";
    Log.i("haiyang:createDB=", sql);

    try {
        db.execSQL("DROP TABLE IF EXISTS diary");
        db.execSQL(sql);
        setTitle("重建数据表成功");
    } catch (SQLException e) {
        setTitle("重建错误");
    }
}
```

在上述代码中，如果已经存在 diary 表，则需要先删除原来的，因为在同一个数据库当中不能出现两张同样名字的表。

4.3 使用ContentProvider存储

Android 系统中的数据是私有的，这些私有数据包括文件数据和数据库数据以及一些其他类型的数据。Android 中的两个程序之间可以进行数据交换，此功能就是通过 ContentProvider 实现的。一个 Content Provider 类实现了一组标准的方法接口，从而能够让其他的应用保存或读取此 Content Provider 的各种数据类型。也就是说，一个程序可以通过实现一个 Content Provider 的抽象接口将自己的数据暴露出去。外界根本看不到，也不用看到这个应用暴露的数据在应用当中是如何存储的，或者是用数据库存储还是用文件存储，还是通过网上获得，这些一切都不重要，重要的是外界可以通过这一套标准及统一的接口和程序里的数据打交道，可以读取程序的数据，也可以删除程序的数据，在中间也会涉及一些权限的问题。

本实例演示了使用 ContentProvider 存储的基本流程，实现源码保存在【光盘 \daima\ 第 4 章\CP】，具体实现流程如下所示。

step 01 单击模拟器的键，单击在桌面上弹出的“Add”选项，如图 4-8 所示。

step 02 进入应用后，依次单击选择“Shortcuts”、“Contact”和“Create new contact”，如图 4-9 所示。



图4-8 出现的桌面

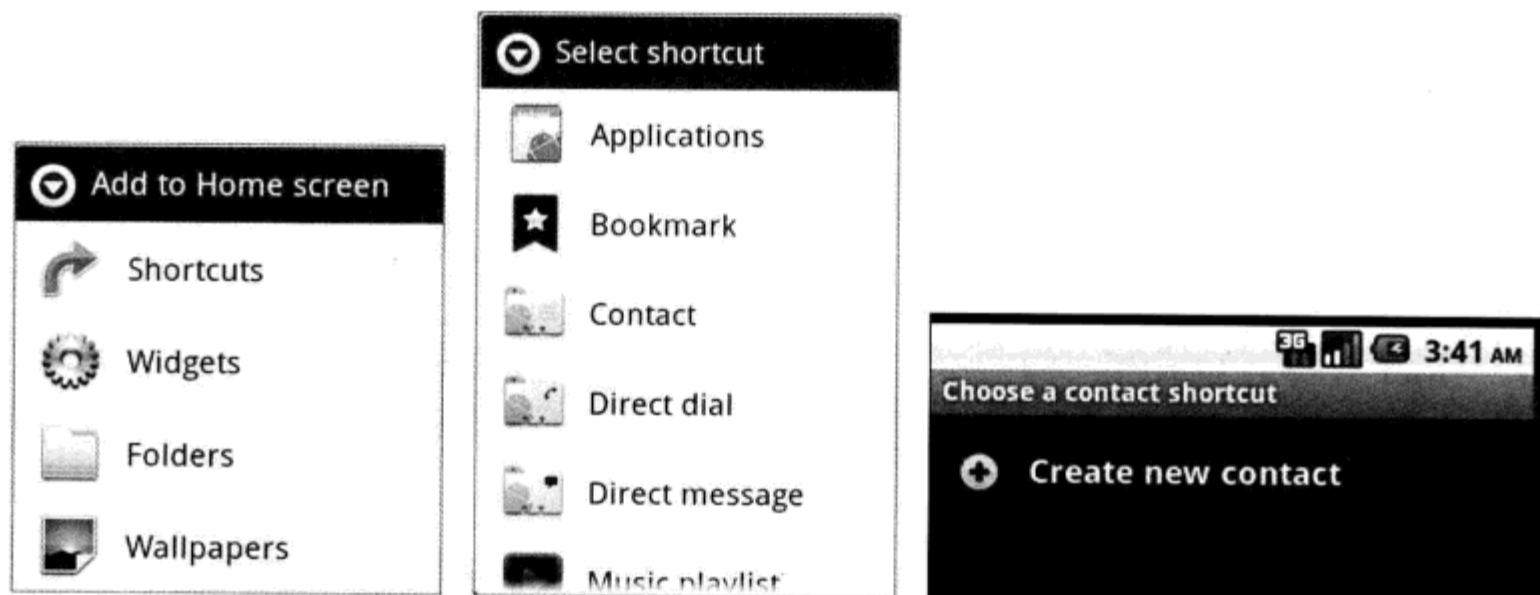


图4-9 单击Create new contact

step 03 在弹出界面中添加联系人姓名和电话号码信息，如图 4-10 所示。

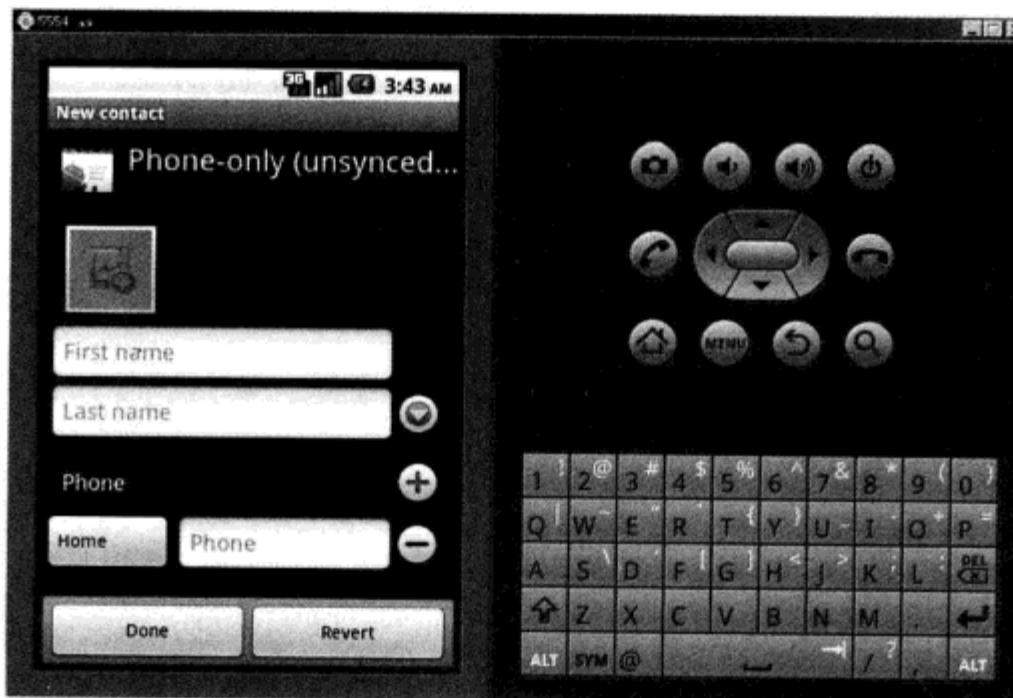


图4-10 添加联系人姓名和电话号码



step 04 填写资料完毕后，单击【Done】按钮完成保存，如图 4-11 所示。



图4-11 单击Save保存

step 05 按照上述操作步骤，即可添加 1 条联系人数据，效果如图 4-12 所示。

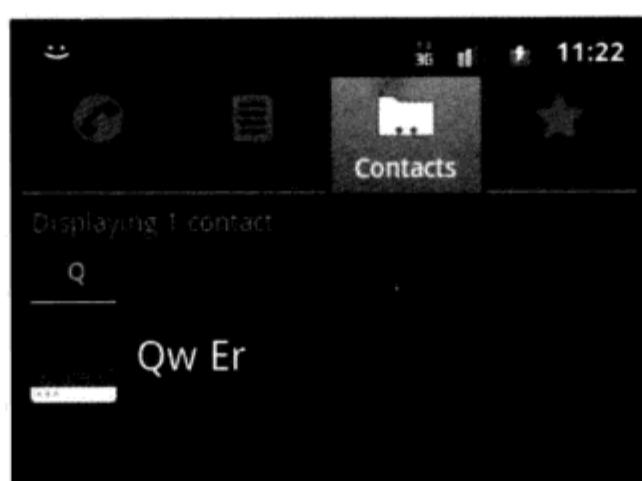


图4-12 添加后的数据

上述存储联系人的操作过程就是将信息使用 ContentProvider 进行存储的过程。接下来再看实例文件代码，文件 shengji2.java 的主要代码如下所示。

```
public class cp extends ListActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Cursor c = getContentResolver().query(Phones.CONTENT_URI,
null, null, null, null);
        startManagingCursor(c);
       ListAdapter adapter = new SimpleCursorAdapter(this,
            android.R.layout.simple_list_item_2, c,
            new String[] { Phones.NAME, Phones.NUMBER },
            new int[] { android.R.id.text1, android.R.id.text2 });
        setListAdapter(adapter);
    }
}
```

4.4 开发一个日记簿项目

本实例实现了一个基本的笔记簿项目，我们不但可以实现增、删、改、查的操作，而且还能够把数据库当中的数据显示在一个 ListView 当中，通过对 ListView 的操作，实现对数据的增、删、改、查操作。

本实例的源码保存在【光盘\daima\第4章\riji】，具体实现流程如下所示。

step 01 编写布局文件 diary_list.xml，此文件是项目执行后首先显示的界面，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8 "?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ListView android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:id="@+id/android:empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="按下MENU写日记！！" />
</LinearLayout>
```

执行后效果如图 4-13 所示。



图4-13 首先显示的界面

step 02 编写文件 DiaryDbAdapter.java，此文件的实现流程如下所示。

- 定义类 DiaryDbAdapter，此类用于封装 DatabaseHelper 和 SQLiteDatabase 类，使我们对数据库的操作更加安全和方便。定义 DatabaseHelper 类的方法非常简单，只不过这个例子里边，在 onUpgrade 增加了升级的代码，具体如下所示。

```
private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) { //生成数据库
```

```

        db.execSQL(DATABASE_CREATE);

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS diary");
        onCreate(db);
    }
}

```

在 DiaryDbAdapter 类中提供了如下方法。

- open()：调用这个方法后，如果数据库还没有建立，那么会建立数据库，如果数据库已经建立了，那么会返回可写的数据库实例。
- close()：调用此方法，DatabaseHelper 会关闭对数据库的访问。
- createDiary (String title, String body) 通过一个 title 和 body 字段在数据库当中创建一条新的纪录。
- deleteDiary (long rowId)：通过记录的 id，删除数据库中的那条记录。
- getAllNotes()：得到 diary 表中所有的记录，并且以一个 Cursor 的形式进行返回。
- getDiary (long rowId)：通过记录的主键 id，得到特定的一条记录。
- updateDiary (long rowId, String title, String body)：更新主键 id 为 rowId 那条记录中的两个字段 title 和 body 字段的内容。
- 实现插入日记功能，通过 SQL 语句实现插入操作，SQL 语句的好处是比较直观。在这个例子当中，将要插入的值都放到一个 ContentValues 的实例中，然后执行插入操作，主要代码如下所示。

```

public long createDiary(String title, String body) {
    //实例化一个contentValues类
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_TITLE, title);
    initialValues.put(KEY_BODY, body); // 将列名和对应的列值放置到
initialValues里边
    Calendar calendar = Calendar.getInstance();
    String created = calendar.get(Calendar.YEAR) + "年"
        + calendar.get(Calendar.MONTH) + "月"
        + calendar.get(Calendar.DAY_OF_MONTH) + "日"
        + calendar.get(Calendar.HOUR_OF_DAY) + "小时"
        + calendar.get(Calendar.MINUTE) + "分钟";
    initialValues.put(KEY_CREATED, created);
    return mDb.insert(DATABASE_TABLE, null, initialValues); // 插入
一条新的纪录
}

```



- 实现修改日记功能，在此采用 ContentValues 机制来更新一条记录，主要代码如下所示。

```
public boolean updateDiary(long rowId, String title, String body) {  
    ContentValues args = new ContentValues();  
    args.put(KEY_TITLE, title);  
    args.put(KEY_BODY, body);  
    Calendar calendar = Calendar.getInstance();  
    String created = calendar.get(Calendar.YEAR) + "年"  
        + calendar.get(Calendar.MONTH) + "月"  
        + calendar.get(Calendar.DAY_OF_MONTH) + "日"  
        + calendar.get(Calendar.HOUR_OF_DAY) + "小时"  
        + calendar.get(Calendar.MINUTE) + "分钟";  
    args.put(KEY_CREATED, created);  
    return mDb.update(DATABASE_TABLE, args, KEY_ROWID + "=" +  
        rowId, null) > 0;  
}
```

此时返回到程序的主界面，对应的 Activity 是 riji。当按下 Menu 键后会出现如图 4-14 所示界面。



图4-14 运行界面

step 03 编写文件 riji.java，此文件的实现流程如下所示。

- 编写方法 onMenuItemSelected，用于处理选择按下 Menu 时弹出的选项，主要代码如下所示。

```
@Override  
public boolean onMenuItemSelected(int featureId, MenuItem item) {  
    switch (item.getItemId()) {
```

```

        case INSERT_ID:
            createDiary();
            return true;
        case DELETE_ID:
            mDbHelper.deleteDiary(getListView().getSelectedItemId());
            renderListView();
            return true;
    }
    return super.onOptionsItemSelected(featureId, item);
}

```

在上述代码中，如果单击添加一篇新日记按钮那么会执行到 `createDiary()` 语句；如果单击删除一条记录，会执行 “`mDbHelper.deleteDiary(getListView().getSelectedItemId())`” 语句，首先删除当前被选中的某一项所对应的数据库当中的记录；“`renderListView()`” 语句用于重新对界面刷新。

- ◆ 方法 `createDiary` 首先构造了一个 intent，此 intent 负责跳转到 `ActivityDiaryEdit` 里，然后启动这个 intent，并且需要返回值。主要代码如下所示。

```

private void createDiary() {
    Intent i = new Intent(this, ActivityDiaryEdit.class);
    startActivityForResult(i, ACTIVITY_CREATE);
}

```

- ◆ 在此文件中有多处地方都用到了方法 `renderListView`。在 `onCreate` 里边用这个方法显示 `ListView`。当 `ListView` 需要发生变化后，例如，删除了一条记录或者增加了一条记录的时候，我们调用这个方法来刷新 `ListView`，方法 `renderListView()` 的主要代码如下所示。

```

private void renderListView() {
    mDiaryCursor = mDbHelper.getAllNotes();
    startManagingCursor(mDiaryCursor);
    String[] from = new String[] { DiaryDbAdapter.KEY_TITLE,
        DiaryDbAdapter.KEY_CREATED };
    int[] to = new int[] { R.id.text1, R.id.created };
    SimpleCursorAdapter notes = new SimpleCursorAdapter(this,
        R.layout.diary_row, mDiaryCursor, from, to);
    setListAdapter(notes);
}

```

- ◆ `mDiaryCursor = mDbHelper.getAllNotes()` 语句：首先获取数据库当中的所有数据，这些数据以 Cursor 的形式存在。
- ◆ `startManagingCursor (mDiaryCursor)` 语句：将生成的 Cursor 交给 Activity 来管理，这样的好处是系统能自动做很多事情，比如当程序暂停的时候，这个系统可以卸



载 Cursor 以节省空间，当程序重新启动的时候系统重新查询生成 Cursor。

- ◆ String[] from：定义了 ListView 每一排对应的数据是从数据库中的哪个列表里选取。
- ◆ int[] to：和 SimpleAdapter 类似，里边是一个 View 的数组。这些 View 只能是 TextView 或者 ImageView。这些 View 是以 id 的形式来表示的，如 Android.R.id.text1。
- ◆ setListAdapter (notes) 语句：将 SimpleCursorAdapter 和 ListActivity 里边的 ListView 绑定起来，至此在界面当中才会显示出列表来。

当单击【写新的】按钮后的界面如图 4-15 所示。

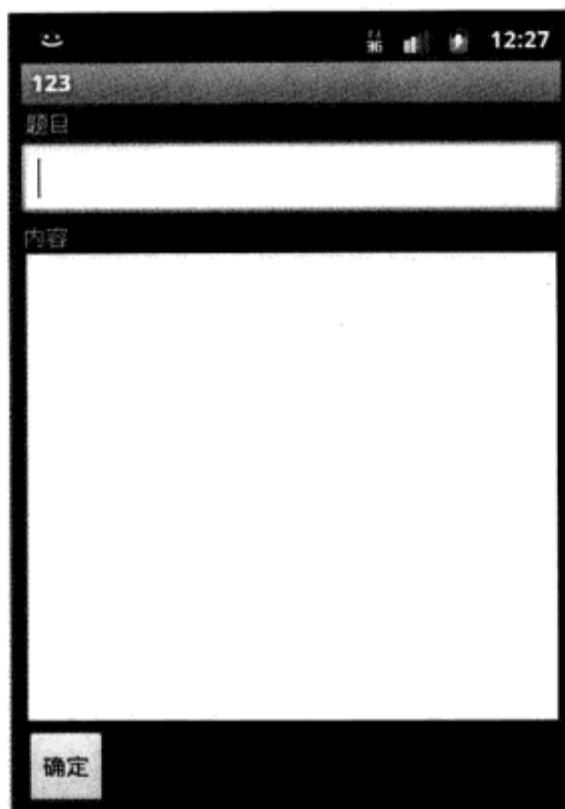


图4-15 添加日记界面

图 4-15 所示的 Activity 是 ActivityDiaryEdit，对应的布局文件是 edit.xml，在里面插入了 LinearLayout 控件、TextView 控件和 EditText 控件，在 LinearLayout 中放置了一些文本框、输入框和一些 Button 按钮。主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="@string/title"
            android:padding="2px" />
        <EditText android:id="@+id/title" />
    
```



```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:layout_
weight="1" />
    </LinearLayout>
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/
body" />
    <EditText android:id="@+id/body" android:layout_width="fill_
parent"
        android:layout_height="fill_parent" android:layout_weight="1"
        android:scrollbars="vertical" android:gravity="top" />
    <Button android:id="@+id/confirm" android:text="@string/confirm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>

```

输入日记内容并单击【确定】按钮后，调用按钮绑定的单击监听器里边的 onClick 方法，主要代码如下所示。

```

public void onClick(View view) {
    String title = mTitleText.getText().toString();
    String body = mBodyText.getText().toString();
    if (mRowId != null) {
        mDbHelper.updateDiary(mRowId, title, body);
    } else mDbHelper.createDiary(title, body);
    Intent mIntent = new Intent();
    setResult(RESULT_OK, mIntent);
    finish();
}

```

目前 mRowId 为 null，所以执行 mDbHelper.createDiary (title, body) 语句将数据保存到数据当中。通过 setResult (RESULT_OK, mIntent) 语句用于设置返回值。执行完上边的代码后，系统跳转到 riji，并执行回调函数 onActivityResult，主要代码如下所示。

```

protected void onActivityResult(int requestCode, int resultCode,
    Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    renderListView();
}

```

- 修改已经添加的日记进行处理。当单击 ListView 里边的条列后，可以对刚才保存的数据进行编辑。上述功能是通过 ActivityMain 中的 onListItemClick 方法实现的，主要代码如下所示。



```
// 需要对position和id进行一个很好的区分  
// position指的是点击的这个ViewItem在当前ListView中的位置  
// 每一个和ViewItem绑定的数据，肯定都有一个id，通过这个id可以找到那条数据。  
protected void onListItemClick(ListView l, View v, int position,  
long id) {  
    super.onListItemClick(l, v, position, id);  
    Cursor c = mDiaryCursor;  
    c.moveToPosition(position);  
    Intent i = new Intent(this, ActivityDiaryEdit.class);  
    i.putExtra(DiaryDbAdapter.KEY_ROWID, id);  
    i.putExtra(DiaryDbAdapter.KEY_TITLE, c.getString(c  
        .getColumnIndexOrThrow(DiaryDbAdapter.KEY_TITLE)));  
    i.putExtra(DiaryDbAdapter.KEY_BODY, c.getString(c  
        .getColumnIndexOrThrow(DiaryDbAdapter.KEY_BODY)));  
    startActivityForResult(i, ACTIVITY_EDIT);  
}
```

关于对上述代码的几点说明：

- ◆ c.moveToPosition (position) 语句：将在 Cursor 当中的指针移到 position 位置，这个 position 是我们单击的这个一列在整个列表中的位置。
- ◆ Intent i = new Intent (this, ActivityDiaryEdit.class) 语句：构造一个跳转到 ActivityDiaryEdit 的 intent。
- ◆ putExtra() 方法：将要传递的数据放到 intent 当中。
- ◆ c.getString (c.getColumnIndexOrThrow(DiaryDbAdapter.KEY_TITLE))：得到这一条数据中列名为 title 的值。
- ◆ c.getString (c.getColumnIndexOrThrow(DiaryDbAdapter.KEY_BODY))：得到这一条数据中列名为 body 的值。
- ◆ startActivityForResult (i, ACTIVITY_EDIT) 语句：启动 intent，发生 Activity 的跳转。

(4) 编写文件 ActivityDiaryEdit，其中方法 onCreate 的主要代码如下所示。

```
Bundle extras = getIntent().getExtras();  
if (extras != null) {  
    String title = extras.getString(DiaryDbAdapter.KEY_TITLE);  
    String body = extras.getString(DiaryDbAdapter.KEY_BODY);  
    mRowId = extras.getLong(DiaryDbAdapter.KEY_ROWID);  
    if (title != null) {  
        mTitleText.setText(title);  
    } if (body != null) {  
        mBodyText.setText(body);  
    }  
}
```

在上述代码中，ActivityDiaryEdit 的 Activity 中有 2 个 intent 可以跳转进来。



- ◆ 第一种是新建一篇日记的时候，此时 intent 中的 extras 部分没有任何数据；
- ◆ 第二种情况是在单击列表的某一个条列的时候，此时的 intent 如上所示会携带 extras 数据。所以在 ActivityDiaryEdit 中我们通过判断 extras 是否为 null，就可以判断是哪种 intent 启动的。

发表日记后会在屏幕中列表显示发布的日记标题，如图 4-16 所示。上下移动焦点（可以通过键盘的上下键或者模拟器中的上下按键），可以对拥有焦点的那一项进行删除，如图 4-17 所示。

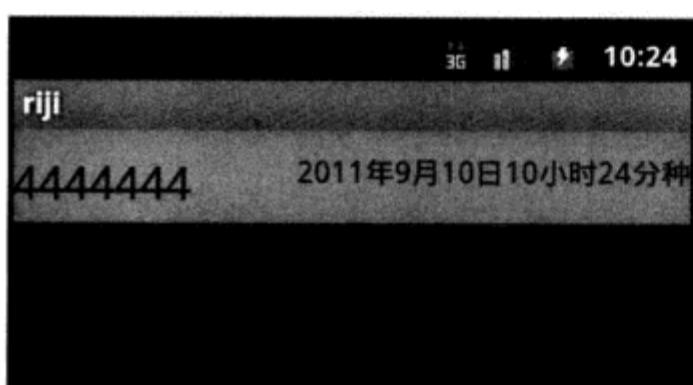


图4-16 上下移动焦点界面

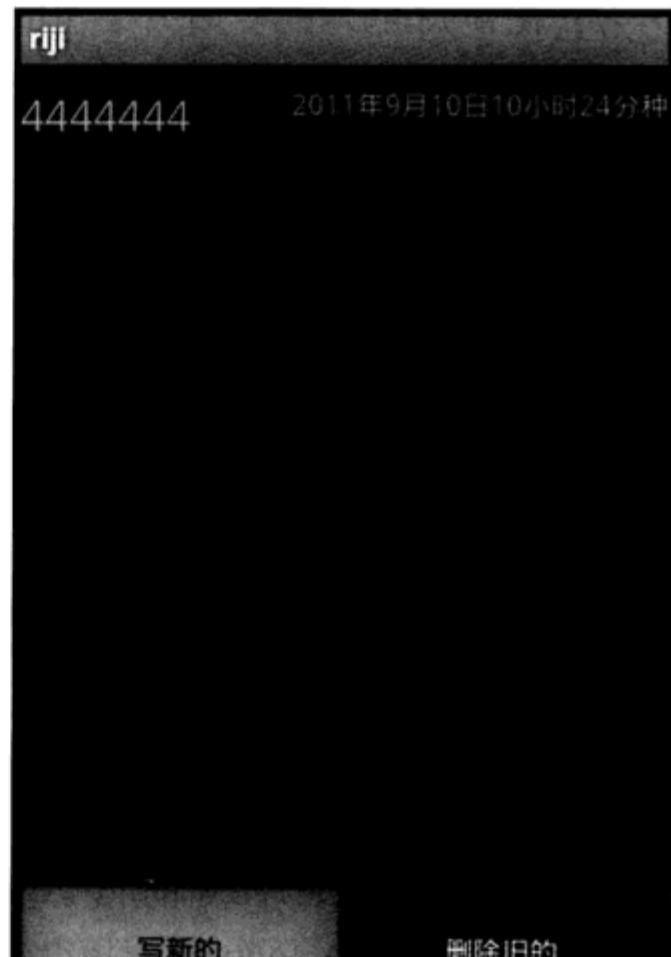


图4-17 开始删除一条日记

删除操作时，首先执行 onMenuItemSelected 中的代码。

```
mDbHelper.deleteDiary(getListView().getSelectedItemID());
renderListView();
```

4.5 升级日记簿功能

在本书前面的内容中已经学习过 ContentProvider 的内容，并且使用了系统中的一个联系人的 ContentProvider，在本实例中，日记本功能是通过 ContentProvider 实现的，而不是直接用数据库实现。这样外界的程序就可以访问得到日记本这个应用数据。

本实例的功能是用 ContentProvider 实现 4.4 节中的日记簿功能，如图 4-18 所示。

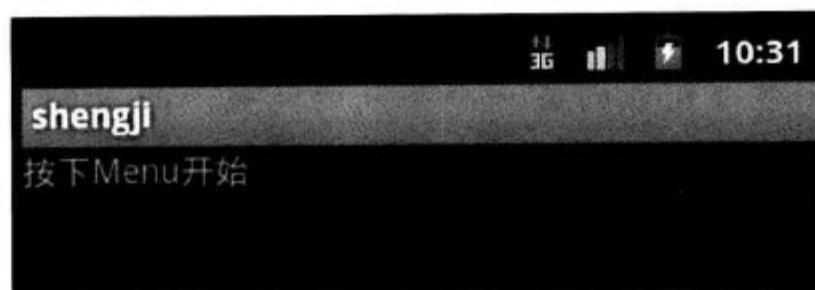


图4-18 执行效果

本实例的源码保存在【光盘\daima\第4章\shengji】，具体实现流程如下所示。

step 01 图 4-18 所示的界面效果在前面的例子中已经见过。本例的 Activity 是一个 ListActivity，对应的布局文件是 diary_list.xml，其主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ListView android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:id="@+id/android:empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="按下Menu开始"/>
</LinearLayout>
```

在上述代码中，ListView 的 id 必须定义成 Android:id="@+id/Android:list" 的形式，这样系统才可以引用到。

step 02 本实例日记本的数据存储在 SQLite 数据库中，执行增、删、改、查操作时不是直接访问数据库实现的，而是通过日记本程序的 ContentProvider 实现的。接下来编写文件 Diary.java，在里面定义了 Diary 类，在此类中有一个内部静态类 DiaryColumns，和它的名字意思一样，此类里主要定义了日记本数据库的列表字段的名字，主要代码如下所示。

```
public static final class DiaryColumns implements BaseColumns {
    private DiaryColumns() {}
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/diaries");
    public static final String CONTENT_TYPE = "vnd.Android.cursor.dir/vnd.google.diary";
    public static final String CONTENT_ITEM_TYPE = "vnd.Android.cursor.item/vnd.google.diary";
    public static final String DEFAULT_SORT_ORDER = "created DESC";
    public static final String TITLE = "title";
```

```

    public static final String BODY = "body";
    public static final String CREATED = "created";
}

```

在上述代码中，BaseColumns 是一个接口，里边有 2 个变量，一个是 _ID=" _id"，一个是 _COUNT=" _ count"。在 Android 中，每一个数据库表至少有一个字段，而且这个字段是 _id。所以当我们构造列名的辅助类时，直接实现 BaseColumns，这样便默认地拥有了 _id 字段。

在本实例的数据表里一共有 4 个字段，分别是 “id”、“title”、“body”、“created”。

step 03 编写文件 DiaryContentProvider.java，先分析类 DiaryContentProvider，此类继承自 ContentProvider，在里面实现了 ContentProvider 的一些接口方法，并且成为日记本的一个 ContentProvider。现在通过学习这个类，来详细了解实现一个自定义的 ContentProvider 的具体过程。

- 先定义一个 public static final 的 Uri 类型的变量，并且命名为 CONTENT_URI。Diary Content Provider 所能处理的 Uri 都是基于 CONTENT_URI 来构建的，需要注意的是，这个 CONTENT_URI 中的内容以 content:// 开头，并且全部小写，且全局唯一。

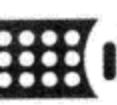
下边是在这个例子中的一个普通 URI，通过分析这个 URI，可以了解 content URI 的构成，具体格式如下所示。

```
content://com.shengji4.diarycontentprovider/diaries/1
```

- 第一部分 “content://”：这部分是固定存在的，不需要做任何修改。
- 第二部分是授权（AUTHORITY）部分：即 com.shengji4.diarycontentprovider，授权部分是唯一的，在程序中一般是我们实现的那个 Content Provider 的全称，并且全都小写。这部分是和在 AndroidManifest.xml 文件当中的如下部分是对应的。

```
<provider android:name="DiaryContentProvider"
    android:authorities="shiyongcontentprovider.diarycontentprovider" />
```

- 第三部分：是请求数据的类型，例如，在这个例子当中定义的类型是 diaries。当然这一部分可以是由 0 个片段或者多个片段构成，如果 Content Provider 只是暴露出了一种类型的数据，那么这部分可以为空，但是如果暴露出了多种，尤其是包含子类的时候，就不能为空，例如，日记本程序里边可以暴露出两种数据，一种是用户自己的 "diaries/my"，另一种是其他人的 "diaries/others"。
- 第四部分：是 "1"，当然这部分是允许为空的。如果为空，表示请求全部数据；如果不为空，表示请求特定 ID 的数据。
- 构建用户的数据存储系统。在这个例子当中，是将数据存储到数据库系统当中。通常是将数据存储在数据库系统中，但是也可以将数据存储在其他的地方，如文件系统等。



- ◆ 实现 ContentProvider 这个抽象类的抽象方法，具体如下所示。
 - public boolean onCreate()：当 ContentProvider 生成的时候调用此方法。
 - public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)：此方法返回一个 Cursor 对象作为查询结果集。
 - public Uri insert (Uri uri, ContentValues initialValues)：此方法负责往数据集当中插入一列，并返回这一列的 Uri。
 - public int delete (Uri uri, String where, String[] whereArgs)：此方法负责删除指定 Uri 的数据。
 - public int update (Uri uri, ContentValues values, String where, String[] whereArgs)：此方法负责更新指定 Uri 的数据。
 - public String getType (Uri uri)：返回所给 Uri 的 MIME 类型。
- ◆ 在 AndroidManifest.xml 文件中增加<provider>标签，本例中的实现的代码如下所示。

```
<provider android:name="DiaryContentProvider" android:authorities="com.shengji4.diarycontentprovider" />
```

其中“Android:name”是我们实现的这个 content provider 的类名；“Android:authorities”是 content URI 的第二部分，即授权部分，实现代码是“com.shiyongcontentprovider.diarycontentprovider”。

step 04 继续看实现 DiaryContentProvider 类的代码，因为此实例的数据是存储在数据库当中，所以需要定义 DatabaseHelper 辅助类。主要代码如下所示。

```
private static class DatabaseHelper extends SQLiteOpenHelper {  
    DatabaseHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
        Log.i("jinyan", "DATABASE_VERSION=" + DATABASE_VERSION);  
    }  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        Log.i("jinyan", "onCreate(SQLiteDatabase db)");  
        String sql = "CREATE TABLE " + DIARY_TABLE_NAME + " ("  
            + DiaryColumns._ID + " INTEGER PRIMARY KEY,"  
            + DiaryColumns.TITLE + " TEXT," + DiaryColumns.BODY  
            + " TEXT," + DiaryColumns.CREATED + " TEXT" + ");";  
        //  
        sql = "CREATE TABLE " + DIARY_TABLE_NAME + " ("  
            + DiaryColumns._ID + " INTEGER PRIMARY KEY,"  
            + DiaryColumns.TITLE + " varchar(255)," + DiaryColumns.BODY  
            + " TEXT," + DiaryColumns.CREATED + " TEXT" + ");";  
        Log.i("jinyan", "sql=" + sql);  
        db.execSQL(sql);  
    }  
}
```

```

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        Log.i("jinyan",
              "onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion)="
              + newVersion);
        db.execSQL("DROP TABLE IF EXISTS diary");
        onCreate(db);
    }
}

```

通过上述代码，在 DatabaseHelper 中操作数据库的辅助类，通过此类可以生成数据库，并且维护这个数据库。

step 05 然后在类 DiaryContentProvider 中定义需要的变量和常量，其中这些常量主要是描述数据库的信息。

```

private static final String DATABASE_NAME = "database";
private static final int DATABASE_VERSION = 1;
private static final String DIARY_TABLE_NAME = "diary";
private static HashMap<String, String> sDiariesProjectionMap;
private static final int DIARIES = 1;
private static final int DIARY_ID = 2;
private static final UriMatcher sUriMatcher;

```

step 06 编写 static 模块，具体代码如下所示。

```

sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
sUriMatcher.addURI(Diary.AUTHORITY, "diaries", DIARIES);
sUriMatcher.addURI(Diary.AUTHORITY, "diaries/#", DIARY_ID);

```

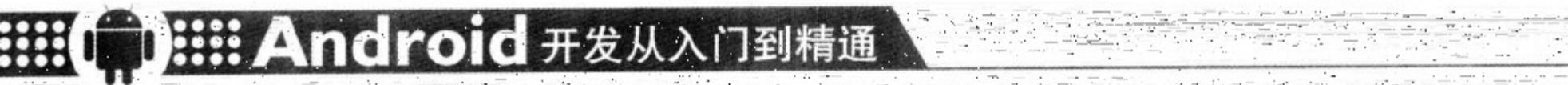
上面代码中的 UriMatcher 是匹配 Uri 的一个辅助类，具体说明如下所示。

- ◆ sUriMatcher.addURI(Diary.AUTHORITY, "diaries", sUriMatcher.match(uri))：如果 Uri 是 content://com.shiyongcontentprovider.diarycontentprovider/diaries/，那么 sUriMatcher.match(uri) 的返回值就是 DIARIES。
- ◆ sUriMatcher.addURI(Diary.AUTHORITY, "diaries/#", DIARY_ID)：如果我们的 Uri 是 content://com.shiyongcontentprovider.diarycontentprovider/diaries/id（其中后边的 id 是一个数字），那么 sUriMatcher.match(uri) 的返回值就是 DIARY_ID。

通过 UriMatcher 类可以很方便地判断一个 Uri 的类型，特别是判断这个 Uri 是对单个数据的请求，还是对全部数据的请求。

step 07 继续编写文件 DiaryContentProvider.java，开始编写抽象方法，具体实现流程如下所示。

- ◆ 编写插入方法 insert()，具体代码如下所示。



```
@Override
public Uri insert(Uri uri, ContentValues initialValues) {
    if (sUriMatcher.match(uri) != DIARIES) {
        throw new IllegalArgumentException("Unknown URI " + uri);
    }
    ContentValues values;
    if (initialValues != null) {
        values = new ContentValues(initialValues);
    } else {
        values = new ContentValues();
    }
    if (values.containsKey(Diary.DiaryColumns.CREATED) == false) {
        values.put(Diary.DiaryColumns.CREATED, getFormateCreatedDate());
    }
    if (values.containsKey(Diary.DiaryColumns.TITLE) == false) {
        Resources r = Resources.getSystem();
        values.put(Diary.DiaryColumns.TITLE, r
                .getString(Android.R.string.untitled));
    }
    if (values.containsKey(Diary.DiaryColumns.BODY) == false) {
        values.put(Diary.DiaryColumns.BODY, "");
    }
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    long rowId = db.insert(DIARY_TABLE_NAME, DiaryColumns.BODY, values);
    if (rowId > 0) {
        Uri diaryUri = ContentUris.withAppendedId(Diary.DiaryColumns.
                CONTENT_URI, rowId);
        return diaryUri;
    }
    throw new SQLException("Failed to insert row into " + uri);
}
```

- sUriMatcher.match(uri) != DIARIES 语句：判断传进来的 Uri，如果这个 Uri 不是 DIARIES 类型，那么这个 Uri 就是一个非法的 Uri。
- SQLiteDatabase db = mOpenHelper.getWritableDatabase() 语句：负责得到一个 SQLiteDatabase 的实例。
- db.insert (DIARY_TABLE_NAME, DiaryColumns.BODY, values) 语句：负责插入一条记录到数据库中。

方法 insert() 返回的是一个 Uri，而不是一个记录的 id，所以，需要把记录的 id 构造成一个 Uri: Uri diaryUri = ContentUris.withAppendedId(Diary.DiaryColumns.CONTENT_URI, rowId).withAppendedId() 方法在下边的小知识当中详细介绍。



ContentUris 是 content URI 的一个辅助类，在里面要有如下 2 个常用的方法。

- public static Uri withAppendedId (Uri contentUri, long id)：负责把 id 和 contentUri 连接成一个新的 Uri。比如在例子中是这么使用的：

```
ContentUris.withAppendedId (Diary. DiaryColumns.CONTENT_URI, rowId)
```

如果 rowId 为 100，则现在的 Uri 内容就是：

```
content://com.shengji4.diarycontentprovider/diaries/100
```

- public static long parseId (Uri contentUri)：负责把 content URI 后边的 id 解析出来，比如现在这个 content URI 是 content://com.shiyongcontentprovider.diarycontentprovider/diaries/100，那么这个函数的返回值就是 100。
- 编写删除方法 delete()，具体实现代码如下所示。

```
public int delete(Uri uri, String where, String[] whereArgs) {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String rowId = uri.getPathSegments().get(1);
    return db.delete(DIARY_TABLE_NAME, DiaryColumns._ID + "=" + rowId, null);
}
```

- rowId = uri.getPathSegments().get(1)：用于得到 rowId 的值。
- getPathSegments() 方法：得到一个 String 的 List，在我们例子当中 uri.getPathSegments().get(1) 为 rowId，如果是 uri.getPathSegments().get(0) 那值就为 "diaries"。
- db.delete(DIARY_TABLE_NAME, DiaryColumns._ID + "=" + rowId, null)：是标准的 SQLite 删除操作。第一个参数是数据表的名字，第二个参数相当于 SQL 语句当中的 where 部分。
- 编写更新数据的方法 update()，具体实现代码如下所示。

```
@Override
public int update(Uri uri, ContentValues values, String where,
                  String[] whereArgs) {
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    String rowId = uri.getPathSegments().get(1);
    return db.update(DIARY_TABLE_NAME, values, DiaryColumns._ID + "="
                     + rowId, null);
}
```

通过上述代码，先得到 SQLiteDatabase 的实例，然后得到 rowId，最后再调用 db.update(DIARY_TABLE_NAME, values, DiaryColumns._ID + "=" + rowId, null) 语句执行更新工作。

- 编写查寻一条数据方法 query()，具体代码如下所示。



```
@Override  
public Cursor query(Uri uri, String[] projection, String selection,  
                      String[] selectionArgs, String sortOrder) {  
    SQLiteQueryBuilder qb = new SQLiteQueryBuilder();  
    switch (sUriMatcher.match(uri)) {  
        case DIARIES:  
            qb.setTables(DIARY_TABLE_NAME);  
            break;  
        case DIARY_ID:  
            qb.setTables(DIARY_TABLE_NAME);  
            qb.appendWhere(DiaryColumns._ID + "="  
                           + uri.getPathSegments().get(1));  
            break;  
        default:  
            throw new IllegalArgumentException("Unknown URI " + uri);  
    }  
    String orderBy;  
    if (TextUtils.isEmpty(sortOrder)) {  
        orderBy = Diary.DiaryColumns.DEFAULT_SORT_ORDER;  
    } else {  
        orderBy = sortOrder;  
    }  
    SQLiteDatabase db = mOpenHelper.getReadableDatabase();  
    Cursor c = qb.query(db, projection, selection, selectionArgs, null,  
                       null, orderBy);  
    return c;  
}
```

关于对上述代码的具体说明如下：

- SQLiteQueryBuilder：是一个构造 SQL 查询语句的辅助类。
- sUriMatcher.match(uri)：根据返回值可以判断这次查询请求时，它是请求全部数据还是某个 id 的数据。如果返回值是 DIARIES，那么只需要执行 qb.setTables(DIARY_TABLE_NAME) 语句就可以了；如果返回值是 DIARY_ID，那么还需要将 where 部分的参数设置进去，代码为 qb.appendWhere(DiaryColumns._ID + "=" + uri.getPathSegments().get(1))。
- SQLiteDatabase db = mOpenHelper.getReadableDatabase()：得到一个可读的 SQLiteDatabase 实例。
- Cursor c = qb.query(db, projection, selection, selectionArgs, null, null, orderBy) 语句：类似于一个标准的 SQL 查询，但是此查询是 SQLiteQueryBuilder 来发起的，而不是 SQLiteDatabase 直接发起的，所以在参数方面略有不同。此函数的定义格式如下所示。



```
query (SQLiteDatabase db, String[] projectionIn, String selection,
String[] selectionArgs, String groupBy, String having, String sortOrder,
String limit)
```

各个参数的具体说明如下所示。

- ◆ 第 1 个参数为要查询的数据库实例。
- ◆ 第 2 个参数是一个字符串数组，里边的每一项代表了需要返回的列名。
- ◆ 第 3 个参数相当于 SQL 语句中的 where 部分。
- ◆ 第 4 个参数是一个字符串数组，里边的每一项依次替代在第三个参数中出现的问号 (?)。
- ◆ 第 5 个参数相当于 SQL 语句当中的 groupby 部分。
- ◆ 第 6 个参数相当于 SQL 语句当中的 having 部分。
- ◆ 第 7 个参数描述是怎么进行排序。
- ◆ 第 8 个参数相当于 SQL 当中的 limit 部分，控制返回的数据的个数。

在 DiaryContentProvider 类里边还有 2 个重要的方法，分别是 getType 和 getFormateCreatedDate。后者的功能是根据时间得到一个特定格式的字符串。前者是一个必须要重写的方法，重写 getType 方法的主要代码如下所示。

```
public String getType(Uri uri) {
    switch (sUriMatcher.match(uri)) {
        case DIARIES:
            return DiaryColumns.CONTENT_TYPE;
        case DIARY_ID:
            return DiaryColumns.CONTENT_ITEM_TYPE;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
    }
}
```

此方法用于返回一个所给 Uri 的指定数据的 MIME 类型。它的返回值如果以 vnd.Android.cursor.item 开头，那么就代表这个 Uri 指定的是单条数据。如果是以 vnd.Android.cursor.dir 开头的话，那么说明这个 Uri 指定的是全部数据。

step 08 单击 MENU 键后会在主 Activity 中执行如下代码。

```
@Override public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, MENU_ITEM_INSERT, 0, R.string.menu_insert);
    return true;
}
```

执行后按下 Menu 键会弹出“添加日记”选项，如图 4-19 所示。

单击【添加日记】选项后进入如图 4-20 所示界面。

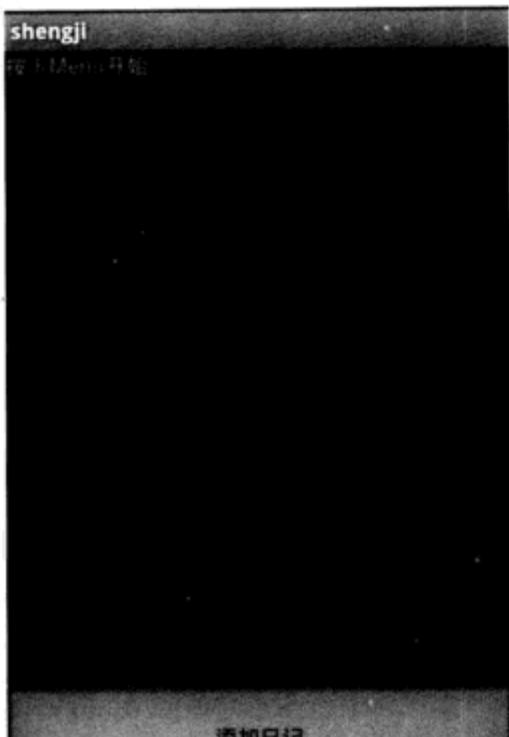


图4-19 单击Menu后的运行界面

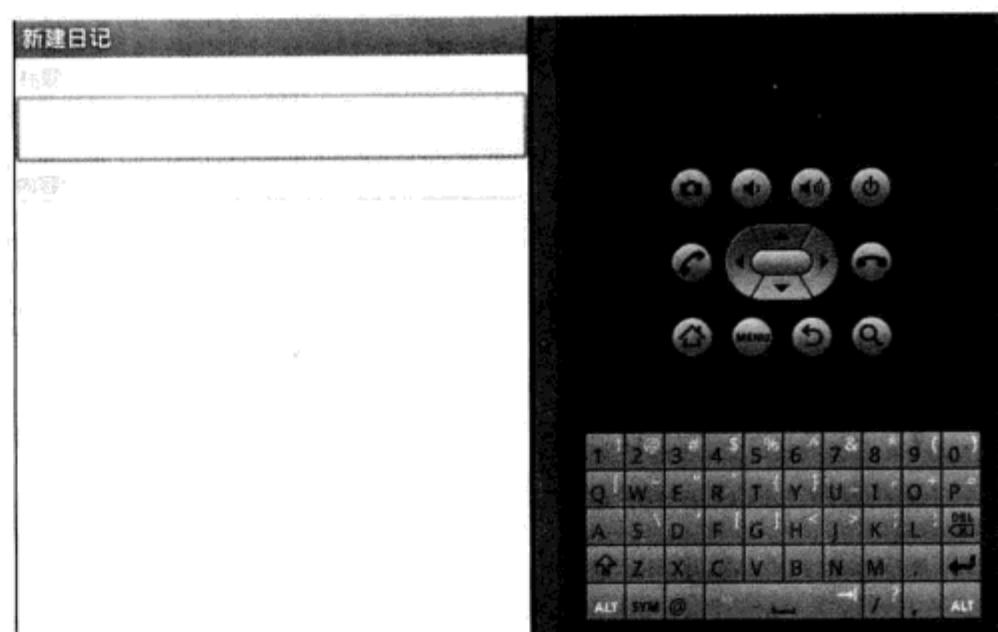


图4-20 添加日记界面

step 09 在文件 ActivityDiaryEditor.java 中定义了 ActivityDiaryEditor 类，在里面有处理两种方法进入日记本程序界面。一种是通过新建日记进入此界面，另一种是经过编辑日记进入此日记运行界面。下面是在文件 ActivityDiaryEditor 中使用的方法 onCreate()，其代码如下所示。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setTheme(android.R.style.Theme_Black);  
    final Intent intent = getIntent();  
    final String action = intent.getAction();  
    setContentView(R.layout.diary_edit);  
  
    mTitleText = (EditText) findViewById(R.id.title);  
    mBodyText = (EditText) findViewById(R.id.body);  
    confirmButton = (Button) findViewById(R.id.confirm);  
  
    if (EDIT_DIARY_ACTION.equals(action)) { // 编辑日记  
        mState = STATE_EDIT;  
        mUri = intent.getData();  
        mCursor = managedQuery(mUri, PROJECTION, null, null, null);  
        mCursor.moveToFirst();  
        String title = mCursor.getString(1);  
        mTitleText.setTextKeepState(title);  
        String body = mCursor.getString(2);  
        mBodyText.setTextKeepState(body);  
    }  
}
```



```

        setResult(RESULT_OK, (new Intent()).setAction(mUri.
toString()));
        setTitle("编辑日记");
    } else if (INSERT_DIARY_ACTION.equals(action)) { // 新建日记
        mState = STATE_INSERT;
        setTitle("新建日记");
    } else {
        Log.e(TAG, "no such action error");
        finish();
        return;
    }
    confirmButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            if (mState == STATE_INSERT) {
                insertDiary();
            } else {
                updateDiary();
            }
            Intent mIntent = new Intent();
            setResult(RESULT_OK, mIntent);
            finish();
        }
    });
}

```

在上述代码中，通过 getIntent() 得到启动当前 Activity 的那个 Intent。

通过 intent.getAction() 得到这个 Intent 的动作（Action）。在此操作中，当前的 Activity 是通过单击先前的【新建日记】按钮进来的，所以很有必要了解文件 shengji4.java 中方法 onOptionsItemSelected 的动作（Action）是怎么设置的，其代码如下所示。

```

Intent intent0 = new Intent(this, ActivityDiaryEditor.class);
intent0.setAction(ActivityDiaryEditor.INSERT_DIARY_ACTION);
intent0.setData(getIntent().getData());
startActivity(intent0);

```

从上述代码可以看到，通过语句 intent0.setAction (ActivityDiaryEditor.INSERT_DIARY_ACTION) 设置的 action 是 ActivityDiaryEditor.INSERT_DIARY_ACTION 实现的。

- 在 ActivityDiaryEditor 中的方法 onCreate() 中，当动作（Action）为 INSERT_DIARY_ACTION 时候执行 mState = STATE_INSERT，也就是标记当前的状态为插入状态。
- 如果当前的动作（Action）是 EDIT_DIARY_ACTION，那么当前就是编辑状态：mState = STATE_EDIT。



当运行程序填充数据后单击【确定】按钮，执行单击监听方法 onClick()，此方法可以根据不同的状态执行插入或者更新数据的操作。

写完日记并单击【确定】按钮后，程序将执行插入操作，具体代码如下所示。

```
private void insertDiary() {  
    String title = mTitleText.getText().toString();  
    String body = mBodyText.getText().toString();  
    ContentValues values = new ContentValues();  
    values.put(Diary.DiaryColumns.CREATED, DiaryContentProvider  
        .getFormateCreatedDate());  
    values.put(Diary.DiaryColumns.TITLE, title);  
    values.put(Diary.DiaryColumns.BODY, body);  
    getContentResolver().insert(Diary.DiaryColumns.CONTENT_URI,  
        values);  
}
```

在上述代码中，首先通过方法 getText() 读取编辑框中的数据都，然后将要插入的数据都放到一个 ContentValues 实例当中。调用 getContentResolver() 得到当前应用的一个 ContentResolver 的实例。

step 10 当单击确定按钮后，程序运行出现如图 4-21 所示的界面。按方向键中的向下键将焦点移动到这条数据上，然后单击 Menu 键会出现如图 4-22 所示界面。

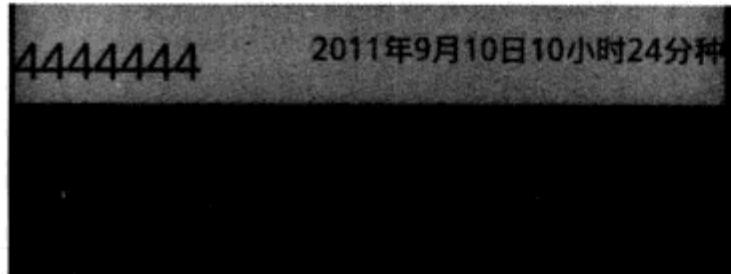


图 4-21 已经插入的数据

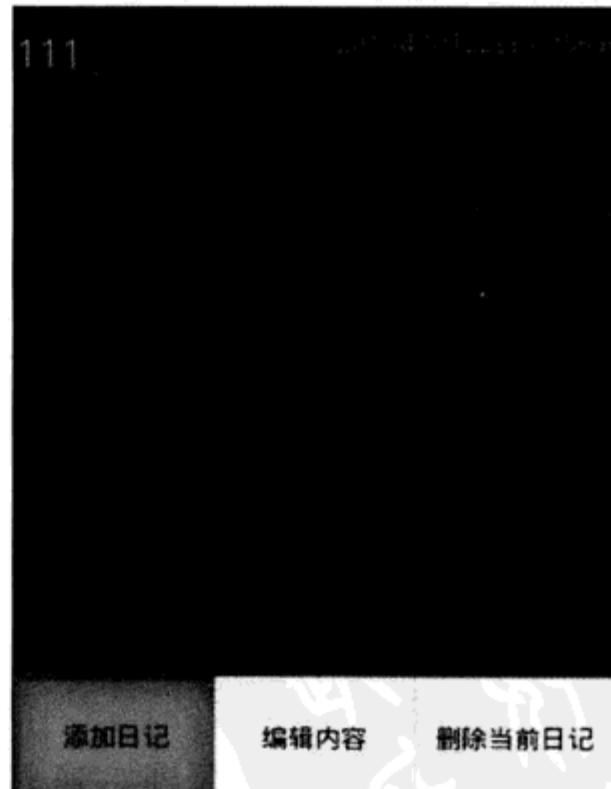


图 4-22 单击 Menu 键界面

图 4-22 所示的这个界面和刚才讲到的单击 Menu 键出现的界面一样，此界面的生成代码如下所示。

```
/* 在每一次menu生成的时候前都会调用这个方法，在这个方法里边可以动态的修改生成  
的menu。 */  
public boolean onPrepareOptionsMenu(Menu menu) {
```

```

super.onCreateOptionsMenu(menu);
final boolean haveItems = getListAdapter().getCount() > 0;
if (haveItems) {
    // 如果选中一个Item的话
    if (getListView().getSelectedItemId() > 0) {
        menu.removeGroup(1);
        Uri uri = ContentUris.withAppendedId(getIntent().getData(),
            getSelectedItemId());
        Intent intent = new Intent(null, uri);
        menu.add(1, MENU_ITEM_EDIT, 1, "编辑").setIntent(intent);
        menu.add(1, MENU_ITEM_DELETE, 1, "删除");
    }
} else{
    menu.removeGroup(1);
}
return true;
}

```

在上述代码中，和这个 ListView 相关联的 ListAdapter 里边元素的个数大于零的时候 haveItems 为真，否则为假。具体说明如下：

- ◆ menu.removeGroup(1)：如果 menu 里边有分组为 1 组的子项的话，那么就先全部删除。
- ◆ getIntent().getData()：得到的 Uri 是 DiaryColumns.CONTENT_URI。
- ◆ ContentUris.withAppendedId(getIntent().getData(),getSelectedItemId())：生成一个和 ListView 中获得焦点这一项的数据的 Uri。
- ◆ menu.add(1, MENU_ITEM_EDIT, 1, " 编辑内容 ").setIntent(intent)：在 menu 当中增加了一项“编辑内容”，并且这一项和一个 intent 关联，这个 intent 主要用于携带数据，它里边携带的就是一个 Uri 的数据，在下边的 onOptionsItemSelected() 函数当中会用到。
- ◆ menu.add(1, MENU_ITEM_DELETE, 1, " 删除当前日记 ")：用于增加另外一项。

如果 haveItems 为假，也就是当前和这个 ListView 相关联的 ListAdapter 里边元素的个数为零，即数据显示在 ListView 上的时候，单击 Menu 按钮的话，如果 Menu 当中还有第 1 分组的子项的话，就给删除了，实现语句为：menu.removeGroup(1)。

当单击 MENU 按钮时，在 Activity 中可能有如下两个回调的函数。

- ◆ onOptionsItemSelected()：此函数只在第一次在当前应用当中单击 Menu 键的时候回调，以后再不回调。
- ◆ onPrepareOptionsMenu()：此函数在每次单击 Menu 键后显示 menu 的时候被系统回调，每次 menu 显示前都会回调此函数。我们一般根据条件改变 Menu 显示的逻辑都放在这个函数里边。

当单击 MENU 键后会弹出 3 个选项，单击其中的某一个按钮会触发 Android 系统回

调 `onOptionsItemSelected()` 函数，此函数的实现代码如下所示。

```
@Override public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case MENU_ITEM_INSERT:  
            Intent intent0 = new Intent(this, ActivityDiaryEditor.class);  
            intent0.setAction(ActivityDiaryEditor.INSERT_DIARY_ACTION);  
            intent0.setData(getIntent().getData());  
            startActivity(intent0);  
            return true; //插入一条数据  
        case MENU_ITEM_EDIT:  
            Intent intent = new Intent(this, ActivityDiaryEditor.class);  
            intent.setData(item.getIntent().getData()); intent.  
            setAction(ActivityDiaryEditor.EDIT_DIARY_ACTION);  
            startActivity(intent); return true; //编辑当前数据内容  
        case MENU_ITEM_DELETE:  
            Uri uri = ContentUris.withAppendedId(getIntent().getData(),  
            getListView().getSelectedItemId());  
            getContentResolver().delete(uri, null, null);  
            renderListView();  
    }  
    return super.onOptionsItemSelected(item);  
}
```

当用户单击添加新日记按钮后，程序新建一个跳转Activity的intent0，并且设置了Action和data，这两个部分在程序跳转到ActivityDiaryEditor后会用到。

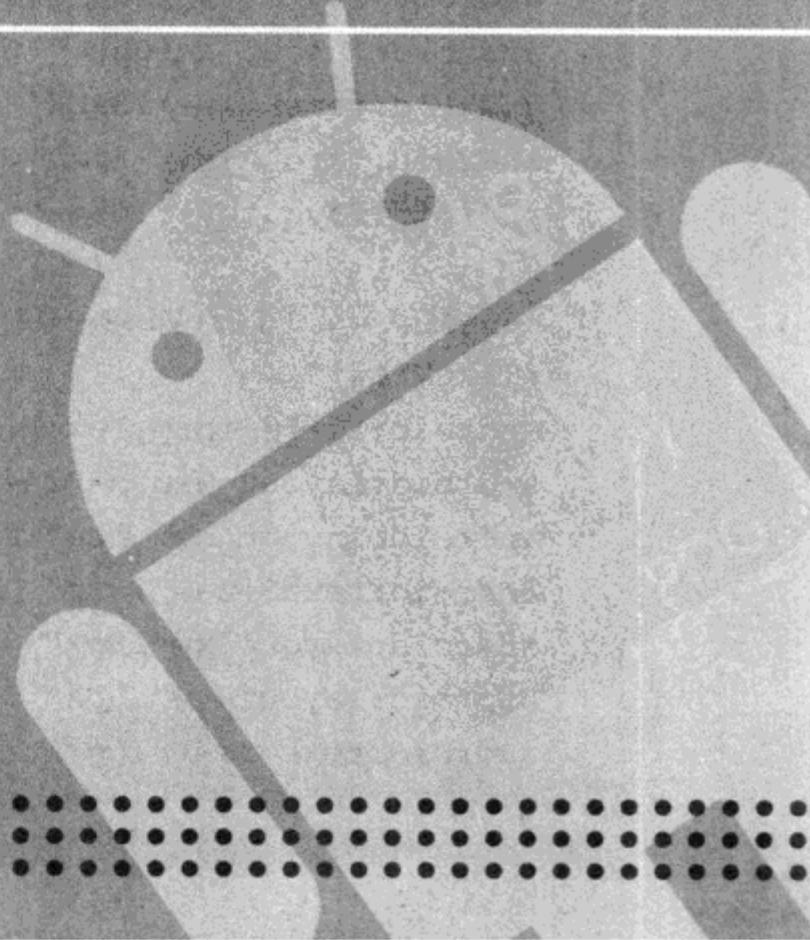
当用户单击编辑按钮后，程序也新建了一个跳转Activity的intent，并且也设置了Action和data。同样，这两部分在程序跳转到ActivityDiaryEditor后会用到。需要注意的是，通过item.getIntent().getData()得到了所需要的Uri。

当用户单击删除按钮后，程序通过ContentUris.withAppendedId(getIntent().getData(), getListView().getSelectedItemId())先得到需要删除数据的Uri，然后得到当前的ContentResolvor，然后调用它的delete()方法删除数据。删除数据后调用函数renderListView()来刷新ListView。



通信领域实战演练

作为一个移动手机设备，当然需要具备交互通信的功能。手机中的交互通信方式有多种，例如常见的通话、短信、邮件和蓝牙等。在本章的内容中，将通过具体的实例来详细讲解 Android 中交互式通信应用的具体实现流程。





5.1 拨号、邮件和网址处理

本实例的功能是提供一个文本输入框，当用户输入电话号码后，可以进行拨号处理；当输入一个网址后，可以登录到这个地址；当输入邮箱后，就发送邮件。此功能是通过 Linkify 对象实现的，Linkify 可以让系统动态获取，并作出一个判断，判断是电话、邮箱还是网址，并随之做出对应的处理。通过 TextView 和 EditText 交互，就可以在此显示自己输入的数据值。具体实现上，只需重写 EditText.setOnKeyListener() 即可实现。

本实例的源码保存在【光盘\daima\第 5 章\bo】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <!--建立第一个TextView-->
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="244px"
        android:layout_height="76px"
        android:textSize="18sp"
        android:layout_x="18px"
        android:layout_y="105px"
    />
    <!--建立第二个TextView-->
    <TextView
        android:id="@+id/myTextView2"
        android:layout_width="244px"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_x="18px"
        android:layout_y="20px"
        android:text="@string/str_text"
```

```
    />
    <!--建立一个EditText-->
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="262px"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_x="14px"
        android:layout_y="41px"
    />
</AbsoluteLayout>
```

通过上述代码，插入了3个TextView和1个EditText。

step 02 编写文件 strings.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">bo</string>
    <string name="app_name">bo</string>
    <string name="str_button1">按下</string>
    <string name="str_text">请输入电话 号码或E-mail地址 或 网址</string>
</resources>
```

step 03 编写主程序文件 bo.java，具体代码如下。

```
package irdc.bo;

import irdc.practicel.R;
import android.app.Activity;
import android.os.Bundle;
import android.text.util.Linkify;
import android.view.KeyEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class practicel extends Activity
{
    private TextView mTextView01;
    private EditText mEditText01;

    /** Called when the activity is first created. */
}
```

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

        mTextView01 = (TextView) findViewById(R.id.myTextView1);
        mEditText01 = (EditText) findViewById(R.id.myEditText1);

        mEditText01.setOnKeyListener(new EditText.OnKeyListener()
        {
            @Override
            public boolean onKey(View arg0, int arg1, KeyEvent arg2)
            {
                // TODO Auto-generated method stub
                mTextView01.setText(mEditText01.getText());
                /*判断输入的类型是何种，并与系统连接*/
                Linkify.addLinks(mTextView01, Linkify.WEB_URLS|Linkify.
                    EMAIL_ADDRESSES|Linkify.PHONE_NUMBERS);
                return false;
            }
        });
    }
}
```

在上述代码中，设置了 Linkify 的处理事件，Linkify 是在创建 TextView 之后设置的，否则设置的 RE 不会起效果。设置了 mTextView01 拥有自动链接功能，这样就能来到指定的页面。

执行后的效果如图 5-1 所示，用户可以输入电话号码、邮箱地址或网址，输入后可显示对应的操作。例如输入 www.163.com 后，会在下面自动显示输入的网址，如图 5-2 所示；当单击网址后会转到对应的页面，如图 5-3 所示。

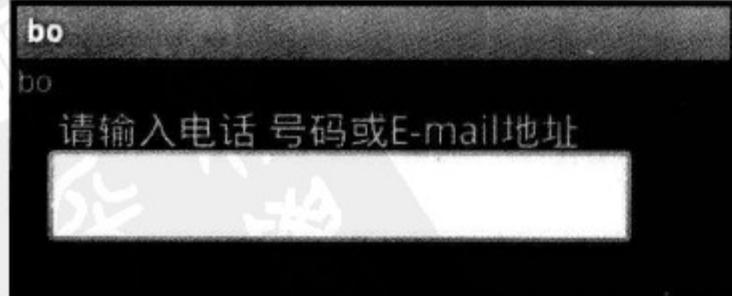


图5-1 执行效果

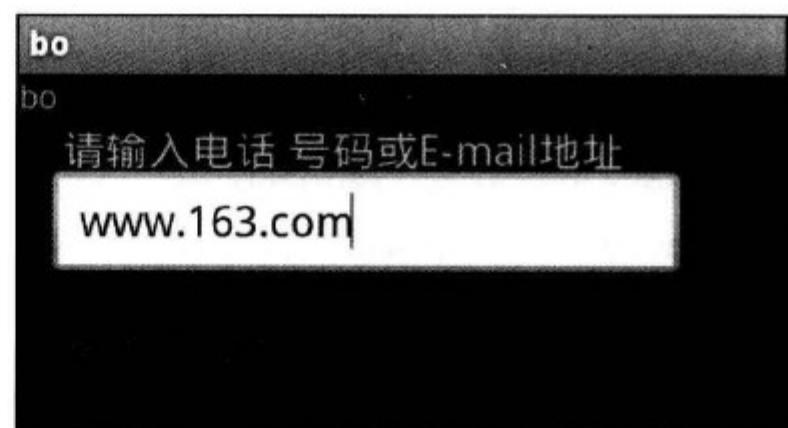


图5-2 自动显示输入数据



图5-3 转到指定网址

5.2 拨打电话

在具体拨打电话时，首先要在 AndroidManifest 中添加 uses-permission，这样就实现了对拨打电话的声明；然后通过自定义的 Intent 对象，通过“ACTION_CALL”键和 Uri.parse() 方法将用户输入的电话号码写入；在最后，通过 startActivity 方法即可完成程序拨打电话的功能。

本实例实现了一个基本的手机拨打电话的过程，首先使用了一个 EditText 用于获取输入的电话号码，当单击 Button 后执行拨打电话程序，并且通过自定义的 isPhoneNumberValid(String phoneNumber) 来确保用户输入的是合法的电话号码。

本实例的源码保存在【光盘\daima\第5章\boda】，具体实现流程如下所示。

step 01 编写主布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/myWidget1"
    android:background="@drawable/white"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="18sp"
    android:layout_x="12px"
    android:layout_y="15px"
  >
</EditText>
<Button
    android:id="@+id/myButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button1"
    android:layout_x="10px"
    android:layout_y="65px"
  >
</Button>
</AbsoluteLayout>
```

step 02 编写主程序文件是 boda.java，其具体实现流程如下所示。

- ◆ 引用类和对象，具体代码如下所示。

```
package irdc.boda;

import android.app.Activity;
import android.content.Intent;
/*引用Uri类才能使用Uri.parse()*/
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
/*引用 widget.Button才能声明使用Button对象*/
import android.widget.Button;
import android.widget.Toast;
/*引用 widget.EditText才能声明使用EditText对象*/
import android.widget.EditText;
/*引用 java.util.regex才能使用Regular Expression*/
import irdc.boda.R;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class boda extends Activity
{
  /*声明Button与EditText对象名称*/
  private Button mButton1;
```

```

private EditText mEditText1;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*通过findViewById构造器来构造EditText与Button对象*/
    mEditText1 = (EditText) findViewById(R.id.myEditText1);
    mButton1 = (Button) findViewById(R.id.myButton1);
}

```

- ◆ 设置 Button 对象的 OnClickListener 来聆听 OnClick 事件，具体代码如下所示。

```

mButton1.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        try
        {
            /*取得EditText中用户输入的字符串*/
            String strInput = mEditText1.getText().toString();
            if (isPhoneNumberValid(strInput)==true)
            {
                /*建构一个新的Intent
                运行action.CALL的常数与通过Uri将字符串带入*/
                Intent myIntentDial = new
                Intent
                (
                    "android.intent.action.CALL",
                    Uri.parse("tel:"+strInput)
                );
                /*在startActivity()方法中
                带入自定义的Intent对象以运行拨打电话的工作.*/
                startActivity(myIntentDial);
                mEditText1.setText("");
            }
            else
            {
                mEditText1.setText("");
                Toast.makeText(
                    boda.this, "输入的电话格式不符",
                    Toast.LENGTH_SHORT).show();
            }
        }
    }
});

```



```
        Toast.LENGTH_LONG).show();
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
}
});
```

- ◆ 定义 `isPhoneNumberValid(String phoneNumber)` 方法，检查字符串是否为电话号码的方法，并返回 true or false 的判断值。具体代码如下所示。

```
public static boolean isPhoneNumberValid(String phoneNumber)
{
    boolean isValid = false;
    /* 可接受的电话格式有：
     * ^\\(? : 可以使用 "(" 作为开头
     * (\\d{3})：紧接着三个数字
     * \\)? : 可以使用 ")" 接续
     * [- ]? : 在上述格式后可以使用具选择性的 "-".
     * (\\d{4}) : 再紧接着三个数字
     * [- ]? : 可以使用具选择性的 "--" 接续.
     * (\\d{4})$ : 以四个数字结束.
     * 可以比较下列数字格式：
     * (123) 456-78900, 123-4560-7890, 12345678900, (123)-4560-7890
     */
    String expression = "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{5})$";
    String expression2 ="^\\(?(\\d{3})\\)?[- ]?(\\d{4})[- ]?(\\d{4})$";
    CharSequence inputStr = phoneNumber;
    /*创建Pattern*/
    Pattern pattern = Pattern.compile(expression);
    /*将Pattern 以参数传入Matcher作Regular expression*/
    Matcher matcher = pattern.matcher(inputStr);
    /*创建Pattern2*/
    Pattern pattern2 =Pattern.compile(expression2);
    /*将Pattern2 以参数传入Matcher2作Regular expression*/
    Matcher matcher2= pattern2.matcher(inputStr);
    if(matcher.matches() || matcher2.matches())
    {
        isValid = true;
    }
}
```

```

    return isValid;
}
}

```

程序执行后的效果如图 5-4 所示；如果输入的电话号码不规范则输出对应的提示，如图 5-5 所示；当输入规范的号码并单击【拨打】按钮后，会实现拨号处理并显示拨号界面，如图 5-6 所示。

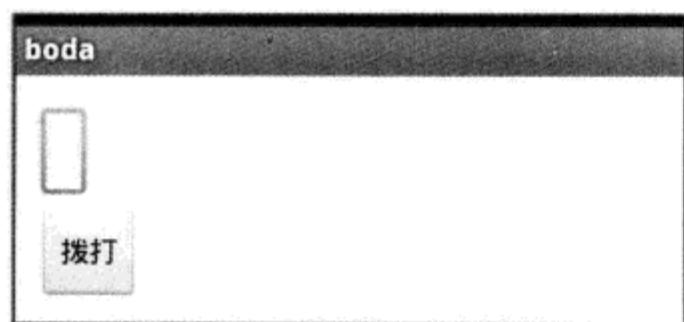


图 5-4 执行效果

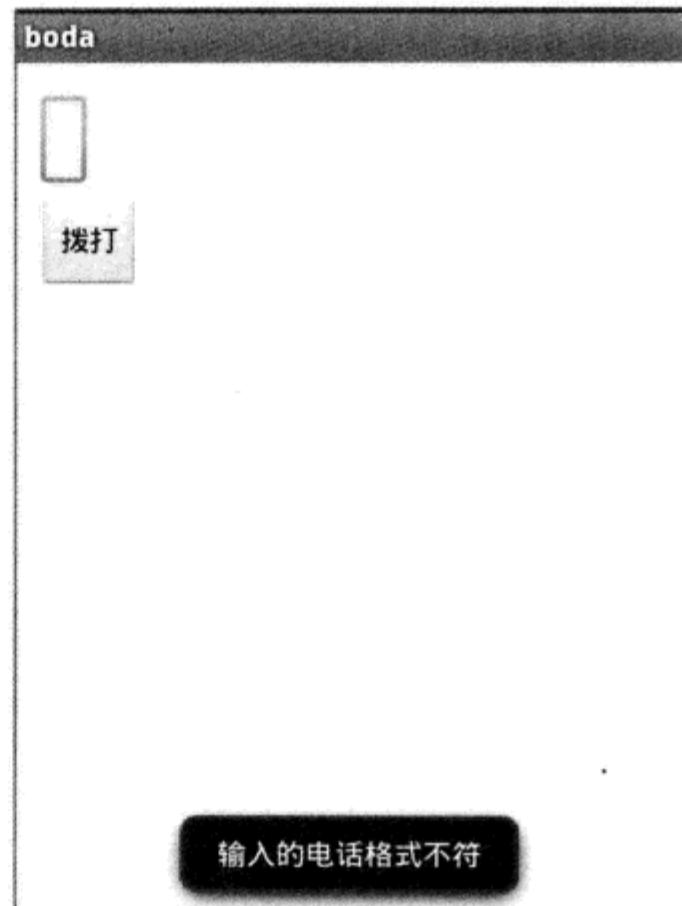


图 5-5 自动显示输入数据

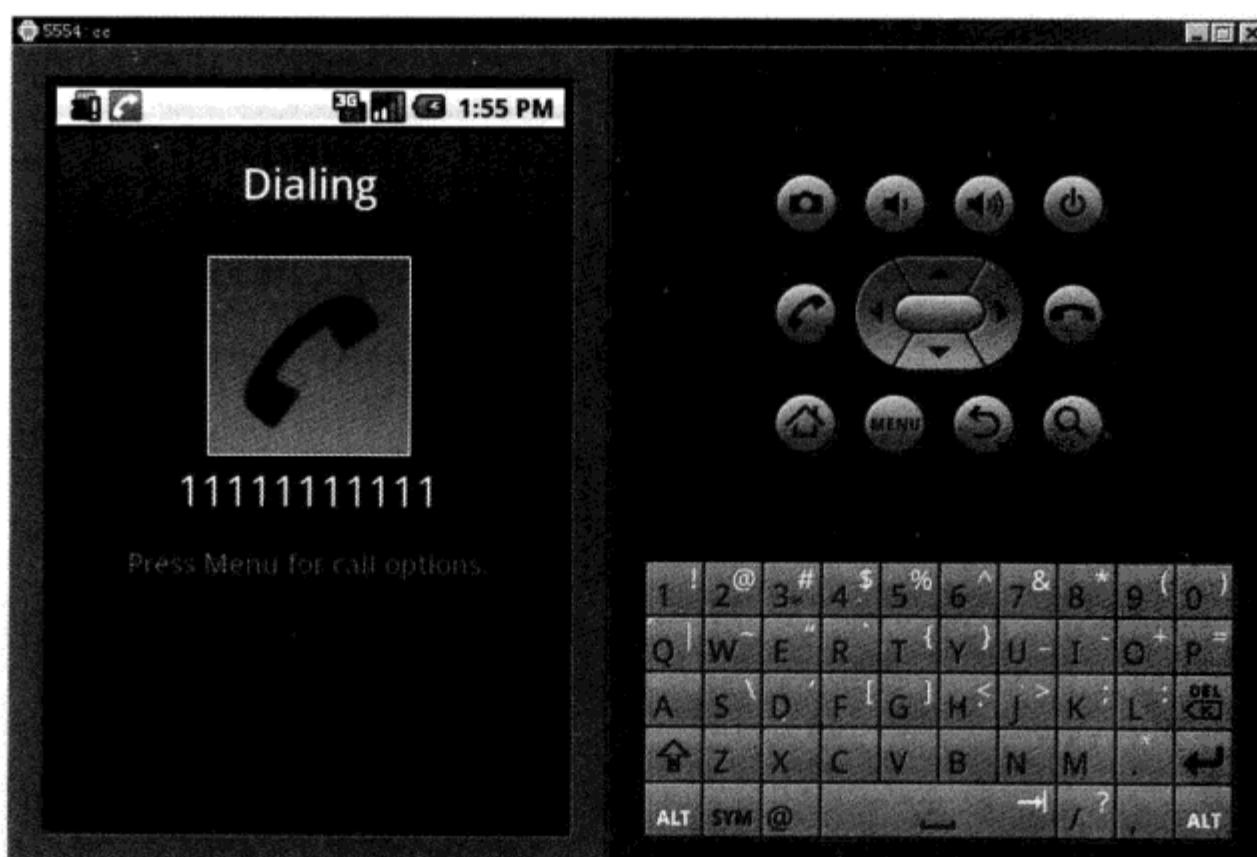


图 5-6 拨打界面



5.3 发送短信交互

作为一个手机，除了拨打电话外，发送短信也是另外一个极为重要的功能。在具体实现上，是通过 SmsManager 对象的 sendTextMessage() 方法来完成的。在 sendTextMessage() 方法中要传入 5 个值，分别是：收件人地址 String、发送地址 String、正文 String、发送服务 PendingIntent 和送达服务服务 PendingIntent。

在本实例中，定义了 2 个 EditText 控件，用于分别获取收信人电话和短信正文，设置了判断手机号码规范化的方法，并且设置了短信的字数不超过 70 个字符。本实例的源码保存在【光盘\daima\第 5 章\duan】，具体实现流程如下所示。

step 01 先编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget1"
    android:background="@drawable/white"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/widget27"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_textview"
        android:textSize="16sp"
        android:layout_x="0px"
        android:layout_y="12px"
    >
    </TextView>
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="18sp"
        android:layout_x="60px"
        android:layout_y="2px"
    >
    </EditText>
    <EditText
```



```
    android:id="@+id/myEditText2"
    android:layout_width="fill_parent"
    android:layout_height="223px"
    android:text=""
    android:textSize="18sp"
    android:layout_x="0px"
    android:layout_y="52px"
  >
</EditText>
<Button
    android:id="@+id/myButton1"
    android:layout_width="162px"
    android:layout_height="wrap_content"
    android:text="@string/str_button1"
    android:layout_x="80px"
    android:layout_y="302px"
  >
</Button>
</AbsoluteLayout>
```

step 02 本实例的主程序文件是 duan.java，其具体代码如下所示。

```
package irdc.duan;

import android.app.Activity;
/*引用PendingIntent类才能使用getBroadcast()*/
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
/*引用telephony.gsm.SmsManager类才能使用sendTextMessage()*/
import android.telephony.gsm.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import irdc.duan.R;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class duan extends Activity
{
    /*声明变量一个Button与两个EditText*/
    private Button mButton1;
    private EditText mEditText1;
```



```
private EditText mEditText2;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*
     * 通过findViewById构造器来建构
     * EditText1,EditText2与Button对象
     */
    mEditText1 = (EditText) findViewById(R.id.myEditText1);
    mEditText2 = (EditText) findViewById(R.id.myEditText2);
    mButton1 = (Button) findViewById(R.id.myButton1);

    /*将默认文字加载EditText中*/
    mEditText1.setText("请输入号码");
    mEditText2.setText("请输入内容!!");

    /*设置onClickListener 让用户点击EditText时做出反应*/
    mEditText1.setOnClickListener(new EditText.OnClickListener()
    {
        public void onClick(View v)
        {
            /*点击EditText时清空正文*/
            mEditText1.setText("");
        }
    });
    /*设置onClickListener 让用户点击EditText时做出反应*/
    mEditText2.setOnClickListener(new EditText.OnClickListener()
    {
        public void onClick(View v)
        {
            /*点击EditText时清空正文*/
            mEditText2.setText("");
        }
    });
    /*设置onClickListener 让用户点击Button时做出反应*/
    mButton1.setOnClickListener(new Button.OnClickListener()
```

```
    @Override
    public void onClick(View v)
    {
        /*由EditText1取得短信收件人电话*/
        String strDestAddress = mEditText1.getText().toString();
        /*由EditText2取得短信文字内容*/
        String strMessage = mEditText2.getText().toString();
        /*建构一取得default instance的 SmsManager对象 */
        SmsManager smsManager = SmsManager.getDefault();

        // TODO Auto-generated method stub
        /*检查收件人电话格式与短信字数是否超过70字符*/
        if(isPhoneNumberValid(strDestAddress)==true &&
           iswithin70(strMessage)==true)
        {
            try
            {
                /*
                 * 两个条件都检查通过的情况下,发送短信
                 * 先建构一PendingIntent对象并使用getBroadcast()广播
                 * 将PendingIntent, 电话, 短信文字等参数
                 * 传入sendTextMessage()方法发送短信
                 */
                PendingIntent mPI = PendingIntent.getBroadcast
                    (duan.this, 0, new Intent(), 0);
                smsManager.sendTextMessage
                    (strDestAddress, null, strMessage, mPI, null);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            Toast.makeText
            (
                duan.this,"送出成功!!" ,
                Toast.LENGTH_SHORT
            ).show();
            mEditText1.setText("");
            mEditText2.setText("");
        }
        else
        {
```



```
/* 电话格式与短信文字不符合条件时,以Toast提醒 */
if (isPhoneNumberValid(strDestAddress)==false)
{ /*且字数超过70字符*/
    if(iswithin70(strMessage)==false)
    {
        Toast.makeText
        (
            duan.this,
            "电话号码格式错误+短信内容超过70字,请检查!!",
            Toast.LENGTH_SHORT
        ).show();
    }
    else
    {
        Toast.makeText
        (
            duan.this,
            "电话号码格式错误,请检查!!",
            Toast.LENGTH_SHORT
        ).show();
    }
}
/*字数超过70字符*/
else if (iswithin70(strMessage)==false)
{
    Toast.makeText
    (
        duan.this,
        "短信内容超过70字,请删除部分内容!!",
        Toast.LENGTH_SHORT
    ).show();
}
}
});

}

/*检查字符串是否为电话号码的方法,并返回true or false的判断值*/
public static boolean isPhoneNumberValid(String phoneNumber)
{
    boolean isValid = false;
    /* 可接受的电话格式有:
     * ^\\(? : 可以使用 "(" 作为开头
    
```

```

* (\d{3})：紧接着三个数字
* \)?：可以使用")"接续
* [-]？：在上述格式后可以使用具选择性的 "-".
* (\d{3})：再紧接着三个数字
* [-]？：可以使用具选择性的 "-" 接续.
* (\d{5})$：以五个数字结束.
* 可以比较下列数字格式：
* (123) 456-7890, 123-456-7890, 1234567890, (123)-456-7890
*/
String expression =
"^\(\d{3}\)\)?[-]?\d{3}[-]?\d{5}$";

/* 可接受的电话格式有：
* ^\(?：可以使用 "(" 作为开头
* (\d{3})：紧接着三个数字
* \)?：可以使用")"接续
* [-]？：在上述格式后可以使用具选择性的 "-".
* (\d{4})：再紧接着四个数字
* [-]？：可以使用具选择性的 "-" 接续.
* (\d{4})$：以四个数字结束.
* 可以比较下列数字格式：
* (02) 3456-7890, 02-3456-7890, 0234567890, (02)-3456-7890
*/
String expression2=
"^\(\d{3}\)\)?[-]?\d{4}[-]?\d{4}$";

CharSequence inputStr = phoneNumber;
/*创建Pattern*/
Pattern pattern = Pattern.compile(expression);
/*将Pattern 以参数传入Matcher作Regular expression*/
Matcher matcher = pattern.matcher(inputStr);
/*创建Pattern2*/
Pattern pattern2 = Pattern.compile(expression2);
/*将Pattern2 以参数传入Matcher2作Regular expression*/
Matcher matcher2 = pattern2.matcher(inputStr);
if(matcher.matches() || matcher2.matches())
{
    isValid = true;
}
return isValid;
}

public static boolean iswithin70(String text)

