Android 开发详解

明日科技 编著

吉林大学出版社

版权页

前言

前言

目 录
Content

第 1 章

Android 旅程之起点

随着移动设备的不断普及与发展，相关软件的开发也越来越受到程序员的青睐。目前，移动开发领域以 Android 的发展最为迅猛。作为 Android 开发的起步，本章将先对学习 Android 需要了解的一些基础内容进行简单介绍，然后重点介绍如何搭建 Android 开发环境与创建 Android 第一个应用。

1.1 走近 Android

1.1.1 智能手机操作系统

对于智能手机大家都不陌生，现在大多数人使用的都是智能手机。而智能手机操作系统，就是智能手机所使用的系统，它和计算机的操作系统类似。目前，智能手机操作系统主要包括 Android、iOS、Windows Mobile、Windows Phone、BlackBerry、Symbian、PalmOS 和 Linux 等，各操作系统占据的市场份额如图 1.1 所示。

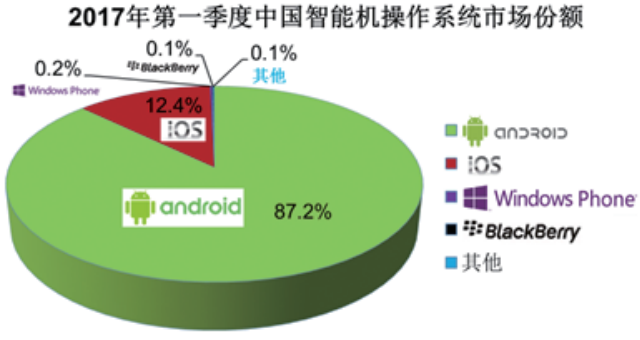


图 1.1　各智能手机操作系统的市场份额

下面将对主流的智能手机操作系统分别进行介绍。

1. Android

Android 是 Google（谷歌）公司发布的基于 Linux 内核的专门为移动设备开发的平台，其中包含了操作系统、中间件和核心应用等。Android 是一个完全免费的手机平台，使用它不需要授权费，

可以完全定制。另外，由于 Android 底层架构使用开源的 Linux 操作系统，同时开放了应用程序开发工具，使所有程序开发人员都可以在统一的、开放的平台上进行开发，从而保证了 Android 应用程序的可移植性。

由于 Android 使用 Java 作为其主要的程序开发语言，所以不少 Java 开发人员加入到此开发阵营，这无疑加快了 Android 队伍的发展速度。

2. iOS

iOS 是苹果公司开发的移动操作系统，主要应用在 iPhone、iPad、iPod touch、MacBook Air 以及 Apple TV 等产品上。iOS 使用 Objective-C 和 Swift 作为程序开发语言，并且苹果公司提供了 SDK（开发工具包），为 iOS 应用程序开发、测试、运行和调试提供工具。

3. Windows

Windows 手机操作系统是 Microsoft（微软）公司推出的移动设备操作系统。开始时命名为 Windows Mobile。由于其界面类似于计算机中使用的 Windows 操作系统，所以用户操作起来比较容易上手。后来，微软公司又推出了 Windows Phone，它是微软公司于 2010 年 10 月推出的新一代移动操作系统。该系统与 Windows Mobile 有很大不同，它具有独特的“方格子”用户界面，并且增加了多点触控和动力感应功能，同时还集成了 Xbox Live 游戏和 Zune 音乐功能。现在，Microsoft 公司又推出了 Windows 10 Mobile，该系统是迄今为止最好的 Windows 手机操作系统。

4. BlackBerry

BlackBerry（黑莓）操作系统是由加拿大的 RIM 公司推出的与黑莓手机配套使用的系统，它提供了手提电脑、文字短信、互联网传真、网页浏览以及其他无线信息服务功能。其中，最主要的特色就是支持电子邮件推送功能，邮件服务器主动将收到的邮件推送到用户的手持设备上，用户不必频繁地连接网络查看是否有新邮件。黑莓系统主要针对商务应用，具有很高的安全性和可靠性。

1.1.2 Android 发展史

Android（发音 [ˈænˌdrɔɪd] ）本义是指“机器人”，标志也是一个机器人，如图 1.2 所示。它是 Google 公司专门为移动设备开发的平台。Android 最早由 Andy Rubin 创办，于 2005 年被搜索巨人 Google 公司收购。2007 年 11 月 5 日，Google 公司正式发布了 Android 1.0 手机操作系统。在 2010 年底，Android 超越称霸 10 年的诺基亚 Symbian 系统，成为全球最受欢迎的智能手机平台。

图 1.2　Android 的标志

在 Android 的发展过程中，已经经历了十多个主要版本的变化，每个版本的代号都是以甜点来命名的，该命名方法开始于从 Android 1.5 版本，并按照首字母排序：纸杯蛋糕、甜甜圈、松饼、冻酸奶、姜饼、蜂巢……。Android 迄今为止发布的主要版本如图 1.3 所示。

图 1.3　Android 发布的主要版本

1.1.3 Android 应用领域

Android 作为移动设备开发的平台不仅可以作为手机的操作系统，而且还可以作为可穿戴设备（如智能手表）和 Android 电视等的操作系统，下面分别进行介绍。

1. Phones/Tablets（手机 / 平板电脑）

Phones/Tablets 是 Google 为智能手机 / 平板电脑打造的操作系统，如图 1.5 所示。它是一个完全免费的开放平台，允许第三方厂商加入和定制。目前，采用 Android 平台的手机厂商主要包括 Google Nexus、HTC、Samsung、LG、Sony、华为、联想和中兴等。

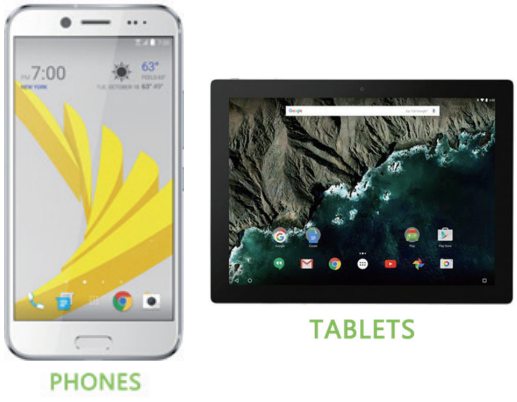


图 1.4　Android Phones/Tablets

图 1.5　Android Wear

2. Android Wear（可穿戴设备）

Android Wear 是 Google 为智能手表等可穿戴设备打造的智能平台。和 Android 一样，Android Wear 也是一个开放平台，它允许第三方厂商加入进来生产各式各样的 Android Wear 兼容设备。目前主要是指智能手表，如图 1.5 所示。

3. Android TV（智能电视）

Android TV 是 Google 在 I/O 会议上宣布的一种名为谷歌电视（Google TV）的替代品，如图 1.6

所示。经过 Google 精心优化的 Android TV 支持 Google Now 语音输入和 D-Pad 遥控，甚至可以连接和匹配游戏手柄。另外，Android TV 完美集成 Google 服务于一体，尤其是 Google Play 上的多媒体内容。例如，Google Play 中成千上万的电影、电视节目和音乐都是 Android TV 的基础内容。

图 1.6　Android TV

图 1.7　Android Auto

4. Android Auto（智能车载）

Android Auto 是 Google 推出的专门为汽车设计的、需要连接 Android 手机使用的设备，旨在取代汽车制造商的原生车载系统来执行 Android 应用与服务，并访问和存取 Android 手机内容，如图 1.7 所示。

1.2 搭建 Android 开发环境

1.2.1 开发环境概述

进行 Android 应用开发需要具备两方面的环境要求。一个是硬件方面，要求 CPU 和内存尽量大。由于开发过程中需要反复重启模拟器，如果每次重启都会消耗几分钟的时间（视机器配置而定），那么将严重影响工作和学习的效率，因此，推荐使用高配置的机器。另一个是软件方面，它需要有相应的开发环境、SDK 及开发工具。下面将从以下两个方面进行介绍。

1. 系统需求

要进行 Android 应用开发，需要有合适的系统环境。表 1.1 中列出了进行 Android 开发所必须的系统环境需求。

表 1.1　进行 Android 开发所必须的系统环境需求

操 作 系 统	系 统 版 本	内　　存	屏幕分辨率
Windows	Microsoft Windows 7/8/10（32 位或 64 位）	最小 4G，推荐 8G	1280×800
Mac OS	Mac OS X 10.10 (Yosemite) 或更高版本，最高为 10.12 (macOS Sierra)	最小 4G，推荐 8G	1280×800
Linux	Linux GNOME 或 KDE（K 桌面环境）	最小 4G，推荐 8G	1280×800

2. 软件需求

要进行 Android 应用开发，除了要有合适的系统环境，还需要有一些软件的支持。通常情况下，需要如图 1.8 所示的这些软件支持。

图 1.8　进行 Android 应用开发所需的软件

在进行 Android 应用开发时，首先需要有 JDK 和 Android SDK 的支持。然后还需要准备合适的开发工具，目前 Android 官方网站推荐的是使用 Android Studio 进行开发。从 Android Studio 2.2 开始在安装 Android Studio 时会自动包含 JDK 和 Android SDK。

说明 JDK（全称为Java Development Kit）是Java开发工具包，包括运行Java程序所必须的JRE（全称为Java Runtime Environment）环境及开发过程中常用的库文件；Android SDK是Android开发工具包，它包括了Android开发相关的API。

3. Android 开发环境的下载与安装过程

Android 开发环境的下载与安装过程如图 1.9 所示。

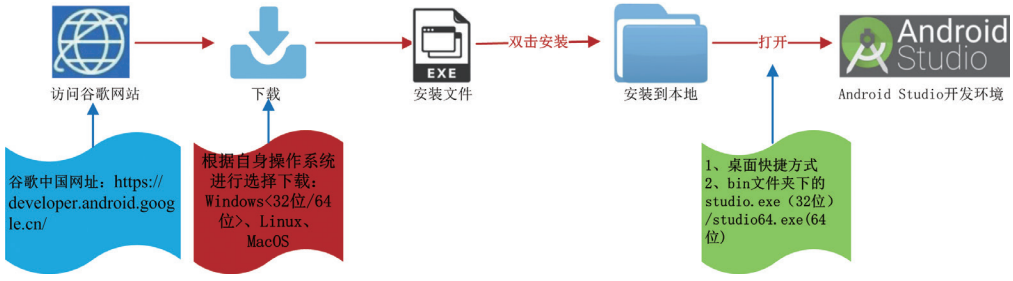


图 1.9　Android 开发环境的下载与安装过程

1.2.2 集成 Android 开发环境的下载

通常情况下，为了提高开发效率，需要使用相应的开发工具。在 Android 发布初期，推荐使用的开发工具是 Eclipse，随着 2015 年 Android Studio 正式版推出，标志着 Google 公司推荐的 Android 开发工具已从 Eclipse 更改为 Android Studio。而且在 Android 的官方网站中，也提供了集成 Android 开发环境的工具包。在该工具包中，不仅包含了开发工具 Android Studio，还包括 JDK 以及最新版本的 Android SDK。下载并安装该工具包后，就可以成功地搭建好 Android 的开发环境。

说明 Android Studio是基于IntelliJ IDEA的Android开发环境。实际上，IntelliJ IDEA是一款非常优秀的Java IDE工具，只是由于它是一款商业工具软件，而且技术文档也少之又少，所以应用并不是很广泛。现在，Google在IntelliJ IDEA的基础上推出的Android Studio则是完全免费的，同时又有Google的技术，相信Android Studio一定会成为开发Android应用的最佳工具。

在 Android 的官方网站中，可以很方便地下载到集成 Android 开发环境的工具包。对于 Android 的官方网站，主要有两个，一个是英文版的，网址为 http://www.android.com/；另一个是中文版的，网址

为 https://developer.android.google.cn/。下面以中文版官方网站为例进行介绍，集成 Android 开发环境的下载步骤如下。

（1）打开浏览器（推荐使用 Google Chrome 浏览器），输入网址 https://developer.android.google.cn/，进入中文版 Android 官方网站首页，如图 1.10 所示。

图 1.10　中文版 Android 官方网站首页

注意 Android官方网站若有打不开的情况下，可以直接到网盘（https://pan.baidu.com/s/ 1dFIQCCT）中复制我们已经下载好的工具包。

（2）单击“获取 Android Studio”超链接，进入到下载 Android Studio 开发工具页面。在该页面中，可以直接单击“下载 ANDROID STUDIO 3.0.1 FOR WINDOWS（683 MB）”按钮，下载 Android Studio3.0 开发工具，如图 1.11 所示。也可以向下滚动页面，找到“选择其他平台”的表格，然后下载所需内容。

图 1.11　下载 Android Studio 和 SDK 工具页面

说明 由于Android Studio的版本更新较快，若官网上的版本与图1.11所示的版本不一致的话，下载最新版本即可。但为了便于读者使用本书，推荐使用我们在云盘中提供的与本书所用版本一致的开发工具。

（3）单击“下载 ANDROID STUDIO 3.0 FOR WINDOWS（683 MB）”按钮，将进入到接受许可协议页面，在该页面中，选中“我已阅读并同意上述条款及条件”复选框，此时“下载 ANDROID STUDIO FOR WINDOWS”按钮将变为可用状态，如图 1.12 所示。

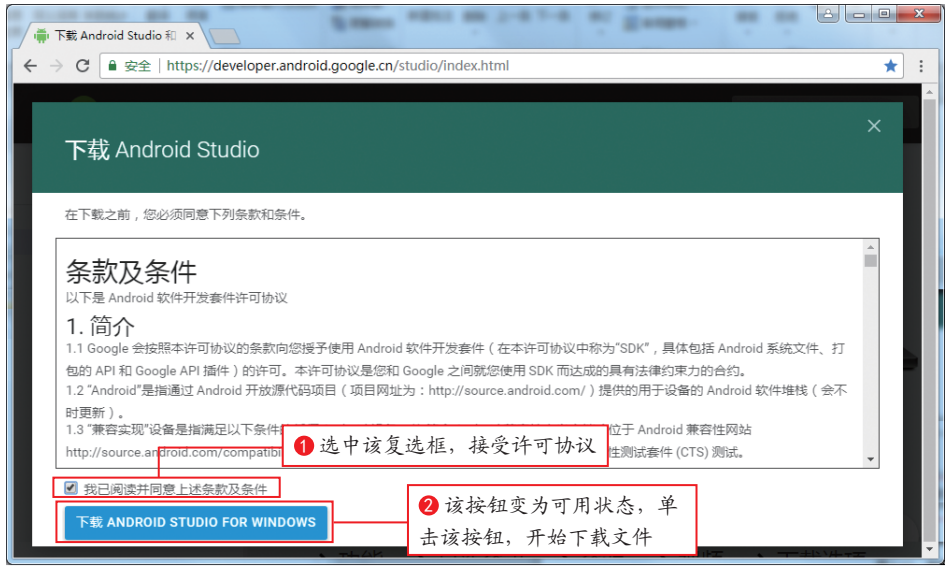


图 1.12　接受许可协议页面

（4）单击“下载 ANDROID STUDIO FOR WINDOWS”按钮，将开始下载 Windows 系统下的 Android 集成开发环境，同时页面会跳转到“安装说明”页面，在该页面中，可以查看安装 Android Studio 的基本步骤。如图 1.13 所示。

图 1.13　正在下载 Android 集成开发环境

（5）下载完成后，浏览器会自动提示“此类型的文件可能会损害您的计算机。您仍然要保留android-studio-….exe 吗？”，此时单击“保留”按钮，保留该文件即可。

（6）下载完成后，将得到一个名称为“android-studio-ide-171.4443003-windows.exe”的安装文件。

说明 由于AndroidStudio开发工具更新速度较快，下载版本可能有所不同。

1.2.3 集成 Android 开发环境的安装

在 Windows 操作系统下，安装 Android 集成开发环境前，需要先检测电脑的 BIOS 中“Intel Virtualization Technology”是否启用，如果没有启用需要先启用它。

说明 每台电脑的BIOS版本不同，开启（Intel（R） Virtualization Technology）状态的位置也会有所不同，这里不进行具体介绍，读者根据自己电脑BIOS版本找到相应的开启位置即可。如果不清楚自己电脑如何开启，可以到网络上搜索一下。

集成 Android 开发环境的安装步骤如下。

（1）双击下载后得到的安装文件“android-studio-ide-171.4443003-windows.exe”，将会弹出“打开文件 - 安全警告”对话框，单击“运行”按钮，将显示如图 1.14 所示的加载进度框。

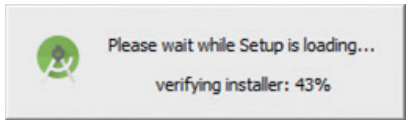


图 1.14　加载进度框

（2）加载完成后，将进入到如图 1.15 所示的安装欢迎对话框。

图 1.15　欢迎安装页面

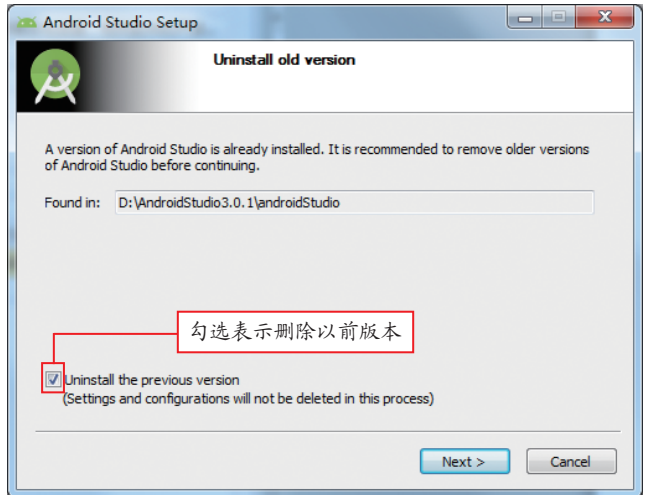


图 1.16　是否卸载旧版本

说明 如果以前安装过旧版本的Android Studio，将会弹出卸载旧版本对话框，如图1.16所示。在该对话框中，勾选Uninstall the previous version复选框（表示删除旧版本），然后单击Next按钮，将会打开如图1.15所示的欢迎安装页面。

（3）在如图 1.15 所示的欢迎安装页面中，单击 Next 按钮，将打开选择安装组件对话框，在该对话框中采用默认设置，如图 1.17 所示。

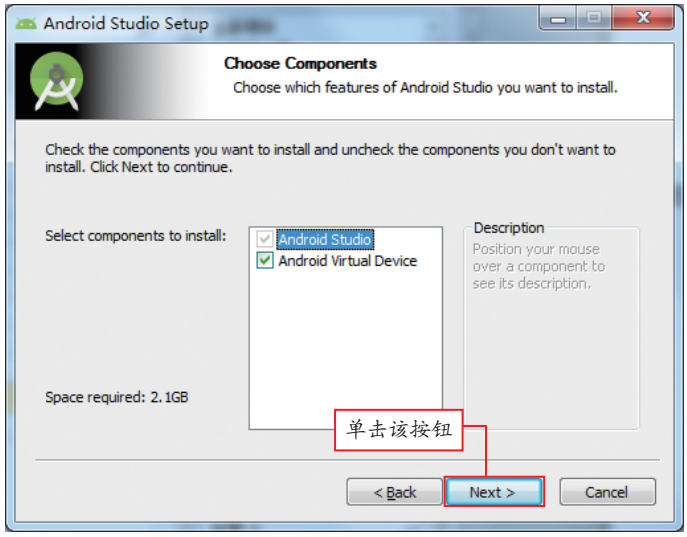


图 1.17　选择安装组件对话框

（4）单击 Next 按钮，将进入到配置安装路径对话框，在该对话框中，指定 Android Studio 的安装路径（如 D:\AndroidStudio3.0.1\androidStudio），如图 1.18 所示。

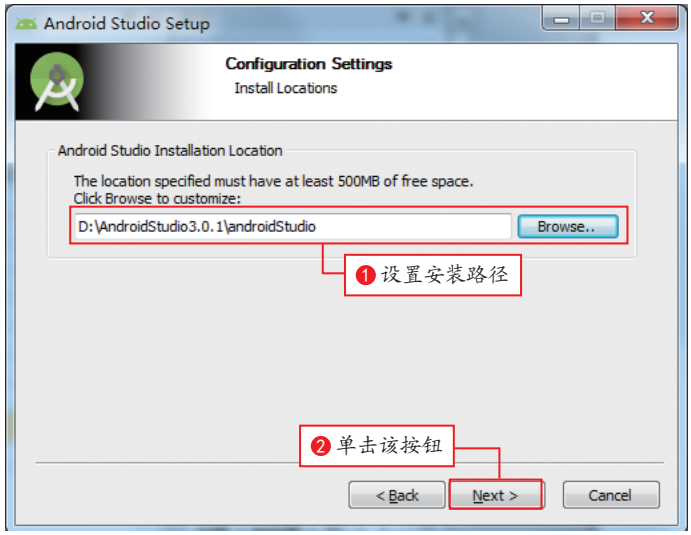


图 1.18　配置安装路径对话框

说明 在配置Android Studio的安装路径时，尽量不要使用C盘，也不要使用包含中文的路径。为了开发工具安装位置的整洁性，可以提前创建好AndroidStudio开发工具所需要的安装路径。（例如，在D盘中创建AndroidStudio3.0.1文件夹，然后在该文件夹内分别创建androidStudio与sdk文件夹用于保存AndroidStudio开发工具与在线下载的SDK工具）。

（6）单击 Next 按钮，将进入到选择开始菜单文件夹对话框，在该对话框中，选择 Android Studio 的快捷方式创建在开始菜单中的哪个文件夹下，默认为新创建的 Android Studio 文件夹，这里采用默认设置，如图 1.19 所示。

说明 在图1.19中，勾选Do not create shortcuts复选框，将不创建快捷方式。

（8）单击 Install 按钮，将显示如图 1.20 所示的安装进度对话框，此时需要等待一段时间。

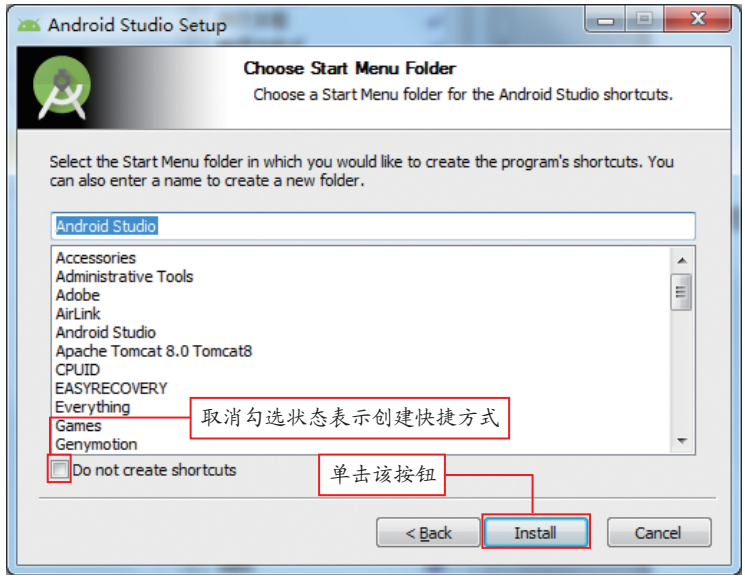


图 1.19　选择快捷方式所在的开始菜单文件夹对话框

（9）安装完成后，将显示如图 1.21 所示的安装完成对话框。单击 Next 按钮，将弹出如图 1.22 所示的对话框。在该对话框中，直接单击 Finish 按钮完成 Android Studio 的安装，并且开启 Android Studio。

说明 在图1.22所示的对话框中，也可以取消Start Android Studio复选框的勾选状态，然后再单击Finish按钮，完成Android Studio的安装。

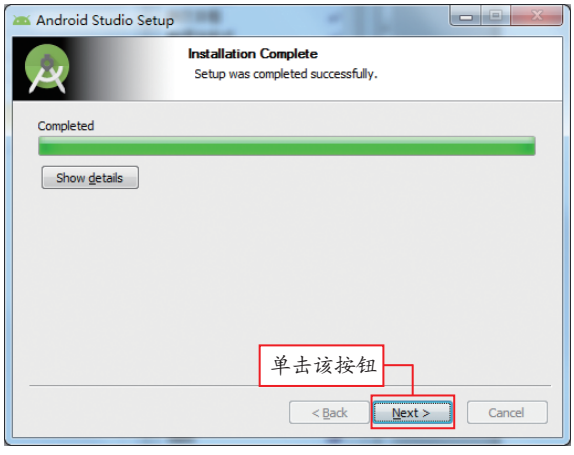


图 1.21　安装成功对话框

图 1.22　安装完成对话框

（10）启动 Android Studio，将弹出如图 1.23 所示的对话框（如果电脑中以前安装过 Android Studio，可能会出现如图 1.24 所示的带 3 个单选按钮的对话框），该对话框用于指定是否从以前版本的 Android Studio 导入设置。这里都选中“Do not import settings”单选按钮，不导入任何设置。

（11）单击 OK 按钮，继续启动 Android Studio，此时会弹出如图 1.25 所示的对话框，该对话框用于询问是否设置代理，如果您有有效的代理地址，可以单击 Setup Proxy 按钮，添加代理地址，否则直接单击 Cancel 按钮。这里直接单击 Cancel 按钮。

图 1.23　询问是否导入设置对话框

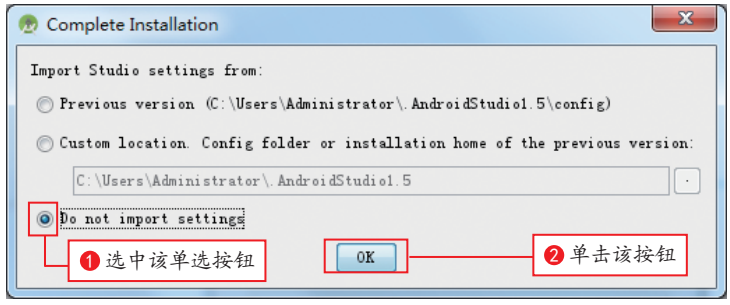


图 1.24　以前安装过 Android Studio 弹出的询问对话框

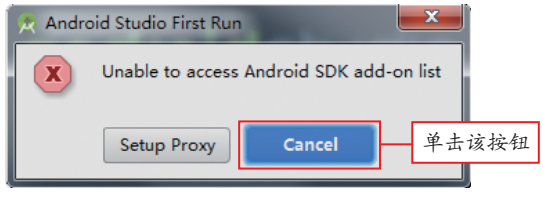


图 1.25　询问是否设置代理

（12）显示如图 1.26 所示的欢迎对话框。检测更新或者安装的组件以及改变默认的主题方案，需要单击 Next 按钮，进行 SDK 与相关组件的下载与安装。

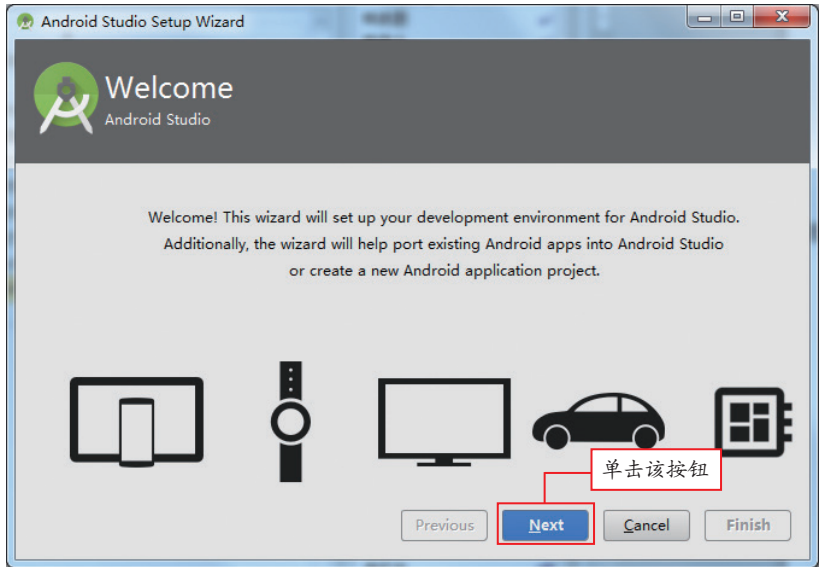


图 1.26　Android Studio 欢迎对话框

（13）在弹出的安装类型对话框中，在该对话框中有两种类型提供选择，Standard 选项为标准选项，该选项中会将默认的 SDK 工具包以及相关组件自动下载并安装到 C 盘中。这里选择 Custom 选项自定义安装路径，如图 1.27 所示。

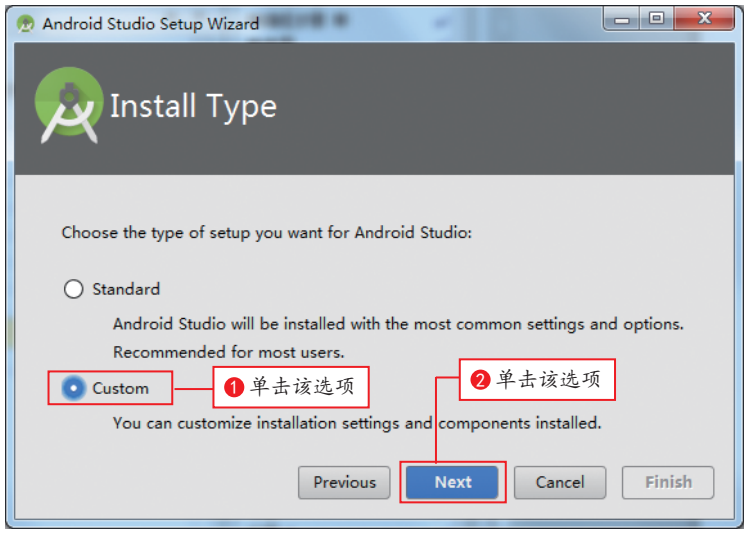


图 1.27　选择安装类型

（14）单击 Next 按钮将显示如图 1.28 所示的选择 UI 主题的对话框，在该对话框中默认选择直接单击 Next 按钮即可。

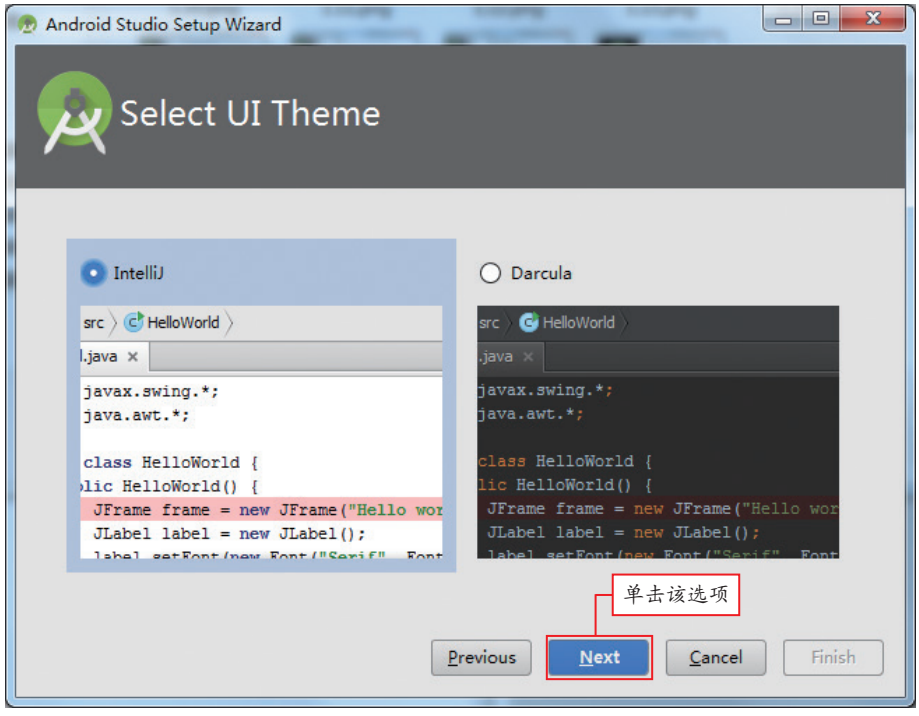


图 1.28　选择 UI 主题对话框

（15）显示 SDK 组件安装对话框，在该对话框中勾选“Android Virtual Device”的 SDK 组件，其它默认勾选即可，然后将 SDK 安装路径修改，最后单击 Next 按钮。如图 1.29 所示。

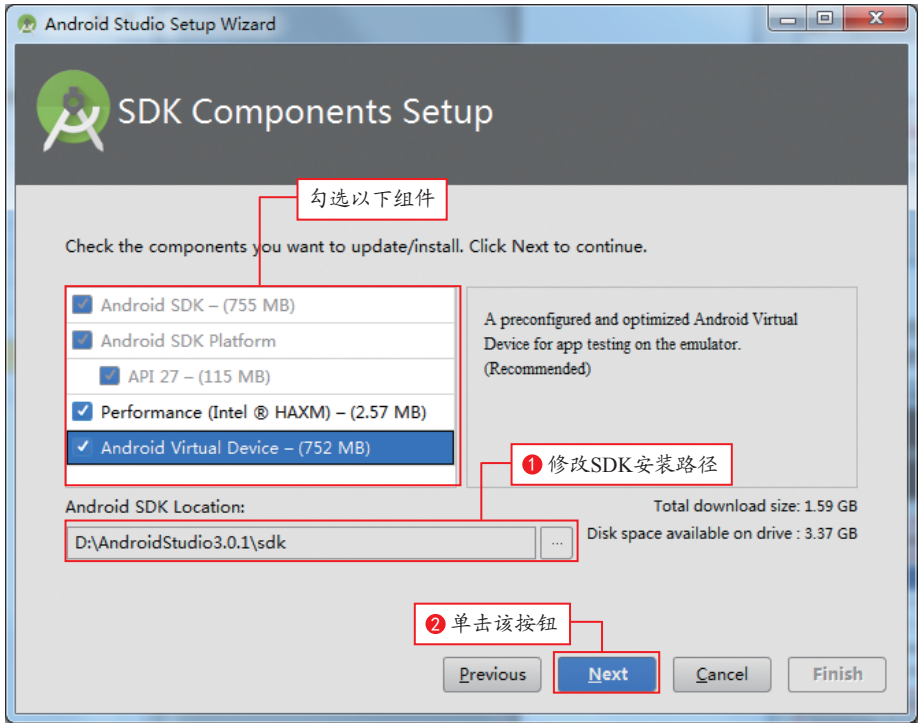


图 1.29　SDK 组件安装对话框

（16）显示模拟器设置对话框，在该对话框中为模拟器设置硬件加速的最大内存，此处默认设置，使用系统推荐即可，然后单击 Next 按钮。如图 1.30 所示。

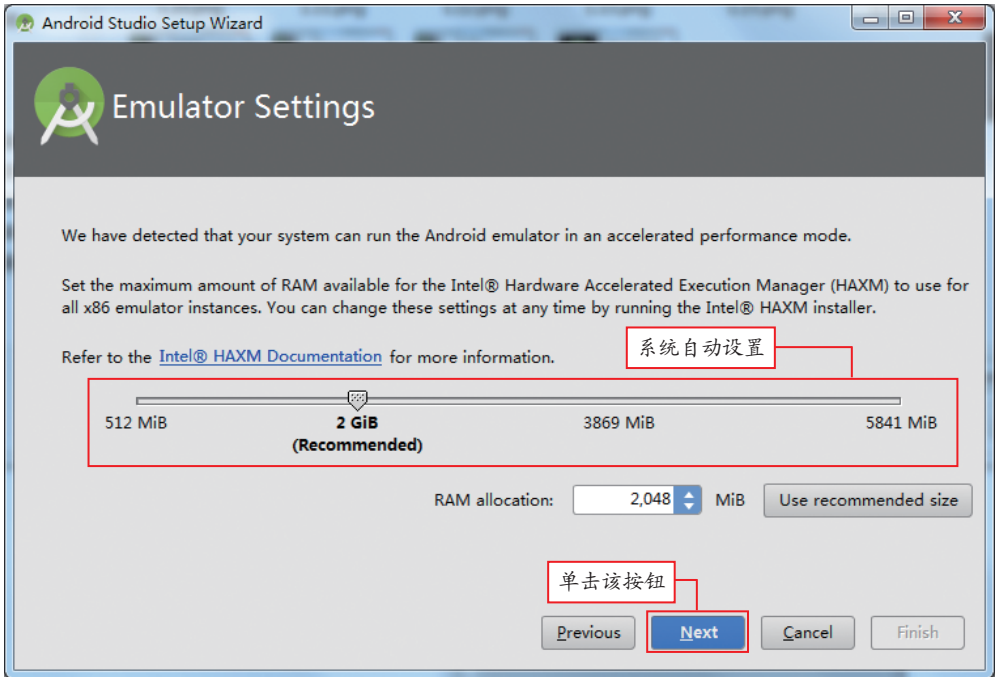


图 1.30　模拟器设置对话框

（17）显示验证设置对话框，在该对话框中可以看到当前需要下载的 SDK 等相关组件，直接

单击 Finish 按钮，显示下载组件对话框，在该对话框中进行组件的下载与安装，安装完成后单击 Finish 即可，如图 1.32 所示。

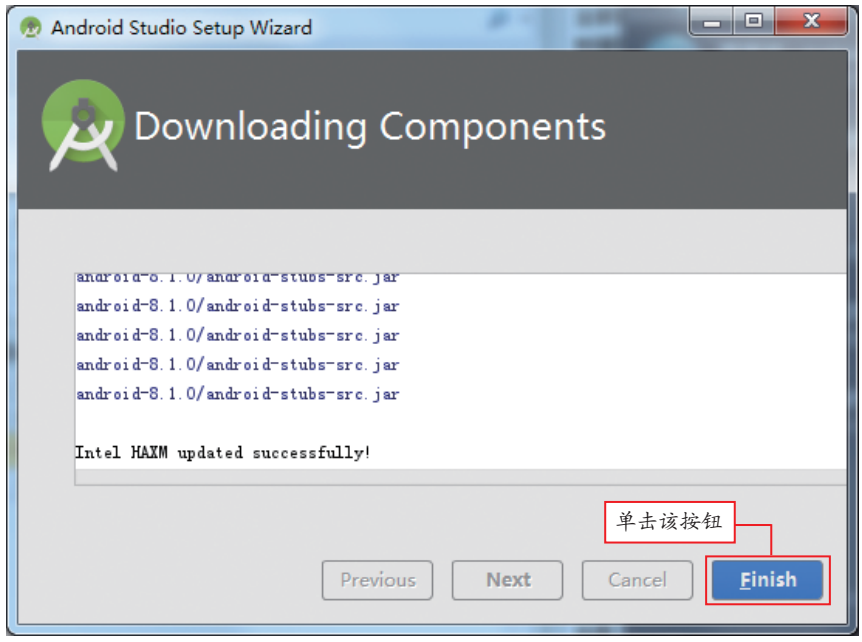


图 1.31　验证设置对话框

（19）显示如图 1.33 所示的欢迎对话框，这时就表示 Android Studio 已经安装完毕，并且启动成功。

图 1.33　欢迎对话框

AndroidStudio 安装完成后，其安装目录结构如图 1.34 所示。

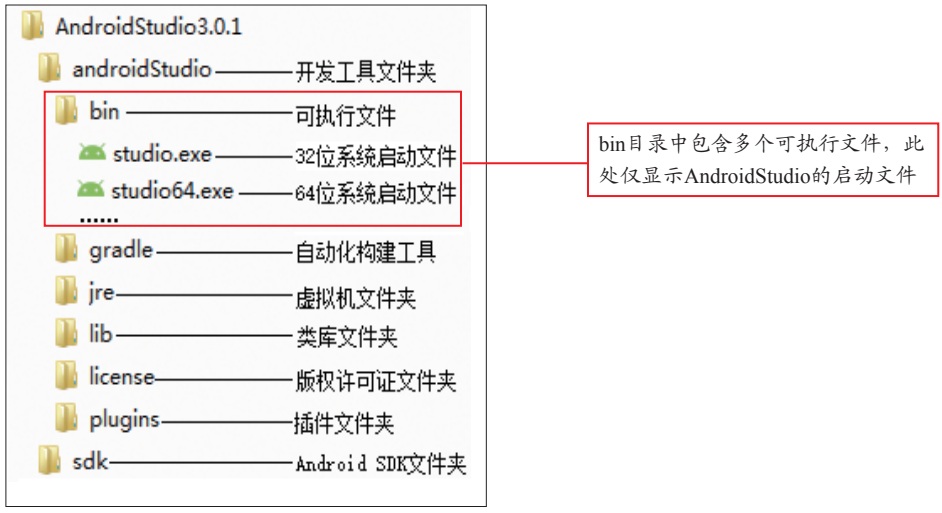


图 1.34　AndroidStudio 安装目录结构

1.3　与 Android 应用初次相见

作为程序开发人员，学习新语言的第一个应用一般都是输出“Hello World”。学习 Android 开发也不例外，第一个应用也是从输出“Hello World”开始。下面将介绍如何编写并运行一个 Android 应用。

1.3.1　创建 Android 应用程序

Android Studio 安装完成后，如果还没有创建项目，将进入到欢迎对话框。在该对话框中，可以创建新项目、打开已经存在的项目、导入项目等。在 Android Studio 中，一个 project（项目）相当于一个工作空间，一个工作空间中可以包含多个 Module（模块），每个 Module 对应一个 Android 应用。下面将通过一个具体的实例来介绍如何创建项目，即创建第一个 Android 应用。

说明　在首次创建项目时，需要联网加载数据，所以此时需要保证电脑可以正常连接互联网。

例 1.1　在屏幕上输出文字“Hello World”

在 Android Studio 中创建项目，名称为“第一个 Android 应用”，具体步骤如下：

（1）在 Android Studio 的欢迎对话框中，单击 Start a new Android Studio project 按钮，进入到 Create New Project 对话框中，在该对话框的 Application name 文本框中输入应用程序名称（例如“第一个 Android 应用”）；

在 Company domain 文本框中输入公司域名（如 mingrisoft.com），将自动生成相应的 Package name（包名），并且默认为不可修改状态，如果想要修改，可以单击 Package name 右侧的 Edit 按钮，使其变为可编辑状态，然后输入想要的包名（如 com.mingrisoft），单击 Done 按钮即可；

在 Project Location 文本框中输入项目保存的位置（如 F:\android_studio\AndroidStudioProjects），如图 1.35 所示。

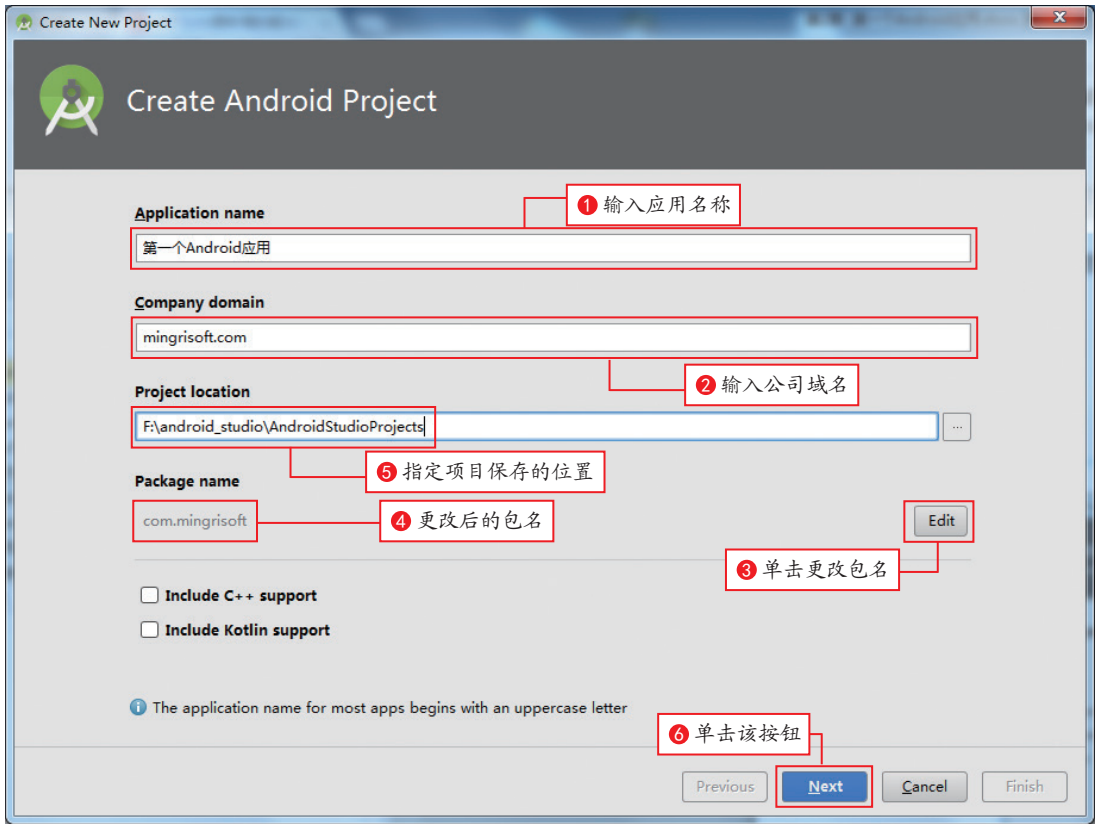


图 1.35　创建新项目对话框

说明 Include C++ support与Include Kotlin support分别为支持C++语言与Kotlin语言。

注意

① 设置Package name时，一定不能使用中文（如com.明日科技）和空格，或者单纯的数字（如com.mr.03），并且也不能以“.”结束，否则项目将无法创建。

②在设置Project location时，一定不能使用中文（如D:\第一个Android应用）和空格。否则项目将无法创建。

（2）单击 Next 按钮，将进入选择目标设备对话框，在该对话框中，首先选中 Phone and tablet 复选框，然后在选择最小 SDK 版本的 Minimum SDK 下拉列表框中选择默认的 API 15，即 Android 4.0.3，如图 1.36 所示。

注意 Minimum SDK用于指定应用程序运行时，所需设备的最低SDK版本，如果所用设备低于这个版本，那么应用程序将不能在该设备上运行，所以这里一般设置得要比所用的SDK版本低。

（3）单击 Next 按钮，将进入选择创建 Activity 类型对话框，在该对话框中，将列出一些用于创建 Activity 的模板，我们可以根据需要进行选择，也可以选择不创建 Activity（即选择 Add No Activity），这里我们选择创建一个空白的 Activity，即 Empty Activity，如图 1.37 所示。

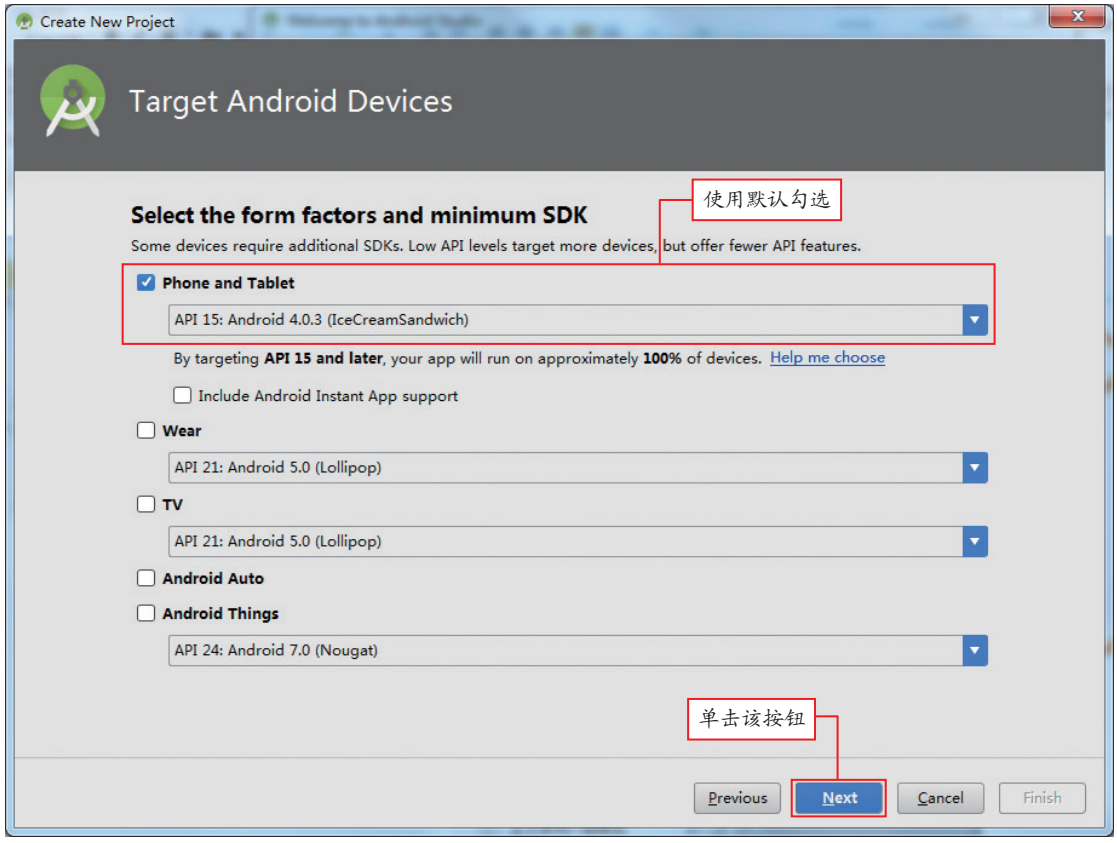


图 1.36　选择目标设备对话框

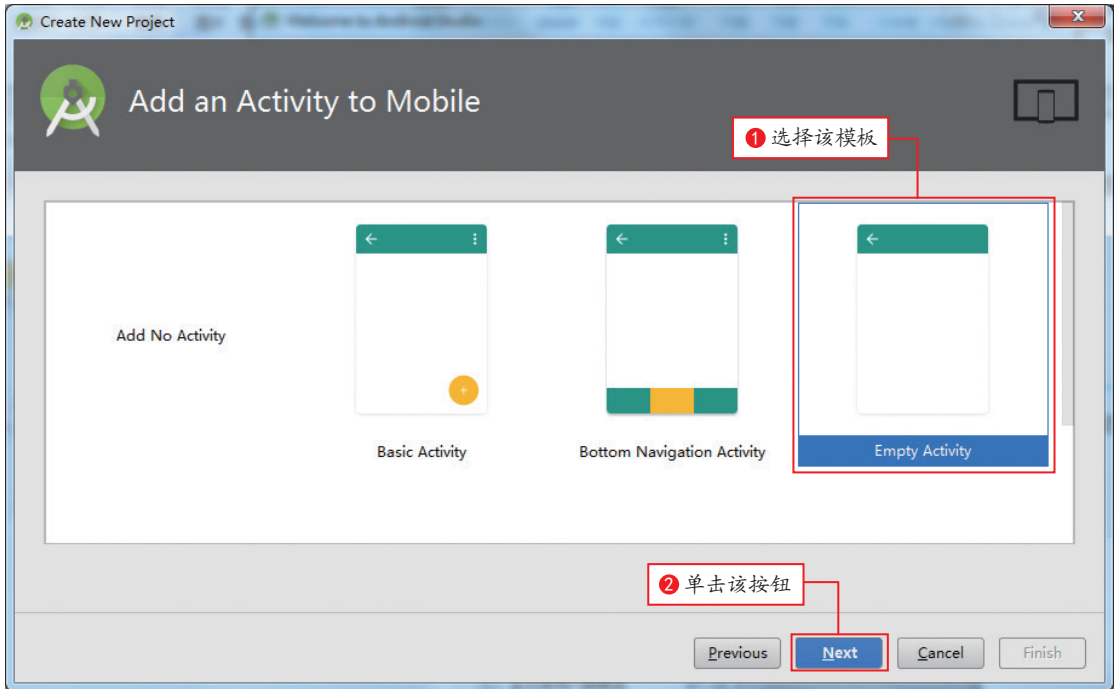


图 1.37　选择创建 Activity 的类型

（4）单击 Next 按钮，将进入到自定义 Activity 对话框，在该对话框中，可以设置自动创建的 Activity 的类名和布局文件名称，这里采用默认设置，如图 1.38 所示。

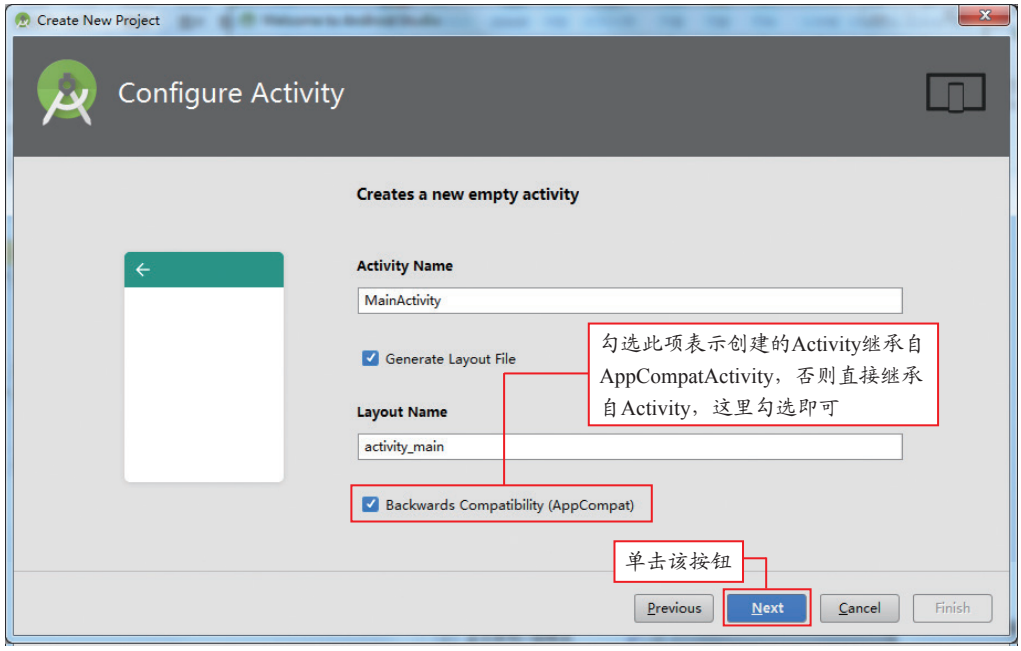


图 1.38　自定义 Activity

（5）单击 Next 按钮将显示安装要求组件的对话框，在该对话框中加载完成后单击 Finish 按钮如图 1.39 所示。

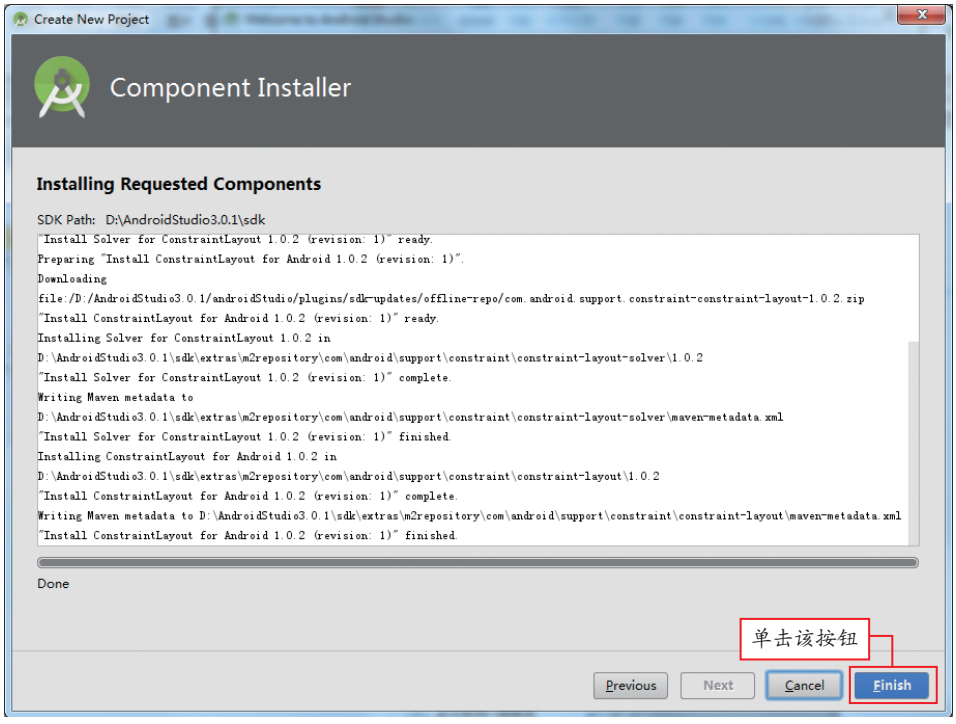


图 1.39　安装要求组件的对话框

（6）单击 Finish 按钮，将显示如图 1.40 所示的创建进度对话框（加载时间可能会比较长，请耐心等待），创建完成后，该对话框自动消失，同时打开该项目。

图 1.40 创建进度对话框

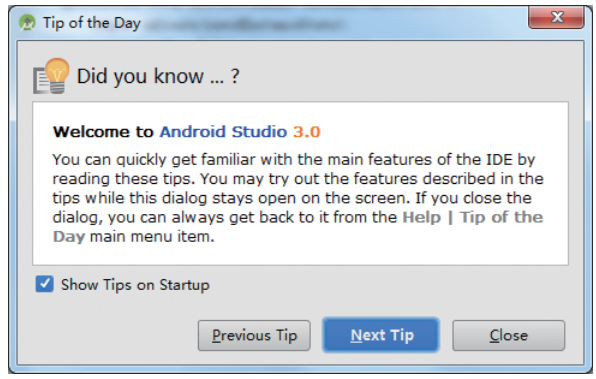


图 1.41 小贴士对话框

（7）在默认情况下，启动项目时会弹出如图 1.41 所示的小贴士，单击 Close 按钮，关闭即可进入到 Android Studio 的主页，同时打开创建好的项目，此时主窗口底部将显示无法找到 SDK 需要的‘android-26’提示，此处单击“Install missing platform(s) and sync project”超链接，如图 1.42 所示。

图 1.42 无法找到 SDK 需要的‘android-26’

（8）单击超链接后将显示如图 1.43 所示的下载许可协议对话框，在该对话框中单击“Accept”接受许可协议，然后单击 Next 按钮即可下载 SDK 中所需要的‘android-26’。

图 1.43 接受下载许可协议

（9）下载完成后单击 Finish 按钮。如图 1.44 所示。

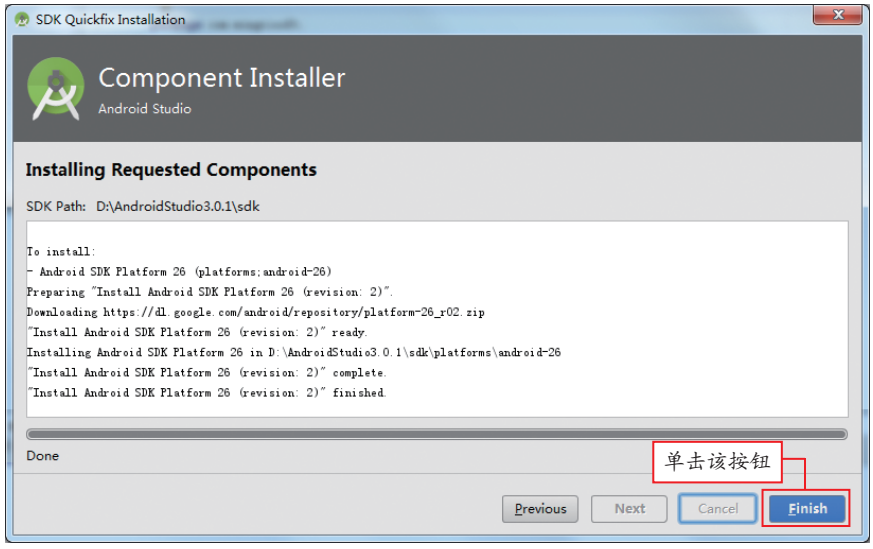


图 1.44 完成 SDK‘android-26’的下载与安装

（10）单击 Finish 按钮后，主窗口底部将显示无法找到构建工具 26.0.2 提示，此处同样单击“Install Build Tools 26.0.2 and sync project”超链接，如图 1.45 所示。

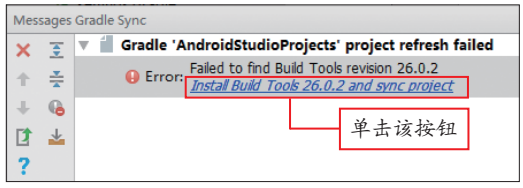


图 1.45 无法找到构建工具 26.0.2

（11）单击超链接后将显示下载构建工具 26.0.2 的对话框，下载完成后单击 Finish 按钮如图 1.46 所示。

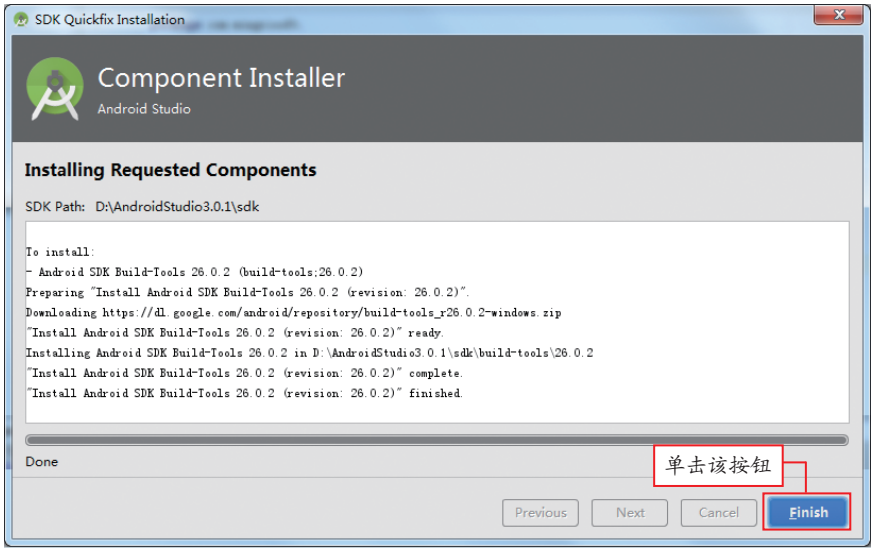


图 1.46 完成构建工具 26.0.2 的下载与安装

单击 Finish 按钮后，Android Studio 将自动同步 SDK 安装后的组件，同步完成后，默认显示 MainActivity.java 文件的内容，如图 1.47 所示。

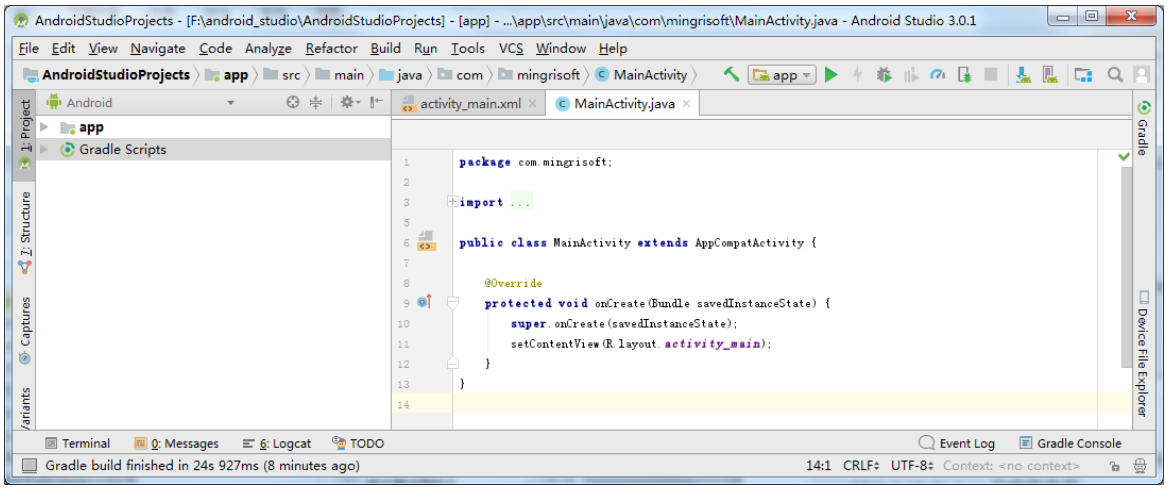


图 1.47　Android Studio 的主页

说明 在使用Android Studio创建一个项目时，默认会创建一个名称为“app”的Module（一个Module就是一个Android应用），展开app结构如图1.48所示。

图 1.48　默认创建的 Module

说明 在使用Android Studio创建过一个项目后，再次创建项目时，需要在已经打开的项目中，选择File →New→ New Project菜单项，打开创建新项目对话框。

1.3.2 Android 项目结构

在 Android Studio 中，提供了如图 1.49 所示的多种项目结构类型。其中，最常用的是 Android 和 Project。由于 Android 项目结构类型是创建项目后默认采用的，所以这里我们就使用这种结构类型。

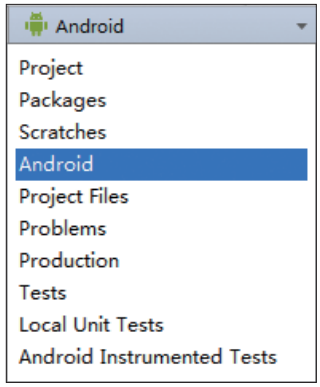


图 1.49　Android Studio 提供的项目结构类型

采用 Android 项目结构时，各个子节点的作用如图 1.50 所示。

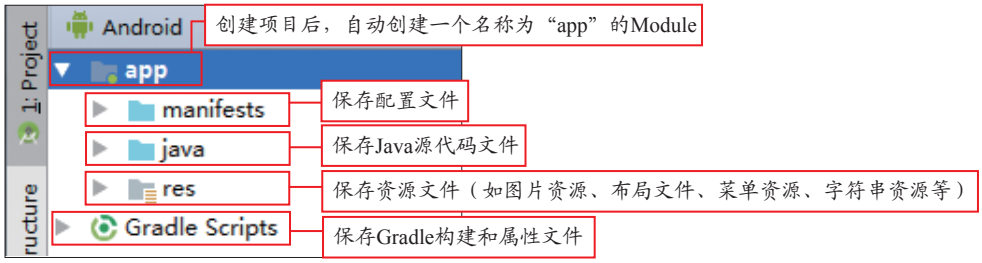


图 1.50　Android 项目结构类型说明

下面再对一些常用的节点进行详细介绍。

1. manifests 节点

manifests 节点用于显示 Android 应用程序的配置文件。通常情况下，每个 Android 应用程序必须包含一个 AndroidManifest.xml 文件，位于 manifests 节点下。它是整个 Android 应用的全局描述文件。在该文件内，需要标明应用的名称、使用图标、Activity 和 Service 等信息，否则程序不能正常启动。例如，“第一个 Android 应用”中的 AndroidManifest.xml 文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml 文件中的重要元素及说明如表 2.1 所示。

表 1.2　AndroidManifest.xml 文件中的重要元素及说明

元　素	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，其属性值为 http://schemas.android.com/apk/res/android，表示 Android 中的各种标准属性能在该 xml 文件中使用，它提供了大部分元素中的数据
package	声明应用程序包

续表

元　素	说　明
application	包含 package 中 application 级别组件声明的根节点，一个 manifest 中可以包含零个或者一个该元素
android:icon	应用程序图标
android:label	应用程序标签，即为应用程序指定名称
android:theme	应用程序采用的主题，例如，Android Studio 创建的项目默认采用 @style/AppTheme
activity	与用户交互的主要工具，它是用户打开一个应用程序的初始页面
intent-filter	配置 Intent 过滤器
action	组件支持的 Intent Action
category	组件支持的 Intent Category，这里通常用来指定应用程序默认启动的 Activity

注意 在Android程序中，每一个Activity都需要在AndroidManifest.xml文件中有一个对应的<activity>标记。

2. java 节点

java 节点用于显示包含了 Android 程序的所有包及源文件（.java），例如，2.1 节的“第一个 Android 应用”项目的 java 节点展开效果如图 1.51 所示。

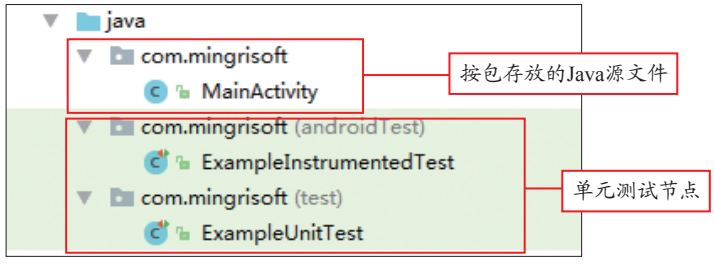


图 1.51　java 节点

默认生成的 MainActivity.java 文件的关键代码如下：

```
01 package com.mingrisoft;                              //指定包
02 //导入Support v7库中的AppCompatActivity类
03 import android.support.v7.app.AppCompatActivity;
04 import android.os.Bundle;                             //导入Bundle类
05 public class MainActivity extends AppCompatActivity {
06     //该方法在创建Activity时被回调，用于对该Activity执行初始化
07     @Override
08     protected void onCreate(Bundle savedInstanceState) {
09         super.onCreate(savedInstanceState);
10         setContentView(R.layout.activity_main);
11     }
12 }
```

从上面的代码中可以看出，Android Studio 创建的 MainActivity 类默认继承自 AppCompatActivity 类（继承自 AppCompatActivity 类的 Activity 将带有 Action Bar），并且在该类中，重写了 Activity 类中的 onCreate() 方法，在 onCreate() 方法中通过 setContentView(R.layout.activity_main) 方法设置当前的 Activity 要显示的布局文件。

说明 这里使用R.layout.activity_main来获取layout目录中的activity_main.xml布局文件，这是因为在Android程序中，每个资源都会在R.java文件中生成一个索引，而通过这个索引，开发人员可以很方便的调用Android程序中的资源文件。

注意 应用Android Studio创建的项目，R.java文件位于新创建应用的<应用名称>build\generated\source\r\debug\<包路径>目录下。R.java文件是只读文件，开发人员不能对其进行修改，当res包中资源发生变化时，该文件会自动修改。

3. res 节点

该节点用来显示保存在 res 目录下的资源文件，当 res 目录中的文件发生变化时，R 文件会自动修改。在 res 目录中还包括一些子目录，下面将对这些子目录进行详细说明。

◆ drawable 子目录

drawable 子目录通常用来保存图片资源（如 PNG\JPEG\GIF 图片、9-Patch 图片或者 Shape 资源文件等）。

◆ layout 子目录

layout 子目录主要用来存储 Android 程序中的布局文件，在创建 Android 程序时，会默认生成一个 activity_main.xml 布局文件。例如，“第一个 Android 应用”项目的 layout 子目录的结构如图 1.52 所示。

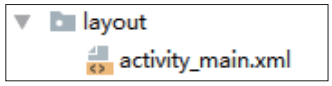


图 1.52　layout 子目录的结构

在 Android Stuido 中打开 activity_main.xml 布局文件的代码如图 1.53 所示。

activity_main.xml ×

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context="com.mingrisoft.MainActivity">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hello World!"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent" />
17
18 </android.support.constraint.ConstraintLayout>
```

图 1.53　在 Android Stuido 中打开 activity_main.xml 文件的代码

activity_main.xml 布局文件中的重要元素及说明如表 1.2 所示。

表 1.2 activity_main.xml 布局文件中的重要元素及说明

元 素	说 明
ConstraintLayout	布局管理器
xmlns:android	包含命名空间的声明，其属性值为 http://schemas.android.com/apk/res/android，表示 Android 中的各种标准属性能在该 xml 文件中使用，它提供了大部分元素中的数据，该属性一定不能省略
xmlns:tools	指定布局的默认工具
android:layout_width	指定当前视图在屏幕上所占的宽度
android:layout_height	指定当前视图在屏幕上所占的高度
TextView	文本框组件，用来显示文本
android:text	文本框组件显示的文本

技巧 开发人员在指定各个元素的属性值时，可以按下<Ctrl+Alt + Space>快捷键来显示帮助列表，然后在帮助列表中选择系统提供的值，如图1.54所示。

另外，Android Studio 提供了可视化的布局编辑器来辅助用户开发布局文件，如图 1.55 所示。在该编辑器内，可以通过拖动组件实现界面布局。

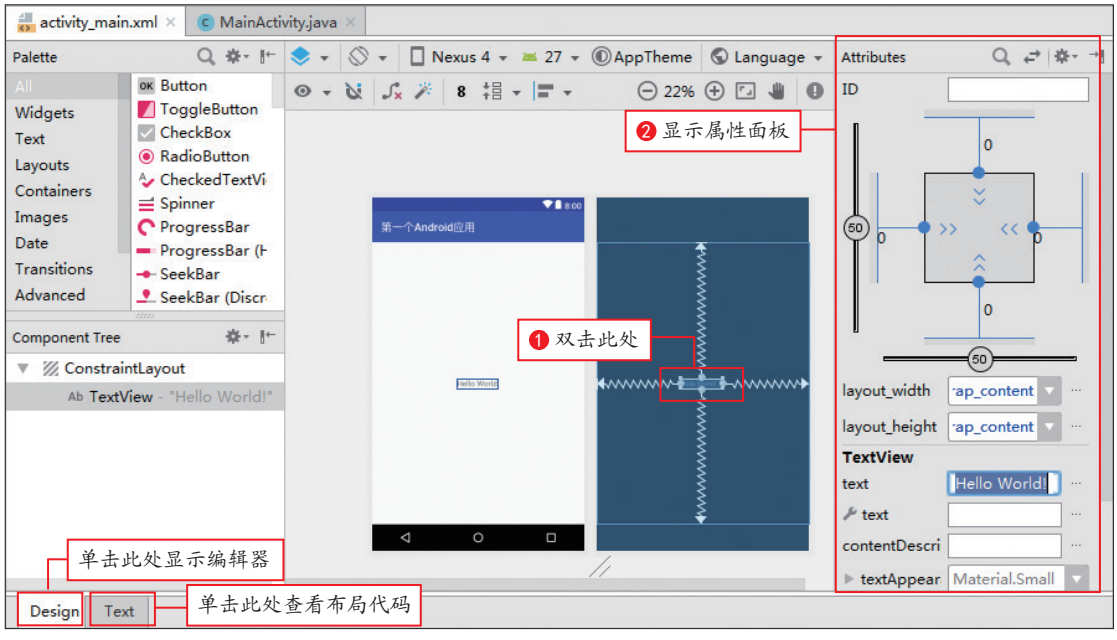


图 1.55 布局编辑器

◆ mipmap 子目录

mipmap 子目录用于保存项目中应用的启动图标。为了保证良好的用户体验，需要为不同的分辨率提供不同的图片，并且分别存放在不同的目录中。通常情况下，Android Studio 会自动创建 mipmap-xxxhdpi（超超超高）、mipmap-xxhdpi（超超高）、mipmap-xhdpi（超高）、mipmap-hdpi

（高）和 mipmap-mdpi（中）等 5 个目录，分别用于存放超超超高分辨率图片、超超高分辨率图片、超高分辨率图片、高分辨率图片和中分辨率图片。并且会自动创建对应 5 种分辨率的启动图标文件（ic_launcher.png）以及 mipmap-anydpi-v26 目录中的两个自适应图标文件，如图 1.56 所示。

图 1.56　mipmap 子目录

说明 由于从android 7.1.1开始系统中应用图标为圆形图标，所以为了让其他版本的系统正常使用高版本开发的应用，mipmap子目录中将自动生成圆形与方形两种图标。

◆ values 子目录

values 子目录通常用于保存应用中使用的字符串、样式和颜色资源。例如，“第一个 Android 应用”的 values 子目录的结构如图 1.57 所示。

图 1.57　values 子目录的结构

在开发国际化程序时，这种方式尤为方便。strings.xml 文件的代码如下：

```
<resources>
    <string name="app_name">第一个Android应用</string>
</resources>
```

说明 关于如何实现国际化程序，将在本书的12.7节进行详细介绍。

1.3.3 使用 Android 模拟器

Android 模拟器是 Google 官方提供的一款运行 Android 程序的虚拟机，可以模拟手机 / 平板电脑等设备。作为 Android 开发人员，不管你有没有基于 Android 操作系统的设备，都可能需要在 Android 模拟器上测试自己开发的 Android 程序。

由于启动 Android 模拟器需要配置 AVD，所以在运行 Android 程序之前，首先需要创建 AVD。创建 AVD 并启动 Android 模拟器的步骤如下。

说明 AVD是Android Virtual Device的简称。通过它可以对Android模拟器进行自定义的配置，能够配置Android模拟器的硬件列表、模拟器的外观、支持的Android系统版本、附加SDK库和存储设置等。开发人员配置好AVD以后就可以按照这些配置来模拟真实的设备。

（1）单击 Android Studio 工具栏上 图标，显示 AVD 管理器对话框，如图 1.58 所示。

图 1.58 AVD 管理器对话框

（2）单击 Create Virtual Device... 按钮，将弹出“Select Hardware”对话框，在该对话框中，选择想要模拟的设备。例如，如果我们要模拟 3.2 寸 HVGA 的设备，那么可以选择“32” HVGA slider(ADP1)”，如图 1.59 所示。

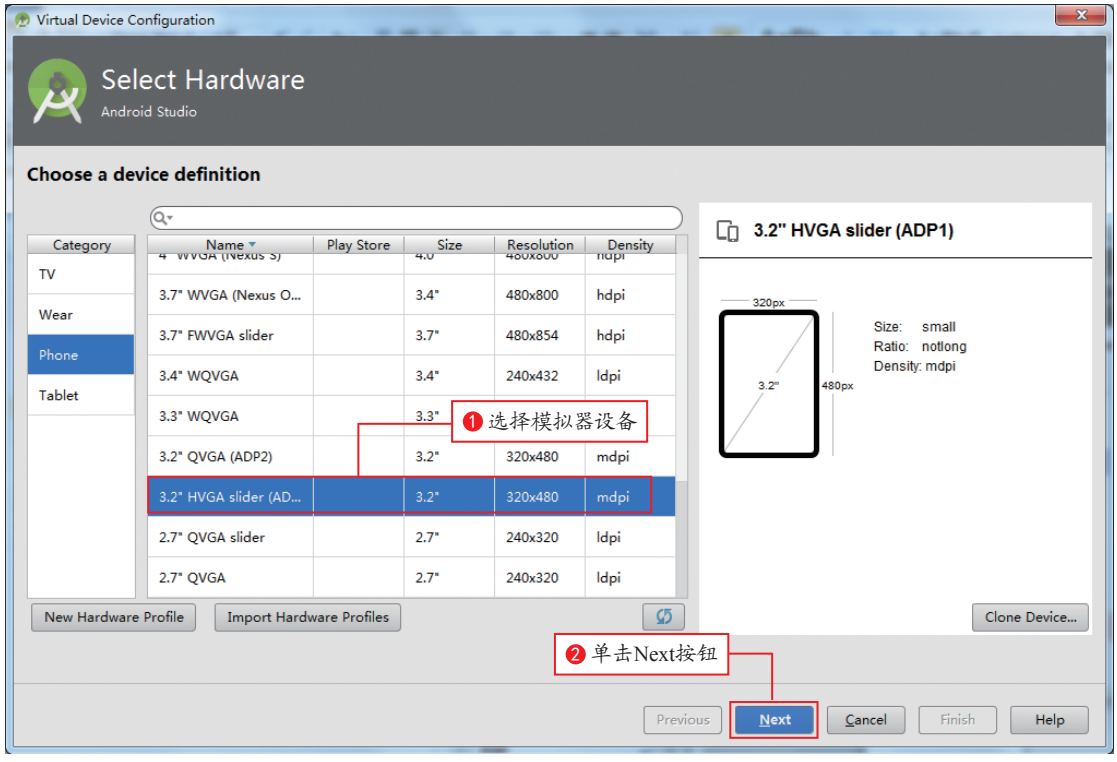


图 1.59 创建 AVD 对话框

（3）单击 Next 按钮，将弹出选择系统镜像对话框，在该对话框中，默认情况下，只能使用已经下载好的 ABI 为 x86 的系统镜像，如图 1.60 所示，这里选择它即可。

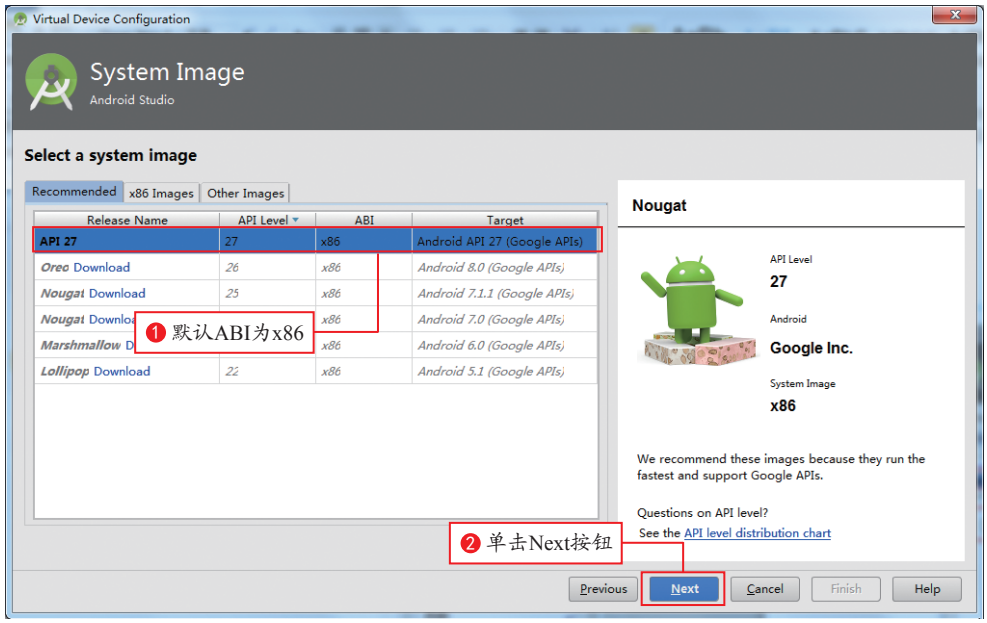


图 1.60　选择系统镜像对话框

说明　如果没有可选择的系统镜像，也可以单击Download下载需要的其他版本的系统镜像。

（4）单击 Next 按钮，将弹出验证配置对话框，在该对话框的 AVD Name 文本框中输入 AVD 名称，这里设置为 AVD，其他采用默认，如图 1.61 所示。

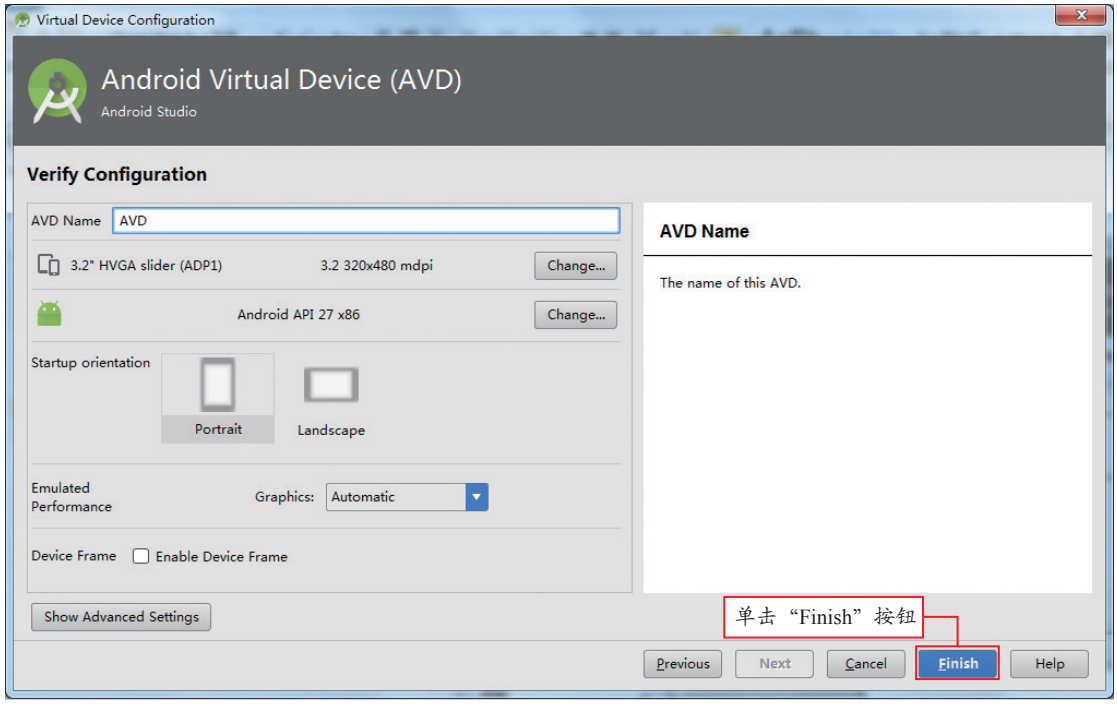


图 1.61　验证配置对话框

（5）单击 Finish 按钮，完成 AVD 的创建。AVD 创建完成后，将显示在 AVD Manager 对话框中，如图 1.62 所示。

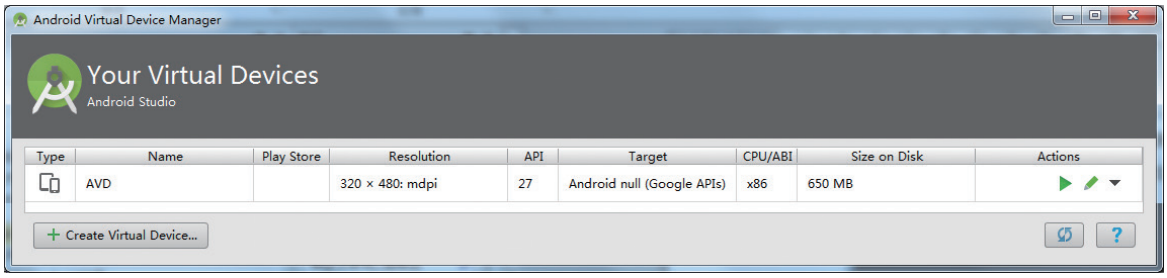


图 1.62　创建完成的 AVD

说明 在图1.62所示的AVD Manager对话框中，单击▶按钮，可以启动AVD；单击✎按钮，可以编辑当前AVD的配置信息；单击▾按钮，将弹出如图1.63所示的快捷菜单，通过该菜单可以实现查看AVD的详细配置、删除AVD或者停止已经启动的模拟器等操作；单击 + Create Virtual Device... 按钮，可以再创建新的AVD。

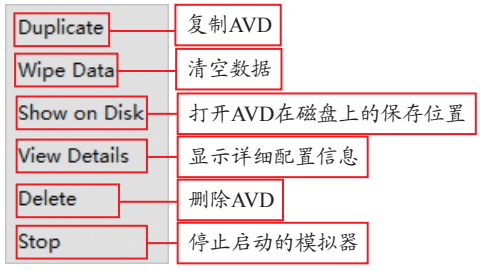


图 1.63　AVD 快捷菜单

（6）单击▶按钮，即可启动 AVD。第一次启动将显示如图 1.64 所示界面。

图 1.64　第一次启动时的界面

1.3.4 运行 Android 应用

创建 Android 应用程序后，还需要运行查看其显示结果。要运行 Android 应用程序，可以有两种方法，一种是通过 Android 提供的模拟器来运行应用；另一种是在电脑上连接手机，然后通过该手机来运行应用。下面分别进行介绍。

1 使用模拟器运行 Android 应用

使用模拟器来运行 Android 应用适用于没有 Android 手机，或者测试各系统版本下应用的兼容性的情况。在 Android Studio 中，通过模拟器运行 1.3.1 节编写的“第一个 Android 应用”的具体步骤如下。

（1）启动模拟器。

（2）在工具栏中找到 app 下拉列表框，选择要运行的应用（这里为 app），再单击右侧的▶按钮，将弹出如图 1.65 所示的选择设备对话框。

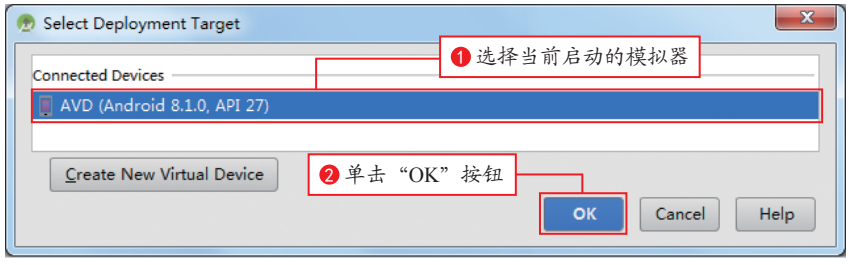


图 1.65 选择设备对话框

（3）单击 OK 按钮，启动模拟器，并运行程序。启动完毕后，在模拟器中将显示刚刚创建的应用，运行效果如图 1.66 所示。

图 1.66 应用程序的运行效果

2 连接手机运行 Android 应用

在上一节中我们已经介绍了如何通过模拟器来运行 Android 应用，本节我们来学习如何连接手机运行 Android 应用。该方法更加符合用户场景，易于发现一些与硬件有关的问题，但是一般手机的系统很难与最新的 SDK 版本统一，所以对于一些 SDK 的新特性还是需要使用模拟器测试。

通过手机来运行 1.3.1 节编写的“第一个 Android 应用”，具体步骤如下。

（1）将 Android 系统的手机连接到电脑上，通常情况下电脑桌面的右下角将提示正在安装设备驱动程序软件，如图 1.67 所示。

图 1.67 安装设备驱动

（2）下载电脑管家并安装，安装完成后电脑管家将自动显示主界面。在主界面中首先单击“工具箱”按钮，然后单击“应用宝”进行应用宝的下载与安装，如图 1.68 所示。

图 1.68　安装应用宝软件

（3）应用宝安装完成后将自动显示如图 1.69 所示的授权电脑管理手机的提示对话框。

图 1.69　授权电脑管理手机

同时，在手机上将显示如图 1.70 所示的是否允许 USB 调试的对话框。

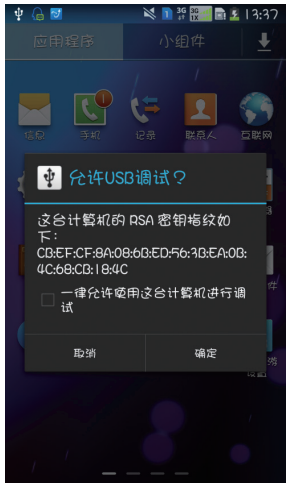


图 1.70　手机中显示的是否允许 USB 调试的对话框

说明　对于不同的手机，显示的是否允许USB调试的方式可能不同，只要根据自己的手机提示进行选择就可以。

（4）在图 1.71 所示的手机界面中，单击“确定”按钮，允许 USB 调试，在电脑中将显示如图 1.70 所示的电脑与手机连接窗口。数秒后将显示如图 1.72 所示的连接成功窗口。

图 1.71　电脑与手机进行连接

图 1.72　连接成功窗口

（5）返回到 Android Studio 中，在工具栏中，找到 app 下拉列表框，然后单击要运行的项目（这里为 app），再单击▶按钮，将显示如图 1.73 所示的选择要运行的设备对话框。在该对话框中，选中表格中显示的已经运行的设备三星手机，单击 OK 按钮即可在手机上运行该应用。

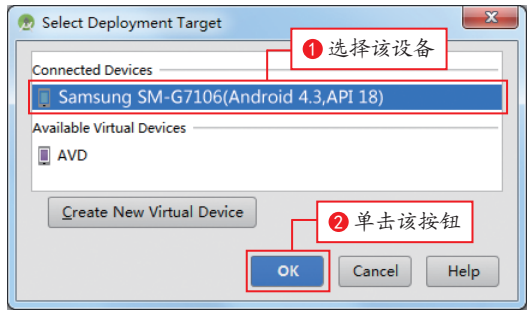


图 1.73　选择要运行的设备对话框

（6）项目运行后，在手机上将显示运行效果，如图 1.74 所示。

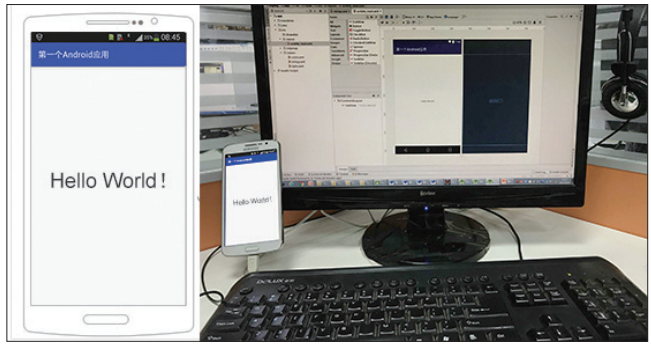


图 1.74　应用的运行效果

说明　根据手机品牌的不同，有的会弹出该应用没有申请权限对话框。这时只需单击“确定”按钮，即可运行本程序。

1.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。

第 2 章

Android Studio 的常用技巧

从 2013 年的 5 月 16 日 Google 在 I/O 大会上发布的 Android Studio 第一个预览版本开始，到如今的 3.0 版本，Google 对它进行不断地增强与优化，并且不断的为其添加很多实用的功能。本章将对 Android Studio 常用功能与技巧进行详细介绍。

2.1 Android Studio 的基本了解

2.1.1 了解配置界面

Android Studio 开发工具提供了快速配置的功能，在不打开项目的情况下即可进行开发项目前的配置。在 Android Studio 欢迎界面中单击 Configure 选项，即可显示如图 2.1 所示的配置功能列表。

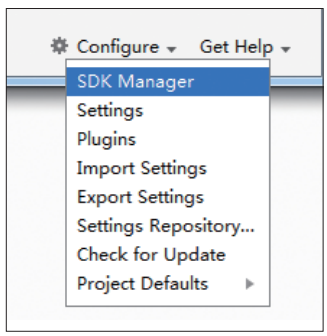


图 2.1　配置功能列表

配置功能列表的概览如下：

- SDK Manager：SDK 管理。
- Settings：开发工具的设置窗口。
- Plugins：插件。
- Import Settings：导入设置。
- Export Settings：导出设置。
- Settings Repository...：设置仓库。
- Check for Update：检查更新。

◆ Project Defaults：默认设置。

下面将对配置功能列表中的常用功能进行介绍。

1. Android SDK 的管理

在 Android Studio 开发工具当中需要使用 SDK Manager 进行 Android SDK 的安装、更新以及下载的操作。启动 SDK Manager 的方式有以下两种：

◆ 在环境界面中单击 Configure → SDK Manager 即可。

◆ 在 Android Studio 创建项目后的主界面的工具栏当中直接单击 SDK Manager 启动图标进行启动。如图 2.2 所示。

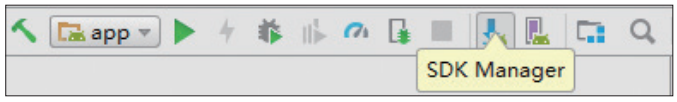


图 2.2　工具栏当中的 SDK Manager 启动图标

打开 SDK Manager 界面后将显示如图 2.3 所示对话框。

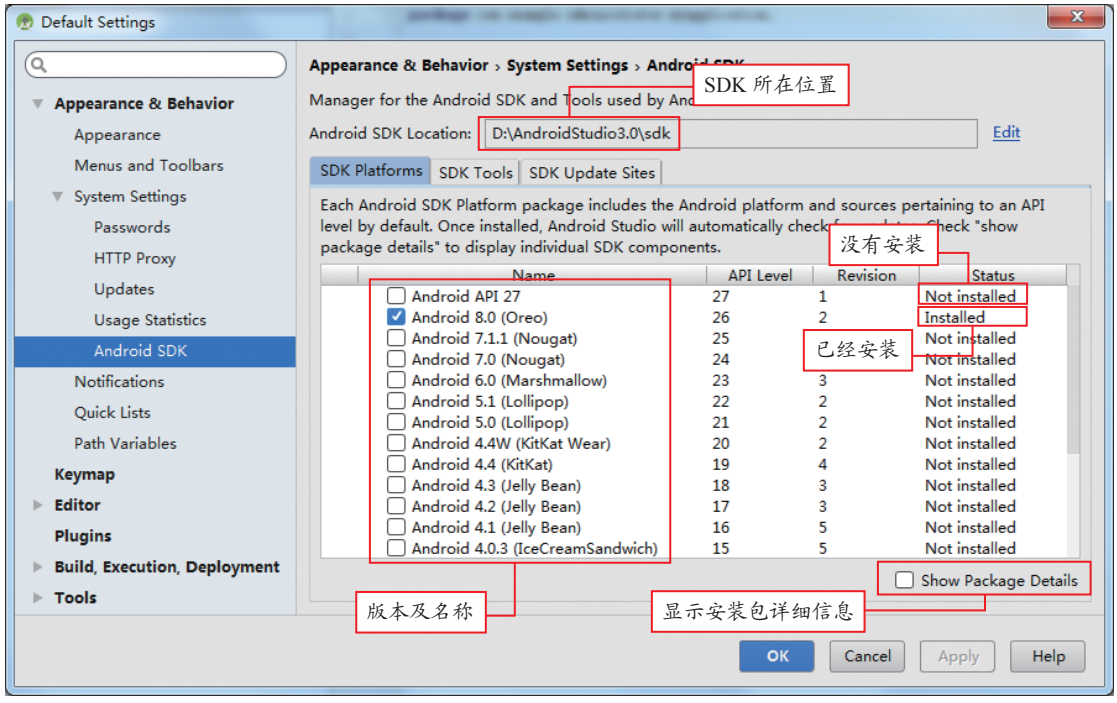


图 2.3　SDK Manager 界面

说明 在上图SDK Manager界面当中 API Level为API级别，Revision为版本修订次数。其中Status当中会出现Partially Installed状态，此类状态为部分安装，可能该版本有更新内容需要下载。

2. 插件

启动插件管理界面的方式有以下两种：

◆ 在欢迎界面中单击 Configure → Plugins 即可。

◆ 在 Android Studio 创建项目后的主界面中单击 File → Settings → Plugins 即可。打开插件管理界面将显示如图 2.4 所示对话框。

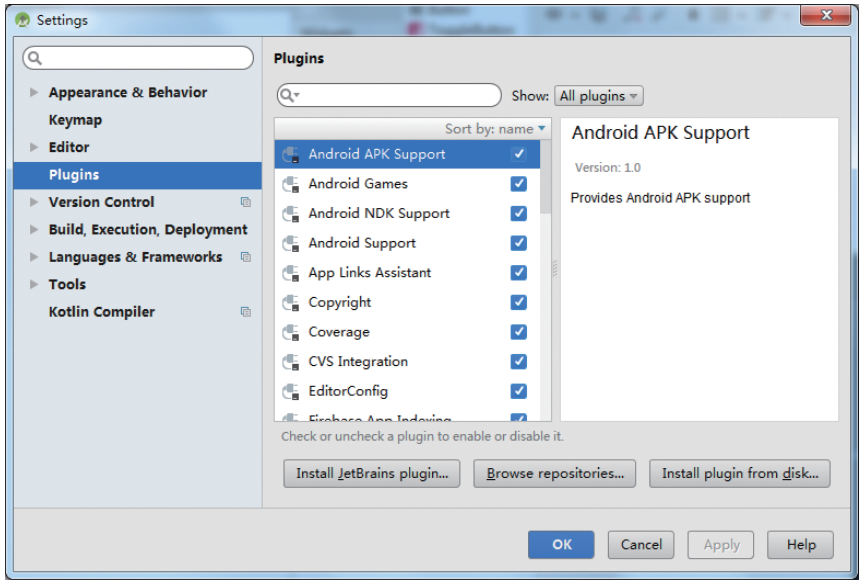


图 2.4　插件管理界面

AndroidStudio 有着非常强大的插件支持功能，更多的插件管理内容将在本章的 2.3 小节中详细介绍。

3.　导出设置

由于 AndroidStudio 可以将导出设置文件以 jar 包的形式进行共享，因此我们可以将设置文件导出。打开导出 AndroidStudio 的设置文件界面，有以下两种方式：

（1）在欢迎界面中单击 Configure → Export Settings 即可。

（2）在 Android Studio 创建项目后的主界面中单击 File → Export Settings 即可。打开导出设置界面将显示如图 2.5 所示对话框。

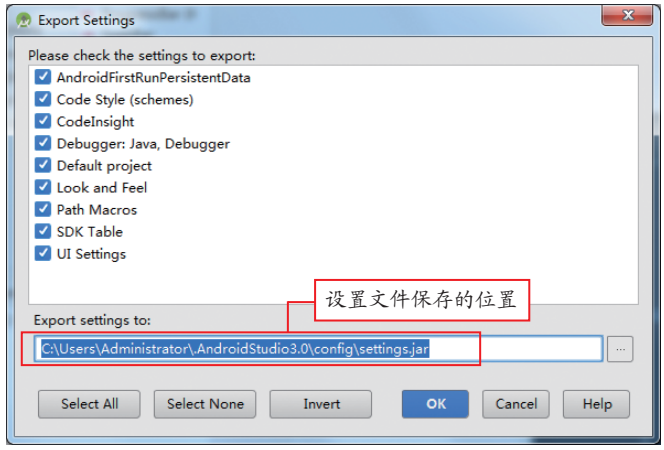


图 2.5　导出设置界面

说明　在图2.5导出设置界面中选择需要导出的设置名称，默认为全选状态。

4.　导入设置

导入设置的操作步骤及打开方式与导出设置类似，只是需要选择 Import Settings → 选择设置文

件 .jar，或者在 File → Import Settings → 选择设置文件 .jar。打开导入设置界面后选择需要导入的设置选项单击 OK 即可。

2.1.2 外观设置

1. 主题

主题可以让开发者在适合自己的风格下进行开发，AndroidStudio 提供了三种不同的主题风格。除了默认显示的风格以外还有如图 2.6 所示的 Darcula 主题以及如图 2.7 所示的 Windows 主题。

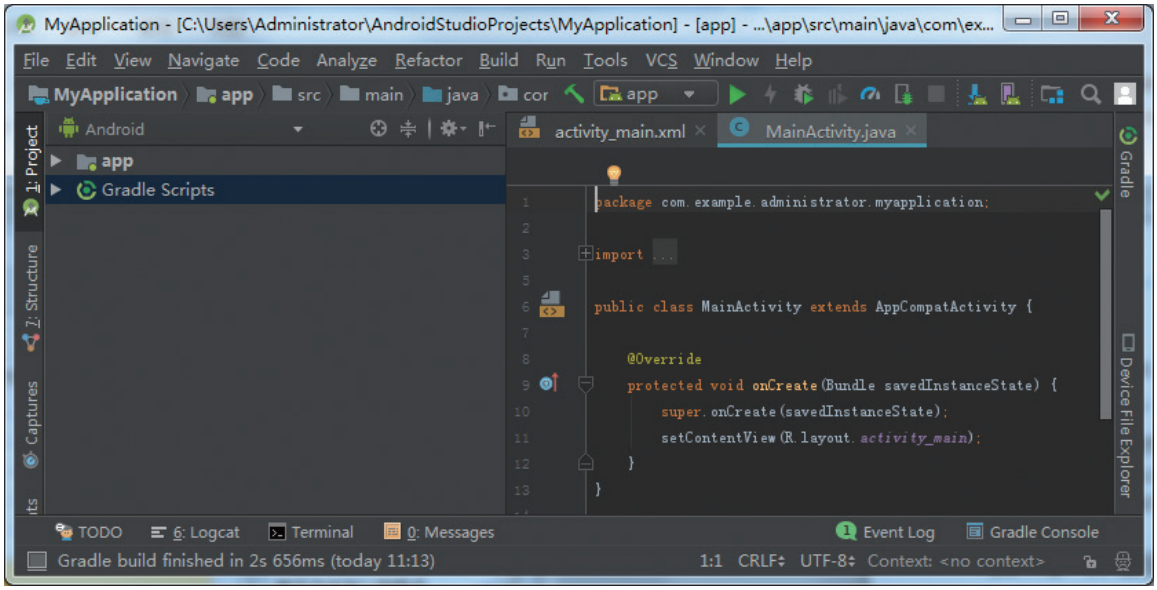


图 2.6 Darcula 主题

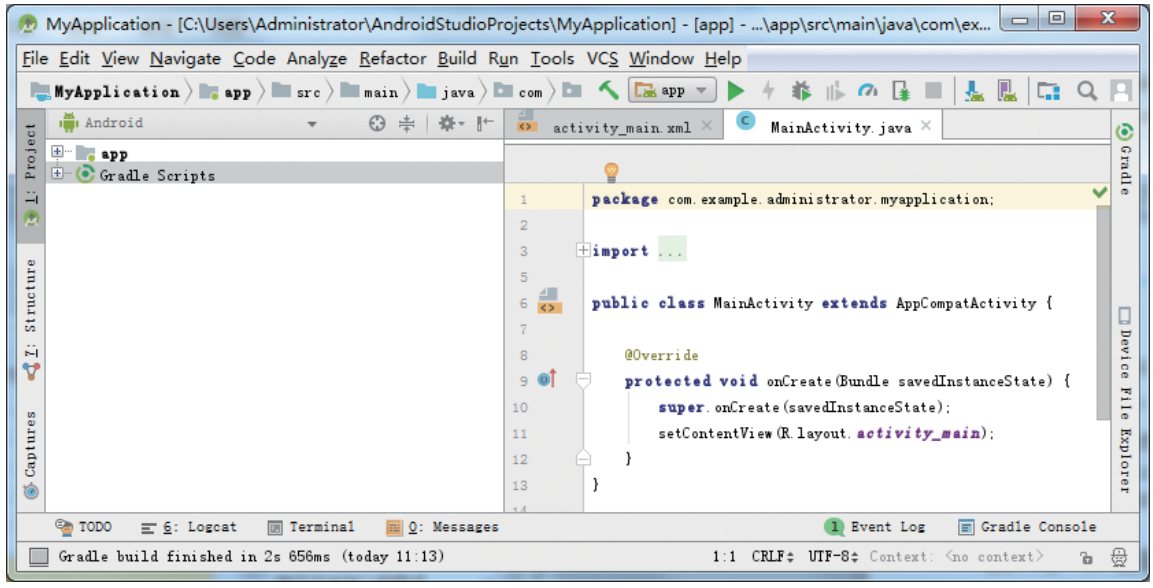


图 2.7 Windows 主题

AndroidStudio 设置主题的方法如下：

单击菜单栏 File → Settings 选项将显示如图 2.8 所示的设置界面，然后在该界面中单击 Appearance&Behavior → Appearance → Theme 进行主题的更换即可。

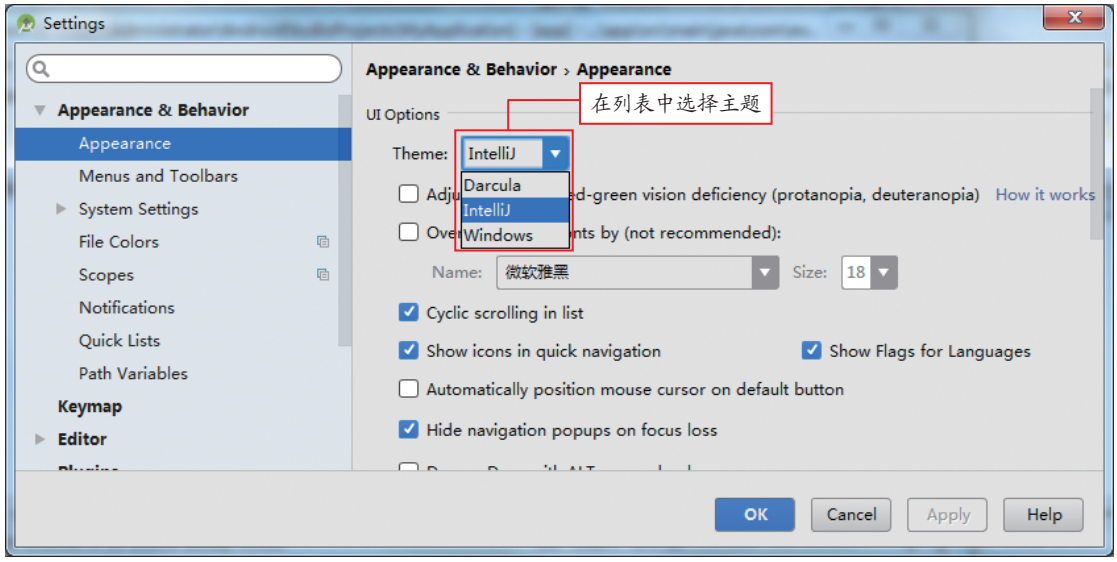


图 2.8　设置主题界面

2.　设置字体大小

在开发中有时工程师会觉得开发工具中文字很小或者是字体看着不舒服，AndroidStudio 同样可以按照开发者所需要的字体与字号进行设置。设置方法如下：

单击菜单栏中 File → Settings 选项将显示如图 2.9 所示的设置界面，然后在该界面中单击 Appearance&Behavior → Appearance →勾选 Override default fonts by (not recommended) 然后选择需要设置的字体与字号即可。

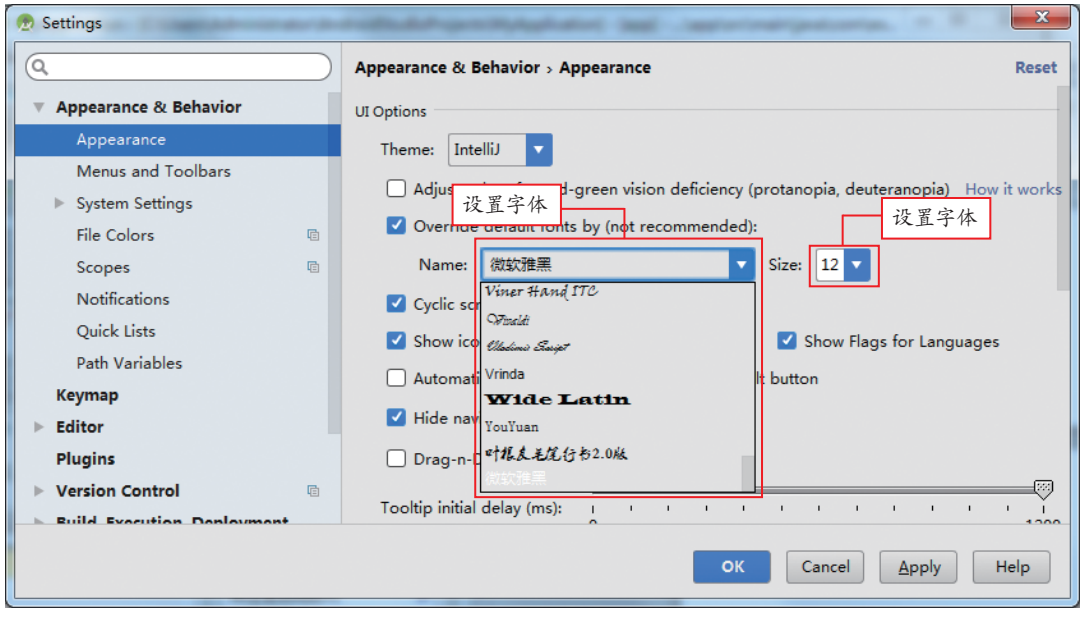


图 2.9　设置字体、字号

3. 设置代码字体的大小

字体与字号设置完成以后，AndroidStudio 整体风格都会根据个人而变化。只是代码的大小并没有变化。设置代码的大小方式如图 2.10 所示。需要在 Settings → Editor → Colors&Fonts → Font → Size 当中进行代码字号大小的设置。

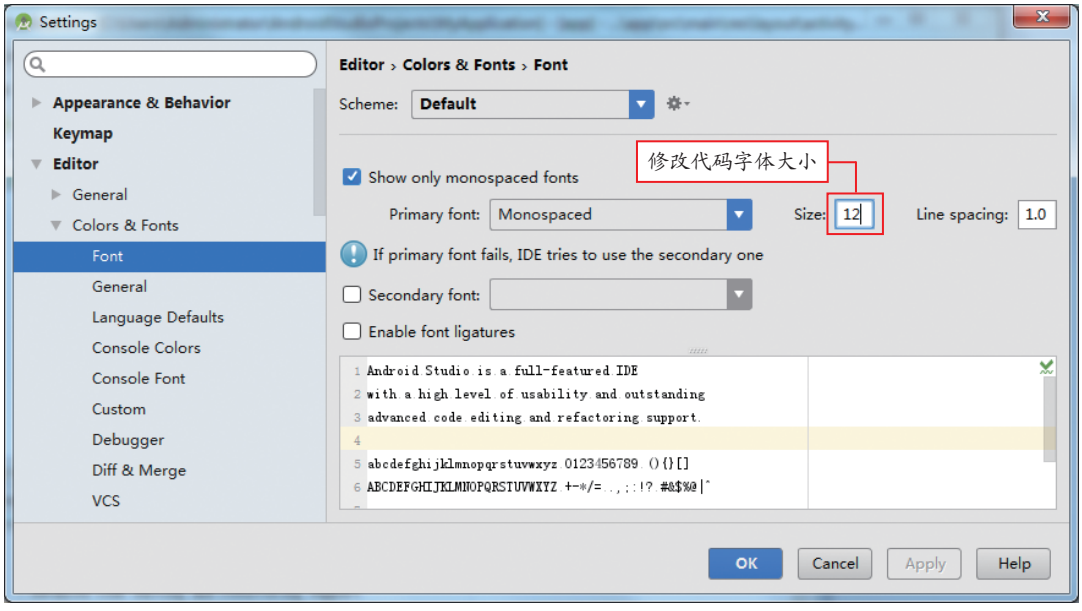


图 2.10 设置代码字号大小

说明 在Settings → Editor → Colors&Fonts设置选项中不仅可以设置代码字体的大小，还可以设置其他语言代码字号的大小以及颜色。例如，C/C++、Java、HTML、XML等。

2.1.3 AndroidStudio 的主窗口

AndroidStudio 的主窗口是由菜单栏、导航栏、工具栏、编辑器窗口、工具窗口栏、工具窗口以及状态栏所组成。如图 2.11 所示。

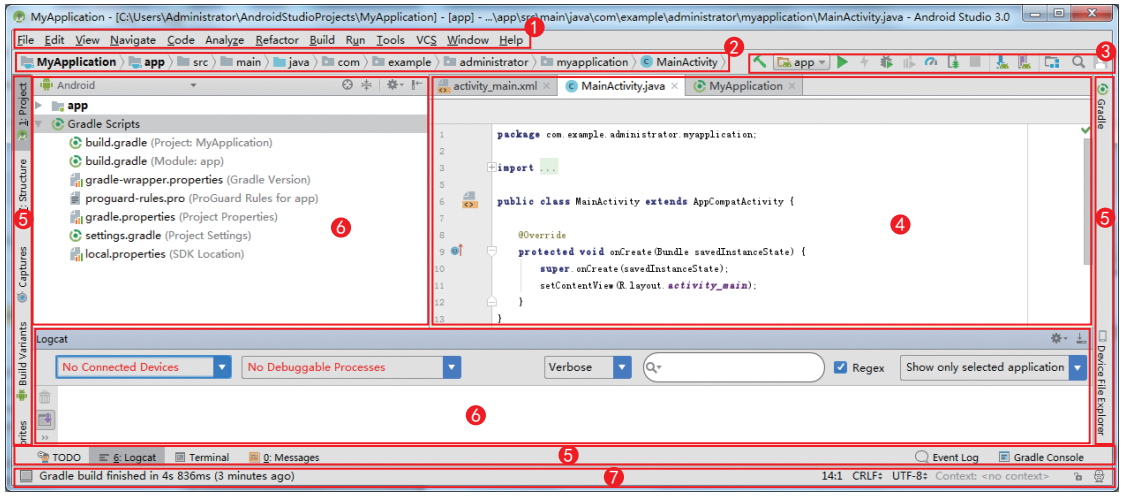


图 2.11 AndroidStudio 的主窗口

❶ 菜单栏

菜单栏一共分为 13 项，分别是文件、编辑、视图、导航、代码、分析、重构、构建、运行、工具、版本控制、窗口、帮助等菜单项，如图 2.12 所示。在每一个菜单项中还包含很多实用的子项。

图 2.12　菜单栏

❷ 导航栏

导航栏可帮助您在项目中导航，以及查看打开的项目和文件。此区域提供 Project 窗口所示结构的精简视图。当前打开的项目文件位置如图 2.13 所示。

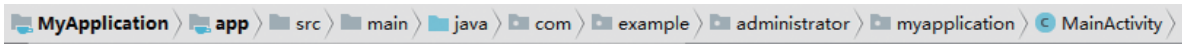


图 2.13　导航栏

说明　在菜单栏中单击View → 勾选Navigation Bar即可显示导航栏，取消勾选则关闭导航栏。

❸ 工具栏

工具栏提供了执行各种操作的工具，包括运行应用和启动 Android 工具。它是从菜单栏中提取出比较常用的一些功能，这样可以方便快速操作。如图 2.14 所示。

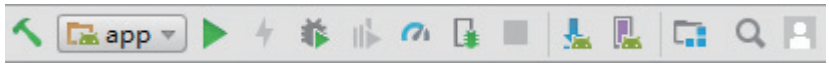


图 2.14　工具栏

❹ 编辑器窗口

编辑器窗口是创建和修改代码的区域如图 2.15 所示。 编辑器可能因当前文件类型的不同而有所差异。例如，在查看布局文件时，编辑器显示布局编辑器。如图 2.16 所示。

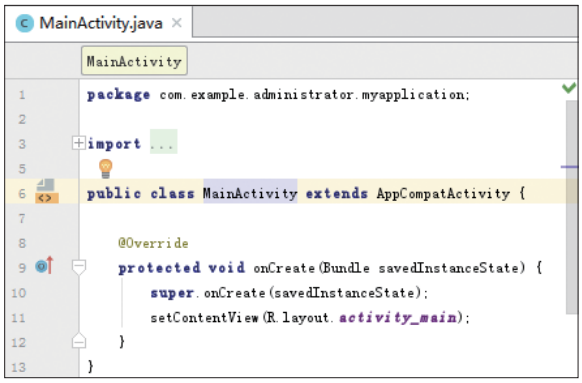


图 2.15　Java 代码编辑器

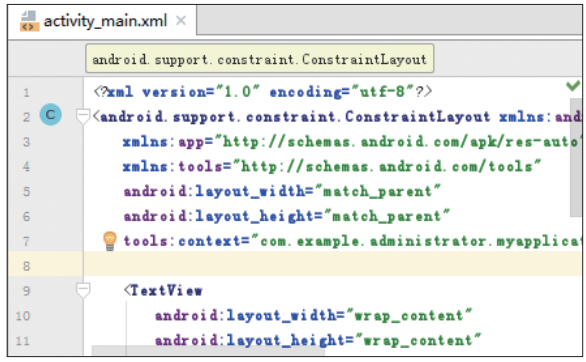


图 2.16　布局代码编辑器

❺ 工具窗口栏

工具窗口栏在 IDE 窗口外部运行，并且包含可用于展开或折叠各个工具窗口的按钮。

❻ 工具窗口

工具窗口提供对特定任务的访问，例如项目管理、运行信息、信息日志等。您可以展开和折叠这些窗口。

❼ 状态栏

状态栏在主窗口的最底部，主要用于显示项目和 IDE 本身的状态以及任何警告或消息。如图 2.17 所示。

图 2.17　状态栏

状态栏信息如下：

❶ 工具窗口管理。
❷ 当前执行过的任务信息。
❸ 光标当前所在的位置。
❹ 切换行分隔符。
❺ 切换文件编码。
❻ 文件只读开关。
❼ 语法高亮级别切换。
❽ 事件日志。

2.1.4 常用的工具窗口

工具窗口被隐藏在工具窗口栏当中，通过一定的规则显示或隐藏，主要用于显示或操作某个工具。下面将对 AndroidStudio 常用的工具窗口进行介绍。

1. Run：运行工具窗口

运行工具窗口主要显示运行 App 应用时的过程，以及运行过程中的提示信息。如图 2.18 所示。

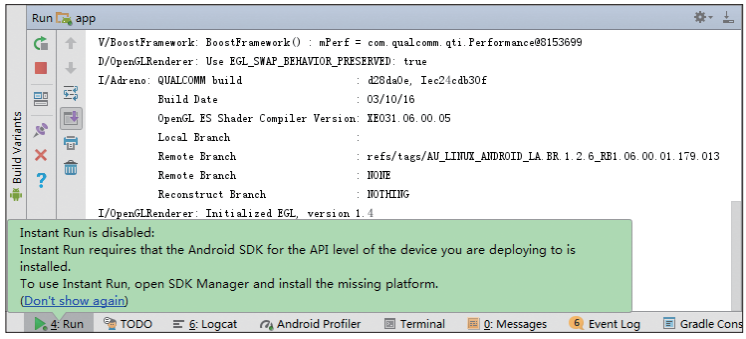


图 2.18　运行工具窗口

2. TODO：待处理任务工具窗口

在使用 AndroidStudio 进行开发的过程中，如果某段代码是未完成的功能或者是此段代码中可能出现 BUG 以及需要优化时，可以使用注释“//TODO 注释内容”来进行标识，如图 2.19 所示。此时通过待处理任务工具窗口进行整个项目待处理任务的查看。如图 2.20 所示。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //TODO 待处理任务测试
}
```

图 2.19　待处理注释内容

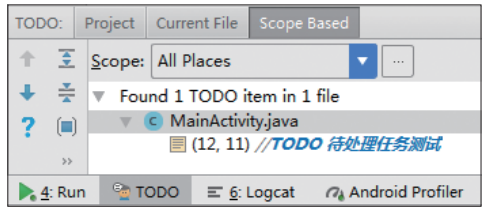


图 2.20　待处理位置

3. Logcat：日志信息工具窗口

在日志信息工具窗口中可以进行日志级别的筛选、关键字的筛选以及过滤配置。如图 2.21 所示。

图 2.21　日志信息工具窗口

4. Android Profiler：Android 监控窗口

AndroidStudio 在 3.0 版本中将内存、CPU、网络的监控窗口集中在 Android Profiler 工具窗口栏当中。如图 2.22 所示。

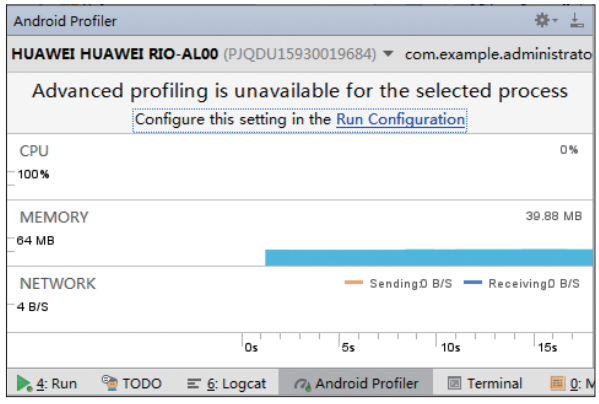


图 2.22　Android 监控窗口

5. Terminal：终端工具窗口

AndroidStudio 提供了终端工具窗口，在该窗口中直接输入终端命令即可，例如，运行 javac 命令符将显示如图 2.23 所示内容。

图 2.23　终端工具窗口

6. Gradle：Gradle 工具窗口

在 Gradle 工具窗口中列出了当前项目和 Module 中所支持的所有 Gradle 任务与运行配置，可以很方便地快速操作，如图 2.24 所示。

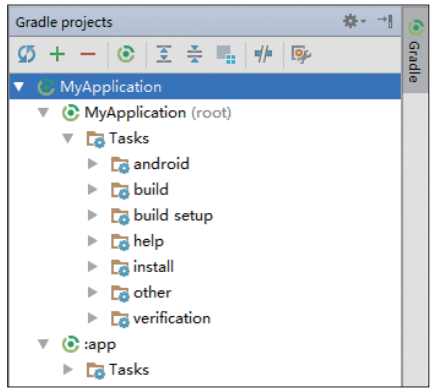


图 2.24 Gradle 工具窗口

2.2 导入项目与模块

2.2.1 导入 AndroidStudio 项目

导入 AndroidStudio 项目的操作方式有以下两种：

◆ 在欢迎界面中单击 Import Project 将显示如图 2.25 所示的选择项目对话框，选择需要导入的项目文件即可。

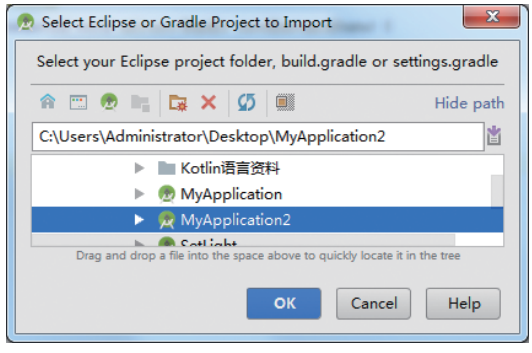


图 2.25 选择项目对话框

◆ 在 Androidstudio 的主窗口当中单击菜单栏中的 File → New → Import Project → 选项选择需要导入的项目即可，AndroidStudio 所开发的项目都会显示 AndroidStudio 图标。

说明 如果导入的项目sdk版本和gradle版本符合当前的AndroidStudio开发工具，即可很轻松地成功导入。

例如，高版本的开发工具导入低版本或者是其他版本的 AndroidStudio 项目时将显示如图 2.26 所示的同步 Android SDK 的提示对话框，该提示对话框表示导入项目的 SDK 位置与开发工具 SDK 位置不同，此处单击 ok 即可。然后将显示如图 2.27 所示的构建项目信息工具的进度提示框。该进度提示将会下载导入项目中所对应的 Gradle 构建工具，下载时间完全取决于网络速度，也可以选择离线下载对应版本的 Gradle 构建工具。

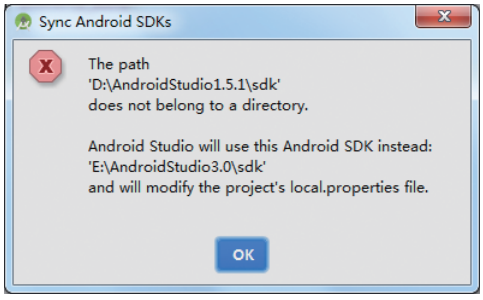


图 2.26　同步 Android SDK 的提示框

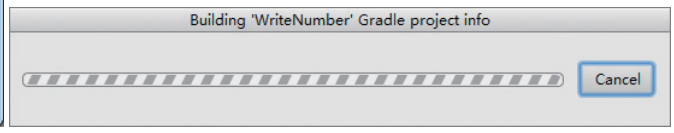


图 2.27　构建项目信息工具的进度提示框

Gradle 构建工具下载并加载完成后将显示如图 2.28 所示的找不到对应的 SDK 版本的提示，此处单击“Install missing platform(s) and sync project”超链接即可下载并安装导入项目中对应的 SDK 版本。安装完成后在提示对话框中单击 Finish 按钮即可。

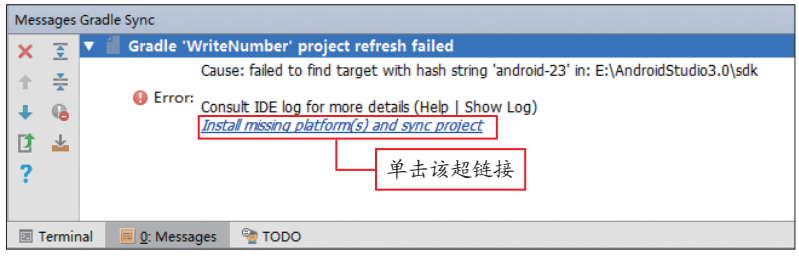


图 2.28　找不到对应的 SDK 版本

导入项目中对应的 SDK 版本安装完成后，将显示如图 2.29 所示的找不到对应的 Android 构建工具“Build Tools”版本，此处单击 Install Build Tools XXXX（对应版本） and sync project 超链接即可下载并安装导入项目中对应的 Android 构建工具版本。

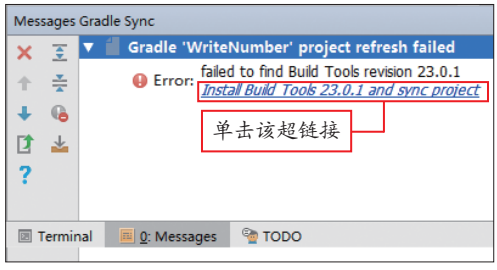


图 2.29　找不到对应的 Build Tools 版本

安装导入项目中对应的 Android 构建工具版本安装完成后，将显示如图 2.30 所示的同步导入项目中的“com.android.support:appcompat-v7:XXX（对应版本）”依赖包。此处单击“Add Google Maven repository and sync project”超链接即可同步。

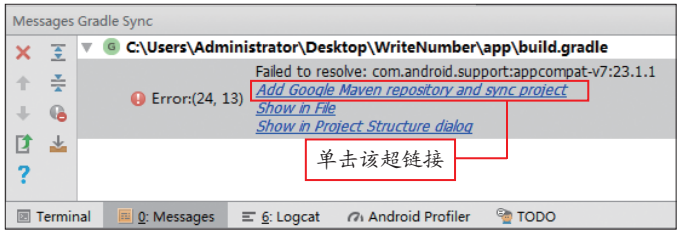


图 2.30　同步依赖包

2.2.2 导入 Eclipse 项目

在使用 AndroidStudio 导入 Eclipse 项目之前，需要为 Eclipse 项目添加 Gradle 构建工具。首先将 Eclipse 项目导出，在弹出的导出对话框中选择 Android → Generate Gradle build files，如图 2.31 所示，然后将项目导出即可。

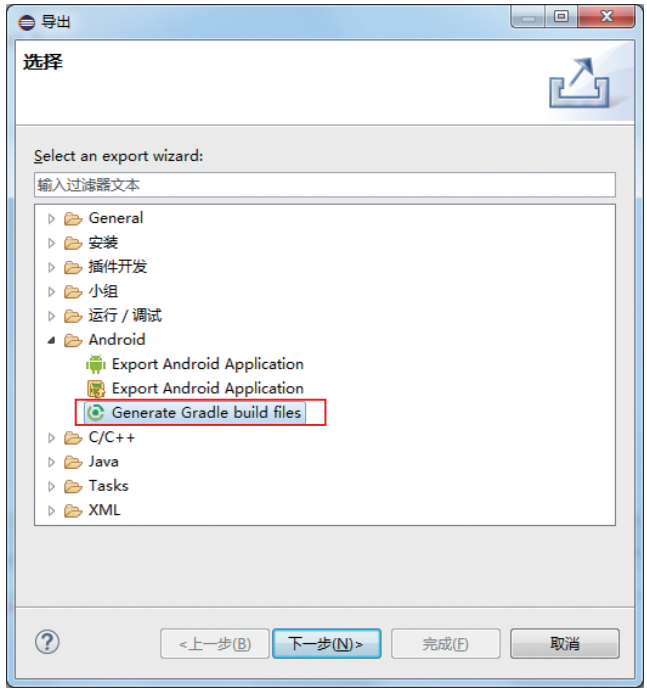


图 2.31 为 Eclipse 项目添加 Gradle 构建工具

导入 Eclipse 项目时可以通过从欢迎界面中单击“Import Project”或者在 Android Studio 的主窗口中单击菜单栏中的 File → New → Import Project 选项，然后在选择项目的对话框中找到需要导入的 Eclipse 项目即可。

项目导入后首先会自动弹出构建项目信息工具的对话框，加载完成后将显示找不到对应 SDK 版本的提示信息，单击超链接下载并安装完成后同样会提示找不到对应的 Android 构建工具“Build Tools”版本，此处参考 2.2.1 知识回顾即可。

以上两个问题解决完成后将显示如图 2.32 所示的 Gradle 同步错误的提示信息，修改该错误信息需要 3 个步骤来解决。

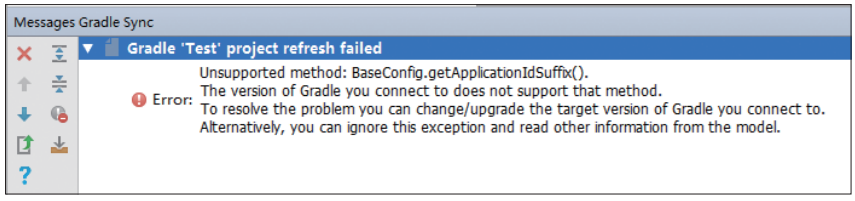


图 2.32 Gradle 同步错误信息

（1）在 AndroidStudio 中创建或打开一个新的项目，例如，打开一个已经创建好的名称为“MyApplication”的 AndroidStudio 项目，然后在该项目中打开 Gradle Scripts 目录下的 build.gradle（Project：MyApplication）文件，在该文件中找到依赖的构建工具包版本号，如图 2.33 所示。

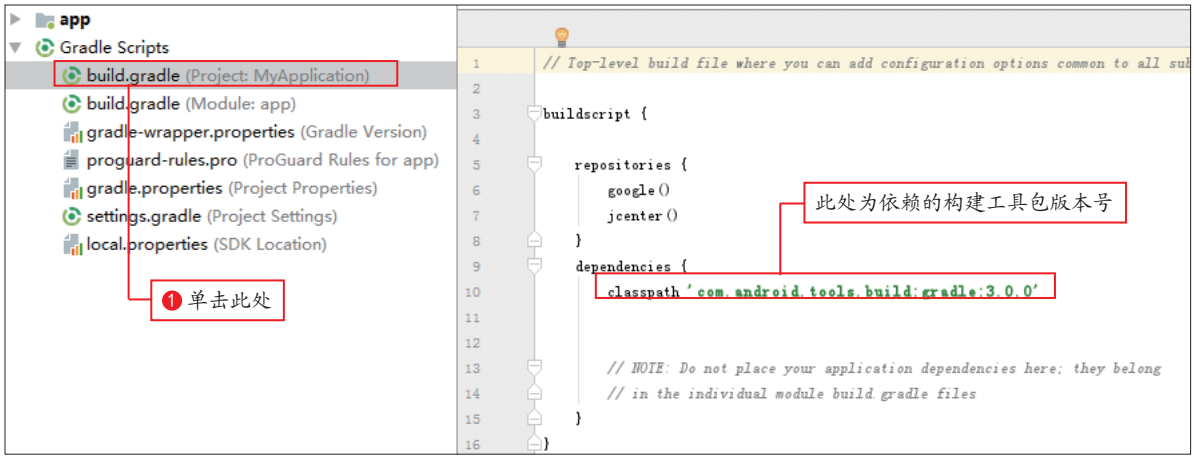


图 2.33　找到依赖的构建工具包版本号

（2）打开 AndroidStudio 导入的 Eclipse 项目，然后同样打开该项目中 Gradle Scripts 目录下的 build.gradle（Project：Test）文件，在该文件中找到依赖的构建工具包版本号，并将该版本号修改为步骤（1）中 AndroidStudio 项目的构建工具包版本号，如图 2.34 所示。最后单击右上角的“Try Again”（再次同步）。

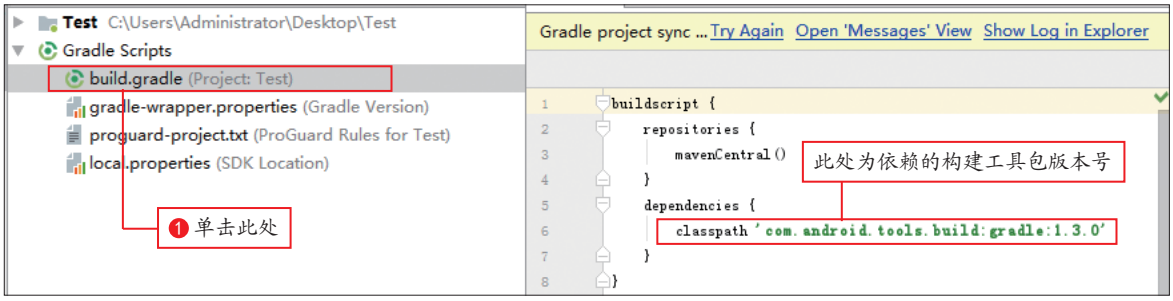


图 2.34　修改 Eclipse 项目中构建工具包版本号

（3）再次同步后将显示如图 2.35 所示的错误信息，提示 Gradle 版本不同。此处单击超链接即可显示如图 2.36 所示的 Android SDK 构建工具版本不同的错误信息。此处同样单击超链接升级 Eclipse 项目中的 Android SDK 构建工具版本即可。

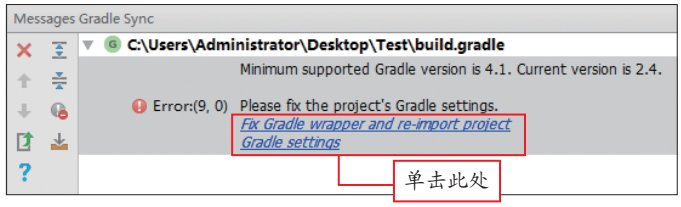


图 2.35　提示 Gradle 版本不同

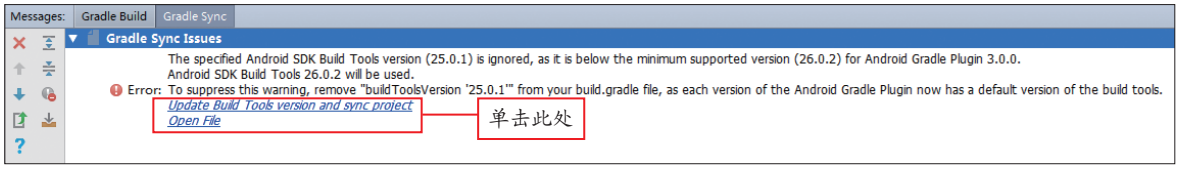


图 2.36　Android SDK 构建工具版本不同的错误信息

2.2.3 导入模块

导入模块需要单击菜单栏中的 File → New → Import Module 选项，在弹出的选择模块对话框中选择指定的模块即可。在导入模块的过程中会出现一些错误提示，下面将介绍导入模块时常见的错误提示。

1. 项目中已经包含有此名称的模块，解决方法可以将模块名称进行修改即可导入。如图 2.37 所示。

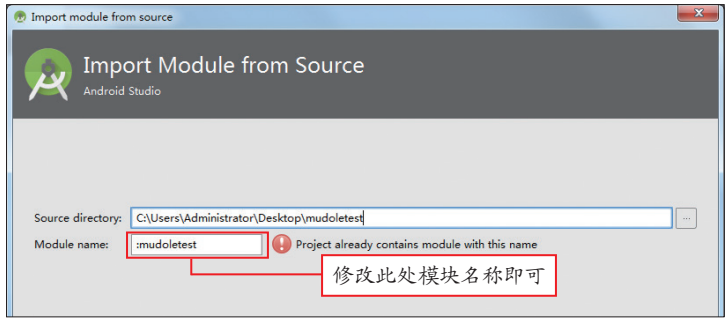


图 2.37 项目中已经包含有此名称的模块

2. 导入路径不存在，解决方法可以重新选择指定的模块即可。如图 2.38 所示。

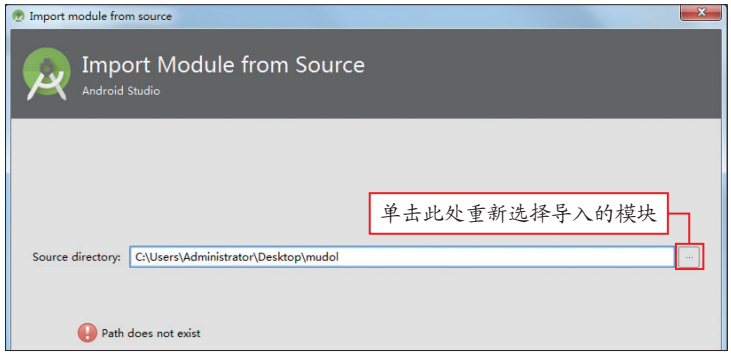


图 2.38 导入路径不存在

说明 Module成功导入后将显示如图2.39所示内容。

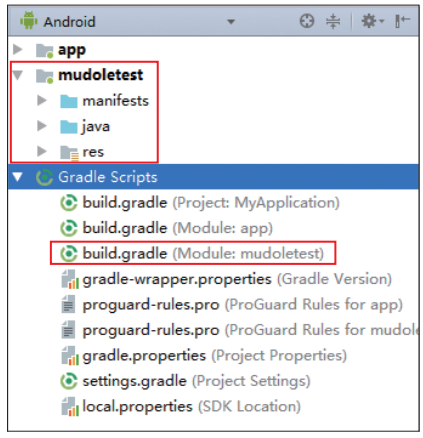


图 2.39 Module 成功导入

2.2.4 创建与导入 .AAR 包

.AAR 包与 JAVA 中的 .JAR 包相似，不同的是 .JAR 包中只包含了 class 文件与清单文件，不包含资源文件，如图片等所有 res 中的文件。而 .AAR 包是一个 Android 库项目的二进制归档文件，其中包含 AndroidManifest.xml、class 文件、res 资源文件等。

1. 创建 .AAR 包

在 AndroidStudio 主窗口的菜单栏中，单击 File → New → NewModule，选项将显示创建 Module 对话框，在该对话框中选择 Android Library 选项，如图 2.40 所示，然后单击 Next 按钮，修改库名称或者包的名称，这里使用默认，单击 Finish 按钮，如图 2.41 所示。

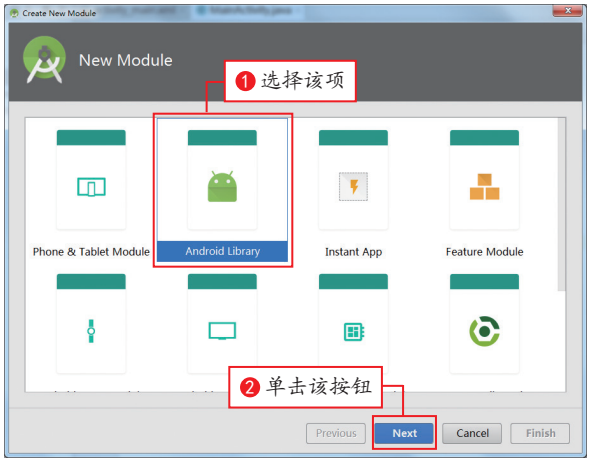


图 2.40 创建 Module 对话框

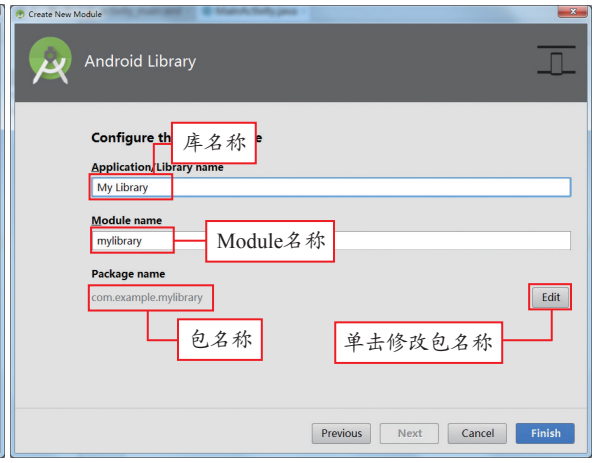


图 2.41 Android Library 对话框

Android Library 创建完成后将以 Module 的形式显示在 AndroidStudio 的项目结构工具窗口中，如图 2.42 所示，在该 Module 中可以创建需要的 class 文件及 res 资源文件等。

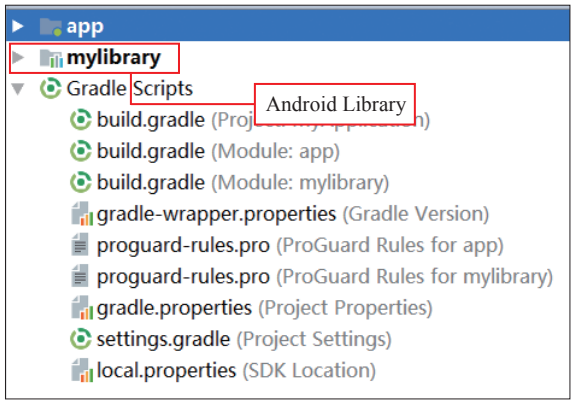


图 2.42 完成创建 Android Library

最后单击菜单栏中的 Build → Rebuild Project 选项，将自动生成该 Module 的 .ARR 包，该包生成后将项目结构类型切换为 Project，然后在 Module 名称 /build/aar 目录下找到该 Module 的 .AAR 包，如图 2.43 所示。

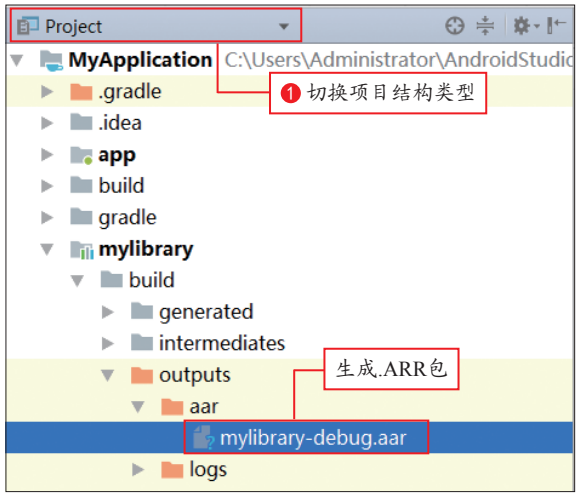


图 2.43　找到该 Module 的 .AAR 包

说明 将Module的.AAR包复制并粘贴在指定路径下，在AndroidStudio中导入该包即可使用。

2.　导入 .AAR 包

在 Android Studio 主窗口的菜单栏中，单击 File → New → NewModule 选项，将显示创建的 Module 对话框，在该对话框中选择 Import .JAR/.ARR Package 选项，然后单击 Next 按钮，如图 2.44 所示。进入“Import Module from Library”对话框中，首先选择 aar 包的存放路径，然后为子项目命名，最后单击 Finish 按钮，如图 2.45 所示。

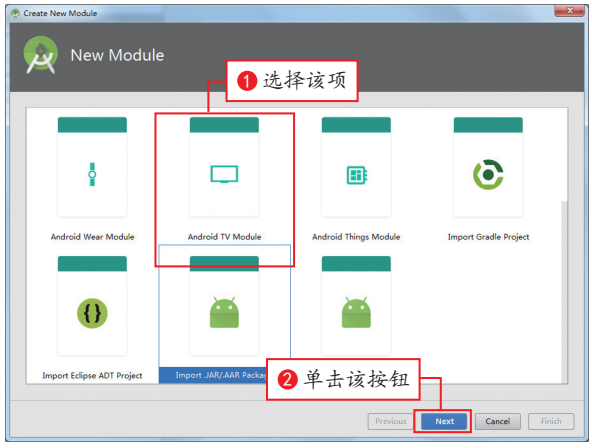


图 2.44　创建 Module 对话框

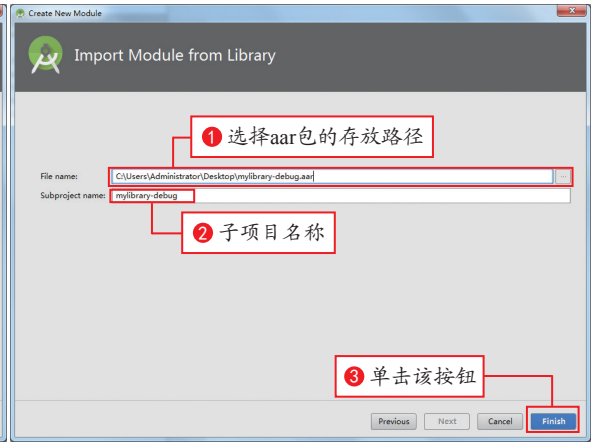


图 2.45　Import Module from Library 对话框

完成后可以看到作为 Module 导入的 .AAR 包，如图 2.46 所示。同时打开 Gradle Scripts 目录下的 settings.gradle（Project Settings）文件，该文件中将自动添加 mylibrary-debug 模块，如图 2.47 所示。

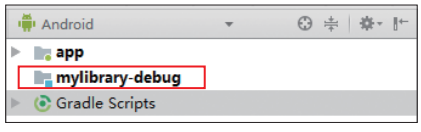


图 2.46　作为 Module 导入的 .AAR 包

```
include ':app', ':mylibrary-debug'
```

图 2.47　项目自动添加 mylibrary-debug 模块

将新导入的 mylibrary-debug 模块添加为应用程序 app 模块的依赖，单击菜单栏中的 File → Project Structure 选项，然后在 Project Structure 对话框中将 mylibrary-debug 模块添加为应用程序 app 模块的依赖，具体步骤如图 2.48 所示。

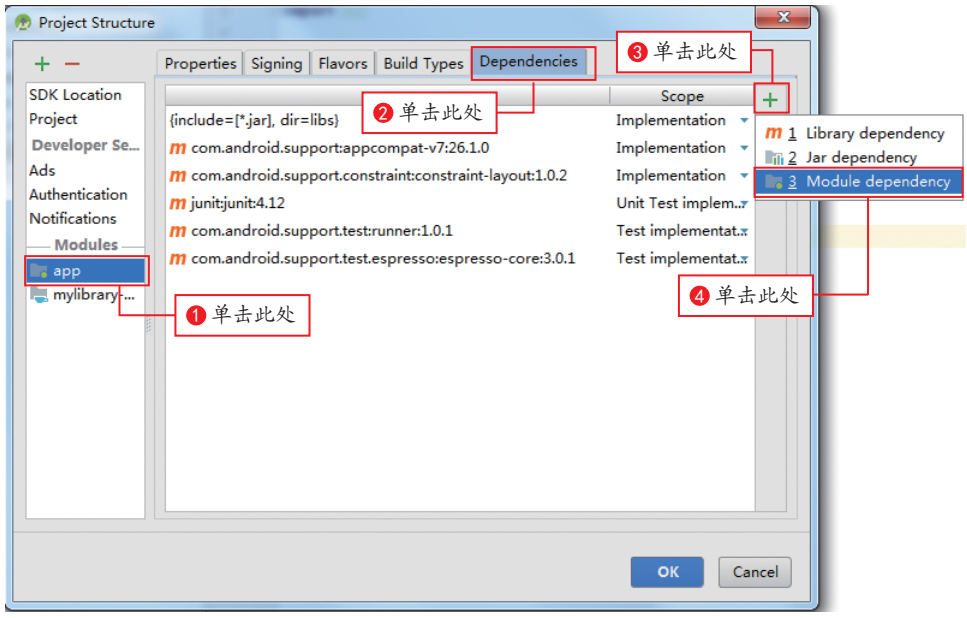


图 2.48　Project Structure 对话框

单击 Module dependency 选项后，将弹出 Choose Modules 对话框并确认需要依赖的 Module。单击 ok 返回 Project Structure 对话框，继续单击 ok 确认即可。如图 2.49 所示。

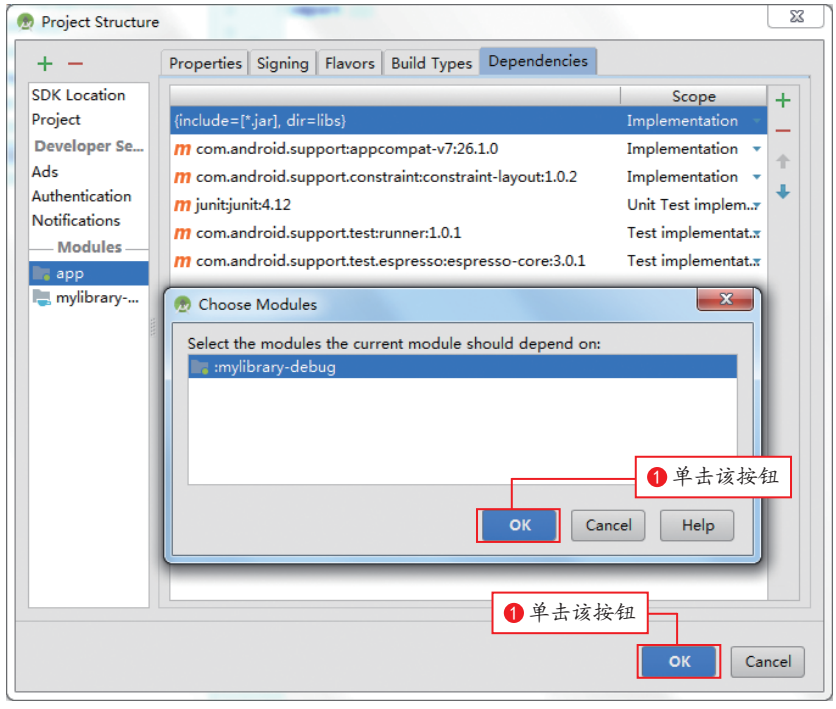


图 2.49　确认需要依赖的 Module

依赖完成后打开 Gradle Scripts 目录下的 build.gradle（Module app）文件，该文件中将自动添加依赖代码。如图 2.50 所示。

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
    implementation project(':mylibrary-debug')
}
```

图 2.50　自动添加的依赖代码

说明　为了测试导入的.AAR包是否可以使用，在创建.AAR包时可以在包中封装测试类与方法，导入与依赖成功后创建类的对象并调用其中的方法进行测试。

2.3 自定义图标

在 AndroidStudio 开发工具中可以快速地将现有的图片、文本或者剪贴画生成不同分辨率的应用图标。在项目结构工具窗口中右击 Module 名称，例如单击 app → New → Image Asset 选项，将弹出 Configure Image Asset 对话框，如图 2.51 所示。

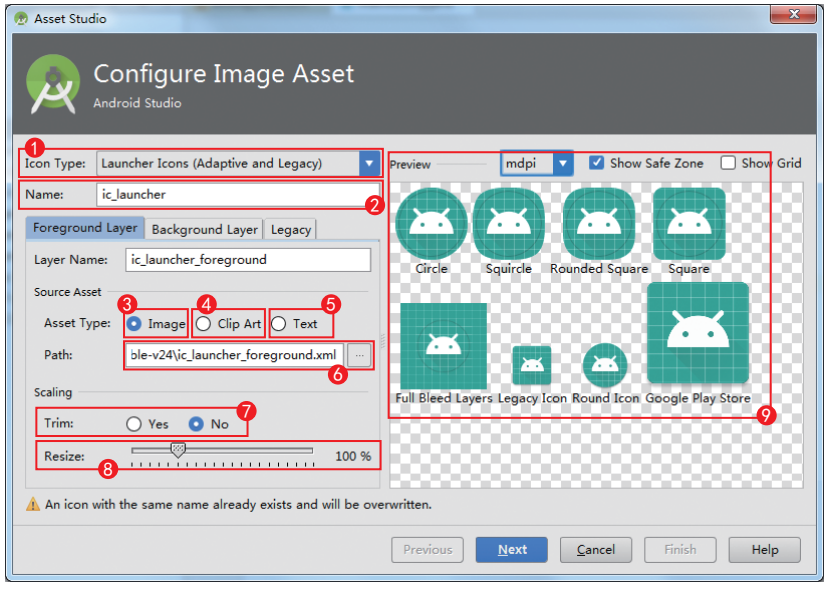


图 2.51　Configure Image Asset 对话框

❶ 图标类型　❷ 图标名称　❸ 图标　❹ 剪贴画　❺ 文本
❻ 选择图片的路径　❼ 修剪　❽ 调整大小　❾ 图标预览区

1. 启动图标

在图标类型(Icon Type)中选择Launcher Icons(Legacy only)，然后选择一个生成图标的本地图片，预览区将自动生成不同分辨率的图标，如图 2.52 所示。

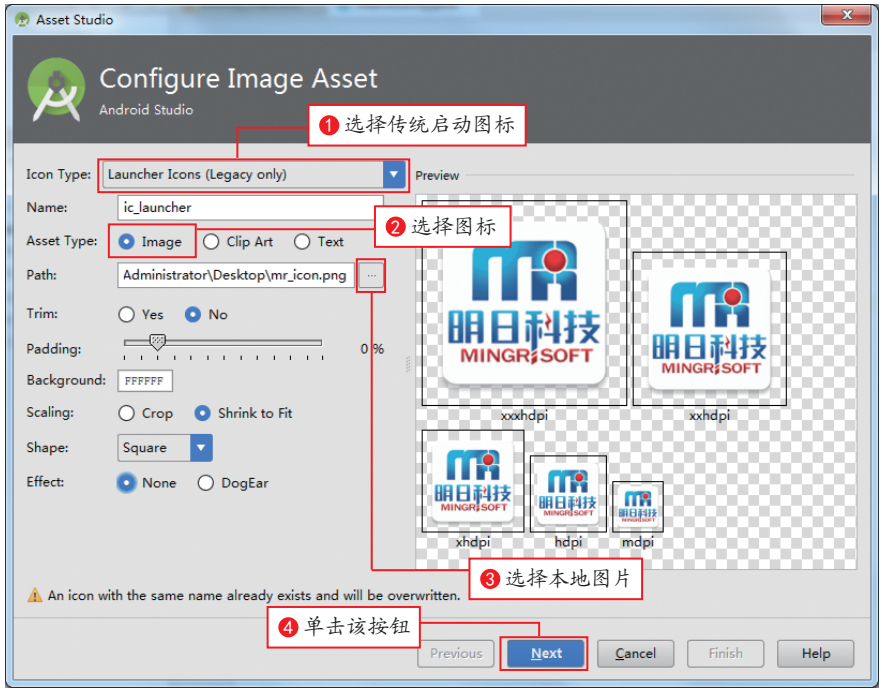


图 2.52　自定义启动图标

单击 Next 按钮后将显示确认图标路径的对话框，在该对话框中直接单击 Finish 按钮即可生成不同分辨率的自定义图标。如图 2.53 所示。

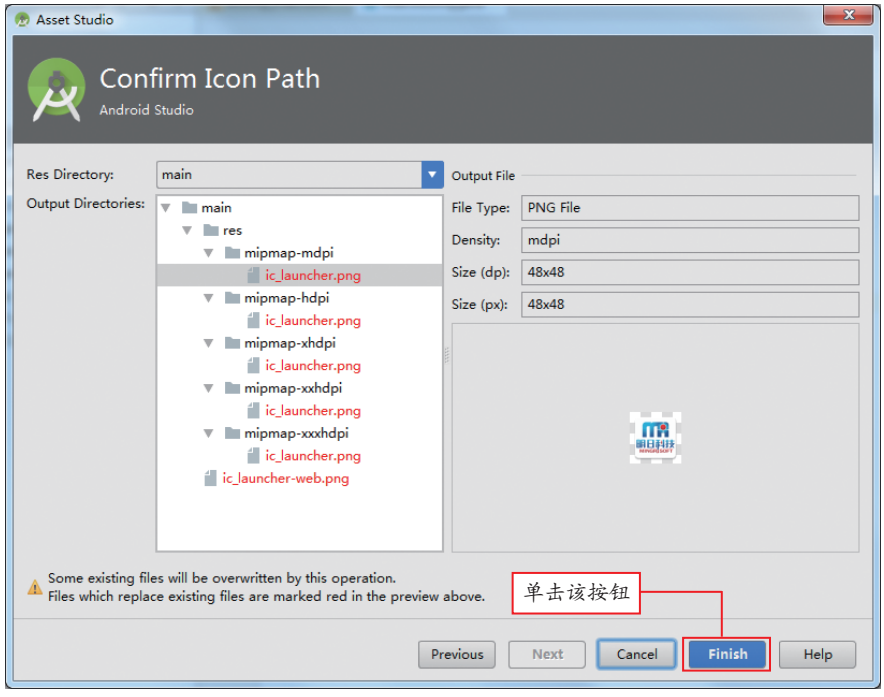


图 2.53　确认图标路径的对话框

生成后的自定义图标将保存在项目结构工具窗口中的 Module 名称 /res/mipmap 目录下，如图 2.54 所示。

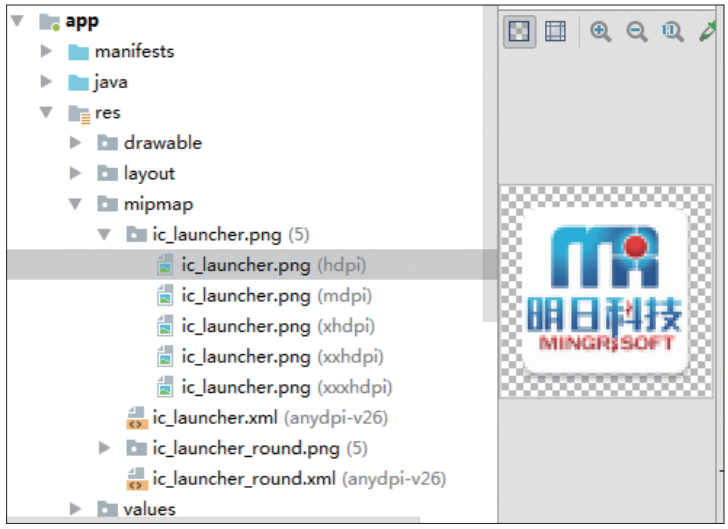


图 2.54　自定义图标的保存位置

2.　剪贴画图标

在 Asset Type 中选择 Clip Art 选项，然后选择需要使用的剪贴画，如图 2.55 所示。剪贴画选择完成后单击 Next 按钮，然后确认保存路径即可。

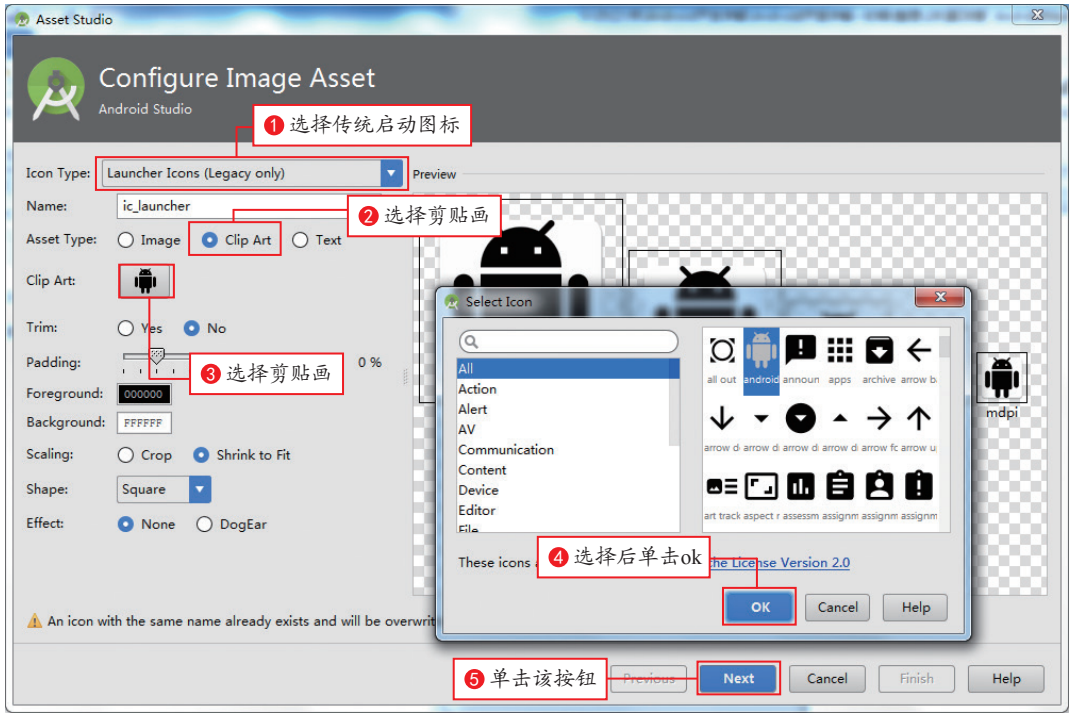


图 2.55　选择需要使用的剪贴画

3.　文本图标

在 Asset Type 中选择 Text 选项，然后设置文字与字体，如图 2.56 所示，然后确认保存路径即可。

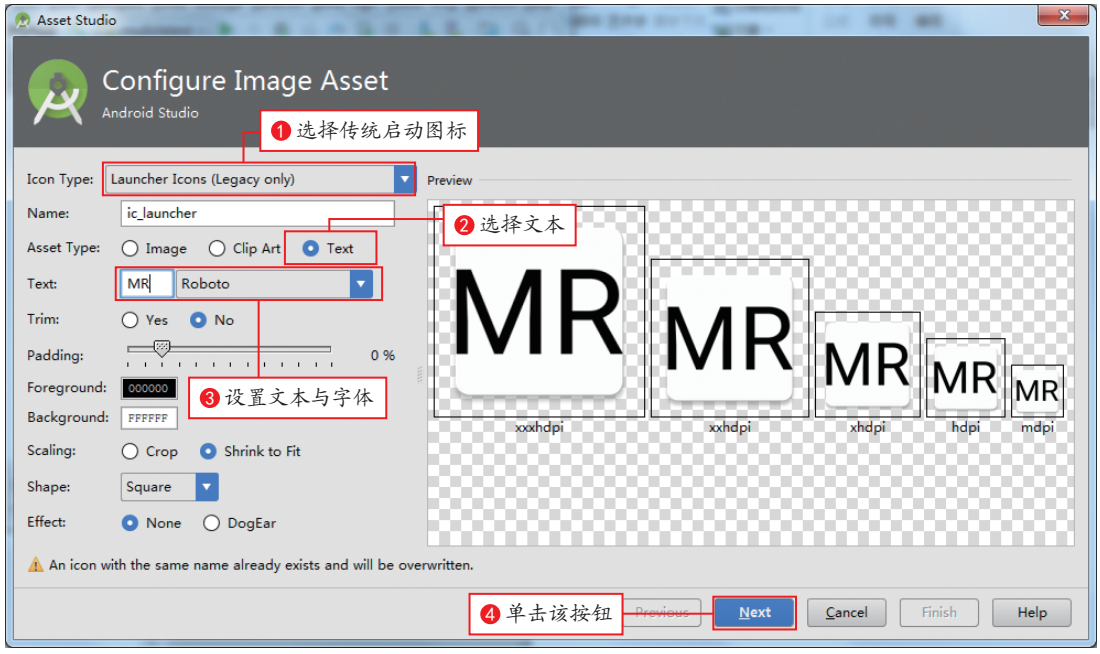


图 2.56　设置文本图标

说明　根据以上的操作方式，在图标类型（Icon Type）中选择其他的图标类型如活动栏与选项卡或者是通知即可创建对应类型的图标。

4.　创建矢量资源

Android 在 5.0（API Level21）中增加了一个 VectorDrawable 类，用于定义矢量资源，但是必须要求 5.0 及更高版本才可以使用。使用矢量资源可以自动适应不同尺寸和分辨率的屏幕，因此图片被缩放时也不会失真。

在项目结构工具窗口中右击 Module 名称 → New → Vector Asset 选项，将打开 Configure Vector Asset 对话框，在 Asset Type 中选择 Material Icon，然后单击 图标，选择内置的图标资源，如图 2.57 所示。

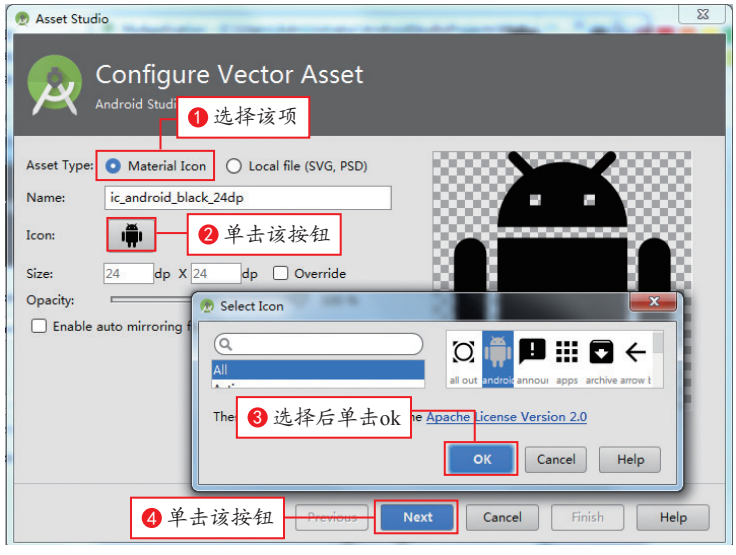


图 2.57　选择内置图标资源

单击 Next 按钮后将显示确认图标路径的对话框，如图 2.58 所示，单击 Finish 按钮矢量资源将保存在当前 Module 的 res/drawable 目录下，如图 2.59 所示。

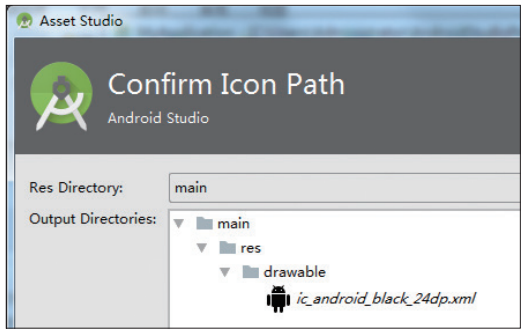


图 2.58　确认图标路径

图 2.59　打开矢量资源文件

说明　在Asset Type中不仅可以选择Material Icon，使用AndroidStudio内置的图标资源，还可以选择Local File(SVG,PSD)使用其他本地资源的SVG与PSD文件。

2.4 AndroidStudio 快捷键的使用

2.4.1 编辑类

在编写代码的过程中会经常使用到复制、粘贴、查找、替换等提升编写代码效率的一些功能。本节将介绍如何在 Android Studio 中通过快捷键以及菜单栏快速地使用这些操作技巧。

1. 复制类名及方法名的具体位置

将一个类名或者是方法名的整个路径复制下来，可以方便查找该类或方法的准确位置。具体方法有以下两种：

◆ 首先鼠标左键单击选中需要复制的类名或者是方法名，然后在菜单栏中单击 Edit → Copy Reference 选项即可复制，再将复制的路径粘贴在需要的位置即可。

◆ 选中需要复制的类名或者是方法名，使用快捷键 <Ctrl + Alt + Shift + C>（Windows/Linux）或者在苹果系统中使用快捷键 <Alt + command + Shift + C>（macOS）即可，如图 2.60 所示。

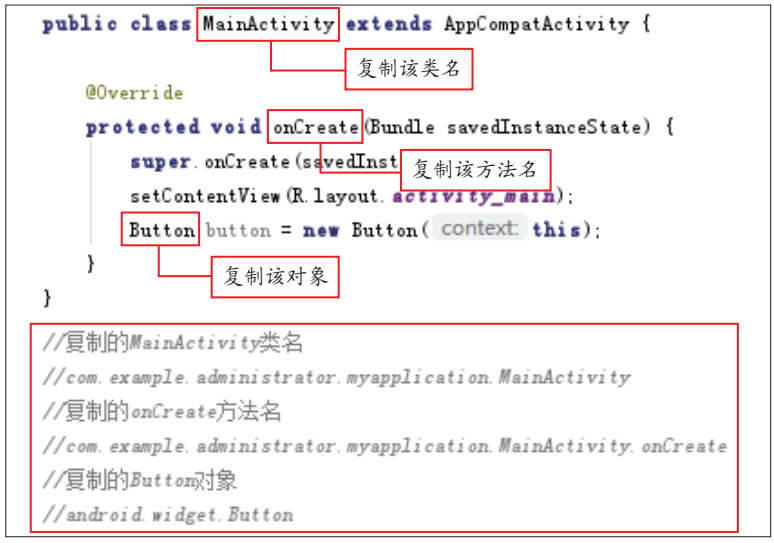


图 2.60　复制类名及方法名的具体位置

2.　合并两行代码

AndroidStudio 提供了一个自动合并两行代码的功能，可以智能地将注释、字符串等代码合并为一行。具体方法有以下两种：

◆ 选中需要合并的代码，然后在菜单栏中单击 Edit → Join Lines 选项即可合并两行代码。

◆ 选中需要合并的代码，然后使用快捷键 <Ctrl + Shift + J>（Windows/Linux）或者在苹果系统中使用快捷键 <control + Shift + j>（macOS）即可，如图 2.61 所示。

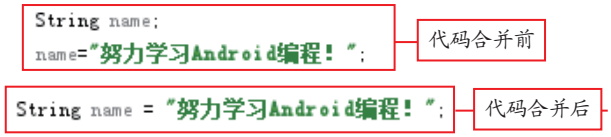


图 2.61　合并两行代码

3.　自动补全当前代码

自动补全当前代码可以完成编写代码的剩余部分，如自动添加代码中的大括号等。具体方法有以下两种：

◆ 在菜单栏中单击 Edit → Complete Current Statement 选项即可。

◆ 使用快捷键 <Ctrl + Shift + Enter>（Windows/Linux）或者在苹果系统中使用快捷键 <Shift + command + Enter >（macOS）即可。

示例：自动补全方法代码

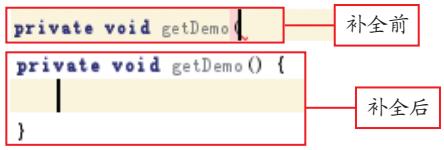
例如，自定义方法 getDemo()，光标需要在代码的后面，然后使用快捷键 <Ctrl + Shift + Enter>（Windows/Linux），如图 2.62 所示。

4.　切换字母大小写

可以通过以下的两种方式实现一键切换字母大小写的功能。

◆ 选中需要切换的字母，然后使用在菜单栏中单击 Edit → Toggle Case 选项即可。

◆ 选中需要切换的字母，然后使用快捷键 <Ctrl + Shift + U>（Windows/Linux）或者在苹果系统中使用快捷键 <Shift + command + U>（macOS）即可，如图 2.63 所示。



2.62　自动补全方法代码

图 2.63　切换字母大小写

5.　查找工具栏

AndroidStudio 提供了非常方便的查找工具，可以很轻松地在编辑区域查找自己需要的内容，下面通过以下示例来实现查找工具的使用方法。

示例 1：打开查找工具

使用快捷键 <Ctrl + F>（Windows/Linux）或者在苹果系统中使用快捷键 command + F（macOS）即可打开查找工具，在搜索框中输入需要查找的内容将显示如图 2.64 所示的界面效果。

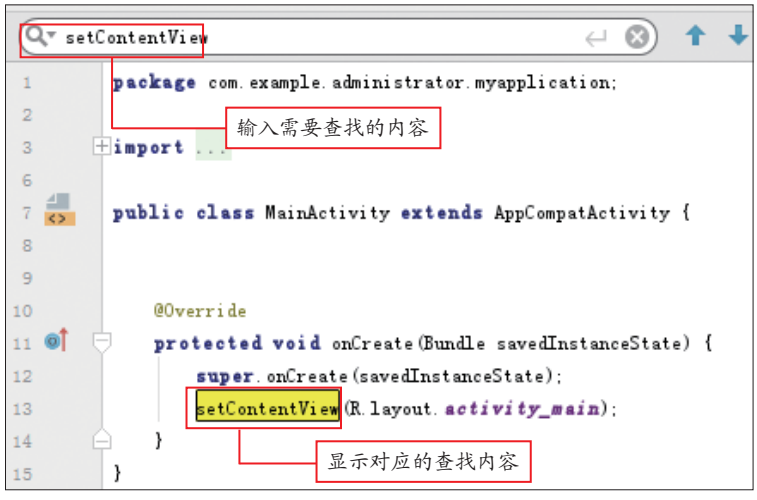


图 2.64　打开查找工具

示例 2：设置查找范围

在查找工具栏中可以设置查找范围，如图 2.65 所示。

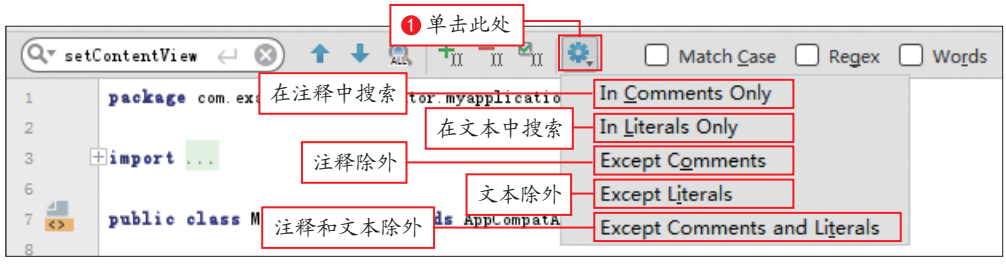


图 2.65　设置查找范围

例如，设置为只在注释中进行搜索将显示如图 2.66 所示。

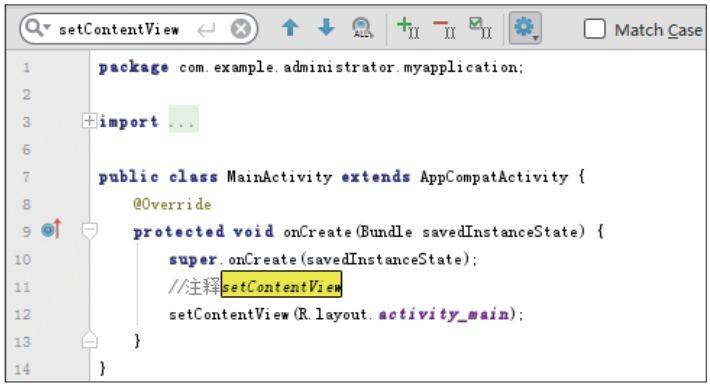


图 2.66　只在注释中进行搜索

示例 3：显示查找内容的路径

在一个项目中会出现多个需要查询的内容，此时可以根据查询内容的具体路径来判断是否是自己需要的内容。打开查找路径的窗口有以下两种方式：

◆ 在菜单栏中单击 Edit → Find → Find in Path…选项。

◆ 使用快捷键 Ctrl + Shift + F（Windows/Linux）或者在苹果系统中使用快捷键 shift + command + F（macOS）即可打开查找路径的窗口。

例如，在整个项目中查找 MainActivity 关键词，由于该项目中有两个 Module，所以将显示如图 2.67 所示的界面效果，两个红色框圈起的位置就是两个 Module 下相同关键字的内容。

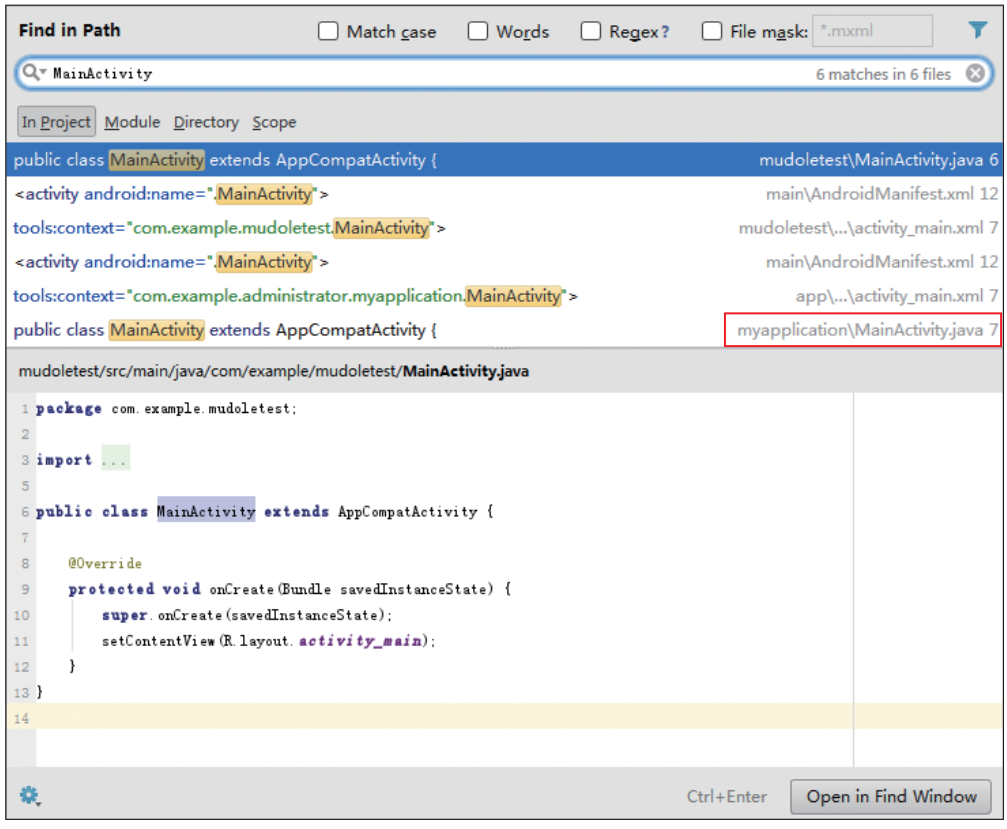


图 2.67　打开查找路径的窗口

单击 Open in Find Window 按钮将显示如图 2.68 所示的详细目录。

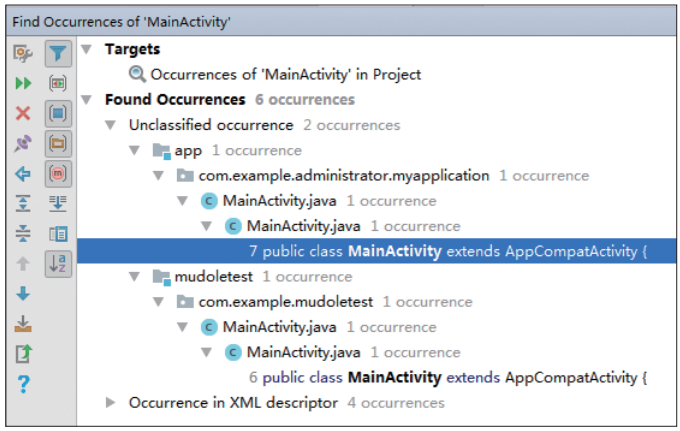


图 2.68　显示详细目录

6.　替换

学习了如何使用查找工具就该了解如何将查找的关键词替换成新的内容。下面通过以下示例来实现替换的使用方法。

示例 1：打开替换工具窗口

打开替换工具的窗口有以下两种方式：

◆ 打开需要替换内容的文件，然后在菜单栏中单击 Edit → Find → Replace 选项。

◆ 首先打开需要替换内容的文件，然后使用快捷键 <Ctrl + R>（Windows/Linux）或者在苹果系统中使用快捷键 <command + R>（macOS）即可打开替换工具窗口，如果有不需要替换的内容也可以单击上、下按钮选中，然后单击 Exclude 选项即可排除替换，如图 2.69 所示。

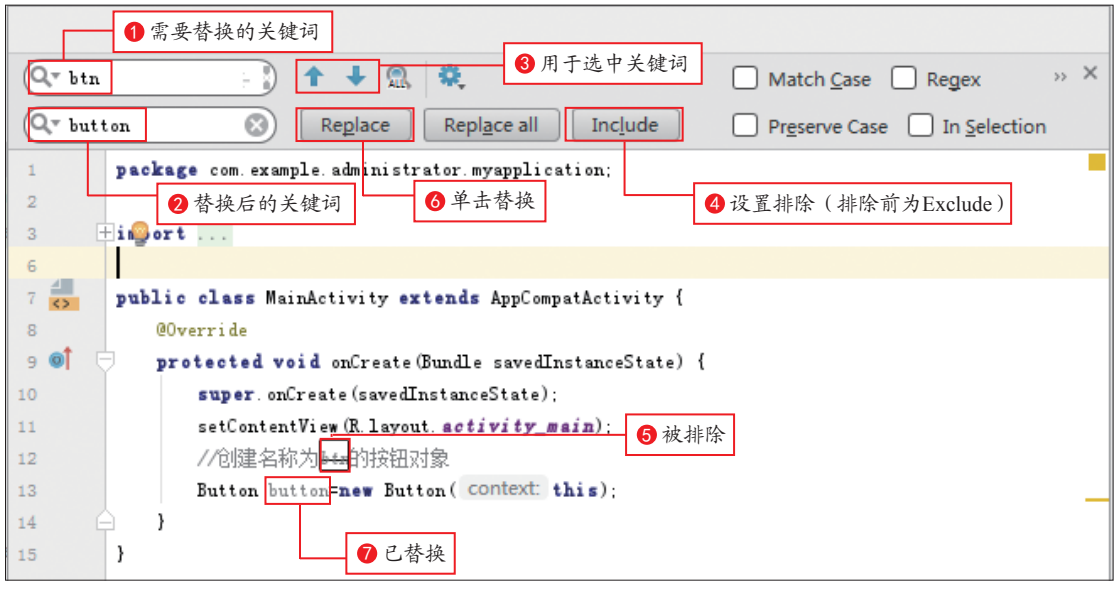


图 2.69　显示替换过程

示例 2：设置替换路径

替换功能不仅可以在指定的文件中进行替换，还可以指定其他的路径进行替换，比如在整个项目中、Module 中、指定的目录中或者设置筛选范围。

打开替换路径的窗口有以下两种方式：

◆ 在菜单栏中单击 Edit → Find → Replace in Path 选项即可。

◆ 使用快捷键 Ctrl + Shift + R（Windows/Linux）或者在苹果系统中使用快捷键 command + shift + R（macOS）即可打开设置替换路径的窗口，例如，在指定的 Module 当中进行关键字的替换，如图 2.70 所示。

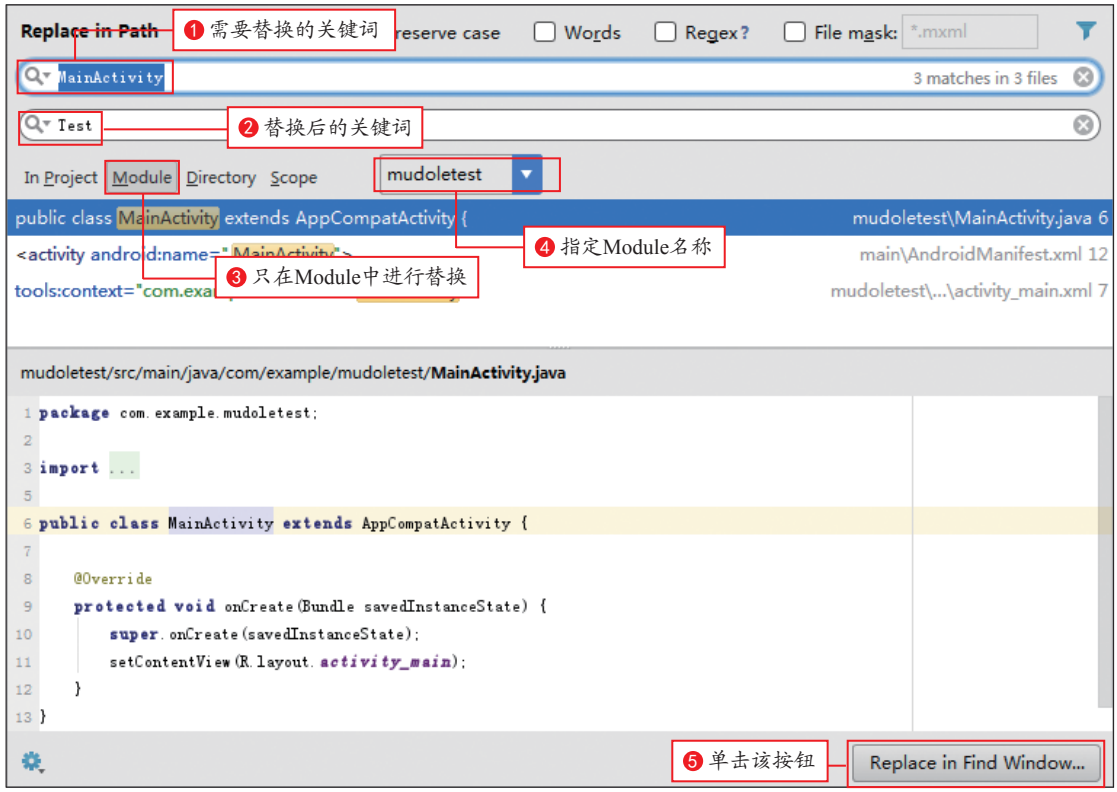


图 2.70　在指定的 Module 当中进行关键字的替换

单击 Replace in Find Window…按钮后将弹出确认替换的对话框，如图 2.71 所示。

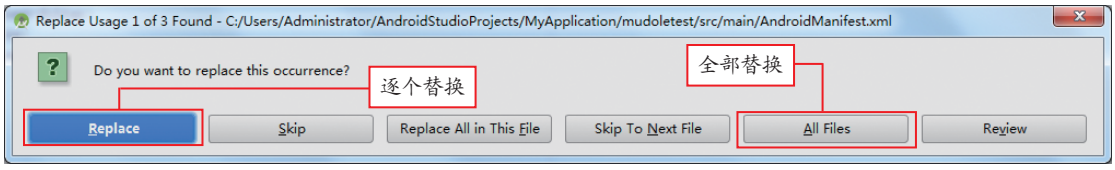


图 2.71　确认替换对话框

7.　查找使用位置

在代码中如果想知道某个方法或者是变量都在哪些位置用到时，可以通过以下的三种方式进行查找：

◆ 鼠标左键选中需要查询的方法，然后在菜单栏中单击 Edit → Find → Find Usages 选项将弹出确认对话框，单击 Yes 即可。

◆ 鼠标右键单击需要查询方法，然后在菜单中选择 Find Usages 选项，将弹出确认对话框，单击 Yes 即可。

◆ 鼠标左键选中需要查询的方法，然后使用快捷键 Alt + F7（Windows/Linux）或者在苹果系统中使用快捷键 fn + option + F7（macOS），将弹出确认对话框，单击 Yes 即可。

例如，查看 setContentView() 方法将显示如图 2.72 所示。

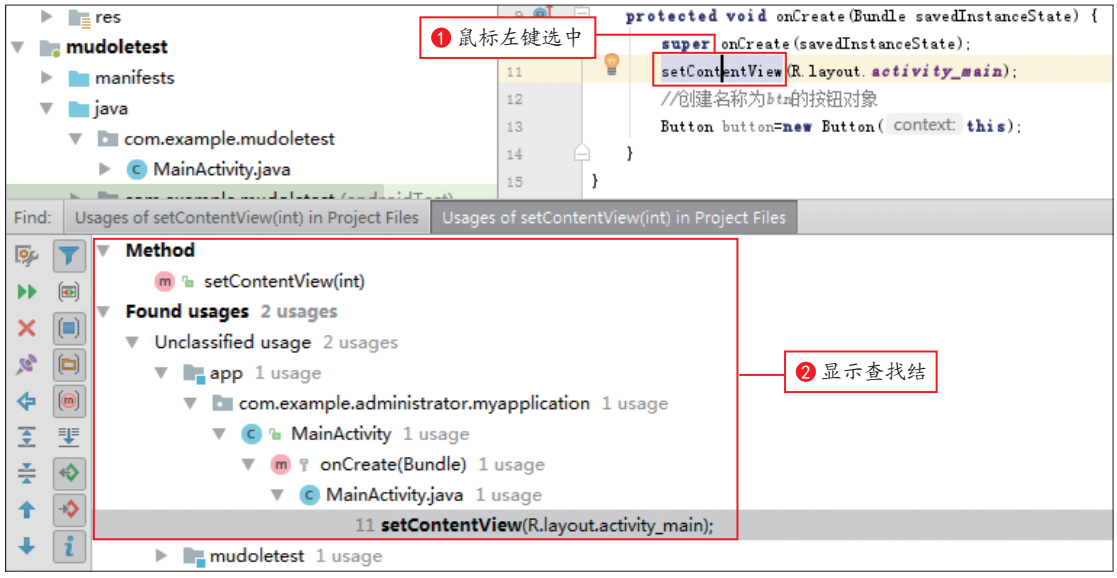


图 2.72　查看 setContentView() 方法在哪些地方使用

8.　标注关键词位置

如果想查看某个关键词在整个文件中哪些位置出现过，此时可以通过高亮显示的方法进行查看。启动高亮显示的方式有以下两种：

◆ 鼠标左键选中需要高亮显示的关键字，然后在菜单栏中单击 Edit → Find → Highlight Usages in File 选项即可。

◆鼠标左键选中需要高亮显示的关键字，然后使用快捷键 Ctrl + Shift + F7（Windows/Linux）或者在苹果系统中使用快捷键 <fn + shift + command + F7>（macOS）即可。

例如，将 Button 关键字进行高亮显示后 , 显示效果如图 2.73 所示。

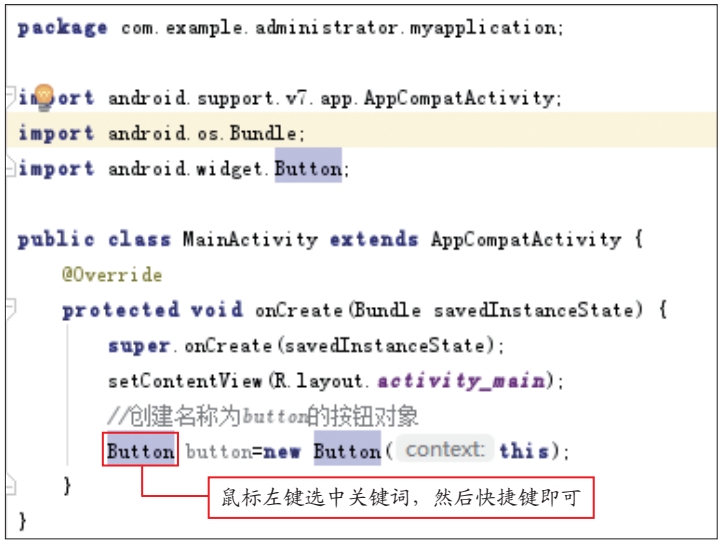


图 2.73　高亮显示关键字

2.4.2 视图类

AndroidStudio 开发工具中的视图操作也有很多的功能，例如，工具窗口的显示与隐藏、查看代码文档信息等。本节将介绍在 AndroidStudio 中通过快捷键以及菜单栏快速的使用这些操作技巧。

1. 显示与隐藏工具窗口

在 AndroidStudio 开发工具中单击菜单栏中的 View 即可显示如图 2.75 所示的菜单项，在该菜单中可以快速的设置 ToolBar（工具栏）、Tool Buttons（工具按钮）、Status Bar（状态栏）、Navigation Bar（导航栏）的显示与隐藏，还可以在 Tool Windows 选项中显示或隐藏常用的工具窗口，如图 2.74 所示。

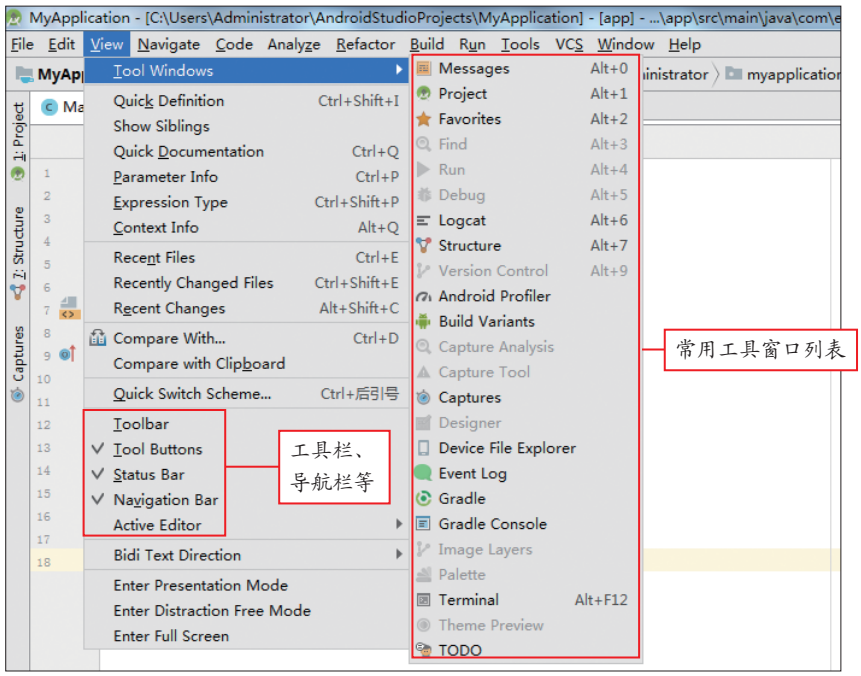


图 2.74 显示与隐藏工具窗口

不仅可以在工具栏中设置工具窗口的显示与隐藏，还可以通过快捷键来设置工具窗口的显示与隐藏，如表 2.1 所示。

表 2.1 来设置工具窗口的显示与隐藏的快捷键

窗 口 名 称	Windows/Linux 系统	macOS 系统
信息工具窗口	Alt + 0	command +0
项目工具窗口	Alt + 1	command + 1
收藏工具窗口	Alt + 2	command + 2
日志工具窗口	Alt + 6	command + 6
项目结构窗口	Alt + 7	command + 7
终端工具窗口	Alt + F12	fn + option + F12

除了以上的两种方式可以显示与隐藏工具窗口外，还可以在状态栏中进行设置，将鼠标移动至状态栏的左下角即可显示设置菜单，如图 2.75 所示，选择需要打开的工具窗口即可。

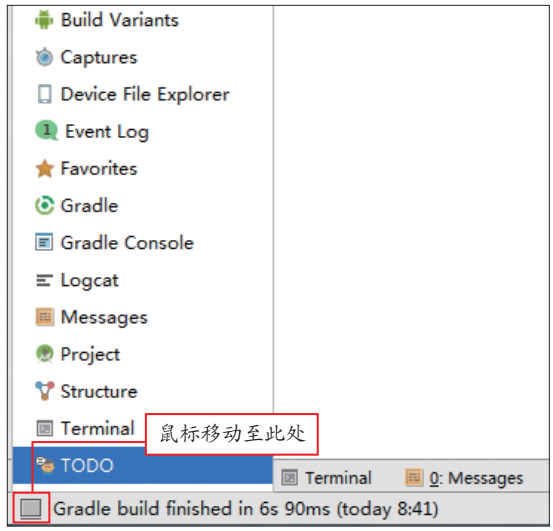


图 2.75　在状态栏中打开工具窗口

2.　快速查看帮助文档

在需要查询某个类或者是方法的帮助文档时，不一定非要在官网中进行搜索。在 AndroidStudio 中同样可以查询您需要的帮助文档，具体的查询方式有以下两种：

◆ 鼠标左键选中需要查询的类或方法，然后单击菜单栏中的 View → Quick Documentation 即可显示需要查询的帮助文档。

◆ 鼠标左键选中需要查询的类或方法，然后使用快捷键 Ctrl + Q（Windows/Linux）或者在苹果系统中使用快捷键 fn + F1（macOS）即可。

例如，查询 Button 控件的帮助文档如图 2.76 所示。

图 2.76　查询 Button 控件的帮助文档

说明　除了以上的两种方式可以快速地查询帮助文档以外，还可以设置鼠标悬停在某个元素上显示对应的帮助文档。在工具栏中单击File → Settings...打开设置对话框，然后在设置对话框中展开Editor选项，再选中General选项，最后勾选“Show quick documentation on mouse move”，如图2.77所示。

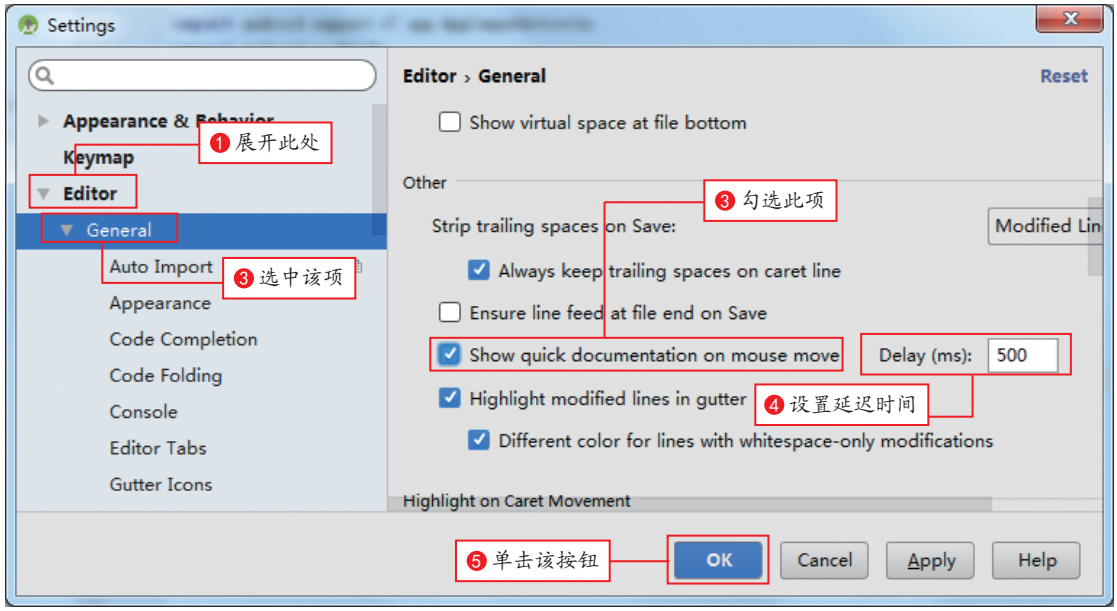


图 2.77　设置鼠标悬停在某个元素上显示对应的帮助文档

注意 在快速查看帮助文档的过程中如果出现如图2.78所示的现象，不要着急，等待一会即可显示帮助文档。

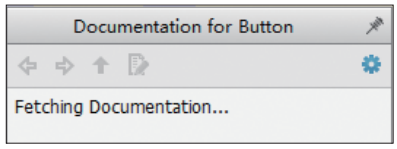


图 2.78　获取帮助文档

3.　快速查看方法参数信息

想快速查看某个方法参数的信息，可以通过以下的两种方式查看：

◆ 鼠标左键选中需要查询方法的括号内，然后单击菜单栏中的 View → Parameter Info 选项即可显示方法参数信息。

◆鼠标左键选中需要查询方法的括号内，然后使用快捷键 Ctrl + P（Windows/Linux）或者在苹果系统中使用快捷键 <command + P>（macOS）即可。

例如，查看 setContentView() 方法的参数信息如图 2.79 所示。

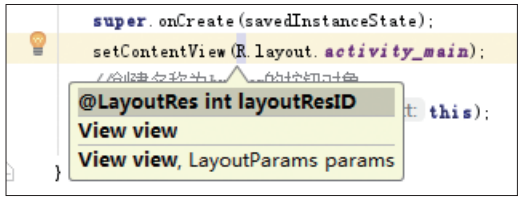


图 2.79　查看 setContentView() 方法的参数信息

4.　查看源码信息

如果要查看一个类或者方法的源码信息可以通过以下的三种方式进行查看：

◆ 鼠标左键选中需要查询内容，然后单击菜单栏中的 View → Jump to Source 选项即可。

◆ 鼠标左键选中需要查询内容，使用快捷键 F4（Windows/Linux）或者在苹果系统中使用快捷键 command + ↓（macOS）即可。

◆ 按住 Ctrl + 鼠标左键单击对应的方法或者是类，如果在苹果系统中需要按住 command 键，同样鼠标左键单击对应的方法或者是类即可。

2.4.3 编码类

AndroidStudio 提供了快速的重写方法或实现方法、代码自动补全以及代码格式化等功能，可以极大地提高编写代码的效率。本节将介绍在 AndroidStudio 中通过快捷键以及菜单栏如何提高编码效率。

1. 重写父类的方法

在编写代码的过程中如果需要重写父类的方法时，可以通过以下两种方法来快速地找到需要重写的方法：

◆ 打开需要重写方法的类，然后单击菜单栏中的 Code → Override Methods…选项，将显示需要重写方法的对话框。

◆ 打开需要重写方法的类，然后使用快捷键 Ctrl + O（Windows/Linux）或者在苹果系统中使用快捷键 control + O（macOS）即可。

例如，打开默认创建的 MainActivity 类重写该类中的 onStart()，然后使用快捷键 Ctrl + O 将显示如图 2.80 所示的选择重写方法的对话框。在该对话框中选择需要重写的方法，然后单击 ok 按钮，在代码编辑区将添加如图 2.81 所示的代码。

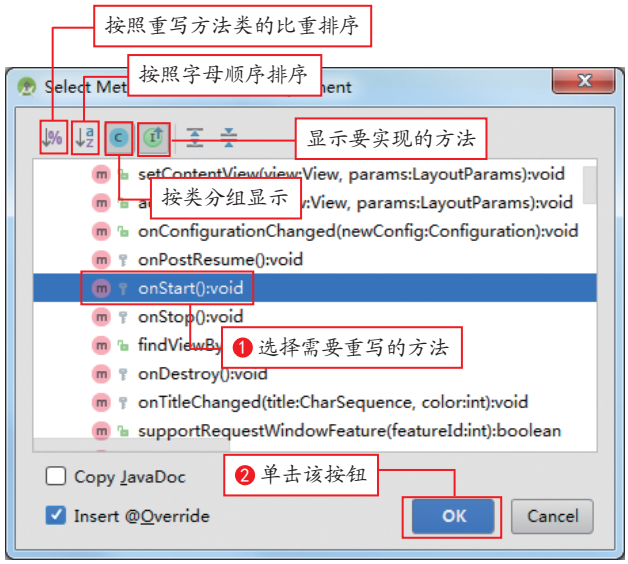


图 2.80 选择重写方法的对话框

图 2.81 添加重写方法的代码

说明

1.Override Method既可以重写父类的方法又可以实现接口中的方法。

2.在选择重写方法的对话框中，不仅可以选择一个方法，按住Ctrl键可以同时选择多个方法并且可以直接搜索您所需要重写方法的名称。

2. 实现接口中的方法

在只需要实现接口中的方法时，可以使用 Implement Methods 功能，具体使用方法有以下两种：

◆ 打开需要实现接口的类，然后添加实现接口的代码，单击菜单栏中的 Code → Override Methods 选项，将显示选择实现方法的对话框。

◆ 打开需要实现接口的类，然后添加实现接口的代码，使用快捷键 <Ctrl + I>（Windows/Linux）或者在苹果系统中使用快捷键 <control + L>（macOS）即可。

例如，打开默认创建的 MainActivity 类为该类实现 View.OnClickListener 接口，然后使用快捷键 Ctrl + I 将显示如图 2.82 所示的选择实现方法的对话框。在该对话框中选择需要实现的方法，然后单击 ok 按钮，在代码编辑区将添加如图 2.83 所示的代码。

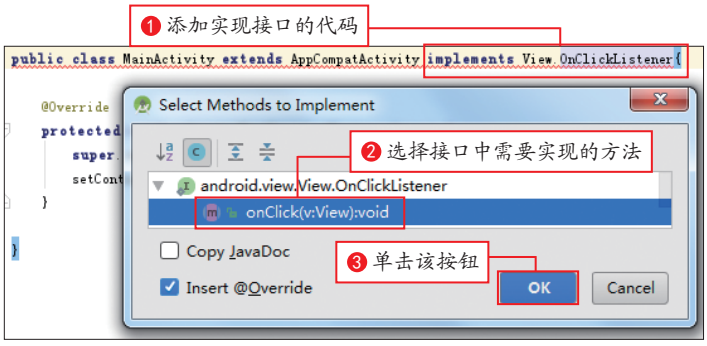


图 2.82 选择实现方法的对话框