```



```
    {
        if (text.length() <= 70)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

在上述代码中，通过 PendingIntent.getBroadcast 方法自定义了 PendingIntent 并进行 Broadcast，然后使用 SmsManager.getDefault() 预先构建的 SmsManager，使用 sendTextMessage() 方法将有关的数据以参数形式带入，这样即可完成发短信的任务。上述代码的实现流程如下所示。

- ◆ 分别引用 dingIntent 类和 telephony.gsm.SmsManager 类；
- ◆ 声明变量 1 个 Button 和 2 个 EditText，EditText 供获取输入收信人电话号码和短信内容，Button 按钮用于激活发信处理程序；
- ◆ 设置 onClickListener() 方法，用于响应用户点击 EditText 时做出反应；
- ◆ 设置 onClickListener 方法，用于用户点击 Button 时做出反应；
- ◆ 检查收件人电话格式与短信字数是否超过 70 字符，然后通过 smsManager.sendTextMessage 实现发送短信处理。

执行后的效果如图 5-7 所示，输入手机号码，编写短信内容后，单击【开始发送】按钮后即可完成短信发送功能，系统会提示成功信息，如图 5-8 所示。

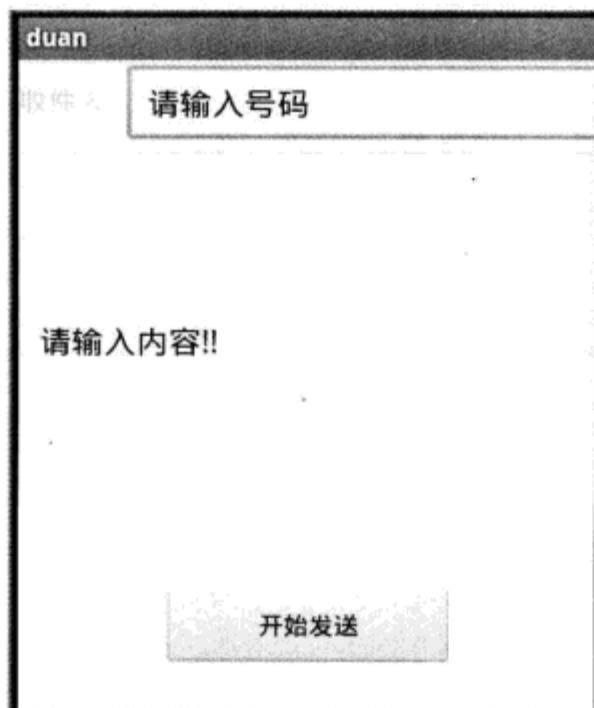


图5-7 初始效果



图5-8 发送成功

如果短信内容和收信人号码格式不规范，会输出对应的错误提示。



5.4 发送邮件

作为一个手机，除了拨打电话和发送短信外，发送邮件也是另外一个极为重要的功能。在具体实现上，邮件的收发过程是通过 Android 内置的 Gmail 程序实现的，而并不是使用 SMTP 的 Protocol 实现的。为了确保邮件能够发出，必须在收件人字段上输入标准的邮件地址格式，如果格式不规范，则发送按钮处于不可用状态。

在本实例中，自定义了 Intent，并使用 `Android.content.Intent.ACTION_SEND` 的参数来通过手机寄发 E-mail 的服务，整个过程还算简单。本实例的源码保存在【光盘\daima\第 5 章\youjian】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget34"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_receive"
        android:layout_x="60px"
        android:layout_y="22px"
        >
    </TextView>
    <TextView
        android:id="@+id/myTextView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_cc"
        android:layout_x="60px"
        android:layout_y="82px"
        >
    </TextView>
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_x="120px"
    android:layout_y="12px"
  >
</EditText>
<EditText
    android:id="@+id/myEditText2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_x="120px"
    android:layout_y="72px"
  >
</EditText>

<TextView
    android:id="@+id/myTextView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_subject"
    android:layout_x="60px"
    android:layout_y="142px"
  >
</TextView>
<EditText
    android:id="@+id/myEditText3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_x="120px"
    android:layout_y="132px"
  >
</EditText>
<EditText
    android:id="@+id/myEditText4"
    android:layout_width="fill_parent"
    android:layout_height="209px"
    android:textSize="18sp"
    android:layout_x="0px"
    android:layout_y="202px"
  >
</EditText>
```



```

<Button android:id="@+id/myButton1" android:layout_width="wrap_content"
        android:layout_height="124px" android:text="@string/str_button"
        android:layout_x="0px" android:layout_y="2px">
</Button>
</AbsoluteLayout>

```

step 02 编写主程序文件是 youjian.java，其具体实现流程如下所示。

- ◆ 引用 content.Intent 类打开 email client，具体代码如下所示。

```

package irdc.youjian;

import irdc.youjian.R;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
import android.app.Activity;
/*必须引用content.Intent类来打开email client*/
import android.content.Intent;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

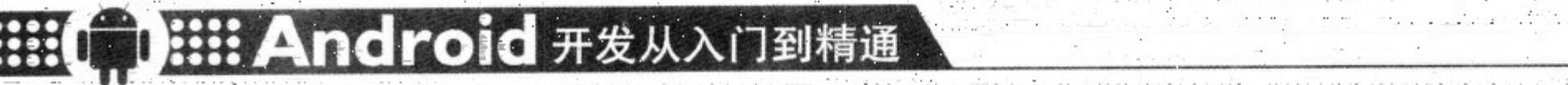
```

- ◆ 声明四个 EditText 一个 Button 以及四个 String 变量，用于输入邮箱地址、邮件主体、副本，具体代码如下所示。

```

public class youjian extends Activity
{
    /*声明四个EditText一个Button以及四个String变量*/
    private EditText mEditText01;
    private EditText mEditText02;
    private EditText mEditText03;
    private EditText mEditText04;
    private Button mButton01;
    private String[] strEmailReciver;
    private String strEmailSubject;
    private String[] strEmailCc;
    private String strEmailBody ;
    /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {

```



```
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
/*通过findViewById构造器来建构Button对象*/
mButton01 = (Button)findViewById(R.id.myButton1);
/*通过findViewById构造器来构造所有EditText对象*/
mButton01.setEnabled(false);
/*设置OnKeyListener,当key事件发生时进行反应*/
mEditText01 = (EditText)findViewById(R.id.myEditText1);
mEditText02 = (EditText)findViewById(R.id.myEditText2);
mEditText03 = (EditText)findViewById(R.id.myEditText3);
mEditText04 = (EditText)findViewById(R.id.myEditText4);
```

- ◆ 定义 setOnKeyListener 方法，如果用户输入为正规 E-mail 文字，则按钮可用，反之则按钮不可用，具体代码如下所示。

```
/*若用户键入为正规E-mail文字，则enable 按钮 反之则disable 按钮*/
mEditText01.setOnKeyListener(new EditText.OnKeyListener()
{
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event)
    {
        // TODO Auto-generated method stub
        /*如果是邮件地址格式，则按钮可按下*/
        if(isEmail(mEditText01.getText().toString()))
        {
            mButton01.setEnabled(true);
        }
        else
        {
            mButton01.setEnabled(false);
        }
        return false;
    }
});
```

定义onClickListener响应按钮，当单击按钮后实现邮件发送处理。具体代码如下所示。

```
/*定义onClickListener 响应按钮*/
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        Intent mEmailIntent = new Intent(android.content.Intent.
ACTION_SEND);
```

```

        mEmailIntent.setType("plain/text");

        strEmailReciver = new String[]{mEditText01.getText().
        toString()};
        strEmailCc = new String[]{mEditText02.getText().toString()};
        strEmailSubject = mEditText03.getText().toString();
        strEmailBody = mEditText04.getText().toString();

        mEmailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,
        strEmailReciver);
        mEmailIntent.putExtra(android.content.Intent.EXTRA_CC,
        strEmailCc);
        mEmailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
        strEmailSubject);
        mEmailIntent.putExtra(android.content.Intent.EXTRA_TEXT,
        strEmailBody);
        startActivityForResult(Intent.createChooser(mEmailIntent,
        getResources().getString(R.string.str_message)));
    }
}
}
}

```

- ◆ 定义 isEmail(String strEmail) 方法，检查是否是规范的邮件地址格式。具体代码如下所示。

```

public static boolean isEmail(String strEmail)
{
    String strPattern = “^ [a-zA-Z] [\w\.-]* [a-zA-Z0-9] @ [a-zA-Z0-9]
    [\w\.-]* [a-zA-Z0-9] \. [a-zA-Z] [a-zA-Z\.\.]* [a-zA-Z] $” ;
    Pattern p = Pattern.compile(strPattern);
    Matcher m = p.matcher(strEmail);
    return m.matches();
}
}

```

执行后的效果如图 5-9 所示，输入手机号码，编写短信内容后，单击【发送邮件】按钮后即可完成短信发送功能，系统会提示成功信息，如图 5-10 所示。



注意 因为在Android模拟器中没有内置Gmail Client端程序，所以当使用本实例发送邮件后，会显示“*No application can perform this action*”的提示。但是如果在实体手机设备上，运行本实例程序后会调用内置的Gmail程序来发送邮件。

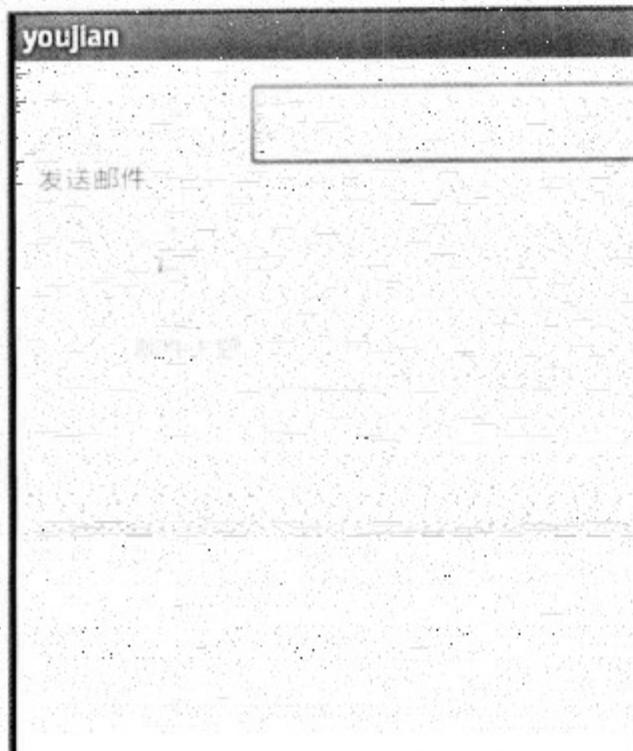


图5-9 初始效果



图5-10 发信中提示

5.5 实现震动效果

手机除了具备基本的通话和短信外，震动也是另外一个极为重要的功能。通过手机震动，能够帮助我们及时感知打来的电话或发来的短信。在本实例中，读者将会了解到触发手机震动事件的方法。Android 中的震动事件 Vibration，需要设置震动的时间长短和周期，并且设置单位是毫秒。如果要建立手机震动，则必须建立 Vibratior 对象，并通过调用 Vibrate 来实现震动目的。在 Vibratior 构造器中有 4 个参数，前三个用于设置震动大小，最后那个用于设置震动持续时间。在本范例中的震动方式是不同的，分为一直持续震动和只震动一次两种。

本实例的源码保存在【光盘\daima\第 5 章\zhendong】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
    <!-- 建立第一个TextView -->
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
    android:text="@string/hello"
/>
<!-- 建立第二个TextView -->
<TextView
    android:id="@+id/myTextView2"
    android:layout_width="127px"
    android:layout_height="35px"
    android:layout_x="90px"
    android:layout_y="33px"
    android:text="@string/str_text1"
/>
<!-- 建立第三个TextView -->
<TextView
    android:id="@+id/myTextView3"
    android:layout_width="127px"
    android:layout_height="35px"
    android:layout_x="90px"
    android:layout_y="115px"
    android:text="@string/str_text2"
/>
<!-- 建立第四个TextView -->
<TextView
    android:id="@+id/myTextView4"
    android:layout_width="127px"
    android:layout_height="35px"
    android:layout_x="90px"
    android:layout_y="216px"
    android:text="@string/str_text3"
/>
<!-- 建立第一个ToggleButton -->
<ToggleButton
    android:id="@+id/myToggleButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="29px"
    android:layout_y="31px"
/>
<!-- 建立第二个ToggleButton -->
<ToggleButton
    android:id="@+id/myToggleButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="29px"
```

```
    android:layout_y="114px"
    />
    <!-- 建立第三个ToggleButton -->
    <ToggleButton
        android:id="@+id/myTogglebutton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="29px"
        android:layout_y="214px"
    />
</AbsoluteLayout>
```

step 02 编写主程序文件是 zhendong.java，其具体实现流程如下所示。

- ◆ 设置 ToggleButton 的对象，检测 ToggleButton 是否被启动，如果【ON】按钮则启动震动模式，如果单击【OFF】按钮则关闭震动模式。具体代码如下所示。

```
package irdc.zhendong;

import irdc.zhendong.R;
import android.app.Activity;
import android.app.Service;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
import android.widget.ToggleButton;

public class zhendong extends Activity
{
    private Vibrator mVibrator01;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /*设置ToggleButton的对象*/
        mVibrator01 = (Vibrator) getApplication().getSystemService
            (Service.VIBRATOR_SERVICE);
```

```

final ToggleButton mtogglebutton1 =
(ToggleButton) findViewById(R.id.myToggleButton1);

final ToggleButton mtogglebutton2 =
(ToggleButton) findViewById(R.id.myToggleButton2);

final ToggleButton mtogglebutton3 =
(ToggleButton) findViewById(R.id.myToggleButton3);

```

- ◆ 设置短震动模式，通过 mVibrator01.vibrate(new long[]{100,10,100,1000},-1) 设置了此模式下的震动周期，具体代码如下所示。

```

/* 短震动 */
mtogglebutton1.setOnClickListener(new OnClickListener()
{
    public void onClick(View v)
    {
        if (mtogglebutton1.isChecked())
        {
            /* 设置震动的周期 */
            mVibrator01.vibrate( new long[]{100,10,100,1000},-1);
            /*用Toast显示震动启动*/
            Toast.makeText
            (
                zhendong.this,
                getString(R.string.str_ok),
                Toast.LENGTH_SHORT
            ).show();
        }
        else
        {
            /* 取消震动 */
            mVibrator01.cancel();
            /*用Toast显示震动已被取消*/
            Toast.makeText
            (
                zhendong.this,
                getString(R.string.str_end),
                Toast.LENGTH_SHORT
            ).show();
        }
    }
});
```

- android与iphone及ipad开发书籍 持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等 持续不断更新中.....
- delphi《书籍》及《视频》教程 持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修炼班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程（个人博客系统49课全），外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班，php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库 ←

文件名 <ul style="list-style-type: none"> <input type="checkbox"/> 办公类软件使用指南 <input type="checkbox"/> 医学 <input type="checkbox"/> 历史人物传记 <input type="checkbox"/> 哲学宗教 <input type="checkbox"/> 外语资料（除英语外）（除英语外） <input type="checkbox"/> 官场类小说 <input type="checkbox"/> 建筑工程类 <input type="checkbox"/> 情感生活类小说 <input type="checkbox"/> 政治军事 <input type="checkbox"/> 教育学习科普大全 网址：WLSAM168.400GB.COM <input type="checkbox"/> 文学理论 <input type="checkbox"/> 智力开发、增强记忆、快速阅读技巧大全 <input type="checkbox"/> 社会生活 <input type="checkbox"/> 科学技术 <input type="checkbox"/> 程序编程类 <input type="checkbox"/> 经济管理 <input type="checkbox"/> 网络安全及管理 <input type="checkbox"/> 网赚系列 <input type="checkbox"/> 美食小吃烹饪煲汤大全 <input type="checkbox"/> 课外读物 	P D F 电子书专题库，内容详尽，每天不断更新！！ <p>本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习，如用于商业或非法用途的后果自负！</p>
--	--

OE Foxit PDF Editor ±à¼-ºæ·É·ÉùÓÐ (c) by Foxit Software Company, 2004 VIP培训教程，易语言黑月VIP视频教程，网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)

- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等 持续不断更新中....
- 网建设相关的pdf书籍及各种vip视频教程 持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价营销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等 持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。
- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 >



- ◆ 设置长震动模式，通过 mVibrator01.vibrate(new long[]{100,100,100,1000},0); 设置了此模式下的震动周期，具体代码如下所示。

```
/* 长震动 */
mtogglebutton2.setOnClickListener(new OnClickListener()
{
    public void onClick(View v)
    {
        if (mtogglebutton2.isChecked())
        {
            /*设置震动的周期*/
            mVibrator01.vibrate(new long[]{100,100,100,1000},0);

            /*用Toast显示震动启动*/
            Toast.makeText
            (
                zhendong.this,
                getString(R.string.str_ok),
                Toast.LENGTH_SHORT
            ).show();
        }
        else
        {
            /* 取消震动 */
            mVibrator01.cancel();

            /* 用Toast显示震动取消 */
            Toast.makeText
            (
                zhendong.this,
                getString(R.string.str_end),
                Toast.LENGTH_SHORT
            ).show();
        }
    }
});
```

- ◆ 设置节奏震动模式，通过 “{1000,50,1000,50,1000}” 设置此模式下的震动周期，具体代码如下所示。

```
/* 节奏震动 */
mtogglebutton3.setOnClickListener(new OnClickListener()
{
    public void onClick(View v)
```

```
    if (mtogglebutton3.isChecked())
    {
        /* 设置震动的周期 */
        mVibrator01.vibrate( new long[]{1000,50,1000,50,1000},0);

        /*用Toast显示震动启动*/
        Toast.makeText(
        (
            zhendong.this, getString(R.string.str_ok),
            Toast.LENGTH_SHORT
        ).show();
    }
    else
    {
        /* 取消震动 */
        mVibrator01.cancel();
        /* 用Toast显示震动取消 */
        Toast.makeText(
        (
            zhendong.this,
            getString(R.string.str_end),
            Toast.LENGTH_SHORT
        ).show();
    }
}
});
```

step 03 在文件 AndroidManifest.xml 中声明 Android.permission.VIBRATE 的权限，具体代码如下所示。

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0.0" package="irdc.zhendong">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity
            android:label="@string/app_name"
            android:name=".zhendong">
            <intent-filter>
```



```
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
<uses-permission android:name="android.permission.VIBRATE" />
</manifest>
```

执行后的效果如图 5-11 所示，当选择一种模式，并单击按钮后会启动对应的震动模式，如图 5-12 所示的是启动了“长时间”震动模式。

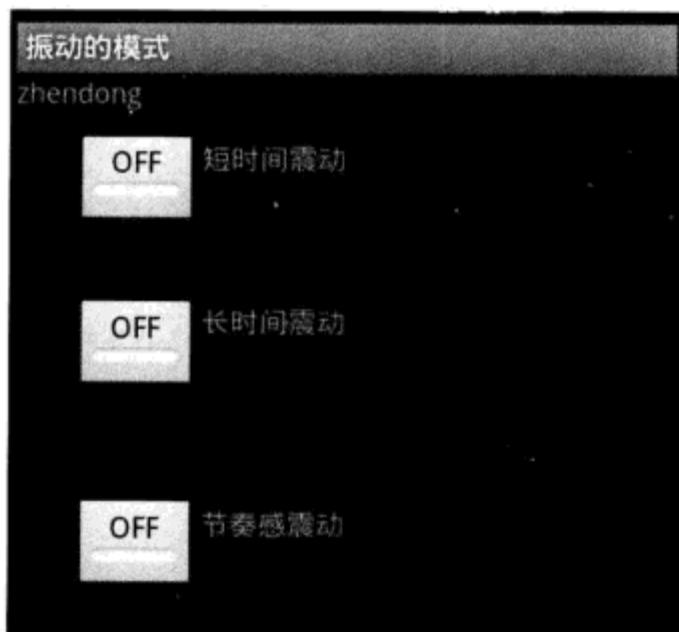


图5-11 初始效果

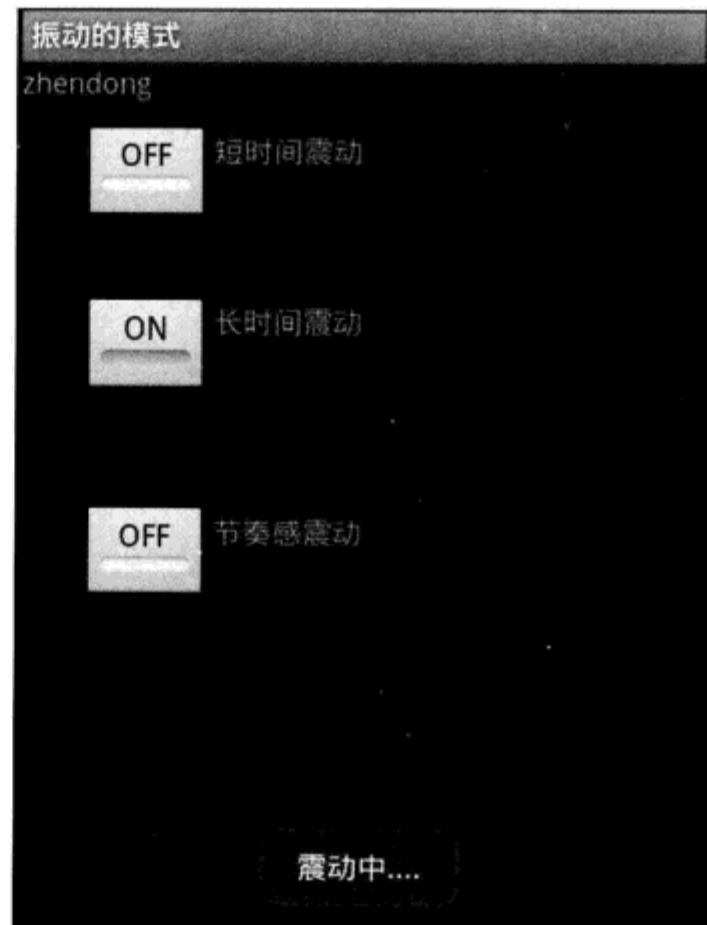


图5-12 长时间震动

5.6 搜索通讯录

在使用手机时经常需要搜索联系人信息，我们可以通过手机的查询系统迅速找到通讯录中的联系人电话。在本节的内容中，将通过一个具体实例的实现过程，介绍通过 ContentResolver 实现检索手机通讯录的方法。本实例的源码保存在【光盘\daima\第 5 章\sousuo】，具体实现流程如下所示。

step 01 编写文件 sousuo.java，首先通过 String[] 获取通讯录中的字段；然后通过 ContentResolver content = getContentResolver(); 和 contactCursor = content.query 来获取通讯录里的数据；最后定义 myAutoCompleteTextView.setOnItemClickListener 事件，即当用户单击联系人姓名后的拦截事件处理，并通过 ContactsAdapter.

getCursor() 来获取联系人的电话号码。具体代码如下所示。

```
package irdc.sousuo;

import irdc.sousuo.R;
import android.app.Activity;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.Contacts;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.AutoCompleteTextView;
import android.widget.TextView;

public class sousuo extends Activity
{
    private AutoCompleteTextView myAutoCompleteTextView;
    private TextView myTextView1;
    private Cursor contactCursor;
    private ContactsAdapter myContactsAdapter;
    /* 要获取通讯录的字段 */
    public static final String[] PEOPLE_PROJECTION = new String[]
    { Contacts.People._ID, Contacts.People.PRIMARY_PHONE_ID,
        Contacts.People.TYPE, Contacts.People.NUMBER, Contacts.People.
LABEL,
        Contacts.People.NAME };

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        myAutoCompleteTextView = (AutoCompleteTextView)
            findViewById(R.id.myAutoCompleteTextView);
        myTextView1 = (TextView) findViewById(R.id.myTextView1);

        /* 取得ContentResolver */
        ContentResolver content = getContentResolver();

        /* 取得通讯录的Cursor */
        contactCursor = content.query
```



```
        Contacts.People.CONTENT_URI,
        PEOPLE_PROJECTION, null, null,
        Contacts.People.DEFAULT_SORT_ORDER
    );
}

/* 将Cursor传入自己实现的ContactsAdapter */
myContactsAdapter = new ContactsAdapter(this, contactCursor);

myAutoCompleteTextView.setAdapter(myContactsAdapter);

myAutoCompleteTextView.setOnItemClickListener
(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick
    (AdapterView<?> arg0, View arg1, int arg2, long arg3)
    {
        /* 取得Cursor */
        Cursor c = myContactsAdapter.getCursor();
        /* 移到所点击的位置 */
        c.moveToPosition(arg2);
        String number = c.getString
        (c.getColumnIndexOrThrow(Contacts.People.NUMBER));
        /* 当找不到电话时显示无输入电话 */
        number = number == null ? "无输入电话" : number;
        myTextview1.setText(c.getString
        (c.getColumnIndexOrThrow(Contacts.People.NAME))
        + "的电话是" + number);
    }
});  }
}
```

step 02 编写文件 ContactsAdapter.java，此文件是继承了 CursorAdapter，并以 curse 作为下拉菜单 data 的 class，并且重写了 runQueryOnBackgroundThread(CharSequence constraint)，设置当输入 * 时，显示出所有的联系人电话信息。具体代码如下所示。

```
package irdc.sousuo;

import android.content.ContentResolver;
import android.content.Context;
import android.database.Cursor;
import android.provider.Contacts;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

public class ContactsAdapter extends CursorAdapter
{
    private ContentResolver mContent;

    public ContactsAdapter(Context context, Cursor c)
    {
        super(context, c);
        mContent = context.getContentResolver();
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor)
    {
        /* 取得通讯录人员的名字 */
        ((TextView) view).setText(cursor.getString(cursor
            .getColumnIndexOrThrow(Contacts.People.NAME)));
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup
parent)
    {
        final LayoutInflater inflater = LayoutInflater.from(context);
        final TextView view = (TextView) inflater.inflate(
            android.R.layout.simple_dropdown_item_1line, parent, false);
        view.setText(cursor.getString(cursor
            .getColumnIndexOrThrow(Contacts.People.NAME)));
        return view;
    }

    @Override
    public String convertToString(Cursor cursor)
    {
        return cursor.getString(cursor.getColumnIndexOrThrow(Contacts.
People.NAME));
    }
}
```



```
@Override  
public Cursor runQueryOnBackgroundThread(CharSequence constraint)  
{  
    if (getFilterQueryProvider() != null)  
    {  
        return getFilterQueryProvider().runQuery(constraint);  
    }  
  
    StringBuilder buffer = null;  
    String[] args = null;  
    if (constraint != null)  
    {  
        buffer = new StringBuilder();  
        buffer.append("UPPER(");  
        buffer.append(ContactMethods.NAME);  
        buffer.append(") GLOB ?");  
        args = new String[]  
        { constraint.toString().toUpperCase() + "*" };  
    }  
    return mContent.query(ContactMethods.CONTENT_URI,  
                           ContactMethods.PROJECTION, buffer == null ? null : buffer.  
                           toString(),  
                           args, ContactMethods.DEFAULT_SORT_ORDER);  
}  
}
```

step 03 在文件 AndroidManifest.xml 中声明打开访问通讯录的权限，具体代码如下所示。

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="irdc.sousuo"  
    android:versionCode="1"  
    android:versionName="1.0.0">  
    <application android:icon="@drawable/icon" android:label="@string/  
app_name">  
        <activity android:name=".sousuo"  
            android:label="@string/app_name">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.  
LAUNCHER" />  
            </intent-filter>  
        </activity>  
    </application>  
<uses-permission android:name="android.permission.READ_CONTACTS"></
```



```
uses-permission>
</manifest>
```

step 04 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    >
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:textColor="@drawable/black"
        >
    </TextView>
    <AutoCompleteTextView
        android:id="@+id/my.AutoCompleteTextView"
        android:completionThreshold="1"
        android:completionHint="@string/strHint"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >
    </AutoCompleteTextView>
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/black"
        >
    </TextView>
</LinearLayout>
```

执行后的初始效果如图 5-13 所示。当输入一个联系人字符后，系统能够根据联系人自动提示显示对应的信息，当输入 * 后会显示通讯录中所有联系人的信息。

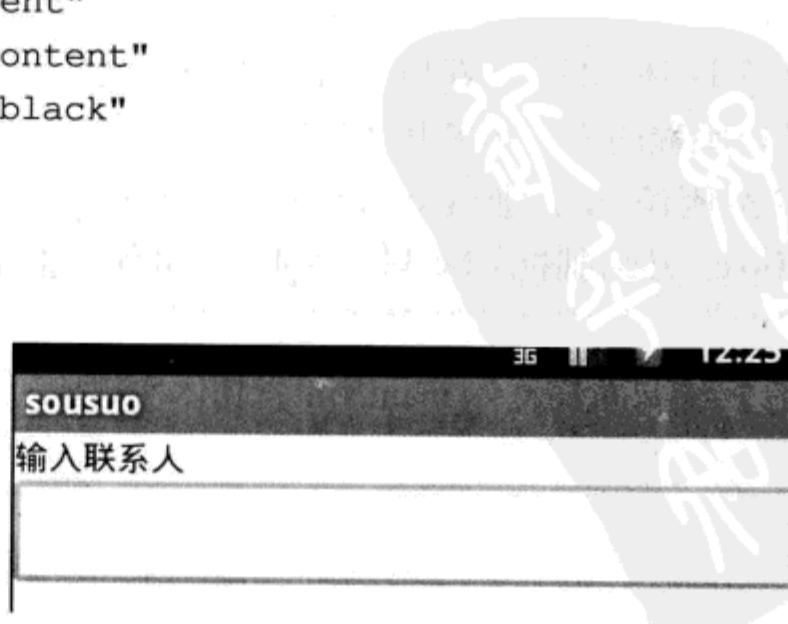


图 5-13 执行效果

5.7 使用Wi-Fi

Wi-Fi是一种可以将个人电脑、手持设备（如PDA、手机）等终端以无线方式互相连接的技术。Wi-Fi是一个无线网路通信技术的品牌，由 WiFi 联盟（WiFi Alliance）所持有。目的是改善基于 IEEE 802.11 标准的无线网路产品之间的互通性。一般人会把 WiFi 及 IEEE 802.11 混为一谈，甚至把 WiFi 等同于无线网际网路。

Wi-Fi就是一种无线联网的技术，以前通过网线连接电脑，而现在则是通过无线电波来连网；常见的就是一个无线路由器，那么在这个无线路由器的电波覆盖的有效范围都可以采用 WIFI 连接方式进行联网，如果无线路由器连接了一条 ADSL 线路或者别的上网线路，则又被称为“热点”。

在Android系统中，存在了一个无线控制模块。打开方式如下：依次单击“Menu” | “Settings” | “Wireless” | “Wi-Fi settings”，进入图5-14所示界面。

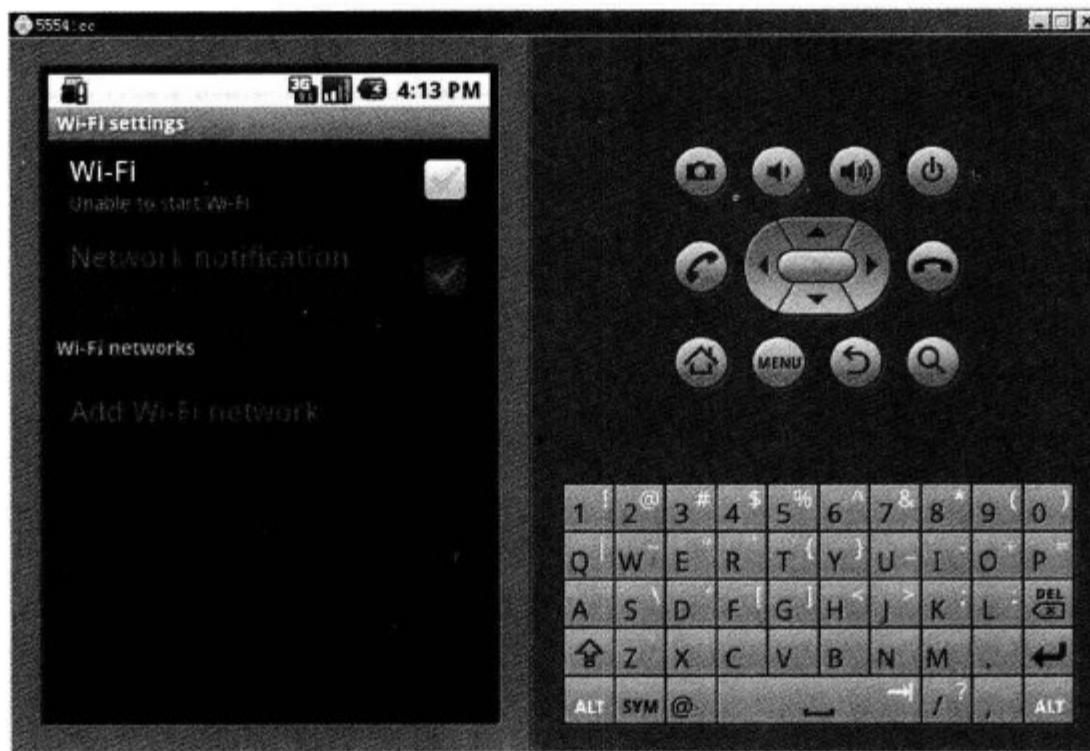


图5-14 WiFi控制界面

图 5-14 所示的是 Wi-Fi 的控制界面，在此可以控制 Wi-Fi 的打开和关闭，而本实例的目的是以编程的方式实现同样类似的功能。

本实例演示了使用 Wi-Fi 的基本流程，在具体实现上，会首先定义一个复选框 CheckBox，然后捕捉 CheckBox 的点击事件，根据对应的状态显示对应的提示。本实例的源码保存在【光盘\daima\第 5 章\WiFi】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
```

```

    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:id="@+id/myTextView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@drawable/blue"
    android:text="@string/hello"
/>
<CheckBox
    android:id="@+id/myCheckBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_checked"
    android:textColor="@drawable/blue"
/>
</LinearLayout>

```

step 02 编写主程序文件是wifi.java，其具体实现流程如下所示。

- ◆ 引入 wifi.WifiManager 和 widget.CheckBox，然后创建 WiFiManager 对象 mWiFiManager01。具体代码如下所示。

```

package irdc.wifi;

import irdc.wifi.R;
import android.app.Activity;
import android.content.Context;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;

public class wifi extends Activity
{
    private TextView mTextView01;
    private CheckBox mCheckBox01;

    /* 创建 WiFiManager 对象 */
    private WifiManager mWiFiManager01;
}

```



- ◆ 分别定义 mTextView01 和 mCheckBox01，分别用于显示提示文本和获取复选框的选择状态。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mTextView01 = (TextView) findViewById(R.id.myTextView1);
    mCheckBox01 = (CheckBox) findViewById(R.id.myCheckBox1);
```

- ◆ 以 getSystemService 取得 WIFI_SERVICE，然后通过 if 语句来判断运行程序后的 WiFi 状态是否打开或打开中，这样便可显示对应的提示信息。具体代码如下所示。

```
mWiFiManager01 = (WifiManager)
    this.getSystemService(Context.WIFI_SERVICE);
/* 判断运行程序后的 WiFi 状态是否打开或打开中 */
if (mWiFiManager01.isWifiEnabled())
{
    /* 判断 WiFi 状态是否“已打开” */
    if (mWiFiManager01.getWifiState() ==
        WifiManager.WIFI_STATE_ENABLED)
    {
        /* 若 WiFi 已打开，将选取项打勾 */
        mCheckBox01.setChecked(true);
        /* 更改选取项文字为关闭 WiFi */
        mCheckBox01.setText(R.string.str_uncheck);
    }
    else
    {
        /* 若 WiFi 未打开，将选取项勾选取消 */
        mCheckBox01.setChecked(false);
        /* 更改选取项文字为打开 WiFi */
        mCheckBox01.setText(R.string.str_checked);
    }
}
else
{
    mCheckBox01.setChecked(false);
    mCheckBox01.setText(R.string.str_checked);
}
```

- ◆ 通过 mCheckBox01.setOnClickListener 来捕捉 CheckBox 的点击事件，用

onClick(View v) 方法获取用户的单击，然后根据 if 语句根据操作需求来执行对应操作，并根据需要输出对应的提示信息。具体代码如下所示。

```
mCheckBox01.setOnClickListener(  
    new CheckBox.OnClickListener()  
    {  
        @Override  
        public void onClick(View v)  
        {  
            // TODO Auto-generated method stub  
  
            /* 当选取项为取消选取状态 */  
            if(mCheckBox01.isChecked()==false)  
            {  
                /* 尝试关闭Wi-Fi服务 */  
                try  
                {  
                    /* 判断WiFi状态是否为已打开 */  
                    if(mWiFiManager01.isWifiEnabled())  
                    {  
                        /* 关闭WiFi */  
                        if(mWiFiManager01.setWifiEnabled(false))  
                        {  
                            mTextView01.setText(R.string.str_stop_wifi_done);  
                        }  
                        else  
                        {  
                            mTextView01.setText(R.string.str_stop_wifi_failed);  
                        }  
                    }  
                    else  
                    {  
                        /* WiFi状态不为已打开状态时 */  
                        switch(mWiFiManager01.getWifiState())  
                        {  
                            /* WiFi正在打开过程中，导致无法关闭... */  
                            case WifiManager.WIFI_STATE_ENABLING:  
                                mTextView01.setText  
                                (  
                                    getResources().getText  
                                    (R.string.str_stop_wifi_failed)+"."  
                                    getResources().getText  
                                    (R.string.str_wifi_enabling)  
                                );  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```
        break;
    /* WiFi正在关闭过程中，导致无法关闭... */
    case WifiManager.WIFI_STATE_DISABLING:
        mTextView01.setText
        (
            getResources().getText
            (R.string.str_stop_wifi_failed)+"："+getResources().getText
            (R.string.str_wifi_disabling)
        );
        break;
    /* WiFi已经关闭 */
    case WifiManager.WIFI_STATE_DISABLED:
        mTextView01.setText
        (
            getResources().getText
            (R.string.str_stop_wifi_failed)+"："+getResources().getText
            (R.string.str_wifi_disabled)
        );
        break;
    /* 无法取得或辨识WiFi状态 */
    case WifiManager.WIFI_STATE_UNKNOWN:
    default:
        mTextView01.setText
        (
            getResources().getText
            (R.string.str_stop_wifi_failed)+"："+getResources().getText
            (R.string.str_wifi_unknown)
        );
        break;
    }
    mCheckBox01.setText(R.string.str_checked);
}
}
catch (Exception e)
{
    Log.i("HIPPO", e.toString());
    e.printStackTrace();
}
}
else if(mCheckBox01.isChecked()==true) .
```

```
{  
    /* 尝试打开Wi-Fi服务 */  
    try  
    {  
        /* 确认WiFi服务是关闭且不在打开作业中 */  
        if (!mWiFiManager01.isWifiEnabled() &&  
            mWiFiManager01.getWifiState() !=  
            WifiManager.WIFI_STATE_ENABLING)  
        {  
            if (mWiFiManager01.setWifiEnabled(true))  
            {  
                switch (mWiFiManager01.getWifiState())  
                {  
                    /* WiFi正在打开过程中，导致无法打开... */  
                    case WifiManager.WIFI_STATE_ENABLING:  
                        mTextView01.setText  
                        (  
                            getResources().getText  
                            (R.string.str_wifi_enabling)  
                        );  
                        break;  
                    /* WiFi已经为打开，无法再次打开... */  
                    case WifiManager.WIFI_STATE_ENABLED:  
                        mTextView01.setText  
                        (  
                            getResources().getText  
                            (R.string.str_start_wifi_done)  
                        );  
                        break;  
                    /* 其他未知的错误 */  
                    default:  
                        mTextView01.setText  
                        (  
                            getResources().getText  
                            (R.string.str_start_wifi_failed)+ ":"+  
                            getResources().getText  
                            (R.string.str_wifi_unknown)  
                        );  
                        break;  
                }  
            }  
        }  
    }  
}
```

```
mTextView01.setText(R.string.str_start_wifi_failed);
}

else
{
    switch(mWifiManager01.getWifiState())
    {
        /* WiFi正在打开过程中，导致无法打开... */
        case WifiManager.WIFI_STATE_ENABLING:
            mTextView01.setText
            (
                getResources().getText
                (R.string.str_start_wifi_failed)+ ":" +
                getResources().getText
                (R.string.str_wifi_enabling)
            );
            break;
        /* WiFi正在关闭过程中，导致无法打开... */
        case WifiManager.WIFI_STATE_DISABLING:
            mTextView01.setText
            (
                getResources().getText
                (R.string.str_start_wifi_failed)+ ":" +
                getResources().getText
                (R.string.str_wifi_disabling)
            );
            break;
        /* WiFi已经关闭 */
        case WifiManager.WIFI_STATE_DISABLED:
            mTextView01.setText
            (
                getResources().getText
                (R.string.str_start_wifi_failed)+ ":" +
                getResources().getText
                (R.string.str_wifi_disabled)
            );
            break;
        /* 无法取得或识别WiFi状态 */
        case WifiManager.WIFI_STATE_UNKNOWN:
        default:
            mTextView01.setText
            (
                getResources().getText
```



```
        (R.string.str_start_wifi_failed)+"："+  
        getResources().getText  
        (R.string.str_wifi_unknown)  
    );  
    break;  
}  
}  
mCheckBox01.setText(R.string.str_uncheck);  
}  
catch (Exception e)  
{  
    Log.i("HIPPO", e.toString());  
    e.printStackTrace();  
}  
}  
}  
});  
}
```

- ◆ 定义 mMakeTextToast 方法() 来根据当前操作显示对应的提示性信息，具体代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)
{
    if(isLong==true)
    {
        Toast.makeText(wifi.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
        Toast.makeText(wifi.this, str, Toast.LENGTH_SHORT).show();
    }
}

@Override
protected void onResume()
{
    // TODO Auto-generated method stub

/* 在onResume重写事件为取得打开程序当下WiFi的状态 */
try
{
    switch(mWiFiManager01.getWifiState())
    {

```

```
/* WiFi已经在打开状态... */
case WifiManager.WIFI_STATE_ENABLED:
    mTextView01.setText(
    (
        getResources().getText(R.string.str_wifi_enabling)
    );
    break;
/* WiFi正在打开过程中状态... */
case WifiManager.WIFI_STATE_ENABLING:
    mTextView01.setText(
    (
        getResources().getText(R.string.str_wifi_enabling)
    );
    break;
/* WiFi正在关闭过程中... */
case WifiManager.WIFI_STATE_DISABLING:
    mTextView01.setText(
    (
        getResources().getText(R.string.str_wifi_disabling)
    );
    break;
/* WiFi已经关闭 */
case WifiManager.WIFI_STATE_DISABLED:
    mTextView01.setText(
    (
        getResources().getText(R.string.str_wifi_disabled)
    );
    break;
/* 无法取得或识别WiFi状态 */
case WifiManager.WIFI_STATE_UNKNOWN:
default:
    mTextView01.setText(
    (
        getResources().getText(R.string.str_wifi_unknown)
    );
    break;
}
}
catch(Exception e)
{
    mTextView01.setText(e.toString());
    e.printStackTrace();
}
```

```

        super.onResume();
    }

    @Override
    protected void onPause()
    {
        // TODO Auto-generated method stub
        super.onPause();
    }
}

```

step 03 编写文件 `AndroidManifest.xml`，在里面声明 WiFi 的访问以及对网络状态权限。具体代码如下所示。

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="irdc.wifi" android:versionCode="1" android:versionName="1.0.0" >
    - <application android:icon="@drawable/icon" android:label="@string/app_name" >
        - <activity android:name=".wifi" android:label="@string/app_name" >
            - <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <!-- 声明存取WIFI以及网络等相关权限 -->
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
</manifest>

```

执行后会显示 1 个按钮，如图 5-15 所示。当选择复选框后会执行对应的操作处理，并且显示对应的提示信息。



图5-15 初始效果



5.8 触摸拨号

触摸拨号功能十分常见，本节将通过一个具体实例来演示在 Android 中实现类似触摸拨号按钮的基本流程。在本实例中，通过 Intent 方式将电话号码传递给内置的拨号程序，然后内置拨号程序实现拨号处理操作。利用了 startActivity() 方法将程序焦点交给内置的拨号程序，这样原来的 Activity 会成为失焦状态，并且还会发生 onPause() 事件，直到关闭拨号程序，焦点也交还给原来的 Activity。

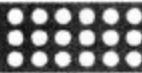
在具体实现上，先插入了一个按钮，当单击按钮后会调用手机内置的默认拨号界面。本实例的源码保存在【光盘\daima\第 5 章\chumo】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/white"
    >
    <ImageButton
        android:id="@+id/myImageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@android:drawable/sym_action_call"
    >
    </ImageButton>
</LinearLayout>
```

step 02 编写主程序文件 practice16.java，其功能是当用户单击按钮后通过 android.intent.action.CALL_BUTTON 调用默认的拨号界面。具体代码如下所示。

```
public class practice16 extends Activity
{
    private ImageButton myImageButton;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myImageButton = (ImageButton) findViewById(R.id.myImageButton);
```



```
myImageButton.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        /* 调用拨号的画面 */
        Intent myIntentDial = new Intent("android.intent.action.CALL_
BUTTON");
        startActivity(myIntentDial);
    }
});
```

执行后会显示对应的按钮，如图 5-16 所示。触摸拨号图标后，会自动来到系统内置的默认拨号界面，如图 5-17 所示。

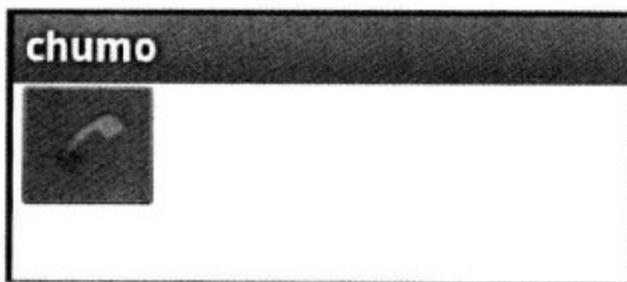


图5-16 初始效果

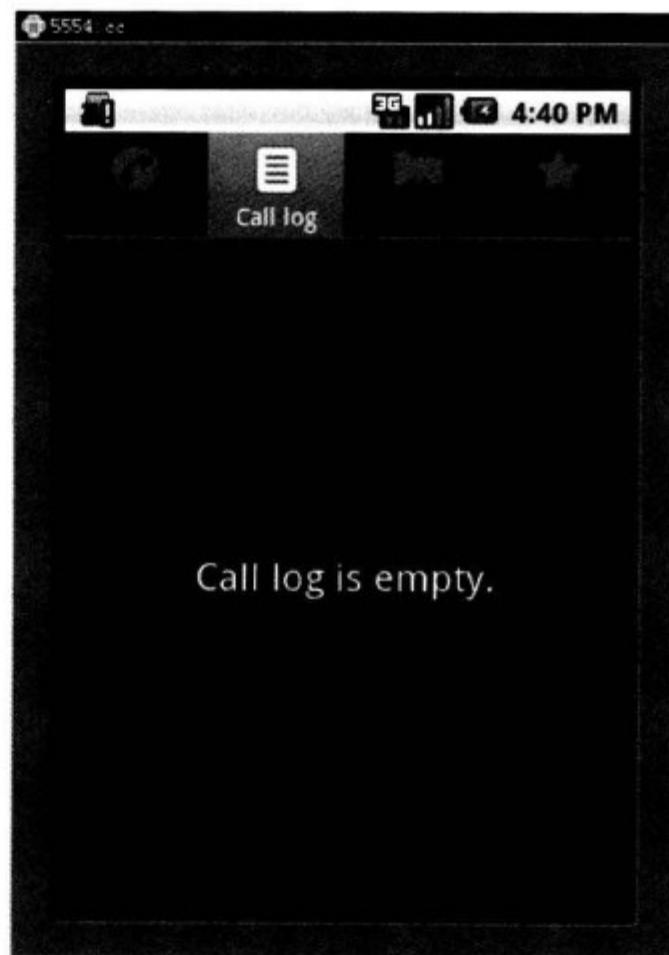


图5-17 内置的拨号界面

5.9 获取设备运营商信息

在使用手机的过程中，经常因为某种需求而查看设备运营商信息的信息，例如手机和网络的相关信息。在本实例中，将以 `getSystemService` 来获取 `TelephonyManager` 对



象，然后通过 TelephonyManager 的方法来获取和电信有关的网络信息。然后通过 Android.provider.Settings.System.getString() 来获取手机的相关设置信息，并将获取的信息存入自定义的 MyAdapter 中，最后以 setListAdapter 内的信息显示在 ListView 中。

本实例的源码保存在【光盘\daima\第 5 章\huoqu】，具体实现流程如下所示。

step 01 编写文件 huoqu.java，其具体实现流程如下所示。

- ◆ 先引入 import 相关 class，然后载入 main.xml 布局。具体代码如下所示。

```
/* import相关class */
import irdc.huoqu.R;

import java.util.ArrayList;
import java.util.List;
import android.app.ListActivity;
import android.content.ContentResolver;
import android.os.Bundle;
import android.telephony.TelephonyManager;

public class huoqu extends ListActivity
{
    private TelephonyManager telMgr;
    private List<String> item=new ArrayList<String>();
    private List<String> value=new ArrayList<String>();

    @SuppressWarnings("static-access")
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 载入main.xml Layout */
        setContentView(R.layout.main);

        telMgr = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
```

- ◆ 将取得的信息写入List中，具体包含下面的信息：

- 取得手机电话号码；
- 取得电信网络国别；
- 取得电信公司代码；
- 取得电信公司名称；
- 取得行动通信类型；
- 取得网络类型；
- 取得漫游状态；

- 取得手机IMEI;
- 取得IMEI SV;
- 取得IMSI;
- 取得蓝牙状态;
- 取得WIFI状态;
- 飞行模式是否打开;
- 取得数据漫游是否打开。

如果上述信息都无法获得，则输出“无法取得”的提示。具体代码如下所示。

```
/* 将取得的信息写入List中 */
item.add(getResources().getText(R.string.str_list0).toString());
if(telMgr.getLine1Number()!=null)
{
    value.add(telMgr.getLine1Number());
}
else
{
    value.add("无法取得");
}

/* 取得电信网络国别 */
item.add(getResources().getText(R.string.str_list1).toString());
if(telMgr.getNetworkCountryIso().equals(""))
{
    value.add("无法取得");
}
else
{
    value.add(""+telMgr.getNetworkCountryIso());
}

/* 取得电信公司代码 */
item.add(getResources().getText(R.string.str_list2).toString());
if(telMgr.getNetworkOperator().equals(""))
{
    value.add("无法取得");
}
else
{
    value.add(telMgr.getNetworkOperator());
}
```



```
/* 取得电信公司名称 */
item.add(getResources().getText(R.string.str_list3).toString());
if(telMgr.getNetworkOperatorName().equals(""))
{
    value.add("无法取得");
}
else
{
    value.add(telMgr.getNetworkOperatorName());
}

/* 取得行动通信类型 */
item.add(getResources().getText(R.string.str_list4).toString());
if(telMgr.getPhoneType() == telMgr.PHONE_TYPE_GSM)
{
    value.add("GSM");
}
else
{
    value.add("未知");
}

/* 取得网络类型 */
item.add(getResources().getText(R.string.str_list5).toString());
if(telMgr.getNetworkType() == telMgr.NETWORK_TYPE_EDGE)
{
    value.add("EDGE");
}
else if(telMgr.getNetworkType() == telMgr.NETWORK_TYPE_GPRS)
{
    value.add("GPRS");
}
else if(telMgr.getNetworkType() == telMgr.NETWORK_TYPE_UMTS)
{
    value.add("UMTS");
}
else if(telMgr.getNetworkType() == 4)
{
    value.add("HSDPA");
}
else
{
    value.add("未知");
}
```

```
}

/* 取得漫游状态 */
item.add(getResources().getText(R.string.str_list6).toString());
if(telMgr.isNetworkRoaming())
{
    value.add("漫游中");
}
else{
    value.add("无漫游");
}

/* 取得手机IMEI */
item.add(getResources().getText(R.string.str_list7).toString());
value.add(telMgr.getDeviceId());

/* 取得IMEI SV */
item.add(getResources().getText(R.string.str_list8).toString());
if(telMgr.getDeviceSoftwareVersion()!=null)
{
    value.add(telMgr.getDeviceSoftwareVersion());
}
else
{
    value.add("无法取得");
}

/* 取得手机IMSI */
item.add(getResources().getText(R.string.str_list9).toString());
if(telMgr.getSubscriberId() !=null)
{
    value.add(telMgr.getSubscriberId());
}
else
{
    value.add("无法取得");
}

/* 取得ContentResolver */
ContentResolver cv = huoqu.this.getContentResolver();
String tmpS="";

/* 取得蓝牙状态 */
```

```
item.add(getResources().getText(R.string.str_list10)
        .toString());
tmpS=android.provider.Settings.System.getString(cv,
        android.provider.Settings.System.BLUETOOTH_ON);
if(tmpS.equals("1"))
{
    value.add("已打开");
}
else{
    value.add("未打开");
}

/* 取得WIFI状态 */
item.add(getResources().getText(R.string.str_list11)
        .toString());
tmpS=android.provider.Settings.System.getString(cv,
        android.provider.Settings.System.WIFI_ON);
if(tmpS.equals("1"))
{
    value.add("已打开");
}
else{
    value.add("未打开");
}

/* 取得飞行模式是否打开 */
item.add(getResources().getText(R.string.str_list12)
        .toString());
tmpS=android.provider.Settings.System.getString(cv,
        android.provider.Settings.System.AIRPLANE_MODE_ON);
if(tmpS.equals("1"))
{
    value.add("打开中");
}
else{
    value.add("未打开");
}

/* 取得数据漫游是否打开 */
item.add(getResources().getText(R.string.str_list13)
        .toString());
tmpS=android.provider.Settings.System.getString(cv,
        android.provider.Settings.System.DATA_ROAMING);
```

```

    if(tmpS.equals("1"))
    {
        value.add("打开中");
    }
    else{
        value.add("未打开");
    }
}

```

- 使用自定义的 MyAdapter 来将数据传入 ListActivity，具体代码如下所示。

```

    setListAdapter(new MyAdapter(this,item,value));
}
}

```

step 02 编写文件 MyAdapter.java，其具体实现流程如下所示。

- 先引入相关 class 类，然后自定义 Adapter，并继承于 android.widget.BaseAdapter。具体代码如下所示。

```

/* import相关class */
import irdc.huoqu.R;

import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

/* 自定义的Adapter，继承android.widget.BaseAdapter */
public class MyAdapter extends BaseAdapter
{
    /* 变量声明 */
    private LayoutInflater mInflater;
    private List<String> items;
    private List<String> values;
    /* MyAdapter的构造器，传入三个参数 */
    public MyAdapter(Context context,List<String> item,
                    List<String> value)
    {
        /* 参数初始化 */
        mInflater = LayoutInflater.from(context);
        items = item;
        values = value;
    }
}

```



- ◆ 通过 @Override 分别覆盖方法 getCount(), getItem(int position), getItemId(int position), getView(int position, View convertView, ViewGroup par)。具体代码如下所示。

```
/* 因继承BaseAdapter, 需覆盖以下方法 */  
@Override  
public int getCount()  
{  
    return items.size();  
}  
  
@Override  
public Object getItem(int position)  
{  
    return items.get(position);  
}  
  
@Override  
public long getItemId(int position)  
{  
    return position;  
}  
  
@Override  
public View getView(int position, View convertView, ViewGroup par)  
{  
    ViewHolder holder;  
  
    if(convertView == null)  
    {  
        /* 使用自定义的file_row作为Layout */  
        convertView = mInflater.inflate(R.layout.row_layout,null);  
        /* 初始化holder的text与icon */  
        holder = new ViewHolder();  
        holder.text1=(TextView)convertView.findViewById(R.id.myText1);  
        holder.text2=(TextView)convertView.findViewById(R.id.myText2);  
  
        convertView.setTag(holder);  
    }  
    else  
    {  
        holder = (ViewHolder) convertView.getTag();  
    }  
    /* 设置要显示的信息 */
```

```
holder.text1.setText(items.get(position).toString());
holder.text2.setText(values.get(position).toString());

return convertView;
}

/* class ViewHolder */
private class ViewHolder
{
    /* text1: 信息名称
     * text2: 信息内容 */
    TextView text1;
    TextView text2;
}
}
```

执行后会按照指定样式显示获取的网络信息和手机信息，如图 5-18 所示。

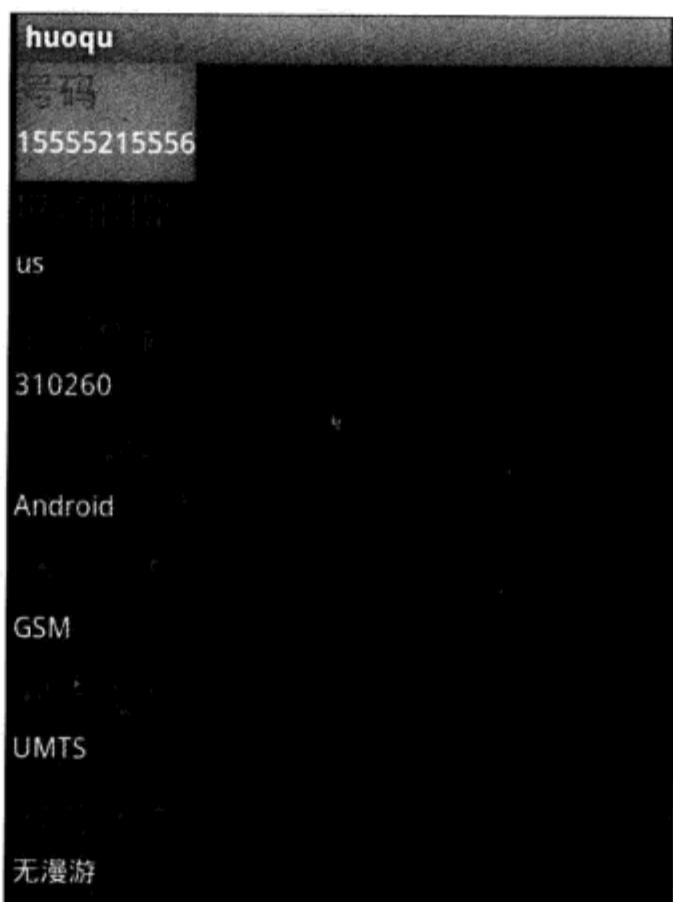


图5-18 获取的信息

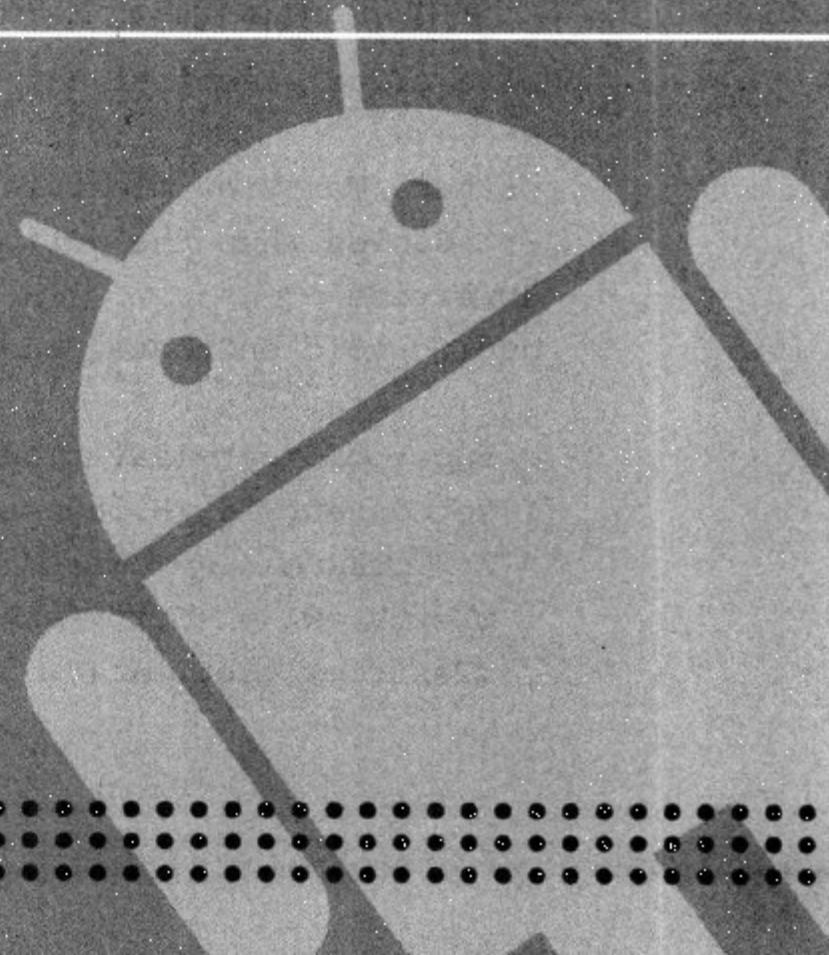




第6章

自动服务实战演练

在手机应用中，有很多的自动服务功能。例如，剩余电量、存储卡容量提示和黑名单自动屏蔽等。这些自动服务功能，很好地为用户提供了人性化的服务，整个操作过程更加方便。本章将通过几个典型实例的实现过程，详细介绍这些自动手机服务功能的实现流程。





6.1 来短信自动提醒

我们使用的手机如果收到了短信后会自动在屏幕显示“有短信”提示。在手机系统中，有广播系统 BroadcastReseiver，它实时监听手机内的短信状况。当手机收到短信后，会通过 Notification 在状态栏中显示短信的摘要信息。在本实例中，会将接收到的短信对象解析为可以识别发信人号码和短信正文的字符串。本实例的难点是如何向系统注册一个常驻的 BroadcastReseiver 对象，然后在后台中聆听短信事件，最后将短信内容编译出来。

本实例的源码保存在【光盘\daima\第 6 章\ziti】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

step 02 编写文件 ziti.java，其功能是以 TextView 的文字显示“正在等待接收短信……”的提示。具体代码如下所示。

```
public class practice1 extends Activity
{
    private TextView mTextView1;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*通过findViewById构造器创建TextView对象*/
        mTextView1 = (TextView) findViewById(R.id.myTextView1);
        mTextView1.setText("正在等待接收短信...");
```

step 03 编写文件 ziti_SMSreceiver.java，下面开始讲解其具体实现流程。

先引用 BroadcastReceiver 类，然后引用 telephony.gsm.SmsMessage 来收取短信，接着引用 Toast 类来告知用户收到短信。具体代码如下所示。

```
package irdc.ziti;

/*必须引用BroadcastReceiver类*/
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
/*必须引用telephony.gsm.SmsMessage来收取短信*/
import android.telephony.gsm.SmsMessage;
/*必须引用Toast类来告知用户收到短信*/
import android.widget.Toast;
```

- ◆ 自定义继承自 BroadcastReceiver 类，用于聆听系统服务广播的信息。具体实现流程如下：
 - 声明静态字符串，并使用 android.provider.Telephony.SMS_RECEIVED 作为 Action 为短信的依据。
 - 通过 if 语句判断传来的 Intent 是否为短信，如果是则先建构一字符串集合变量 sb，然后接收由 Intent 传来的数据。
 - 通过 if 语句判断 Intent 是有数据。

对应代码如下所示。

```
public class ziti_SMSreceiver extends BroadcastReceiver
{
    /*声明静态字符串，并使用android.provider.Telephony.SMS_RECEIVED
    作为Action为短信的依据*/
    private static final String mACTION =
        "android.provider.Telephony.SMS_RECEIVED";

    @Override
    public void onReceive(Context context, Intent intent)
    {
        // TODO Auto-generated method stub
        /* 判断传来的Intent是否为短信*/
        if (intent.getAction().equals(mACTION))
        {
            /*建构一字符串集合变量sb*/
            StringBuilder sb = new StringBuilder();
            /*接收由Intent传来的数据*/
            Bundle bundle = intent.getExtras();
```



```
/*判断Intent是有数据*/
if (bundle != null)
{
    /* pdus为 android内置短信参数 identifier
     * 通过bundle.get("")返回一包含pdus的对象*/
    Object[] myOBJpdus = (Object[]) bundle.get("pdus");
    /*构建短信对象array，并依据收到的对象长度来创建array的大小*/
    SmsMessage[] messages = new SmsMessage[myOBJpdus.length];
    for (int i = 0; i<myOBJpdus.length; i++)
    {
        messages[i] =
            SmsMessage.createFromPdu((byte[]) myOBJpdus[i]);
    }

    /* 将送来的短信合并自定义信息于StringBuilder当中 */
    for (SmsMessage currentMessage : messages)
    {
        sb.append("接收到来自:\n");
        /* 来信者的电话号码 */
        sb.append(currentMessage.getDisplayOriginatingAddress());
        sb.append("\n-----传来的短信-----\n");
        /* 取得传来信息的BODY */
        sb.append(currentMessage.getDisplayMessageBody());
    }
}
}
```

- ◆ 以 Notification(Toase) 显示来信信息，然后返回主 Activity，并设置让其以一个全新的 task 任务来运行。具体代码如下所示。

```
Toast.makeText
(
    context, sb.toString(), Toast.LENGTH_LONG
).show();

/* 返回主Activity */
Intent i = new Intent(context, ziti.class);
/*设置让以一个全新的task来运行*/
i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(i);
}
```

step 04 在文件 AndroidManifest.xml 中向系统注册常驻的 receiver，并设置这个 receiver



的 intent-filter 名为 “`android.provider.Telephony.SMS_RECEIVED`”。另外还需要设置 `permission.RECEIVE_SMS` 权限。具体代码如下所示。

```
<!-- 建立receiver来聆听系统广播信息 -->
<receiver android:name="practice1_SMSreceiver">
    <!—设定要捕捉的信息名称为provider中Telephony.SMS_RECEIVED -->
    <intent-filter>
        <action
            android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
</application>
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-
permission>
```

执行后的初始效果如图 6-1 所示。当接收到短信后会在屏幕中显示对应的提示信息，如图 6-2 所示。

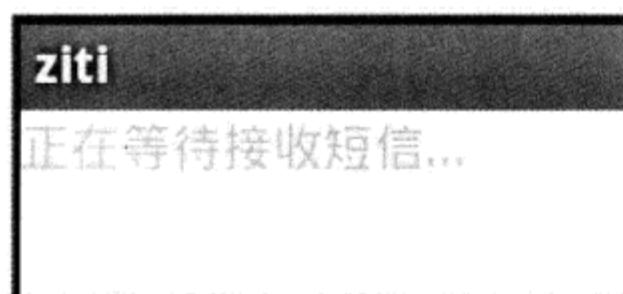


图6-1 执行效果



图6-2 短信提示

在具体测试时，我们需要同时运行两个模拟器，一个用于发送短信，另外的则接收。但是如果机器太慢，无法启动两个模拟器。也可以只启动一个模拟器。然后在 eclipse 菜单中依次单击 Windows->Show View->Other->Android->Emulator Control，打开 Emulator Control 面板。在 Telephony Actions 分组框中，Voice 是呼叫，SMS 是发送短信。Incoming number 是模拟器的端口号，用户也可以使用这个功能给模拟器拨打电话或发送短信，如图 6-3 所示。

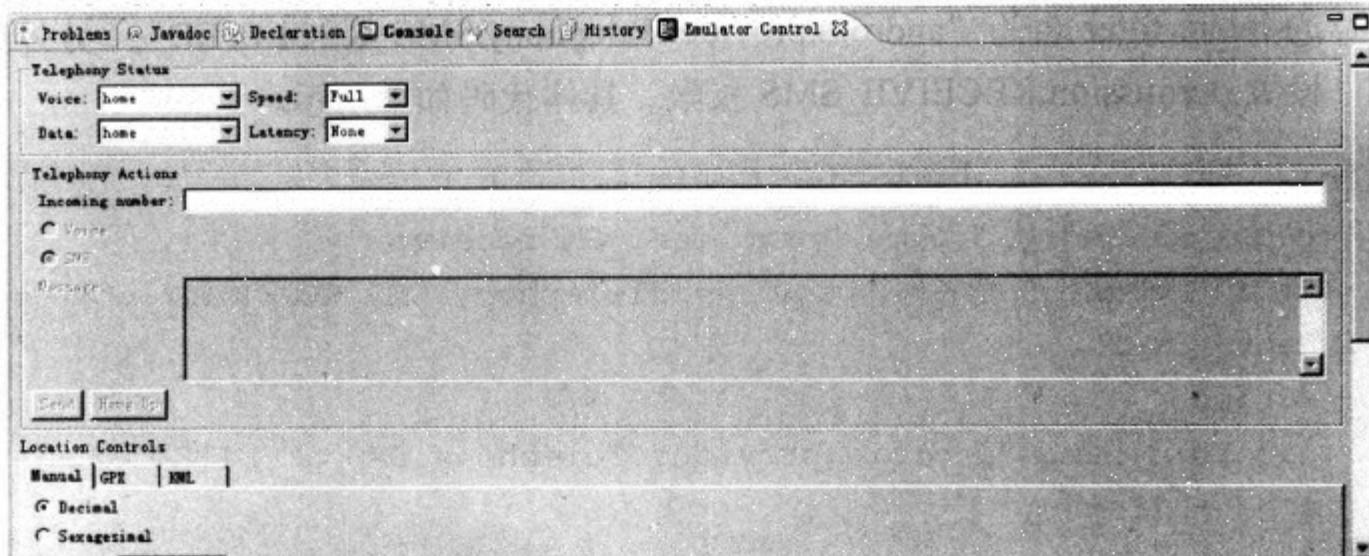


图6-3 Emulator Control面板

6.2 自动显示剩余电量

在使用过程中，最担心的是手机没电而影响业务，所以及时显示电池容量功能变得愈发重要了。

我们可以使用 Android API 里面的 BroadcastReseiver 类 和 Button 的 Listener 类，当 Reseiver 被注册后会在后台等待被其他程序调用。当指定要捕捉的 Action 发生时，Reseiver 就会被调用，并运行 onReseiver 来实现里面的程序。

在本实例中，将利用 BroadcastReseiver 的特性来获取手机电池的容量。通过注册 BroadcastReseiver 时设置的 IntentFilter 来获取系统发出的 Intent.ACTION_BATTERY_CHANGED，然后以此来获取电池的容量。本实例的源码保存在【光盘\daima\第6章\dian】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/layout1"
    android:background="@drawable/white"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:textSize="20sp">
```

```

    android:text="@string/str_title"
    android:layout_x="60px"
    android:layout_y="40px"
  >
</TextView>
<Button
    android:id="@+id/myButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button1"
    android:textColor="@drawable/black"
    android:textSize="14sp"
    android:layout_x="80px"
    android:layout_y="90px"
  >
</Button>
</AbsoluteLayout>

```

step 02 编写主程序文件是 dian.java，其具体实现流程如下所示。

- ◆ 声明 3 个变量 intLevel、intScale 和 mButton01，然后创建 BroadcastReceiver，如果捕捉到的 action 是 ACTION_BATTERY_CHANGED，则运行 onBatteryInfoReceiver()。具体代码如下所示。

```

public class dian extends Activity
{
    /* 变量声明 */
    private int intLevel;
    private int intScale;
    private Button mButton01;

    /* 创建BroadcastReceiver */
    private BroadcastReceiver mBatInfoReceiver=new BroadcastReceiver()
    {
        public void onReceive(Context context, Intent intent)
        {
            String action = intent.getAction();
            /* 如果捕捉到的action是ACTION_BATTERY_CHANGED,
             * 就运行onBatteryInfoReceiver() */
            if (Intent.ACTION_BATTERY_CHANGED.equals(action))
            {
                intLevel = intent.getIntExtra("level", 0);
                intScale = intent.getIntExtra("scale", 100);
                onBatteryInfoReceiver(intLevel,intScale);
            }
        }
    }
}

```

```
    }  
};
```

- 在 onCreate 中载入主布局文件 main.xml，然后初始化 Button 和设置点击后的动作，并注册一个系统 BroadcastReceiver，用于访问电池计量。具体代码如下所示。

```
/** Called when the activity is first created. */  
@Override  
public void onCreate(Bundle savedInstanceState)  
{  
    super.onCreate(savedInstanceState);  
    /* 载入main.xml Layout */  
    setContentView(R.layout.main);  
  
    /* 初始化Button，并设置点击后的动作 */  
    mButton01 = (Button) findViewById(R.id.myButton1);  
    mButton01.setOnClickListener(new Button.OnClickListener()  
    {  
        @Override  
        public void onClick(View v)  
        {  
            /* 注册一个系统 BroadcastReceiver，作为访问电池计量之用 */  
            registerReceiver  
            (  
                mBatInfoReceiver,  
                new IntentFilter(Intent.ACTION_BATTERY_CHANGED)  
            );  
        }  
    });  
}
```

- 定义方法 onBatteryInfoReceiver，当捕捉到 ACTION_BATTERY_CHANGED 时要运行的 method，首先创建一个背景模糊的 Window，且将对话窗口放在前景，然后将取得的电池计量显示于 Dialog 中，最后设置返回主画面的按钮。具体代码如下所示。

```
/* 捕捉到ACTION_BATTERY_CHANGED时要运行的method */  
public void onBatteryInfoReceiver(int intLevel, int intScale)  
{  
    /* create 跳出的对话窗口 */  
    final Dialog d = new Dialog(dian.this);  
    d.setTitle(R.string.str_dialog_title);  
    d.setContentView(R.layout.mydialog);  
  
    /* 创建一个背景模糊的window，且将对话窗口放在前景 */
```



```

Window window = d.getWindow();
window.setFlags
(
    WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
    WindowManager.LayoutParams.FLAG_BLUR_BEHIND
);

/* 将取得的电池计量显示于Dialog中 */
TextView mTextView02=(TextView)d.findViewById(R.id.myTextView2);
mTextView02.setText
(
    getResources().getText(R.string.str_dialog_body) +
    String.valueOf(intLevel * 100 / intScale) + "%"
);

/* 设置返回主画面的按钮 */
Button mButton02 = (Button)d.findViewById(R.id.myButton2);
mButton02.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        /* 反注册Receiver，并关闭对话窗口 */
        unregisterReceiver(mBatInfoReceiver);
        d.dismiss();
    }
});
d.show();
}
}

```

执行后的初始效果如图 6-4 所示。当单击【获取】按钮后会显示当前电池的容量，如图 6-5 所示。

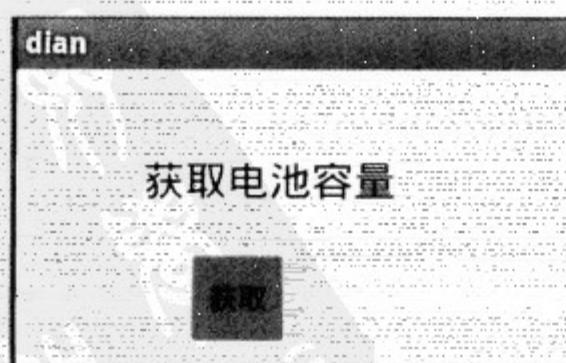


图6-4 初始效果

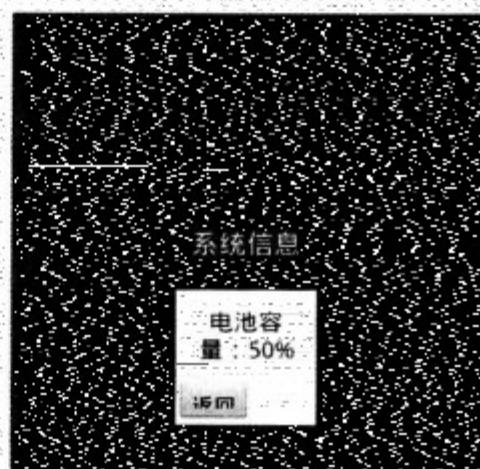


图6-5 显示电池容量

6.3 来短信E-mail通知

本实例的设计流程是：先在后台设计一个 BroadcastReseiver 用于等待接收短信，当收到短信后，以 Bundle 的方式来封装短信的内容，然后通过 Intent 方式返回给主程序 Activity。因为 Receiver 无法直接发送 E-mail，所以需要将控制权回给主程序，通过主程序来运行发送 E-mail 的工作。当主程序收到 Bundle 后，会以 Bundle.getString 的方法来取得返回短信的内容，然后以 Intent.setType(“plain/text”) 来设置要打开的 Intent 类型，并以关键程序 Intent.putExtra(android.content.Intent.EXTRA_EMAIL,strEmailReciver) 来指定要打开的是 E-mail 所需要的 Extra 参数，当 Android 系统收到这些参数后，就会打开内置的 E-mail 发送程序。

本实例的最终目的是：当收到一条短信后，先用 Toast 来提示获取了短信，然后再通过 E-mail 发送到用户的邮箱中，这样就可以将重要的短信放在邮箱中保存，从而不用担心短信容量的问题了。本实例的源码保存在【光盘\daima\第 6 章\youtong】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

step 02 编写文件 practice1.java，其具体实现流程如下。

- 声明一个 TextView，String 数组与两个文本字符串变量，具体代码如下所示。

```
public class youtong extends Activity {
    /*声明一个TextView,String数组与两个文本字符串变量*/
    private TextView mTextView1;
    public String[] strEmailReciver;
    public String strEmailSubject;
```

```
public String strEmailBody;
```

- ◆ 在 onCreate 中通过 findViewById 构造器来创建 TextView 对象，并通过 TextView 来显示“等待接收短信……”的提示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*通过findViewById构造器创建TextView对象*/
    mTextView1 = (TextView) findViewById(R.id.myTextView1);
    mTextView1.setText("等待接收短信……");
```

- ◆ 通过 try 语句取得短信传来的 bundle 对象，并将 bunde 内的字符串取出，然后自定义 Intent 来运行寄送 E-mail 的工作，同时设置邮件格式为 "plain/text"，并分别取得 EditText01,02,03,04 的值作为收件人地址、附件、主题和正文，最后将取得的字符串放入 mEmailIntent 中。具体代码如下所示。

```
try
{
    /*取得短信传来的bundle*/
    Bundle bunde = this.getIntent().getExtras();
    if (bunde!= null)
    {
        /*将bunde内的字符串取出*/
        String sb = bunde.getString("STR_INPUT");
        /*自定义一Intent来运行寄送E-mail的工作*/
        Intent mEmailIntent =
            new Intent(android.content.Intent.ACTION_SEND);
        /*设置邮件格式为"plain/text"*/
        mEmailIntent.setType("plain/text");

        /*
         * 取得EditText01,02,03,04的值作为
         * 收件人地址、附件、主题、正文
         */
        strEmailReciver =new String[]{"jay.mingchieh@gmail.com"};
        strEmailSubject = "你有一封短信!!";
        strEmailBody = sb.toString();

        /*将取得的字符串放入mEmailIntent中*/
    }
}
```



```
mEmailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,
    strEmailReciver);
mEmailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
    strEmailSubject);
mEmailIntent.putExtra(android.content.Intent.EXTRA_TEXT,
    strEmailBody);
startActivity(Intent.createChooser(mEmailIntent,
    getResources().getString(R.string.str_message)));
}
else
{
    finish();
}
}
catch(Exception e)
{
    e.printStackTrace();
}
}
}
```

step 03 编写文件 youtongSMSreceiver.java，其具体实现流程如下。

- ◆ 引用 BroadcastReceiver 类，然后引用 telephony.gsm.SmsMessage 来收取短信，引用 Toast 类来告知用户收到短信，具体代码如下所示。

```
package irdc.youtong;

/*引用BroadcastReceiver类*/
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
/*引用telephony.gsm.SmsMessage来收取短信*/
import android.telephony.gsm.SmsMessage;
/*引用Toast类来告知用户收到短信*/
import android.widget.Toast;
```

- ◆ 自定义继承自 BroadcastReceiver 类，用于聆听系统服务广播的信息，然后声明静态字符串并作为 Action 为短信的依据，具体代码如下所示。

```
public class youtongSMSreceiver extends BroadcastReceiver
{
    * android.provider.Telephony.SMS_RECEIVED
    private static final String mACTION =
```



```
"android.provider.Telephony.SMS_RECEIVED";

private String str_receive="收到短信!";
```

- ◆ 定义 onReceive(Context context, Intent intent) 用于获取短信，先通过 if 语句判断传来 Intent 是否为短信，是则建构一字符串集合变量 sb 并接收由 Intent 传来的数据。具体代码如下所示。

```
@Override
public void onReceive(Context context, Intent intent)
{
    // TODO Auto-generated method stub
    Toast.makeText(context, str_receive.toString(),
    Toast.LENGTH_LONG).show();

    /*判断传来Intent是否为短信*/
    if (intent.getAction().equals(mACTION))
    {
        /*建构一字符串集合变量sb*/
        StringBuilder sb = new StringBuilder();
        /*接收由Intent传来的数据*/
        Bundle bundle = intent.getExtras();
```

- ◆ 用 if 语句判断 Intent 中是否有数据，用 pdus 作为 android 内置短信参数 identifier，并通过 bundle.get("") 返回一包含 pdus 的对象。具体代码如下所示。

```
/*判断Intent是有数据*/
if (bundle != null)
{
    Object[] myOBJpdus = (Object[]) bundle.get("pdus");
```

- ◆ 构造短信对象 array，并依据收到的对象长度来创建 array 的大小。具体代码如下所示。

```
SmsMessage[] messages = new SmsMessage[myOBJpdus.length];

for (int i = 0; i<myOBJpdus.length; i++)
{
    messages[i] =
        SmsMessage.createFromPdu((byte[]) myOBJpdus[i]);
}
```

- ◆ 将传递来的短信集合，然后自定义信息于 StringBuilder 当中。即分别获取收信人的电话号码，传来信息的 BODY。具体代码如下所示。

```
for (SmsMessage currentMessage : messages)
```



```
{  
    sb.append("接收到来自:\n");  
    /* 收信人的电话号码 */  
    sb.append(currentMessage.getDisplayOriginatingAddress());  
    sb.append("\n-----传来的短信-----\n");  
    /* 取得传来信息的BODY */  
    sb.append(currentMessage.getDisplayMessageBody());  
    Toast.makeText  
(  
        context, sb.toString(), Toast.LENGTH_LONG  
    ).show();  
}  
}
```

- ◆ 用 Notification(Toase) 显示来信信息，具体代码如下所示。

```
Toast.makeText  
(  
    context, sb.toString(), Toast.LENGTH_LONG  
).show();
```

- ◆ 返回主 Activity，然后自定义一 Bundle，将短信信息以 putString() 方法存入自定义的 bundle 内，最后设置 Intent 的 Flag 以一个全新的 task 来运行。具体代码如下所示。

```
Intent i = new Intent(context, youtong.class);  
/*自定义一Bundle*/  
Bundle mbundle = new Bundle();  
/*将短信信息以putString()方法存入自定义的bundle内*/  
mbundle.putString("STR_INPUT", sb.toString());  
/*将自定义bundle写入Intent中*/  
i.putExtras(mbundle);  
/*设置Intent的Flag以一个全新的task来运行*/  
i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
context.startActivity(i);  
}  
}  
}
```

- step 04** 在文件 AndroidManifest.xml 中向系统注册一个常驻的 BroadcastReseiver，并设置这个 Reseiver 的 intent-filter，让其 SMSreceiver 针对收到短信事件做出反应，并添加 android.permission.RECEIVE_SMS 权限。具体代码如下所示。

```
<receiver android:name="youtongSMSreceiver">  
    <!--设定要捕捉的信息名称是provider中Telephony.SMS_RECEIVED -->  
    <intent-filter>
```



```

<action
    android:name="android.provider.Telephony.SMS_RECEIVED" />
</intent-filter>
</receiver>
</application>
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-
permission>

```

执行后可以向其发送一条短信，收到会显示提示信息，并生成邮件提示。



注意 读者在此需要注意的是，在模拟器运行后会显示“*No application can perform this action*”的提示，而在真实机器上不会出现此问题。

6.4 来电后显示提示信息

在使用手机时，如果有来电则会在屏幕中显示拨打用户的姓名等基本信息。在 Android 中，可以通过 PhoneStateListener 提供的方法来监听来电状态。在具体实施时，需要创建 PhoneStateListener 对象，并重写其中的 `onCallStateChanged()` 方法，并通过传入的“state”来判断来电状态。

要获取来电状态，需要用户读取电话状态的权限，否则不能成功获取状态。在具体实施上，需要在模拟器中先添加一个联系人记录，并为其起名。这样当电话进来后，会在屏幕上显示他的名字。如果是非通讯录的来电，则在屏幕上显示 Unknown Caller。

本实例实现了来电后屏幕提醒的功能，本实例的源码保存在【光盘\daima\第6章\tixing】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    >
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        >

```



```
</TextView>  
</LinearLayout>
```

step 02 编写主程序文件是 tixing.java，其具体实现流程如下所示。

- ◆ 通过 setContentView 来引用主布局文件 main.xml，然后通过 myTextView1 显示提示。
具体代码如下所示。

```
package irdc.tixing;  
  
import irdc.tixing.R;  
import android.app.Activity;  
import android.content.ContentResolver;  
import android.content.Context;  
import android.database.Cursor;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.provider.Contacts;  
import android.telephony.PhoneStateListener;  
import android.telephony.TelephonyManager;  
import android.widget.TextView;  
  
public class tixing extends Activity  
{  
    private TextView myTextView1;  
  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState)  
    {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        myTextView1 = (TextView) findViewById(R.id.myTextView1);
```

- ◆ 定义 TelephonyManager 对象 tm，用于获取电话服务，然后通过 tm.listen 来注册电话通信 Listener。具体代码如下所示。

```
/* 添加自己实现的PhoneStateListener */  
exPhoneCallListener myPhoneCallListener =  
new exPhoneCallListener();  
  
/* 取得电话服务 */  
TelephonyManager tm =  
(TelephonyManager) this.getSystemService
```

```

(Context.TELEPHONY_SERVICE);

/* 注册电话通信Listener */
tm.listen
(
    myPhoneCallListener,
    PhoneStateListener.LISTEN_CALL_STATE
);
}

```

- ◆ 使用内部 class 来继承 PhoneStateListener，重写 onCallStateChanged，这样当状态改变时改变 myTextView1 的文字及颜色。然后分别设置无任何状态、接起电话、电话进来的显示。具体代码如下所示。

```

public class exPhoneCallListener extends PhoneStateListener
{
    /* 重写onCallStateChanged
    当状态改变时改变myTextView1的文字及颜色 */
    public void onCallStateChanged(int state, String incomingNumber)
    {
        switch (state)
        {
            /* 无任何状态时 */
            case TelephonyManager.CALL_STATE_IDLE:
                myTextView1.setTextColor
                (
                    getResources().getColor(R.drawable.red)
                );
                myTextView1.setText("CALL_STATE_IDLE");
                break;
            /* 接起电话时 */
            case TelephonyManager.CALL_STATE_OFFHOOK:
                myTextView1.setTextColor
                (
                    getResources().getColor(R.drawable.green)
                );
                myTextView1.setText("CALL_STATE_OFFHOOK");
                break;
            /* 电话进来时 */
            case TelephonyManager.CALL_STATE_RINGING:
                getContactPeople(incomingNumber);
                break;
            default:

```



```
        break;
    }
    super.onCallStateChanged(state, incomingNumber);
}
}
```

- ◆ 通过方法 `getContactPeople(String incomingNumber)` 来获取机器内的联系人信息，然后在 `cursor` 里存放要放的字段名称。具体代码如下所示。

```
private void getContactPeople(String incomingNumber)
{
    myTextView1.setTextColor(Color.BLUE);
    ContentResolver contentResolver = getContentResolver();
    Cursor cursor = null;

    /* cursor里要放的字段名称 */
    String[] projection = new String[]
    {
        Contacts.People._ID,
        Contacts.People.NAME,
        Contacts.People.NUMBER
    };
}
```

- ◆ 通过来电号码来查找对应的联系人，查找到则显示姓名，没有查找到则只显示号码。具体代码如下所示。

```
/* 用来自电号码去找该联系人 */
cursor = contentResolver.query(
    Contacts.People.CONTENT_URI, projection,
    Contacts.People.NUMBER + "=?",
    new String[]
    {
        incomingNumber
    },
    Contacts.People.DEFAULT_SORT_ORDER
);

/* 找不到联系人 */
if (cursor.getCount() == 0)
{
    myTextView1.setText("unknown Number:" + incomingNumber);
}
else if (cursor.getCount() > 0)
```

```
    {
        cursor.moveToFirst();
        /* 在projection这个数组里名字是放在第1个位置 */
        String name = cursor.getString(1);
        myTextView1.setText(name + ":" + incomingNumber);
    }
}
```

step 03 在文件 AndroidManifest.xml 中获取如下两个权限。

- 读取通讯录权限： android.permission.READ_CONTACTS
 - 获取电话状态权限： android.permission.READ_PHONE_STATE

设置权限代码如下所示。

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

执行后的效果如图 6-6 所示，当拨打电话后会显示来电的基本信息，如图 6-7 所示。

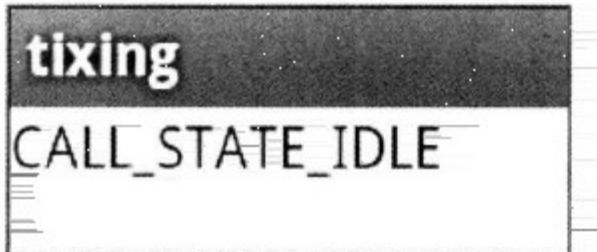


图6-6 初始效果



图6-7 来电后界面

6.5 获取手机存储卡的容量

存储卡是可以随时插拔的，每次插拔时会对操作系统进行 ACTION broadcast。在使用手机时经常需要及时了解存储卡的容量信息。在本实例中，将通过 StatFs 文件系统的方法来取得 MicroSD 存储卡的剩余容量。在具体实施时，需要首先判断是否安装存储卡，如



果不存在则直接不计算。为了更好地显示容量，在布局中插入了一个 ProgressBar Widget，这样将更加一目了然。

6.5.1 一些基本知识

要想在模拟器中使用存储卡功能，读者需要了解下面的知识。

Android 模拟器能够让我们使用 fat32 格式的磁盘镜像作为 SD 卡的模拟，具体过程如下：

step 01 进入 Android SDK 目录下的 tools 子目录，运行：

```
mksdcard -l sdcard 512M /your_path_for_img/sdcard.img
```

这样就创建了一个 512MB 的 SD 卡镜像文件。

step 02 运行模拟器的时候指定路径（注意需要完整路径）：

```
emulator -sdcard /your_path_for_img/sdcard.img
```

这样模拟器中就可以使用 “/sdcard” 这个路径来指向模拟的 SD 卡了。

那么如何复制本机文件到 SD 卡，或者管理 SD 卡上的内容呢？有如下两种方案：

第 1 种：在 Linux 系统我们可以 mount 成一个 loop 设备，先创建一个目录，比如叫 android_sdcard，然后执行：

```
mount -o loop sdcard.img android_sdcard
```

这样管理这个目录就是管理 sdcard 内容了。

第 2 种：在 Windows 系统我们可以用 mtools 来做管理，也可以用 android SDK 带的命令（这个命令在 linux 下面也可以用）：

```
adb push local_file sdcard/remote_file
```

在使用 mksdcard 命令时要注意如下 6 点：

- mycard 命令可以使用三种尺寸：字节、K 和 M。如果只使用数字，表示字节。后面还可以跟 K，如 262144K，也表示 256M。
- mycard 建立的虚拟文件最小为 8MB，也就是说，模拟器只支持大于 8MB 的虚拟文件。
- -l 命令行参数表示虚拟磁盘的卷标，可以没有该参数。
- 虚拟文件的扩展名可以是任意的，如 mycard.abc。
- mksdcard 命令不会自动建立不存在的目录，因此，在执行上面命令之前，要先在当前目录中建立一个 card 目录。
- mksdcard 命令是按实际大小生成的 sdcard 虚拟文件。也就是说，生成 256MB 的虚拟文件的尺寸就是 256MB，如果生成较大的虚拟文件，要看看自己的硬盘空间是否够不够。

在执行完上面的命令后，执行下面的命令启动 android 模拟器：

```
emulator -avd avd1 -sdcard card/mycard.img
```

如果在开发环境（Eclipse）中，可以在 Run Configuration 对话框中设置启动参数，当然，也可以在 Preferences 对话框中设置默认启动参数。这样在新建立的 Android 工程中就自动加入了装载 sdcard 虚拟文件的命令行参数。

如果读者使用 OPhone 虚拟机，设置的方法也是完全一样的。然后在虚拟机中的 Setting 里看看 sdcard，是否找到，那么如何查看 sdcard 虚拟设备中的内容呢？方法很多，最简单的就是使用 android eclipse 插件带的 DDMS 透视图。

6.5.2 具体实现

本实例的源码保存在【光盘\daima\第6章\cunchu】，具体实现流程如下所示。

step 01 编写布局文件 main.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:id="@+id/myButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/strButton"
        >
    </Button>
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >
    </TextView>
    <ProgressBar
        android:id="@+id/myProgressBar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="0"
        >
    </ProgressBar>
</LinearLayout>
```

**step 02** 编写主程序文件是 cunxhu.java，其具体实现流程如下所示。

- ◆ 先分别定义 3 个参数 myButton、myProgressBar 和 myTextView，然后通过 findViewById 构造 3 个对象 myButton、myProgressBar 和 myTextView。具体代码如下所示。

```
package irdc.cunxhu;

import irdc.cunxhu.R;

import java.io.File;
import java.text.DecimalFormat;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import android.os.StatFs;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;

public class cunxhu extends Activity
{
    private Button myButton;
    private ProgressBar myProgressBar;
    private TextView myTextView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        myButton = (Button) findViewById(R.id.myButton);
        myProgressBar = (ProgressBar) findViewById(R.id.myProgressBar);
        myTextView = (TextView) findViewById(R.id.myTextView);
    }

    • 定义 setOnClickListener，用于触发按钮单击事件处理程序。具体代码如下所示。
        myButton.setOnClickListener(new Button.OnClickListener()
        {

            @Override
            public void onClick(View arg0)
```

```

    {
        showSize();
    }
});
}
}

```

- ◆ 定义方法 showSize() 来显示存储卡的容量大小，具体代码如下所示。

```

private void showSize()
{
    /* 将TextView及ProgressBar设置为空值及0 */
    myTextView.setText("");
    myProgressBar.setProgress(0);
    /* 判断存储卡是否插入 */
    if (Environment.getExternalStorageState().equals(Environment.
MEDIA_MOUNTED))
    {
        /* 取得SD CARD文件路径一般是/sdcard */
        File path = Environment.getExternalStorageDirectory();

        /* StatFs看文件系统空间使用状况 */
        StatFs statFs = new StatFs(path.getPath());
        /* Block的size*/
        long blockSize = statFs.getBlockSize();
        /* 总Block数量 */
        long totalBlocks = statFs.getBlockCount();
        /* 已使用的Block数量 */
        long availableBlocks = statFs.getAvailableBlocks();

        String[] total = fileSize(totalBlocks * blockSize);
        String[] available = fileSize(availableBlocks * blockSize);

        /* getMax取得在main.xml里ProgressBar设置的最大值 */
        int ss = Integer.parseInt(available[0]) * myProgressBar.getMax()
            / Integer.parseInt(total[0]);

        myProgressBar.setProgress(ss);
        String text = "总共" + total[0] + total[1] + "\n";
        text += "可用" + available[0] + available[1];
        myTextView.setText(text);

    } else if (Environment.getExternalStorageState().equals(
        Environment.MEDIA_REMOVED))
    {
        myTextView.setText("存储卡未插入");
    }
}

```



```
    String text = "SD CARD已删除";
    myTextView.setText(text);
}

/*返回为字符串数组[0]为大小[1]为单位KB或MB*/
private String[] fileSize(long size)
{
    String str = "";
    if (size >= 1024) -
    {
        str = "KB";
        size /= 1024;
        if (size >= 1024)
        {
            str = "MB";
            size /= 1024;
        }
    }
    DecimalFormat formatter = new DecimalFormat();
    /* 每3个数字用,分隔如: 1,000 */
    formatter.setGroupingSize(3);
    String result[] = new String[2];
    result[0] = formatter.format(size);
    result[1] = str;
    return result;
}
}
```

上述代码的具体实现流程如下所示。

- 将 TextView 及 ProgressBar 设置为空值及 0。
- 获取 SD CARD 文件路径。
- 通过 StatFs 来查看文件系统空间使用状况。
- 分别获取总 Block 数量和已使用的 Block 数量。
- 通过 getMax 获取在 main.xml 里 ProgressBar 设置的最大值。
- 显示出容量信息。
- 如果没有 SD 卡则输出“SD CARD 已删除”的提示。

执行后的效果如图 6-8 所示。

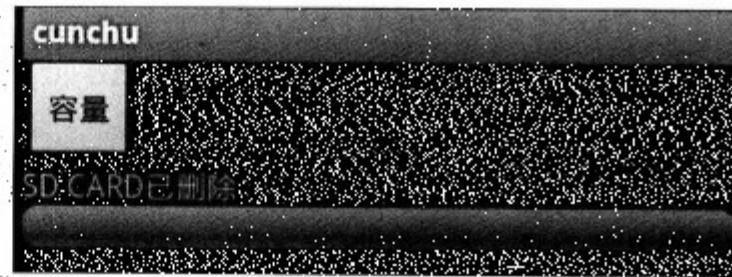


图6-8 执行效果



6.6 闹钟到时响

手机闹钟，对大家来说都不陌生。在Android中提供了AlarmManager类，通过此类可以设置在指定时间运行某些动作。在Android中，内置了Alarm Clock，通过它可以实现闹钟功能。本实例通过编程方式实现了一个闹钟功能，本实例的源码保存在【光盘\ daima\ 第6章\ nao】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/layout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
>
    <DigitalClock
        android:id="@+id/dClock"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="40sp"
        android:textColor="@drawable/blue"
        android:layout_x="70px"
        android:layout_y="32px"
    >
    </DigitalClock>
    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_title1"
        android:textSize="20sp"
        android:textColor="@drawable/black"
        android:layout_x="10px"
        android:layout_y="102px"
    >
    </TextView>
    <Button
        android:id="@+id/mButton1"
        android:layout_width="wrap_content"
        android:layout_height="40px"
```

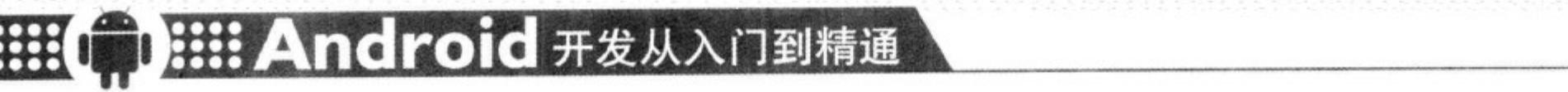
```
    android:text="@string/str_button1"
    android:textColor="@drawable/black"
    android:textSize="18sp"
    android:layout_x="190px"
    android:layout_y="102px"
>
</Button>
<Button
    android:id="@+id/mButton2"
    android:layout_width="wrap_content"
    android:layout_height="40px"
    android:text="@string/str_button2"
    android:textColor="@drawable/black"
    android:textSize="18sp"
    android:layout_x="190px"
    android:layout_y="145px"
>
</Button>
<TextView
    android:id="@+id/text2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_title2"
    android:textSize="20sp"
    android:textColor="@drawable/black"
    android:layout_x="10px"
    android:layout_y="212px"
>
</TextView>
<Button
    android:id="@+id/mButton3"
    android:layout_width="wrap_content"
    android:layout_height="40px"
    android:text="@string/str_button1"
    android:textColor="@drawable/black"
    android:textSize="18sp"
    android:layout_x="190px"
    android:layout_y="212px"
>
</Button>
<Button
    android:id="@+id/mButton4"
    android:layout_width="wrap_content"
```

```
    android:layout_height="40px"
    android:text="@string/str_button2"
    android:textColor="@drawable/black"
    android:textSize="18sp"
    android:layout_x="190px"
    android:layout_y="249px"
  >
</Button>
<TextView
    android:id="@+id/setTime1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_default"
    android:textColor="@drawable/red"
    android:textSize="16sp"
    android:layout_x="10px"
    android:layout_y="142px"
  >
</TextView>
<TextView
    android:id="@+id/setTime2"
    android:layout_width="170px"
    android:layout_height="wrap_content"
    android:text="@string/str_default"
    android:textColor="@drawable/red"
    android:textSize="16sp"
    android:layout_x="10px"
    android:layout_y="252px"
  >
</TextView>
</AbsoluteLayout>
```

step 02 编写文件 nao.java，其具体实现流程如下。

- ◆ 分别定义 6 个对象变量 setTime1、setTime2、mButton1、mButton2、mButton3、mButton4，然后通过 findViewById 构造 3 个对象 myButton、myProgressBar 和 myTextView。具体代码如下所示。

```
public class nao extends Activity
{
    /* 声明对象变量 */
    TextView setTime1;
    TextView setTime2;
    Button mButton1;
    Button mButton2;
```



```
Button mButton3;
Button mButton4;
Calendar c=Calendar.getInstance();
```

- ◆ 载入主布局文件 main.xml，通过 setTime1 实现只响一次的闹钟的设置，然后实现只响一次的闹钟的设置 Button1。具体代码如下所示。

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    /* 载入main.xml Layout */
    setContentView(R.layout.main);

    /* 以下为只响一次的闹钟的设置 */
    setTime1=(TextView) findViewById(R.id.setTime1);
    /* 只响一次的闹钟的设置Button */
    mButton1=(Button)findViewById(R.id.mButton1);
    mButton1.setOnClickListener(new View.OnClickListener()
    {
        public void onClick(View v)
        {
            /* 取得点击按钮时的时间作为TimePickerDialog的默认值 */
            c.setTimeInMillis(System.currentTimeMillis());
            int mHour=c.get(Calendar.HOUR_OF_DAY);
            int mMinute=c.get(Calendar.MINUTE);
        }
    });
}
```

- ◆ 通过 TimePickerDialog 来弹出一个对话框供用户来设置时间，具体代码如下所示。

```
/* 跳出TimePickerDialog来设置时间 */
new TimePickerDialog(nao.this,
    new TimePickerDialog.OnTimeSetListener()
    {
        public void onTimeSet(TimePicker view, int hourOfDay,
                             int minute)
        {
            /* 取得设置后的时间，秒跟毫秒设为0 */
            c.setTimeInMillis(System.currentTimeMillis());
            c.set(Calendar.HOUR_OF_DAY,hourOfDay);
            c.set(Calendar.MINUTE,minute);
            c.set(Calendar.SECOND,0);
            c.set(Calendar.MILLISECOND,0);

            /* 指定闹钟设置时间到时要运行CallAlarm.class */
        }
    });
}
```

```

Intent intent = new Intent(nao.this, nao_2.class);
/* 创建PendingIntent */
PendingIntent sender=PendingIntent.getBroadcast(
    nao.this, 0, intent, 0);
/* AlarmManager.RTC_WAKEUP设置服务在系统休眠时同样会运行
 * 以set()设置的PendingIntent只会运行一次
 */
AlarmManager am;
am = (AlarmManager) getSystemService(ALARM_SERVICE);
am.set(AlarmManager.RTC_WAKEUP,
    c.getTimeInMillis(),
    sender
);
/* 更新显示的设置闹钟时间 */
String tmpS=format(hourOfDay)+": "+format(minute);
setTime1.setText(tmpS);
/* 以Toast提示设置已完成 */
Toast.makeText(nao.this,"设置闹钟时间为"+tmpS,
    Toast.LENGTH_SHORT)
.show();
}
},mHour,mMinute,true).show();
}
});

```

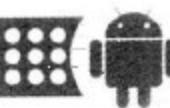
上述代码的具体实现流程如下所示。

- 获取设置后的时间。
- 指定闹钟设置时间到时要运行 CallAlarm.class。
- 创建 PendingIntent。
- 通过 AlarmManager.RTC_WAKEUP 设置服务在系统休眠时同样会运行。
- 定义 tmpS，用于更新显示的设置闹钟时间。
- 以 Toast 提示设置已完成。
- ◆ 设置按钮 mButton2，用于实现只响一次的闹钟的删除。首先在 AlarmManager 中实现删除，然后通过 Toast 提示已删除设置，并更新显示的闹钟时间。具体代码如下所示。

```

/* 只响一次的闹钟的删除Button */
mButton2=(Button) findViewById(R.id.mButton2);
mButton2.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v)
    {

```



```
Intent intent = new Intent(nao.this, nao_2.class);
PendingIntent sender=PendingIntent.getBroadcast(
    nao.this,0, intent, 0);
/* 由AlarmManager中删除 */
AlarmManager am;
am =(AlarmManager) getSystemService(ALARM_SERVICE);
am.cancel(sender);
/* 以Toast提示已删除设置，并更新显示的闹钟时间 */
Toast.makeText(nao.this,"闹钟时间解除",
    Toast.LENGTH_SHORT).show();
setTime1.setText("目前无设置");
}
});
```

- ◆ 开始设置重复响起的闹钟，具体代码如下所示。

```
/* 以下为重复响起的闹钟的设置 */
setTime2=(TextView) findViewById(R.id.setTime2);
/* create重复响起的闹钟的设置画面 */
/* 引用timeset.xml为Layout */
LayoutInflater factory = LayoutInflater.from(this);
final View setView = factory.inflate(R.layout.timeset,null);
final TimePicker tPicker=(TimePicker)setView
    .findViewById(R.id.tPicker);
tPicker.setIs24HourView(true);

/* create重复响起闹钟的设置Dialog */
final AlertDialog di=new AlertDialog.Builder(nao.this)
    .setIcon(R.drawable.clock)
    .setTitle("设置")
    .setView(setView)
    .setPositiveButton("确定",
        new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int which)
        {
            /* 取得设置的间隔秒数 */
            EditText ed=(EditText)setView.findViewById(R.id.mEdit);
            int times=Integer.parseInt(ed.getText().toString())
                *1000;
            /* 取得设置的开始时间，秒及毫秒设为0 */
            c.setTimeInMillis(System.currentTimeMillis());
            c.set(Calendar.HOUR_OF_DAY,tPicker.getCurrentHour());
            c.set(Calendar.MINUTE,tPicker.getCurrentMinute());
```



```
c.set(Calendar.SECOND, 0);
c.set(Calendar.MILLISECOND, 0);

/* 指定闹钟设置时间到时要运行CallAlarm.class */
Intent intent = new Intent(nao.this,
                             nao_2.class);
PendingIntent sender = PendingIntent.getBroadcast(
    nao.this, 1, intent, 0);
/* setRepeating() 可让闹钟重复运行 */
AlarmManager am;
am = (AlarmManager) getSystemService(ALARM_SERVICE);
am.setRepeating(AlarmManager.RTC_WAKEUP,
                c.getTimeInMillis(), times, sender);
/* 更新显示的设置闹钟时间 */
String tmpS=format(tPicker.getCurrentHour())+": "+
             format(tPicker.getCurrentMinute());
setText2.setText("设置闹钟时间为"+tmpS+
                 "开始，重复间隔为"+times/1000+"秒");
/* 以Toast提示设置已完成 */
Toast.makeText(nao.this,"设置闹钟时间为"+tmpS+
               "开始，重复间隔为"+times/1000+"秒",
               Toast.LENGTH_SHORT).show();
}
})
.setNegativeButton("取消",
new DialogInterface.OnClickListener()
{
    public void onClick(DialogInterface dialog, int which)
    {
    }
}).create();
```

上述代码的具体流程如下所示。

- 以 create 重复响起的闹钟的设置画面，并引用 timeset.xml 为布局文件。
- 以 create 重复响起闹钟的设置 Dialog 对话框。
- 获取设置的间隔秒数。
- 获取设置的开始时间，秒及毫秒都设为 0。
- 指定闹钟设置时间到时要运行 CallAlarm.class。
- 通过 setRepeating() 可让闹钟重复运行。
- 通过 tmpS 更新显示的设置闹钟时间。
- 通过以 Toast 提示设置已完成。
- 实现重复响起的闹钟的设置 Button 按钮处理事件，具体代码如下所示。

```
/* 重复响起的闹钟的设置Button */
mButton3=(Button) findViewById(R.id.mButton3);
mButton3.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v)
    {
        /* 取得点击按钮时的时间作为tPicker的默认值 */
        c.setTimeInMillis(System.currentTimeMillis());
        tPicker.setCurrentHour(c.get(Calendar.HOUR_OF_DAY));
        tPicker.setCurrentMinute(c.get(Calendar.MINUTE));
        /* 跳出设置画面di */
        di.show();
    }
});
```

- ◆ 开始设置重复响起的闹钟的删除 Button 按钮处理事件，首先在 AlarmManager 中删除，然后以 Toast 提示已删除设置，并更新显示的闹钟时间。具体代码如下所示。

```
/* 重复响起的闹钟的删除Button */
mButton4=(Button) findViewById(R.id.mButton4);
mButton4.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v)
    {
        Intent intent = new Intent(nao.this, nao_2.class);
        PendingIntent sender = PendingIntent.getBroadcast(
            nao.this, 1, intent, 0);
        /* 由AlarmManager中删除 */
        AlarmManager am;
        am = (AlarmManager) getSystemService(ALARM_SERVICE);
        am.cancel(sender);
        /* 以Toast提示已删除设置，并更新显示的闹钟时间 */
        Toast.makeText(nao.this, "闹钟时间解除",
                      Toast.LENGTH_SHORT).show();
        setTime2.setText("目前无设置");
    }
});
```

- ◆ 定义方法 format(int x)，用于设置日期时间示两位数的显示格式。具体代码如下所示。

```
/* 日期时间显示两位数的方法 */
private String format(int x)
```

```

        String s="" +x;
        if(s.length()==1) s="0"+s;
        return s;
    }
}

```

step 03 文件 nao_1.java，其具体实现流程如下。

- ◆ import 相关 class。
- ◆ 实现实际跳出闹铃 Dialog 的 Activity。
- ◆ 通过 AlertDialog.Builder(nao_1.this) 实现弹出闹钟警示框。
- ◆ 通过 onClick(DialogInterface dialog, int whichButton) 关闭 Activity。

文件 nao_1.java 的具体代码如下所示。

```

/* import相关class */
import irdc.nao.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;

/* 实际跳出闹铃Dialog的Activity */
public class nao_1 extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 跳出的闹铃警示 */
        new AlertDialog.Builder(nao_1.this)
            .setIcon(R.drawable.clock)
            .setTitle("闹钟响了!!!")
            .setMessage("赶快起床吧!!!")
            .setPositiveButton("关掉他",
                new DialogInterface.OnClickListener()
                {
                    public void onClick(DialogInterface dialog, int whichButton)
                    {
                        /* 关闭Activity */
                        nao_1.this.finish();
                    }
                })
            .show();
    }
}

```

step 04 编写文件 nao_2.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类。
- ◆ 调用闹钟 Alert 的 Receiver。
- ◆ 创建 Intent，用于调用 AlarmAlert.class。

文件 nao_2.java 的具体代码如下所示。

```
package irdc.nao;

/* import相关class */
import android.content.Context;
import android.content.Intent;
import android.content.BroadcastReceiver;
import android.os.Bundle;

/* 调用闹钟Alert的Receiver */
public class nao_2 extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        /* create Intent，调用AlarmAlert.class */
        Intent i = new Intent(context, nao_1.class);

        Bundle bundleRet = new Bundle();
        bundleRet.putString("STR_CALLER", "");
        i.putExtras(bundleRet);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
    }
}
```

step 05 在文件 AndroidManifest.xml 中添加对 CallAlarm 的 receiver 设置，具体代码如下所示。

```
<!—注册receiver CallAlarm -->
<receiver android:name=".nao_2" android:process=":remote" />
<activity android:name=".nao_1" android:label="@string/app_name">
</activity>
```

执行后的初始效果如图 6-9 所示，单击第一个【设置】按钮后在弹出的界面中可以设置闹钟时间，如图 6-10 所示。单击第二个按钮可以设置重复响起的时间，如图 6-11 所示。闹钟响起后的界面如图 6-12 所示。



图6-9 初始效果

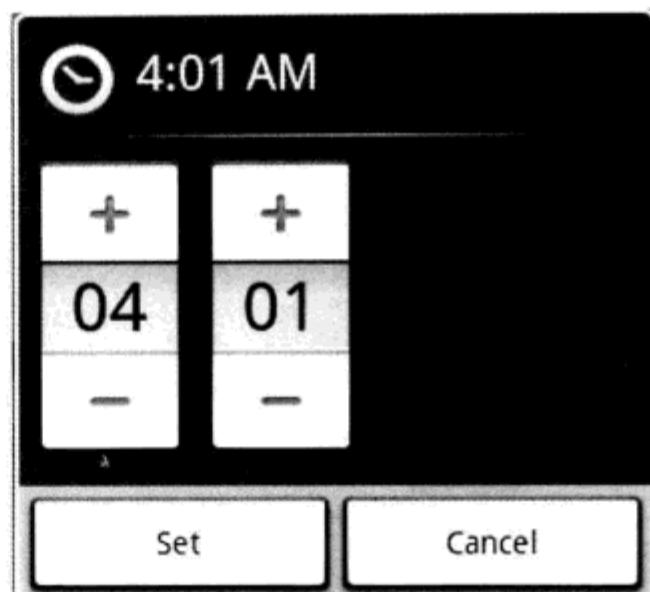


图6-10 响一次的设置界面



图6-11 重复响的设置界面

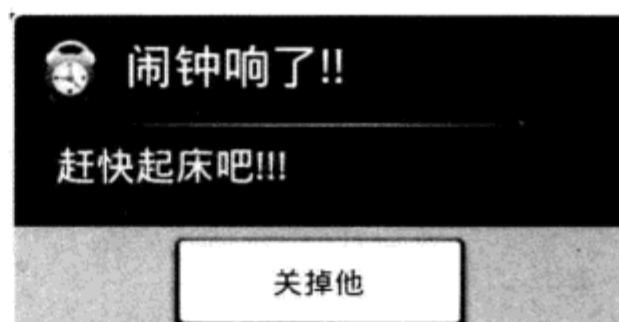


图6-12 闹钟响起后界面

6.7 黑名单来电自动静音

当前几乎每个手机中都有黑名单功能，被列入黑名单的用户不能打进电话和发进短信。在本实例中，首先添加一个 EditText，用户在里面可以输入黑名单用户的号码。当此号码来电时，系统会自动切换为静音模式。当对方挂机后，再自动转换为正常模式，并使用 Toast 实现提示。本实例的源码保存在【光盘\daima\第 6 章\hei】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```



```
    android:layout_height="fill_parent"
    >
    <!-- 建立第一个TextView -->
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <!-- 建立第二个TextView -->
    <TextView
        android:id="@+id/myTextView2"
        android:layout_height="wrap_content"
        android:layout_x="3px"
        android:text="@string/input" android:layout_y="80px" android:layout_
width="135px"/>
    <!-- 建立第三个TextView -->
    <TextView
        android:id="@+id/myTextView3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_y="80px"
        android:layout_x="130px"/>
    <!-- 建立一个EditText -->
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="185px"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:phoneNumber="true"
        android:layout_x="3px"
        android:layout_y="32px"
    />
</AbsoluteLayout>
```

step 02 编写主程序文件 hei.java，其具体实现流程如下所示。

- ◆ 先分别定义 3 个对象变量 mTextView01、mEditText1 和 mEditText03，具体代码如下所示。

```
package irdc.hei;

import irdc.hei.R;
import android.app.Activity;
```

```
import android.content.Context;
import android.media.AudioManager;
import android.os.Bundle;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.view.KeyEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class hei extends Activity
{
    private TextView mTextView01;
    private TextView mTextView03;
    private EditText mEditText1;
```

- ◆ 设置PhoneCallListener对象phoneListener，用TelephonyManager抓取Telephony Sevice，然后设置Listen Call，并查找TextView EditText的数据。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*设置PhoneCallListener*/
    mPhoneCallListener phoneListener=new mPhoneCallListener();
    /*用TelephonyManager抓取Telephony Sevice*/
    TelephonyManager telMgr = (TelephonyManager) getSystemService(
        TELEPHONY_SERVICE);
    /*设置Listen Call*/
    telMgr.listen(phoneListener, mPhoneCallListener,
        LISTEN_CALL_STATE);

    /*查找TextView  EditText*/
    mTextView01 = (TextView)findViewById(R.id.myTextView1);
    mTextView03 = (TextView)findViewById(R.id.myTextView3);
    mEditText1 = (EditText)findViewById(R.id.myEditText1);
}
```

- 判断 PhoneStateListener 当前状态，具体代码如下所示。

```
/* 判断PhoneStateListener当前状态 */
public class mPhoneCallListener extends PhoneStateListener
{
    @Override
    public void onCallStateChanged(int state, String incomingNumber)
    {
        // TODO Auto-generated method stub
        switch(state)
        {
            /* 获取手机待机状态 */
            case TelephonyManager.CALL_STATE_IDLE:
                mTextView01.setText(R.string.str_CALL_STATE_IDLE);

                try
                {
                    AudioManager audioManager = (AudioManager)
                        getSystemService(Context.AUDIO_SERVICE);
                    if (audioManager != null)
                    {
                        /*设置手机为待机时响铃正常*/
                        audioManager.setRingerMode(AudioManager.
                            RINGER_MODE_NORMAL);
                        audioManager.getStreamVolume(
                            AudioManager.STREAM_RING);
                    }
                }
                catch(Exception e)
                {
                    mTextView01.setText(e.toString());
                    e.printStackTrace();
                }
                break;

            /* 获取手机状态为通话中 */
            case TelephonyManager.CALL_STATE_OFFHOOK:
                mTextView01.setText(R.string.str_CALL_STATE_OFFHOOK);
                break;

            /* 获取手机状态为来电 */
            case TelephonyManager.CALL_STATE_RINGING:
                /*显示来电信息 */
                mTextView01.setText(
```

```

        getResources().getText(R.string.str_CALL_STATE_RINGING) +
        incomingNumber);

    /* 判断输入电话是否一致，一样时用静音 */
    if(incomingNumber.equals(mTextView03.getText().toString()))
    {
        try
        {
            AudioManager audioManager = (AudioManager)
                getSystemService(Context.AUDIO_SERVICE);
            if (audioManager != null)
            {
                /*设置响铃为静音 */
                audioManager.setRingerMode(AudioManager.
                    RINGER_MODE_SILENT);
                audioManager.getStreamVolume(
                    AudioManager.STREAM_RING);
                Toast.makeText(hei.this, getString(R.string.str_msg)
                    ,Toast.LENGTH_SHORT).show();
            }
        }
        catch(Exception e)
        {
            mTextView01.setText(e.toString());
            e.printStackTrace();
            break;
        }
    }

    super.onCallStateChanged(state, incomingNumber);

    mEditText1.setOnKeyListener(new EditText.OnKeyListener()
    {

```

- ◆ 定义 onKey(View v, int keyCode, KeyEvent event)，用于设置 EditText 的输入数据显示在 TextView。具体代码如下所示。

```

public boolean onKey(View v, int keyCode, KeyEvent event)
{
    // TODO Auto-generated method stub
    /*设置EditText的输入数据显示在TextView*/
    mTextView03.setText(mEditText1.getText());
    return false;

```



执行后的初始效果如图 6-13 所示，在输入框中可以输入黑名单号码，如图 6-14 所示。这样当此号码来电时会是静音模式，如图 6-15 所示。

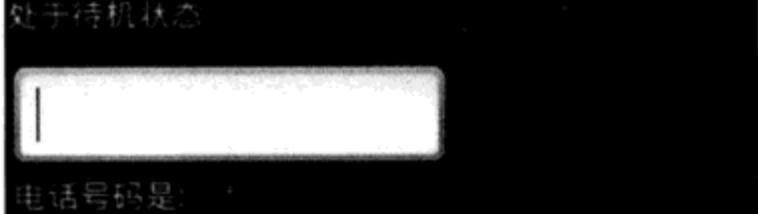


图6-13 初始效果

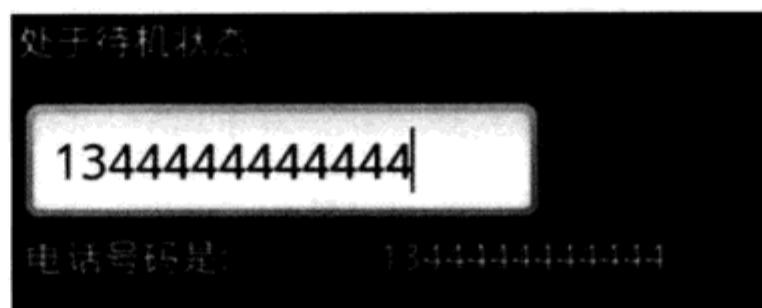


图6-14 设置黑名单号码

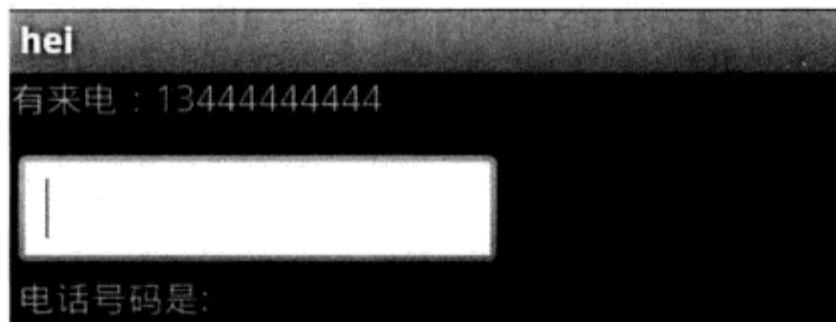
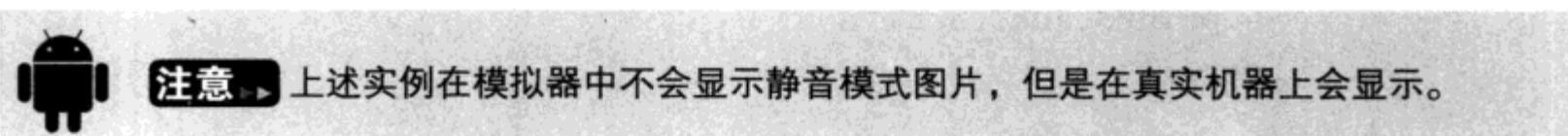


图6-15 来电静音



6.8 监听发送的短信是否成功

当发送短信后，我们往往比较关心是否发送成功。本实例通过编程的方式实现了检测短信是否发送成功的功能。实例设置当发送一条短信后，会及时提供一条信息，说明短信是发送成功还是失败。手机的默认程序可以捕捉到发送状态，这是因为经过系统广播的信息，程序可以捕捉到发送结果。本实例的源码保存在【光盘\daima\第6章\jianting】，具体实现流程如下所示。

step 01 编写布局文件 strings.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```

<string name="hello">jianting</string>
<string name="app_name">jianting</string>
<string name="str_mobile_phone">发送手机号:</string>
<string name="str_sms_body">短信内容:</string>
<string name="str_button1">发送</string>
<string name="str_sms_sending">在发送SMS...</string>
<string name="str_sms_sent_success">成功发送</string>
<string name="str_sms_sent_failed">发送失败</string>
</resources>

```

step 02 编写主程序文件 jianting.java，其具体实现流程如下所示。

- ◆ 创建两个 mServiceReceiver 对象，作为类成员变量。然后分别创建 mButton1、mButton2、mTextView01、mEditText1 和 mEditText2。具体代码如下所示。

```

package irdc.jianting;

import irdc.jianting.R;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.telephony.gsm.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class jianting extends Activity
{
    /* 创建两个mServiceReceiver对象，作为类成员变量 */
    private mServiceReceiver mReceiver01, mReceiver02;
    private Button mButton1;
    private TextView mTextView01;
    private EditText mEditText1, mEditText2;
}

```

- ◆ 自定义 ACTION 常数，作为广播的 Intent Filter 识别常数。然后通过 mEditText1 获得电话号码，通过 mEditText2 获得信息内容，然后设置默认为 5556 表示第二个模拟器的 Port。具体代码如下所示。

```
/* 自定义ACTION常数，作为广播的Intent Filter识别常数 */
private String SMS_SEND_ACTION = "SMS_SEND_ACTION";
private String SMS_DELIVERED_ACTION = "SMS_DELIVERED_ACTION";

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mTextView01 = (TextView) findViewById(R.id.myTextView1);

    /* 电话号码 */
    mEditText1 = (EditText) findViewById(R.id.myEditText1);

    /* 短信内容 */
    mEditText2 = (EditText) findViewById(R.id.myEditText2);
    mButton1 = (Button) findViewById(R.id.myButton1);

    //mEditText1.setText("+12345678");
    /* 设置默认为5556表示第二个模拟器的Port */
    mEditText1.setText("5556");
    mEditText2.setText("Hello DavidLanz!");
}
```

- ◆ 定义 setOnClickListener 作为发送 SMS 短信按钮事件处理程序，strDestAddress 对象是欲发送的电话号码，strMessage 对象是要发送的内容。具体代码如下所示。

```
/* 发送SMS短信按钮事件处理 */
mButton1.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub

        /* 欲发送的电话号码 */
        String strDestAddress = mEditText1.getText().toString();

        /* 欲发送的短信内容 */
        String strMessage = mEditText2.getText().toString();
    }
});
```

- ◆ 创建 SmsManager 对象 smsManager 来发送短信，具体代码如下所示。

```

/* 创建SmsManager对象 */
SmsManager smsManager = SmsManager.getDefault();

// TODO Auto-generated method stub
try
{
    /* 创建自定义Action常数的Intent(给PendingIntent参数之用) */
    Intent itSend = new Intent(SMS_SEND_ACTION);
    Intent itDeliver = new Intent(SMS_DELIVERED_ACTION);

    /* sentIntent参数为传送后接受的广播信息PendingIntent */
    PendingIntent mSendPI = PendingIntent.getBroadcast
        (getApplicationContext(), 0, itSend, 0);

    /* deliveryIntent参数为送达后接受的广播信息PendingIntent */
    PendingIntent mDeliverPI = PendingIntent.getBroadcast
        (getApplicationContext(), 0, itDeliver, 0);

    /* 发送SMS短信, 注意倒数的两个PendingIntent参数 */
    smsManager.sendTextMessage
        (strDestAddress, null, strMessage, mSendPI, mDeliverPI);

    mTextView01.setText(R.string.str_sms_sending);
}
catch(Exception e)
{
    mTextView01.setText(e.toString());
    e.printStackTrace();
}
}
);
}
}

```

- ◆ 自定义 mServiceReceiver，用于覆盖 BroadcastReceiver 聆听短信状态信息。如果发送短信成功则输出“成功发送”提示，如果发送短信失败则输出“发送失败”提示。具体代码如下所示。

```

/* 自定义mServiceReceiver覆盖BroadcastReceiver聆听短信状态信息 */
public class mServiceReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)

```

```
{  
    // TODO Auto-generated method stub  
  
    try  
    {  
        /* android.content.BroadcastReceiver.getResultCode() 方法 */  
        switch(getResultCode())  
        {  
            case Activity.RESULT_OK:  
                /* 发送短信成功 */  
                //mTextView01.setText(R.string.str_sms_sent_success);  
                mMakeTextToast  
                (  
                    getResources().getText  
                    (R.string.str_sms_sent_success).toString(),  
                    true  
                );  
                break;  
            case SmsManager.RESULT_ERROR_GENERIC_FAILURE:  
                /* 发送短信失败 */  
                //mTextView01.setText(R.string.str_sms_sent_failed);  
                mMakeTextToast  
                (  
                    getResources().getText  
                    (R.string.str_sms_sent_failed).toString(),  
                    true  
                );  
                break;  
            case SmsManager.RESULT_ERROR_RADIO_OFF:  
                break;  
            case SmsManager.RESULT_ERROR_NULL_PDU:  
                break;  
        }  
    }  
    catch(Exception e)  
    {  
        mTextView01.setText(e.toString());  
        e.printStackTrace();  
    }  
}
```



- ◆ 定义方法 mMakeTextToast(String str, boolean isLong)，用于输出发送成功还是失败的提示。具体代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)
{
    if(isLong==true)
    {
        Toast.makeText(jianting.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
        Toast.makeText(jianting.this, str, Toast.LENGTH_SHORT).show();
    }
}
```

- ◆ 定义 onResume()，此时 Activity 会被 Resume。具体代码如下所示。

```
@Override
protected void onResume()
{
    /* 自定义IntentFilter为SENT_SMS_ACTION Receiver */
    IntentFilter mFilter01;
    mFilter01 = new IntentFilter(SMS_SEND_ACTION);
    mReceiver01 = new mServiceReceiver();
    registerReceiver(mReceiver01, mFilter01);

    /* 自定义IntentFilter为DELIVERED_SMS_ACTION Receiver */
    mFilter01 = new IntentFilter(SMS_DELIVERED_ACTION);
    mReceiver02 = new mServiceReceiver();
    registerReceiver(mReceiver02, mFilter01);

    super.onResume();
}
```

- ◆ 定义 onPause()，此时 Activity 会被 Pause 暂停。具体代码如下所示。

```
@Override
protected void onPause()
{
    // TODO Auto-generated method stub

    /* 取消注册自定义Receiver */
    unregisterReceiver(mReceiver01);
```

```
unregisterReceiver(mReceiver02);  
  
super.onPause();  
}  
}
```

发送短信后能够显示短信是否成功的提示，如图 6-16 所示。



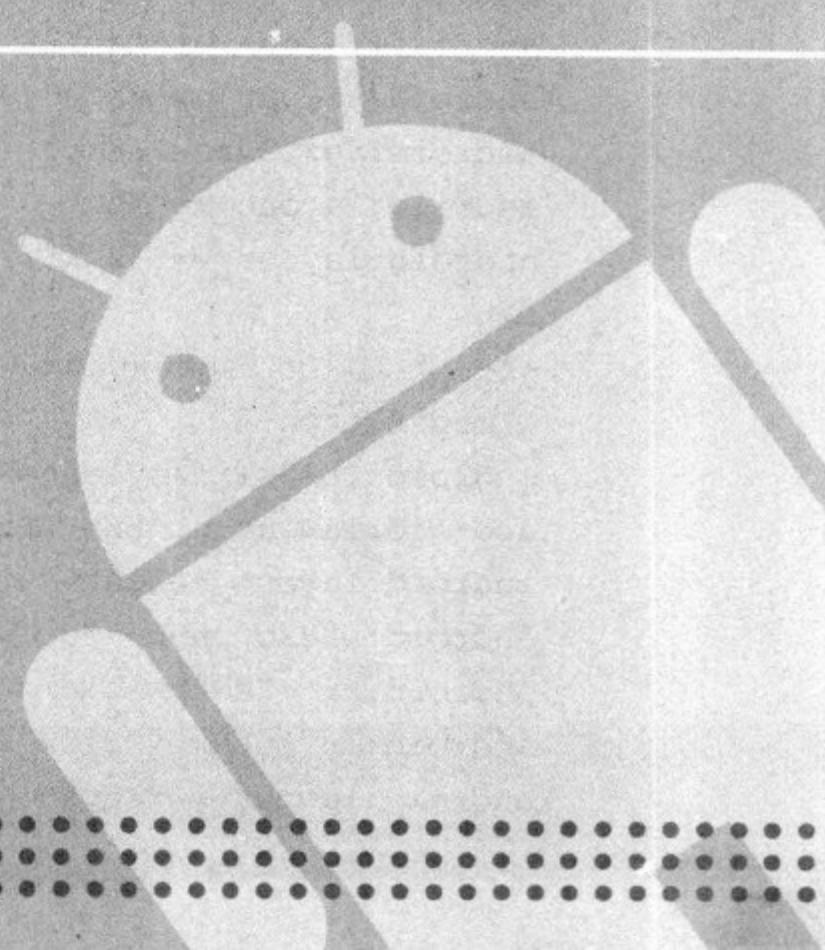
图6-16 短信提示





互联网实战演练

随着 3G 技术的普及，人们将手机当做移动电脑使用，经常利用手机浏览网页、聊天或玩偷菜游戏等。本章将通过几个典型实例的实现过程，详细介绍在 Android 系统中实现网络功能的基本知识。





7.1 浏览指定的网页

在 Android 系统中内置了一个 WebKit 引擎，使用里面的 WebView Widget 可以浏览一个网页。在本实例中，通过 WebView.loadUrl 来加载网址，所以从 EditText 中传入要浏览的网址后，就可以在 WebView 中加载网页的内容了。本实例的源码保存在【光盘\daima\第 7 章\wang】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:background="@drawable/white"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <!-- 建立一个TextView -->
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <!-- 建立一个EditText -->
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="267px"
        android:layout_height="40px"
        android:textSize="18sp"
        android:layout_x="5px"
        android:layout_y="32px"
        />
    <!-- 建立一个ImageButton -->
    <ImageButton
        android:id="@+id/myImageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/white"
        android:src="@drawable/go"
        android:layout_x="275px"
        android:layout_y="35px"
        />
```



```

    />
    <!-- 建立一个WebView -->
    <WebView
        android:id="@+id/myWebView1"
        android:layout_height="330px"
        android:layout_width="300px"
        android:layout_x="7px"
        android:layout_y="90px"
        android:background="@drawable/black"
        android:focusable="false"
    />
</AbsoluteLayout>

```

step 02 编写主程序文件 wang.java，其具体实现流程如下所示。

- ◆ 引入相关对象，然后分别定义对象 mImageButton1、mEditText1 和 mWebView1。具体代码如下所示。

```

package irdc.wang;

import irdc.wang.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.webkit.WebView;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

public class wang extends Activity
{
    private ImageButton mImageButton1;
    private EditText mEditText1;
    private WebView mWebView1;
}

```

step 03 通过 setOnClickListener 监听按钮单击事件，单击箭头后先抓取 EditText 中的数据，然后打开此网址，并在 WebView 中显示网页内容。具体代码如下所示。

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}

```



```
mImageButton1 = (ImageButton) findViewById(R.id.myImageButton1);
mEditText1 = (EditText) findViewById(R.id.myEditText1);
mWebView1 = (WebView) findViewById(R.id.myWebView1);

/*当单击箭头后*/
mImageButton1.setOnClickListener(new
    ImageButton.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub
        {

            mImageButton1.setImageResource(R.drawable.go_2);
            /*抓取EditText中的数据*/
            String strURI = (mEditText1.getText().toString());
            /* WebView显示网页内容 */
            mWebView1.loadUrl(strURI);
            Toast.makeText(
                wang.this,getString(R.string.load)+strURI,
                Toast.LENGTH_LONG)
                .show();
        }
    }
});
```

执行后显示一个文本框，在此可以输入网址，如图 7-1 所示。输入网址并单击后面的▶后，将显示此网页的内容，如图 7-2 所示。

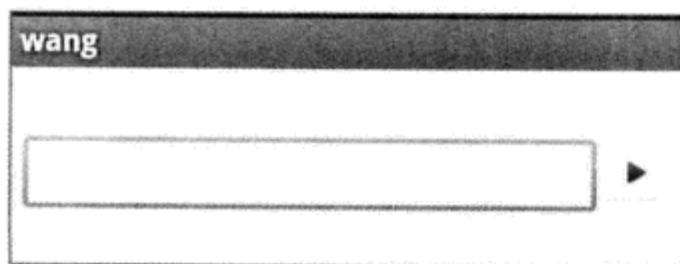


图7-1 输入网址



图7-2 打开的网页

7.2 加载显示HTML程序

HTML 语言是当前主流的网页技术，在 Android 手机中也可以显示 HTML 程序。使用 WebView 这个嵌入式的浏览器，可以将 HTML 标记传递给 WebView 对象，让 Android 手机程序变为 Web 浏览器。这样，网页程序被放在了 WebView 中运行，如同一个 Web Application。本实例的源码保存在【光盘\daima\第 7 章\ht】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:background="@drawable/white"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <!-- 创建一个TextView -->
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:text="@string/hello"
        />
    <!-- 创建一个WebView -->
    <WebView
        android:id="@+id/myWebView1"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        />
</LinearLayout>
```

step 02 编写主程序文件 ht.java，在 loadData 中插入了我们指定的 HTML 代码，通过 HTML 代码，显示了一幅图片和文字，并且插入了超级链接功能。具体代码如下所示。

```
package irdc.ht;

import irdc.ht.R;
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;
```



```
public class ht extends Activity
{
    private WebView mWebView1;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mWebView1 = (WebView) findViewById(R.id.myWebView1);

        /*自行设置WebView要显示的网页内容*/
        mWebView1.
            loadData(
                "<html><body><p>aaaaaaaa</p>" +
                "<div class='widget-content'> " +
                "<a href=http://www.sohu.com>" +
                " <img src=http://hiphotos.baidu.com/chaojihedan/pic/item/
bbddf5efc260f133fd3cd7.jpg />" +
                "<a href=http://www.sohu.com>Link Blog</a>" +
                "</body></html>", "text/html", "utf-8");
    }
}
```

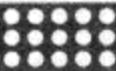
执行后将显示 HTML 产生的页面，如图 7-3 所示。单击超链接后会来到指定的网页，如图 7-4 所示。



图7-3 执行效果



图7-4 来到指定网页



7.3 使用浏览器打开网页

在Android系统中，我们可以直接调用它的内置浏览器来实现上网操作功能。在本实例中，定义了一个ListView，里面列表显示了4个菜单，单击菜单后会连接到指定的页面。当ListView的ItemClick()事件发生时，通过Intent(Intent.ACTION_VIEW,uri)来打开内置的浏览器，并浏览ListView中创建的网页URL。本实例的源码保存在【光盘\daima\第7章\nei】，具体实现流程如下所示。

step 01 编写布局文件 main.xml 的具体实现代码如下所示。

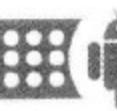
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"/>
    <ListView
        android:id="@+id/myListView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

step 02 编写主程序文件 nei.java，下面开始介绍其实现流程。

- ◆ 先声明一个 ListView 和 TextView 对象变量，然后声明一个 String array 变量来存储收藏夹列表，最后声明一个 String 变量来存储网址。具体代码如下所示。

```
package irdc.nei;

import irdc.nei.R;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
```



```
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

public class nei extends Activity
{
    /*声明一个ListView,TextView对象变量
     * 一个String array变量存储收藏夹列表
     * 与String变量来存储网址*/
    private ListView mListview1;
    private TextView mTextView1;
    private String[] myFavor;
    private String myUrl;
```

- 通过 findViewById 构造器创建 ListView 与 TextView 对象，将 string.xml 中的信息导入到列表中。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*通过findViewById构造器创建ListView与TextView对象*/
    mListview1 = (ListView) findViewById(R.id myListview1);
    mTextView1 = (TextView) findViewById(R.id myTextView1);
    mTextView1.setText(getResources().getString(R.string.hello));
    /*将列表通过string.xml中导入*/
    myFavor = new String[] {
        getResources().getString
        (R.string.str_list_url1),
        getResources().getString
        (R.string.str_list_url2),
        getResources().getString
        (R.string.str_list_url3),
        getResources().getString
        (R.string.str_list_url4)
    };
```

- 自定义一 ArrayAdapter 准备传入 ListView 中，然后将 myFavor 列表以参数传入。接下来自定义完成的 ArrayAdapter 传入自定义的 ListView 中，并将ListAdapter 的

可选 (Focusable) 菜单选项打开，最后设置 ListView 选项的 nItemClickListener。具体代码如下所示。

```
/*自定义一 ArrayAdapter准备传入ListView中，并将myFavor列表以参数传入*/
ArrayAdapter<String> adapter = new
ArrayAdapter<String>
(nei.this, android.R.layout.simple_list_item_1, myFavor);

/*将自定义完成的ArrayAdapter传入自定义的ListView中*/
mListView1.setAdapter(adapter);
/*将ListAdapter的可选(Focusable)菜单选项打开*/
mListView1.setItemsCanFocus(true);
/*设置ListView菜单选项设为每次只能单一选项*/
mListView1.setChoiceMode
(ListView.CHOICE_MODE_SINGLE);
/*设置ListView选项的nItemClickListener*/
mListView1.setOnItemClickListener
(new ListView.OnItemClickListener()
{

```

- ◆ 定义覆盖 onItemClick，当用户单击一个 Item 后，会进行比较，从 string.xml 中取出对应的 URL 网址，并将字符串转换为 URL 对象。具体代码如下所示。

```
@Override
/*覆盖OnItemClick方法*/
public void onItemClick
(AdapterView<?> arg0, View arg1, int arg2, long arg3)
{
    // TODO Auto-generated method stub
    /*若所选菜单的文字与myFavor字符串数组第一个文字相同*/
    if(arg0.getAdapter().getItem(arg2).toString() ==
myFavor[0].toString())
    {
        /*取得网址并调用goToUrl()方法*/
        myUrl=getResources().getString(R.string.str_url1);
        goToUrl(myUrl);
    }
    /*若所选菜单的文字与myFavor字符串数组第二个文字相同*/
    else if (arg0.getAdapter().getItem(arg2).toString() ==
myFavor[1].toString())
    {
        /*取得网址并调用goToUrl()方法*/
        myUrl=getResources().getString(R.string.str_url2);
```

```
        goToUrl(myUrl);
    }
    /*若所选菜单的文字与myFavor字符串数组第三个文字相同*/
    else if (arg0.getAdapter().getItem(arg2).toString() ==
    myFavor[2].toString())
    {
        /*取得网址并调用goToUrl()方法*/
        myUrl=getResources().getString(R.string.str_url3);
        goToUrl(myUrl);
    }
    /*若所选菜单的文字与myFavor字符串数组第四个文字相同*/
    else if (arg0.getAdapter().getItem(arg2).toString() ==
    myFavor[3].toString())
    {
        /*取得网址并调用goToUrl()方法*/
        myUrl=getResources().getString(R.string.str_url4);
        goToUrl(myUrl);
    }
    /*以上皆非*/
    else
    {
        /*显示错误信息*/
        mTextView1.setText("Ooops!!出错了");
    }
}
});
```

- ◆ 定义方法 goToUrl(String url) 来打开网址为 URL 的网页，具体代码如下所示。

```
/*打开网页的方法*/
private void goToUrl(String url)
{
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
```

执行后将列表显示 4 个菜单，如图 7-5 所示。当单击一个菜单后，会来到对应的目标页面，如图 7-6 所示。



图7-5 4个菜单



图7-6 打开的网页

7.4 显示网络中的照片

在本实例中，我们直接从网络中调用照片，并在 Gallery 中显示出来，这样可以节约手机的存储空间。在具体实现时，需要将 URL 网址的相片实时处理下载后，以 InputStream 转换为 Bitmap，这样才能放入 BaseAdapter 中取用。在运行实例前，需要预先准备照片并上传到网络空间中，在获取相片的连接后，再以 String 数组方式放在程序中，并对 BaseAdapter 稍作修改，加上 URL 对象的访问以及 URLConnection 连接的处理。本实例的源码保存在【光盘\daima\第7章\zhao】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLinearLayout"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Gallery
        android:id="@+id/myGallery01"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
```

```
</Gallery>
</LinearLayout>
```

step 02 编写主程序文件 zhao.java，其具体实现流程如下所示。

- ◆ 声明 Gallery 中要显示 5 副图片的地址栏字符串，具体代码如下所示。

```
package irdc.zhao;

import irdc.zhao.R;

import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.netURLConnection;

import android.app.Activity;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.Gallery;
import android.widget.ImageView;

public class zhao extends Activity
{
    private Gallery myGallery01;
    /* 地址栏字符串 */
    private String[] myImageURL = new String[]
    {
        "http://b27.photo.store.qq.com/http_imgload.cgi?/"
        +
        "rurl4_b=086a67cbd6a8cfb4389ea2b48efab6f322f755a085107a7aeeaa56fc1358b1bd
124186254e021f0655732688e69f060725491f8ae82e8e5508dbe9821670e2baf04e92ded
c97e3bbf28e5605596aa991c13220f1&a=27&b=27",
        "http://b27.photo.store.qq.com/http_imgload.cgi?/"
        +
        "rurl4_b=086a67cbd6a8cfb4389ea2b48efab6f3ea78f5797abbbaa617259f2d2a980a54
68f2801897cfcc2b78af92fbb87565ed7a3a08041daff2dd9ccd26d3cc6198e41f2d205c8
a0c445325771e8a179215999afaf9f3&a=27&b=27",
        "http://b27.photo.store.qq.com/http_imgload.cgi?/"
```

```

+
"url4_b=2a9dcf1fd909a7ed3ce8951f738608982f26d812b3a5fc96e221b85fc085e7cc
3264ee20730f0fd3a1f7aca06740db7a6153d9357467ca39f82b866b6fbe3cd94bbdd10ed
01841e67c95d8e4af8890b7ced40869&a=30&b=27",
    "http://b27.photo.store.qq.com/http_imgload.cgi?/"
+
"url4_b=2a9dcf1fd909a7ed3ce8951f73860898bb7ff57a8cb7747c9f0eb6a02124850b
709c0b86f086a4ba5653eeb71dd4b01e4a58f407e2eec9433cd8d4bc0b88fda56260c2c8b
eb34ebab77b610c7131393f82e774ef&a=27&b=27",
    "http://b27.photo.store.qq.com/http_imgload.cgi?/"
+
"url4_b=2a9dcf1fd909a7ed3ce8951f73860898158d252489f84e7d2a83d44c01b7bb12
b2c19ca0efdd555dba788407fd01e9de45524b11a9793f532624197bc8d14c84ae78ddeba
fe4357e4eedc60e9e510224367490bf&a=27&b=27" };

```

- ◆ 引入布局文件 main.xml，定义类成员 myContext Context 对象，然后设置只有一个参数 C 的构造器，即要存储的 Context。先获取 Gallery 属性的 Index id，并设置对象的 styleable 属性能够反复使用。具体代码如下所示。

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    myGallery01 = (Gallery) findViewById(R.id.myGallery01);
    myGallery01.setAdapter(new myInternetGalleryAdapter(this));
}

/* 用BaseAdapter */
public class myInternetGalleryAdapter extends BaseAdapter
{
    /* 类成员myContext Context对象 */
    private Context myContext;
    private int mGalleryItemBackground;

    /*构造器只有一个参数，即要存储的Context */
    public myInternetGalleryAdapter(Context c)
    {
        this.myContext = c;
        TypedArray a = myContext
            .obtainStyledAttributes(R.styleable.Gallery);
    }
}

```

```
/* 获取Gallery属性的Index id */  
mGalleryItemBackground = a.getResourceId(  
    R.styleable.Gallery_android_galleryItemBackground, 0);
```

```
/* 把对象的styleable属性能够反复使用 */  
a.recycle();
```

```
}
```

- ◆ 使用方法 getCount() 来返回全部已定义图片的总量。然后定义 getItem(int position)，用于使用 getItem 方法获取当前容器中图像数的数组 ID。具体代码如下所示。

```
/* 返回全部已定义图片的总量 */
```

```
public int getCount()  
{  
    return myImageURL.length;  
}
```

```
/* 使用getItem方法获取当前容器中图像数的数组ID */
```

```
public Object getItem(int position)  
{  
    return position;  
}
```

```
public long getItemId(int position)  
{  
    return position;  
}
```

- ◆ 定义方法 getScale(boolean focused, int offset)，能够根据中央位移量，利用 getScale 返回 views 的大小 (0.0f to 1.0f)。具体代码如下所示。

```
/* 根据中央位移量，利用getScale返回views的大小(0.0f to 1.0f) */
```

```
public float getScale(boolean focused, int offset)  
{  
    /* Formula: 1 / (2 ^ offset) */  
    return Math.max(0, 1.0f / (float) Math.pow(2, Math  
        .abs(offset)));  
}
```

定义方法 getView 根据中央位移量来获取当前要显示的图像 View，传入数组 ID 值使之读取并成像处理。具体代码如下所示。

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

```

/* 创建ImageView对象 */
ImageView imageView = new ImageView(this.myContext);
try
{
    /* new URL将对象网址传入 */
    URL aryURI = new URL(myImageURL[position]);
    /* 获取连接 */
    URLConnection conn = aryURI.openConnection();
    conn.connect();
    /*获取返回的InputStream*/
    InputStream is = conn.getInputStream();
    /* 将InputStream变为Bitmap */
    Bitmap bm = BitmapFactory.decodeStream(is);
    /* 关闭InputStream */
    is.close();
    /* 设置Bitmap到ImageView中 */
    imageView.setImageBitmap(bm);
} catch (IOException e)
{
    e.printStackTrace();
}
imageView.setScaleType(ImageView.ScaleType.FIT_XY);
/* 设置ImageView的宽和高, 单位是dip */
imageView.setLayoutParams(new Gallery.LayoutParams(200, 150));
/* 设置Gallery背景图*/
imageView.setBackgroundResource(mGalleryItemBackground);
return imageView;
}
}
}

```

执行后将在 Gallery 中显示网络中的图片, 如图 7-7 所示。



图7-7 执行效果



7.5 播放在线音乐

为了节约手机的存储空间，我们可以从网络中下载一个 MP3 然后再播放。在本实例中我们插入 4 个按钮，分别用于播放、暂停、重新播放和停止处理。执行后，通过 Runnable 发起运行线程，在线程中远程下载指定的 MP3 文件，是通过网络传输方式下载的。下载完毕后，临时保存到 SD 卡中，这样可以通过 4 个按钮对其进行控制。当程序关闭后，删除 SD 卡中的临时性文件。

本实例的源码保存在【光盘\daima\第 7 章\yinyue】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:text="@string/hello"
    />
    <LinearLayout
        android:orientation="horizontal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:padding="10dip"
    >
        <ImageButton android:id="@+id/play"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/play"
        />
        <ImageButton android:id="@+id/pause"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/pause"
        />
    
```

```
>
<ImageButton android:id="@+id/reset"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/reset"
/>
<ImageButton android:id="@+id/stop"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/stop"
/>
</LinearLayout>
</LinearLayout>
```

Step 02 编写主程序文件 yinyue.java，其实现流程如下所示。

- ◆ 引入相关 class，然后 private 声明系统中需要的对象，具体代码如下所示。

```
package irdc.yinyue;

import irdc.yinyue.R;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

import android.app.Activity;
import android.graphics.PixelFormat;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.webkit.URLUtil;
import android.widget.ImageButton;
import android.widget.TextView;

public class yinyue extends Activity
{
    private TextView mTextView01;
    private MediaPlayer mMediaPlayer01;
    private ImageButton mPlay, mReset, mPause, mStop;
    private boolean bIsReleased = false;
```



```
private boolean bIsPaused = false;  
private static final String TAG = "Hippo_URL_MP3_Player";
```

- ◆ 用 currentFilePath 用于记录当前正在播放 MP3 的地址 URL，定义 currentTempFilePath 表示当前播放 MP3 的路径。具体代码如下所示。

```
/* 记录当前正在播放MP3的地址URL */  
private String currentFilePath = "";  
  
/*当前播放MP3的路径*/  
private String currentTempFilePath = "";  
private String strVideoURL = "";
```

- ◆ 引入主布局文件 main.xml，然后通过 strVideoURL 设置要播放 MP3 文件的网址，并设置透明度。具体代码如下所示。

```
/** Called when the activity is first created. */  
@Override  
public void onCreate(Bundle savedInstanceState)  
{  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
  
    /* mp3文件不会被下载到local*/  
    strVideoURL = "http://www.lrn.cn/zywh/xyyy/yyxs/200805/  
W020080505536315331317.mp3";
```

```
mTextView01 = (TextView) findViewById(R.id.myTextView1);
```

```
/*设置透明度*/  
getWindow().setFormat(PixelFormat.TRANSPARENT);
```

```
mPlay = (ImageButton) findViewById(R.id.play);  
mReset = (ImageButton) findViewById(R.id.reset);  
mPause = (ImageButton) findViewById(R.id.pause);  
mStop = (ImageButton) findViewById(R.id.stop);
```

- ◆ 设置单击【播放】按钮所触发的处理事件，具体代码如下所示。

```
/* 播放按钮 */  
mPlay.setOnClickListener(new ImageButton.OnClickListener()  
{  
    public void onClick(View view)  
    {
```

```

    /* 调用播放影片Function */
    playVideo(strVideoURL);
    mTextView01.setText
    (
        getResources().getText(R.string.str_play).toString()+
        "\n"+ strVideoURL
    );
}
});

```

- ◆ 设置单击【重新播放】按钮所触发的处理事件，具体代码如下所示。

```

/* 重新播放 */
mReset.setOnClickListener(new ImageButton.OnClickListener()
{
    public void onClick(View view)
    {
        if(bIsReleased == false)
        {
            if (mMediaPlayer01 != null)
            {
                mMediaPlayer01.seekTo(0);
                mTextView01.setText(R.string.str_play);
            }
        }
    }
});

```

- ◆ 设置单击【暂停播放】按钮所触发的处理事件，具体代码如下所示。

```

/* 暂停播放 */
mPause.setOnClickListener(new ImageButton.OnClickListener()
{
    public void onClick(View view)
    {
        if (mMediaPlayer01 != null)
        {
            if(bIsReleased == false)
            {
                if(bIsPaused==false)
                {
                    mMediaPlayer01.pause();
                    bIsPaused = true;
                    mTextView01.setText(R.string.str_pause);
                }
            }
        }
    }
});

```

```
        else if(bIsPaused==true)
        {
            mMediaPlayer01.start();
            bIsPaused = false;
            mTextView01.setText(R.string.str_play);
        }
    }
}
});
```

- ◆ 设置单击【停止播放】按钮所触发的处理事件，具体代码如下所示。

```
/* 终止 */
mStop.setOnClickListener(new ImageButton.OnClickListener()
{
    public void onClick(View view)
    {
        try
        {
            if (mMediaPlayer01 != null)
            {
                if(bIsReleased==false)
                {
                    mMediaPlayer01.seekTo(0);
                    mMediaPlayer01.pause();
                    mTextView01.setText(R.string.str_stop);
                }
            }
            catch(Exception e)
            {
                mTextView01.setText(e.toString());
                Log.e(TAG, e.toString());
                e.printStackTrace();
            }
        }
    }
});
```

- ◆ 定义方法 playVideo(final String strPath)，用于播放指定的 MP3，其播放的是存储卡中暂时保存的 MP3 文件，具体代码如下所示。

```
private void playVideo(final String strPath)
{
```

```

try
{
    if (strPath.equals(currentFilePath) && mMediaPlayer01 != null)
    {
        mMediaPlayer01.start();
        return;
    }

    currentFilePath = strPath;

    mMediaPlayer01 = new MediaPlayer();
    mMediaPlayer01.setAudioStreamType(2);
}

```

- ◆ 定义 setOnErrorListener 处理事件来处理错误，具体代码如下所示。

```

/* 错误事件 */
mMediaPlayer01.setOnErrorListener(new MediaPlayer.
OnErrorListener()
{
    @Override
    public boolean onError(MediaPlayer mp, int what, int extra)
    {
        Log.i(TAG, "Error on Listener, what: " + what + "extra: " + extra);
        return false;
    }
});

```

- ◆ 定义 setOnBufferingUpdateListener 来捕捉使用 MediaPlayer 缓冲区的更新，具体代码如下所示。

```

/* 捕捉使用MediaPlayer缓冲区的更新事件 */
mMediaPlayer01.setOnBufferingUpdateListener(new MediaPlayer.
OnBufferingUpdateListener()
{
    @Override
    public void onBufferingUpdate(MediaPlayer mp, int percent)
    {
        Log.i(TAG, "Update buffer: " + Integer.toString(percent) + "%");
    }
});

```

- ◆ 定义 setOnCompletionListener，用于实现播放完毕所触发的事件。具体代码如下所示。

```

/* 播放完毕所触发的事件 */
mMediaPlayer01.setOnCompletionListener(new MediaPlayer.

```

```
OnCompletionListener()
{
    @Override
    public void onCompletion(MediaPlayer mp)
    {
        Log.i(TAG, "mMediaPlayer01 Listener Completed");
    }
});
```

- ◆ 定义 setOnPreparedListener，用于开始阶段的监听 Listener。具体代码如下所示。

```
/* 开始阶段的监听Listener */
mMediaPlayer01.setOnPreparedListener(new MediaPlayer.
OnPreparedListener()
{
    @Override
    public void onPrepared(MediaPlayer mp)
    {
        Log.i(TAG, "Prepared Listener");
    }
});
```

- ◆ 定义 Runnable 对象 r，用 Runnable 来确保文件在存储完毕后才开始 start()。先将文件存到 SD 卡，然后在 setDataSource 后运行 prepare()，最后通过 mMediaPlayer01.start() 开始播放 MP3。如果有异常则输出提示。具体代码如下所示。

```
/* 用Runnable来确保文件在存储完毕后才开始start() */
Runnable r = new Runnable()
{
    public void run()
    {
        try
        {
            /* setDataSource将文件存到SD卡 */
            setDataSource(strPath);
            /* 因为线程顺利进行，所以在setDataSource后运行prepare() */
            mMediaPlayer01.prepare();
            Log.i(TAG, "Duration: " + mMediaPlayer01.getDuration());

            /* 开始播放mp3 */
            mMediaPlayer01.start();
            bIsReleased = false;
        }
        catch (Exception e)
```

```

    {
        Log.e(TAG, e.getMessage(), e);
    }
}

new Thread(r).start();
}
catch (Exception e)
{
    if (mMediaPlayer01 != null)
    {
        /* 线程发生异常则停止播放 */
        mMediaPlayer01.stop();
        mMediaPlayer01.release();
    }
    e.printStackTrace();
}
}
}

```

- ◆ 定义函数 setDataSource，用于存储 URL 的 MP3 文件到存储卡。首先判断传入的地址是否为 URL，然后创建 URL 对象和临时文件，当 fos 存储完毕调用 MediaPlayer.setDataSource。具体代码如下所示。

```

/* 定义函数用于存储URL的mp3文件到存储卡 */
private void setDataSource(String strPath) throws Exception
{
    /* 判断传入的地址是否为URL */
    if (!URLUtil.isNetworkUrl(strPath))
    {
        mMediaPlayer01.setDataSource(strPath);
    }
    else
    {
        if (bIsReleased == false)
        {
            /* 创建URL对象 */
            URL myURL = new URL(strPath);
            URLConnection conn = myURL.openConnection();
            conn.connect();

            /* 获取URLConnection的InputStream */
            InputStream is = conn.getInputStream();
            if (is == null)
            {

```



```
        throw new RuntimeException("stream is null");
    }

    /* 创建临时文件 */
    File myTempFile = File.createTempFile("yinyue",
".."+getFileExtension(strPath));
    currentTempFilePath = myTempFile.getAbsolutePath();

    /* currentTempFilePath = /sdcard/hippoplayertmp39327.mp3 */

    /*
    if(currentTempFilePath!="")
    {
        Log.i(TAG, currentTempFilePath);
    }
    */

    FileOutputStream fos = new FileOutputStream(myTempFile);
    byte buf[] = new byte[128];
    do
    {
        int numread = is.read(buf);
        if (numread <= 0)
        {
            break;
        }
        fos.write(buf, 0, numread);
    }while (true);

    /* 直到fos存储完毕，调用MediaPlayer.setDataSource */
    mMediaPlayer01.setDataSource(currentTempFilePath);
    try
    {
        is.close();
    }
    catch (Exception ex)
    {
        Log.e(TAG, "error: " + ex.getMessage(), ex);
    }
}

}
```

- 定义方法 `getFileExtension(String strFileName)`, 用于获取音乐文件扩展名, 如果无

- android与iphone及ipad开发书籍 持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等 持续不断更新中.....
- delphi《书籍》及《视频》教程 持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修炼班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程（个人博客系统49课全），外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班，php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库 ←

文件名 <ul style="list-style-type: none"> <input type="checkbox"/> 办公类软件使用指南 <input type="checkbox"/> 医学 <input type="checkbox"/> 历史人物传记 <input type="checkbox"/> 哲学宗教 <input type="checkbox"/> 外语资料（除英语外）（除英语外） <input type="checkbox"/> 官场类小说 <input type="checkbox"/> 建筑工程类 <input type="checkbox"/> 情感生活类小说 <input type="checkbox"/> 政治军事 <input type="checkbox"/> 教育学习科普大全 网址：WLSAM168.400GB.COM <input type="checkbox"/> 文学理论 <input type="checkbox"/> 智力开发、增强记忆、快速阅读技巧大全 <input type="checkbox"/> 社会生活 <input type="checkbox"/> 科学技术 <input type="checkbox"/> 程序编程类 <input type="checkbox"/> 经济管理 <input type="checkbox"/> 网络安全及管理 <input type="checkbox"/> 网赚系列 <input type="checkbox"/> 美食小吃烹饪煲汤大全 <input type="checkbox"/> 课外读物 	P D F 电子书专题库，内容详尽，每天不断更新！！ <p>本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习，如用于商业或非法用途的后果自负！</p>
--	--

OE Foxit PDF Editor ±à¼-ºæ·É·ÉùÓÐ (c) by Foxit Software Company, 2004 VIP培训教程，易语言黑月VIP视频教程，网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)

- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等 持续不断更新中....
- 网建设相关的pdf书籍及各种vip视频教程 持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价营销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等 持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。
- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 >

法顺利获取扩展名则默认为.dat。具体代码如下所示。

```
/* 获取音乐文件扩展名自定义函数 */
private String getFileExtension(String strFileName)
{
    File myFile = new File(strFileName);
    String strFileExtension=myFile.getName();
    strFileExtension=(strFileExtension.substring(strFileExtension.lastIndexOf("."))+1)).toLowerCase();
    if(strFileExtension=="")
    {
        /* 如果无法顺利获取扩展名则默认为.dat */
        strFileExtension = "dat";
    }
    return strFileExtension;
}
```

- ◆ 定义方法 delFile(String strFileName)，用于当离开程序时删除临时音乐文件。具体代码如下所示。

```
/* 离开程序时需要调用自定义函数删除临时音乐文件*/
private void delFile(String strFileName)
{
    File myFile = new File(strFileName);
    if(myFile.exists())
    {
        myFile.delete();
    }
}

@Override
protected void onPause()
{
    //TODO Auto-generated method stub

    /* 删除临时文件 */
    try
    {
        delFile(currentTempFilePath);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

```
    super.onPause();  
}  
}
```

执行后可以通过播放、暂停、重新播放和停止四个按钮，控制指定的 MP3 音乐，如图 7-8 所示。

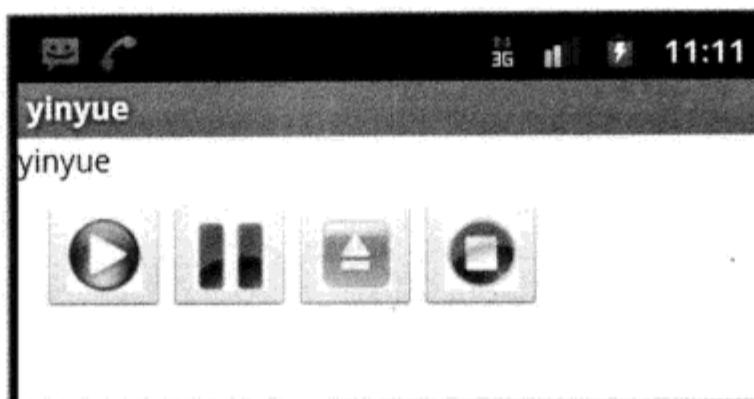


图7-8 执行效果

7.6 下载在线手机铃声

铃声是移动手机的重要功能之一，我们可以从网络中直接下载一个 MP3 文件作为手机的铃声。在本实例中，通过 EditText 让用户输入一个指定的网址，当下载完成后打开 RingtoneManager.ACTION_RINGTONE_PICKER 这个 Intent，在打开 Intent 的同时传入一个参数，这个 ACTION_RINGTONE_PICKER 的 Intent 会带入刚才下载文件让用户选择。在实现过程中，会判断下载文件是否完整，并判断用户已设置铃声，下载后的铃声文件存储在哪里。在具体实现时，会以 SD 卡中的铃声文件作为存储网络下载音乐文件的路径，打开 RingtoneManager 的 ACTION_RINGTONE_PICKER 的 Intent 让用户找到下载的音乐，并作为铃声。

本实例的源码保存在【光盘\daima\第 7 章\lingsheng】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    >  
    <!-- 建立一个TextView -->  
    <TextView  
        android:id="@+id/myTextView1"
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/app_name"
/>
<!-- 建立一个EditText -->
<EditText
    android:id="@+id/myEditText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/url"
/>
<!-- 建立一个Button -->
<Button
    android:id="@+id/myButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button1"
/>
</LinearLayout>
```

step 02 编写主程序文件 lingsheng.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类，然后 private 私有声明系统中需要的对象，具体代码如下所示。

```
package irdc.lingsheng;

import irdc.lingsheng.R;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

import android.app.Activity;
import android.content.Intent;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.webkit.URLUtil;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
```



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

public class lingsheng extends Activity
{
    protected static final String APP_TAG = "DOWNLOAD_RINGTONE";
    private Button mButton1;
    private TextView mTextView1;
    private EditText mEditText1;
    private String strURL = "";
    public static final int RINGTONE_PICKED = 0x108;
    private String currentFilePath = "";
    private String currentTempFilePath = "";
    private String fileEx="";
    private String fileNa="";
    private String strRingtoneFolder = "/sdcard/music/ling";
```

- ◆ 判断是否有“/sdcard/music/ringtones”文件夹，如果不存在则输出提示。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mButton1 = (Button) findViewById(R.id.myButton1);
    mTextView1 = (TextView) findViewById(R.id.myTextView1);
    mEditText1 = (EditText) findViewById(R.id.myEditText1);

    /*判断是否有/sdcard/music/ringtones文件夹*/
    if(bIfExistRingtoneFolder(strRingtoneFolder))
    {
        Log.i(APP_TAG, "Ringtone Folder exists.");
    }
```

- ◆ 设置 strURL 是 String 中设置的，通过 fileEx 和 getFile 取得文件的名称。具体代码如下所示。

```
mButton1.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
```

```

// TODO Auto-generated method stub
strURL = mEditText1.getText().toString();

Toast.makeText(lingsheng.this, getString(R.string.str_msg),
    Toast.LENGTH_SHORT).show();

/*取得文件名称*/
fileEx = strURL.substring(strURL.lastIndexOf(".") + 1, strURL.
    length()).toLowerCase();
fileNa = strURL.substring(strURL.lastIndexOf("/") + 1, strURL.
    lastIndexOf("."));

getFile(strURL);
}
);
}
}

```

- ◆ 定义方法 getMIMETYPE(File f)，用于判断文件 MimeType 的 method。首先取得扩展名，然后根据扩展名的类型决定MimeType。具体代码如下所示。

```

/* 判断文件MimeType的method */
private String getMIMETYPE(File f)
{
    String type="";
    String fName=f.getName();
    /* 取得扩展名 */
    String end=fName.substring(fName.lastIndexOf(".")+1,
        fName.length()).toLowerCase();

    /* 依扩展名的类型决定MimeType */
    if(end.equals("m4a") || end.equals("mp3") || end.equals("mid") ||
       end.equals("xmf") || end.equals("ogg") || end.equals("wav"))
    {
        type = "audio";
    }
    else if(end.equals("3gp") || end.equals("mp4"))
    {
        type = "video";
    }
    else if(end.equals("jpg") || end.equals("gif") ||
            end.equals("png") || end.equals("jpeg") ||
            end.equals("bmp"))
    {
        type = "image";
    }
}

```

```
    else
    {
        type="*";
    }
    /*如果无法直接打开，就跳出软件列表给用户选择 */
    if(end.equals("image"))
    {
    }
    else
    {
        type += "/*";
    }
    return type;
}
```

- ◆ 定义方法 getFile(final String strPath) 来获取最后的文件，如果地址和当前地址一样，则直接用 getDataSource 数据。如果有异常，则输出异常信息。具体代码如下所示。

```
private void getFile(final String strPath)
{
    try
    {
        if (strPath.equals(currentFilePath) )
        {
            getDataSource(strPath);
        }
        currentFilePath = strPath;
        Runnable r = new Runnable()
        {
            public void run()
            {
                try
                {
                    getDataSource(strPath);
                }
                catch (Exception e)
                {
                    Log.e(APP_TAG, e.getMessage(), e);
                }
            }
        };
        new Thread(r).start();
    }
```

```
catch (Exception e)
{
    e.printStackTrace();
}
```

- ◆ 定义方法 `getDataSource(String strPath)` 来获取远程文件，如果地址错误则输出错误信息。具体代码如下所示。

```
/*取得远程文件*/
private void getDataSource(String strPath) throws Exception
{
    if (!URLUtil.isNetworkUrl(strPath))
    {
        mTextView1.setText("错误的URL");
    }
    else
    {
        /*取得URL*/
        URL myURL = new URL(strPath);
        /*创建连接*/
        URLConnection conn = myURL.openConnection();
        conn.connect();
        /*InputStream 下载文件*/
        InputStream is = conn.getInputStream();
        if (is == null)
        {
            throw new RuntimeException("stream is null");
        }

        /*创建文件地址*/
        File myTempFile = new File("/sdcard/music/ling/",
                                   fileNa+"."+fileEx);
        /*取得在内存盘案路径*/
        currentTempFilePath = myTempFile.getAbsolutePath();
        /*将文件写入暂存盘*/
        FileOutputStream fos = new FileOutputStream(myTempFile);
        byte buf[] = new byte[128];
        do
        {
            int numread = is.read(buf);
            if (numread <= 0)
            {
                break;
            }
            fos.write(buf, 0, numread);
        }
        while (true);
        fos.close();
    }
}
```



```
    }
    fos.write(buf, 0, numread);
}while (true);
```

- ◆ 打开 RingtonManager 来选择铃声，通过 Intent 对象 intent 来设置铃声，然后设置显示铃声的文件夹和显示铃声开头。如果有异常则输出异常。具体代码如下所示。