```
@Override
public void onClick(View v) {

}
```

图 2.83 添加实现接口中方法的代码

3. 生成构造方法

在 AndroidStudio 开发工具中快速生成构造方法有以下两种方式：

◆ 单击菜单栏 Code → Generate 选择 Constructor 选项即可在代码编辑区域添加构造方法的代码。

◆ 使用快捷键 Alt + Insert 选择 Constructor（Windows/Linux）或者在苹果系统中使用快捷键 command + N 选择 Constructor 选项（macOS）即可。

例如，在 Demo 类中生成一个构造方法，首先将光标放在需要插入构造方法的位置，然后使用快捷键 Alt + Insert 选择 Constructor 选项，将显示如图 2.84 所示的选择在构造方法中初始化字段的对话框。选择后单击 ok 按钮，在代码编辑区将添加如图 2.85 所示的代码。

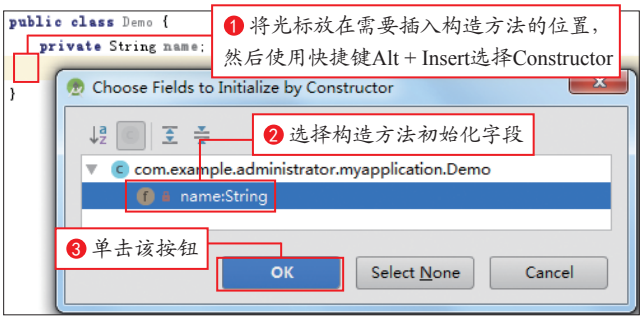


图 2.87 选择构造方法初始化字段的对话框

```
public class Demo {
    private String name;

    public Demo(String name) {
        this.name = name;
    }
}
```

图 2.85 添加构造方法代码

4. 生成 Getter 和 Setter 方法

生成 Getter 和 Setter 方法与生成构造方法类似，都需要在 Generate 列表中进行选择，具体的生成方式有以下两种：◆ 单击菜单栏 Code → Generate，选择 Getter and Setter 选项，即可在代码编辑区域添加 Getter 和 Setter 方法的代码。

◆ 使用快捷键 Alt + Insert 选择 Getter and Setter（Windows/Linux）或者在苹果系统中使用快捷键 command + N 选择 Getter and Setter 选项（macOS）即可。

说明 具体操作方式参照生成构造方法即可。

5. 注释代码

注释单行代码：

◆ 将光标移动到需要注释代码的位置，然后单击菜单栏中的 Code → Comment with Line Comment 即可。

◆ 将光标移动到需要注释代码的位置，使用快捷键 Ctrl + /（Windows/Linux）或者在苹果系统中快捷键 command+/（macOS）。如图 2.86 所示。

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

图 2.86 注释单行代码

注释代码块：

◆ 鼠标左键选中需要注释的代码块，然后单击菜单栏中的 Code → Comment with Block Comment 即可。

◆鼠标左键选中需要注释的代码块，然后使用快捷键 Ctrl + Shift + /（Windows/Linux）或者在苹果系统中使用快捷键 option + command + /（macOS）。如图 2.87 所示。

```
/*    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }*/
```

图 2.87 注释代码块

6. 格式化代码

为了让编写的代码看起来非常规整，AndroidStudio 设置了默认的代码风格，通过快速格式化功能可以让代码统一风格。快速格式化代码可以通过以下两种方式：

◆ 单击菜单栏中的 Code → Reformat Code 选项即可。

◆ 使用快捷键 Ctrl + Alt + L（Windows/Linux）或者在苹果系统中使用快捷键 option + command + L（macOS）即可。如图 2.88 所示。

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

格式化前

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

格式化后

图 2.88 格式化代码

2.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。

第 3 章

App UI 设计

通过前面的学习，相信读者已经对 Android 有了一定的了解，本章将学习 Android 开发中一项很重要的内容——App UI 设计。Android 提供了多种控制 UI 界面的方法、布局方式，以及大量功能丰富的 UI 组件，通过这些组件，可以像搭积木一样，开发出优秀的用户界面。

3.1 UI 设计相关的概念

我们要开发的 Android 应用是运行在手机或者平板电脑上的程序，这些程序给用户的第一印象就是用户界面，也就是 User Interface，简称 UI。在 Android 中，进行用户界面设计可以称为 UI 设计，在进行 UI 设计时，经常会用到 View 和 ViewGroup 类。对于初识 Android 的人来说，一般不好理解。下面将对这两个概念进行详细介绍。

3.1.1 View

View 在 Android 中可以理解为视图。它占据屏幕上的一个矩形区域，负责提供组件绘制和事件处理的方法。如果把 Android 界面比喻成窗户，那么每块玻璃都是一个 view，如图 3.1 所示。View 类是所有的 UI 组件（如第 2 章创建的实例“第一个 Android 应用”中使用的 TextView 就是 UI 组件）的基类。

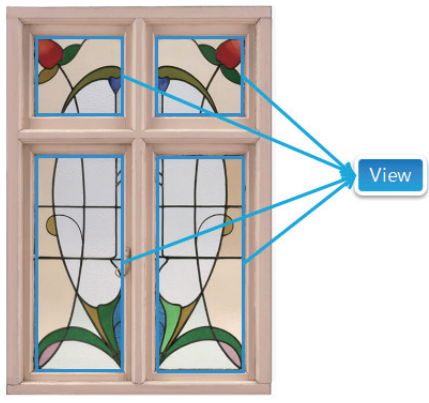


图 3.1　View 示意图

说明 View类位于android.view包中；文本框组件TextView是View类的子类，位于android.widget包中。

在 Android 中，View 类及其子类的相关属性，既可以在 XML 布局文件中进行设置，也可以通过成员方法在 Java 代码中动态设置。View 类常用的属性及对应的方法如表 3.1 所示。

表 3.1 View 类支持的常用 XML 属性及对应的方法

XML 属性	方 法	描 述
android:background	setBackgroundResource(int)	设置背景，其属性值为 Drawable 资源或者颜色值
android:clickable	setClickable(boolean)	设置是否响应单击事件，其属性值为 boolean 型的 true 或者 false
android:elevation	setElevation(float)	Android API 21 新添加的，用于设置 z 轴深度，其属性值为带单位的有效浮点数
android:id	setId(int)	设置组件的唯一标识符 ID，可以通过 findViewById() 方法获取
android:longClickable	setLongClickable(boolean)	设置是否响应长单击事件，其属性值为 boolean 型的 true 或者 false
android:minHeight	setMinimumHeight(int)	设置最小高度，其属性值为带单位的整数
android:minWidth	setMinimumWidth(int)	设置最小宽度，其属性值为带单位的整数
android:onClick		设置单击事件触发的方法
android:padding	setPaddingRelative(int,int,int,int)	设置 4 个边的内边距
android:paddingBottom	setPaddingRelative(int,int,int,int)	设置底边的内边距
android:paddingEnd	setPaddingRelative(int,int,int,int)	设置右边的内边距
android:paddingLeft	setPadding(int,int,int,int)	设置左边的内边距
android:paddingRight	setPadding(int,int,int,int)	设置右边的内边距
android:paddingStart	setPaddingRelative(int,int,int,int)	设置左边的内边距
android:paddingTop	setPaddingRelative(int,int,int,int)	设置顶边的内边距
android:visibility	setVisibility(int)	设置 View 的可见性

3.1.2 ViewGroup

ViewGroup 在 Android 中代表容器。如果还用窗户来比喻的话，ViewGroup 就相当于窗户框，用于控制玻璃的安放，如图 3.2 所示。ViewGroup 类继承自 View 类，它是 View 类的扩展，是用来容纳其他组件的容器，但是由于 ViewGroup 是一个抽象类，所以在实际应用中通常是使用 ViewGroup 的子类来作为容器，例如，在 3.3 节中将要介绍的布局管理器。

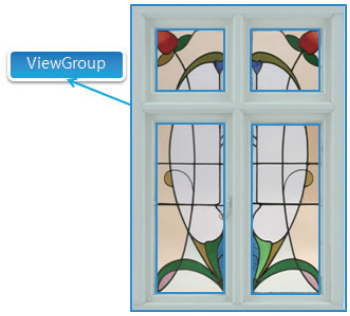


图 3.2　ViewGroup 示意图

ViewGroup 控制其子组件的分布时（例如，设置子组件的内边距、宽度和高度等），还经常依赖于 ViewGroup.LayoutParams 和 ViewGroup.MarginLayoutParams 两个内部类，下面分别进行介绍。

◆ ViewGroup.LayoutParams 类

ViewGroup.LayoutParams 类封装了布局的位置、高和宽等信息。它支持 android:layout_height 和 android:layout_width 两个 XML 属性，它们的属性值，可以使用精确的数值，也可以使用 FILL_PARENT（表示与父容器相同）、MATCH_PARENT（表示与父容器相同，需要 API 8 或以上版本才支持）或者 WRAP_CONTENT（表示包裹其自身的内容）指定。

◆ ViewGroup.MarginLayoutParams 类

ViewGroup.MarginLayoutParams 类用于控制其子组件的外边距。它支持的常用 XML 属性如表 3.2 所示。

表 3.2　ViewGroup.MarginLayoutParams 类支持的常用 XML 属性

XML 属性	描　　述
android:layout_marginBottom	设置底外边距
android:layout_marginEnd	该属性为 Android 4.2 新增加的属性，设置右外边距
android:layout_marginLeft	设置左外边距
android:layout_marginRight	设置右外边距
android:layout_marginStart	该属性为 Android 4.2 新增加的属性，用于设置左外边距
android:layout_marginTop	设置顶外边距

在 Android 中，所有的 UI 界面都是由 View 类和 ViewGroup 类及其子类组合而成的。在 ViewGroup 类中，除了可以包含普通的 View 类外，还可以再次包含 ViewGroup 类。实际上，这使用了 Composite（组合）设计模式。View 类和 ViewGroup 类的层次结构如图 3.3 所示。

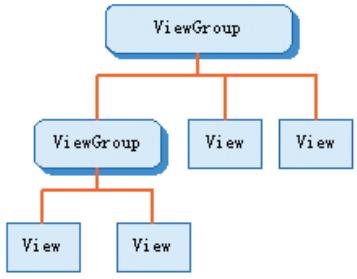


图 3.3　View 类和 ViewGroup 类的层次结构

3.2 控制 UI 界面

用户界面设计是 Android 应用开发的一项重要内容。在进行用户界面设计时，需要先了解界面中的 UI 元素如何呈现给用户，也就是采用何种控制 UI 界面的方法呈现给用户。

3.2.1 使用 XML 布局文件控制 UI 界面

Android 提供了一种非常简单、方便的方法用于控制 UI 界面。该方法采用 XML 文件来进行界面布局，从而将布局界面的代码和逻辑控制的 Java 代码分离开来，使程序的结构更加清晰、明了。

使用 XML 布局文件控制 UI 界面可以分为以下两个关键步骤。

（1）在 Android 应用的 res\layout 目录下创建 XML 布局文件，该布局文件的名称可以采用任何符合 Java 命名规则的文件名。

（2）在 Activity 中使用以下 Java 代码显示 XML 文件中布局的内容。

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

在上面的代码中，activity_main 是 XML 布局文件的文件名。

通过上面的步骤就可以轻松实现布局并显示 UI 界面的功能。下面通过一个例子来演示如何使用 XML 布局文件控制 UI 界面。

例 3.1 游戏的进入界面

（1）在 Android Studio 中打开一个已经存在的项目，然后在主菜单中选择 File → New → New Module 菜单项，将打开新建模块对话框，如图 3.4 所示。在该对话框中选择 Phone & Tablet Module 选项，创建针对手机或平板电脑的应用。

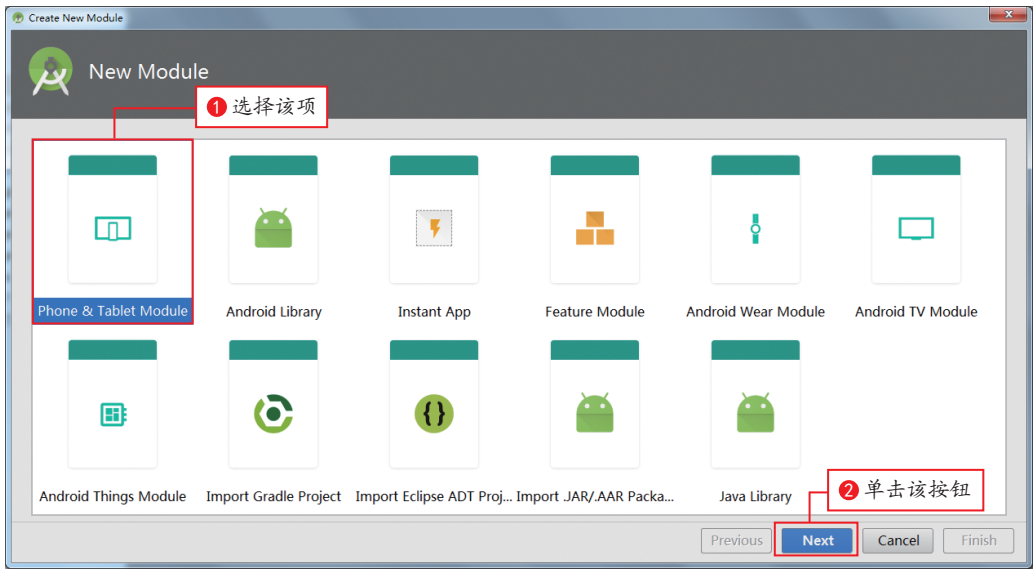


图 3.4 新建模块对话框

（2）单击 Next 按钮，将进入到配置新模块对话框，在该对话框中指定应用名称、模块名称、包名和最小 SDK 版本等信息，如图 3.5 所示。

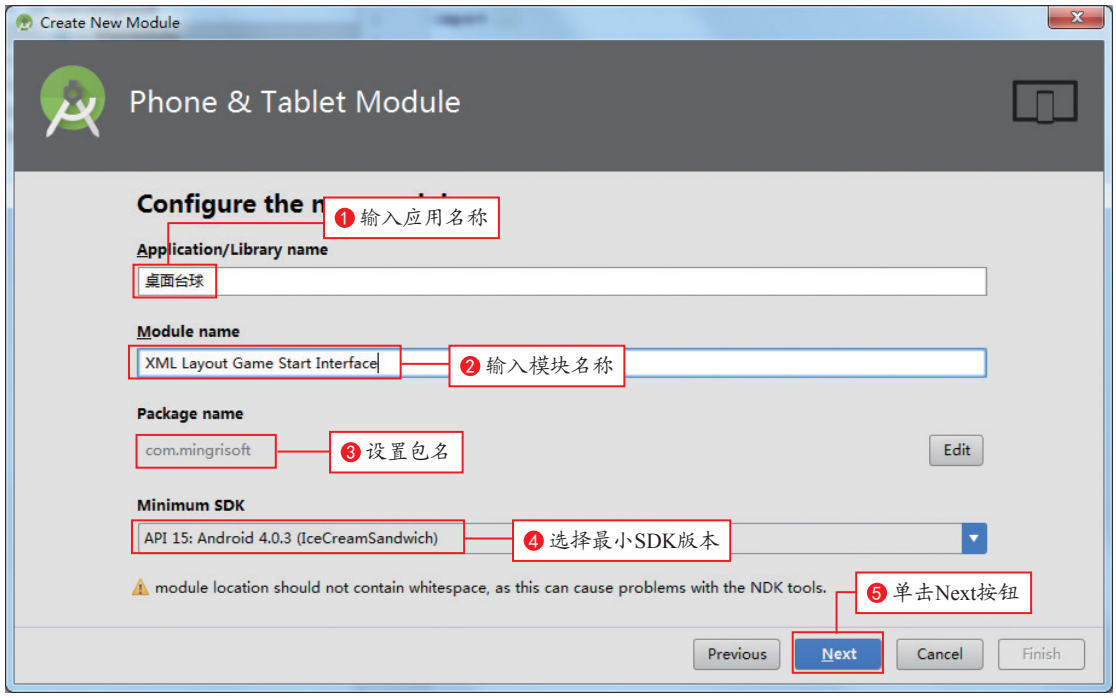


图 3.5　配置新的 Module

（3）单击 Next 按钮，将进入到选择创建 Activity 类型对话框，在该对话框中，将列出一些用于创建 Activity 的模板，这里我们选择创建一个空白的 Activity，即 Empty Activity。然后单击 Next 按钮，在进入的自定义 Activity 对话框中，设置自动创建的 Activity 的类名和布局文件名称，这里采用默认设置，单击 Finish 按钮完成 Module 的创建。

（4）在源码中，把名称为 bg.png 的背景图片复制到 mipmap-xhdpi 目录。

（5）修改 res\values 节点下的 strings.xml 文件，并且在该文件中添加一个用于定义开始按钮内容的常量，名称为 start，内容为“开始游戏”。修改后的代码如下：

```
<resources>
    <string name="app_name">桌面台球</string>
    <string name="start">开始游戏</string>
</resources>
```

说明　strings.xml文件用于定义程序中应用的字符串常量。其中，每一个<string>子元素都可以定义一个字符串常量，常量名称由name属性指定，常量内容写在起始标记<string>和结束标记</string>之间。

（6）修改新建 Module 的 res\layout 节点下的布局文件 activity_main.xml，将默认创建的布局管理器修改为帧布局管理器 FrameLayout，并且为其设置背景，然后修改默认添加的 TextView 组件，用于实现在窗体的正中间位置显示开始游戏按钮。修改后的代码如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="@mipmap/bg"
     tools:context="com.mingrisoft.MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textSize="18sp"
        android:textColor="#115572"
        android:text="@string/start" />
</FrameLayout>
```

说明 在布局文件activity_main中，通过设置布局管理器的android:background属性，可以为窗体设置背景图片；使用android:layout_gravity=" center" 可以让该组件在帧布局中居中显示；android:textSize属性用于设置字体大小；android:textColor属性用于设置文字的颜色。

（7）在主活动中，也就是 MainActivity 中，应用 setContentView() 方法指定活动应用的布局文件。不过，在应用 Android Studio 创建 Android 应用时，Android Studio 会自动在主活动的 onCreate() 方法中添加以下代码指定使用的布局文件，不需要我们手动添加。

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

说明 由于目前还没有学习Android中的UI组件，所以这里的“开始游戏”按钮先使用文本框组件代替。在实际应用开发时，通常采用按钮组件实现。

（8）运行本实例，将显示如图 3.6 所示的运行结果。

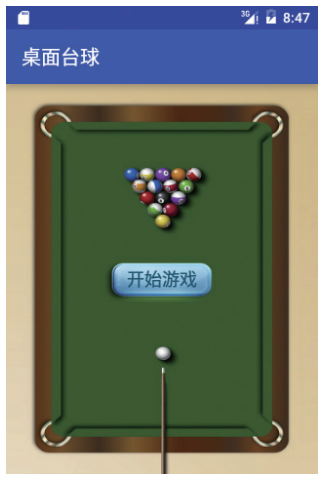


图 3.6　实现游戏的开始界面

3.2.2 在 Java 代码中控制 UI 界面

在 Android 中，支持像 Java Swing 那样完全通过代码控制 UI 界面。也就是所有的 UI 组件都通过 new 关键字创建出来，然后将这些 UI 组件添加到布局管理器中，从而实现用户界面。

在代码中控制 UI 界面可以分为以下 3 个关键步骤。

（1）创建布局管理器，例如，帧布局管理器、表格布局管理器、线性布局管理器、相对布局管理器和网格布局管理器等，并且设置布局管理器的属性。例如，为布局管理器设置背景图片等。

（2）创建具体的组件，例如，TextView、ImageView、EditText 和 Button 等任何 Android 提供的组件，并且设置组件的布局和属性。

（3）将创建的组件添加到布局管理器中。

下面我们将通过一个具体的例子来演示如何使用 Java 代码控制 UI 界面。

例 3.2 游戏的进入界面

在 Android Studio 中创建 Module，名称为“Game entry interface”。实现本实例的具体步骤如下：

（1）在新创建的 Module 中，打开 java/com.mingrisoft 节点下的 MainActivity.java 文件，然后将默认生成的下面这行代码删除。

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

（2）在 MainActivity 的 onCreate() 方法的上方声明一个 TextView 组件 text1，关键代码如下：

```
public TextView text1;
```

说明 在输入代码public TextView后，按下快捷键〈Ctrl+Enter〉，将显示如图3.7所示的提示框，单击Import class导入TextView类。

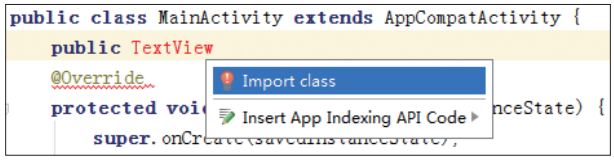


图 3.7 导入类的提示框

（3）在 MainActivity 的 onCreate() 方法中，创建一个帧布局管理器，并为该布局管理器设置背景，关键代码如下：

```
01  FrameLayout frameLayout = new FrameLayout(this); //创建帧布局管理器
02  frameLayout.setBackgroundResource(R.mipmap.bg);  //设置背景
03  setContentView(frameLayout);                     //设置在Activity中显示frameLayout
```

（4）实例化 text1 组件，设置其显示文字、文字大小、颜色和布局，具体代码如下：

```
01  text1 = new TextView(this);
02  text1.setText("开始游戏");                              //设置显示文字
03  text1.setTextSize(TypedValue.COMPLEX_UNIT_SP, 18); //设置文字大小，单位为SP（缩放像素）
04  text1.setTextColor(Color.rgb(17, 85, 114));         //设置文字的颜色
05  FrameLayout .LayoutParams params = new FrameLayout .LayoutParams(
06          ViewGroup.LayoutParams.WRAP_CONTENT,
07          ViewGroup.LayoutParams.WRAP_CONTENT);       //创建保存布局参数的对象
08  params.gravity = Gravity.CENTER;                    //设置居中显示
09  text1.setLayoutParams(params);                      //设置布局参数
```

说明 在通过setTextSize()方法设置TextView的文字大小时，可以指定使用的单位，在上面的代码中，int型的常量TypedValue.COMPLEX_UNIT_SP表示单位是可缩放像素，如果要设置单位是像素，可以使用常量TypedValue.COMPLEX_UNIT_PX，这些常量可以在Android官方提供的API中找到。

（5）实现单击“开始游戏”文本框时，显示询问对话框。具体方法是：为 text1 组件添加单击事件监听器，并在重写的 onClick() 方法中，显示询问对话框，关键代码如下：

```
text1.setOnClickListener(new View.OnClickListener() {  //为text1添加单击事件监听器
    @Override
    public void onClick(View v) {
        //设置对话框的标题
        new AlertDialog.Builder(MainActivity.this).setTitle("系统提示")
                //设置对话框的显示内容
                .setMessage("游戏有风险，进入需谨慎，真的要进入吗？")
                .setPositiveButton("确定",                    //为确定按钮添加单击事件
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                Log.i("桌面台球", "进入游戏");//输出消息日志
                            }
                        }).setNegativeButton("退出",          //为取消按钮添加单击事件
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,    int which) {
                        Log.i("桌面台球", "退出游戏");         //输出消息日志
                        finish();                             //结束游戏
                    }
                }).show();                                    //显示对话框
    }
});
```

说明 如果想要实现同例3.1一样的效果，那么步骤（5）可以省略。

（6）将文本框组件 text1 添加到布局管理器中，具体代码如下：

```
frameLayout.addView(text1);                          //将text1添加到布局管理器中
```

（7）运行本实例，将显示如图 3.8 所示的运行结果。

图 3.8　通过代码布局游戏开始界面

说明 完全通过代码控制UI界面，虽然该方法比较灵活，但是其开发过程比较烦琐，而且使得各模块之间的依赖性提高，进而降低了代码的重用性，因此不推荐采用这种方式控制UI界面。

3.2.3 使用 XML 和 Java 代码混合控制 UI 界面

完全通过 XML 布局文件控制 UI 界面，实现比较方便快捷，但是有失灵活；而完全通过 Java 代码控制 UI 界面，虽然比较灵活，但是开发过程比较烦琐。鉴于这两种方法的优缺点，下面来看另一种控制 UI 界面的方法，即使用 XML 和 Java 代码混合控制 UI 界面。

使用 XML 和 Java 代码混合控制 UI 界面，习惯上把变化小、行为比较固定的组件放在 XML 布局文件中，把变化较多、行为控制比较复杂的组件交给 Java 代码来管理。下面通过一个具体的实例来演示如何使用 XML 和 Java 代码混合控制 UI 界面。

例 3.3 QQ 相册照片列表

在 Android Studio 中创建 Module，名称为“QQ Album Photo List”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 节点下的布局文件 activity_main.xml，将默认添加的相对布局管理器修改为网格布局管理器，并将默认创建的 <TextView> 组件删除，然后为该网格布局管理器设置 android:id 属性，以及按水平方向排列，再设置该网格布局管理器包括 3 行 4 列。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:rowCount="3"
    android:columnCount="4"
    tools:context="com.mingrisoft.MainActivity">
</GridLayout>
```

（2）在 MainActivity 中，声明 img 和 imagePath 两个成员变量，其中，img 是一个 ImageView 类型的一维数组，用于保存 ImageView 组件；imagePath 是一个 int 型的一维数组，用于保存要访问的图片资源。关键代码如下：

```
private  ImageView[] img=new ImageView[12];          //声明一个保存ImageView组件的数组
private int[] imagePath=new int[]{
        R.mipmap.img01,R.mipmap.img02,R.mipmap.img03,R.mipmap.img04,
        R.mipmap.img05,R.mipmap.img06,R.mipmap.img07,R.mipmap.img08,
        R.mipmap.img09,R.mipmap.img10,R.mipmap.img11,R.mipmap.img12
};
```

（3）在 MainActivity 的 onCreate() 方法中，首先获取在 XML 布局文件中创建的网格布局管理器，然后通过一个 for 循环创建 12 个显示图片的 ImageView 组件，并将其添加到布局管理器中。关键代码如下：

```
01  //获取XML文件中定义的网格布局管理器
02  GridLayout layout=(GridLayout)findViewById(R.id.layout);
03  for(int i=0;i<imagePath.length;i++){
04      img[i]=new ImageView(MainActivity.this);        //创建一个ImageView组件
05      img[i].setImageResource(imagePath[i]);          //为ImageView组件指定要显示的图片
06      img[i].setPadding(2, 2,2, 2);                   //设置ImageView组件的内边距
07      //设置图片的宽度和高度
08      ViewGroup.LayoutParams params=new ViewGroup.LayoutParams(116,68);
09      img[i].setLayoutParams(params);                 //为ImageView组件设置布局参数
10      layout.addView(img[i]);                         //将ImageView组件添加到布局管理器中
11  }
```

（4）运行本实例，将显示如图 3.9 所示的运行结果。

说明 单击模拟器右侧菜单栏中的旋转按钮，可以将模拟器屏幕切换为横屏状态。

图 3.9　实现 QQ 相册照片列表页面

3.2.4 开发自定义的 View

一般情况下，开发 Android 应用程序的 UI 界面，都不直接使用 View 类和 ViewGroup 类，而是使用这两个类的子类。例如，要显示一个图片，就可以使用 View 类的子类 ImageView。虽然 Android 提供了很多继承了 View 类的 UI 组件，但是在实际开发时，还会出现不足以满足程序需要的情况。这时，我们就可以通过继承 View 类来开发自己的组件。开发自定义的 View 组件大致分为以下 3 个步骤。

（1）创建一个继承 android.view.View 类的 Java 类，并且重写构造方法。

注意 在自定义的View类中，至少需要重写一个构造方法。

（2）根据需要重写其他的方法。被重写的方法可以通过下面的方法找到。

在代码中单击鼠标右键，在弹出的快捷菜单中选择 Generate 菜单项，将打开如图 3.10 所示的快捷菜单，在该菜单中选择 Override Methods 菜单项，将打开如图 3.11 所示的选择覆盖或实现的方法对话框，在该对话框的列表中显示出了可以被重写的方法。我们只需要选中要重写方法前面的复选框，并单击“确定”按钮，Android Studio 将自动重写指定的方法。通常情况下，不需要重写全部的方法。

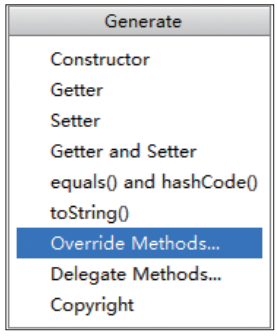


图 3.10　Generate 菜单项

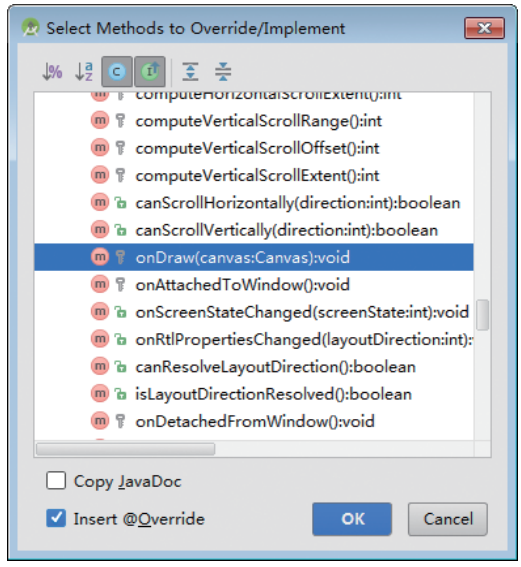


图 3.11　选择覆盖或实现的方法对话框

（3）在项目的活动中，创建并实例化自定义 View 类，然后将其添加到布局管理器中。下面通过一个实例演示如何开发自定义的 View。

例 3.4　跟随手指的小兔子

在 Android Studio 中创建 Module，名称为“Follow Finger Bunny”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/layout 节点下的布局文件 activity_main.xml，将默认创建的布局管理器修改为帧布局管理器 FrameLayout，并且设置其背景和 id 属性，然后将 TextView 组件删除。修改后的代码如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/background"
    android:id="@+id/mylayout"
    tools:context="com.mingrisoft.MainActivity" >
</FrameLayout>
```

（2）在 com.mingrisoft 包上单击右键，选择 View → Java Class 选项，新建一个名称为 RabbitView 的 Java 类，该类继承自 android.view.View 类，重写带一个参数 Context 的构造方法和 onDraw() 方法。其中，在构造方法中设置兔子的默认显示位置，在 onDraw() 方法中根据图片绘制小兔子。RabbitView 类的关键代码如下：

```
public class RabbitView extends View {
        public float bitmapX;                    //兔子显示位置的X坐标
        public float bitmapY;                    //兔子显示位置的Y坐标
        public RabbitView(Context context) {  //重写构造方法
```

```
            super(context);
            bitmapX = 210;                          //设置兔子的默认显示位置的X坐标
            bitmapY = 130;                          //设置兔子的默认显示位置的Y坐标
        }
    @Override
    protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);
            Paint paint = new Paint();              //创建并实例化Paint对象
            Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),
                    R.mipmap.rabbit);               //根据图片生成位图对象
            canvas.drawBitmap(bitmap, bitmapX, bitmapY, paint); //绘制小兔子
            if (bitmap.isRecycled()) {              //判断图片是否回收
                bitmap.recycle();                   //强制回收图片
            }
        }
  }
```

（3）在 MainActivity 的 onCreate() 方法中，首先获取帧布局管理器，并实例化小兔子对象 rabbit，然后为 rabbit 添加触摸事件监听器，在重写的触摸事件中设置 rabbit 的显示位置，并重绘 rabbit 组件，最后将 rabbit 添加到布局管理器中，关键代码如下：

```
FrameLayout frameLayout=(FrameLayout)findViewById(R.id.mylayout); //获取帧布局管理器
final RabbitView rabbit=new RabbitView(this);   //创建并实例化RabbitView类
//为小兔子添加触摸事件监听
rabbit.setOnTouchListener(new View.OnTouchListener() {

            @Override
    public boolean onTouch(View v, MotionEvent event) {
            rabbit.bitmapX=event.getX();        //设置小兔子显示位置的X坐标
            rabbit.bitmapY=event.getY();        //设置小兔子显示位置的Y坐标
            rabbit.invalidate();                //重绘rabbit组件
            return true;
        }
    });
frameLayout.addView(rabbit);                    //将rabbit自定义view添加到布局管理器中
```

（4）运行本实例，将显示如图 3.12 所示的运行结果。当用手指在屏幕上拖动时，小兔子将跟随手指的拖动轨迹移动。

图 3.12　跟随手指的小兔子

说明 单击模拟器右侧菜单栏中的旋转按钮，可以将模拟器屏幕切换为横屏状态。

3.3 布局管理器

在 Android 中，每个组件在窗体中都有具体的位置和大小，在窗体中摆放各种组件时，很难进行判断。不过，使用 Android 布局管理器可以很方便地控制各组件的位置和大小。Android 提供了以下 5 种布局管理器。

相对布局管理器（RelativeLayout）：通过相对定位的方式来控制组件的摆放位置。

线性布局管理器（LinearLayout）：是指在垂直或水平方向上依次摆放组件。

帧布局管理器（FrameLayout）：没有任何定位方式，默认情况下，所有的组件都会摆放在容器的左上角，逐个覆盖。

表格布局管理器（TableLayout）：使用表格的方式按行、列来摆放组件。

绝对布局管理器（AbsoluteLayout）：通过绝对定位（x、y 坐标）的方式来控制组件的摆放位置。

其中，绝对布局在 Android 2.0 中被标记为已过期，不过可以使用帧布局或相对布局替代。另外，在 Android 4.0 版本以后，又提供了一个新的布局管理器，网格布局管理器（GridLayout）。通过它可以实现跨行或跨列摆放组件。

Android 提供的布局管理器均直接或间接继承自 ViewGroup 类，如图 3.13 所示。因此，所有的布局管理器都可以作为容器使用，我们可以向布局管理器中添加多个 UI 组件。当然，也可以将一个或多个布局管理器嵌套到其他的布局管理器中，在本章的 3.3.6 节将对布局管理器的嵌套进行介绍。

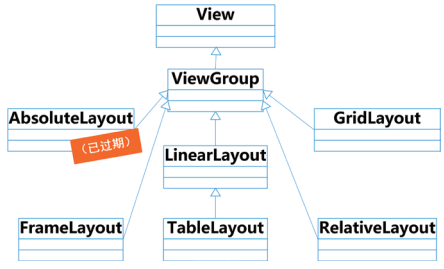


图 3.13 Android 布局管理器的类图

3.3.1 相对布局管理器

相对布局管理器是通过相对定位的方式让组件出现在布局的任何位置的。例如如图 3.14 所示的界面就是采用相对布局管理器来进行布局的，其中先放置组件 A，然后放置组件 B，让其位于组件 A 的下方，再放置组件 C，让其位于组件 A 的下方，并且位于组件 B 的右侧。

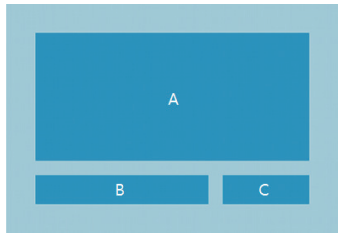


图 3.14 相对布局管理器示意图

在 Android 中，可以在 XML 布局文件中定义相对布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义相对布局管理器。在 XML 布局文件中，定义相对布局管理器可以使用 <RelativeLayout> 标记，其基本的语法格式如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
>
</RelativeLayout>
```

在上面的语法中，<RelativeLayout> 为起始标记，</RelativeLayout> 为结束标记。在起始标记中的 xmlns:android 为设置 XML 命名空间的属性，其属性值为固定写法。

说明 在Android中，无论是创建哪一种布局管理器都有两种方法，一种是在XML布局文件中定义，另一种是使用Java代码来创建。推荐使用的是在XML布局文件中定义，所以在本书中将只介绍在XML布局文件中创建这一种方法。

RelativeLayout 支持的常用 XML 属性如表 3.3 所示。

表 3.3 RelativeLayout 支持的常用 XML 属性

XML 属性	描　述
android:gravity	用于设置布局管理器中各子组件的对齐方式
android:ignoreGravity	用于指定哪个组件不受 gravity 属性的影响

在相对布局管理器中，只有上面介绍的两个属性是不够的，为了更好地控制该布局管理器中各子组件的布局分布，RelativeLayout 提供了一个内部类 RelativeLayout.LayoutParams，通过该类提供的大量 XML 属性，可以很好地控制相对布局管理器中各组件的分布方式。RelativeLayout.LayoutParams 支持的 XML 属性如表 3.4 所示。

表 3.4 RelativeLayout.LayoutParams 支持的常用 XML 属性

XML 属性	描　述
android:layout_above	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的上方
android:layout_alignBottom	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的下边界对齐
android:layout_alignLeft	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的左边界对齐
android:layout_alignParentBottom	其属性值为 boolean 值，用于指定该组件是否与布局管理器底端对齐
android:layout_alignParentLeft	其属性值为 boolean 值，用于指定该组件是否与布局管理器左边对齐
android:layout_alignParentRight	其属性值为 boolean 值，用于指定该组件是否与布局管理器右边对齐
android:layout_alignParentTop	其属性值为 boolean 值，用于指定该组件是否与布局管理器顶端对齐
android:layout_alignRight	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的右边界对齐
android:layout_alignTop	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的上边界对齐
android:layout_below	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的下方

续表

XML 属性	描　述
android:layout_centerHorizontal	其属性值为 boolean 值，用于指定该组件是否位于布局管理器水平居中的位置
android:layout_centerInParent	其属性值为 boolean 值，用于指定该组件是否位于布局管理器的中央位置
android:layout_centerVertical	其属性值为 boolean 值，用于指定该组件是否位于布局管理器垂直居中的位置
android:layout_toLeftOf	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的左侧
android:layout_toRightOf	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的右侧

下面编写一个在程序中使用相对布局管理器的实例。

例 3.5 软件更新提示界面

在 Android Studio 中创建 Module，名称为“Software Update Tips”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，把背景图片复制到 mipmap-xhdpi 目录中，将默认添加的布局管理器修改为相对布局管理器（RelativeLayout），然后为其设置背景，再设置默认添加的文本框（TextView）居中显示，并且为其设置 ID 和要显示的文字，最后在该布局管理器中，添加两个 Button，并设置它们的显示位置及对齐方式。修改后的代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg"
    tools:context="com.mingrisoft.MainActivity" >
    <!-- 添加一个居中显示的文本视图textView1 -->
    <TextView android:text="发现有Widget的新版本，您想现在就安装吗？"
        android:id="@+id/textView1"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        />
    <!-- 添加一个按钮button2，该按钮与textView1的右边界对齐 -->
    <Button
        android:text="以后再说"
        android:id="@+id/button2"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_alignRight="@id/textView1"
        android:layout_below="@id/textView1"
        />
    <!-- 添加一个在button2左侧显示的按钮button1 -->
    <Button
        android:text="现在更新"
```

```
26          android:id="@+id/button1"
27          android:layout_height="wrap_content"
28          android:layout_width="wrap_content"
29          android:layout_below="@id/textView1"
30          android:layout_toLeftOf="@id/button2"
31          />
32  </RelativeLayout>
```

说明 在上面的代码中，将提示文本组件textView1设置为在屏幕中央显示，然后设置“以后再说”按钮button2在textView1的下方居右边界对齐，最后设置“现在更新”按钮button1在“以后再说”按钮的左侧显示。

（2）运行本实例，将显示如图 3.15 所示的运行结果。

图 3.15　软件更新提示页面

3.3.2 线性布局管理器

线性布局管理器是将放入其中的组件按照垂直或水平方向来布局，也就是控制放入其中的组件横向排列或纵向排列。其中，纵向排列的称为垂直线性布局管理器，如图 3.16 所示；横向排列的称为水平线性布局管理器，如图 3.17 所示。在垂直线性布局管理器中，每一行中只能放一个组件，而在水平线性布局管理器中，每一列中只能放一个组件。另外 Android 的线性布局管理器中的组件不会换行，当组件一个挨着一个排列到窗体的边缘后，剩下的组件将不会被显示出来。

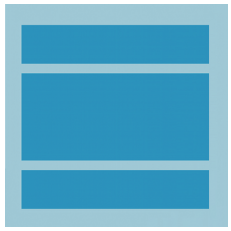


图 3.16　垂直线性布局管理器

图 3.17　水平线性布局管理器

说明 在线性布局管理器中，排列方式由android:orientation属性来控制，对齐方式由android:gravity属性来控制。

在 XML 布局文件中定义线性布局管理器，需要使用 <LinearLayout> 标记，其基本的语法格式如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    属性列表
>
</LinearLayout>
```

1. LinearLayout 的常用属性

LinearLayout 支持的常用 XML 属性如表 3.5 所示。

表 3.5 LinearLayout 支持的常用 XML 属性

XML 属性	描　述
android:orientation	用于设置布局管理器内组件的排列方式，其可选值为 horizontal 和 vertical，默认值为 vertical。其中，horizontal 表示水平排列，vertical 表示垂直排列
android:gravity	android:gravity 属性用于设置布局管理器内组件的显示位置，其可选值包括 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal。这些属性值也可以同时指定，各属性值之间用竖线隔开（竖线前后不能有空格）。例如要指定组件靠右下角对齐，可以使用属性值 right|bottom
android:layout_width	用于设置该组件的基本宽度，其可选值有 fill_parent、match_parent 和 wrap_content，其中 fill_parent 表示该组件的宽度与父容器的宽度相同；match_parent 与 fill_parent 的作用完全相同，从 Android 2.2 开始推荐使用；wrap_content 表示该组件的宽度恰好能包裹它的内容
android:layout_height	用于设置该组件的基本高度，其可选值有 fill_parent、match_parent 和 wrap_content，其中 fill_parent 表示该组件的高度与父容器的高度相同；match_parent 与 fill_parent 的作用完全相同，从 Android 2.2 开始推荐使用；wrap_content 表示该组件的高度恰好能包裹它的内容
android:id	用于为当前布局管理器指定一个 ID 属性，在 Java 代码中可以应用该属性单独引用这个布局管理器。为布局管理器指定 id 属性后，在 R.java 文件中，会自动派生一个对应的属性，在 Java 代码中，可以通过 findViewById() 方法来获取它
android:background	用于为该组件设置背景。可以是背景图片，也可以是背景颜色。为组件指定背景图片时，可以将准备好的背景图片复制到 drawable 目录下，然后使用下面的代码进行设置：android:background=”@drawable/background”如果想指定背景颜色，可以使用颜色值，例如，要想指定背景颜色为白色，可以使用下面的代码：android:background=”#FFFFFFFF”

说明 android:layout_width和android:layout_height属性是ViewGroup.LayoutParams所支持的XML属性。对于其他的布局管理器同样适用。

注意 在水平线性布局管理器中，子组件的android:layout_width属性值通常不设置为match_parent或fill_parent，如果这样设置，在该布局管理器中一行将只能显示一个组件；在垂直线性布局管理器中，android:layout_height属性值通常不设置为match_parent或fill_parent，如果这样设置，在该布局管理器中一列将只能显示一个组件。

2. 子组件在 LinearLayout 中的常用属性

在 LinearLayout 中放置的子组件，还经常用到如表 3.6 所示的两个属性。

表 3.6 LinearLayout 子组件的常用 XML 属性

XML 属性	描　述
android:layout_gravity	用于设置组件在其父容器中的位置。它的属性值与 android:gravity 属性相同，也是 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal。这些属性值也可以同时指定，各属性值之间用竖线隔开，但竖线前后一定不能有空格
android:layout_weight	用于设置组件所占的权重，即用于设置组件占父容器剩余空间的比例。该属性的默认值为 0，表示需要显示多大的视图就占据多大的屏幕空间。当设置一个高于零的值时，则将父容器的剩余空间分割，分割的大小取决于每个组件的 layout_weight 属性值。例如，在一个 320*480 的屏幕中，放置一个水平的线性布局管理器，并且在该布局管理器中放置两个组件，并且这两个组件的 android:layout_weight 属性值都设置为 1，那么，每个组件将分配到父容器的 1/2 的剩余空间。如图 3.18 所示

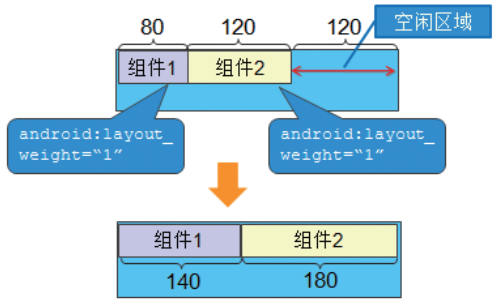


图 3.18 android:layout_weight 属性示意图

注意 在线性布局管理器的定义中，使用android:layout_gravity属性设置放入其中的组件的摆放位置不起作用，要想实现这一功能，需要使用android:gravity属性。

下面编写一个在程序中使用线性布局的实例。

例 3.6 登录微信界面

在 Android Studio 中创建 Module，名称为“WeChat Login”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为线性布局管理器 LinearLayout，然后将其设置为垂直线性布局管理器。修改后的代码如下：

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

```
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    tools:context="com.mingrisoft.MainActivity">
</LinearLayout>
```

（2）将名称为 zhanghao.png 和 mima.png 的图片复制到 mipmap-xxhdpi 目录中，并且在线性布局管理器中添加两个 EditText 组件，用于输入账号和密码，然后添加一个登录按钮，并且在登录按钮下面再添加一个 TextView，用来填写登录遇到的问题，关键代码如下：

```
<!--第一行-->
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="20dp"
    android:hint="QQ号/微信号/Email"
    android:drawableLeft="@mipmap/zhanghao"
    />
<!--第二行-->
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="20dp"
    android:hint="密码"
    android:drawableLeft="@mipmap/mima"
    />
<!--第三行-->
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="登录"
    android:textColor="#FFFFFF"
    android:background="#FF009688"/>
<!--第四行-->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="登录遇到问题?"
    android:gravity="center_horizontal"
    android:paddingTop="20dp"/>
```

说明 关于EditText（编辑框）、TextView（文本框）和Button（按钮）的详细介绍请参照第4章，这里知道这样用就可以。

（3）改变默认的主题为深色 ActionBar 主题。打开 AndroidManifest.xml 文件，将其中的 <application> 标记的 android:theme 属性值“@style/AppTheme”修改为“@style/Theme.AppCompat.Light.DarkActionBar”，修改后的 android:theme 属性的代码如下：

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

（4）运行本实例，将显示如图 3.19 所示的运行结果。

图 3.19　登录微信界面

3.3.3 帧布局管理器

在帧布局管理器中，每加入一个组件都将创建一个空白的区域，通常称为一帧，默认情况下，这些帧都会被放置在屏幕的左上角，即帧布局是从屏幕的左上角（0，0）坐标点开始布局。多个组件层叠排序，后面的组件覆盖前面的组件，如图 3.20 所示。

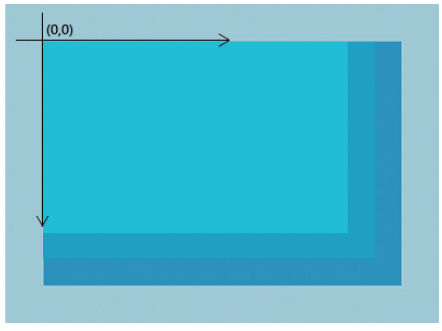


图 3.20　帧布局管理器

在 XML 布局文件中定义帧布局管理器可以使用 <FrameLayout> 标记，其基本的语法格式如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
    >
</FrameLayout>
```

FrameLayout 支持的常用 XML 属性如表 3.7 所示。

表 3.7　FrameLayout 支持的常用 XML 属性

XML 属性	描　　述
android:foreground	设置该帧布局管理器的前景图像
android:foregroundGravity	定义绘制前景图像的 gravity 属性，即前景图像显示的位置

下面编写一个在程序中使用帧布局管理器的实例。

例 3.7 居中显示层叠的正方形

在 Android Studio 中创建 Module，名称为“Frame Layout”实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局代码删除，然后添加一个 FrameLayout 帧布局管理器，并且为其设置背景和前景图像，以及前景图像显示的位置，之后再将前景图像文件 mr.png 复制到 mipmap-hdpi 目录下，最后在该布局管理器中，添加 3 个居中显示的 TextView 组件，并且为其指定不同的颜色和大小，用于更好地体现层叠效果。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:foreground="@mipmap/mr"
    android:foregroundGravity="bottom|right"
    tools:context="com.mingrisoft.MainActivity" >
    <!--添加居中显示的蓝色背景的TextView，将显示在最下层-->
    <TextView
        android:id="@+id/textView1"
        android:layout_width="280dp"
        android:layout_height="280dp"
        android:layout_gravity="center"
        android:background="#FF0000FF"
        android:textColor="#FFFFFF"
        android:text="蓝色背景的TextView" />
    <!--添加居中显示的天蓝色背景的TextView组件，将显示在中间层-->
    <TextView
        android:id="@+id/textView2"
        android:layout_width="230dp"
        android:layout_height="230dp"
        android:layout_gravity="center"
        android:background="#FF0077FF"
        android:textColor="#FFFFFF"
        android:text="天蓝色背景的TextView" />
    <!--添加居中显示的水蓝色背景的TextView组件，将显示在最上层-->
    <TextView
        android:id="@+id/textView3"
        android:layout_width="180dp"
        android:layout_height="180dp"
        android:layout_gravity="center"
        android:background="#FF00B4FF"
        android:textColor="#FFFFFF"
        android:text="水蓝色背景的TextView" />
</FrameLayout>
```

说明 帧布局管理器经常应用在游戏开发中，用于显示自定义的视图。例如，在3.2.4节的实例中，实现跟随手指的小兔子时就应用了帧布局管理器。

（2）运行本实例，将显示如图 3.21 所示的运行结果。

图 3.21　应用帧布局管理器居中显示层叠的正方形

3.3.4 表格布局管理器

表格布局管理器与常见的表格类似，它以行、列的形式来管理放入其中的 UI 组件，如图 3.22 所示。表格布局管理器使用 <TableLayout> 标记定义，在表格布局管理器中，可以添加多个 <TableRow> 标记，每个 <TableRow> 标记占用一行。由于 <TableRow> 标记也是容器，所以在该标记中还可添加其他组件，在 <TableRow> 标记中，每添加一个组件，表格就会增加一列。在表格布局管理器中，列可以被隐藏，也可以被设置为伸展的，从而填充可利用的屏幕空间，还可以设置为强制收缩，直到表格匹配屏幕大小。

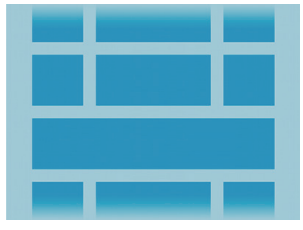


图 3.22　表格布局管理器

说明 如果在表格布局中，直接向<TableLayout>中添加UI组件，那么这个组件将独占一行。

在 XML 布局文件中定义表格布局管理器的基本语法格式如下：

```
<TableLayout  xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
   >
<TableRow 属性列表> 需要添加的UI组件 </TableRow>
多个<TableRow>
</TableLayout>
```

TableLayout 继承了 LinearLayout，因此它完全支持 LinearLayout 所支持的全部 XML 属性，此外，TableLayout 还支持如表 3.8 所示的 XML 属性。

表 3.8 TableLayout 支持的 XML 属性

XML 属性	描　述
android:collapseColumns	设置需要被隐藏的列的列序号（序号从 0 开始），多个列序号之间用逗号“,”分隔
android:shrinkColumns	设置允许被收缩的列的列序号（序号从 0 开始），多个列序号之间用逗号“,”分隔
android:stretchColumns	设置允许被拉伸的列的列序号（序号从 0 开始），多个列序号之间用逗号“,”分隔

下面编写一个在程序中使用表格布局的实例。

例 3.8 仿喜马拉雅的用户登录界面

在 Android Studio 中创建 Module，名称为“Xmly Login”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认添加的布局代码删除，然后添加一个 TableLayout 表格布局管理器，并且在该布局管理器中，添加一个背景图片，将需要的背景图片复制到 mipmap-mdpi 当中，然后添加 4 个 TableRow 表格行，接下来在每个表格行中添加相关的图片组件，最后设置表格的第 1 列和第 4 列允许被拉伸。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/biaoge"
    android:stretchColumns="0,3"
    tools:context="com.mingrisoft.MainActivity">
    <!--第一行-->
    <TableRow
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="200dp"
        >
        <TextView />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:text="账 号:"
            android:gravity="center_horizontal"
            />
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="邮箱或者手机号"
            />
```

```
28          <TextView />
29      </TableRow>
30      <!--第二行-->
31      <TableRow
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:paddingTop="20dp"
35          >
36          <TextView />
37          <TextView
38              android:layout_width="wrap_content"
39              android:layout_height="wrap_content"
40              android:textSize="18sp"
41              android:text="密 码:"
42              android:gravity="center_horizontal"
43              />
44          <EditText
45              android:layout_width="wrap_content"
46              android:layout_height="wrap_content"
47              android:hint="输入6～16位数字或字母"
48              />
49          <TextView />
50      </TableRow>
51      <!--第三行-->
52      <TableRow
53          android:layout_width="wrap_content"
54          android:layout_height="wrap_content">
55          <TextView />
56          <Button
57              android:layout_width="wrap_content"
58              android:layout_height="wrap_content"
59              android:text="注 册"
60              />
61          <Button
62              android:layout_width="wrap_content"
63              android:layout_height="wrap_content"
64              android:background="#FF8247"
65              android:text="登 录"/>
66          <TextView />
67      </TableRow>
68      <!--第四行-->
69      <TableRow
70          android:layout_width="wrap_content"
71          android:layout_height="wrap_content"
72          android:paddingTop="20dp"
73          >
74          <TextView />
```

```
75          <TextView />
76          <TextView
77              android:text="忘记密码？"
78              android:textColor="#FF4500"
79              android:gravity="right"
80              />
81          <TextView />
82      </TableRow>
83  </TableLayout>
```

说明 在本实例中，添加了6个<TextView />，并且设置对应列允许拉伸，这是为了让登录相关组件在水平方向上居中显示而设置的。

（2）运行本实例，将显示如图 3.23 所示的运行结果。

图 3.23 应用表格布局实现仿喜马拉雅的用户登录页面

3.3.5 网格布局管理器

网格布局管理器是在 Android 4.0 版本中提出的，使用 GridLayout 表示。在网格布局管理器中，屏幕被虚拟的细线划分成行、列和单元格，每个单元格放置一个组件，并且这个组件也可以跨行或跨列摆放，如图 3.24 所示。

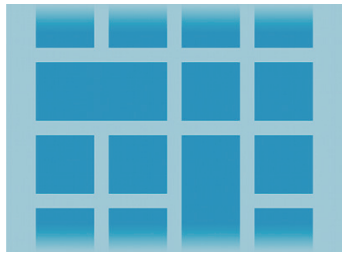


图 3.24 网格布局管理器示意图

说明 网格布局管理器与表格布局有些类似，都可以以行、列的形式管理放入其中的组件，但是它们之间最大的不同就是网格布局管理器可以跨行显示组件，而表格布局管理器则不能。

在 XML 布局文件中，定义网格布局管理器可以使用 <GridLayout> 标记，其基本的语法格式如下：

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
   >
</GridLayout >
```

GridLayout 支持的常用 XML 属性如表 3.9 所示。

表 3.9　GridLayout 支持的常用 XML 属性

XML 属性	描　　述
android:columnCount	用于指定网格的最大列数
android:orientation	用于当没有为放入其中的组件分配行和列时，指定其排列方式。其属性值为 horizontal 表示水平排列；属性值为 vertical 表示垂直排列
android:rowCount	用于指定网格的最大行数
android:useDefaultMargins	用于指定是否使用默认的边距，其属性值设置为 true 时，表示使用；属性值为 false 时，表示不使用
android:alignmentMode	用于指定该布局管理器采用的对齐模式，其属性值为 alignBounds 时，表示对齐边界；值为 alignMargins 时，表示对齐边距，默认值为 alignMargins
android:rowOrderPreserved	用于设置行边界显示的顺序和行索引的顺序是否相同，其属性值为 true，表示相同，属性值为 false，表示不相同
android:columnOrderPreserved	用于设置列边界显示的顺序和列索引的顺序是否相同，其属性值为 true，表示相同， 属性值为 false，表示不相同

为了控制网格布局管理器中各子组件的布局分布，网格布局管理器提供了 GridLayout.LayoutParams 内部类，在该类中提供了如表 3.10 所示的 XML 属性，用于控制网格布局管理器中各子组件的布局分布。

表 3.10　GridLayout.LayoutParams 支持的常用 XML 属性

XML 属性	描　　述
android:layout_column	用于指定该子组件位于网格的第几列
android:layout_columnSpan	用于指定该子组件横向跨几列（索引从 0 开始）
android:layout_columnWeight	用于指定该子组件在水平方向上的权重，即该组件分配水平剩余空间的比例
android:layout_gravity	用于指定该子组件采用什么方式占据该网格的空间，其可选值有 top（放置在顶部）、bottom（放置在底部）、left（放置在左侧）、right（放置在右侧）、center_vertical（垂直居中）、fill_vertical（垂直填满）、center_horizontal（水平居中）、fill_horizontal（水平填满）、center（放置在中间）、fill（填满）、clip_vertical（垂直剪切）、clip_horizontal（水平剪切）、start（放置在开始位置）、end（放置在结束位置）
android:layout_row	用于指定该子组件位于网格的第几行（索引从 0 开始）
android:layout_rowSpan	用于指定该子组件纵向跨几行
android:layout_rowWeight	用于指定该子组件在垂直方向上的权重，即该组件分配垂直剩余空间的比例

说明 在网格布局管理器中，如果想让某个组件跨行或跨列，那么需要先通过android:layout_columnSpan或者android:layout_rowSpan设置跨越的行或列数，然后再设置其layout_gravity属性为fill，表示该组件填满跨越的行或者列。

例 3.9 QQ 聊天信息列表界面

在 Android Studio 中创建 Module，名称为“QQ Chat Message”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/yout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为网格布局管理器，并且将默认添加的文本框组件删除，将需要的图片复制到 mipmap-mdpi 目录下，然后为该网格布局设置背景和列数。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="@mipmap/bg"
    android:columnCount="6"
    tools:context="com.mingrisoft.MainActivity" >
</GridLayout>
```

（2）添加第 1 行要显示的信息和头像，这里需要两个图像视图组件（ImageView），其中第 1 个 ImageView 用于显示聊天信息，占 4 个单元格，从第 2 列开始，居右放置；第 2 个 ImageView 用于显示头像，占 1 个单元格，位于第 6 列。具体代码如下：

```
<!-- 第一行 -->
<ImageView
    android:id="@+id/imageView1"
    android:src="@mipmap/a1"
    android:layout_gravity="end"
    android:layout_columnSpan="4"
    android:layout_column="1"
    android:layout_row="0"
    android:layout_marginRight="5dp"
    android:layout_marginBottom="20dp"
    />
<ImageView
    android:id="@+id/imageView2"
    android:src="@mipmap/ico2"
    android:layout_column="5"
    android:layout_row="0"
    />
```

代码注解

❶ 第5行代码，用于设置组件居右放置。

❷ 第6行代码，用于设置组件占4个单元格的位置。

❸ 第7行代码，用于指定组件放置在第2列。

❹ 第8行代码，用于指定组件放置在第1行。

❺ 第15行代码，用于指定组件放置在第6列。

（3）添加第 2 行要显示的信息和头像，这里也需要两个图像视图组件（ImageView），其中第 1 个 ImageView 用于显示头像，位于第二行的第二列；第 2 个 ImageView 用于显示聊天信息，位于第二行头像组件的下一列，具体代码如下：

```
<!-- 第二行 -->
<ImageView
    android:id="@+id/imageView3"
    android:src="@mipmap/ico1"
    android:layout_column="0"
    android:layout_row="1"
    />
<ImageView
    android:id="@+id/imageView4"
    android:src="@mipmap/b1"
    android:layout_row="1"
    android:layout_marginBottom="20dp"
    />
```

（4）按照步骤（2）和步骤（3）的方法再添加两行聊天信息。

（5）运行本实例，将显示如图 3.25 所示的运行结果。

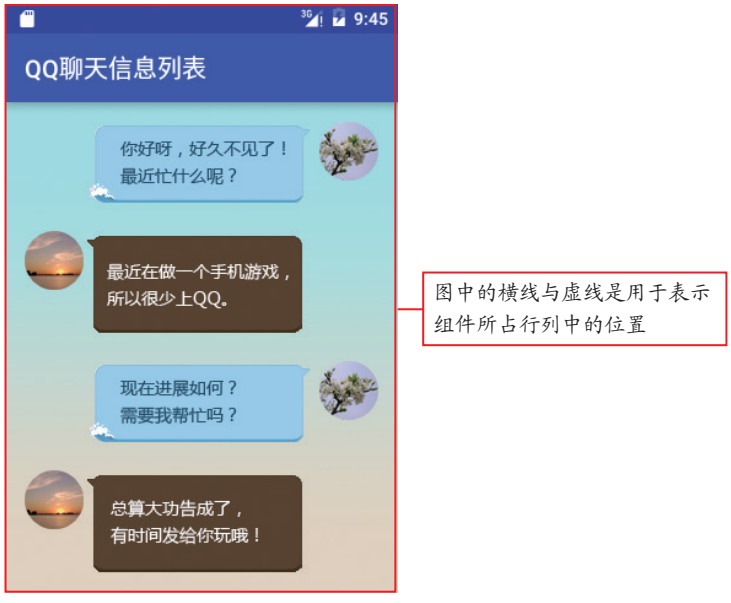


图 3.25　手机 QQ 聊天信息列表

3.3.6 布局管理器的嵌套

在进行用户界面设计时，很多时候只通过一种布局管理器很难实现想要的界面效果，这时就需要将多种布局管理器混合使用，即布局管理器的嵌套。在实现布局管理器的嵌套时，只需要记住以下几点原则就可以了。

◆ 根布局管理器必须包含 xmlns 属性。

◆ 在一个布局文件中，最多只能有一个根布局管理器。如果想要使用多个布局管理器，就需要使用一个根布局管理器将它们括起来。

◆ 不能嵌套太深。如果嵌套太深，则会影响性能，主要会降低页面的加载速度。

例 3.10 微信朋友圈界面

在 Android Studio 中创建 Module，名称为“WeChat Circle Of Friends”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 节点下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，然后将默认添加的文本框组件删除。修改后的代码如下：

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

（2）在步骤（1）中添加的垂直线性布局管理器中，添加一个用于显示第 1 条朋友圈信息的相对布局管理器，然后在该布局管理器中添加一个显示头像的图像视图组件（ImageView），让它与父容器左对齐，具体代码如下：

```
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp" >
<ImageView
    android:id="@+id/ico1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_margin="10dp"
    android:src="@mipmap/v_ico1" />
</RelativeLayout>
```

（3）在步骤（2）添加的相对布局管理器中，头像 ImageView 组件的右侧添加 3 个文本框组件，分别用于显示发布人、内容和时间，具体代码如下：

```
<TextView
android:id="@+id/name1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:layout_toRightOf="@+id/ico1"
android:text="雪绒花"
android:textColor="#576B95" />
<TextView
android:id="@+id/content1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/name1"
android:layout_marginBottom="5dp"
android:layout_marginTop="5dp"
android:layout_toRightOf="@+id/ico1"
android:minLines="3"
android:text="祝我的亲人、朋友们新年快乐！" />
<TextView
android:id="@+id/time1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/content1"
android:layout_marginTop="3dp"
android:layout_toRightOf="@id/ico1"
android:text="昨天"
android:textColor="#9A9A9A" />
```

（4）在上段代码的下面继续添加一个 ImageView 组件，用于显示评论图标，具体代码如下：

```
<ImageView
android:id="@+id/comment1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:layout_below="@id/content1"
android:src="@mipmap/comment" />
```

（5）在相对布局管理器的外面，线性布局管理器里面添加一个 ImageView 组件，用于显示一个分隔线，具体代码如下：

```
<ImageView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@mipmap/line" />
```

（6）按照步骤（2）到步骤（4）的方法再添加显示第 2 条朋友圈信息的代码。运行本实例，将显示如图 3.26 所示的运行结果。

图 3.26　微信朋友圈页面

3.4 常用 App UI 界面设计

对于一个 App 应用程序来说，简洁友好的用户界面（UI），自然会给用户优秀的体验，自然很容易就得到用户的认可和赞许，所以作为开发 App 的第一步，UI 尤为重要，通过以上的学习内容，相信读者已经掌握了多种控制 UI 界面的方法与布局方式，下面通过一个具体的实战来完成一个简单新闻类应用。

例 3.11　微信精选

在 Android Studio 中新建 Module，名称为“MRNews”，具体步骤如下：

（1）打开 build.gradle（Module: MRNews）文件，然后在该文件中的 dependencies 节点中添加依赖库的代码。具体代码如下：

```
//网络okhttp3请求框架
compile 'com.squareup.okhttp3:logging-interceptor:3.4.1'
//阿里巴巴提供的开源 json处理框架
//https://mvnrepository.com/artifact/com.alibaba/fastjson
compile group: 'com.alibaba', name: 'fastjson', version: '1.2.41'
//Fresco图片处理框架 https://www.fresco-cn.org/
compile 'com.facebook.fresco:fresco:1.5.0'
//安卓依赖库控件
compile 'com.android.support:design:26.+'
//封装好的开源频道管理功能
compile 'com.github.andyoom:draggrid:v1.0.1'
```

说明　由于使用开源频道管理的依赖框架，所以还需要在build.gradle（Module: MRNews）文件中添加以下代码用于引入频道管理框架的库文件。具体代码如下：

```
allprojects {
```

```
    repositories {
        jcenter()
        //频道管理的引入库地址
        maven { url "https://jitpack.io" }
    }
}
```

（2）修改新建Module的res/layout目录下的布局文件activity_main.xml，根据界面划分为标题栏，横向菜单栏，内容显示区域。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <!--标题栏布局-->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="55dp"
            android:background="@android:color/holo_red_dark">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginLeft="10dp"
                android:text="微信精选"
                android:textColor="#FFFFFF"
                android:textSize="20sp" />
        </LinearLayout>
        <!--横向选择新闻类型的布局-->
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <!--横向滚动控件-->
            <HorizontalScrollView
                android:layout_width="wrap_content"
                android:layout_height="50dp"
                android:layout_gravity="center_vertical"
                android:layout_marginRight="40dp"
                android:scrollbars="none">
                <LinearLayout
                    android:layout_width="match_parent"
```

```
38                      android:layout_height="match_parent">
39                      <!--自定义显示新闻类型的控件-->
40                      <com.mingrisoft.mrnews.view.MyGridView
41                          android:id="@+id/gv_topmen"
42                          android:layout_width="2000dp"
43                          android:layout_height="match_parent"
44                          android:layout_gravity="center_vertical"
45                          android:listSelector="@android:color/transparent">
46                      </com.mingrisoft.mrnews.view.MyGridView>
47                  </LinearLayout>
48              </HorizontalScrollView>
49              <!--添加新闻类型的按钮布局-->
50              <LinearLayout
51                  android:id="@+id/ll_channel"
52                  android:layout_width="wrap_content"
53                  android:layout_height="50dp"
54                  android:layout_alignParentRight="true"
55                  android:layout_centerVertical="true"
56                  android:background="#e0F8f8f8"
57                  android:gravity="center_vertical"
58                  android:paddingLeft="10dp"
59                  android:paddingRight="10dp">
60                  <!--添加按钮-->
61                  <ImageView
62                      android:layout_width="wrap_content"
63                      android:layout_height="wrap_content"
64                      android:background="@drawable/addbj" />
65              </LinearLayout>
66          </RelativeLayout>
67          <!--横线-->
68          <View
69              android:layout_width="match_parent"
70              android:layout_height="1dp"
71              android:background="#50666666" />
72          <!--自定义View实现SwipeRefreshLayout，添加上拉加载更多的功能的内容显示区域-->
73          <com.mingrisoft.mrnews.view.SwipeRefreshView
74              android:id="@+id/srl"
75              android:layout_width="match_parent"
76              android:layout_height="match_parent">
77
78              <ListView
79                  android:id="@+id/lv"
80                  android:layout_width="match_parent"
81                  android:layout_height="match_parent"
82                  android:divider="#00000000"
83                  android:dividerHeight="0dip"
```

```
84                  android:listSelector="@android:color/transparent" />
85          </com.mingrisoft.mrnews.view.SwipeRefreshView>
86      </LinearLayout>
87  </RelativeLayout>
```

（3）创建 CategorylistAdapter 类与 SearchlistAdapter 类，分别是显示顶部新闻分类名称的列表适配器与分类所对应的新闻数据列表适配器，此处代码参照源码即可。

（4）在 MainActivity 中初始化控件并创建 DownloadTopMeandata() 方法获取网络数据，创建 inview() 方法监听点击事件。具体代码如下：

```
01  protected void onCreate(Bundle savedInstanceState) {
02      super.onCreate(savedInstanceState);
03      setContentView(R.layout.activity_main);
04      //初始化控件
05      myGridView = (MyGridView) findViewById(R.id.gv_topmen);
06      lv = (ListView) findViewById(R.id.lv);
07      swipeRefreshView = (SwipeRefreshView) findViewById(R.id.srl);
08      ll_channel = (LinearLayout) findViewById(R.id.ll_channel);
09      manager = OkhttpUtil.getInstance();//初始化网络请求工具
10      //获取顶部数据
11      DownloadTopMeandata();
12      //监听 下拉刷新上拉加载事件 点击事件
13      inview();
14  }
```

（5）DownloadTopMeandata() 方法中通过导入的工具类 OkhttpUtil 请求网络并添加到顶部栏目中，具体代码如下：

```
01  //请求顶部数据
02  private void DownloadTopMeandata() {
03      //封装请求阐述
04      Map<String, String> map = new HashMap<String, String>();
05      map.put("key", AppKey.appKey);
06      //使用自定义网络请求工具类 请求网络获取数据
07      manager.postAsync(AppKey.TopUrl, map, new OkhttpUtil.DataCallBack() {
08          //数据接收失败的时候调用
09          @Override
10          public void requestFailure(Request request, IOException e) {
11              Toast.makeText(MainActivity.this,
12                      "数据请求失败，请重新进入", Toast.LENGTH_SHORT).show();
13          }
14          //数据接收成功时候调用该方法
15          @Override
16          public void requestSuccess(String result) throws Exception {
17              categoryBean = new CategoryBean();
18              categoryBean = JSON.parseObject(result, CategoryBean.class);
19              if (!categoryBean.getRetCode().equals("200")) {//判断数据是否正确返回
```

```
                //错误码   说明
                //10001    appkey不合法
                //10020    接口维护
                //10021    接口停用
                //200  成功
                String ErrorCode = "";
                if (categoryBean.getRetCode().equals("10001")) {
                    ErrorCode = "appkey不合法";
                }
                if (categoryBean.getRetCode().equals("10020")) {
                    ErrorCode = "接口维护";
                }
                if (categoryBean.getRetCode().equals("10021")) {
                    ErrorCode = "接口停用";
                }
                Toast.makeText(MainActivity.this,
                        "数据错误!原因:" + ErrorCode, Toast.LENGTH_SHORT).show();
            } else {
                //正确返回后处理数据
                //更新最新所有频道
                TopMeans = categoryBean.getResult();

                //判断显示频道是否为空
                if (TopMean.isEmpty()) {
                    //设置显示频道  显示全部
                    TopMean = categoryBean.getResult();
                }
                //设置初始cati值
                catidType = TopMean.get(0).getCid();
                //设置频道显示个数
                myGridView.setNumColumns(TopMean.size());
                //初始化适配器绑定数据
                categorylistAdapter = new
                        CategorylistAdapter(TopMean, catidType, MainActivity.this);
                //绑定适配器  显示数据
                myGridView.setAdapter(categorylistAdapter);
                //根据citid值  获取最新数据
                Refresh(catidType);
            }
        }
    });
}
```

说明 DownloadTopMeandata()方法中的OkhttpUtil为网络请求工具类，通过调用该类发送网络请求并获取网络数据，该类为封装类在源码中查看即可。

（6）事件监听包括列表的下拉与加载，顶部的菜单单击，以及列表条目的单击事件都将在inview() 方法中执行，具体代码如下：

```
01      private void inview() {
02       //下拉时触发SwipeRefreshLayout的下拉动画，动画完毕之后就会回调这个方法
03       swipeRefreshView.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
04              @Override
05              public void onRefresh() {
06                  //开始刷新，设置当前为刷新状态
07                  //swipeRefreshLayout.setRefreshing(true);
08                  Refresh(catidType);
09              }
10          });
11          //设置下拉加载更多
12       swipeRefreshView.setOnLoadMoreListener(new SwipeRefreshView.OnLoadMoreListener() {
13              @Override
14              public void onLoadMore() {
15                  Load(catidType);
16              }
17          });
18          //列表单击事件
19          lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
20              @Override
21        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
22                  //要跳转页面 使用intent初始化intent
23                  Intent intent = new Intent();
24                  //在Intent对象当中添加一个键值对 传递要详情url
25                  intent.putExtra("url", DataList.get(position).getSourceUrl());
26                  //设置Intent对象要启动的Activity
27                  intent.setClass(MainActivity.this, ListDetaActivity.class);
28                  //跳转页面
29                  startActivity(intent);
30              }
31          });
32          //顶部菜单点击事件
33          myGridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
34              @Override
35        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
36                  //根据按钮重新设置citid值
37                  catidType = TopMean.get(position).getCid();
38                  //重新加载 顶部菜单
39                  myGridView.setNumColumns(TopMean.size());
40                  categorylistAdapter = new
41                          CategorylistAdapter(TopMean, catidType, MainActivity.this);
42                  myGridView.setAdapter(categorylistAdapter);
43                  //根据citid值 获取最新数据
```

```
                Refresh(catidType);
            }
        });
        //频道按钮点击事件
        ll_channel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (list == null) {//判断集合中是否已有数据，没有则创建
                    list = new ArrayList<>();
                    //全部频道
                    for (int i = 0; i < TopMeans.size(); i++) {
                        list.add(new ChannelBean(TopMeans.get(i).getName(), true));
                    }
                    ChannelActivity.startChannelActivity(MainActivity.this, list);
                } //当判断保存的字符串不为空的时候，直接加载已经有了的字符串
                else if (jsonStr != null) {
                    ChannelActivity.startChannelActivity(MainActivity.this, jsonStr);
                }
            }
        });

    }
```

（7）创建 ListDetaActivity 类用于详情页逻辑处理，将需要展示信息的 url 通过 webview 显示在界面当中，具体代码如下：

```
public class ListDetaActivity extends AppCompatActivity {
    private TextView tv_back;
    private WebView web;
    private String url;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_deta);
        //返回按钮
        tv_back = (TextView) findViewById(R.id.tv_back);
        tv_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        web = (WebView) findViewById(R.id.web);
        //取得从上一个Activity当中传递过来的Intent对象
```

```
        Intent intent = getIntent();
        url = intent.getStringExtra("url");
        web.loadUrl(url);
        //启用支持javascript
        WebSettings settings = web.getSettings();
        settings.setJavaScriptEnabled(true);
    }
    //改写物理按键—返回的逻辑
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        //TODO Auto-generated method stub
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            if (web.canGoBack()) {
                web.goBack();//返回上一页面
                return true;
            } else {
                finish();
                //关闭activity动画显示
                overridePendingTransition(0, R.anim.activity_close);
            }
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

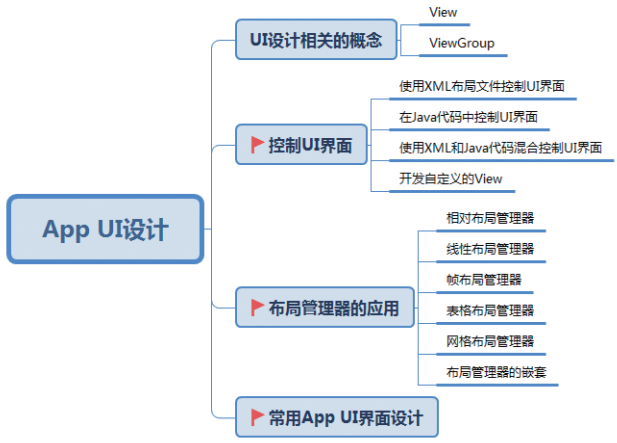
（8）运行本实例，将显示如图 3.27 所示的新闻信息主界面，单击感兴趣的新闻信息将显示如图 3.28 所示的新闻信息详情界面。

图 3.27　新闻信息主界面

图 3.28　新闻信息详情界面

3.5 知 识 回 顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 4 章

初级 UI 组件

组件是 Android 程序设计的基本组成单位，通过使用组件可以高效地开发 Android 应用程序。所以熟练掌握组件的使用是合理、有效地进行 Android 程序开发的重要前提。本章将对 Android 中提供的初级组件进行详细介绍。

4.1 文本类组件（初级）

Android 中提供了一些与文本显示、输入相关的组件，通过这些组件可以显示或输入文字。其中，用于显示文本的组件为文本框组件，用 TextView 类表示；用于编辑文本的组件为编辑框组件，用 EditText 类表示。这两个组件最大的区别是 TextView 类不允许用户编辑文本内容，而 EditText 类则允许用户编辑文本内容。它们的继承关系如图 4.1 所示。

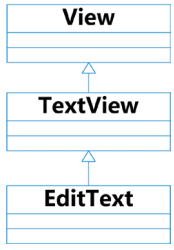


图 4.1　文本类组件继承关系图

从图 4.1 中可以看出：TextView 组件继承自 View，而 EditText 组件又继承自 TextView 组件。下面将对这两个组件分别进行介绍。

4.1.1 文本框

在 Android 手机应用中，文本框应用十分广泛。文本框主要用于在页面中显示文本信息。例如，蚂蚁庄园的动态界面（如图 4.2 所示），以及支付宝的福卡排行榜界面（如图 4.3 所示），都应用了文本框。

图 4.2　蚂蚁庄园动态页界面

图 4.3　福卡排行榜界面

说明 在图4.2和图4.3中用圆角矩形框圈起的内容均为文本框显示的内容。

在 Android 中，可以使用两种方法向屏幕中添加文本框：一种是通过在 XML 布局文件中使用 <TextView> 标记添加；另一种是在 Java 文件中，通过 new 关键字创建。推荐采用第一种方法，也就是通过 <TextView> 标记在 XML 布局文件中添加文本框，其基本的语法格式如下：

```
<TextView
属性列表
    >
</TextView>
```

说明 在Android中，无论是创建哪一种UI组件都有两种方法，一种是在XML布局文件中定义，另一种是使用Java代码来创建。Android官网中推荐使用的是在XML布局文件中定义。所以在本书中只介绍如何在XML布局文件中创建这一种方法。

TextView 支持的常用 XML 属性如表 4.1 所示。

表 4.1　TextView 支持的 XML 属性

XML 属性	描　　述
android:autoLink	用于指定是否将指定格式的文本转换为可单击的超链接形式，其属性值有 none、web、email、phone、map 和 all
android:drawableBottom	用于在文本框内文本的底部绘制指定图像，该图像可以是放在 res/mipmap 目录下的图片，通过“@mipmap/ 文件名（不包括文件的扩展名）”设置
android:drawableLeft	用于在文本框内文本的左侧绘制指定图像，该图像可以是放在 res/mipmap 目录下的图片，通过“@mipmap/ 文件名（不包括文件的扩展名）”设置
android:drawableStart	在 Android 4.2 中新增的属性，用于在文本框内文本的左侧绘制指定图像，该图像可以是放在 res\mipmap 目录下的图片，通过“@mipmap/ 文件名（不包括文件的扩展名）”设置

续表

XML 属性	描　述
android:drawableRight	用于在文本框内文本的右侧绘制指定图像，该图像可以是放在 res/mipmap 目录下的图片，通过“@mipmap/ 文件名（不包括文件的扩展名）”设置
android:drawableEnd	在 Android 4.2 中新增的属性，用于在文本框内文本的右侧绘制指定图像，该图像可以是放在 res\mipmap 目录下的图片，通过“@mipmap/ 文件名（不包括文件的扩展名）”设置
android:drawableTop	用于在文本框内文本的顶部绘制指定图像，该图像可以是放在 res/mipmap 目录下的图片，通过“@mipmap/ 文件名（不包括文件的扩展名）”设置
android:gravity	用于设置文本框内文本的对齐方式，可选值有 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal 等。这些属性值也可以同时指定，各属性值之间用竖线隔开。例如，要指定组件靠右下角对齐，可以使用属性值 right|bottom
android:hint	用于设置当文本框中文本内容为空时，默认显示的提示文本
android:inputType	用于指定当前文本框显示内容的文本类型，其可选值有 textPassword、textEmailAddress、phone 和 date 等，可以同时指定多个，使用“|”分隔
android:singleLine	用于指定该文本框是否为单行模式，其属性值为 true 或 false，为 true 表示该文本框不会换行，当文本框中的文本超过一行时，其超出的部分将被省略，同时在结尾处添加“…”
android:text	用于指定该文本框中显示的文本内容，可以直接在该属性值中指定，也可以通过在 strings.xml 文件中定义文本常量的方式指定
android:textColor	用于设置文本框内文本的颜色，其属性值可以是 #rgb、#argb、#rrggbb 或 #aarrggbb 格式指定的颜色值
android:textSize	用于设置文本框内文本的字体大小，其属性由代表大小的数值和单位组成，其单位可以是 dp、px、pt、sp 和 in 等
android:width	用于指定文本框的宽度，其单位可以是 dp、px、pt、sp 和 in 等
android:height	用于指定文本框的高度，其单位可以是 dp、px、pt、sp 和 in 等

说明 在表4.1中，只给出了TextView组件常用的部分属性，关于该组件的其他属性，可以参阅Android官方提供的API文档。在下载SDK时，如果已经下载Android API文档，那么可以在已经下载好的SDK文件夹下找到（docs文件夹中的内容即为API文档），否则需要自行下载。下载完成后，打开Android API文档主页（index.html），在Develop/ Reference中左侧的Android APIs列表中，单击android.widget节点，在下方找到TextView类并单击，在右侧就可以看到该类的相关介绍，其中XML Attributes表格中列出的就是该类的全部属性。

例如，在屏幕中添加一个文本框，显示文字为“奋斗就是每一天都很难，可一年比一年容易。不奋斗就是每一天都很容易，可一年比一年难。”，代码如下：

```
01  <TextView
02  android:layout_width="wrap_content"
03  android:layout_height="wrap_content"
```

```
04  android:text="奋斗就是每一天都很难，可一年比一年容易。不奋斗就是每一天都很容易，可一年比一年难。"
05  android:id="@+id/textView" />
```

在模拟器中运行上面这段代码，将显示如图 4.4 所示的运行结果。

图 4.4　添加一个文本框

图 4.5　添加单行文本框

对于文本框组件，默认为多行文本框，也可以设置为单行文本框，只需要将 android:singleLine 属性设置为 ture 就可以显示为单行文本框，例如，上面的多行文本框设置“android:singleLine=”true”” 属性后，将显示如图 4.5 所示的单行文本框。

下面通过一个实例来演示文本框的具体应用。

例 4.1　模拟福卡排行榜列表

在 Android Studio 中创建 Module，名称为“Ranking”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，添加 2 个 TextView 控件用于显示排行榜中第一名与第二名的人物名称，关键代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
06      android:layout_height="match_parent"
07      android:background="@drawable/bg"
08      android:orientation="vertical"
09      tools:context="com.mingrisoft.ranking.MainActivity">
10      <!--显示排行第一的名称-->
11      <TextView
12          android:id="@+id/text1"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:layout_marginLeft="100dp"
16          android:layout_marginTop="232dp" />
17      <!--显示排行第二的名称-->
18      <TextView
19          android:id="@+id/text2"
20          android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
22          android:layout_marginLeft="100dp"
23          android:layout_marginTop="48dp" />
24  </LinearLayout>
```

说明 根据第二名显示人物名称的控件与第一名控件的上下距离，添加其他4个控件即可。

（2）打开 MainActivity 类，该类继承 Activity 类，然后创建显示人物控件的 id 数组与显示人物名称的文字数组，代码如下：

```
01  private int[] text_Id = {
02          R.id.text1, R.id.text2, R.id.text3,
03          R.id.text4, R.id.text5, R.id.text6,
04  };
05  //控件需要显示的文字
06  private String[] text_String = {
07          "巴拉巴拉一大堆", "小科",
08          "鞋盒宝宝", "赵颖", "2047", "流浪的风"
09  };
```

（3）创建 initView() 方法，用于实现控件的初始化工作并设置排行榜人物名称，代码如下：

```
01  private void initView() {
02      for (int i=0;i<text_Id.length;i++){
03          TextView textView=findViewById(text_Id[i]);     //获取所有显示排名人物名称的控件
04          textView.setText(text_String[i]);               //设置人物名称
05      }
06  }
```

（4）运行本实例，将显示如图 4.6 所示的运行结果。

图 4.6　福卡排行榜列表

4.1.2 编辑框

在 Android 手机应用中，编辑框组件的应用非常普遍。例如，聚划算 App 的账号登录页面（如图 4.7 所示），以及微信的发送朋友圈信息页面（如图 4.8 所示）等，都应用了编辑框组件。

图 4.7 聚划算 App 账号登录页面

图 4.8 微信发送朋友圈消息页面

通过 <EditText> 标记在 XML 布局文件中添加编辑框的基本语法格式如下：

```
<EditText
属性列表
    >
</EditText>
```

由于 EditText 类是 TextView 的子类，所以表 4.1 中列出的 TextView 支持的 XML 属性，同样适用于 EditText 组件。需要特别注意的是，在 EditText 组件中，android:inputType 属性可以控制输入框的显示类型。例如，要添加一个密码框，可以将 android:inputType 属性设置为 textPassword。

技巧 在Android Studio中，打开布局文件，通过Design视图，可以在可视化界面中通过拖曳的方式添加编辑框组件，编辑框组件位于Palette面板的Text Fields栏目中，并且在该栏目中还列出了不同类型的输入框（如Password密码框、Password(Numeric)数字密码框和Phone输入电话号码的编辑框等），只需要将其拖曳到布局文件中即可。

在屏幕中添加编辑框后，还需要获取编辑框中输入的内容，这可以通过编辑框组件提供的 getText() 方法实现。使用该方法时，先要获取到编辑框组件，然后再调用 getText() 方法。例如，要获取布局文件中添加的 id 属性为 login 的编辑框的内容，可以通过以下代码实现：

```
01  EditText login=(EditText)findViewById(R.id.login);
02  String loginText=login.getText().toString();
```

下面给出一个关于编辑框的实例。

例 4.2 手机 QQ 空间写说说界面

在 Android Studio 中创建 Module，名称为“QQ Zone”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml。将默认添加的布局管理器修改为垂直的线性布局管理器，并删除默认添加的文本框组件，然后再将所需要的图片复制到 mipmap-mdpi 文件夹中，修改后的代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:background="#EAEAEA"
tools:context="com.mingrisoft.MainActivity" >
</LinearLayout>
```

（2）在线性布局管理器中，添加一个编辑框组件用于输入说说内容。设置其 android:inputType 属性值为 textMultiLine，表示该编辑框为多行编辑框，显示提示文本为“说点什么吧 ...”，并设置其为顶对齐、白色背景、内边框为 5dp、底外边距为 10dp，具体代码如下：

```
<!-- 添加写说说编辑框 -->
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:lines="6"
    android:hint="说点什么吧..."
    android:padding="5dp"
    android:background="#FFFFFF"
    android:gravity="top"
    android:layout_marginBottom="10dp"
    android:inputType="textMultiLine" >
</EditText>
```

（3）添加一个用于设置添加照片栏目的文本框组件。设置该文本框的显示文本为“添加照片”、在起始位置绘制一个图标、图标与文字的间距为 8dp、垂直居中对齐、白色背景、文字颜色为灰色，具体代码如下：

```
<!-- 设置添加照片栏目 -->
<TextView
android:id="@+id/textView1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:drawableLeft="@mipmap/addpicture"
android:text="添加照片"
android:drawablePadding="8dp"
android:gravity="center_vertical"
android:padding="8dp"
android:background="#FFFFFF"
android:textColor="#767676" />
```

（4）添加一个用于设置底部分享的文本框组件，该文本框只绘制一个图像就可以。代码如下：

```
<!-- 设置底部分享栏目 -->
```

```
02  <TextView
03  android:id="@+id/textView2"
04  android:layout_width="match_parent"
05  android:layout_height="wrap_content"
06  android:drawableLeft="@mipmap/bottom" />
```

说明 实际上，在屏幕中添加图片还可以使用图像组件（ImageView）来实现，将在后面的章节中进行介绍。

（5）运行本实例，将显示如图 4.9 所示的运行结果。

图 4.9 布局手机 QQ 空间写说说页面

4.2 按钮类组件（初级）

在 Android 中，提供了一些按钮类的组件，主要包括普通按钮、图片按钮、单选按钮和复选框等。其中，普通按钮使用 Button 类表示，用于触发一个指定的事件；图片按钮使用 ImageButton 类表示，也用于触发一个指定的事件，只不过该按钮将以图像来表现；单选按钮使用 RadioButton 类表示；复选框使用 CheckBox 类表示。这两个组件最大的区别是一组 RadioButton 中，只能有一个被选中，而一组 CheckBox 中，则可以同时选中多个。按钮类组件的继承关系如图 4.10 所示。

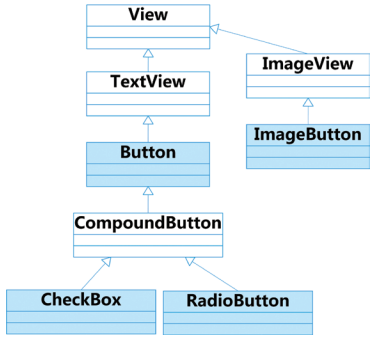


图 4.10 按钮类组件继承关系图

从图 4.10 中可以看出：Button 组件继承自 TextView，而 ImageButton 组件继承自 ImageView 组件，所以这两个组件在添加上是不同的，但是作用是一样的，都可以触发一个事件；RadioButton 和 CheckBox 都间接继承自 Button，都可以直接使用 Button 支持的属性和方法，所不同的是它们都比 Button 多了可选中的功能。下面将对 4 个按钮类组件分别进行介绍。

4.2.1 普通按钮

在 Android 手机应用中，按钮应用十分广泛。例如，QQ 登录方界面中的登录按钮（如图 4.11 所示），以及微信界面中的登录按钮（如图 4.12 所示），都应用了普通按钮。

图 4.11 QQ 登录按钮

图 4.12 微信登录按钮

通过 <Button> 标记在 XML 布局文件中添加普通按钮的基本格式如下：

```
<Button
android:id="@+id/ID号"
android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="显示文本"
    >
</Button>
```

说明 由于Button是TextView的子类，所以TextView支持的属性Button都是支持的。

例如：在屏幕中添加一个“开始游戏”按钮，代码如下：

```
01  <Button
02  android:id="@+id/start"
03  android:layout_width="wrap_content"
```

```
04  android:layout_height="wrap_content"
05  android:text="开始游戏" />
```

在模拟器中运行上面这段代码，将显示如图 4.13 所示的运行结果。

图 4.13　添加一个“开始游戏”按钮

在屏幕上添加按钮后，还需要为按钮添加单击事件监听器，才能让按钮发挥其特有的用途。Android 提供了两种为按钮添加单击事件监听器的方法，一种是在 Java 代码中完成，例如，在 Activity 的 onCreate() 方法中添加如下代码：

```
01  import android.view.View.OnClickListener;
02  import android.widget.Button;
03  Button login=(Button)findViewById(R.id.login);    //通过ID获取布局文件中添加的按钮
04  login.setOnClickListener(new View.OnClickListener() { //为按钮添加单击事件监听器
05      @Override
06      public void onClick(View v) {
07          //编写要执行的动作代码
08      }
09  });
```

说明 监听器类似于安防系统中安装的红外线报警器。安装了红外线报警器后，当有物体阻断红外线光束时，就会自动报警。同理，当我们为组件设置监听器后，如果有动作触发该监听器，那么就执行监听器中编写的代码。例如，为按钮设置一个单击事件监听器，那么单击这个按钮时，就会触发这个监听器，从而执行一些操作（如弹出一个对话框）。

另一种是在 Activity 中编写一个包含 View 类型参数的方法，并且将要触发的动作代码放在该方法中，然后在布局文件中，通过 android:onClick 属性指定对应的方法名实现。例如，在 Activity 中编写一个名为 myClick() 的方法，关键代码如下：

```
01  public void myClick(View view){
02      //此处编写要执行的动作代码
03  }
```

那么就可以在布局文件中通过 android:onClick=”myClick”语句为按钮添加单击事件监听器。

下面将通过一个实例来介绍如何添加普通按钮，并为按钮添加单击事件监听器。

例 4.3　模拟微信登录按钮

在 Android Studio 中创建 Module，名称为“Login Button”。在该 Module 中实现本实例，具体步骤如下。

（1）在 res/drawable 节点上单击鼠标右键，在弹出的快捷菜单中选中 New → Drawable Resource File 菜单项，在打开的新建资源文件对话框中，输入文件名称“shape”，单击 OK 按钮，创建 Shape

资源文件，然后删除默认生成的源代码，在该资源文件中绘制圆角矩形，并设置文字与按钮边界的间距，关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <!-- 设置填充的颜色 -->
    <solid android:color="#1AAD19" />
    <!-- 设置4个角的弧形半径 -->
    <corners android:radius="5dp" />
    <!-- 设置文字与按钮边界的间隔 -->
    <padding
        android:bottom="10dp"
        android:left="15dp"
        android:right="15dp"
        android:top="10dp" />
</shape>
```

（2）在默认生成的布局文件 activity_main.xml 中，将默认添加的布局管理器修改为相对布局管理器，然后添加 Button 普通按钮控件并设置按钮的背景资源 shape，最后为按钮设置单击事件的方法名称，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/loginbutton_bg"
    tools:context="com.mingrisoft.loginbutton.MainActivity">
<!--登录按钮-->
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="180dp"
        android:background="@drawable/shape"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:onClick="onLogin"
        android:text="登录"
        android:textColor="#FFFFFF" />
</RelativeLayout>
```

（3）打开 MainActivity 类，该类继承 Activity，然后创建 onLogin() 方法，实现登录按钮的单击事件，代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //全屏
07          getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN ,
08                  WindowManager.LayoutParams. FLAG_FULLSCREEN);
09      }
10      //登录按钮的单击事件
11      public void onLogin(View view) {
12          Toast.makeText(this,
13                  "您单击了登录按钮！", Toast.LENGTH_SHORT).show();
14      }
15  }
```

（4）运行本实例，再单击右侧的▶按钮，运行效果如图 4.14 所示，单击“登录”按钮，将显示如图 4.15 所示的消息提示框。

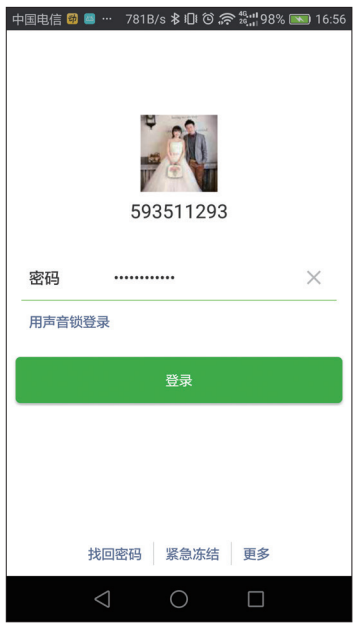


图 4.14　微信登录按钮

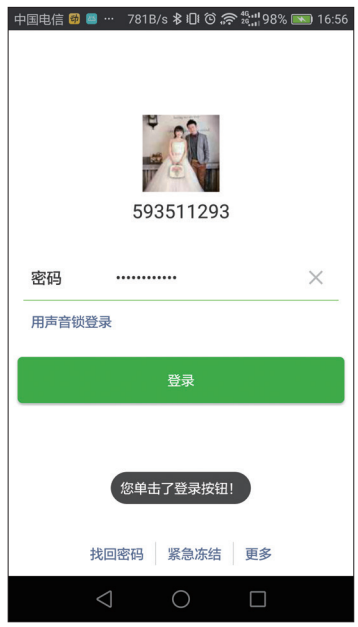


图 4.15　消息提示框

4.2.2 图片按钮

在 Android 手机应用中，图片按钮应用也很常见。例如，开心消消乐游戏的开始游戏界面中的“开始游戏”和“切换账号”按钮（如图 4.16 所示），以及全民飞机大战游戏的破纪录页面中的“确定”“炫耀一下”和查看奖励明细的按钮（如图 4.17 所示），都应用了图片按钮。

图 4.16　开心消消乐的开始游戏界面

图 4.17　全民飞机大战的破纪录页面

图片按钮与普通按钮的使用方法基本相同，只不过图片按钮使用 <ImageButton> 标记定义，并且可以为其指定 android:src 属性，用于设置要显示的图片。在布局文件中添加图片按钮的基本语法格式如下：

```
<ImageButton
android:id="@+id/ID号"
android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:src="@mipmap/图片文件名"
android:scaleType="缩放方式"
    >
</ImageButton>
```

重要属性说明：

☑ android:src 属性：用于指定按钮上显示的图片。

☑ android:scaleType 属性：用于指定图片的缩放方式。其属性值如表 4.2 所示。

表 4.2　android:scaleType 属性的属性值说明

属　性　值	描　　述
matrix	使用 matrix 方式进行缩放
fitXY	对图片横向、纵向独立缩放，使得该图片完全适应于该 ImageButton，图片的纵横比可能会改变
fitStart	保持纵横比缩放图片，直到该图片能完全显示在 ImageButton 中，缩放完成后该图片放在 ImageButton 的左上角
fitCenter	保持纵横比缩放图片，直到该图片能完全显示在 ImageButton 中，缩放完成后该图片放在 ImageButton 的中间

续表

属 性 值	描 述
fitEnd	保持纵横比缩放图片，直到该图片能完全显示在 ImageButton 中，缩放完成后该图片放在 ImageButton 的右下角
center	把图像放在 ImageButton 的中间，但不进行任何缩放
centerCrop	保持纵横比缩放图片，使图片能完全覆盖 ImageButton
centerInside	保持纵横比缩放图片，使 ImageButton 能完全显示该图片

例如，在屏幕中添加一个代表播放的图片按钮，代码如下：

```
<ImageButton
android:id="@+id/play"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@mipmap/play"
    >
</ImageButton>
```

在模拟器中运行上面这段代码，将显示如图 4.18 所示的运行结果。

图 4.18 添加一个播放按钮

说明 如果在添加图片按钮时，不为其设置android:background属性，那么作为按钮的图片将显示在一个灰色的按钮上，也就是说所添加的图片按钮，将带有一个灰色立体的边框。不过这时的图片按钮，将会随着用户的动作而改变。一旦为其设置了android:background属性后，它将不会随着用户的动作而改变。如果要让其随着用户的动作而改变，就需要使用StateListDrawable资源来对其进行设置。

同普通按钮一样，也需要为图片按钮添加单击事件监听器，具体添加方法同普通按钮，这里不再赘述。下面通过一个实例来演示图片按钮的使用。

例 4.4 开心消消乐的开始游戏和切换账号按钮

在 Android Studio 中创建 Module，名称为“Start Game Button”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直的线性布局管理器，并将默认添加的文本框组件删除。

（2）设置线性布局管理器的背景为复制到 mipmap 目录下的开心消消乐的背景图片，并设置底部居中对齐，以及底边距为 20dp，关键代码如下：