```
/* 打开RingtonManager进行铃声选择 */
String uri = null;
if(bIfExistRingtoneFolder(strRingtoneFolder))
{
    /*设置铃声*/
    Intent intent = new Intent( RingtonManager.
        ACTION_RINGTONE_PICKER);
    /*设置显示铃声的文件夹*/
    intent.putExtra( RingtonManager.EXTRA_RINGTONE_TYPE,
        RingtonManager.TYPE_RINGTONE);
    /*设置显示铃声开头*/
    intent.putExtra( RingtonManager.EXTRA_RINGTONE_TITLE,
        "设置铃声");
    if( uri != null)
    {
        intent.putExtra( RingtonManager.
            EXTRA_RINGTONE_EXISTING_URI, Uri.parse( uri));
    }
    else
    {
        intent.putExtra( RingtonManager.
            EXTRA_RINGTONE_EXISTING_URI, (Uri)null);
    }
    startActivityForResult(intent, RINGTONE_PICKED);
}

try
{
    is.close();
}
catch (Exception ex)
{
    Log.e(APP_TAG, "error: " + ex.getMessage(), ex);
}
```

- ◆ 定义方法 `onActivityResult()` 根据用户选择的铃声设置保存对应的信息。当选择完毕后，会再次返回选择 Activity。具体代码如下所示。

```

@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data)
{
    // TODO Auto-generated method stub
    if (resultCode != RESULT_OK)
    {
        return;
    }
    switch (requestCode)
    {
        case (RINGTONE_PICKED):
            try
            {
                Uri pickedUri = data.getParcelableExtra
                    (RingtoneManager.EXTRA_RINGTONE_PICKED_URI);
                if(pickedUri!=null)
                {
                    RingtoneManager.setActualDefaultRingtoneUri
                        (lingsheng.this,RingtoneManager.TYPE_RINGTONE,
                         pickedUri);
                }
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            break;
        default:
            break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}

```

- ◆ 定义方法 `bIfExistRingtoneFolder` 来判断是否包含了“/sdcard/music/ringtones”文件夹，具体代码如下所示。

```

/*判断是否/sdcard/music/ringtones文件夹*/
private boolean bIfExistRingtoneFolder(String strFolder)
{
    boolean bReturn = false;

```

```
File f = new File(strFolder);
if(!f.exists())
{
    /*创建/sdcard/music/ringtones文件夹*/
    if(f.mkdirs())
    {
        bReturn = true;
    }
    else
    {
        bReturn = false;
    }
}
else
{
    bReturn = true;
}
return bReturn;
}
```

执行后会先显示一个下载界面，如图 7-9 所示。单击【下载音乐】按钮后开始下载指定的 MP3 文件，下载完成后会弹出“铃声设置”界面，如图 7-10 所示。选择一种选项，并单击【OK】按钮后，完成铃声设置。

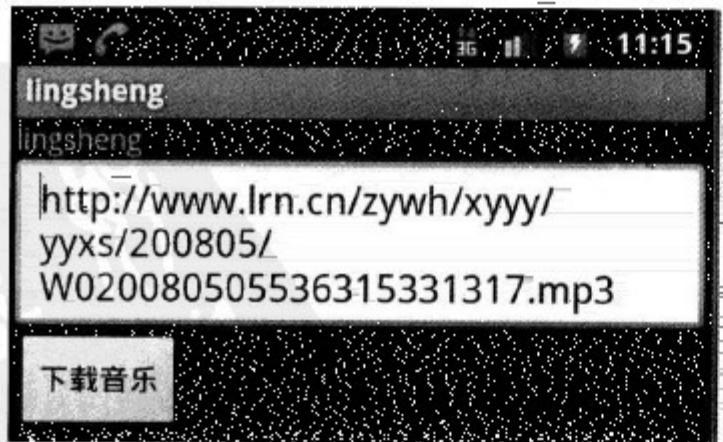


图7-9 初始界面



图7-10 铃声设置界面

7.7 开发一个简易RSS系统

RSS 是一个开放的新闻模式，是在线共享内容的一种简易方式（Really Simple Syndication，聚合内容）。通常在时效性比较强的内容上使用 RSS 订阅能更快速获取信息，网站提供 RSS 输出，有利于让用户获取网站内容的最新更新。通过本实例的实现过程，读者将会掌握在 Android 系统中开发 RSS 的基本流程。本实例的源码保存在【光盘\ daima\ 第 7 章\RSSL】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/layout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <EditText
        android:id="@+id/myEdit"
        android:layout_width="280px"
        android:layout_height="wrap_content"
        android:text="http://"
        android:textSize="12sp"
        android:layout_x="20px"
        android:layout_y="42px"
    >
    </EditText>
    <TextView
        android:id="@+id/myText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_title"
        android:textColor="@drawable/black"
        android:textSize="16sp"
        android:layout_x="20px"
        android:layout_y="12px"
    >
    </TextView>
    <Button
        android:id="@+id/myButton"
    >
```



```
    android:layout_width="86px"
    android:layout_height="46px"
    android:text="@string/str_button"
    android:textColor="@drawable/black"
    android:layout_x="100px"
    android:layout_y="112px"
  >
</Button>
</AbsoluteLayout>
```

step 02 分别编写主程序文件RSSL.java、RSSL_1.java、RSSL_2.java、News.java、MyHandler.java和MyAdapter.java，其中，在文件RSSL.java中以EditText来作为输入RSS连接组件，当输入网址并单击【开始解析】按钮后，按钮的onClick会被触发，运行EditText的空白检查。当检查无误后，将输入的网址写入Bundle对象中，再将Bundle对象assign给Intent，并通过startActivityForResult()来触发RSSL_1这个Activity。

先编写文件 RSSL.java，具体实现代码如下所示。

```
package irdc.RSSL;

/* import相关class */
import irdc.RSSL.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class RSSL extends Activity
{
    /* 变量声明 */
    private Button mButton;
    private EditText mEditText;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 初始化对象 */
        mEditText=(EditText) findViewById(R.id.myEdit);
```

```
mButton=(Button) findViewById(R.id.myButton);
/* 设置Button的onClick事件 */
mButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        String path=mEditText.getText().toString();
        if(path.equals(""))
        {
            showDialog("网址不可为空白!");
        }
        else
        {
            /* new一个Intent对象，并指定class */
            Intent intent = new Intent();
            intent.setClass(RSSL.this,RSSL_1.class);

            /* new一个Bundle对象，并将要传递的数据传入 */
            Bundle bundle = new Bundle();
            bundle.putString("path",path);
            /* 将Bundle对象assign给Intent */
            intent.putExtras(bundle);
            /* 调用Activity EX08_13_1 */
            startActivityForResult(intent,0);
        }
    }
});
});
```

```
/* 覆盖 onActivityResult() */
@Override
protected void onActivityResult(int requestCode,int resultCode,
                               Intent data)
{
    switch (resultCode)
    {
        case 99:
            /* 返回错误时以Dialog显示 */
            Bundle bunde = data.getExtras();
            String error = bunde.getString("error");
            showDialog(error);
            break;
    }
}
```



```
    default:  
        break;  
    }  
}  
  
/* 显示Dialog的方法 */  
private void showDialog(String mess){  
    new AlertDialog.Builder(RSSL.this).setTitle("Message")  
    .setMessage(mess)  
    .setNegativeButton("确定", new DialogInterface.OnClickListener()  
{  
        public void onClick(DialogInterface dialog, int which)  
        {  
        }  
    })  
    .show();  
}  
}
```

step 03 然后编写文件 RSSL_1.java，此文件是一个 ListActivity，是通过主程序 RSSL.java 来调用的，用于显示订阅的 RSS 内容列表。其实现流程如下所示。

- ◆ 引入相关 class，分别变量声明 TextView mText、title 和 li。具体代码如下所示。

```
package irdc.RSSL;  
  
/* import相关class */  
import irdc.RSSL.R;  
  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.List;  
import android.app.ListActivity;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.ListView;  
import android.widget.TextView;  
import javax.xml.parsers.*;  
import org.xml.sax.InputSource;  
import org.xml.sax.XMLReader;  
  
public class RSSL_1 extends ListActivity  
{  
    /* 变量声明 */
```

```

private TextView mText;
private String title="";
private List<News> li=new ArrayList<News>();

```

- ◆ 设置 layout 为 newslist.xml，取得 Intent 中的 Bundle 对象，并取得 Bundle 对象中的数据，然后调用 getRss() 取得解析后的 List。具体代码如下所示。

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /* 设置layout为newslist.xml */
    setContentView(R.layout.newslist);

    mText=(TextView) findViewById(R.id.myText);
    /* 取得Intent中的Bundle对象 */
    Intent intent=this.getIntent();
    Bundle bunde = intent.getExtras();
    /* 取得Bundle对象中的数据 */
    String path = bunde.getString("path");
    /* 调用getRss()取得解析后的List */
    li=getRss(path);
    mText.setText(title);
    /* 设置自定义的MyAdapter */
    setListAdapter(new MyAdapter(this,li));
}

```

- ◆ 定义 onListItemClick 来监听 ListItem 被点击时要做的动作。具体代码如下所示。

```

/* 设置ListItem被点击时要做的动作 */
@Override
protected void onListItemClick(ListView l, View v, int position,
                             long id)
{
    /* 取得News对象 */
    News ns=(News)li.get(position);
    /* new一个Intent对象，并指定class */
    Intent intent = new Intent();
    intent.setClass(RSSL_1.this,RSSL_2.class);
    /* new一个Bundle对象，并将要传递的数据传入 */
    Bundle bundle = new Bundle();
    bundle.putString("title",ns.getTitle());
    bundle.putString("desc",ns.getDesc());
    bundle.putString("link",ns.getLink());
    /* 将Bundle对象assign给Intent */

```

```
    intent.putExtras(bundle);
    /* 调用Activity RSSL_2 */
    startActivity(intent);
}
```

- ◆ 定义方法 getRss(String path) 来解析 XML，如果有异常则输出错误提示对话框。具体代码如下所示。

```
/* 解析XML的方法 */
private List<News> getRss(String path)
{
    List<News> data=new ArrayList<News>();
    URL url = null;
    try
    {
        url = new URL(path);
        /* 产生SAXParser对象 */
        SAXParserFactory spf = SAXParserFactory.newInstance();
        SAXParser sp = spf.newSAXParser();
        /* 产生XMLReader对象 */
        XMLReader xr = sp.getXMLReader();
        /* 设置自定义的MyHandler给XMLReader */
        MyHandler myExampleHandler = new MyHandler();
        xr.setContentHandler(myExampleHandler);
        /* 解析XML */
        xr.parse(new InputSource(url.openStream()));
        /* 取得RSS标题与内容列表 */
        data =myExampleHandler.getParsedData();
        title=myExampleHandler.getRssTitle();
    }
    catch (Exception e)
    {
        /* 发生错误时返回result回上一个activity */
        Intent intent=new Intent();
        Bundle bundle = new Bundle();
        bundle.putString("error",""+e);
        intent.putExtras(bundle);
        /* 错误的返回值设置为99 */
        RSSL_1.this.setResult(99, intent);
        RSSL_1.this.finish();
    }
    return data;
}
```



step 04 编写文件 RSSL_2.java，此文件是由 RSSL_1 唤起的，用于显示上一个 Activity 所单击的新闻内容。当程序被唤起后，会首先从 Bundle 对象中获取 News 的 title、link 和 desc，显示在画面中。以 Linkify.addLinks() 将 link 设置为一个 WEB_URLS 形式的链接。当用户单击链接后，会通过设置的网址直接打开 Web 浏览器来浏览网页。文件 RSSL_2.java 的具体代码如下所示。

```
package irdc.RSSL;

/* import相关class */
import irdc.RSSL.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.util.Linkify;
import android.widget.TextView;

public class RSSL_2 extends Activity
{
    /* 变量声明 */
    private TextView mTitle;
    private TextView mDesc;
    private TextView mLink;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 设置layout为newscontent.xml */
        setContentView(R.layout.newscontent);
        /* 初始化对象 */
        mTitle=(TextView) findViewById(R.id.myTitle);
        mDesc=(TextView) findViewById(R.id.myDesc);
        mLink=(TextView) findViewById(R.id.myLink);

        /* 取得Intent中的Bundle对象 */
        Intent intent=this.getIntent();
        Bundle bunde = intent.getExtras();
        /* 取得Bundle对象中的数据 */
        mTitle.setText(bunde.getString("title"));
        mDesc.setText(bunde.getString("desc")+"....");
        mLink.setText(bunde.getString("link"));
        /* 设置mLink为网页连接 */
        Linkify.addLinks(mLink,Linkify.WEB_URLS);
```

```
    }  
}
```

step 05 编写文件 News.java，在此定义了一个 JavaBean 类来存放每一篇新闻信息。每一个 News 对象代表了一条新闻，在 News 对象中定义了新闻的标题、描述、网站链接和发布时间这 4 个属性。JavaBean 类中的方法都是以 setAAA() 和 getAAA() 方式来命名的，所以用 setAAA() 来设置属性值，或通过 getAAA() 来获取属性值。文件 News.java 的具体代码如下所示。

```
package irdc.RSSL;  
  
public class News  
{  
    /* News      跑计 */  
    private String _title="";  
    private String _link="";  
    private String _desc="";  
    private String _date="";  
  
    public String getTitle()  
    {  
        return _title;  
    }  
    public String getLink()  
    {  
        return _link;  
    }  
    public String getDesc()  
    {  
        return _desc;  
    }  
    public String getDate()  
    {  
        return _date;  
    }  
    public void setTitle(String title)  
    {  
        _title=title;  
    }  
    public void setLink(String link)  
    {  
        _link=link;  
    }
```

```
public void setDesc(String desc)
{
    _desc=desc;
}
public void setDate(String date)
{
    _date=date;
}
```

step 06 编写文件 MyAdapter.java，在此定义一个继承自 android.widget.BaseAdapter 的 Adapter 对象，通过此对象来设置 ListView 中要显示的信息，以 news_row.xml 作为布局。文件 MyAdapter.java 的实现代码如下所示。

```
package irdc.RSSL;

/* import相关class */
import irdc.RSSL.R;

import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

/* 自定义的Adapter，继承android.widget.BaseAdapter */
public class MyAdapter extends BaseAdapter
{
    /* 变量声明 */
    private LayoutInflater mInflater;
    private List<News> items;

    /* MyAdapter的构造器，传递两个参数 */
    public MyAdapter(Context context, List<News> it)
    {
        /* 参数初始化 */
        mInflater = LayoutInflater.from(context);
        items = it;
    }

    /* 因继承BaseAdapter，需重写以下方法 */
}
```

```
@Override
public int getCount()
{
    return items.size();
}

@Override
public Object getItem(int position)
{
    return items.get(position);
}

@Override
public long getItemId(int position)
{
    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup par)
{
    ViewHolder holder;

    if(convertView == null)
    {
        /* 使用自定义的news_row作为Layout */
        convertView = mInflater.inflate(R.layout.news_row, null);
        /* 初始化holder的text与icon */
        holder = new ViewHolder();
        holder.text = (TextView) convertView.findViewById(R.id.text);
        convertView.setTag(holder);
    }
    else
    {
        holder = (ViewHolder) convertView.getTag();
    }

    News tmpN=(News) items.get(position);
    holder.text.setText(tmpN.getTitle());

    return convertView;
}

/* class ViewHolder */
```

```

private class ViewHolder
{
    TextView text;
}
}

```

step 07 编写文件 MyHandler.java，在此定义了继承自 org.xml.sax.helpers.DefaultHandler 的 MyHandler 对象，此对象用于解析 XML 文件，并获取对应的信息。文件 MyHandler.java 的具体实现流程如下所示。

- 引入相关 class，然后分别声明各个变量。具体代码如下所示。

```

package irdc.RSSL;

/* import相关class */
import java.util.ArrayList;
import java.util.List;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class MyHandler extends DefaultHandler
{
    /* 变量声明 */
    private boolean in_item = false;
    private boolean in_title = false;
    private boolean in_link = false;
    private boolean in_desc = false;
    private boolean in_date = false;
    private boolean in_mainTitle = false;
    private List<News> li;
    private News news;
    private String title="";
    private StringBuffer buf=new StringBuffer();
}

```

- 分别将转换成 List<News> 的 XML 数据返回，将解析出的 RSS title 返回。然后调用 startDocument()，开始解析操作。当解析结束时，调用 endDocument()。当解析到 Element 开头时，调用 startElement 方法。具体代码如下所示。

```

/* 将转换成List<News>的XML数据返回 */
public List<News> getParsedData()
{
    return li;
}

```

```
/* 将解析出的RSS title返回 */
public String getRssTitle()
{
    return title;
}
/* XML文件开始解析时调用此方法 */
@Override
public void startDocument() throws SAXException
{
    li = new ArrayList<News>();
}
/* XML文件结束解析时调用此方法 */
@Override
public void endDocument() throws SAXException
{
}
/* 解析到Element的开头时调用此方法 */
@Override
public void startElement(String namespaceURI, String localName,
                         String qName, Attributes atts) throws SAXException
{
    if (localName.equals("item"))
    {
        this.in_item = true;
        /* 解析到item的开头时new一个News对象 */
        news=new News();
    }
    else if (localName.equals("title"))
    {
        if(this.in_item)
        {
            this.in_title = true;
        }
        else
        {
            this.in_mainTitle = true;
        }
    }
    else if (localName.equals("link"))
    {
        if(this.in_item)
        {
            this.in_link = true;
        }
    }
}
```

```
    }
}
else if (localName.equals("description"))
{
    if(this.in_item)
    {
        this.in_desc = true;
    }
}
else if (localName.equals("pubDate"))
{
    if(this.in_item)
    {
        this.in_date = true;
    }
}
}
```

- ◆ 当解析到 Element 的结尾时调用 endElement 方法，具体代码如下所示。

```
/* 解析到Element的结尾时调用此方法 */
@Override
public void endElement(String namespaceURI, String localName,
                       String qName) throws SAXException
{
    if (localName.equals("item"))
    {
        this.in_item = false;
        /* 解析到item的结尾时将News对象写入List中 */
        li.add(news);
    }
    else if (localName.equals("title"))
    {
        if(this.in_item)
        {
            /* 设置News对象的title */
            news.setTitle(buf.toString().trim());
            buf.setLength(0);
            this.in_title = false;
        }
        else
        {
            /* 设置RSS的title */
            title=buf.toString().trim();
        }
    }
}
```



```
buf.setLength(0);
this.in_mainTitle = false;
}
}
else if (localName.equals("link"))
{
if(this.in_item)
{
/* 设置News对象的link */
news.setLink(buf.toString().trim());
buf.setLength(0);
this.in_link = false;
}
}
else if (localName.equals("description"))
{
if(in_item)
{
/* 设置News对象的description */
news.setDesc(buf.toString().trim());
buf.setLength(0);
this.in_desc = false;
}
}
else if (localName.equals("pubDate"))
{
if(in_item)
{
/* 设置News对象的pubDate */
news.setDate(buf.toString().trim());
buf.setLength(0);
this.in_date = false;
}
}
}
```

- ◆ 定义方法 characters 来于获取 Element 开头和结尾中间的字符串，具体代码如下所示。

```
/* 取得Element的开头结尾中间夹的字符串 */
@Override
public void characters(char ch[], int start, int length)
{
if(this.in_item||this.in_mainTitle)
```



```

    {
        /* 将char[]添加StringBuffer */
        buf.append(ch,start,length);
    }
}
}

```

执行后的效果如图 7-11 所示。在文本框中输入 RSS 网址 <http://rss.sina.com.cn/news/marquee/ddt.xml>，然后单击【开始解析】按钮后会在屏幕中列表显示 RSS 新闻，如图 7-12 所示。单击某条新闻后，会显示此新闻的详情，如图 7-13 所示。

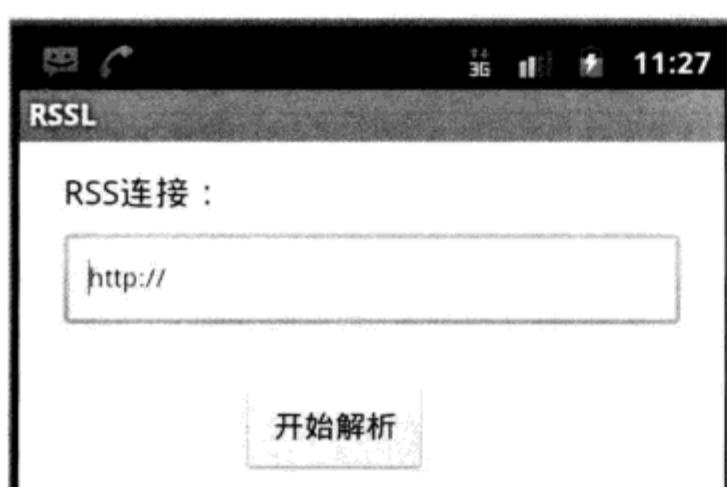


图7-11 初始效果

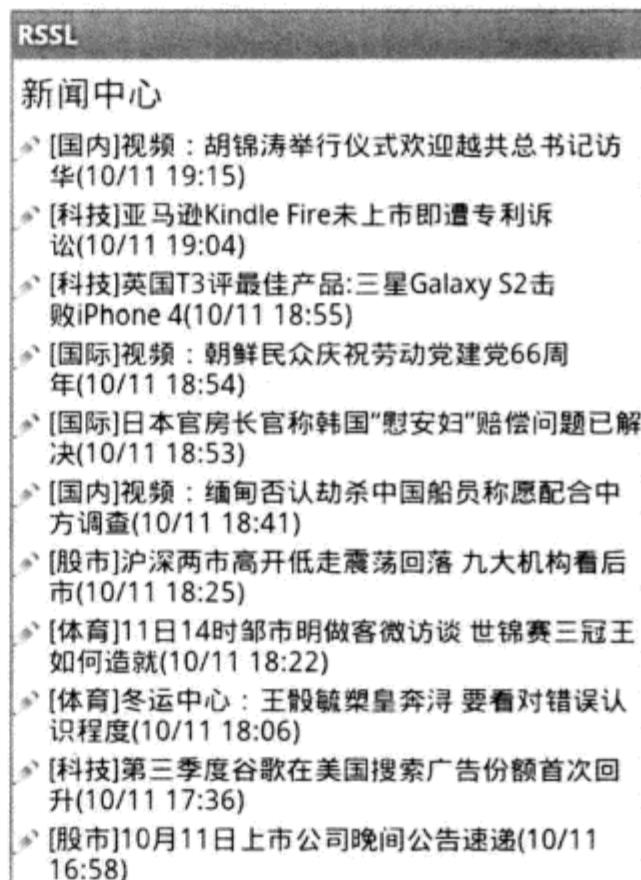


图7-12 RSS列表

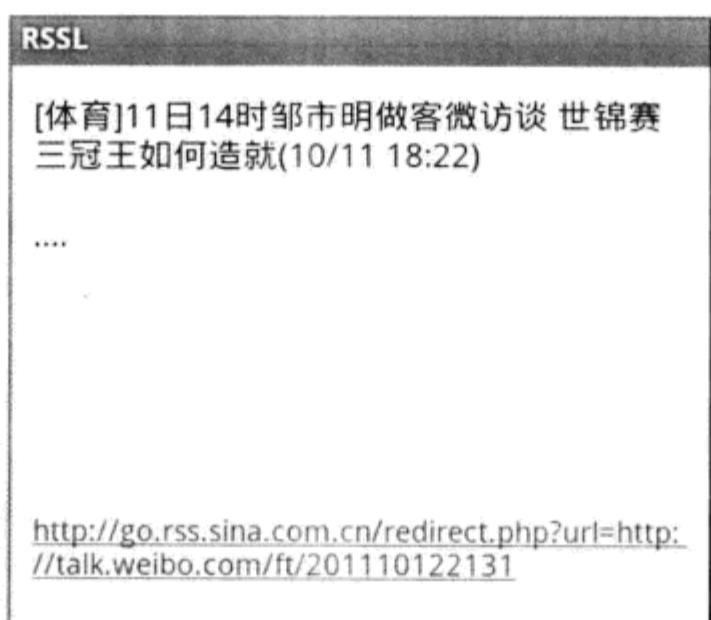


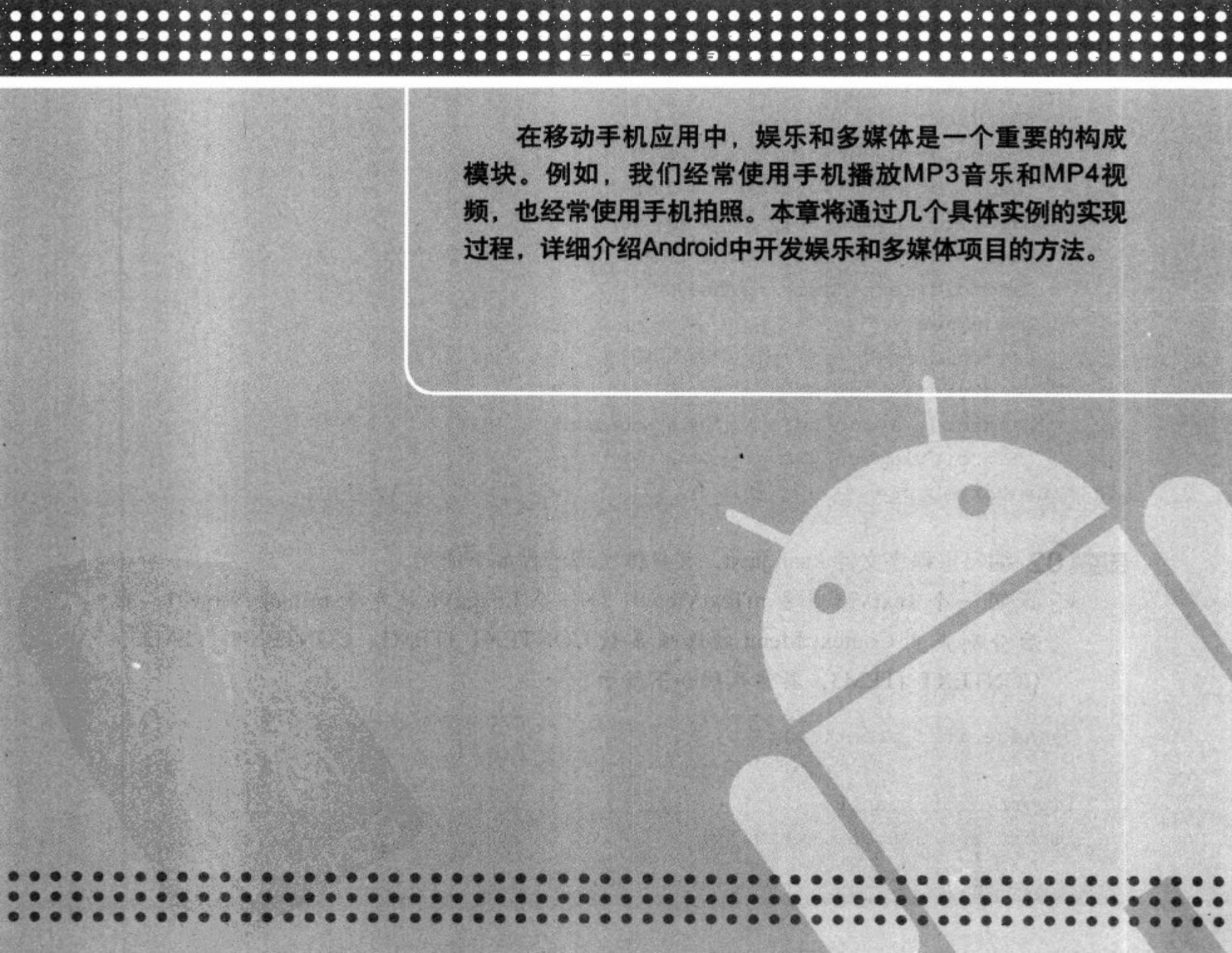
图7-13 新闻详情





多媒体实战演练

在移动手机应用中，娱乐和多媒体是一个重要的构成模块。例如，我们经常使用手机播放MP3音乐和MP4视频，也经常使用手机拍照。本章将通过几个具体实例的实现过程，详细介绍Android中开发娱乐和多媒体项目的方法。



8.1 获取图片的宽和高

在本实例中，通过一个 ListView 对象并设置了 setOnCreateContextMenuListener，当用户单击一个选项后，能够弹出多个 ContextView Menu，并以 onContextItemSelected 来判断用户是单击了哪个选项，最后分别获取图片的宽和高。在具体实现上，通过 Bitmap 对象的 BitmapFactory.decodeResource() 方法来获取预先设定的图片“baby.png”，然后再通过 Bitmap 对象的 getHeight() 和 getWidth() 来获取图片的宽和高。

本实例的源码保存在【光盘\daima\第 8 章\kuan】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:text="@string/hello"/>
    <ImageView
        android:id="@+id/myImageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />
</LinearLayout>
```

step 02 编写主程序文件 kuan.java，其具体实现流程如下所示。

- 声明一个 TextView 变量 mTextView01 和一个 ImageView 变量 mImageView01，然后分别声明 Context Menu 的选项常数 CONTEXT_ITEM1、CONTEXT_ITEM2 和 CONTEXT_ITEM3。具体代码如下所示。

```
package irdc.practice1;

import irdc.kuan.R;
import android.app.Activity;
```

```

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
public class kuan extends Activity
{
    /*声明一个 TextView变量与一个ImageView变量*/
    private TextView mTextView01;
    private ImageView mImageView01;
    /*声明Context Menu的选项常数*/
    protected static final int CONTEXT_ITEM1 = Menu.FIRST;
    protected static final int CONTEXT_ITEM2 = Menu.FIRST+1;
    protected static final int CONTEXT_ITEM3 = Menu.FIRST+2;

```

- 通过构造器创建 TextView 和 ImageView 对象，然后将 Drawable 中的图片 baby.png 放入自定义的 ImageView 中。具体代码如下所示。

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*通过findViewById构造器创建TextView与ImageView对象*/
    mTextView01 = (TextView) findViewById(R.id.myTextView1);
    mImageView01= (ImageView) findViewById(R.id.myImageView1);
    /*将Drawable中的图片baby.png放入自定义的ImageView中*/
    mImageView01.setImageDrawable(getResources().
        getDrawable(R.drawable.baby));

```

- 设置 OnCreateContextMenuListener 给 TextView，这样图片上可以使用 ContextMenu，然后覆盖 OnCreateContextMenu 来创建 ContextMenu 的选项。具体代码如下所示。

```

/*设置OnCreateContextMenuListener给TextView
 * 让图片上可以使用ContextMenu*/
mImageView01.setOnCreateContextMenuListener

```



```
(new ListView.OnCreateContextMenuListener()
{
    @Override
    /*覆盖OnCreateContextMenu来创建ContextMenu的选项*/
    public void onCreateContextMenu
    (ContextMenu menu, View v, ContextMenuInfo menuInfo)
    {
        // TODO Auto-generated method stub
        menu.add(Menu.NONE, CONTEXT_ITEM1, 0, R.string.str_context1);
        menu.add(Menu.NONE, CONTEXT_ITEM2, 0, R.string.str_context2);
        menu.add(Menu.NONE, CONTEXT_ITEM3, 0, R.string.str_context3);
    }
});
```

- ◆ 覆盖 OnContextItemSelected 来定义用户点击 Menu 后的动作，然后通过自定义 Bitmap 对象 BitmapFactory.decodeResource 来取得预设的图片资源。具体代码如下所示。

```
@Override
/*覆盖OnContextItemSelected来定义用户点击menu后的动作*/
public boolean onContextItemSelected(MenuItem item)
{
    // TODO Auto-generated method stub
    /*自定义Bitmap对象并通过BitmapFactory.decodeResource取得
     *预先Import至Drawable的baby.png图档*/
    Bitmap myBmp = BitmapFactory.decodeResource
        (getResources(), R.drawable.baby);
    /*通过Bitmap对象的getHeight与getWidth来取得图片宽高*/
    int intHeight = myBmp.getHeight();
    int intWidth = myBmp.getWidth();
```

- ◆ 根据用户选择的选项，分别通过 getHeight() 和 getWidth() 获取对应图片宽度和高度。具体代码如下所示。

```
try
{
    /*菜单选项与动作*/
    switch(item.getItemId())
    {
        /*将图片宽度显示在TextView中*/
        case CONTEXT_ITEM1:
            String strOpt =
                getResources().getString(R.string.str_width)
```

```

        +"="+Integer.toString(intWidth);
        mTextView01.setText(strOpt);
        break;
    /*将图片高度显示在TextView中*/
    case CONTEXT_ITEM2:
        String strOpt2 =
            getResources().getString(R.string.str_height)
        +"="+Integer.toString(intHeight);
        mTextView01.setText(strOpt2);
        break;
    /*将图片宽高显示在TextView中*/
    case CONTEXT_ITEM3:
        String strOpt3 =
            getResources().getString(R.string.str_width)
        +"="+Integer.toString(intWidth)+"\n"
        +getResources().getString(R.string.str_height)
        +"="+Integer.toString(intHeight);
        mTextView01.setText(strOpt3);
        break;
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
return super.onContextItemSelected(item);
}
}

```

执行后的初始效果如图 8-1 所示，当长时间选中图片后会弹出用户选项，如图 8-2 所示。当选择一个选项后，会弹出对应的获取数值，如图 8-3 所示。

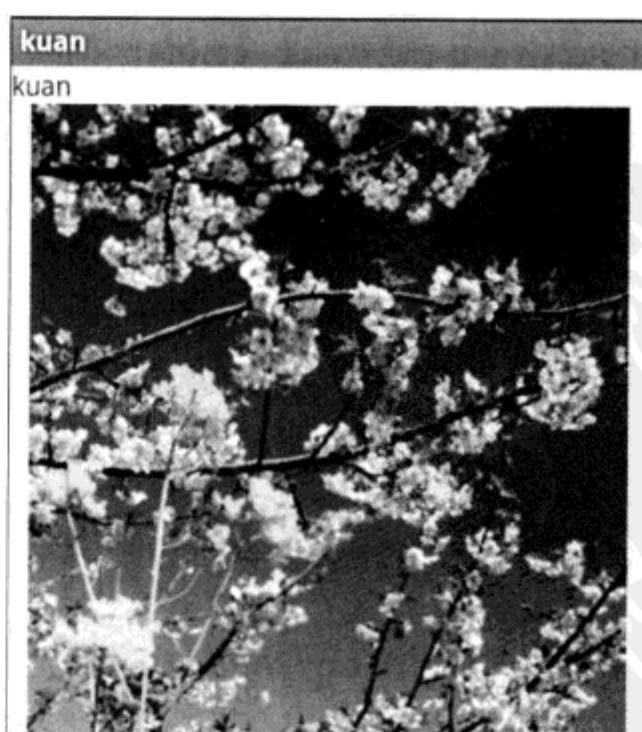


图8-1 初始效果

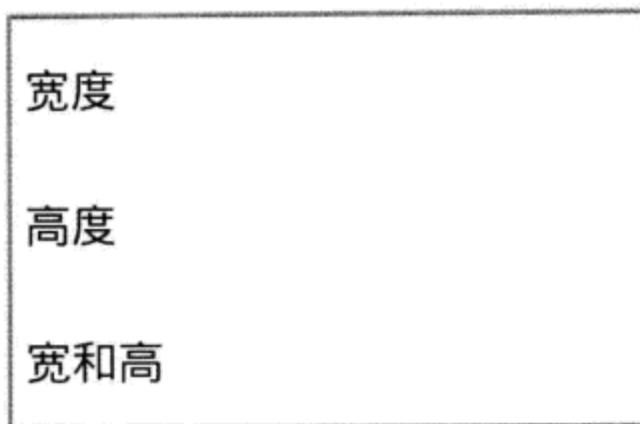


图8-2 弹出选项



图8-3 获取图片宽和高

8.2 绘制各种几何图形

当前很多手机游戏程序需要绘制几何图形来完成。在本实例中，通过 `Android.graphics` 的类来绘制 2D 向量图。在其 package 中提供了很多在手机上绘制图形的类和方法。例如 `Canvas` 相当于一个图纸，所有的图形都在它上面绘制并显示出来。而 `Paint` 则像铅笔，可以设置为不同的颜色，从而绘制不同颜色的图形。本实例的源码保存在【光盘\daima\第8章\jihe】，具体实现流程如下所示。

step 01 编写值文件 `strings.xml`，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">jihe</string>
    <string name="app_name">jihe</string>
    <string name="str_text1">圆形</string>
    <string name="str_text2">正方形</string>
    <string name="str_text3">长方形</string>
    <string name="str_text4">椭圆形</string>
    <string name="str_text5">三角形</string>
    <string name="str_text6">梯形</string>
</resources>
```

step 02 编写主程序文件 `jihe.java`，其具体实现流程如下所示。

- ◆ 引入相关 class 类，然后设置 `ContentView` 为自定义的 `MyView`。具体代码如下所示。

```
package irdc.jihe;

/* import相关class */
import irdc.jihe.R;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
```

```
import android.graphics.Color;
import android.graphics.LinearGradient;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.RectF;
import android.graphics.Shader;
import android.os.Bundle;
import android.view.View;

public class jihe extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        /* 设置ContentView为自定义的MyView */
        MyView myView = new MyView(this);
        setContentView(myView);
    }
}
```

- ◆ 自定义继承 View 的 MyView，具体代码如下所示。

```
/* 自定义继承View的MyView */
private class MyView extends View
{
    public MyView(Context context)
    {
        super(context);
    }
}
```

- ◆ 设置背景和消除锯齿，然后分别绘制空心圆形、空心正方形、空心长方形、空心椭圆形、空心三角形和空心梯形。具体代码如下所示。

```
/* 覆盖onDraw() */
@Override
protected void onDraw(Canvas canvas)
{
    super.onDraw(canvas);
    /* 设置背景为白色 */
    canvas.drawColor(Color.WHITE);
    Paint paint = new Paint();
    /* 去锯齿 */
    paint.setAntiAlias(true);
    /* 设置paint的颜色 */
}
```



```
paint.setColor(Color.RED);
/* 设置paint的style为STROKE: 空心的 */
paint.setStyle(Paint.Style.STROKE);
/* 设置paint的外框宽度 */
paint.setStrokeWidth(3);
/* 画一个空心圆形 */
canvas.drawCircle(40, 40, 30, paint);
/* 画一个空心正方形 */
canvas.drawRect(10, 90, 70, 150,paint);
/* 画一个空心长方形 */
canvas.drawRect(10, 170, 70, 200,paint);
/* 画一个空心椭圆形 */
RectF re=new RectF(10,220,70,250);
canvas.drawOval(re, paint);
/* 画一个空心三角形 */
Path path = new Path();
path.moveTo(10,330);
path.lineTo(70,330);
path.lineTo(40,270);
path.close();
canvas.drawPath(path, paint);
/* 画一个空心梯形 */
Path path1 = new Path();
path1.moveTo(10,410);
path1.lineTo(70,410);
path1.lineTo(55,350);
path1.lineTo(25,350);
path1.close();
canvas.drawPath(path1, paint);
```

- ◆ 设置实心样式和颜色，然后分别绘制实心圆形、实心正方形、实心长方形、实心椭圆形、实心三角形和实心梯形。具体代码如下所示。

```
/* 设置paint的style为FILL: 实心 */
paint.setStyle(Paint.Style.FILL);
/* 设置paint的颜色 */
paint.setColor(Color.BLUE);

/* 画一个实心圆 */
canvas.drawCircle(120, 40, 30, paint);
/* 画一个实心正方形 */
canvas.drawRect(90, 90, 150, 150,paint);
/* 画一个实心长方形 */
```

```
canvas.drawRect(90,170,150,200,paint);
/* 画一个实心椭圆形 */
RectF re2=new RectF(90,220,150,250);
canvas.drawOval(re2, paint);
/* 画一个实心三角形 */
Path path2 = new Path();
path2.moveTo(90,330);
path2.lineTo(150,330);
path2.lineTo(120,270);
path2.close();
canvas.drawPath(path2, paint);
/* 画一个实心梯形 */
Path path3 = new Path();
path3.moveTo(90,410);
path3.lineTo(150,410);
path3.lineTo(135,350);
path3.lineTo(105,350);
path3.close();
canvas.drawPath(path3, paint);
```

- ◆ 设置渐变样式和颜色，然后分别绘制渐变圆形、渐变正方形、渐变长方形、渐变椭圆形、渐变三角形和渐变梯形。具体代码如下所示。

```
/* 设置渐变色 */
Shader mShader=new LinearGradient(0, 0,100,100,
    new int[]{Color.RED, Color.GREEN,Color.BLUE,Color.YELLOW},
    null, Shader.TileMode.REPEAT);
paint.setShader(mShader);
/* 画一个渐变色的圆形 */
canvas.drawCircle(200,40, 30, paint);
/* 画一个渐变色的正方形 */
canvas.drawRect(170,90,230,150,paint);
/* 画一个渐变色的长方形 */
canvas.drawRect(170,170,230,200,paint);
/* 画一个渐变色的椭圆形 */
RectF re3=new RectF(170,220,230,250);
canvas.drawOval(re3, paint);
/* 画一个渐变色的三角形 */
Path path4 = new Path();
path4.moveTo(170,330);
path4.lineTo(230,330);
path4.lineTo(200,270);
path4.close();
```



```
canvas.drawPath(path4, paint);
/* 画一个渐变色的梯形 */
Path path5 = new Path();
path5.moveTo(170, 410);
path5.lineTo(230, 410);
path5.lineTo(215, 350);
path5.lineTo(185, 350);
path5.close();
canvas.drawPath(path5, paint);
```

- ◆ 通过 canvas.drawText 实现写字功能，具体代码如下所示。

```
/* 写字 */
paint.setTextSize(24);
canvas.drawText(getResources().getString(R.string.str_text1),
               240, 50, paint);
canvas.drawText(getResources().getString(R.string.str_text2),
               240, 120, paint);
canvas.drawText(getResources().getString(R.string.str_text3),
               240, 190, paint);
canvas.drawText(getResources().getString(R.string.str_text4),
               240, 250, paint);
canvas.drawText(getResources().getString(R.string.str_text5),
               240, 320, paint);
canvas.drawText(getResources().getString(R.string.str_text6),
               240, 390, paint);
}
}
```

执行后将会在屏幕内显示不同的图形，并在后面显示对应的文字描述。效果如图 8-4 所示。

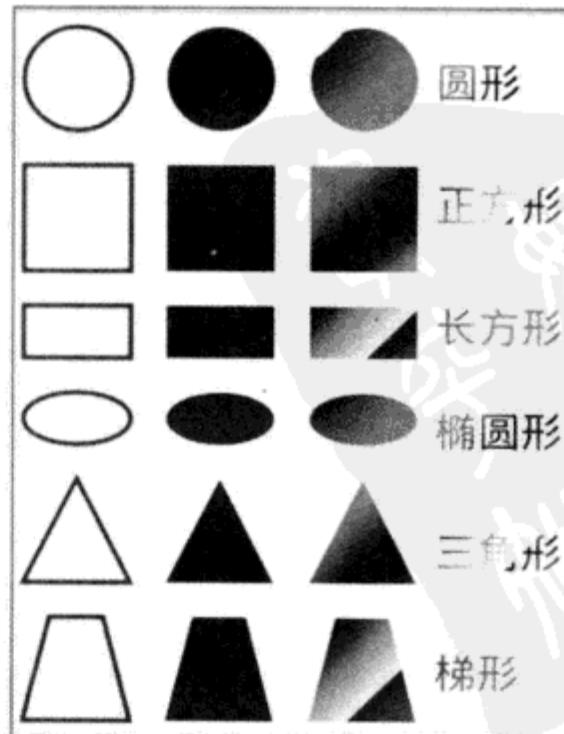


图8-4 执行效果



8.3 开发一个手机屏保程序

本实例实现了一个手机屏保功能，通过控制和判断用户静止未触动手机键盘或屏幕的时间及其事件，并且通过动态全屏幕淡入、淡出和图片交换效果。上述功能是通过线程实现的，我们以时间戳记的方式，判断距离上一次单击键盘或屏幕的时间，并计算两次的间隔。当超过了设置了时间后，会进入屏保程序，本实例设置的时间间隔为5秒。本实例的源码保存在【光盘\daima\第8章\pingbao】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ImageView
        android:id="@+id/myImageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitCenter"
        android:layout_gravity="center" />
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:visible="true"
        android:text="@string/str_set_pwd"/>
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""/>
</LinearLayout>
```

step 02 编写主程序文件 pingbao.java，其具体实现流程如下所示。

- 引入相关 class 类，然后设置 LayoutInflater 对象作为新建的 AlertDialog。具体代码如下所示。



```
package irdc.pingbao;

import irdc.pingbao.R;

import java.util.Date;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

public class pingbao extends Activity
{
    private TextView mTextView01;
    private ImageView mImageView01;

    /* LayoutInflater对象作为新建AlertDialog之用 */
    private LayoutInflater mInflater01;
```

- ◆ 用 mView01 来输入解锁的 View。通过 menu 选项 identifier，用以识别对应的事件。
具体代码如下所示。

```
/* 输入解锁的View */
private View mView01;
private EditText mEditText01,mEditText02;
/* menu选项identifier，用以识别事件 */
```

```

static final private int MENU_ABOUT = Menu.FIRST;
static final private int MENU_EXIT = Menu.FIRST+1;
private Handler mHandler01 = new Handler();
private Handler mHandler02 = new Handler();
private Handler mHandler03 = new Handler();
private Handler mHandler04 = new Handler();

```

- ◆ 分别定义控制 User 静止与否的 Counter，控制 FadeIn 与 Fade Out 的 Counter，控制循序替换背景图 ID 的 Counter。具体代码如下所示。

```

/* 控制User静止与否的Counter */
private int intCounter1, intCounter2;
/* 控制FadeIn与Fade Out的Counter */
private int intCounter3, intCounter4;
/* 控制循序替换背景图ID的Counter */
private int intDrawable=0;

```

- ◆ 设置 timePeriod，设置当静止超过 n 秒将自动进入屏幕保护。具体代码如下所示。

```

/* 上一次User有动作的Time Stamp */
private Date lastUpdateTime;
/* 计算User共几秒没有动作 */
private long timePeriod;
/* 静止超过n秒将自动进入屏幕保护 */
private float fHoldStillSecond = (float) 5;
private boolean bIfRunScreenSaver;
private boolean bFadeFlagOut, bFadeFlagIn = false;
private long intervalScreenSaver = 1000;
private long intervalKeypadeSaver = 1000;
private long intervalFade = 100;
private int screenWidth, screenHeight;

```

- ◆ 设置每 5 秒置换图片，用 Screen Saver 保存需要用到的背景图。具体代码如下所示。

```

/* 每n秒置换图片 */
private int intSecondsToChange = 5;

/* 设置Screen Saver需要用到的背景图 */
private static int[] screenDrawable = new int[]
{
    R.drawable.screen1,
    R.drawable.screen2,
    R.drawable.screen3,
    R.drawable.screen4,
    R.drawable.screen5
};

```

- ◆ 设置在 setContentView 之前调用全屏幕显示，通过 lastUpdateTime 初始取得 User 触碰手机的时间，并用 recoverOriginalLayout() 来初始化 Layout 上的 Widget 可见性。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    /* 必须在setContentView之前调用全屏幕显示 */
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags
    (
        WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN
    );
    setContentView(R.layout.main);

    /* onCreate all Widget */
    mTextView01 = (TextView) findViewById(R.id.myTextView1);
    mImageView01 = (ImageView) findViewById(R.id.myImageView1);
    mEditText01 = (EditText) findViewById(R.id.myEditText1);
    /* 初始取得用户触碰手机的时间 */
    lastUpdateTime = new Date(System.currentTimeMillis());
    /* 初始化Layout上的Widget可见性 */
    recoverOriginalLayout();
}
```

- ◆ 设置 menu 群组 ID，然后通过 menu.add 创建具有 SubMenu 的 menu，最后创建退出 Menu。具体代码如下所示。

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    // TODO Auto-generated method stub

    /* menu群组ID */
    int idGroup1 = 0;

    /* The order position of the item */
    int orderMenuItem1 = Menu.NONE;
    int orderMenuItem2 = Menu.NONE+1;

    /* 创建具有SubMenu的menu */
}
```

```

menu.add
(
    idGroup1, MENU_ABOUT, orderMenuItem1, R.string.app_about
);
/* 创建退出Menu */
menu.add(idGroup1, MENU_EXIT, orderMenuItem2, R.string.str_exit);
menu.setGroupCheckable(idGroup1, true, true);

return super.onCreateOptionsMenu(menu);
}

```

- ◆ 根据用户选择的 Menu 选项显示对应的 AlertDialog 提示框，具体代码如下所示。

```

public boolean onOptionsItemSelected(MenuItem item)
{
    switch(item.getItemId())
    {
        case (MENU_ABOUT):
            new AlertDialog.Builder
            (
                pingbao.this
            ).setTitle(R.string.app_about).setIcon
            (
                R.drawable.hippo
            ).setMessage
            (
                R.string.app_about_msg
            ).setPositiveButton(R.string.str_ok,
            new DialogInterface.OnClickListener()
            {
                public void onClick
                (DialogInterface dialoginterface, int i)
                {
                }
            })
            .show();
            break;
        case (MENU_EXIT):
            /* 离开程序 */
            finish();
            break;
    }
    return super.onOptionsItemSelected(item);
}

```

- ◆ 用 mTasks01 监控 User 没有动作的运行线程，通过 timePeriod 计算 User 静止不动



作的时间间距，如果静止不动，超过设置的5秒，则运行对应的线程。具体代码如下所示。

```
/* 监控User没有动作的运行线程 */
private Runnable mTasks01 = new Runnable()
{
    public void run()
    {
        intCounter1++;
        Date timeNow = new Date(System.currentTimeMillis());

        /* 计算用户静止不动作的时间间隔 */
        timePeriod =
            (long)timeNow.getTime() - (long)lastUpdateTime.getTime();

        float timePeriodSecond = ((float)timePeriod/1000);

        /* 如果超过时间静止不动 */
        if(timePeriodSecond>fHoldStillSecond)
        {
            /* 静止超过时间第一次的标记 */
            if(bIfRunScreenSaver==false)
            {
                /* 启动运行线程2 */
                mHandler02.postDelayed(mTasks02, intervalScreenSaver);
                if(intCounter1%(intSecondsToChange)==0)
                {
                    bFadeFlagOut=true;
                    mHandler03.postDelayed(mTasks03, intervalFade);
                }
                else
                {
                    /* 在Fade Out后立即Fade In */
                    if(bFadeFlagOut==true)
                    {
                        bFadeFlagIn=true;
                        mHandler04.postDelayed(mTasks04, intervalFade);
                    }
                    else
                    {
                        bFadeFlagIn=false;
                        intCounter4 = 0;
                        mHandler04.removeCallbacks(mTasks04);
                    }
                }
            }
        }
    }
}
```

```
    intCounter3 = 0;
    bFadeFlagOut = false;
}
bIfRunScreenSaver = true;
}
else
{
    /* screen saver 正在运行中 */
    if(intCounter1%(intSecondsToChange)==0)
    {
        bFadeFlagOut=true;
        mHandler03.postDelayed(mTasks03, intervalFade);
    }
    else
    {
        /* 在Fade Out后立即Fade In */
        if(bFadeFlagOut==true)
        {
            bFadeFlagIn=true;
            mHandler04.postDelayed(mTasks04, intervalFade);
        }
        else
        {
            bFadeFlagIn=false;
            intCounter4 = 0;
            mHandler04.removeCallbacks(mTasks04);
        }
        intCounter3 = 0;
        bFadeFlagOut=false;
    }
}
}
else
{
    /* 当User没有动作的间距未超过时间 */
    bIfRunScreenSaver = false;
    /* 恢复原来的Layout Visible*/
    recoverOriginalLayout();
}

/* 以LogCat监看User静止不动的时间间距 */
Log.i
(
```



```
"HIPPO",
"Counter1:"+Integer.toString(intCounter1)+"/"+
Float.toString(timePeriodSecond));

/* 反复运行运行线程1 */
mHandler01.postDelayed(mTasks01, intervalKeypadSaver);
}

};
```

- ◆ 使用 mTasks02 设置每 1 秒运行一次屏保程序，隐藏原有 Layout 上面的 Widget，并调用 ScreenSaver() 加载图片，即轮换显示预设的 5 副图片。具体代码如下所示。

```
/* Screen Saver Runnable */
private Runnable mTasks02 =new Runnable()
{
    public void run()
    {
        if(bIfRunScreenSaver==true)
        {
            intCounter2++;
            hideOriginalLayout();
            showScreenSaver();
            mHandler02.postDelayed(mTasks02, intervalScreenSaver);
        }
        else
        {
            mHandler02.removeCallbacks(mTasks02);
        }
    }
};
```

- ◆ 定义 mTasks03，通过 setAlpha 设置 ImageView 的透明度渐暗下去。具体代码如下所示。

```
/* Fade Out特效Runnable */
private Runnable mTasks03 = new Runnable()
{
    public void run()
    {
        if(bIfRunScreenSaver==true && bFadeFlagOut==true)
        {
            intCounter3++;
            /* 设置ImageView的透明度渐暗下去 */
        }
    }
};
```

```

        mImageView01.setAlpha(255-intCounter3*28);

        Log.i("HIPPO", "Fade out:"+Integer.toString(intCounter3));
        mHandler03.postDelayed(mTasks03, intervalFade);
    }
    else
    {
        mHandler03.removeCallbacks(mTasks03);
    }
}
;

```

- ◆ 定义 mTasks03，通过 setAlpha 设置设置 ImageView 的透明度渐亮起来。具体代码如下所示。

```

/* Fade In特效Runnable */
private Runnable mTasks04 = new Runnable()
{
    public void run()
    {
        if(bIfRunScreenSaver==true && bFadeFlagIn==true)
        {
            intCounter4++;

            /* 设置ImageView的透明度渐亮起来 */
            mImageView01.setAlpha(intCounter4*28);

            mHandler04.postDelayed(mTasks04, intervalFade);
            Log.i("HIPPO", "Fade In:"+Integer.toString(intCounter4));
        }
        else
        {
            mHandler04.removeCallbacks(mTasks04);
        }
    }
};

```

- ◆ 定义 recoverOriginalLayout() 方法来恢复原有的 Layout 可视性，定义 hideOriginalLayout() 方法来隐藏原有应用程序里的布局配置组件。具体代码如下所示。

```

/* 恢复原有的Layout可视性 */
private void recoverOriginalLayout()
{
    mTextView01.setVisibility(View.VISIBLE);
}

```



```
mEditText01.setVisibility(View.VISIBLE);
mImageView01.setVisibility(View.GONE);
}

/* 隐藏原有应用程序里的布局配置组件 */
private void hideOriginalLayout()
{
    /* 将欲隐藏的Widget写在此 */
    mTextView01.setVisibility(View.INVISIBLE);
    mEditText01.setVisibility(View.INVISIBLE);
}

/* 开始ScreenSaver */
private void showScreenSaver()
{
    /* 屏幕保护之后要做的事件写在此*/

    if(intDrawable>4)
    {
        intDrawable = 0;
    }

    DisplayMetrics dm=new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);
    screenWidth = dm.widthPixels;
    screenHeight = dm.heightPixels;

    Bitmap bmp=BitmapFactory.decodeResource(getResources(),
screenDrawable[intDrawable]);

```

- ◆ 通过 Matrix 设置比例，使用 Matrix.postScale 设置维度 ReSize，通过 resizedBitmap 对象设置图文件至屏幕分辨率，新建 Drawable 对象 myNewBitmapDrawable 用于放大图文件至全屏幕，通过 setVisibility(View.VISIBLE) 使 ImageView 可见。具体代码如下所示。

```
/* Matrix比例 */
float scaleWidth = ((float) screenWidth) / bmp.getWidth();
float scaleHeight = ((float) screenHeight) / bmp.getHeight();
Matrix matrix = new Matrix();
/* 使用Matrix.postScale设置维度ReSize */
matrix.postScale(scaleWidth, scaleHeight);

/* ReSize图文件至屏幕分辨率 */
Bitmap resizedBitmap = Bitmap.createBitmap(
(
    bmp, 0, 0, bmp.getWidth(), bmp.getHeight(), matrix, true
)
```

```

);
/* 新建Drawable放大图文件至全屏幕 */
BitmapDrawable myNewBitmapDrawable =
    new BitmapDrawable(resizedBitmap);
mImageView01.setImageDrawable(myNewBitmapDrawable);

/* 使ImageView可见 */
mImageView01.setVisibility(View.VISIBLE);

/* 每间隔设置秒数替换图片ID，于下一个Runnable才会生效 */
if(intCounter2%intSecondsToChange==0)
{
    intDrawable++;
}
}

```

- ◆ 定义方法onUserWakeUpEvent() 实现解锁和加密处理，具体代码如下所示。

```

public void onUserWakeUpEvent()
{
    if(bIfRunScreenSaver==true)
    {
        try
        {
            /* LayoutInflater.from取得此Activity的context */
            mInflater01 = LayoutInflater.from(pingbao.this);

            /* 创建解锁密码使用View的Layout */
            mView01 = mInflater01.inflate(R.layout.securescreen, null);

            /* 于对话框中唯一的EditText等待输入解锁密码 */
            mEditText02 =
                (EditText) mView01.findViewById(R.id.myEditText2);

            /* 创建AlertDialog */
            new AlertDialog.Builder(this)
                .setView(mView01)
                .setPositiveButton("OK",
                    new DialogInterface.OnClickListener()
                {
                    public void onClick(DialogInterface dialog, int whichButton)
                    {

```

```
    /* 比较输入的密码与原Activity里的设置是否相符 */
    if (mEditText01.getText().toString().equals
        (mEditText02.getText().toString()))
    {
        /* 当密码正确才真的解锁屏幕保护装置 */
        resetScreenSaverListener();
    }
}
}).show();
}
catch (Exception e)
{
    e.printStackTrace();
}
}
}
}
```

- ◆ 定义 updateUserActionTime(), 用于统计用户单击键盘或屏幕的时间间隔。首先取得点击按键事件时的系统 Time Millis，然后重新计算点击按键距离上一次静止的时间间距。具体代码如下所示。

```
public void updateUserActionTime()
{
    /* 取得点击按键事件时的系统Time Millis */
    Date timeNow = new Date(System.currentTimeMillis());

    /* 重新计算点击按键距离上一次静止的时间间距 */
    timePeriod =
        (long)timeNow.getTime() - (long)lastUpdateTime.getTime();
    lastUpdateTime.setTime(timeNow.getTime());
}
```

- ◆ 定义方法 resetScreenSaverListener() 来重新设屏幕。具体代码如下所示。

```
public void resetScreenSaverListener()
{
    /* 删除现有的Runnable */
    mHandler01.removeCallbacks(mTasks01);
    mHandler02.removeCallbacks(mTasks02);

    /* 取得点击按键事件时的系统Time Millis */
    Date timeNow = new Date(System.currentTimeMillis());
    /* 重新计算点击按键距离上一次静止的时间间距 */
}
```

```

timePeriod =
    (long)timeNow.getTime() - (long)lastUpdateTime.getTime();
lastUpdateTime.setTime(timeNow.getTime());

/* for Runnable2, 取消屏幕保护 */
bIfRunScreenSaver = false;

/* 重置Runnable1与Runnable2的Counter */
intCounter1 = 0;
intCounter2 = 0;

/* 恢复原来的Layout Visible*/
recoverOriginalLayout();

/* 重新postDelayed()新的Runnable */
mHandler01.postDelayed(mTasks01, intervalKeypadeSaver);
}

```

- ◆ 定义 onKeyDown(int keyCode, KeyEvent event) 来监听用户的触摸单击事件，具体代码如下所示。

```

@Override
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    // TODO Auto-generated method stub
    if(bIfRunScreenSaver==true&& keyCode!=4)
    {
        /* 当屏幕保护程序正在运行中，触动解除屏幕保护程序 */
        onUserWakeUpEvent();
    }
    else
    {
        /* 更新User未触动手机的时间戳记 */
        updateUserActionTime();
    }
    return super.onKeyDown(keyCode, event);
}

@Override
public boolean onTouchEvent(MotionEvent event)
{
    // TODO Auto-generated method stub
    if(bIfRunScreenSaver==true)
    {
        /* 当屏幕保护程序正在运行中，触动解除屏幕保护程序 */
    }
}

```



```
    onUserWakeUpEvent();  
    ...  
    else  
    {  
        /* 更新User未触动手机的时间戳记 */  
        updateUserActionTime();  
  
        ...  
        return super.onTouchEvent(event);  
    }  
  
    @Override  
    protected void onResume()  
    {  
        // -TODO Auto-generated method stub  
        mHandler01.postDelayed(mTasks01, intervalKeypadeSaver);  
        super.onResume();  
    }  
}
```

- ◆ 定义方法 onPause() 来删除运行中的运行线程 mHandler01、mHandler02、mHandler03 和 mHandler04，具体代码如下所示。

```
protected void onPause()  
{  
    // TODO Auto-generated method stub  
  
    try  
    {  
        /* 删除运行中的运行线程 */  
        mHandler01.removeCallbacks(mTasks01);  
        mHandler02.removeCallbacks(mTasks02);  
        mHandler03.removeCallbacks(mTasks03);  
        mHandler04.removeCallbacks(mTasks04);  
    }  
    catch (Exception e)  
    {  
        e.printStackTrace();  
    }  
    super.onPause();  
}
```

执行后如果超过 5 秒不动键盘或屏幕，则会进入屏保状态，如图 8-5 所示。可以设置屏保密码，当输入正确的密码后才能解除屏保。

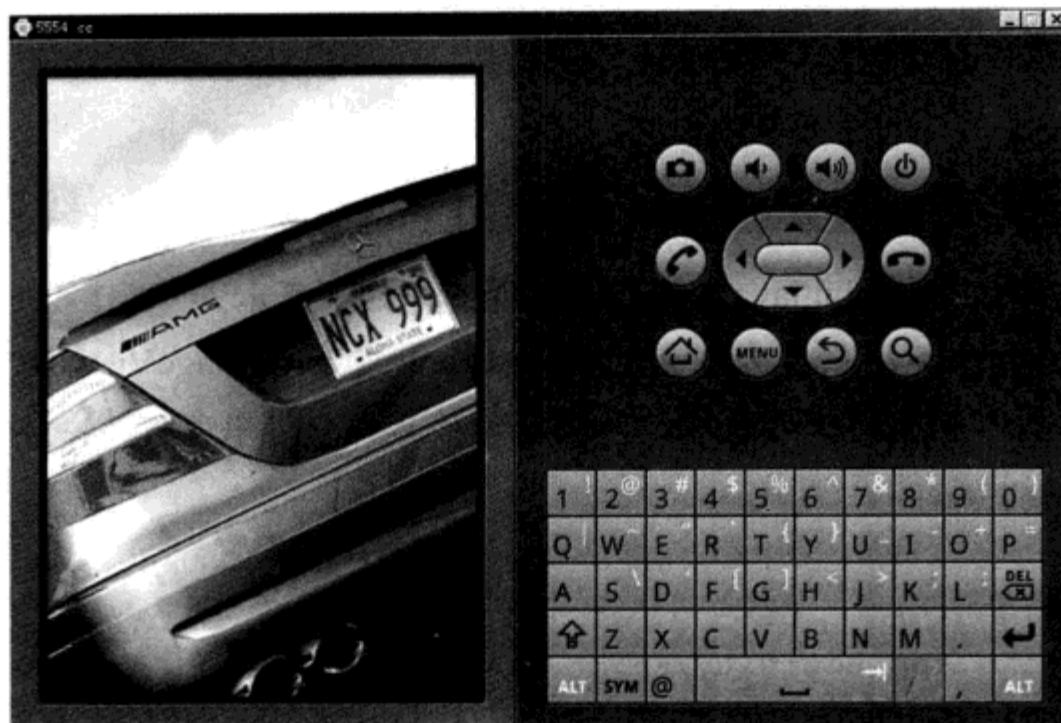


图8-5 执行效果

8.4 在屏幕上触摸移动照片

在触摸屏手机中，触屏移动照片的功能十分常见。在本实例中，设计了一个 ImageView 并使用了 Drawale 的照片，将照片在程序运行的开始将放在屏幕中央。通过 onTouchEvent 来处理点击、拖动、放开等事件来完成拖动图片的功能。设置了 ImageView 的 onClick-Listener 让用户在单击图片的同时，恢复到图片的初始位置。本实例的源码保存在【光盘\daima\第 8 章\chu】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget27"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <ImageView
        android:id="@+id/myImageView1"
        android:layout_width="wrap_content"
```



```
        android:layout_height="wrap_content"
    >
</ImageView>
</AbsoluteLayout>
```

step 02 编写主程序文件 chu.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类，然后分别声明 ImageView 变量 mImageView01，声明相关变量作为存储图片宽高和位置使用，声明存储屏幕的分辨率变量。具体代码如下所示。

```
package irdc.chu;

import irdc.chu.R;
import android.app.Activity;
import android.os.Bundle;
/**/
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.AbsoluteLayout;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

public class chu extends Activity
{
    /*声明ImageView变量*/
    private ImageView mImageView01;
    /*声明相关变量作为存储图片宽高，位置使用*/
    private int intWidth, intHeight, intDefaultX, intDefaultY;
    private float mX, mY;
    /*声明存储屏幕的分辨率变量 */
    private int intScreenX, intScreenY;
```

- ◆ 通过 DisplayMetrics 取得屏幕对象，然后通过 intScreenX 和 intScreenY 取得屏幕解析像素，最后分别设置图片的宽高。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* 取得屏幕对象 */
```



```
DisplayMetrics dm = new DisplayMetrics();
get.WindowManager().getDefaultDisplay().getMetrics(dm);
/* 取得屏幕解析像素 */
intScreenX = dm.widthPixels;
intScreenY = dm.heightPixels;
/* 设置图片的宽高 */
intWidth = 100;
intHeight = 100;
```

- ◆ 通过 `findViewById` 构造器创建 `ImageView` 对象，然后将图片从 `Drawable` 赋值给 `ImageView` 来呈现，并通过 `RestoreButton()` 初始化按钮位置居中。具体代码如下所示。

```
/*通过findViewById构造器创建ImageView对象*/
mImageView01 = (ImageView) findViewById(R.id.myImageView1);
/*将图片从Drawable赋值给ImageView来呈现*/
mImageView01.setImageResource(R.drawable.baby);

/* 初始化按钮位置居中 */
RestoreButton();
```

- ◆ 定义 `setOnTouchListener(new Button.OnTouchListener())`，用于当点击 `ImageView`，还原初始位置。具体代码如下所示。

```
/* 当点击ImageView，还原初始位置 */
mImageView01.setOnTouchListener(new Button.OnTouchListener()
{
    @Override
    public void onClick(View v)
    {
        RestoreButton();
    }
});
```

- ◆ 定义 `onTouchEvent(MotionEvent event)` 覆盖触控事件。首先取得手指触控屏幕的位置，然后实现触控事件的处理，分别实现点击屏幕、移动位置、离开屏幕的动作处理。具体代码如下所示。

```
/*覆盖触控事件*/
public boolean onTouchEvent (MotionEvent event)
{
    /*取得手指触控屏幕的位置*/
    float x = event.getX();
    float y = event.getY();
    try
```



```
    {
        /*触控事件的处理*/
        switch (event.getAction())
        {
            /*点击屏幕*/
            case MotionEvent.ACTION_DOWN:
                picMove(x, y);
                break;
            /*移动位置*/
            case MotionEvent.ACTION_MOVE:
                picMove(x, y);
                break;
            /*离开屏幕*/
            case MotionEvent.ACTION_UP:
                picMove(x, y);
                break;
        }
    }catch (Exception e)
    {
        e.printStackTrace();
    }
    return true;
}
```

- 定义 picMove(float x, float y) 来移动图片，具体代码如下所示。

```
/*移动图片的方法*/
private void picMove (float x, float y)
{
    /*默认微调图片与指针的相对位置*/
    mX=x-(intWidth/2);
    mY=y-(intHeight/2);

    /*防图片超过屏幕的相关处理*/
    /*防止屏幕向右超过屏幕*/
    if ((mX+intWidth)>intScreenX)
    {
        mX = intScreenX-intWidth;
    }
    /*防止屏幕向左超过屏幕*/
    else if (mX<0)
    {
        mX = 0;
    }
}
```

```
/*防止屏幕向下超过屏幕*/
else if ((mY+intHeight)>intScreenY)
{
    mY=intScreenY-intHeight;
}
/*防止屏幕向上超过屏幕*/
else if (mY<0)
{
    mY = 0;
}
/*通过log 来查看图片位置*/
Log.i("jay", Float.toString(mX)+","+Float.toString(mY));
/* 以setLayoutParams方法，重新安排Layout上的位置 */
mImageView01.setLayoutParams
(
    new AbsoluteLayout.LayoutParams
    (intWidth,intHeight,(int) mX,(int)mY)
);
}
```

- ◆ 定义 RestoreButton() 来还原 ImageView 的位置，具体代码如下所示。

```
/* 还原ImageView位置的事件处理 */
public void RestoreButton()
{
    intDefaultX = ((intScreenX-intWidth)/2);
    intDefaultY = ((intScreenY-intHeight)/2);
    /*Toast还原位置坐标*/
    mMakeTextToast
    (
        "("+
        Integer.toString(intDefaultX)+",
        "+
        Integer.toString(intDefaultY)+") ",true
    );
    /* 以setLayoutParams方法，重新安排Layout上的位置 */
    mImageView01.setLayoutParams
    (
        new AbsoluteLayout.LayoutParams
        (intWidth,intHeight,intDefaultX,intDefaultY)
    );
}
```

- ◆ 定义 mMakeTextToast(String str, boolean isLong) 来自定义发出信息，具体代码如下所示。

```
/*自定义一发出信息的方法*/  
public void mMakeTextToast(String str, boolean isLong)  
{  
    if(isLong==true)  
    {  
        Toast.makeText(chu.this, str, Toast.LENGTH_LONG).show();  
    }  
    else  
    {  
        Toast.makeText(chu.this, str, Toast.LENGTH_SHORT).show();  
    }  
}
```

执行后在屏幕上显示一幅图片，并且可以通过鼠标来移动这幅图片，如图 8-6 所示。



图8-6 执行效果

8.5 调节音量

在使用移动手机时，我们经常根据需要来调节手机的音量大小。在 Android API 中，可以使用 AudioManager 中的相关方法来实现音量调节功能，通过这些方法可以在程序中控制手机音量的大小，也可以切换声音的模式为震动或静音。本实例的源码保存在【光盘 \daima\ 第 8 章 \yinliang】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/layout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/myText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_text1"
        android:textSize="16sp"
        android:textColor="@drawable/black"
        android:layout_x="20px"
        android:layout_y="42px"
    >
    </TextView>
    <ImageView
        android:id="@+id/myImage"
        android:layout_width="48px"
        android:layout_height="48px"
        android:layout_x="110px"
        android:layout_y="32px"
    >
    </ImageView>
    <TextView
        android:id="@+id/myText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_text2"
        android:textSize="16sp"
        android:textColor="@drawable/black"
        android:layout_x="20px"
        android:layout_y="102px"
    >
    </TextView>
    <ProgressBar
        android:id="@+id/myProgress"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="160dip"
```



```
    android:layout_height="wrap_content"
    android:max="7"
    android:progress="5"
    android:layout_x="110px"
    android:layout_y="102px"
    >
</ProgressBar>
<ImageButton
    android:id="@+id/downButton"
    android:layout_width="100px"
    android:layout_height="100px"
    android:layout_x="50px"
    android:layout_y="162px"
    android:src="@drawable/down"
    >
</ImageButton>
<ImageButton
    android:id="@+id/upButton"
    android:layout_width="100px"
    android:layout_height="100px"
    android:layout_x="150px"
    android:layout_y="162px"
    android:src="@drawable/up"
    >
</ImageButton>
<ImageButton
    android:id="@+id/normalButton"
    android:layout_width="60px"
    android:layout_height="60px"
    android:layout_x="50px"
    android:layout_y="272px"
    android:src="@drawable/normal"
    >
</ImageButton>
<ImageButton
    android:id="@+id/muteButton"
    android:layout_width="60px"
    android:layout_height="60px"
    android:layout_x="120px"
    android:layout_y="272px"
    android:src="@drawable/mute"
    >
</ImageButton>
```

```

<ImageButton
    android:id="@+id/vibrateButton"
    android:layout_width="60px"
    android:layout_height="60px"
    android:layout_x="190px"
    android:layout_y="272px"
    android:src="@drawable/vibrate"
    >
</ImageButton>
</AbsoluteLayout>

```

step 02 编写主程序文件 yinliang.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类，然后声明各个需要的变量。具体代码如下所示。

```

package irdc.yinliang;

/* import相关class */
import irdc.yinyue.R;
import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;

public class yinliang extends Activity
{
    /* 变量声明 */
    private ImageView myImage;
    private ImageButton downButton;
    private ImageButton upButton;
    private ImageButton normalButton;
    private ImageButton muteButton;
    private ImageButton vibrateButton;
    private ProgressBar myProgress;
    private AudioManager audioMa;
    private int volume=0;
}

```

- ◆ 依次对象初始化变量 audioMa、myImage、myProgress、downButton、upButton、normalButton、muteButton、muteButton 和 vibrateButton。具体代码如下所示。



```
@Override  
public void onCreate(Bundle savedInstanceState)  
{  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    /* 对象初始化 */  
    audioMa = (AudioManager) getSystemService(Context.AUDIO_SERVICE);  
    myImage、 = (ImageView) findViewById(R.id.myImage);  
    myProgress = (ProgressBar) findViewById(R.id.myProgress);  
    downButton = (ImageButton) findViewById(R.id.downButton);  
    upButton = (ImageButton) findViewById(R.id.upButton);  
    normalButton = (ImageButton) findViewById(R.id.normalButton);  
    muteButton = (ImageButton) findViewById(R.id.muteButton);  
    vibrateButton = (ImageButton) findViewById(R.id.vibrateButton);
```

- ◆ 分别设置初始的手机音量和初始的声音模式，具体代码如下所示。

```
/* 设置初始的手机音量 */  
volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);  
myProgress.setProgress(volume);  
/* 设置初始的声音模式 */  
int mode=audioMa.getRingerMode();  
if(mode==AudioManager.RINGER_MODE_NORMAL)  
{  
    myImage.setImageDrawable(getResources()  
        .getDrawable(R.drawable.normal));  
}  
else if(mode==AudioManager.RINGER_MODE_SILENT)  
{  
    myImage.setImageDrawable(getResources()  
        .getDrawable(R.drawable.mute));  
}  
else if(mode==AudioManager.RINGER_MODE_VIBRATE)  
{  
    myImage.setImageDrawable(getResources()  
        .getDrawable(R.drawable.vibrate));  
}
```

- ◆ 设置音量调小按钮 downButton 的处理事件 setOnClickListener，首先设置音量调小声一格，然后设置调整后声音模式。具体代码如下所示。

```
/* 音量调小声的Button */  
downButton.setOnClickListener(new Button.OnClickListener()  
{  
    @Override
```

```

public void onClick(View arg0)
{
    /* 设置音量调小声一格 */
    audioMa.adjustVolume(AudioManager.ADJUST_LOWER, 0);
    volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
    myProgress.setProgress(volume);
    /* 设置调整后声音模式 */
    int mode=audioMa.getRingerMode();
    if (mode==AudioManager.RINGER_MODE_NORMAL)
    {
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.normal));
    }
    else if (mode==AudioManager.RINGER_MODE_SILENT)
    {
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.mute));
    }
    else if (mode==AudioManager.RINGER_MODE_VIBRATE)
    {
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.vibrate));
    }
}
});

```

- ◆ 设置音量调小按钮 upButton 的处理事件 setOnClickListener，先设置音量调大声一格，然后设置调整后声音模式。具体代码如下所示。

```

/* 音量调大声的Button */
upButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        /* 设置音量调大声一格 */
        audioMa.adjustVolume(AudioManager.ADJUST_RAISE, 0);
        volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
        myProgress.setProgress(volume);
        /* 设置调整后的声音模式 */
        int mode=audioMa.getRingerMode();
        if (mode==AudioManager.RINGER_MODE_NORMAL)
        {
            myImage.setImageDrawable(getResources()

```

```
        .getDrawable(R.drawable.normal));
    }
    else if(mode==AudioManager.RINGER_MODE_SILENT)
    {
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.mute));
    }
    else if(mode==AudioManager.RINGER_MODE_VIBRATE)
    {
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.vibrate));
    }
}
});
```

- ◆ 设置调整正常铃声模式按钮 normalButton 的处理事件 setOnClickListener，先设置铃声模式为 NORMAL，然后设置音量与声音模式。具体代码如下所示。

```
/* 调整铃声模式为正常模式的Button */
normalButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        /* 设置铃声模式为NORMAL */
        audioMa.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
        /* 设置音量与声音模式 */
        volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
        myProgress.setProgress(volume);
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.normal));
    }
});
```

- ◆ 设置调整静音铃声模式按钮 muteButton 的处理事件 setOnClickListener，先设置铃声模式为 SILENT，然后设置音量与声音状态。具体代码如下所示。

```
/* 调整铃声模式为静音模式的Button */
muteButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        /* 设置铃声模式为SILENT */
        audioMa.setRingerMode(AudioManager.RINGER_MODE_SILENT);
```



```

/* 设置音量与声音状态 */
volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
myProgress.setProgress(volume);
myImage.setImageDrawable(getResources()
    .getDrawable(R.drawable.mute));
}
});

```

- ◆ 设置调整震动铃声模式按钮 vibrateButton 的处理事件 setOnClickListener，先设置铃声模式为 VIBRATE，然后设置音量与声音状态。具体代码如下所示。

```

/* 调整铃声模式为震动模式的Button */
vibrateButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        /* 设置铃声模式为VIBRATE */
        audioMa.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
        /* 设置音量与声音状态 */
        volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
        myProgress.setProgress(volume);
        myImage.setImageDrawable(getResources()
            .getDrawable(R.drawable.vibrate));
    }
});
}
}

```

执行后将会显示一个音量调节界面。具体如图 8-7 所示。

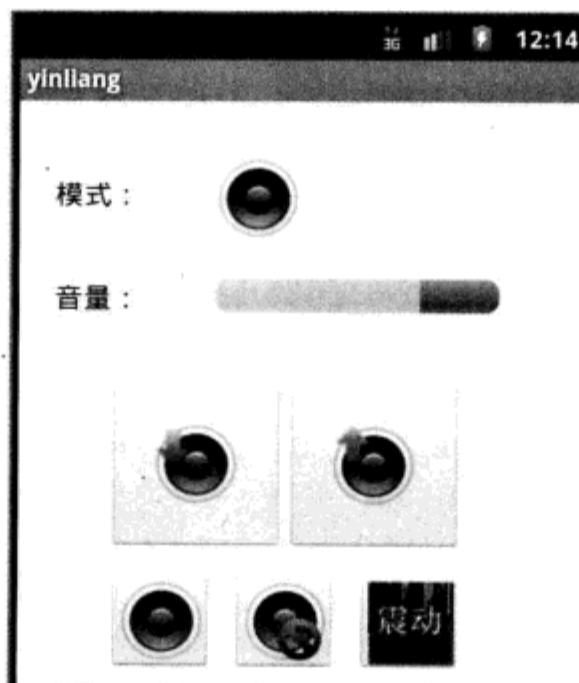


图8-7 执行效果

8.6 播放MP3音乐

在当前的移动手机中，播放 MP3 文件已经是一个十分重要并且十分普通的功能。在本实例中，插入了 3 个按钮，分别用于播放、暂停和停止。当单击播放按钮后会从指定的手机资源中获取 MP3 文件，并执行播放处理。在具体实现上，先添加一个 MediaPlayer 对象，并使用 MediaPlayer.create() 方法来创建播放器资源，然后通过 MediaPlay.create()、MediaPlay.stop() 和 MediaPlay.pause() 分别来实现开始、停止和暂停功能。为了处理按钮所需要的各个事件，需要覆盖各个 ImageButton 的 onClick()，用于通过按钮来控制 MediaPlayer 的状态。本实例的源码保存在【光盘\daima\第 8 章\yinyue】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/widget32"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:textColor="@drawable/black"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
    >
    </TextView>
    <ImageButton
        android:id="@+id/myButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/play"
        android:layout_below="@+id/myTextView1"
    >
    </ImageButton>
    <ImageButton
        android:id="@+id/myButton3"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:src="@drawable/pause"
    android:layout_alignTop="@+id/myButton1"
    android:layout_toRightOf="@+id/myButton1"
  >
</ImageButton>
<ImageButton
    android:id="@+id/myButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/stop"
    android:layout_alignTop="@+id/myButton1"
    android:layout_toRightOf="@+id/myButton3"
  >
</ImageButton>
</RelativeLayout>
```

step 02 编写主程序文件 yinyue.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类，然后分别声明 ImageButton、TextView 和 MediaPlayer 变量，声明一个 Flag 作为确认音乐是否暂停的变量并默认为 false。具体代码如下所示。

```
package irdc.yinyue;

import irdc.yinyue.R;
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class yinyue extends Activity
{
    /*声明一个 ImageButton, TextView, MediaPlayer变量*/
    private ImageButton mButton01, mButton02, mButton03;
    private TextView mTextView01;
    private MediaPlayer mMediaPlayer01;
    /*声明一个Flag作为确认音乐是否暂停的变量并默认为false*/
    private boolean bIsPaused = false;
```

- ◆ 引入主布局文件 main.xml，通过 findViewById 构造器创建 TextView 与 ImageView 对象，然后创建 MediaPlayer 对象，并将音乐以 Import 的方式存储在“res/raw/always.mp3”目录中。具体代码如下所示。



```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*通过findViewById构造器创建TextView与ImageView对象*/
    mButton01 = (ImageButton) findViewById(R.id.myButton1);
    mButton02 = (ImageButton) findViewById(R.id.myButton2);
    mButton03 = (ImageButton) findViewById(R.id.myButton3);
    mTextView01 = (TextView) findViewById(R.id.myTextView1);

    /* onCreate时创建MediaPlayer对象 */
    mMediaPlayer01 = new MediaPlayer();
    /* 将音乐以Import的方式存储在res/raw/always.mp3 */
    mMediaPlayer01 = MediaPlayer.create(yinyue.this, R.raw.big);
```

- 单击按钮开始播放 MP3，先覆盖 OnClick 事件，然后在 MediaPlayer 取得播放资源与 stop() 之后，开始或回复播放，最后改变 TextView 为开始播放状态。具体代码如下所示。

```
/* 运行播放音乐的按钮 */
mButton01.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    /*覆盖onClick事件*/
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        try
        {

            if (mMediaPlayer01 != null)
            {
                mMediaPlayer01.stop();
            }
            /*在MediaPlayer取得播放资源与stop()之后
             * 要准备Playback的状态前一定要使用MediaPlayer.prepare()*/
            mMediaPlayer01.prepare();
            /*开始或回复播放*/
            mMediaPlayer01.start();
            /*改变TextView为开始播放状态*/
            mTextView01.setText(R.string.str_start);
        }
    }
});
```

```

    }
    catch (Exception e)
    {
        // TODO Auto-generated catch block
        mTextView01.setText(e.toString());
        e.printStackTrace();
    }
}
});

```

- ◆ 定义停止播放处理事件，通过 mMediaPlayer01.stop() 停止播放 MP3，改变 TextView 为停止播放状态。具体代码如下所示。

```

/* 停止播放 */
mButton02.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub
        try
        {
            if (mMediaPlayer01 != null)
            {
                /*停止播放*/
                mMediaPlayer01.stop();
                /*改变TextView为停止播放状态*/
                mTextView01.setText(R.string.str_close);
            }
        }
        catch (Exception e)
        {
            // TODO Auto-generated catch block
            mTextView01.setText(e.toString());
            e.printStackTrace();
        }
    }
});

```

- ◆ 定义暂停播放处理事件，首先判断是否处于暂停状态，如果不在暂停状态，先暂停。具体代码如下所示。

```

/* 暂停播放 */
mButton03.setOnClickListener(new ImageButton.OnClickListener()

```



```
{\n    @Override\n    public void onClick(View arg0)\n    {\n        // TODO Auto-generated method stub\n        try\n        {\n            if (mMediaPlayer01 != null)\n            {\n                /*是否为暂停状态=否*/\n                if (bIsPaused==false)\n                {\n                    /*暂停播放*/\n                    mMediaPlayer01.pause();\n                    /*设置Flag为true表示 Player状态为暂停*/\n                    bIsPaused = true;\n                    /*改变TextView为暂停播放*/\n                    mTextView01.setText(R.string.str_pause);\n                }\n                /*是否为暂停状态=是*/\n                else if (bIsPaused==true)\n                {\n                    /*回复播出状态*/\n                    mMediaPlayer01.start();\n                    /*设置Flag为false表示 Player状态为非暂停状态*/\n                    bIsPaused = false;\n                    /*改变TextView为开始播放*/\n                    mTextView01.setText(R.string.str_start);\n                }\n            }\n            catch (Exception e)\n            {\n                // TODO Auto-generated catch block\n                mTextView01.setText(e.toString());\n                e.printStackTrace();\n            }\n        }\n    );\n}
```

- 当播放完毕后会运行 Listener，覆盖文件播出完毕事件，并改变 TextView 为播放结束。具体代码如下所示。

```
/* 当MediaPlayer.OnCompletionListener会运行的Listener */
```

```

mMediaPlayer01.setOnCompletionListener(
    new MediaPlayer.OnCompletionListener()
{
    // @Override
    /*覆盖文件播出完毕事件*/
    public void onCompletion(MediaPlayer arg0)
    {
        try
        {
            /*解除资源与MediaPlayer的赋值关系
             * 让资源可以为其他程序利用*/
            mMediaPlayer01.release();
            /*改变TextView为播放结束*/
            mTextView01.setText(R.string.str_OnCompletionListener);
        }
        catch (Exception e)
        {
            mTextView01.setText(e.toString());
            e.printStackTrace();
        }
    }
});

```

- ◆ 当出错时会运行的 Listener， 覆盖错误处理事件， 当发生错误时也解除资源与 MediaPlayer 的赋值。具体代码如下所示。

```

/* 当MediaPlayer.OnErrorListener会运行的Listener */
mMediaPlayer01.setOnErrorListener(new MediaPlayer.
OnErrorListener()
{
    @Override
    /*覆盖错误处理事件*/
    public boolean onError(MediaPlayer arg0, int arg1, int arg2)
    {
        // TODO Auto-generated method stub
        try
        {
            /*发生错误时也解除资源与MediaPlayer的赋值*/
            mMediaPlayer01.release();
            mTextView01.setText(R.string.str_OnErrorListener);
        }
        catch (Exception e)
        {
            mTextView01.setText(e.toString());
        }
    }
});

```

```
        e.printStackTrace();
    }
    return false;
}
);
}
}
```

- ◆ 覆盖主程序暂停状态事件，在主程序暂停时解除资源与 MediaPlayer 的赋值关系，通过 catch 实现异常处理。具体代码如下所示。

```
protected void onPause()
{
    // TODO Auto-generated method stub
    try
    {
        /*在主程序暂停时解除资源与MediaPlayer的赋值关系*/
        mMediaPlayer01.release();
    }
    catch (Exception e)
    {
        mTextView01.setText(e.toString());
        e.printStackTrace();
    }
    super.onPause();
}
}
```

执行后会在屏幕中通过 3 个播放按钮播放指定的 MP3 文件，如图 8-8 所示。

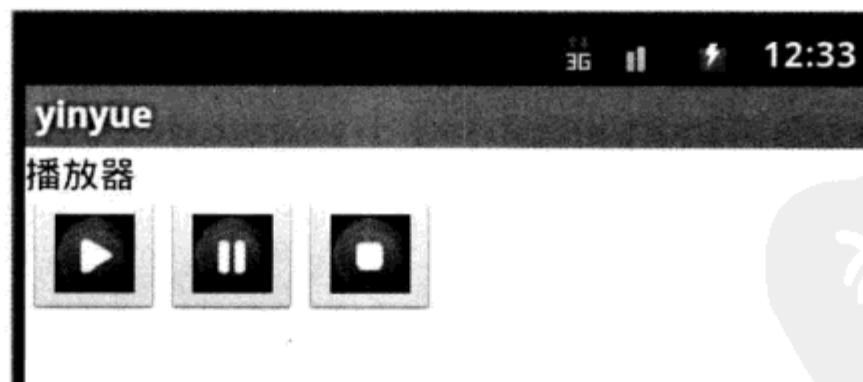


图8-8 执行效果

8.7 开发一个录音机程序

在当前使用的智能移动手机中，录音功能也十分常见。在本实例中，插入了 4 个按钮，分别用于录音、停止录音、播放录音和删除录音。为了能够不限制录音的长度，现将录音

暂时保存到存储卡，当录音完毕后，再将录音文件显示在 ListView。单击文件后，可以播放或删除录音文件。本实例的源码保存在【光盘\daima\第8章\luyin】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/white">
    <LinearLayout
        android:id="@+id/LinearLayout01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <ImageButton
            android:id="@+id/ImageButton01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/record">
        </ImageButton>
        <ImageButton
            android:id="@+id/ImageButton02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/stop">
        </ImageButton>
        <ImageButton
            android:id="@+id/ImageButton03"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/play">
        </ImageButton>
        <ImageButton
            android:id="@+id/ImageButton04"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/delete">
        </ImageButton>
    </LinearLayout>
    <TextView
        android:id="@+id/TextView01"
```



```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@drawable/black">
  </TextView>
  <ListView
    android:id="@+id/ListView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/black">
  </ListView>
</LinearLayout>
```

step 02 编写主程序文件 luyin.java，其具体实现流程如下所示。

- ◆ 引入相关 class，然后分别声明各个需要的变量。具体代码如下所示。

```
package irdc.luyin;

import irdc.luyin.R;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Intent;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.CheckedTextView;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class luyin extends Activity
{
    private ImageButton myButton1;
    private ImageButton myButton2;
    private ImageButton myButton3;
    private ImageButton myButton4;
```

```

private ListView myListview1;
private String strTempFile = "ex07_11_";
private File myRecAudiofile;
private File myRecAudioDir;
private File myPlayFile;
private MediaRecorder mMediaRecorder01;

private ArrayList<String> recordFiles;
private ArrayAdapter<String> adapter;
private TextView myTextview1;
private boolean sdCardExit;
private boolean isStopRecord;

```

- 通过 `findViewById` 分别构造 4 个按钮对象和 2 个文本对象，然后设置按钮状态不可选。具体代码如下所示。

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /* 设置4个按钮和2个文本 */
    myButton1 = (ImageButton) findViewById(R.id.ImageButton01);
    myButton2 = (ImageButton) findViewById(R.id.ImageButton02);
    myButton3 = (ImageButton) findViewById(R.id.ImageButton03);
    myButton4 = (ImageButton) findViewById(R.id.ImageButton04);
    myListview1 = (ListView) findViewById(R.id.ListView01);
    myTextview1 = (TextView) findViewById(R.id.TextView01);
    /* 设置按钮状态不可选 */
    myButton2.setEnabled(false);
    myButton3.setEnabled(false);
    myButton4.setEnabled(false);

```

- 通过 `sdCardExit` 判断插入 SD 卡，然后获取 SD Card 路径作为录音的文件位置，并取得 SD 卡目录里的所有 `.amr` 文件，最后将 `ArrayAdapter` 添加 `ListView` 对象中。具体代码如下所示。

```

/* 判断SD Card是否插入 */
sdCardExit = Environment.getExternalStorageState().equals(
    android.os.Environment.MEDIA_MOUNTED);
/* 取得SD Card路径作为录音的文件位置 */
if (sdCardExit)
    myRecAudioDir = Environment.getExternalStorageDirectory();

```

```
/* 取得SD Card目录里的所有.amr文件 */
getRecordFiles();
adapter = new ArrayAdapter<String>(this,
    R.layout.my_simple_list_item, recordFiles);
/* 将ArrayAdapter添加ListView对象中 */
myListView1.setAdapter(adapter);
```

- ◆ 设置单击录音按钮后的录音处理事件，先创建录音音频文件，然后设置录音来源为麦克风，最后通过 myTextView1.setText("录音中") 设置录音过程显示的提示文本。具体代码如下所示。

```
/* 录音 */
myButton1.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        try
        {
            if (!sdCardExit)
            {
                Toast.makeText(luyin.this, "请插入SD Card",
                    Toast.LENGTH_LONG).show();
                return;
            }
            /* 创建录音音频文件 */
            myRecAudioFile = File.createTempFile(strTempFile, ".amr",
                myRecAudioDir);
            mMediaRecorder01 = new MediaRecorder();
            /* 设置录音来源为麦克风 */
            mMediaRecorder01
                .set AudioSource(MediaRecorder.AudioSource.MIC);
            mMediaRecorder01
                .setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
            mMediaRecorder01
                .set AudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
            mMediaRecorder01.setOutputFile(myRecAudioFile
                .getAbsolutePath());
            mMediaRecorder01.prepare();
            mMediaRecorder01.start();
            myTextView1.setText("录音中");
            myButton2.setEnabled(true);
            myButton3.setEnabled(false);
        }
    }
});
```

```

        myButton4.setEnabled(false);
        isStopRecord = false;
    }
    catch (IOException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
});

```

- ◆ 设置单击停止按钮后的处理事件，先通过 mMediaRecorder01.stop() 停止录音，然后将录音文件名给 Adapter。具体代码如下所示。

```

/* 停止 */
myButton2.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub
        if (myRecAudioFile != null)
        {
            /* 停止录音 */
            mMediaRecorder01.stop();
            mMediaRecorder01.release();
            mMediaRecorder01 = null;
            /* 将录音文件名给Adapter */
            adapter.add(myRecAudioFile.getName());
            myTextView1.setText("停止：" + myRecAudioFile.getName());
            myButton2.setEnabled(false);
            isStopRecord = true;
        }
    }
});

```

- ◆ 置单击播放按钮后的处理事件，单击后将打开播放的程序。具体代码如下所示。

```

/* 播放 */
myButton3.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub

```

```
    if (myPlayFile != null && myPlayFile.exists())
    {
        /* 打开播放的程序 */
        openFile(myPlayFile);
    }
}
});
```

- ◆ 设置单击删除按钮后的处理事件，先将 Adapter 删除文件名，然后删除存在的文件。具体代码如下所示。

```
/* 删除 */
myButton4.setOnClickListener(new ImageButton.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub
        if (myPlayFile != null)
        {
            /* 先将Adapter删除文件名 */
            adapter.remove(myPlayFile.getName());
            /* 删除文件 */
            if (myPlayFile.exists())
                myPlayFile.delete();
            myTextView1.setText("完成删除");
        }
    }
});
```

- ◆ 设置单击列表中文件的处理事件，当有文件单击后将删除及播放按钮 Enable，然后输出选择提示语句。具体代码如下所示。

```
myListView1.setOnItemClickListener
(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1,
            int arg2, long arg3)
    {
        /* 当有点击档名时将删除及播放按钮Enable */
        myButton3.setEnabled(true);
        myButton4.setEnabled(true);
        myPlayFile = new File(myRecAudioDir.getAbsolutePath())
```

```

        + File.separator
        + ((CheckedTextView) arg1).getText());
myTextView1.setText("你选的是: "
        + ((CheckedTextView) arg1).getText());
    }
});
}
}

```

- ◆ 使用方法 `onStop()` 来停止录音处理，具体代码如下所示。

```

@Override
protected void onStop()
{
    if (mMediaRecorder01 != null && !isStopRecord)
    {
        /* 停止录音 */
        mMediaRecorder01.stop();
        mMediaRecorder01.release();
        mMediaRecorder01 = null;
    }
    super.onStop();
}

```

- ◆ 定义方法 `getRecordFiles()` 来获取文件的长度，并设置只获取“.amr”格式的文件。具体代码如下所示。

```

private void getRecordFiles()
{
    recordFiles = new ArrayList<String>();
    if (sdCardExit)
    {
        File files[] = myRecAudioDir.listFiles();
        if (files != null)
        {
            for (int i = 0; i < files.length; i++)
            {
                if (files[i].getName().indexOf(".") >= 0)
                {
                    /* 只取.amr文件 */
                    String fileS = files[i].getName().substring(
                        files[i].getName().indexOf("."));
                    if (fileS.toLowerCase().equals(".amr"))
                        recordFiles.add(files[i].getName());
                }
            }
        }
    }
}

```

```
        }
    }
}
```

- ◆ 定义方法 openFile(File f) 来打开播放指定的录音文件，具体代码如下所示。

```
/* 打开播放录音文件的程序 */
private void openFile(File f)
{
    Intent intent = new Intent();
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.setAction(android.content.Intent.ACTION_VIEW);
    String type = getMIMEType(f);
    intent.setDataAndType(Uri.fromFile(f), type);
    startActivity(intent);
}
```

- ◆ 定义方法 getMIMEType(File f) 来获取文件的类型，在此设置了 audio 类型、image 类型和其他类型，共三大类。具体代码如下所示。

```
private String getMIMEType(File f)
{
    String end = f.getName().substring(
        f.getName().lastIndexOf(".") + 1, f.getName().length())
        .toLowerCase();
    String type = "";
    if (end.equals("mp3") || end.equals("aac") || end.equals("aac"))
        || end.equals("amr") || end.equals("mpeg")
        || end.equals("mp4"))
    {
        type = "audio";
    }
    else if (end.equals("jpg") || end.equals("gif")
        || end.equals("png") || end.equals("jpeg"))
    {
        type = "image";
    }
    else
    {
        type = "*";
    }
    type += "/*";
    return type;
}
```



step 03 在文件 AndroidManifest.xml 中声明打开录音权限，具体代码如下所示。

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

执行后的效果如图 8-9 所示。当单击【录音】按钮时开始录音处理，如图 8-10 所示。当单击【停止】按钮后停止录音处理，并在列表中显示录制的音频文件，如图 8-11 所示。当选中音频文件，单击“播放”按钮后会播放选中的音频文件，如图 8-12 所示。



图8-9 初始效果



图8-10 正在录音



图8-11 显示录制的文件



图8-12 播放音频

当选中音频文件，单击【删除】按钮后会删除选中的音频文件。

8.8 开发一个拍照程序

在当前移动手机中，拍照也是一个十分重要的功能。在 Android 相机系统中，使用 Preview 功能能够实现预览处理。在 Preview 的 API Demo 中，没有示范如何在“Activity”中配置 Preview 的讲解，也没有提到如何获取 Preview 中画面的方法。本实例实现了一个简单的拍照功能，和 API Demo 的 Preview 程序不同。实例中将以 Activity 为基础，在 Layout 中配置了 3 个按钮，分别实现打开预览、关闭相机和拍照处理这 3 个功能。带单击“拍照”按钮后，常年供需将画面截取下来，并存储到 SD 卡中，并将拍下来的图片显示在 Activity 中的 ImageView 中。为避免拍照相片造成的存储卡垃圾暂存堆栈，在离开程序



前将临时文件删除。

本实例的源码保存在【光盘\daima\第8章\paizao】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:text="@string/hello"
    />
    <SurfaceView
        android:id="@+id/mSurfaceView1"
        android:visibility="visible"
        android:layout_width="320px"
        android:layout_height="240px">
    </SurfaceView>
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    >
        <Button
            android:id="@+id/myButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/str_button1"/>
        <Button
            android:id="@+id/myButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/str_button2"/>
        <Button
            android:id="@+id/myButton3"
            android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="@string/str_take_picture"/>
</LinearLayout>
<ImageView
    android:id="@+id/myImageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center" />
</LinearLayout>
```

step 02 编写主程序文件 paizao.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类，当拍照成功后引用 PictureCallback 作为拍照后的获取事件。具体代码如下所示。

```
package irdc.paizao;

import irdc.paizao.R;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.PixelFormat;

/* 引用Camera类 */
import android.hardware.Camera;

/* 引用PictureCallback作为取得拍照后的事件 */
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.ShutterCallback;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
```



- ◆ 创建私有 Camera 对象，然后分别创建 mImageView01、mTextView01、TAG、mSurfaceView01、mSurfaceHolder01，作为 review 照下来的相片之用。具体代码如下所示。

```
/* 使Activity实现SurfaceHolder.Callback */
public class paizao extends Activity
implements SurfaceHolder.Callback
{
    /* 创建私有Camera对象 */
    private Camera mCamera01;
    private Button mButton01, mButton02, mButton03;

    /* 作为review照下来的相片之用 */
    private ImageView mImageView01;
    private TextView mTextView01;
    private String TAG = "HIPPO";
    private SurfaceView mSurfaceView01;
    private SurfaceHolder mSurfaceHolder01;
```

- ◆ 默认相机预览模式为 false，将照下来的图片存储在 "/sdcard/camera_snap.jpg"。具体代码如下所示。

```
/* 默认相机预览模式为false */
private boolean bIfPreview = false;

/* 将照下来的图片存储在此 */
private String strCaptureFilePath = "/sdcard/camera_snap.jpg";
```

- ◆ 通过 requestWindowFeature(Window.FEATURE_NO_TITLE)，设置程序全屏幕运行，而不使用 title bar。然后判断存储卡是否存在，并提醒用户未安装存储卡。具体代码如下所示。

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    /* 使应用程序全屏幕运行，不使用title bar */
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.main);

    /* 判断存储卡是否存在 */
    if(!checkSDCard())
    {
```

```
/* 提醒User未安装存储卡 */
mMakeTextToast
(
    getResources().getText(R.string.str_err_nosd).toString(),
    true
);
}
```

- ◆ 通过 DisplayMetrics 对象 dm 取得屏幕解析像素，然后以 SurfaceView 作为相机 Preview 之用，并绑定 SurfaceView，取得 SurfaceHolder 对象，最后通过 setFixedSize 额外设置预览大小。具体代码如下所示。

```
/* 取得屏幕解析像素 */
DisplayMetrics dm = new DisplayMetrics();
get WindowManager().getDefaultDisplay().getMetrics(dm);
//intScreenX = dm.widthPixels;
//intScreenY = dm.heightPixels;
//Log.i(TAG, Integer.toString(intScreenX));

mTextView01 = (TextView) findViewById(R.id.myTextView1);
mImageView01 = (ImageView) findViewById(R.id.myImageView1);

/* 以SurfaceView作为相机Preview之用 */
mSurfaceView01 = (SurfaceView) findViewById(R.id.mSurfaceView1);

/* 绑定SurfaceView，取得SurfaceHolder对象 */
mSurfaceHolder01 = mSurfaceView01.getHolder();

/* Activity必须实现SurfaceHolder.Callback */
mSurfaceHolder01.addCallback(paizao.this);

/* 额外的设置预览大小设置，在此不使用 */
//mSurfaceHolder01.setFixedSize(320, 240);

/*
 * 以SURFACE_TYPE_PUSH_BUFFERS(3)
 * 作为SurfaceHolder显示类型
 */
mSurfaceHolder01.setType
(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

mButton01 = (Button) findViewById(R.id.myButton1);
mButton02 = (Button) findViewById(R.id.myButton2);
mButton03 = (Button) findViewById(R.id.myButton3);
```

- ◆ 设置打开相机及 Preview 按钮事件，自定义初始化打开相机函数。具体代码如下所示。

```
/* 打开相机及Preview */
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub

        /* 自定义初始化打开相机函数 */
        initCamera();
    }
});
```

- ◆ 设置停止 Preview 及相机按钮事件，自定义重置相机，并关闭相机预览函数。具体代码如下所示。

```
/* 停止Preview及相机 */
mButton02.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub

        /* 自定义重置相机，并关闭相机预览函数 */
        resetCamera();
    }
});
```

- ◆ 设置停止拍照按钮事件，当存储卡存在才允许拍照，存储暂存图像文件，并自定义拍照函数 takePicture()。具体代码如下所示。

```
/* 拍照 */
mButton03.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub

        /* 当存储卡存在才允许拍照，存储暂存图像文件 */
        if(checkSDCard())
        {
```

```
    /* 自定义拍照函数 */
    takePicture();
}
else
{
    /* 存储卡不存在显示提示 */
    mTextView01.setText
    (
        getResources().getText(R.string.str_err_nosd).toString()
    );
}
});
}
```

- ◆ 定义 initCamera() 方法，如果相机非在预览模式则打开相机。具体代码如下所示。

```
/* 自定义初始相机函数 */
private void initCamera()
{
    if (!bIfPreview)
    {
        /* 若相机非在预览模式，则打开相机 */
        mCamera01 = Camera.open();
    }

    if (mCamera01 != null && !bIfPreview)
    {
        Log.i(TAG, "inside the camera");

        /* 创建Camera.Parameters对象 */
        Camera.Parameters parameters = mCamera01.getParameters();

        /* 设置相片格式为JPEG */
        parameters.setPictureFormat(PixelFormat.JPEG);

        /* 指定preview的屏幕大小 */
        parameters.setPreviewSize(320, 240);

        /* 设置图片分辨率大小 */
        parameters.setPictureSize(320, 240);

        /* 将Camera.Parameters设置予Camera */
        mCamera01.setParameters(parameters);
    }
}
```

```
/* setPreviewDisplay唯一的参数为SurfaceHolder */
mCamera01.setPreviewDisplay(mSurfaceHolder01);

/* 立即运行Preview */
mCamera01.startPreview();
bIfPreview = true;
}
```

- ◆ 定义 takePicture() 方法来调用 takePicture() 方法拍照并采集图像，具体代码如下所示。

```
/* 拍照采集图像 */
private void takePicture()
{
    if (mCamera01 != null && bIfPreview)
    {
        /* 调用takePicture()方法拍照 */
        mCamera01.takePicture
        (shutterCallback, rawCallback, jpegCallback);
    }
}
```

- ◆ 定义 resetCamera() 方法实现相机重置，具体代码如下所示。

```
/* 相机重置 */
private void resetCamera()
{
    if (mCamera01 != null && bIfPreview)
    {
        mCamera01.stopPreview();
        /* 扩展学习，释放Camera对象 */
        //mCamera01.release();
        mCamera01 = null;
        bIfPreview = false;
    }
}

private ShutterCallback shutterCallback = new ShutterCallback()
{
    public void onShutter()
    {
        // Shutter has closed
    }
};
```

```

private PictureCallback rawCallback = new PictureCallback()
{
    public void onPictureTaken(byte[] _data, Camera _camera)
    {
        // TODO Handle RAW image data
    }
};

```

- 通过 onPictureTaken 传入的第一个参数即为相片的 byte，并用 new File(strCaptureFilePath) 创建新文件，并采用压缩转档方法压缩图片，然后调用 flush() 方法更新 BufferStream，最后将拍照下来且存储完毕的图文件显示出来。具体代码如下所示。

```

private PictureCallback jpegCallback = new PictureCallback()
{
    public void onPictureTaken(byte[] _data, Camera _camera)
    {
        /* onPictureTaken传入的第一个参数即为相片的byte */
        Bitmap bm = BitmapFactory.decodeByteArray
            (_data, 0, _data.length);
        /* 创建新文件 */
        File myCaptureFile = new File(strCaptureFilePath);
        try
        {
            BufferedOutputStream bos = new BufferedOutputStream
                (new FileOutputStream(myCaptureFile));
            /* 采用压缩转档方法 */
            bm.compress(Bitmap.CompressFormat.JPEG, 80, bos);
            /* 调用flush()方法，更新BufferStream */
            bos.flush();
            /* 结束OutputStream */
            bos.close();
            /* 将拍照下来且存储完毕的图文件，显示出来 */
            mImageView01.setImageBitmap(bm);
            /* 显示完图文件，立即重置相机，并关闭预览 */
            resetCamera();
            /* 再重新启动相机继续预览 */
            initCamera();
        }
        catch (Exception e)
        {
            Log.e(TAG, e.getMessage());
        }
    }
};

```



- ◆ 定义方法 delFile(String strFileName) 自定义删除临时文件，具体代码如下所示。

```
/* 自定义删除文件函数 */
private void delFile(String strFileName)
{
    try
    {
        File myFile = new File(strFileName);
        if(myFile.exists())
        {
            myFile.delete();
        }
    }
    catch (Exception e)
    {
        Log.e(TAG, e.toString());
        e.printStackTrace();
    }
}
```

- ◆ 定义方法 mMakeTextToast(String str, boolean isLong) 来输出提示语句，具体代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)
{
    if(isLong==true)
    {
        Toast.makeText(paizao.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
        Toast.makeText(paizao.this, str, Toast.LENGTH_SHORT).show();
    }
}
```

- ◆ 定义方法 checkSDCard() 来检查是否有存储卡，具体代码如下所示。

```
private boolean checkSDCard()
{
    /* 判断存储卡是否存在 */
    if(android.os.Environment.getExternalStorageState().equals(
        android.os.Environment.MEDIA_MOUNTED))
    {
        return true;
    }
}
```

```

    else
    {
        return false;
    }
}

@Override
public void surfaceChanged
(SurfaceHolder surfaceholder, int format, int w, int h)
{
    Log.i(TAG, "Surface Changed");
}

@Override
public void surfaceCreated(SurfaceHolder surfaceholder)
{
    // TODO Auto-generated method stub
    Log.i(TAG, "Surface Created");
}

```

- ◆ 当 Surface 不存在时需要删除图片，具体代码如下所示。

```

@Override
public void surfaceDestroyed(SurfaceHolder surfaceholder)
{
    /* 当Surface不存在，需要删除图片 */
    try
    {
        delFile(strCaptureFilePath);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    Log.i(TAG, "Surface Destroyed");
}

```

- (3) 在文件 AndroidManifest.xml 中声明 android.permission.CAMERA 权限，具体代码如下所示。

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

执行后的效果如图 8-13 所示。单击【预览】、【拍照】和【关闭】按钮后，能够实现对应的功能。

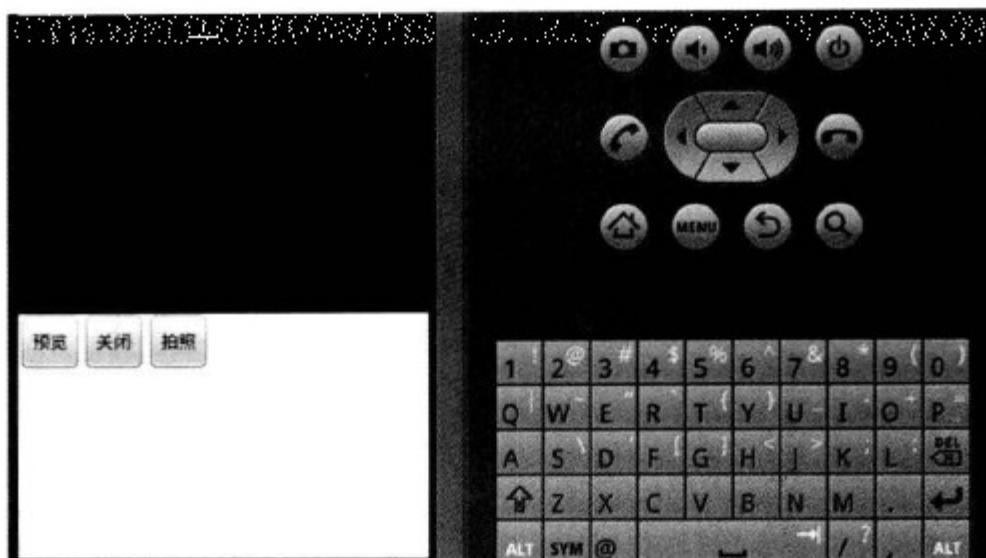


图8-13 执行效果

8.9 开发一个视频播放器

在当前的移动手机中，视频播放也是一个十分重要的功能。在 Android 系统中，内置了 VideoView Widget 作为多媒体视频播放器，这样可以浏览电影视频。VideoView Widget 和前面介绍的 Widget 私用方法类似，必须先在 Layout XML 中定义 VideoView 属性，在程序中通过 findViewById() 方法即可创建 VideoView 对象。

在本实例中，预先准备了 2 个 .3gp 格式的视频文件，上传到了虚拟 SD 卡中。然后插入 2 个按钮，当单击按钮后分别实现对这 2 个视频文件的播放。本实例的源码保存在【光盘\daima\第 8 章\shipin】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:text="@string/hello"
    />
    <VideoView
        android:id="@+id/myVideoView1"
        android:layout_width="320px"
```

```

        android:layout_height="240px"
    />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1">
    <Button
        android:id="@+id/myButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_button1" />
    <Button
        android:id="@+id/myButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_button2" />

```

```
</LinearLayout>
</LinearLayout>
```

step 02 编写主程序文件 shipin.java，其具体实现流程如下所示。

- ◆ 引入相关 class 类，然后分别声明变量 mTextView01、mVideoView01、strVideoPath、mButton01、mButton02 和 TAG。具体代码如下所示。

```

package irdc.shipin;

import irdc.shipin.R;
import android.app.Activity;
import android.graphics.PixelFormat;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.MediaController;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.VideoView;

public class shipin extends Activity
{
    private TextView mTextView01;
    private VideoView mVideoView01;
}

```



```
private String strVideoPath = "";
private Button mButton01, mButton02;
private String TAG = "HIPPO_VIDEOVIEW";
```

- ◆ 设置默认判别是否安装存储卡 flag 为 false，然后设置全屏幕显示。具体代码如下所示。

```
/* 默认判别是否安装存储卡flag为false */
private boolean bIfSDExist = false;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    /* 全屏幕 */
    getWindow().setFormat(PixelFormat.TRANSLUCENT);
    setContentView(R.layout.main);
```

- ◆ 判断存储卡是否存在，如果不存在则通过 mMakeTextToast 输出提示语句。具体代码如下所示。

```
/* 判断存储卡是否存在 */
if(android.os.Environment.getExternalStorageState().equals
    (android.os.Environment.MEDIA_MOUNTED))
{
    bIfSDExist = true;
}
else
{
    bIfSDExist = false;
    mMakeTextToast
    (
        getResources().getText(R.string.str_err_nosd).toString(),
        true
    );
}
```

- ◆ 分别获取 TextView、EditText 的值，然后通过 findViewById 构造 2 个对象。具体代码如下所示。

```
mTextView01 = (TextView)findViewById(R.id.myTextView1);
mVideoView01 = (VideoView)findViewById(R.id.myVideoView1);

mButton01 = (Button)findViewById(R.id.myButton1);
```



```
mButton02 = (Button) findViewById(R.id.myButton2);
```

- ◆ 定义按钮单击处理事件，通过 playVideo(strVideoPath) 播放第一个视频 A。具体代码如下所示。

```
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub
        if(bIfSDExist)
        {
            /* 播放影片路径1 */
            strVideoPath = "file:///sdcard/hello.3gp";
            playVideo(strVideoPath);
        }
    }
});
```

- ◆ 定义按钮单击处理事件，通过 playVideo(strVideoPath) 播放第二个视频 B。具体代码如下所示。

```
mButton02.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        // TODO Auto-generated method stub
        if(bIfSDExist)
        {
            /* 播放影片路径2 */
            strVideoPath = "file:///sdcard/test.3gp";
            playVideo(strVideoPath);
        }
    }
});
```

- ◆ 自定义 VideoView 方法来播放指定路径的影片，具体代码如下所示。

```
/* 自定义以VideoView播放影片 */
private void playVideo(String strPath)
{
    if(strPath!="")
    {
        /* 调用VideoURI方法，指定解析路径 */
    }
}
```

```
mVideoView01.setVideoURI(Uri.parse(strPath));  
  
/* 设置控制Bar显示于此Context中 */  
mVideoView01.setMediaController  
(new MediaController(shipin.this));  
  
mVideoView01.requestFocus();  
  
/* 调用VideoView.start()自动播放 */  
mVideoView01.start();  
if(mVideoView01.isPlaying())  
{  
    /* 下程序不会被运行，因start()后尚需要preparing() */  
    mTextView01.setText("Now Playing:"+strPath);  
    Log.i(TAG, strPath);  
}  
}  
}  
}
```

- ◆ 定义 mMakeTextToast(String str, boolean isLong) 来输出提醒语句，具体代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)  
{  
    if(isLong==true)  
    {  
        Toast.makeText(shipin.this, str, Toast.LENGTH_LONG).show();  
    }  
    else  
    {  
        Toast.makeText(shipin.this, str, Toast.LENGTH_SHORT).show();  
    }  
}
```

执行后的效果如图 8-14 所示。当单击“播放影片 A”和“播放影片 B”按钮后，分别播放预设的影片。



图8-14 执行效果

第 9 章



Google地图实战演练

在当前智能手机系统应用中，地图导航已经成为了必不可少的功能之一。我们永远不会忘记谷歌地图给我的生活带来的巨大方便，作为谷歌旗下的产品——安卓，当然具备了谷歌地图的所有优秀功能。本章将通过几个典型实例的实现过程，详细介绍 Android 系统中使用 Google 地图服务的基本知识。





9.1 获取当前位置的坐标

Android 支持 GPS 和网络地图，通常将各种不同的定位技术成为 LBS。LBS 是基于位置的服务 (Location Based Service) 的简称，它是通过电信移动运营商的无线电通信网络（如 GSM 网、CDMA 网）或外部定位方式（如 GPS）获取移动终端用户的位置信息（地理坐标，或大地坐标），在 GIS (Geographic Information System, 地理信息系统) 平台的支持下，为用户提供相应服务的一种增值业务。在本实例中，使用 GPS 定位技术获取了当前位置的坐标信息。

本实例的源码保存在【光盘\daima\第 9 章\dang】，具体实现流程如下所示。

step 01 编写文件 AndroidManifest.xml，在里面声明 ACCESS_FINE_LOCATION 权限，具体代码如下所示。

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

step 02 编写主文件 main.xml 来布局屏幕界面，具体代码如下所示。

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

step 03 编写文件 dang.java，在 onCreate(Bundle savedInstanceState) 中获取当前位置信息，主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    LocationManager locationManager;
    String serviceName = Context.LOCATION_SERVICE;
    locationManager = (LocationManager) getSystemService(serviceName);
```

```

        Criteria criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setAltitudeRequired(false);
        criteria.setBearingRequired(false);
        criteria.setCostAllowed(true);
        criteria.setPowerRequirement(Criteria.POWER_LOW);
        String provider = locationManager.getBestProvider(criteria, true);
        Location location = locationManager.getLastKnownLocation(provider);
        updateWithNewLocation(location);
        /*每隔1000ms更新一次，并且不考虑位置的变化。*/
        locationManager.requestLocationUpdates(provider, 2000, 10,
            locationListener);
    }
}

```

在上述代码中，使用 LocationManager 周期获得当前设备的一个类。要获取 LocationManager 实例，需要调用方法 Context.getSystemService 并传入服务名 LOCATION_SERVICE("location")。在创建 LocationManager 实例后可以调用 getLastKnownLocation 方法将上一次 LocationManager 获得有效位置信息以 Location 对象的形式返回。

step 04 定义方法 updateWithNewLocation(Location location) 来更新显示用户界面，主要代码如下所示。

```

private void updateWithNewLocation(Location location) {
    String latLongString;
    TextView myLocationText;
    myLocationText = (TextView)findViewById(R.id.myLocationText);
    if (location != null) {
        double lat = location.getLatitude();
        double lng = location.getLongitude();
        latLongString = "纬度：" + lat + "\n经度：" + lng;
    } else {
        latLongString = "获取地理信息失败";
    }
    myLocationText.setText("当前坐标位置是:\n" +
        latLongString);
}

```

step 05 定义 LocationListener 对象来监听坐标改变时事件，如果 Provider 传进相同的坐标，它就不会被触发。主要代码如下所示。

```

private final LocationListener locationListener = new
LocationListener() {
    public void onLocationChanged(Location location) {
        updateWithNewLocation(location);
    }
}

```

```
}

public void onProviderDisabled(String provider) {
    updateWithNewLocation(null);
}

public void onProviderEnabled(String provider){ }

public void onStatusChanged(String provider, int status,
    Bundle extras){ }

};
```

因为模拟器上没有 GPS 设备，所以需要在 Eclipse 的 DDMS 工具中提供模拟的 GPS 数据。即依次单击“DDMS” | “Emulator Control”，在弹出对话框中找到“Location Control”选项，在此输入坐标，完成后单击【Send】按钮，如图 9-1 所示。

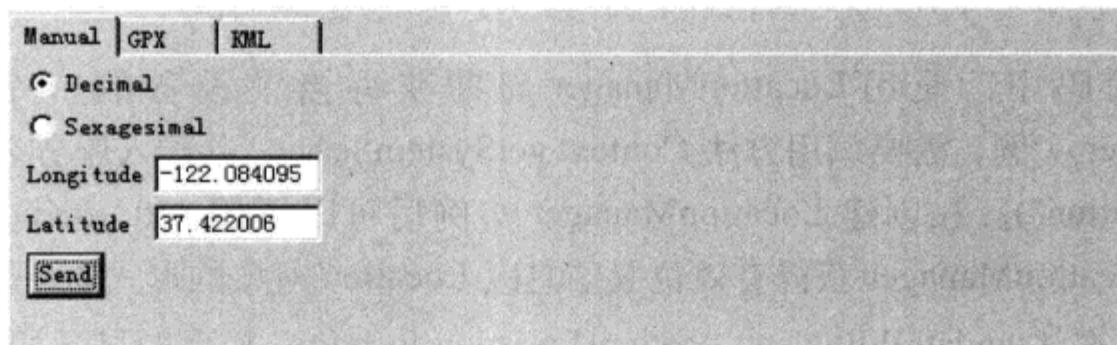


图9-1 设置坐标

因为用到了 Google API，所以在项目中引入 Google API，邮件单击项目选择“Properties”，在弹出对话框中选择 Google API 版本，如图 9-2 所示。

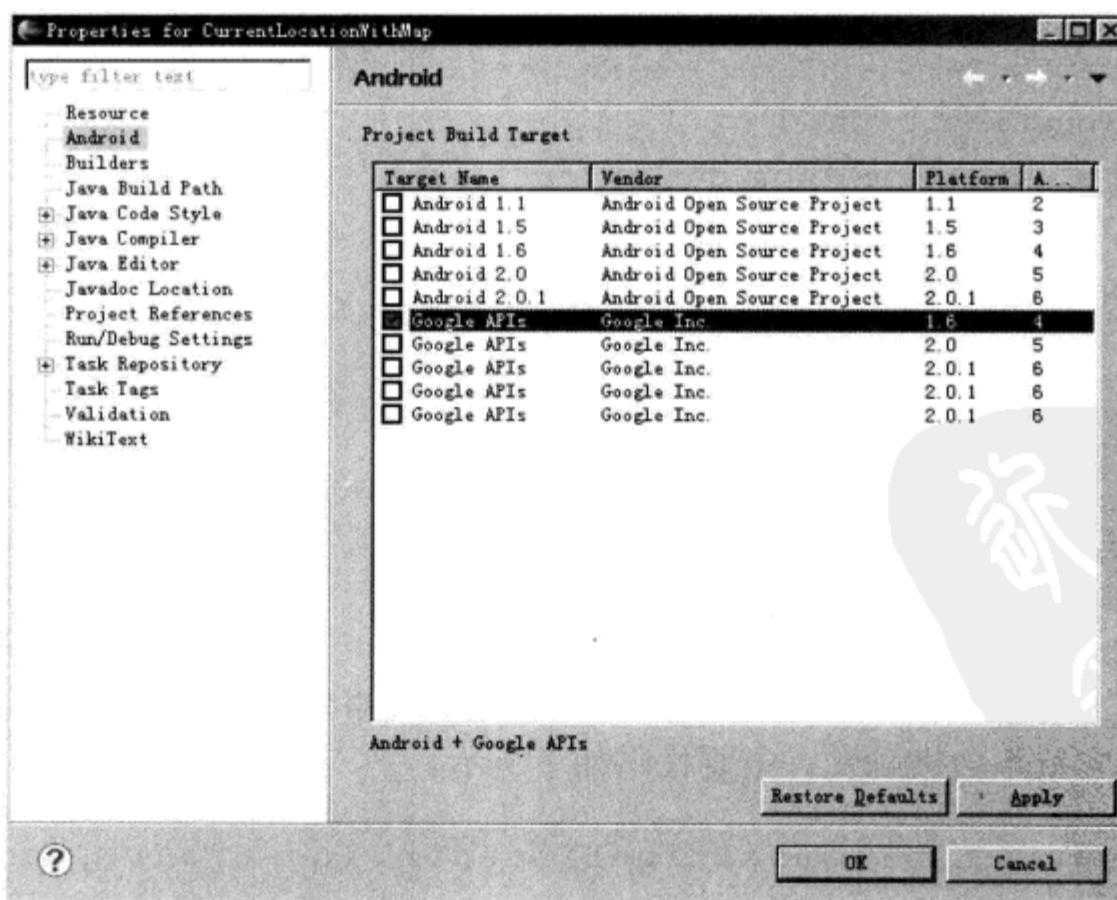


图9-2 引用Google API

此时模拟器运行后会显示当前的坐标，如图 9-3 所示。



当前坐标位置是：
纬度：37.422005
经度：-122.084095

图9-3 执行效果

9.2 在手机中使用谷歌地图

在本实例中，我们使用 Map API 密钥在手机屏幕中显示谷歌地图。Google 地图给人们的生活带来了极大的方便，例如，可以通过 Google 地图查找商户信息、查看地图和获取行车路线等。Android 平台也提供了一个 map 包 (com.google.android.maps)，通过其中的MapView 就能够方便地利用 Google 地图的资源来进行编程。

在使用谷歌地图之前需要预先进行如下设置。

(1) 添加 maps.jar 到项目

在 Android SDK 中，以 JAR 库的形式提供了和 MAP 有关的 API，此 JAR 库位与“android-sdk-windows\add-ons\google_apis-4”目录下。要把 maps.jar 添加到项目中，可以在项目属性中的“Android”栏中指定使用包含 Google API 的 Target 作为项目的构建目标，如图 9-4 所示。

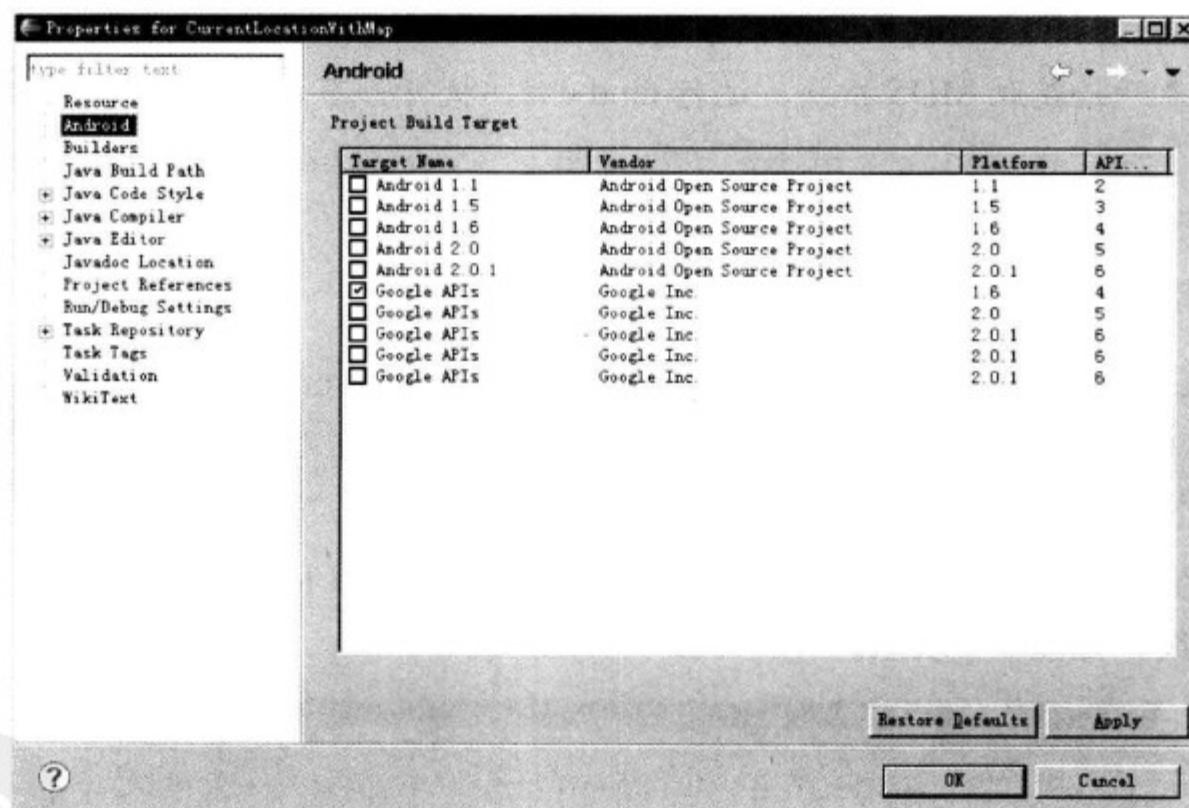


图9-4 在项目中包含Google API

(2) 将地图嵌入到应用

通过使用 MapActivity 和 MapView 控件，可以轻松地将地图嵌入到应用程序当中。在此步骤中，需要将 Google API 添加到构建路径中。方法是在图 9-4 所示界面中选择“Java Build Path”，然后在 Target 中选择选中 Google API，设置项目中包含 Google API，如图 9-5 所示。

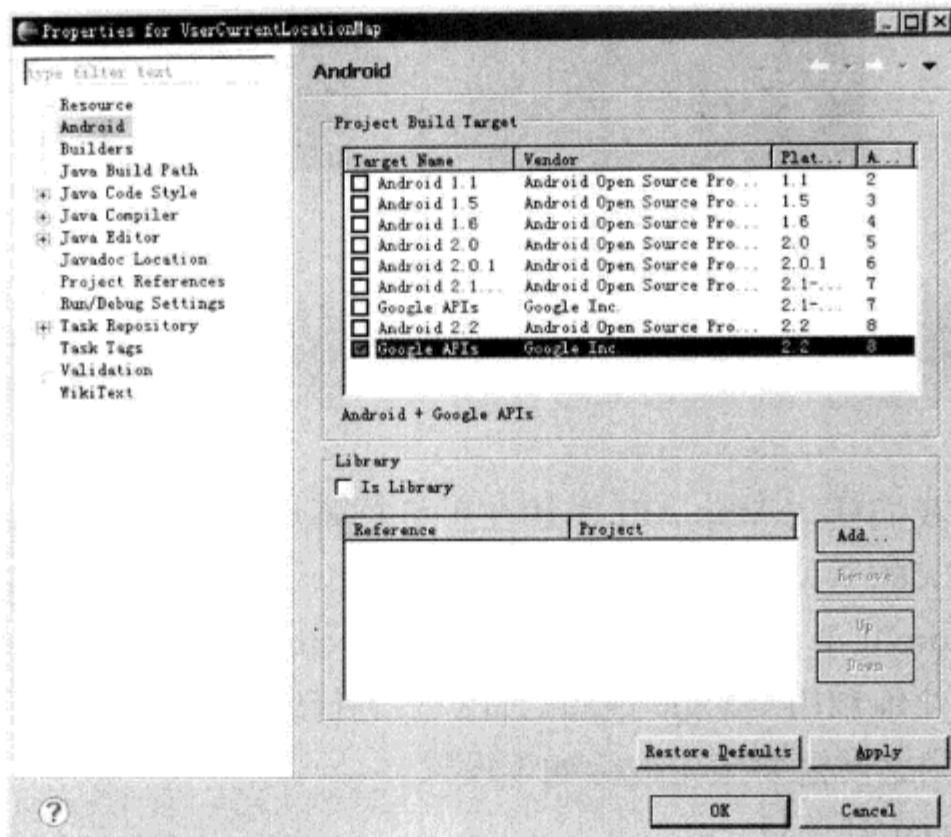


图9-5 将Google API添加到构建路径

(3) 获取 Map API 密钥

在利用MapView之前，必须要先申请一个Android Map API Key。具体步骤如下所示。

- ◆ 找到我们的 debug.keystore 文件，通常位于如下目录中。

C:\Documents and Settings\你的当前用户\Local Settings\Application Data\Android

- ◆ 接下来开始获取 MD5 指纹，运行 cmd.exe，执行如下命令获取 MD5 指纹。

```
>keytool -list -alias androiddebugkey -keystore "debug.keystore的路径"  
-storepass android -keypass android
```

例如笔者在个人机器中输入如下命令。

```
keytool -list -alias androiddebugkey -keystore "C:\Documents and  
Settings\Administrator\.android\debug.keystore" -storepass android  
-keypass android
```

此时系统会提示输入 keystore 密码，此时输入 android 后系统会输出我们申请到的 MD5 认证指纹，如图 9-6 所示。

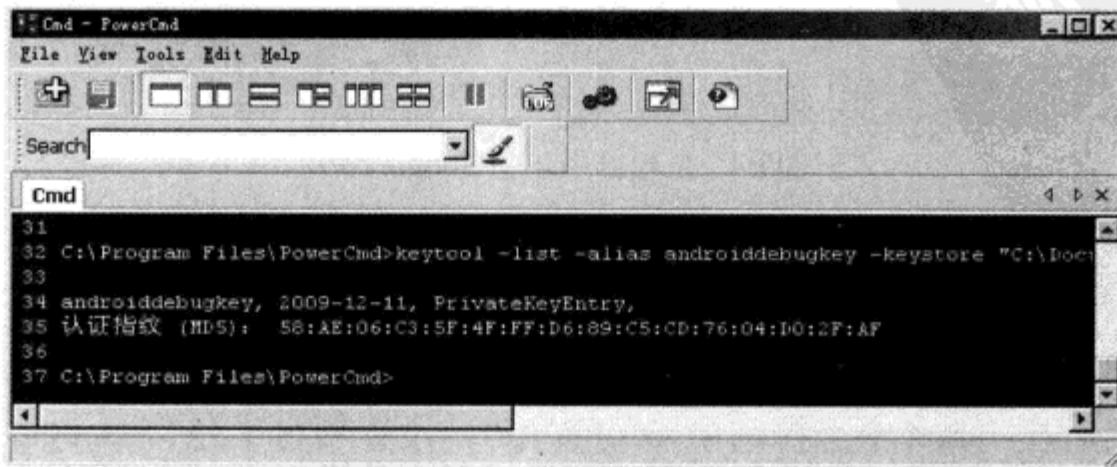


图9-6 获取的认证指纹



注意 因为在CMD中不能直接复制、粘贴使用CMD命令，这样很影响我们的编程效率，所以笔者使用了第三方软件PowerCmd来代替机器中自带的CMD工具。

- 开始申请 Android map 的 API Key

在浏览器中输入网址：<http://code.google.com/intl/zh-CN/android/maps-api-signup.html>，如图 9-7 所示。

If you use different keys for signing development builds and release builds, you will need to obtain a separate Maps API key for each certificate. Each key will on the corresponding certificate.

You also need a [Google Account](#) to get a Maps API key, and your API key will be connected to your Google Account.

Android Maps APIs Terms of Service

Last Updated: October 13, 2008

Thanks for your interest in the Android Maps APIs. The Android Maps APIs are a collection of services (including, but not limited to, the "com.google.android.maps.MapView" and "android.location.Geocoder" classes) that allow you to include maps, geocoding, and other content from Google and its content providers in your Android applications. The Android Maps APIs explicitly do not include any driving directions data or local search data that may be owned or licensed by Google.

1. Your relationship with Google.
1.1. Your use of any of the Android Maps APIs (referred to in this document as the "Maps API(s)" or the "Service") is subject to the

I have read and agree with the terms and conditions ([printable version](#))

My certificate's MD5 fingerprint: 58:AE:06:C3:5F:4F:FF:D6:89:C5:CD:76:04:D0:2F:AF

Generate API Key

图9-7 申请主页

在谷歌的 android map API Key 申请页面上输入图 9-6 中得到的 MD5 认证指纹，按下【Generate API Key】按钮转到图 9-8 所示的界面，在下方是我们申请到的 API Key。

感谢您注册 Android 地图 API 密钥！

您的密钥是：

0b97ff69320A_LUXeKCHTVAbSCqHAlqvzezEqjQ

此密钥适用于所有使用以下指纹所对应证书进行验证的应用程序：

58:AE:06:C3:5F:4F:FF:D6:89:C5:CD:76:04:D0:2F:AF

下面是一个 xml 格式的示例，帮助您了解地图功能：

```
<com.google.android.maps.MapView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0b97ff69320A_LUXeKCHTVAbSCqHAlqvzezEqjQ"/>
```

图9-8 得到的API Key

本实例的源码保存在【光盘\daima\第 9 章\miyue】，具体实现流程如下所示。

step 01 编写主布局文件 main.xml，在里面插入了 2 个 Button，分别用于“放大”和“缩小”地图，然后通过 ToggleButton 来控制是否显示卫星地图，最后设置申请的 apiKey。主要代码如下所示。

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

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:id="@+id/myLocationText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
<Button
    android:id="@+id/in"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="放大" />
<Button
    android:id="@+id/out"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="缩小" />
</LinearLayout>
<ToggleButton
    android:id="@+id/switchMap"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="卫星视图(关)"
    android:textOn="卫星视图(开)"/>
<com.google.android.maps.MapView
    android:id="@+id/myMapView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHALqvzetFqjQ"
    />
</LinearLayout>
```

step 02 在文件 `AndroidManifest.xml` 中声明 `android.permission.INTERNET` 和 `INTERNET` 权限，主要代码如下所示。

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

step 03 编写文件 miyue.java，其具体实现流程如下所示。

- 将MapView绘制到屏幕上，因为MapView只能继承自MapActivity的活动中，所以必须用方法onCreate将MapView绘制到屏幕上，并同时覆盖方法isRouteDisplayed()，它表示是否需要在地图上绘制导航线路。具体代码如下所示。

```
public class miyue extends MapActivity {
    MapView map;
    MapController ctrlMap;
    Button inBtn;
    Button outBtn;
    ToggleButton switchMap;
    @Override
    protected boolean isRouteDisplayed() {
        return false;
    }
}
```

- 引入主布局 main.xml，调用getOverlays()方法获取其Overlay链表，并将构建好的MyLocationOverlay对象添加到链表中去。其中MyLocationOverlay对象调用的enableMyLocation()方法表示尝试通过位置服务来获取当前的位置。具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    map = (MapView) findViewById(R.id.myMapView);
    List<Overlay> overlays = map.getOverlays();
    MyLocationOverlay myLocation = new MyLocationOverlay(this, map);
    myLocation.enableMyLocation();
    overlays.add(myLocation);
```

- 为【放大】和【缩小】这两个按钮设置单击响应程序，首先通过方法getController()获取MapView的MapController对象，然后在【放大】和【缩小】两个按钮单击事件监听器的回放方法里，根据按钮的不同实现对MapView的缩放。具体代码如下所示。

```
ctrlMap = map.getController();
inBtn = (Button) findViewById(R.id.in);
outBtn = (Button) findViewById(R.id.out);
OnClickListener listener = new OnClickListener() {
    @Override
```



```
public void onClick(View v) {  
    switch (v.getId()) {  
        case R.id.in: /*如果是缩放*/  
            ctrlMap.zoomIn();  
            break;  
        case R.id.out: /*如果是放大*/  
            ctrlMap.zoomOut();  
            break;  
        default:  
            break;  
    }  
}  
  
inBtn.setOnClickListener(listener);  
outBtn.setOnClickListener(listener);
```

- 通过方法 `onCheckedChanged` 获取是否选择了 `switchMap` 卫星试图，如果选择了则显示卫星地图。首先通过方法 `findViewById` 获取对应 id 的 `ToggleButton` 对象的引用，然后调用 `setOnCheckedChangeListener` 方法，设置对事件监听器选中的事件进行处理。根据 `ToggleButton` 是否被选中，进而通过 `setSatellite()` 方法启用或禁用卫星试图功能。具体代码如下所示。

```
- switchMap = (ToggleButton)findViewById(R.id.switchMap);  
switchMap.setOnCheckedChangeListener(new OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChanged(CompoundButton cBtn, boolean  
isChecked) {  
        if (isChecked == true) {  
            map.setSatellite(true);  
        } else {  
            map.setSatellite(false);  
        }  
    }  
});
```

- 使用 `LocationManager` 获取当前的位置，然后通过方法 `getBestProvider` 获取查询条件，最后设置更新位置信息的最小间隔为 2 秒，位移变化在 10 米以上。具体代码如下所示。

```
LocationManager locationManager;  
String context = Context.LOCATION_SERVICE;  
locationManager = (LocationManager) getSystemService(context);  
Criteria criteria = new Criteria();  
criteria.setAccuracy(Criteria.ACCURACY_FINE);
```

```

criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
String provider = locationManager.getBestProvider(criteria, true);
Location location = locationManager.getLastKnownLocation(provider);
updateWithNewLocation(location);
locationManager.requestLocationUpdates(provider, 2000, 10,
    locationListener);

```

- ◆ 定义方法 updateWithNewLocation(Location location) 来显示地里信息和地图信息，具体代码如下所示。

```

private void updateWithNewLocation(Location location) {
    String latLongString;
    TextView myLocationText;
    myLocationText = (TextView)findViewById(R.id.myLocationText);
    if (location != null) {
        double lat = location.getLatitude();
        double lng = location.getLongitude();
        latLongString = "纬度:" + lat + "\n经度:" + lng;
        ctrlMap.animateTo(new GeoPoint((int)(lat*1E6), (int)(lng*1E6)));
    } else {
        latLongString = "无法获取地理信息";
    }
    myLocationText.setText("您当前的位置是:\n" +
        latLongString);
}
}

```

此时在图 9-9 中选定一个经度和维度位置后可以显示此位置的定位信息，并且定位信息分别以文字和地图形式显示出来，如图 9-10 所示。

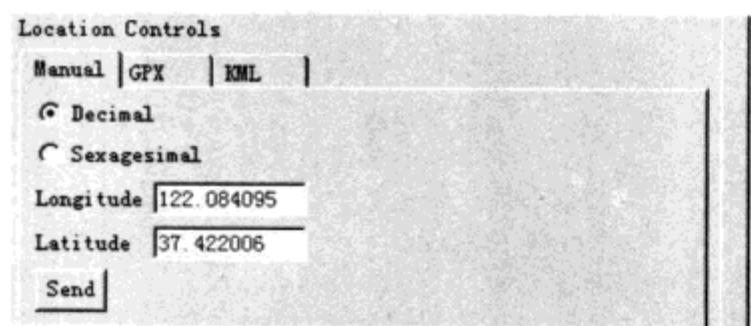


图9-9 指定位置



图9-10 显示对应信息



单击【放大】和【缩小】按钮后可以制地图的大小，如图 9-11 所示。打开卫星视图后可以显示此位置范围对应的卫星地图，如图 9-12 所示。



图9-11 放大后效果



图9-12 卫星地图

9.3 输入坐标后在地图中实现定位

在本实例中，通过 Google MapView 和 GeoPoint 实现地图经纬应用的流程。当在表单中输入一个经度和维度值后，单击【查询】按钮会在地图中显示此位置。本实例的源码保存在【光盘 \daima\ 第 9 章 \dingwei】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，主要代码如下所示。

```
<EditText  
    android:id="@+id/myEdit1"  
    android:layout_width="105px"  
    android:layout_height="40px"  
    android:textSize="16sp"  
    android:layout_x="130px"  
    android:layout_y="12px"  
    android:numeric="decimal"  
>  
</EditText>  
<EditText  
    android:id="@+id/myEdit2"  
    android:layout_width="105px"  
    android:layout_height="40px"  
    android:textSize="16sp"  
    android:layout_x="130px"  
    android:layout_y="52px"
```

```
    android:numeric="decimal"
>
</EditText>
<TextView
    android:id="@+id/myText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_longitude"
    android:textColor="@drawable/blue"
    android:layout_x="10px"
    android:layout_y="22px"
>
</TextView>
<TextView
    android:id="@+id/myText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_latitude"
    android:textColor="@drawable/blue"
    android:layout_x="10px"
    android:layout_y="62px"
>
</TextView>
<Button
    android:id="@+id/myButton1"
    android:layout_width="77px"
    android:layout_height="45px"
    android:text="@string/str_button1"
    android:layout_x="240px"
    android:layout_y="12px"
>
</Button>
<Button
    android:id="@+id/myButton2"
    android:layout_width="40px"
    android:layout_height="40px"
    android:text="@string/str_button2"
    android:textSize="20sp"
    android:layout_x="277px"
    android:layout_y="55px"
>
</Button>
<Button
```



```
    android:id="@+id/myButton3"
    android:layout_width="40px"
    android:layout_height="40px"
    android:text="@string/str_button3"
    android:textSize="20sp"
    android:layout_x="240px"
    android:layout_y="55px"
  >
</Button>
<!-- Google MapView Widget -->
<com.google.android.maps.MapView
    android:id="@+id/myMapView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_x="0px"
    android:layout_y="102px"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHAlqvzetFqjQ"
  >
</com.google.android.maps.MapView>
```

step 02 编写文件 dingwei.java，在 EditText 输入框中输入坐标的经度和维度，将坐标转换为 GeoPoint 对象后，再利用 MapController 的 animateTo() 方法将地图的中心点移到 GeoPoint 坐标上。文件 dingwei.java 的具体实现代码如下所示。

```
public class dingwei extends MapActivity
{
    private MapController mMapController01;
    private MapView mMapView01;
    private Button mButton01,mButton02,mButton03;
    private EditText mEditText01;
    private EditText mEditText02;
    private int intZoomLevel=0;
    /* Map启动时的默认坐标*/
    private double dLat=120.391177;
    private double dLng=39.9067452;
    @Override
    protected void onCreate(Bundle icicle)
    {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        /* 创建MapView对象 */
        mMapView01 = (MapView) findViewById(R.id.myMapView1);
```

```
mMapController01 = mMapView01.getController();
/* 设置MapView的显示选项(卫星、街道) */
mMapView01.setSatellite(false);
mMapView01.setStreetView(true);
/* 默认放大的层级 */
intZoomLevel = 17;
mMapController01.setZoom(intZoomLevel);
/* 设置Map的中点为默认经纬度 */
refreshMapView();

mEditText01 = (EditText)findViewById(R.id.myEdit1);
mEditText02 = (EditText)findViewById(R.id.myEdit2);

/* 送出查询的Button */
mButton01 = (Button)findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        /* 经纬度空白检查 */
        if(mEditText01.getText().toString().equals("") ||
           mEditText02.getText().toString().equals(""))
        {
            showDialog("经度或纬度填写不正确!");
        }
        else
        {
            /* 取得输入的经纬度 */
            dLng=Double.parseDouble(mEditText01.getText().toString());
            dLat=Double.parseDouble(mEditText02.getText().toString());
            /* 依输入的经纬度重整Map */
            refreshMapView();
        }
    }
});

/* 放大Map的Button */
mButton02 = (Button)findViewById(R.id.myButton2);
mButton02.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
```

```
{  
    intZoomLevel++;  
    if(intZoomLevel>mMapView01.getMaxZoomLevel())  
    {  
        intZoomLevel = mMapView01.getMaxZoomLevel();  
    }  
    mMapController01.setZoom(intZoomLevel);  
}  
});  
  
/* 缩小Map的Button */  
mButton03 = (Button)findViewById(R.id.myButton3);  
mButton03.setOnClickListener(new Button.OnClickListener()  
{  
    @Override  
    public void onClick(View v)  
    {  
        intZoomLevel--;  
        if(intZoomLevel<1)  
        {  
            intZoomLevel = 1;  
        }  
        mMapController01.setZoom(intZoomLevel);  
    }  
});  
}  
  
/* 重整Map的方法 */  
public void refreshMapView()  
{  
    GeoPoint p = new GeoPoint((int)(dLat* 1E6), (int)(dLng* 1E6));  
    mMapView01.displayZoomControls(true);  
    /* 将Map的中点移至GeoPoint */  
    mMapController01.animateTo(p);  
    mMapController01.setZoom(intZoomLevel);  
}  
  
@Override  
protected boolean isRouteDisplayed()  
{  
    return false;  
}
```

```

/* 显示Dialog的方法 */
private void showDialog(String mess) {
    new AlertDialog.Builder(dingwei.this).setTitle("Message")
        .setMessage(mess)
        .setNegativeButton("确定", new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int which)
        {
        }
    })
    .show();
}
}

```

执行后的效果如图 9-13 所示。



图9-13 执行效果

9.4 在手机中实现地址查询

在 Google 中提供了一个 Geocoder 服务，在非商用情况下，使用 Geocoder 可以反查 Address 地址对象服务。在本实例中，通过使用 Geocoder 实现地址反查功能。本实例的源码保存在【光盘 \daima\ 第 9 章 \dicha】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，主要代码如下所示。

```

<TextView
    android:id="@+id/myTextView1"
    android:layout_width="fill_parent"

```

```
    android:layout_height="wrap_content"
    android:textColor="@drawable/blue"
    android:text="@string/hello"/>

<EditText
    android:id="@+id/myEditText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="" />

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
    <Button
        android:id="@+id/myButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_button1"/>
    <Button
        android:id="@+id/myButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_button2"/>
    <Button
        android:id="@+id/myButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/str_button3"/>
</LinearLayout>
<com.google.android.maps.MapView
    android:id="@+id/myMapView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHALqvzetFqjQ" />
```

step 02 编写文件 dicha.java，在此文件中通过地址来获取 GeoPoint 方法，在自定义函数 getGeoByAddress() 中传入唯一的值字符串的方式传入地址，使用方法 Geocoder.getFromLocationName() 来获取从 Google 服务器找到的结果。文件 dicha.java 的主要代码如下所示。

```
protected void onCreate(Bundle icicle)
```

```
{  
    super.onCreate(icicle);  
    setContentView(R.layout.main);  
  
    mEditText01 = (EditText) findViewById(R.id.myEditText1);  
    mEditText01.setText  
(  
        getResources().getText(R.string.str_default_address).toString()  
    );  
  
    /* 创建MapView对象 */  
    mMapView01 = (MapView) findViewById(R.id.myMapView1);  
    mMapController01 = mMapView01.getController();  
    // 设置MapView的显示选项(卫星、街道)  
    mMapView01.setSatellite(true);  
    mMapView01.setStreetView(true);  
    mButton01 = (Button) findViewById(R.id.myButton1);  
    mButton01.setOnClickListener(new Button.OnClickListener()  
    {  
        @Override  
        public void onClick(View v)  
        {  
            if (mEditText01.getText().toString() != "")  
            {  
                refreshMapViewByGeoPoint  
                (  
                    getGeoByAddress  
                    (  
                        mEditText01.getText().toString()  
                    ), mMapView01, intZoomLevel, true  
                );  
            }  
        }  
    });  
  
    /* 放大 */  
    mButton02 = (Button) findViewById(R.id.myButton2);  
    mButton02.setOnClickListener(new Button.OnClickListener()  
    {  
        @Override  
        public void onClick(View v)  
        {  
            // TODO Auto-generated method stub  
        }  
    });  
}
```

```
        intZoomLevel++;
        if(intZoomLevel>mMapView01.getMaxZoomLevel())
        {
            intZoomLevel = mMapView01.getMaxZoomLevel();
        }
        mMapController01.setZoom(intZoomLevel);
    }
});

/* 缩小 */
mButton03 = (Button)findViewById(R.id.myButton3);
mButton03.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        intZoomLevel--;
        if(intZoomLevel<1)
        {
            intZoomLevel = 1;
        }
        mMapController01.setZoom(intZoomLevel);
    }
});

/* 初次查询地点 */
refreshMapViewByGeoPoint
(
    getGeoByAddress
    (
        getResources().getText(R.string.str_default_address).toString()
    ),mMapView01,intZoomLevel,true
);
}

private GeoPoint getGeoByAddress(String strSearchAddress)
{
    GeoPoint gp = null;
    try
    {
        if(strSearchAddress!="")
        {
            Geocoder mGeocoder01 = new Geocoder(dicha.this, Locale.getDefault());
        }
    }
}
```



```

List<Address> lstAddress = mGeocoder01.getFromLocationName(str
SearchAddress, 1);
if (!lstAddress.isEmpty())
{
    // Address[addressLines=[0:"U.S PIZZA",1:"15th Main Rd,
    Phase II, J P Nagar",2:"Bengaluru, Karnataka",3:"India"],feature=U.S
    PIZZA,admin=Karnataka,sub-admin=Bengaluru,locality=Bengaluru,thoroughfare
    =15th Main Rd,postalCode=null,countryCode=IN,countryName=India,hasLatitud
    e=true,latitude=18.508933,hasLongitude=true,longitude=73.8042,phone=null,
    url=null,extras=null]
    /*
    for (int i = 0; i < lstAddress.size(); ++i)
    {
        Address adsLocation = lstAddress.get(i);
        Log.i(TAG, "Address found = " + adsLocation.toString());
    }
    Address adsLocation = lstAddress.get(0);
    double geoLatitude = adsLocation.getLatitude()*1E6;
    double geoLongitude = adsLocation.getLongitude()*1E6;
    gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
}
else
{
    Log.i(TAG, "Address GeoPoint NOT Found.");
}
}
catch (Exception e)
{
    e.printStackTrace();
}
return gp;
}

public static void refreshMapViewByGeoPoint(GeoPoint gp, MapView mv,
int zoomLevel, boolean bIfSatellite)
{
try
{
    mv.displayZoomControls(true);
    /* 取得MapView的MapController */
    MapController mc = mv.getController();
    /* 移至该地理坐标地址 */
    mc.animateTo(gp);
}

```

```
/* 放大地图层级 */
mc.setZoom(zoomLevel);
/* 设置MapView的显示选项（卫星、街道）*/
if(bIfSatellite)
{
    mv.setSatellite(true);
    mv.setStreetView(true);
}
else
{
    mv.setSatellite(false);
}
}
catch(Exception e)
{
    e.printStackTrace();
}
}

@Override
protected boolean isRouteDisplayed()
{
    // TODO Auto-generated method stub
    return false;
}
```

执行后能够实现地址反查处理，如图 9-14 所示。



图9-14 地址反查

9.5 实现路径导航

在Android SDK中，可以使用手机内置地图程序传递导航坐标的方式来规划路径。在本实例中，通过Directions Route实现了路径导航功能。在具体实现上，先调用getProvider()来获取当前Location，以取得当前所在位置的地里坐标。并通过提供的EditText Widget来让用户输入将要前往的地址，通过地址来反复取得目的地的地理坐标。通过2个GeoPoint对象，并通过Intent方式来调用内置的地图程序。

本实例的源码保存在【光盘\daima\第9章\daohang】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，主要代码如下所示。

```
<!-- 建立一个TextView -->
<TextView
    android:id="@+id/myTextView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@drawable/blue"
    android:text="@string/hello"
/>
<!-- 建立一个EditText -->
<EditText
    android:id="@+id/myEditText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
/>
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
>
<!-- 建立第一个Button -->
<Button
    android:id="@+id/myButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button1"
/>
<!-- 建立第二个Button -->
<Button
    android:id="@+id/myButton2"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button2"
/>
<!-- 建立第三个Button -->
<Button
    android:id="@+id/myButton3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button3"
/>
</LinearLayout>
<!-- 设定Google map的API Key -->
<com.google.android.maps.MapView
    android:id="@+id/myMapView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHAlqvzetFqjQ"
/>
```

step 02 编写文件 daohang.java，其具体实现流程如下所示。

- ◆ 创建 LocationManager 对象以获取系统 LOCATION 本地服务，然后设置使用 MapView 控件显示选项（卫星、街道）。具体实现代码如下所示。

```
public class daohang extends MapActivity
{
    private TextView mTextView01;
    private LocationManager mLocationManager01;
    private String strLocationProvider = "";
    private Location mLocation01;
    private MapController mMapController01;
    private MapView mMapView01;
    private Button mButton01,mButton02,mButton03;
    private EditText mEditText01;
    private int intZoomLevel=0;
    private GeoPoint fromGeoPoint, toGeoPoint;

    @Override
    protected void onCreate(Bundle icicle)
    {
        // TODO Auto-generated method stub
```

```
super.onCreate(icicle);
setContentView(R.layout.main);

mTextView01 = (TextView) findViewById(R.id.myTextView1);

mEditText01 = (EditText) findViewById(R.id.myEditText1);
mEditText01.setText
(
    getResources().getText
    (R.string.str_default_address).toString()
);

/* 创建MapView对象 */
mMapView01 = (MapView) findViewById(R.id.myMapView1);
mMapController01 = mMapView01.getController();

// 设置MapView的显示选项（卫星、街道）
mMapView01.setSatellite(true);
mMapView01.setStreetView(true);

// 放大的层级
intZoomLevel = 15;
mMapController01.setZoom(intZoomLevel);

/* 创建LocationManager对象取得系统LOCATION服务 */
mLocationManager01 =
(LocationManager) getSystemService(Context.LOCATION_SERVICE);

/*
 * 自定义函数，访问Location Provider,
 * 并将之存储在strLocationProvider当中
 */
getLocationProvider();

/* 传入Location对象，显示于MapView */
fromGeoPoint = getGeoByLocation(mLocation01);
refreshMapViewByGeoPoint(fromGeoPoint,
    mMapView01, intZoomLevel);

/* 创建LocationManager对象，监听
 * Location更改时事件，更新MapView*/
mLocationManager01.requestLocationUpdates
(strLocationProvider, 2000, 10, mLocationListener01);
```



- ◆ 定义单击 mButton01 按钮的处理事件，先获取用户要前往地址的 GeoPoint 对象，传入路径规划所需要的地标地址。具体实现代码如下所示。

```
mButton01 = (Button) findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        if(mEditText01.getText().toString()!="")
        {
            /* 取得User要前往地址的GeoPoint对象 */
            toGeoPoint =
                getGeoByAddress(mEditText01.getText().toString());

            /* 路径规划Intent */
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);

            /* 传入路径规划所需要的地标地址 */
            intent.setData
            (
                Uri.parse("http://maps.google.com/maps?f=d&saddr="+
                    GeoPointToString(fromGeoPoint)+"
                    "&daddr="+GeoPointToString(toGeoPoint)+"
                    "&hl=cn" +
                    "")
            );
            startActivity(intent);
        }
    }
});
```

- ◆ 定义单击 mButton02 按钮的处理事件，单击后实现地图放大处理。具体代码如下所示。

```
/* 放大地图 */
mButton02 = (Button) findViewById(R.id.myButton2);
mButton02.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
```

```

    // TODO Auto-generated method stub
    intZoomLevel++;
    if(intZoomLevel>mMapView01.getMaxZoomLevel())
    {
        intZoomLevel = mMapView01.getMaxZoomLevel();
    }
    mMapController01.setZoom(intZoomLevel);
}
});

```

- ◆ 定义单击 mButton03 按钮的处理事件，单击实现地图缩小处理。具体代码如下所示。

```

/* 缩小地图 */
mButton03 = (Button)findViewById(R.id.myButton3);
mButton03.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        intZoomLevel--;
        if(intZoomLevel<1)
        {
            intZoomLevel = 1;
        }
        mMapController01.setZoom(intZoomLevel);
    }
});

```

- ◆ 捕捉当手机 GPS 坐标更新时的事件，当手机收到位置更改时，将 location 传入 getMyLocation 对象。具体代码如下所示。

```

/* 捕捉当手机GPS坐标更新时的事件 */
public final LocationListener mLocationListener01 =
new LocationListener()
{
    @Override
    public void onLocationChanged(Location location)
    {
        /* 当手机收到位置更改时，将location传入getMyLocation */
        mLocation01 = location;
        fromGeoPoint = getGeoByLocation(location);
        refreshMapViewByGeoPoint(fromGeoPoint,
            mMapView01, intZoomLevel);
    }
};

```



```
    }  
};
```

- ◆ 定义方法 getGeoByLocation，设置当传入 Location 对象时取回其 GeoPoint 对象。具体代码如下所示。

```
/* 传入Location对象，取回其GeoPoint对象 */  
private GeoPoint getGeoByLocation(Location location)  
{  
    GeoPoint gp = null;  
    try  
    {  
        /* 如果Location存在 */  
        if (location != null)  
        {  
            double geoLatitude = location.getLatitude()*1E6;  
            double geoLongitude = location.getLongitude()*1E6;  
            gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);  
        }  
    }  
    catch (Exception e)  
    {  
        e.printStackTrace();  
    }  
    return gp;  
}
```

- ◆ 定义方法 getGeoByAddress，设置当输入地址时获取其 GeoPoint 对象。具体代码如下所示。

```
/* 输入地址，取得其GeoPoint对象 */  
private GeoPoint getGeoByAddress(String strSearchAddress)  
{  
    GeoPoint gp = null;  
    try  

```

```

        Address adsLocation = lstAddress.get(0);
        double geoLatitude = adsLocation.getLatitude()*1E6;
        double geoLongitude = adsLocation.getLongitude()*1E6;
        gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
    }
}
catch (Exception e)
{
    e.printStackTrace();
}
return gp;
}

```

- ◆ 定义方法 refreshMapViewByGeoPoint 和 refreshMapViewByCode，用于分别传入 geoPoint 更新MapView 里的谷歌地图和传入经纬度更新MapView 里的谷歌地图。具体代码如下所示。

```

/* 传入geoPoint更新MapView里的Google Map */
public static void refreshMapViewByGeoPoint
(GeoPoint gp, MapView mapview, int zoomLevel)
{
    try
    {
        mapview.displayZoomControls(true);
        MapController myMC = mapview.getController();
        myMC.animateTo(gp);
        myMC.setZoom(zoomLevel);
        mapview.setSatellite(false);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

/* 传入经纬度更新MapView里的Google Map */
public static void refreshMapViewByCode
(double latitude, double longitude,
     MapView mapview, int zoomLevel)
{
    try
    {

```



```
GeoPoint p = new GeoPoint((int) latitude, (int) longitude);
mapview.displayZoomControls(true);
MapController myMC = mapview.getController();
myMC.animateTo(p);
myMC.setZoom(zoomLevel);
mapview.setSatellite(false);
}
catch(Exception e)
{
    e.printStackTrace();
}
}
```

- ◆ 定义方法 GeoPointToString 将 GeoPoint 中的经纬度以“String, String”的格式返回，具体代码如下所示。

```
/* 将GeoPoint里的经纬度以String, String返回 */
private String GeoPointToString(GeoPoint gp)
{
    String strReturn="";
    try
    {
        /* 当Location存在 */
        if (gp != null)
        {
            double geoLatitude = (int)gp.getLatitudeE6()/1E6;
            double geoLongitude = (int)gp.getLongitudeE6()/1E6;
            strReturn = String.valueOf(geoLatitude)+","+
                String.valueOf(geoLongitude);
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return strReturn;
}
```

执行后的效果如图 9-15 所示；单击【开始规划路径】按钮后弹出选择对话框，如图 9-16 所示。在此选择“Maps”后弹出规划界面，如图 9-17 所示；在图 9-17 所示界面中，在第一个文本框中设置出发位置，例如“beijing”，在第二个文本框中设置目的地位置，例如“tianjin”，选择汽车前往标志按钮，如图 9-18 所示；单击按钮“Go”，系统将实现北京出发，目的地到天津的线路规划，产生线路规划图，最终的执行界面如图 9-19 所示。



图9-15 执行效果

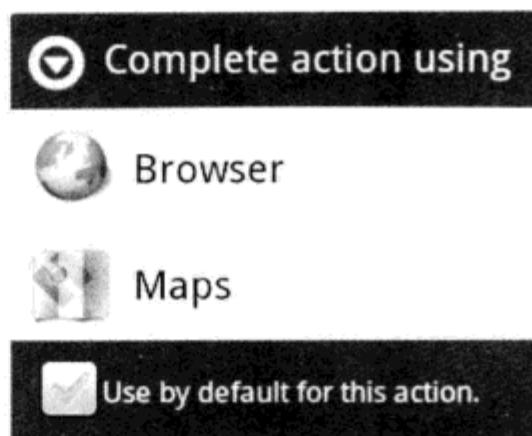


图9-16 选择对话框

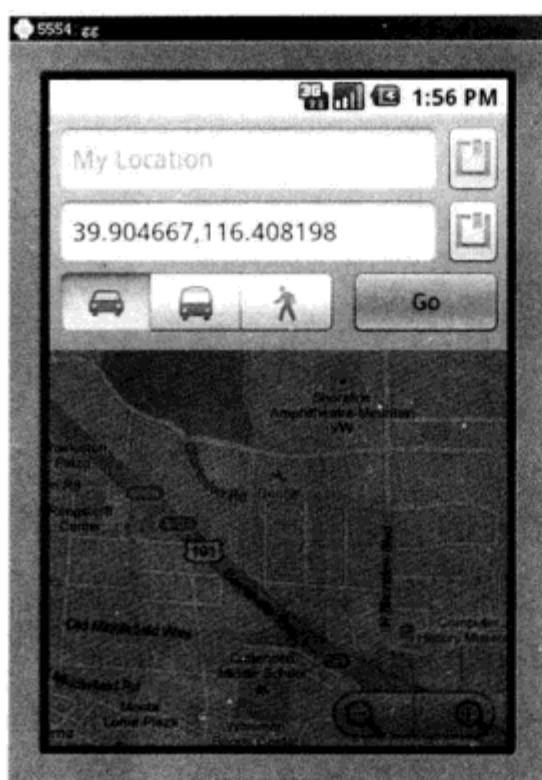


图9-17 规划界面



图9-18 设置出发地和目的地

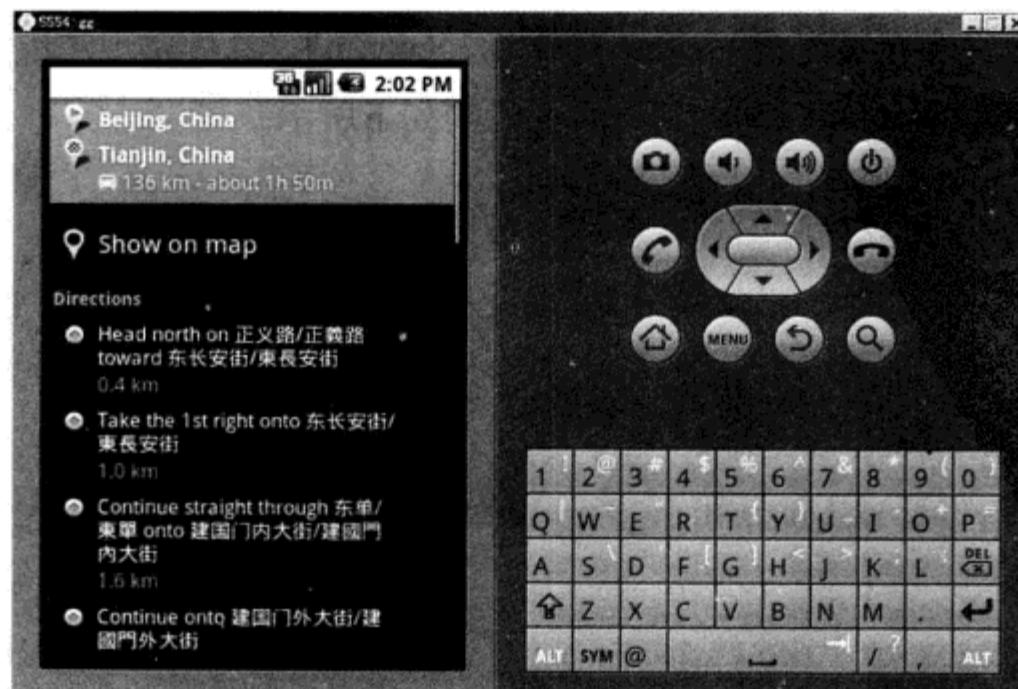


图9-19 生成的线路规划

单击图 9-19 中的“Show on map”后将会在地图中显示行走线路，如图 9-20 所示。



图9-20 地图中的线路规划

9.6

移动手机时自动更新位置

在现实应用中，GPS 的使用越来越广泛。但是每一个位置和路况都不是固定不变的，这就要求系统能够根据各种变化而变化，不能误导了人们的使用。在 Android SDK 中，支持手机 GPS 定位事件处理。在 Android 手机中内置了 Google Map，但是不能随着手机的移动而更新。在本实例中，将插入一个 TextView 和 MapView，当手机移动时，会触发内置的 GPS 定位坐标的改变事件。只要程序发现地理位置的变化，便实时更新MapView 里的 Google Map，并反查地理坐标系统的信息。从而实现了在 Android 中 GPS 实时更新的功能。

本实例的源码保存在【光盘\daima\第 9 章\gengxin】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```

    android:textColor="@drawable/blue"
    android:text="@string/hello"/>
<com.google.android.maps.MapView
    android:id="@+id/myMapView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHALqvzetFqjQ"
/>
</LinearLayout>