```
android:background="@mipmap/bg"
android:gravity="bottom|center_horizontal"
android:paddingBottom="20dp"
```

（3）在线性布局管理器中，添加两个图片按钮，分别为开始游戏按钮和切换账号按钮。主要是通过 android:src 属性设置图片，以及通过 android:background 属性设置背景透明，具体代码如下：

```
<!--开始游戏图片按钮-->
<ImageButton
    android:id="@+id/start"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#0000"
    android:src="@mipmap/bt_start"
    />
<!--切换账号图片按钮-->
<ImageButton
    android:id="@+id/switch1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#0000"
    android:src="@mipmap/bt_switch"
    android:layout_marginTop="10dp"
    />
```

注意 在设置组件ID时，不能使用Java关键字。例如上面代码中，就不能把ID属性值设置为@+id/switch，否则将出现资源名不能使用Java关键字（Resource name cannot be a Java keyword …）的错误。

（4）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中添加设置全屏的代码。修改后的具体代码如下：

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);        //设置全屏显示
    }
}
```

（5）在主活动 MainActivity 的 onCreate() 方法中，为“开始游戏”图片按钮添加单击事件监听器，在重写的 onClick() 方法中弹出相应的消息提示框，关键代码如下：

```
ImageButton st= (ImageButton) findViewById(R.id.start);//通过ID获取布局开始游戏图片按钮
//为开始游戏图片按钮添加单击事件监听器
st.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this,
```

```
07                  "您单击了开始游戏按钮",Toast.LENGTH_SHORT).show();
08          }
09  });
```

（6）按照步骤（5）的方法再为“切换账号”图片按钮添加单击事件监听器，具体代码请参见光盘。

（7）运行效果如图 4.19 所示，单击“开始游戏”按钮，将显示如图 4.20 所示的消息提示框。

图 4.19　开心消消乐的开始游戏和切换账号按钮

图 4.20　单击“开始游戏”按钮显示的消息提示框

4.4 图像类组件

在 Android 中，提供了比较丰富的图像类组件。用于显示图像的组件称为图像视图组件，用 ImageView 表示；用于按照行、列的方式来显示多个元素（如图片、文字等）的组件称为网格视图，用 GridView 表示。它们的继承关系如图 4.21 所示。

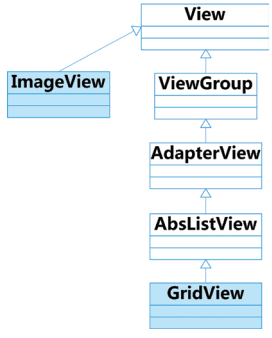


图 4.21　图像类组件继承关系图

从图 4.21 中可以看出：ImageView 组件继承自 View，它主要用于呈现图像；GridView 组件间接继承自 AdapterView 组件，可以包括多个列表项，并且可以通过合适的方式显示。下面将对这三个组件分别进行介绍。

说明　AdapterView是一个抽象基类，它继承自ViewGroup，属于容器，可以包括多个列表项，并且可以通过合适的方式显示。在指定多个列表项时，使用Adapter对象提供。

4.4.1 图像视图

图像视图（ImageView）用于在屏幕中显示任何 Drawable 对象，通常用来显示图片。例如，美图秀秀的美化图片界面中显示的图片（如图 4.22 所示），以及有道词典的主界面中的图片（如图 4.23 所示）。

图 4.22　美图秀秀美化图片界面

图 4.23　有道词典的主界面

图 4.22　美图秀秀美化图片界面

图 4.23　有道词典的主界面

说明 在使用ImageView组件显示图像时，通常需要将要显示的图片放置在res\drawable或者res\mipmap目录中。

在布局文件中添加图像视图，可以使用 <ImageView> 标记来实现，具体的语法格式如下：

```
<ImageView
    属性列表
    >
</ImageView>
```

ImageView 组件支持的常用 XML 属性如表 4.3 所示。

表 4.3　ImageView 组件支持的 XML 属性

XML 属性	描　述
android:adjustViewBounds	用于设置 ImageView 组件是否调整自己的边界来保持所显示图片的长宽比
android:maxHeight	设置 ImageView 组件的最大高度，需要设置 android:adjustViewBounds 属性值为 true，否则不起作用
android:maxWidth	设置 ImageView 组件的最大宽度，需要设置 android:adjustViewBounds 属性值为 true，否则不起作用

续表

XML 属性	描　述
android:scaleType	用于设置所显示的图片如何缩放或移动以适应 ImageView 的大小，其属性值可以是： matrix（使用 matrix 方式进行缩放）、fitXY（对图片横向、纵向独立缩放，使得该图片完全适应于该 ImageView，图片的纵横比可能会改变） fitStart（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的左上角） fitCenter（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的中间） fitEnd（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的右下角） center（把图像放在 ImageView 的中间，但不进行任何缩放） centerCrop（保持纵横比缩放图片，以使得图片能完全覆盖 ImageView）或 centerInside（保持纵横比缩放图片，以使得 ImageView 能完全显示该图片）
android:src	用于设置 ImageView 所显示的 Drawable 对象的 ID，例如，设置显示保存在 res/drawable 目录下的名称为 flower.jpg 的图片，可以将属性值设置为 android:src=”@drawable/flower”
android:tint	用于为图片着色，其属性值可以是 #rgb、#argb、#rrggbb 或 #aarrggbb 表示的颜色值

说明 在表4.3中，只给出了ImageView组件常用的部分属性，关于该组件的其他属性，可以参阅Android官方提供的API文档。

下面编写一个关于 ImageView 组件的实例。

例 4.5 单击 ImageView 更换显示的图像

在 Android Studio 中创建 Module，名称为“ImageView”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并将默认添加的 TextView 组件删除，然后在布局管理器中添加一个 ImageView 组件并设置默认显示的图片与单击事件的方法，修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.imageview.MainActivity">
    <!--图像视图控件-->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:onClick="onImageView"
        android:layout_margin="50dp"
        android:background="@drawable/bg2" />

</RelativeLayout>
```

（2）打开 MainActivity 类，首先定义图像视图控件与图像标记，然后在 onCreate() 方法中获取图像视图控件，最后创建 onImageView() 方法，用于实现更换图像，代码如下：

```
public class MainActivity extends AppCompatActivity {
    private ImageView imageView;                          //图像视图控件
    private int i = 2;                                    //设置图像标记
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView = findViewById(R.id.imageView);         //获取图像视图控件
    }
    //图像视图控件的单击事件
    public void onImageView(View view) {
        if (i == 2) {
            imageView.setBackgroundResource(R.drawable.bg1);//更换图片
            i=1;                                          //更改图像标记
        } else {
            imageView.setBackgroundResource(R.drawable.bg2);
            i=2;                                          //更改图像标记
        }
    }
}
```

（3）运行本实例，将显示如图 4.24 所示和图 4.25 所示的界面效果。

图 4.24　默认显示的图像

图 4.25　更换显示的图像

4.4.2 网格视图

网格视图（GridView）是按照行、列分布的方式来显示多个组件，通常用于显示图片或图标等。例如，QQ 相册相片预览界面（如图 4.26 所示），以及口袋购物浏览商品界面（如图 4.27 所示），都应用了网格视图。

图 4.26 QQ 相册相片预览界面

图 4.27 口袋购物浏览商品界面

在使用网格视图时，需要在屏幕上添加 GridView 组件，通常在 XML 布局文件中使用 <GridView> 标记实现，其基本语法如下：

```
<GridView
    属性列表
    >
</GridView>
```

GridView 组件支持的 XML 属性如表 4.4 所示。

表 4.4 GridView 组件支持的 XML 属性

XML 属性	描　述
android:columnWidth	用于设置列的宽度
android:gravity	用于设置对齐方式
android:horizontalSpacing	用于设置各元素之间的水平间距
android:numColumns	用于设置列数，其属性值通常为大于 1 的值，如果只有 1 列，那么最好使用 ListView 实现
android:stretchMode	用于设置拉伸模式，其中属性值可以是 none（不拉伸）、spacingWidth（仅拉伸元素之间的间距）、columnWidth（仅拉伸表格元素本身）或 spacingWidthUniform（表格元素本身、元素之间的间距一起拉伸）
android:verticalSpacing	用于设置各元素之间的垂直间距

在使用 GridView 组件时，通常使用 Adapter 类为 GridView 组件提供数据。

Apapter 类是一个接口，代表适配器对象。它是组件与数据之间的桥梁，通过它可以处理数据并将其绑定到相应的组件上。它的常用实现类包括以下几个。

☑ ArrayAdapter：数组适配器，通常用于将数组的多个值包装成多个列表项，只能显示一行文字。

☑ SmipleAdapter：简单适配器，通常用于将 List 集合的多个值包装成多个列表项。可以自定义各种效果，功能强大。

☑ SmipleCursorAdapter：与 SmipleAdapter 类似，只不过它需将 Cursor（数据库的游标对象）的字段与组件 ID 对应，从而实现将数据库的内容以列表形式展示出来。

☑ BaseAdapter：是一个抽象类，继承它需要实现较多的方法，通常它可以对各列表项进行最大限度的定制，也具有很高的灵活性。

下面通过一个具体的实例演示如何通过 BaseAdapter 适配器指定内容的方式创建 GridView。

例 4.6 手机 QQ 相册界面

在 Android Studio 中创建 Module，名称为“QQ Album”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，然后在 TextView 组件上面添加一个 ImageView，最后在 ImageView 下面添加 ID 为 gridView 的 GridView 组件，并设置其列数为自动排列。修改后的代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.mingrisoft.MainActivity">
<!--标题栏-->
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@mipmap/qqxiang" />
<!--年月日-->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:paddingTop="10dp"
        android:text="2016年1月19号" />
<!--网格布局-->
    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:columnWidth="100dp"
        android:gravity="center"
        android:numColumns="auto_fit"
```

```
        android:stretchMode="columnWidth"
        android:verticalSpacing="5dp">
    </GridView>
</LinearLayout>
```

（2）打开主活动 MainActivity，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在该类中创建一个用于保存图片资源 ID 的数组。修改后的关键代码如下：

```
public class MainActivity extends Activity {
    //显示的图片资源ID的数组
    private Integer[] picture = {R.mipmap.img01, R.mipmap.img02, R.mipmap.img03,
            R.mipmap.img04, R.mipmap.img05, R.mipmap.img06, R.mipmap.img07,
            R.mipmap.img08, R.mipmap.img09, R.mipmap.img10, R.mipmap.img11,
            R.mipmap.img12,};
}
```

（3）在 MainActivity 中，创建一个 ImageAdapter 图片适配器，在该适配器中，首先创建一个新的 ImageView，然后将图片通过适配器加载到新的 ImageView 中，具体代码如下：

```
//创建ImageAdapter
public class ImageAdapter extends BaseAdapter {
    private Context mContext;                    //获取上下文
    public ImageAdapter(Context c){
        mContext=c;
    }
    @Override
    public int getCount() {
        return picture.length;                   //图片数组的长度
    }
    @Override
    public Object getItem(int position) {
        return null;
    }
    @Override
    public long getItemId(int position) {
        return 0;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if(convertView==null){                   //判断传过来的值是否为空
            imageView=new ImageView(mContext);//创建ImageView组件
            imageView.setLayoutParams(new GridView.LayoutParams(100, 90));//为组件设置宽高
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP); //选择图片铺设方式
        }else{
            imageView= (ImageView) convertView;
        }
```

```
29          imageView.setImageResource(picture[position]);//将获取图片放到ImageView组件中
30          return imageView;                              //返回ImageView
31      }
32  }
```

（4）在主活动的 onCreate() 方法中，获取布局文件中添加的 GridView 组件，并且为其设置适配器，关键代码如下：

```
01  GridView gridView= (GridView) findViewById(R.id.gridView);//获取布局文件中的GridView组件
02  gridView.setAdapter(new ImageAdapter(this));           //调用ImageAdapter
```

（5）打开 AndroidManifest.xml 文件，将其中的 <application> 标记的 android:theme 属性值“@style/AppTheme”修改为“@style/Theme.AppCompat.Light.DarkActionBar”，修改后的 android:theme 属性的代码如下：

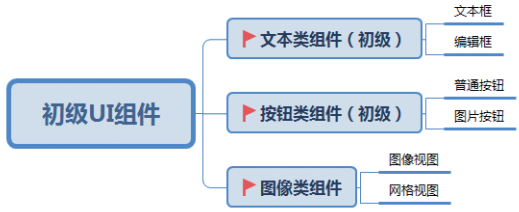
```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

（6）运行本实例，将显示如图 4.28 所示的界面。

图 4.28　通过 GridView 实现手机 QQ 相册页面

4.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 5 章

中级 UI 组件

在第 4 章当中已经介绍了初级 UI 组件的用法以及组件经常出现在 App 的哪些位置。本章将继续学习 Android 开发中的用户界面设计，主要涉及一些常用的中级 UI 组件、自动完成文本框和文本切换器等，通过这些组件，可以开发出更加优秀的用户界面。

5.1 文本类组件（中级）

5.1.1 自动完成文本框

在百度上面进行关键词搜索的时候，输入需要搜索的关键字就会显示一个下拉列表，列表中会匹配到用户输入的关键词汇，效果如图 5.1 所示。Android 也为开发者提供了一个这样的组件，那就是 AutoCompleteTextView 组件也叫做动完成文本框。AutoCompleteTextView 是 EditText 的子类，从外表上看它就是一个普通的编辑框组件，其实它内在的功能就是在用户输入一定的字符时，该组件会显示一个下拉列表。在这个列表中用户单击需要的字符，该字符将自动填写在编辑框当中。

图 5.1　百度关键字提示列表

通过 <AutoCompleteTextView> 在 XML 布局文件中添加自动完成文本框的基本语法格式如下：

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

自动完成文本框（AutoCompleteTextView）所支持的 XML 属性如表 5.1 所示。

表 5.1　AutoCompleteTextView 支持的 XML 属性

XML 属性	描　　述
android:completionHint	设置下拉菜单中显示的提示
android:completionHintView	设置下拉菜单中显示的提示视图
android:completionThreshold	设置用户至少输入几个字符才显示提示列表
android:dropDownAnchor	设置提示列表显示在某个组件的下面，值为某组件的 id
android:dropDownHeight	设置下拉列表的高度
android:dropDownHorizontalOffset	设置下拉列表与文本框之间的水平偏移，下拉菜单默认与文本框左对齐
android:dropDownSelector	设置下拉列表选择器的背景
android:dropDownVerticalOffset	设置下拉列表与文本框之间的垂直偏移，下拉菜单默认在文本框下面
android:dropDownWidth	设置下拉列表的宽度
android:popupBackground	设置下拉列表的背景

AutoCompleteTextView 组件的用法其实很简单，只需要为它设置一个 Adapter 适配器，并在该适配器中添加 AutoCompleteTextView 提示列表中显示的文本信息即可。

下面通过一个具体的实例演示自动完成文本框（AutoCompleteTextView）的应用。

例 5.1　模拟淘宝搜索宝贝时的提示列表

在 Android Studio 中创建 Module，名称为“Auto Complete TextView”，实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后为布局管理器设置背景图片，最后添加 1 个 AutoCompleteTextView 组件用于显示淘宝搜索宝贝时的提示列表。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
06      android:layout_height="match_parent"
07      android:background="@mipmap/bg"
```

```
    tools:context="com.mingrisoft.MainActivity">
    <!--添加自动完成文本框组件-->
    <AutoCompleteTextView
        android:id="@+id/at"
        android:layout_width="150dp"
        android:layout_height="40dp"
        android:layout_centerHorizontal="true"
        android:background="@null"
        android:dropDownWidth="match_parent"/>
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后创建 1 个名称为 book 并且是 String 类型的数组，该数组用于显示列表中的书名。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    //创建列表显示文字的数组
    private static final String[] book = new String[] {
            "Android入门到精通", "Android开发实战",
            "Android范例宝典", "Android精彩编程200例",
            "Android项目开发|实战入门"
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

（3）在主活动的 onCreate() 方法中，首先添加设置当前 Activity 的全屏代码，然后获取布局文件中的 AutoCompleteTextView 组件，再创建 1 个数组适配器，最后为 AutoCompleteTextView 组件设置数组适配器。关键代码如下：

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置全屏显示
//获取自动完成文本框组件
AutoCompleteTextView autoCompleteTextView = (AutoCompleteTextView)
        findViewById(R.id.at);
//创建数组适配器
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, book);
autoCompleteTextView.setAdapter(adapter);            //设置数组适配器
```

（4）运行本实例，将显示如图 5.2 所示的界面。输入关键字“An”将显示如图 5.3 所示的界面。

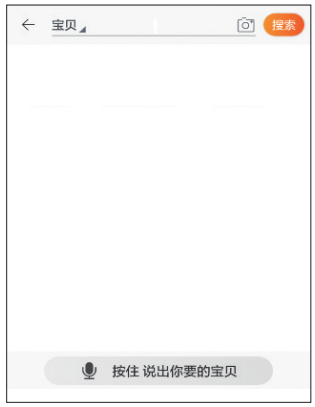


图 5.2　宝贝搜索界面

图 5.3　显示关键字对应的提示列表

5.1.2 文本切换器

TextSwitcher 是 ViewSwitcher 的子类，它与 ImageSwitcher 很相似，都需要设置一个 ViewFactory，并且都可以设置切换时所显示的动画效果。不同的是 TextSwitcher 在实现 ViewFactory 接口中的 makeView() 方法时，需要返回一个 TextView 组件。文本切换器可以用来实现一个通过垂直滚动的动画显示唐诗中的文字，如图 5.4 所示。

图 5.4　显示垂直滚动的文字

通过 <TextSwitcher> 在 XML 布局文件中添加状态开关的基本语法格式如下：

```
<TextSwitcher
    android:id="@+id/textSwitcher"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

文本切换器所支持的 XML 属性如表 5.2 所示。

表 5.2　文本切换器支持的 XML 属性

XML 属性	描　述
android:inAnimation	设置文本进入时的动画文件
android:outAnimation	设置文本退出时的动画文件

下面通过一个具体的实例演示文本切换器的具体应用。

例 5.2 单击屏幕滚动切换每句唐诗

在 Android Studio 中创建 Module，名称为“Text Switcher”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后为布局管理器设置背景图片，最后添加一个 TextSwitcher 组件用于实现切换每句唐诗。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg"
    tools:context="com.mingrisoft.MainActivity">
    <!--定义文本切换器并设置文本切换时的动画效果-->
    <TextSwitcher
        android:id="@+id/textSwitcher"
        android:layout_width="260dp"
        android:layout_height="100dp"
        android:layout_marginTop="50dp"
        android:layout_marginLeft="50dp"
        android:inAnimation="@anim/in_animation"
        android:outAnimation="@anim/out_animation">
    </TextSwitcher>
</RelativeLayout>
```

说明 在XML文件中设置TextSwitcher组件时，需要指定文字切换时进入与退出的动画文件，如代码中的“anim/in_animation”文字切换时的进入动画文件，“anim/out_animation”文字切换时的退出动画文件。这两个动画文件需要手动创建，具体的创建方式会在第16图形图像处理技术中进行介绍，完成本实例可以在云盘的源码中进行拷贝。

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义所需的成员变量与用于显示唐诗的数组，最后在 onCreate() 方法中添加设置当前 Activity 全屏的代码。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    private TextSwitcher textSwitcher;      //定义文本切换器
    private int index;                      //定义用于计算诗句数组下标
    String[] poetry = new String[]{         //定义唐诗数组
            "望庐山瀑布",
            "[作者] 李白",
            "日照香炉生紫烟，",
```

```
            "遥看瀑布挂前川。",
            "飞流直下三千尺，",
            "疑是银河落九天。"
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置全屏显示
    }
}
```

（3）在主活动的 onCreate() 方法中，首先获取布局文件中的 TextSwitcher 组件，然后为其设置 1 个 ViewFactory，并重写 makeView() 方法，最后在重写的 makeView() 方法中创建 1 个用于显示唐诗文字的 TextView。关键代码如下：

```
//获取文本切换器组件
textSwitcher = (TextSwitcher) findViewById(R.id.textSwitcher);
//为文本切换器设置Factory
textSwitcher.setFactory(new ViewSwitcher.ViewFactory() {
    @Override
    public View makeView() {
        //创建文本框组件用于显示唐诗文字
        TextView textView = new TextView(MainActivity.this);
        textView.setTextSize(30);               //设置文字大小
        textView.setTextColor(Color.BLACK);     //设置文字颜色为黑色
        return textView;
    }
});
```

（4）重写 onTouchEvent() 方法，在该方法中实现单击手机或模拟器屏幕时切换显示的唐诗文字。具体代码如下：

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    //判断手指单击屏幕时
    if (event.getAction()==MotionEvent.ACTION_DOWN){
        //切换显示的诗句文字
        textSwitcher.setText(poetry[index++%poetry.length]);
    }
    return super.onTouchEvent(event);
}
```

（5）运行本实例，将显示如图 5.5 所示的背景图片。单击手机或模拟器屏幕，唐诗数组中的第一句文字将从箭头起始位置向上滚动并显示如图 5.6 所示。

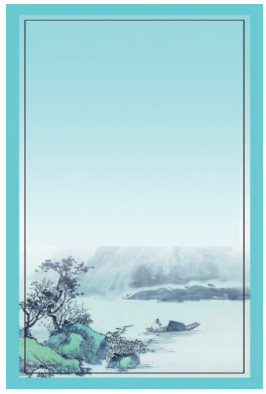


图 5.5　显示背景图片

图 5.6　滚动显示唐诗中的文字

5.2 按钮类组件（中级）

5.2.1 状态开关按钮

状态开关按钮（Switch）也是由 Button 派生出来的，所以在本质上它也算是一个比较高级的按钮，用户可以来回拖动“拇指”控制该按钮的开启与关闭或者只需轻按，就像选择复选框一样来开启该组件。在 Android 中，该组件经常被使用到 App 的设置界面当中。例如，酷狗音乐的设置界面中的功能开关（如图 5.7 所示），以及计步器 App 设置界面中的步数通知开关（如图 5.8 所示）。

图 5.7　酷狗音乐设置界面

图 5.8　计步器设置界面

通过 <Switch> 在 XML 布局文件中添加状态开关的基本语法格式如下：

```
<Switch
    android:id="@+id/ID号"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

状态开关（Switch）所支持的 XML 属性如表 5.3 所示。

表 5.3　Switch 支持的 XML 属性

XML 属性	描　　述
android:switchMinWidth	设置开关组件的最小宽度
android:switchPadding	设置开关组件与左侧标题文本之间的距离
android:switchTextAppearance	设置开关组件标题文本的样式
android:textOff	设置开关组件关闭时显示的文本
android:textOn	设置开关组件开启时显示的文本
android:showText	是否显示该组件打开或关闭时所显示的文本
android:textStyle	设置开关组件的文本风格
android:thumb	指定该组件切换开关的图标
android:track	指定该组件开关滑动的轨道

由于状态开关在使用时无法确定当前开关处于开启或者是关闭的状态。所以需要为该组件设置一个事件监听器。例如，为名称 swich1 的状态开关添加状态改变事件监听器，可以使用下面的代码：

```
01  final Switch swich1 = (Switch) findViewById(R.id.swich1);//根据id属性获取状态开关
02  swich1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
03      @Override
04      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
05          if (swich1.isChecked()) {                          //判断状态开关是否被选中
06              Toast.makeText(MainActivity.this, "开启", Toast.LENGTH_SHORT).show();
07          } else {
08              Toast.makeText(MainActivity.this, "关闭", Toast.LENGTH_SHORT).show();
09          }
10      }
11  });
```

下面通过一个具体的实例演示状态开关（Switch）的具体应用。

例 5.3　模拟微信通用界面中开启横屏模式的开关

在 Android Studio 中创建 Module，名称为“Status Switch”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后为布局管理器设置背景图片，最后添加一个 Switch 组件用于实现开启横屏模式的开关。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:background="@mipmap/bg1"
    tools:context="com.mingrisoft.MainActivity">
<!--状态开关-->
<Switch
    android:id="@+id/sw"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="75dp"
    android:text="开启横屏模式"
    android:textColor="#000000"
    android:switchPadding="170dp"
    android:layout_marginLeft="13dp"/>
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中添加设置当前 Activity 全屏的代码。修改后的具体代码如下：

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置全屏显示
    }
}
```

（3）在主活动的 onCreate() 方法中，首先获取布局文件中的 switch 开关组件，然后为该组件设置状态改变监听器，最后在监听器中实现显示开关状态的提示功能。关键代码如下：

```
final Switch switch1= (Switch) findViewById(R.id.sw); //获取布局文件中switch开关组件
//设置开关状态改变监听器
switch1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
     if (switch1.isChecked()){                          //判断状态开关是否被选中
       Toast.makeText(MainActivity.this, "开启横屏模式！", Toast.LENGTH_SHORT).show();
     }else {
       Toast.makeText(MainActivity.this, "关闭横屏模式！", Toast.LENGTH_SHORT).show();
     }
    }
});
```

（4）运行本实例，将显示如图 5.9 所示。滑动或单击“开启横屏模式”开关将显示如图 5.10 所示。

图 5.9　微信通用界面

图 5.10　显示开启横屏模式提示

5.2.2 单选按钮

在默认情况下，单选按钮显示为一个圆形图标，并且在该图标旁边放置一些说明性文字。在程序中，一般将多个单选按钮放置在按钮组中，使这些单选按钮表现出某种功能，当用户选中某个单选按钮后，按钮组中的其他按钮将被自动取消选中状态。在 Android 手机应用中，单选按钮应用也十分广泛。例如，在使用陌陌社交工具注册新用户填写基本资料时，填写基本资料界面中的选择性别单选按钮（如图 5.11 所示），以及如图 5.12 所示的显示智力问答题的备选答案时的单选按钮。

图 5.11　陌陌的注册界面

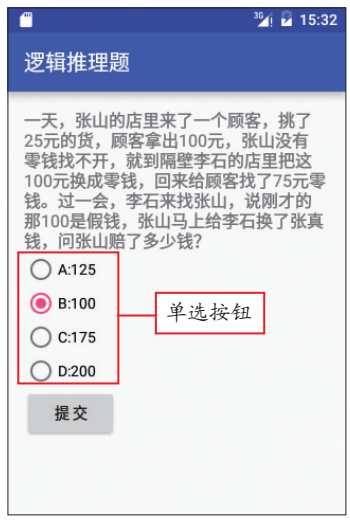


图 5.12　智力问答题的备选答案的单选按钮

通过 <RadioButton> 在 XML 布局文件中添加单选按钮的基本语法格式如下：

```
<RadioButton
android:text="显示文本"
android:id="@+id/ID号"
```

```
android:checked="true|false"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
    >
</RadioButton>
```

RadioButton 组件的 android:checked 属性用于指定选中状态，属性值为 true 时，表示选中；属性值为 false 时，表示取消选中，默认为 false。

通常情况下，RadioButton 组件需要与 RadioGroup 组件一起使用，组成一个单选按钮组。在 XML 布局文件中，添加 RadioGroup 组件的基本格式如下：

```
<RadioGroup
android:id="@+id/radioGroup1"
android:orientation="horizontal"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
<!-- 添加多个RadioGroup组件 -->
</RadioGroup>
```

例如，在页面中添加一个选择性别的单选按钮组和一个“提交”按钮，可以使用下面的代码：

```
<RadioGroup
android:id="@+id/radioGroup1"
android:orientation="horizontal"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
<RadioButton
    android:layout_height="wrap_content"
    android:id="@+id/radio0"
    android:text="男"
    android:layout_width="wrap_content"
    android:checked="true"/>
<RadioButton
    android:layout_height="wrap_content"
    android:id="@+id/radio1"
    android:text="女"
    android:layout_width="wrap_content"/>
</RadioGroup>
<Button android:text="提交"
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
</Button>
```

在模拟器中运行上面这段代码，将显示如图 5.13 所示的运行结果。

图 5.13　添加一个单选按钮组

在屏幕中添加单选按钮组后，还需要获取单选按钮组中选中项的值，通常存在以下两种情况。

☑ 在改变单选按钮组的值时获取

在改变单选按钮组的值时获取选中的单选按钮的值，首先需要获取单选按钮组，然后为其添加 OnCheckedChangeListener，并在其 onCheckedChanged() 方法中根据参数 checkedId 获取被选中的单选按钮，并通过其 getText() 方法获取该单选按钮对应的值。例如，要获取 id 属性为 radioGroup1 的单选按钮组的值，可以通过下面的代码实现。

```
RadioGroup sex=(RadioGroup)findViewById(R.id.radioGroup1);
sex.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton r=(RadioButton)findViewById(checkedId);
        r.getText();          //获取被选中的单选按钮的值
    }
});
```

☑ 单击其他按钮时获取

单击其他按钮时获取选中项的值时，首先需要在该按钮的单击事件监听器的 onClick() 方法中，通过 for 循环语句遍历当前单选按钮组，并根据被遍历到的单选按钮的 isChecked() 方法判断该按钮是否被选中，当被选中时，通过单选按钮的 getText() 方法获取对应的值。例如，要在单击“提交”按钮时，获取 id 属性为 radioGroup1 的单选按钮组的值，可以通过下面的代码实现。

```
final RadioGroup sex=(RadioGroup)findViewById(R.id.radioGroup1);
Button button=(Button)findViewById(R.id.button1); //获取一个提交按钮
button.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        for(int i=0;i<sex.getChildCount();i++){
            RadioButton r=(RadioButton)sex.getChildAt(i); //根据索引值获取单选按钮
            if(r.isChecked()){                    //判断单选按钮是否被选中
                r.getText();                      //获取被选中的单选按钮的值
                break;                            //跳出for循环
            }
        }
    }
});
```

下面通过一个实例演示单选按钮的具体应用。

例 5.4 逻辑推理题

在 Android Studio 中创建 Module，名称为“Radio Button”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，并修改默认添加的 TextView 组件用于显示逻辑推理题的文字内容，然后添加一个包含 4 个单选按钮的单选按钮组和一个提交按钮，关键代码如下：

```
<!--逻辑问题-->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="一天，张山的店里来了一个顾客，挑了25元的货，顾客拿出100元，
    张山没有零钱找不开，就到隔壁李石的店里把这100元换成零钱，回来给顾客找了75元零钱。
    过一会，李石来找张山，说刚才的那100是假钱，张山马上给李石换了张真钱，问张山赔了多少钱？"
    android:textSize="16sp"/>
<!--单选按钮组-->
<RadioGroup
    android:id="@+id/rg"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <!--单选按钮A-->
    <RadioButton
        android:id="@+id/rb_a"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="A:125" />
    <!--单选按钮B-->
    <RadioButton
        android:id="@+id/rb_b"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B:100" />
    <!--单选按钮C-->
    <RadioButton
        android:id="@+id/rb_c"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="C:175" />
    <!--单选按钮D-->
    <RadioButton
        android:id="@+id/rb_d"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="D:200" />
</RadioGroup>
```

```
<!--提交按钮-->
<Button
    android:id="@+id/bt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="提 交" />
```

（2）在主活动 MainActivity 的 onCreate() 方法中，获取提交按钮并为其添加单击事件监听器，在按钮的单击事件监听器的 onClick() 方法中，通过 for 循环语句遍历当前单选按钮组，并根据被遍历到的单选按钮的 isChecked() 方法判断该按钮是否被选中，当被选中时，通过单选按钮的 getText() 方法获取对应的值，并且将获取的值与正确答案的值进行比较，如果相同，则提示“回答正确”，否则给出解析及正确答案。关键代码如下：

```
public class MainActivity extends AppCompatActivity {
 Button bt;                                          //定义提交按钮
 RadioGroup rg;                                      //定义单选按钮组
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 bt = (Button) findViewById(R.id.bt);                //通过ID获取布局中的提交按钮
 rg = (RadioGroup) findViewById(R.id.rg);            //通过ID获取布局中的单选按钮组
 bt.setOnClickListener(new View.OnClickListener() {  //为提交按钮设置单击事件监听器
     @Override
     public void onClick(View v) {
         for (int i = 0; i < rg.getChildCount(); i++) {
            //根据索引值获取单选按钮
            RadioButton radioButton = (RadioButton) rg.getChildAt(i);
             if (radioButton.isChecked()) {          //判断单选按钮是否被选中
                 if (radioButton.getText().equals("B:100")) {  //判断答案是否正确
                           Toast.makeText(MainActivity.this,
                                   "回答正确", Toast.LENGTH_LONG).show();
                       } else {
                  //错误消息提示框
                 AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                      builder.setMessage("回答错误，下面请看解析：当张山换完零钱之后，" +
                              "给了顾客75还有价值25元的商品，自己还剩下了25元。这时，" +
                              "李石来找张山要钱，张山把自己剩下的相当于是李石的25元给了李石，" +
                             "另外自己掏了75元。这样张山赔了一个25元的商品和75元的人民币，" +
                                     "总共价值100元。");
                          builder.setPositiveButton("确定", null).show();//单击确定消失
                      }
                      break;                         //跳出for循环
                  }
              }
          }
```

```
34            });
35        }
36    }
```

（3）运行本实例，将显示如图 5.14 所示的界面。

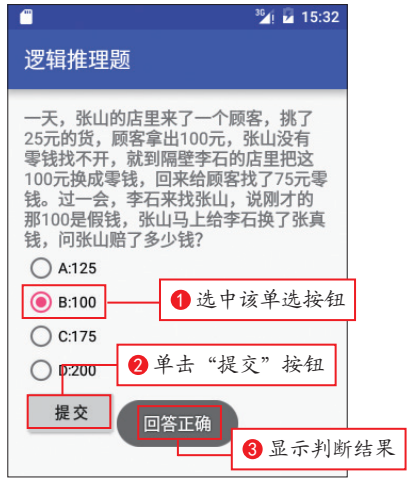


图 5.14　逻辑推理题的单选按钮组

5.2.3 复选框

在默认情况下，复选框显示为一个方块图标，并且在该图标旁边放置一些说明性文字。与单选按钮唯一不同的是，复选框可以进行多选设置，每一个复选框都提供“选中”和“不选中”两种状态。在 Android 手机应用中，复选框组件的应用也十分广泛。例如，在全民飞机大战游戏中，通过微信登录游戏时显示的授予权限界面（如图 5.15 所示），在该页面中将通过复选框显示已经授予的权限；亚马逊手机客户端的用户登录页面中的是否显示密码的复选框（如图 5.16 所示）。

图 5.15　全民飞机大战授予权限界面

图 5.16　亚马逊登录界面

通过 <CheckBox> 在 XML 布局文件中添加复选框的基本语法格式如下：

```
<CheckBox android:text="显示文本"
android:id="@+id/ID号"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
    >
</CheckBox>
```

由于使用复选框可以选中多项，所以为了确定用户是否选择了某一项，还需要为每一个选项添加事件监听器。例如，要为 id 为 like1 的复选框添加状态改变事件监听器，可以使用下面的代码：

```
01  final CheckBox like1=(CheckBox)findViewById(R.id.like1);          //根据id属性获取复选框
02  like1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
03
04      @Override
05      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
06          if(like1.isChecked()){                                    //判断该复选框是否被选中
07              like1.getText();                                      //获取选中项的值
08          }
09      }
10  });
```

下面通过一个实例来演示复选框的应用。

例 5.5 模拟 12306 车票预订

在 Android Studio 中创建 Module，名称为“Check Box”。实现本实例的骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后添加两个 CheckBox 复选框控件与 1 个 Button 按钮控件，关键代码如下：

```
01  <!--学生复选框-->
02  <CheckBox
03      android:id="@+id/checkbox1"
04      android:layout_width="wrap_content"
05      android:layout_height="wrap_content"
06      android:layout_alignEnd="@+id/checkbox2"
07      android:layout_alignParentBottom="true"
08      android:layout_alignRight="@+id/checkbox2"
09      android:layout_marginBottom="258dp"
10      android:layout_marginEnd="69dp"
11      android:layout_marginRight="69dp"
12      android:scaleX="0.9"
13      android:scaleY="0.9"
14      android:text="学生"
15      android:textColor="#000000" />
16  <!--车次复选框-->
```

```
<CheckBox
    android:id="@+id/checkbox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_alignTop="@+id/checkbox1"
    android:layout_marginEnd="22dp"
    android:layout_marginRight="22dp"
    android:layout_marginTop="51dp"
    android:scaleX="0.9"
    android:scaleY="0.9"
    android:text="兑换车次"
    android:textColor="#000000" />
<!--查询按钮-->
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/checkbox2"
    android:layout_marginLeft="15dp"
    android:layout_marginRight="15dp"
    android:layout_marginTop="25dp"
    android:background="@drawable/shape"
    android:onClick="onClick"
    android:text="查询"
    android:textColor="#FFFFFF"
    android:textSize="18sp" />
```

（2）在主活动 MainActivity 中，定义 2 个复选框并且在 onCreate() 方法中获取这 2 个复选框控件，代码如下：

```
CheckBox checkBox1, checkBox2;                                    //定义复选框
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //全屏
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
    checkBox1 = (CheckBox) findViewById(R.id.checkbox1);     //通过ID获取布局复选框1
    checkBox2 = (CheckBox) findViewById(R.id.checkbox2);     //通过ID获取布局复选框2
}
```

（3）创建 onClick() 方法，该方法是查询按钮的单击事件，在该方法中通过 if 语句获取被选中的复选框所对应的值，然后通过一个提示信息框显示，代码如下：

```
01  public void onClick(View view) {
02      String checked = "";                                    //保存选中的值
03      if (checkBox1.isChecked()) {                            //当第一个复选框被选中
04          checked += checkBox1.getText().toString();          //输出第一个复选框内的信息
05      }
06      if (checkBox2.isChecked()) {                            //当第二个复选框被选中
07          checked += checkBox2.getText().toString();          //输出第二个复选框内的信息
08      }
09      if (checked.equals("")){
10          Toast.makeText(this, "请选择复选框内容！", Toast.LENGTH_SHORT).show();
11      }else {
12          //显示被选中复选框对应的信息
13          Toast.makeText(MainActivity.this, checked, Toast.LENGTH_LONG).show();
14      }
15  }
```

（4）运行本实例，将显示如图 5.17 所示的界面，选中复选框，单击“查询”按钮，将弹出如图 5.18 所示的消息提示框。

图 5.17　选中复选框

图 5.18　获取复选框对应的值

5.3 进度条类组件

在 Android 中，提供了进度条、拖动条和星级评分条等进度条类组件。其中，用于显示某个耗时操作完成的百分比的组件称为进度条组件，用 ProgressBar 表示；允许用户通过拖动滑块来改变值的组件称为拖动条组件，用 SeekBar 表示；同样也是允许用户通过拖动来改变进度，但是使用星星图案表示进度的组件称为星级评分条，用 RatingBar 表示。它们的继承关系如图 5.19 所示。

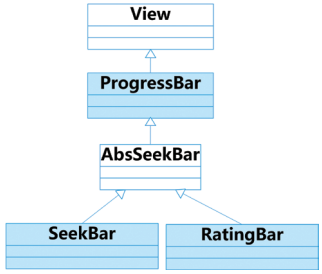


图 5.19　进度条类组件继承关系图

从图 5.19 可以看出：ProgressBar 组件继承自 View，而 SeekBar 和 RatingBar 组件又间接继承自 ProgressBar 组件。所以对于 ProgressBar 的属性，同样适用于 SeekBar 和 RatingBar 组件。下面将对这 3 个组件分别进行介绍。

5.3.1 进度条

当一个应用在后台执行时，前台界面不会有任何信息，这时用户根本不知道程序是否在执行以及执行进度等，因此需要使用进度条来提示程序执行的进度。在 Android 中，提供了两种进度条，一种是水平进度条，另一种是圆形进度条。例如，开心消消乐的启动页面中的进度条（如图 5.20 所示），为水平进度条，而一键清理大师的垃圾清理界面的进度条（如图 5.21 所示）为圆形进度条。

图 5.20　开心消消乐的启动界面

图 5.21　垃圾清理界面

在屏幕中添加进度条，可以在 XML 布局文件中通过 <ProgressBar> 标记添加，基本语法格式如下：

```
<ProgressBar
    属性列表
    >
</ProgressBar>
```

ProgressBar 组件支持的 XML 属性如表 5.4 所示。

表 5.4 ProgressBar 支持的 XML 属性

XML 属性	描　述
android:max	用于设置进度条的最大值
android:progress	用于指定进度条已完成的进度值
android:progressDrawable	用于设置进度条轨道的绘制形式

除了表 5.4 介绍的属性外，进度条组件还提供了下面两个常用方法用于操作进度。

☑ setProgress(int progress) 方法：用于设置进度完成的百分比。

☑ incrementProgressBy(int diff) 方法：用于设置进度条的进度增加或减少。当参数值为正数时，表示进度增加；当参数值为负数时，表示进度减少。

下面编写一个关于在屏幕中使用进度条的实例。

例 5.6 模拟开心消消乐启动界面

在 Android Studio 中创建 Module，名称为“Horizontal Progress Bar”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加一个水平进度条，并且将名称为 xxll.jpg 的背景图片复制到 mipmap-mdpi 目录中，修改后的代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/xxll"
    tools:context="com.mingrisoft.MainActivity">
    <!-- 水平进度条 -->
    <ProgressBar
        android:id="@+id/progressBar1"
        style="@android:style/Widget.ProgressBar.Horizontal"
        android:layout_width="match_parent"
        android:layout_height="25dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginBottom="60dp"
        android:max="100" />
</RelativeLayout>
```

说明 在上面的代码中，通过android:max属性设置水平进度条的最大进度值；通过style属性为ProgressBar指定风格，常用的style属性值如表5.5所示。

表 5.5 ProgressBar 的 style 属性的可选值

XML 属性	描　述
?android:attr/progressBarStyleHorizontal	细水平长条进度条
?android:attr/progressBarStyleLarge	大圆形进度条
?android:attr/progressBarStyleSmall	小圆形进度条
@android:style/Widget.ProgressBar.Large	大跳跃、旋转画面的进度条
@android:style/Widget.ProgressBar.Small	小跳跃、旋转画面的进度条
@android:style/Widget.ProgressBar.Horizontal	粗水平长条进度条

（2）打开主活动 MainActivity，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中添加设置当前 Activity 全屏的代码。修改后的具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
07                  WindowManager.LayoutParams.FLAG_FULLSCREEN);  //设置全屏显示
08      }
09  }
```

（3）在主活动 MainActivity 中，定义一个 ProgressBar 类的对象（用于表示水平进度条）、一个 int 型的变量（用于表示完成进度）和一个处理消息的 Handler 类的对象，具体代码如下：

```
01  private ProgressBar horizonP;                    //水平进度条
02  private int mProgressStatus = 0;                 //完成进度
03  private Handler mHandler;                        //声明一个用于处理消息的Handler类的对象
```

（4）在主活动的 onCreate() 方法中，首先获取水平进度条，然后通过匿名内部类实例化处理消息的 Handler 类（位于 android.os 包中）的对象，并重写 handleMessage() 方法，实现当耗时操作没有完成时更新进度，否则设置进度条不显示，关键代码如下：

```
01  horizonP = (ProgressBar) findViewById(R.id.progressBar1);      //获取水平进度条
02  mHandler = new Handler() {
03      @Override
04      public void handleMessage(Message msg) {
05          if (msg.what == 0x111) {
06              horizonP.setProgress(mProgressStatus);           //更新进度
07          } else {
08              Toast.makeText(MainActivity.this,
09                      "耗时操作已经完成", Toast.LENGTH_SHORT).show();
```

```
            horizonP.setVisibility(View.GONE);//设置进度条不显示，并且不占用空间
        }
    }
};
```

说明 在上面的代码中，0x111为自定义的消息代码，通过它可以区分消息，以便进行不同的处理。

（5）开启一个线程，用于模拟一个耗时操作。在该线程中，调用 sendMessage() 方法发送处理消息，具体代码如下：

```
new Thread(new Runnable() {
    public void run() {
        while (true) {          //循环获取耗时操作完成的百分比，直到耗时操作结束
            mProgressStatus = doWork();          //获取耗时操作完成的百分比
            Message m = new Message();           //创建并实例化一个消息对象
            if (mProgressStatus < 100) {         //当完成进度不到100时,表示耗时任务未完成
                m.what = 0x111;                  //设置代表耗时操作未完成的消息代码
                mHandler.sendMessage(m);         //发送信息
            } else {                             //当完成进度到达100时,表示耗时操作完成
                m.what = 0x110;                  //设置代表耗时操作已经完成的消息代码
                mHandler.sendMessage(m);         //发送消息
                break;                           //退出循环
            }
        }
    }
    //模拟一个耗时操作
    private int doWork() {
        mProgressStatus += Math.random() * 10;//改变完成进度
        try {
            Thread.sleep(200);                   //线程休眠200毫秒
        } catch (InterruptedException e) {
            e.printStackTrace();                 //输出异常信息
        }
        return mProgressStatus;                  //返回新的进度
    }
}).start();                                      //开启一个线程
```

（6）运行本实例，将显示如图 5.22 所示的运行结果。

图 5.22 模拟开心消消乐启动界面的水平进度条

5.3.2 拖动条

拖动条与进度条类似，所不同的是，拖动条允许用户拖动滑块来改变值，通常用于实现对某种数值的调节。例如，美图秀秀中的调整相片亮度的界面（如图 5.23 所示），以及在一键清理大师的设置界面中设置延迟时间和摇晃灵敏度的拖动条（如图 5.24 所示），都应用了拖动条。

图 5.23 调整相片亮度界面

图 5.24 一键清理大师的设置界面

在 Android 中，如果想在屏幕中添加拖动条，可以在 XML 布局文件中通过 <SeekBar> 标记添加，基本语法格式如下：

```
<SeekBar
    android:layout_height="wrap_content"
    android:id="@+id/seekBar1"
    android:layout_width="match_parent">
</SeekBar>
```

SeekBar 组件允许用户改变拖动滑块的外观，这可以使用 android:thumb 属性实现，该属性的属性值为一个 Drawable 对象，该 Drawable 对象将作为自定义滑块。

由于拖动条可以被用户控制，所以需要为其添加 OnSeekBarChangeListener 监听器，基本代码如下：

```
01  seekbar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
02      @Override
03      public void onStopTrackingTouch(SeekBar seekBar) {
04          //要执行的代码
05      }
06      @Override
07      public void onStartTrackingTouch(SeekBar seekBar) {
08          //要执行的代码
09      }
```

```
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
                                  boolean fromUser) {
        //其他要执行的代码
    }
});
```

说明 在上面的代码中，onProgressChanged()方法中的参数progress表示当前进度，也就是拖动条的值。

下面通过一个实例说明拖动条的应用。

例 5.7 可以设置屏幕亮度的拖动条

在 Android Studio 中创建 Module，名称为“SeekBar”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml。首先将默认添加的布局管理器修改为相对布局管理器，并将默认添加的 TextView 组件删除；然后添加一个拖动条，关键代码如下：

```
<SeekBar
    android:id="@+id/set_light"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true" />
```

（2）创建 BrightnessUtils 类，该类为亮度工具类，首先在该类中创建 getScreenBrightness () 方法用于获取当前屏幕的亮度，代码如下：

```
public static int getScreenBrightness(Context context) {
    int nowBrightnessValue = 0;
    //创建内容解析对象
    ContentResolver resolver = context.getContentResolver();
    try {
        //获取当前屏幕亮度
        nowBrightnessValue = Settings.System.getInt(resolver,
                Settings.System.SCREEN_BRIGHTNESS);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return nowBrightnessValue;
}
```

（3）创建 setSystemBrightness() 方法，用于设置系统屏幕亮度，代码如下：

```
public static void setSystemBrightness(Context context, int brightness) {
    //异常处理
    if (brightness < 1) {
        brightness = 1;
```

```
    }
    //异常处理
    if (brightness > 255) {
        brightness = 255;
    }
    saveBrightness(context, brightness);  //调用保存屏幕亮度的方法
}
```

（4）创建 saveBrightness() 方法，用于保存亮度设置状态，代码如下：

```
public static void saveBrightness(Context context, int brightness) {
    //获取设置系统屏幕亮度的uri
    Uri uri = Settings.System.getUriFor("screen_brightness");
    //保存修改后的屏幕亮度值
    Settings.System.putInt(context.getContentResolver(),
            "screen_brightness", brightness);
    //更新亮度值
    context.getContentResolver().notifyChange(uri, null);
}
```

（5）在主活动 MainActivity 中，声明一个 SeekBar 类的对象用于表示拖动条，再声明一个 BrightnessUtils 对象用于代表屏幕亮度工具类，代码如下：

```
private BrightnessUtils utils;        //屏幕亮度工具类
private SeekBar seekBar;              //拖动条
```

（6）在 onCreate() 方法中判断当前系统的版本是否大于或等于 6.0，符合条件将跳转开启系统设置权限的界面，代码如下：

```
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {    //判断系统版本是否大于或等于6.0
    if (!Settings.System.canWrite(this)) {              //判断系统权限设置
        //跳转系统权限界面
        Intent intent = new Intent
                (android.provider.Settings.ACTION_MANAGE_WRITE_SETTINGS);
        intent.setData(Uri.parse("package:" + getPackageName()));
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }
}
```

（7）初始化工具类，然后获取拖动条控件，获取当前屏幕亮度并同步拖动条，最后设置拖动条的最大值，代码如下：

```
utils =new BrightnessUtils();                          //初始化工具类
seekBar = (SeekBar) findViewById(R.id.set_light);      //获取拖动条控件
seekBar.setProgress(utils.getScreenBrightness(this)); //获取当前屏幕亮度并同步拖动条
seekBar.setMax(225);                                   //设置拖动条的最大值为225
```

（8）设置拖动条滑动监听器，用于监听拖动条变化并与屏幕亮度同步，代码如下：

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) { //改变中的监听
        utils.setSystemBrightness(MainActivity.this,i);  //将当前的进度条的值传给系统
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {  //开始改变时监听
    }
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {   //结束改变时监听
    }
});
```

（9）在 AndroidManifest.xml 文件中添加修改系统设置的权限，代码如下：

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

（10）运行本实例，将显示如图 5.25 所示的运行结果。

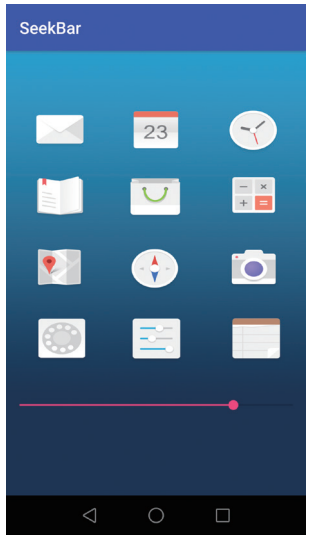


图 5.25　改变屏幕亮度的拖动条

注意 使用拖动条设置屏幕亮度前，需要将手机自动调节屏幕亮度的功能关闭。

5.3.3 星级评分条

星级评分条与拖动条类似，都允许用户通过拖动的方式来改变进度，所不同的是，星级评分条是通过五角星图案来表示进度的。通常情况下，使用星级评分条表示对某一事物的支持度或对某种服务的满意程度等。例如，淘宝中对卖家的好评度就是通过星级评分条实现的（如图 5.26 所示），百度外卖的添加评论界面也应用了星级评分条（如图 5.27 所示）。

图 5.26 淘宝的发表评价界面

图 5.27 百度外卖的添加评价界面

在 Android 中，如果想在屏幕中添加星级评分条，可以在 XML 布局文件中通过 <RatingBar> 标记添加，基本语法格式如下：

```
<RatingBar
    属性列表
    >
</RatingBar>
```

RatingBar 组件支持的 XML 属性如表 5.6 所示。

表 5.6 RatingBar 组件支持的 XML 属性

XML 属性	描　述
android:isIndicator	用于指定该星级评分条是否允许用户改变，true 为不允许改变
android:numStars	用于指定该星级评分条总共有多少个星
android:rating	用于指定该星级评分条默认的星级
android:stepSize	用于指定每次最少需要改变多少个星级，默认为 0.5 个

除了表 5.6 介绍的属性外，星级评分条还提供了以下 3 个比较常用的方法。

☑ getRating() 方法：用于获取等级，表示选中了几颗星。

☑ getStepSize()：用于获取每次最少要改变多少个星级。

☑ getProgress() 方法：用于获取进度，获取到的进度值为 getRating() 方法返回值与 getStepSize() 方法返回值之商。

下面通过一个具体的实例来说明星级评分条的应用。

例 5.8 模拟淘宝评价界面

在 Android Studio 中创建 Module，名称为“Star Rating”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，首先将默认添加的布

局管理器修改为相对布局管理器，然后添加一个星级评分条和一个普通按钮，并将背景图片复制到 mipmap-mdpi 文件夹中，修改后的代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/xing1"
    android:padding="16dp"
    tools:context="com.mingrisoft.MainActivity">
    <!-- 店铺评分-->
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn"
        android:layout_marginBottom="130dp"
        android:text="店铺评分"
        android:textSize="20sp" />
    <!-- 星级评分条 -->
    <RatingBar
        android:id="@+id/ratingBar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn"
        android:layout_marginBottom="60dp"
        android:numStars="5"
        android:rating="0" />
    <!--发表评价-->
    <Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:background="#FF5000"
        android:text="发表评价" />
</RelativeLayout>
```

（2）打开主活动 MainActivity，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义一个 RatingBar 类的对象，用于表示星级评分条。修改后的具体代码如下：

```
public class MainActivity extends Activity {
```

```
    private RatingBar ratingbar;    //星级评分条
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

（3）在主活动的 onCreate() 方法中，首先获取布局文件中添加的星级评分条，然后获取提交按钮，并为其添加单击事件监听器，在重写的 onClick() 事件中，实现获取进度、等级和每次最少要改变多少个星级并显示到日志中，同时通过消息提示框显示获得的星的个数，关键代码如下：

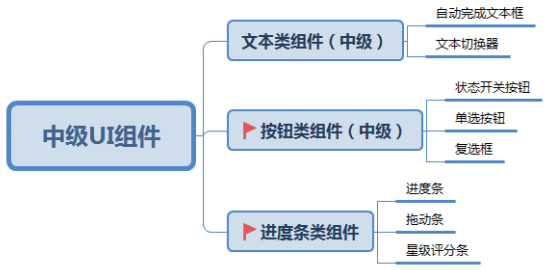
```
ratingbar = (RatingBar) findViewById(R.id.ratingBar1);  //获取星级评分条
Button button=(Button)findViewById(R.id.btn);           //获取"提交"按钮
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int result = ratingbar.getProgress();           //获取进度
        float rating = ratingbar.getRating();           //获取等级
        float step = ratingbar.getStepSize();           //获取每次最少要改变多少个星级
        Log.i("星级评分条","step="+step+" result="+result+" rating="+rating);
        Toast.makeText(MainActivity.this,
                "你得到了" + rating + "颗星", Toast.LENGTH_SHORT).show();
    }
});
```

（4）运行本实例，将显示如图 5.28 所示的运行结果。

图 5.28　单击“发表评价”按钮显示选择了几颗星

5.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 6 章

高级 UI 组件

本章介绍 Android 开发常用的一些高级 UI 组件，主要包括图像类组件（高级）、列表类组件、切换类组件以及通用组件。本章主要学习一些可以滑动的列表或是切换类的组件，通过此类组件可以实现动态 UI 界面的效果。

6.1 图像类组件（高级）

6.1.1 图像切换器

图像切换器（ImageSwitcher），用于实现带动画效果的图片切换功能。例如，手机相册中滑动查看相片的功能（如图 6.1 所示），以及蘑菇街中挑选图片界面（如图 6.2 所示）。

图 6.1　手机相册滑动查看相片界面

图 6.2　蘑菇街中挑选图片界面

在使用 ImageSwitcher 时，必须通过它的 setFactory() 方法为 ImageSwitcher 类设置一个 ViewFactory，用于将显示的图片和父窗口区分开，对于 setFactory() 方法的参数，需要通过实例化 ViewSwitcher.ViewFactory 接口的实现类来指定。在创建 ViewSwitcher.ViewFactory 接口的实现类时，需要重写 makeView() 方法，用于创建显示图片的 ImageView。makeView() 方法将返回一个显示图

片的 ImageView。另外，在使用图像切换器时，还有一个方法非常重要，那就是 setImageResource() 方法，该方法用于指定要在 ImageSwitcher 中显示的图片资源。

下面通过一个具体的实例来说明图像切换器的用法。

例 6.1 模拟手机相册的滑动查看相片

在 Android Studio 中创建 Module，名称为“ImageSwitcher”，具体步骤如下：

（1）首先在 res 目录中单击鼠标右键 New → Directory 创建一个名称为 anim 的目录，然后在该目录中单击右键 New → Animation resource file 创建名称为 slide_in_left 的从左平移进入动画。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<translate
    android:fromXDelta="-100%p"
    android:toXDelta="0"
    android:duration="1000"/>
</set>
```

（2）按照以上步骤分别创建名称为 slide_in_right、slide_out_left、slide_out_right 的三个动画文件。

① slide_in_right：从右平移进入动画，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="1000"
        android:fromXDelta="100%p"
        android:toXDelta="0"/>
</set>
```

② slide_out_left：从左退出动画，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="1000"
        android:fromXDelta="0"
        android:toXDelta="-100%p"/>
</set>
```

③ slide_out_right：从右退出动画，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="1000"
        android:fromXDelta="0"
```

```
        android:toXDelta="100%p"/>
</set>
```

（3）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并将默认添加的 TextView 组件删除，然后添加一个图像切换器 ImageSwitcher。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <!--图像切换器-->
    <ImageSwitcher
        android:id="@+id/imageswitcher"
        android:layout_centerVertical="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </ImageSwitcher>
</RelativeLayout>
```

（4）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中添加设置全屏的代码。修改后的具体代码如下：

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置全屏显示
    }
}
```

（5）在主活动的 onCreate() 方法的上方，首先声明并初始化一个保存要显示图像 ID 的数组，再声明一个图像切换器的对象，然后声明一个保存当前显示图像索引的变量，最后声明手指按下和抬起时的 X 坐标，具体代码如下：

```
private int[] arrayPictures = new int[]{
        R.mipmap.img01, R.mipmap.img02, R.mipmap.img03,
        R.mipmap.img04, R.mipmap.img05, R.mipmap.img06,
        R.mipmap.img07, R.mipmap.img08, R.mipmap.img09,
};//声明并初始化一个保存要显示图像ID的数组
```

```
06  private ImageSwitcher imageSwitcher; //声明一个图像切换器对象
07  //要显示的图片在图片数组中的Index
08  private int pictutureIndex;
09  //左右滑动时手指按下的X坐标
10  private float touchDownX;
11  //左右滑动时手指抬起的X坐标
12  private float touchUpX;
```

（6）在主活动的 onCreate() 方法中，首先获取布局文件中添加的图像切换器，然后为其设置一个 ImageSwitcher.ViewFactory，并重写 makeView() 方法，最后为图像切换器设置默认显示的图像，关键代码如下：

```
01 imageSwitcher = (ImageSwitcher) findViewById(R.id.imageswitcher); //获取图像切换器
02 //为ImageSwicher设置Factory，用来为ImageSwicher制造ImageView
03 imageSwitcher.setFactory(new ViewSwitcher.ViewFactory() {
04     @Override
05     public View makeView() {
06         //实例化一个ImageView类的对象
07         ImageView imageView = new ImageView(MainActivity.this);
08         //根据ID加载默认显示图片
09         imageView.setImageResource(arrayPictures[pictutureIndex]);
10         return imageView; //返回imageView对象
11     }
12 });
```

说明 在上面的代码中，使用ImageSwitcher类的父类ViewAnimator的setInAnimation()方法和setOutAnimation()方法为图像切换器设置动画效果；调用其父类ViewSwitcher的setFactory()方法指定视图切换工厂，其参数为ViewSwitcher.ViewFactory类型的对象。

（7）为 imageSwitcher 创建一个触摸事件，首先获取左右滑动按下和抬起的 X 坐标，然后判断当手指从左向右滑动时，设置图片切换的动画为左进右出，相反再判断当手指从右向左滑动时，设置图片切换的动画为右进左出，具体代码如下：

```
01  imageSwitcher.setOnTouchListener(new View.OnTouchListener() {
02      @Override
03      public boolean onTouch(View v, MotionEvent event) {
04          if (event.getAction() == MotionEvent.ACTION_DOWN) {
05              //取得左右滑动时手指按下的X坐标
06              touchDownX = event.getX();
07              return true;
08          } else if (event.getAction() == MotionEvent.ACTION_UP) {
09              //取得左右滑动时手指松开的X坐标
10              touchUpX = event.getX();
11              //从左往右，看下一张
12              if (touchUpX - touchDownX > 100) {
13                  //取得当前要看的图片的index
14                  pictutureIndex = pictutureIndex == 0 ?
```

```
                        arrayPictures.length - 1 : pictutureIndex - 1;
                //设置图片切换的动画
                imageSwitcher.setInAnimation(AnimationUtils.
                        loadAnimation(MainActivity.this, R.anim.slide_in_left));
                imageSwitcher.setOutAnimation(AnimationUtils.
                        loadAnimation(MainActivity.this, R.anim.slide_out_right));
                //设置当前要看的图片
                imageSwitcher.setImageResource(arrayPictures[pictutureIndex]);
                //从右往左，看上一张
            } else if (touchDownX - touchUpX > 100) {
                //取得当前要看的图片index
                pictutureIndex = pictutureIndex == arrayPictures.length - 1
                        ? 0 : pictutureIndex + 1;
                //设置切换动画
                imageSwitcher.setOutAnimation(AnimationUtils.
                        loadAnimation(MainActivity.this, R.anim.slide_out_left));
                imageSwitcher.setInAnimation(AnimationUtils.
                        loadAnimation(MainActivity.this, R.anim.slide_in_right));
                //设置要看的图片
                imageSwitcher.setImageResource(arrayPictures[pictutureIndex]);
            }
            return true;
        }
        return false;
    }
});
```

（8）打开 AndroidManifest.xml 文件，将其中的 <application> 标记的 android:theme 属性值“@style/AppTheme”修改为“@style/Theme.AppCompat”，修改后的 android:theme 属性的代码如下：

```
android:theme="@style/Theme.AppCompat"
```

（9）运行本实例，将显示如图 6.3 所示的运行效果。从左向右滑动屏幕将显示如图 6.4 所示。

图 6.3　默认的运行效果

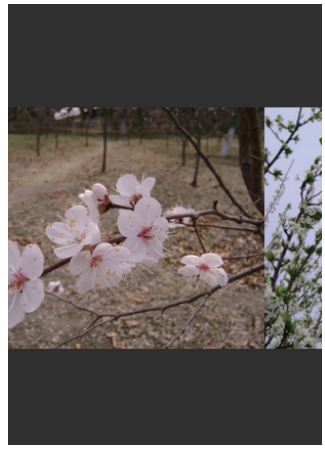


图 6.4　从左向右滑动屏幕

6.1.2 画廊视图

画廊视图（Gallery）表示，能够按水平方向显示内容，并且可用手指直接拖动图片移动，一般用来浏览图片，被选中的选项位于中间，并且可以响应事件显示信息。在使用画廊视图时，首先需要在屏幕上添加 Gallery 组件，通常使用 <Gallery> 标记在 XML 布局文件中添加，其基本语法如下：

```
< Gallery
    属性列表
    >
</Gallery>
```

Gallery 组件支持的 XML 属性如表 6.1 所示。

表 6.1　Gallery 支持的 XML 属性

XML 属性	描　　述
android:animationDuration	用于设置列表项切换时的动画持续时间
android:gravity	用于设置对齐方式
android:spacing	用于设置列表项之间的间距
android:unselectedAlpha	用于设置没有选中的列表项的透明度

使用画廊视图，也需要使用 Adapter 提供要显示的数据。通常使用 BaseAdapter 类为 Gallery 组件提供数据。下面通过一个具体的实例演示通过 BaseAdapter 适配器为 Gallery 组件提供要显示的图片。

例 6.2　模拟淘宝搜索宝贝时的提示列表

在 Android Studio 中创建 Module，名称为“Gallery View”，在该 Module 中实现本实例，具体步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的相对布局管理器修改为垂直线性布局管理器，并将默认添加的 TextView 组件删除，然后添加一个 id 属性为 gallery1 的 Gallery 组件，并设置其列表项之间的间距为 5dp，以及设置未选中项的透明度。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:tools="http://schemas.android.com/tools"
04      android:layout_width="match_parent"
05      android:layout_height="match_parent"
06      android:orientation="vertical"
07      tools:context=".MainActivity">
08      <Gallery
09          android:id="@+id/gallery1"
10          android:spacing="5dp"
11          android:unselectedAlpha="0.6"
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content" />
14  </LinearLayout>
```

（2）在 res\values 目录中点击鼠标右键 New → Values resource file，创建一个名称为 attr.xml 的文件，在该文件中定义一个 styleable 对象，用于组合多个属性。这里只指定了一个系统自带的 android:galleryItemBackground 属性，用于设置各选项的背景，具体代码如下：

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

（3）在主活动 MainActivity 中，定义一个用于保存要显示图片 id 的数组（需要将要显示的图片复制到 res/mipmap 文件夹中），关键代码如下：

```
private int[] imageId = new int[] { R.mipmap.img01, R.mipmap.img02,
        R.mipmap.img03, R.mipmap.img04, R.mipmap.img05,
        R.mipmap.img06, R.mipmap.img07, R.mipmap.img08,
        R.mipmap.img09, R.mipmap.img10, R.mipmap.img11,
        R.mipmap.img12, };          //定义并初始化保存图片id的数组
```

（4）在主活动的 onCreate() 方法中，获取在布局文件中添加的画廊视图，关键代码如下：

```
Gallery gallery = (Gallery) findViewById(R.id.gallery1);          //获取Gallery组件
```

（5）顺序添加代码，创建 BaseAdapter 类的对象，并重写其中的 getView()、getItemId()、getItem() 和 getCount() 方法，其中最主要的是重写 getView() 方法来设置显示图片的格式，具体代码如下：

```
BaseAdapter adapter = new BaseAdapter() {
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageview;                                //声明ImageView的对象
        if (convertView == null) {
            imageview = new ImageView(MainActivity.this);//实例化ImageView的对象
            imageview.setScaleType(ImageView.ScaleType.FIT_XY);//设置缩放方式
            imageview
                    .setLayoutParams(new Gallery.LayoutParams(120, 90));
            TypedArray typedArray = obtainStyledAttributes(R.styleable.Gallery);
            imageview.setBackgroundResource(typedArray.getResourceId(
                    R.styleable.Gallery_android_galleryItemBackground,0));
            imageview.setPadding(5, 0, 5, 0);           //设置ImageView的内边距
        } else {
            imageview = (ImageView) convertView;
        }
        imageview.setImageResource(imageId[position]); //为ImageView设置要显示的图片
        return imageview;                               //返回ImageView
    }
    /*
```

```
21          * 功能：获取当前选项的id
22          */
23         @Override
24         public long getItemId(int position) {
25             return position;
26         }
27         /*
28          * 功能：获取当前选项
29          */
30         @Override
31         public Object getItem(int position) {
32             return position;
33         }
34         /*
35          * 获取数量
36          */
37         @Override
38         public int getCount() {
39             return imageId.length;
40         }
41     };
```

（6）顺序添加代码，将步骤（5）中创建的适配器与 Gallery 关联，并且让中间的图片选中，为了在用户单击某张图片时显示对应的位置，还需要为Gallery 添加单击事件监听器，具体代码如下：

```
01  gallery.setAdapter(adapter);                          //将适配器与Gallery关联
02  gallery.setSelection(imageId.length / 2);             //选中中间的图片
03  gallery.setOnItemClickListener(new AdapterView.OnItemClickListener() {
04      @Override
05      public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
06     Toast.makeText(MainActivity.this,"您选择了第" + String.valueOf(position) + "张图片",
07                  Toast.LENGTH_SHORT).show();
08      }
09  });
```

（7）运行本实例，默认的运行效果如图 6.5 所示。单击图片后将显示对应的提示信息，如图 6.6 所示。

图 6.5　应用画廊视图显示图片列表

图 6.6　单击图片后显示提示信息

6.2 列表类组件

在 Android 中提供了两种列表类组件，一种是下拉列表框，通常用于弹出一个下拉菜单供用户选择，用 Spinner 表示；另一种是列表视图，通常用于实现在一个窗口中只显示一个列表，使用 ListView 表示。它们的继承关系如图 6.7 所示。

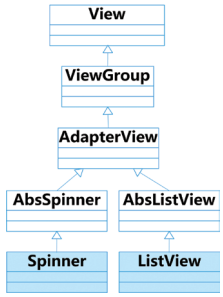


图 6.7　列表类组件继承关系图

从图 6.7 中可以看出：Spinner 和 ListView 组件都间接继承自 ViewGroup，所以它们都属于容器类组件。而由于它们又间接继承自 AdapterView，所以它们都可以采用合适的方式显示多个列表项。下面将对这两个组件分别进行介绍。

6.2.1 下拉列表框

Android 中提供的下拉列表框（Spinner），通常用于提供一系列列表项供用户进行选择。例如，豆瓣网的搜索界面中的选择搜索类型的下拉列表框（如图 6.8 所示），以及手机相册的选择相片显示方式的下拉列表框（如图 6.9 所示）。

图 6.8　豆瓣网的搜索界面

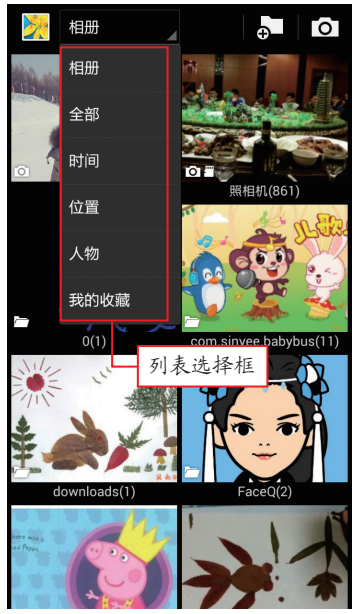


图 6.9　手机相册的相片显示方式界面

在 Android 中，在 XML 布局文件中通过 <Spinner> 标记添加下拉列表框，基本语法格式如下：

```
<Spinner
android:entries="@array/数组名称"
android:prompt="@string/info"
其他属性
    >
</Spinner>
```

其中，android:entries 为可选属性，用于指定列表项，如果在布局文件中不指定该属性，可以在 Java 代码中通过为其指定适配器的方式指定；android:prompt 属性也是可选属性，用于指定下拉列表框的标题。

说明 在Android 5.0中，当应用采用默认的主题（Theme.Holo）时，设置android:prompt属性看不到具体的效果，如果采用Theme.Black，就可以在弹出的下拉框中显示该标题。

通常情况下，如果下拉列表框中要显示的列表项是可知的，那么可将其保存在数组资源文件中，然后通过数组资源来为下拉列表框指定列表项。这样，就可以在不编写 Java 代码的情况下实现一个下拉列表框。下面将通过一个具体的实例来说明如何在不编写 Java 代码的情况下，在屏幕中添加下拉列表框。

例 3.5 豆瓣网搜索下拉列表

在 Android Studio 中创建 Module，名称为“Spinner”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml。首先将默认添加的布局管理器修改为水平线性布局管理器，然后在布局文件中添加一个 <spinner> 标记，并为其指定 android:entries 属性，最后添加一个 EditText 组件，将需要的背景图片复制到 mipmap-mdpi 文件夹中，具体代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/xila"
    android:orientation="horizontal"
    tools:context="com.mingrisoft.MainActivity">
    <!--列表选择框-->
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:entries="@array/ctype">
    </Spinner>
    <!--搜索文本框-->
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="搜索"
```

```
        android:textColor="#F8F8FF" />
</LinearLayout>
```

（2）编写用于指定列表项的数组资源文件，并将其保存在 res/values 目录中，这里将其命名为“arrays.xml”，在该文件中添加一个字符串数组，名称为“ctype”，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string-array name="ctype">
    <item>全部</item>
    <item>电影/电视</item>
    <item>图书</item>
    <item>唱片</item>
    <item>小事</item>
    <item>用户</item>
    <item>小组</item>
    <item>群聊</item>
    <item>游戏/应用</item>
    <item>活动</item>
</string-array>
</resources>
```

（3）打开主活动 MainActivity，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类。修改后的关键代码如下：

```
import android.app.Activity;
public class MainActivity extends Activity {
}
```

（4）打开 AndroidManifest.xml 文件，将其中的 <application> 标记的 android:theme 属性值“@style/AppTheme”修改为“@style/Theme.AppCompat.Light.DarkActionBar”，修改后的 android:theme 属性的代码如下：

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

（5）添加下拉列表框后，如果需要在用户选择不同的列表项后，执行相应的处理，则可以为该下拉列表框添加 OnItemSelectedListener 事件监听器。例如，为 spinner 添加选择列表项事件监听器，并通过 getItemAtPosition() 方法获取选中的值，然后用 Toast.makeText() 方法将获取的值显示出来，可以在 onCreate() 方法中使用下面的代码：

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);                    //获取下拉列表
 //为下拉列表创建监听事件
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view,
            int position, long id) {
        String result = parent.getItemAtPosition(position).toString(); //获取选择项的值
```

```
08            //显示被选中的值
09            Toast.makeText(MainActivity.this, result, Toast.LENGTH_SHORT).show();
10        }
11        @Override
12        public void onNothingSelected(AdapterView<?> parent) {
13        }
14    });
```

（6）运行本实例，将显示类似豆瓣网搜索页面，单击下拉列表框右侧的黑色倒三角，可以显示下拉列表框的各个列表项，如图 6.10 所示，选择某一列表项（如电影 / 电视）后，将显示选择项的值，如图 6.11 所示。

图 6.10　豆瓣网搜索下拉列表框

图 6.11　显示选择的结果

在使用下拉列表框时，如果不在布局文件中直接为其指定要显示的列表项，也可以通过为其指定适配器的方式指定。下面仍然以例 6.3 为例介绍通过指定适配器的方式指定列表项的方法。

为下拉列表框指定适配器，通常分为以下 3 个步骤。

（1）创建一个适配器对象，通常使用 ArrayAdapter 类。首先需要创建一个一维的字符串数组，用于保存要显示的列表项，然后使用 ArrayAdapter 类的构造方法 ArrayAdapter(Context context, int textViewResourceId, T[] objects) 实例化一个 ArrayAdapter 类的实例，具体代码如下：

```
String[] ctype=new String[]{"全部","电影","图书","唱片","小事",
            "用户","小组","群聊","游戏","活动"};
ArrayAdapter<String> adapter=new ArrayAdapter<String>
            (this,android.R.layout.simple_spinner_item,ctype);
```

（2）为适配器设置列表框下拉时的选项样式，具体代码如下：

```
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

（3）将适配器与下拉列表框关联，具体代码如下：

```
spinner.setAdapter(adapter);                    //将适配器与选择列表框关联
```

在屏幕上添加下拉列表框后，可以使用下拉列表框的 getSelectedItem() 方法获取下拉列表框的选中值，例如，要获取图 6.11 所示下拉列表框选中项的值，可以使用下面的代码：

```
01  Spinner spinner = (Spinner) findViewById(R.id.spinner);    //获取下拉列表框
02  spinner.getSelectedItem();                                  //获取选中项的值
```

6.2.2 列表视图

列表视图（ListView）是 Android 中最常用的一种视图组件，它以垂直列表的形式列出需要显示的列表项。例如，微信通讯录界面中的联系人列表（如图 6.12 所示），以及 QQ 的图片浏览设置界面（如图 6.13 所示）。

图 6.12 微信通讯录界面

图 6.13 QQ 的图片浏览设置界面

在 Android 中，可以通过在 XML 布局文件中使用 <ListView> 标记添加列表视图，其基本语法格式如下：

```
<ListView
属性列表
    >
</ListView>
```

ListView 组件支持的常用 XML 属性如表 6.2 所示。

表 6.2 ListView 组件支持的 XML 属性

XML 属性	描　述
android:divider	用于为列表视图设置分隔条，既可以用颜色分隔，也可以用 Drawable 资源分隔
android:dividerHeight	用于设置分隔条的高度

续表

XML 属性	描　　述
android:entries	用于通过数组资源为 ListView 指定列表项
android:footerDividersEnabled	用于设置是否在 footer View（底部视图）之前绘制分隔条，默认值为 true，设置为 false 时，表示不绘制。使用该属性时，需要通过 ListView 组件提供的 addFooterView() 方法为 ListView 设置 footer View
android:headerDividersEnabled	用于设置是否在 header View（头部视图）之后绘制分隔条，默认值为 true，设置为 false 时，表示不绘制。使用该属性时，需要通过 ListView 组件提供的 addHeaderView() 方法为 ListView 设置 header View

例如，在布局文件中添加一个列表视图，并通过数组资源为其设置列表项。具体代码如下：

```
<ListView android:id="@+id/listView1"
    android:entries="@array/ctype"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"/>
```

在上面的代码中，使用了名称为 ctype 的数组资源，因此，需要在 res/values 目录中创建一个定义数组资源的 XML 文件 arrays.xml，并在该文件中添加名称为 ctype 的字符串数组，关键代码如下：

```
<resources>
    <string-array name="ctype">
        <item>情景模式</item>
        …      <!-- 省略了其他项的代码 -->
        <item>连接功能</item>
    </string-array>
</resources>
```

运行上面的代码，将显示如图 6.14 所示的列表视图。

图 6.14　在布局文件中添加的列表视图

在使用列表视图时，重要的是如何设置选项内容。同 Spinner 下拉列表框一样，如果没有在布局文件中为 ListView 指定要显示的列表项，也可以通过为其设置 Adapter 来指定需要显示的列表项。

通过 Adapter 来为 ListView 指定要显示的列表项，可以分为以下两个步骤。

（1）创建 Adapter 对象。对于纯文字的列表项，通常使用 ArrayAdapter 对象。创建 ArrayAdapter 对象通常可以有两种方式：一种是通过数组资源文件创建；另一种是通过在 Java 文件中使用字符串数组创建。这与 5.2.1 节 Spinner 下拉列表框中介绍的创建 ArrayAdapter 对象基本相同，所不同的就是在创建该对象时，指定列表项的外观形式。在 Android API 中默认提供了一些用于设置外观形式的布局文件，通过这些布局文件，可以很方便地指定 ListView 的外观形式。常用的布局文件有以下几个。

- ☑ simple_list_item_1：每个列表项都是一个普通的文本。
- ☑ simple_list_item_2：每个列表项都是一个普通的文本（字体略大）。
- ☑ simple_list_item_checked：每个列表项都有一个已选中的列表项。
- ☑ simple_list_item_multiple_choice：每个列表项都是带复选框的文本。
- ☑ simple_list_item_single_choice：每个列表项都是带单选按钮的文本。

（2）将创建的适配器对象与 ListView 相关联，可以通过 ListView 对象的 setAdapter() 方法实现，具体代码如下：

```
listview.setAdapter(adapter); //将适配器与ListView关联
```

下面通过一个具体的实例演示通过适配器指定列表项来创建 ListView。

例 3.5 模拟支付宝朋友列表

在 Android Studio 中创建 Module，名称为“Alipay Friends”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，将默认添加的 TextView 组件删除，添加一个 ListView 组件，用于显示朋友列表，关键代码如下：

```
<ListView
    android:id="@+id/listview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="180dp"
    android:layout_marginBottom="58dp">
</ListView>
```

（2）在新建项目的 res/layout 目录下单击鼠标右键新建一个布局文件，命名为“item”，用来显示图片、名称以及信息内容，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <!--图标-->
    <ImageView
```

```
        android:id="@+id/img"
        android:layout_width="60dp"
        android:layout_height="60dp" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="60dp"
        android:layout_toRightOf="@+id/img"
        android:gravity="center_vertical"
        android:orientation="vertical">
        <!--名称-->
        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:textColor="#000000"
            android:textSize="18sp" />
        <!--信息-->
        <TextView
            android:id="@+id/info"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:layout_marginRight="15dp"
            android:singleLine="true"
            android:textSize="13sp" />
    </LinearLayout>
</LinearLayout>
```

（3）打开主活动 MainActivity，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后分别定义图标数组、名字数组、信息数组。

（4）在主活动的 onCreate() 方法中，首先获取布局文件中添加的 ListView，然后创建一个 List 集合，通过 for 循环将图标、名字、信息放到 Map 中，并添加到 List 集合中，关键代码如下：

```
ListView listview = (ListView) findViewById(R.id.listview);    //获取列表视图
//创建一个list集合
List<Map<String, Object>> listItems = new ArrayList<Map<String, Object>>();
//通过for循环将图片id和列表项文字放到Map中，并添加到List集合中
for (int i = 0; i < icons.length; i++) {
    Map<String, Object> map = new HashMap<String, Object>(); //实例化Map对象
    map.put("图标", icons[i]);
    map.put("名字", names[i]);
    map.put("信息",infos[i]);
    listItems.add(map);                                    //将map对象添加到List集合中
   }
```

（5）创建 SimpleAdapter 适配器，并且将适配器与 ListView 关联，为 ListView 创建监听事件，然后通过 getItemAtPosition() 方法，获取选中的值，最后通过 Toast.makeText() 方法将获取的值显示出来，具体代码如下：

```
SimpleAdapter adapter = new SimpleAdapter(this, listItems,
        R.layout.item, new String[]{"名字", "图标","信息"}, new int[]{
        R.id.name, R.id.img,R.id.info});    //创建SimpleAdapter
listview.setAdapter(adapter);               //将适配器与ListView关联
listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    //获取选择项的值
    Map<String, Object> map = (Map<String, Object>) parent.getItemAtPosition(position);
        Toast.makeText(MainActivity.this, map.get("名字").
                toString(), Toast.LENGTH_SHORT).show();
    }
});
```

（6）运行本实例，将显示如图 6.15 所示的运行结果。

图 6.15　支付宝朋友列表

6.3　切换类组件

本节将介绍如何使用 ViewFlipper 组件实现组件之间的切换，然后介绍翻页组件 ViewPager 配合适配器 PagerAdapter 的用法以及翻页标题栏 PagerTabStrip 的用法。下面将对这三种切换类组件进行详细的介绍。

6.3.1 组件的切换（ViewFlipper）

ViewFilpper 是 ViewAnimator 的子类，该类可以通过 addView() 方法添加多个组件，然后再通过 ViewFilpper 来实现多个组件之间的切换效果。例如，手机图库中自动播放幻灯片的运行效果就可以通过它来实现。

在 XML 布局文件中 ViewFilpper 的基本语法格式如下：

```
<ViewFlipper
    android:id="@+id/ID号"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</ViewFlipper>
```

ViewFilpper 所支持的相关方法如表 6.3 所示。

表 6.3 ViewFilpper 所支持的相关方法

相 关 方 法	描　　述
startFlipping()	启动组件切换功能
stopFlipping()	停止组件切换功能
setAutoStart()	设置是否自动开始
setFilpInterval()	设置组件切换的间隔时间
showPrevious()	调用该方法显示上一个组件，适用于手动切换
showNext()	调用该方法显示下一个组件，适用于手动切换
setInAnimation()	设置组件进入动画文件
setOutAnimation()	设置组件退出动画文件

下面通过一个具体的实例演示 ViewFilpper 组件的具体应用。

例 3.5　自动播放幻灯片

在 Android Studio 中创建 Module，名称为“View Filpper”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加一个 View Filpper 组件用于实现图片组件之间的切换，最后在 View Filpper 组件当中添加 5 个 ImageView 组件用于显示 5 本不同语言的图书。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
06      android:layout_height="match_parent"
07      tools:context="com.mingrisoft.MainActivity">
```

```
    <!--组件切换-->
    <ViewFlipper
        android:id="@+id/vf"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <!--图片1-->
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/img1"/>
        <!--图片2-->
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/img2"/>
        <!--图片3-->
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/img3"/>
        <!--图片4-->
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/img4"/>
        <!--图片5-->
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/img5"/>
    </ViewFlipper>
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，首先让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义所需的成员变量，再获取布局文件中的 ViewFilpper 组件并设置切换图片组件的间隔时间，最后设置切换组件时进入与退出的动画文件。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    private boolean isStart;              //定义启动与暂停标记
    private ViewFlipper viewFlipper;      //定义组件切换
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        viewFlipper= (ViewFlipper) findViewById(R.id.vf);  //获取ViewFlipper组件
```

```
09          viewFlipper.setFlipInterval(2000);                    //设置切换组件的间隔时间
10          viewFlipper.setInAnimation(this,R.anim.in_animation); //设置组件进入动画
11          viewFlipper.setOutAnimation(this,R.anim.out_animation);//设置组件退出动画
12      }
13  }
```

说明 参考源码在res目录中找到anim动画文件，然后将该文件复制在项目代码的res目录下即可。

（3）重写 onTouchEvent() 方法，在该方法中实现切换组件的启动与停止功能。具体代码如下：

```
01  @Override
02  public boolean onTouchEvent(MotionEvent event) {
03      if (event.getAction()==MotionEvent.ACTION_DOWN){
04          if (isStart==false){                                  //判断标记是否为暂停状态
05              viewFlipper.setAutoStart(true);                   //开启自动播放
06              viewFlipper.startFlipping();                      //启动组件切换
07              isStart=true;                                     //修改标记为开启状态
08          }else {
09              viewFlipper.stopFlipping();                       //停止组件切换
10              viewFlipper.setAutoStart(false);                  //关闭自动播放
11              isStart=false;                                    //恢复标记暂停状态
12          }
13      }
14      return super.onTouchEvent(event);
15  }
```

（4）运行本实例，将显示如图 6.16 所示的运行效果。单击手机或模拟器屏幕时每两秒钟自动更换下一个 ImageView 组件所显示的图片，效果如图 6.17 所示。

图 6.16　默认显示的运行效果

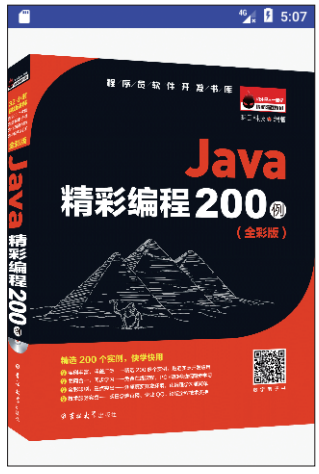


图 6.17　切换第二个组件所显示的图片

6.3.2 翻页组件（ViewPager）

ViewPager 是由 Android v4 包提供的一个组件，它是 ViewGroup 的子类，所以它也是一个容器类，可以在其中添加其他的 view 类。ViewPager 需要一个 PagerAdapter 适配器给它提供数据，它使

用的监听器是 OnPageChangeListener，用于监听页面切换的事件。在 Android 中，该组件经常被使用在左右滑动时显示多个页面的运行效果。例如，App 中的引导界面（如图 6.18 所示），以及京东商城中自动轮播的广告（如图 6.19 所示）都可以使用 ViewPager 来实现。

图 6.18　App 引导界面

图 6.19　京东商城轮播广告

在 XML 布局文件中添加 ViewPager 的基本语法格式如下：

```
<android.support.v4.view.ViewPager
    android:id="@+id/ID号"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</android.support.v4.view.ViewPager>
```

下面对 ViewPager 的重要方法进行介绍。

（1）PagerAdapter：ViewPager 的适配器。创建该对象时需要重写它的 4 个方法。具体方法如下：

◆ instantiateItem(): 将当前视图添加到视图容器中并返回当前所显示的视图。

◆ destroyItem()：从视图容器中移除指定位置的页面。

◆ getCount()：可用滑动页面的数量。

◆ isViewFromObject(): 确定页面视图是否与返回的对象相关联 , 返回 view == object 即可。

（2）ViewPager.OnPageChangeListener：页面切换时的监听事件，其中需要实现的 3 个方法如下：

◆ onPageScrollStateChanged(): 在翻译状态发生改变时被调用。

◆ onPageScrolled()：在翻页过程中该方法被调用。

◆ onPageSelected(): 当页面被选中时该方法被调用。

例如，为名称 viewPager 添加页面切换时的监听事件，可以使用下面的代码：

```
01  ViewPager viewPager= (ViewPager) findViewById(R.id.viewPager);     //获取ViewPager组件
02  //添加页面切换时的监听事件
03  viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
04      @Override
05      public void onPageScrolled(int position, float positionOffset,
```

```
                                int positionOffsetPixels) {
    }
    @Override
    public void onPageSelected(int position) {
    }
    @Override
    public void onPageScrollStateChanged(int state) {
    }
});
```

下面通过一个实例演示 ViewPager 的具体应用。

例 6.6 模拟 App 引导界面

在 Android Studio 中创建 Module，名称为“View Pager”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加一个 ViewPager 组件用于实现引导页面的切换。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--ViewPager组件-->
    <android.support.v4.view.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </android.support.v4.view.ViewPager>
</RelativeLayout>
```

（2）在 java/com.mingrisoft 目录中创建 1 个新的 Activity，名称为 Main2Activity。首先让 Main2Activity 直接继承 Activity 并导入 android.app.Activity 类，然后在 onCreate() 方法中设置当前界面的全屏代码，最后在 activity_main2.xml 的布局文件中设置显示主界面 2 的背景图片。

（3）在 res/layout 目录中分别创建名称为 layout1、layout2、layout3、layout4.xml 文件，然后在前三个文件的布局管理器中分别设置三个页面的背景图片，最后在 layout4.xml 文件中设置第 4 个页面的背景图片并设置 1 个 Button 按钮用于启动主界面 2。layout4.xml 的布局代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg4"
```

```
    android:orientation="vertical">
    <!--启动主界面2的按钮-->
    <Button
        android:id="@+id/btn"
        android:layout_width="170dp"
        android:layout_height="50dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="360dp"
        android:background="@mipmap/btn_bg"
        android:onClick="onEnter"
        android:textSize="20sp" />
</LinearLayout>
```

（4）打开主活动 MainActivity.java 文件，修改默认生成的代码，首先让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义所需的全局变量并在 onCreate() 方法中加载需要显示在页面中的布局文件，最后将所有页面添加在数组列表中并且为 ViewPager 指定所使用的适配器。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    private View view1,view2,view3,view4;     //4个页面视图
    private List<View> viewList;              //保存页面的数组列表
    private ViewPager viewPager;              //ViewPager组件
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);     //设置全屏显示
        LayoutInflater lf = getLayoutInflater().from(this);      //获取布局填充器
        viewPager= (ViewPager) findViewById(R.id.viewPager);     //获取ViewPager组件
        view1 = lf.inflate(R.layout.layout1, null);      //加载页面1的布局文件
        view2 = lf.inflate(R.layout.layout2, null);      //加载页面2的布局文件
        view3 = lf.inflate(R.layout.layout3, null);      //加载页面3的布局文件
        view4 = lf.inflate(R.layout.layout4, null);      //加载页面4的布局文件
        viewList = new ArrayList<View>();                //创建保存4个页面的数组列表
        viewList.add(view1);                             //向数组列表中添加第1个页面
        viewList.add(view2);                             //向数组列表中添加第2个页面
        viewList.add(view3);                             //向数组列表中添加第3个页面
        viewList.add(view4);                             //向数组列表中添加第4个页面
        viewPager.setAdapter(adapter);                   //设置适配器
    }
}
```

（5）创建 PagerAdapter 对象并重写所需要的 4 个方法来实现所有页面的切换功能。具体代码如下：

```
PagerAdapter adapter = new PagerAdapter() {              //创建适配器
```

```
    @Override
    public int getCount() {          //获取页面个数
        return viewList.size();      //返回页面数量
    }
    //确定页面视图是否与返回的对象相关联
    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }
    //从视图容器中移除指定位置的页面
    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        container.removeView(viewList.get(position));
    }
    //返回当前所显示的视图
    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        container.addView(viewList.get(position));
        return viewList.get(position);
    }
};
```

（6）创建 onEnter() 方法，该方法是第 4 个页面中按钮的单击事件，用于实现单击“开启私人订制”按钮后跳转至主界面 2 当中。具体代码如下：

```
public void onEnter(View view){ //第4个页面中按钮单击事件方法
    //创建Intent跳转主界面2中
    Intent intent=new Intent(MainActivity.this,Main2Activity.class);
    startActivity(intent);      //启动Intent
}
```

（7）运行本实例，将显示如图 6.20 所示。从右向左滑动 3 次将显示如图 6.21 所示的第 4 个界面，单击“开启私人订制”按钮将显示如图 6.22 所示的主界面 2 中。

图 6.20　引导第 1 页

图 6.21　引导第 4 页

图 6.22　主界面 2

6.3.3 翻页的标题栏（PagerTabStrip）

ViewPager 还有两个比较好的搭档，分别是 PagerTitleStrip 类与 PagerTabStrip 类，PagerTitleStrip 类直接继承自 ViewGroup 类，而 PagerTabStrip 是 PagerTitleStrip 的子类，所以这两个类也是容器类。二者都是可以实现在 ViewPager 页面上方显示标题文字，不同的是 PagerTitleStrip 只能显示单纯的标题文字，而 PagerTabStrip 类似于选项卡，标题文字下面有条横线，单击或者左右滑动都可以切换标题文字所对应的页面。

在 XML 布局文件中添加 PagerTabStrip 的时候，它必须是 ViewPager 标签的子标签。基本语法格式如下：

```
<!--ViewPager组件-->
<android.support.v4.view.ViewPager
    android:id="@+id/ID号"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <!--PagerTabStrip组件-->
    <android.support.v4.view.PagerTabStrip
        android:id="@+id/ID号"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </android.support.v4.view.PagerTabStrip>
</android.support.v4.view.ViewPager>
```

下面对 PagerTabStrip 类的常用方法进行介绍。

- setDrawFullUnderline: 参数为布尔类型，true 为显示标题文字底部长横线，false 为不显示。
- setTabIndicatorColor：设置标题文字指示条的颜色。
- setTextColor：设置标题文字的颜色。
- setTextSize：设置标题文字的大小。

在标题栏中显示文字时需要重写 PagerAdapter 中的 getPageTitle() 方法，代码示例如下：

```
//设置标题文字
@Override
public CharSequence getPageTitle(int position) {
    return tabList.get(position);
}
```

下面通过一个实例来演示 PagerTabStrip 的具体应用。

例 6.7 可以翻页的标题栏

在 Android Studio 中创建 Module，名称为“PagerTabStrip”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 ViewPager 组件并且在该组件中添加 1 个 PagerTabStrip 组件用于显示标题文字。具体代码如下：

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

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--设置ViewPager组件-->
    <android.support.v4.view.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <!--设置PagerTabStrip组件-->
        <android.support.v4.view.PagerTabStrip
            android:id="@+id/pagerTabStrip"
            android:layout_width="wrap_content"
            android:layout_height="50dp">
        </android.support.v4.view.PagerTabStrip>
    </android.support.v4.view.ViewPager>
</RelativeLayout>
```

（2）在 res/layout 目录中分别创建名称为 layout1、layout2、layout3、layout4 的 xml 文件，然后在每个文件的布局管理器中添加 1 个 ImageView 组件用于显示页面内容。layout1.xml 布局代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
<!--显示页面中的图片-->
    <ImageView
        android:layout_width="300dp"
        android:layout_height="450dp"
        android:layout_gravity="center_horizontal"
        android:src="@mipmap/img1" />
</LinearLayout>
```

（3）打开主活动 MainActivity.java 文件，修改默认生成的代码，首先让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义所需的全局变量并在 onCreate() 方法中加载需要显示在页面中的布局文件，最后将所有页面与相对应的标题文字添加在数组列表中并且为 ViewPager 指定所使用的适配器。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    private List<View> pageList = new ArrayList<View>();    //存放页面
    private List<String> tabList = new ArrayList<String>();//存放标题文字
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);       //设置全屏显示
        //获取ViewPager组件
        ViewPager  viewPager = (ViewPager) findViewById(R.id.viewPager);
        //加载4个页面的布局文件
        View view_1 = LayoutInflater.from(getApplicationContext()).
                inflate(R.layout.layout1,null);
        View view_2 = LayoutInflater.from(getApplicationContext()).
                inflate(R.layout.layout2,null);
        View view_3 = LayoutInflater.from(getApplicationContext()).
                inflate(R.layout.layout3,null);
        View view_4 = LayoutInflater.from(getApplicationContext()).
                inflate(R.layout.layout4,null);
        //添加ViewPage的4个页面
        pageList.add(view_1);
        pageList.add(view_2);
        pageList.add(view_3);
        pageList.add(view_4);
        //添加标题文字
        tabList.add("Android 精彩编程200例");
        tabList.add("Java 精彩编程200例");
        tabList.add("ASP.NET 精彩编程200例");
        tabList.add("C# 精彩编程200例");
        //设置适配器
        viewPager.setAdapter(adapter);
    }
}
```

（4）创建 PagerAdapter 对象并重写所需要的 5 个方法来实现通过标题栏切换页面的功能。具体代码如下：

```
PagerAdapter adapter=new PagerAdapter() {
    @Override
    public int getCount() {          //获取页面个数
        return pageList.size();      //返回页面数量
    }
    //确定页面视图是否与返回的对象相关联
    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }
    //从视图容器中移除指定位置的页面
    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        container.removeView(pageList.get(position));
    }
```

```
16      //返回当前所显示的视图
17      @Override
18      public Object instantiateItem(ViewGroup container, int position) {
19          container.addView(pageList.get(position));
20          return pageList.get(position);
21      }
22      //设置标题文字
23      @Override
24      public CharSequence getPageTitle(int position) {
25          return tabList.get(position);
26      }
27  };
```

（5）运行本实例，将显示如图 6.23 所示的运行效果。单击标题栏或滑动页面将显示如图 6.24 所示的运行效果。

图 6.23　显示第 1 个页面

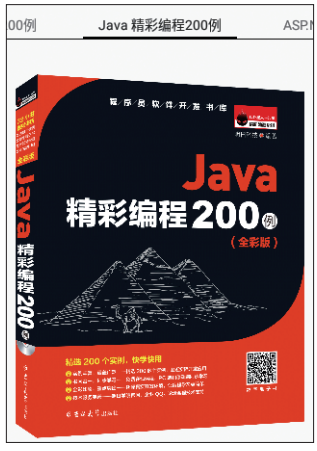


图 6.24　显示第 2 个页面

6.4 通用组件

在 Android 中，提供了用于为其他组件添加滚动条的滚动视图，使用 ScrollView 表示。另外，还提供了选项卡，它主要涉及 3 个组件，分别为 TabHost、TabWidget 和 FrameLayout。其中，TabHost 表示承载选项卡的容器；TabWidget 表示显示选项卡栏，主要用于当用户选择一个选项卡时，向父容器对象 TabHost 发送一个消息，通知 TabHost 切换到对应的页面；FrameLayout 是用于指定选项卡内容的。其次还有搜索框（SearchView）用于搜索相关数据，下面将分别进行介绍。

6.4.1 滚动视图

在默认的情况下，当窗体中的内容比较多而一屏显示不下时，超出的部分将不能被用户所看到。因为 Android 的布局管理器本身没有提供滚动屏幕的功能。如果要让其滚动，就需要使用滚动视图

（ScrollView），这样用户可以通过滚动屏幕查看完整的内容。例如，今日头条的新闻界面就应用了滚动视图（如图 6.25 所示），以及 QQ 的聊天窗口也应用了滚动视图（如图 6.26 所示）。

图 6.25　今日头条的新闻界面

图 6.26　QQ 的聊天窗口

滚动视图是 android.widget.FrameLayout（帧布局管理器）的子类。因此，在滚动视图中，可以添加任何想要放入其中的组件。但是，一个滚动视图中，只能放置一个组件。如果想要放置多个，可以在滚动视图中放置一个布局管理器，再将要放置的其他多个组件放置到该布局管理器中。在滚动视图中，使用比较多的是线性布局管理器。

说明 滚动视图ScrollView只支持垂直滚动。如果想要实现水平滚动条，可以使用水平滚动视图（HorizontalScrollView）来实现。

在 Android 中，可以使用两种方法向屏幕中添加滚动视图，一种是通过在 XML 布局文件中使用 <ScrollView> 标记添加，另一种是在 Java 文件中，通过 new 关键字创建出来。下面分别进行介绍。

1．在 XML 布局文件中添加

在 XML 布局文件中添加滚动视图，比较简单，只需要在要添加滚动条的组件外面使用下面的布局代码添加即可。

```
<ScrollView
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <!-- 要添加滚动条的组件 -->
</ScrollView>
```

例如，要为一个显示公司简介的 TextView 文本框添加滚动条，可以使用下面的代码。

```
01  <ScrollView
02      android:id="@+id/scrollView1"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
<TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:textSize="20sp"
    android:text="@string/content" />
</ScrollView>
```

2．通过 new 关键字创建

在 Java 代码中，通过 new 关键字创建滚动视图需要经过以下 3 个步骤。

（1）使用构造方法 ScrollView(Context context) 就可以创建一个滚动视图。

（2）创建或者获取需要添加滚动条的组件，并应用 addView() 方法将其添加到滚动视图中。

（3）将滚动视图添加到整个布局管理器中，用于显示该滚动视图。

下面通过一个具体的实例来介绍如何通过 new 关键字创建滚动视图。

例 6.8 为编程词典目录添加垂直滚动条

在 Android Studio 中创建 Module，名称为“ScrollView”，实现本实例的具体步骤如下：

（1）在 res/values/Strings.xml 文件中，添加一个名称为 cidian 的字符串资源，关键代码如下：

```
<resources>
    <string name="app_name">编程词典目录滚动视图</string>
    <string name="cidian">Java Web  编程词典（个人版）主干目录
入门训练营\n
  第1部分   从零开始\n
      第1课   第 1 课  搭建开发环境\n
          第1讲   课堂讲解\n
          第2讲   照猫画虎——基本功训练\n
          第3讲   情景应用——拓展与实践\n
      第2课   第 2 课  JSP中的Java程序\n
          第4讲   课堂讲解\n
          第5讲   照猫画虎——基本功训练\n
          第6讲   情景应用——拓展与实践\n
      第3课   第 3 课  HTML语言与CSS样式\n
          第7讲   课堂讲解\n
          第8讲   照猫画虎——基本功训练\n
          第9讲   情景应用——拓展与实践\n
      第4课   第 4 课  JavaScript脚本语言\n
          第10讲   课堂讲解\n
          第11讲   照猫画虎——基本功训练\n
          第12讲   情景应用——拓展与实践\n
      第5课   第 5 课  掌握JSP语法\n
          第13讲   课堂讲解\n
          第14讲   照猫画虎——基本功训练\n
```

```
25              第15讲    情景应用—拓展与实践\n
26          第6课    第 6 课   使用JSP内置对象\n
27              第16讲    课堂讲解\n
28              第17讲    照猫画虎—基本功训练\n
29              第18讲    情景应用—拓展与实践\n</string>
30  </resources>
```