```

step 02 编写文件 gengxin.java，其具体实现流程如下所示。

- 创建 LocationManager 对象 mLocationManager01 来获取系统 LOCATION 服务，具体代码如下所示。

```

public class gengxin extends MapActivity
{
    private LocationManager mLocationManager01;
    private String strLocationProvider = "";
    private Location mLocation01=null;
    private TextView mTextView01;
    private MapView mMapView01;
    private GeoPoint currentGeoPoint;
    private int intZoomLevel = 20;
    @Override
    protected void onCreate(Bundle icicle)
    {
        // TODO Auto-generated method stub
        super.onCreate(icicle);
        setContentView(R.layout.main);

        mTextView01 = (TextView) findViewById(R.id.myTextView1);
        /* 创建MapView对象 */
        mMapView01 = (MapView) findViewById(R.id.myMapView1);

        /* 创建LocationManager对象取得系统LOCATION服务 */
        mLocationManager01 =
            (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        /* 第一次运行向Location Provider取得Location */
        mLocation01 = getLocationProvider(mLocationManager01);
    }
}

```

- 通过方法 getLocationProvider 取得当前 Location 位置，创建 LocationManager 对象用于监听 Location 更改时事件，并更新 MapView 控件中的地图。具体代码如下所示。

```

/* 第一次运行向Location Provider取得Location */
mLocation01 = getLocationProvider(mLocationManager01);

```



```
if(mLocation01!=null)
{
    processLocationUpdated(mLocation01);
}
else
{
    mTextView01.setText
    (
        getResources().getText(R.string.str_err_location).toString()
    );
}
/* 创建LocationManager对象，监听Location更改时事件，更新MapView */
mLocationManager01.requestLocationUpdates
(strLocationProvider, 2000, 10, mLocationListener01);
}
```

- 通过方法 LocationListener 来监听定位信息，当手机收到位置更改时将 location 位置传入并取得当前地理坐标。具体代码如下所示。

```
public final LocationListener
mLocationListener01 = new LocationListener()
{
    @Override
    public void onLocationChanged(Location location)
    {
        /* 当手机收到位置更改时，将location传入取得地理坐标 */
        processLocationUpdated(location);
    }
    public void onProviderDisabled(String provider)
    {
        /* 当Provider已离开服务范围时 */
    }
    public void onProviderEnabled(String provider)
    {
        // TODO Auto-generated method stub
    }
}
```

- 定义方法 getAddressbyGeoPoint(GeoPoint gp) 来获取定位地址的信息，先创建 Geocoder 对象并取出地理坐标经纬度，然后判断地址是否为多行，最后将采集到的地址组合后放在 StringBuilder 对象中输出。具体代码如下所示。

```
public String getAddressbyGeoPoint(GeoPoint gp)
{
    Geocoder gc = new Geocoder(this);
    List<Address> addressList = gc.getFromLocation(gp.getLatitude(),
                                                gp.getLongitude(), 1);
    if(addressList != null && addressList.size() > 0)
    {
        Address address = addressList.get(0);
        if(address.getThoroughfare() != null && address.getThoroughfare().length() > 0)
        {
            sb.append(address.getThoroughfare());
        }
        if(address.getSubThoroughfare() != null && address.getSubThoroughfare().length() > 0)
        {
            sb.append(" " + address.getSubThoroughfare());
        }
        if(address.getLocality() != null && address.getLocality().length() > 0)
        {
            sb.append(" " + address.getLocality());
        }
        if(address.getSubLocality() != null && address.getSubLocality().length() > 0)
        {
            sb.append(" " + address.getSubLocality());
        }
        if(address.getCountryName() != null && address.getCountryName().length() > 0)
        {
            sb.append(" " + address.getCountryName());
        }
        if(address.getPostalCode() != null && address.getPostalCode().length() > 0)
        {
            sb.append(" " + address.getPostalCode());
        }
    }
    return sb.toString();
}
```

```

String strReturn = "";
try
{
    /* 当GeoPoint不等于null */
    if (gp != null)
    {
        /* 创建Geocoder对象 */
        Geocoder gc = new Geocoder
(gengxin.this, Locale.getDefault());

        /* 取出地理坐标经纬度 */
        double geoLatitude = (int)gp.getLatitudeE6()/1E6;
        double geoLongitude = (int)gp.getLongitudeE6()/1E6;

        /* 自经纬度取得地址（可能有多行地址） */
        List<Address> lstAddress =
gc.getFromLocation(geoLatitude, geoLongitude, 1);

        StringBuilder sb = new StringBuilder();

        /* 判断地址是否为多行 */
        if (lstAddress.size() > 0)
        {
            Address adsLocation = lstAddress.get(0);
            for(int i=0;i<adsLocation.getMaxAddressLineIndex();i++)
            {
                sb.append(adsLocation.getAddressLine(i)).append("\n");
            }
            sb.append(adsLocation.getLocality()).append("\n");
            sb.append(adsLocation.getPostalCode()).append("\n");
            sb.append(adsLocation.getCountryName());
        }

        /* 将采集到的地址，组合后放在StringBuilder对象中输出用*/
        strReturn = sb.toString();
    }
}
catch(Exception e)
{
    e.printStackTrace();
}

return strReturn;
}

```

```
public Location getLocationProvider(LocationManager lm)
{
    Location retLocation = null;
    try
    {
        Criteria mCriteria01 = new Criteria();
        mCriteria01.setAccuracy(Criteria.ACCURACY_FINE);
        mCriteria01.setAltitudeRequired(false);
        mCriteria01.setBearingRequired(false);
        mCriteria01.setCostAllowed(true);
        mCriteria01.setPowerRequirement(Criteria.POWER_LOW);
        strLocationProvider = lm.getBestProvider(mCriteria01, true);
        retLocation = lm.getLastKnownLocation(strLocationProvider);
    }
    catch(Exception e)
    {
        mTextView01.setText(e.toString());
        e.printStackTrace();
    }
    return retLocation;
}

private GeoPoint getGeoByLocation(Location location)
{
    GeoPoint gp = null;
    try
    {
        /* 当Location存在 */
        if (location != null)
        {
            double geoLatitude = location.getLatitude()*1E6;
            double geoLongitude = location.getLongitude()*1E6;
            gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return gp;
}

public static void refreshMapViewByGeoPoint
(GeoPoint gp, MapView mv, int zoomLevel, boolean bIfSatellite)
```

```

{
    try
    {
        mv.displayZoomControls(true);
        /* 取得MapView的MapController */
        MapController mc = mv.getController();
        /* 移至该地理坐标地址 */
        mc.animateTo(gp);

        /* 放大地图层级 */
        mc.setZoom(zoomLevel);
        /* 设置MapView的显示选项（卫星、街道）*/
        if(bIfSatellite)
        {
            mv.setSatellite(true);
            mv.setStreetView(true);
        }
        else
        {
            mv.setSatellite(false);
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
}

```

- ◆ 定义方法 processLocationUpdated，设置当手机获取的位置发生变化时将 location 位置传入到 GeoPoint，并同时在MapView 控件中更新显示地图。具体代码如下所示。

```

/* 当手机收到位置更改，将location传入GeoPoint及MapView */
private void processLocationUpdated(Location location)
{
    /* 传入Location对象，取得GeoPoint地理坐标 */
    currentGeoPoint = getGeoByLocation(location);
    /* 更新MapView显示Google Map */
    refreshMapViewByGeoPoint
    (currentGeoPoint, mMapView01, intZoomLevel, true);
    mTextView01.setText
    (
        getResources().getText(R.string.str_my_location).toString()+
        "\n"+getAddressbyGeoPoint(currentGeoPoint)
    );
}

```



```
    }  
    protected boolean isRouteDisplayed()  
    {  
        // TODO Auto-generated method stub  
        return false;  
    }  
}
```

执行后将显示当前位置的定位信息，并能实现及时更新功能，如图 9-21 所示。

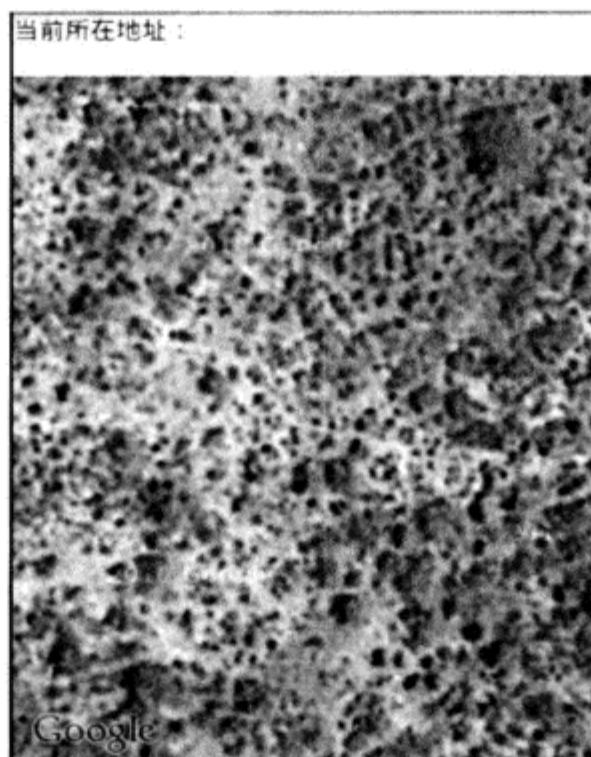


图9-21 执行效果

9.7 在地图中绘制线路并计算距离

通过前面的演示实例，相信大家了解了在 Android 实现 GPS 导航的方法。其实除了导航功能之外，还可以在地图上绘制线路并计算距离，实现一个完整的导航功能。在本实例中，实现了在 Android 地图上绘制线路并计算距离的功能。本实例的源码保存在【光盘\daima\第 9 章\juli】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，主要实现代码如下所示。

```
<AbsoluteLayout  
    android:id="@+id/widget0"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:background="@drawable/white"  
    xmlns:android="http://schemas.android.com/apk/res/android"  
>
```

```
<Button  
    android:id="@+id/myButton1"  
    android:layout_width="80px"  
    android:layout_height="40px"  
    android:text="@string/str_button1"  
    android:layout_x="10px"  
    android:layout_y="12px"  
/>  
<Button  
    android:id="@+id/myButton2"  
    android:layout_width="80px"  
    android:layout_height="40px"  
    android:text="@string/str_button2"  
    android:layout_x="90px"  
    android:layout_y="12px"  
/>  
<Button  
    android:id="@+id/myButton3"  
    android:layout_width="40px"  
    android:layout_height="40px"  
    android:text="@string/str_button3"  
    android:textSize="20sp"  
    android:layout_x="170px"  
    android:layout_y="12px"  
/>  
<Button  
    android:id="@+id/myButton4"  
    android:layout_width="40px"  
    android:layout_height="40px"  
    android:text="@string/str_button4"  
    android:textSize="20sp"  
    android:layout_x="210px"  
    android:layout_y="12px"  
/>  
<TextView  
    android:id="@+id/myText1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/str_text1"  
    android:textColor="@drawable/blue"  
    android:textSize="16sp"  
    android:layout_x="10px"  
    android:layout_y="52px"
```



```
>
<com.google.android.maps.MapView
    android:id="@+id/myMapView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_x="0px"
    android:layout_y="82px"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHALqvzetFqjQ"
/>
```

step 02 编写文件 juli.java，通过自定义的 MyOverLay 类在 MapView 控件中绘制标记。
文件 juli.java 的主要代码如下所示。

```
public class juli extends MapActivity
{
    private TextView mTextView;
    private Button mButton01;
    private Button mButton02;
    private Button mButton03;
    private Button mButton04;
    private MapView mMapView;
    private MapController mMapController;
    private LocationManager mLocationManager;
    private Location mLocation;
    private String mLocationPrvider="";
    private int zoomLevel=0;
    private GeoPoint gp1;
    private GeoPoint gp2;
    private boolean _run=false;
    private double distance=0;
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /* 创建MapView对象 */
        mMapView = (MapView) findViewById(R.id.myMapView1);
        mMapController = mMapView.getController();
        /* 对象初始化 */
        mTextView = (TextView) findViewById(R.id.myText1);
        mButton01 = (Button) findViewById(R.id.myButton1);
        mButton02 = (Button) findViewById(R.id.myButton2);
        mButton03 = (Button) findViewById(R.id.myButton3);
```

```
mButton04 = (Button) findViewById(R.id.myButton4);
/* 设置默认的放大层级 */
zoomLevel = 17;
mMapController.setZoom(zoomLevel);

/* Provider初始化 */
mLocationManager = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);
/* 取得Provider与Location */
getLocationPriveder();
if(mLocation!=null)
{
    /* 取得目前的经纬度 */
    gp1=getGeoByLocation(mLocation);
    gp2=gp1;
    /* 将MapView的中点移至当前位置 */
    refreshMapView();
    /* 设置事件的Listener */
    mLocationManager.requestLocationUpdates(mLocationPriveder,
        2000, 10, mLocationListener);
}
else
{
    new AlertDialog.Builder(juli.this).setTitle("系统信息")
        .setMessage(getResources().getString(R.string.str_message))
        .setNegativeButton("确定",new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int which)
        {
            juli.this.finish();
        }
    })
    .show();
}
/* 开始记录的Button */
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        gp1=gp2;
        /* 清除Overlay */
        resetOverlay();
```

```
/* 画起点 */
setStartPoint();
/* 更新MapView */
refreshMapView();
/* 重设移动距离为0，并更新TextView */
distance=0;
mTextView.setText("移动距离: 0M");
/* 启动画路线的机制 */
_run=true;
}
});
/* 结束记录的Button */
mButton02.setOnClickListener(new Button.OnClickListener()
{
@Override
public void onClick(View v)
{
/* 画终点 */
setEndPoint();
/* 更新MapView */
refreshMapView();
/* 终止画路线的机制 */
_run=false;
}
});
/* 缩小地图的Button */
mButton03.setOnClickListener(new Button.OnClickListener()
{
@Override
public void onClick(View v)
{
zoomLevel--;
if(zoomLevel<1)
{
zoomLevel = 1;
}
mMapController.setZoom(zoomLevel);
}
});
/* 放大地图的Button */
mButton04.setOnClickListener(new Button.OnClickListener()
{
@Override
```

```
public void onClick(View v)
{
    zoomLevel++;
    if(zoomLevel>mMapView.getMaxZoomLevel())
    {
        zoomLevel = mMapView.getMaxZoomLevel();
    }
    mMapController.setZoom(zoomLevel);
}
});

/* MapView的Listener */
public final LocationListener mLocationListener =
    new LocationListener()
{
    @Override
    public void onLocationChanged(Location location)
    {
        /* 如果记录进行中，就画路线并更新移动距离 */
        if(_run)
        {
            /* 记下移动后的位置 */
            gp2=getGeoByLocation(location);
            /* 画路线 */
            setRoute();
            /* 更新MapView */
            refreshMapView();
            /* 取得移动距离 */
            distance+=GetDistance(gp1,gp2);
            mTextView.setText("移动距离: "+format(distance)+"M");
            gp1=gp2;
        }
    }
}

/* 取得GeoPoint的方法 */
private GeoPoint getGeoByLocation(Location location)
{
    GeoPoint gp = null;
    try
    {
        if (location != null)
        {
            double geoLatitude = location.getLatitude()*1E6;
            double geoLongitude = location.getLongitude()*1E6;
        }
    }
}
```

```
        gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
return gp;
}
/* 取得LocationProvider */
public void getLocationPrivider()
{
    Criteria mCriteria01 = new Criteria();
    mCriteria01.setAccuracy(Criteria.ACCURACY_FINE);
    mCriteria01.setAltitudeRequired(false);
    mCriteria01.setBearingRequired(false);
    mCriteria01.setCostAllowed(true);
    mCriteria01.setPowerRequirement(Criteria.POWER_LOW);

    mLocationPrivider = mLocationManager
                        .getBestProvider(mCriteria01, true);
    mLocation = mLocationManager
                .getLastKnownLocation(mLocationPrivider);
}

/* 设置起点的方法 */
private void setStartPoint()
{
    int mode=1;
    OverLay mOverlay = new OverLay(gp1, gp2, mode);
    List<Overlay> overlays = mMapView.getOverlays();
    overlays.add(mOverlay);
}

/* 设置路线的方法 */
private void setRoute()
{
    int mode=2;
    OverLay mOverlay = new OverLay(gp1, gp2, mode);
    List<Overlay> overlays = mMapView.getOverlays();
    overlays.add(mOverlay);
}

/* 设置终点的方法 */
private void setEndPoint()
{
```

```

int mode=3;
OverLay mOverlay = new OverLay(gp1,gp2,mode);
List<Overlay> overlays = mMapView.getOverlays();
overlays.add(mOverlay);

}
/* 重设Overlay的方法 */
private void resetOverlay()
{
    List<Overlay> overlays = mMapView.getOverlays();
    overlays.clear();
}

/*
 * 更新MapView的方法
 */
public void refreshMapView()
{
    mMapView.displayZoomControls(true);
    MapController myMC = mMapView.getController();
    myMC.animateTo(gp2);
    myMC.setZoom(zoomLevel);
    mMapView.setSatellite(false);
}

/*
 * 取得两点间的距离的方法
 */
public double GetDistance(GeoPoint gp1,GeoPoint gp2)
{
    double Lat1r = ConvertDegreeToRadians(gp1.getLatitudeE6()/1E6);
    double Lat2r = ConvertDegreeToRadians(gp2.getLatitudeE6()/1E6);
    double Long1r= ConvertDegreeToRadians(gp1.getLongitudeE6()/1E6);
    double Long2r= ConvertDegreeToRadians(gp2.getLongitudeE6()/1E6);
    /* 地球半径(KM) */
    double R = 6371;
    double d = Math.acos(Math.sin(Lat1r)*Math.sin(Lat2r) +
        Math.cos(Lat1r)*Math.cos(Lat2r)*
        Math.cos(Long2r-Long1r))*R;
    return d*1000;
}

/*
 * format移动距离的方法
 */
public String format(double num)
{
    NumberFormat formatter = new DecimalFormat("###");
    String s=formatter.format(num);
    return s;
}

```

step 03 编写文件 OverLay.java，其功能是在MapView地图上绘制图形，并使用方法 getProjection 来获取 Projection 对象，然后使用 projection.toPixels(gp1, point) 将



getProjection() 转换成 Point，最后根据 Point 对象的对应位置来绘制图形。文件 OverLay.java 的主要代码如下所示。

```
public class OverLay extends Overlay
{
    private GeoPoint gp1;
    private GeoPoint gp2;
    private int mRadius=6;
    private int mode=0;
    /* 构造器，传入起点与终点的GeoPoint与mode */
    public OverLay(GeoPoint gp1,GeoPoint gp2,int mode)
    {
        this.gp1 = gp1;
        this.gp2 = gp2;
        this.mode = mode;
    }
    public boolean draw
    (Canvas canvas, MapView mapView, boolean shadow, long when)
    {
        Projection projection = mapView.getProjection();
        if (shadow == false)
        {
            /* 设置笔刷 */
            Paint paint = new Paint();
            paint.setAntiAlias(true);
            paint.setColor(Color.BLUE);
            Point point = new Point();
            projection.toPixels(gp1, point);
            /* mode=1: 创建起点 */
            if (mode==1)
            {
                /* 定义RectF对象 */
                RectF oval=new RectF(point.x - mRadius, point.y - mRadius,
                                     point.x + mRadius, point.y + mRadius);
                /* 绘制起点的圆形 */
                canvas.drawOval(oval, paint);
            }
            /* mode=2: 画路线 */
            else if(mode==2)
            {
                Point point2 = new Point();
                projection.toPixels(gp2, point2);
                paint.setColor(Color.BLACK);
```

```

paint.setStrokeWidth(5);
paint.setAlpha(120);
/* 画线 */
canvas.drawLine(point.x, point.y, point2.x, point2.y, paint);
}
/* mode=3: 创建终点 */
else if(mode==3)
{
    /* 避免误差, 先画最后一段的路线 */
    Point point2 = new Point();
    projection.toPixels(gp2, point2);
    paint.setStrokeWidth(5);
    paint.setAlpha(120);
    canvas.drawLine(point.x, point.y, point2.x, point2.y, paint);
    /* 定义RectF对象 */
    RectF oval=new RectF(point2.x - mRadius, point2.y - mRadius,
                        point2.x + mRadius, point2.y + mRadius);
    /* 绘制终点的圆形 */
    paint.setAlpha(255);
    canvas.drawOval(oval, paint);
}
}
return super.draw(canvas, mapView, shadow, when);
}
}

```

执行后依次单击按钮【开始记录】和【结束记录】按钮后，可以在地图中绘制 GPS 轨迹记录，执行效果如图 9-22 所示。



图9-22 执行效果

9.8 在谷歌地图中显示指定的位置

在前面的实例中，功能都十分人性化，能够根据当前的位置来显示对应的定位信息。而在本实例中，在代码中指定一个坐标位置，然后在手机屏幕中用谷歌地图显示此位置。本例假设的坐标是济南泉城广场的坐标。本实例的源码保存在【光盘\daima\第9章\weizhi】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，在里面引用了 Map 密钥，主要代码如下所示。

```
<?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"
    >
    <com.google.android.maps.MapView
        android:id="@+id/MapView01"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:apiKey="0by7ffx8jX0C4kaxou6vckW6pkvss4ZwxjYIofg"/>
</RelativeLayout>
```

step 02 编写文件 weizhi.java，在里面设置了坐标位置是 (36.662223, 117.026095)，主要代码如下所示。

```
public class weizhi extends MapActivity
{
    private MapView mMapView;
    private MapController mMapController;
    private GeoPoint mGeoPoint;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mMapView = (MapView) findViewById(R.id.MapView01);
        //设置为交通模式
        //mMapView.setTraffic(true);
        //设置为卫星模式
        mMapView.setSatellite(true);
        //设置为街景模式
    }
}
```



```
//mMapView.setStreetView(false);
//取得MapController对象(控制MapView)
mMapController = mMapView.getController();
mMapView.setEnabled(true);
mMapView.setClickable(true);
//设置地图支持缩放
mMapView.setBuiltInZoomControls(true);
//设置起点为济南
mGeoPoint = new GeoPoint((int) (36.662223 * 1000000), (int)
(117.026095 * 1000000));
//定位到济南
mMapController.animateTo(mGeoPoint);
//设置倍数(1-21)
mMapController.setZoom(15);
//添加Overlay, 用于显示标注信息
MyLocationOverlay myLocationOverlay = new MyLocationOverlay();
List<Overlay> list = mMapView.getOverlays();
list.add(myLocationOverlay);
}
protected boolean isRouteDisplayed()
{
    return false;
}
class MyLocationOverlay extends Overlay
{
    @Override
    public boolean draw(Canvas canvas, MapView mapView, boolean
shadow, long when)
    {
        super.draw(canvas, mapView, shadow);
        Paint paint = new Paint();
        Point myScreenCoords = new Point();
        // 将经纬度转换成实际屏幕坐标
        mapView.getProjection().toPixels(mGeoPoint, myScreenCoords);
        paint.setStrokeWidth(1);
        paint.setARGB(255, 255, 0, 0);
        paint.setStyle(Paint.Style.STROKE);
        Bitmap bmp = BitmapFactory.decodeResource(getResources(),
R.drawable.home);
        canvas.drawBitmap(bmp, myScreenCoords.x, myScreenCoords.y,
paint);
        canvas.drawText("泉城广场", myScreenCoords.x, myScreenCoords.
y, paint);
    }
}
```

```
    return true;  
}  
}  
}
```

执行后的效果如图 9-23 所示。



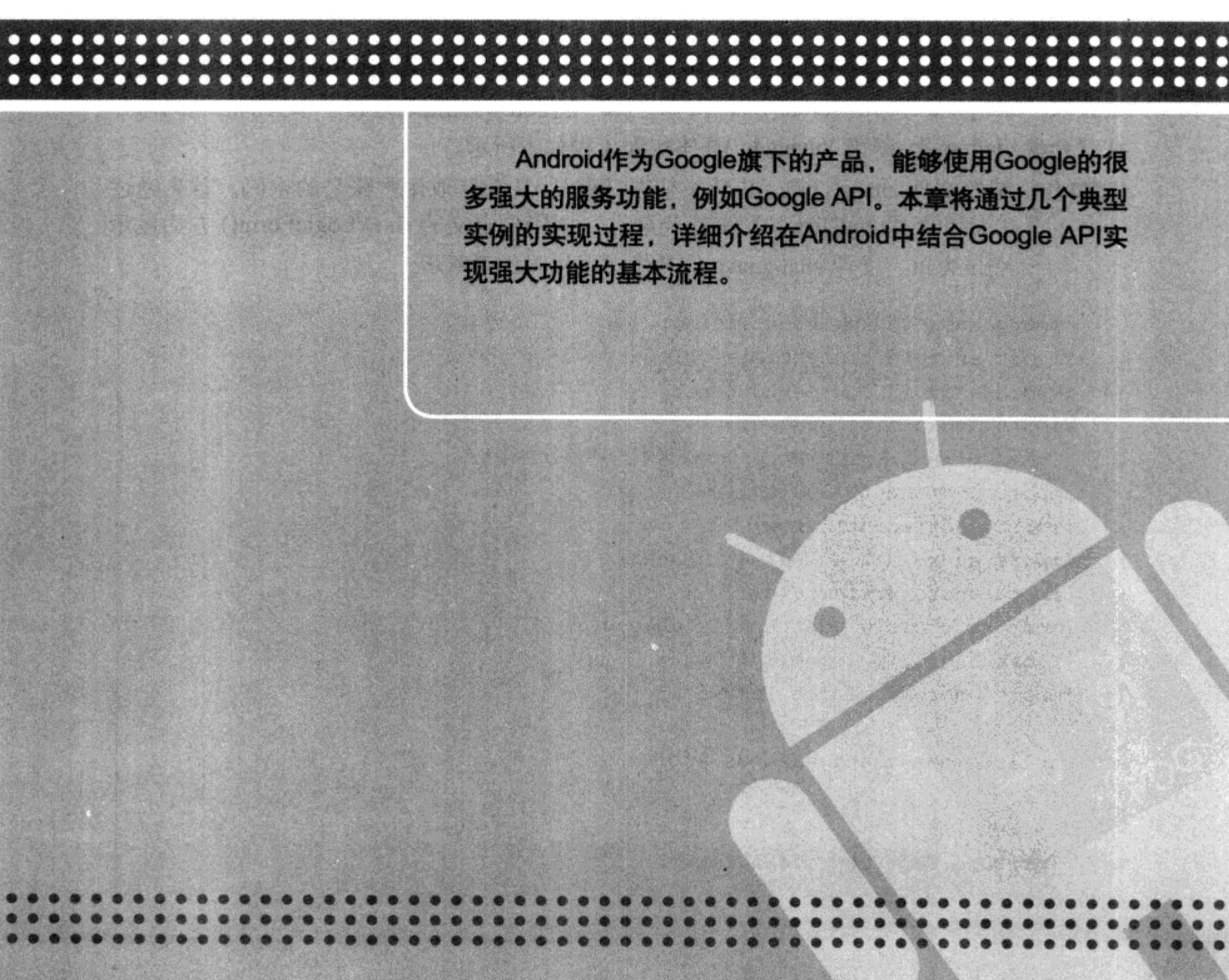
图9-23 执行效果



第 10 章

Google API实战演练

Android作为Google旗下的产品，能够使用Google的很多强大的服务功能，例如Google API。本章将通过几个典型实例的实现过程，详细介绍在Android中结合Google API实现强大功能的基本流程。



10.1 模拟验证官方账号

在现实应用中，基于 Google 的大多数服务都需要通过官方进行的账号验证。在 Google 中是通过 Google Account 实现账号验证的，主要包括如下组成部分。

- ◆ AuthSub：提供单独的 google service access，例如 gmail contact。如果你要一次使用多个 google service，比如同时需要读取 picasaweb 和 youtube 的数据，那么用 authsub 就不那么合适了，因为需要让用户登录两次 Google。
- ◆ OAuth：和 authsub 相比，提供了 sign-on，一次登录取得的 authtoken 可以 access 好几个 google services。
- ◆ Federated：本质上是 openid，它是唯一可以让你在使用 google account 登录自己网站的同时，拿到用户电子邮件的方法。
- ◆ Hybrid：集成了 oauth 和 openid，可以让你的网站同时拿到 google service 访问权限，以及用户的电子邮件。

使用 Authsub 的好处是不需要在 Google 中注册自己的网站。

在本节的内容中，将通过一个具体实例的实现过程，介绍在 Android 中通过账号验证来获取“Google Account Authentication Service”所发出的凭据的过程。本实例的源码保存在【光盘\daima\第10章\zhang】，具体实现流程如下所示。

step 01 编写文件 zhang.java，其功能是进行登录 UI 和获取用户账号的密码。首先通过 TextView 的 onClick() 事件为起点，调用自定义的 showLoginForm() 方法显示登录表单。文件 zhang.java 的具体实现代码如下所示。

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
/*必须引用util.DisplayMetrics类来取得屏幕分辨率*/
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.AbsoluteLayout;
import android.widget.EditText;
import android.widget.TextView;

public class zhang extends Activity
{
    /*声明变量*/
    private TextView mTextView01;
```



```
private LayoutInflater mInflater01;
private View mView01;
private EditText mEditText01, mEditText02;
private String TAG = "HIPPO_DEBUG";
/* 中文字的间距 */
private int intShiftPadding = 14;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /* 创建DisplayMetrics对象，取得屏幕分辨率 */
    DisplayMetrics dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);

    mTextView01 = (TextView) findViewById(R.id.myTextView1);

    /* 将文字Label放在屏幕右上方 */
    mTextView01.setLayoutParams(
        new AbsoluteLayout.LayoutParams(intShiftPadding*mTextView01.getText().toString().length(), 18, (dm.widthPixels - (intShiftPadding*mTextView01.getText().toString().length()))-10, 0));
}

/* 处理用户点击TextView文字的事件处理 - 登录 */
mTextView01.setOnClickListener(new TextView.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub

        /* 显示登录对话框 */
        showLoginForm();
    }
});

/* 自定义登录对话框函数 */
}
```

```
private void showLoginForm()
{
    try
    { /* 以LayoutInflater取得主Activity的context */
        mInflater01 = LayoutInflater.from(example1.this);
        /* 设置创建的View所要使用的Layout Resource */
        mView01 = mInflater01.inflate(R.layout.login, null);

        /* 账号EditText */
        mEditText01=(EditText)mView01.findViewById(R.id.myEditText1);

        /* 密码EditText */
        mEditText02=(EditText)mView01.findViewById(R.id.myEditText2);

        /*创建AlertDialog窗口来取得用户账号密码*/
        new AlertDialog.Builder(this)
            .setView(mView01)
            .setPositiveButton("OK",
                new DialogInterface.OnClickListener()
                {
                    /*覆盖onClick()来触发取得Token事件与完成登录事件*/
                    public void onClick(DialogInterface dialog, int whichButton)
                    {
                        if(processGoogleLogin(mEditText01.getText().toString(),
                            mEditText02.getText().toString()))
                        {
                            Intent i = new Intent();
                            /*登录后调用注销程序(zhang_01_02.java)*/
                            i.setClass(example1.this, example1_01_02.class);
                            startActivity(i);
                            finish();
                        }
                    }
                }).show();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
/*调用GoogleAuthSub来取的Google账号的Authentication Token*/
private boolean processGoogleLogin(String strUID, String strUPW)
{
```

```

try
{
    /*建构自定义的GoogleAuthSub对象*/
    GoogleAuthSub gas = new GoogleAuthSub(strUID, strUPW);
    /*取得Google Token*/
    String strAuth = gas.getAuthSubToken();
    /*将取回的Google Token写入log中*/
    Log.i(TAG, strAuth);

}
catch (Exception e)
{
    e.printStackTrace();
}
return true;
}
}

```

在上述代码中，方法 showLoginForm() 是一个使用 LayoutInflater 获取主 Activity 的 context，搭配 AlertDialog 所构建的 Login Form 表单。当用户输入账号和密码后，开始重写 DialogInterface.OnClickListener() 的 onClick() 事件来调用自定义的 processGoogleLogin 处理和 Google 账号验证的连接事件。当通过 Google 验证后取得 Google Authentication Token 后，通过 Intent 打开 zhang_01_02.java 以改变 UI 的状态。

step 02 编写文件 zhang_01_02.java，其功能是注销返回登录界面的工作。即将原来的登录状态，改为注销状态，并实现 TextView 的 onClick() 方法。当用户单击 TextView，则通过自定义的 Intent 来调用 zhang.java，返回到程序的等待状态。文件 zhang_01_02.java 的具体实现代码如下所示。

```

package irdc.zhang;

import irdc.zhang.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.widget.AbsoluteLayout;
import android.widget.TextView;

public class zhang_01_02 extends Activity
{
    private TextView mTextView03;

```



```
/* 中文字的间距 */
private int intShiftPadding = 14;

@Override
protected void onCreate(Bundle savedInstanceState)
{
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.loginok);

    /* 创建DisplayMetrics对象，取得屏幕分辨率 */
    DisplayMetrics dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);

    /*通过 findViewById() 来取得TextView对象*/
    mTextView03 = (TextView) findViewById(R.id.myTextView3);

    /* 将文字Label放在屏幕右上方 */
    mTextView03.setLayoutParams
    (
        new AbsoluteLayout.LayoutParams(intShiftPadding*mTextView03.getText().toString().length(), 18, (dm.widthPixels-(intShiftPadding*mTextView03.getText().toString().length()))-10, 0)
    );

    /* 处理用户点击TextView文字的事件处理-注销 */
    mTextView03.setOnClickListener(new TextView.OnClickListener()
    {
        /*覆盖onClick()事件*/
        @Override
        public void onClick(View v)
        {
            // TODO Auto-generated method stub
            Intent i = new Intent();
            /*注销后调用登录程序(zhang_01.java)*/
            i.setClass(zhang_01.this, zhang.class);
            startActivity(i);
            finish();
        }
    });
}

}

}*/
```

step 03 编写文件 GoogleAuthSub.java，其功能是将 HttpGet 和实际取得 Google Calendar

的服务 Token 作为示范。此文件是我们这个实例的核心，通过 Google 提供的 ClientLogin 的机制，使用HttpPost 连接到 https://www.google.com/accounts/ClientLogin，并同时将用户账号和密码及其相关参数以 Name Value Pair 字符串带入，通过自定义的 getAuth() 方法获取 Google 认证的 Authentication Token，然后模拟 Google 网络服务的 AuthSub 的方法，将自定义的 Header 和HttpGet 方法带入 Token 来获取用户 Google Calendar 服务中的所有日历数据，并以 XML 文件存储于临时文件中，作为使用 Google 服务的规范。文件 GoogleAuthSub.java 的具体实现代码如下所示。

```
package irdc.zhang;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.Header;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import android.util.Log;

public class GoogleAuthSub
{
    /*声明变量*/
    private DefaultHttpClient httpclient;
    private HttpPost httppost;
    private HttpResponse response;
    private String strGoogleAccount;
    private String strGooglePassword;
    private String TAG = "IRDC_DEBUG";
```

```
/*GoogleAuthSub对象的构造器*/
public GoogleAuthSub(String strUID, String strPWD)
{
    this.strGoogleAccount = strUID;
    this.strGooglePassword = strPWD;
    httpclient = new DefaultHttpClient();
    httppost = new HttpPost("https://www.google.com/accounts/
ClientLogin");
}

/*取得Google Token方法*/
public String getAuthSubToken()
{
    /*创建Name Value Pair字符串*/
    List <NameValuePair> nvps = new ArrayList <NameValuePair>();
    nvps.add(new BasicNameValuePair("Email", this.strGoogleAccount));
    nvps.add(new BasicNameValuePair("Passwd", this.strGooglePassword));
    nvps.add(new BasicNameValuePair("source", "MyApiV1"));
    nvps.add(new BasicNameValuePair("service", "cl"));
    String GoogleLoginAuth="";
    try
    {
        /*创建Http Post连接*/
        httppost.setEntity(new UrlEncodedFormEntity(nvps, HTTP.DEFAULT_
CONTENT_CHARSET));
        response = httpclient.execute(httppost);
        if( response.getStatusLine().getStatusCode() != 200 )
        {
            return "";
        }
        /*取回Google Token*/
        InputStream is = response.getEntity().getContent();
        GoogleLoginAuth = getAuth(is);

        /*模拟HTTP Header*/
        Header[] headers = new BasicHeader[6];
        headers[0] = new BasicHeader("Content-type", "application/x-www-
form-urlencoded");
        headers[1] = new BasicHeader("Authorization", "GoogleLogin auth=
\""+GoogleLoginAuth+"\"");
        headers[2] = new BasicHeader("User-Agent", "Java/1.5.0_06");
        headers[3] = new BasicHeader("Accept", "text/html, image/gif,
```



```
image/jpeg, *; q=.2, /*/*; q=.2");
    headers[4] = new BasicHeader("Connection", "keep-alive");
    /*发出Http Get请求登录Google Calendar服务作范例*/
    HttpGet httpget;
    String feedUrl2 = "http://www.google.com/calendar/feeds/default/
allcalendars/full";
    httpget = new HttpGet(feedUrl2);
    httpget.addHeader(headers[0]);
    httpget.addHeader(headers[1]);
    httpget.addHeader(headers[2]);
    httpget.addHeader(headers[3]);
    httpget.addHeader(headers[4]);
    /*取得Google Calendar服务应答*/
    response = httpclient.execute(httpget);
    String strTemp01 = convertStreamToString(response.getEntity() .
getContent());
    Log.i(TAG, strTemp01);
    /*指定暂存盘位置*/
    String strEarthLog = "/sdcard/googleauth.log";
    BufferedWriter bw;
    bw = new BufferedWriter (new FileWriter(strEarthLog));
    /*将取回文件写入暂存盘中*/
    bw.write( strTemp01, 0, strTemp01.length() );
    bw.flush();
}

catch (UnsupportedEncodingException e)
{
    e.printStackTrace();
}
catch (ClientProtocolException e)
{
    e.printStackTrace();
}
catch (IOException e)
{
    e.printStackTrace();
}
catch (Exception e)
{
    e.printStackTrace();
}
return GoogleLoginAuth;
```

```
}

/*自定义读取token内容的方法*/
public String getAuth(InputStream is)
{
    BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
    String line = null;
    String strAuth="";
    try
    {
        while ((line = reader.readLine()) != null)
        {
            Log.d(TAG, ":" +line);
            if( line.startsWith("Auth="))
            {
                strAuth=line.substring(5);
                Log.i("auth",":" +strAuth);
            }
        }
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
    finally
    {
        try
        {
            is.close();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
    return strAuth;
}
/*将数据转为字符串方法*/
public String convertStreamToString(InputStream is)
{
    BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;
    try
    {
        while ((line = reader.readLine()) != null)
        {
            sb.append(line);
        }
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
    return sb.toString();
}
```

```

String line = null;
try
{
    while ((line = reader.readLine()) != null)
    {
        sb.append(line);
    }
}
catch (IOException e)
{
    e.printStackTrace();
}
finally
{
    try
    {
        is.close();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}
return sb.toString();
}
}

```

执行后的效果如图 10-1 所示；单击“登录”链接后显示登录表单界面，如图 10-2 所示；输入账号和密码并单击【OK】按钮后弹出成功获取 Token 提示。如图 10-3 所示。

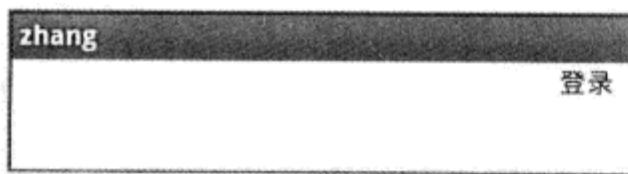


图 10-1 执行效果

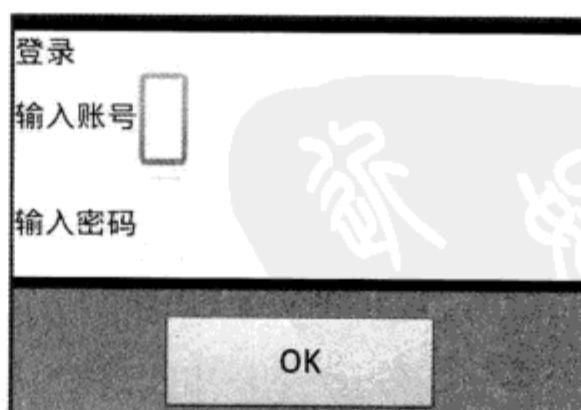


图 10-2 登录表单

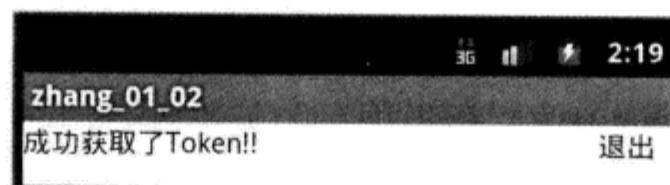


图 10-3 成功获取Token提示



10.2 实现Google搜索

在 Android 系统中，我们可以通过 Google Search API 实现搜索处理功能。使用 Google Search API 的基本流程如下所示。

step 01 构造一个搜索服务“容器”。

step 02 构造一个搜索“服务”对象。

step 03 像容器中添加“搜索服务”。

step 04 用 SearchControl 对象调用 draw() 方法，按照里面的 DrawOptions 参数画出搜索框。

本实例的源码保存在【光盘\daima\第 10 章\sousuo】，具体实现流程如下所示。

step 01 编写文件 sousuo.java 创建一个 MyAdapter 对象，此对象是自己实现的 BaseMyAdapter 类。文件 sousuo.java 的具体实现代码如下所示。

```
package irdc.sousuo;

import irdc.sousuo.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.AutoCompleteTextView;

public class sousuo extends Activity
{
    private AutoCompleteTextView myAutoCompleteTextView1;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myAutoCompleteTextView1 = (AutoCompleteTextView) findViewById(R.
id.myAutoCompleteTextView1);

        /* new一个自己实现的BaseAdapter */
        MyAdapter adapter = new MyAdapter(this);
        myAutoCompleteTextView1.setAdapter(adapter);
    }
}
```

step 02 编写文件 MyAdapter.java，在里面创建一个继承于 BaseAdapter 的 MyAdapter，可以通过覆盖 Filterable 对象中的 getFilter() 方法来对输入的关键字进行动态处理。当用户输入关键字时，performFiltering() 所返回的 FilterResults 就是查询后

的结果。文件 MyAdapter.java 的具体实现代码如下所示。

```
package irdc.sousuo;

import irdc.sousuo.R;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Filter;
import android.widget.Filterable;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MyAdapter extends BaseAdapter implements Filterable
{
    ArrayList<String> keyWithValue = new ArrayList<String>();
    ArrayList<String> resultValue = new ArrayList<String>();
    private Context mContext;
    LinearLayout.LayoutParams param1;

    public MyAdapter(Context context)
    {
        mContext = context;

        param1 = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
    }

    @Override
    public int getCount()
    {
```

```
    return keyWordValue.size();
}

@Override
public Object getItem(int position)
{
    return keyWordValue.get(position);
}

@Override
public long getItemId(int position)
{
    return position;
}

@Override
public View getView(int position, View view, ViewGroup viewGroup)
{
    LinearLayout myLinearLayout = new LinearLayout(mContext);
    myLinearLayout.setOrientation(LinearLayout.HORIZONTAL);

    if (position >= keyWordValue.size())
        return myLinearLayout;
    /* 第一个TextView放关键字结果数量 */
    TextView keyWordTextView = new TextView(this.mContext);
    keyWordTextView.setTextColor(mContext.getResources().getColor(
        R.drawable.blue));
    keyWordTextView.setTextSize(18);
    keyWordTextView.setWidth(180);
    try
    {
        keyWordTextView
            .setText(keyWordValue.get(position).toString());
    } catch (java.lang.IndexOutOfBoundsException i)
    {
        keyWordTextView.setText("");
    }
    /* 第二个TextView放关键字结果数量 */
    TextView resultTextView = new TextView(this.mContext);
    resultTextView.setTextColor(mContext.getResources().getColor(
        R.drawable.red));
    resultTextView.setTextSize(18);
    try
```

```
{  
    resultTextView.setText(resultValue.get(position).toString());  
} catch (java.lang.IndexOutOfBoundsException i)  
{  
    resultTextView.setText("");  
}  
myLinearLayout.addView(keyWordTextView, param1);  
myLinearLayout.addView(resultTextView, param1);  
  
return myLinearLayout;  
}  
  
@Override  
public Filter getFilter()  
{  
    // TODO Auto-generated method stub  
    Filter myFilter = new Filter()  
    {  
  
        @Override  
        protected FilterResults performFiltering(CharSequence text)  
        {  
  
            FilterResults fr = new FilterResults();  
            keyWordValue = new java.util.ArrayList<String>();  
            resultValue = new java.util.ArrayList<String>();  
            if (text == null || text.length() == 0)  
            {  
                fr.count = keyWordValue.size();  
                fr.values = keyWordValue;  
                return fr;  
            }  
  
            /* 输入关键字后调用google 关键字API */  
            changeResult(getGoogleAPI(text.toString()));  
  
            fr.count = keyWordValue.size();  
            fr.values = keyWordValue;  
            return fr;  
        }  
  
        @Override  
        protected void publishResults(CharSequence text,  
        FilterResults results)
```

```
        FilterResults filterResults)
    {
        if (filterResults != null && filterResults.count > 0)
            notifyDataSetChanged();
        else
            notifyDataSetChanged();
    }

};

return myFilter;
}

/* 访问GoogleAPI取得返回的结果字符串 */
private String getGoogleAPI(String text)
{
    String uri = "";
    try
    {
        /* 输入的字要encode */
        uri = "http://www.google.com/complete/"
            + "search?hl=en&js=true&qu="
            + URLEncoder.encode(text, "utf-8");
    } catch (UnsupportedEncodingException e1)
    {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    URL googleUrl = null;
    HttpURLConnection conn = null;
    InputStream is = null;
    BufferedReader in = null;
    String resultStr = "";
    /* 取得连接 */
    try
    {
        googleUrl = new URL(uri);
        /* 打开连接 */
        conn = (HttpURLConnection) googleUrl.openConnection();
        int code = conn.getResponseCode();
        /* 连接OK时 */
        if (code == HttpURLConnection.HTTP_OK)
        {
```

```
/* 取得返回的InputStream */
is = conn.getInputStream();

in = new BufferedReader(new InputStreamReader(is));
String inputLine;

/* 一行一行读取 */
while ((inputLine = in.readLine()) != null)
{
    resultStr += inputLine;
}

}

} catch (IOException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
} finally
{
    try
    {
        if (is != null)
            is.close();
        if (conn != null)
            conn.disconnect();
    } catch (Exception e)
    {
    }
}

}

return resultStr;
}

/* 处理返回的字符串变成ArrayList */
private void changeResult(String text)
{

String resultStr = "";
String startSub = "new Array(2, ";
String endSub = "), new Array";
int start = text.indexOf(startSub);
int end = text.indexOf(endSub);
if (start != -1 && end != -1)
```



```
        {
            resultStr = text.substring(start + startSub.length(), end);
            /* 去掉前后的" */
            resultStr = resultStr.substring(1, resultStr.length() - 1);
            /* 以 ", "来分隔字符串变成字符串数组 */
            String total[] = resultStr.split("\\\\\", \\\\\"");
            for (int i = 0; i < total.length / 2; i++)
            {
                keyWordValue.add(total[i * 2]);
                /* 将results字符串去掉 */
                resultValue
                    .add(total[i * 2 + 1].replaceAll(" results", ""));
            }
        }
    }
}
```

执行后的效果如图 10-4 所示。

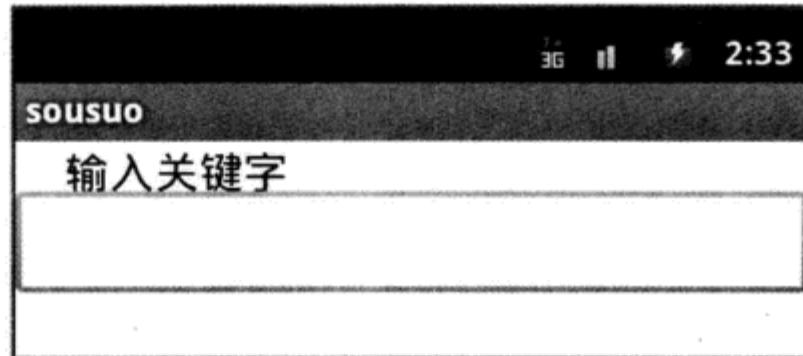


图10-4 执行效果

10.3 在手机中生成二维条码

在 Andorid 系统中，可以使用 Google 提供的 Google Chart API 生成动态二维条码。Google Chart API 为每个请求返回一个 PNG 格式图片。目前提供如下类型图表：折线图、柱状图、饼图、维恩图、散点图。用户可以设定图表尺寸、颜色和图例，可以在网页中使用 `` 元素插入图表，当浏览器打开该网页时，Chart API 提供即时图表。

所有 Chart API URL 都应使用如下格式：

```
http://chart.apis.google.com/chart?<parameter 1>&<parameter 2>&<parameter n>
```

每个 URL 所有字符必须在同一行内。

本实例的源码保存在【光盘\daima\第10章\tiaoma】，具体实现流程如下所示。

step 01 编写文件 tiaoma.java，通过自定义函数 genGoogleQRChart() 构成要显示远程图像的网址，然后以“”的方式来组成 HTML TAG。在具体成像时，通过了 WebView 来显示 HTML 的内容。文件 tiaoma.java 的具体实现代码如下所示。

```
package irdc.tiaoma;
import irdc.tiaoma.R;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.EditText;

public class tiaoma extends Activity
{
    private Button mButton01;
    private EditText mEditText01;
    private WebView mWebView01;
    private boolean bInternetConnectivity=false;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /* 测试手机是否具有连接Google API的能力 */
        if(checkInternetConnection
            ("http://code.google.com/intl/zh-TW/apis/chart/", "utf-8")
        )
        {
            bInternetConnectivity = true;
        }

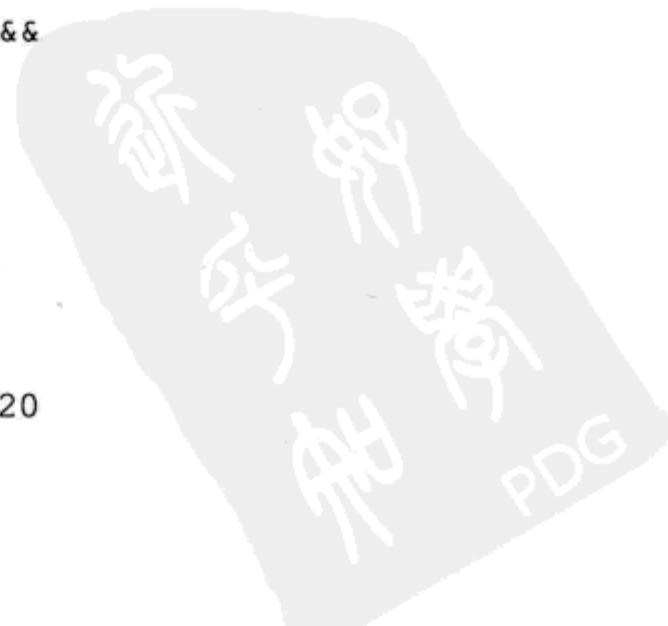
        mWebView01 = (WebView) findViewById(R.id.myWebView1);
        mButton01 = (Button) findViewById(R.id.myButton1);
    }
}
```

```
mButton01.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        if(mEditText01.getText().toString()!=""
        && bInternetConnectivity==true)
        {
            mWebView01.loadData
            (
                /* 调用自定义云端生成QR Code函数 */
                genGoogleQRChart
                (
                    mEditText01.getText().toString(),120
                    , "text/html", "utf-8"
                );
            }
        }
    });
});

mEditText01 = (EditText)findViewById(R.id.myEditText1);
mEditText01.setText(R.string.str_text);

mEditText01.setOnKeyListener(new EditText.OnKeyListener()
{
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event)
    {
        // TODO Auto-generated method stub

        if(mEditText01.getText().toString()!=""
        && bInternetConnectivity==true)
        {
            mWebView01.loadData
            (
                genGoogleQRChart
                (
                    mEditText01.getText().toString(),120
                    , "text/html", "utf-8"
                );
            }
        }
        return false;
    }
});
```



```
    }
  });
}

/* 调用Google API, 产生QR Code二维条形码 */
public String genGoogleQRChart(String strToQRCode, int strWidth)
{
  String strReturn="";
  try
  {
    strReturn = new String(strToQRCode.getBytes("utf-8"));

    /* 组成Google API需要的传输参数字符串 */
    strReturn = "<html><body>" +
    "<img src=http://chart.apis.google.com/chart?chs="+
    strWidth+"x"+strWidth+"&chl="+
    URLEncoder.encode(strReturn, "utf-8")+
    "&choe=UTF-8&cht=qr></body></html>";
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
  return strReturn;
}

/* 检查网络连接是否正常 */
public boolean checkInternetConnection
(String strURL, String strEncoding)
{
  /* 最多延时n秒若无应答则表示无法连接 */
  int intTimeout = 5;
  try
  {
    HttpURLConnection urlConnection= null;
    URL url = new URL(strURL);
    urlConnection=(HttpURLConnection)url.openConnection();
    urlConnection.setRequestMethod("GET");
    urlConnection.setDoOutput(true);
    urlConnection.setDoInput(true);
    urlConnection.setRequestProperty(
      "User-Agent", "Mozilla/4.0" +
```



```
    " (compatible; MSIE 6.0; Windows 2000)"  
);  
  
    urlConnection.setRequestProperty  
    ("Content-type", "text/html; charset="+strEncoding);  
    urlConnection.setConnectTimeout(1000*intTimeout);  
    urlConnection.connect();  
    if (urlConnection.getResponseCode() == 200)  
    {  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}  
catch (Exception e)  
{  
    e.printStackTrace();  
    return false;  
}  
}  
/* 自定义BIG5转UTF-8 */  
public String big52unicode(String strBIG5)  
{  
    String strReturn="";  
    try  
    {  
        strReturn = new String(strBIG5.getBytes("big5"), "UTF-8");  
    }  
    catch (Exception e)  
    {  
        e.printStackTrace();  
    }  
    return strReturn;  
}  
  
/* 自定义UTF-8转BIG5 */  
public String unicode2big5(String strUTF8)  
{  
    String strReturn="";  
    try  
    {
```

```
        strReturn = new String(strUTF8.getBytes("UTF-8"), "big5");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return strReturn;
}
}
```

step 02 编写布局文件 main.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@drawable/white"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <!-- 建立一个TextView -->
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="@drawable/blue"
        android:text="@string/hello"
    />
    <!-- 建立一个EditText -->
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
    />
    <!-- 建立一个WebView -->
    <WebView
        android:id="@+id/myWebView1"
        android:background="@drawable/white"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
    />
    <!-- 建立一个Button -->
    <Button
```

```
    android:id="@+id/myButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_button1"
/>
</LinearLayout>
```

执行后的效果如图 10-5 所示。



图10-5 执行效果

10.4 手机翻译

在 Android 系统中，是没有手机翻译功能的。但是 Android 官方为其提供了 API 支持，通过调用 Google Translate API 即可实现翻译功能。Google 的在线翻译功能十分强大，现在 Google 已经开放了其 Ajax 的 API。使用 Ajax 语言的 API，可以仅使用 Javascript 来翻译和检测网页中文本块的语言。此外，可以在网页的任何文本字段或文本区域启用音译。例如，如果用户已音译为北印度语，则该 API 会允许用户使用英语按照发音拼出北印度语单词，并以北印度语脚本中显示出来。

google-api-translate-java 是 Java 语言对 Google 翻译引擎的封装类库，使用方法如下所示。

```
import com.google.api.translate.Language;
import com.google.api.translate.Translate;

public class Main {
    public static void main(String[] args) {
        try {
            String translatedText =
```

```
        Translate.translate("Salut le monde", Language.FRENCH,
Language.ENGLISH);
        System.out.println(translatedText);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
}
```

本实例的源码保存在【光盘\daima\第10章\fanyi】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/myAbsoluteLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/myTextView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="English"
        android:textColor="@drawable/red"
        android:layout_x="130px"
        android:layout_y="0px"
    >
    </TextView>
    <EditText
        android:id="@+id/myEditText1"
        android:layout_width="262px"
        android:layout_height="38px"
        android:textSize="18sp"
        android:layout_x="30px"
        android:layout_y="20px"
    >
    </EditText>
    <WebView
        android:id="@+id/myWebView1"
        android:background="@drawable/white"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:focusable="false"
```



```
    android:layout_x="0px"
    android:layout_y="60px"
  />
</AbsoluteLayout>
```

step 02 编写主程序文件是 fanyi.java，其具体实现流程如下所示。

- ◆ 定义 WebSettings 对象 webSettings，用于获取 WebSettings。具体代码如下所示。

```
package irdc.fanyi;

import irdc.fanyi.R;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.webkit.JsResult;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.EditText;

public class fanyi extends Activity
{
    private EditText myEditText1;
    private WebView myWebView1;
    private Handler mHandler01 = new Handler();
    private static final String LOG_TAG = "DEBUG";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        myEditText1 = (EditText) findViewById(R.id.myEditText1);
        myWebView1 = (WebView) findViewById(R.id.myWebView1);

        /* 取得WebSettings */
        WebSettings webSettings = myWebView1.getSettings();
        /* 设置可运行JavaScript */
        webSettings.setJavaScriptEnabled(true);
        webSettings.setSaveFormData(false);
        webSettings.setSavePassword(false);
```

```

    webSettings.setSupportZoom(false);
    myWebView1.setWebChromeClient(new MyWebChromeClient());
    /* 设置给html调用的对象及名称 */
    myWebView1.addJavascriptInterface(new runJavaScript(), "irdc");
    /* 将assets/google_translate.html载入 */
    String url = "file:///android_asset/google_translate.html";
    myWebView1.loadUrl(url);
}

```

- ◆ 定义 runJavaScript，用于调用 google_translate.html 里的 javascript 以显示结果。具体代码如下所示。

```

final class runJavaScript
{
    public void runOnAndroidJavaScript()
    {
        mHandler01.post(new Runnable()
        {
            public void run()
            {
                if (myEditText1.getText().toString() != "")
                {
                    /* 调用google_translate.html里的javascript */
                    myWebView1.loadUrl("javascript:translate('"
                        + myEditText1.getText().toString() + "')");
                }
            }
        });
    }
}

```

- ◆ 定义 onJsAlert，用于捕捉网页里的 alert javascript 作为 js 调试之用，并输出至 LogCat。具体代码如下所示。

```

/**
 * 捕捉网页里的alert javascript作为.js调试之用，并输出至LogCat
 */
final class MyWebChromeClient extends WebChromeClient
{
    @Override
    public boolean onJsAlert(WebView view, String url,
                           String message, JsResult result)
    {
        // TODO Auto-generated method stub
        Log.d(LOG_TAG, message);
    }
}

```



```
// result.confirm();  
return super.onJsAlert(view, url, message, result);  
}  
}  
}
```

执行后的效果如图 10-6 所示；当输入英文字符并单击“中文”链接后，将分别弹出两个对话框，如图 10-7 和图 10-8 所示；依次单击【OK】按钮，即可实现翻译处理，如图 10-9 所示显示的翻译结果。

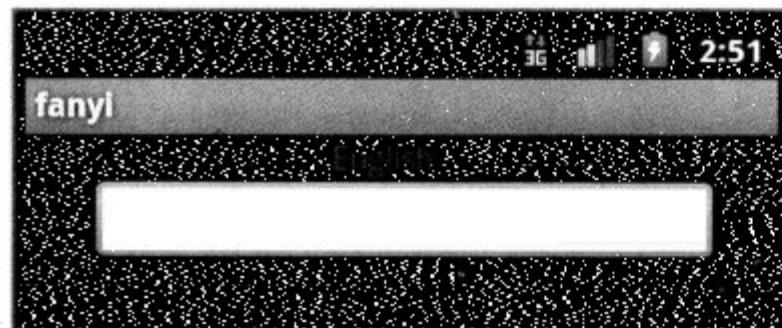


图10-6 初始效果

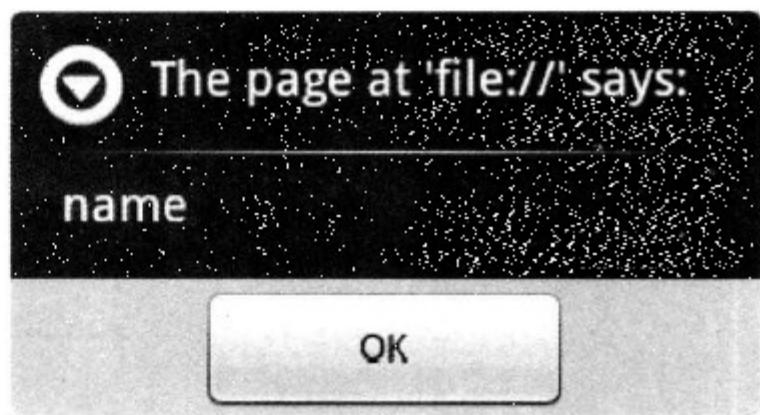


图10-7 单击【OK】按钮

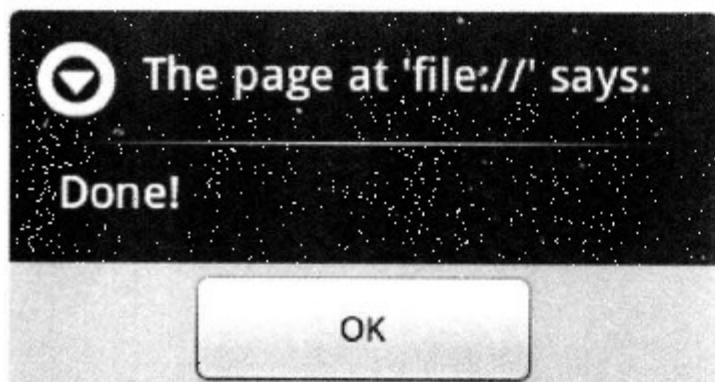


图10-8 单击【OK】按钮

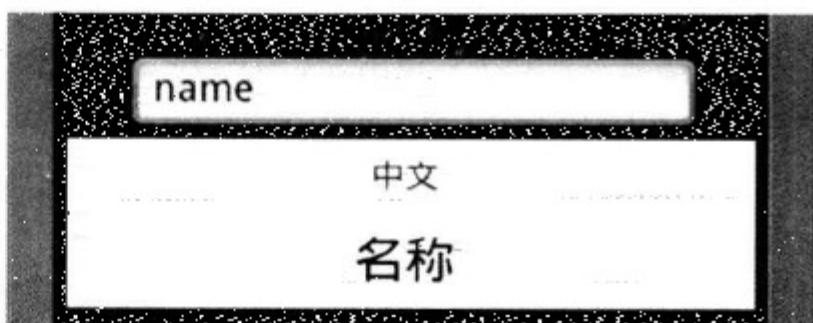
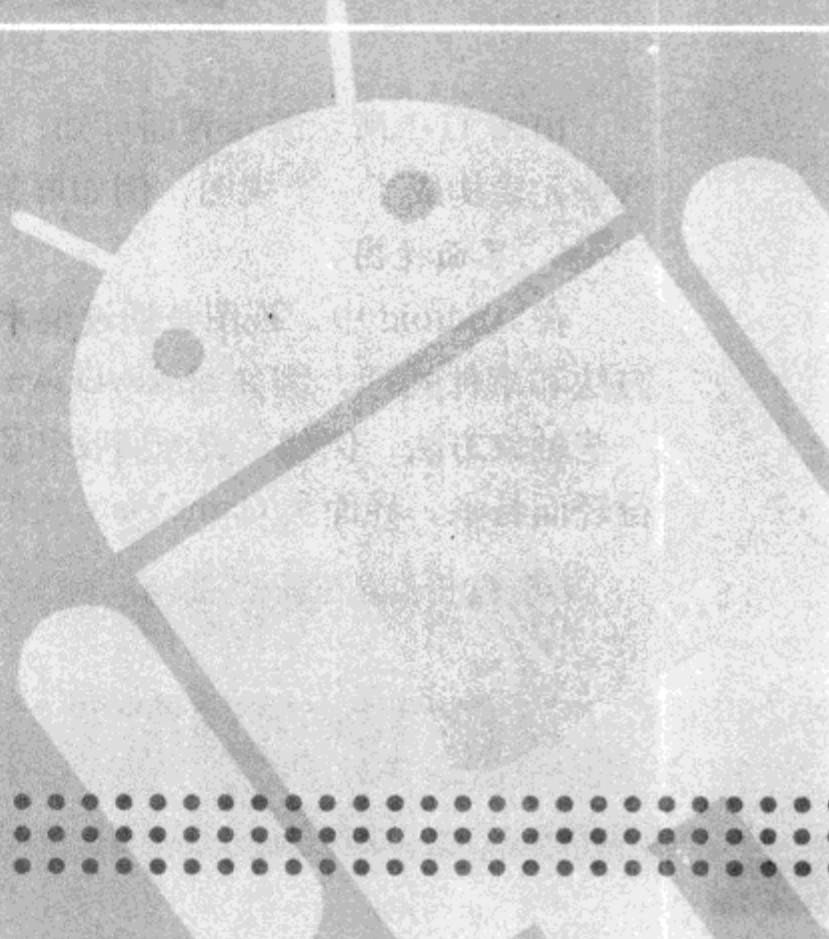


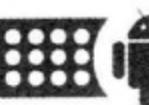
图10-9 翻译结果



游戏实战演练

自从手持设备诞生以来，游戏就成为了最重要的应用之一。无论是在旅行和上下班的路上，还是躺在家中的床上，都可以用手机游戏来打发无聊的时间。本章将通过几个典型实例的实现过程，详细介绍在 Android 系统中开发游戏项目的基本流程。





11.1 益智类游戏——魔塔

魔塔是一种固定数值的 RPG 游戏，在玩此游戏时需要动很多脑筋，任何一个轻率的选择都可能导致游戏的失败，还可以锻炼游戏者的数学能力。在本实例中，基于 Android 平台，开发了一个经典魔塔游戏。本实例的源码保存在【光盘\daima\第 11 章\mota】，具体实现流程如下所示。

(1) 设计游戏框架

因为所有游戏是基于框架的，所以设计一个合理、科学的框架尤为重要。为了使我们的框架做到完美，先看市面上魔塔游戏的界面，吸取其精华和经验，如图 11-1 所示。

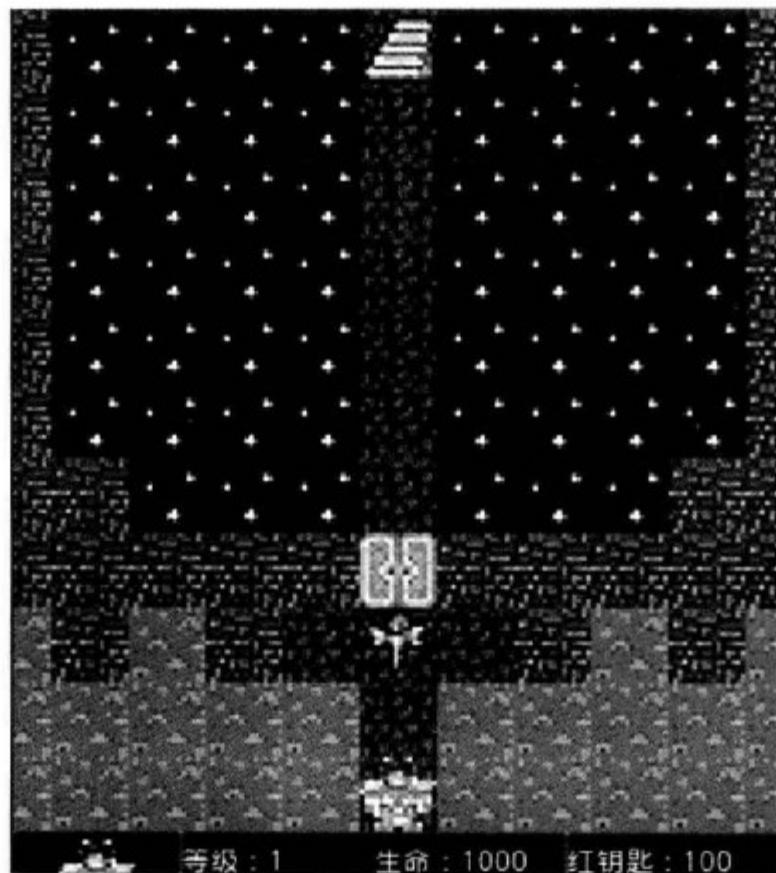


图 11-1 市面魔塔游戏界面

由图 11-1 所示游戏界面可知，在游戏中包含了地图、角色、屏幕界面、道具等元素，这些元素构成了一个视图，例如屏幕视图、道具视图、角色视图等。

◆ 界面视图

在 Android 中，视图是通过继承 View 类实现的，在 View 类中包含了各种绘制图形的方法和事件处理，例如 onKeyDown、onKeyUp 等。在构造此视图类时还可以加入自己的一些抽象方法，例如，资源回收和刷新等，有了这些内容，这样构建一个游戏界面类将变得轻而易举。界面类 GameView 类的主要实现代码如下所示。

```
public abstract class GameView extends View
{
    public GameView(Context context)
    {
```

```

        super(context);
    }
    /**绘图*/
    protected abstract void onDraw(Canvas canvas);
    /**按键按下*/
    public abstract boolean onKeyDown(int keyCode);
    /*按键弹起*/
    public abstract boolean onKeyUp(int keyCode);
    /*回收资源*/
    protected abstract void reCycle();
    /*刷新*/
    protected abstract void refurbish();
}

```

◆ 屏幕显示

前面的视图类用于显示游戏界面，我们还需要控制当前的屏幕显示哪一个界面，并且能够对界面进行一些逻辑上的处理，这就需要建立整个游戏的 MainGame 类，在此类中需要根据不同的游戏状态来设置屏幕需要显示的视图。在此需要编写类 MainGame，具体代码如下所示。

```

public class MainGame
{
    private static GameView m_GameView = null; // 当前需要显示的对象
    private Context m_Context = null;
    private MagicTower m_MagicTower = null;
    private int m_Status = -1; // 游戏状态
    public CMIDIPlayer m_CMIDIPlayer;
    public byte mbMusic = 0;
    public MainGame(Context context)
    {
        m_Context = context;
        m_MagicTower = (MagicTower) context;
        m_Status = -1;

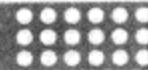
        initGame();
    }
    // 初始化游戏
    public void initGame()
    {
        controlView(yarin.GAME_SPLASH);
        m_CMIDIPlayer = new CMIDIPlayer(m_MagicTower);
    }
}

```

```
//得到游戏状态
public int getStatus()
{
    return m_status;
}
//设置游戏状态
public void setStatus(int status)
{
    m_status = status;
}
//得到主类对象
public Activity getMagicTower()
{
    return m_MagicTower;
}

//得到当前需要显示的对象
public static GameView getMainView()
{
    return m_GameView;
}
//控制显示什么界面
public void controlView(int status)
{
    if(m_status != status)
    {
        if(m_GameView != null)
        {
            m_GameView.reCycle();
            System.gc();
        }
    }
    freeGameView(m_GameView);
    switch (status)
    {
        case yarin.GAME_SPLASH:
            m_GameView = new SplashScreen(m_Context, this);
            break;
        case yarin.GAME_MENU:
            m_GameView = new MainMenu(m_Context, this);
            break;
        case yarin.GAME_HELP:
    }
}
```





```

        m_GameView = new HelpScreen(m_Context, this);
        break;
    case yarin.GAME_ABOUT:
        m_GameView = new AboutScreen(m_Context, this);
        break;
    case yarin.GAME_RUN:
        m_GameView = new GameScreen(m_Context, m_MagicTower,
this, true);
        break;
    case yarin.GAME_CONTINUE:
        m_GameView = new GameScreen(m_Context, m_MagicTower,
this, false);
        break;
    }
    setStatus(status);
}

//释放界面对象
public void freeGameView(GameView gameView)
{
    if(gameView != null)
    {
        gameView = null;
        System.gc();
    }
}
}

```

◆ 更新线程

当创建和控制视图显示以后，还需要让游戏能够动起来，此时需要线程来实现界面的自动更新。在此可以为游戏开启一个主线程，并通过方法 MainGame.getMainView() 来获取当前显示的界面，并根据不同的界面来进行游戏更新。此更能是在文件 ThreadCanvas.java 中实现，主要代码如下所示。

```

package com.mota;

public class ThreadCanvas extends View implements Runnable
{
    private String m_Tag      = "ThreadCanvas_Tag";
    public ThreadCanvas(Context context)
    {
        super(context);
    }
}

```

```
/*绘图*/
protected void onDraw(Canvas canvas)
{
    if (MainGame.getMainView() != null)
    {
        MainGame.getMainView().onDraw(canvas);
    }
    else
    {
        Log.i(m_Tag, "null");
    }
}

/*绘图显示*/
public void start()
{
    Thread t = new Thread(this);
    t.start();
}

// 刷新界面
public void refurbish()
{
    if (MainGame.getMainView() != null)
    {
        MainGame.getMainView().refurbish();
    }
}

/*游戏循环*/
public void run()
{
    while (true)
    {
        try
        {
            Thread.sleep(yarin.GAME_LOOP);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        refurbish(); // 更新显示
        postInvalidate(); // 刷新屏幕
    }
}
```

学习
笔记
第1步
进阶
篇
PDG



```
}

// 按键处理(按键按下)
boolean onKeyDown(int keyCode)
{
    if (MainGame.getMainView() != null)
    {
        MainGame.getMainView().onKeyDown(keyCode);
    }
    else
    {
        Log.i(m_Tag, "null");
    }
    return true;
}

// 按键弹起
boolean onKeyDown(int keyCode)
{
    if (MainGame.getMainView() != null)
    {
        MainGame.getMainView().onKeyUp(keyCode);
    }
    else
    {
        Log.i(m_Tag, "null");
    }
    return true;
}
}
```

◆ 显示

接下来需要调用 Activity 来显示具体的界面。因为是在 ThreadCanvas 中控制界面的，所以需要用 setContentView 来显示一个 ThreadCanvas 对象。本实例的显示功能通过文件 MagicTower.java 实现的，主要代码如下所示。

```
public class MagicTower extends Activity
{
    private ThreadCanvas mThreadCanvas = null;
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setTheme(android.R.style.Theme_Black_NoTitleBar_Fullscreen);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_
```

```
FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
        new MainGame(this);
        mThreadCanvas = new ThreadCanvas(this);
        setContentView(mThreadCanvas);
    }

/*暂停*/
protected void onPause()
{
    super.onPause();
}

/*重绘*/
protected void onResume()
{
    super.onResume();
    mThreadCanvas.requestFocus();
    mThreadCanvas.start();
}

/*按键按下*/
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    mThreadCanvas.onKeyDown(keyCode);
    return false;
}

/*按键弹起*/
public boolean onKeyUp(int keyCode, KeyEvent event)
{
    mThreadCanvas.onKeyUp(keyCode);
    return false;
}
}
```

(2) 后面的视图

经过前面的游戏框架设计后，标志着整个游戏项目的根基已经打好了。接下来需要我们在根基的基础上增砖添瓦，此魔塔游戏后面的功能也都是基于视图的，接下来将详细介绍具体实现过程。

- ◆ 设计地图

地图在游戏项目中十分重要，游戏中的所有角色都需要在某个场景中生存。但是不可能让美工制作一张巨大的地图供用户使用，因为这样的游戏将会很大，不利于在手机中有限的空间和配置中使用。通常游戏中的地图是由多个小块构成的一个完整大图，所以完全可以用一个二维数组来存储小块数据，然后通过程序将地图数据对应的小块映射到屏幕，从而组成一副完整的地图。当前主流的做法是用mappy来生成地图，然后用



脚本语言为 mappy 写一个保存格式的程序。一般情况下，地图分为 45° 角、仰视角和俯视角。

实现地图的思路非常清晰，具体流程如下所示。

- 创建一个对象类，在此命名为 TiledLayer，具体代码如下所示。

```
public class TiledLayer extends Layer

    ● 新建对应的构造函数，具体代码如下所示。

    public TiledLayer(int columns, int rows, Bitmap image, int
tileWidth,
                    int tileHeight) {
        super(columns < 1 || tileWidth < 1 ? -1 : columns *
tileWidth, rows < 1
                || tileHeight < 1 ? -1 : rows * tileHeight);
        if (((image.getWidth() % tileWidth) != 0)
            || ((image.getHeight() % tileHeight) != 0)) {
            throw new IllegalArgumentException();
        }
        this.columns = columns;
        this.rows = rows;
        cellMatrix = new int[rows][columns];
        int noOfFrames = (image.getWidth() / tileWidth)
                        * (image.getHeight() / tileHeight);
        createStaticSet(image, noOfFrames + 1, tileWidth, tileHeight,
true);
    }
```

其中参数 columns 和 rows 分别表示地图的行数和列数，image 代表地图土块中的一块，tileWidth 和 tileHeight 分别表示土块的宽度和高度。

- 定义方法 setCell(int col, int row, int tileIndex) 来设置地图使用的地图数据，此方法的具体实现代码如下所示。

```
public void setCell(int col, int row, int tileIndex) {
    if (col < 0 || col >= this.columns || row < 0 || row >= this.
rows) {
        throw new IndexOutOfBoundsException();
    }
    if (tileIndex > 0) {
        if (tileIndex >= numberOfTiles) {
            throw new IndexOutOfBoundsException();
        }
    } else if (tileIndex < 0) {
        // do animated tile index check
    }
}
```

```
        if (anim_to_static == null || (-tileIndex) >=  
numOfAnimTiles) {  
            throw new IndexOutOfBoundsException();  
        }  
    }  
    cellMatrix[row][col] = tileIndex;  
}
```

- 通过方法 createAnimatedTile(int staticTileIndex) 实现动态贴图功能，此方法在 J2ME 中十分常见。动态贴图可以通过调用这个方法来创建，该方法返回一个索引号，用于标记新创建的动态贴图。动态贴图的索引号总是负数，并且也是连续的，起始值为 -1。一旦被创建，与之关联的静态贴图可以通过调用 setAnimatedTile(int, int) 方法来改变。具体代码如下所示。

```
public int createAnimatedTile(int staticTileIndex) {  
    if (staticTileIndex < 0 || staticTileIndex >= numberoftiles) {  
        throw new IndexOutOfBoundsException();  
    }  
    if (anim_to_static == null) {  
        anim_to_static = new int[4];  
        numOfAnimTiles = 1;  
    } else if (numOfAnimTiles == anim_to_static.length) {  
        // grow anim_to_static table if needed  
        int new_anim_tbl[] = new int[anim_to_static.length * 2];  
        System.arraycopy(anim_to_static, 0, new_anim_tbl, 0,  
                         anim_to_static.length);  
        anim_to_static = new_anim_tbl;  
    }  
    anim_to_static[numOfAnimTiles] = staticTileIndex;  
    numOfAnimTiles++;  
    return -(numOfAnimTiles - 1));  
}
```

定义方法 setAnimatedTile(int animatedTileIndex, int staticTileIndex) 实现动态图块的内容，其第一个参数是动态图块的编号，第二个参数是 Tile 的编号。具体代码如下所示。

```
public void setAnimatedTile(int animatedTileIndex, int  
staticTileIndex) {  
    if (staticTileIndex < 0 || staticTileIndex >= numberoftiles) {  
        throw new IndexOutOfBoundsException();  
    }  
    animatedTileIndex = -animatedTileIndex;  
    if (anim_to_static == null || animatedTileIndex <= 0
```

```

        || animatedTileIndex >= numAnimTiles) {
    throw new IndexOutOfBoundsException();
}
anim_to_static[animatedTileIndex] = staticTileIndex;
}

```

- 使用函数 getAnimatedTile(int animatedTileIndex) 得到序号为 animatedTileIndex 的动画分块实际的图像分块序号。具体代码如下所示。

```

public int getAnimatedTile(int animatedTileIndex) {
    animatedTileIndex = -animatedTileIndex;
    if (anim_to_static == null || animatedTileIndex <= 0
        || animatedTileIndex >= numAnimTiles) {
        throw new IndexOutOfBoundsException();
    }
    return anim_to_static[animatedTileIndex];
}

```

- 使用方法 paint 将图绘制在屏幕上，具体代码如下所示。

```

public void fillCells(int col, int row, int numCols, int numRows,
                      int tileIndex) {
    if (col < 0 || col >= this.columns || row < 0 || row >= this.rows
        || numCols < 0 || col + numCols > this.columns || numRows
        < 0
        || row + numRows > this.rows) {
        throw new IndexOutOfBoundsException();
    }
    if (tileIndex > 0) {
        if (tileIndex >= numberTiles) {
            throw new IndexOutOfBoundsException();
        }
    } else if (tileIndex < 0) {
        if (anim_to_static == null || (-tileIndex) >=
            numAnimTiles) {
            throw new IndexOutOfBoundsException();
        }
    }
    for (int rowCount = row; rowCount < row + numRows; rowCount++) {
        for (int columnCount = col; columnCount < col + numCols;
            columnCount++) {
            cellMatrix[rowCount][columnCount] = tileIndex;
        }
    }
}

```

- ◆ 设计主角

在本实例游戏中只有一个主角，在魔塔游戏中被称之为“精灵”。这个“精灵”角色还有很多的动作，例如向4个方向的移动，在移动的时候就是一个动画。动画的本身是将图片一帧一帧地连接起来、循环播放每一帧所形成的。在一些大型游戏中，可以使用精灵编辑器去编写精灵，将精灵拆解为很多部分，然后再组合起来，这样可以节省大量的空间。在此笔者决定使用Sprite类实现魔塔中的主角，此类是一个用于显示图像的类。下面开始讲解主角的具体实现过程。

- ◆ 构建Sprite对象

先定义Sprite类，代码如下所示。

```
public class Sprite extends Layer
```

然后在Sprite类中提供了如下3个构造方法来构建完整的Sprite类。

- 方法Sprite(Bitmap image)的主要代码如下所示。

```
public Sprite(Bitmap image) {  
    super(image.getWidth(), image.getHeight());  
    initializeFrames(image, image.getWidth(), image.getHeight(), false);  
    initCollisionRectBounds();  
    this.setTransformImpl(TRANS_NONE);  
}
```

- 方法Sprite(Bitmap image, int frameWidth, int frameHeight)的具体实现代码如下所示。

```
public Sprite(Bitmap image, int frameWidth, int frameHeight) {  
    super(frameWidth, frameHeight);  
    if ((frameWidth < 1 || frameHeight < 1)  
        || ((image.getWidth() % frameWidth) != 0)  
        || ((image.getHeight() % frameHeight) != 0)) {  
        throw new IllegalArgumentException();  
    }  
    initializeFrames(image, frameWidth, frameHeight, false);  
    initCollisionRectBounds();  
    this.setTransformImpl(TRANS_NONE);  
}
```

- 方法Sprite(Sprite s)的具体实现代码如下所示。

```
public Sprite(Sprite s) {  
    super(s != null ? s.getWidth() : 0, s != null ? s.getHeight() : 0);  
    if (s == null) {  
        throw new NullPointerException();  
    }  
    this.sourceImage = s.sourceImage;
```

```

this.numberFrames = s.numberFrames;
this.frameCoordsX = new int[this.numberFrames];
this.frameCoordsY = new int[this.numberFrames];
System.arraycopy(s.frameCoordsX, 0, this.frameCoordsX, 0, s
    .getRawFrameCount());
System.arraycopy(s.frameCoordsY, 0, this.frameCoordsY, 0, s
    .getRawFrameCount());
this.x = s.getX();
this.y = s.getY();
this.dRefX = s.dRefX;
this.dRefY = s.dRefY;
this.collisionRectX = s.collisionRectX;
this.collisionRectY = s.collisionRectY;
this.collisionRectWidth = s.collisionRectWidth;
this.collisionRectHeight = s.collisionRectHeight;
this.srcFrameWidth = s.srcFrameWidth;
this.srcFrameHeight = s.srcFrameHeight;
setTransformImpl(s.t_currentTransformation);
this.setVisible(s.isVisible());
this.frameSequence = new int[s.getFrameSequenceLength()];
this.setFrameSequence(s.frameSequence);
this setFrame(s.getFrame());

this.setRefPixelPosition(s.getRefPixelX(), s.getRefPixelY());
}

```

在上述3个构造函数中，参数image为精灵的图片，参数frameWidth和frameHeight分别设置了精灵图片的每一帧的宽度和高度，参数s表示通过一个精灵来创建另一个精灵。构造Sprite类的时候需要指定精灵的高度和宽度（像素值）。图像的高度和宽度必须分别是精灵的高度和宽度的整数倍。换句话说，要能正好让电脑把图像按照精灵的大小划分成几个类，通过上面的例子也看到了，这些帧是横着排还是竖着排，抑或横竖都有，排成一个方阵，都无所谓。接着就可以指定帧数了，左上方是编号0，然后从左到右、从上到下依次排列。可以使用setFrame(int sequenceIndex)选择哪一帧被显示，只要把它的编号作为参数传递即可。

◆ 设置Sprite属性

因为类TiledLayer可以自动根据精灵的位置来判断地图绘制的位置，所以可以使用一个方法来设置精灵的位置。在此定义为setRefPixelPosition(int x, int y)方法，具体代码如下所示。

```

public void setRefPixelPosition(int x, int y) {
    // update this.x and this.y
    this.x = x
}

```

```
        - getTransformedPtX(dRefX, dRefY, this.t_
currentTransformation);
        this.y = y
        - getTransformedPtY(dRefX, dRefY, this.t_
currentTransformation);
    }
```

其中参数 x 和 y 是精灵的位置。

◆ 实现碰撞检测处理

碰撞即相遇，当精灵和外物相碰撞时就需要对应的处理。我的精灵类提供了以下 3 个碰撞检测函数：

- public final boolean collidesWith(TiledLayer t, boolean pixelLevel)
- public final boolean collidesWith(Sprite s, boolean pixelLevel)
- public final boolean collidesWith(Bitmap image, int x, int y, boolean pixelLevel)

使用函数 collidesWith(TiledLayer t, boolean pixelLevel) 实现精灵和 TiledLayer 的碰撞，主要代码如下所示。

```
public final boolean collidesWith(TiledLayer t, boolean
pixelLevel) {
    if (!(t.visible && this.visible)) {
        return false;
    }
    int tLx1 = t.x;
    int tLy1 = t.y;
    int tLx2 = tLx1 + t.width;
    int tLy2 = tLy1 + t.height;
    int tW = t.getCellWidth();
    int tH = t.getCellHeight();
    int sx1 = this.x + this.t_collisionRectX;
    int sy1 = this.y + this.t_collisionRectY;
    int sx2 = sx1 + this.t_collisionRectWidth;
    int sy2 = sy1 + this.t_collisionRectHeight;
    int tNumCols = t.getColumns();
    int tNumRows = t.getRows();
    int startCol; // = 0;
    int endCol; // = 0;
    int startRow; // = 0;
    int endRow; // = 0;
    if (!intersectRect(tLx1, tLy1, tLx2, tLy2, sx1, sy1, sx2, sy2)) {
        return false;
    }
    startCol = (sx1 <= tLx1) ? 0 : (sx1 - tLx1) / tW;
```



```

startRow = (sy1 <= tLy1) ? 0 : (sy1 - tLy1) / tH;
endCol = (sx2 < tLx2) ? ((sx2 - 1 - tLx1) / tW) : tNumCols - 1;
endRow = (sy2 < tLy2) ? ((sy2 - 1 - tLy1) / tH) : tNumRows - 1;
if (!pixelLevel) {
    for (int row = startRow; row <= endRow; row++) {
        for (int col = startCol; col <= endCol; col++) {
            if (t.getCell(col, row) != 0) {
                return true;
            }
        }
    }
    return false;
} else {
    if (this.t_collisionRectX < 0) {
        sx1 = this.x;
    }
    if (this.t_collisionRectY < 0) {
        sy1 = this.y;
    }
    if ((this.t_collisionRectX + this.t_collisionRectWidth) >
this.width) {
        sx2 = this.x + this.width;
    }
    if ((this.t_collisionRectY + this.t_collisionRectHeight) >
this.height) {
        sy2 = this.y + this.height;
    }
    if (!intersectRect(tLx1, tLy1, tLx2, tLy2, sx1, sy1, sx2, sy2)) {
        return (false);
    }
    startCol = (sx1 <= tLx1) ? 0 : (sx1 - tLx1) / tW;
    startRow = (sy1 <= tLy1) ? 0 : (sy1 - tLy1) / tH;
    endCol = (sx2 < tLx2) ? ((sx2 - 1 - tLx1) / tW) : tNumCols - 1;
    endRow = (sy2 < tLy2) ? ((sy2 - 1 - tLy1) / tH) : tNumRows - 1;
    int cellTop = startRow * tH + tLy1;
    int cellBottom = cellTop + tH;
    int tileIndex; // = 0;
    for (int row = startRow; row <= endRow; row++, cellTop += tH, cellBottom += tH) {
        int cellLeft = startCol * tW + tLx1;
        int cellRight = cellLeft + tW;
        for (int col = startCol; col <= endCol; col++, cellLeft += tW, cellRight += tW) {

```

```
        tileIndex = t.getCell(col, row);
        if (tileIndex != 0) {
            int intersectLeft = (sx1 < cellLeft) ? cellLeft : sx1;
            int intersectTop = (sy1 < cellTop) ? cellTop : sy1;
            int intersectRight = (sx2 < cellRight) ? sx2
                : cellRight;
            int intersectBottom = (sy2 < cellBottom) ? sy2
                : cellBottom;
            if (intersectLeft > intersectRight) {
                int temp = intersectRight;
                intersectRight = intersectLeft;
                intersectLeft = temp;
            }
            if (intersectTop > intersectBottom) {
                int temp = intersectBottom;
                intersectBottom = intersectTop;
                intersectTop = temp;
            }
            int intersectWidth = intersectRight -
                intersectLeft;
            int intersectHeight = intersectBottom -
                intersectTop;
            int image1XOffset = getImageTopLeftX(
                intersectLeft,
                intersectTop, intersectRight,
                intersectBottom);
            int image1YOffset = getImageTopLeftY(
                intersectLeft,
                intersectTop, intersectRight,
                intersectBottom);
            int image2XOffset = t.tileSetX [tileIndex]
                + (intersectLeft - cellLeft);
            int image2YOffset = t.tileSetY [tileIndex]
                + (intersectTop - cellTop);
            if (doPixelCollision(image1XOffset, image1YOffset,
                image2XOffset, image2YOffset, this.sourceImage,
                this.t_currentTransformation, t.sourceImage,
                TRANS_NONE, intersectWidth, intersectHeight)) {
                return true;
            }
        }
    }
}
```

```
        return false;
    }
}
```

使用函数 `collidesWith(Sprite s, boolean pixelLevel)` 实现精灵和精灵的碰撞，具体代码如下所示。

```
public final boolean collidesWith(Sprite s, boolean pixelLevel) {
    if (!(s.visible && this.visible)) {
        return false;
    }

    int otherLeft = s.x + s.t_collisionRectX;
    int otherTop = s.y + s.t_collisionRectY;
    int otherRight = otherLeft + s.t_collisionRectWidth;
    int otherBottom = otherTop + s.t_collisionRectHeight;
    int left = this.x + this.t_collisionRectX;
    int top = this.y + this.t_collisionRectY;
    int right = left + this.t_collisionRectWidth;
    int bottom = top + this.t_collisionRectHeight;
    if (intersectRect(otherLeft, otherTop, otherRight, otherBottom, left,
                      top, right, bottom)) {
        if (pixelLevel) {
            if (this.t_collisionRectX < 0) {
                left = this.x;
            }
            if (this.t_collisionRectY < 0) {
                top = this.y;
            }
            if ((this.t_collisionRectX + this.t_collisionRectWidth) >
                this.width) {
                right = this.x + this.width;
            }
            if ((this.t_collisionRectY + this.t_collisionRectHeight) >
                this.height) {
                bottom = this.y + this.height;
            }
            if (s.t_collisionRectX < 0) {
                otherLeft = s.x;
            }
            if (s.t_collisionRectY < 0) {
                otherTop = s.y;
            }
            if ((s.t_collisionRectX + s.t_collisionRectWidth) >
                s.width) {
```

```
        otherRight = s.x + s.width;
    }
    if ((s.t_collisionRectY + s.t_collisionRectHeight) >
s.height) {
        otherBottom = s.y + s.height;
    }
    if (!intersectRect(otherLeft, otherTop, otherRight,
            otherBottom, left, top, right, bottom)) {
        return false;
    }
    int intersectLeft = (left < otherLeft) ? otherLeft : left;
    int intersectTop = (top < otherTop) ? otherTop : top;
    int intersectRight = (right < otherRight) ? right :
otherRight;
    int intersectBottom = (bottom < otherBottom) ? bottom
            : otherBottom;
    int intersectWidth = Math.abs(intersectRight -
intersectLeft);
    int intersectHeight = Math.abs(intersectBottom -
intersectTop);
    int thisImageXOffset = getImageTopLeftX(intersectLeft,
            intersectTop, intersectRight, intersectBottom);
    int thisImageYOffset = getImageTopLeftY(intersectLeft,
            intersectTop, intersectRight, intersectBottom);
    int otherImageXOffset = s.getImageTopLeftX(intersectLeft,
            intersectTop, intersectRight, intersectBottom);
    int otherImageYOffset = s.getImageTopLeftY(intersectLeft,
            intersectTop, intersectRight, intersectBottom);
    return doPixelCollision(thisImageXOffset, thisImageYOffset,
            otherImageXOffset, otherImageYOffset, this.
sourceImage,
            this.t_currentTransformation, s.sourceImage,
            s.t_currentTransformation, intersectWidth,
            intersectHeight);
} else {
    return true;
}
}
return false;
}
```

使用函数 collidesWith(Bitmap image, int x, int y, boolean pixelLevel) 用于实现精灵和图片的碰撞。具体代码如下所示。



```
public final boolean collidesWith(Bitmap image, int x, int y,
        boolean pixelLevel) {
    if (!(this.visible)) {
        return false;
    }
    int otherLeft = x;
    int otherTop = y;
    int otherRight = x + image.getWidth();
    int otherBottom = y + image.getHeight();
    int left = this.x + this.t_collisionRectX;
    int top = this.y + this.t_collisionRectY;
    int right = left + this.t_collisionRectWidth;
    int bottom = top + this.t_collisionRectHeight;
    if (intersectRect(otherLeft, otherTop, otherRight,
otherBottom, left,
            top, right, bottom)) {
        if (pixelLevel) {
            if (this.t_collisionRectX < 0) {
                left = this.x;
            }
            if (this.t_collisionRectY < 0) {
                top = this.y;
            }
            if ((this.t_collisionRectX + this.t_collisionRectWidth) >
this.width) {
                right = this.x + this.width;
            }
            if ((this.t_collisionRectY + this.t_collisionRectHeight) >
this.height) {
                bottom = this.y + this.height;
            }
            if (!intersectRect(otherLeft, otherTop, otherRight,
otherBottom, left, top, right, bottom)) {
                return false;
            }
            int intersectLeft = (left < otherLeft) ? otherLeft : left;
            int intersectTop = (top < otherTop) ? otherTop : top;
            int intersectRight = (right < otherRight) ? right :
otherRight;
            int intersectBottom = (bottom < otherBottom) ? bottom :
otherBottom;
            int intersectWidth = Math.abs(intersectRight -
intersectLeft);
```

```
        int intersectHeight = Math.abs(intersectBottom -  
intersectTop);  
        int thisImageXOffset = getImageTopLeftX(intersectLeft,  
intersectTop, intersectRight, intersectBottom);  
        int thisImageYOffset = getImageTopLeftY(intersectLeft,  
intersectTop, intersectRight, intersectBottom);  
        int otherImageXOffset = intersectLeft - x;  
        int otherImageYOffset = intersectTop - y;  
        return doPixelCollision(thisImageXOffset,  
thisImageYOffset,  
otherImageXOffset, otherImageYOffset, this.  
sourceImage,  
this.t_currentTransformation, image, Sprite.  
TRANS_NONE,  
intersectWidth, intersectHeight);  
    } else {  
        return true;  
    }  
}  
return false;  
}
```

在上3个函数中，参数 pixelLevel 表示使用像素检测还是矩形检测。矩形检测只需要将精灵对应成相应的矩形范围来进行检查，这种检测速度很快，但不是很准确，对于碰撞要求不高的游戏可以使用它。同时还可以将一个 Sprite 分解成很多矩形来使用矩形检测以提高准确性。而像素检测则比较准确，但是速度必然会减慢。

- ◆ 实现简单的主角对话

本例设计的《魔塔》主角可以和 NPC 对话来获得一些信息，然后进行游戏。笔者准备通过一个浮动的对话框来显示对话内容，这个对话框只是一个矩形框，然后在右边绘制出对应的 NPC 的头像即可。这里的对话内容可以通过前面介绍的 TextUtil 类来实现自动换行。

- ◆ 实现精灵旋转和镜像

在游戏开发时，一般都需要使用旋转和镜像。例如做一个飞机游戏时，飞机会有几个方向的图片，这样会增加游戏开发出来的包的大小，所以可以制作一个方向的图片，然后通过精灵的旋转方法来将图片在各个方向进行旋转。这里我们只实现了 90° 的倍数旋转，分别是我们在 Sprite 类中定义的常量：TRANS_NONE、TRANS_ROT90、TRANS_ROT180、TRANS_ROT270、TRANS_MIRROR、TRANS_MIRROR_ROT90、TRANS_MIRROR_ROT180、TRANS_MIRROR_ROT270。我可以通过 setTransform 方法来传输上面这些常量，设置 Sprite 的旋转和镜像。当然，可以查看该方法的具体实现来更深入地理解 Sprite 的旋转和镜像。

◆ 实现战斗界面

当主角和怪物发生碰撞时就会发生战斗，这时需要一个界面来显示战斗效果。本实例中的战斗界面很简单，只分别显示玩家和怪物的头像以及属性（包括生命、攻击和防御）。战斗界面功能是由文件 FightScreen.java 实现的，其中绘制战斗界面功能的实现代码如下所示。

```

protected void onDraw(Canvas canvas)
{
    mcanvas = canvas;
    int tx, ty, tw, th;
    tw = yarin.SCREENW;
    th = yarin.MessageBoxH;
    tx = 0;
    ty = (yarin.SCREENH - yarin.MessageBoxH) / 2;
    showMessage();
    if (!isFighting)
    {
        tu.DrawText(mcanvas);
    }
    else
    {
        yarin.drawImage(canvas, orgeImage, 0, ty + (th - GameMap.
TILE_WIDTH) / 2, GameMap.TILE_WIDTH, GameMap.TILE_WIDTH, orgeSrcX,
orgeSrcY);
        yarin.drawImage(canvas, heroImage, (tw - GameMap.TILE_
WIDTH), ty + (th - GameMap.TILE_WIDTH) / 2, GameMap.TILE_WIDTH, GameMap.
TILE_WIDTH, 0, 0);
        paint.setColor(Color.WHITE);
        // 怪物
        {
            tx = 40;
            ty = (yarin.SCREENH - yarin.MessageBoxH) / 2 + 5;
            yarin.drawString(canvas, "生命:" + orgeHp, tx, ty,
paint);
            yarin.drawString(canvas, "攻击:" + orgeAttack, tx, ty
+ yarin.TextSize, paint);
            yarin.drawString(canvas, "防御:" + orgeDefend, tx, ty
+ 2 * yarin.TextSize, paint);
        }
        // 英雄
        {
            String string = "";
            ty = (yarin.SCREENH - yarin.MessageBoxH) / 2 + 5;
        }
    }
}

```

```
        string = hero.getHp() + ":生命";
        yarin.drawString(canvas, string, (tw - 40 - paint.
measureText(string)), ty, paint);
        string = hero.getAttack() + ":攻击";
        yarin.drawString(canvas, string, (tw - 40 - paint.
measureText(string)), ty + yarin.TextSize, paint);
        string = hero.getDefend() + ":防御";
        yarin.drawString(canvas, string, (tw - 40 - paint.
measureText(string)), ty + 2 * yarin.TextSize, paint);
    }
}
tick();
}
public void showMessage()
{
    int x = 0;
    int y = (yarin.SCREENH - yarin.MessageBoxH) / 2;
    int w = yarin.SCREENW;
    int h = yarin.MessageBoxH;
    Paint ptmPaint = new Paint();
    ptmPaint.setARGB(255, 0, 0, 0);
    yarin.fillRect(mcanvas, x, y, w, h, ptmPaint);
    ptmPaint = null;
}
```

每次战斗过后都会得到经验值，增加经验值功能的实现代码如下所示。

```
private void tick()
{
    if (orgeHp <= 0)
    {
        isFighting = false;
        tu.InitText("得到" + orgeMoney + "个金币" + "经验值增加"
" + orgeExperience, 0, (yarin.SCREENH - yarin.MessageBoxH) / 2, yarin.
SCREENW, yarin.MessageBoxH,
                0x0, 0xff0000, 255, yarin.TextSize);
    }
    else if (heroFirst == true)
    {
        orgeHp -= orgeDamagePerBout;
        if (orgeHp <= 0)
        {
            orgeHp = 0;
        }
    }
}
```

```

        else
        {
            hero.cutHp(heroDamagePerBout);
        }
        heroFirst = !heroFirst;
    }
}

```

在文件 Sprite.java 中编写函数 paint(Canvas canvas), 通过此函数将 Sprite 显示在屏幕上。主要实现代码如下所示。

```

public final void paint(Canvas canvas) {
    if (canvas == null) {
        throw new NullPointerException();
    }
    if (visible) {
        drawImage(canvas, this.x, this.y, sourceImage, frameCoordsX[frameSequence[sequenceIndex]],
                  frameCoordsY[frameSequence[sequenceIndex]], srcFrameWidth,
                  srcFrameHeight);
    }
}

```



注意 ► Sprite的编号是从0开始的，但是 TiledLayer却是从1开始的。在TiledLayer中，序号0表示一个空白的元素（比如在某个位置你什么都不想画，那就把它设置成0）。Sprite只由一个单元组成，所以如果你想让它不显示这个单元，简单地设置成 setVisible(false)就可以了，因而Sprite不需要一个特殊的编号来表示空白的单元。

◆ 游戏音效

音效在游戏开发中起了很重要的作用，在开发游戏时，人们常常忽视游戏的音效。开发者往往把主要精力花费在游戏的图像和动画等方面，而忽视了背景音乐和声音效果。这种做法是不可取的，因为好的游戏音效和音乐可以使玩家融入游戏世界，产生共鸣。音效的作用还不仅限于此。如果没有高超的游戏音效的映衬，再好的图像技巧也无法使游戏的表现摆脱平庸，对玩家也没有足够的吸引力。首先，我们将游戏中的音效分为如下几类：背景音乐、剧情音乐、音效（动作的音效、使用道具音效、辅助音效）等。背景音乐一般需要一直播放，而剧情音乐则只需要在剧情需要的时候播放，音效则是很短小的一段，比如挥刀的声音、怪物叫声等。可准备为此游戏添加两个背景音乐，一个是菜单背景音乐，一个是游戏中的背景音乐。操作流程如下所示。

- ◆ 准备两个符合游戏剧情的背景音乐放到“res\raw”目录下面。
- ◆ 创建一个 CMIDIPlayer 类，控制音乐播放。

因为，在 Android 中是通过 MediaPlayer 来播放音乐的，所以在类 CMIDIPlayer 中需



要构建一个 MediaPlayer 对象，通过 MediaPlayer.create 来装载音乐文件。上述功能的实现文件是 CMIDIPlayer.java，主要实现代码如下所示。

```
public class CMIDIPlayer
{
    public MediaPlayer           playerMusic;
    public MagicTower magicTower = null;
    public CMIDIPlayer(MagicTower magicTower)
    {
        this.magicTower = magicTower;
    }
    // 播放音乐
    public void PlayMusic(int ID)
    {
        FreeMusic();
        switch (ID)
        {
            case 1:
                //装载音乐
                playerMusic = MediaPlayer.create(magicTower, R.raw.
menu);
                //设置循环
                playerMusic.setLooping(true);
                try
                {
                    //准备
                    playerMusic.prepare();
                }
                catch (IllegalStateException e)
                {
                    e.printStackTrace();
                }
                catch (IOException e)
                {
                    e.printStackTrace();
                }
                //开始
                playerMusic.start();
                break;
            case 2:
                playerMusic = MediaPlayer.create(magicTower, R.raw.
run);
                playerMusic.setLooping(true);
        }
    }
}
```

```

        try
        {
            playerMusic.prepare();
        }
        catch (IllegalStateException e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        playerMusic.start();
        break;
    }
}
// 退出释放资源
public void FreeMusic()
{
    if (playerMusic != null)
    {
        playerMusic.stop();
        playerMusic.release();
    }
}
// 停止播放
public void StopMusic()
{
    if (playerMusic != null)
    {
        playerMusic.stop();
    }
}
}

```

在上述代码中，通过 PlayMusic 加上 ID 来确定播放什么音乐，通过 StopMusic 来停止正在播放的音乐，当程序退出时调用 FreeMusic 方法来释放播放音乐产生的资源。

◆ 创建“是否开启音效”界面

在游戏中不可能强制玩家接受要播放的音乐，所以设计一个界面来供玩家选择是否开启音乐很有必要。播放音乐代码如下所示。

```
mCMIDIPlayer.PlayMusic(1);
```

mCMIDIPlayer 是我们构建的一个 CMIDIPlayer 类的对象。在进入游戏时，又需要播



放另一首背景音乐，和上面的代码一样，只需要通过参数的 ID 来设置要播放的音乐。

- ◆ 释放资源

当退出游戏时，需要调用类 CMIDPlayer 的方法 FreeMusic 来释放资源，主要代码如下所示。

```
mCMIDIPlayer.FreeMusic();
```

在大多数游戏中，控制音效的界面也不只是这一个，可以在主菜单界面里设置音效，同样还可以在游戏中通过一个弹出菜单等，来控制音效，但是实现方式都相同，大家可以自己将其完善，尽可能地方便用户随时控制音乐开关。

执行后的界面效果如图 11-2 所示。

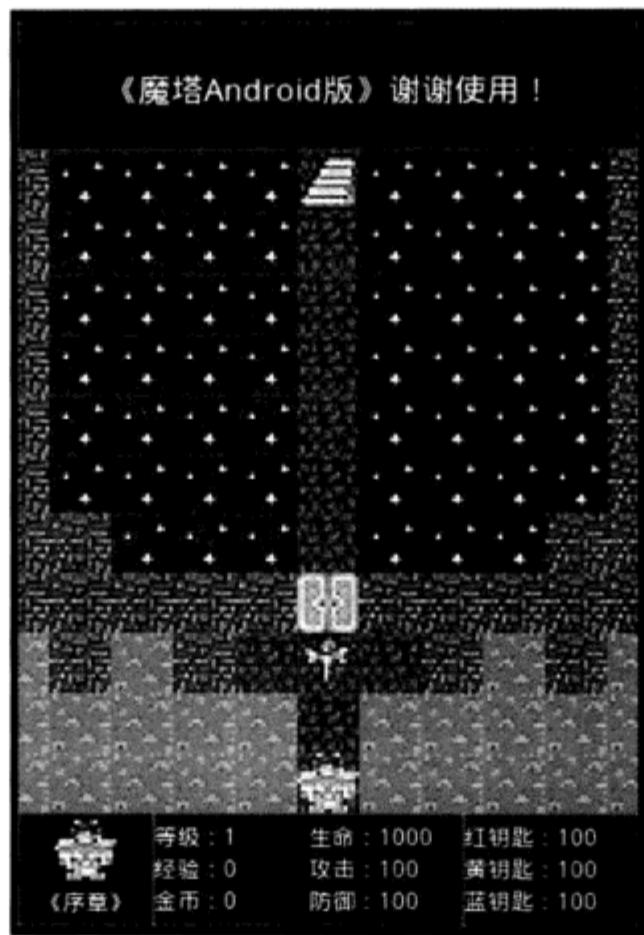


图11-2 执行效果

11.2 竞技类游戏——中国象棋

象棋，又称中国象棋，英文现译作 Xiangqi。象棋在中国有着悠久的历史，属于二人对抗性游戏的一种，由于用具简单，趣味性强，成为流行极为广泛的棋艺活动。中国象棋是我国正式开展的 78 个体育运动项目之一，为促进该项目在世界范围内的普及和推广，现将“中国象棋”项目名称更改为“象棋”。此外，高材质的象棋也具有收藏价值，例如高档木材、玉石等为材料的象棋。更有文人墨客为象棋谱写了诗篇，使象棋更具有文化色彩。本实例实现了一个中国象棋游戏，希望读者好好阅读代码，读懂每一段代码的真正功能。

本实例的源码保存在【光盘\daima\第11章\xiangqi】，具体实现流程如下所示。

step 01 编写文件 XIActivity.java，在此文件中定义了类 XIActivity，这是本实例游戏控制器类，功能是在合适的时候初始化相应的用户界面，根据其他界面的要求切换到需要的界面。文件 XIActivity.java 的主要代码如下所示。

```
public class XIActivity extends Activity {  
    boolean isSound = true; //是否播放声音  
    MediaPlayer startSound; //开始和菜单时的音乐  
    MediaPlayer gamesound; //游戏声音  
  
    Handler myHandler = new Handler() { //用来更新UI线程中的控件  
        public void handleMessage(Message msg) {  
            if (msg.what == 1) { //WelcomeView或HelpView或GameView  
                //传来的消息，切换到MenuView  
                initMenuView(); //初始化并切换到菜单界面  
            }  
            else if (msg.what == 2) { //MenuView传来的消息，切换到GameView  
                initGameView(); //初始化并切换到游戏界面  
            }  
            else if (msg.what == 3) { //MenuView传来的消息，切换到HelpView  
                initHelpView(); //初始化并切换到帮助界面  
            }  
        }  
    };  
    public void onCreate(Bundle savedInstanceState) { //重写的onCreate  
        super.onCreate(savedInstanceState);  
        //全屏  
        requestWindowFeature(Window.FEATURE_NO_TITLE);  
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,  
                            WindowManager.LayoutParams.FLAG_FULLSCREEN);  
        startSound = MediaPlayer.create(this, R.raw.startsound); //加载  
        //欢迎声音  
        startSound.setLooping(true); //设置游戏声音循环播放  
        gamesound = MediaPlayer.create(this, R.raw.gamesound); //游戏  
        //过程的背景声音  
        gamesound.setLooping(true); //设置游戏声音循环播放  
        this.initWelcomeView(); //初始化欢迎界面  
    }  
    public void initWelcomeView() { //初始化欢迎界面  
        this.setContentView(new WelcomeView(this, this)); //切换到欢迎界面  
        if (isSound) { //需要播放声音时  
            startSound.start(); //播放声音  
        }  
    }  
}
```



```
}

public void initGameView() { // 初始化游戏界面
    this.setContentView(new GameView(this, this)); // 切换到游戏界面
}

public void initMenuView() { // 初始化菜单界面
    if (startSound != null) { // 停止
        startSound.stop(); // 停止播放声音
        startSound = null;
    }
    if (this.isSound) { // 是否播放声音
        gamesound.start(); // 播放声音
    }
    this.setContentView(new MenuView(this, this)); // 切换View
}
public void initHelpView() { // 初始化帮助界面
    this.setContentView(new HelpView(this, this)); // 切换到帮助界面
}
}
```

step 02 编写文件 WelcomeView.java，在此文件中定义了类 WelcomeView，此类是一个辅助界面类，是刚进入游戏系统后显示的欢迎界面框架。文件 WelcomeView.java 的主要代码如下所示。

```
public class WelcomeView extends SurfaceView implements SurfaceHolder.Callback {
    XIActivity activity;
    private TutorialThread thread; // 刷帧的线程
    private WelcomeThread moveThread; // 物件移动的线程
    Bitmap welcomebackage; // 大背景
    Bitmap logo;
    Bitmap boy; // 小孩的图片
    Bitmap oldboy; // 老头的图片
    Bitmap bordbackground; // 文字背景
    Bitmap logo2;
    Bitmap menu; // 菜单按钮

    int logoX = -120; // 初始化需要移动的图片的相应坐标
    int boyX = -100;
    int oldboyX = -120;
    int logo2X = 320;
```

```

int bordbackgroundY = -100; //背景框的y坐标
int menuY = 520; //菜单的y坐标
public WelcomeView(Context context, XIActivity activity) { //构造器
    super(context);
    this.activity = activity; //得到activity引用
    getHolder().addCallback(this);
    this.thread = new TutorialThread(getHolder(), this); //初始化刷帧
    线程
    this.moveThread = new WelcomeThread(this); //初始化图片移动线程
    initBitmap(); //初始化所以图片
}
public void initBitmap() { //初始化所以图片
    welcomebackage = BitmapFactory.decodeResource(getResources(),
R.drawable.welcomebackage);
    logo = BitmapFactory.decodeResource(getResources(),
R.drawable.logo);
    boy = BitmapFactory.decodeResource(getResources(),
R.drawable.boy);
    oldboy = BitmapFactory.decodeResource(getResources(),
R.drawable.oldboy);
    bordbackground = BitmapFactory.decodeResource(getResources(),
R.drawable.bordbackground);
    logo2 = BitmapFactory.decodeResource(getResources(),
R.drawable.logo2);
    menu = BitmapFactory.decodeResource(getResources(),
R.drawable.menu);
}
public void onDraw(Canvas canvas) { //自己写的绘制方法，并非重写的
    //画的内容是z轴的，后画的会覆盖前面画的
    canvas.drawColor(Color.BLACK); //清屏
    canvas.drawBitmap(welcomebackage, 0, 100, null); //绘制
    welcomebackage
    canvas.drawBitmap(logo, logoX, 110, null); //绘制logo
    canvas.drawBitmap(boy, boyX, 210, null); //绘制boy
    canvas.drawBitmap(oldboy, oldboyX, 270, null); //绘制oldboy
    canvas.drawBitmap(bordbackground, 150, bordbackgroundY,
null); //绘制bordbackground
    canvas.drawBitmap(logo2, logo2X, 100, null); //绘制logo2
    canvas.drawBitmap(menu, 200, menuY, null); //绘制menu
}
public void surfaceChanged(SurfaceHolder holder, int format, int
width, int height) {
}

```

```
public void surfaceCreated(SurfaceHolder holder) { //创建时启动相应进程
    this.thread.setFlag(true); //设置循环标志位
    this.thread.start(); //启动线程

    this.moveThread.setFlag(true); //设置循环标志位
    this.moveThread.start(); //启动线程
}

public void surfaceDestroyed(SurfaceHolder holder) { //摧毁时释放相应进程
    boolean retry = true;
    thread.setFlag(false); //设置循环标志位
    moveThread.setFlag(false);
    while (retry) { //循环
        try {
            thread.join(); //等待线程结束
            moveThread.join();
            retry = false; //停止循环
        }
        catch (InterruptedException e) { //不断地循环，直到刷帧线程结束
        }
    }
}

@Override
public boolean onTouchEvent(MotionEvent event) { //屏幕监听
    if(event.getAction() == MotionEvent.ACTION_DOWN) {
        if(event.getX() > 200 && event.getX() < 200+menu.getWidth()
           && event.getY() > 355 && event.getY() < 355+menu.
getHeight()) { //点击菜单按钮
            activity.myHandler.sendMessage(Message.obtain(activity.myHandler, 1));
        }
    }
    return super.onTouchEvent(event);
}

class TutorialThread extends Thread { //刷帧线程
    private int span = 100; //睡眠的毫秒数
    private SurfaceHolder surfaceHolder; //SurfaceHolder引用
    private WelcomeView welcomeView; //WelcomeView引用
    private boolean flag = false;
    public TutorialThread(SurfaceHolder surfaceHolder, WelcomeView
welcomeView) { //构造器
        this.surfaceHolder = surfaceHolder; //得到SurfaceHolder引用
        this.welcomeView = welcomeView; //得到WelcomeView引用
    }
    public void setFlag(boolean flag) { //设置循环标记位
    }
}
```

```

        this.flag = flag;
    }

    @Override
    public void run() { //重写的run方法
        Canvas c; //画布
        while (this.flag) { //循环
            c = null;
            try {
                // 锁定整个画布，在内存要求比较高的情况下，建议参数不要为null
                c = this.surfaceHolder.lockCanvas(null);
                synchronized (this.surfaceHolder) { //同步
                    welcomeView.onDraw(c); //绘制
                }
            } finally { //使用finally语句保证下面的代码一定会被执行
                if (c != null) {
                    //更新屏幕显示内容
                    this.surfaceHolder.unlockCanvasAndPost(c);
                }
            }
            try {
                Thread.sleep(span); //睡眠指定毫秒数
            }
            catch (Exception e) { //捕获异常
                e.printStackTrace(); //打印堆栈信息
            }
        }
    }
}
}

```

step 03 编写文件 WelcomeThread.java，在此文件中定义了类 WelcomeThread，此类也是一个辅助界面类，用于生成欢迎界面的动画效果。文件 WelcomeThread.java 的主要代码如下所示。

```

public class WelcomeThread extends Thread{
    private boolean flag = true; //循环标志位
    WelcomeView welcomeView; //WelcomeView的引用
    public WelcomeThread(WelcomeView welcomeView) { //构造器
        this.welcomeView = welcomeView; //得到WelcomeView的引用
    }
    public void setFlag(boolean flag) { //设置循环标志位
        this.flag = flag;
    }
}

```

```
}

public void run() { //重写的run方法
    try {
        Thread.sleep(300); //睡眠三百毫秒，保证界面已经显示
    }
    catch (Exception e) { //捕获异常
        e.printStackTrace(); //打印异常信息
    }

    while (flag) {
        welcomeView.logoX += 10; //移动欢迎界面的logo
        if (welcomeView.logoX > 0) { //到位后停止移动
            welcomeView.logoX = 0;
        }

        welcomeView.boyX += 20; //移动小男孩图片
        if (welcomeView.boyX > 70) { //到位置后停止移动
            welcomeView.boyX = 70;
        }

        welcomeView.oldboyX += 15; //移动小老头
        if (welcomeView.oldboyX > 0) { //到位后停止移动
            welcomeView.oldboyX = 0;
        }

        welcomeView.bordbackgroundY += 50; //移动文字背景
        if (welcomeView.bordbackgroundY > 240) {
            welcomeView.bordbackgroundY = 240;
        }

        welcomeView.logo2X -= 30; //更改图片的坐标
        if (welcomeView.logo2X < 150) {
            welcomeView.logo2X = 150; //停止移动
        }

        if (welcomeView.logo2X == 150) { //当logo2到位后按钮才移动出现
            welcomeView.menuY -= 30;
            if (welcomeView.menuY < 355) {
                welcomeView.menuY = 355;
            }
        }
    }

    try {
        Thread.sleep(100); //睡眠指定毫秒数
    } catch (Exception e) { //捕获异常
        e.printStackTrace(); //打印异常信息
    }
}
}
```

step 04 编写文件 CAIMenuView.java，在此定义了类 CAIMenuView，功能是在欢迎界面单击【菜单】按钮时进入菜单界面。文件 CAIMenuView.java 的主要代码如下所示。

```

public class CAIMenuView extends SurfaceView implements SurfaceHolder.Callback {
    XIActivity activity;//总Activity的引用
    private TutorialThread thread;//刷帧的线程
    Bitmap startGame;//开始游戏图片
    Bitmap openSound;//打开声音图片
    Bitmap closeSound;//关闭声音的图片
    Bitmap help;//帮助的图片
    Bitmap exit;//退出游戏的图片
    public CAIMenuView(Context context, XIActivity activity) {//构造器
        super(context);
        this.activity = activity;//得到activity引用
        getHolder().addCallback(this);
        this.thread = new TutorialThread(getHolder(), this);//启动刷帧线程
        initBitmap();//初始化图片资源
    }
    public void initBitmap() {//初始化图片资源图片
        //开始游戏按钮
        startGame = BitmapFactory.decodeResource(getResources(),
R.drawable.startgame);
        //开始声音按钮
        openSound = BitmapFactory.decodeResource(getResources(),
R.drawable.opensound);
        closeSound = BitmapFactory.decodeResource(getResources(),
R.drawable.closesound);//关闭声音按钮
        help = BitmapFactory.decodeResource(getResources(),
R.drawable.help);//帮助按钮
        exit = BitmapFactory.decodeResource(getResources(),
R.drawable.exit);//退出按钮
    }
    public void onDraw(Canvas canvas) {//自己写的绘制方法
        canvas.drawColor(Color.BLACK);//清屏
        canvas.drawBitmap(startGame, 50, 50, null);//绘制图片
        if(activity.isSound){//放声音时，绘制关闭声音图片
            canvas.drawBitmap(closeSound, 50, 150, null);//绘制关闭声音
        }else{//没有放声音时绘制打开声音图片
            canvas.drawBitmap(openSound, 50, 150, null);//绘制开始声音
        }
        canvas.drawBitmap(help, 50, 250, null);//绘制帮助按钮
        canvas.drawBitmap(exit, 50, 350, null);//绘制退出按钮
    }
}

```



```
}

public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
}

public void surfaceCreated(SurfaceHolder holder) { //创建时启动刷帧
    this.thread.setFlag(true); //设置循环标志位
    this.thread.start(); //启动线程
}

public void surfaceDestroyed(SurfaceHolder holder) { //摧毁时释放刷帧
线程
    boolean retry = true; //循环标志位
    thread.setFlag(false); //设置循环标志位
    while (retry) { //循环
        try {
            thread.join(); //等待线程结束
            retry = false; //停止循环
        } catch (InterruptedException e) {} //不断地循环，直到刷帧线程结束
    }
}

public boolean onTouchEvent(MotionEvent event) { //屏幕监听
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        if (event.getX() > 105 && event.getX() < 220
            && event.getY() > 60 && event.getY() < 95) { //点击的是开始游戏
            activity.myHandler.sendEmptyMessage(2);
        } else if (event.getX() > 105 && event.getX() < 220
            && event.getY() > 160 && event.getY() < 195) { //点击的是声音按钮
            activity.isSound = !activity.isSound; //将声音开关取反
            if (!activity.isSound) { //当没有放声音时
                if (activity.gamesound != null) { //检查当前是否已经有声音正在播放
                    if (activity.gamesound.isPlaying()) { //当游戏声音正在播放时，
                        activity.gamesound.pause(); //停止声音的播放
                }
            }
        } else { //当需要播放声音时
            if (activity.gamesound != null) { //当gamesound不为空时
                if (!activity.gamesound.isPlaying()) { //且当前声音没有在播放
                    activity.gamesound.start(); //则播放声音
                }
            }
        }
    }
}
```

```

        }
    }

    } else if(event.getX()>105 && event.getX()<220
              &&event.getY()>260 && event.getY()<295) { //点击的是
帮助按钮
        activity.myHandler.sendEmptyMessage(3); //向activity发
送Hander消息通知切换View
    } else if(event.getX()>105 && event.getX()<220
              &&event.getY()>360 && event.getY()<395) { //点击的是
退出游戏
        System.exit(0); //直接退出游戏
    }
}

return super.onTouchEvent(event);
}

class TutorialThread extends Thread{ //刷帧线程
    private int span = 500; //睡眠的毫秒数
    private SurfaceHolder surfaceHolder; //SurfaceHolder的引用
    private CAIMenuView menuView; //MenuView的引用
    private boolean flag = false; //循环标记位
    public TutorialThread(SurfaceHolder surfaceHolder, CAIMenuView
menuView) { //构造器
        this.surfaceHolder = surfaceHolder; //得到surfaceHolder引用
        this.menuView = menuView; //得到menuView引用
    }
    public void setFlag(boolean flag) { //设置循环标记位
        this.flag = flag;
    }
    public void run() { //重写的run方法
        Canvas c; //画布
        while (this.flag) { //循环
            c = null;
            try {
                // 锁定整个画布，在内存要求比较高的情况下，建议参数不要为null
                c = this.surfaceHolder.lockCanvas(null);
                synchronized (this.surfaceHolder) { //同步锁
                    menuView.onDraw(c); //调用绘制方法
                }
            } finally { //使用finally保证下面代码一定被执行
                if (c != null) {
                    //更新屏幕显示内容
                    this.surfaceHolder.unlockCanvasAndPost(c);
                }
            }
        }
    }
}

```



```
        }
        try{
            Thread.sleep(span); //睡眠指定毫秒数
        }catch(Exception e){ //捕获异常
            e.printStackTrace(); //有异常时打印异常堆栈信息
        }
    }
}
```

step 05 编写文件 Help.java，在此定义了类 Help，这也是一个辅助界面类，功能是显示游戏系统的使用方法。文件 Help.java 的主要代码如下所示。

```
public class Help extends SurfaceView implements SurfaceHolder.Callback {
    XIActivity activity; //Activity的引用
    private TutorialThread thread; //刷帧的线程
    Bitmap back; //返回按钮
    Bitmap helpBackground; //背景图片
    public Help(Context context, XIActivity activity) { //构造器
        super(context);
        this.activity = activity; //得到activity引用
        getHolder().addCallback(this);
        this.thread = new TutorialThread(getHolder(), this); //初始化重
绘线程
    }
    public void initBitmap() { //初始化图片资源
        back = BitmapFactory.decodeResource(getResources(),
R.drawable.back); //返回按钮
        helpBackground = BitmapFactory.decodeResource(
            getResources(),
            R.drawable.helpbackground); //初始化背景图片
    }
    public void onDraw(Canvas canvas) { //自己写的绘制方法
        canvas.drawBitmap(helpBackground, 0, 90, new Paint()); //绘制背
景图片
        canvas.drawBitmap(back, 200, 370, new Paint()); //绘制按钮
    }
    public void surfaceChanged(SurfaceHolder holder, int format, int
width, int height) {}
    public void surfaceCreated(SurfaceHolder holder) { //被创建时启动刷帧
线程
    }
}
```



```

    this.thread.setFlag(true); //设置循环标志位
    this.thread.start(); //启动刷帧线程
}

public void surfaceDestroyed(SurfaceHolder holder) { //被摧毁时停止刷
    帧线程
    boolean retry = true; //循环标志位
    thread.setFlag(false); //设置循环标志位
    while (retry) {
        try {
            thread.join(); //等待线程结束
            retry = false; //停止循环
        } catch (InterruptedException e) {} //不断地循环，直到刷帧线程结束
    }
}

public boolean onTouchEvent(MotionEvent event) { //屏幕监听
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        if (event.getX() > 200 && event.getX() < 200 + back.getWidth()
            && event.getY() > 370 && event.getY() < 370 + back.
        getHeight()) { //点击了返回按钮
            activity.myHandler.sendMessage(Message.obtain(activity.myHandler, 1)); //发送Handler消息
        }
    }
    return super.onTouchEvent(event);
}

class TutorialThread extends Thread { //刷帧线程
    private int span = 1000; //睡眠的毫秒数
    private SurfaceHolder surfaceHolder; //SurfaceHolder的引用
    private Help helpView; //父类的引用
    private boolean flag = false; //循环标记位
    public TutorialThread(SurfaceHolder surfaceHolder, Help
    helpView) { //构造器
        this.surfaceHolder = surfaceHolder; //得到surfaceHolder引用
        this.helpView = helpView; //得到helpView引用
    }
    public void setFlag(boolean flag) { //设置循环标记位
        this.flag = flag;
    }
    public void run() { //重写的run方法
        Canvas c; //画布
        while (this.flag) { //循环
            c = null;
            try {
                c = this.surfaceHolder.lockCanvas(null);
            }
        }
    }
}

```



```
        synchronized (this.surfaceHolder) { //同步
            helpView.onDraw(c); //调用绘制方法
        }
    } finally { //用finally语句保证下面的代码一定会被执行
        if (c != null) { //更新屏幕显示内容
            this.surfaceHolder.unlockCanvasAndPost(c);
        }
    }
}
try{
    Thread.sleep(span); //睡眠指定毫秒数
} catch (Exception e) { //捕获异常
    e.printStackTrace(); //打印异常堆栈信息
}
}
}
```

step 06 编写文件 Game.java，此文件和前面介绍的界面辅助类不一样，在此文件中定义的 Game 类是一个核心类，功能是实现游戏界面框架。文件 Game.java 的实现流程如下所示。

- ◆ 定义继承于 SurfaceView 的类 Game，然后定义了类中需要的成员变量。主要代码如下所示。

```
public class Game extends SurfaceView implements SurfaceHolder.Callback{  
    private TutorialThread thread; //刷帧的线程  
    TimeThread timeThread ;  
    XIActivity activity; //声明Activity的引用  
    Bitmap qipan; //棋盘  
    Bitmap qizibackground; //棋子的背景图片  
    Bitmap win; //胜利的图片  
    Bitmap lost; //失败的图片  
    Bitmap ok; //确定按钮  
    Bitmap vs; //黑方红方VS的图片  
    Bitmap right; //向右的指针  
    Bitmap left; //向左的指针  
    Bitmap current; // "当前"文字  
    Bitmap exit2; //退出按钮图片  
    Bitmap sound2; //声音按钮图片  
    Bitmap sound3; //当前是否播放了声音  
    Bitmap time; //冒号  
    Bitmap redtime; //红色冒号  
    Bitmap background; //背景图片
```

```

MediaPlayer go;//下棋声音
Paint paint;//画笔
boolean caiPan = true;//是否为玩家走棋
boolean focus = false;//当前是否有选中的棋子
int selectqizi = 0; //当然选中的棋子
int startI, startJ;//记录当前棋子的开始位置
int endI, endJ;//记录当前棋子的目标位置
Bitmap[] heiZi = new Bitmap[7];//黑子的图片数组
Bitmap[] hongZi = new Bitmap[7];//红子的图片数组
Bitmap[] number = new Bitmap[10];//数字的图片数组，用于显示时间
Bitmap[] redNumber = new Bitmap[10];//红色数字的图片，用于显示时间

GuiZe guiZe;//规则类
int status = 0;//游戏状态。0游戏中，1胜利，2失败
int heiTime = 0;//黑方总共思考时间
int hongTime = 0;//红方总共思考时间

```

- ◆ 分别定义系统中的构造器和对应构造方法，主要代码如下所示。

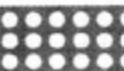
```

public Game(Context context, XIActivity activity) {//构造器
    super(context);
    this.activity = activity;//得到Activity的引用
    getHolder().addCallback(this);
    go = MediaPlayer.create(this.getContext(), R.raw.go);//加载下
    棋的声音
    this.thread = new TutorialThread(getHolder(), this);//初始化刷
    帧线程
    this.timeThread = new TimeThread(this);//初始化思考时间的线程
    init();//初始化所需资源
    guiZe = new GuiZe();//初始化规则类
}
public void init()//初始化方法
{
    paint = new Paint();//初始化画笔
    qipan = BitmapFactory.decodeResource(getResources(),
    R.drawable.qipan);//棋盘图片
    //棋子的背景
    qizibackground = BitmapFactory.decodeResource(getResources(),
    R.drawable.qizi);
    win = BitmapFactory.decodeResource(getResources(),
    R.drawable.win);//胜利的图片
    lost = BitmapFactory.decodeResource(getResources(),
    R.drawable.lost);//失败的图片
    ok = BitmapFactory.decodeResource(getResources(), R.drawable.
    ok);//确定按钮图片
}

```



```
        vs = BitmapFactory.decodeResource(getResources(), R.drawable.vs); //vs字样的图片
        right = BitmapFactory.decodeResource(getResources(), R.drawable.right); //向右的指针
        left = BitmapFactory.decodeResource(getResources(), R.drawable.left); //向左的指针
        //文字"当前"
        current = BitmapFactory.decodeResource(getResources(), R.drawable.current);
        exit2 = BitmapFactory.decodeResource(getResources(), R.drawable.exit2); //退出按钮图片
        //声音按钮图片
        sound2 = BitmapFactory.decodeResource(getResources(), R.drawable.sound2);
        time = BitmapFactory.decodeResource(getResources(), R.drawable.time); //黑色冒号
        redtime = BitmapFactory.decodeResource(getResources(), R.drawable.redtime); //红色冒号
        sound3 = BitmapFactory.decodeResource(getResources(), R.drawable.sound3);
        heiZi[0] = BitmapFactory.decodeResource(getResources(), R.drawable.heishuai); //黑帅
        heiZi[1] = BitmapFactory.decodeResource(getResources(), R.drawable.heiju); //黑车
        heiZi[2] = BitmapFactory.decodeResource(getResources(), R.drawable.heima); //黑马
        heiZi[3] = BitmapFactory.decodeResource(getResources(), R.drawable.heipao); //黑炮
        heiZi[4] = BitmapFactory.decodeResource(getResources(), R.drawable.heishi); //黑士
        heiZi[5] = BitmapFactory.decodeResource(getResources(), R.drawable.heixiang); //黑象
        heiZi[6] = BitmapFactory.decodeResource(getResources(), R.drawable.heibing); //黑兵
        hongZi[0] = BitmapFactory.decodeResource(getResources(), R.drawable.hongjiang); //红将
        hongZi[1] = BitmapFactory.decodeResource(getResources(), R.drawable.hongju); //红车
        hongZi[2] = BitmapFactory.decodeResource(getResources(), R.drawable.hongma); //红马
        hongZi[3] = BitmapFactory.decodeResource(getResources(), R.drawable.hongpao); //红炮
        hongZi[4] = BitmapFactory.decodeResource(getResources(),
```

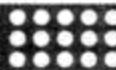


```
R.drawable.hongshi);//红仕
    hongZi[5] = BitmapFactory.decodeResource(getResources(),
R.drawable.hongxiang);//红相
    hongZi[6] = BitmapFactory.decodeResource(getResources(),
R.drawable.hongzu);//红卒
    //黑色数字0
    number[0] = BitmapFactory.decodeResource(getResources(),
R.drawable.number0);
    number[1] = BitmapFactory.decodeResource(getResources(),
R.drawable.number1);//黑色数字1
    number[2] = BitmapFactory.decodeResource(getResources(),
R.drawable.number2);//黑色数字2
    number[3] = BitmapFactory.decodeResource(getResources(),
R.drawable.number3);//黑色数字3
    number[4] = BitmapFactory.decodeResource(getResources(),
R.drawable.number4);//黑色数字4
    number[5] = BitmapFactory.decodeResource(getResources(),
R.drawable.number5);//黑色数字5
    number[6] = BitmapFactory.decodeResource(getResources(),
R.drawable.number6);//黑色数字6
    number[7] = BitmapFactory.decodeResource(getResources(),
R.drawable.number7);//黑色数字7
    number[8] = BitmapFactory.decodeResource(getResources(),
R.drawable.number8);//黑色数字8
    number[9] = BitmapFactory.decodeResource(getResources(),
R.drawable.number9);//黑色数字9
    redNumber[0] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber0);//红色数字0
    redNumber[1] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber1);//红色数字1
    redNumber[2] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber2);//红色数字2
    redNumber[3] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber3);//红色数字3
    redNumber[4] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber4);//红色数字4
    redNumber[5] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber5);//红色数字5
    redNumber[6] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber6);//红色数字6
    redNumber[7] = BitmapFactory.decodeResource(getResources(),
R.drawable.rednumber7);//红色数字7
    redNumber[8] = BitmapFactory.decodeResource(getResources(),
```

```
R.drawable.rednumber8); //红色数字8  
redNumber[9] = BitmapFactory.decodeResource(getResources(),  
R.drawable.rednumber9); //红色数字9  
background = BitmapFactory.decodeResource(getResources(),  
R.drawable.bacnground);  
}
```

- ◆ 定义绘制方法 onDraw，该方法是自己定义的并非重写的，只会根据数据绘制屏幕。主要代码如下所示。

```
public void onDraw(Canvas canvas) { //自己写的绘制方法  
    canvas.drawColor(Color.WHITE);  
    canvas.drawBitmap(background, 0, 0, null); //清背景  
    canvas.drawBitmap(qiPan, 10, 10, null); //绘制棋盘  
    for (int i=0; i<qizi.length; i++) {  
        for (int j=0; j<qizi[i].length; j++) { //绘制棋子  
            if (qizi[i][j] != 0) {  
                canvas.drawBitmap(qizibackground, 9+j*34, 10+i*35,  
null); //绘制棋子的背景  
                if (qizi[i][j] == 1) { //为黑帅时  
                    canvas.drawBitmap(heiZi[0], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 2) { //为黑车时  
                    canvas.drawBitmap(heiZi[1], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 3) { //为黑马时  
                    canvas.drawBitmap(heiZi[2], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 4) { //为黑炮时  
                    canvas.drawBitmap(heiZi[3], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 5) { //为黑士时  
                    canvas.drawBitmap(heiZi[4], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 6) { //为黑象时  
                    canvas.drawBitmap(heiZi[5], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 7) { //为黑兵时  
                    canvas.drawBitmap(heiZi[6], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 8) { //为红将时  
                    canvas.drawBitmap(hongZi[0], 12+j*34, 13+i*35, paint);  
                }  
                else if (qizi[i][j] == 9) { //为红车时  
                    canvas.drawBitmap(hongZi[1], 12+j*34, 13+i*35, paint);  
                }  
            }  
        }  
    }  
}
```



```
        canvas.drawBitmap(hongZi[1], 12+j*34, 13+i*35, paint);
    }
    else if(qizi[i][j] == 10){//为红马时
        canvas.drawBitmap(hongZi[2], 12+j*34, 13+i*35, paint);
    }
    else if(qizi[i][j] == 11){//为红炮时
        canvas.drawBitmap(hongZi[3], 12+j*34, 13+i*35, paint);
    }
    else if(qizi[i][j] == 12){//为红仕时
        canvas.drawBitmap(hongZi[4], 12+j*34, 13+i*35, paint);
    }
    else if(qizi[i][j] == 13){//为红相时
        canvas.drawBitmap(hongZi[5], 12+j*34, 13+i*35, paint);
    }
    else if(qizi[i][j] == 14){//为红卒时
        canvas.drawBitmap(hongZi[6], 12+j*34, 13+i*35, paint);
    }
}
}

canvas.drawBitmap(vs, 10, 360, paint);//绘制vs背景图
//绘制黑方的时间
canvas.drawBitmap(time, 81, 411, paint);//绘制冒号
int temp = this.heiTime/60;//换算时间
String timeStr = temp+"";//转换成字符串
if(timeStr.length()<2){//当不足两位时前面填0
    timeStr = "0" + timeStr;
}
for(int i=0;i<2;i++){//循环绘制时间
    int tempScore=timeStr.charAt(i)-'0';
    canvas.drawBitmap(number[tempScore], 65+i*7, 412, paint);
}
//画分钟
temp = this.heiTime%60;
timeStr = temp+"";//转换成字符串
if(timeStr.length()<2){
    timeStr = "0" + timeStr;//当长度小于2时在前面添加一个0
}
for(int i=0;i<2;i++){//循环
    int tempScore=timeStr.charAt(i)-'0';
    canvas.drawBitmap(number[tempScore], 85+i*7, 412, paint);//绘制
}
//开始绘制红方时间
```



```
    canvas.drawBitmap(this.redtime, 262, 410, paint); //红方的冒号
    int temp2 = this.hongTime/60; //换算时间
    String timeStr2 = temp2+""; //转换成字符串
    if(timeStr2.length()<2) { //当不足两位时前面填0
        timeStr2 = "0" + timeStr2;
    }
    for(int i=0;i<2;i++) { //循环绘制时间
        int tempScore=timeStr2.charAt(i)-'0';
        canvas.drawBitmap(redNumber[tempScore], 247+i*7, 411,
paint); //绘制
    }
    //画分钟
    temp2 = this.hongTime%60; //求出当前的秒数
    timeStr2 = temp2+""; //转换成字符串
    if(timeStr2.length()<2) { //不足两位时前面用0补
        timeStr2 = "0" + timeStr2;
    }
    for(int i=0;i<2;i++) { //循环绘制
        int tempScore=timeStr2.charAt(i)-'0';
        canvas.drawBitmap(redNumber[tempScore], 267+i*7, 411,
paint); //绘制时间数字
    }
    if(caiPan == true) { //当该玩家走棋时，即红方走棋
        canvas.drawBitmap(right, 155, 420, paint); //绘制向右的指针
    }
    else { //黑方走棋，即电脑走棋时
        canvas.drawBitmap(left, 120, 420, paint); //绘制向左的指针
    }

    canvas.drawBitmap(current, 138, 445, paint); //绘制当前文字
    canvas.drawBitmap(sound2, 10, 440, paint); //绘制声音
    if(activity.isPlaying) { //如果正在播放声音
        canvas.drawBitmap(sound3, 80, 452, paint); //绘制
    }

    canvas.drawBitmap(exit2, 250, 440, paint); //绘制退出按钮
    if(status == 1) { //当胜利时
        canvas.drawBitmap(win, 85, 150, paint); //绘制胜利图片
        canvas.drawBitmap(ok, 113, 240, paint);
    }
    if(status == 2) { //失败后
        canvas.drawBitmap(lost, 85, 150, paint); //绘制失败界面
        canvas.drawBitmap(ok, 113, 236, paint);
    }
```

```

    }
}

```

- ◆ 定义重写的屏幕监听方法 onTouchEvent，该方法是游戏主要逻辑接口，用于接受玩家输入。它会根据点击的位置和当前的游戏状态做出相应的处理，当需要切换 View 界面时，通过给 Activity 发送 Handler 消息来处理。

```

public boolean onTouchEvent(MotionEvent event) {
    if(event.getAction() == MotionEvent.ACTION_DOWN){ //只取鼠标按下的事件
        if(event.getX()>10&&event.getX()<10+sound2.getWidth()
        && event.getY()>440 && event.getY()<440+sound2.getHeight()){//按下了声音
            activity.isSound = !activity.isSound;//声音取反
            if(activity.isSound){ //当需要放声音时
                if(activity.gamesound != null){ //gamesound不为空时
                    if(!activity.gamesound.isPlaying()){//当前没有音乐时
                        activity.gamesound.start(); //播放音乐
                    }
                }
            }
            else{
                if(activity.gamesound != null){ //gamesound不为空时
                    if(activity.gamesound.isPlaying()){//当前有音乐时
                        activity.gamesound.pause(); //停止音乐
                    }
                }
            }
        }//end 按下了声音按钮
        if(event.getX()>250&&event.getX()<250+exit2.getWidth()
        && event.getY()>440 && event.getY()<440+exit2.getHeight()){//按下了退出
            activity.myHandler.sendEmptyMessage(1); //发送消息，切换到MenuView
        }
        if(status == 1){ //胜利后
            if(event.getX()>135&&event.getX()<190
            && event.getY()>249 && event.getY()<269){ //点击了确定按钮
                activity.myHandler.sendEmptyMessage(1); //发送消息，切换到MenuView
            }
        }
        else if(status == 2){ //失败后
            if(event.getX()>135&&event.getX()<190
            && event.getY()>245 && event.getY()<265){ //点击了确定按钮
                activity.myHandler.sendEmptyMessage(1); //发送消息，切换到MenuView
            }
        }
    }
}

```

```
    }
}

else if(status == 0){//游戏中时
    if(event.getX()>10&&event.getX()<310
        && event.getY()>10 && event.getY()<360){//点击的位置在棋盘内时
        if(caiPan == true){//如果是该玩家走棋
            int i = -1, j = -1;
            int[] pos = getPos(event);//根据坐标换算成所在的行和列
            i = pos[0];
            j = pos[1];
            if(focus == false){//之前没有选中的棋子
                if(qizi[i][j] != 0){//点击的位置有棋子
                    if(qizi[i][j] > 7){//点击的是自己的棋子。即下面的黑色棋子
                        selectqizi = qizi[i][j];//将该棋子设为选中的棋子
                        focus = true;//标记当前有选中的棋子
                        startI = i;
                        startJ = j;
                    }
                }
            }
            else{//之前选中过棋子
                if(qizi[i][j] != 0){//点击的位置有棋子
                    if(qizi[i][j] > 7){//如果是自己的棋子.
                        selectqizi = qizi[i][j];//将该棋子设为选中的棋子
                        startI = i;
                        startJ = j;
                    }
                }
                else{//如果是对方的棋子
                    endI = i;
                    endJ = j;//保存该点
                    boolean canMove = guiZe.canMove(qizi, startI, startJ, endI, endJ);
                    if(canMove){//如果可以移动过去
                        caiPan = false;//不让玩家走了
                        if(qizi[endI][endJ] == 1 || qizi[endI][endJ] == 8){//如果是"帅"或"将"
                            this.success();//胜利了
                        }
                    }
                }
            }
            if(activity.isSound){
                go.start();//播放下棋声音
            }
            qizi[endI][endJ] = qizi[startI][startJ];//移动棋子
            qizi[startI][startJ] = 0;//将原来处设空
            startI = -1;
        }
    }
}
```



```

startJ = -1;
endI = -1;
endJ = -1;//还原保存点
focus = false;//标记当前没有选中棋子

ChessMove cm = guiZe.searchAGoodMove(qizi); //根据当前局势查询一个最好的走法
if(activity.isSound) {
    go.start(); //播放下棋声音
}
qizi[cm.toX][cm.toY] = qizi[cm.fromX][cm.fromY]; //移动棋子
qizi[cm.fromX][cm.fromY] = 0;
caiPan = true; //恢复玩家响应
}
}
}

} //end点击的位置有棋子
else { //如果点击的位置没有棋子
    endI = i;
    endJ = j;
    boolean canMove = guiZe.canMove(qizi, startI, startJ, endI, endJ); //查看是否可走
    if(canMove) { //如果可以移动
        caiPan = false; //不让玩家走了
        if(activity.isSound) {
            go.start(); //播放下棋声音
        }
        qizi[endI][endJ] = qizi[startI][startJ]; //移动棋子
        qizi[startI][startJ] = 0; //将原来处置空
        startI = -1;
        startJ = -1;
        endI = -1;
        endJ = -1; //还原保存点
        focus = false; //标志位设false
    }

    ChessMove cm = guiZe.searchAGoodMove(qizi); //得到一步走法
    if(qizi[cm.toX][cm.toY] == 8) { //电脑吃了您的将
        status = 2; //切换游戏状态为失败
    }
    if(activity.isSound) { //需要播放声音时
        go.start(); //播放下棋声音
    }
    qizi[cm.toX][cm.toY] = qizi[cm.fromX][cm.fromY]; //移动棋子
    qizi[cm.fromX][cm.fromY] = 0;
}

```



```
    caiPan = true; //恢复玩家响应
}
}
}//end 之前选中过棋子
}
}//end点击的位置在棋盘内时
}//end游戏中时
}
return super.onTouchEvent(event);
}
```

- ◆ 定义方法 getPo，用于将坐标换算成数组的维数。主要代码如下所示。

```
public int[] getPos(MotionEvent e) {
    int[] pos = new int[2];
    double x = e.getX(); //得到点击位置的x坐标
    double y = e.getY(); //得到点击位置的y坐标
    if (x > 10 && y > 10 && x < 10 + qipan.getWidth() && y < 10 + qipan.getHeight()) { //点击的是棋盘时
        pos[0] = Math.round((float) ((y - 21) / 36)); //取得所在的行
        pos[1] = Math.round((float) ((x - 21) / 35)); //取得所在的列
    } else { //点击的位置不是棋盘时
        pos[0] = -1; //将位置设为不可用
        pos[1] = -1;
    }
    return pos; //将坐标数组返回
}

public void success() { //胜利了
    status = 1; //切换到胜利状态
}

public void surfaceChanged(SurfaceHolder holder, int format, int width,
                           int height) {

}

public void surfaceCreated(SurfaceHolder holder) { //重写的
    this.thread.setFlag(true);
    this.thread.start(); //启动刷帧线程
    timeThread.setFlag(true);
    timeThread.start(); //启动思考时间的线程
}
```

step 07 编写文件 Move.java，在此文件中定义了象棋的走法类 Move，在走法中包含了什么棋子、起始点的位置、目标点的位置以及估值时所用到的 score。文件 Move.java 的主要代码如下所示。

```
package com.xiangqi;
public class Move {
    int ChessID;//表明是什么棋子
    int fromX;//起始的坐标
    int fromY;
    int toX;//目的地的坐标
    int toY;
    int score;//值，估值时会用到
    public Move(int ChessID, int fromX, int fromY, int toX, int toY, int score) {//构造器
        this.ChessID = ChessID;//棋子的类型
        this.fromX = fromX;//棋子的起始坐标
        this.fromY = fromY;
        this.toX = toX;//棋子的目标点x坐标
        this.toY = toY;//棋子的目标点y坐标
        this.score = score;
    }
}
```

step 08 编写文件 TimeThread.java，在此文件中定义了类 TimeThread，此类能够根据是哪一方该走字，并增加这一方的思考时间。文件 TimeThread.java 的主要代码如下所示。

```
public class TimeThread extends Thread{
    private boolean flag = true;//循环标志
    Game gameView;
    public TimeThread(Game gameView) {//构造器
        this.gameView = gameView;//得到GameView引用
    }
    public void setFlag(boolean flag) {//设置循环标记位
        this.flag = flag;
    }
    @Override
    public void run() {//重写的run方法
        while(flag) {//循环
            if(gameView.caiPan == false) {//当前为黑方走棋、思考
                gameView.heiTime++; //黑方时间自加
            }
            else if(gameView.caiPan == true) {//当前为红方走棋、思考
                gameView.hongTime++; //红方时间自加
            }
        }
    }
}
```



```
        }
        try{
            Thread.sleep(1000); //睡眠一秒种
        }
        catch(Exception e){ //捕获异常
            e.printStackTrace(); //打印异常信息
        }
    }
}
```

step 09 编写文件 GuiZe.java，在里面定义了象棋规则类 GuiZe。我们知道象棋是有规则的，例如马走日，象走田。主要代码如下所示。

```
public class GuiZe {
    boolean isRedGo = false; //是不是红方走棋
    public boolean canMove(int[][] qizi, int fromY, int fromX, int
toY, int toX){
        int i = 0;
        int j = 0;
        int moveChessID; //起始位置是什么棋子
        int targetID; //目的地是什么棋子或空地
        if(toX<0){ //当左边出界时
            return false;
        }
        if(toX>8){ //当右边出界时
            return false;
        }
        if(toY<0){ //当上边出界时
            return false;
        }
        if(toY>9){ //当下边出界时
            return false;
        }
        if(fromX==toX && fromY==toY){ //目的地与出发点相同,
            return false;
        }
        moveChessID = qizi[fromY][fromX]; //得到起始棋子
        targetID = qizi[toY][toX]; //得带终点棋子
        if(isSameSide(moveChessID, targetID)){ //如果是同一阵营的
            return false;
        }
        switch(moveChessID){
            case 1://黑帅
```

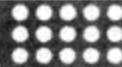


```

        if(toY>2||toX<3||toX>5){//出了九宫格
            return false;
        }
        if((Math.abs(fromY-toY)+Math.abs(toX-fromX))>1){//只能走一步
            return false;
        }
        break;
    case 5://黑士
        if(toY>2||toX<3||toX>5){//出了九宫格
            return false;
        }
        if(Math.abs(fromY-toY) != 1 || Math.abs(toX-fromX) != 1){ //走斜线
            return false;
        }
        break;
    case 6://黑象
        if(toY>4){//不能过河
            return false;
        }
        if(Math.abs(fromX-toX) != 2 || Math.abs(fromY-toY) != 2){ //相走"田"字
            return false;
        }
        if(qizi[((fromY+toY)/2)][((fromX+toX)/2)] != 0){
            return false;//相眼处有棋子
        }
        break;
    case 7://黑兵
        if(toY < fromY){//不能回头
            return false;
        }
        if(fromY<5 && fromY == toY){//过河前只能直走
            return false;
        }
        if(toY - fromY + Math.abs(toX-fromX) > 1){ //只能走一步，并且是直线
            return false;
        }
        break;
    case 8://红将
        if(toY<7||toX>5||toX<3){//出了九宫格
            return false;
        }
        break;
}

```

```
        return false;
    }
    if((Math.abs(fromY-toY)+Math.abs(toX-fromX))>1){//只能走一步
        return false;
    }
    break;
case 2://黑车
case 9://红车
    if(fromY != toY && fromX != toX){//只能走直线
        return false;
    }
    if(fromY == toY){//走横线
        if(fromX < toX){//向右走
            for(i = fromX + 1; i < toX; i++){//循环
                if(qizi[fromY][i] != 0){
                    return false;//返回false
                }
            }
        }
        else{//向左走
            for(i = toX + 1; i < fromX; i++){//循环
                if(qizi[fromY][i] != 0){
                    return false;//返回false
                }
            }
        }
    }
    else{//走的是竖线
        if(fromY < toY){//向右走
            for(j = fromY + 1; j < toY; j++){
                if(qizi[j][fromX] != 0)
                    return false;//返回false
            }
        }
        else{//向左走
            for(j= toY + 1; j < fromY; j++){
                if(qizi[j][fromX] != 0)
                    return false;//返回false
            }
        }
    }
}
```



```
        }
        break;
    case 10://红马
    case 3://黑马
        if (!(Math.abs(toX - fromX) == 1 && Math.abs(toY - fromY) == 2)
            || (Math.abs(toX - fromX) == 2 && Math.abs(toY - fromY) == 1))) {
            return false;//马走的不是日字
        }
        if (toX - fromX == 2) {//向右走
            i = fromX + 1;//移动
            j = fromY;
        }
        else if (fromX - toX == 2) {//向左走
            i = fromX - 1;//移动
            j = fromY;
        }
        else if (toY - fromY == 2) {//向下走
            i = fromX;//移动
            j = fromY + 1;
        }
        else if (fromY - toY == 2) {//向上走
            i = fromX;//移动
            j = fromY - 1;
        }
        if (qizi[j][i] != 0)
            return false;//绊马腿
        break;
    case 11://红炮
    case 4://黑炮
        if (fromY != toY && fromX != toX) {//炮走直线
            return false;//返回false
        }
        if (qizi[toY][toX] == 0) {//不吃子时
            if (fromY == toY) {//横线
                if (fromX < toX) {//向右走
                    for (i = fromX + 1; i < toX; i++) {
                        if (qizi[fromY][i] != 0) {
                            return false;//返回false
                        }
                    }
                }
            }
        }
    }
```

```
        else{//向走走
            for(i = toX + 1; i < fromX; i++) {
                if(qizi[fromY][i] != 0) {
                    return false;//返回false
                }
            }
        }

        else{//竖线
            if(fromY < toY){//向下走
                for(j = fromY + 1; j < toY; j++) {
                    if(qizi[j][fromX] != 0) {
                        return false;//返回false
                    }
                }
            }
            else{//向上走
                for(j = toY + 1; j < fromY; j++) {
                    if(qizi[j][fromX] != 0) {
                        return false;//返回false
                    }
                }
            }
        }

        else{//吃子时
            int count=0;
            if(fromY == toY){//走的是横线
                if(fromX < toX){//向右走
                    for(i=fromX+1;i<toX;i++) {
                        if(qizi[fromY][i] != 0) {
                            count++;
                        }
                    }
                    if(count != 1) {
                        return false;//返回false
                    }
                }
                else{//向左走
                    for(i=toX+1;i<fromX;i++) {
                        if(qizi[fromY][i] != 0) {
                            count++;
                        }
                    }
                }
            }
        }
    }
}
```





```
        }
        if(count!=1){
            return false;//返回false
        }
    }
else{//走的是竖线
    if(fromY<toY){//向下走
        for(j=fromY+1;j<toY;j++){
            if(qizi[j][fromX]!=0){
                count++;//返回false
            }
        }
        if(count!=1){
            return false;//返回false
        }
    }
else{//向上走
    for(j=toY+1;j<fromY;j++){
        if(qizi[j][fromX] != 0){
            count++;//返回false
        }
    }
    if(count!=1){
        return false;//返回false
    }
}
}
break;
case 12://红仕
if(toY<7||toX>5||toX<3){//出了九宫格
    return false;
}
if(Math.abs(fromY-toY) != 1 || Math.abs(toX-fromX) != 1){//走斜线
    return false;
}
break;
case 13://红相
if(toY<5){//不能过河
    return false;//返回false
}
```



```
if(Math.abs(fromX-toX) != 2 || Math.abs(fromY-toY) !=  
2) {//相走"田"字  
    return false;//返回false  
}  
if(qizi[(fromY+toY)/2][(fromX+toX)/2] != 0){  
    return false;//相眼处有棋子  
}  
break;  
case 14://红卒  
if(toY > fromY){//不能回头  
    return false;  
}  
if(fromY > 4 && fromY == toY){  
    return false;//不让走  
}  
if(fromY - toY + Math.abs(toX - fromX) > 1){//只能走一  
步，并且是直线  
    return false;//返回false不让走  
}  
break;  
default:  
    return false;  
}  
return true;  
}
```

到此为止，此项目的主要功能介绍完毕，执行后的界面效果如图 11-3 所示，游戏界面效果如图 11-4 所示。



图11-3 初始效果

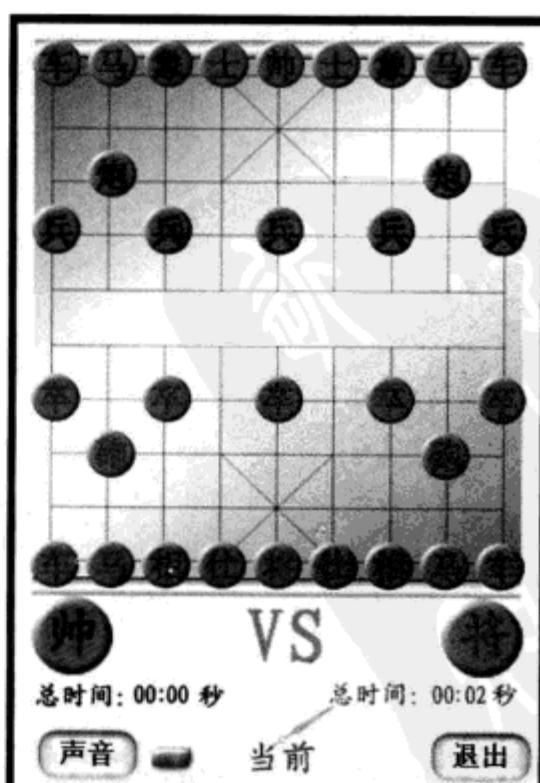


图11-4 游戏界面效果



优化和发布项目

经过本书前面内容的学习，在Android应用领域的主要项目介绍完毕了。当我们开发一个项目完毕后，接下来的工作就是性能优化并发布了。在本章的内容中，将通过几个典型实例的实现过程，详细介绍优化Android项目和发布Android项目的基本流程。

12.1 UI界面中优化之<merge />标签

定义 Android Layout(XML) 时，有四个比较特别的标签是非常重要的，其中有三个是与资源复用有关，分别是 <viewStub/>，<requestFocus/>，<merge/> 和 <include/>。可是以往我们所接触的案例或者官方文档的例子都没有着重去介绍这些标签的重要性。其中 <merge /> 标签十分重要，因为它在优化 UI 结构时起到很重要的作用。<merge /> 标签可以通过删减多余或者额外的层级，从而优化整个 Android Layout 的结构。

在使用 <merge /> 标签的时候需要主题如下两点。

- (1) <merge /> 只可以作为 xml layout 的根节点。
- (2) 当需要扩充的 xml layout 本身是由 merge 作为根节点的话，需要将被导入的 xml layout 置于 viewGroup 中，同时需要设置 attachToRoot 为 True。

其实除了本例外，<merge /> 标签还有另外一个用法。当应用 Include 或者 ViewStub 标签从外部导入 xml 结构时，可以将被导入的 xml 用 merge 作为根节点表示，这样当被嵌入父级结构中后可以很好地将它所包含的子集融合到父级结构中，而不会出现冗余的节点。

本实例将演示 <merge /> 标签在 UI 界面中的优化作用，源码保存在【光盘\daima\第 12 章\mergeC】，具体实现流程如下所示。

step 01 新建一个简单的 Layout 界面，在里面包含了两个 View 元素，分别是 ImageView 和 TextView。在默认状态下将这两个元素放在 FrameLayout 中，效果是在主视图中全屏显示一张图片，之后将标题显示在图片上，并位于视图的下方。文件 main.xml 的主要实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:scaleType="center"
        android:src="@drawable/golden_gate"
        />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dip"
        android:layout_gravity="center_horizontal|bottom"
```



```
    android:padding="12dip"
    android:background="#AA000000"
    android:textColor="#ffffffff"
    android:text="Golden Gate"
/>
</FrameLayout>
```

此时执行后的效果如图 12-1 所示。

step 02 启动 SDK 目录下的“tools”文件夹中的 hierarchyviewer.bat，如图 12-2 所示。

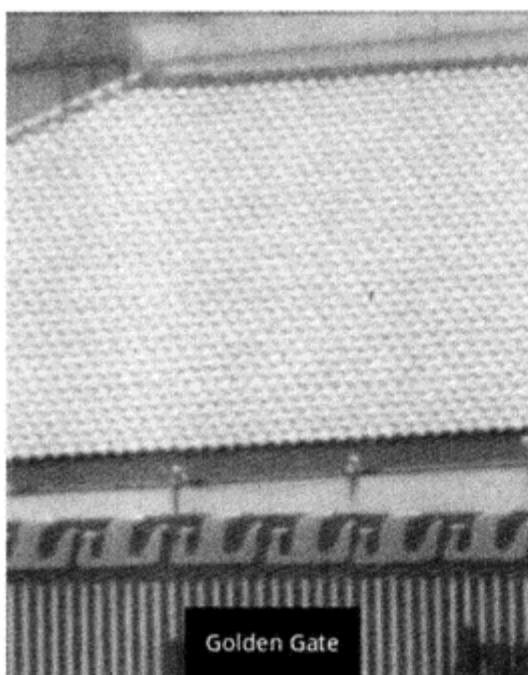


图12-1 执行效果

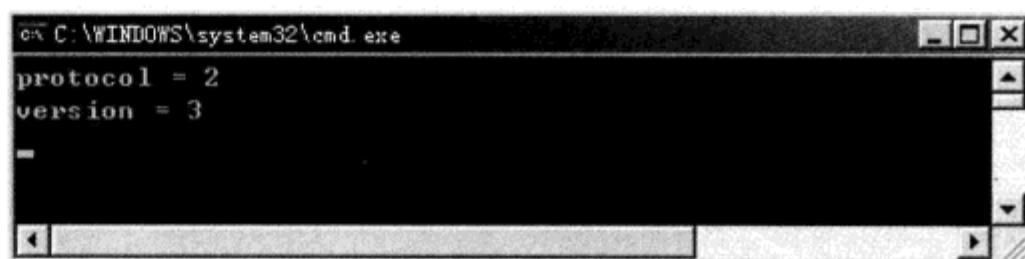


图12-2 启动hierarchyviewer.bat

此时可以查看当前 UI 的结构视图，如图 12-3 所示。

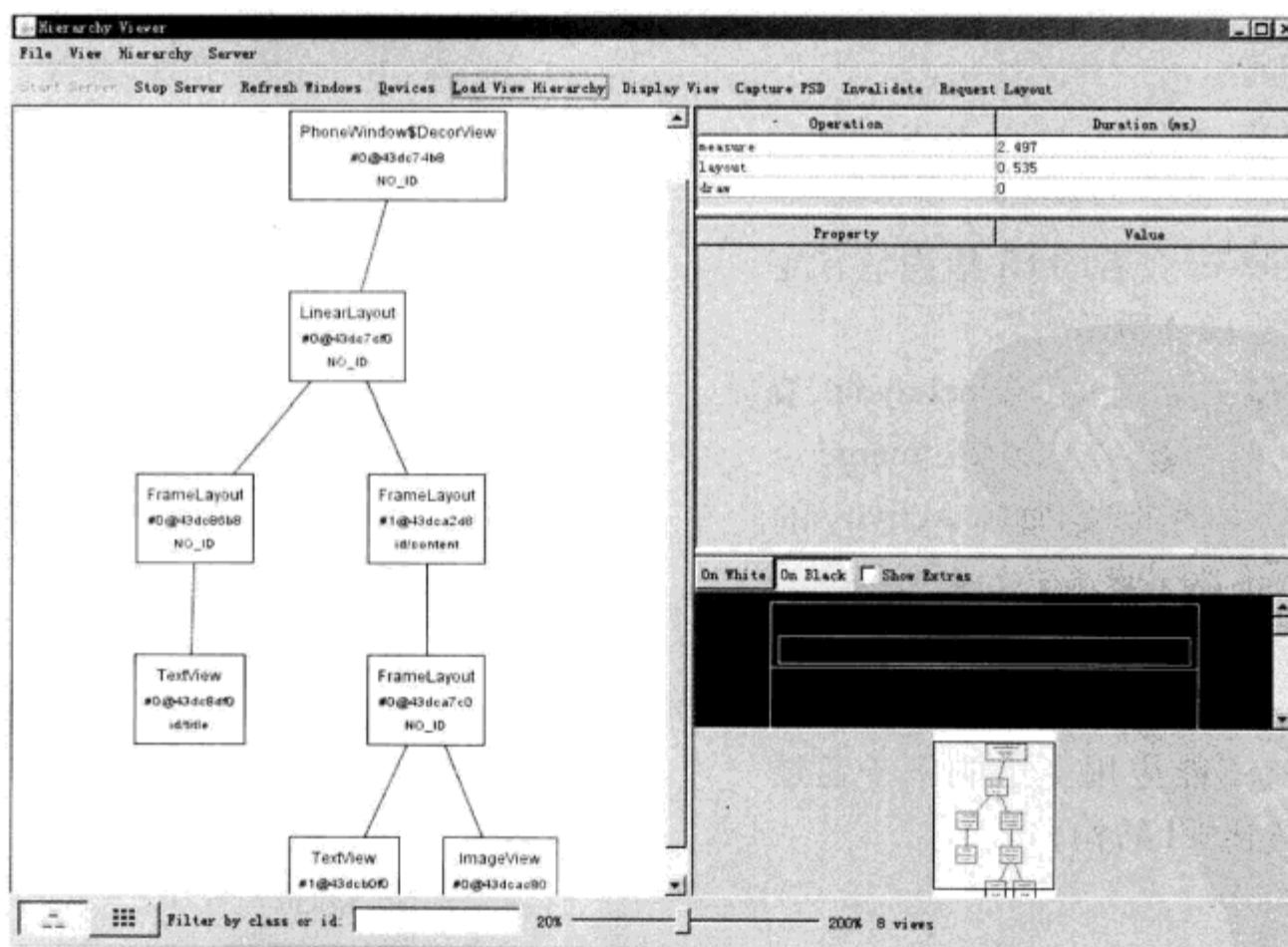


图12-3 文件main.xml的UI结构视图

此时可以很明显地看到由红色线框所包含的结构出现了两个 framelayout 节点，这说明这两个完全意义相同的节点造成了资源浪费，那么如何才能解决呢？这时候就要用到 <merge /> 标签来处理类似的问题了。

step 03 将上边 XML 代码中的 framLayout 换成 merge，实现文件 main2.xml 的具体实现代码如下所示。

```
<merge
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:scaleType="center"
        android:src="@drawable/golden_gate"
        />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dip"
        android:layout_gravity="center_horizontal|bottom"
        android:padding="12dip"
        android:background="#AA000000"
        android:textColor="#ffffffff"
        android:text="Golden Gate"
        />
</merge>
```

此时程序运行后，在 Emulator 中显示的效果是一样的，可是通过 hierarchyviewer 查看的 UI 结构是有变化的，如图 12-4 所示。

此时原来多余的 FrameLayout 节点被合并在一起了，即将 <merge /> 标签中的子集直接加到 Activity 的 FrameLayout 跟节点下。如果所创建的 Layout 并不是用 framLayout 作为根节点（而是应用 LinerLayout 等定义 root 标签），就不能应用上边的例子通过 merge 来优化 UI 结构。