（2）修改 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，为其设置 id，并且将默认添加的 TextView 组件删除，修改后的代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <LinearLayout
03      xmlns:android="http://schemas.android.com/apk/res/android"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:id="@+id/ll"
06      android:orientation="vertical"
07      android:layout_width="match_parent"
08      android:layout_height="match_parent"
09      tools:context=".MainActivity">
10  </LinearLayout>
```

（3）在 MainActivity 类的 onCreate() 方法中，首先获取布局文件中添加的线性布局管理器（linearLayout），然后创建一个滚动视图（scrollView）和一个新的布局管理器（linearLayout2），在默认布局管理器中添加滚动视图组件，然后在滚动视图中添加新创建的布局管理器，具体代码如下：

```
01  public class MainActivity extends AppCompatActivity {
02      //定义linearLayout为默认布局管理器，linearLayout2为新建布局管理器
03      LinearLayout linearLayout, linearLayout2;
04      ScrollView scrollView;                                    //定义滚动视图组件
05      @Override
06      protected void onCreate(Bundle savedInstanceState) {
07          super.onCreate(savedInstanceState);
08          setContentView(R.layout.activity_main);
09          linearLayout = (LinearLayout) findViewById(R.id.ll);    //获取布局管理器
10          linearLayout2 = new LinearLayout(MainActivity.this);  //创建一个新的布局管理器
11          linearLayout2.setOrientation(LinearLayout.VERTICAL);  //设置为纵向排列
12          scrollView = new ScrollView(MainActivity.this);       //创建滚动视图组件
13          linearLayout.addView(scrollView);                  //默认布局中添加滚动视图组件
14          scrollView.addView(linearLayout2);                 //滚动视图组件中添加新建布局
15      }
16  }
```

（4）在 MainActivity 类的 onCreate() 方法中，创建 ImageView 对象和 TextView 对象分别用于存放词典图片和文本目录，并且将这两个对象添加到新的布局管理器（linearLayout2）中。具体代码如下：

```
01  ImageView imageView = new ImageView(MainActivity.this);//创建ImageView组件
02  imageView.setImageResource(R.mipmap.cidian);          //为ImagView添加图片
03  TextView textView = new TextView(MainActivity.this);  //创建TextView组件
04  textView.setText(R.string.cidian);                    //为TextView添加文字
05  linearLayout2.addView(imageView);                     //新建布局中添加ImageView组件
06  linearLayout2.addView(textView);                      //新建布局中添加TextView组件
```

（5）运行本实例，将显示如图 6.27 所示的运行结果。

图 6.27　为编程词典目录添加垂直滚动条

说明　在默认情况下滚动条不显示，向上拖动后方可显示，停止拖动后滚动条消失。

5.4.2 选项卡

选项卡用于实现一个多标签页的用户界面，通过它可以将一个复杂的对话框分割成若干个标签页，实现对信息的分类显示和管理。使用该组件不仅可以使界面简洁大方，还可以有效地减少窗体的个数。例如，微信的表情商店界面（如图 6.28 所示），以及百度贴吧的进吧界面（如图 6.29 所示）。

图 6.28　微信的表情商店界面

图 6.29　百度贴吧的进吧界面

在 Android 中，使用选项卡，不能通过某一个具体的组件在 XML 布局文件中添加。通常需要按照以下步骤来实现。

（1）在布局文件中添加实现选项卡所需的 TabHost、TabWidget 和 FrameLayout 组件。

（2）编写各标签页中要显示内容所对应的 XML 布局文件。

（3）在 Activity 中，获取并初始化 TabHost 组件。

（4）为 TabHost 对象添加标签页。

下面通过一个具体的实例来说明选项卡的应用。

例 6.9 模拟微信表情商店的选项卡

在 Android Studio 中创建 Module，名称为“TabHost”，在该 Module 中实现本实例，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器删除，然后添加实现选项卡所需的 TabHost、TabWidget 和 FrameLayout 组件。具体的步骤是：首先添加一个 TabHost 组件，然后在该组件中添加线性布局管理器，并且在该布局管理器中添加一个作为标签组的 TabWidget 和一个作为标签内容的 FrameLayout 组件，最后删除内边距。在 XML 布局文件中添加选项卡的基本代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@android:id/tabhost"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
        </FrameLayout>
    </LinearLayout>
</TabHost>
```

说明 在应用XML布局文件添加选项卡时，必须使用系统的id来为各组件指定id属性，否则将出现异常。

（2）在 res/layout 目录下单击右键 New → Layout resource file 创建一个 XML 布局文件，名称为 tab1.xml，用于指定第一个标签页中要显示的内容，具体代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/linearlayout1"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@mipmap/biaoqing_left"/>
</LinearLayout>
```

（3）创建第二个 XML 布局文件，名称为 tab2.xml，用于指定第二个标签页中要显示的内容，具体代码如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/framelayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:id="@+id/linearlayout2"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:src="@mipmap/biaoqing_right"/>
    </LinearLayout>
</FrameLayout>
```

说明 在本实例中，除了需要编写名称为tab1.xml的布局文件外，还需要编写名称为tab2.xml的布局文件，用于指定第二个标签页中要显示的内容。

（3）在 MainActivity 中，首先获取并初始化 TabHost 组件，然后为 TabHost 对象添加标签页，这里共添加了两个标签页，一个用于精选表情，另一个用于投稿表情。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    private TabHost tabHost;                                    //声明TabHost组件的对象
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tabHost=(TabHost)findViewById(android.R.id.tabhost); //获取TabHost对象
        tabHost.setup();                                        //初始化TabHost组件
        //声明并实例化一个LayoutInflater对象
        LayoutInflater inflater = LayoutInflater.from(this);
        inflater.inflate(R.layout.tab1, tabHost.getTabContentView());
        inflater.inflate(R.layout.tab2,tabHost.getTabContentView());
        tabHost.addTab(tabHost.newTabSpec("tab1")
```

```
14                  .setIndicator("精选表情")
15                  .setContent(R.id.linearlayout1));              //添加第一个标签页
16          tabHost.addTab(tabHost.newTabSpec("tab2")
17                  .setIndicator("投稿表情")
18                  .setContent(R.id.framelayout)) ;               //添加第二个标签页
19      }
20  }
```

运行本实例，将显示如图 6.30 所示的运行结果。

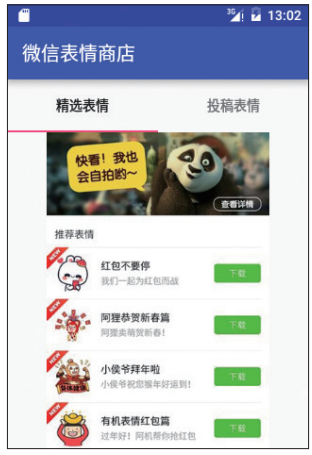


图 6.30　模拟微信表情商店的选项卡

6.4.3 搜索框 (SearchView)

SearchView 是 Android 原生的搜索框组件，它在默认的情况下显示着 1 个搜索图标，单击图标后将展开搜索框。当用户在搜索框内输入文字时可以通过监听器监听用户输入，当用户输入完成后也可以通过监听器来执行搜索功能。例如，在手机的文件管理器中搜索指定文件时，就可以使用该组件来实现，如图 6.31 所示。

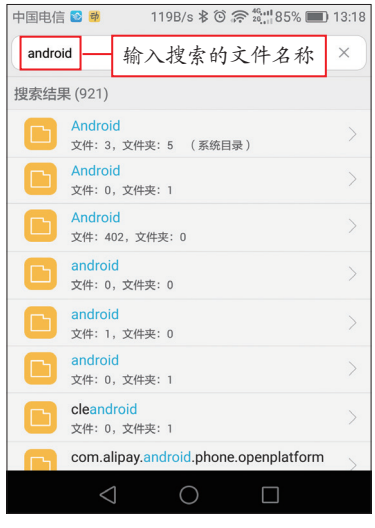


图 6.31　文件管理器中搜索指定文件

在 XML 布局文件中添加搜索框的基本语法格式如下：

```
<SearchView
    android:id="@+id/ID号"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</SearchView>
```

下面对 SearchView 的常用方法进行介绍。

◆ setIconifiedByDefault: 设置搜索图标是否显示在搜索框内。

◆ setQueryHint：设置搜索框内默认显示的提示文字。

◆ setSubmitButtonEnabled：启用搜索框内查询内容不为空时显示提交按钮。

◆ setMaxWidth：设置搜索框组件的最大宽度。

如果想让搜索框实现搜索功能时，还需要为搜索框设置事件监听器。例如，为名称 searchView 的搜索框添加事件监听器，可以使用下面的代码：

```
01  //获取搜索框组件
02  SearchView searchView= (SearchView) findViewById(R.id.searchView);
03  //设置搜索框事件监听器
04  searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
05      //当用户提交查询时调用该方法
06      @Override
07      public boolean onQueryTextSubmit(String query) {
08          return false;
09      }
10      //当用户更改查询文本时调用该方法
11      @Override
12      public boolean onQueryTextChange(String newText) {
13          return false;
14      }
15  });
```

下面通过一个实例演示搜索框的具体应用。

例 6.10 在列表中搜索指定的内容

在 Android Studio 中创建 Module，名称为“SearchView”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器并将 TextView 组件删除，然后先添加 1 个 SearchView 组件用于显示搜索框，再添加 1 个 ListView 组件用于显示列表内容。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.mingrisoft.MainActivity">
    <!--搜索框-->
    <SearchView
        android:id="@+id/searchView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:iconifiedByDefault="false"
        android:queryHint="请输入搜索内容" />
    <!--列表视图-->
    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />
</LinearLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，首先让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义 1 个字母数组用于显示在列表当中。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    //设置列表显示的字母数组
    private String[] content={"a","b","c","d","e","f","g","h","i","j"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

（3）在主活动的 onCreate() 方法中，首先获取布局文件中的搜索框（SearchView）组件与列表（ListView）组件，然后为 ListView 设置适配器用于显示字母内容，最后启用 ListView 的文本过滤器。关键代码如下：

```
//获取搜索框组件
SearchView searchView = (SearchView) findViewById(R.id.searchView);
//获取ListView列表组件
final ListView listView = (ListView) findViewById(R.id.listView);
//设置ListView列表适配器
listView.setAdapter(new ArrayAdapter<String>
        (this, android.R.layout.simple_list_item_1, content));
```

```
08  listView.setTextFilterEnabled(true);          //启用文本过滤器
```

（4）设置搜索框事件监听器，首先在 onQueryTextSubmit() 方法中通过 Toast 显示当前搜索的字母，然后在 onQueryTextChange() 方法中实现通过 ListView 列表显示搜索内容。关键代码如下：

```
01  //设置搜索框事件监听器
02  searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
03      //当用户提交查询时调用该方法
04      @Override
05      public boolean onQueryTextSubmit(String query) {
06          Toast.makeText(MainActivity.this, "搜索字母:"+query,
07                  Toast.LENGTH_SHORT).show();
08          return false;
09      }
10      //当用户更改查询文本时调用该方法
11      @Override
12      public boolean onQueryTextChange(String newText) {
13          if (!TextUtils.isEmpty(newText)) {          //如果搜索框内容长度不为0时
14              listView.setFilterText(newText);        //列表显示搜索内容
15          } else {
16              listView.clearTextFilter();             //否则清除过滤
17          }
18          return false;
19      }
20  });
```

（5）运行本实例，将显示如图 6.32 所示的运行结果。在搜索框中输入指定的字母列表将自动显示搜索内容如图 5.33 所示。单击“搜索”按钮将显示如图 6.34 所示的运行结果。

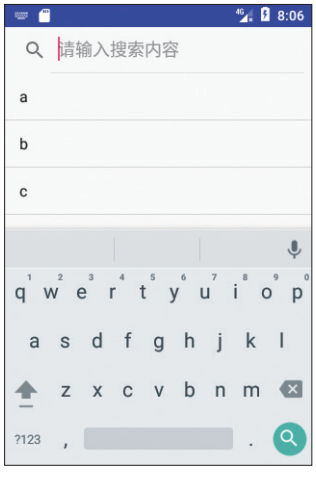


图 6.32　默认效果图

图 6.33　自动显示搜索内容

图 6.34　提示搜索内容

6.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。

第 7 章

Android 权限机制与支持库的应用

Android 是一个权限分隔的操作系统，其中每个应用都有其独特的系统标识（Linux 用户 ID 和组 ID）。Android 支持库提供了许多未内置于框架的功能。这些库提供向后兼容版本的新功能、框架中未包含的实用 UI 元素，以及应用可以利用的一系列实用程序与控件。本章将对 Android 的权限机制与支持库中的常用控件进行详细介绍。

7.1 Android 权限机制

在 Android 系统 6.0 版本之前，安装 App 时会提示用户此 App 都需要使用哪些权限，如图 7.1 所示。但是用户不可以单独对某项权限进行授权或者是拒绝，如果用户安装了该 App，就表示用户已经接受了该 App 对这些权限的使用。从 Android 系统 6.0 版本开始，Android 使用了新的权限管理机制，将 App 可以使用的权限分成了两类，一类是普通权限，例如，设置手机振动或者是访问网络等。另一类是危险权限，例如，开启摄像头或者是获取地理位置等。对于普通权限，仍然和以前一样，开发者只需要在 AndroidManifest 配置文件中添加所需的权限即可，用户只要同意安装就表示已经接受该 App 使用这些权限。对于危险权限来说，因为涉及到个人的隐私，所以不仅需要在 AndroidManifest 配置文件中添加权限，还需要在运行时请求用户授权此类权限，如图 7.2 所示。此时用户可以选择允许或拒绝，如果拒绝某项权限，该 App 将无法使用该权限的功能。

图 7.1　Android 6.0 之前需要使用的权限

图 7.2　Android 6.0 之后授权危险权限

运行 App 时权限的申请步骤如下：

（1）判断当前手机系统版本是否大于或等于 6.0 版本。可以使用以下代码：

```
if(Build.VERSION.SDK_INT>= Build.VERSION_CODES.M){
   Toast.makeText(this, "大于等于6.0", Toast.LENGTH_SHORT).show();
}else {
   Toast.makeText(this, "小于6.0", Toast.LENGTH_SHORT).show();
}
```

（2）检查是否授权了某个权限，这里以读取联系人权限为例。可以使用以下代码：

```
//Manifest.permission.READ_CONTACTS为查询的读取联系人权限
if (checkSelfPermission(Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {
    //没有权限，可以在这里申请权限。
}else{
    //权限已经申请。
}
```

（3）申请一组需要的权限可以是 1 个或者多个以数组的形式添加。可以使用以下代码：

```
requestPermissions(new String[]{Manifest.permission.READ_CONTACTS} ,1);
```

（4）用户处理完授权操作后，查看申请结果的回调。可以使用以下代码：

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[] grantResults) {
    switch (requestCode) {
        case 1: {       //判断授权码为1的申请权限数组
            //判断授权结果是否已经授权
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                //用户已同意权限
            } else {
                //用户已拒绝权限
            }
            return;
        }
    }
}
```

参数含义如表 7.1 所示。

表 7.1　参数含义

参 数 名 称	描　　述
int requestCode	申请权限的请求代码，会在回调 onRequestPermissionsResult() 方法时返回，用来判断是哪个授权申请的回调

续表

参 数 名 称	描　　述
String permissions[]	申请需要权限的数组
int[] grantResults	授权结果数组，对应 permissions，通过 PackageManager 判断是否已经授权

下面通过一个具体的实例演示 Android 系统 6.0 版本申请权限的应用。

例 7.1 模拟申请定位权限

在 Android Studio 中创建 Module，名称为“ApplyForPermission”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 Button 组件用于触发申请定位权限的方法。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--申请定位权限按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:onClick="onPermission"
        android:text="申请定位权限" />
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，在 onCreate() 方法的下面创建 1 个 onPermission() 方法，该方法用于处理申请权限按钮的单击事件。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //创建申请定位权限按钮的单击事件
    public void onPermission(View view){
    }
}
```

（3）在 onPermission() 方法中，首先判断当前手机系统的版本是否大于或者等于 Android 6.0 的系统版本，如果是大于或者等于 Android 6.0 的系统版本，则再判断是否没有授权定位权限，如果没有授权将申请该定位权限。关键代码如下：

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    //判断是否没有授权定位权限
    if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {
        //申请定位权限
        requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
    } else {
        Toast.makeText(this, "定位权限已经授权！", Toast.LENGTH_SHORT).show();
    }
}
```

（4）调用 onRequestPermissionsResult() 申请权限的回调方法，在该方法中首先判断用户是否同意了授权定位的权限，如果同意了将通过 Toast 进行提示，否则通过对话框的方式提示用户该权限的重要性，并提醒用户可以在设置中心进行权限的授权操作。具体代码如下：

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[] grantResults) {
switch (requestCode) {
    case 1: {       //判断授权码为1的申请权限数组
        //判断授权结果是否已经授权
        if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "用户已经同意授权定位的权限！",
                    Toast.LENGTH_SHORT).show();
        } else {
            new AlertDialog.Builder(this)
                    .setTitle("提示信息")
                    .setMessage("当前应用缺少必要权限，该功能暂时无法使用。" +
                            "如若需要，请单击【确定】按钮前往设置中心进行权限授权。")
                    .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                        }
                    })
                     .setPositiveButton("确定", new DialogInterface.OnClickListener()
{
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            //通过Intent跳转设置中心界面
                            Intent intent = new Intent(
                                Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                            intent.setData(Uri.parse("package:" + getPackageName()));
                            startActivity(intent);
                        }
                    }).show();
```

```
31                }
32                return;
33            }
34        }
35    }
```

（5）打开 manifests 目录中的 AndroidManifest.xml 文件，在该文件中的 application 节点上面添加定位权限的代码。关键代码如下：

```
01    <!--定位权限-->
02    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

（6）运行本实例，在默认显示的界面中单击“申请定位权限“按钮将显示如图 7.3 所示的允许定位权限的对话框，若在该对话框中单击“拒绝”按钮将显示如图 7.4 所示的提示用户该权限的重要性的对话框，若在图 7.3 的界面中单击“允许”按钮将显示如图 7.5 所示的定位权限授权成功的消息提示。

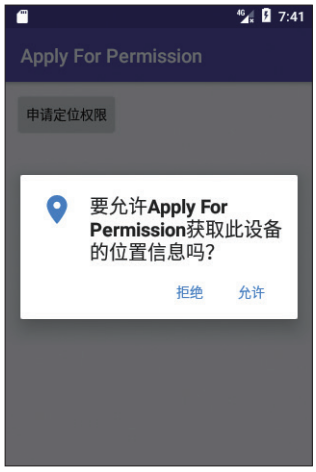


图 7.3　允许定位权限

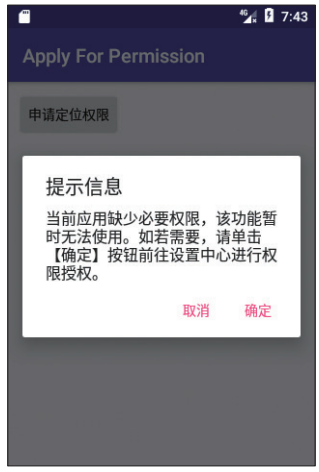


图 7.4　该权限重要性的对话框

图 7.5　定位权限授权成功

7.2 Android 支持库的常用控件

7.2.1 工具栏（ToolBar）

在 Android Studio 中默认创建 1 个 App 时，顶部都会默认添加导航栏 ActionBar 组件。但是 ActionBar 存在一些缺点，如使用不灵活难以扩展等问题。因此在 Android 5.0 时 ToolBar 诞生了，由于它高度的可定制性、灵活性、具有 Material Design 风格等优点，越来越多的 App 开始使用 ToolBar，例如，美团 App 中顶部的工具栏就可以通过 ToolBar 来实现，如图 7.6 所示。由于仍有一部分 Android 用户的手机版本低于 5.0，所以，ToolBar 也被放进了 support v7 包当中，方便低版本系统用户的使用。

图 7.6　美团 App 主界面

为了兼容以前的系统版本，ActionBar 仍然需要保留，但是要想使用 ToolBar 就需要先将 ActionBar 关闭。ToolBar 的使用方法如下：

（1）打开 res/values 目录中的 styles.xml 文件，在该文件中将主题样式修改为没有 ActionBar。关键代码如下：

```
<!--Base application theme.-->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
```

（2）打开 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 ToolBar 组件并在该组件中添加 1 个 TextView 组件，用于显示“我是 ToolBar”文字。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--工具栏ToolBar-->
    <android.support.v7.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorAccent">
        <!--显示文字-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
17              android:text="我是ToolBar" />
18      </android.support.v7.widget.Toolbar>
19  </RelativeLayout>
```

在 ToolBar 组件中可以放置其他组件也是它自身的一个特点，因为 ToolBar 实际上是一个 ViewGroup，因此可以在其内部放入子 View。运行效果如图 7.7 所示。

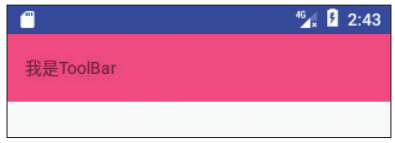


图 7.7　简单实现 ToolBar

ToolBar 之所以比 ActionBar 强大，是因为 ToolBar 有着非常灵活的属性方法。ToolBar 的常用方法如表 7.2 所示。

表 7.2　ToolBar 常用的属性方法

属 性 方 法	描　　述
setNavigation()	设置左侧的导航图标
setLogo()	设置工具栏图标
setTitle()	设置主标题文字
setSubtitle()	设置副标题文字
setNavigationOnClickListener()	设置导航图标的单击事件监听器
setOnMenuItemClickListener()	设置菜单项的单击事件

下面通过一个具体的实例演示工具栏（ToolBar）的应用。

例 7.2　ToolBar 常用功能的实现

在 Android Studio 中创建 Module，名称为“ToolBar”，具体步骤如下：

（1）打开 res/values 目录中的 styles.xml 文件，在该文件中将主题样式修改为没有 ActionBar。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 Toolbar 组件。

（3）在 res 目录中首先创建 1 个名称为“menu”的文件夹，然后在该文件夹中创建 1 个 toolbar_menu.xml 文件，在该文件中实现工具栏（Toolbar）中的选项菜单。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <menu xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto">
04      <item android:id="@+id/menu1"
05          android:title="菜单1"/>
06      <item android:id="@+id/menu2"
07          android:title="菜单2"/>
08      <item android:id="@+id/menu3"
```

```
        android:title="菜单3"/>
</menu>
```

（4）打开主活动 MainActivity.java 文件，让该类实现 View.OnClickListener 与 Toolbar.OnMenuItemClickListener 接口并实现接口中相应的方法。具体代码如下：

```
public class MainActivity extends AppCompatActivity implements
        View.OnClickListener,Toolbar.OnMenuItemClickListener{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //单击事件方法
    @Override
    public void onClick(View v) {
    }
    //菜单项的单击事件方法
    @Override
    public boolean onMenuItemClick(MenuItem item) {
        return false;
    }
}
```

（5）在 onCreate() 方法中，首先获取布局文件中的 ToolBar 组件，然后设置工具栏（Toolbar）的相关属性，如工具栏图标、导航图标、标题文字等，最后为工具栏（Toolbar）分别设置导航图标的单击事件监听器与菜单项的单击事件监听器。关键代码如下：

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolBar);      //获取工具栏（Toolbar）
toolbar.setLogo(R.drawable.ic_launcher);                     //设置工具栏图标
toolbar.setNavigationIcon(R.drawable.ic_launcher_round);     //设置导航图标
toolbar.setTitle("我是主标题！");                              //设置主标题文字
toolbar.setTitleTextColor(Color.RED);                        //设置主标题文字颜色
toolbar.setSubtitle("我是副标题！");                           //设置副标题文字
toolbar.setSubtitleTextColor(Color.BLACK);                   //设置副标题文字颜色
setSupportActionBar(toolbar);                                //设置工具栏为Toolbar
toolbar.setNavigationOnClickListener(this);                  //设置导航图标单击事件
toolbar.setOnMenuItemClickListener(this);                    //设置菜单项的单击事件
```

（6）重写 onCreateOptionsMenu() 方法，在该方法中解析创建的选项菜单资源文件，具体代码如下：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.toolbar_menu,menu);     //解析选项菜单资源
    return true;
}
```

（7）重写 onMenuItemClick() 方法，在该方法中判断菜单中的哪一项被单击并通过 Toast 进行提示。具体代码如下：

```
@Override
public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()){
        case R.id.menu1:
            Toast.makeText(this, "菜单1被单击了！", Toast.LENGTH_SHORT).show();
            break;
        case R.id.menu2:
            Toast.makeText(this, "菜单2被单击了！", Toast.LENGTH_SHORT).show();
            break;
        case R.id.menu3:
            Toast.makeText(this, "菜单3被单击了！", Toast.LENGTH_SHORT).show();
            break;
    }
    return false;
}
```

（8）重写 onClick() 方法，在该方法中通过 Toast 提示导航图标被单击了即可。

（9）运行本实例，单击左上角导航按钮将显示如图 7.8 所示的界面。单击右上角选项菜单将显示如图 7.9 所示的界面。在选项菜单中单击某个选项后项将显示如图 7.10 所示的提示信息。

图 7.8　导航图标提示

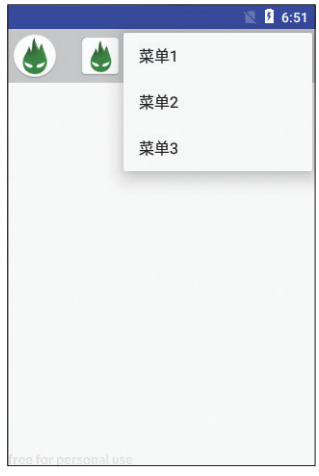


图 7.9　显示选项菜单

图 7.10　单击菜单项提示

7.2.2 折叠标题栏（CollapsingToolbarLayout）

CollapsingToolbarLayout 用于美化 Toolbar 的运行效果，它是由 Design Support 库提供的。CollapsingToolbarLayout 被设计用于 AppBarLayout 的子布局，AppBarLayout 是一个垂直的 LinearLayout，只是为了实现交互动画效果它增加了一些滚动特性。AppBarLayout 只有作为 CoordinatorLayout 的直接子布局才可以正常的工作，如果 CoordinatorLayout 包含了一个不同的 ViewGroup，滚动功能将无法实现。通过 CollapsingToolbarLayout 与它的兄弟们就可以实现折叠式工具栏的运行效果了，例如，哔哩哔

哩App的视频详情界面就可以通过该技术来实现，如图7.11所示，手指在屏幕中向上滑动将显示如图7.12所示的工具栏折叠效果。

图 7.11　视频详情界面

图 7.12　工具栏折叠效果

CollapsingToolbarLayout 所支持的常用属性如表 7.3 所示。

表 7.3　CollapsingToolbarLayout 支持的常用属性

XML 属性	描　　述
app:collapsedTitleGravity	工具栏折叠后标题文字显示在指定位置，默认值为 left
app:expandedTitleGravity	工具栏完全展开后标题文字显示在指定位置，默认值为 left+bottom
app:contentScrim	工具栏折叠后 Toolbar 显示的背景颜色
app:layout_scrollFlags	设置滚动的方式，需要滚动必须设置 scroll 值、exitUntilCollapsed 值向上滚动实现折叠效果、enterAlways 值向下滚动时立即显示 View、enterAlwaysCollapsed 值当 View 设置最小高度时该 View 以最小高度限定进入

由于 CollapsingToolbarLayout 是 Design Support 库所提供的，所以需要在 gradle 文件中添加依赖库代码。具体代码如下：

```
compile 'com.android.support:design:插入版本号'
```

下面通过一个具体的实例演示折叠式工具栏（CollapsingToolbarLayout）的应用。

例 7.3　折叠式工具栏的实现

在 Android Studio 中创建 Module，名称为“CollapsingToolbar”，具体步骤如下：

（1）打开 build.gradle（Module: CollapsingToolbar）文件，在 dependencies 节点中添加 CollapsingToolbar 的依赖库代码。

（2）打开 res/values 目录中的 styles.xml 文件，在该文件中将主题样式修改为没有 ActionBar。

（3）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局

管理器修改为 CoordinatorLayout 布局管理器并将 TextView 组件删除，然后在该布局管理器中添加 1 个 AppBarLayout。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--设置AppBarLayout-->
    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </android.support.design.widget.AppBarLayout>
</android.support.design.widget.CoordinatorLayout>
```

（4）在 AppBarLayout 布局内添加 1 个 CollapsingToolbarLayout 并在该布局中添加 1 个 ImageView 组件用于显示展开工具栏后的图片，然后再添加 1 个 Toolbar 组件用于显示工具栏。具体代码如下：

```
<!--设置CollapsingToolbarLayout-->
<android.support.design.widget.CollapsingToolbarLayout
    android:id="@+id/collapsingToolbarLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_scrollFlags="scroll|exitUntilCollapsed">
    <!--展开工具栏后显示的图片-->
    <ImageView
        android:id="@+id/main.backdrop"
        android:layout_width="wrap_content"
        android:layout_height="300dp"
        android:scaleType="centerCrop"
        android:src="@mipmap/img"
        app:layout_collapseMode="parallax" />
    <!--工具栏Toolbar-->
    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?android:attr/actionBarSize"
        app:layout_collapseMode="pin" />
</android.support.design.widget.CollapsingToolbarLayout>
```

说明 ImageView组件中的app:layout_collapseMode=”parallax”属性表示在折叠的过程中图片产生错位偏移的视觉效果。而ToolBar组件中app:layout_collapseMode=”pin”属性表示在折叠的过程中该组件位置始终保持不变。

（5）在 AppBarLayout 布局的下面添加 1 个 NestedScrollView 组件，然后在该组件中添加 1 个 TextView 组件来实现滚动显示的文字。具体代码如下：

```
<!--滚动组件-->
<android.support.v4.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">
    <!--文本框组件，文字信息需要在资源文件中创建-->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/text"
        android:textSize="20sp" />
</android.support.v4.widget.NestedScrollView>
```

说明 NestedScrollView组件不仅像ScrollView那样使用滚动的方式查看屏幕以外的数据，还增加了响应滚动事件的功能。由于CoordinatorLayout已经可以响应滚动事件了，所以该布局中需要使用NestedScrollView或者是RecyclerView这样的滚动组件与折叠工具栏进行联动。该组件中的app:layout_behavior=”@string/appbar_scrolling_view_behavior”属性这里是指NestedScrollView组件时刻保持在AppBarLayout的下面，改变AppBarLayout内容位置时NestedScrollView组件可以一起滚动，起到联动的作用。

（6）打开主活动 MainActivity.java 文件，在 onCreate() 方法中首先设置界面中的工具栏为 ToolBar，然后获取折叠工具栏布局并设置标题文字与颜色。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);//获取工具栏（Toolbar）
        setSupportActionBar(toolbar);                          //设置工具栏为Toolbar
        //获取折叠工具栏布局
        CollapsingToolbarLayout collapsingToolbarLayout=
                (CollapsingToolbarLayout) findViewById(R.id.collapsingToolbarLayout);
        collapsingToolbarLayout.setTitle("Android 200例");//设置标题文字
        collapsingToolbarLayout.setExpandedTitleColor(
                Color.RED);                                    //设置未折叠时标题文字的颜色
        collapsingToolbarLayout.setCollapsedTitleTextColor(
                Color.WHITE);                                  //设置折叠后的标题文字颜色
        //设置折叠后工具栏的背景颜色
        collapsingToolbarLayout.setContentScrimColor(
                getResources().getColor(R.color.colorPrimary));
    }
}
```

（7）运行本实例，将显示如图 7.13 所示的运行结果。手指在屏幕中向上滑动后将显示如图 7.14 所示的工具栏折叠效果。

图 7.13　工具栏展开效果

图 7.14　工具栏折叠效果

7.2.3 卡片式布局（CardView）

CardView 是 Android 5.0 所推出的组件，它是用于实现卡片式布局的重要组件，由 appcompat-v7 库提供。CardView 是 FrameLayout 的子类，只是单独提供了圆角与阴影的效果，看起来立体感更强、更加美观。CardView 一般被使用在如 ListView、GridView、RecyclerView 等列表视图的子项（item）布局中。实现 CardView 的基本步骤如下：

（1）由于 CardView 是 appcompat-v7 库所提供的，所以在布局管理器中使用该组件需要先在 build.gradle 文件的 dependencies 节点中添加 CardView 依赖库的代码。具体代码如下：

```
compile 'com.android.support:cardview-v7:插入版本号'
```

（2）打开 res/layout 目录下的布局文件 activity_main.xml，在布局管理器中添加 CardView 组件。具体代码如下：

```
<android.support.v7.widget.CardView
    android:id="@+id/cardView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</android.support.v7.widget.CardView>
```

CardView 需要为其设置一些属性才可以让它显示得更加美观，CardView 所支持的 XML 属性如表 7.4 所示。

表 7.4　CardView 支持的 XML 属性

XML 属性	描　述
app:cardBackgroundColor	卡片视图的背景颜色
app:cardCornerRadius	卡片视图的圆角半径
app:cardElevation	卡片视图阴影的大小

续表

XML 属性	描　　述
app: cardMaxElevation	卡片视图最大阴影的大小
app: cardPreventCornerOverlap	防止卡片内容边角的重叠
app: cardUseCompatPadding	保持内边距的计算方式与以前的版本相同

下面通过一个具体的实例演示卡片式布局（CardView）的应用。

例 7.4 CardView 的布局效果

在 Android Studio 中创建 Module，名称为“CardView”，具体步骤如下：

（1）打开 build.gradle（Module:CardView）文件，然后在该文件中的 dependencies 节点中添加依赖库的代码。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 CardView 组件并在该组件中添加 1 个水平线性布局管理器用于显示卡片式布局的内容。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--CardView组件-->
    <android.support.v7.widget.CardView
        android:id="@+id/cardView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp">
        <!--水平线性布局管理器-->
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="150dp"
            android:orientation="horizontal">
        </LinearLayout>
    </android.support.v7.widget.CardView>
</RelativeLayout>
```

（3）在水平线性布局管理器中，首先添加 1 个 ImageView 组件用于显示图片，然后添加 1 个垂直线性布局管理器并在该布局管理器中添加 2 个 TextView 组件用于显示文本信息。具体代码如下：

```
<!--图像组件-->
<ImageView
    android:layout_width="150dp"
```

```
    android:layout_height="match_parent"
    android:layout_margin="5dp"
    android:scaleType="centerCrop"
    android:src="@mipmap/android_img" />
<!--垂直线性布局管理器-->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="零基础快速入门"
        android:textColor="#000000"
        android:textSize="16sp"
        android:textStyle="bold" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="一学就会·一看就懂"
        android:textColor="#000000"
        android:textSize="12sp" />
</LinearLayout>
```

（4）打开主活动 MainActivity.java 文件，在 onCreate() 方法中首先获取 CardView 组件，然后为其分别设置圆角半径、阴影部分的大小以及卡片视图与内容的边距属性。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取布局管理器中的CardView组件
        CardView cardView = (CardView)findViewById(R.id.cardView);
        cardView.setRadius(10);                 //设置卡片视图的圆角半径
        cardView.setCardElevation(10);          //设置卡片视图阴影部分大小
        cardView.setContentPadding(5,5,5,5);    //设置卡片视图与内容的边距
    }
}
```

（5）运行本实例，使用 CardView 布局后的运行效果如图 7.15 所示。

图 7.15　CardView 布局后的运行效果

7.2.4 滑动菜单（DrawerLayout）

DrawerLayout 在 Android v4 包中实现了侧滑菜单效果的布局，它显示在屏幕的最左侧，默认情况下是隐藏的，当用户用手指从屏幕左侧向右侧滑动时该组件将显示，单击该布局外侧或者向原来的方向滑动时该组件消失，例如，图 7.16 所示的这款 App 就使用了该组件实现隐藏在左侧的菜单。

图 7.16　显示隐藏在左侧的菜单

DrawerLayout 是一个布局文件管理器，在这个布局当中允许放入 2 个组件或者是布局，第 1 个用于显示在主屏幕当中，第 2 个用于显示滑动菜单中的内容，在设置滑动菜单实现的内容时在该组件或布局中需要添加 android:layout_gravity=”start”属性将需要显示的内容放在布局的左侧。代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <android.support.v4.widget.DrawerLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawerLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!-- 界面主要内容布局区 -->
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </FrameLayout>
    <!-- 左侧弹出的内容 -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:orientation="vertical">
    </LinearLayout>
```

在使用 DrawerLayout 布局时可以使用 setDrawerListener() 方法为该布局设置监听器，根据监听器中的 4 个方法实现滑动菜单在不同的状态下来完成各种任务。例如，为名称为 drawerLayout 设置状态改变事件监听器，可以使用下面的代码：

```
//获取DrawerLayout布局
DrawerLayout drawerLayout= (DrawerLayout) findViewById(R.id.drawerLayout);
drawerLayout.setDrawerListener(new DrawerLayout.DrawerListener() {
    //当抽屉的位置发生变化时调用
    @Override
    public void onDrawerSlide(View drawerView, float slideOffset) {
    }
    //当抽屉已经处于完全打开的状态时调用
    @Override
    public void onDrawerOpened(View drawerView) {
    }
    //当抽屉已经完全关闭状态时调用
    @Override
    public void onDrawerClosed(View drawerView) {
    }
    //抽屉滑动状态改变时调用
    //状态值STATE_IDLE：闲置、STATE_DRAGGING：拖拽、STATE_SETTLING：固定的
    @Override
    public void onDrawerStateChanged(int newState) {
    }
});
```

下面通过一个具体的实例演示 DrawerLayout 的应用。

例 7.5 抽屉菜单

在 Android Studio 中创建 Module，名称为“DrawerLayout”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为 DrawerLayout 布局管理器，首先添加 1 个 FrameLayout 布局管理器用于显示主屏幕内容，然后添加 1 个 RelativeLayout 相对布局管理器并在该布局管理器中添加 1 个 ListView 组件用于显示左侧弹出的列表菜单。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawerLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--主屏幕显示的内容-->
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:text="我是主屏幕" />
    </FrameLayout>
    <!--左侧菜单弹出的内容-->
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="#87CEEB">
        <!--菜单列表-->
        <ListView
            android:id="@+id/listView"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
        </ListView>
    </RelativeLayout>
</android.support.v4.widget.DrawerLayout>
```

（2）打开主活动 MainActivity.java 文件，创建 1 个用于显示菜单名称的数组。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    //设置菜单名称数组
    private String[] item = new String[]{"菜单一", "菜单二", "菜单三", "菜单四"};
```

```
04
05      @Override
06      protected void onCreate(Bundle savedInstanceState) {
07          super.onCreate(savedInstanceState);
08          setContentView(R.layout.activity_main);
09      }
10  }
```

（3）在主活动的 onCreate() 方法中，首先获取布局文件中用于显示菜单的 ListView 组件，然后为该组件创建并设置相应的适配器，再获取布局文件中的 DrawerLayout 布局，最后为该布局设置状态变化监听器来实现抽屉打开与关闭时的消息提示。关键代码如下：

```
01  ListView listView = (ListView) findViewById(R.id.listView);
02  //创建适配器
03  ArrayAdapter adapter = new ArrayAdapter(this,
04          android.R.layout.simple_list_item_1, item);
05  listView.setAdapter(adapter);        //设置适配器
06  //获取DrawerLayout布局
07  DrawerLayout drawerLayout = (DrawerLayout) findViewById(R.id.drawerLayout);
08  //设置状态变化监听器
09  drawerLayout.addDrawerListener(new DrawerLayout.DrawerListener() {
10      //当抽屉的位置发生变化时调用
11      @Override
12      public void onDrawerSlide(View drawerView, float slideOffset) {
13      }
14      //当抽屉已经处于完全打开的状态时调用
15      @Override
16      public void onDrawerOpened(View drawerView) {
17          Toast.makeText(MainActivity.this, "抽屉菜单已打开！", Toast.LENGTH_SHORT).show();
18      }
19      //当抽屉已经完全关闭状态时调用
20      @Override
21      public void onDrawerClosed(View drawerView) {
22          Toast.makeText(MainActivity.this, "抽屉菜单已关闭！", Toast.LENGTH_SHORT).show();
23      }
24      //抽屉滑动状态改变时调用
25      //状态值STATE_IDLE：闲置、STATE_DRAGGING：拖拽、STATE_SETTLING：固定的
26      @Override
27      public void onDrawerStateChanged(int newState) {
28      }
29  });
```

（4）运行本实例，将显示如图 7.17 所示的界面。通过手指从手机左侧边框位置向右侧滑动后将显示如图 7.18 所示的效果。单击菜单外侧后将显示如图 7.19 所示的效果。

图 7.17　默认显示主屏幕

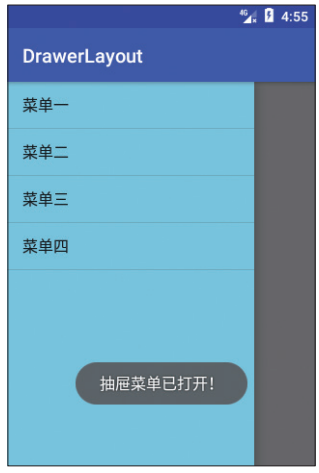


图 7.18　打开抽屉菜单

图 7.19　关闭抽屉菜单

7.2.5　下拉刷新（SwipeRefreshLayout）

SwipeRefreshLayout 是 Android 官方 v4 支持包中的一个组件，该组件当中只接受 1 个子组件也就是需要刷新内容的组件，如果有多个子组件时将只展示第一个子组件中所更新的内容，后面的子组件将不予展示。这个子组件必须是允许滚动的，如 ListView、GridView 或者是 RecyclerView 等。该组件主要用于在屏幕中手指向下滑动时起到下拉刷新数据的作用，例如，计步软件中同步数据时就可以使用 SwipeRefreshLayout 来实现，如图 7.20 所示。

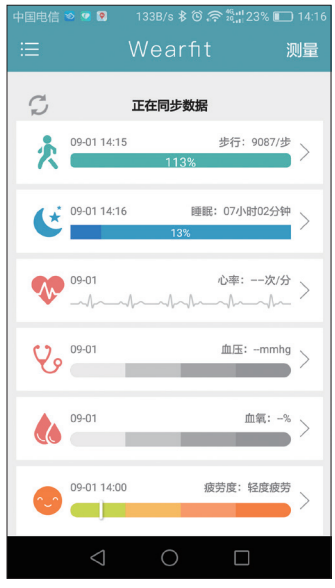


图 7.20　下拉同步数据

SwipeRefreshLayout 的常用方法如下：

◆ setOnRefreshListener: 设置刷新监听器，需要重写 onRefresh() 方法，顶部下拉时会调用这个方法，在里面实现请求数据的逻辑，设置下拉进度条消失等等。

◆ setColorSchemeResources：设置下拉进度条的颜色主题，可变参数并且是颜色资源 id，可以

设置多种不同的颜色，每转一圈就显示一种颜色。

- ◆ setProgressBackgroundColorSchemeResource：设置下拉进度条的背景颜色，默认为白色。
- ◆ setRefreshing: 设置刷新的状态，true 表示正在刷新，false 表示取消刷新。
- ◆ setDistanceToTriggerSync：设置手指向下滑动多少距离后触发刷新功能。
- ◆ isRefreshing：判断当前的状态是否正在刷新。

下面通过一个具体的实例演示下拉刷新（SwipeRefreshLayout）的应用。

例 7.6 下拉刷新后添加列表内容

在 Android Studio 中创建 Module，名称为“DropRefresh”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 SwipeRefreshLayout 组件并在该组件中添加 1 个 ListView 用于实现下拉刷新后添加列表内容。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--下拉刷新组件-->
    <android.support.v4.widget.SwipeRefreshLayout
        android:id="@+id/dropRefresh"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <!--信息列表-->
        <ListView
            android:id="@+id/listView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
        </ListView>
    </android.support.v4.widget.SwipeRefreshLayout>
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，首先定义所需要的全局变量，然后分别定义 Handler 与 Runnable 对象用于实现刷新后向列表中添加新数据，最后定义 1 个集合并在集合中设置默认显示列表中的数据。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    private SwipeRefreshLayout swipeRefreshLayout;          //下拉刷新组件
    private ListView listView;                              //列表组件
    private ArrayAdapter<String> adapter;                   //适配器
    //创建Handler与Runnable处理延迟后消息
    private Handler handler = new Handler();
    private Runnable refresh = new Runnable() {
        @Override
```

```
        public void run() {
            //向列表中添加新书名称
            data.addAll(Arrays.asList(
                    "Android 项目开发实战入门", "零基础学Android", "Java 200例"));
            adapter.notifyDataSetChanged();                //更改适配器中数据
            swipeRefreshLayout.setRefreshing(false);       //结束刷新
        }
    };
    //列表中信息集合
    private List<String> data = new ArrayList<String>(
            Arrays.asList("Android 200例", "Java 全能速查宝典", "Android 从入门到精通  ",
                    "Java开发实战宝典", "Android 开发宝典"));
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

（3）在主活动的 onCreate() 方法中，首先获取用于显示列表的 ListView 组件，然后获取 SwipeRefreshLayout 组件并为其设置时间监听器，再设置刷新进度条的颜色，最后创建列表适配器并为 ListView 列表组件设置该适配器。关键代码如下：

```
listView = (ListView) findViewById(R.id.listView);       //获取列表组件
//获取下拉刷新组件
swipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.dropRefresh);
//设置下拉刷新监听器
swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
    @Override
    public void onRefresh() {
        //延迟5秒显示刷新内容
        handler.postDelayed(refresh, 5000);
    }
});
//设置刷新进度条颜色
swipeRefreshLayout.setColorSchemeResources(android.R.color.holo_red_light,
        android.R.color.holo_blue_dark, android.R.color.holo_green_dark,
        android.R.color.holo_purple);
//创建列表适配器
adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, data);
//设置适配器
listView.setAdapter(adapter);
```

（4）运行本实例，将显示如图 7.21 所示的界面。手指在屏幕顶部向下滑动后将显示如图 7.22 所示的效果。进度条旋转 5 秒后将显示列表添加数据后的效果，如图 7.23 所示。

图 7.21　显示默认的列表信息

图 7.22　5 秒的加载过程

图 7.23　显示刷新后添加在列表中的信息

7.3　增强型滚动控件

RecyclerView 是谷歌公司在 Android 5.0 之后推出的一个新控件，RecyclerView 可以说是 ListView 和 GridView 的增强升级版。首先它可以实现如图 7.24 所示的垂直列表，然后可以实现如图 7.25 所示的网格列表，它还可以实现如图 7.26 所示的网格交错的运行效果。RecyclerView 不仅可以秒杀列表视图 ListView 与网格视图 GridView，甚至还可以秒杀瀑布流网格开源框架中的 PinterestLikeAdapterView，因此，RecyclerView 将会是循环视图中的一把利器。

图 7.24　垂直列表

图 7.25　网格列表

图 7.26　网格交错列表

7.3.1 RecyclerView 的基础用法

为了让 RecyclerView 可以在所有的 Android 版本中都能使用，Android 开发团队将 RecyclerView 定义在 support.v7 包当中。在使用该控件时需要打开当前 Module 的 build.gradle 文件，然后在 dependencies 节点中添加依赖库的代码。具体代码如下：

```
compile 'com.android.support:recyclerview-v7:插入版本号'
```

添加完成以后需要单击右上角的“Sync Now”进行同步。同步完成后可以在布局文件中添加 RecyclerView 控件，语法格式如下：

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

RecyclerView 常用的方法如下：

◆ setLayoutManager: 设置列表项的布局管理器，LinearLayoutManager 为线性布局管理器、GridLayoutManager 为网格布局管理器、StaggeredGridLayoutManager 为瀑布流网格布局管理器。

◆ setItemAnimator：设置列表项增加或者是删除时的动画，可以使用关键字 new 创建 DefaultItemAnimator() 对象实现系统默认的动画效果。

◆ addItemDecoration：添加列表项分割线。

◆ addOnItemTouchListener：添加列表项的触摸监听器。

◆ removeOnItemTouchListener：移除列表项的触摸监听器。

◆ setAdapter：设置列表项的适配器，使用 RecyclerView.Adapter。

RecyclerView.Adapter 是为 RecyclerView 所单独设计的适配器类，RecyclerView.Adapter 的相关方法如下：

◆ getItemCount: 获取列表项的数目。

◆ onBindViewHolder：绑定列表项中所显示的数据。

◆ onCreateViewHolder: 在该方法中可以加载列表 item（子项）中的布局文件。

下面通过一个具体的实例演示 RecyclerView 垂直列表的应用。

例 7.7 模拟 QQ 消息列表

在 Android Studio 中创建 Module，名称为“QQMessageList”，具体步骤如下：

（1）打开 build.gradle（Module: QQMessageList）文件，然后在该文件中的 dependencies 节点中添加依赖库的代码。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 RecyclerView 组件用于显示消息列表。具体代码如下：

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     xmlns:app="http://schemas.android.com/apk/res-auto"
04     xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--列表控件-->
    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:background="#EEEEEE"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        />
</RelativeLayout>
```

（3）在 res/layout 目录中创建一个名称为 layout_item.xml 的布局文件，将默认添加的布局管理器修改为相对布局管理器，首先添加一个 ImageView 组件用于显示头像的图标，然后添加 1 个垂直线性布局管理器，在该布局管理器中添加 2 个 TextView 组件分别用于显示名称与文字消息。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginBottom="1dp"
    android:background="@android:color/white"
    android:gravity="center_vertical">
    <!--图标-->
    <ImageView
        android:id="@+id/img"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_toRightOf="@+id/img"
        android:gravity="center_vertical"
        android:orientation="vertical">
        <!--名称-->
        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:textColor="#000000"
            android:textSize="15dp" />
```

```
        <!--消息-->
        <TextView
            android:id="@+id/info"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:layout_marginRight="15dp"
            android:singleLine="true"
            android:textSize="10dp" />
    </LinearLayout>
</RelativeLayout>
```

（4）在 res/values 目录中的 strings.xml 字符串资源文件内添加名字与文字消息，用于显示在列表当中。

（5）在 com.mingrisoft 包中创建 1 个名称为 Adapter 的 Java 类，首先在该类中创建 1 个名称为 MyViewHolder 的内部类并让该类继承自 RecyclerView.ViewHolder，在该内部类中获取列表中用于显示名称、头像图标与文字信息的组件。然后让 Adapter 类继承自 RecyclerView.Adapter<Adapter.MyViewHolder>，并且实现相对应的方法。具体代码如下：

```
public class Adapter extends RecyclerView.Adapter<Adapter.MyViewHolder> {
    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return null;
    }
    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
    }
    @Override
    public int getItemCount() {
        return 0;
    }
    class MyViewHolder extends RecyclerView.ViewHolder {
        public TextView name, info;              //名字与信息
        public ImageView img;                    //图标

        //获取相关控件
        public MyViewHolder(View itemView) {
            super(itemView);
            name = (TextView) itemView.findViewById(R.id.name);
            info = (TextView) itemView.findViewById(R.id.info);
            img = (ImageView) itemView.findViewById(R.id.img);
        }
    }
}
```

（6）创建 3 个 int 类型的数组分别用于保存头像的图标资源、名字、文字信息，然后创建 3 个 ArrayList 集合分别用于保存数组中的资源文件，最后创建 1 个 Adapter 构造方法，在该方法中设置菜单行数与行内图标、名称与文字信息。具体代码如下：

```
//图标数组
private int[] icons = {
        R.mipmap.icon_1, R.mipmap.icon_2, R.mipmap.icon_3,
        R.mipmap.icon_4, R.mipmap.icon_5, R.mipmap.icon_6,
        R.mipmap.icon_7, R.mipmap.icon_8, R.mipmap.icon_9,
        R.mipmap.icon_10, R.mipmap.icon_11
};
//名字数组，引用资源文件中的文字
private int[] names = {
        R.string.name1, R.string.name2, R.string.name3,
        R.string.name4, R.string.name5, R.string.name6,
        R.string.name7, R.string.name8, R.string.name9,
        R.string.name10, R.string.name11
};
//信息数组
private int[] infos = {
        R.string.info1, R.string.info2, R.string.info3,
        R.string.info4, R.string.info5, R.string.info6,
        R.string.info7, R.string.info8, R.string.info9,
        R.string.info10, R.string.info11
};
private Context lContext;   //上下文
private List<Integer> listIcon = new ArrayList<Integer>();   //图标集合
private List<Integer> listName = new ArrayList<Integer>();   //名称集合
private List<Integer> listInfo = new ArrayList<Integer>();   //信息集合
public Adapter(Context context) {
    lContext = context;
    //设置菜单行数与行内图标、名称、信息
    for (int i = 0; i < 11; i++) {
        listIcon.add(icons[i]);
        listName.add(names[i]);
        listInfo.add(infos[i]);
    }
}
```

（7）重写 MyViewHolder onCreateViewHolder() 方法，在该方法中实现获取列表中每行 item 的布局文件。修改后代码如下：

```
@Override
public MyViewHolder onCreateViewHolder(ViewGroup arg0, int arg1) {
    //获取列表中每行的布局文件
    View view = LayoutInflater.from(lContext).inflate(R.layout.layout_item, arg0, false);
    MyViewHolder holder = new MyViewHolder(view);
    return holder;
}
```

（8）重写 onBindViewHolder() 方法，在该方法中设置列表菜单中 item（子项）所显示的内容。修改后代码如下：

```
@Override
public void onBindViewHolder(final MyViewHolder holder, int position) {
    //设置图标
    holder.img.setBackgroundResource(listIcon.get(position));
    //设置名称
    holder.name.setText(listName.get(position));
    //设置信息
    holder.info.setText(listInfo.get(position));
}
```

（9）重写 getItemCount() 方法，在该方法中实现返回数据集中的项目总数。修改后代码如下：

```
@Override
public int getItemCount() {
    return listIcon.size();
}
```

（10）打开主活动 MainActivity.java 文件，定义所需要的全局变量，在 onCreate() 方法中首先获取 RecyclerView 组件，然后为其设置列表布局管理器，最后为其设置适配器。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    private RecyclerView lRecyclerView;      //列表控件
    private Adapter lAdapter;                 //适配器
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取列表控件
        lRecyclerView = (RecyclerView) findViewById(R.id.recyclerview);
        //设置列表布局管理器
        lRecyclerView.setLayoutManager(new LinearLayoutManager(this));
        //设置适配器
        lRecyclerView.setAdapter(lAdapter = new Adapter(this));
    }
}
```

（11）运行本实例，将显示如图 7.27 所示的运行效果。

图 7.27　模拟 QQ 消息列表

7.3.2 RecyclerView 的网格布局

如果让 RecyclerView 实现 1 个网格布局的运行效果，就需要使用 GridLayoutManager 类也就是网格布局管理器。GridLayoutManager 就像 GridView 一样，需要设置网格的列数才可以正常的显示，GridLayoutManager 可以通过构造方法指定网格的列数，也可以通过以下的两种方法来设置网格的列数，具体方法如下：

◆ setSpanCount：设置网格的列数。

◆ setSpanSizeLookup: 设置网格 item（子项）所占有的位置，默认为 1 项占 1 列，也可以通过 GridLayoutManager.SpanSizeLookup 类中的方法设置某个 item（子项）占用多个列。

下面通过一个具体的实例演示 RecyclerView 网格布局的应用，由于本实例与 7.3.1 小节中的实例代码部分相同，这里只给出不同部分的代码。

例 7.8 RecyclerView 的网格布局

在 Android Studio 中创建 Module，名称为“RecyclerViewGridlayout”，具体步骤如下：

（1）打开 build.gradle（Module: RecyclerViewGridlayout）文件，然后在该文件中的 dependencies 节点中添加依赖库的代码。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 RecyclerView 组件用于显示网格列表。

（3）在 res/layout 目录中创建 1 个名称为 layout_item.xml 的布局文件，将默认添加的布局管理器修改为垂直线性布局管理器并将布局管理器的高度设置为包裹其自身内容，首先添加 1 个 ImageView 组件用于显示图片，然后添加 1 个 TextView 组件用于显示图片的标题文字。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <!--显示图片-->
    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal" />
    <!--显示图片的标题-->
    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="17sp" />
</LinearLayout>
```

（4）在 com.mingrisoft 包中创建 1 个名称为 Adapter 的 Java 类，在该类中获取用于显示图片与标题文字的组件，然后设置网格列表中所显示的内容。

（5）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，定义所需要的全局变量，在 onCreate() 方法中首先设置全

屏代码，然后获取 RecyclerView 组件，再为其设置网格布局管理器并指定第 1 个网格占用两列，最后为其设置适配器。具体代码如下：

```
01  public class MainActivity extends Activity {
02      private RecyclerView lRecyclerView;      //列表控件
03      private Adapter lAdapter;                //适配器
04      @Override
05      protected void onCreate(Bundle savedInstanceState) {
06          super.onCreate(savedInstanceState);
07          setContentView(R.layout.activity_main);
08          //全屏
09          getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN ,
10                  WindowManager.LayoutParams. FLAG_FULLSCREEN);
11          //获取列表控件
12          lRecyclerView = (RecyclerView) findViewById(R.id.recyclerview);
13          //创建网格布局管理器
14          GridLayoutManager manager = new GridLayoutManager(this, 2);
15          //设置网格列表中第1个与第2个item（子项）跨列每个子项占两个位置
16          manager.setSpanSizeLookup(new GridLayoutManager.SpanSizeLookup() {
17              @Override
18              public int getSpanSize(int position) {
19                  if (position == 0 ) {        //判断是第1个或者第2个子项
20                      return 2;                //返回要跨的列数
21                  } else {
22                      return 1;
23                  }
24              }
25          });
26          //设置网格布局管理器
27          lRecyclerView.setLayoutManager(manager);
28          //设置适配器
29          lRecyclerView.setAdapter(lAdapter = new Adapter(this));
30      }
31  }
```

（6）运行本实例，将显示如图 7.28 所示的效果。

图 7.28　RecyclerView 的网格布局

7.3.3 RecyclerView 的瀑布流网格布局

在许多电商的 App 当中都需要网格交错的布局进行商品的展示，因为不同商品的外观与尺寸都是不一样的，例如，一台电视可能需要横向的尺寸比较长一些，而冰箱就需要纵向的尺寸长一些，所以要是通过尺寸相同的网格布局来实现必然会有部分商品的图片被压缩变形。在这种情况下根据商品尺寸的不同来展示各种商品就可以使用瀑布流网格布局。通过 RecyclerView 实现瀑布流网格布局时需要使用 StaggeredGridLayoutManager 类，然后在适配器中动态设置每个网格的尺寸，最后 RecyclerView 将适配器中的数据以瀑布流网格布局的形式显示在界面当中。StaggeredGridLayoutManager 的常用方法如下：

- setOrientation：设置布局排列的方向。
- setReverseLayout：设置相反方向的布局排列。
- setSpanCount：设置网格的列数。
- StaggeredGridLayoutManager：构造方法设置网格的列数和方向。

下面通过一个具体的实例演示 RecyclerView 瀑布流网格布局的应用，由于本实例与 7.3.1 小节中的实例代码部分相同，这里只给出不同部分的代码。

例 7.9 RecyclerView 的瀑布流网格布局

在 Android Studio 中创建 Module，名称为“StaggeredGridLayout”，具体步骤如下：

（1）打开 build.gradle（Module: StaggeredGridLayout）文件，然后在该文件中的 dependencies 节点中添加依赖库的代码。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个 RecyclerView 组件用于显示瀑布流网格列表。

（3）在 res/layout 目录中创建 1 个名称为 item.xml 的布局文件，将默认添加的布局管理器修改为垂直线性布局管理器并将布局管理器的高度设置为包裹其自身内容，然后添加 1 个 ImageView 组件用于显示图片，再添加 1 个 TextView 组件用于显示图片的标题文字。

（4）在 com.mingrisoft 包中创建 1 个名称为 MyRecyclerAdapter 的 Java 类，在该类中创建 getRandomHeight() 方法用于得到随机产生的网格高度，然后定义 OnItemClickListener() 接口用于设置单击事件与长按事件的方法，再创建 setOnClickListener() 方法，用于设置单击事件。关键代码如下：

```
01  //得到随机item的高度
02  private void getRandomHeight(List<String> lists) {
03      heights = new ArrayList<>();                          //网格高度的集合
04      for (int i = 0; i < lists.size(); i++) {
05          //随机产生网格高度并添加到集合中
06          heights.add((int) (200 + Math.random() * 100));
07      }
08  }
09  //单击事件与长按事件的接口
10  public interface OnItemClickListener {
11      void ItemClickListener(View view, int postion);
```

```
    void ItemLongClickListener(View view, int postion);
}
//设置单击事件的方法
public void setOnClickListener(OnItemClickListener listener) {
    this.mListener = listener;
}
```

（5）重写 onBindViewHolder() 方法，在该方法中设置瀑布流网格列表中 item（子项）所显示的内容并设置单击事件与长按事件接口中的方法。修改后代码如下：

```
@Override
public void onBindViewHolder(final MyViewHolder holder, int position) {
    //得到item的LayoutParams布局参数
    ViewGroup.LayoutParams params = holder.itemView.getLayoutParams();
    params.height = heights.get(position);  //把随机的高度赋予item布局
    holder.itemView.setLayoutParams(params);//把params设置给item布局
    holder.mIv.setBackgroundResource(resids.get(position % 15));//设置图片
    holder.mTv.setText("图书" + lists.get(position));            //为控件绑定数据
    if (mListener != null) {      //如果设置了监听那么它就不为空，然后回调相应的方法
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int pos = holder.getLayoutPosition();//获取当前点击item的位置pos
                //把事件交给我们实现的接口那里处理
                mListener.ItemClickListener(holder.itemView, pos);
            }
        });
        holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                int pos = holder.getLayoutPosition();//获取当前点击item的位置pos
                //把事件交给我们实现的接口那里处理
                mListener.ItemLongClickListener(holder.itemView, pos);
                return true;
            }
        });
    }
}
```

（6）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，定义所需要的全局变量，然后创建 1 个 initData() 方法，该方法用于加载图片与图片标题文字的信息。具体代码如下：

```
public class MainActivity extends Activity {
```

```
    private RecyclerView mRecyclerView;          //列表控件
    private List<String> lists;                  //文字信息集合
    private MyRecyclerAdapter adapter;           //适配器
    private List<Integer>  resids;               //图片id集合
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    private void initData() {
        //文字信息集合
        lists = new ArrayList();
        //资源id集合
        resids= new ArrayList();
        for (int i = 0; i < 100; i++) {
            lists.add("" + i);
        }
        //添加图片id
        resids.add(R.mipmap.book0);
        resids.add(R.mipmap.book1);
        resids.add(R.mipmap.book2);
        resids.add(R.mipmap.book3);
        resids.add(R.mipmap.book4);
        resids.add(R.mipmap.book5);
        resids.add(R.mipmap.book6);
        resids.add(R.mipmap.book7);
        resids.add(R.mipmap.book8);
        resids.add(R.mipmap.book9);
        resids.add(R.mipmap.book10);
        resids.add(R.mipmap.book11);
        resids.add(R.mipmap.book12);
        resids.add(R.mipmap.book13);
        resids.add(R.mipmap.book14);
    }
}
```

（7）在 onCreate() 方法中首先获取 RecyclerView 组件，然后为其设置瀑布流网格布局管理器，再为其设置适配器，最后为适配器 adapter 设置单击事件与长按事件，并在长按事件中实现长按网格中的某项后，从瀑布流列表中删除该项内容。关键代码如下：

```
//全屏
getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN ,
        WindowManager.LayoutParams. FLAG_FULLSCREEN);
```

```
04  initData();          //加载图片与图片文字信息
05  //列表控件
06  mRecyclerView = (RecyclerView) this.findViewById(R.id.recyclerView);
07  mRecyclerView.setItemAnimator(new DefaultItemAnimator());
08  //设置RecyclerView布局管理器为2列垂直排布
09  mRecyclerView.setLayoutManager(new StaggeredGridLayoutManager
10          (2, StaggeredGridLayoutManager.VERTICAL));
11  //创建适配器对象
12  adapter = new MyRecyclerAdapter(this,lists,resids);
13  mRecyclerView.setAdapter(adapter);       //设置适配器
14  adapter.setOnClickListener(new MyRecyclerAdapter.OnItemClickListener() {
15      //单击事件
16      @Override
17      public void ItemClickListener(View view, int postion) {
18          Toast.makeText(MainActivity.this,"单击了图书: "
19                  +postion,Toast.LENGTH_SHORT).show();
20      }
21      //长按事件
22      @Override
23      public void ItemLongClickListener(View view, int postion) {
24          //长按删除
25          lists.remove(postion);
26          //在适配器中移除
27          adapter.notifyItemRemoved(postion);
28      }
29  });
```

（8）运行本实例，效果如图 7.29 所示。单击瀑布流列表中的“图书 1”将显示如图 7.30 所示的效果，长按“图书 1”后将在瀑布流列表中移除该项，如图 7.31 所示。

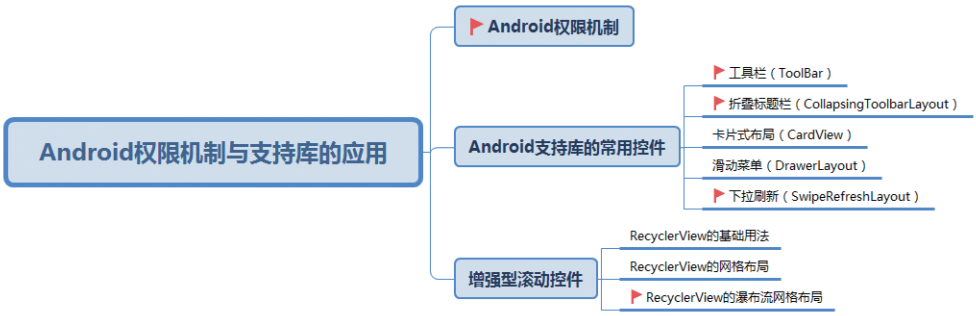
图 7.29　默认显示瀑布流网格列表

图 7.30　单击图书 1 显示效果

图 7.31　长按图书 1 显示效果

7.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 8 章

自定义控件

在日常的 Android 开发中经常会和控件打交道，有时 Android 提供的控件未必能满足业务的需求，此时开发者就可以通过自定义控件的方式来解决业务中的需求。本章将介绍自定义控件的要求和实现的基本原理。

8.1 自定义控件实现流程

8.1.1 自定义属性

自定义一个控件的时候，多数是因为系统自带的控件无法满足开发需要的属性或者是功能，下面以 TextView 控件为例，通过自定义控件的方式，模拟声明 TextView 控件的常用属性，常用属性类型的取值说明如表 8.1 所示。

表 8.1　属性类型的取值说明

属 性 类 型	获 取 方 法	描　　述
float	getFloat()	浮点类型的值
integer	getInt()	整数类型的值
string	getString()	字符串类型的值
color	getColor()	颜色值，取值为 # 开头的六位或八位十六进制数值
boolean	getBoolean()	布尔类型的值
dimension	getDimension()	尺寸值，取值为末尾带有 dp 的值
dimension	getDimensionPixelSize()	字体大小，取值为末尾带有 sp 的值
reference	getResourceId()	引用资源的 ID
fraction	getFraction()	百分数，取值为末尾带有 % 号的值
enum	getInt()	枚举值
flag	getInt()	标志位

1. 声明属性

在 res/values 目录下创建名称为 attrs.xml，然后在该文件中添加需要定义的属性名称。代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <declare-styleable name="MyTextView">
            <attr name="mText" format="string" />
            <attr name="mTextColor" format="color"/>
            <attr name="mTextSize" format="dimension"/>
        </declare-styleable>
</resources>
```

2. 构造对象

在 Android 中每一个控件都有自己的父类，而创建一个自定义的控件同样需要找到自定义控件的目标类作为父类，例如，在开发中经常使用到的 TextView 控件就是需要继承 View 类才可以实现它自身所需要的方法与功能，继承父类以后还需要重写 3 个构造方法。代码如下：

```
public class MyTextView extends View {
    //声明对象时采用单参数构造方法
    public MyTextView(Context context) {
        //引用两个参数的构造方法
        this(context, null);
    }
    //在布局文件中引用自定义视图时采用两个参数的构造方法
    public MyTextView(Context context, AttributeSet attrs) {
        //引用三个参数的构造方法
        this(context, attrs, 0);
    }
    //该构造函数除了可以在布局文件中指定属性以外，还可以指定默认的样式
    public MyTextView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }
}
```

实现了三个构造方法以后，然后在三个参数的构造方法中进行属性值的获取与设置。代码如下：

```
//获取自定义属性的值
TypedArray typedArray = context.getTheme().obtainStyledAttributes
        (attrs, R.styleable.MyTextView, defStyleAttr, 0);
//获取字符串属性值
mText = typedArray.getString(R.styleable.MyTextView_mText);
//获取字体颜色属性值
mTextColor = typedArray.getColor(R.styleable.MyTextView_mTextColor, Color.BLACK);
//获取字体大小属性值
mTextSize = (int) typedArray.getDimension(R.styleable.MyTextView_mTextSize, 100);
typedArray.recycle();                //回收属性资源
```

```
mPaint = new Paint();              //创建画笔
mPaint.setAntiAlias(true);         //开启抗锯齿
mPaint.setTextSize(mTextSize);     //设置字体大小
mPaint.setColor(mTextColor);       //设置字体颜色
//获取绘制文本的宽和高
mBound = new Rect();
mPaint.getTextBounds(mText, 0, mText.length(), mBound);
```

最后通过重写 onDraw() 方法来实现文字的绘制。代码如下：

```
@Override
protected void onDraw(Canvas canvas) {
    //绘制文字
    canvas.drawText(mText, getWidth() / 2 - mBound.width() / 2,
            getHeight() / 2 + mBound.height() / 2, mPaint);
}
```

完成了以上的操作步骤，MyTextView 的自定义控件与属性就已经完成了。在布局文件中使用 MyTextView 自定义控件时还需要在根节点中添加命名空间声明，然后将自定义控件的全路径名称填写在布局文件中即可，如图 8.1 所示。

图 8.1　在布局文件中使用 MyTextView 自定义控件

说明　三个参数的构造方法中，参数defStyleAttr将参照类型（reference，该参数需要在styles.xml文件中定义）使用步骤如下：

（1）在styles.xml文件中定义样式DefaultMyTextStyle。

（2）在attrs.xml文件中声明该样式的参照属性，示例代码如下：

```
<attr name="mTextStyle" format="reference"/>
```

（3）在两个参数的构造方法中，调用三个参数的构造方法时将第三个参数设置为参照属性。示例代码如下：

```
01  public MyTextView(Context context, AttributeSet attrs) {
02      //引用三个参数的构造方法
03      this(context, attrs, R.attr.mTextStyle);
04  }
05  public MyTextView(Context context, AttributeSet attrs, int defStyleAttr) {
06      super(context, attrs, defStyleAttr);
07      //获取自定义属性的值
08      TypedArray typedArray = context.getTheme().obtainStyledAttributes
09          (attrs, R.styleable.MyTextView, defStyleAttr, R.style.DefaultMyTextStyle);
10  }
```

此时系统会首先查找布局文件中所指定的属性，然后查找attrs.xml文件中所声明的R.attr.mTextStyle样式，最后查找styles.xml文件中的R.style.DefaultMyTextStyle的样式。

8.1.2 测量尺寸

测量视图就是为了能够让自定义控件根据各种不同的情况以合适的宽和高去展示自己，一般在布局文件中对视图的宽和高有 3 种赋值方式，具体详情参照表 8.2 所示。

表 8.2　在布局文件中对视图的 3 种赋值方式

XML 中的尺寸类型	子视图的尺寸类型	描　　述
match_parent	MATCH_PARENT	与上一级视图大小一样
wrap_content	WRAP_CONTENT	包裹其自身内容（按照自身尺寸进行适配）
dp	整型数值	设置尺寸的数值

提到测量就必须要提到 onMeasure 方法了，该方法用于设置视图的宽和高，也就是自定义控件的大小。代码如下：

```
01  @Override
02  protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
03      super.onMeasure(widthMeasureSpec, heightMeasureSpec);
04  }
```

在 onMeasure 方法中有两个重要的参数 widthMeasureSpec 与 heightMeasureSpec，这两个参数的测量规则为 int 类型，但是它并不是实际的尺寸，而是尺寸和测量模式的合成值。它们需要通过 MeasureSpec 类提供的静态方法，然后从 widthMeasureSpec 和 heightMeasureSpec 参数中提取测量模式和尺寸。代码如下：

```
01      //获取父 View 的测量模式
02  int specMode = MeasureSpec.getMode(measureSpec);
03      //获取父 View 的测量尺寸
04  int specSize = MeasureSpec.getSize(measureSpec);
```

在 Android 的视图当中一共提供了 3 种测量模式，具体详情参照表 8.3 所示。

表 8.3 Android 视图的 3 种测量模式

测 量 模 式	视图的赋值方式	描 述
UNSPECIFIED	MATCH_PARENT	与上一级视图大小一样
AT_MOST	WRAP_CONTENT	包裹其自身内容（按照自身尺寸进行适配）
EXACTLY	整型数值	设置尺寸的数值

8.1.3 绘制 View

在自定义控件被创建并且测量代码写好之后，就可以用 onDraw() 方法来绘制 View 了，onDraw（）方法包含了一个 Canvas，叫作画布的参数，onDraw() 简单来说就两点：

（1）Canvas 决定要去画什么。

（2）Paint 决定怎么画。

比如，Canvas 提供了画线方法，Paint 就来决定线的颜色；Canvas 提供了画矩形，Paint 可以决定让矩形是空心还是实心的。

在 onDraw（）方法中开始绘制之前，应该让画笔 Paint 对象的信息初始化完毕。这是因为 View 的重新绘制是比较频繁的，这就可能多次调用 onDraw（），所以初始化的代码不应该放在 onDraw（）方法里。

Canvas 和 Paint 提供很多方法在本章中就不一一列举了。具体的相关知识在接下来的章节中进行详细介绍。

下面通过以上自定义控件的实现流程，实现一个自定义绘制视图的实例，效果如图 8.2 所示。

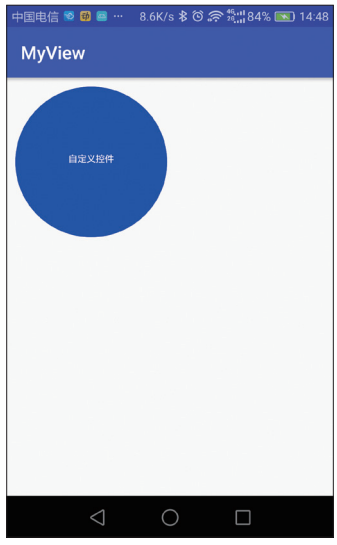


图 8.2 自定义绘制视图

例 8.1 自定义绘制视图

在 Android Studio 中创建 Module，名称为“MyView”。在该 Module 中实现本实例，具体步骤如下：

（1）创建 MyView 类并且让该类继承自 View 类，然后创建构造方法并且在构造方法中对画笔进行初始化工作。代码如下：

```
01 public class MyView extends View {
02     //默认控件宽和高
03     private static final int MY_VIEW_WIDTH = 500;
04     private static final int MY_VIEW_HEIGHT = 500;
05     //声明画笔
06     private Paint mPaint;
07     public MyView(Context context) {
08         super(context);
09         //初始化画笔
10         init();
11     }
12     public MyView(Context context, AttributeSet attrs) {
13         super(context, attrs);
14         init();
15     }
16     public MyView(Context context, AttributeSet attrs, int defStyleAttr) {
17         super(context, attrs, defStyleAttr);
18     }
19     //初始化相关控件
20     private void init(){
21         mPaint = new Paint();
22         mPaint.setAntiAlias(true);
23     }
24 }
```

（2）创建 measureDimension() 方法，在该方法中进行控件尺寸的测量工作。代码如下：

```
01 //测量尺寸
02 protected int measureDimension( int defaultSize, int measureSpec ) {
03     int result = defaultSize;
04     int specMode = MeasureSpec.getMode(measureSpec);
05     int specSize = MeasureSpec.getSize(measureSpec);
06     //1. layout给出了确定的值，比如：100dp
07     //2. layout使用的是match_parent，但父控件的size已经可以确定了，
08     //  比如设置的是具体的值或者match_parent
09     if (specMode == MeasureSpec.EXACTLY) {
10         result = specSize; //建议：result直接使用确定值
11     }
12     //1. layout使用的是wrap_content
13     //2. layout使用的是match_parent,但父控件使用的是确定的值或者wrap_content
14     else if (specMode == MeasureSpec.AT_MOST) {
15         result = Math.min(defaultSize, specSize); //建议：result不能大于specSize
16     }
17     //UNSPECIFIED,没有任何限制，所以可以设置任何大小
18     //多半出现在自定义的父控件的情况下，期望由自定义自行决定大小
19     else {
20         result = defaultSize;
```

```
    }
    return result;
}
```

（3）重写 onMeasure() 方法，在该方法中实现自定义控件的测量并设置控件的大小。代码如下：

```
//测量并设置控件大小
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);
    //获取控件宽度与高度
    int width  = measureDimension(MY_VIEW_WIDTH, widthMeasureSpec);
    int height = measureDimension(MY_VIEW_HEIGHT, heightMeasureSpec);
    //设置测量结果
    setMeasuredDimension(width, height);
}
```

（4）重写 onDraw() 方法，在该方法中实现自定义控件的绘制工作。代码如下：

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    //设置圆形画笔颜色
    mPaint.setColor(Color.BLUE);
    //FILL填充，STROKE描边,FILL_AND_STROKE填充和描边
    mPaint.setStyle(Paint.Style.FILL_AND_STROKE);
    //获取宽和高
    int with = getWidth();
    int height = getHeight();
    float radius = with / 2;
    //绘制圆形
    canvas.drawCircle(with/2 , with/2 , radius, mPaint);
    //设置要绘制文字画笔颜色
    mPaint.setColor(Color.WHITE);
    //设置字号
    mPaint.setTextSize(30f);
    //要绘制的文字
    String text="自定义控件";
    //获取字符串的宽度
    float stringWidth = mPaint.measureText(text);
    //根据宽度计算位置
    float x =with/2-stringWidth/2;
    canvas.drawText("自定义控件",x,height/2,mPaint);
}
```

（5）打开默认生成的 activity_main.xml 布局文件，首先将默认生成的布局管理器修改为相对布局管理器，然后添加自定义控件 MyView。关键代码如下：

```
<com.mingrisoft.myview.MyView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="10dp" />
```

8.2 自定义控件

8.2.1 可嵌套在 ScrollView 中的列表

在 Android Studio 中复杂而页面长渐渐成了流行趋势，例如京东、淘宝等，而实现界面的滑动就需要用到 ScrollView 与列表控件的配合，但是 ScrollView 与 ListView 以及 GridView 控件同时出现滑动时会出现冲突情况，此时就需要自定义列表控件来满足以上的开发需求。

1. 自定义失去滑动特性的 ListView，可以嵌套在 ScrollView 中。具体代码如下：

```
public class MyListView extends ListView {
    public MyListView(Context context, AttributeSet attrs) {
        super(context, attrs);
        //TODO Auto-generated constructor stub
    }
   //通过onMeasure方法重新计算高度
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        //TODO Auto-generated method stub
        int expandSpec = MeasureSpec.makeMeasureSpec(Integer.MAX_VALUE >> 2,
                                                     MeasureSpec.AT_MOST);
        super.onMeasure(widthMeasureSpec, expandSpec);
    }
}
```

2. 自定义失去滑动特性的 MyGridView，可以嵌套在 ScrollView 中。具体代码如下：

```
public class MyGridView extends GridView{
    public MyGridView(Context context, AttributeSet attrs){
         super(context, attrs);
    }
    //通过onMeasure方法重新计算高度
    public void onMeasure(int widthMeasureSpec, int heightMeasureSpec){
         int mExpandSpec = MeasureSpec.makeMeasureSpec(Integer.MAX_VALUE >> 2,
                                                       MeasureSpec.AT_MOST);
         super.onMeasure(widthMeasureSpec, mExpandSpec);
    }
}
```

8.2.2 外星人手机时钟

现在大多数人都是通过手机来看时间的，带手表的人越来越少了。本实例为自定义一款时钟样式—— 外星人手机时钟。

例 8.2 外星人手机时钟

在 Android Studio 中创建 Module，名称为“Clock”。在该 Module 中实现本实例，具体步骤如下。

（1）在 res\values 目录下创建名称为 attrs.xml，然后在该文件中添加需要定义的属性名称。代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="AnalogClock">
        <attr name="dial" format="reference" />
        <attr name="hand_hour" format="reference" />
        <attr name="hand_minute" format="reference" />
        <attr name="hand_second" format="reference" />
    </declare-styleable>
</resources>
```

（2）在 styles.xml 文件中添加时钟样式，代码如下：

```
<style name="AppWidget">
    <item name="dial">@drawable/clock_bj</item>
    <item name="hand_hour">@drawable/clock_hour1</item>
    <item name="hand_minute">@drawable/clock_fenzhen1</item>
    <item name="hand_second">@drawable/clock_miao1</item>
</style>
```

（3）创建 MyClock 类，该类同样继承自 View 类并且实现三个构造方法，然后在带有三个参数的构造方法中首先获取自定义时钟所使用的图片资源，然后进行时间与画笔的初始化工作，代码如下：

```
Resources r = getContext().getResources();
//获取布局文件中自定义控件所使用的图片资源，如果没有指定属性资源则使用默认的
TypedArray a = context.getTheme().obtainStyledAttributes(attrs,
        R.styleable.Clock, defStyle, 0);
//获取表盘
mDial = a.getDrawable(R.styleable.Clock_dial);
//获取时针
mHourHand = a.getDrawable(R.styleable.Clock_hand_hour);
//获取分针
mMinuteHand = a.getDrawable(R.styleable.Clock_hand_minute);
//获取秒针
mSecondHand = a.getDrawable(R.styleable.Clock_hand_second);

//为了整体美观性，只要缺少一张图片，我们就用默认的那套图片
```

```
15  if (mDial == null || mHourHand == null || mMinuteHand == null
16          || mSecondHand == null) {
17      mDial = r.getDrawable(R.drawable.clock_bj);
18      mHourHand = r.getDrawable(R.drawable.clock_hour);
19      mMinuteHand = r.getDrawable(R.drawable.clock_fenzhen);
20      mSecondHand = r.getDrawable(R.drawable.clock_miao);
21  }
22  a.recycle();//不调用这个函数，则上面的都是白费功夫
23  //获取表盘的宽度和高度
24  mDialWidth = mDial.getIntrinsicWidth();
25  mDialHeight = mDial.getIntrinsicHeight();
26  //初始化画笔
27  mPaint = new Paint();
28  mPaint.setColor(Color.parseColor("#3399ff"));
29  mPaint.setTypeface(Typeface.DEFAULT_BOLD);
30  mPaint.setFakeBoldText(true);
31  mPaint.setAntiAlias(true);
32  //初始化Time对象
33  if (mCalendar == null) {
34      mCalendar = new Time();
35  }
```

（4）重写 onMeasure() 方法，通过该方法测量控件的大小，进行相应的处理，这里主要是对控件上显示的图片进行压缩。代码如下：

```
01  @Override
02  protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
03      //根据提供的测量值(格式)提取模式
04      int widthMode = MeasureSpec.getMode(widthMeasureSpec);
05      //根据提供的测量值(格式)提取大小值(这个大小也就是我们通常所说的大小)
06      int widthSize = MeasureSpec.getSize(widthMeasureSpec);
07      //高度与宽度类似
08      int heightMode = MeasureSpec.getMode(heightMeasureSpec);
09      int heightSize = MeasureSpec.getSize(heightMeasureSpec);
10      //缩放值
11      float hScale = 1.0f;
12      float vScale = 1.0f;
13      //如果父元素提供的宽度比图片宽度小，就需要压缩一下子元素的宽度
14      if (widthMode != MeasureSpec.UNSPECIFIED && widthSize < mDialWidth) {
15          hScale = (float) widthSize / (float) mDialWidth;
16      }
17      //如果父元素提供的高度比图片高度小，就需要压缩一下子元素的高度
18      if (heightMode != MeasureSpec.UNSPECIFIED && heightSize < mDialHeight) {
19          vScale = (float) heightSize / (float) mDialHeight;//同上
20      }
21      //取最小的压缩值，值越小，压缩越厉害
22      float scale = Math.min(hScale, vScale);
```

```
23      //最后保存一下，这个函数一定要调用
24      setMeasuredDimension(
25              resolveSizeAndState((int) (mDialWidth * scale),
26                      widthMeasureSpec, 0),
27              resolveSizeAndState((int) (mDialHeight * scale),
28                      heightMeasureSpec, 0));
29  }
```

（5）重写 onAttachedToWindow() 方法，实现当时钟控件添加到窗口中，进行时间的更新。代码如下：

```
01  @Override
02  protected void onAttachedToWindow() {
03      //添加到窗口中就要更新时间了
04      mTickerStopped = false;
05      super.onAttachedToWindow();
06      //更新线程
07      mTicker = new Runnable() {
08          public void run() {
09              //判断是否退出线程
10              if (mTickerStopped)
11                  //退出线程
12                  return;
13              //时间改变方法
14              onTimeChanged();
15              //刷新控件
16              invalidate();
17              //获取当前系统时间
18              long now = SystemClock.uptimeMillis();
19              //计算下次需要更新的时间间隔
20              long next = now + (1000 - now % 1000);
21              //递归执行，就达到秒针一直转动的效果
22              mHandler.postAtTime(mTicker, next);
23          }
24      };
25      mTicker.run();
26  }
```

（6）重写 onDraw() 方法，在该方法中首先获取时钟控件的宽度与高度，然后获取表盘、时针、分针、秒针的宽度与高度，最后进行时钟控件的绘制工作，代码如下：

```
01  @Override
02  protected void onDraw(Canvas canvas) {
03      super.onDraw(canvas);
04      boolean changed = mChanged;
05      if (changed) {
06          mChanged = false;
```

```
    }
    int availableWidth = getRight() - getLeft(); //view可用宽度，通过右坐标减去左坐标
    int availableHeight = getBottom() - getTop();//view可用高度，通过下坐标减去上坐标

    int x = availableWidth / 2;                  //view宽度中心点坐标
    int y = availableHeight / 2;                 //view高度中心点坐标

    final Drawable dial = mDial;                 //表盘图片
    int w = dial.getIntrinsicWidth();            //表盘宽度
    int h = dial.getIntrinsicHeight();

    boolean scaled = false;
    //最先画表盘，最底层的要先画上画板
    //如果view的可用宽和高小于表盘图片，就要缩小图片
    if (availableWidth < w || availableHeight < h) {
        scaled = true;
        float scale = Math.min((float) availableWidth / (float) w,
                (float) availableHeight / (float) h);//计算缩小值
        canvas.save();
        canvas.scale(scale, scale, x, y);        //实际上是缩小的画板
    }
    if (changed) {
        //设置表盘图片位置。组件在容器X轴上的起点;
        //组件在容器Y轴上的起点;  组件的宽度;组件的高度
        dial.setBounds(x - (w / 2), y - (h / 2), x + (w / 2), y + (h / 2));
    }
    dial.draw(canvas); //这里才是真正把表盘图片画在画板上
    canvas.save();     //一定要保存一下
    //再画时针
    //旋转画板，第一个参数为旋转角度，第二、三个参数为旋转坐标点
    canvas.rotate(mHour / 12.0f * 360.0f, x, y);
    final Drawable hourHand = mHourHand;
    if (changed) {
        w = hourHand.getIntrinsicWidth();
        h = hourHand.getIntrinsicHeight();
        hourHand.setBounds(x - (w / 2), y - (h / 2), x + (w / 2), y
                + (h / 2));
    }
    hourHand.draw(canvas);                       //把时针画在画板上
    canvas.restore();                            //恢复画板到最初状态
    canvas.save();
    //然后画分针
    canvas.rotate(mMinutes / 60.0f * 360.0f, x, y);
```

```
50      final Drawable minuteHand = mMinuteHand;
51      if (changed) {
52          w = minuteHand.getIntrinsicWidth();
53          h = minuteHand.getIntrinsicHeight();
54          minuteHand.setBounds(x - (w / 2), y - (h / 2), x + (w / 2), y
55                  + (h / 2));
56      }
57      minuteHand.draw(canvas);
58      canvas.restore();
59      canvas.save();
60      //最后画秒针
61      canvas.rotate(mSecond / 60.0f * 360.0f, x, y);
62      final Drawable secondHand = mSecondHand;
63      if (changed) {
64          w = secondHand.getIntrinsicWidth();
65          h = secondHand.getIntrinsicHeight();
66          secondHand.setBounds(x - (w / 2), y - (h /2), x + (w / 2), y+ (h / 2)-4);
67      }
68      secondHand.draw(canvas);
69      canvas.restore();
70
71      if (scaled) {
72          canvas.restore();
73      }
74  }
```

（7）运行本实例，将显示如图 8.3 所示的界面效果。

图 8.3　自定义手机时钟

8.3 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。

第 9 章

主角（Activity）与配角（Intent）

在前面介绍的实例中已经应用过 Activity，不过那些实例中的所有操作都是在一个 Activity 中进行的，在实际的应用开发中，经常需要包含多个 Activity，而且这些 Activity 之间可以通过 Intent 相互跳转或传递数据。Intent 对象中包含组件名称、动作、数据等内容。根据 Intent 中的内容，Android 系统可以启动需要的组件。本章将对主角 Activity 与配角 Intent 进行详细介绍。

9.1 主角的作用

在 Android 应用中，提供了 4 大基本组件，分别是 Activity、Service、BroadcastReceiver 和 ContentProvider。其中，Activity 是 Android 应用最常见的组件之一，它的中文意思是活动。在 Android 中，Activity 代表手机或者平板电脑中的一屏，它提供了和用户交互的可视化界面。在一个 Activity 中，可以添加很多组件，这些组件负责具体的功能。

在一个 Android 应用中，可以有多个 Activity，这些 Activity 组成了 Activity 栈（Stack），当前活动的 Activity 位于栈顶，之前的 Activity 被压入下面，成为非活动 Activity，等待是否可能被恢复为活动状态。在 Activity 的生命周期中，有如表 9.1 所示的 4 个重要状态。

表 9.1　Activity 的 4 个重要状态

状　态	描　述
运行状态	当前的 Activity，位于 Activity 栈顶，用户可见，并且可以获得焦点
暂停状态	失去焦点的 Activity，仍然可见，但是在内存低的情况下，不能被系统 killed（杀死）
停止状态	该 Activity 被其他 Activity 所覆盖，不可见，但是它仍然保存所有的状态和信息。当内存低的情况下，它将会被系统 killed（杀死）
销毁状态	该 Activity 结束，或 Activity 所在的虚拟器进程结束

在了解了 Activity 的 4 个重要状态后，再来看图 9.1（参照 Android 官方文档），该图显示了一个 Activity 的各种重要状态，以及相关的回调方法。

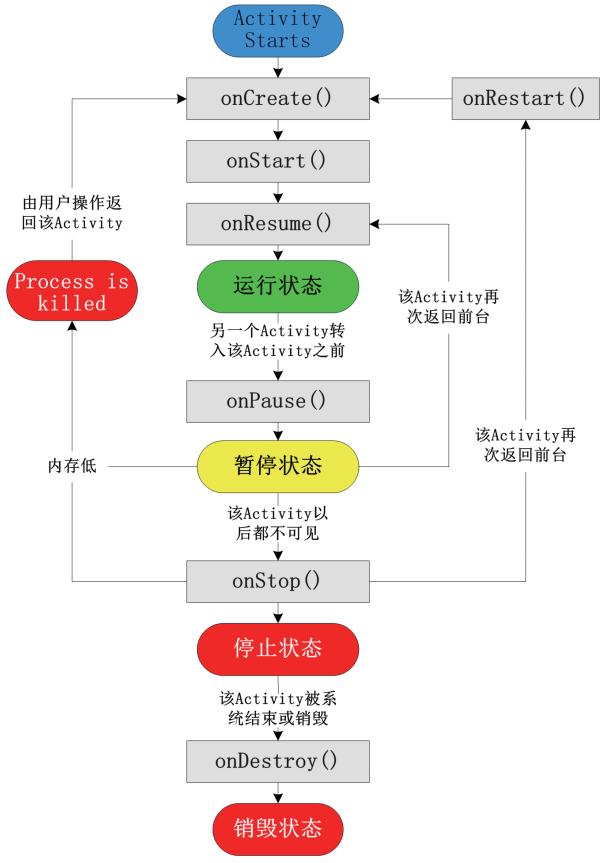


图 9.1　Activity 的生命周期及回调方法

在图 9.1 中，矩形方块表示的内容为可以被回调的方法，而有底色的椭圆形则表示 Activity 的重要状态。从该图可以看出，在一个 Activity 的生命周期中有一些方法会被系统回调，这些方法的名称及其描述如表 9.2 所示。

表 9.2　Activity 生命周期中的回调方法

方 法 名	描　　述
onCreate()	在创建 Activity 时被回调。该方法是最常见的方法，在 Android Studio 中创建 Android 项目时，会自动创建一个 Activity，在该 Activity 中，默认重写了 onCreate(Bundle savedInstanceState) 方法，用于对该 Activity 执行初始化
onStart()	启动 Activity 时被回调，也就是当一个 Activity 变为可见时被回调
onResume()	当 Activity 由暂停状态恢复为活动状态时调用。调用该方法后，该 Activity 位于 Activity 栈的栈顶。该方法总是在 onPause() 方法以后执行
onPause()	暂停 Activity 时被回调。该方法需要被非常快速地执行，因为直到该方法执行完毕后，下一个 Activity 才能被恢复。在该方法中，通常用于持久保存数据。例如，当我们正在玩游戏时，突然来了一个电话，这时就可以在该方法中将游戏状态持久保存起来
onRestart()	重新启动 Activity 时被回调，该方法总是在 onStart() 方法以后执行

续表

方 法 名	描 述
onStop()	停止 Activity 时被回调
onDestroy()	销毁 Activity 时被回调

说明 在Activity中，可以根据程序的需要来重写相应的方法。其中，onCreate()和onPause()方法是最常用的，经常需要重写这两个方法。

9.2 主角的艰辛历程

在 Android 中，Activity 提供了与用户交互的可视化界面。在使用 Activity 时，需要先对其进行创建和配置，然后才可以启动或关闭 Activity。下面将详细介绍创建、配置、启动和关闭 Activity 的方法。

9.2.1 创建 Activity

创建 Activity 的基本步骤如下：

（1）创建一个 Activity，一般是继承 android.app 包中的 Activity 类，不过在不同的应用场景下，也可以继承 Activity 的子类。例如，在一个 Activity 中，如果只实现一个列表，就可以让该 Activity 继承 ListActivity；如果只实现选项卡效果，就可以让该 Activity 继承 TabActivity。创建一个名为 MyActivity 的 Activity，具体代码如下：

```
01 import android.app.Activity;
02 public class MyActivity extends Activity {
03
04 }
```

（2）重写需要的回调方法。通常情况下，都需要重写 onCreate() 方法，并且在该方法中调用 setContentView() 方法设置要显示的页面。例如，在步骤（1）中创建的 Activity 中，重写 onCreate() 方法，并设置要显示的页面为 activity_my.xml，具体代码如下：

```
01 @Override
02 public void onCreate(Bundle savedInstanceState) {
03     super.onCreate(savedInstanceState);
04     setContentView(R.layout.activity_my);
05 }
```

另外，使用 Android Studio 也可以很方便地创建 Activity，具体步骤如下：

（1）在 Module 的包名（如 com.mingrisoft）节点上单击鼠标右键，在弹出的快捷菜单上依次选择 New → Activity → Empty Activity 菜单项，如图 9.2 所示。

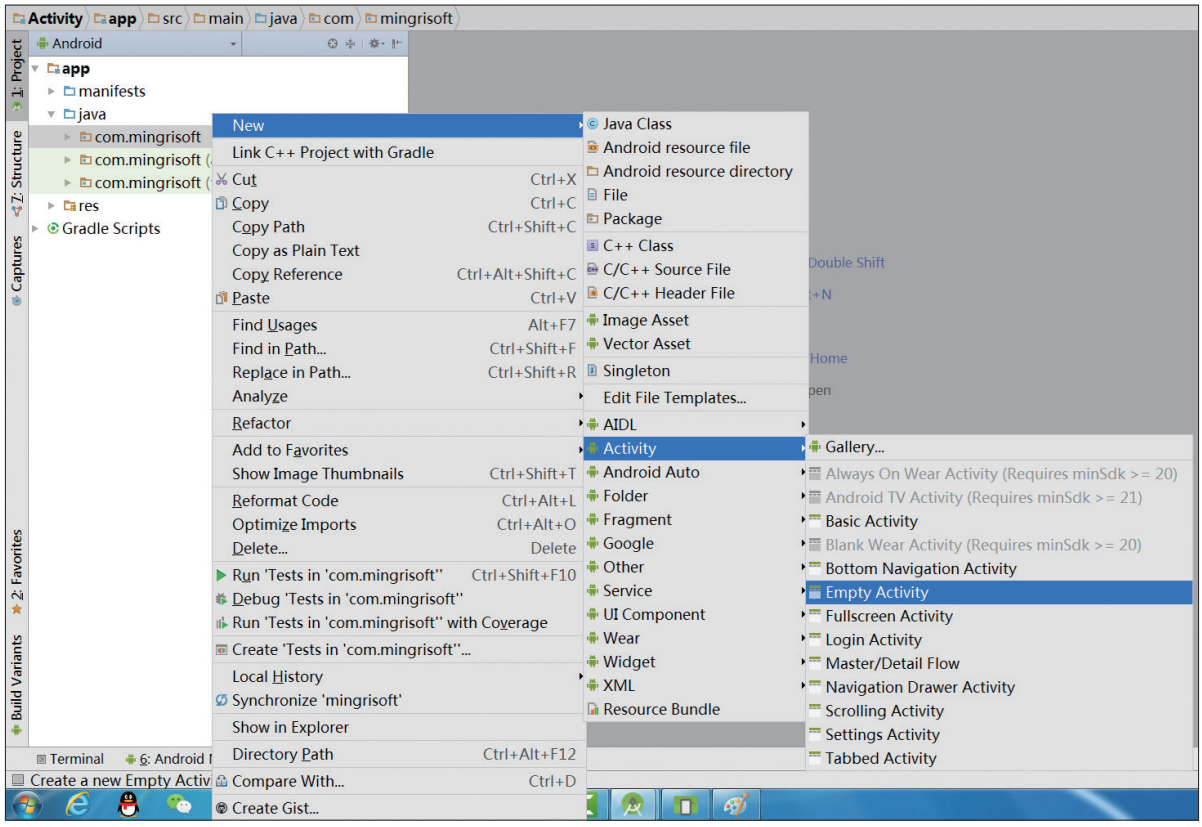


图 9.2　选择 Empty Activity 选项

（2）在弹出的对话框中修改 Activity 的名称，如图 9.3 所示。

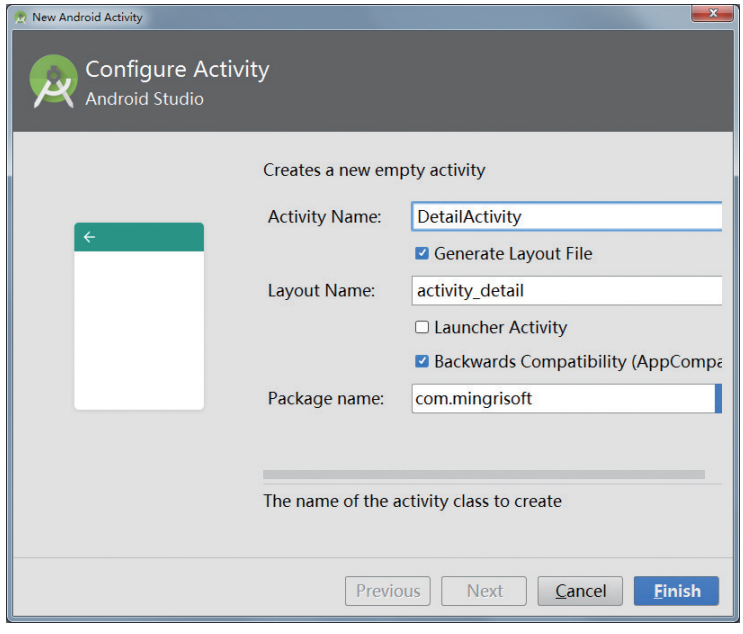


图 9.3　修改创建的 Activity 名称

（3）单击 Finish 按钮即可创建一个空的 Activity，然后就可以在该类中重写需要的回调方法。

9.2.2 配置 Activity

使用 Android Studio 向导创建 Activity 后，会自动在 AndroidManifest.xml 文件中配置该 Activity。如果没有在 AndroidManifest.xml 文件中配置，而又在程序中启动了该 Activity，将会抛出如图 9.4 所示的异常信息。

```
05-09 02:44:22.148 6787-6787/? E/AndroidRuntime: FATAL EXCEPTION: main
                                                 Process: com.mingrisoft, PID: 6787
                                                 android.content.ActivityNotFoundException: Unable to find explicit
 activity class {com.mingrisoft/com.mingrisoft.DetailActivity}; have you declared this activity in your AndroidManifest.xml?
```

图 9.4 LogCat 面板中抛出的异常信息

具体的配置方法是在 <application></application> 标记中添加 <activity></activity> 标记实现（每个 Activity 对应一个 <activity></activity> 标记）。<activity> 标记的基本格式如下：

```
<activity
android:name="实现类"
android:label="说明性文字"
android:theme="要应用的主题"
    …
    >
    …
</activity>
```

从上面格式中可以看出，配置 Activity 时通常需要指定以下几个属性：

- android:name：指定对应的 Activity 实现类。
- android:label：为该 Activity 指定标签。
- android:theme：设置要应用的主题。