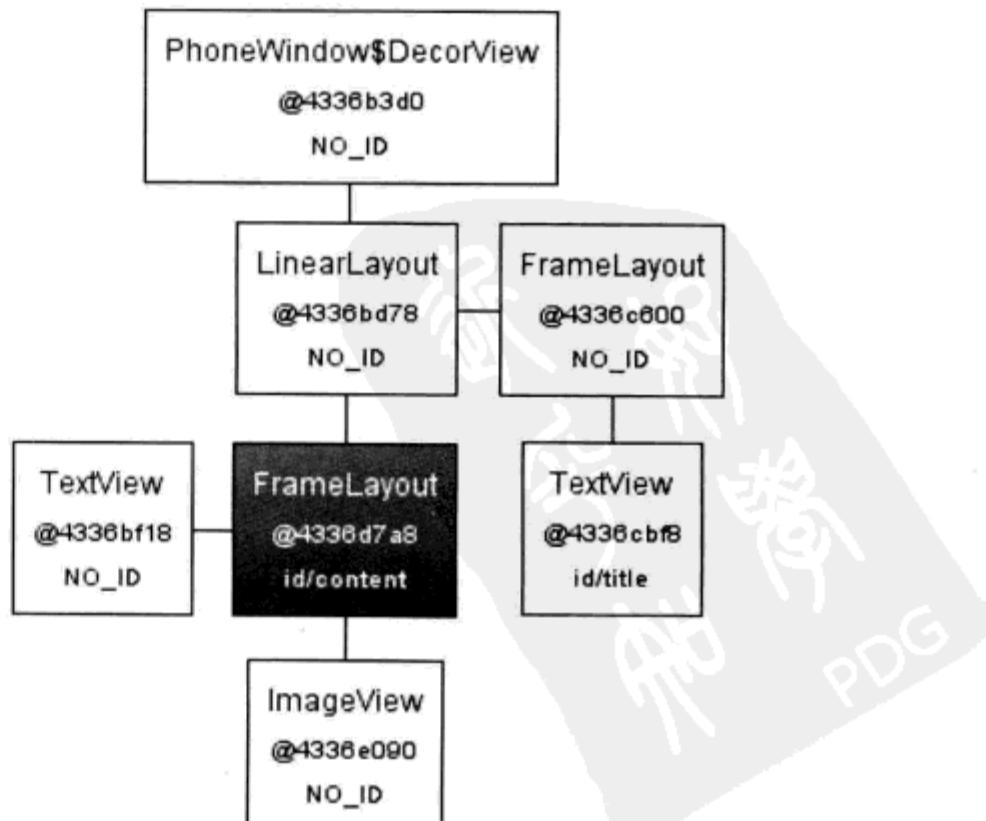


图 12-4 UI 结构视图



12.2 测试计算机的性能

在 Java 中提供了方法 System.currentTimeMillis(), 通过此方法可以得到毫秒级的当前时间, 我们在以前的程序当中一定也写过类似的代码来计算执行某一段代码所消耗的时间。在本实例中, 测试的是 java.util.LinkedList 和 java.util.ArrayList 中的方法 get(int index), ArrayList 要比 LinkedList 高效, 因为前者是随机访问, 而后者需要顺序访问。

本实例的源码保存在【光盘\daima\第 12 章\xing】，具体实现流程如下所示。

step 01 编写布局文件 main.xml, 主要代码如下所示。

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

step 02 编写文件 Testing.java 来创建一个接口, 主要代码如下所示。

```
package com.Xing;
public interface Testing
{
    public void testArrayList();
    public void testLinkedList();
}
```

step 03 编写文件 Xing.java 创建一个测试对象, 用这个对象实现此接口。主要代码如下所示。

```
package com.Xing;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
public class Xing implements Testing
{
    private List link= new LinkedList();
```

```
private List array = new ArrayList();
public Xing()
{
    for (int i = 0; i < 10000; i++)
    {
        array.add(new Integer(i));
        link.add(new Integer(i));
    }
}
public void testArrayList()
{
    for (int i = 0; i < 10000; i++)
        array.get(i);
}
public void testLinkedList()
{
    for (int i = 0; i < 10000; i++)
        link.get(i);
}
```

step 04 编写文件 Handler.java 来实现 InvocationHandler 接口，主要体代码如下所示。

```
package com.Xing;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;
import android.util.Log;
public class Handler implements InvocationHandler
{
    private Object obj;
    public Handler(Object obj)
    {
        this.obj = obj;
    }
    public static Object newInstance(Object obj)
    {
        Object result = Proxy.newProxyInstance(obj.getClass().
getClassLoader(), obj.getClass().getInterfaces(), new Handler(obj));
        return (result);
    }
    public Object invoke(Object proxy, Method method, Object[] args)
throws Throwable
```

```

    {
        Object result;
        try
        {
            Log.i("Handler", "begin method " + method.getName());

            long start = System.currentTimeMillis();
            result = method.invoke(obj, args);
            long end = System.currentTimeMillis();

            Log.i("Handler", "the method " + method.getName() + " lasts " + (end - start) + "ms");
        }
        catch (InvocationTargetException e)
        {
            throw eTargetException();
        }
        catch (Exception e)
        {
            throw new RuntimeException("unexpected invocation exception: " + e.getMessage());
        }
        finally
        {
            Log.i("Handler", "end method " + method.getName());
        }
    }
}

```

step 05 编写文件 Activity01.java，主要代码如下所示。

```

public class Activity01 extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        try
        {
            Testing testing = (Testing) Handler.newInstance(new Xing());
            testing.testArrayList();
        }
    }
}

```

```
        testing.testLinkedList();
    }catch (Exception e){
        e.printStackTrace();
    }
    setContentView(R.layout.main);
}
}
```

执行后在 DDMS 中可以看到详细的 Log 信息，可以看出每个方法的执行时间，如图 12-5 所示。

Time	pid	tag	Message
07-06 19:55	I 65	Activ.	Start proc com.android.email for
07-06 19:55	I 65	Recover.	No recovery log file
07-06 19:55	I 65	Activ.	Start proc android.process.media
07-06 19:55	I 174	Activ.	Publishing provider drm com.android.e
07-06 19:55	I 170	Activ.	Publishing provider com.android.e
07-06 19:55	I 170	Activ.	Publishing provider com.android.e
07-06 19:55	I 170	Activ.	Publishing provider com.android.e
07-06 19:55	I 174	Activ.	Publishing provider media com.an
07-06 19:55	I 150	Activ.	Publishing provider call_log com
07-06 19:55	I 150	Activ.	Publishing provider user_dictionary
07-06 19:55	V 174	Media...	Attached volume: internal
07-06 19:55	D 170	Exchange	BootReceiver onReceive
07-06 19:55	D 170	EAS S...	!!! EAS SyncManager, onCreate
07-06 19:55	I 174	Activ.	Publishing provider downloads: co
07-06 19:55	I 65	Activ.	Start proc com.android.defcontainer
07-06 19:55	D 170	EAS S...	!!! EAS SyncManager, onStartCommand
07-06 19:55	D 170	EAS S...	!!! EAS SyncManager, stopping self
07-06 19:55	I 65	Activ.	Start proc com.android.alarmclock
07-06 19:55	D 22	...	GC_EXPLICIT freed: 1,025

图 12-5 Log 信息

本实例演示了利用动态代理比较两个方法的执行时间，有时候通过一次简单的测试进行比较是片面的，读者可以进行多次执行测试对象，从而计算出最差、最好和平均性能。这样，才能加快经常执行的程序的速度，尽量少调用速度慢的程序。

使用动态代理的好处是用户不必修改原有代码 FooImpl，但是一个缺点是如果用户的类原来没有实现接口的话，不得不写一个接口。

12.3 测试内存性能

当运行一个 Java 应用程序时，会存在很多需要消耗内存的因素，例如对象、加载类、线程等。在本实例中将只考虑程序中的对象所消耗的虚拟机空间，此时可以利用 Runtime 类的方法 freeMemory() 和 totalMemory() 来实现。

本实例测试内存的性能，源码保存在【光盘\daima\第 12 章\neicun】，具体实现流程如下所示。

step 01 编写布局文件 main.xml，主要代码如下所示。

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

```

        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
/>
</LinearLayout>

```

step 02 编写文件 AA.java 用于创建一个接口 AA，主要代码如下所示。

```

package com.Example211;
public interface AA
{
    public void creatArray();
    public void creatHashMap();
}

```

step 03 编写文件 BB.java 用于创建对象实现接口 BB，此接口的功能是分别比较一个长度为 1000 的 ArrayList 和 HashMap 所占内存空间的大小。主要实现代码如下所示。

```

public class BB implements AA
{
    ArrayList arr = null;
    HashMap hash = null;
    public void creatArray()
    {
        arr = new ArrayList(1000);
    }
    public void creatHashMap()
    {
        hash = new HashMap(1000);
    }
}

```

step 04 编写文件 Memory.java，定义了类 Memory 来计算当前的内存消耗。具体代码如下所示。

```

public class Memory
{
    public static long used()
    {
        long total = Runtime.getRuntime().totalMemory();
        long free = Runtime.getRuntime().freeMemory();
    }
}

```



```
        return (total - free);  
    }  
}
```

step 05 编写文件 Handle.java，在里面定义了 invoke() 方法来修改 Handler 类。主要实现代码如下所示。

```
public class Handler implements InvocationHandler  
{  
    private Object obj;  
    public Handler(Object obj)  
    {  
        this.obj = obj;  
    }  
    public static Object newInstance(Object obj)  
    {  
        Object result = Proxy.newProxyInstance(obj.getClass().  
getClassLoader(), obj.getClass().getInterfaces(), new Handler(obj));  
        return (result);  
    }  
    public Object invoke(Object proxy, Method method, Object[] args)  
throws Throwable  
    {  
        Object result;  
        try  
        {  
            Log.i("Handler", "begin method " + method.getName());  
  
            long start = Memory.used();  
            result = method.invoke(obj, args);  
            long end = Memory.used();  
  
            Log.i("Handler", "memory increased by " + (end - start) +  
"bytes");  
        }  
        catch (InvocationTargetException e)  
        {  
            throw e.getTargetException();  
        }  
        catch (Exception e)  
        {  
            throw new RuntimeException("unexpected invocation  
exception: " + e.getMessage());  
        }  
    }  
}
```

```

        finally
        {
            Log.i("Handler", "end method " + method.getName());
        }
        return result;
    }
}

```

执行后在 DDMS 中可以看到详细的 Log 信息，可以看出所占用的内存空间，如图 12-6 所示。

Time	pid	tag	Message
07-06 19:54:00	D 33	dalvikvm	GC_FOR_MALLOC freed 10024 objects...
07-06 19:54:00	D 33	dalvikvm	GC_FOR_MALLOC freed 8084 objects...
07-06 19:54:00	D 33	dalvikvm	GC_FOR_MALLOC freed 7906 objects...
07-06 19:54:00	D 33	dalvikvm	GC_FOR_MALLOC freed 7645 objects...
07-06 19:54:00	D 33	dalvikvm	GC_FOR_MALLOC freed 7674 objects...
07-06 19:54:00	D 33	dalvikvm	GC_EXPLICIT freed 6657 objects / 1...
07-06 19:54:00	D 33	dalvikvm	GC_EXPLICIT freed 1539 objects / 1...
07-06 19:54:00	D 33	dalvikvm	GC_EXPLICIT freed 447 objects / 2...
07-06 19:54:00	D 33	dalvikvm	GC_EXPLICIT freed 315 objects / 2...
07-06 19:54:00	I 33	Zygote	... preloaded 1265 classes in 2216...
07-06 19:54:00	E 33	Zygote	setreuid() failed errno: 17
07-06 19:54:00	D 33	dalvikvm	GC_EXPLICIT freed 104 objects / 1...
07-06 19:54:00	I 33	Zygote	Preloading resources
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10
07-06 19:54:00	I 33	Zygote	Preloaded drawable resource #0x10

图 12-6 Log 信息

在本实例中，笔者没有遵循命名规范，只是随便的起了 AA 和 BB，AOP 通过分解关注点和 OOP 相互结合，使程序更加简洁易懂，通过 Java 语言本身提供的动态代理帮助我们很容易分解关注点，取得了较好的效果。不过测试对象必须实现接口，在一定程度上限制了动态代理的使用，可以借鉴 Spring 中使用的 CGLib 来为没有实现任何接口的类创建动态代理。

12.4 Android Layout 优化

Android 中的 Layout 优化一直是广大程序员们探讨的话题，接下来将给出两段通用的标准 XML 代码，并不是希望广大读者严格遵循下面的布局格式，但是希望自己项目的需求尽力向下面的标准靠拢。

在进行 Layout 布局时，必须注意下面的四点。

(1) 如果可能，尽量不要使用 LinearLayout，而是使用 RelativeLayout 替换它。这是因为 android:layout_alignWithParentIfMissing 只对 RelativeLayout 有用，如果那个视图设置为 gone，这个属性将按照父视图进行调整。

(2) 在使用 Adapter 控件时，例如 list，如果布局中递归太深则会严重影响性能。

(3) 对于 TextView 和 ImageView 组成的 Layout 来说，可以直接使用 TextView 替换。

(4) 如果其父 Layout 是 FrameLayout，如果子 Layout 也是 FrameLayout，此时可以将



FrameLayout 替换为 merge，这样做的好处是可以减少层递归深度。

本实例演示了 Android 中 Layout 优化的过程，源码保存在【光盘\daima\第 12 章\lay】，第一段标注了 Layout 优化 XML 代码。

```
<?xml version="1.0" encoding="utf-8"?>
<!--<FrameLayout-->
<!-- xmlns:android="http://schemas.android.com/apk/res/android"-->
<!-- android:layout_width="fill_parent"-->
<!-- android:layout_height="fill_parent">-->
<!-- <ListView android:id="@+id/list"-->
<!--     android:layout_width="fill_parent"-->
<!--     android:layout_height="fill_parent"/>-->
<!--     <TextView android:id="@+id/no_item_text"-->
<!--         android:layout_width="fill_parent"-->
<!--         android:layout_height="fill_parent"-->
<!--         android:gravity="center"-->
<!--         android:visibility="gone"/>-->
<!--</FrameLayout>-->
<merge xmlns:android="http://schemas.android.com/apk/res/android">
    <ListView android:id="@+id/list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>
    <TextView android:id="@+id/no_item_text"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center"
        android:visibility="gone"/>
</merge>
```

第二段标注了 Layout 优化 XML 的代码。

```
<?xml version="1.0" encoding="utf-8"?>
<!--<LinearLayout-->
<!-- xmlns:android="http://schemas.android.com/apk/res/android"-->
<!-- android:orientation="vertical"-->
<!-- android:layout_width="fill_parent"-->
<!-- android:layout_height="fill_parent">-->
<!-- -->
<!--     <ImageView android:id="@+id/softicon"-->
<!--         android:layout_width="wrap_content"-->
<!--         android:layout_height="wrap_content"-->
<!--         android:layout_marginTop="10dip"-->
<!--         android:layout_gravity="center"/>-->
<TextView
```

```

    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/softname"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dip"
    android:layout_gravity="center"
    android:gravity="center"
    android:drawableTop="@drawable/icon"/>
<!--</LinearLayout>-->

```

12.5 优化模拟器

我们编写程序后，在电脑上运行的都是模拟器，也许是因为模拟的缘故，在调试一个Android项目时，速度会非常慢。即使自己的电脑配置够高，也是很慢。此时就需要我们掌握一些优化知识和相关技巧，以确保我们的调试工作“加速运转”。

(1) 快速调试多个项目

先运行模拟器，到运行成功的速度非常慢。然后在 Eclipse 中右键单击项目，在弹出命令中依次单击【Run As】 | 【Android Application】，开始运行一个项目。如图 12-7 所示。

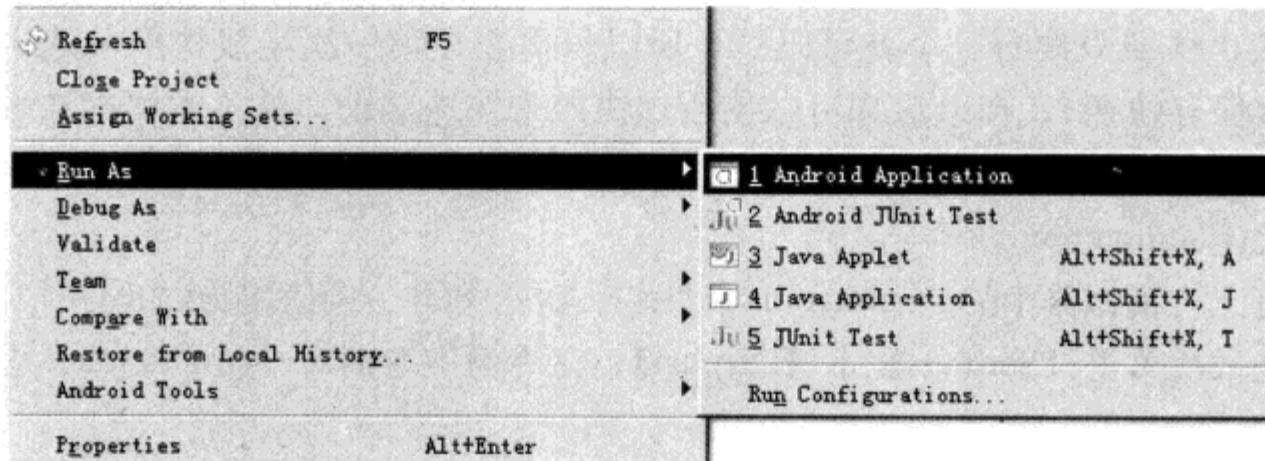


图12-7 运行一个项目

在后面我们不要关闭模拟器，继续在 Eclipse 中打开其他 Android 项目，可以继续用图 12-12 所示的操作方式来调试运行多个 Android 项目，调试多个 Android 项目的速度非常快。这个技巧简称为“一次运行，多次调试！”。

(2) 加速模拟器

我们可以将模拟器中的“自动屏幕适应”功能关闭，关闭后能够加速模拟器的运行。关闭方法是依次单击【Settings】 | 【Display】，在界面中取消对“Auto-rotate screen”选项的选中状态，如图 12-8 所示。

(3) 使用快照 (Snapshot) 功能

首先在 Android 虚拟设备 (AVD) 管理窗口中选中“Snapshot”选项，如图 12-9 所示。这样当以后启动该虚拟设备时，就可以使用快照功能了。

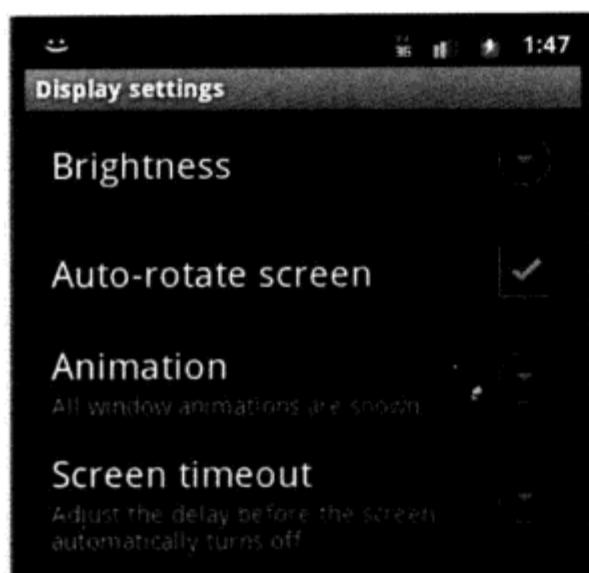


图12-8 取消对“Auto-rotate screen”选项的选中状态



图12-9 选中“Snapshot”选项

然后在虚拟设备的启动窗口中，选中“Launch from snapshot（从快照启动）”和“Save to snapshot（保存快照）”选项，如图 12-10 所示。

这样，在关闭虚拟设备时，就会把虚拟设备的当前状态和设置都自动保存成快照（例如，设置的语言状态、网络状态、甚至你在命令行 adb shell 中的各项设置等）。下次再启动该虚拟设备时，就能立即启动成功，再不用等长长的一段时间了。

当选中了“Save to snapshot”（保存快照），关闭虚拟设备时系统会花一段时间来保存快照。其实，快照只要有一个就可以了。所以只需要在第一次生成快照前选中“Save to snapshot”（保存快照），在以后都可以使用该快照来快速启动，就不需要再选中“Save to snapshot”（保存快照）了。

(4) 关闭 animation 动画

我们可以将模拟器中的“animation（动画）”功能关闭，关闭后能够加速模拟器的运行。关闭方法是依次单击【Settings】 | 【Display】，在界面中取消对“Animation”选项的选中状态，如图 12-11 所示。

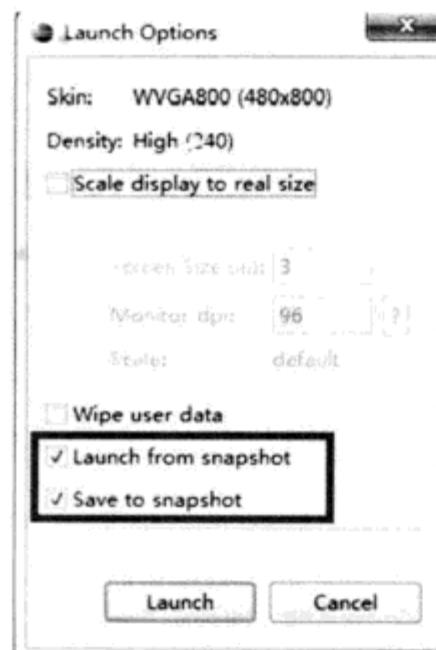


图12-10 选中两项

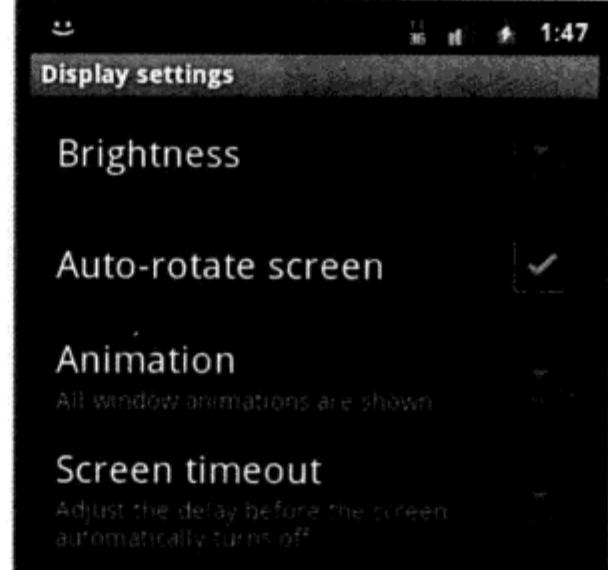
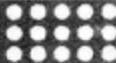


图12-11 取消对“Animation”选项的选中状态



(5) 重置模拟器数据

如果我们多次使用一个模拟器调试 Android 程序，则会在模拟器中保存多个程序。例如图 12-12 的模拟器界面中显示了我们以前调试的程序。如果模拟器中包含原来的程序过多，则会影响调试速度。此时用户可以重置模拟器数据，即删除原来的调试痕迹。具体操作做法如下。

step 01 鼠标右键单击要调试的项目，依次选择【Run As】→【Run Configurations】，如图 12-13 所示。

step 02 在弹出界面中选择“Target”选项卡，在下方选中“Wipe user data”选项，最后单击【RUN】按钮开始运行程序，如图 12-14 所示。

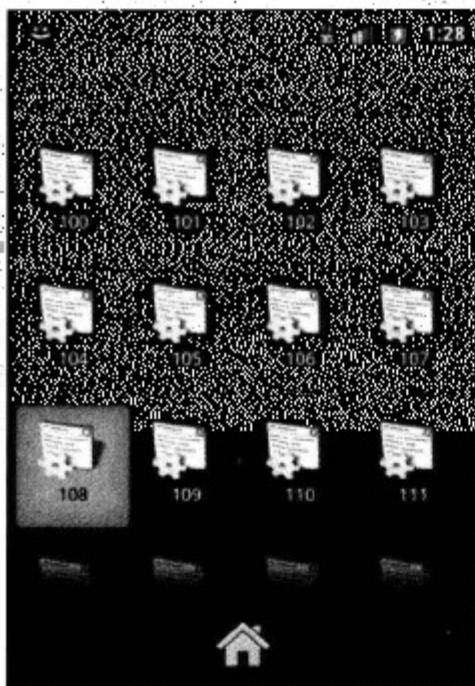


图12-12 以前调试的程序

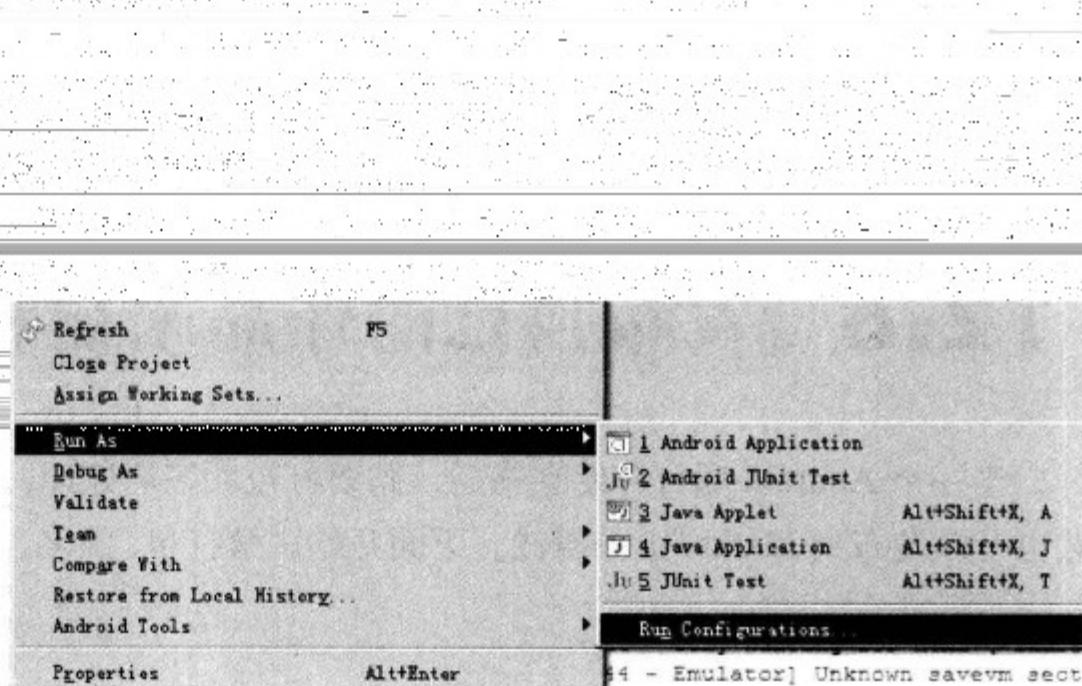


图12-13 运行Android项目

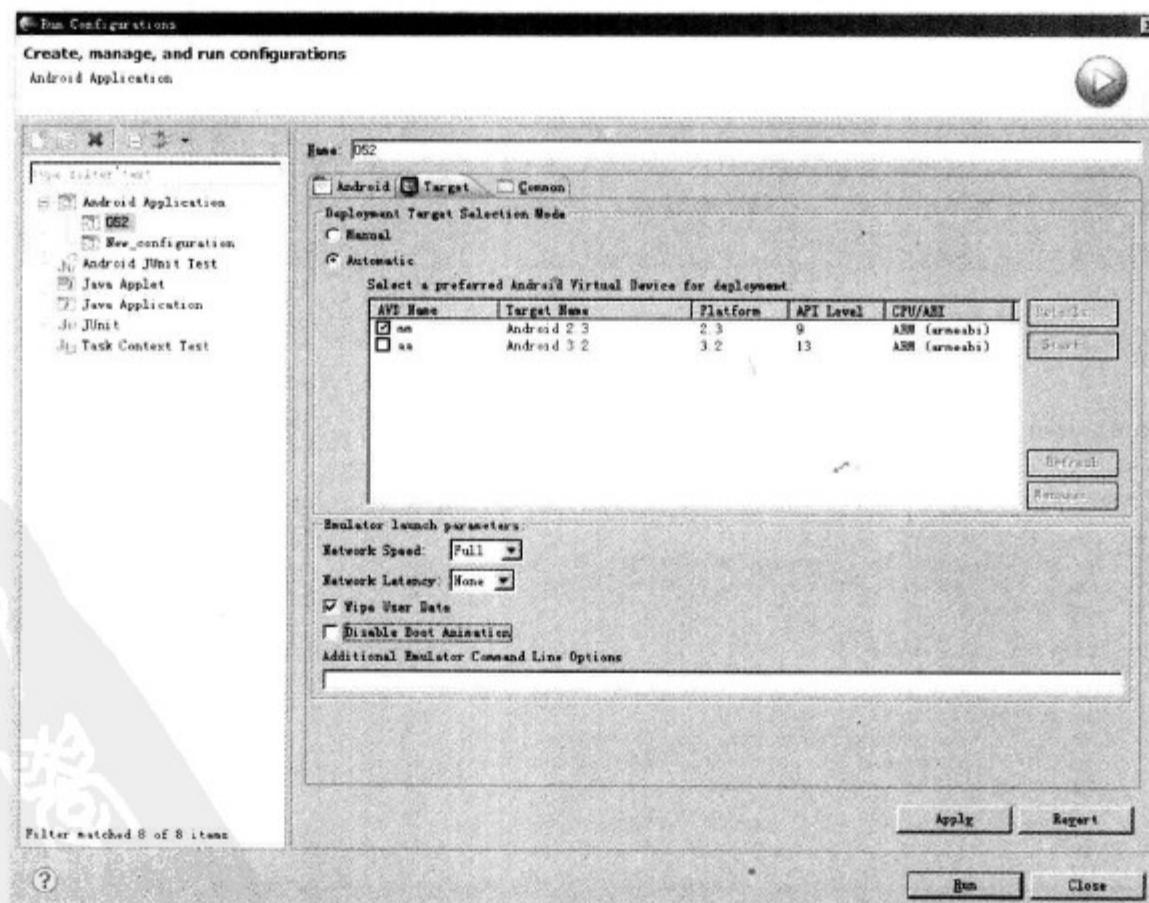


图12-14 选中“Wipe user data”选项



此时模拟器主界面中只显示当前的程序痕迹，如图 12-15 所示。

在图 12-15 中，在“Wipe user data”选项下方有一个“Disable Boot Animations”选项，此选项的功能是取消动画效果。如果选中此选项，模拟器的运行速度会更快，此选项也是优化模拟器的一个重要方法。

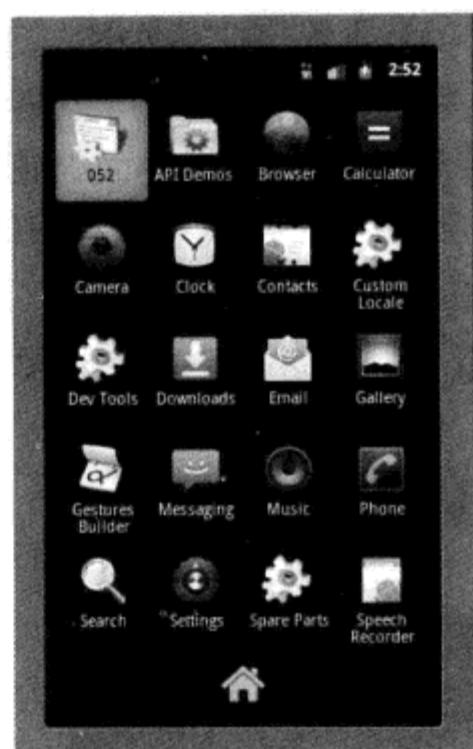


图 12-15 和图 12-12 相比

12.6 发布自己的作品来盈利

当一个 Android 项目开发完毕后，需要打包和签名处理，这样才能放到手机中使用，当然也可以发布到 Market 上去赚钱。下面开始讲解打包、签名、发布 Android 程序的具体过程。

12.6.1 申请会员

即去 Market 申请成为会员，具体流程如下。

step 01 登录 <http://market.android/publish/signup>，如图 12-16 所示。



图 12-16 登录 Market

step 02 单击链接 Create an account now，来到注册页面，如图 12-17 所示。

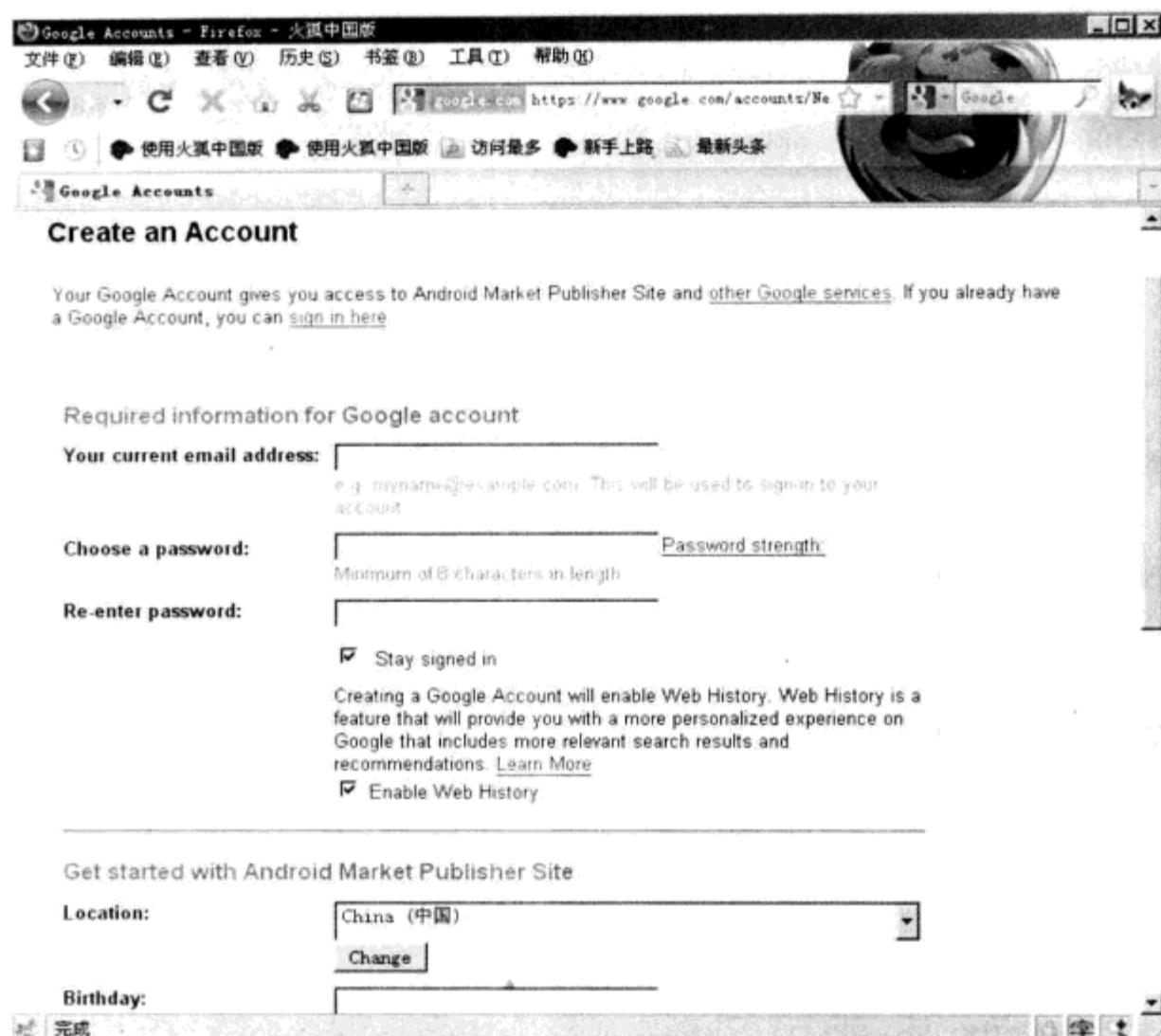


图12-17 注册界面

step 03 单击同意协议后来到下一步页面，在此输入手机号码，如图 12-18 所示。

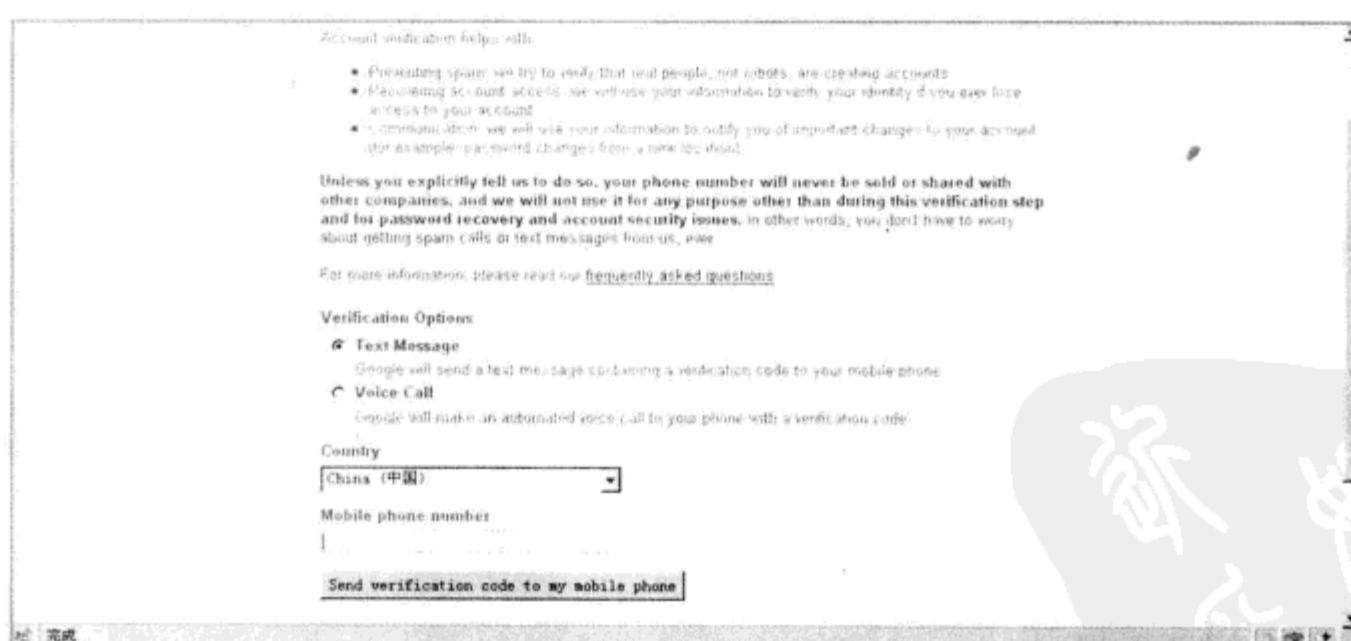


图12-18 输入手机号码

step 04 在新界面中输入手机获取的验证码，如图 12-19 所示。

step 05 验证通过后，在新界面中继续输入信息，如图 12-20 所示。

step 06 单击“Continue »”按钮后，提示需要花费 25 美元，支付后才能成为正式会员，如图 12-21 所示。

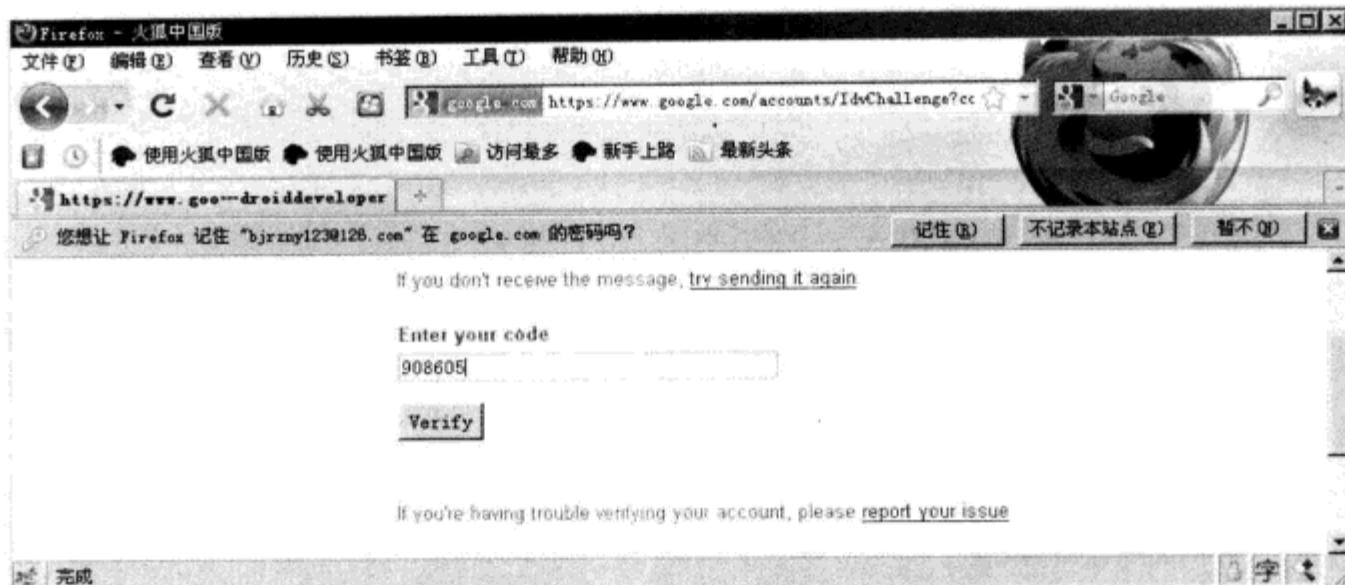


图12-19 输入验证码

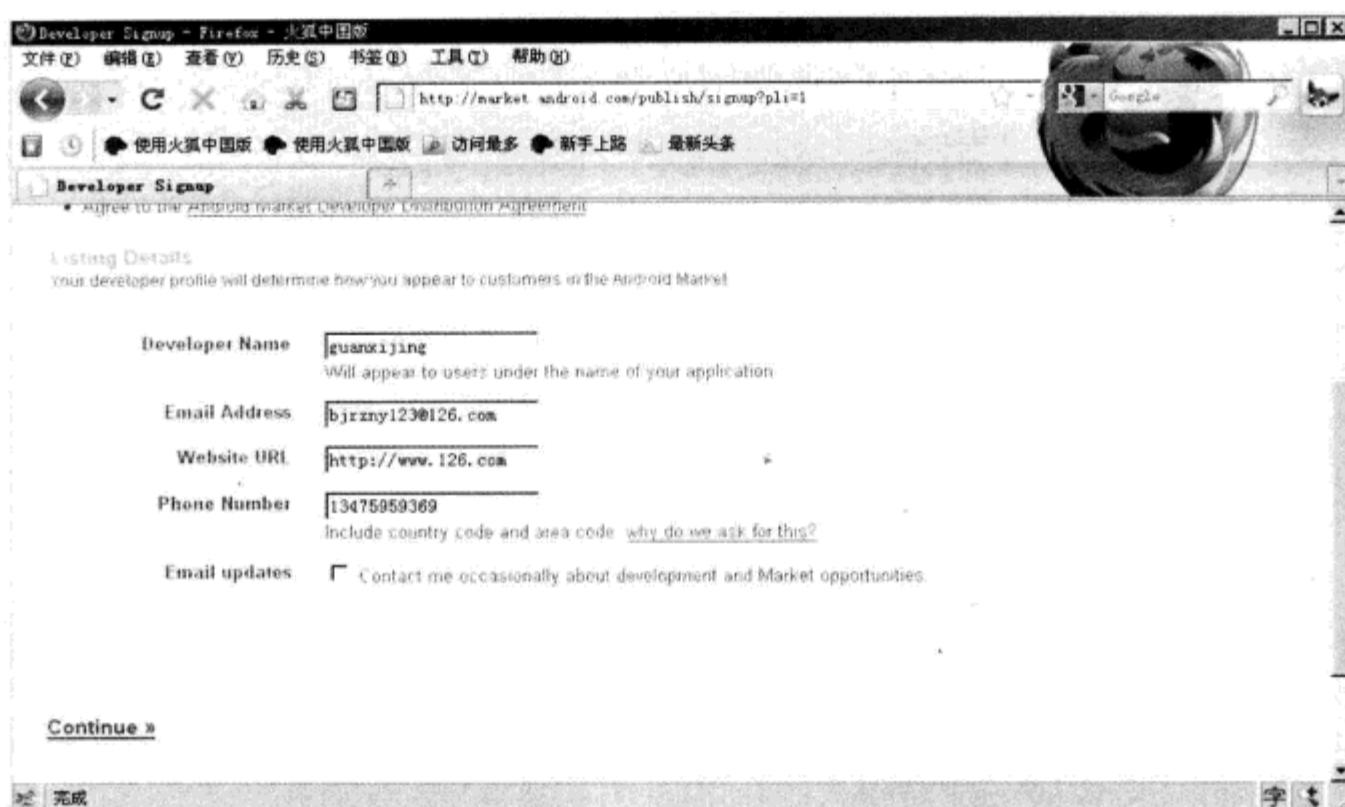


图12-20 输入信息

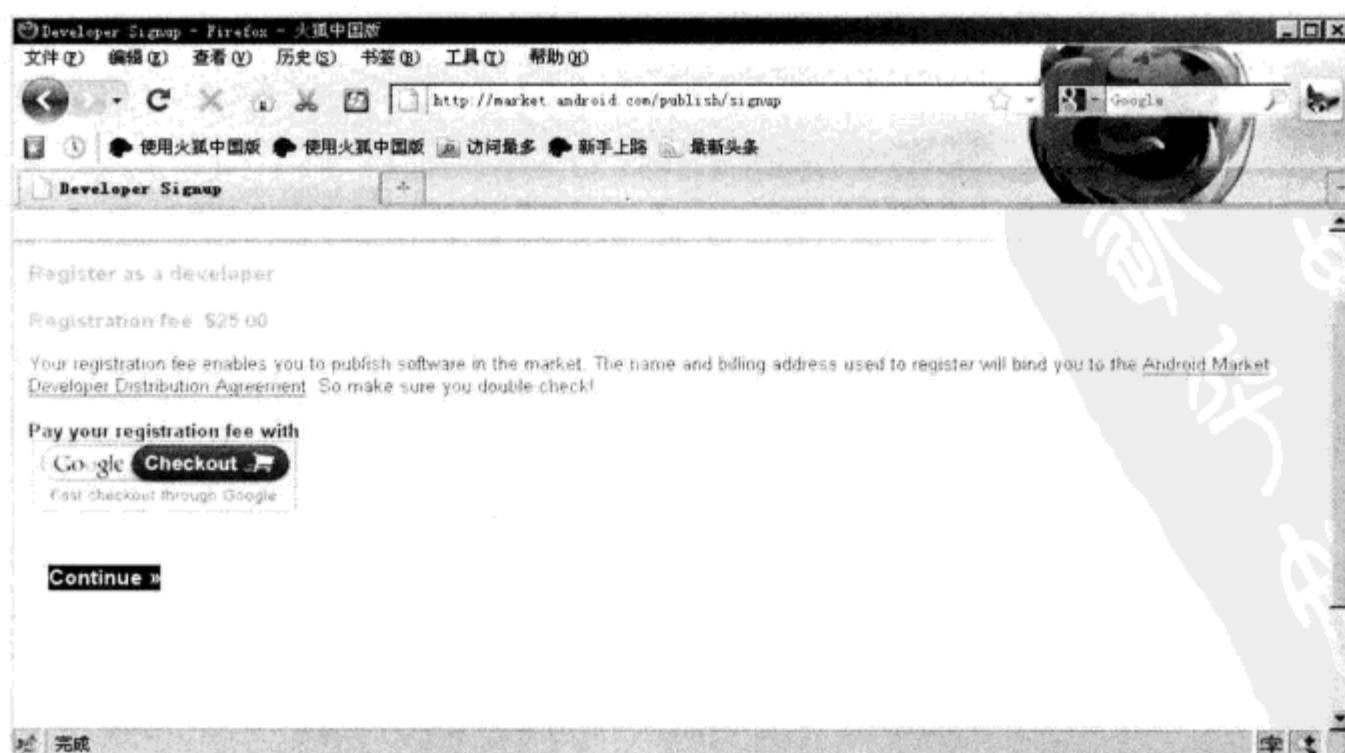


图12-21 需要支付界面



step 07 单击  按钮来到支付界面，如图 12-22 所示。

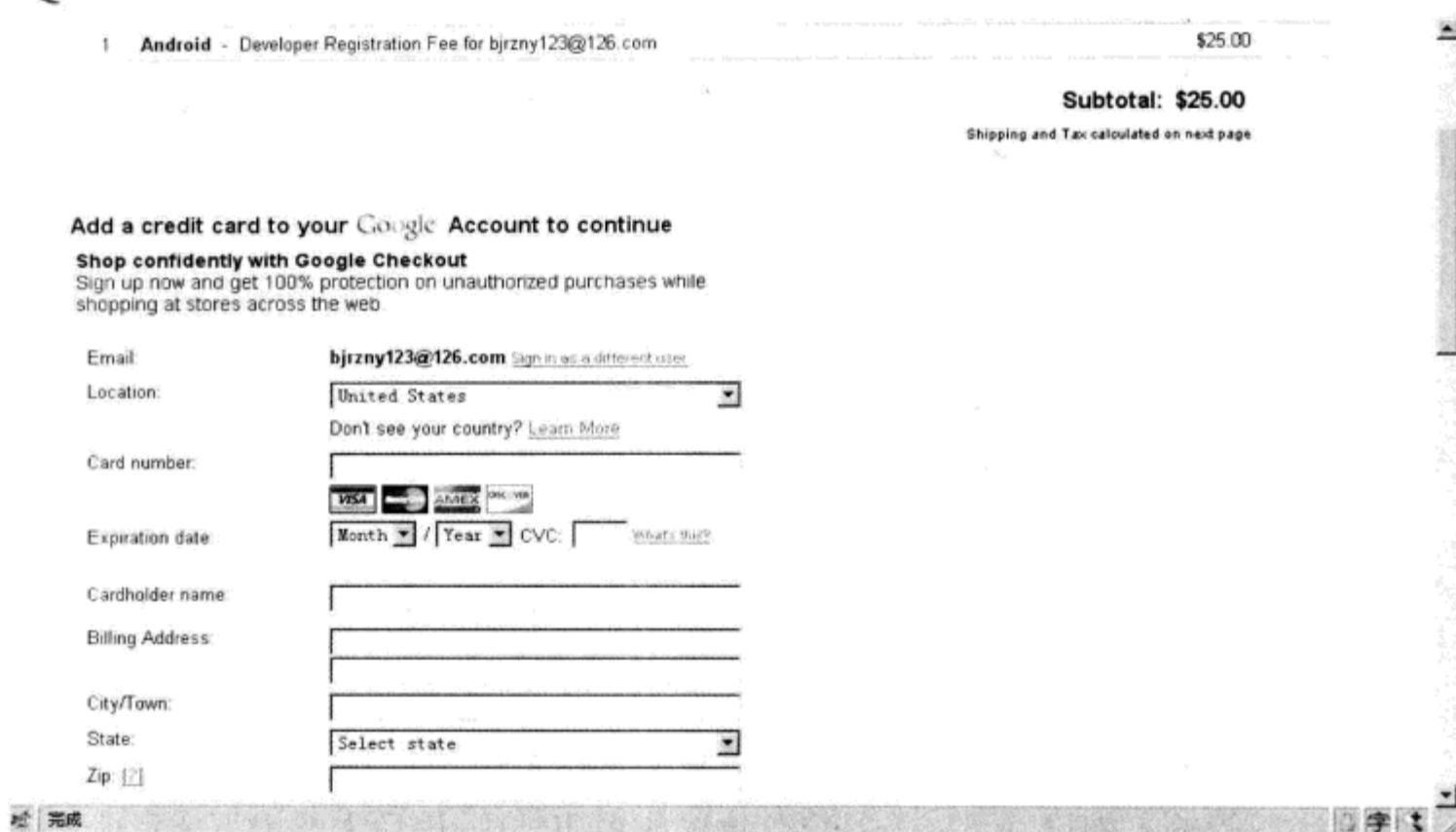


图12-22 支付界面

在此输入你的信用卡信息，完成支付后即可成为正式会员。

12.6.2 生成签名文件

Android 程序的签名和 Symbian 类似都可以自签名 (Self-signed)，但是在 Android 平台上证书初期还显得形同虚设，平时开发时通过 ADB 接口上传的程序会自动被签有 Debug 权限的程序。需要签名验证在上传程序到 Android Market 上时大家都已经发现这个问题了。

Androi 签名文件的制作方法有两种：

第一种：命令行生成

具体流程如下：

(1) cmd 如下命令

```
keytool -genkey -alias android123.keystore -keyalg RSA -validity 20000  
-keystore android123.keystore
```

然后一次提示用户输入如下信息：

输入 keystore 密码：[密码不回显]

再次输入新密码：[密码不回显]

您的名字与姓氏是什么？

[Unknown] : android123

您的组织单位名称是什么？

[Unknown] : www.android123.com.cn

您的组织名称是什么？

[Unknown] : www.android123.com.cn



您的组织名称是什么？

[Unknown] : www.android123.com.cn

您所在的城市或区域名称是什么？

[Unknown] : New York

您所在的州或省份名称是什么？

[Unknown] : New York

该单位的两字母国家代码是什么

[Unknown] : CN

CN=android123, OU=www.android123.com.cn, O=www.android123.com.cn, L>New York, ST

=New York, C=CN 正确吗？

[否] : Y

输入 <android123.keystore> 的主密码（如果和 keystore 密码相同，按 Enter 键）：

其中参数 -validity 为证书有效天数，这里我们写得大些，如 200 天。还有在输入密码时没有回显，只管输入就可以了，一般位数建议使用 20 位，最后需要记下来后面还要用。接下来就可以为 apk 文件签名了。

(2) 执行

```
jarsigner -verbose -keystore android123.keystore -signedjar  
android123_signed.apk android123.apk android123.keystore
```

这样就可以生成签名的 apk 文件，假设输入文件 android123.apk，则最终生成 android123_signed.apk 为 Android 签名后的 APK 执行文件。



keytool用法和jarsigner用法总结

(1) keytool用法：

-certreq [-v] [-protected]

```
[-alias <别名>] [-sigalg <sigalg>]  
[-file <csr_file>] [-keypass <密钥库口令>]  
[-keystore <密钥库>] [-storepass <存储库口令>]  
[-storetype <存储类型>] [-providername <名称>]  
[-providerclass <提供方类名称> [-providerarg <参数>]] ...  
[-providerpath <路径列表>]
```

-changealias [-v] [-protected] -alias <别名> -destalias <目标别名>

```
[-keypass <密钥库口令>]  
[-keystore <密钥库>] [-storepass <存储库口令>]  
[-storetype <存储类型>] [-providername <名称>]  
[-providerclass <提供方类名称> [-providerarg <参数>]] ...  
[-providerpath <路径列表>]
```



```
-delete      [-v] [-protected] -alias <别名>
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-exportcert   [-v] [-rfc] [-protected]
              [-alias <别名>] [-file <认证文件>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-genkeypair   [-v] [-protected]
              [-alias <别名>]
              [-keyalg <keyalg>] [-keysize <密钥大小>]
              [-sigalg <sigalg>] [-dname <dname>]
              [-validity <valDays>] [-keypass <密钥库口令>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-genseckeypair [-v] [-protected]
              [-alias <别名>] [-keypass <密钥库口令>]
              [-keyalg <keyalg>] [-keysize <密钥大小>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-help
-importcert   [-v] [-noprompt] [-trustcacerts] [-protected]
              [-alias <别名>]
              [-file <认证文件>] [-keypass <密钥库口令>]
              [-keystore <密钥库>] [-storepass <存储库口令>]
              [-storetype <存储类型>] [-providername <名称>]
              [-providerclass <提供方类名称> [-providerarg <参数>]] ...
              [-providerpath <路径列表>]

-importkeystore [-v]
              [-srckeystore <源密钥库>] [-destkeystore <目标密钥库>]
              [-srcstoretype <源存储类型>] [-deststoretype <目标存储类型>]
```



```
[ -srcstorepass <源存储库口令> ] [ -deststorepass <目标存储库口令> ]
[ -srcprotected ] [ -destprotected ]
[ -srcprovidername <源提供方名称> ]
[ -destprovidername <目标提供方名称> ]
[ -srcalias <源别名> [ -destalias <目标别名> ]
[ -srckeypass <源密钥库口令> ] [ -destkeypass <目标密钥库口令> ] ]
[ -noprompt ]
[ -providerclass <提供方类名称> [ -providerarg <参数> ] ] ...
[ -providerpath <路径列表> ]

-keystorewd [ -v ] [ -alias <别名> ]
[ -keypass <旧密钥库口令> ] [ -new <新密钥库口令> ]
[ -keystore <密钥库> ] [ -storepass <存储库口令> ]
[ -storetype <存储类型> ] [ -providername <名称> ]
[ -providerclass <提供方类名称> [ -providerarg <参数> ] ] ...
[ -providerpath <路径列表> ]

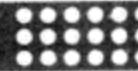
-list [ -v | -rfc ] [ -protected ]
[ -alias <别名> ]
[ -keystore <密钥库> ] [ -storepass <存储库口令> ]
[ -storetype <存储类型> ] [ -providername <名称> ]
[ -providerclass <提供方类名称> [ -providerarg <参数> ] ] ...
[ -providerpath <路径列表> ]

-printcert [ -v ] [ -file <认证文件> ]

-storepasswd [ -v ] [ -new <新存储库口令> ]
[ -keystore <密钥库> ] [ -storepass <存储库口令> ]
[ -storetype <存储类型> ] [ -providername <名称> ]
[ -providerclass <提供方类名称> [ -providerarg <参数> ] ] ...
[ -providerpath <路径列表> ]
```

(2) jarsigner用法：

[选项] jar	文件别名
jarsigner -verify [选项] jar	文件
[-keystore <url>]	密钥库位置
[-storepass <口令>]	用于密钥库完整性的口令
[-storetype <类型>]	密钥库类型
[-keypass <口令>]	专用密钥的口令 (如果不同)
[-sigfile <文件>]	.SF/.DSA 文件的名称
[-signedjar <文件>]	已签名的 JAR 文件的名称
[-digestalg <算法>]	摘要算法的名称



[-sigalg <算法>]	签名算法的名称
[-verify]	验证已签名的 JAR 文件
[-verbose]	签名/验证时输出详细信息
[-certs]	输出详细信息和验证时显示证书
[-tsa <url>]	时间戳机构的位置
[-tsacert <别名>]	时间戳机构的公共密钥证书
[-altsigner <类>]	替代的签名机制的类名
[-altsignerpath <路径列表>]	替代的签名机制的位置
[-internalsf]	在签名块内包含 .SF 文件
[-sectionsonly]	不计算整个清单的散列
[-protected]	密钥库已保护验证路径
[-providerName <名称>]	提供者名称
[-providerClass <类>]	加密服务提供者的名称
[-providerArg <参数>]] ...	主类文件和构造函数参数

第二种：eclipse 的 ADT 生成

实际上，使用 eclipse 可以更加直观、方便地生成签名文件，具体流程如下：

- step 01** 右键单击 eclipse 项目名，依次选择“Android Tools” | “Export Signed Application Package…”，如图 12-23 所示。

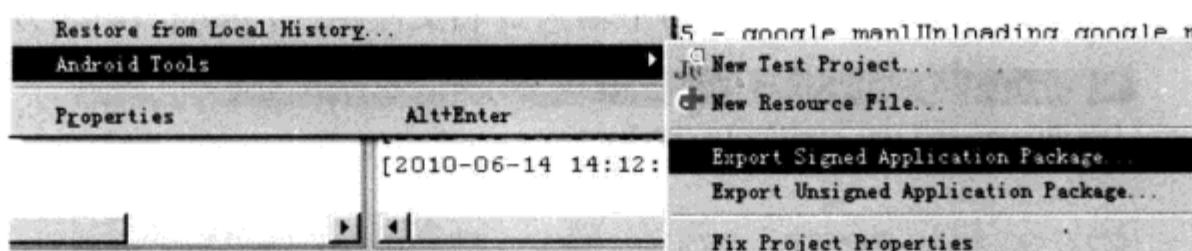


图 12-23 选择导出

- step 02** 在弹出界面中选择要导出的项目，如图 12-24 所示。

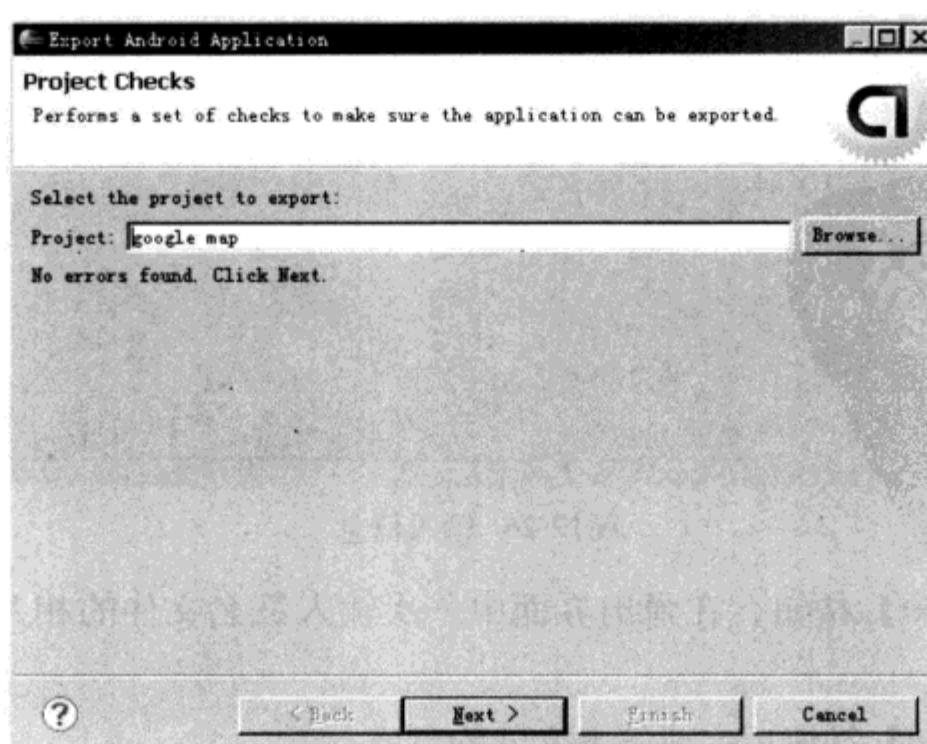


图 12-24 选择要导出的项目

step 03 单击【Next】按钮，在弹出界面中选择 Create new keystore，然后分别输入文件名和密码，如图 12-25 所示。

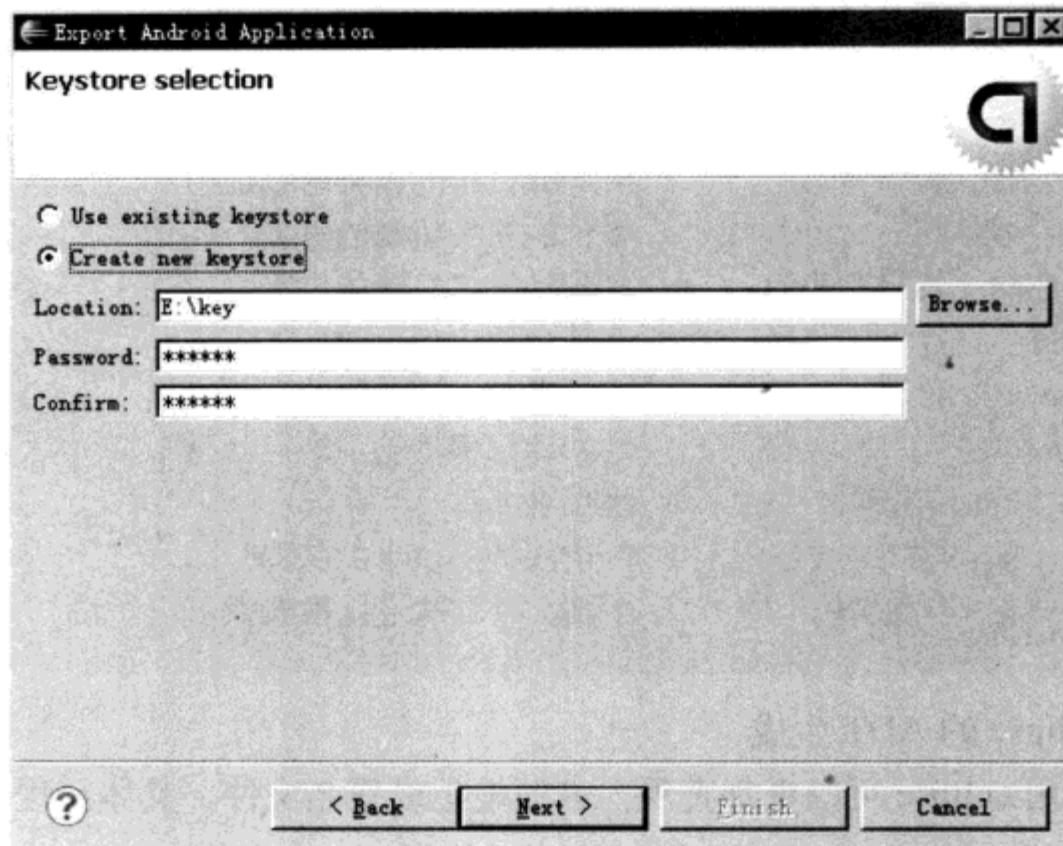


图12-25 文件名和密码

step 04 单击【Next】按钮，在弹出界面中输入签名文件路径，如图 12-26 所示。

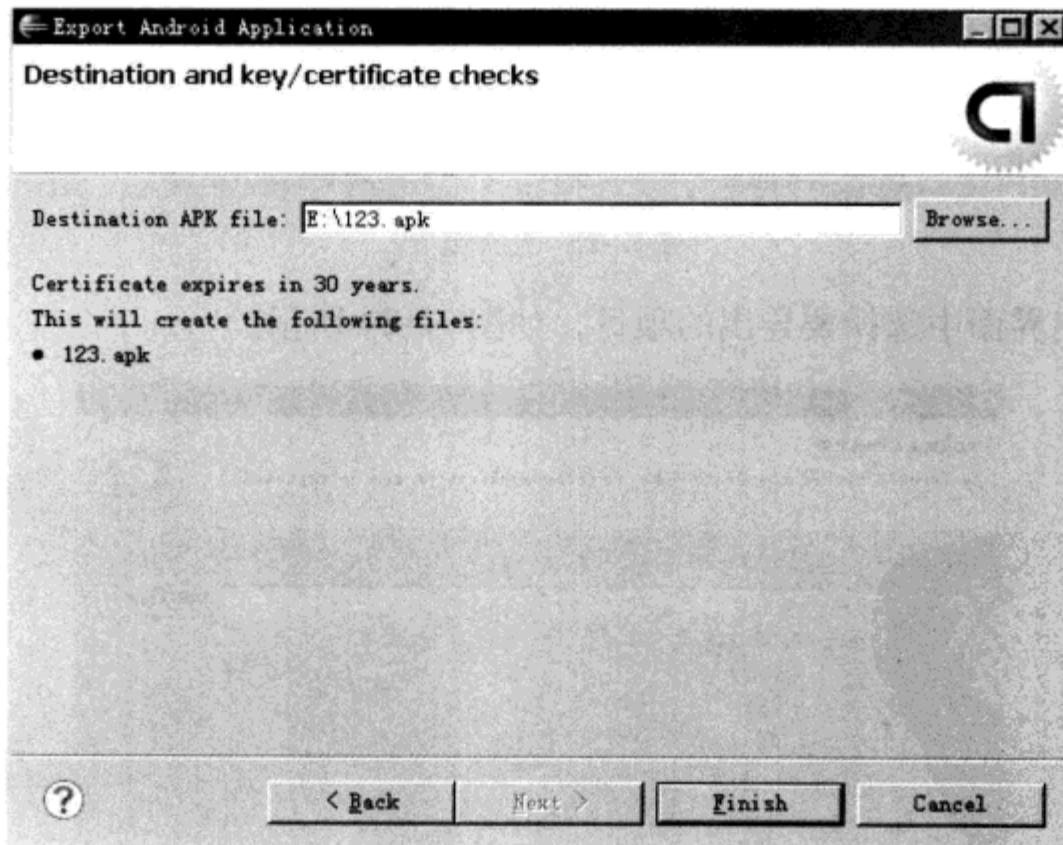


图12-26 输入信息

step 05 单击【Next】按钮，在弹出界面中一次输入签名文件的相关信息，如图 12-27 所示。

step 06 单击【Next】按钮后完成签名文件的创建。

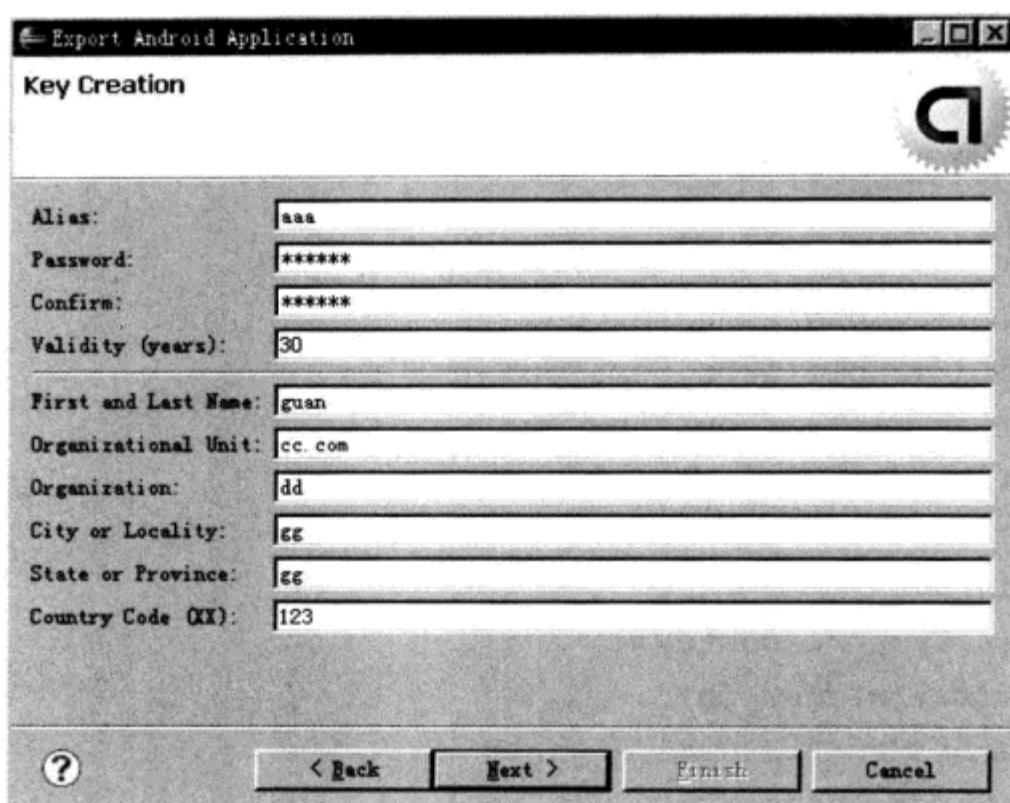


图12-27 输入信息

12.6.3 使用签名文件

生成签名文件后，就可以使用它了，在此也有两种方式。

第一种：命令行

(1) 假设生成的签名文件是 ChangeBackgroundWidget.apk，则最终生成 ChangeBackgroundWidget_signed.apk 为 Android 签名后的 APK 执行文件。

输入以下命令行：

```
jarsigner -verbose -keystore ChangeBackgroundWidget.keystore
-signedjar ChangeBackgroundWidget_signed.apk ChangeBackgroundWidget.apk
ChangeBackgroundWidget.keystore
```

上面命令中间不换行。

(2) 按“Enter”键，根据提示输入密钥库的口令短语（即密码），详细信息如下。

输入密钥库的口令短语：

```
正在添加: META-INF/MANIFEST.MF
正在添加: META-INF/CHANGEBA.SF
正在添加: META-INF/CHANGEBA.RSA
正在签名: res/drawable/icon.png
正在签名: res/drawable/icon_audio.png
正在签名: res/drawable/icon_exit.png
正在签名: res/drawable/icon_folder.png
正在签名: res/drawable/icon_home.png
正在签名: res/drawable/icon_img.png
正在签名: res/drawable/icon_left.png
正在签名: res/drawable/icon_mantou.png
```



```
正在签名: res/drawable/icon_other.png  
正在签名: res/drawable/icon_pause.png  
正在签名: res/drawable/icon_play.png  
正在签名: res/drawable/icon_return.png  
正在签名: res/drawable/icon_right.png  
正在签名: res/drawable/icon_set.png  
正在签名: res/drawable/icon_text.png  
正在签名: res/drawable/icon_xin.png  
正在签名: res/layout/fileitem.xml  
正在签名: res/layout/filelist.xml  
正在签名: res/layout/main.xml  
正在签名: res/layout/widget.xml  
正在签名: res/xml/widget_info.xml  
正在签名: AndroidManifest.xml  
正在签名: resources.arsc  
正在签名: classes.dex
```

通过上述过程处理后，即可将未签名文件 ChangeBackgroundWidget.apk 签名为 ChangeBackgroundWidget_signed.apk。

在上述方式中，读者可能会遇到以下问题：

问题一：jarsigner：无法打开 jar 文件 ChangeBackgroundWidget.apk。

解决方法：将要进行签名的 APK 放到对应的文件下，把要签名的 ChangeBackgroundWidget.apk 放到 JDK 的 bin 文件里。

问题二：jarsigner 无法对 jar 进行签名：java.util.zip.ZipException: invalid entry compressed size (expected 1598 but got 1622 bytes)。

方法一：Android 开发网提示这些问题主要是由于资源文件造成的，对于 android 开发来说应该检查 res 文件夹中的文件，逐个排查。这个问题可以通过升级系统的 JDK 和 JRE 版本来解决。

方法二：这是因为默认给 apk 做了 debug 签名，所以无法做新的签名。这时就必须点工程右键 ->Android Tools ->Export Unsigned Application Package。

或者从 AndroidManifest.xml 的 Exporting 上也是一样的。

然后再基于这个导出的 unsigned apk 做签名，导出的时候最好将其目录选在用户之前产生 keystore 的那个目录下，这样操作起来就方便了。

第二种：Eclipse 的 ADT 生成

实际上，使用 Eclipse 可以更加直观、方便地生成签名文件，具体流程如下。

step 01 手表右键单击 Eclipse 项目名，依次选择“Android Tools” | “Export Signed Application Package…”，如图 12-28 所示。

step 02 在弹出界面中选择项目，如图 12-29 所示。

step 03 单击【Next】按钮，在弹出界面中选择“Use existing keystore”，并输入文件的密码，如图 12-30 所示。

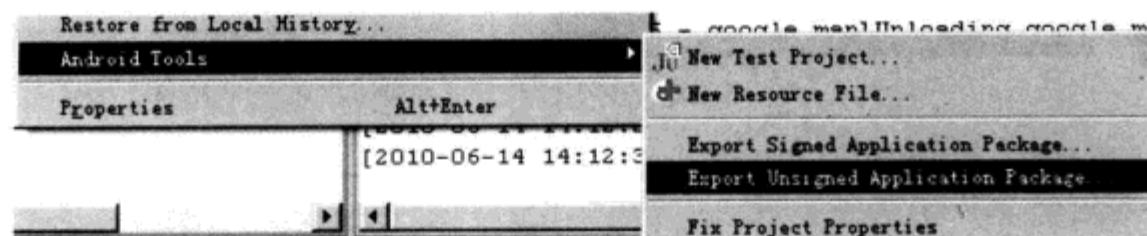


图12-28 Export Unsigned Application Package

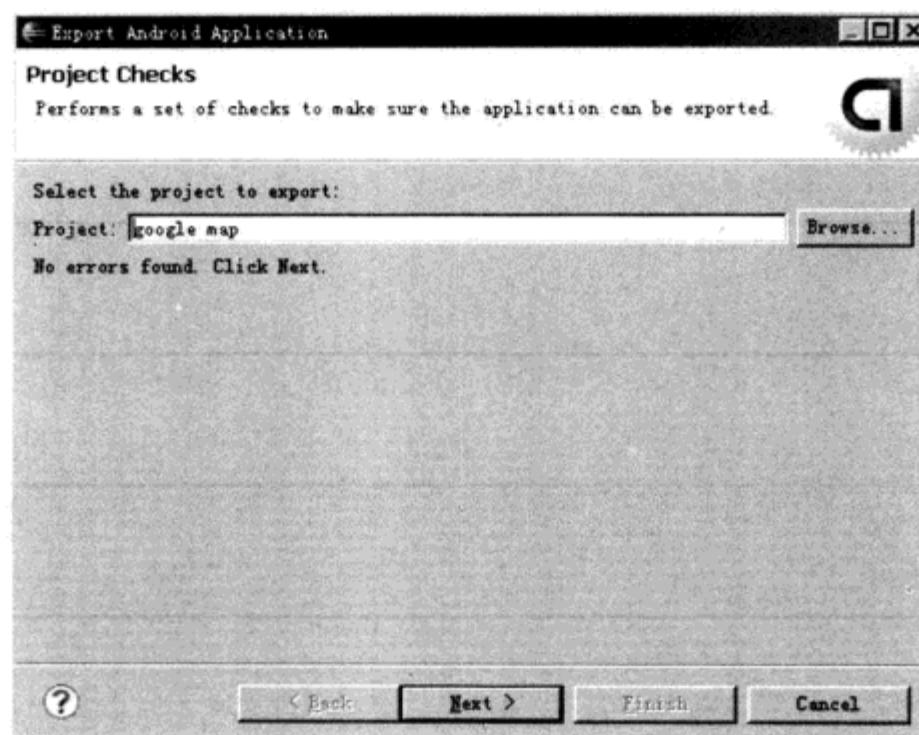


图12-29 选择项目

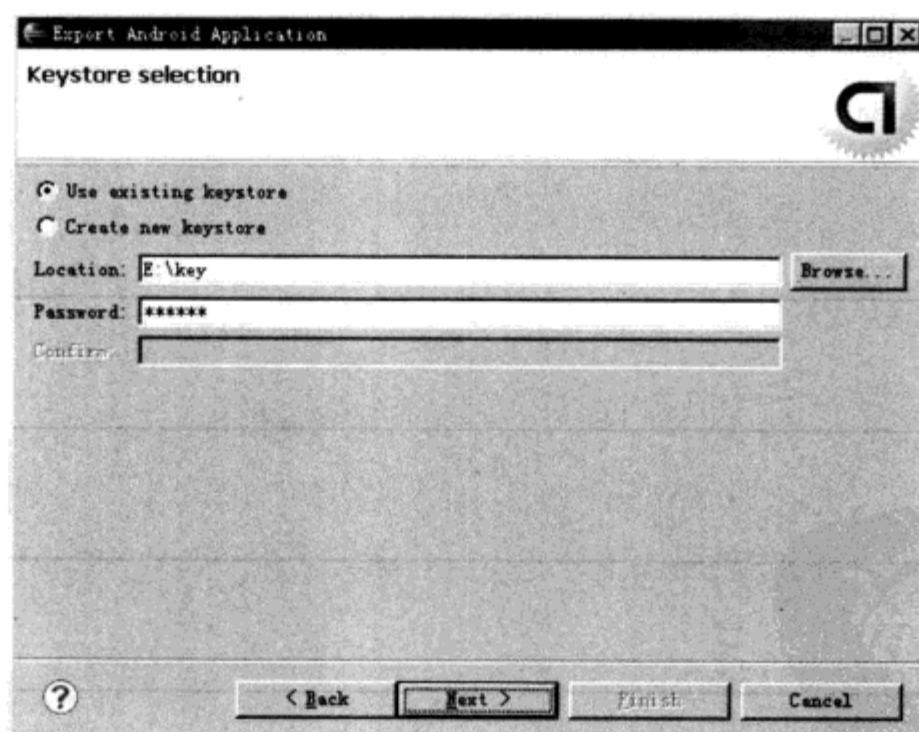


图12-30 输入密码

(4) 单击【Next】按钮，输入原来签名文件的资料和密码，按照默认提示完成签名。

12.6.4 发布

发布的过程比较简单，来到 Market，登录个人中心，上传签名后的文件即可，具体操作流程在 Market 站点上有详细介绍说明。由于本书篇幅有限，在此将不做详细介绍。



- android与iphone及ipad开发书籍 持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等 持续不断更新中.....
- delphi《书籍》及《视频》教程 持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修炼班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程（个人博客系统49课全），外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班，php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库 ←

文件名 <ul style="list-style-type: none"> <input type="checkbox"/> 办公类软件使用指南 <input type="checkbox"/> 医学 <input type="checkbox"/> 历史人物传记 <input type="checkbox"/> 哲学宗教 <input type="checkbox"/> 外语资料（除英语外）（除英语外） <input type="checkbox"/> 官场类小说 <input type="checkbox"/> 建筑工程类 <input type="checkbox"/> 情感生活类小说 <input type="checkbox"/> 政治军事 <input type="checkbox"/> 教育学习科普大全 网址：WLSAM168.400GB.COM <input type="checkbox"/> 文学理论 <input type="checkbox"/> 智力开发、增强记忆、快速阅读技巧大全 <input type="checkbox"/> 社会生活 <input type="checkbox"/> 科学技术 <input type="checkbox"/> 程序编程类 <input type="checkbox"/> 经济管理 <input type="checkbox"/> 网络安全及管理 <input type="checkbox"/> 网赚系列 <input type="checkbox"/> 美食小吃烹饪煲汤大全 <input type="checkbox"/> 课外读物 	P D F 电子书专题库，内容详尽，每天不断更新！！ <p>本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习，如用于商业或非法用途的后果自负！</p>
--	--

OE Foxit PDF Editor ±à¼-ºæ·É·ÉùÓÐ (c) by Foxit Software Company, 2004 VIP培训教程，易语言黑月VIP视频教程，网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)

- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等 持续不断更新中....
- 网建设相关的pdf书籍及各种vip视频教程 持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价营销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等 持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

[电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
 [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
 [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
 [算法之道（第2版）].邹恒明.扫描版.pdf
 [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
 [深入浅出WPF].刘铁猛.扫描版.pdf
 [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
 [精通.NET互操作：P/Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
 [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
 [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
 [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
 [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
 [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
 [Ext江湖].大漠穷秋.扫描版.pdf
 [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 >

[General Information]

书名 = Android 开发从入门到精通

作者 = 扶松柏编

页码 = 542

I S B N = 542

S S 号 = 12875352

d x N u m b e r = 0 0 0 0 0 8 2 2 3 6 2 1

出版时间 = 2012.01

出版社 = 该引擎未能查询到

定价 : 69.00

试读地址 = h t t p : / / b o o k . d u x i u . c o m / b o o k D e t a i l . j s p ? d x N u m b e r = 0 0 0 0 0 8 2 2 3 6 2 1 & d = 3 5 F 3 A 9 1 3 E 1 D 4 1 2 4 F 3 A C 3 3 C 1 3 8 D A 7 2 9 A 7 & f e n l e i = 1 8 1 6 1 1 1 2 0 3 & s w = A n d r o i d % B F % A A % B 7 % A 2 % B 4 % D 3 % C 8 % E B % C 3 % C 5 % B 5 % B D % B E % A B % C D % A 8

全文地址 = h t t p : / / c x 3 0 0 . 5 r e a d . c o m / i m a g e / s s 2 j p g . d l l ? d i d = n 3 4 & p i d = 4 2 6 8 6 0 0 9 1 0 B 5 7 6 B 7 C 1 0 D F 9 D 9 1 B E D 6 6 2 7 B 7 7 5 6 1 B 2 3 9 E 5 0 6 8 3 1 B 7 6 7 8 9 C D 1 5 1 3 3 5 E 2 7 0 A B 1 6 E B 0 5 5 E C A 8 B D C 0 A 6 D E E 8 2 2 4 7 8 1 5 E 3 2 E 2 E E B 8 4 9 B C D 6 3 B B F 3 2 5 9 4 A 5 F 4 7 F 7 2 2 8 4 5 C E 4 F 9 4 5 7 9 B 1 F 5 3 9 3 A B 3 1 3 7 2 4 2 9 9 B 5 F B 7 5 1 5 E 6 E 3 3 1 A F 8 E 2 E 9 5 8 6 4 D F 6 6 A 3 0 4 E 0 C 1 1 A 9 9 9 8 3 2 1 6 4 E 1 E 7 C 7 B B B 7 C 6 7 F 7 5 2 A 0 E & j i d = /