说明 如果该Activity类在<manifest>标记的package属性指定的包中，则android:name属性的属性值可以直接写类名，也可以是“.类名”的形式；如果在package属性指定包的子包中，则属性值需要设置为“.子包序列.类名”或者是完整的类名（包括包路径）。

在 AndroidManifest.xml 文件中配置名称为 DetailActivity 的 Activity，该类保存在 <manifest> 标记指定的包中，关键代码如下：

```
01  <activity
02  android:name=".DetailActivity"
03  android:label="详细"
04      >
05  </activity>
```

9.2.3 启动和关闭 Activity

1. 启动 Activity

启动 Activity 分为以下两种情况。

- 在一个 Android 应用中，只有一个 Activity 时，那么只需要在 AndroidManifest.xml 文件中对

其进行配置，并且将其设置为程序的入口。这样，当运行该项目时，将自动启动该 Activity。

◆ 在一个 Android 应用中，存在多个 Activity 时，需要应用 startActivity() 方法来启动需要的 Activity。startActivity() 方法的语法格式如下：

```
public void startActivity (Intent intent)
```

该方法没有返回值，只有一个 Intent 类型的入口参数，Intent 是 Android 应用里各组件之间的通信方式，一个 Activity 通过 Intent 来表达自己的“意图”。在创建 Intent 对象时，需要指定想要被启动的 Activity。

例如，要启动一个名称为 DetailActivity 的 Activity，可以使用下面的代码：

```
01  Intent intent=new Intent(MainActivity.this,DetailActivity.class);
02  startActivity(intent);
```

2. 关闭 Activity

在 Android 中，如果想要关闭当前的 Activity，可以使用 Activity 类提供的 finish() 方法。finish() 方法的语法格式如下：

```
public void finish ()
```

该方法的使用比较简单，既没有入口参数，也没有返回值，只需要在 Activity 中相应的事件中调用该方法即可。例如，想要在单击按钮时关闭该 Activity，可以使用下面的代码：

```
01  Button button1 = (Button)findViewById(R.id.button1);
02  button1.setOnClickListener(new View.OnClickListener() {
03      @Override
04      public void onClick(View v) {
05          finish();                    //关闭当前Activity
06      }
07  });
```

说明 如果当前的Activity不是主活动，那么执行finish()方法后，将返回到调用它的那个Activity；否则，将返回到主屏幕中。

下面通过一个具体的实例来演示如何启动和关闭 Activity。

例 9.1 仿喜马拉雅 FM 实现跳转到忘记密码界面

在 Android Studio 中创建 Module，名称为“Startup And Shutdown Activity”，具体步骤如下：

（1）修改新建 Module 的 res\layout 目录下的布局文件 activity_main.xml。首先将默认添加的布局管理器修改为表格布局管理器，然后在该布局管理器中添加 1 个背景图片和 4 个 TableRow 表格行，并在每个表格行中添加相关的组件，最后设置表格的第 1 列和第 4 列允许被拉伸。

（2）在 java 目录上单击右键，然后依次选择 New → Activity → Empty Activity，创建一个名称为 PasswordActivity 的 Activity，并且设置它的布局文件为 activity_password.xml。

（3）修改 res\layout 目录中的 activity_password.xml 布局文件。首先将默认添加的布局管理器修改为垂直线性布局管理器，然后为布局管理器设置背景图片，并在该布局管理器中添加一个 ImageButton 组件（用于显示关闭按钮），在关闭按钮下面添加一个 TextView 组件（用于显示提

示文字），在该提示文字下面添加一个 EditText 组件（用于填写邮箱或账号），最后再添加一个 Button 组件（用于提交信息）。

（4）打开 PasswordActivity.java 文件，让 PasswordActivity 直接继承 Activity，并且在 onCreate() 方法中，首先获取“ ”按钮，然后为该图片按钮添加单击事件监听器，在重写的 onClick() 方法中调用 finish() 方法，关闭当前 Activity，具体代码如下：

```
01 public class PasswordActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_password);
06         ImageButton close = (ImageButton) findViewById(R.id.close);//获取布局文件中的关闭按钮
07         close.setOnClickListener(new View.OnClickListener() { //为关闭按钮添加单击监听事件
08             @Override
09             public void onClick(View v) {
10                 finish();                                    //关闭当前Activity
11             }
12         });
13     }
14 }
```

（5）打开默认创建的主活动 MainActivity，首先让 MainActivity 直接继承 Activity，在 onCreate() 方法中，获取“忘记密码”文字，并为其添加单击事件监听器，然后在重写的 onClick() 方法中，创建一个 PasswordActivity 所对应的 Intent 对象，并调用 startActivity() 方法，启动 PasswordActivity，具体代码如下：

```
01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         TextView password = (TextView) findViewById(R.id.wang_mima);//获取布局文件中的忘记密码
07         password.setOnClickListener(new View.OnClickListener() {//为忘记密码添加单击监听事件
08             @Override
09             public void onClick(View v) {
10                 //创建Intent对象
11                 Intent intent = new Intent(MainActivity.this, PasswordActivity.class);
12                 startActivity(intent);                    //启动Activity
13             }
14         });
15     }
16 }
```

（6）运行本实例，效果如图 9.5 所示界面，在第一个 Activity 中单击“忘记密码”，则进入到第二个 Activity 中，如图 9.6 所示，若单击“关闭”按钮，则关闭当前的 Activity，返回到第一个 Activity 中。

图 9.5　用户登录页面

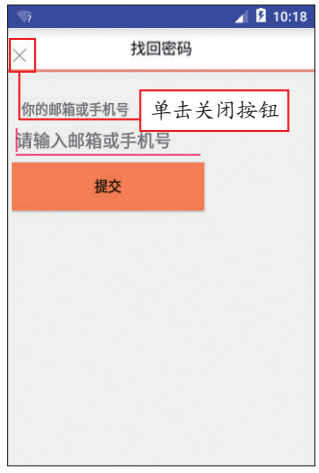


图 9.6　找回密码页面

9.2 主角之间的交流

Intent 中文意思为“意图”。它是 Android 程序中传输数据的核心对象，在 Android 官方文档中，对 Intent 的定义是执行某操作的一个抽象描述。它可以开启新的 Activity，也可以发送广播消息，或者开启 Service 服务。下面将对 Intent 及其基本应用分别进行介绍。

9.2.1 配角很重要

1. Intent 概述

一个 Android 程序主要是由 Activity、Service 和 BroadcastReceiver 三种组件组成，这三种组件是独立的，它们之间可以互相调用、协调工作，最终组成一个真正的 Android 程序。这些组件之间的通讯主要由 Intent 协助完成。Intent 负责对应用中一次操作的 Action（动作）、Action 涉及的 Data（数据）、Extras（附加数据）进行描述，Android 则根据 Intent 的描述，找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。因此，Intent 在这里起着一个媒体中介的作用，专门提供组件间互相调用的相关信息，实现调用者与被调用者之间的解耦。

例如，在一个联系人维护的应用中，在联系人列表界面（假设对应的 Activity 为 ListActivity）中，如图 9.7 所示，当单击联系人 Mr 后，会打开该联系人的详细信息界面（假设对应的 Activity 为 DetailActivity），如图 9.8 所示。

为了实现这个目的，ListActivity 需要构造一个 Intent，这个 Intent 用于告诉系统：需要完成“查看”动作，而此动作对应的查看对象是“某联系人”；然后调用 startActivity(Intent intent) 方法，并将构造的 Intent 传入，系统会根据此 Intent 中的描述，在 AndroidManifest.xml 文件中找到满足此 Intent 要求的 Activity（即 DetailActivity）；最后，DetailActivity 会根据此 Intent 中的描述，执行相应的操作，如图 9.9 所示。

图 9.7 联系人列表界面（ListActivity）

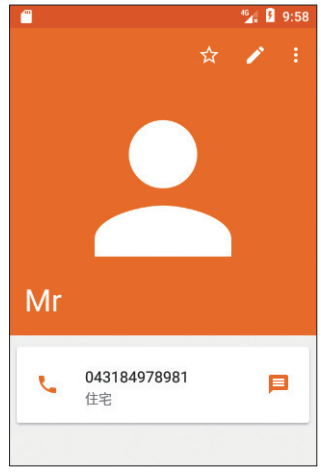


图 9.8 联系人详细信息界面（DetailActivity）

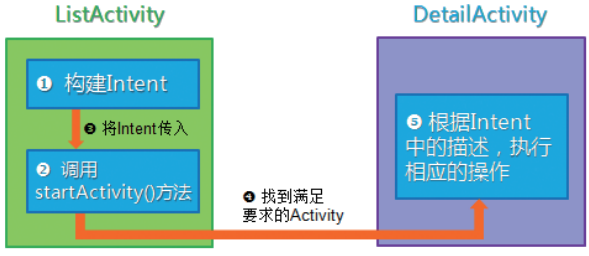


图 9.9 Intent 的作用

2. Intent 的基本应用

Intent 是一个可以从另一个应用程序请求动作的消息处理对象。它可以实现组件间的通信，主要有以下 3 种基本应用。

◆ 开启 Activity

通过将一个 Intent 对象传递给 startActivity() 方法，可以启动一个新的 Activity，并且还可以携带一些必要的数据。另外，也可以将 Intent 对象传递到 startActivityForResult() 方法中，这样，在需要获取返回结果时，就可以在调用它的 Activity 的 onActivityResult() 方法中接收返回结果了。

◆ 开启 Service

通过将一个 Intent 对象传递给 startService() 方法，可以启动一个 Service 来完成一次操作（如下载文件）或者传递一个新的指令给正在运行的 Service。另外，将一个 Intent 对象传递给 bindService() 方法，可以建立调用组件和目标服务之间的连接。

◆ 传递 Broadcast（广播）

通过任何一个广播方法（如 sendBroadcast()、sendOrderedBroadcast() 或 sendStickyBroadcast() 方法等），都可以将广播传递给所有感兴趣的广播接收者。

说明 Android程序会自动查找合适的Activity、Service或者BroadcastReceiver来响应Intent（意图），如果初始化这些消息的系统之间没有重叠，那么BroadcastReceiver的意图只会传递给广播接收者，而不会传递给Activity或Service。

9.2.2 显式 Intent

显式 Intent 是指在创建 Intent 对象时，就指定接收者（如 Activity、Service 或者 BroadcastReceiver），因为我们已经知道要启动的 Activity 或者 Service 的类名称。由于 Service 还没有介绍，所以这里将以 Activity 为例介绍如何使用显式 Intent。

在启动 Activity 时必须在 Intent 中指明要启动的 Activity 所在的类。通常情况下，在一个 Android 项目中，如果只有一个 Activity，那么只需要在 AndroidManifest.xml 文件中配置，并且将其设置为程序的入口。这样，当运行该项目时，将自动启动该 Activity。否则，需要应用 Intent 和 startActivity() 方法来启动需要的 Activity，即通过显式 Intent 来启动，具体步骤如下：

（1）创建 Intent 对象，可以使用下面的语法格式：

```
Intent intent = new Intent(Context packageContext, Class<?> cls)
```

◆ intent：用于指定对象名称；

◆ packageContext：用于指定一个启动 Activity 的上下文对象，可以使用 Activity 名 .this（如 MainActivity.this）来指定；

◆ cls：用于指定要启动的 Activity 所在的类，可以使用 Activity 名 .class（如 DetailActivity.class）来指定。

说明 Intent位于android.content包中，在使用Intent时，需要应用 “import android.content.Intent;” 语句导入该类。

例如，创建一个启动 DetailActivity 的 Intent 对象，可以使用下面的代码。

```
Intent intent=new Intent(MainActivity.this,DetailActivity.class);
```

（2）应用 startActivity() 方法来启动 Activity。startActivity() 方法的语法格式如下：

```
public void startActivity (Intent intent)
```

该方法没有返回值，只有一个 Intent 类型的入口参数，该 Intent 对象为步骤（1）中创建的 Intent 对象。

9.2.3 隐式 Intent

隐式 Intent 是指在创建 Intent 对象时，不指定具体的接收者，而是定义要执行的 Action、Category 和 Data，然后让 Android 系统根据相应的匹配机制找到要启动的 Activity。例如，在 Activity A 中隐式启动 Activity B 需要经过如图 9.10 所示的过程。

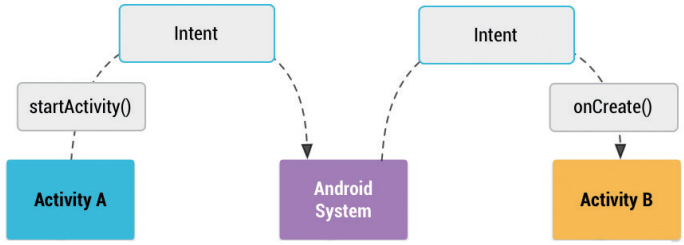


图 9.10　隐式 Intent 示意图

说明 从图9.10可以看出，在Activity A中，创建一个设置了Action的Intent对象，并且把它传递到startActivity()中，然后Android系统将搜索所有的应用程序来匹配这个Intent，当找到匹配后，系统将通过传递Intent到onCreate()方法来启动匹配的Activity B。

使用隐式 Intent 启动 Activity 时，需要为 Intent 对象定义 Action、Category 和 Data 属性，然后再调用 startActivity() 方法来启动匹配的 Activity。

例如，我们要在自己的应用程序中展示一个网页，就可以直接调用系统的浏览器来打开这个网页，而不必自己再编写一个浏览器。可以使用下面的语句实现。

```
01  01 Intent intent = new Intent();                          //创建Intent对象
02  02 intent.setAction(Intent.ACTION_VIEW);                  //为Intent设置动作
03  03 intent.setData(Uri.parse("http://www.mingribook.com")); //为Intent设置数据
04  04 startActivity(intent);                                 //将Intent传递给Activity
```

也可以使用下面的语句实现。

```
01  Intent intent = new Intent(Intent.ACTION_VIEW,
02  Uri.parse("http://www.mingribook.com"));          //创建Intent对象
03  startActivity(intent);                            //将Intent传递给Activity
```

◆ Intent.ACTION_VIEW：为 Intent 的 action，表示需要执行的动作。Android 系统支持的标准 Activity 动作说明如表 9.3 所示。

表 9.3 标准 Activity 动作说明

常　量	对应字符串	说　明
ACTION_MAIN	android.intent.action.MAIN	作为初始的 Activity 启动，没有数据输入输出
ACTION_VIEW	android.intent.action.VIEW	将数据显示给用户
ACTION_ATTACH_DATA	android.intent.action.ATTACH_DATA	用于指示一些数据应该附属于其他地方
ACTION_EDIT	android.intent.action.EDIT	将数据显示给用户用于编辑
ACTION_PICK	android.intent.action.PICK	从数据中选择一项，并返回该项
ACTION_CHOOSER	android.intent.action.CHOOSER	显示一个 Activity 选择器
ACTION_GET_CONTENT	android.intent.action.GET_CONTENT	允许用户选择特定类型的数据并将其返回
ACTION_DIAL	android.intent.action.DIAL	使用提供的数字拨打电话
ACTION_CALL	android.intent.action.CALL	使用提供的数据给某人拨打电话
ACTION_SEND	android.intent.action.SEND	向某人发送消息，接收者未指定
ACTION_SENDTO	android.intent.action.SENDTO	向某人发送消息，接收者已指定
ACTION_ANSWER	android.intent.action.ANSWER	接听电话
ACTION_INSERT	android.intent.action.INSERT	在给定容器中插入空白项

续表

常　量	对应字符串	说　明
ACTION_DELETE	android.intent.action.DELETE	从容器中删除给定数据
ACTION_RUN	android.intent.action.RUN	无条件运行数据
ACTION_SYNC	android.intent.action.SYNC	执行数据同步
ACTION_PICK_ACTIVITY	android.intent.action.PICK_ACTIVITY	选择给定 Intent 的 Activity，返回选择的类
ACTION_SEARCH	android.intent.action.SEARCH	执行查询
ACTION_WEB_SEARCH	android.intent.action.WEB_SEARCH	执行联机查询
ACTION_FACTORY_TEST	android.intent.action.FACTORY_TEST	工厂测试的主入口点

说明 关于表9.3内容的详细说明请参考API文档中Intent类的说明。

◆ Uri.parse() 方法：用于把字符串解释为 URI 对象，表示需要传递的数据。

在执行上面的代码时，系统首先根据 Intent.ACTION_VIEW 得知需要启动具备浏览功能的 Activity，但是具体的浏览内容，还需要根据第二个参数的数据类型来判断。这里面提供的是 Web 地址，所以将使用内置的浏览器显示。

下面通过一个实例演示如何实现隐式启动 Intent。

例 9.2 隐式 Intent 实现拨打电话功能

在 Android Studio 中创建一个 Module，名称为“Intent Dial”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后在布局管理器中添加 4 个用于显示公司信息的文本框，然后添加两个 ImageButton 组件，分别为“拨打电话”按钮和“发送短信”按钮。

（2）修改 MainActivity.java 文件，在 onCreate() 方法中，获取布局文件中的电话图片按钮和短信图片按钮，并为它们设置单击事件监听器，代码如下：

```
01  public class MainActivity extends AppCompatActivity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //获取电话图片按钮
07          ImageButton imageButton = (ImageButton) findViewById(R.id.imageButton_phone);
08          //获取短信图片按钮
09          ImageButton imageButton1 = (ImageButton) findViewById(R.id.imageButton_sms);
10          imageButton.setOnClickListener(listener);    //为电话图片按钮设置单击事件
11          imageButton1.setOnClickListener(listener);   //为短信图片按钮设置单击事件
12      }
13  }
```

（3）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型。因此，在 Activity 中创建该对象，并重写其 onClick() 方法，在该方法中，通过判断单击按钮的 id，分别为两个 ImageButton 组件设置拨打电话和发送短信的 Action 及 Data，代码如下：

```
//创建监听事件对象
View.OnClickListener listener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();                          //创建Intent对象
        switch (v.getId()) {                        //根据ImageButton组件的id进行判断
            case R.id.imageButton_phone:                       //如果是电话图片按钮
                intent.setAction(intent.ACTION_DIAL);          //调用拨号面板
                intent.setData(Uri.parse("tel:043184978981"));//设置要拨打的号码
                startActivity(intent);                         //启动Activity
                break;
            case R.id.imageButton_sms:                         //如果是短信图片按钮
                intent.setAction(intent.ACTION_SENDTO);        //调用发送短信息面板
                intent.setData(Uri.parse("smsto:5554"));       //设置要发送的号码
                intent.putExtra("sms_body", "Welcome to Android!"); //设置要发送的信息内容
                startActivity(intent);                         //启动Activity
        }
    }
};
```

（4）在 AndroidManifest.xml 文件中，设置允许该应用拨打电话和发送短信的权限，代码如下：

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

（5）运行本实例，将显示如图 9.11 所示的关于明日学院界面，在该界面中，若单击电话图标按钮将显示如图 9.12 所示的拨打电话界面；若单击短信图标按钮将显示如图 9.13 所示的发送信息界面。

图 9.11　主 Activity 界面

图 9.12　拨打电话界面

图 9.13　发送信息

9.2.4 Intent 过滤器

使用隐式 Intent 启动 Activity 时，并没有在 Intent 中指明 Activity 所在的类。因此，Android 系统要根据某种匹配机制，找到要启动的 Activity。这种机制就是根据 Intent 过滤器来实现的。

说明 Intent过滤器是一种根据Intent中的Action、Data和Category等属性对适合接收该Intent的组件进行匹配和筛选的机制。

为了使组件能够注册 Intent 过滤器，通常在 AndroidManifest.xml 文件的各个组件声明标记中，使用 <intent-filter> 标记声明该组件所支持的动作、数据和种类等信息。当然，也可以在程序代码中，使用 Intent 对象提供的对应属性的方法来进行设置。这里主要介绍通过 <intent-filter> 标记在 AndroidManifest.xml 文件中进行配置。在 <intent-filter> 标记中，用于设置 Action 属性的标记为 <action>；用于设置 Data 属性的标记为 <data>；用于设置 Category 属性的标记为 <category>。下面将对这几个标记进行详细介绍。

1. 配置 <action> 标记

<action> 标记用于指定组件所能响应的动作，以字符串形式表示，通常由 Java 类名和包的完全限定名组成。<action> 标记的语法格式如下：

```
<action android:name="string" />
```

其中，string 为字符串，可以是表 9.3 中的“对应字符串”列的内容，但不能直接使用类常量。例如，要设置其作为初始启动 Activity（对应常量为 ACTION_MAIN），那么需要将其指定为 android.intent.action.MAIN。代码如下：

```
<action android:name="android.intent.action.MAIN"/>
```

除了使用标准的 Action 常量外，还可以自定义 action 的名字，为了确保名字的唯一性，一定要用该应用程序的包名作为前缀。例如，要设置名字为 DETAIL，可以使用下面的代码。

```
<action android:name="com.mingrisoft.action.DETAIL "/>
```

2. 配置 <data> 标记

<data> 标记用于向 Action 提供要操作的数据。它可以是一个 URI 对象或者数据类型（MIME 媒体类型）。其中，URI 可以分成 scheme（协议或服务方式）、host（主机）、port（端口）以及 path（路径）等，格式如下：

```
<scheme>://<host>:<port>/<path>
```

例如下面的 URI：

```
content://com.example.project:200/folder/subfolder/etc
```

其中， content 是 scheme； com.example.project 是 host； 200 是 port； folder/subfolder/etc 是 path。host 和 port 一起组成了 URI 授权，如果 host 没有指定，则忽略 port。这些属性都是可选的，但是相互之间并非完全独立。如果授权有效，则 scheme 必须指定。如果 path 有效，则 scheme 和授权必须指定。

```
<data>标记的语法格式如下：
<data android:scheme="string"
    android:host="string"
    android:port="string"
    android:path="string"
    android:mimeType="string" />
```

◆ android:scheme：用于指定所需要的特定协议；

◆ android:host：用于指定一个有效的主机名；

◆ android:port：用于指定主机的有效端口号；

◆ android:path：用于指定有效的 URI 路径名；

◆ android:mimeType：用于指定组件能处理的数据类型，支持使用“*”通配符来包含子类型（如 image/* 或者 audio/*）。在过滤器中，该属性比较常用。

例如，要设置数据类型为 JPG 图片，可以使用下面的代码。

```
<data android:mimeType="image/jpeg"/>
```

3. 配置 <category> 标记

<category> 标记用于指定以何种方式去执行 Intent 请求的动作。<category> 标记的语法格式如下：

```
<category android:name="string" />
```

其中，string 为字符串，可以是表 9.4 中的“对应字符串”列的内容，但不能直接使用类常量。

表 9.4 标准 Category 说明

常　量	对应字符串	说　明
CATEGORY_DEFAULT	android.intent.category.DEFAULT	将 Activity 作为默认动作选项
CATEGORY_BROWSABLE	android.intent.category.BROWSABLE	让 Activity 能够安全地从浏览器中调用
CATEGORY_TAB	android.intent.category.TAB	将 Activity 作为 TabActivity 的选项卡
CATEGORY_ALTERNATIVE	android.intent.category.ALTERNATIVE	将 Activity 作为用户正在查看数据的备用动作
CATEGORY_SELECTED_ALTERNATIVE	android.intent.category.SELECTED_ALTERNATIVE	将 Activity 作为用户当前选择数据的备用动作
CATEGORY_LAUNCHER	android.intent.category.LAUNCHER	让 Activity 在顶层启动器中显示
CATEGORY_INFO	android.intent.category.INFO	用于提供 Activity 所在包的信息
CATEGORY_HOME	android.intent.category.HOME	用于返回 Home Activity（系统桌面）
CATEGORY_PREFERENCE	android.intent.category.PREFERENCE	让 Activity 作为一个偏好面板
CATEGORY_TEST	android.intent.category.TEST	用于测试

续表

常　量	对应字符串	说　明
CATEGORY_CAR_DOCK	android.intent.category.CAR_DOCK	用于在设备插入到 car dock 时运行 Activity
CATEGORY_DESK_DOCK	android.intent.category.DESK_DOCK	用于在设备插入到 desk dock 时运行 Activity
CATEGORY_LE_DESK_DOCK	android.intent.category.LE_DESK_DOCK	用于在设备插入到模拟 dock（低端）时运行 Activity
CATEGORY_HE_DESK_DOCK	android.intent.category.HE_DESK_DOCK	用于在设备插入到数字 dock（高端）时运行 Activity
CATEGORY_CAR_MODE	android.intent.category.CAR_MODE	指定 Activity 可以用于汽车环境
CATEGORY_APP_MARKET	android.intent.category.APP_MARKET	让 Activity 允许用户浏览和下载新应用

说明 关于表9.4内容详细说明请参考API文档中Intent类的说明。

例如，要设置其作为用于测试的 Activity（对应常量为 CATEGORY_TEST），那么需要将其指定为 android.intent.category.TEST。代码如下：

```
<action android:name="android.intent.category.TEST"/>
```

除了使用标准的 Category 常量外，还可以自定义 Category 的名字，为了确保名字的唯一性，一定要用该应用程序的包名作为前缀，例如，要设置名字为 DETAIL，可以使用下面的代码。

```
<action android:name="com.mingrisoft.category.DETAIL"/>
```

下面通过一个实例说明 Intent 过滤器的具体应用。

例 9.3 使用 Intent 过滤器实现查看大图功能

在 Android Studio 中创建 Module，名称为“Intent Filter”。在该 Module 中实现本实例，具体步骤如下：

（1）修改新建 Module 的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将文本框组件删除，添加一个 ImageView 组件（用于显示小图）和一个 Button 组件（单击查看大图）。

（2）打开 MainActivity，在 onCreate() 方法中获得布局文件中的 Button 组件并为其添加单击事件监听器。在监听器中传递包含动作的隐式 Intent，其代码如下：

```
01 public class MainActivity extends AppCompatActivity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         Button button= (Button) findViewById(R.id.btn); //获取按钮组件
07         //为按钮创建单击事件
```

```
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent();              //创建Intent对象
                intent.setAction(intent.ACTION_VIEW);    //为Intent设置动作
                startActivity(intent);                   //启动Activity
            }
        });
    }
}
```

注意 在上面的代码中，并没有指定将Intent对象传递给哪个Activity。

（3）创建名称为 ContactsActivity 的 Activity，让 ContactsActivity 直接继承 Activity，并且在 onCreate() 方法中设置全屏显示。

（4）在 res\layout 目录中找到名称为 activity_contacts.xml 的文件，将默认添加的布局管理器修改为相对布局管理器，然后添加一个 ImageView 组件，用于设置图片，关键代码如下：

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ContactsActivity">
<ImageView
    android:id="@+id/image1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:src="@mipmap/hehua"
    android:scaleType="fitXY"/>
</RelativeLayout>
```

（5）编写 AndroidManifest.xml 文件，为两个 Activity 设置不同的 Intent 过滤器，其代码如下：

```
<manifest
    package="com.mingrisoft"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
```

```
15          </activity>
16          <activity android:name=".ContactsActivity">
17              <intent-filter>
18                  <action android:name="android.intent.action.VIEW"/>
19                  <category android:name="android.intent.category.DEFAULT"/>
20              </intent-filter>
21          </activity>
22      </application>
23  </manifest>
```

说明 由于有多种匹配ACTION_VIEW的方式，因此需要用户进行选择。

（6）运行本实例，将显示如图 9.14 所示的主界面，单击“查看大图”按钮，显示如图 9.15 所示的选择打开方式界面，选择“Intent 过滤器”跳转到第二个 Activity，显示完整图片。

图 9.14 选择发送方式界面

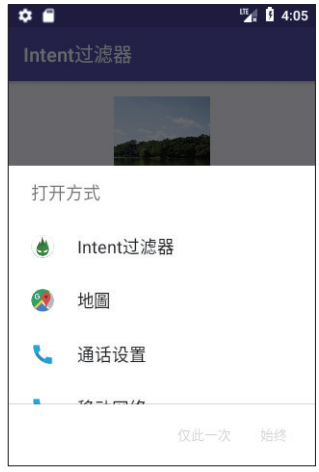


图 9.15 选择打开方式界面

9.2.5 多个 Activity 之间交换数据

当在一个 Activity 中启动另一个 Activity 时，经常需要传递一些数据。这时就可以通过 Intent 来实现，因为 Intent 通常被称为是两个 Activity 之间的信使，通过将要传递的数据保存在 Intent 中，就可以将其传递到另一个 Activity 中了。在 Android 中，可以将要保存的数据存放在 Bundle 对象中，然后通过 Intent 提供的 putExtras() 方法将要携带的数据保存到 Intent 中。通过 Intent 传递数据的示意图如图 9.16 所示。

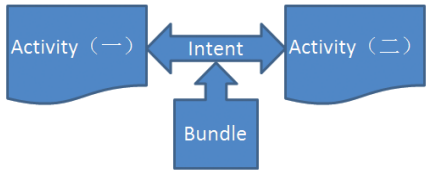


图 9.16 通过 Intent 传递数据

说明 Bundle是一个key-value（键-值）对的组合，用于保存要携带的数据包。这些数据可以是

boolean、byte、int、long、float、double和String等基本类型或者对应的数组，也可以是对象或者对象数组。如果是对象或者对象数组时，必须实现Serializable或者Parcelable接口。

下面通过一个实例介绍如何使用 Bundle 在 Activity 之间交换数据。

例 9.4 模拟保存淘宝收货地址

在 Android Studio 中创建 Module，名称为“Activity Exchange Data”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res\layout 目录下的布局文件 activity_main.xml。首先将 . 默认添加的布局管理器修改为相对布局管理器，然后添加一个 ImageView 组件，用于存放导航条图片，在下面再添加用于输入地址信息的 6 个编辑框和一个“保存”按钮。

（2） 打开默认创建的主活动 MainActivity，然后让 MainActivity 直接继承 Activity，并且在 onCreate() 方法中，获取“保存”按钮，并为其添加单击事件监听器。在重写的 onClick() 方法中，首先获取输入的地区、街道、详细地址、姓名、电话和邮箱，并保存到相应的变量中，然后判断输入信息是否为空，如果为空，则给出消息提示，如果不为空，将输入的信息保存到 Bundle 中，并启动一个新的 Activity 显示输入的收货地址信息，具体代码如下：

```
01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         Button btn = (Button) findViewById(R.id.btn);          //获取保存按钮
07         btn.setOnClickListener(new View.OnClickListener() { //为按钮添加单击监听事件
08             @Override
09             public void onClick(View v) {
10              //获取输入的所在地区
11              String site1 = ((EditText) findViewById(R.id.et_site1)).getText().toString();
12              //获取输入的所在街道
13              String site2 = ((EditText) findViewById(R.id.et_site2)).getText().toString();
14              //获取输入的详细地址
15              String site3 = ((EditText) findViewById(R.id.et_site3)).getText().toString();
16              //获取输入的用户信息
17              String name = ((EditText) findViewById(R.id.et_name)).getText().toString();
18              //获取输入的手机号码
19              String phone = ((EditText) findViewById(R.id.et_phone)).getText().toString();
20              //获取输入的邮箱
21              String email= ((EditText) findViewById(R.id.et_email)).getText().toString();
22                 if (!"".equals(site1) && !"".equals(site2) && !"".equals(site3)&&
23                         !"".equals(name) && !"".equals(phone) &&!"".equals(email) ) {
24                   //将输入的信息保存到Bundle中，通过Intent传递到另一个Activity中显示出来
25                    Intent intent = new Intent(MainActivity.this, AddressActivity.class);
26                     //创建并实例化一个Bundle对象
27                     Bundle bundle = new Bundle();
28                     bundle.putCharSequence("name", name);     //保存姓名
29                     bundle.putCharSequence("phone", phone);  //保存手机号码
```

```
                    bundle.putCharSequence("site1", site1);  //保存所在地区信息
                    bundle.putCharSequence("site2", site2);  //保存所在街道信息
                    bundle.putCharSequence("site3", site3);  //保存详细地址信息
                    intent.putExtras(bundle);      //将Bundle对象添加到Intent对象中
                    startActivity(intent);                  //启动Activity
                }else {
                    Toast.makeText(MainActivity.this,
                            "请将收货地址填写完整！",Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

（3）在工具窗口中的 Activity Exchange Data 节点上单击鼠标右键，在弹出的快捷菜单中选择 New → Activity → Empty Activity 菜单项，然后在弹出的自定义 Activity 对话框中，修改 Activity 的名称为 AddressActivity，单击“完成”按钮，创建一个 AddressActivity，并且自动创建一个名称为 activity_address.xml 的布局文件。

（4）修改 res\layout 目录中的 activity_address.xml 布局文件。首先将默认的布局管理器修改为相对布局管理器，然后添加一个 ImageView 组件，用于存放导航条图片，再添加 3 个 TextView 组件，分别用于显示姓名、电话和地址。由于此处的布局代码比较简单，这里不再给出。

（5）打开 AddressActivity.java，让 AddressActivity 直接继承 Activity，在 onCreate() 方法中，首先获取 Intent 对象和传递的数据包，然后将传递过来的姓名、电话和地址显示到对应的 TextView 组件中。关键代码如下：

```
public class AddressActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_address);
        Intent intent = getIntent();                    //获取Intent对象
        Bundle bundle = intent.getExtras();             //获取传递的Bundle信息
        TextView name = (TextView) findViewById(R.id.name);//获取显示姓名的TextView组件
        name.setText(bundle.getString("name"));//获取输入的姓名并显示到TextView组件中
        //获取显示手机号码的TextView组件
        TextView phone = (TextView) findViewById(R.id.phone);
        //获取输入的电话号码并显示到TextView组件中
        phone.setText(bundle.getString("phone"));
        TextView site = (TextView) findViewById(R.id.site);//获取显示地址的TextView组件
        //获取输入的地址并显示到TextView组件中
        site.setText(bundle.getString("site1")
                + bundle.getString("site2") + bundle.get("site3"));
    }
}
```

（6）运行本实例，将显示收货地址管理界面，如图 9.17 所示，填写收货地址，单击“保存”按钮后，将启动第二个 Activity，并显示填写好的收货地址，如图 9.18 所示。

图 9.17　填写收货地址信息界面

图 9.18　显示收货地址信息界面

9.2.6 调用另一个 Activity 并返回结果

在 Android 应用开发时，有时需要在一个 Activity 中调用另一个 Activity，当用户在第二个 Activity 中选择完成后，程序将自动返回到第一个 Activity 中，第一个 Activity 能够获取并显示用户在第二个 Activity 中选择的结果。例如，用户在修改信息的时候可以对头像进行修改，在修改头像时首先需要调用选择头像的界面，效果如图 9.19 所示。在选择头像后会自动返回到修改信息界面，并显示用户选择的新头像，效果如图 9.20 所示。

图 9.19　选择头像界面

图 9.20　显示选择后的头像界面

此功能也可以通过 Intent 和 Bundle 来实现。与在两个 Activity 之间交换数据不同的是，此处需要使用 startActivityForResult() 方法来启动另一个 Activity。调用 startActivityForResult() 方法启动 Activity 后，关闭新启动的 Activity 时，可以将选择的结果返回到原 Activity 中。startActivityForResult() 方法的语法格式如下：

```
public void startActivityForResult (Intent intent, int requestCode)
```

该方法将以指定的请求码启动 Activity，并且程序将会获取新启动的 Activity 返回的结果（通过重写 onActivityResult() 方法来获取）。requestCode 参数代表了启动 Activity 的请求码，该请求码的值由开发者根据业务自行设置，用于标识请求来源。

下面通过一个实例介绍如何调用另一个 Activity 并返回结果。

例 9.5 模拟喜马拉雅 FM 实现选择头像功能

在 Android Studio 中创建 Module，名称为“Select Avatar”。在该 Module 中实现本实例，具体步骤如下。

（1）修改新建 Module 的 res\layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器，在该布局管理器中，添加一个背景图片，将需要的背景图片复制到 drawable-mdpi 中，并将默认添加的 TextView 组件和内边距代码删除，然后添加一个 ImageView 组件，让这个组件水平居中，最后在 ImageView 组件下方添加一个 Button 按钮组件。

（2）打开默认创建的主活动 MainActivity，然后让 MainActivity 直接继承 Activity，在 onCreate() 方法中，获取“选择头像”按钮，并为其添加单击事件监听器，在重写的 onClick() 方法中，创建一个要启动的 Activity 对应的 Intent 对象，并应用 startActivityForResult() 方法启动指定的 Activity，等待返回结果，具体代码如下：

```
Button button= (Button) findViewById(R.id.btn);          //获取选择头像按钮
button.setOnClickListener(new View.OnClickListener() { //为按钮添加单击事件
    @Override
    public void onClick(View v) {
        //创建Intent对象
        Intent intent=new Intent(MainActivity.this,HeadActivity.class);
        startActivityForResult(intent, 0x11);          //启动intent对应的Activity
    }
});
```

（3）创建一个新的 Activity，名称为 HeadActivity，对应的布局文件为 activity_ head.xml。

（4）在 res\layout 目录中找到名称为 activity_head.xml 的布局文件，在该布局文件中添加一个 GridView 组件，用于显示可选择的头像列表，关键代码如下：

```
<GridView
    android:id="@+id/gridView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10dp"
    android:horizontalSpacing="3dp"
    android:verticalSpacing="3dp"
    android:numColumns="4">
</GridView>
```

（5）打开 HeadActivity.java，让 HeadActivity 直接继承 Activity，然后在 onCreate() 方法的上方定义一个保存要显示头像 ID 的一维数组，关键代码如下：

```
public int[] imageId = new int[]{R.mipmap.touxiang1, R.mipmap.touxiang2,
        R.mipmap.touxiang3, R.mipmap.touxiang4, R.mipmap.touxiang5
       };                                    //定义并初始化保存头像id的数组
```

（6）在重写的 onCreate() 方法中，获取 GridView 组件，并创建一个与之关联的 BaseAdapter 适配器，关键代码如下：

```
GridView gridview = (GridView) findViewById(R.id.gridView);//获取GridView组件
BaseAdapter adapter=new BaseAdapter() {
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageview;                                    //声明ImageView的对象
        if(convertView==null){
            imageview=new ImageView(HeadActivity.this);    //实例化ImageView的对象
            /*************设置图像的宽度和高度*****************/
            imageview.setAdjustViewBounds(true);
            imageview.setMaxWidth(158);
            imageview.setMaxHeight(150);
            /*************************************************/
            imageview.setPadding(5, 5, 5, 5);              //设置ImageView的内边距
        }else{
            imageview=(ImageView)convertView;
        }
        imageview.setImageResource(imageId[position]);  //为ImageView设置要显示的图片
        return imageview;                                       //返回ImageView
    }
    /*
     * 功能：获得当前选项的ID
     */
    @Override
    public long getItemId(int position) {
        return position;
    }
    /*
     * 功能：获得当前选项
     */
    @Override
    public Object getItem(int position) {
        return position;
    }
    /*
```

```
35          * 获得数量
36          */
37         @Override
38         public int getCount() {
39             return imageId.length;
40         }
41     };
42     gridview.setAdapter(adapter);          //将适配器与GridView关联
```

（7）为 GridView 添加 OnItemClickListener 事件监听器，在重写的 onItemClick() 方法中，首先获取 Intent 对象，然后创建一个要传递的数据包，并将选中的头像 ID 保存到该数据包中，再将要传递的数据包保存到 Intent 中，并设置返回的结果码及返回的 Activity，最后关闭当前 Activity。关键代码如下：

```
01  gridview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
02      @Override
03      public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
04          Intent intent = getIntent(); //获取Intent对象
05          Bundle bundle = new Bundle();//实例化要传递的数据包
06          bundle.putInt("imageId", imageId[position]);//显示选中的图片
07          intent.putExtras(bundle);     //将数据包保存到intent中
08          setResult(0x11, intent);      //设置返回的结果码，并返回调用该Activity的Activity
09          finish();                     //关闭当前Activity
10      }
11  });
```

（8）重新打开 MainActivity，在该类中，重写 onActivityResult() 方法，在该方法中，需要判断 requestCode 请求码和 resultCode 结果码是否与预先设置的相同，如果相同，则获取传递的数据包，并从该数据包中获取选择的头像 ID 并显示。具体代码如下：

```
01  @Override
02  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
03      super.onActivityResult(requestCode, resultCode, data);
04      if(requestCode==0x11 && resultCode==0x11){ //判断是否为待处理的结果
05          Bundle bundle=data.getExtras();         //获取传递的数据包
06          int imageId=bundle.getInt("imageId");   //获取选择的头像ID
07          //获取布局文件中添加的ImageView组件
08          ImageView iv=(ImageView)findViewById(R.id.imageView);
09          iv.setImageResource(imageId);           //显示选择的头像
10      }
11  }
```

（9）运行本实例，在选择头像界面中单击“选择头像”按钮，如图 9.21 所示，打开新的 Activity 选择头像后，如图 9.22 所示，将选择的头像返回到原 Activity 中。

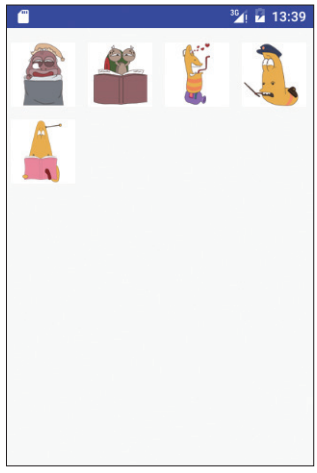


图 9.21　选择头像界面

图 9.22　模拟喜马拉雅 FM 选择头像的界面

9.3 使用碎片（Fragment）

Fragment 是 Android 3.0 新增的概念，其中文意思是碎片，它与 Activity 十分相似，用来在一个 Activity 中描述一些行为或一部分用户界面。使用多个 Fragment 可以在一个单独的 Activity 中建立多个 UI 面板，也可以在多个 Activity 中重用 Fragment。例如，微信主界面就相当于一个 Activity，在这个 Activity 中包含多个 Fragment，其中“微信”“通讯录”“发现”和“我”这 4 个功能界面，每一个功能界面就相当于一个 Fragment，它们之间可以随意切换，如图 9.23 和图 9.24 所示。

图 9.23　“发现”功能界面

图 9.24　“我”功能界面

9.3.1 Fragment 的生命周期

和 Activity 一样，Fragment 也有自己的生命周期。一个 Fragment 必须被嵌入到一个 Activity 中，

它的生命周期直接受其所属的宿主 Activity 的生命周期影响。例如，当 Activity 被暂停时，其中的所有 Fragment 也被暂停；当 Activity 被销毁时，所有隶属于它的 Fragment 也将被销毁。然而，当一个 Activity 正在运行时（处于 resumed 状态），可以单独地对每一个 Fragment 进行操作，如添加或删除等。Fragment 完整的生命周期如图 9.25 所示。

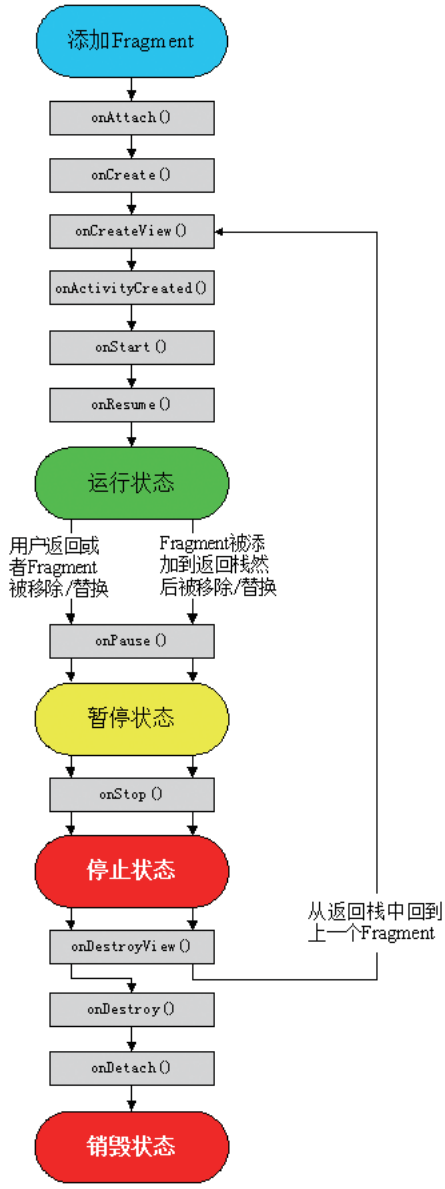


图 9.25　Fragment 的生命周期示意图

9.3.2 创建 Fragment

要创建一个 Fragment，必须创建一个 Fragment 的子类，或者继承自另一个已经存在的 Fragment 的子类。例如，要创建一个名称为 NewsFragment 的 Fragment，并重写 onCreateView() 方法，可以使用下面的代码：

```
01  public class NewsFragment extends Fragment {
02      @Override
03      public View onCreateView(LayoutInflater inflater,
04                               ViewGroup container, Bundle savedInstanceState) {
05          //从布局文件news.xml 加载一个布局文件
06          View v = inflater.inflate(R.layout.news, container, false);
07          return v;
08      }
09  }
```

说明 当系统首次调用Fragment时，如果想绘制一个UI界面，那么在Fragment中，必须重写onCreateView()方法返回一个View；如果Fragment没有UI界面，可以返回null。

9.3.3 在 Activity 中添加 Fragment

向 Activity 中添加 Fragment，有两种方法：一种是直接在布局文件中添加，将 Fragment 作为 Activity 整个布局的一部分；另一种是当 Activity 运行时，将 Fragment 放入 Activity 布局中。下面分别进行介绍。

1. 直接在布局文件中添加 Fragment

直接在布局文件中添加 Fragment 可以使用 <fragment></fragment> 标记实现。例如，要在一个布局文件中添加两个 Fragment，可以使用下面的代码：

```
01  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
02      android:layout_width="fill_parent"
03      android:layout_height="fill_parent"
04      android:orientation="horizontal" >
05     <fragment android:name="com.mingrisoft.ListFragment"
06         android:id="@+id/list"
07         android:layout_weight="1"
08         android:layout_width="0dp"
09         android:layout_height="match_parent" />
10     <fragment android:name="com.mingrisoft.DetailFragment"
11         android:id="@+id/detail"
12         android:layout_weight="2"
13         android:layout_marginLeft="20dp"
14         android:layout_width="0dp"
15         android:layout_height="match_parent" />
16  </LinearLayout>
```

说明 在<fragment></fragment>标记中，android:name属性用于指定要添加的Fragment。

2. 当 Activity 运行时添加 Fragment

当 Activity 运行时，也可以将 Fragment 添加到 Activity 的布局中，实现方法是获取一个 FragmentTransaction 的实例，然后使用 add() 方法添加一个 Fragment，add() 方法的第一个参数是

Fragment 要放入的 ViewGroup（由 Resource ID 指定），第二个参数是需要添加的 Fragment，最后为了使改变生效，还必须调用 commit() 方法提交事务。例如，要在 Activity 运行时添加一个名称为 DetailFragment 的 Fragment，可以使用下面的代码：

```
01  DetailFragment details = new DetailFragment();     //实例化DetailFragment的对象
02  FragmentTransaction ft = getFragmentManager()
03          .beginTransaction();                        //获得一个FragmentTransaction的实例
04  ft.add(android.R.id.content, details);              //添加一个显示详细内容的Fragment
05  ft.commit();                                        //提交事务
```

Fragment 比较强大的功能之一就是可以合并两个 Activity，从而让这两个 Activity 在一个屏幕上显示。如图 9.26 所示（参照 Android 官方文档），左边的两个图分别代表两个 Activity，右边的图表示包括两个 Fragment 的 Activity，其中第一个 Fragment 的内容是 Activity A，第二个 Fragment 的内容是 Activity B。

图 9.26　使用 Fragment 合并两个 Activity

下面通过一个实例介绍 Fragment 在 App 中的实际应用。

例 3.5　模拟微信切换界面功能

在 Android Studio 中创建 Module，名称为“WeChat Interface Change”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后在布局管理器中添加一个 Fragment 组件，并为其设置 id 属性，在 Fragment 组件下方添加一个水平线性布局管理器，并设置其显示在容器底部，最后在水平线性布局管理器中添加 4 个布局宽度相同的 ImageView，将它们的 layout_weight 属性均设置为 1。具体代码如下：

```
01  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
02      xmlns:tools="http://schemas.android.com/tools"
03      android:layout_width="match_parent"
04      android:layout_height="match_parent"
05      tools:context="com.mingrisoft.MainActivity">
```

```
    <!--Fragment组件-->
    <fragment
        android:id="@+id/fragment"
        android:name="com.mingrisoft.WeChat_Fragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal">
        <!--微信图标-->
        <ImageView
            android:id="@+id/image1"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:src="@mipmap/bottom_1" />
        <!--通讯录图标-->
        <ImageView
            android:id="@+id/image2"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:src="@mipmap/bottom_2" />
        <!--发现图标-->
        <ImageView
            android:id="@+id/image3"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:src="@mipmap/bottom_3" />
        <!--我图标-->
        <ImageView
            android:id="@+id/image4"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:src="@mipmap/bottom_4" />
    </LinearLayout>
</RelativeLayout>
```

（2）在 res/layout 目录下创建一个名称为 wechat_fragment.xml 的布局文件，并且将默认创建的线性布局管理器修改为相对布局管理器，然后在该布局文件中添加一个 ImageView 组件，用于存放要显示的图片。

（3）在工具窗口中的 WeChat Interface Change\java 节点的第一个 com.mingrisoft 包中，创建一个名称为“WeChat_Fragment”的类，让这个类继承 Fragment 类，并且重写 onCreateView() 方法，然后为 WeChatFragment 添加 wechat_fragment.xml 布局文件，具体代码如下：

```
01  public class WeChat_Fragment extends Fragment {
02      @Override
03      public View onCreateView(LayoutInflater inflater,
04                               ViewGroup container, Bundle savedInstanceState) {
05          View view=inflater.inflate(R.layout.wechat_fragment,null);
06          return view;
07      }
08  }
```

说明 按照步骤（2）和步骤（3）的方法，再创建3个Fragment类和3个对应的布局文件，分别用于实现“通讯录”“发现”和“我”3个界面。

（4）打开默认创建的 MainActivity，让 MainActivity 直接继承 Activity，获取布局文件中的 4 张 Tab 标签图片，并且为每一张图片设置单击事件监听器，然后通过 switch 语句判断单击哪张导航图片，并创建相应的 Fragment 替换原有的 Fragment。具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //获取布局文件的第一张导航图片
07          ImageView imageView1 = (ImageView) findViewById(R.id.image1);
08          //获取布局文件的第二张导航图片
09          ImageView imageView2 = (ImageView) findViewById(R.id.image2);
10          //获取布局文件的第三张导航图片
11          ImageView imageView3 = (ImageView) findViewById(R.id.image3);
12          //获取布局文件的第四张导航图片
13          ImageView imageView4 = (ImageView) findViewById(R.id.image4);
14          imageView1.setOnClickListener(l);          //为第一张导航图片添加单击事件监听器
15          imageView2.setOnClickListener(l);          //为第二张导航图片添加单击事件监听器
16          imageView3.setOnClickListener(l);          //为第三张导航图片添加单击事件监听器
17          imageView4.setOnClickListener(l);          //为第四张导航图片添加单击事件监听器
18      }
19      //创建单击事件监听器
20      View.OnClickListener l = new View.OnClickListener() {
21          @Override
```

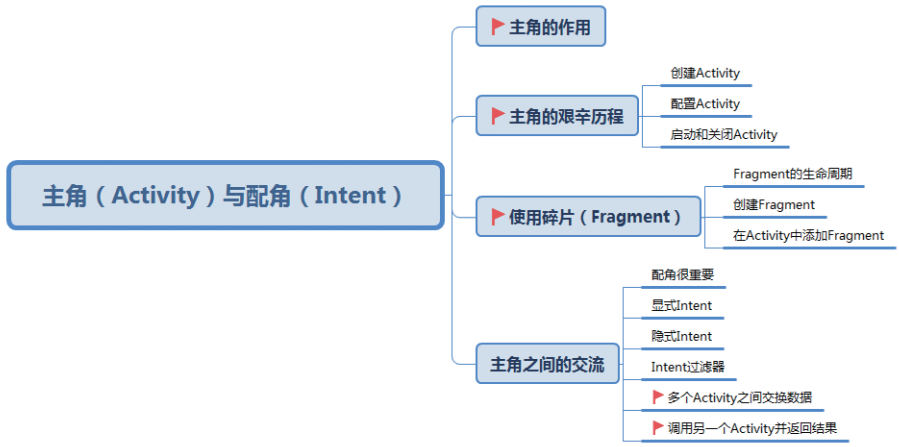
```
22          public void onClick(View v) {
23              FragmentManager fm = getFragmentManager();          //获取Fragment
24              FragmentTransaction ft = fm.beginTransaction();     //开启一个事务
25              Fragment f = null;                                  //为Fragment初始化
26              switch (v.getId()) {                    //通过获取单击的id判断单击了哪张图片
27                  case R.id.image1:
28                      f = new WeChat_Fragment();      //创建第一个Fragment
29                      break;
30                  case R.id.image2:
31                      f = new Message_Fragment();     //创建第二个Fragment
32                      break;
33                  case R.id.image3:
34                      f = new Find_Fragment();        //创建第三个Fragment
35                      break;
36                  case R.id.image4:
37                      f = new Me_Fragment();          //创建第四个Fragment
38                      break;
39                  default:
40                      break;
41              }
42              ft.replace(R.id.fragment, f);           //替换Fragment
43              ft.commit();                            //提交事务
44          }
45      };
46  }
```

（5）运行本实例，将显示如图 9.27 所示的效果。

图 9.27　模拟微信界面

9.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 10 章

Android 程序的修理工

开发 Android 程序时，不仅要注意程序代码的准确性与合理性，还要处理程序中可能出现的异常情况。Android SDK 中提供了 Log 类来获取程序的日志信息；另外，还提供了 LogCat 管理器，用来查看程序运行的日志信息及错误日志。本章将详细讲解如何对 Android 程序进行调试及 DDMS 工具的使用方法。

10.1 DDMS 工具的使用

DDMS（全称为 Dalvik Debug Monitor Service），是 Android 开发环境中的 Dalvik 虚拟机调试监控服务。在应用开发时，经常使用它来调试程序。可以进行的操作有：查看指定进程中正在运行的线程信息、内存信息、内存分配信息，为测试设备截屏、LogCat 日志信息、模拟电话呼叫和模拟发送短信等。

10.1.1 Android Studio 中打开 DDMS

打开 Android Studio，单击 Tools → Android → Android Device Monitor 选项，将打开 DDMS 工具。由于 DDMS 工具主要是用于监控虚拟机的，所以在打开 DDMS 之前，通常需要先启动一个 Dalvik 虚拟机（启动模拟器或者连接手机）。例如，启动模拟器后，看到的 DDMS 工具窗口如图 10.1 所示。

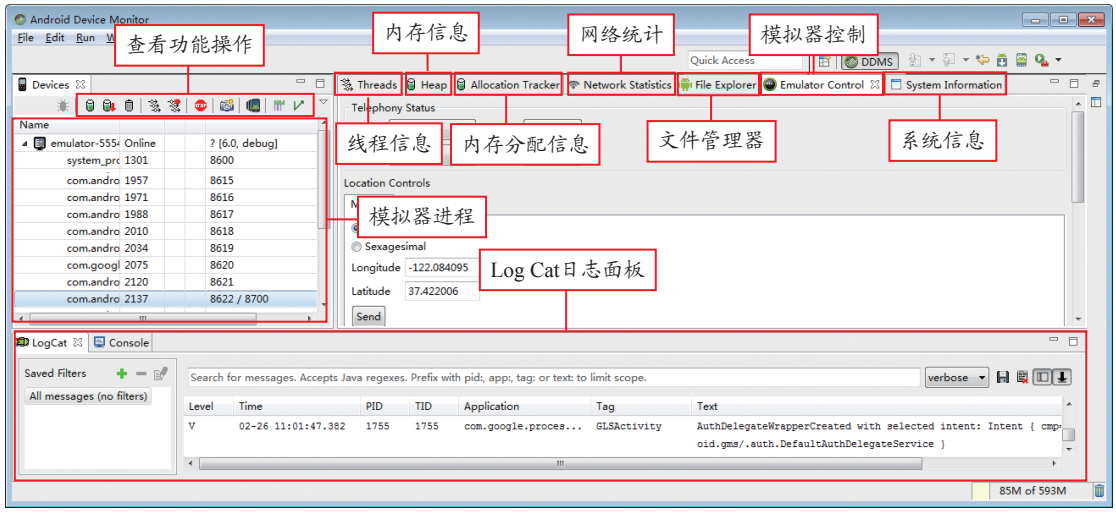


图 10.1　DDMS 工具窗口

说明 需要配置JDK的环境变量才可以使用DDMS功能。

10.1.2 DDMS 常用功能详解

1. 日志查看器（LogCat）

在应用开发调试过程中，经常需要查看日志信息，通过日志可以了解程序的执行情况，以及出现的异常信息。在 DDMS 的底部，单击 LogCat 选项卡，即可打开日志查看器。通过它可以查看和过滤日志信息的输出。

2. 线程查看器（Threads）

在应用开发调试过程中，有时会出现死锁或者信号量卡死的情况，对于这种情况，单看代码很难找出问题的原因。这时就可以借助 DDMS 中的线程查看器来查看线程的运行情况。通常情况下，可以通过线程的状态（Stauts）来确定出现问题的线程，一般情况下，当某个线程的 Stauts 变为 Monitor 状态时（即正在等待获取一个监听锁时），就需要特别留意它，并结合刷新查看它的执行情况，再结合代码找出问题的原因。

在 Android Studio 中，使用模拟器运行程序后，打开 DDMS 工具窗口。选择模拟器 App 进程区域中想要查询的一个 App 进程，然后单击上方查看功能中的 按钮更新线程信息。最后单击右侧线程信息面板，查看线程信息，如图 10.2 所示。

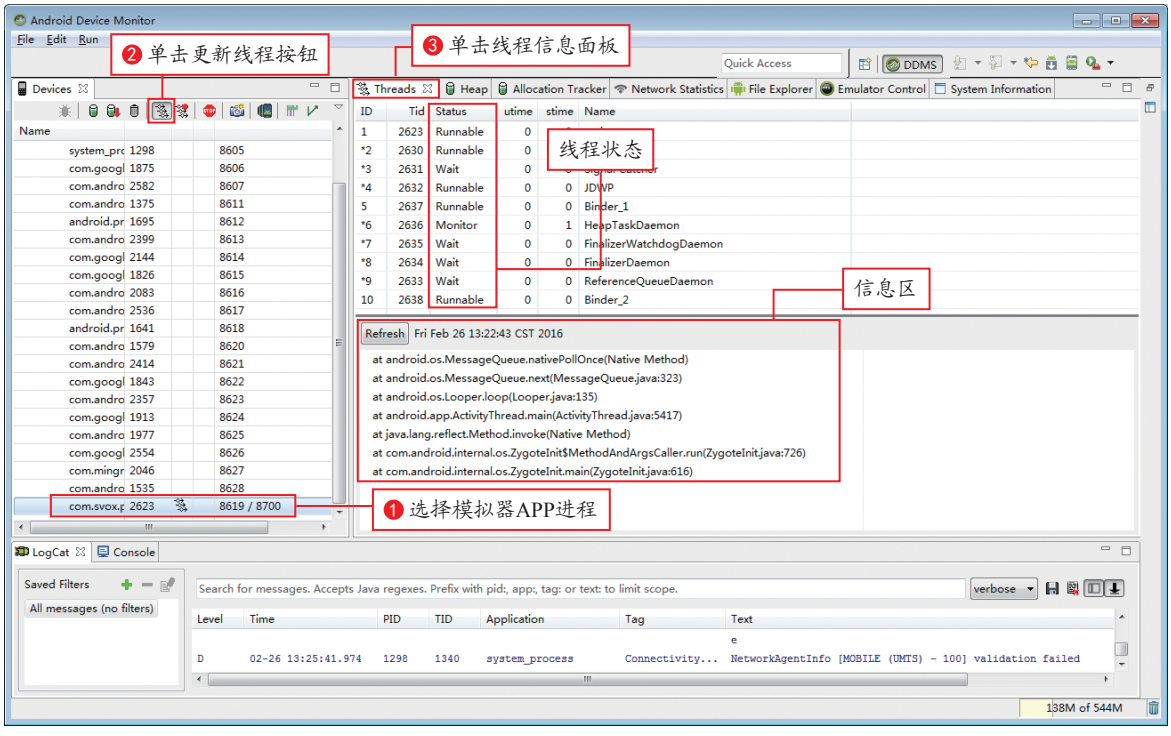


图 10.2 查看线程信息

说明 一个模拟器进程就是一个App应用，根据进程最前面的包名选择想要查询的App进程。

3．堆内存查看器（Heap）

在进行手机应用开发时，如有不慎，可能造成内存泄漏，从而导致系统运行变慢或应用程序崩溃。为此，DDMS 工具提供了一个内存监测工具，即 Heap。通过它可以检测一个正在运行的 App 应用的内存变化，从而检测出某个应用是否存在内存泄漏的可能。

在 Android Studio 工具中，使用模拟器运行程序后，打开 DDMS 工具窗口。选择模拟器 APP 进程区域中想要查询的一个 App 进程，然后单击上方查看功能中的按钮更新堆内存信息。最后单击右侧堆内存信息面板，单击 Cause GC 按钮更新堆内存信息，然后在下面的列表中选择显示的类型，查看堆内存信息，如图 10.3 所示。

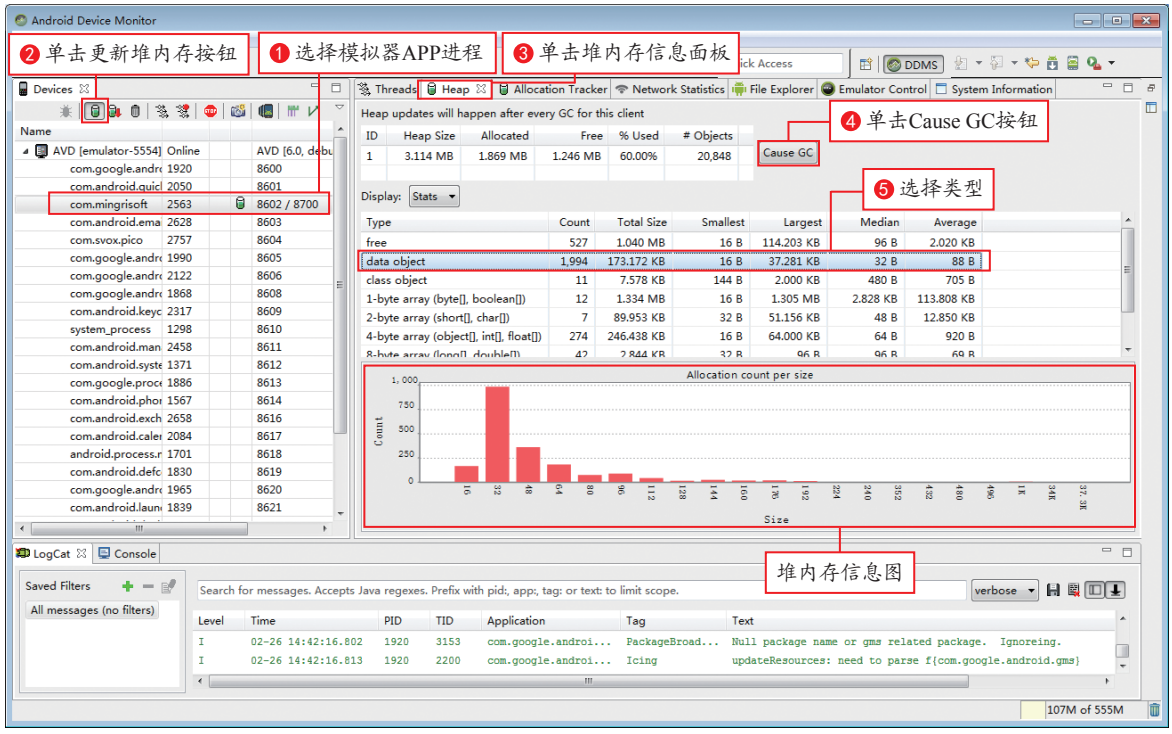


图 10.3　查看堆内存信息

说明　一般情况下，可以通过数据对象（data object）来判断内存是否泄漏。当我们进入某个App，并不断操作，然后仔细观察data object的Total Size列的值，当我们每次单击 Cause GC 按钮进行垃圾回收并更新堆状态后，Total Size的值不仅不会有明显的回落，反而越来越大，这时该应用就可能存在内存泄漏。

4．文件管理器（File Explorer）

File Explorer 用于管理 Android 模拟器中的文件，可以很方便地导入 / 导出文件。

在 Android Studio 工具中，使用模拟器运行程序后，打开 DDMS 工具窗口。单击右侧文件管理器面板，在下面的文件列表中选择一个需要导出的文件，单击右上角按钮，弹出选择指定位置对话框，单击确定按钮即可导出指定文件；选择导入文件存放的位置，单击右上角按钮，弹出选择指定文件对话框，单击确定按钮即可将指定文件导入到模拟器的指定目录中。操作步骤如图 10.4 所示。

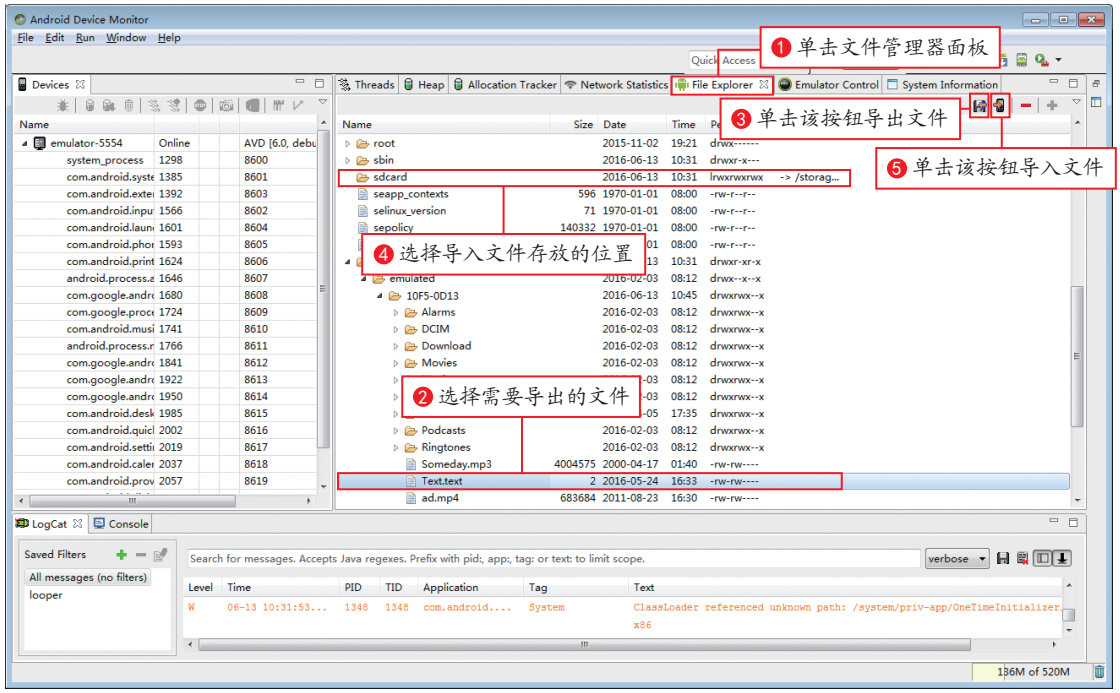


图 10.4　文件管理器

5. 模拟器控制器（Emulator Control）

Emulator Control 用于实现对模拟器的控制。例如，给模拟器打电话，根据选项模拟各种不同的网络，或者给模拟器发短信息等。

在 Android Studio 工具中，启动模拟器，然后打开 DDMS 工具窗口。在模拟器 APP 进程区域中选中模拟器，单击右侧上方模拟器控制面板，将显示控制面板，默认选中打电话单选按钮（Voice）。在 Incoming number 文本框中输入一个虚拟拨打给模拟器的号码，单击 Call 按钮，模拟器将显示如图 10.5 所示的效果，将单选按钮选中 SMS 选项，在 Message 文本框当中输入发送给模拟器的文本信息，单击 send 按钮发送短信给模拟器，模拟器将显示如图 10.6 所示的效果。操作步骤如图 10.7 所示。

说明　向模拟器发短信息时，文本框内不可以填写中文，模拟器无法识别。

图 10.5　给模拟器打电话

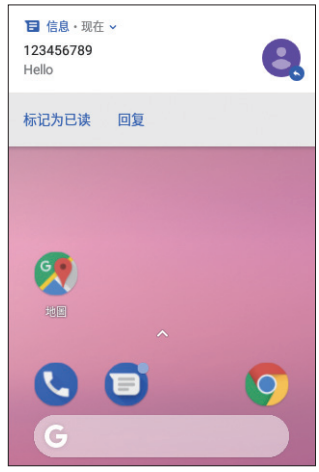


图 10.6　给模拟器发短信

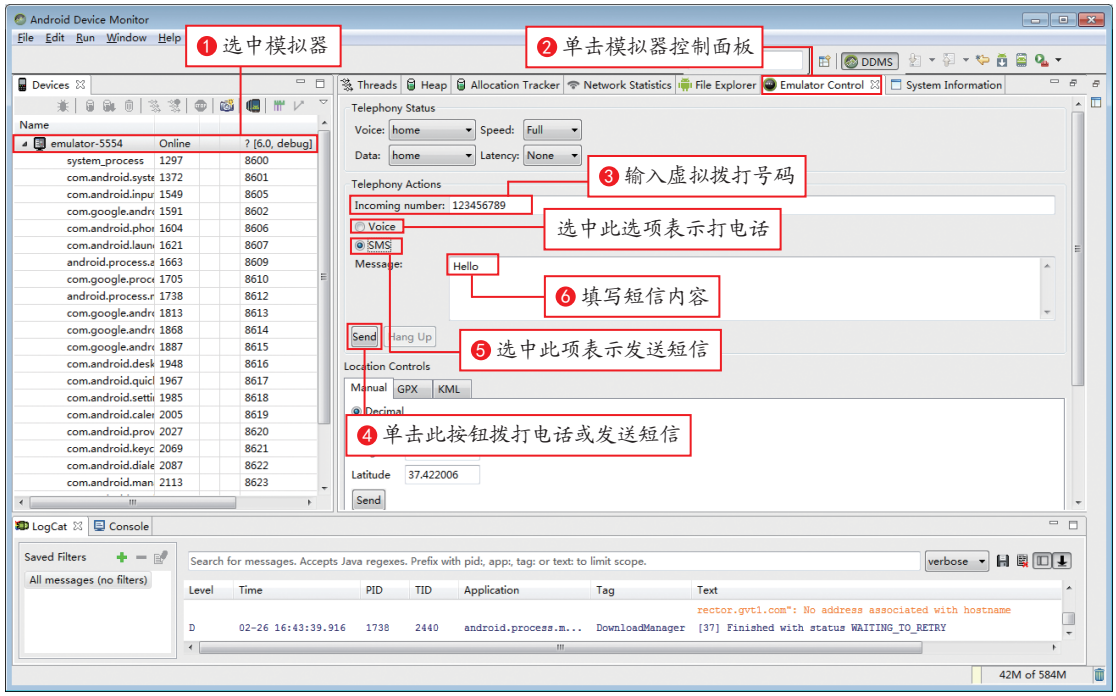


图 10.7　模拟器控制的操作步骤

6. 屏幕截取（Screen Capture）

Screen Capture 用于实现对 Dalvik 虚拟机运行界面的屏幕截取。例如，在 Android Studio 工具中，启动模拟器，然后打开 DDMS 工具窗口。在模拟器 App 进程区域中选中模拟器，单击查看功能操作区域的按钮，将弹出如图 10.8 所示的截图窗口，单击 Save 按钮将图片保存到指定位置。

图 10.8　屏幕截取

10.2 输出日志信息

Android SDK 中提供了 Log 类来获取程序运行时的日志信息，该类位于 android.util 包中，它继

承自 java.lang.Object 类。Log 类提供了一些方法，用来输出日志信息，其常用方法及说明如表 10.1 所示。

表 10.1　Log 类的常用方法及说明

方　法	说　明
e()	输出 ERROR 错误日志信息
w()	输出 WARN 警告日志信息
i()	输出 INFO 程序日志信息
d()	输出 DEBUG 调试日志信息
v()	输出 VERBOSE 冗余日志信息

表 10.1 中列出的 Log 类的相关方法都有多种重载形式，下面将介绍经常用到的重载形式。

10.2.1 Log.e() 方法

Log.e() 方法用来输出 ERROR 错误日志信息。开发人员经常用到的重载形式语法如下：

```
public static int e (String tag, String msg)
```

☑ tag：String 字符串，用来为日志信息指定标签，它通常指定为可能出现错误的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

说明 ERROR错误日志在DDMS的LogCat面板中，使用红颜色的文字表示。

10.2.2 Log.w() 方法

Log.w() 方法用来输出 WARN 警告日志信息。开发人员经常用到的重载形式语法如下：

```
public static int w (String tag, String msg)
```

☑ tag：String 字符串，用来为日志信息指定标签，它通常指定为可能出现警告的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

说明 WARN警告日志在DDMS的LogCat面板中，使用橘黄色的文字表示。

10.2.3 Log.i() 方法

Log.i() 方法用来输出 INFO 程序日志信息。开发人员经常用到的重载形式语法如下：

```
public static int i (String tag, String msg)
```

☑ tag：String 字符串，用来为日志信息指定标签，它通常指定为类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

说明 INFO程序日志在DDMS的LogCat面板中，使用绿色的文字表示。

10.2.4 Log.d() 方法

Log.d() 方法用来输出 DEBUG 调试日志信息。开发人员经常用到的重载形式语法如下：

```
public static int d (String tag, String msg)
```

☑ tag：String 字符串，用来为日志信息指定标签，它通常指定为可能出现 Debug 的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

说明 DEBUG调试日志在DDMS的LogCat面板中，使用蓝色的文字表示。

10.2.5 Log.v() 方法

Log.v() 方法用来输出 VERBOSE 冗余日志信息。开发人员经常用到的重载形式语法如下：

```
public static int v (String tag, String msg)
```

☑ tag：String 字符串，用来为日志信息指定标签，它通常指定为可能出现冗余的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

说明 VERBOSE冗余日志在DDMS的LogCat面板中，使用黑色的文字表示。

下面通过一个具体的实例演示 Log 类常用方法的使用。

例 10.1 输出警告日志信息

（1）打开 MainActivity，让 MainActivity 直接继承 Activity。创建一个 String 类型 TAG，为其指定名称 MainActivity。在重写的 onCreate() 方法中，使用日志 API 方法输出错误、警告、普通、调试和冗余日志信息，代码如下：

```
01  public class MainActivity extends AppCompatActivity {
02      private static String TAG = "MainActivity";
03      @Override
04      protected void onCreate(Bundle savedInstanceState) {
05          super.onCreate(savedInstanceState);
06           setContentView(R.layout.activity_main);
07           Log.e(TAG, "错误信息");
08           Log.w(TAG, "警告信息");
09           Log.i(TAG, "普通信息");
10           Log.d(TAG, "调试信息");
11           Log.v(TAG, "冗余信息");
12      }
13  }
```

（2）单击按钮打开 DDMS 工具。单击 LogCat 日志面板中的按钮，填写过滤器名称（Filter Name）和日志标记名称（by Log Tag），名称均为 MainActivity。通过 LogCat 过滤器准确查找日志信息。如图 10.9 所示。

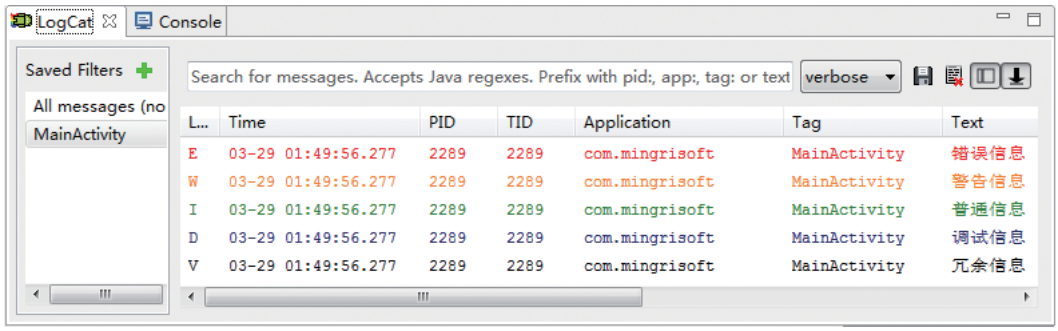


图 10.9　使用日志 API 方法输出日志信息

10.3 程序调试

在程序开发过程中会不断体会到程序调试的重要性。为验证 Android 的运行状况，会经常在某个方法调用的开始和结束位置分别使用 Log.i() 方法输出信息，并根据这些信息判断程序执行状况，这是非常古老的程序调试方法，而且经常导致程序代码混乱。下面将介绍几种使用 Android 常用的调试工具来调试 Android 应用程序的方法。

10.3.1 Android Studio 编辑器调试

在使用 Android Studio 开发 Android 应用时，使用的编辑器不但能够为开发者提供代码编写、辅助提示和实时编译等常用功能，而且还能够对 Java 源代码进行快捷修改、重构和语法纠错等高级操作。通过 Android Studio 编辑器，可以很方便地找到一些语法错误，并根据提示进行快速修正。下面对 Android Studio 编辑器提供的调试功能进行介绍。

☑ 代码下方的红色波浪线

在出现错误的代码下方，会显示红色的波浪线，将鼠标移动到红色波浪线上，将显示如图 10.10 所示提示报错的原因，同时此行代码左侧将显示图标，单击图标的三角号将显示可用的快速修正，如图 10.11 所示的提示框，单击具体的方法，可进行快速修正错误，也可以通过快捷键〈Alt+Enter〉显示快速修正错误提示框。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ListView button=findViewById(R.id.listview);
}
```

图 10.10　提示报错的原因

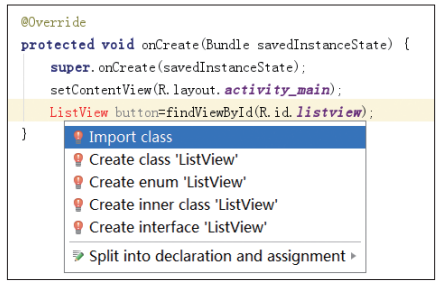


图 10.11　提示快速修正方法

☑ 编辑器窗口的右上角的 标记

它的作用是提示开发者该文档的某个位置存在错误，将鼠标移动到该图标上，将显示具体的错误个数，如图 10.12 所示。

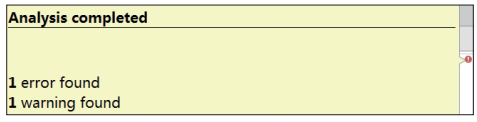


图 10.12　显示具体的错误个数

☑ 编辑器窗口右侧的 标记

在编辑器右侧空白位置可以看到一个或者多个红色的矩形框，它指明了错误所在行的大致位置，如果代码量非常大，可以单击这个红色矩形块，快速定位错误所在行以及错误信息等。

10.3.2 Android Studio 调试器调试

本节将介绍使用 Android Studio 内置的 Android 调试器调试 Android 程序的方法，使用该调试器可以设置程序的断点、实现程序单步执行、在调试过程中查看变量和表达式的值等调试操作，这样可以避免在程序中编写大量的 Log.i () 方法输出调试信息。

使用 Android Studio 的 Android 调试器需要设置程序断点，然后使用单步调试分别执行程序代码的每一行。

1．断点

设置断点是程序调试中必不可少的有效手段，Android 调试器每次遇到程序断点时都会将当前线程挂起，即暂停当前程序的运行。在 Android Studio 中，可以在 Android 编辑器中，单击显示代码行号的位置即可添加或删除当前行的断点，如图 10.13 所示。

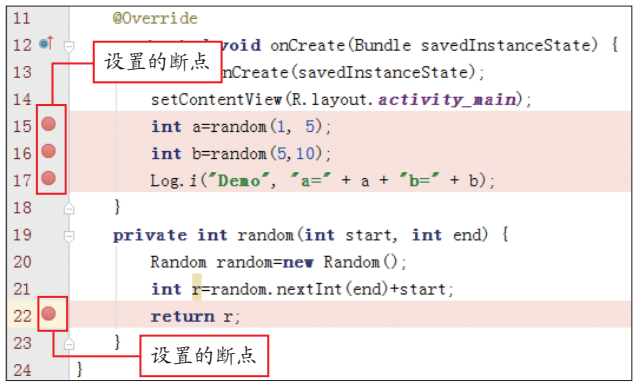


图 10.13　选择“切换断点”命令

技巧 在Android Studio的编辑中，默认不显示行号，在代码左侧的灰色区域中单击鼠标右键，在弹出的快捷菜单中，选择Show Line Numbers菜单项，即可显示行号。

2．简单调试

为程序设置断点后，单击工具栏上的按钮，将在 Android Studio 的底部显示调试面板，如图 10.14 所示。

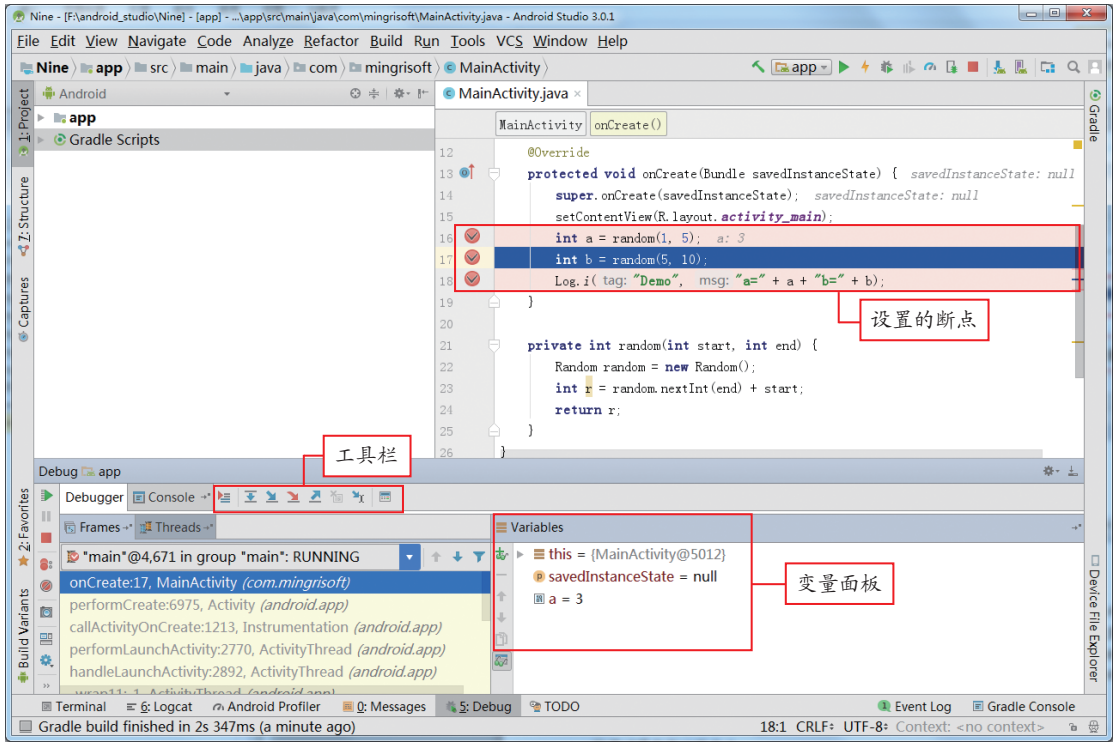


图 10.14　调试面板

说明 在调试面板中，Variables面板用于显示程序运行时涉及的变量的值，并且在某变量上单击鼠标右键，在弹出的快捷菜单中选择“Set Values...”菜单项，可以为该变量设置新的值。如果程序中变量过多，在该面板中不方便查看，可以在右侧的Watches面板中，单击+按钮，输入要查看的变量，并按下〈Enter〉键，将在下方显示该变量的值。如果不想显示某个变量的值，也可以选中该变量，然后单击—按钮，即可从列表中删除该变量的查看。

在调试面板中，可以通过工具栏上的按钮执行相应的调试操作，如单步跳过、单步跳入等。常用的调试操作有以下几个。

☑ 单步跳过

在工具栏中单击 按钮或按 F8 键，将执行单步跳过操作，即运行单独的一行程序代码，但是不进入调用方法的内部，然后跳到下一个可执行点。

说明 不停地执行单步跳过操作，会每次执行一行程序代码，直到程序结束或等待用户操作。

☑ 单步跳入

在工具栏中单击 按钮或按 F7 键，将跳入到调用方法或对象的内部单步执行程序。

☑ 强制单步跳入

在工具栏中单击按钮，会跳入所有被调用的方法。

☑ 单步跳出

在工具栏中单击按钮，如果在调试的时候跳入了一个方法，并觉得该方法没有问题，此时就可以执行该操作跳出该方法，返回到该方法被调用处的下一行代码。

☑ 跳到下一断点

在工具栏中单击按钮，会继续向下执行，直到下一个断点的位置。如果程序中没有断点或者异常，将直接运行到程序结束。

3．高级调试

☑ 跨断点调试

在工具栏中单击按钮，如果在程序中设置了多个断点，执行该操作会直接执行到下一个断点。

☑ 查看断点

在工具栏中单击按钮，可以看到所有的断点，也可以设置一些属性。

☑ 停止调试

在工具栏中单击按钮，会停止程序的调试。需要注意的是，该操作不会停止程序的运行，而是会跳过所有调试。

☑ 实时计算变量的值

在工具栏中单击按钮，可以打开如图 10.15 所示的 Evaluate Expression 对话框，在该对话框中，可以实时计算程序中变量的值。例如，在图 10.15 中，在 Expression 列表框中输入变量名 a，并按下 Evaluate 按钮，将显示变量 a 的值为 3，就是在执行了代码“int a=random(1,5);”以后的结果。

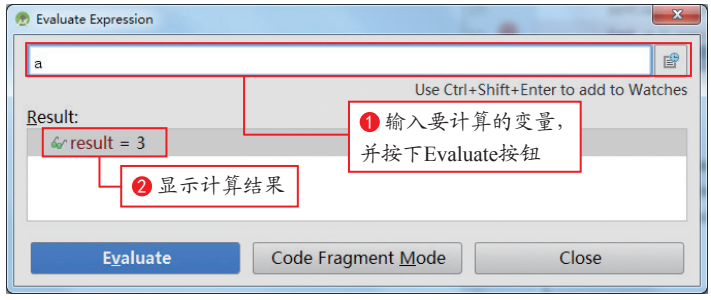
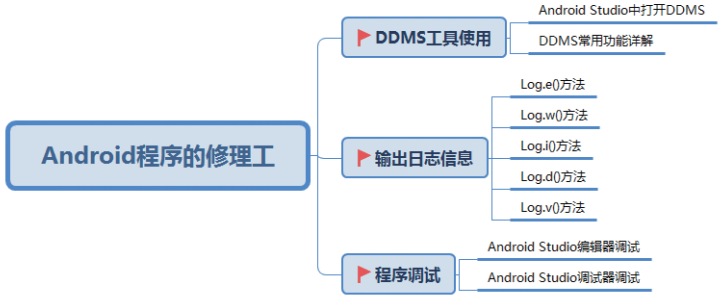


图 10.15　Evaluate Expression 对话框

说明 在Evaluate Expression对话框中，也可以计算表达式的值。例如，在图10.15的Expression列表框中输入“a+b”，并按下〈Enter〉键，将显示表达式“a+b”的值。

10.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 11 章

一起来互动（事件与手势）

用户在使用手机、平板电脑时，是通过各种操作来与软件进行交互，比较常见的方式包括物理按键操作、触摸屏操作和手势等。在 Android 中，这些操作都将转换为对应的事件进行处理，本章对 Android 中事件处理及手势检测进行介绍。

11.1 互动规则

现在的图形界面应用程序，都是通过事件来实现人机交互的。事件就是用户对图形界面的操作。在 Android 手机和平板电脑上，主要包括物理按键事件和触摸屏事件两大类。物理按键事件包括按下、抬起和长按等；触摸屏事件包括单击事件、长按事件和触摸事件等。

在 Android 组件中，提供了事件处理的相关方法。例如，在 View 类中，提供了 onTouchEvent() 方法，我们可以重写该方法来处理触摸屏事件。这种方式主要适用于重写组件的场景。但是仅仅通过重写这个方法来完成事件处理是不够的。为此，Android 提供了使用 setOnTouchListener() 方法为组件设置监听器来处理触摸屏事件，这在日常开发中更加常用。

在 Android 中提供了两种方式的事件处理，一种是基于监听的事件处理，另一种是基于回调的事件处理，下面分别进行介绍。

11.1.1 基于监听的事件处理

实现基于监听的事件处理，主要做法就是为 Android 的 UI 组件绑定特定的事件监听器。在事件监听的处理模型中，主要有以下 3 类对象。

◆ Event Source（事件源）：即产生事件的来源，通常是各种组件，例如，按钮、窗口和菜单等。

◆ Event（事件）：事件中封装了 UI 组件上发生的特定事件的具体信息，如果监听器需要获取 UI 组件上所发生事件的相关信息，一般通过 Event 对象来传递。

◆ Event Listener（事件监听器）：监听事件源所发生的事件，并对不同的事件做出相应的响应。

事件处理流程示意图如图 11.1 所示。

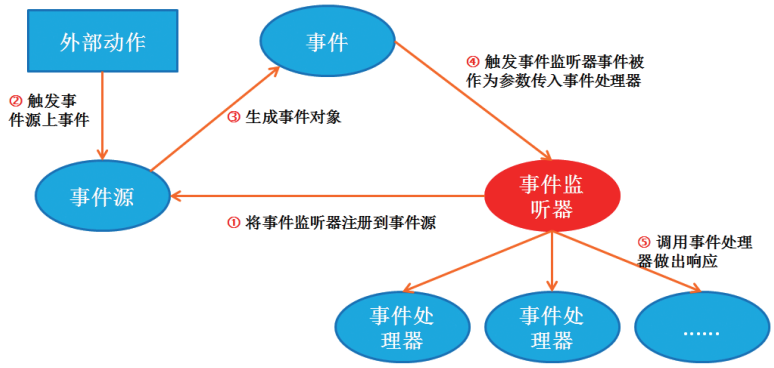


图 11.1　事件处理流程示意图

11.1.2 基于回调的事件处理

实现基于回调的事件处理，主要做法就是重写 Android 组件特定的回调方法，或者重写 Activity 的回调方法。从代码实现的角度来看，基于回调的事件处理模型更加简单。为了使用回调机制来处理 GUI 组件上所发生的事件，需要为该组件提供对应的事件处理方法，可以通过继承 GUI 组件类，并重写该类的事件处理方法来实现。

为了实现回调机制的事件处理，Android 为所有 GUI 组件都提供了一些事件处理的回调方法，例如，在 View 类中就包含了一些事件处理的回调方法，这些方法如表 11.1 所示。

表 11.1　View 类中事件处理的回调方法

方　法	说　明
boolean onKeyDown(int keyCode, KeyEvent event)	当用户在该组件上按下某个按键时触发
boolean onKeyLongPress(int keyCode, KeyEvent event)	当用户在该组件上长按某个按键时触发
boolean onKeyShortcut(int keyCode, KeyEvent event)	当一个键盘快捷键事件发生时触发
boolean onKeyUp(int keyCode, KeyEvent event)	当用户在该组件上松开某个按键时触发
boolean onTouchEvent (MotionEvent event)	当用户在该组件上触发触摸屏事件时触发
boolean onTrackballEvent(MotionEvent event)	当用户在该组件上触发轨迹球事件时触发

一般来说，基于回调的事件处理方式可用于处理一些通用性的事件，事件处理的代码会比较简洁。但对于某些特定的事件，无法采用基于回调的事件处理方式实现时，就只能采用基于监听的事件处理方式了。

11.2 物理按键互动

对于一个标准的 Android 设备，包含了多个能够触发事件的物理按键。例如，手机上的常用物理按键，如图 11.2 所示。

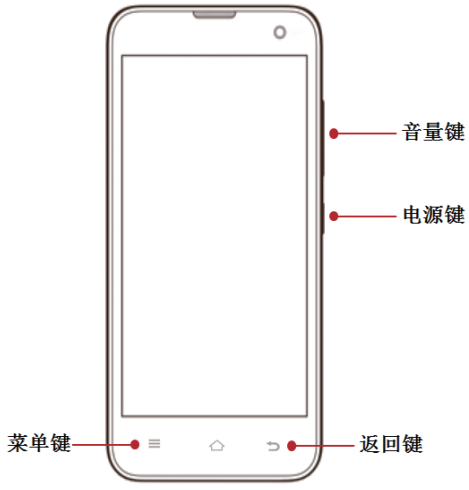


图 11.2　Android 手机常用物理按键

Android 设备常用物理按键能够触发的事件及其说明如表 11.2 所示。

表 11.2　Android 设备常用物理按键及其触发事件

物 理 按 键	KeyEvent	说　　明
音量键	KEYCODE_VOLUME_UP KEYCODE_VOLUME_DOWN	控制当前上下文音量，如音乐播放器、手机铃声、通话音量等
返回键	KEYCODE_BACK	返回到前一个界面
菜单键	KEYCODE_MENU	显示当前应用的可用菜单

在 Android 中处理物理按键事件时，常用的回调方法有以下 3 个。

◆ onKeyUp()：当用户松开某个按键时触发该方法。

◆ onKeyDown()：当用户按下（未松开）某个按键时触发该方法。

◆ onKeyLongPress()：当用户长按某个按键时触发该方法。

下面通过一个实例演示返回按钮的触发事件。

例 11.1　模拟退出手机淘宝的消息提示

在 Android Studio 中创建一个 Module，名称为“Quit TaoBao”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后将默认添加的 TextView 组件删除，为布局管理器添加背景图片。

（2）修改默认创建的 MainActivity，让 MainActivity 直接继承 Activity，并重写 onKeyDown() 方法来拦截用户单击返回键的事件。关键代码如下：

```
01  public class MainActivity extends Activity {
02      private long exitTime = 0;              //退出时间变量值
03      @Override
04      protected void onCreate(Bundle savedInstanceState) {
05          super.onCreate(savedInstanceState);
06          setContentView(R.layout.activity_main);
```

```
        //全屏
        getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN ,
                WindowManager.LayoutParams. FLAG_FULLSCREEN);
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        //判断是否单击了返回键
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            exit();                                    //创建并调用退出方法
            return true;                               //拦截返回键
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

说明 由于此步骤中没有创建退出方法exit()，所以代码显示为红色。创建退出方法后代码将显示为黑色。

（3）在 MainActivity 中，创建退出方法 exit()，在该方法中判断按键时间差是否大于 2 秒，如果大于 2 秒，则弹出消息提示，否则退出当前应用，具体代码如下：

```
public void exit() {
    if ((System.currentTimeMillis() - exitTime) > 2000) { //判断按键时间差是否大于2秒
        Toast.makeText(getApplicationContext(),
                "再按一次返回键退出手机淘宝", Toast.LENGTH_SHORT).show();
        exitTime = System.currentTimeMillis();
    } else {
        finish();
        System.exit(0);                                    //销毁，并强制退出
    }
}
```

（4）运行本实例，将显示如图 11.3 所示的效果。

图 11.3　退出手机淘宝的消息提示

11.3 触摸屏互动

当下，主流的 Android 手机 / 平板电脑都以较大的屏幕取代了外置键盘，很多操作都是通过触摸屏幕来实现的。其中，常用的触摸屏事件主要包括单击事件、长按事件和触摸事件等。下面分别进行介绍。

11.3.1 单击事件

在手机应用中，经常需要实现在屏幕中单击某个按钮或组件执行一些操作。这时就可以通过单击事件来完成。在处理单击事件时，可以通过为组件添加单击事件监听器的方法来实现。Android 为组件提供了 setOnClickListener() 方法，用于为组件设置单击事件监听器。该方法的参数是一个 View.OnClickListener 接口的实现类对象。View.OnClickListener 接口的定义如下：

```
public static interface View.OnClickListener{
public void onClick(View v);
}
```

从上面接口的定义中可以看出，在实现 View.OnClickListener 接口时，需要重写 onClick() 方法。当单击事件触发后，将调用 onClick() 方法执行具体的事件处理操作。

例如，要为名称为 button1 的按钮添加一个单击事件监听器，并且实现在单击该按钮时弹出消息提示框显示“单击了按钮”，可以通过下面的代码实现。

```
01  Button button1=new Button(this);
02  button1.setOnClickListener(new View.OnClickListener() {
03      @Override
04      public void onClick(View v) {
05          Toast.makeText(MainActivity.this, "单击了按钮", Toast.LENGTH_SHORT).show();
06      }
07  });
```

11.3.2 长按事件

在 Android 中还提供了长按事件的处理操作，长按事件与单击事件不同，该事件需要长按某一个组件 2 秒之后才会触发。在处理长按事件时，可以通过为组件添加长按事件监听器的方法来实现。Android 为组件提供了 setOnLongClickListener () 方法，用于为组件设置长按事件监听器。该方法的参数是一个 View.OnLongClickListener 接口的实现类对象。View.OnLongClickListener 接口的定义如下：

```
public static interface View.OnLongClickListener{
public boolean onLongClick(View v);
}
```

从上面接口的定义中可以看出，在实现 View.OnLongClickListener 接口时，需要重写 onLongClick() 方法。当长按事件触发后，将调用 onLongClick() 方法执行具体的事件处理操作。

下面通过一个实例演示长按事件的使用。

例 11.2 模拟长按朋友圈图片弹出菜单功能

在 Android Studio 中创建 Module，名称为“Long Press Event”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认生成的布局管理器修改为垂直线性布局管理器，删除默认添加的 TextView 组件，然后添加 3 个 ImageView 组件用于放置图片。

（2）修改主活动 MainActivity，让其直接继承 Activity。使用 findViewById() 方法获得布局文件中定义的长按图片，并为其添加 OnLongClickListener 事件监听器，然后再重写 onCreateContextMenu() 方法，为菜单添加选项值。最后将长按事件注册到菜单中。代码如下：

```
01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         ImageView imageView = (ImageView) findViewById(R.id.imageView);//获取图片组件
07         //创建长按监听事件
08         imageView.setOnLongClickListener(new View.OnLongClickListener() {
09             @Override
10             public boolean onLongClick(View v) {
11                 registerForContextMenu(v);                    //将长按事件注册菜单中
12                 openContextMenu(v);                           //打开菜单
13                 return true;
14             }
15         });
16     }
17     @Override
18     public void onCreateContextMenu(ContextMenu menu, View v,
19                                     ContextMenu.ContextMenuInfo menuInfo) {//创建菜单
20         super.onCreateContextMenu(menu, v, menuInfo);
21         menu.add("收藏");                                      //为菜单添加参数
22         menu.add("举报");
23     }
24 }
```

（3）打开 AndroidManifest.xml 文件，修改 <application> 标记的 android:theme 属性，修改后的代码如下：

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

（4）运行本实例，效果如图 11.4 所示。长按朋友圈图片将自动弹出菜单，效果如图 11.5 所示。

图 11.4　长按朋友圈图片

图 11.5　显示菜单

11.3.3 触摸事件

触摸事件就是指当用户触摸屏幕之后产生的一种事件，当用户在屏幕上划过时，可以通过触摸事件获取用户当前的坐标。在处理触摸事件时，可以通过为组件添加触摸事件监听器的方法来实现。Android 为组件提供了 setOnTouchListener() 方法，用于为组件设置触摸事件监听器。该方法的参数是一个 View.OnTouchListener 接口的实现类对象。View.OnTouchListener 接口的定义如下：

```
public interface View.OnTouchListener{
public abstract boolean onTouch(View v, MotionEvent event);
}
```

从上面接口的定义中可以看出，在实现 View.OnTouchListener 接口时，需要重写 onTouch() 方法。触摸事件触发后，将调用 onTouch() 方法执行具体的事件处理操作，同时会产生一个 MotionEvent 事件类的对象，通过该对象可以获取用户当前的 X 坐标和 Y 坐标。

下面通过一个实例演示触摸事件的使用。

例 11.3　触摸屏幕帮助企鹅戴好帽子

在 Android Studio 中创建 Module，名称为“Touch Events”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ayout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并设置其背景和 id 属性，然后将 TextView 组件删除。

（2）在 com.mingrisoft 包中新建一个名称为 HatView 的 Java 类，该类继承自 View 类，重写带一个参数 Context 的构造方法和 onDraw() 方法。然后在构造方法中设置帽子的默认显示位置，最后在 onDraw() 方法中根据图片绘制帽子。HatView 类的关键代码如下：

```
01  public class HatView extends View {
02      public float bitmapX;                    //帽子显示位置的X坐标
03      public float bitmapY;                    //帽子显示位置的Y坐标
04      public HatView(Context context) {        //重写构造方法
05          super(context);
```

```
        bitmapX = 65;                          //设置帽子的默认显示位置的X坐标
        bitmapY = 0;                           //设置帽子的默认显示位置的Y坐标
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();             //创建Paint对象
        //根据图片生成位图对象
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),
 R.mipmap.hat);
        canvas.drawBitmap(bitmap, bitmapX, bitmapY, paint);  //绘制帽子
        if (bitmap.isRecycled()) {             //判断图片是否回收
            bitmap.recycle();                  //强制回收图片
        }
    }
}
```

（3）修改主活动 MainActivity，让 MainActivity 直接继承 Activity。在 onCreate() 方法中，首先获取相对布局管理器，并实例化帽子对象 hat，然后为 hat 添加触摸事件监听器，再在重写的触摸事件中，设置 hat 的显示位置，并重绘 hat 组件，最后将 hat 添加到布局管理器中，关键代码如下：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取相对局管理器
        RelativeLayout relativeLayout = (RelativeLayout) findViewById(R.id.relativeLayout);
        final HatView hat = new HatView(MainActivity.this);  //创建并实例化HatView类
        //为帽子添加触摸事件监听器
        hat.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                hat.bitmapX = event.getX()-80;    //设置帽子显示位置的X坐标
                hat.bitmapY = event.getY()-50;    //设置帽子显示位置的Y坐标
                hat.invalidate();                 //重绘hat组件
                return true;
            }
        });
        relativeLayout.addView(hat);              //将hat添加到布局管理器中
    }
}
```

（4）运行本实例，将显示如图 11.6 所示的界面。然后通过用户触摸屏幕帮助企鹅戴好帽子，如图 11.7 所示。

图 11.6　企鹅未戴好帽子的效果

图 11.7　企鹅已戴好帽子的效果

11.3.4 单击事件与触摸事件的区别

针对屏幕上的一个 View 组件，Android 如何区分应当触发 onTouch 事件，还是 onClick 事件呢。在 Android 中，一次用户操作可以被不同的 View 按次序分别处理，并将完全响应了用户的一次 UI 操作称之为消耗了该事件（consume），那么 Android 是按什么次序将事件传递的，又在什么情况下判定为消耗了该事件？下面通过一个具体的实例进行说明。

例 11.4　单击事件与触摸事件的区别

在 Android Studio 中创建 Module，名称为“Click And Touch”。实现本实例的具体步骤如下：

（1）修改新建项目的 res/layout 节点下的布局文件 activity_main.xml，将默认添加的 TextView 组件删除，并且添加一个 Button 按钮，同时为其设置 id 属性。

（2）修改 MainActivity 类，让 MainActivity 直接继承 Activity。在 onCreate() 方法中，使用 findViewById() 方法获得布局文件中定义的按钮，并且为按钮添加单击事件监听器，通过 Log.i() 方法输出 onClick 单击事件。然后为按钮添加触摸事件，通过判断方式输出手指是按下还是抬起。具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          Button button= (Button) findViewById(R.id.btn);
07          //为按钮添加单击事件监听器
08          button.setOnClickListener(new View.OnClickListener() {
09              @Override
10              public void onClick(View v) {
11                  Log.i("onClick","单击事件");
12              }
13          });
```

```
        //为按钮添加触摸事件监听器
        button.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction()==MotionEvent.ACTION_DOWN){    //表示手指按下时
                    Log.i("onTouch","按下");
                }else if (event.getAction()==MotionEvent.ACTION_UP){//表示手指抬起时
                    Log.i("onTouch","抬起");
                }
                return false;          //表示未消耗掉这个事件
            }
        });
    }
}
```

（3）运行本实例，单击按钮将会在 LogCat 视图中看到如图 11.8 所示的结果。

```
02-11 16:09:13.539 7321-7321/com.mingrisoft.clickandtouch I/onTouch: 按下
02-11 16:09:13.669 7321-7321/com.mingrisoft.clickandtouch I/onTouch: 抬起
02-11 16:09:13.669 7321-7321/com.mingrisoft.clickandtouch V/AudioManager: playSoundEffect   effectType: 0
02-11 16:09:13.669 7321-7321/com.mingrisoft.clickandtouch V/AudioManager: querySoundEffectsEnabled...
02-11 16:09:13.669 7321-7321/com.mingrisoft.clickandtouch I/onClick: 单击事件
```

图 11.8　显示执行顺序

说明　当触摸事件监听器返回值为真时，说明消耗掉了这个事件，将不再执行单击事件。

11.3.5 事件的综合应用

在实际开发中，事件的综合应用非常广泛，下面通过一个具体的实例进行说明。

例 11.5　微信朋友圈发送消息

在 Android Studio 中创建 Module，名称为“Click And Long”。实现本实例的具体步骤如下：

（1）修改新建项目的 res/layout 节点下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并设置背景图片，然后默认添加的 TextView 组件删除，再添加一个显示在右上角的 Button 按钮，同时为其设置 id 属性。

（2）修改 MainActivity 类，让 MainActivity 直接继承 Activity。然后分别实现 View.OnClickListener（单击事件）与 View.OnLongClickListener（长按事件）接口与对应的方法，然后在 onCreate() 方法中获取右上角所显示的按钮并为该按钮设置单击事件与长按事件，关键代码如下：

```
//全屏
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
Button button = findViewById(R.id.btn);       //获取右上角按钮
button.setOnClickListener(this);              //设置单击事件
button.setOnLongClickListener(this);          //设置长按事件
```

（3）在 res/layout 节点下创建名称为 dialog_layout.xml 的布局文件，该文件是对话框布局文件，用于显示单击右上角按钮后的菜单对话框。

（4）在 MainActivity 类中重写 onClick() 方法，该方法为右上角按钮的单击事件。在该方法中首先创建对话框实例，然后获取对话框窗口并设置对话框显示位置，再设置菜单对话框所显示的布局文件，最后获取菜单对话框中的选项并调用初始化选项事件的方法，代码如下：

```
@Override
public void onClick(View view) {
    //创建对话框实例
    dlg = new AlertDialog.Builder(this).create();
    dlg.show();                                  //显示对话框
    Window window = dlg.getWindow();             //获取对话框窗口
    window.setGravity(Gravity.CENTER);           //此处可以设置dialog显示的中心位置
    window.setContentView(R.layout.dialog_layout); //设置对话框布局文件
    //菜单对话框中的选项
    item1 = window.findViewById(R.id.layout1);
    item2 = window.findViewById(R.id.layout2);
    initEvent();                                 //调用初始化事件方法
}
```

（5）创建 initEvent() 方法，在该方法中实现菜单对话框两个选项的单击事件。代码如下：

```
//选项事件处理
private void initEvent() {
    item1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(MainActivity.this,
                    "您单击了拍摄！", Toast.LENGTH_SHORT).show();
        }
    });
    item2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(MainActivity.this,
                    "您单击了从相册选择！", Toast.LENGTH_SHORT).show();
        }
    });
}
```

（6）在该项目中创建一个新的 Activity 名称为“Main2Activity”，该 Activity 用于模拟发表这一刻想法的界面，在长按朋友圈右上角按钮时跳转该界面。首先在 Main2Activity 类中设置全屏代码即可，然后在 activity_main2.xml 文件中将布局管理求修改为相对布局并设置背景图片，最后添加 EditText（编辑框组件）用于填写文字，布局代码如下：

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

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg2"
    tools:context="com.mingrisoft.clickandlong.Main2Activity">
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="这一刻的想法..."
        android:layout_marginLeft="20dp"
        android:layout_marginTop="85dp"
        android:background="@null"/>
</RelativeLayout>
```

（7）在 MainActivity 类中重写 onLongClick() 方法，该方法为右上角按钮的长按事件。在该方法中首先需要将返回值修改为 true，否则长按事件将与单击事件冲突。然后通过 Intent 对象进行界面的跳转，代码如下：

```
@Override
public boolean onLongClick(View view) {
    //从当前界面跳转至发表这一刻想法的界面
    Intent intent = new Intent(MainActivity.this,Main2Activity.class);
    startActivity(intent);
    return true;        //此处设置为true否则长按事件将与单击事件冲突
}
```

（8）运行本实例，效果如图 11.9 所示，单击右上角按钮将显示如图 11.10 所示的菜单对话框，长按右上角按钮将显示如图 11.11 所示的发表这一刻想法的界面。

图 11.9　微信朋友圈

图 11.10　菜单对话框

图 11.11　发表这一刻想法的界面

11.4 手势检测

手势是指用户手指或触摸笔在屏幕上连续触碰的行为。例如，在屏幕上从左到右或从上到下划出的一个动作就是手势。Android 为手势行为提供了支持。最常用的就是手势检测。下面将详细介绍如何实现手势检测。

Android 为手势检测提供了一个 GestureDetector 类，该类代表了一个手势检测器。创建 GestureDetector 时，需要传入一个 GestureDetector.OnGestureListener 实例。GestureDetector.OnGestureListener 代表一个监听器，负责对用户的手势行为提供响应。GestureDetector.OnGestureListener 中包含的事件处理方法如表 11.3 所示。

表 11.3 GestureDetector.OnGestureListener 中的事件处理方法

方 法	说 明
boolean onDown(MotionEvent e)	当触摸事件按下时触发
boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)	当用户手指在触摸屏上“滑过”时触发，其中，velocityX、velocityY 代表“滑过”动作在横向、纵向上的速度
abstract void onLongPress(MotionEvent e)	当用户手指在触摸屏上长按时触发
boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)	当用户手指在触摸屏上连续向上或向下时触发
void onShowPress(MotionEvent e)	当用户手指在触摸屏上按下，并且未移动和松开时触发
boolean onSingleTapUp(MotionEvent e)	当用户手指在触摸屏上的轻击事件发生时触发

使用 Android 的手势检测只需要以下两个步骤：

（1）创建一个 GestureDetector 对象。创建该对象时必须实现一个 GestureDetector.OnGestureListener 监听器实例。

（2）为应用程序的 Activity 的 TouchEvent 事件绑定监听器，在事件处理中指定把 Activity 上的 TouchEvent 事件交给 GestureDetector 处理。这样 GestureDetector 就会检测是否触发了特定的手势动作。

下面通过一个实例演示手势的检测。

例 11.6 通过手势查看相片功能

在 Android Studio 中创建 Module，名称为“Gesture Detection”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 节点下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后将 TextView 组件删除，再添加一个 ViewFlipper 组件，并设置其 ID 属性为 flipper，最后在 res 目录下创建动画资源文件夹 anim，用于保存动画资源文件。

（2）修改主活动 MainActivity，让它实现 GestureDetector.OnGestureListener 接口，并且重写 onFling() 和 onTouchEvent() 方法。然后定义一个 ViewFlipper 类的对象、一个 GestureDetector 类的对象、

一个 Animation 动画数组、一个 int 型变量 distance（用于指定两点之间最小的距离）、一个图片资源数组，具体代码如下：

```
public class MainActivity extends AppCompatActivity
        implements GestureDetector.OnGestureListener {
    ViewFlipper flipper;                        //定义ViewFlipper
    GestureDetector detector;                   //定义手势检测器
    Animation[] animation = new Animation[4];//定义动画数组，为ViewFlipper指定切换动画
    final int distance = 50;                    //定义手势动作两点之间最小距离
    //定义图片数组
    private int[] images = new int[]{R.mipmap.img01, R.mipmap.img02, R.mipmap.img03,
            R.mipmap.img04, R.mipmap.img05, R.mipmap.img06,
            R.mipmap.img07, R.mipmap.img08, R.mipmap.img09,
    };
    @Override
    public boolean onDown(MotionEvent e) {
        return false;
    }
    @Override
    public void onShowPress(MotionEvent e) {

    }
    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        return false;
    }
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        return false;
    }
    @Override
    public void onLongPress(MotionEvent e) {
    }
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        return false;
    }
   }
```

（3）在 onCreate() 方法中，创建手势检测器，然后获取 ViewFlipper，并通过 for 循环加载图片数组中的图片，最后初始化动画数组。具体代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    detector = new GestureDetector(this, this);          //创建手势检测器
```

```
06      flipper = (ViewFlipper) findViewById(R.id.flipper);    //获取ViewFlipper组件
07      for (int i = 0; i < images.length; i++) {
08          ImageView imageView = new ImageView(this);         //创建图像组件
09          imageView.setImageResource(images[i]);             //设置图片资源
10          flipper.addView(imageView);                        //加载图片
11      }
12      //初始化动画数组
13      animation[0] = AnimationUtils.loadAnimation(this, R.anim.slide_in_left);
14      animation[1] = AnimationUtils.loadAnimation(this, R.anim.slide_out_left);
15      animation[2] = AnimationUtils.loadAnimation(this, R.anim.slide_in_right);
16      animation[3] = AnimationUtils.loadAnimation(this, R.anim.slide_out_right);
17  }
```

（4）在 onFling() 方法中通过触摸事件的 X 坐标判断是向左滑动还是向右滑动，并且为其设置动画。最后创建 onTouchEvent() 方法，将该 Activity 上的触摸事件交给 GestureDetector 处理。具体代码如下：

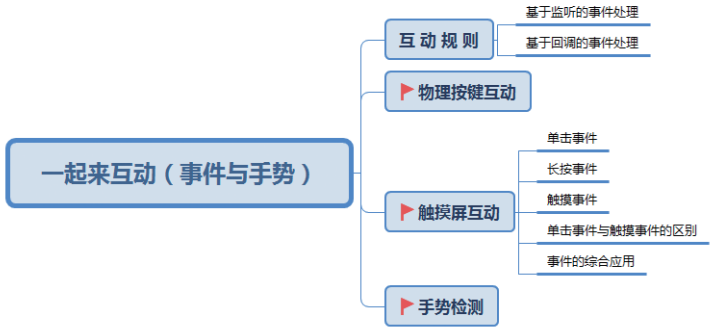
```
01  @Override
02  public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float
velocityY) {
03      /*
04      如果第一个触摸事件的X坐标到第二个触摸事件的X坐标的距离超过distance就是从右向左滑动
05       */
06      if (e1.getX() - e2.getX() > distance) {
07          //为flipper设置切换的动画效果
08          flipper.setInAnimation(animation[2]);
09          flipper.setOutAnimation(animation[1]);
10          flipper.showPrevious();
11          return true;
12          /*
13         如果第二个触摸事件的X坐标到第一个触摸事件的X坐标的距离超过distance就是从左向右滑动
14           */
15      } else if (e2.getX() - e1.getX() > distance) {
16          //为flipper设置切换的动画
17          flipper.setInAnimation(animation[0]);
18          flipper.setOutAnimation(animation[3]);
19          flipper.showNext();
20          return true;
21      }
22      return false;
23  }
24  @Override
25  public boolean onTouchEvent(MotionEvent event) {
26      //将该Activity上的触摸事件交给GestureDetector处理
27      return detector.onTouchEvent(event);
28  }
```

（5）运行本实例，效果如图 11.12 所示，左右滑动可以查看相片。

图 11.12　显示滑动查看相片

11.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 12 章

Android 的基本资源访问

Android 中的资源是指可以在代码中使用的外部文件，这些文件作为应用程序的一部分，被编译到应用程序中。在 Android 中，资源文件都被保存到 Android 应用的 res 目录下对应的子目录中，这些资源既可以在 Java 文件中使用，又可以在其他 XML 资源中使用。在 Android 中，利用这些资源可以为程序开发提供很多方便，使用资源不但有利于对程序进行修改，而且还可以非常方便地实现程序国际化。本章将对 Android 中的基本资源进行详细介绍。

12.1 字符串（string）资源

在 Android 中，当需要使用大量的字符串作为提示信息时，可以将这些字符串声明在资源文件中，从而实现程序的可配置性。下面对字符串资源进行详细介绍。

12.1.1 定义字符串资源文件

字符串资源文件位于 res\values 目录下，在使用 Android Studio 创建 Android 应用时，values 目录下会自动创建字符串资源文件 strings.xml（默认），该文件的基本结构如图 12.1 所示。

```
<resources>
    <string name="app_name">MyDemo</string>
</resources>
```

图 12.1　strings.xml 文件的基本结构

从图 12.1 中可以看出，在 strings.xml 文件中，根元素是 <resources> </resources> 标记，在该元素中，使用 <string></string> 标记定义各字符串资源。其中，<string> 标记的 name 属性用于指定字符串的名称，在 <string> 和 </string> 中间添加的内容为字符串资源的内容。

例如，在 strings.xml 资源文件中添加一个名称为 introduce 的字符串，内容是公司简介，具体代码如下：

```
01  <resources>
02      <string name="app_name">MyDemo</string>
03      <string name="introduce">明日科技有限公司是一家以计算机软件为核心的高科技企业，
```

```
04          多年来始终致力于行业管理软件开发、数字化出版物制作、
05          计算机网络系统综合应用以及行业电子商务网站开发等领域。</string>
06  </resources>
```

此外，还可以创建新的字符串资源文件，具体方法如下：

(1) 在 values 文件夹上单击鼠标右键，在弹出的快捷菜单中依次选择 New → XML → Values XML file 菜单项，将弹出创建资源文件的对话框，如图 12.2 所示。

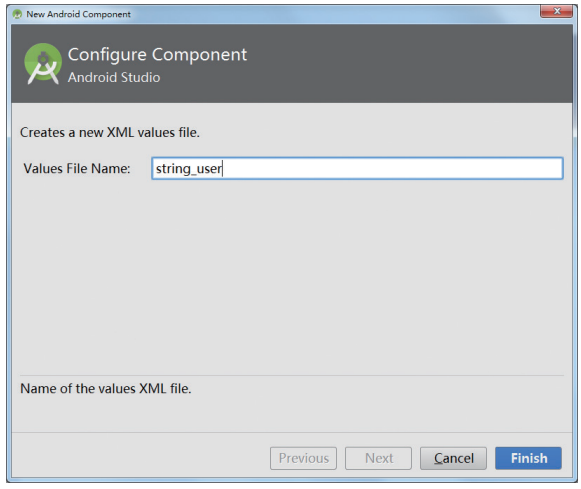


图 12.2　创建资源文件

（2）在文本框中输入自定义的资源文件名称（例如 string_user），单击 Finish 按钮，即可创建一个空的资源文件，代码结构如图 12.3 所示。

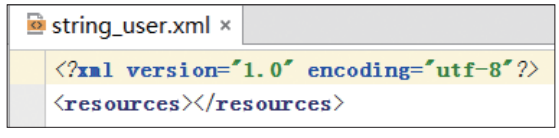


图 12.3　新创建的资源文件的代码结构

（3）在 <resources> </resources> 标记中使用 <string></string> 标记来创建字符串资源。

12.1.2 使用字符串资源

定义字符串资源后，就可以在 Java 文件或 XML 文件中使用字符串资源。

◆ 在 Java 文件中使用字符串资源的语法格式如下：

```
[<package>.]R.string.字符串名
```

例如，在 MainActivity 中要获取名称为 introduce 的字符串，可以使用下面的代码：

```
getResources().getString(R.string.introduce)
```

◆ 在 XML 文件中使用字符串资源的基本语法格式如下：

```
@[<package>:]string/字符串名
```

例如，在定义 TextView 组件时，通过字符串资源设置 android:text 属性值的代码如下：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/introduce" />
```

12.2 颜色（color）资源

颜色资源也是 Android 应用开发常用的资源，用于设置文字和背景的颜色等。下面将对颜色资源进行详细介绍。

12.2.1 颜色值的定义

在 Android 中，颜色值通过 RGB（红、绿、蓝）色值和一个透明度（Alpha）值表示。它必须以“#”开头，后面用 Alpha-Red-Green-Blue 形式的内容。其中，Alpha 值可以省略，如果省略则表示颜色默认是完全不透明的。在通常情况下，颜色值表示方式有如表 12.1 所示的 4 种形式，可以使用任意一种形式。

表 12.1　Android 支持的颜色值及其描述

颜 色 格 式	描　　述	举　　例
#RGB	使用红、绿、蓝三原色的值来表示颜色，其中，红、绿和蓝采用 0~f 来表示	要表示红色，可以使用 #f00
#ARGB	使用透明度以及红、绿、蓝三原色来表示颜色，其中，透明度、红、绿和蓝均采用 0~f 来表示	要表示半透明的红色，可以使用 #6f00
#RRGGBB	使用红、绿、蓝三原色的值来表示颜色，与 #RGB 不同的是，这里的红、绿和蓝使用 00~ff 来表示	要表示蓝色，可以使用 #0000ff
#AARRGGBB	使用透明度以及红、绿、蓝三原色来表示颜色，其中，透明度、红、绿和蓝均采用 00~ff 来表示	要表示半透明的绿色，可以使用 #6600ff00

说明　在表示透明度时，0表示完全透明，f表示完全不透明。

12.2.2 定义颜色资源文件

颜色资源文件位于 res\values 目录下，在使用 Android Studio 创建 Android 项目时，values 目录下会自动创建默认的颜色资源文件 colors.xml，该文件的基本结构如图 12.4 所示。

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

图 12.4　colors.xml 文件的基本结构

从图 12.4 中可以看出，在 colors.xml 文件中，根元素是 <resources></resources> 标记，在该元素中，使用 <color></color> 标记定义各颜色资源，其中，name 属性用于指定颜色资源的名称，在 <color> 和 </color> 标记中间添加颜色值。

例如，在 colors.xml 资源文件中添加 4 个颜色资源，其中第 1 个名称为 title，颜色值采用 #AARRGGBB 格式；第 2 个名称为 title1，颜色值采用 #ARGB 格式；第 3 个名称为 content，颜色值采用 #RRGGBB 格式；第 4 个名称为 content1，颜色值采用 #RGB 格式，关键代码如下：

```
01  <color name="title">#66ff0000</color>
02  <color name="title1">#6f00</color>
03  <color name="content">#ff0000</color>
04  <color name="content1">#f00</color>
```

说明 第1个和第2个资源都表示半透明的红色；第3个和第4个资源都表示完全不透明的红色。

另外，还可以创建新的颜色资源文件，具体方法同 12.1.1 节中介绍的创建新的字符串资源类似，只是设置的文件名不同，这里不再赘述。

在图 12.4 中可以看到每一行 <color></color> 标记的左侧都有一个小色块儿，如果不想使用该颜色，可以单击小色块，将打开选择颜色的对话框，如图 12.5 所示。在该对话框中选择想要使用的颜色，选定颜色后单击 Choose 按钮即可完成选择颜色的操作。

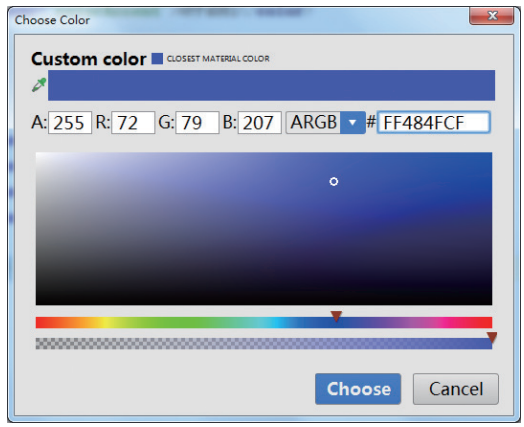


图 12.5　选择颜色的对话框

12.2.3 使用颜色资源

定义颜色资源后，就可以在 Java 或 XML 文件中使用该颜色资源了。

◆ 在 Java 文件中使用颜色资源的语法格式如下：

```
[<package>.]R.color.颜色资源名
```

例如，在 MainActivity 中，通过颜色资源为 TextView 组件设置文字颜色，可以使用下面的代码：

```
01  TextView tv=(TextView)findViewById(R.id.title);
02  tv.setTextColor(getResources().getColor(R.color.title));
```

◆ 在 XML 文件中使用颜色资源的基本语法格式如下：

```
@[<package>:]color/颜色资源名
```

例如，在定义 TextView 组件时，通过颜色资源为其指定 android:textColor 属性，即设置组件内文字的颜色，代码如下：

```
01  <TextView
02      android:layout_width="wrap_content"
03      android:layout_height="wrap_content"
04      android:textColor="@color/title" />
```

12.3 尺寸（dimen）资源

尺寸资源也是进行 Android 应用开发时比较常用的资源，它通常用于设置文字的大小、组件的间距等。下面对尺寸资源进行详细介绍。

12.3.1 Android 支持的尺寸单位

在 Android 中，支持的尺寸单位及其描述如表 12.2 所示。

表 12.2　Android 支持的尺寸单位及其描述

尺 寸 单 位	描　　述	适　用　于
dip 或 dp（设置独立像素）	一种基于屏幕密度的抽象单位	屏幕的清晰度
sp（比例像素）	主要用于处理字体的大小，可以根据用户字体大小首选项进行缩放	字体大小
px（Pixels，像素）	每个 px 对应屏幕上的一个点	屏幕横向、纵向的像素个数
pt（points，磅）	屏幕物理长度单位，1 磅为 1/72 英寸	设置字体大小（不常用）
in（Inches，英寸）	标准长度单位。1 英寸等于 2.54 厘米	屏幕对角线长度
mm（Millimeters，毫米）	屏幕物理长度单位	屏幕物理长度

在表 12.2 列出的几种尺寸单位中，比较常用的是 dp 和 sp。

◆ dp

在屏幕密度为 160dpi（每英寸 160 点）的显示器上，1dp=1px。随着屏幕密度的改变，dp 与 px 的换算也会发生改变。例如，在屏幕密度为 320dpi 的显示器上，1dp=2px。

◆ sp

与 dp 类似，该尺寸单位主要用于字体显示，它可以根据用户对字体大小的首选项进行缩放。因此，字体大小使用 sp 单位可以确保文字按照用户选择的大小显示。

12.3.2 使用尺寸资源

尺寸资源文件位于 res\values 目录下，在使用当前版本的 Android Studio 创建 Android 项目时，

不会自动创建尺寸资源文件，需要在 values 目录下手动创建尺寸资源文件 dimens.xml，在 values 节点上单击鼠标右键，在弹出的菜单中依次选择 New → Values resource file 选项，如图 12.6 所示。

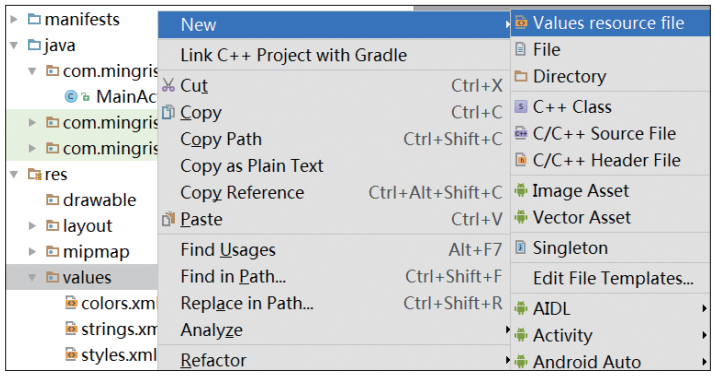


图 12.6　新建 Values resource file

将弹出 New Resouce File 对话框，在该对话框中填写资源文件名称，然后单击 OK 按钮，如图 12.7 所示。

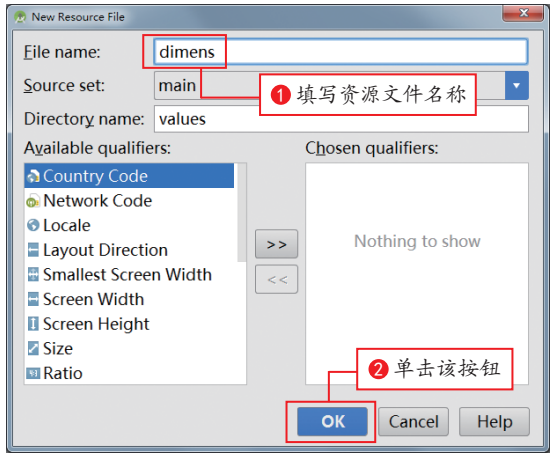


图 12.7　创建尺寸资源文件

该文件默认的基本结构如图 12.8 所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources></resources>
```

图 12.8　dimens.xml 文件默认的基本结构

在 dimens.xml 文件中，根元素是 <resources></resources> 标记，在该元素中，使用 <dimen></dimen> 标记定义各尺寸资源。其中，name 属性用于指定尺寸资源的名称，在 <dimen> 和 </dimen> 标记中间定义一个尺寸常量。

例如，在 dimens.xml 资源文件中添加两个尺寸资源，其中一个名称为 title，尺寸值为 24sp；另一个名称为 content，尺寸值为 14dp，具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <resources>
03      <dimen name="title">24sp</dimen>
```

```
    <dimen name="content">14dp</dimen>
</resources>
```

定义尺寸资源后，就可以在 Java 或 XML 文件中使用该尺寸资源了。

◆ 在 Java 文件中使用尺寸资源的语法格式如下：

```
[<package>.]R.color.尺寸资源名
```

例如，在 MainActivity 中，通过尺寸资源为 TextView 组件设置文字大小，可以使用下面的代码：

```
TextView tv=(TextView)findViewById(R.id.title);
tv.setTextSize(getResources().getDimension(R.dimen.title));
```

◆ 在 XML 文件中使用尺寸资源的基本语法格式如下：

```
@[<package>:]dimen/尺寸资源名
```

例如，在定义 TextView 组件时，通过尺寸资源为其指定 android: textSize 属性，即设置组件内文字的大小，代码如下：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="@dimen/title" />
```

下面通过一个实例来演示字符串、颜色和尺寸资源的应用。

例 12.1 Windows Phone “方格子”界面

在 Android Studio 中创建一个 Module，名称为“Windows Phone”。实现本实例的具体步骤如下：

（1）修改默认创建的布局文件 activity_main.xml。首先将默认添加的布局管理器修改为垂直的线性布局管理器，并设置上边距和下边距为 80dp，然后在该线性布局管理器中再添加 3 个水平线性布局管理器，设置它们的 android:layout_weight 属性均为 1，接下来在每个水平线性布局管理器中，添加 3 个文本框组件，并设置它们的 android:layout_weight 属性均为 1。

（2）创建尺寸资源文件 dimens.xml，然后在该文件中添加两个尺寸资源，一个用于指定文字的大小，另一个用于指定文本框的外边距，具体代码如下：

```
<dimen name="wordsize">18sp</dimen>
<dimen name="margin">5dp</dimen>
```

（3）为第二个和第三个水平线性布局管理器设置顶外边距，使用定义的尺寸资源 margin，关键代码如下：

```
android:layout_marginTop="@dimen/margin"
```

（4）为 ID 为 textView2、textView3、textView5、textView6、textView8 和 textView9 的文本框设置左外边距，使用定义的尺寸资源 margin，关键代码如下：

```
android:layout_marginLeft="@dimen/margin"
```

（5）使用尺寸资源 wordsize 来为每个文本框设置文字大小，关键代码如下：

```
android:textSize="@dimen/wordsize"
```

（6）在默认创建的颜色资源文件 colors.xml 中，添加一个名称为 wordcolor 的颜色资源，设置它的颜色值为不透明的白色，然后再添加名称为 textView1 至 textView9 的背景颜色资源，并设置它们的颜色值均为半透明，具体代码如下：

```
<color name="wordcolor">#FFFFFF</color>
<color name="textView1">#BBE24A83</color>
<color name="textView2">#BB318AD6</color>
<color name="textView3">#BBD73943</color>
<color name="textView4">#BBE69A08</color>
<color name="textView5">#BBBD9663</color>
<color name="textView6">#BBD45ABC</color>
<color name="textView7">#BB4AA6D6</color>
<color name="textView8">#BB8064D2</color>
<color name="textView9">#BBF7A81E</color>
```

（7）为每个文本框设置背景颜色，关键代码如下：

```
android:layout_weight="1"
```

（8）在默认创建的字符串资源文件 strings.xml 中，添加名称为 textView1 至 textView9 的字符串资源，具体代码如下：

```
<string name="textView1">微信</string>
<string name="textView2">通讯录</string>
<string name="textView3">QQ</string>
<string name="textView4">相机</string>
<string name="textView5">时钟</string>
<string name="textView6">备忘录</string>
<string name="textView7">音乐</string>
<string name="textView8">互联网</string>
<string name="textView9">邮件</string>
```

说明 在Android的资源文件中，需要使用“ ”表示空格。

（9）使用步骤（8）中定义的字符串资源为每个文本框设置文字，例如，为第一个文本框设置使用字符串资源 textView1，可以使用下面的代码。

```
android:text="@string/textView1"
```

（10）为每个文本框设置文字颜色为颜色资源 wordcolor 的值，关键代码如下：

```
android:textColor="@color/wordcolor"
```

（11）运行本实例，效果如图 12.9 所示。

图 12.9　Windows Phone 的“方格子”界面

12.4 布局（layout）资源

布局资源是 Android 中最常用的一种资源，从第一个 Android 应用开始，我们就已经在使用布局资源了，而且在 3.3 节中已经详细介绍了各种布局管理器的应用。因此，这里不再详细介绍布局管理器的知识，只对如何使用布局资源进行简单的归纳。

在 Android 中，将布局资源文件放置在 res/layout 目录下，布局资源文件的根元素通常为各种布局管理器，在该布局管理器中，放置各种 View 组件或是嵌套的其他布局管理器。

布局文件创建完成后，可以在 Java 代码或是 XML 文件中使用。在 Java 代码中，可以通过下面的语法格式访问布局文件：

```
[<package>.]R.layout.<文件名>
```

例如，在 MainActivity 的 onCreate() 方法中，可以通过下面的代码指定该 Activity 应用的布局文件为 main.xml。

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

在 XML 文件中，可以通过下面的语法格式访问布局资源文件：

```
@[<package>:]layout.文件名
```

例如，如果要在一个布局文件 main.xml 中包含另一个布局文件 image.xml，可以在 main.xml 文件中使用下面的代码：

```
<include layout="@layout/image" />
```

12.5 数组（array）资源

同 Java 一样，Android 中也允许使用数组。但是在 Android 中，不推荐在 Java 文件中定义数组，而是推荐使用数组资源文件来定义数组。下面对数组资源进行详细介绍。

12.5.1 定义数组资源文件

数组资源文件需要放置在 res\values 目录下。在使用 Android Studio 创建 Android 项目后，并没有在 values 目录下自动创建数组资源文件，需要手动创建（例如 arrays.xml）。定义数组时 XML 资源文件的根元素是 <resources></resources> 标记，在该元素中可以包括以下 3 个子元素。

- ◆ <array> 子元素：用于定义普通类型的数组。
- ◆ <integer-array> 子元素：用于定义整数数组。
- ◆ <string-array> 子元素：用于定义字符串数组。

说明 创建数组资源文件的方法可参考12.3.2小节中的创建尺寸资源文件的方法。

无论使用哪一个子元素，都可以在子元素的起始标记中使用 name 属性定义数组名称，并且在子元素的起始标记和结束标记中间使用 <item></item> 标记定义数组元素。

例如，要定义一个名称为 listitem.xml 的数组资源文件，并在该文件中添加一个名称为 listItem，包括 3 个数组元素的字符串数组，可以使用下面的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="listItem">
        <item>账号管理</item>
        <item>手机号码</item>
        <item>辅助功能</item>
    </string-array>
</resources>
```

12.5.2 使用数组资源

定义数组资源后，就可以在 Java 或 XML 文件中使用该数组资源了。

- ◆ 在 Java 文件中使用数组资源的语法格式如下：

```
[<package>.]R.array.数组名
```

例如，在 MainActivity 中，要获取名称为 listItem 的字符串数组，可以使用下面的代码：

```
String[] arr=getResources().getStringArray(R.array.listItem);
```

- ◆ 在 XML 文件中使用数组资源的基本语法格式如下：

```
@[<package>:]array/数组名
```

例如，在定义 ListView 组件时，通过字符串数组资源为其指定 android:entries 属性的代码如下：

```
<ListView
    android:id="@+id/listView1"
    android:entries="@array/listItem"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</ListView>
```

例 12.2 使用数组资源实现 Windows Phone 界面

在 Android Studio 中创建 Module，名称为“Array Resource”。实现本实例的具体步骤如下：

（1）将例 12.1 中的布局代码和资源代码都复制到本实例中，然后将 textView1 至 textView9 所有的背景和文字删除。

（2）在 res/values 目录中创建一个名称为 arrays.xml 的数组资源文件，在该文件中添加两个数组，一个是整型数组，用于定义背景颜色，另一个是字符串数组，用于定义文本框上显示的文字，关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!--颜色代码-->
    <integer-array name="bgcolor">
        <item>0xBBE24A83</item>
        <item>0xBB318AD6</item>
        <item>0xBBD73943</item>
        <item>0xBBE69A08</item>
        <item>0xBBBD9663</item>
        <item>0xBBD45ABC</item>
        <item>0xBB4AA6D6</item>
        <item>0xBB8064D2</item>
        <item>0xBBF7A81E</item>
    </integer-array>
    <!--文字代码-->
    <string-array name="word">
        <item>微信</item>
        <item>通讯录</item>
        <item>QQ</item>
        <item>相机</item>
        <item>时钟</item>
        <item>备忘录</item>
        <item>音乐</item>
        <item>互联网</item>
        <item>邮件</item>
    </string-array>
</resources>
```

（3）打开默认创建的 MainActivity，让其直接继承 Activity，然后定义一个用于保存文本框组件 ID 的数组，具体代码如下：

```
int[] tvid = {R.id.textView1, R.id.textView2, R.id.textView3, R.id.textView4,
        R.id.textView5, R.id.textView6, R.id.textView7, R.id.textView8,
        R.id.textView9};    //文本框组件ID
```

（4）在 onCreate() 方法中，首先获取保存在数组资源文件中的两个数组资源，然后通过循环为每个文本框设置背景颜色和显示文字，具体代码如下：

```
int[] color=getResources().getIntArray(R.array.bgcolor);      //获取保存背景颜色的数组
String[] word=getResources().getStringArray(R.array.word);    //获取保存显示文字的数组
//通过循环为每个文本框设置背景颜色和显示文字
for(int i=0;i<9;i++){
    TextView tv=(TextView)findViewById(tvid[i]);               //获取文本框组件对象
    tv.setBackgroundColor(color[ i]);                          //设置背景颜色
    tv.setText(word[i]);                                       //设置显示文字
}
```

（5）运行本实例，将显示如图 12.10 所示的效果。

图 12.10　Windows Phone 的“方格子”界面

12.6 样式（style）资源

有时我们需要为某个类型的组件设置相似的格式，比如字体、颜色、背景色等。若每次都要为该组件指定这些属性，不仅会增加工作量，还不利于项目的后期维护。

在编写 Word 文档的时候，如果为某段文本设置了样式，那么该样式下的所有格式都会应用于这段文本中。Android 的样式与此类似，每种样式都会包含一组格式，一旦为某个组件设置了样式，该样式下的所有格式都会应用于该组件中。

样式资源主要用于对组件的显示样式进行控制，如改变文本框显示文字的大小和颜色等。样式资源文件位于 res\values 目录下，它的根元素是 <resources></resources> 标记，在该元素中，使用 <style> 标记定义样式。其中，name 属性用于指定样式的名称；在 <style> 和 </style> 标记中间添加 <item></item> 标记来定义格式项，在一个 <style></style> 标记中，可以包括多个 <item></item> 标记。

例如，在默认创建的 styles.xml 中定义一个名称为 title 的样式，在该样式中定义两个样式，一个是设置文字大小的样式，另一个是设置文字颜色的样式，关键代码如下：

```
<style name="title">
    <item name="android:textSize">30sp</item>
    <item name="android:textColor">#f60</item>
</style>
```

在 Android 中，还支持继承样式的功能，只需要在 <style></style> 标记中使用 parent 属性进行设置。例如，定义一个名称为 basic 的样式，再定义一个名称为 title 的样式，并让该样式继承 basic 样式，关键代码如下：

```
<style name="basic">
    <item name="android:textSize">30sp</item>
    <item name="android:textColor">#f60</item>
</style>
<style name="title" parent="basic">
<item name="android:padding">10dp</item>
<item name="android:gravity">center</item>
</style>
```

说明 当一个样式（子样式）继承自另一个样式（父样式）后，如果在该子样式中，出现了与父样式相同的属性，将使用子样式中定义的属性值。

例 12.3 模拟今日头条的新闻界面

在 Android Studio 中创建 Module，名称为“Style Resource”。实现本实例的具体步骤如下：

（1）首先在 colors.xml 文件中添加黑色颜色代码，然后打开 values 目录下的 styles.xml，在该文件中添加名称为 black 的样式资源，设置字体样式为加粗，字体颜色为黑色。具体代码如下：

```
<style name="black">
    <item name="android:textStyle">bold</item>
    <item name="android:textColor">@color/black</item>
</style>
```

（2）打开默认创建的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，并为其设置背景图片，然后在布局管理器中添加一个 TextView 组件用于显示标题，并且设置其使用名称为 black 的样式资源。关键代码如下：

```
<!-- 标题-->
<TextView
    android:id="@+id/title"
    style="@style/black"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="70dp"
    android:layout_marginRight="70dp"
    android:layout_marginTop="50dp"
    android:text="@string/title"
    android:textSize="20sp" />
```

（3）在 styles.xml 文件中，添加名称为 text_down 的样式，让其继承自 black 样式，然后再添加使组件在布局管理器的水平居中位置显示的样式。具体代码如下：

```
<style name="text_down" parent="black">
    <item name="android:layout_gravity">center_horizontal</item>
</style>
```

（4）在标题下面添加一个 TextView 组件（用于显示新闻内容）、一个 ImageView 组件（用于显示图片），以及一个 TextView 组件（用于显示图注），并且让显示图注的文本框使用步骤（3）中编写的名称为 text_down 的样式。其中，图注文本框的代码如下：

```
<!--图注字-->
<TextView
    android:id="@+id/text_down"
    style="@style/text_down"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/image"
    android:layout_marginTop="10dp "
    android:text="@string/text_down"/>
```

（5）运行本实例，将显示如图 12.11 所示的界面。

图 12.11　模拟今日头条的新闻页面

12.7 Android 程序国际化

国际化的英文单词是 Internationalization，因为该单词较长，将其简称为 I18N，其中，I 是该单词的第一个字母；18 表示中间省略的字母个数；N 是该单词的最后一个字母。Android 程序国际化，是指程序可以根据系统所使用的语言，将界面中的文字翻译成与之对应的语言。这样，可以让程序更加通用。Android 可以通过资源文件非常方便地实现程序的国际化。下面将以国际字符串资源为例，介绍如何实现 Android 程序的国际化。

在编写 Android 项目时，通常都是将程序中要使用的字符串资源放置在 res\values 目录下的 strings.xml 文件中，为了实现这些字符串资源的国际化，可以在 Android 项目的 res 目录下创建对应于各个语言的资源文件夹（例如，为了让程序兼容简体中文、繁体中文和美式英文，可以分别创建

名称为 values-zh-rCN、values-zh-rTW 和 values-en-rUS 的文件夹，如图 12.12 所示），然后在每个文件夹中创建一个对应的 strings.xml 文件，并在该文件中定义对应语言的字符串即可。这样，当程序运行时，就会自动根据操作系统所使用的语言来显示对应的字符串信息。

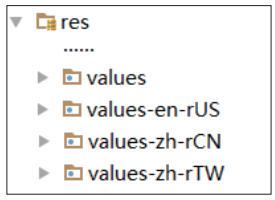


图 12.12　创建对应于各个语言的资源文件夹

下面通过一个实例来说明 Android 程序的国际化。

例 12.4　系统语言不同显示文字不同

在 Android Studio 中创建 Module，名称为“International Language”。实现本实例的具体步骤如下：

（1）将 12.5.2 小节中的例 12.2 中的代码全部复制到该 Module 中。

（2）打开新建项目的 res\values 目录，在默认创建的 strings.xml 文件中，将默认添加的字符串变量全部删除，然后再将实例 02 中的字符串资源代码也全部复制到此实例的字符串资源文件中。

说明　在res\values目录中创建的strings.xml文件，为默认使用的字符串资源文件。当在后面创建的资源文件（与各语言对应的资源文件）中没有与系统使用的语言相对应的文件时，将使用该资源文件。

（3）在 res 目录中分别创建名称为 values-zh-rCN（简体中文）、values-zh-rTW（繁体中文）和 values-en-rUS（美式英文）目录，并将 res\values 目录下的 strings.xml 文件分别复制到这 3 个目录中，如图 12.13 所示（需要切换到 project 项目结构类型）。

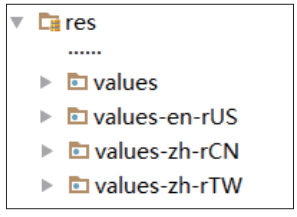


图 12.13　完成后的文件夹

技巧　复制string.xml文件的方法为：选中strings.xml，并按下快捷键〈Ctrl+C〉，然后选中values目录，并按下快捷键〈Ctrl+V〉，将弹出如图12.14所示的对话框，在该对话框中单击“...”按钮，将弹出如图12.15所示的选择目标路径的对话框，在该对话框中选择目标目录，单击OK按钮。将返回如图12.14所示的确认粘贴目录的对话框，在该对话框中单击OK按钮。

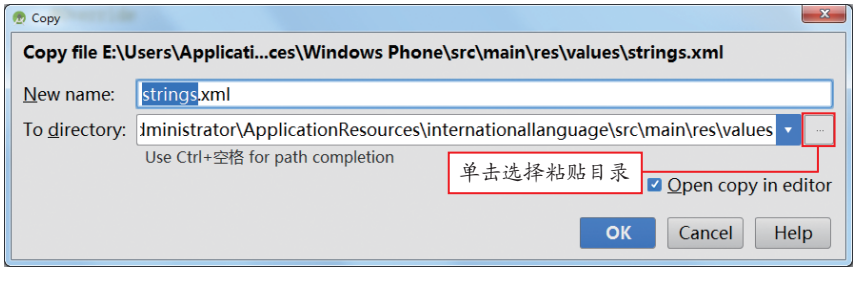


图 12.14　确认粘贴目录

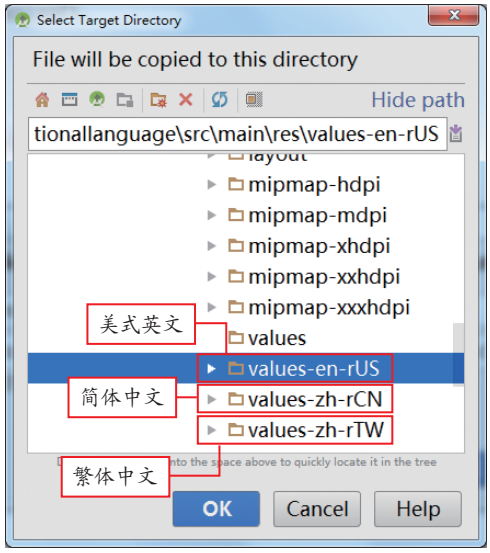


图 12.15　选择粘贴目录

（4）修改 res\values-zh-rTW 目录中的 strings.xml 文件，关键代码如下：

```
<resources>
    <string name="app_name">Windows Phone 10.1</string>
    <string name="textView1">微信</string>
    <string name="textView2">通訊錄</string>
    <string name="textView3">QQ</string>
    <string name="textView4">相機</string>
    <string name="textView5">時鐘</string>
    <string name="textView6">備忘錄</string>
    <string name="textView7">音樂</string>
    <string name="textView8">互聯網</string>
    <string name="textView9">郵件</string>
</resources>
```

（5）修改 res\values-en-rUS 目录中的 strings.xml 文件，关键代码如下：

```
<resources>
    <string name="app_name">Windows Phone</string>
    <string name="textView1">WeChat</string>
    <string name="textView2">Contacts</string>
    <string name="textView3">QQ</string>
    <string name="textView4">Camera</string>
    <string name="textView5">Clock</string>
    <string name="textView6">Notes</string>
    <string name="textView7">Music</string>
    <string name="textView8">Browser</string>
    <string name="textView9">E-mail</string>
</resources>
```

（6）运行本实例，将 Module“International Language”进行国际化，即实现在不同语言的操作系统下显示不同的文字，如图 12.16 所示，显示简体中文，如图 12.17 所示，显示繁体中文，如图 12.18 所示，显示美式英语。

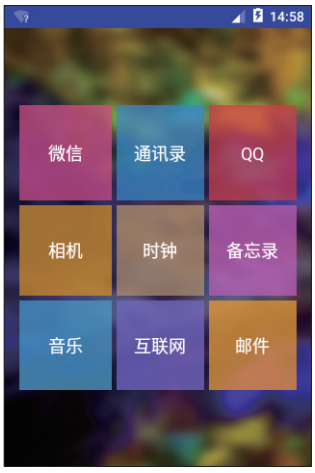


图 12.16　简体中文环境中的运行结果

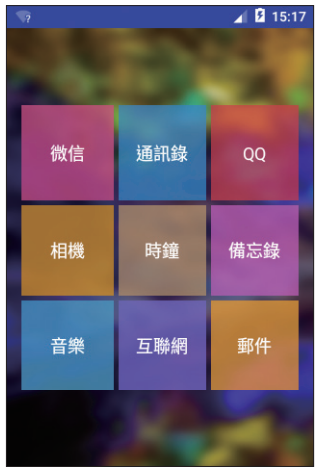


图 12.17　繁体中文环境中的运行结果

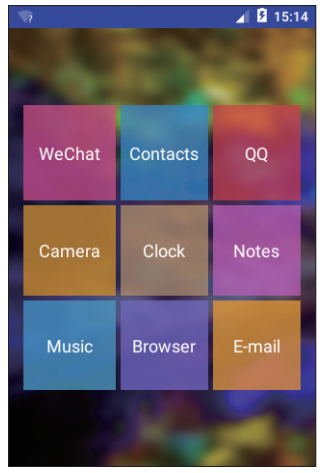


图 12.18　美式英语环境中的运行结果

说明　本实例需要修改系统语言设置，才可以显示以上图片中的运行效果。

12.8 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。

第 13 章

Android 高级资源的使用

在上一章已经介绍了 Android 基本资源的使用，本章将介绍 Android 的高级资源，主要包括菜单资源、图像资源的使用。下面将对这两类高级资源进行详细的介绍。

13.1 菜单（menu）资源

在桌面应用程序中，菜单的使用十分广泛。但是在 Android 应用中，菜单大幅度减少。不过 Android 中提供了两种实现菜单的方法，分别是通过 Java 代码创建菜单和使用菜单资源文件创建菜单，Android 推荐使用菜单资源来定义菜单，下面进行详细介绍。

13.1.1 定义菜单资源文件

菜单资源文件通常放置在 res\menu 目录下，在 Android Studio 中创建项目时，默认是不能自动创建 menu 目录的，所以需要手动创建。菜单资源的根元素通常使用 <menu></menu> 标记，在该标记中可以包含多个 <item></item> 标记，用于定义菜单项，可以通过表 13.1 所示的各属性来为菜单项设置标题等内容。

表 13.1 <item></item> 标记的常用属性

属　性	描　述
android:id	用于为菜单项设置 ID，也就是唯一标识
android:title	用于为菜单项设置标题
android:alphabeticShortcut	用于为菜单项指定字符快捷键
android:numericShortcut	用于为菜单项指定数字快捷键
android:icon	用于为菜单项指定图标
android:enabled	用于指定该菜单项是否可用
android:checkable	用于指定该菜单项是否可选
android:checked	用于指定该菜单项是否已选中
android:visible	用于指定该菜单项是否可见

13.1.2 使用菜单资源

在 Android 中，定义的菜单资源可以用来创建选项菜单（Option Menu）和上下文菜单（Content Menu）。使用菜单资源创建这两种菜单的方法是不同的，下面分别进行介绍。

1．选项菜单

当用户单击菜单按钮时，弹出的菜单就是选项菜单。使用菜单资源创建选项菜单的具体步骤如下。

（1）重写 Activity 中的 onCreateOptionsMenu() 方法。在该方法中，首先创建一个用于解析菜单资源文件的 MenuInflater 对象，然后调用该对象的 inflate() 方法解析一个菜单资源文件，并把解析后的菜单保存在 menu 中，关键代码如下：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater=new MenuInflater(this);      //实例化一个MenuInflater对象
    inflater.inflate(R.menu.optionmenu, menu);         //解析菜单文件
    return super.onCreateOptionsMenu(menu);
}
```

（2）重写 onOptionsItemSelected() 方法，用于当菜单项被选择时，做出相应的处理。例如，当菜单项被选择时，弹出一个消息提示框显示被选中菜单项的标题，可以使用下面的代码：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Toast.makeText(MainActivity.this, item.getTitle(), Toast.LENGTH_SHORT).show();
    return super.onOptionsItemSelected(item);
}
```

例 13.1 明日学院的选项菜单

在 Android Studio 中创建 Module，名称为“Menu Resource”。实现本实例的体步骤如下：

（1）打开默认创建的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后为布局管理器添加一张背景图片。

（2）在 res 目录上单击鼠标右键，在弹出的快捷菜单中选择 New → Android resource directory 菜单项，在弹出的对话框中，在 Resource type 类型的下拉列表框中选择 menu，单击 OK 按钮。创建一个 menu 目录，并在该目录中创建一个名称为 menu.xml 的菜单资源文件，在 menu.xml 文件中，定义两个菜单项分别是“设置”和“关于”，显示文字通过字符串资源指定。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/settings"
        android:title="@string/menu_title_settings"></item>
    <item
        android:id="@+id/regard"
        android:title="@string/menu_title_regard"></item>
</menu>
```

（3）在 MainActivity 中重写 onCreateOptionsMenu() 方法，在该方法中，首先创建一个用于解析菜单资源文件的 MenuInflater 对象，然后调用该对象的 inflate() 方法解析一个菜单资源文件，并把解析后的菜单保存在 menu 中，最后将菜单返回，关键代码如下：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater menuInflater = new MenuInflater(this);  //实例化一个MenuInflater对象
    menuInflater.inflate(R.menu.menu, menu);             //解析菜单文件
    return super.onCreateOptionsMenu(menu);
}
```

（4）在 com.mingrisoft 包中创建一个名称为 Settings 的 Activity，用于显示设置界面，然后再创建一个名称为 Regard 的 Activity 用于显示关于界面。

（5）在 MainActivity 中重写 onOptionsItemSelected() 方法，然后通过 switch() 语句根据选中的菜单 id 跳转到指定的 Activity，关键代码如下：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {                          //获取选中的菜单id
        case R.id.settings:                              //跳转到设置页面
            Intent intent = new Intent(MainActivity.this, Settings.class);
            startActivity(intent);
            break;
        case R.id.regard:                                //跳转到关于页面
            Intent intent1 = new Intent(MainActivity.this, Regard.class);
            startActivity(intent1);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

（6）运行本实例，将显示如图 13.1 所示的界面，单击屏幕右上方的菜单按钮，如图 13.2 所示，单击“关于”菜单项将跳转到关于界面。

图 13.1　显示界面

图 13.2　显示选项菜单

2．上下文菜单

当用户长按组件时，弹出的菜单就是上下文菜单。使用菜单资源创建上下文菜单的具体步骤如下。

（1）在 Activity 的 onCreate() 方法中注册上下文菜单。例如，为文本框组件注册上下文菜单，当长按该文本框时，显示上下文菜单，可以使用下面的代码。

```
TextView tv=(TextView)findViewById(R.id.show);
registerForContextMenu(tv);                          //为文本框注册上下文菜单
```

（2）重写 Activity 中的 onCreateContextMenu() 方法。在该方法中，首先创建一个用于解析菜单资源文件的 MenuInflater 对象，然后调用该对象的 inflate() 方法解析一个菜单资源文件，并把解析后的菜单保存在 menu 中，最后为菜单头设置图标和标题，关键代码如下：

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    MenuInflater inflator=new MenuInflater(this);      //实例化一个MenuInflater对象
    inflator.inflate(R.menu.menus, menu);              //解析菜单文件
    menu.setHeaderIcon(R.mipmap.ic_launcher);          //为菜单头设置图标
    menu.setHeaderTitle("请选择");                      //为菜单头设置标题
}
```

（3）重写 onContextItemSelected() 方法，用于当菜单项被选择时，做出相应的处理。例如，当菜单项被选择时，弹出一个消息提示框显示被选中菜单项的标题，可以使用下面的代码：

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    Toast.makeText(MainActivity.this, item.getTitle(), Toast.LENGTH_SHORT).show();
    return super.onContextItemSelected(item);
}
```

例 13.2 模拟微信朋友圈的消息菜单

在 Android Studio 中创建 Module，名称为“Context Menu”。实现本实例的具体步骤如下：

（1）在 res 目录下创建一个 menu 目录，并在该目录中创建一个名称为 introduce_menu.xml 的菜单资源文件，在该文件中定义 4 个菜单项，分别是复制、收藏、翻译和举报。菜单项标题通过字符串资源指定，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menu_copy"
        android:title="@string/introduce_copy"></item>
    <item
        android:id="@+id/menu_collect"
        android:title="@string/introduce_collect"></item>
    <item
        android:id="@+id/menu_translate"
        android:title="@string/introduce_translate"></item>
```

```
    <item
        android:id="@+id/menu_report"
        android:title="@string/introduce_report"></item>
</menu>
```

（2）打开默认创建的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后添加实现类似朋友圈中显示一条消息的每个组件。例如，将用于显示消息详细内容的文本框的 ID 属性设置为 introduce，可以使用下面的代码。

```
<!—显示消息的TextView组件—>
<TextView
    android:id="@+id/introduce"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/name"
    android:layout_below="@+id/name"
    android:layout_marginTop="@dimen/margin"
    android:text="@string/introduce"/>
```

（3）修改默认创建的 MainActivity 类，让 MainActivity 直接继承 Activity，并声明 TextView 组件，然后在重写的 onCreate() 方法中，获取要添加上下文菜单的 TextView 组件，并为其注册上下文菜单，关键代码如下：

```
public class MainActivity extends Activity {
    TextView introduce;                                    //声明TextView组件
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        introduce = (TextView) findViewById(R.id.introduce);//获取显示消息的TextView组件
        registerForContextMenu(introduce);                 //为文本框注册上下文菜单
    }
}
```

（4）在 MainActivity 中重写 onCreateContextMenu() 方法，在该方法中，创建一个用于解析菜单资源文件的 MenuInflater 对象，然后调用该对象的 inflate() 方法解析文本框的菜单资源文件，并把解析后的菜单保存在 menu 中，关键代码如下：

```
@Override
//创建上下文菜单
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    MenuInflater inflater = new MenuInflater(this);    //实例化一个MenuInflater对象
    inflater.inflate(R.menu.introduce_menu, menu);     //解析菜单文件
}
```

（5）重写 onContextItemSelected() 方法，在该方法中，通过 switch 语句判断用户选择的菜单选项来显示相应的提示信息，具体代码如下：

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_copy:                          //选中介绍文字菜单中的"复制"菜单项时
            Toast.makeText(MainActivity.this, "已复制", Toast.LENGTH_SHORT).show();
            break;
        case R.id.menu_collect:                       //选中介绍文字菜单中的"收藏"菜单项时
            Toast.makeText(MainActivity.this,"已收藏",Toast.LENGTH_SHORT).show();
            break;
    }
    return true;
}
```

（6）运行本实例，将显示如图 13.3 所示的界面。长按文字信息，将显示如图 13.4 所示的上下文菜单。

图 13.3　模拟微信朋友圈界面

图 13.4　显示上下文菜单

13.2 图像资源

13.2.1 图片资源

Drawable 资源是 Android 应用中使用最广泛、灵活的资源。它不仅可以直接使用图片作为资源，而且可以使用 XML 文件作为资源，只要一个 XML 文件可以被系统编译成 Drawable 子类的对象，那么该 XML 文件就可以作为 Drawable 资源。

说明 Drawable资源保存在res\drawable目录中，实际上，为了适应不同屏幕分辨率，通常情况下将其保存在res\drawable-mdpi、res\drawable-hdpi、res\drawable-xhdpi等目录下，这几个目录需要用户手动创建。其中，res\drawable-mdpi保存的是中等分辨率的图片；res\drawable-hdpi保存的是高分辨率的图片；res\drawable-xhdpi保存的是超高分辨率的图片。

1．图片资源

在 Android 中，不仅可以将扩展名为 .png、.jpg 和 .gif 的普通图片作为图片资源，而且还可以将扩展名为 .9.png 的 9-Patch 图片作为图片资源。扩展名为 .png、.jpg 和 .gif 的普通图片较常见，它们通常是通过绘图软件完成的，下面来对扩展名为 .9.png 的 9-Patch 图片进行简要介绍。

9-Patch 图片是一种被特殊处理过的 png 图片，通常用作背景。与普通图片不同的是，当使用 9-Patch 图片作为屏幕或按钮的背景时，如果屏幕尺寸或者按钮大小改变时，图片可自动缩放，达到不失真效果。例如，在制作聊天界面的时候，如果使用 .png、.jpg 等普通格式的图片作为收发消息的背景图，那么图片被拉伸后就会出现变形或失真，而使用 9-Patch 图片就可以完美地解决这个问题。

通过 Android Studio 生成扩展名为 .9.png 的图片的具体步骤如下。

（1）将需要生成扩展名为 .9.png 的图片复制到 Drawable 目录下，然后选中要制作为 9-Patch 图片的图片，单击鼠标右键，在弹出的快捷菜单中选择 Create 9-Patch file 选项，将显示 Save As .9.png（保存 .9 图片）对话框，在该对话框中选择要保存的目录，再修改 .9 图片名称，如图 13.5 所示。单击 OK 按钮后，Drawable 目录结构的效果如图 13.6 所示。

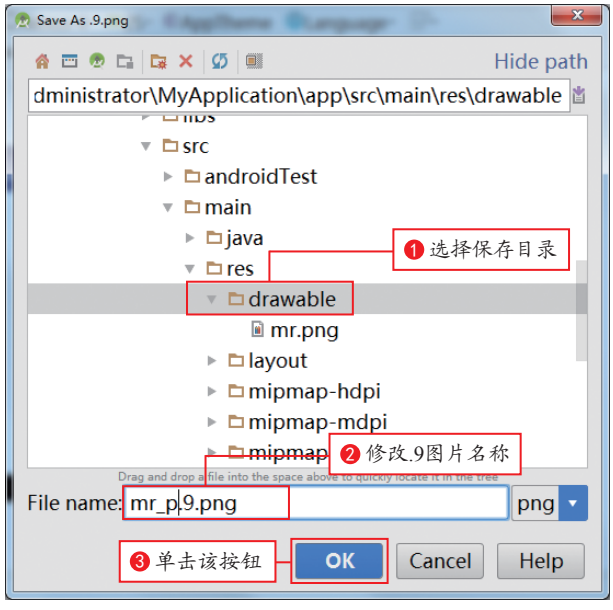


图 13.5　选择保存 .9 图片目录

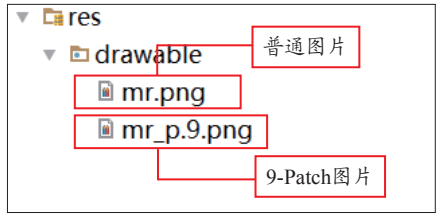


图 13.6　9-Patch 图片与普通图片

注意 生成的9-Patch图片名称不能与普通图片名称相同，否则使用后程序将出错。

（2）双击打开已经生成的 9-Patch 图片，将显示如图 13.7 所示的界面。

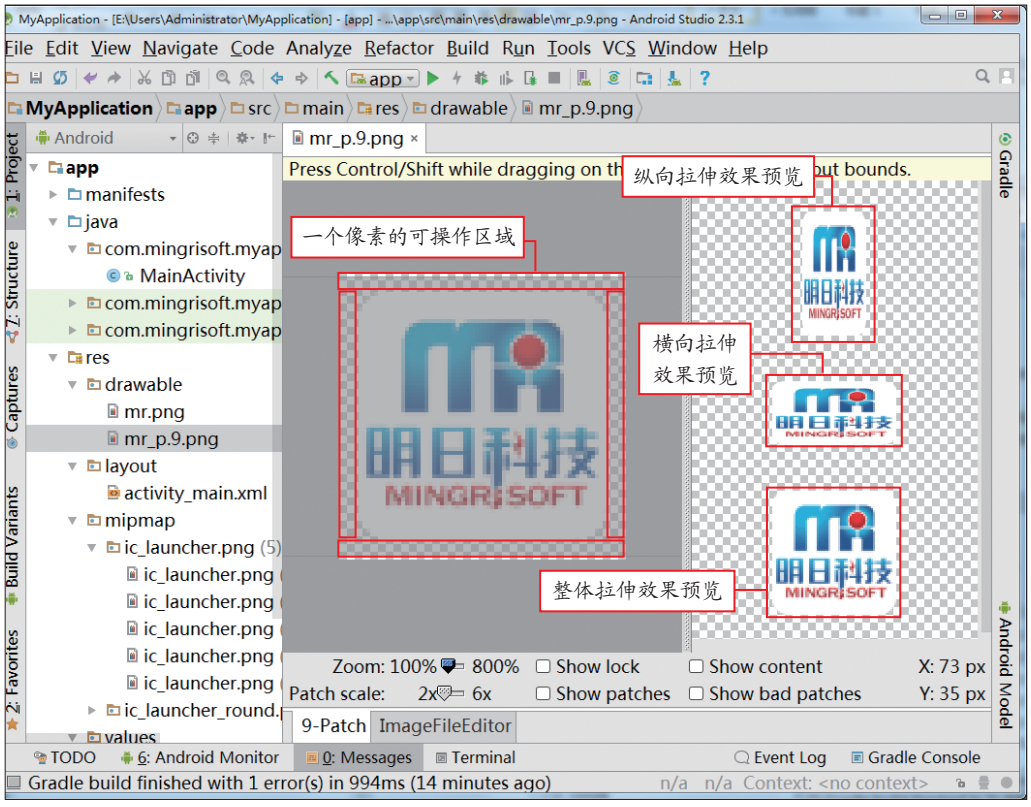


图 13.7　默认图片的拉伸效果

说明　在图片的四周多了一圈一个像素的可操作区域，在该可操作区域上单击，可以绘制一个像素的黑线，水平方向黑线与垂直方向黑线的交集为可缩放区域，在已经绘制的黑线上单击鼠标右键（或者按下Shift键后单击），可以清除已经绘制的内容。

（3）在打开的图片上定义如图 13.8 所示的可缩放区域和内容显示区域。

图 13.8　定义 9-Patch 图片

此时该图片就可以作为图片资源使用。在模拟器中使用 9-Patch 图片和普通 PNG 图像作为按钮背景的对比效果，如图 13.9 所示。

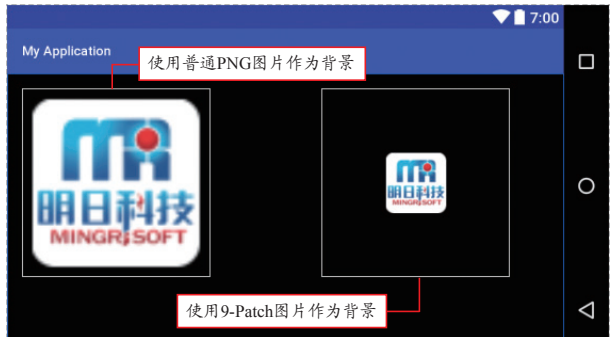


图 13.9　普通 PNG 图片与 9-Patch 图片的对比

在了解了如何制作 9-Patch 图片后，下面来介绍如何使用图片资源。在使用图片资源时，首先将准备好的图片放置在 res\drawable-xxx 目录中，然后就可以在 Java 文件或 XML 文件中访问该资源。在 Java 代码中，可以通过下面的语法格式访问图片。

```
[<package>.]R.drawable.<文件名>
```

注意 Android中不允许图片资源的文件名中出现大写字母，且不能以数字开头。

例如，在 MainActivity 中，通过图片资源为 ImageView 组件设置要显示的图片，可以使用下面的代码：

```
01  ImageView iv=(ImageView)findViewById(R.id.imageView1);
02  iv.setImageResource(R.drawable.head);
```

在 XML 文件中，可以通过下面的语法访问图片资源：

```
@[<package>:]drawable/文件名
```

例如，在定义 ImageView 组件时，通过图片资源为其指定 android:src 属性，也就是设置要显示的图片，具体代码如下：

```
01  <ImageView
02      android:id="@+id/imageView1"
03      android:layout_width="wrap_content"
04      android:layout_height="wrap_content"
05      android:src="@drawable/head" />
```

说明 在Android应用中，使用9-Patch图片时不需要加扩展名.9.png。例如，要在XML文件中使用一个名称为mrbiao.9.png的9-Patch图片，可以使用@drawable/mrbiao。

13.2.2 StateListDrawable 资源

StateListDrawable 资源是定义在 XML 文件中的 Drawable 对象，能根据状态来呈现不同的图像。例如，一个 Button 组件存在多种不同的状态（pressed、enabled 或 focused 等），使用 StateListDrawable

资源可以为按钮的每个状态提供不同的按钮图片。StateListDrawable 资源在实际开发中用处很多，例如，今日头条中选择登录方式的按钮，效果如图 13.10 所示。

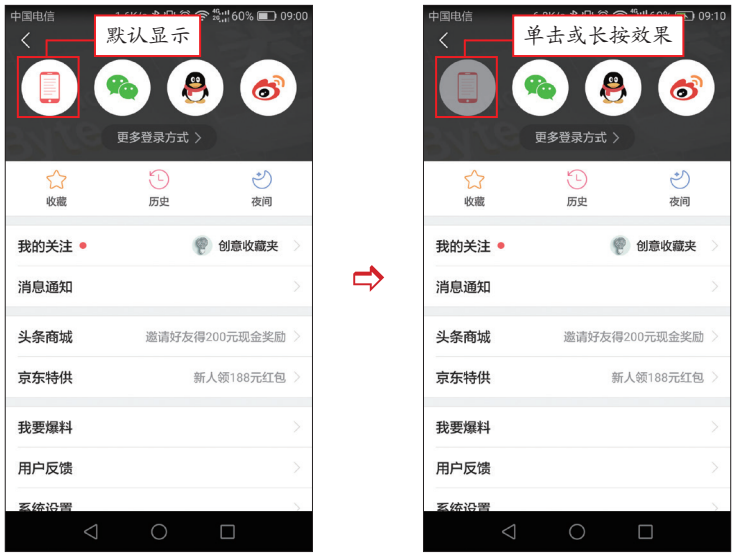


图 13.10　今日头条选择登录方式的按钮

StateListDrawable 资源文件同图片资源一样，也是放在 res\drawable 目录中。StateListDrawable 资源文件的根元素为 <selector></selector>，在该元素中可以包括多个 <item></item> 元素。每个 Item 元素可以设置以下两个属性。

◆ android:color 或 android:drawable：用于指定颜色或 Drawable 资源。

◆ android:state_xxx：用于指定一个特定的状态，常用的状态属性如表 13.2 所示。

表 13.2　StateListDrawable 支持的常用状态属性

状 态 属 性	描　　述
android:state_active	表示是否处于激活状态，属性值为 true 或 false
android:state_checked	表示是否处于选中状态，属性值为 true 或 false
android:state_enabled	表示是否处于可用状态，属性值为 true 或 false
android:state_first	表示是否处于开始状态，属性值为 true 或 false
android:state_focused	表示是否处于获得焦点状态，属性值为 true 或 false
android:state_last	表示是否处于结束状态，属性值为 true 或 false
android:state_middle	表示是否处于中间状态，属性值为 true 或 false
android:state_pressed	表示是否处于被按下状态，属性值为 true 或 false
android:state_selected	表示是否处于被选择状态，属性值为 true 或 false
android:state_window_focused	表示窗口是否已经得到焦点状态，属性值为 true 或 false

例如，创建一个根据编辑框是否获得焦点来改变文本框内文字颜色的 StateListDrawable 资源，名称为 edittext_focused.xml，可以使用下面的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:color="#f60" android:state_focused="true"/>
    <item android:color="#0a0" android:state_focused="false"/>
</selector>
```

创建一个 StateListDrawable 资源后，可以将该文件放置在 res\drawable 目录下，然后在相应的组件中使用该资源即可。例如，要在编辑框中使用名称为 edittext_focused.xml 的 StateListDrawable 资源，可以使用下面的代码：

```
<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="@drawable/edittext_focused"
    android:text="请输入文字" />
```

例 13.3 模拟微信登录

在 Android Studio 中创建 Module，名称为“Drawable Resource”。实现本实例的具体步骤如下：

（1）在 Android Studio 中，将已经准备好的 green_mint.png 和 green.png 图片复制在 drawable 目录下并制作成 9-Patch 图片。最终完成后的图片如图 13.11 所示。

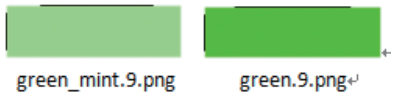


图 13.11 完成后的图片

（2）在 res/rawable 目录中创建一个名称为 button_enable.xml 的 StateListDrawable 资源文件，在该文件中，指定 android:state_enabled 属性为 true 时使用的背景图片（green.9.png）和 android:state_enabled 属性为 false 时使用的背景图片（green_mint.9.png）。button_enable.xml 文件的具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/green" android:state_enabled="true"/>
    <item android:drawable="@drawable/green_mint" android:state_enabled="false"/>
</selector>
```

（3）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml。首先将默认添加的布局管理器修改为垂直线性布局管理器，并且添加一个 ImageView 组件（用于保存头像）、一个 TextView 组件（用于显示账号）、一个 EditText 组件（用于填写密码）、一个 Button 组件（用于登录按钮）和两个 TextView 组件（用于填写文字信息）。其中，登录按钮的背景需要设置为 StateListDrawable 资源（用于让按钮的背景图片随按钮状态而动态改变），并且让该按钮默认为不可用状态。关键代码如下：

```
<Button
    android:id="@+id/btn_login"
```

```
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:background="@drawable/button_enable"
    android:text="@string/text_login"
    android:enabled="false"
    android:textColor="@color/button_text_white"/>
```

（4）打开默认创建的 MainActivity 类，在 onCreate() 方法中，首先获取密码编辑框和登录按钮，并且使用 final 关键字修饰，然后为密码编辑框设置文字改变监听器，用于监视密码编辑框的状态变化。在重写的 onTextChanged() 方法中判断密码编辑框是否输入文字，当输入文字时登录按钮设置为可用状态，无文字输入时登录按钮设置为不可用状态。具体代码如下：

```
//获取密码编辑框
final EditText editText = (EditText) findViewById(R.id.editText);
//获取登录按钮
final Button button = (Button) findViewById(R.id.btn_login);
//为编辑框设置监听事件
editText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        if (editText.length() > 0) {            //判断编辑框内输入文字时
            button.setEnabled(true);            //登录按钮为可用状态
        } else {                                //编辑框内没有文字时
            button.setEnabled(false);           //登录按钮为不可用状态
        }
    }
    @Override
    public void afterTextChanged(Editable s) {
    }
});
```

（5）打开 AndroidManifest.xml 文件，将其中的 <application> 标记的 android:theme 属性值“@style/AppTheme”修改为“@style/Theme.AppCompat.Light.DarkActionBar”，修改后的 android:theme 属性的代码如下：

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

（6）运行本实例，将显示如图 13.12 所示的登录界面，此界面中尚未输入密码。输入密码后将显示如图 13.13 所示的效果。

说明 头像图片采用的是普通PNG图片，设置固定尺寸不可拉伸，而登录按钮背景图片采用StateList-Drawable资源，可以根据按钮的状态自动改变。

图 13.12　未输入密码时

图 13.13　输入密码后

13.2.3 ShapeDrawable 资源

ShapeDrawable 用于绘制一些基本的几何图形，例如，矩形、圆形、线条和环形，ShapeDrawable 资源文件的根元素为 <shape></shape>，在该元素中可以设置以下 4 种属性：

- ◆ rectangle：绘制矩形，这是默认的形状。
- ◆ oval：绘制圆形。
- ◆ line：绘制线条。
- ◆ ring：绘制环形。

ShapeDrawable 资源的语法格式如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!--选择绘制图形-->
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
<!--设置几何图形4个角的弧度-->
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
<!--设置几何图形4个角的弧度-->
    设置渐变色填充-->
    <gradient
        android:angle="integer"
        android:centerX="integer"
        android:centerY="integer"
        android:centerColor="integer"
        android:endColor="color"
```

```
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:useLevel=["true" | "false"] />
<!--设置几何图形4个角的弧度-->
    设置几何图形内边距-->
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer" />
<!--设置几何图形4个角的弧度-->
    设置几何图形大小-->
    <size
        android:width="integer"
        android:height="integer" />
<!--设置几何图形4个角的弧度-->
    设置单颜色填充-->
    <solid
        android:color="color" />
<!--设置几何图形边框-->
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
</shape>
```

ShapeDrawable 资源常用的 XML 属性如表 13.3 所示。

表 13.3　ShapeDrawable 资源常用的 XML 属性

渐变类 XML 属性	描　　述
android:angle	整数类型，渐变角度。0 表示从左到右，90 是从下到上
android:centerX	浮点类型，梯度中心的相对 X 位置（0 - 1.0）
android:centerY	浮点类型，梯度中心的相对 Y 位置（0 - 1.0）
android:centerColor	开始和结束颜色之间的可选颜色
android:startColor	起始颜色
android:endColor	结尾颜色
android:gradientRadius	梯度的半径。只适用于 android:type=”radial”
android:type	渐变模式的类型。默认值为”linear”线性梯度。”radial”径向梯度。开始颜色是中心颜色。”sweep”一条扫线的渐变
android:useLevel	布尔值，true 为开启 LevelListDrawable

续表

线条类 XML 属性	描　　述
android:width	边框线条的粗细
android:dashGap	虚线中线与线间的空隙，只有设置 android:dashWidth 属性有效
android:dashWidth	虚线的长度只有设置 android:dashGap 属性有效
圆环类 XML 属性	**描　　述**
android:innerRadius	环的内部半径（中间的孔）尺寸值
android:innerRadiusRatio	环的内部部分的半径，以环的宽度的比率表示。例如，如果 android:innerRadiusRatio="5"，则内半径等于环的宽度除以 5。该值被覆盖 android:innerRadius。默认值为 9
android:thickness	环的厚度
android:thicknessRatio	环的厚度，以环的宽度的比率表示。例如，如果 android:thicknessRatio="2"，则厚度等于环的宽度除以 2。该值被覆盖 android:innerRadius。默认值为 3
android:useLevel	如果这是用作“true”为开启 LevelListDrawable。这通常是“假”，否则您的形状可能不会出现

下面通过一个实例演示 ShapeDrawable 资源的具体应用。

例 13.4　4 种不同的按钮背景

在 Android Studio 中创建 Module，名称为“ShapeDrawable”，具体步骤如下：

（1）在 res/drawable 目录当中创建名称为“solidline_drawable”的 XML 资源文件，在该文件中通过 ShapeDrawable 实现 1 个红色实线边框的按钮背景图。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <!--设置填充色-->
    <solid android:color="#CCCCCC" />
    <!--设置实线-->
    <stroke
        android:width="2dp"
        android:color="#FF0000" />
</shape>
```

（2）在 res/drawable 目录当中创建名称为“dottedline_drawable”的 XML 资源文件，在该文件中通过 ShapeDrawable 实现 1 个红色虚线边框的按钮背景图。具体代码如下：

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

```
    android:shape="rectangle">
    <!--设置填充色-->
    <solid android:color="#CCCCCC" />
    <!--设置虚线-->
    <stroke
        android:width="2dp"
        android:color="#FF0000"
        android:dashGap="2dp"
        android:dashWidth="10dp" />
</shape>
```

（3）在 res/drawable 目录当中创建名称为“gradient_drawable”的 XML 资源文件，在该文件中通过 ShapeDrawable 实现 1 个圆角渐变色矩形按钮背景图。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <!--设置圆角-->
    <corners android:radius="10dp" />
    <!--设置渐变色-->
    <gradient
        android:angle="45"
        android:endColor="#66CD00"
        android:startColor="#FF0000" />
</shape>
```

（4）在 res/drawable 目录当中创建名称为“ellipse_drawable”的 XML 资源文件，在该文件中通过 ShapeDrawable 实现 1 个椭圆渐变色按钮背景图。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <!--设置渐变色-->
    <gradient
        android:angle="90"
        android:endColor="#66CD00"
        android:startColor="#FF0000" />
</shape>
```

（5）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并设置布局内组件水平居中显示，然后将默认添加的 TextView 组件删除，最后添加 4 个 Button 组件并为这 4 个按钮设置对应的背景资源。具体代码如下：

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

```
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    tools:context="com.mingrisoft.MainActivity">
    <!--实线背景按钮-->
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="@drawable/solidline_drawable"
        android:padding="10dp"
        android:text="实线边框按钮" />
    <!--虚线背景按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="@drawable/dottedline_drawable"
        android:padding="10dp"
        android:text="虚线边框按钮"
        android:layout_toRightOf="@+id/btn1"
        android:id="@+id/button" />
    <!--圆角矩形渐变背景按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/gradient_drawable"
        android:padding="10dp"
        android:text="矩形渐变按钮"
        android:layout_below="@+id/btn1"
        android:layout_alignLeft="@+id/btn1"
        android:layout_alignStart="@+id/btn1" />
    <!--椭圆背景按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/ellipse_drawable"
        android:padding="10dp"
        android:text="椭圆背景按钮"
        android:layout_below="@+id/button"
        android:layout_alignLeft="@+id/button"
        android:layout_alignStart="@+id/button" />
</RelativeLayout>
```

（6）运行本实例，将显示如图 13.14 所示的界面。

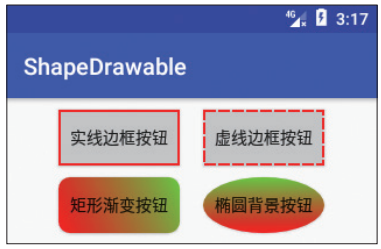


图 13.14　4 种不同的按钮背景

13.2.4 LevelListDrawable 资源

LeveListDrawable 可以管理着一组交替的 drawable 资源。LeveListDrawable 里面的一个 drawable 资源可以根据 setLevel() 方法进行级别的设置，当 level-list 中某项的 Android:maxLevel 数值大于或者等于 setLevel 设置的数值时，该项中的 drawable 资源就会被加载。默认的 level 值为 0，如果没有和 0 匹配的 level，那么将会不显示。

LeveListDrawable 资源的语法格式如下：

```
<?xml version="1.0" encoding="utf-8"?>
<level-list
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@drawable/drawable_resource"
        android:maxLevel="integer"
        android:minLevel="integer" />
</level-list>
```

LeveListDrawable 资源文件的根元素为 <level-list></level-list>，子元素为 <item></item>，在该元素中可以设置以下 3 种属性：

◆ android:drawable：设置 1 个 drawable 资源。

◆ android:maxLevel：设置最大级别。

◆ android:minLevel：设置最小级别。

下面通过一个实例演示 LeveListDrawable 资源的具体应用。

例 13.5　自动切换的图片

在 Android Studio 中创建 Module，名称为“LevelListDrawable”，具体步骤如下：

（1）在 res/drawable 目录当中创建名称为“levellist_drawable”的 XML 资源文件，在该文件中添加 4 个 <item></item> 并加载 4 张图片。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <level-list xmlns:android="http://schemas.android.com/apk/res/android">
03      <!--第一张图片-->
04      <item
05      android:drawable="@drawable/bg1"
06      android:maxLevel="0"/>
```

```
    <!--第二张图片-->
    <item
        android:drawable="@drawable/bg2"
        android:maxLevel="1"/>
    <!--第三张图片-->
    <item
        android:drawable="@drawable/bg3"
        android:maxLevel="2"/>
    <!--第四张图片-->
    <item
        android:drawable="@drawable/bg4"
        android:maxLevel="3"/>
</level-list>
```

（2）修改新建项目的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后将默认添加的 TextView 组件删除，最后添加 1 个 ImageView 组件用于显示自动切换的图片。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示图片-->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"
        android:src="@drawable/levellist_drawable" />
</RelativeLayout>
```

（3）打开主活动 MainActivity.java 文件，定义所需的全局变量。具体代码如下：

```
final int FLAG_MSG = 0x001;       //定义要发送的消息代码
private Message message;          //定义消息对象
private int i = 0;                //用于计算余数
private LevelListDrawable levelListDrawable;       //定义LevelListDrawable
```

（4）在 MainActivity 中创建 android.os.Handler 对象，并重写 handleMessage() 方法，在该方法中实现每 2 秒钟设置 1 次级别值并根据级别值加载相应的图片资源，具体代码如下：

```
Handler handler = new Handler() {  //创建android.os.Handler对象
    @Override
    public void handleMessage(Message msg) {
```

```
04          if (msg.what == FLAG_MSG) {    //如果接收到的是发送的标记消息
05              int a = i % 4;             //利用余数计算当前显示的级别
06          levelListDrawable.setLevel(a);//设置级别
07              i++;
08          }
09          message = handler.obtainMessage(FLAG_MSG);    //获取要发送的消息
10          handler.sendMessageDelayed(message, 2000);    //延迟2秒发送消息
11      }
12  };
```

（5）重写 onStop() 方法，在该方法中实现界面停止后停止 Handler。具体代码如下：

```
01  //界面停止后移除消息,停止Handler
02  @Override
03  protected void onStop() {
04      super.onStop();
05      handler.removeMessages(FLAG_MSG); //移除消息
06  }
```

（6）在 onCreate() 方法中首先获取显示图片的 ImageView 组件，然后获取 LevelListDrawable，最后设置 Message 消息并发送消息。关键代码如下：

```
01  //获取图像组件
02  ImageView imageView = (ImageView) findViewById(R.id.imageView);
03  //获取LevelListDrawable
04  levelListDrawable = (LevelListDrawable) imageView.getDrawable();
05  message=Message.obtain();              //获得消息对象
06  message.what=FLAG_MSG;                 //设置消息代码
07  handler.sendMessage(message);          //发送消息
```

（7）运行本实例，将显示如图 13.15 所示默认显示的图片，2 秒后将自动切换如图 13.16 所示的图片。

图 13.15　默认显示的图片

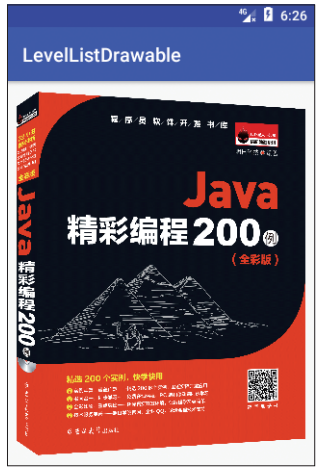


图 13.16　2 秒后自动切换的图片

说明 在级别匹配时以android:maxLevel属性值为优先。如果一个item当中android:minLevel值为1，android:maxLevel值为2；另一个item，android:minLevel值为2，android:maxLevel值为3。如果此时setLevel()方法中设置的值为2，将优先匹配第一个item。

13.2.5 ClipDrawable 资源

ClipDrawable 资源是对一个 Drawable 进行裁剪的操作，可以控制当前 Drawable 的裁剪区域，以及相对于容器的对齐方式，它需要根据 level 的属性值，决定裁剪区域的大小。默认的 level 值为 0，表示全部裁剪掉 Drawable 不可见。当 level 值为 10000 时图片相当于没裁剪，表示完全可见。

ClipDrawable 资源的语法格式如下：

```
<?xml version="1.0" encoding="utf-8"?>
<clip
xmlns:android="http://schemas.android.com/apk/res/android"
android:drawable="@drawable/drawable_resource"
android:clipOrientation=["horizontal"|"vertical"]
android:gravity=["top"|"bottom"|"left"|"right"|
"center_vertical"|"fill_vertical"|"center_horizontal"|
"fill_horizontal"|"center"|"fill"|"clip_vertical"|"clip_horizontal"]/>
```

ClipDrawable 资源文件的根元素为 <clip></clip>，在该元素中可以设置以下 3 种属性：

- android:drawable：设置裁剪的 Drawable 资源。
- android:clipOrientation：设置裁剪方向，horizontal 为水平裁剪、vertical 为垂直裁剪。
- android:gravity：指定在 drawable 中裁剪的位置，属性值如表 13.4 所示。

表 13.4 android:gravity 属性值

属 性 值	描 述
top	将对象放在容器的顶部，而不改变其大小。当 clipOrientation 是 "vertical"，裁剪发生在 drawable 的底部
bottom	将对象放在容器的底部，不改变其大小。当 clipOrientation 是 "vertical"，裁剪发生在 drawable 的顶部
left	将对象放在容器的左侧，不改变其大小。当 clipOrientation 是 "horizontal"，裁剪发生在 drawable 的右边，默认情况下为该值
right	将对象放在容器的右侧，不改变其大小。当 clipOrientation 是 "horizontal"，裁剪发生在 drawable 的左边
center_vertical	将对象放在容器垂直中心，不改变其大小。裁剪的情况和”center”一样
fill_vertical	如果 clipOrientation 是 "vertical"，不发生任何裁剪。（除非 drawable 的 level 是 0，才会不可见，表示全部裁剪完）
center_horizontal	将对象放在容器水平中心，不改变其大小。裁剪的情况和”center”一样
fill_horizontal	如果 clipOrientation 是 "horizontal"，不发生任何剪裁。（除非绘制级别为 0，在这种情况下是不可见的）

续表

属 性 值	描　　述
center	将物体放置在容器中心的垂直和水平轴上，而不改变其尺寸。 如果 clipOrientation 是 "horizontal"，剪切发生在左侧和右侧。当 clipOrientation 是 "vertical"，发生削波的顶部和底部
fill	如果需要，增加对象的水平和垂直尺寸，以便完全填充其容器。没有剪切发生，因为 drawable 填充水平和垂直空间（除非可绘制级别为 0，在这种情况下不可见）
clip_vertical	附加选项，用于按照容器的边来剪切对象的顶部和 / 或底部的内容，剪切基于其纵向对齐设置：顶部对齐时剪切底部；底部对齐时剪切顶部；除此之外剪切顶部和底部
clip_horizontal	附加选项，用于按照容器的边来剪切对象的左侧和 / 或右侧的内容。剪切基于其横向对齐设置：左侧对齐时剪切右侧；右侧对齐时剪切左侧；除此之外剪切左侧和右侧

下面通过一个实例演示 ClipDrawable 资源的具体应用。

例 13.6 自动展开的图片

在 Android Studio 中创建 Module，名称为“ClipDrawable”，具体步骤如下：

（1）在 res/drawable 目录当中创建名称为“clip_drawable”的 XML 资源文件，在该文件中首先设置 Drawable 的裁剪方式为水平裁剪，然后设置需要裁剪的 Drawable 资源，最后设置裁剪位置为 left。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<clip xmlns:android="http://schemas.android.com/apk/res/android"
    android:clipOrientation="horizontal"
    android:drawable="@drawable/bg1"
    android:gravity="left">
</clip>
```

（2）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后将默认添加的 TextView 组件删除，最后添加 1 个 ImageView 组件用于显示自动展开的图片。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示图片-->
    <ImageView
```

```
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"
        android:src="@drawable/clip_drawable" />
</RelativeLayout>
```

（3）打开主活动 MainActivity.java 文件，定义所需的全局变量。具体代码如下：

```
final int FLAG_MSG = 0x001;          //定义要发送的消息代码
private Message message;             //定义消息对象
private ClipDrawable clipDrawable; //定义ClipDrawable资源
```

（4）在 MainActivity 中创建 android.os.Handler 对象，并重写 handleMessage() 方法，在该方法中实现每 0.1 秒钟设置 1 次级别值，具体代码如下：

```
Handler handler = new Handler() {  //创建android.os.Handler对象
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == FLAG_MSG) {//如果接收到的是发送的标记消息
            //判断当前级别是否大于或等于最大值
            if (clipDrawable.getLevel() >= 10000) {
                handler.removeMessages(FLAG_MSG);    //移除消息停止Handler
            } else {
                //在当前级别基础上增加值100
                clipDrawable.setLevel(clipDrawable.getLevel() + 100);
            }
        }
        message = handler.obtainMessage(FLAG_MSG);    //获取要发送的消息
        handler.sendMessageDelayed(message, 100);     //延迟0.1秒发送消息
    }
};
```

（5）在 onCreate() 方法中首先获取显示图片的 ImageView 组件，然后获取 ClipDrawable，最后设置 Message 消息并发送消息。关键代码如下：

```
//获取图像组件
ImageView imageView = (ImageView) findViewById(R.id.imageView);
clipDrawable = (ClipDrawable) imageView.getDrawable();  //获取ClipDrawable资源
message = Message.obtain();                  //获得消息对象
message.what = FLAG_MSG;                     //设置消息代码
handler.sendMessage(message);                //发送消息
```

（6）运行本实例，图片每 0.1 秒将自动展开图片的一部分，效果如图 13.17 所示，图片完全展开后将显示如图 13.18 所示的效果。

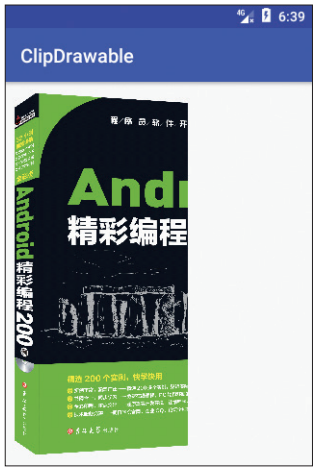


图 13.17　自动展开的图片

图 13.18　完全展开的图片

13.2.6 LayerDrawable 资源

LayerDrawable 表示一种层次化的 Drawable 数组，系统将会按照这些 Drawable 数组的顺序来绘制它们，索引最大的 Drawable 对象将会被绘制在最上面。LayerDrawable 资源文件也是放在 res\drawable-xxx 目录中。LayerDrawable 资源文件的根元素为 <layer-list></layer-list>，在该元素中可以包含多个 <item></item> 子元素。每个 Item 元素可以设置以下属性：

- ◆ android:drawable：设置 LayerDrawable 元素之一的 Drawable 对象。
- ◆ android:id：设置 Drawable 对象的标识。
- ◆ android:top：设置 Drawable 对象绘制在指定组件顶部的指定位置。
- ◆ android:bottom：设置 Drawable 对象绘制在指定组件底部的指定位置。
- ◆ android:left：设置 Drawable 对象绘制在指定组件左侧的指定位置。
- ◆ android:right：设置 Drawable 对象绘制在指定组件右侧的指定位置。

例如，指定一个 Drawable 元素可以使用下面的代码：

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <!--指定Drawable元素-->
    <item
        android:id="@+id/seekBar"
        android:drawable="@drawable/ic">
    </item>
</layer-list>
```

下面通过一个实例演示 LayerDrawable 资源的具体应用。

例 13.7　改变拖动条外观与绘制叠在一起的 Android 图标

在 Android Studio 中创建 Module，名称为“LayerDrawable”，具体步骤如下：

（1）在 res/drawable 目录当中创建名称为“seekbar_drawable”的 XML 资源文件，在该文件中通过 LayerDrawable 实现拖动条拖动前与拖动后的背景图。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 设置拖动前的背景图 -->
    <item android:id="@android:id/background"
        android:drawable="@drawable/ic" />
    <!-- 设置拖动后的背景图-->
    <item android:id="@android:id/progress"
        android:drawable="@drawable/ic1" />
</layer-list>
```

（2）在 res/drawable 目录当中创建名称为“imageview_drawable”的 XML 资源文件，在该文件中通过 LayerDrawable 实现绘制叠在一起的 Android 图标。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <!--第一个Android图标-->
    <item>
        <bitmap
            android:gravity="left"
            android:src="@drawable/ic" />
    </item>
    <!--第二个Android图标-->
    <item
        android:left="10dp" android:top="20dp">
        <bitmap
            android:gravity="left"
            android:src="@drawable/ic"/>
    </item>
    <!--第三个Android图标-->
    <item
        android:left="20dp" android:top="30dp">
        <bitmap
            android:gravity="left"
            android:src="@drawable/ic"/>
    </item>
</layer-list>
```

（3）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，并将默认添加的 TextView 组件删除，然后先添加 1 个 SeekBar 组件用于显示拖动条，再添加 1 个 ImageView 组件用于显示绘制的 Android 图标。具体代码如下：

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

```
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
06      android:layout_height="match_parent"
07      android:orientation="vertical"
08      tools:context="com.mingrisoft.MainActivity">
09      <!--拖动条组件-->
10      <SeekBar
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:max="100"
14          android:progressDrawable="@drawable/seekbar_drawable"
15          />
16      <!--图像组件-->
17      <ImageView
18          android:layout_width="wrap_content"
19          android:layout_height="wrap_content"
20          android:src="@drawable/imageview_drawable"/>
21  </LinearLayout>
```

（4）运行本实例，效果如图 13.19 所示。

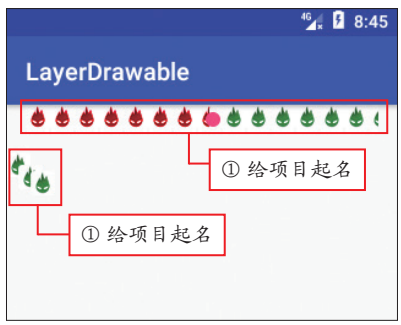


图 13.19　改变拖动条外观与绘制叠在一起的 Android 图标

13.2.7 mipmap 资源

在使用 Android Studio 创建 Android 项目时，res 目录下有一个 mipmap 文件夹，该目录在 1.3.2 小节中已经详细介绍过，本节将主要讲述 mipmap 资源的语法格式。

◆ 在 Java 代码中，可以通过下面的语法格式访问 mipmap 资源：

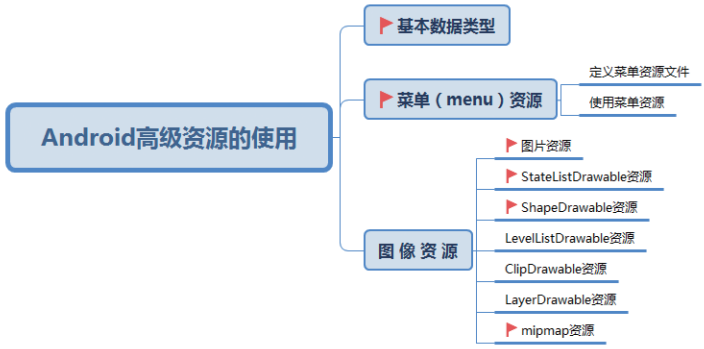
```
[<package>.]R.mipmap.<文件名>
```

◆ 在 XML 文件中，可以通过下面的语法访问 mipmap 资源：

```
@[<package>:]mipmap/文件名
```

13.3 知 识 回 顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 14 章

Action Bar 的使用

Action Bar，即动作栏，它是一种窗体特性，它用于标识应用和用户位置，并提供用户动作和导航模式。开发人员在大多数需要显示用户动作或全局导航的 Activity 中使用 Action Bar，Action Bar 为用户在跨应用程序时提供了连续的界面，Android 系统也能让其外观适应不同屏幕的设置。本章将对 Android 中的 Action Bar 进行详细讲解。

14.1 Action Bar 概述

Action Bar 是用来代替显示标题和应用图标的传统标题栏的。图 14.1 就是一个 Action Bar，左侧显示了应用的图标和 Activity 标题，右侧显示了一些主要操作以及 overflow 菜单。

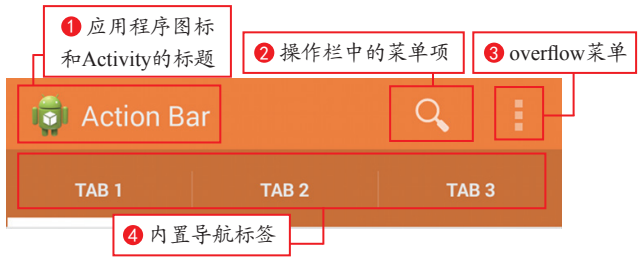


图 14.1　Action Bar 示例

说明　（1）如果当前应用不在顶层界面，那么在应用程序图标的左侧通常会放置一个向左的箭头，表示“向上”按钮，用于返回到上一界面。

（2）overflow菜单即溢出菜单，通常位于Action Bar的右侧，屏幕的右上角，以三个点来表示，在该菜单中可以定义一些菜单项，这些菜单项是以下拉菜单形式显示的。

Action Bar 的主要用途如下：

◆ 提供一个用来标识应用程序的图标和标题。

这个空间的左边是应用程序的图标或 Logo，以及 Activity 的标题，如图 14.1 中的第 1 部分所示。

◆显示选项菜单的菜单项。

开发人员通过把菜单项直接放到 Action Bar 中，从而为用户提供直接访问，如图 14.1 中的第 2 部分所示。

◆提供基于下拉的导航方式。

与 Action Item 无关的菜单项可以放到 overflow 菜单中，通过单击设备的菜单键或者 Action Bar 中的 overflow 菜单按钮来访问，如图 14.1 中的第 3 部分所示。

◆提供基于 Tab 的导航方式，可以在多个 Fragment 之间进行切换。

Action Bar 提供了在多个 Fragment 之间切换的内置导航标签，如图 14.1 中的第 4 部分所示。

14.2 Action Bar 基本应用

从 Android 3.0（API 11）开始，Activity 中就默认包含 Action Bar 组件。如果想要使用 Action Bar 的全部特性，必须指定 minSdkVersion 的版本为 11 或 11 以上，即设置 android:minSdkVersion 的值为 11 或 11 以上。例如，在 Android Studio 中，默认创建的 app 将包含如图 14.2 所示的 Action Bar。

图 14.2 Android Studio 中默认创建应用的 Action Bar

说明 如果不想在Activity上包含Action Bar，可以在AndroidManifest.xml中将android:theme属性设置为后缀带“.NoActionBar”，如“@style/Theme.AppCompat.NoActionBar”。

14.2.1 显示和隐藏 Action Bar

在 Java 代码中，可以控制已经添加的 Action Bar 的显示和隐藏。如果希望在某个 Activity 中不使用 Action Bar，则可以在 Java 代码中调用 ActionBar 对象的 hide() 方法来隐藏它，另外，如果想要显示已经隐藏的 Action Bar，可以调用 ActionBar 对象的 show() 方法来显示它。例如，使用 hide() 方法隐藏 Action Bar 可以使用下面的代码：

```
01  ActionBar actionbar = getActionBar();
02  actionbar.hide();
```

说明 在获取Action Bar对象时，如果当前的Activity继承自V7包中的Activity时，需要通过getSupportActionBar()方法来获取Action Bar对象。当隐藏Action Bar时，系统会将Activity的内容填充至整个空间。在实际项目中，推荐使用Java代码的方式来控制Action Bar的显示和隐藏。

下面通过一个实例来演示 Action Bar 的应用。

例 14.1 Action Bar 的显示与隐藏

在 Android Studio 中创建一个 Module，名称为“Display And Hide”。实现本实例的具体步骤如下：

（1）修改 res/layout 包中的 activity_main.xml 文件，首先将默认添加的布局管理器修改为水平线性布局管理器，然后删除默认添加的 TextView 组件，最后添加两个按钮组件，分别设置 id 为 actionBar_show 与 actionBar_hide，用于显示与隐藏 Action Bar。

（2）打开默认创建的 MainActivity 类，然后定义显示与隐藏按钮，再定义一个 V7 包下的 ActionBar。

```
01  Button action_show, action_hide;                    //定义显示与隐藏按钮
02  android.support.v7.app.ActionBar actionBar;          //定义V7包下的ActionBar
```

说明 Android为了保证高版本SDK开发时的向下兼容性，提供了Android Support Library package系列的包，从而实现在低版本上，可以使用高版本的有些特性。其中，V7包是为了兼容2.1（API 7）及以上版本而设计的。

（3）重写 onCreate() 方法，在该方法中首先获取 Action Bar，然后获得两个按钮组件并为其设置事件监听器，再创建一个新的监听事件，最后通过 id 判断选择了哪个按钮，并设置显示与隐藏 Action Bar。其代码如下：

```
01  @Override
02  protected void onCreate(Bundle savedInstanceState) {
03      super.onCreate(savedInstanceState);
04      setContentView(R.layout.activity_main);
05      actionBar = getSupportActionBar(); //获取ActionBar
06      action_show = (Button) findViewById(R.id.actionBar_show); //获取显示按钮
07      action_hide = (Button) findViewById(R.id.actionBar_hide); //获取隐藏按钮
08      action_show.setOnClickListener(l); //为显示按钮设置监听事件
09      action_hide.setOnClickListener(l); //为隐藏按钮设置监听事件
10  }
11  View.OnClickListener l = new View.OnClickListener() { //创建一个新的监听事件l
12      @Override
13      public void onClick(View v) {
14          switch (v.getId()) {            //根据选择按钮的id判断
15              case R.id.actionBar_show:
16                  actionBar.show();       //显示ActionBar
17                  break;
18              case R.id.actionBar_hide:
19                  actionBar.hide();       //隐藏ActionBar
20                  break;
21          }
22      }
23  };
```

（4）运行本实例，效果如图 14.3 所示。单击隐藏 ActionBar 按钮将显示如图 14.4 所示的效果。

图 14.3　显示 Action Bar

图 14.4　隐藏 Action Bar

14.2.2 添加 Action Item 选项

一个 Action Item（动作项）实际上就是一个直接在 Action Bar 上显示的菜单项，它可以包含图标或文本标题。如果一个菜单项没有作为 Action Item 显示在 Action Bar 上，那么系统会将其放置在 overflow 菜单中。

当 Activity 第一次启动时，系统会通过调用 onCreateOptionsMenu() 方法来将菜单项放置于 Action Bar 或者 overflow 菜单中。在该方法中，可以使用 menu 文件夹中定义的资源文件来生成菜单。例如，加载名称为 actions.xml 的菜单文件，可以使用下面的代码：

```
01  public boolean onCreateOptionsMenu(Menu menu) {
02      MenuInflater inflater = getMenuInflater();
03      inflater.inflate(R.menu.actions, menu);
04      return true;
05  }
```

在定义菜单资源文件时，同 12.1.1 节介绍的方法类似，也是使用 <menu> 标记作为根元素，然后通过 <item> 标记指定菜单项。通常需要为 <item> 标记指定以下属性。

◆ android:title 属性

该属性用来设置 Action Item 的标题。

◆ android:icon 属性

该属性用来设置 Action Item 的图标。它是可选的，但是通常会定义该属性，即为 Action Item 提供图标。

◆ android:showAsAction 属性

该属性用来设置是否将该菜单项显示在 Action Bar 上。它主要有以下 4 个属性值：

（1）oom：当 Action Bar 有可用空间时，就会显示该 Action Item。如果没有足够的空间，就会在 overflow 菜单中显示。

ays：总是将该菜单项显示在 Action Bar 上。

er：不将该菜单项显示在 Action Bar 上。

（4）hText：将该菜单项显示在 Action Bar 上，并显示该菜单项的文本。

例 14.2 添加 Action Item 选项

在 Android Studio 中创建 Module，名称为“Action Item”，实现本实例的具体步骤如下：

（1）修改 res/layout 目录中的 activity_main.xml 文件，将默认添加的布局管理器修改为相对布局管理器，然后删除默认添加的 TextView 组件，再为布局管理器添加背景图片。

（2）在 res 目录中新建 menu 子文件夹，然后在 menu 文件夹中新建名称为 menu.xml 的菜单资源文件，再在该菜单资源文件中添加 4 个菜单项，并设置前两个显示在 Action Bar 上，后两个只有在 Action Bar 上空间充足时才显示到 Action Bar 上，代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <menu xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto">
04      <item
05          android:id="@+id/search"
```

```
        android:icon="@mipmap/search"
        android:title="@string/search"
        app:showAsAction="always"></item>
    <item
        android:id="@+id/bell"
        android:icon="@mipmap/bell"
        android:title="@string/bell"
        app:showAsAction="always"></item>
    <item
        android:id="@+id/settings"
        android:title="@string/settings"
        app:showAsAction="ifRoom"></item>
    <item
        android:id="@+id/about"
        android:title="@string/about"
        app:showAsAction="never"></item>
</menu>
```

（3）打开默认创建的 MainActivity 类，然后重写 onCreateOptionsMenu() 方法，并且在该方法中解析步骤（2）中创建的菜单文件，实现创建菜单，关键代码如下：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater=getMenuInflater();     //实例化一个MenuInflater对象
    inflater.inflate(R.menu.menu,menu);          //解析菜单文件
    return super.onCreateOptionsMenu(menu);
}
```

（4）运行本实例，将显示如图 14.5 所示的界面。

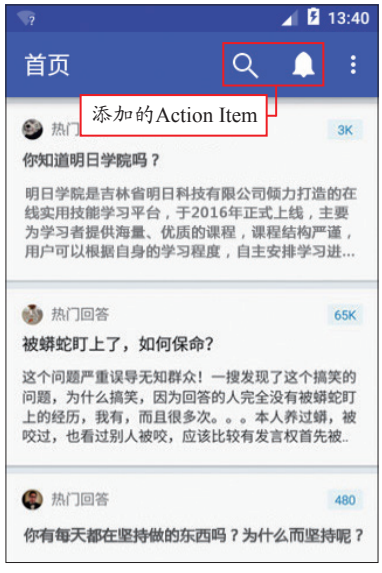


图 14.5　添加 Action Item 选项

14.2.3 添加 Action View

Action View（动作视图）是出现在 Action Bar 中，用于替换 Action Item 按钮并显示在 Action Bar 上的一种可视组件。例如，可以在 Action Bar 中添加如图 14.6 所示的“搜索”Action Item。

图 14.6 在 Action Bar 中添加“搜索”Action Item

为了在 Action Bar 上添加 Action View，可以使用如下两种方式：

◆ 义 Action Item 时使用 android:actionViewClass 属性指定 Action View 的实现类。

◆ 义 Action Item 时使用 android:actionLayout 属性指定 Action View 对应的视图资源。

下面通过一个例子来演示添加 ActionView 的两种方法。

例 14.3 模拟支付宝朋友页面的 Action Bar

在 Android Studio 中创建 Module，名称为“Action View”，实现本实例的具体步骤如下：

（1）修改 res/layout 目录中的 activity_main.xml 文件，将默认添加的布局管理器修改为相对布局管理器，然后删除默认添加的 TextView 组件，并为布局管理器添加背景图片。

（2）分别创建两个名称为 img_message.xml 和 img_add.xml 布局文件，用于显示代表通讯录和添加的图标。

（3）在 res 文件夹中新建 menu 子文件夹，然后在 menu 文件夹中新建 menu.xml 文件，再增加三个菜单项，并设置其属性，为第一个设置 actionViewClass 属性为”android.support.v7.widget.SearchView”，用于显示搜索，第二个设置 app:actionLayout 属性为”@layout/ img_message”，用于显示通讯录图标，第三个设置 app:actionLayout 属性为” @layout/img_add”，用于显示添加图标，代码如下：

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

```
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/search"
        android:title="@string/search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always"></item>
    <item
        android:id="@+id/img1"
        android:title="@string/img1"
        app:actionLayout="@layout/img_message"
        app:showAsAction="always"></item>
    <item
        android:id="@+id/img2"
        android:title="@string/img2"
        app:actionLayout="@layout/img_add"
        app:showAsAction="always"></item>
</menu>
```

（4）修改默认创建的 MainActivity 类，首先在 onCreate() 方法中隐藏 ActionBar 中显示的标题，然后重写 onCreateOptionsMenu() 方法，并在该方法中解析菜单资源文件，实现创建菜单。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      //隐藏ActionBar中显示的标题
      getSupportActionBar().setDisplayShowTitleEnabled(false);
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater=getMenuInflater();     //实例化一个MenuInflater对象
      inflater.inflate(R.menu.menu,menu);          //解析菜单文件
      return super.onCreateOptionsMenu(menu);
  }
}
```

（5）打开 AndroidManifest.xml 文件，修改 <application> 标记的 android:theme 属性值，修改后的 android:theme 属性的代码如下：

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

（6）运行本实例，效果如图 14.7 所示。

图 14.7　模拟支付宝 app 的 Action Bar

14.2.4 Action Bar 与 Tab

Action Bar 提供基于选项卡模式的导航方式，在实际项目中为了更好地展现 Tab 导航效果，Action Bar 通常会与 Fragment 结合使用。它运行在一个 Activity 中，可以在不同的 Fragment 之间进行切换。如图 14.8 所示。

图 14.8　基于选项卡模式的导航方式

同时，针对用户选择选项卡事件，还专门定义了一个事件监听器。在 ActionBar 类中，定义的与 Tab 相关的常用方法如表 14.1 所示。

表 14.1　ActionBar 类中与 Tab 相关的常用方法

方　法　名	描　　述	语 法 格 式
addTab()	为 Action Bar 增加选项卡	public abstract void addTab(ActionBar.Tab tab)
getSelectedTab()	获得当前选择的选项卡	public abstract ActionBar.Tab getSelectedTab()
getTabAt()	获得指定索引位置的选项卡	public abstract ActionBar.Tab getTabAt(int index)
getTabCount()	获得选项卡的个数	public abstract int getTabCount()
newTab()	获得一个选项卡，但是它并没有被添加到 Action Bar，需要调用 addTab 方法添加	public abstract ActionBar.Tab newTab()
removeAllTabs()	移除全部选项卡	public abstract void removeAllTabs()
removeTab()	移除指定选项卡	public abstract void removeTab(ActionBar.Tab tab)
removeTabAt()	移除指定位置的选项卡	public abstract void removeTabAt(int position)
selectTab()	设置选项卡被选中	public abstract void selectTab(ActionBar.Tab tab)
onTabReselected()	处理选项卡再次被选中事件	public abstract void onTabReselected (ActionBar.Tab tab, FragmentTransaction ft)
onTabSelected()	处理选项卡选中事件	public abstract void onTabSelected (ActionBar.Tab tab, FragmentTransaction ft)
onTabUnselected()	处理选项卡退出选中状态	public abstract void onTabUnselected (ActionBar.Tab tab, FragmentTransaction ft)

说明　表14.1中介绍的方法有的具有多种重载形式，详细介绍请参考API文档。

使用 Action Bar 实现 Tab 导航，大致可以通过以下两个步骤实现。

（1）调用 Action Bar 的 setNavigationMode(ActionBar.NAVIGATION_MODE_TABS) 方法设置使用 Tab 导航方式。

（2）调用 Action Bar 的 addTab() 方法添加多个 Tab 标签，并为每个 Tab 标签添加事件监听器。

下面通过一个实例来演示 Action Bar 与 Tab 的应用。

例 14.4　模拟有道词典 App 的 Action Bar

在 Android Studio 中创建 Module，名称为“ActionBar And Tab”，实现本实例的具体步骤如下：

（1）在 res/layout 目录下创建名称为 fragment_1.xml 布局文件，用于实现第一个标签页界面，代码如下：

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

```
        android:scaleType="fitXY"
        android:src="@mipmap/fragment_1"
        />
</LinearLayout>
```

说明 按照以上步骤依次创建5个布局文件，用于显示5个不同页面。

（2）创建 MyTabListener 类，让其继承 android.support.v4.app.Fragment 并实现 android.support.v7.app.ActionBar.TabListener 接口，然后在该类中处理标签页相关事件，代码如下：

```
public class MyTabListener<T extends android.support.v4.app.Fragment> implements
        android.support.v7.app.ActionBar.TabListener {
    private android.support.v4.app.Fragment fragment;       //定义Fragment
    private final Activity activity;                        //定义Activity
    private final Class aClass;                             //定义Class
    public MyTabListener(Activity activity, Class aClass) { //添加构造函数
        this.activity = activity;
        this.aClass = aClass;
    }
    @Override
    public void onTabSelected(android.support.v7.app.ActionBar.Tab tab,
                              android.support.v4.app.FragmentTransaction ft) {
        //判断碎片是否初始化
        if (fragment == null) {                             //如果没有初始化，将其初始化
            fragment =
              android.support.v4.app.Fragment.instantiate(activity, aClass.getName());
            ft.add(android.R.id.content, fragment, null);
        }
        ft.attach(fragment);                                //显示新画面
    }
    @Override
    public void onTabUnselected(android.support.v7.app.ActionBar.Tab tab,
                                android.support.v4.app.FragmentTransaction ft) {
        if (fragment != null) {
            ft.detach(fragment);                            //删除旧画面
        }
    }
    @Override
    public void onTabReselected(android.support.v7.app.ActionBar.Tab tab,
                                android.support.v4.app.FragmentTransaction ft) {
    }
}
```

（3）创建 Fragment1 类，让其继承 Fragment，并且在该类中重写 onCreateView() 方法，用于加载布局文件页面，其代码如下：

```
public class Fragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_1,null);    //加载布局页面
```

```
06      }
07  }
```

说明 按照以上步骤依次创建5个Fragment类，分别加载5个不同布局文件。

（4）打开默认创建的 MainActivity 类，然后在 onCreate() 方法中获取 ActionBar，再设置 ActionBar 为 Tab 导航方式，并且隐藏标题栏，最后将标签页添加到 ActionBar 中，代码如下：

```
01  public class MainActivity extends AppCompatActivity {
02      @Override
03              protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          ActionBar actionBar=getSupportActionBar(); //获取ActionBar
07          //设置ActionBar为选项卡模式
08          actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
09          actionBar.setDisplayOptions(0, ActionBar.DISPLAY_SHOW_TITLE);  //隐藏标题栏
10          actionBar.addTab(actionBar.newTab().setText("词典").//将标签页添加ActionBar中
11                  setTabListener(new MyTabListener(this, Fragment1.class)));
12          actionBar.addTab(actionBar.newTab().setText("百科").//将标签页添加ActionBar中
13                  setTabListener(new MyTabListener(this, Fragment2.class)));
14          actionBar.addTab(actionBar.newTab().setText("翻译").////将标签页添加ActionBar中
15                  setTabListener(new MyTabListener(this, Fragment3.class)));
16          actionBar.addTab(actionBar.newTab().setText("发现").//将标签页添加ActionBar中
17                  setTabListener(new MyTabListener(this, Fragment4.class)));
18          actionBar.addTab(actionBar.newTab().setText("我的").//将标签页添加ActionBar中
19                  setTabListener(new MyTabListener(this, Fragment5.class)));
20      }
21  }
```

（5）运行本实例，效果如图 14.9 所示。

图 14.9　Tab 在 Action Bar 中的使用

14.3 实现层级式导航

在 Android 应用中，可以使用后退键来实现应用内导航功能。这种方式又称为临时性导航，它只能返回到上一次的用户界面。而有时需要实现层级式导航，即逐级向上在应用内导航。在 Android 中，利用 Action Bar 上的应用图标可以实现层级式导航，另外，利用应用图标也可以实现直接退至应用的主界面。这一功能，在现在的手机应用中非常实用。下面将详细介绍如何实现层级式导航。

14.3.1 启用程序图标导航

通常情况下，应用程序图标启用向上导航功能时，就会在左侧显示一个向左指向的图标，如图 14.10 所示。

图 14.10 向上导航功能图标

这是通过启用应用图标向上导航按钮的功能实现的。要实现该功能，需要调用以下方法设置 Activity 或 Fragment 的 DisplayHomeAsUpEnabled 属性为 true。

例如，要为 Activity 设置启用应用程序图标向上导航功能，可以在它的 onCreate() 方法中获取 ActionBar 对象，并调用其 setDisplayHomeAsUpEnabled() 方法实现。具体代码如下：

```
01 @Override
02 protected void onCreate(Bundle savedInstanceState) {
03     super.onCreate(savedInstanceState);
04     setContentView(R.layout.activity_main);
05     getActionBar().setDisplayHomeAsUpEnabled(true);
06 }
```

14.3.2 配置父 Activity

调用 setDisplayHomeAsUpEnabled() 方法只是让应用程序图标转变为按钮，并显示一个向左的图标而已，要实现向上回退的功能，还需要在 AndroidManifest.xml 中，使用 <activity> 标记的子标记 <meta-data> 配置 Activity 的父 Activity。例如，配置 DetailActivity 的父 Activity 为 MainActivity，可以使用下面的代码：

```
01 <activity
02     android:name=".DetailActivity"
03     android:label="详细信息" >
04 <meta-data android:name="android.support.PARENT_ACTIVITY"
05     android:value=".MainActivity"/>
06 </activity>
```

14.3.3 控制导航图标的显示

添加了启用程序图标导航功能后，还需要控制导航图标是否显示，即当该 Activity 未指定父 Activity 时，无需再显示向左的箭头图标，避免误导用户。要实现该功能，只需要加上以下判断语句即可。

```
if(NavUtils.getParentActivityName(DetailActivity.this)!= null){
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);  //显示向左的箭头图标
}
```

例 14.5 模拟微信向上导航按钮

在 Android Studio 中创建 Module，名称为“Navigation Button”，在该 Module 中实现本实例，具体步骤如下：

（1）修改新建项目的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后为布局管理器添加背景图片，再将默认添加的 TextView 组件删除，最后添加一个 ImageView 组件，用于显示朋友圈选项。

（2）创建一个 Empty Activity 界面，名称为 FriendsActivity。然后在 activity_friends.xml 布局文件中为布局管理器添加背景图片，用于显示朋友圈页面。

（3）在默认创建 MainActivity 的 onCreate() 方法中，获取布局文件中添加的朋友圈选项图片，并为其设置单击事件监听器，然后实现启动另一个 Activity 功能，具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    ImageView imageView; //定义 ImageView
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView= (ImageView) findViewById(R.id.imageView);       //获取朋友圈图片
        imageView.setOnClickListener(new View.OnClickListener() {//为图片设置单击事件
            @Override
            public void onClick(View v) {
                //创建Intent对象
                Intent intent=new Intent(MainActivity.this,FriendsActivity.class);
                startActivity(intent);                              //启动Activity

            }
        });
    }
}
```

（4）打开 FriendsActivity 类，在重写的 onCreate（）方法中判断父 Activity 是否为空，不为空则设置导航图标显示，具体代码如下：

```
public class FriendsActivity extends AppCompatActivity {
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_friends);
        if (NavUtils.getParentActivityName(FriendsActivity.this) != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);//显示向左的箭头图标
        }
    }
}
```

（5）在 AndroidManifest.xml 中，使用 <activity> 标记的子标记 <meta-data> 配置 Activity 的父 Activity，关键代码如下：

```
<activity android:name=".FriendsActivity"
    android:label="朋友圈">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity"/>
</activity>
```

（6）在 AndroidManifest.xml 文件，修改 <application> 标记的 android:theme 属性值，修改后的 android:theme 属性的代码如下：

```
android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
```

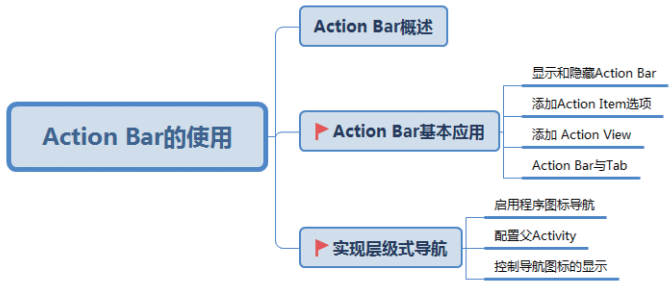
（7）运行本实例，效果如图 14.11 所示。单击朋友圈选项图片，将进入到如图 14.12 所示的子 Activity。

图 14.11　父 Activity

图 14.12　子 Activity

14.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 15 章

消息、通知、广播与闹钟

在图形界面中，对话框和通知是人机交互的两种重要形式，在开发 Android 应用时，经常需要弹出消息提示框、对话框和显示通知等内容。另外，手机中发送和接收广播以及设置闹钟也都是比较常用的功能。本章将对 Android 中如何弹出对话框、显示通知、使用广播和设置闹钟等功能进行详细介绍。

15.1 通过 Toast 类显示消息提示框

Toast 类通常用于显示一些快速提示信息，应用范围非常广泛。在前面各章的实例中，已经应用过 Toast 类来显示一个简单的消息提示框。在本节中，将对 Toast 类进行详细介绍。应用 Toast 类在屏幕中显示的消息提示框具有如下几个特点：

◆ 没有任何控制按钮。

◆ 不会获得焦点。

◆ 经过一段时间后会自动消失。

使用 Toast 类来显示消息提示框比较简单，只需要以下 3 个操作步骤即可实现。

（1）创建一个 Toast 对象。通常有两种方法，一种是使用构造方法进行创建，另一种是调用 Toast 类的 makeText() 方法创建。

使用构造方法创建一个名称为 toast 的 Toast 对象的基本代码如下：

```
Toast toast=new Toast(this);
```

调用 Toast 类的 makeText() 方法创建一个名称为 toast 的 Toast 对象的基本代码如下：

```
Toast toast=Toast.makeText(this, "要显示的内容", Toast.LENGTH_SHORT);
```

（2）调用 Toast 类提供的方法来设置该消息提示的对齐方式、页边距以及显示的内容等。常用的方法如表 15.1 所示。

表 15.1　Toast 类的常用方法

方　法	描　述
setDuration(int duration)	用于设置消息提示框持续的时间的长短，通常使用 Toast.LENGTH_LONG 或 Toast.LENGTH_SHORT 参数值

续表

方　法	描　述
setGravity(int gravity, int xOffset, int yOffset)	用于设置消息提示框的位置，参数 gravity 用于指定对齐方式，xOffset 和 yOffset 用于指定具体的偏移值
setMargin(float horizontalMargin, float verticalMargin)	用于设置消息提示的页边距
setText(CharSequence s)	用于设置要显示的文本内容
setView(View view)	用于设置将要在消息提示框中显示的视图

（3）调用 Toast 类的 show() 方法显示消息提示框。需要注意的是，一定要调用该方法，否则设置的消息提示框将不显示。

例如，在手机淘宝的主界面单击返回键时，会在屏幕下方出现一个消息提示框，提示用户“再按一次返回键退出手机淘宝”，如图 15.1 所示。这个就是通过 Toast 类显示消息提示框的一个典型应用。

图 15.1　屏幕下方出现消息提示框

说明　由于使用Toast来显示消息提示框在前面的章节中多次应用，这里就不再举例说明。

15.2　对话框与弹出窗口的使用

15.2.1　使用 AlertDialog 实现对话框

AlertDialog 类的功能非常强大，它不仅可以生成带按钮的提示对话框，还可以生成带列表的列表对话框。例如，在 360 手机助手中，单击“全部更新”按钮会弹出提示对话框，提示用户是否要进行全部更新的操作，效果如图 15.2 所示。

图 15.2　弹出提示对话框

图 15.3　提示对话框的 4 个区域

使用 AlertDialog 生成的对话框通常可分为 4 个区域，分别是图标区、标题区、内容区和按钮区。例如，图 15.2 中的提示对话框可分为如图 15.3 所示的 4 个区域。

使用 AlertDialog 可以生成的对话框，概括起来有以下 4 种。

◆ 带确定、中立和取消等多个按钮的提示对话框，其中的按钮个数不是固定的，可以根据需要添加。例如，不需要有中立按钮，那么就可以生成只带有确定和取消按钮的对话框，也可以是只带有一个按钮的对话框。

◆ 带列表的列表对话框。

◆ 带多个单选列表项和多个按钮的列表对话框。

◆ 带多个多选列表项和多个按钮的列表对话框。

在使用 AlertDialog 类生成对话框时，常用的方法如表 15.2 所示。

表 15.2　AlertDialog 类的常用方法

方　法	描　述
setTitle(CharSequence title)	为对话框设置标题
setIcon(Drawable icon)	使用 Drawable 资源为对话框设置图标
setIcon(int resId)	使用资源 ID 所指的 Drawable 资源为对话框设置图标
setMessage(CharSequence message)	为提示对话框设置要显示的内容
setButton()	为提示对话框添加按钮，可以是取消按钮、中立按钮和确定按钮。需要通过为其指定 int 类型的 whichButton 参数实现，其参数值可以是 DialogInterface.BUTTON_POSITIVE（确定按钮）、BUTTON_NEGATIVE（取消按钮）或者 BUTTON_NEUTRAL（中立按钮）

通常情况下，使用 AlertDialog 类只能生成带 N 个按钮的提示对话框，如果要生成另外 3 种列表对话框，就需要使用 AlertDialog.Builder 类，AlertDialog.Builder 类提供的常用方法如表 15.3 所示。

表 15.3 AlertDialog.Builder 类的常用方法

方　法	描　述
setTitle(CharSequence title)	用于为对话框设置标题
setIcon(Drawable icon)	使用 Drawable 资源为对话框设置图标
setIcon(int resId)	使用资源 ID 所指的 Drawable 资源为对话框设置图标
setMessage(CharSequence message)	用于为提示对话框设置要显示的内容
setNegativeButton()	用于为对话框添加取消按钮
setPositiveButton()	用于为对话框添加确定按钮
setNeutralButton()	用于为对话框添加中立按钮
setItems()	用于为对话框添加列表项
setSingleChoiceItems()	用于为对话框添加单选列表项

下面通过一个实例说明如何应用 AlertDialog 类生成各种提示对话框和列表对话框。

例 15.1　4 种不同类型的对话框

在 Android Studio 中创建一个 Module，名称为“AlertDialog”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ layout 目录下的布局文件 activity_main.xml。首先将默认添加的布局管理器修改为垂直线性布局管理器，然后将默认添加的 TextView 组件删除，再添加 4 个用于控制各种对话框显示的按钮。

（2）在 MainActivity 的 onCreate() 方法中，首先获取布局文件中添加的第 1 个按钮，也就是“显示带取消和确定按钮的对话框”按钮，并为其添加单击事件监听器，然后在重写的 onClick() 方法中，应用 AlertDialog 类创建一个带取消和确定按钮的提示对话框，具体代码如下：

```
01  //获取"显示带取消、确定按钮的对话框"按钮
02  Button button1 = (Button) findViewById(R.id.button1);
03  //为"显示带取消、确定按钮的对话框"按钮添加单击事件监听器
04  button1.setOnClickListener(new View.OnClickListener() {
05      @Override
06      public void onClick(View v) {
07          //创建对话框对象
08          AlertDialog alertDialog = new AlertDialog.Builder(MainActivity.this).create();
09          alertDialog.setIcon(R.mipmap.advise);       //设置对话框的图标
10          alertDialog.setTitle("乔布斯:");            //设置对话框的标题
11          //设置要显示的内容
12          alertDialog.setMessage("活着就是为了改变世界，难道还有其他原因吗？");
13          //添加取消按钮
14          alertDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "否",
15                  new DialogInterface.OnClickListener() {
16                      @Override
17                      public void onClick(DialogInterface dialog, int which) {
```

```
                        Toast.makeText(MainActivity.this, "您单击了否按钮",
                                Toast.LENGTH_SHORT).show();
                    }
                });
        //添加确定按钮
        alertDialog.setButton(DialogInterface.BUTTON_POSITIVE, "是",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(MainActivity.this, "您单击了是按钮 ",
                                Toast.LENGTH_SHORT).show();
                    }
                });
        alertDialog.show();                             //显示对话框
    }
});
```

（3）在 MainActivity 的 onCreate() 方法中，首先获取布局文件中添加的第 2 个按钮，也就是“显示带列表的对话框”按钮，并为其添加单击事件监听器，然后在重写的 onClick() 方法中，应用 AlertDialog 类创建一个带 4 个列表项的列表对话框，具体代码如下：

```
Button button2 = (Button) findViewById(R.id.button2);  //获取"显示带列表的对话框"按钮
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //创建名言字符串数组
        final String[] items = new String[]{"当你有使命，它会让你更专注", "要么出众，"
                + "要么出局", "活着就是为了改变世界", "求知若饥，虚心若愚"};
        //创建列表对话框对象
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(R.mipmap.advise1);              //设置对话框的图标
        builder.setTitle("请选择你喜欢的名言：");        //设置对话框的标题
        //添加列表项
        builder.setItems(items, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this,
                        "您选择了" + items[which], Toast.LENGTH_SHORT).show();
            }
        });
        builder.create().show();                        //创建对话框并显示
    }
});
```

（4）在 MainActivity 的 onCreate() 方法中，首先获取布局文件中添加的第 3 个按钮，也就是“显示带单选列表项的对话框”按钮，并为其添加单击事件监听器，然后在重写的 onClick() 方法中，应用 AlertDialog 类创建一个带 5 个单选列表项和一个确定按钮的列表对话框，具体代码如下：

```
//获取"显示带单选列表项的对话框"按钮
Button button3 = (Button) findViewById(R.id.button3);
button3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //创建名字字符串数组
        final String[] items = new String[]{"扎克伯格", "乔布斯", "拉里.埃里森",
                "安迪.鲁宾", "马云"};
        //显示带单选列表项的对话框
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(R.mipmap.advise2);                    //设置对话框的图标
        builder.setTitle("如果让你选择，你最想做哪一个：");  //设置对话框的标题
        builder.setSingleChoiceItems(items, 0, new DialogInterface.OnClickListener()
{
            @Override
            public void onClick(DialogInterface dialog, int which) {
                //显示选择结果
                Toast.makeText(MainActivity.this,
                        "您选择了" + items[which], Toast.LENGTH_SHORT).show();
            }
        });
        builder.setPositiveButton("确定", null);          //添加确定按钮
        builder.create().show();                          //创建对话框并显示
    }
});
```

（5）在 MainActivity 中定义一个 boolean 类型的数组（用于记录各列表项的状态）和一个 String 类型的数组（用于记录各列表项要显示的内容），关键代码如下：

```
private boolean[] checkedItems;          //记录各列表项的状态
private String[] items;                  //各列表项要显示的内容
```

（6）在 MainActivity 的 onCreate() 方法中，首先获取布局文件中添加的第 4 个按钮，也就是“显示带多选列表项的对话框”按钮，并为其添加单击事件监听器，然后在重写的 onClick() 方法中，应用 AlertDialog 类创建一个带 5 个多选列表项和一个确定按钮的列表对话框，具体代码如下：

```
//获取"显示带多选列表项的对话框"按钮
Button button4 = (Button) findViewById(R.id.button4);
button4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        //记录各列表项的状态
        checkedItems = new boolean[]{false, true, false, true, false};
        //各列表项要显示的内容
        items = new String[]{"开心消消乐", "球球大作战", "欢乐斗地主",
                "梦幻西游", "超级玛丽"};
        //显示带单选列表项的对话框
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(R.mipmap.advise2);              //设置对话框的图标
        builder.setTitle("请选择您喜爱的游戏：");         //设置对话框标题
        builder.setMultiChoiceItems(items, checkedItems,
                new DialogInterface.OnMultiChoiceClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                checkedItems[which] = isChecked;        //改变被操作列表项的状态

            }
        });
        //为对话框添加"确定"按钮
        builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String result = "";
                for (int i = 0; i < checkedItems.length; i++) {
                    if (checkedItems[i]) {              //当选项被选择时
                        result += items[i] + "、";      //将选项的内容添加到result中
                    }
                }
                //当result不为空时，通过消息提示框显示选择的结果
                if (!"".equals(result)) {
                    //去掉最后面添加的"、"号
                    result = result.substring(0, result.length() - 1);
                    Toast.makeText(MainActivity.this,
                            "您选择了[ " + result + " ]", Toast.LENGTH_LONG).show();
                }

            }
        });
        builder.create().show();                        //创建对话框并显示
    }
});
```

（7）运行本实例，效果分别是：j 显示带“是”和“否”按钮的对话框，如图 15.4 所示；k 显示带列表项的对话框，如图 15.5 所示；l 显示带单选列表项的对话框，如图 15.6 所示；m 显示带多选列表项的对话框，如图 15.7 所示。

图 15.4　带“是”和“否”按钮的对话框

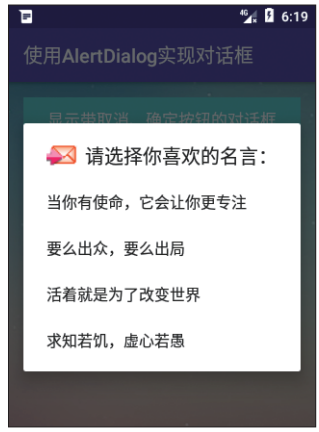


图 15.5　带列表的对话框

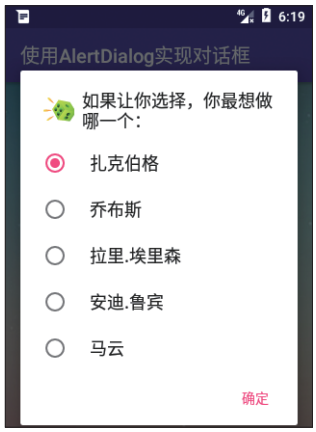


图 15.6　带单选列表项的对话框

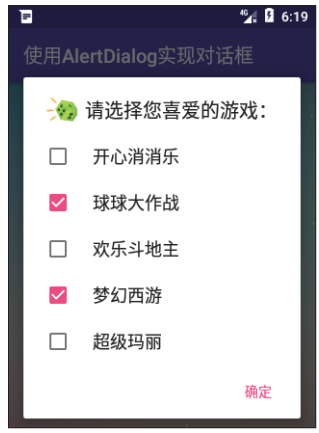


图 15.7　带多选列表项的对话框

15.2.2　TimePickerDialog 与 DatePickerDialog 的使用

TimePickerDialog 是时间选择对话框，而 DatePickerDialog 是日期选择对话框。它们两个在使用的时候要比单独 TimePicke 组件和 DatePicker 组件好用多了，并且用法也比较简单，只要通过 new 关键字创建相对应的对象即可，然后调用它们的 show() 方法就可以将时间选择对话框与日期选择对话框显示出来。TimePickerDialog 与 DatePickerDialog 同样也有相对应的事件监听器，用于获取当前所设置的事件。

下面通过一个具体的实例演示两种对话框的应用。

例 15.2　显示选择的日期与时间

在 Android Studio 中创建 Module，名称为“DateAndTime”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为水平线性布局管理器并将 TextView 组件删除，然后添加 2 个 Button 组件分别用于显示日期选择对话框与时间选择对话框。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="horizontal"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示日期选择对话框的按钮-->
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="设置日期" />
    <!--显示时间选择对话框的按钮-->
    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="设置时间" />
</LinearLayout>
```

（2）打开主活动 MainActivity.java 文件，在 onCreate() 方法中首先获取用于显示日期对话框的按钮，并为其设置单击事件监听器，然后在 onClick() 方法中实现日期选择对话框的显示，再为该对话框设置事件监听器 OnDateSetListener()，最后在 onDateSet() 方法中通过 Toast 显示当前选择的日期。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.btn1);//获取显示日期选择对话框的按钮
        //设置单击事件
        button.setOnClickListener(new View.OnClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.N)
            @Override
            public void onClick(View v) {
                Calendar calendar = Calendar.getInstance();  //用于初始化日期
                new DatePickerDialog(MainActivity.this,      //创建日期选择对话框
                        new DatePickerDialog.OnDateSetListener() {
                            @Override
                            public void onDateSet(DatePicker view, int year,
                                                  int monthOfYear, int dayOfMonth) {
```

```
                                Toast.makeText(MainActivity.this,
                                        "您选择了: " + year + "年" + (monthOfYear+1)
                                                + "月" + dayOfMonth + "日",
                                                        Toast.LENGTH_SHORT).show();
                            }
                        }
                        //设置初始化时间
                        , calendar.get(Calendar.YEAR)
                        , calendar.get(Calendar.MONTH)
                        , calendar.get(Calendar.DAY_OF_MONTH)).show();
            }
        });
    }
}
```

（3）在 onCreate() 方法中添加代码，首先获取用于显示时间选择对话框的按钮并为其设置单击事件监听器，然后在该监听器的 onClick() 方法中实现时间选择对话框的显示。关键代码如下：

```
Button button1= (Button) findViewById(R.id.btn2);  //获取显示时间选择对话框的按钮
//设置单击事件
button1.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    public void onClick(View v) {
        Calendar c = Calendar.getInstance();       //用于初始化时间
        new TimePickerDialog(MainActivity.this,    //创建时间选择对话框
                //绑定监听器
                new TimePickerDialog.OnTimeSetListener() {
                    @Override
                    public void onTimeSet(TimePicker view,
                                          int hourOfDay, int minute) {
                        Toast.makeText(MainActivity.this, "您选择了: " + hourOfDay
                                + "时" + minute + "分", Toast.LENGTH_SHORT).show();
                    }
                }
                //设置初始时间
                , c.get(Calendar.HOUR_OF_DAY)
                , c.get(Calendar.MINUTE),
                //true表示采用24小时制
                true).show();
    }
});
```

（4）运行本实例，将显示如图 15.8 所示显示触发日期、时间对话框的按钮。然后单击设置日期或设置时间按钮后将显示如图 15.9 所示的界面。选择指定的日期与时间后，单击“确定”按钮将显示如图 15.10 所示的效果。

图 15.8　触发日期、时间对话框的按钮

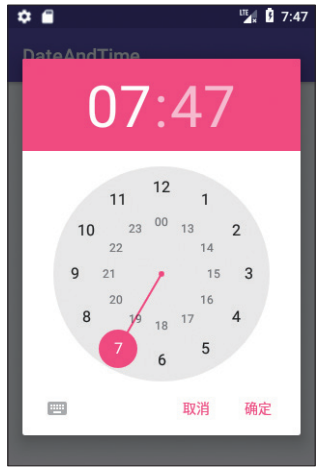


图 15.9　选择日期、时间对话框

图 15.10　显示选择的日期与时间

说明　在获取设置月份信息时，由于monthOfYear参数仅返回（0-11）所以需要在该值后面+1，即可获取准确的月份参数。

15.2.3　进度对话框 (ProgressDialog)

ProgressDialog 是 AlertDialog 的子类，它的主要功能就是可以在对话框中显示 1 个进度。使用 ProgressDialog 是非常简单的，只需要通过关键字 new 创建其对象，再通过 show() 方法将进度对话框显示出来即可。ProgressDialog 在日常生活中的 App 当中是很常见的，例如，手机银行账号登录时等待账号密码验证的进度对话框，如图 15.11 所示。或者某 App 在线升级时所显示的进度对话框，如图 15.12 所示。

图 15.11　等待账号密码验证的进度对话框

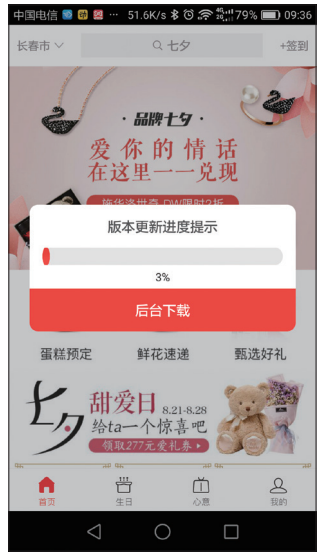


图 15.12　App 在线升级时的进度对话框

使用 ProgressDialog 创建进度对话框有以下两种方法，语法格式如下：

方法一：

```
ProgressDialog(Context context)
```

方法二：

```
ProgressDialog(Context context, int theme)
```

参数含义如表 15.4 所示。

表 15.4　参数含义

参 数 名 称	描　　述
context	上下文参数
theme	设置样式

ProgressDialog 常用的属性方法有以下几种：

◆ setMax：设置进度对话框内进度条的最大值。

◆ setMessage: 设置进度对话框内显示的消息。

◆ setProgress：设置进度对话框内进度条的进度值。

◆ setProgressStyle：设置进度对话框进度条的样式。

下面通过一个具体的实例演示 ProgressDialog 的应用。

例 15.3　模拟手机银行正在载入进度对话框

在 Android Studio 中创建 Module，名称为“ProgressDialog”，具体步骤如下：

（1）在 com.mingrisoft 包中创建 1 个名称为 Main2Activity 的 Activity 用于显示登录后的界面，然后打开 activity_main2.xml 文件，将默认添加的布局管理器修改为相对布局管理器，然后为其设置背景图片。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg2"
    tools:context="com.mingrisoft.Main2Activity">
</RelativeLayout>
```

（2）打开 Main2Activity.java 文件，首先让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类包，然后在 onCreate() 方法中设置全屏显示。修改后代码如下：

```
public class Main2Activity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main2);
        //全屏
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
    }
}
```

（3）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并删除 TextView 组件，然后添加 1 个 ImageView 组件用于实现登录按钮并为该组件设置单击事件方法，名称为“onProgressDialog”。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg1"
    tools:context="com.mingrisoft.MainActivity">
    <!--登录按钮-->
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="210dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:onClick="onProgressDialog"
        android:src="@mipmap/btn_bg" />
</RelativeLayout>
```

（4）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中设置全屏显示。修改后代码如下：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //全屏
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
    }
}
```

（5）在 onCreate() 方法的下面创建 onProgressDialog() 方法用于实现登录按钮的单击事件，在该方法中首先创建 1 个进度对话框，然后创建 1 个延迟消息用于实现进度对话框显示 3 秒后自动关闭进度对话框并跳转到登录后的界面当中。具体代码如下：

```
01  //登录按钮的单击事件
02  public void onProgressDialog(View view) {
03      //创建进度对话框
04      final ProgressDialog progressDialog = new ProgressDialog(this);
05      progressDialog.setMessage("正在载入...");        //进度对话框显示的文字
06      progressDialog.show();                           //显示进度对话框
07      new Handler().postDelayed(new Runnable() {       //创建延迟
08          @Override
09          public void run() {
10              progressDialog.dismiss();                //3秒后关闭对话框
11              //跳转界面
12              Intent intent=new Intent(MainActivity.this,Main2Activity.class);
13              startActivity(intent);
14          }
15      }, 3000);
16  }
```

（6）运行本实例，效果如图 15.13 所示，单击“登录”按钮将显示如图 15.14 所示的进度对话框，3 秒后将自动跳转至如图 15.15 所示登录后的界面当中。

图 15.13　默认显示的效果

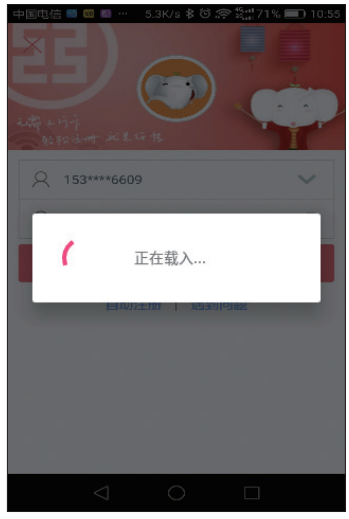


图 15.14　进度对话框

图 15.15　登录后的界面

15.2.4 弹出窗口（PopupWindow）

PopupWindow 是 Android v7 包中可以实现弹出窗口的 1 个组件。PopupWindow 与 AlertDialog 有些相似，都是通过某个组件触发后所弹出的一个窗口，不同的是 AlertDialog 不能直接指定其显示的位置，只能通过设置 WindowManager 参数来改变位置。而 PopupWindow 是可以随意指定其显示的位置的，对于开发者来说 PopupWindow 会更加灵活方便一些。例如，手机 QQ 消息界面右上角的菜单就可以通过 PopupWindow 来实现如图 15.16 所示。

图 15.16　QQ 消息界面右上角的菜单

生成 1 个 PopupWindow 一般情况下需要下面的 4 个构造方法：

方法一：

```
PopupWindow(Context context);
```

方法二：

```
public PopupWindow(View contentView);
```

方法三：

```
public PopupWindow(View contentView, int width, int height);
```

方法四：

```
public PopupWindow(View contentView, int width, int height, boolean focusable);
```

参数含义如表 15.5 所示。

表 15.5　参数含义

参 数 名 称	描　　述
context	上下文参数
contentView	显示在窗口中的内容
width	窗口的宽度
height	窗口的高度
focusable	如果弹出窗口可以聚焦，则为 true，否则为 false

PopupWindow 常用的属性方法有以下几种：

◆ setFocusable()：设置 PopupWindow 是否可以获得焦点。

◆ setTouchable()：设置 PopupWindow 是否可以触摸。
◆ setOutsideTouchable()：设置非 PopupWindow 区域可触摸。
◆ setAnimationStyle()：设置 PopupWindow 的动画样式。
◆ setContentView()：更改 PopupWindow 显示的内容。
◆ showAsDropDown()：在锚定视图的左下角弹出窗口。在该方法中可以设置以下 4 种参数：
◎ anchor：用于设置弹出窗口显示位置的组件。
◎ xoff：设置与组件的水平偏移。
◎ yoff：设置与组件的垂直偏移。
◎ gravity：设置弹出窗口与组件对齐的方式。
◆ showAtLocation(): 在指定的位置弹出窗口。在该方法中可以设置以下 4 种参数：
◎ parent：用于设置弹出窗口显示位置的组件。
◎ gravity：设置以组件对齐位置为原点。
◎ x：以组件的原点，设置水平偏移。
◎ y：以组件的原点，设置垂直偏移。

下面通过一个具体的实例演示 PopupWindow 的应用。

例 15.4 模拟微信右上角菜单

在 Android Studio 中创建 Module，名称为“PopupWindow”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，首先修改默认添加的 TextView 组件用于显示标题文字，然后添加 1 个 Button 组件用于显示右上角触发菜单的按钮并为该按钮设置单击事件方法的名称为“OnMenu”。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/bg"
    tools:context="com.mingrisoft.MainActivity">
    <!--微信标题-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="微信"
        android:textColor="#FFFFFF"
        android:textSize="25sp" />
    <!--菜单按钮-->
    <Button
        android:id="@+id/btn_menu"
        android:layout_width="40dp"
        android:layout_height="40dp"
```

```
        android:layout_alignParentRight="true"
        android:background="@mipmap/btn_add"
        android:onClick="OnMenu" />
</RelativeLayout>
```

（2）在 res/layout 目录下创建 1 个名称为“menu”的布局文件，首先在该布局文件中设置为垂直线性布局管理器，然后添加 5 个 Button 组件用于实现菜单中的 5 个按钮。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="right"
    android:orientation="vertical">
    <!--菜单发起群聊按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:background="#333333"
        android:drawableLeft="@mipmap/icon_group"
        android:drawablePadding="10dip"
        android:paddingLeft="20dip"
        android:paddingRight="49dp"
        android:text="发起群聊"
        android:textColor="#FFFFFF"
        android:textSize="12sp" />
 ......<!--此处省略部分代码，请参照源码补全-->
    <!--菜单帮助与反馈按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:background="#333333"
        android:drawableLeft="@mipmap/icon_help"
        android:drawablePadding="10dip"
        android:paddingLeft="20dip"
        android:paddingRight="40dp"
        android:text="帮助与反馈"
        android:textColor="#FFFFFF"
        android:textSize="12sp" />
</LinearLayout>
```

（3）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后定义所需的全局变量。具体代码如下：

```
01  public class MainActivity extends Activity {
02      PopupWindow popupWindow;     //定义PopupWindow
03      @Override
04      protected void onCreate(Bundle savedInstanceState) {
05          super.onCreate(savedInstanceState);
06          setContentView(R.layout.activity_main);
07      }
08  }
```

（4）在 onCreate() 方法的下面，创建 1 个名称为 OnMenu() 方法，在该方法中首先获取自定义菜单的布局文件，然后通过构造方法创建 1 个 PopupWindow 菜单并设置其显示在触发按钮的下面。具体代码如下：

```
01  public void OnMenu(View view){
02      //获取自定义菜单的布局文件
03      View popupWindow_view = getLayoutInflater().inflate(R.layout.menu, null,false);
04      //创建PopupWindow实例,设置菜单宽度和高度为包裹其自身内容
05      popupWindow = new PopupWindow(popupWindow_view, ActionBar.LayoutParams.WRAP_CONTENT,
06              ActionBar.LayoutParams.WRAP_CONTENT, true);
07      //设置菜单显示在按钮的下面
08      popupWindow.showAsDropDown(findViewById(R.id.btn_menu),0,0);
09  }
```

（5）打开 res/values 目录中的 colors.xml 文件，将名称为“colorPrimaryDark”的颜色代码修改为“#333333”黑色。关键代码如下：

```
<color name="colorPrimaryDark">#333333</color>
```

（6）运行本实例，效果如图 15.17 所示，单击右上角的触发按钮将显示如图 15.18 所示的 PopupWindow 菜单。

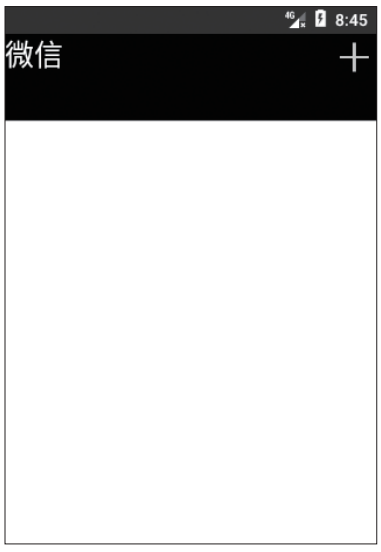


图 15.17　默认显示的效果

图 15.18　显示 PopupWindow 菜单

15.3 使用 Notification 在状态栏上显示通知

状态栏位于手机屏幕的最上方，一般用于显示手机当前的网络状态、系统时间以及电池状态等信息。在使用手机时，当有未接来电或有新短消息时，手机会给出相应的提示信息，这些提示信息通常会显示到手机屏幕的状态栏上。例如，手机在接收到短信时，会在状态栏中出现一个短信息的通知，如图 15.19 所示。

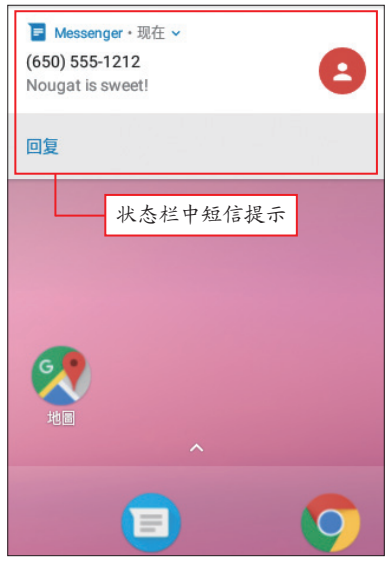


图 15.19 状态栏中的短信提示

Android 也提供了用于处理这些信息的类，它们是 Notification 和 NotificationManager。其中 Notification 代表的是具有全局效果的通知，而 NotificationManager 则是用来发送 Notification 通知的系统服务。在 Android 8.0 系统中还添加了 NotificationChannel，用于帮助用户管理通知。

实现发送和显示通知也比较简单，大致可以分为以下 5 个步骤实现。

（1）调用 getSystemService() 方法获取系统的 NotificationManager 服务。

（2）创建 NotificationChannel 对象，实现创建通知渠道。

（3）创建一个 Notification 对象。

（4）为 Notification 对象设置各种属性，其中常用的方法如表 15.6 所示。

表 15.6 Notification 对象中的常用方法

方　法	描　述
setDefaults()	设置通知 LED 灯、音乐、振动等
setAutoCancel()	设置单击通知后，状态栏自动删除通知
setContentTitle()	设置通知标题
setContentText()	设置通知内容

续表

方　法	描　述
setSmalllcon()	为通知设置图标
setLargelcon()	为通知设置大图标
setContentIntent()	设置单击通知后将要启动的程序组件对应的 PendingIntent

（5）通过 NotificationManager 类的 notify() 方法发送 Notification 通知。

说明 通过NotificationChannel类创建通知渠道时，将显示如图15.20所示的错误提示，需要将Module的最小版本设置为API 26即Android 8.0版本，此处单击红色错误浪线按快捷键Alt+Enter在提示列表中选择Add @TargetApi（O）Annotation添加目标Api（O）8.0注释，如图15.21所示。

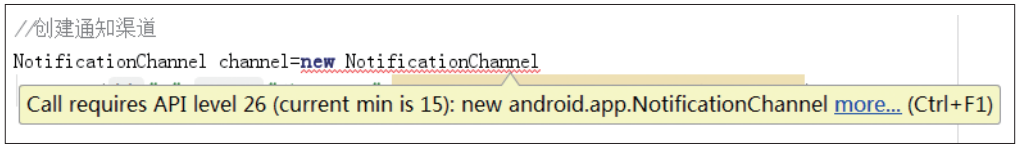


图 15.20　设置最小版本为 API 26

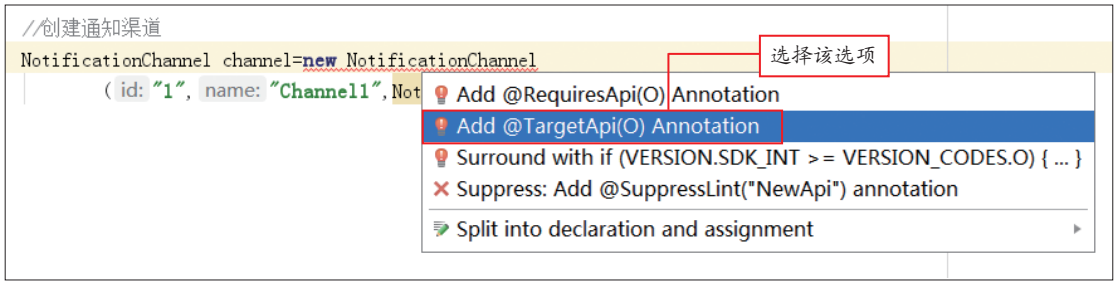


图 15.21　添加 Api 版本注释

下面通过一个实例说明如何使用 Notification 在状态栏上显示通知。

例 15.5　模拟淘宝在状态栏上显示活动通知

在 Android Studio 中创建 Module，名称为“Notification”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/ layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后删除 TextView 组件，再为布局管理器添加背景图片。

（2）在 MainActivity 中创建一个常量，用于保存通知的 ID，关键代码如下：

```
final int NOTIFYID = 0x123;                    //通知的ID
```

（3）在 MainActivity 的 onCreate() 方法中，调用 getSystemService() 方法获取系统的 NotificationManager 服务，关键代码如下：

```
01  //获取通知管理器服务，用于发送通知
02  final NotificationManager notificationManager =
03          (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

（4）首先创建 NotificationChannel 对象，然后创建一个 Notification.Builder 对象，并设置其相关属性，再创建一个启动其他 Activity 的 Intent，并通过 setContentIntent() 方法设置通知栏单击跳转，最后通过通知管理器发送通知，具体代码如下：

```
//创建通知渠道
NotificationChannel channel=new NotificationChannel
        ("1","Channel1",NotificationManager.IMPORTANCE_DEFAULT);
notificationManager.createNotificationChannel(channel);
//创建一个Notification对象
Notification.Builder notification = new Notification.Builder(this,"1");
//设置打开该通知，该通知自动消失
notification.setAutoCancel(true);
//设置通知的图标
notification.setSmallIcon(R.mipmap.packet);
//设置下拉列表中的大图标
notification.setLargeIcon
                (BitmapFactory.decodeResource(getResources(),R.mipmap.packet));
//设置通知内容的标题
notification.setContentTitle("奖励百万红包！！！");
//设置通知内容
notification.setContentText("点击查看详情！");
//设置使用系统默认的声音，默认震动
notification.setDefaults(Notification.DEFAULT_SOUND
        | Notification.DEFAULT_VIBRATE);
//设置发送时间
notification.setWhen(System.currentTimeMillis());
//创建一个启动其他Activity的Intent
Intent intent = new Intent(MainActivity.this, DetailActivity.class);
PendingIntent pi = PendingIntent.getActivity(MainActivity.this, 0, intent, 0);
//设置通知栏单击跳转
notification.setContentIntent(pi);
//发送通知
notificationManager.notify(NOTIFYID, notification.build());
```

注意 在程序中需要访问系统振动器，这就需要在AndroidManifest.xml中声明使用权限，具体代码如下：

```
<!--添加振动器权限-->
<uses-permission android:name="android.permission.VIBRATE"/>
```

（5）在 com.mingrisoft 包中创建一个 Empty Activity，名称为“DetailActivity”，用于实现页面跳转。然后在 activity_detail.xml 布局文件中将默认添加的布局管理器修改为相对布局管理器，并且为布局管理器添加背景图片。

（6）运行本实例，将显示如图 15.22 所示的界面，按住状态栏并向下滑动，直到出现如图 15.23 所示的通知窗口，单击第一项，如图 15.24 所示，查看后该通知的图标将不在状态栏中显示。

图 15.22　淘宝的活动通知

图 15.23　显示通知列表

图 15.24　活动通知的详细内容

说明　本实例的通知方式需要在Android 8.0系统版本下运行，所以建议使用模拟器进行测试。

15.4　BroadcastReceiver 使用

15.4.1　BroadcastReceiver 简介

BroadcastReceiver 是接收广播通知的组件。广播是一种同时通知多个对象的事件通知机制。类似日常生活中的广播，允许多个人同时收听，也允许不收听。通常情况下，广播通知是以消息提示框、对话框或者通知的形式进行体现的。例如，在接收到一条短信之后，系统会发出一条广播，当广播接收器接收到该广播时，将以通知和对话框两种形式提示，如图 15.25 所示。再如，当更新系

统日期时间时，系统也会发出一条广播，当广播接收器接收到该广播时，将以消息提示框的形式提示用户，如图 15.26 所示。

图 15.25　以对话框的形式提示

图 15.26　以消息提示框的形式提示

Android 中广播来源是系统事件，例如按下拍照键、电池电量低以及安装新应用等，还有普通应用程序，例如启动特定线程、文件下载完毕等。这里主要介绍系统广播，下面列举出一些常用的系统事件，当这些事件发生时就会发送系统广播。

- 电池电量低。
- 系统启动完成。
- 系统日期发生改变。
- 系统时间发生改变。
- 系统连接电源。
- 系统被关闭。

BroadcastReceiver 类是所有广播接收器的抽象基类。其实现类用来对发送出来的广播进行筛选并做出响应。广播接收器的生命周期非常简单。当消息到达时，接收器调用 onReceive() 方法，在该方法结束后，BroadcastReceiver 实例失效。

说明 onReceive()方法是实现BroadcastReceiver类时需要重写的方法。

当用户需要进行广播时，可以通过 Activity 程序中的 sendBroadcast() 方法触发所有的广播组件，而每一个广播组件在进行广播启动之前，也必须判断用户所传递的广播操作是否是指定的 Action 类型，如果是，则进行广播的处理。广播的处理过程如图 15.27 所示。

图 15.27　广播处理过程

在 Android 操作系统中，每启动一个广播都需要重新实例化一个新的广播组件对象，并自动调用类中的 onReceive() 方法对广播事件进行处理。

用于接收的广播有以下两大类：

◆ 普通广播

使用 Context.sendBroadcast() 方法发送，它们完全是异步的。广播的全部接收者以未定义的顺序运行，通常在同一时间。这使得消息传递的效率比较高，但缺点是接收者不能将处理结果传递给下一个接收者，并且无法终止广播的传播。

◆ 有序广播

使用 Context.sendOrderedBroadcast() 方法发送，它们每次只发送给优先级较高的接收者，然后由优先级较高的接收者再传播到优先级较低的接收者。由于每个接收者依次运行，它能为下一个接收者生成一个结果，或者它能完全终止广播以便不传递给其他接收者。有序接收者运行顺序由匹配的 intent-filter 的 android:priority 属性控制，具有相同优先级的接收者运行顺序任意。

综上所述，普通广播和有序广播的特点如表 15.7 所示。

表 15.7　普通广播和有序广播的特点

普 通 广 播	有 序 广 播
可以在同一时刻被所有接收者接收到	相同优先级的接收者接收顺序是随机的，不同优先级的接收者按照优先级由高到低的顺序依次接收
接收者不能将处理结果传递给下一个接收者	接收者可以将处理结果传递给下一个接收者
无法终止广播的传播	可以终止广播的传播

15.4.2 BroadcastReceiver 应用

例 15.6　实现一个广播接收器的功能

在 Android Studio 中创建一个 Module，名称为“BroadcastReceiver”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后删除文本框组件，再添加一个按钮组件并为其设置 id。

（2）打开默认创建的 MainActivity，在 onCreate() 方法中首先创建过滤器，然后创建广播接收者并注册广播，再设置按钮单击事件监听器，最后创建一个 Intent，并且为 Intent 添加一个动作，然后通过 sendBroadcast() 方法发送广播。具体代码如下：

```
01   //过滤器
02   IntentFilter intentFilter = new IntentFilter("com.mingrisoft");
03   //创建广播接收者的对象
04  MyReceiver myReceiver=new MyReceiver();
05   //注册广播接收者的对象
06   registerReceiver(myReceiver,intentFilter);
07   Button button= (Button) findViewById(R.id.Broadcast);    //获取布局文件中的广播按钮
08   button.setOnClickListener(new View.OnClickListener() { //为按钮设置单击事件
09       @Override
10       public void onClick(View v) {
```

```
11          Intent intent=new Intent();                 //创建Intent对象
12          intent.setAction("com.mingrisoft");         //为Intent添加动作com.mingrisoft
13          sendBroadcast(intent);                      //发送广播
14      }
15  });
```

（3）在 java\com.mingrisoft 包下，单击鼠标右键，在弹出的快捷菜单中选择 New → Other → Broadcast Receiver 菜单项，创建一个名称为“MyReceiver.java”的广播接收器。然后在该类中声明两个动作，最后在重写的 onReceive() 方法中根据广播发送的动作给出不同的提示，具体代码如下：

```
01 public class MyReceiver extends BroadcastReceiver {
02     private static final String action1="com.mingrisoft";   //声明第一个动作
03     private static final String action2="mingrisoft";       //声明第二个动作
04     public MyReceiver() {
05     }
06     @Override
07     public void onReceive(Context context, Intent intent) {
08         if (intent.getAction().equals(action1)){
09             Toast.makeText(context, "MyReceiver收到:com.mingrisoft的广播",
10                     Toast.LENGTH_SHORT).show();          //回复该动作收到广播
11         }else if (intent.getAction().equals(action2)){
12             Toast.makeText(context, "MyReceiver收到:mingrisoft的广播",
13                     Toast.LENGTH_SHORT).show();          //回复该动作收到广播
14         }
15     }
16 }
```

（4）运行本实例，将显示如图 15.28 所示的界面。

图 15.28　接收广播提示界面

15.5 使用 AlarmManager 设置闹钟

AlarmManager 类是 Android 提供的用于在未来的指定时间弹出一个警告信息，或者完成指定操作的类。实际上 AlarmManager 是一个全局的定时器，使用它可以在指定的时间或指定的周期启动其他的组件（包括 Activity、Service 和 BroadcastReceiver）。使用 AlarmManager 设置警告后，Android 将自动开启目标应用，即使手机处于休眠状态。因此，使用 AlarmManager 也可以实现关机后仍可响应的闹钟。

15.5.1 AlarmManager 简介

在 Android 中，要获取 AlarmManager 对象，类似于获取 NotificationManager 服务，也需要使用 Context 类的 getSystemService() 方法来实现，具体代码如下：

```
Context.getSystemService(Context.ALARM_SERVICE)
```

获取 AlarmManager 对象后，就可以应用该对象提供的相关方法来设置警告。AlarmManager 对象提供的常用方法如表 15.8 所示。

表 15.8　AlarmManager 对象的常用方法

方　法	描　述
cancel(PendingIntent operation)	取消 AlarmManager 的定时服务
set(int type, long triggerAtTime, PendingIntent operation)	设置当到达参数 triggerAtTime 所指定的时间时，按照 type 参数所指定的服务类型启动由 operation 参数指定的组件
setInexactRepeating(int type, long triggerAtTime, long interval, PendingIntent operation)	设置一个非精确的周期性任务。例如，我们设置一个每个小时启动一次的闹钟，但是系统并不一定总在每个小时开始时启动闹钟
setRepeating(int type, long triggerAtTime, long interval, PendingIntent operation)	设置一个周期性执行的定时服务
setTime(long millis)	设置定时的时间
setTimeZone(String timeZone)	设置系统默认的时区

在设置定时服务时，AlarmManager 提供了以下 4 种类型。

◆ ELAPSED_REALTIME

用于设置从现在时间开始过了一定时间后启动提醒功能。当系统进入睡眠状态时，这种类型的定时不会唤醒系统。直到系统下次被唤醒才传递它，该定时所用的时间是相对时间，是从系统启动后开始计时的（包括睡眠时间），可以通过调用 SystemClock.elapsedRealtime() 方法获得。

◆ ELAPSED_REALTIME_WAKEUP

用于设置从现在时间开始过了一定时间后启动提醒功能。这种类型的定时能够唤醒系统，即使

系统处于休眠状态也会启动提醒功能。使用方法与 ELAPSED_REALTIME 类似，也可以通过调用 SystemClock.elapsedRealtime() 方法获得。

◆ RTC

用于设置当系统调用 System.currentTimeMillis() 方法的返回值与指定的触发时间相等时启动提醒功能。当系统进入睡眠状态时，这种类型的定时不会唤醒系统。直到系统下次被唤醒才传递它，该定时所用的时间是绝对时间（采用 UTC 时间），可以通过调用 System.currentTimeMillis() 方法获得。

◆ RTC_WAKEUP

用于设置当系统调用 System.currentTimeMillis() 方法的返回值与指定的触发时间相等时启动提醒功能。这种类型的定时能够唤醒系统，即使系统处于休眠状态也会启动提醒功能。

15.5.2 设置一个简单的闹钟

在 Android 中，使用 AlarmManager 设置闹钟比较简单，下面通过一个实例来介绍如何实现一个简单的闹钟。

例 15.7 实现一个定时启动的闹钟

在 Android Studio 中创建 Module，名称为“Timing Alarm Clock”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后删除 TextView 组件，再添加一个时间拾取器和一个设置闹钟的按钮，关键代码如下：

```
<TimePicker
    android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:text="设置闹钟" />
```

（2）打开默认创建的 MainActivity，然后在该类中创建两个成员变量，分别是时间拾取器和日历对象，具体代码如下：

```
TimePicker timepicker;                    //时间拾取器
Calendar c;                               //日历对象
```

（3）在 MainActivity 的 onCreate() 方法中，首先获取日历对象，然后获取时间拾取器组件，并设置其采用 24 小时制。具体代码如下：

```
c=Calendar.getInstance();                                    //获取日历对象
timepicker = (TimePicker) findViewById(R.id.timePicker1);    //获取时间拾取组件
timepicker.setIs24HourView(true);                            //设置使用24小时制
```

（4）获取布局管理器中添加的“设置闹钟”按钮，并为其添加单击事件监听器，在重写的onClick() 方法中，首先创建一个 Intent 对象，并获取显示闹钟的 PendingIntent 对象，然后再获取AlarmManager 对象，并且用时间拾取器中设置的小时数和分钟数设置日历对象的时间，接下来再调用 AlarmManager 对象的 set() 方法设置一个闹钟，最后显示一个提示闹钟设置成功的消息提示。具体代码如下：

```
Button button1 = (Button) findViewById(R.id.button1);              //获取"设置闹钟"按钮
//为"设置闹钟"按钮添加单击事件监听器
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,
                AlarmActivity.class);                               //创建一个Intent对象
        PendingIntent pendingIntent = PendingIntent.getActivity(
                MainActivity.this, 0, intent, 0);   //获取显示闹钟的PendingIntent对象
        //获取AlarmManager对象
        AlarmManager alarm = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
        c.set(Calendar.HOUR_OF_DAY, timepicker.getCurrentHour());//设置闹钟的小时数
        c.set(Calendar.MINUTE, timepicker.getCurrentMinute());    //设置闹钟的分钟数
        c.set(Calendar.SECOND,0);                                   //设置闹钟的秒数
        alarm.set(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(),
                pendingIntent);                                     //设置一个闹钟
        Toast.makeText(MainActivity.this, "闹钟设置成功", Toast.LENGTH_SHORT)
                .show();                                            //显示一个消息提示
    }
});
```

（5）创建一个 AlarmActivity，用于显示闹钟提示内容。在该 Activity 中重写 onCreate() 方法，在该方法中创建并显示一个带“确定”按钮的对话框，显示闹钟的提示内容。关键代码如下：

```
public class AlarmActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        AlertDialog alert = new AlertDialog.Builder(this).create();
        alert.setIcon(R.mipmap.alarm);                  //设置对话框的图标
        alert.setTitle("传递正能量：");                  //设置对话框的标题
        alert.setMessage("要么出众，要么出局");           //设置要显示的内容
        //添加确定按钮
        alert.setButton(DialogInterface.BUTTON_POSITIVE,"确定",
                new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {}
        });
        alert.show();                                   //显示对话框
    }
}
```

（6）在 AndroidManifest.xml 文件中配置 AlarmActivity，配置的主要属性有 Activity 使用的实现类和标签，具体代码如下：

```
<activity
    android:name=".AlarmActivity"
    android:label="@string/clock">
</activity>
```

说明 在测试该实例前，需要在模拟器“设置”→“日期和时间”中关闭“自动确定日期和时间”与“自动确定时区”，然后将“选择时区”设置为“中国标准时间”。

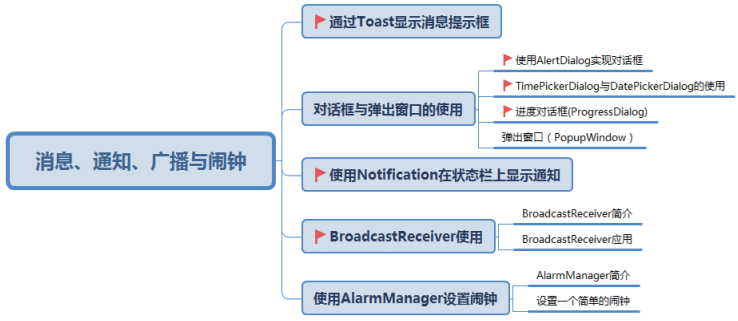
（7）运行本实例，在显示的界面中，设置如图 15.29 所示的闹钟，当到达闹钟所设定的时间时，将显示如图 15.30 所示的提示对话框。

图 15.29　设置闹钟

图 15.30　显示闹钟提示

15.6 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 16 章 图形图像处理技术

图形图像处理技术在 Android 中非常重要，特别是在开发益智类游戏或者 2D 游戏时，都离不开图形图像处理技术的支持。本章将对 Android 中的图形图像处理技术进行详细介绍。

16.1 常用绘图类

在 Android 中，绘制图像时最常应用的就是 Paint 类、Canvas 类、Path 类、Bitmap 类和 BitmapFactory 类。有关这几个类的作用描述如表 16.1 所示。

表 16.1 常用绘图类

类	描　述
Paint 类	画笔类，用来描述图形的颜色和风格
Canvas 类	画布类，用于绘制各种图形
Path 类	路径类，用于绘制路径
Bitmap 类	位图类，用于获取图像文件信息，主要对图像进行剪切、旋转、缩放等操作
BitmapFactory 类	位图工厂类，用于从不同的数据源来解析、创建 Bitmap 对象

在现实生活中，有画笔和画布就可以作画了，在 Android 中也是如此，通过 Paint 类和 Canvas 类即可绘制图像。

16.1.1 Paint 类

Paint 类代表画笔，用来描述图形的颜色和风格，如线宽、颜色、透明度和填充效果等信息。使用 Paint 类时，首先需要创建该类的对象，这可以通过该类提供的构造方法来实现。通常情况下，只需要使用无参数的构造方法来创建一个使用默认设置的 Paint 对象，具体代码如下：

```
Paint paint=new Paint();
```

创建 Paint 类的对象后，还可以通过该对象提供的方法来对画笔的默认设置进行改变，例如，改变画笔的颜色、笔触宽度等。用于改变画笔设置的常用方法如表 16.2 所示。

表 16.2 Paint 类的常用方法

方 法	描 述
setARGB(int a, int r, int g, int b)	用于设置颜色，各参数值均为 0~255 之间的整数，分别用于表示透明度、红色、绿色和蓝色值
setColor(int color)	用于设置颜色，参数 color 可以通过 Color 类提供的颜色常量指定，也可以通过 Color.rgb(int red,int green,int blue) 方法指定
setAlpha(int a)	用于设置透明度，值为 0~255 之间的整数
setAntiAlias(boolean aa)	用于指定是否使用抗锯齿功能，如果使用，会使绘图速度变慢
setPathEffect(PathEffect effect)	用于设置绘制路径时的路径效果，如点划线
setShader(Shader shader)	用于设置渐变，可以使用 LinearGradient（线性渐变）、RadialGradient（径向渐变）或者 SweepGradient（角度渐变）
setShadowLayer(float radius, float dx, float dy, int color)	用于设置阴影，参数 radius 为阴影的角度；dx 和 dy 为阴影在 x 轴和 y 轴上的距离；color 为阴影的颜色。如果参数 radius 的值为 0，那么将没有阴影
setStrokeJoin(Paint.Join join)	用于设置画笔转弯处的连接风格，参数值为 Join.BEVEL、Join.MITER 或 Join.ROUND
setStrokeWidth(float width)	用于设置笔触的宽度
setStyle(Paint.Style style)	用于设置填充风格，参数值为 Style.FILL、Style.FILL_AND_STROKE 或 Style.STROKE
setTextAlign(Paint.Align align)	用于设置绘制文本时的文字对齐方式，参数值为 Align.CENTER、Align.LEFT 或 Align.RIGHT
setTextSize(float textSize)	用于设置绘制文本时的文字的大小

例如，要定义一个画笔，然后在画布上绘制一个以线性渐变颜色填充的矩形，可以使用下面的代码：

```
01  Paint paint=new Paint();                          //定义一个默认的画笔
02  //线性渐变
03  Shader shader=new LinearGradient(0, 0, 100, 100, Color.RED,
04          Color.GREEN, Shader.TileMode.MIRROR);
05  paint.setShader(shader);                          //为画笔设置渐变器
```

应用该画笔绘制的矩形的效果如图 16.1 所示。

图 16.1 绘制以线性渐变颜色填充的矩形

说明 关于如何在画布上绘制矩形，将在16.2.1节进行介绍。

16.1.2 Canvas 类

Canvas 类代表画布，通过该类提供的方法，可以绘制各种图形（如矩形、圆形和线条等）。通常情况下，要在 Android 中绘图，需要先创建一个继承自 View 类的视图，并且在该类中重写其 onDraw(Canvas canvas) 方法，然后在显示绘图的 Activity 中添加该视图。下面将通过一个具体的实例来说明如何创建用于绘图的画布。

下面通过一个实例来演示画笔与画布的应用。

例 16.1 绘制渐变色矩形

在 Android Studio 中创建一个 Module，名称为“Draw Rectangle”。实现本实例的具体步骤如下：

（1）修改新建项目的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为帧布局管理器并设置其 id 为 frameLayout，然后将默认添加的 TextView 组件删除。

（2）打开默认创建的 MainActivity，在该文件中，创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas) 方法，关键代码如下：

```
01  private class MyView extends View {
02      public MyView(Context context) {
03          super(context);
04      }
05      @Override
06      protected void onDraw(Canvas canvas) {        //重写onDraw()方法
07          super.onDraw(canvas);
08      }
09  }
```

（3）在 MyView 的 onDraw() 方法中，首先定义一个默认的画笔，然后添加一个线性渐变器，再为画笔设置渐变器，最后绘制矩形，关键代码如下：

```
01  Paint paint = new Paint();                        //定义一个默认的画笔
02  //线性渐变
03  Shader shader = new LinearGradient(0, 0, 100, 100,
04          Color.RED, Color.GREEN, Shader.TileMode.MIRROR);
05  paint.setShader(shader);                          //为画笔设置渐变器
06  canvas.drawRect(10, 10, 280, 150, paint);         //绘制矩形
```

（4）在 MainActivity 的 onCreate() 方法中，获取布局文件中添加的帧布局管理器，并将步骤（2）中创建的 MyView 视图添加到该帧布局管理器中，关键代码如下：

```
01  //获取帧布局管理器
02  FrameLayout fragment = (FrameLayout) findViewById(R.id.frameLayout);
03  fragment.addView(new MyView(this));  //将自定义视图的内部类添加到布局管理器中
```

（5）运行本实例，将显示如图 16.2 所示的效果。

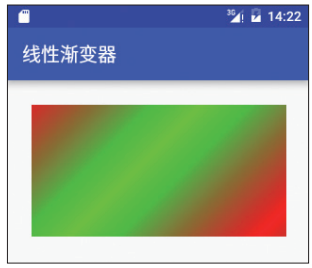


图 16.2　绘制以渐变色填充的矩形

16.1.3 Path 类

Path 类用于绘制路径。该类中包含一组矢量绘图方法，如画圆、矩形、弧、线条等。常用的绘图方法如表 16.3 所示。

表 16.3　Path 类的常用绘图方法

方　法	描　述
addArc(RectF oval, float startAngle, float sweepAngle)	添加弧形路径
addCircle(float x, float y, float radius, Path.Direction dir)	添加圆形路径
addOval(RectF oval, Path.Direction dir)	添加椭圆形路径
addRect(RectF rect, Path.Direction dir)	添加矩形路径
addRoundRect(RectF rect, float rx, float ry, Path.Direction dir)	添加圆角矩形路径
moveTo(float x, float y)	设置开始绘制直线的起始点
lineTo(float x, float y)	在 moveTo() 方法设置的起始点与该方法指定的结束点之间画一条直线，如果在调用该方法之前没使用 moveTo() 方法设置起始点，那么将从（0,0）点开始绘制直线
quadTo(float x1, float y1, float x2, float y2)	用于根据指定的参数绘制一条线段轨迹
close()	闭合路径

说明　在使用addCircle()、addOval()、addRect()和addRoundRect()方法时，需要指定Path.Direction类型的常量，可选值为Path.Direction.CW（顺时针）和Path.Direction.CCW（逆时针）。

例如，创建一个折线，可以使用下面的代码：

```
01  Path path=new Path();          //创建并实例化一个path对象
02  path.moveTo(50,50);            //设置起始点
03  path.lineTo(100, 10);          //设置第1条边的结束点，也是第2条边的起始点
04  path.lineTo(150, 50);          //设置第2条边的结束点
```

将该路径绘制到画布上的效果如图 16.3 所示。

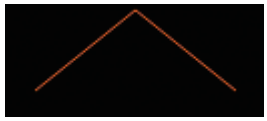


图 16.3　绘制两条线组成的折线

图 16.4　绘制一个三角形

如果要绘制一个三角形路径，可以在上面的代码的末尾使用 close() 方法闭合路径，代码如下：

```
path.close();          //闭合路径
```

将该路径绘制到画布上的效果如图 16.4 所示。

16.1.4 Bitmap 类

Bitmap 类代表位图，是 Android 系统中图像处理的一个重要类。使用该类，不仅可以获取图像文件信息，进行图像剪切、旋转、缩放等操作，而且还可以指定格式保存图像文件。对于这些操作，都可以通过 Bitmap 类提供的方法来实现。Bitmap 类提供的常用方法如表 16.4 所示。

表 16.4　Bitmap 类的常用方法

方　法	描　述
compress(Bitmap.CompressFormat format, int quality, OutputStream stream)	用于将 Bitmap 对象压缩为指定格式并保存到指定的文件输出流中，其中 format 参数值可以是 Bitmap.CompressFormat.PNG、Bitmap.CompressFormat. JPEG 和 Bitmap.CompressFormat.WEBP
createBitmap(Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)	用于从源位图的指定坐标点开始，“挖取”指定宽度和高度的一块图像来创建新的 Bitmap 对象，并按 Matrix 指定规则进行变换
createBitmap(int width, int height, Bitmap.Config config)	用于创建一个指定宽度和高度的新的 Bitmap 对象
createBitmap(Bitmap source, int x, int y, int width, int height)	用于从源位图的指定坐标点开始，“挖取”指定宽度和高度的一块图像来创建新的 Bitmap 对象
createBitmap(int[] colors, int width, int height, Bitmap.Config config)	使用颜色数组创建一个指定宽度和高度的新的 Bitimap 对象，其中，数组元素的个数为 width*height
createBitmap(Bitmap src)	用于使用源位图创建一个新的 Bitmap 对象
createScaledBitmap(Bitmap src, int dstWidth, int dstHeight, boolean filter)	用于将源位图缩放为指定宽度和高度的新的 Bitmap 对象
isRecycled()	用于判断 Bitmap 对象是否被回收
recycle()	强制回收 Bitmap 对象

例如，创建一个包括 4 个像素（每个像素对应一种颜色）的 Bitmap 对象的代码如下：

```
Bitmap bitmap=Bitmap.createBitmap(new int[]{
        Color.RED,
        Color.GREEN,
        Color.BLUE,
        Color.MAGENTA}, 4, 1,
        Bitmap.Config.RGB_565);
```

16.1.5 BitmapFactory 类

在 Android 中，还提供了一个 BitmapFactory 类，该类是一个工具类，用于从不同的数据源来解析、创建 Bitmap 对象。BitmapFactory 类提供的创建 Bitmap 对象的常用方法如表 16.5 所示。

表 16.5 BitmapFactory 类的常用方法

方 法	描 述
decodeFile(String pathName)	用于从给定的路径所指定的文件中解析、创建 Bitmap 对象
decodeFileDescriptor(FileDescriptor fd)	用于从 FileDescriptor 对应的文件中解析、创建 Bitmap 对象
decodeResource(Resources res, int id)	用于根据给定的资源 id，从指定的资源中解析、创建 Bitmap 对象
decodeStream(InputStream is)	用于从指定的输入流中解析、创建 Bitmap 对象

例如，要解析 SD 卡上的图片文件 img01.jpg 并创建对应的 Bitmap 对象，可以使用下面的代码：

```
String path = Environment.getExternalStorageDirectory() + "/img01.jpg";
Bitmap bm= BitmapFactory.decodeFile(path);
```

要解析 Drawable 资源中保存的图片文件 img02.jpg 并创建对应的 Bitmap 对象，可以使用下面的代码：

```
Bitmap bm=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.img02);
```

16.2 绘制 2D 图像

Android 提供了非常强大的本机二维图形库，用于绘制 2D 图像。在 Android 应用中，比较常用的是绘制几何图形、文本、路径和图片等，下面分别进行介绍。

16.2.1 绘制几何图形

常见的几何图形包括点、线、弧、圆形、矩形等。在 Android 中，Canvas 类提供了丰富的绘制几何图形的方法，通过这些方法，可以绘制出各种几何图形。常用的绘制几何图形的方法如表 16.6 所示。

表 16.6　Canvas 类提供的绘制几何图形的方法

方　法	描　述	举　例	绘图效果
drawArc(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint)	绘制弧	RectF rectf=new RectF(10, 20, 100, 110); canvas.drawArc(rectf, 0, 60, true, paint);	
		RectF rectf1=new RectF(10, 20, 100, 110); canvas.drawArc(rectf1, 0, 60, false, paint);	
drawCircle(float cx, float cy, float radius, Paint paint)	绘制圆形	paint.setStyle(Style.STROKE); canvas.drawCircle(50, 50, 15, paint);	
drawLine(float startX, float startY, float stopX, float stopY, Paint paint)	绘制一条线	canvas.drawLine(100, 10, 150, 10, paint);	
drawLines(float[] pts, Paint paint)	绘制多条线	canvas.drawLines(new float[]{10,10, 30,10, 30,10, 15,30, 15,30, 10,10}, paint);	
drawOval(RectF oval, Paint paint)	绘制椭圆	RectF rectf=new RectF(40, 20, 80, 40); canvas.drawOval(rectf,paint);	
drawPoint(float x, float y, Paint paint)	绘制一个点	canvas.drawPoint(10, 10, paint);	
drawPoints(float[] pts, Paint paint)	绘制多个点	canvas.drawPoints(new float[]{10,10, 15,10, 20,15, 25,10, 30,10}, paint);	
drawRect(float left, float top, float right, float bottom, Paint paint)	绘制矩形	canvas.drawRect(10, 10, 40, 30, paint);	
drawRoundRect(RectF rect, float rx, float ry, Paint paint)	绘制圆角矩形	RectF rectf=new RectF(40, 20, 80, 40); canvas.drawRoundRect(rectf, 6, 6, paint);	

说明　表16.6中给出的绘图效果使用的画笔均为以下代码所定义的画笔。

```
Paint paint=new Paint();                //创建一个采用默认设置的画笔
paint.setAntiAlias(true);               //使用抗锯齿功能
paint.setColor(Color.RED);              //设置颜色为红色
paint.setStrokeWidth(2);                //笔触的宽度为2像素
paint.setStyle(Style.STROKE);           //填充样式为描边
```

下面通过一个实例来演示绘制几何图形的应用。

例 16.2　绘制 Android 机器人

在 Android Studio 中创建 Module，名称为“Draw Robot”，实现本实例的具体步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为帧布局管理器并设置其 id，然后将默认添加的 TextView 组件删除。

（2）创建一个名称为 MyView 的内部类，让其继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas) 方法。

（3）在 MyView 的 onDraw() 方法中，首先创建一个画笔，并设置画笔的相关属性，然后绘制机器人的头、眼睛、天线、身体、胳膊和腿，具体代码如下：

```
Paint paint=new Paint();                              //默认设置创建一个画笔
paint.setAntiAlias(true);                             //使用抗锯齿功能
paint.setColor(0xFFA4C739);                           //设置画笔的颜色为绿色
//绘制机器人的头
RectF rectf_head=new RectF(10, 10, 100, 100);
rectf_head.offset(90, 20);
canvas.drawArc(rectf_head, -10, -160, false, paint); //绘制弧
//绘制眼睛
paint.setColor(Color.WHITE);                          //设置画笔的颜色为白色
canvas.drawCircle(125, 53, 4, paint);                 //绘制圆
canvas.drawCircle(165, 53, 4, paint);                 //绘制圆
paint.setColor(0xFFA4C739);                           //设置画笔的颜色为绿色
//绘制天线
paint.setStrokeWidth(2);                              //设置笔触的宽度
canvas.drawLine(110, 15, 125, 35, paint);             //绘制线
canvas.drawLine(180, 15, 165, 35, paint);             //绘制线
//绘制身体
canvas.drawRect(100, 75, 190, 150, paint);            //绘制矩形
RectF rectf_body=new RectF(100,140,190,160);
canvas.drawRoundRect(rectf_body, 10, 10, paint);//绘制圆角矩形
//绘制胳膊
RectF rectf_arm=new RectF(75,75,95,140);
canvas.drawRoundRect(rectf_arm, 10, 10, paint); //绘制左侧的胳膊
rectf_arm.offset(120, 0);                             //设置在X轴上偏移120像素
canvas.drawRoundRect(rectf_arm, 10, 10, paint); //绘制右侧的胳膊
//绘制腿
RectF rectf_leg=new RectF(115,150,135,200);
canvas.drawRoundRect(rectf_leg, 10, 10, paint); //绘制左侧的腿
rectf_leg.offset(40, 0);                              //设置在X轴上偏移40像素
canvas.drawRoundRect(rectf_leg, 10, 10, paint); //绘制右侧的腿
```

（4）在 MainActivity 的 onCreate() 方法中获取布局文件中的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。代码如下：

```
//获取帧布局管理器
FrameLayout fragment = (FrameLayout) findViewById(R.id.frameLayout);
fragment.addView(new MyView(this));              //将自定义视图的内部类添加到布局管理器中
```

（5）运行本实例，将显示如图 16.5 所示的界面效果。

图 16.5　在屏幕上绘制 Android 机器人

16.2.2 绘制文本

在 Android 中，虽然可以通过 TextView 或图片显示文本，但是在开发游戏，特别是开发 RPG（角色）类游戏时，会包含很多文字，使用 TextView 和图片显示文本不太合适，这时，就需要通过绘制文本的方式来实现。例如，图 16.6 和图 16.7 所示的游戏中的对话就是通过绘制文本的方式实现的。

图 16.6　在游戏中绘制对白文字

图 16.7　在游戏中绘制提示文字

Canvas 类提供了 drawText() 方法用于在画布的指定位置绘制文字。该方法比较常用的语法格式如下：

```
drawText(String text, float x, float y, Paint paint)
```

在该语法中，参数 text 用于指定要绘制的文字；x 用于指定文字起始位置的 X 坐标；y 用于指定文字起始位置的 Y 坐标；paint 用于指定使用的画笔。

例如，要在画布上输出文字“求知若饥，虚心若愚”，可以使用下面的代码：

```
01  Paint paintText=new Paint();
02  paintText.setTextSize(20);
03  canvas.drawText("求知若饥，虚心若愚", 165,65, paintText);
```

下面通过一个实例来演示绘制文本的应用。

例 16.3　绘制游戏对白文字

在 Android Studio 中创建 Module，名称为“Draw Text”，实现本实例的具体步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为帧布局管理器，然后为帧布局管理器添加背景图片与 id，再删除默认添加的 TextView 组件。

（2）创建一个名称为 MyView 的内部类，让其继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas) 方法。

（3）在 MyView 的 onDraw() 方法中，首先创建一个采用默认设置的画笔，然后设置画笔颜色以及对齐方式、文字大小和使用抗锯齿功能，再通过 drawText() 方法绘制文字。具体代码如下：

```
01  Paint paintText=new Paint();                                  //创建一个采用默认设置的画笔
02  paintText.setColor(Color.BLACK);                              //设置画笔颜色
03  paintText.setTextAlign(Paint.Align.LEFT);                     //设置文字左对齐
04  paintText.setTextSize(12);                                    //设置文字大小
05  paintText.setAntiAlias(true);                                 //使用抗锯齿功能
06  canvas.drawText("不，我不想去！", 245, 45, paintText);          //通过drawText()方法绘制文字
07  canvas.drawText("你想和我一起",175,160,paintText);             //通过drawText()方法绘制文字
08  canvas.drawText("去探险吗？",175,175,paintText);               //通过drawText()方法绘制文字
```

（4）在 MainActivity 的 onCreate() 方法中获取布局文件中的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。然后让 MainActivity 直接继承自 Activity。

（5）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（6）运行本实例，效果如图 16.8 所示。

图 16.8　在画布上绘制文字

16.2.3 绘制图片

在 Android 中，Canvas 类不仅可以绘制几何图形、文件和路径，还可用来绘制图片。要想使用 Canvas 类绘制图片，只需要使用 Canvas 类提供的如表 16.7 所示的方法，将 Bitmap 对象中保存的图片绘制到画布上即可。

表 16.7　Canvas 类提供的绘制图片的常用方法

方　法	描　述
drawBitmap(Bitmap bitmap, Rect src, RectF dst, Paint paint)	用于从指定点绘制从源位图中“挖取”的一块
drawBitmap(Bitmap bitmap, float left, float top, Paint paint)	用于在指定点绘制位图
drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)	用于从指定点绘制从源位图中“挖取”的一块

例如，从源位图上“挖取”从（0,0）点到（500,300）点的一块图像，然后绘制到画布的（50,50）点到（450,350）点所指区域，可以使用下面的代码：

```
Rect src=new Rect(0,0,500,300);                    //设置挖取的区域
Rect dst=new Rect(50,50,450,350);                  //设置绘制的区域
canvas.drawBitmap(bm, src, dst, paint);            //绘制图片
```

下面通过一个实例来演示绘制图片的应用。

例 16.4 绘制挖取后图片

在 Android Studio 中创建 Module，名称为“Draw Pictures”，实现本实例的具体步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为帧布局管理器并且为其设置 id，然后将默认添加的 TextView 组件删除。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas) 方法，然后在 onCreate() 方法中，获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 DDMS 中，将名称为 img01.jpg 图片导入到模拟器的 sdcard 路径下（具体方法请参见 9.1.2 节的文件管理器的使用）。

（4）在 MyView 的 onDraw() 方法中，首先创建一个画笔，并指定要绘制图片的路径，获取要绘制图片所对应的 Bitmap 对象，再在画布的指定位置绘制 Bitmap 对象，以及从源图片中挖取指定区域并绘制挖取到的图像。具体代码如下：

```
Paint paint = new Paint();                         //创建一个采用默认设置的画笔
//指定图片文件的路径
String path = Environment.getExternalStorageDirectory() + "/img01.jpg";
Bitmap bm = BitmapFactory.decodeFile(path);        //获取图片文件对应的Bitmap对象
canvas.drawBitmap(bm, 0, 30, paint);               //将获取的Bitmap对象绘制在画布的指定位置
Rect src = new Rect(105, 70, 220, 170);            //设置挖取的区域
Rect dst = new Rect(350, 90, 465, 190);            //设置绘制的区域
canvas.drawBitmap(bm, src, dst, paint);            //绘制挖取到的图像
```

（5）打开 AndroidManifest.xml 文件，在其中设置 SD 卡的读取权限，具体代码如下：

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

（6）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（7）运行本实例，效果如图 16.9 所示。

图 16.9 绘制图片

说明 第一次运行本实例将会出现空指针异常，无法获取图片路径。用户可以在模拟器中依次进入→“设置”→“应用和通知”→“你的App应用名称”→“权限”开启存储空间权限，即可获取图片指定路径。

16.2.4 绘制路径

绘制路径对于 2D 图像或者游戏开发来说都是很重要的。例如，在开发画板涂鸦游戏时就需要应用路径记录绘制路线，如图 16.10 所示。另外，在进行某些游戏开发的时候也会需要沿着路径绘制图形，如图 16.11 所示。

图 16.10 画板涂鸦中应用路径

图 16.11 在游戏中绘制路径

绘制一条路径可以分为创建路径和将定义好的路径绘制在画布上两个步骤。在 16.1.3 节已经介绍了如何创建路径，下面将使用 Canvas 类提供的 drawPath() 方法，可以将定义好的路径绘制在画布上。该方法的语法格式如下：

```
drawPath(Path path, Paint paint)
```

在该语法中，path 用于指定创建的 Path 对象实例，paint 用于指定使用的画笔。

在 Android 的 Canvas 类中，还提供了另一个应用路径的方法 drawTextOnPath()，也就是沿着指定的路径绘制字符串。使用该方法可绘制环形文字。该方法的语法格式如下：

```
drawTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint)
```

在该语法中，参数 text 用于指定要绘制的文字；path 用于指定创建的 Path 对象实例，hOffset 用于指定文字的水平偏移；vOffset 用于指定文字的垂直偏移；paint 用于指定使用的画笔。

下面通过一个实例来演示绘制路径的应用。

例 16.5 简易涂鸦板

在 Android Studio 中创建 Module，名称为“Graffiti Board”，实现本实例的具体步骤如下：

（1）创建一个名称为 DrawView 的类，该类继承自 android.view.View 类。在该类中，首先定义程序中所需的属性，然后添加构造方法，并重写 onDraw(Canvas canvas) 方法，具体代码如下：

```
01  public class DrawView extends View {
02      private int view_width = 0;      //屏幕的宽度
03      private int view_height = 0;     //屏幕的高度
04      private float preX;              //起始点的X坐标值
05      private float preY;              //起始点的y坐标值
06      private Path path;               //路径
07      public Paint paint = null;       //画笔
08      Bitmap cacheBitmap = null;       //定义一个内存中的图片，该图片将作为缓冲区
09      Canvas cacheCanvas = null;       //定义cacheBitmap上的Canvas对象
10      public DrawView(Context context, AttributeSet set) { //构造方法
11          super(context, set);
12      }
13      @Override
14      protected void onDraw(Canvas canvas) {  //重写onDraw()方法
15          super.onDraw(canvas);
16      }
17  }
```

（2）修改新建项目的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为帧布局管理器，然后将内边距与默认添加的 TextView 组件删除。并且在帧布局管理器中添加步骤（1）中创建的自定义视图。最后添加一个 ImageButton 组件，用于清空画板。修改后的代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <FrameLayout
03      xmlns:android="http://schemas.android.com/apk/res/android"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
06      android:layout_height="match_parent"
07      tools:context="com.mingrisoft.MainActivity">
08      <!--自定义View-->
09      <com.mingrisoft.DrawView
10          android:id="@+id/dv"
11          android:layout_width="match_parent"
12          android:layout_height="match_parent"
13          android:layout_gravity="center_horizontal|bottom"/>
14      <!--清除按钮-->
15      <ImageButton
16          android:id="@+id/btn_clear"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:src="@mipmap/clear"
20          android:layout_gravity="right|bottom"
21          />
22  </FrameLayout>
```

（3）在 DrawView 类的构造方法中，首先获取屏幕的宽度和高度，并创建一个与该 View

相同大小的缓存区，然后创建一个新的画布，并实例化一个路径，再将内存中的位图绘制到 cacheCanvas 中，最后实例化一个画笔，并设置画笔的相关属性。关键代码如下：

```
view_width = context.getResources().getDisplayMetrics().widthPixels; //获取屏幕的宽度
view_height = context.getResources().getDisplayMetrics().heightPixels;//获取屏幕的高度
System.out.println(view_width + "*" + view_height);                    //屏幕宽和高
//创建一个与该View相同大小的缓存区
cacheBitmap = Bitmap.createBitmap(view_width, view_height, Bitmap.Config.ARGB_8888);
cacheCanvas = new Canvas();                    //创建一个新的画布
path = new Path();                             //实例化路径
cacheCanvas.setBitmap(cacheBitmap);            //在cacheCanvas上绘制cacheBitmap
paint = new Paint();                           //实例化画笔
paint.setColor(Color.RED);                     //设置默认的画笔颜色
//设置画笔风格
paint.setStyle(Paint.Style.STROKE);            //设置填充方式为描边
paint.setStrokeWidth(2);                       //设置默认笔触的宽度为1像素
paint.setAntiAlias(true);                      //使用抗锯齿功能
```

（4）在 DrawView 类的 onDraw() 方法中，添加以下代码，用于绘制 cacheBitmap、绘制路径，关键代码如下：

```
canvas.drawBitmap(cacheBitmap, 0, 0, null);//绘制cacheBitmap
canvas.drawPath(path, paint);              //绘制路径
```

（5）在 DrawView 类中，重写 onTouchEvent() 方法，为该视图添加触摸事件监听器，在该方法中，首先获取触摸事件发生的位置，然后应用 switch 语句对事件的不同状态添加响应代码，最后调用 invalidate() 方法更新视图。具体代码如下：

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    //获取触摸事件的发生位置
    float x = event.getX();                    //获取x坐标
    float y = event.getY();                    //获取y坐标
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:          //当手指按下时
            path.moveTo(x, y);                 //将绘图的起始点移到（x,y）坐标点的位置
            preX = x;
            preY = y;
            break;
        case MotionEvent.ACTION_MOVE:          //根据触摸过程与位置绘制线条
            float dx = Math.abs(x - preX);     //计算x值的移动距离
            float dy = Math.abs(y - preY);     //计算y值的移动距离
            if (dx >= 5 || dy >= 5) {          //判断是否在允许的范围内
                path.quadTo(preX, preY, (x + preX) / 2, (y + preY) / 2); //贝塞尔曲线
                preX = x;
                preY = y;
            }
```

```
            break;
        case MotionEvent.ACTION_UP:                 //当手指抬起时
            cacheCanvas.drawPath(path, paint);      //绘制路径
            path.reset();                           //重置路径
            break;
    }
    invalidate();                                   //刷新
    return true;                                    //返回true表明处理方法已经处理该事件
}
```

（6）编写 clear() 方法，用于清空画板功能，具体代码如下：

```
public void clear() {
    if (cacheCanvas != null) {          //如果绘制路径不为空
        path.reset();                   //重置路径
        cacheCanvas.drawColor(Color.TRANSPARENT, PorterDuff.Mode.CLEAR);
        invalidate();                   //刷新
    }
}
```

（7）在 MainActivity 的 onCreate() 方法中，首先获取自定义的视图，然后获取一个清空画板的图片按钮并为其设置监听事件，最后在 onClick() 方法中获取 DrawView 类中的 clear() 方法，用于清空画板。关键代码如下：

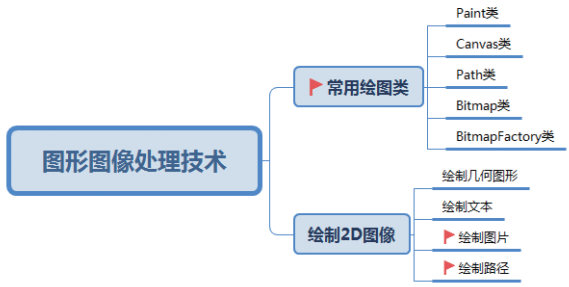
```
final DrawView drawView= (DrawView) findViewById(R.id.dv);          //获取自定义的绘图视图
ImageButton button= (ImageButton) findViewById(R.id.btn_clear); //获取清空按钮
button.setOnClickListener(new View.OnClickListener() {          //为按钮设置监听事件
    @Override
    public void onClick(View v) {
        drawView.clear();                                       //调用清除方法
    }
});
```

（8）运行本实例，将显示如图 16.12 所示的效果。

图 13.12　简易涂鸦板

16.3 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 17 章

Android 中的动画

动画技术在 Android 中非常重要，现在的手机应用中，动画已经成为了不可或缺的功能。本章将对 Android 中的逐帧动画、补间动画以及属性动画进行详细的介绍。在应用 Android 进行项目开发时，特别是在进行游戏开发时，经常需要涉及动画。下面将分别介绍如何实现这三种动画。

17.1 逐帧动画

逐帧（Frame）动画就是顺序播放提前准备好的静态图像，利用人眼的“视觉暂留”原理，给用户造成动画的错觉。实现逐帧动画比较简单，只需要以下两个步骤。

（1）在 Android 的 XML 资源文件中定义一组用于生成动画的图片资源，可以使用包含一系列 <item></item> 子标记的 <animation-list></animation-list> 标记来实现，具体语法格式如下：

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true|false">
    <item android:drawable="@drawable/图片资源名1" android:duration="integer" />
    …  <!-- 省略了部分<item></item>标记 -->
    <item android:drawable="@drawable/图片资源名n" android:duration="integer" />
</animation-list>
```

在上面的语法中，android:oneshot 属性用于设置是否循环播放，默认值为 true，表示循环播放；android:drawable 属性用于指定要显示的图片资源；android:duration 属性用于指定图片资源持续的时间。

（2）使用步骤（1）中定义的动画资源。通常情况下，可以将其作为组件的背景使用。例如，可以在布局文件中添加一个线性布局管理器，然后将该布局管理器的 android:background 属性设置为所定义的动画资源。也可以将定义的动画资源作为 ImageView 的背景使用。

说明 在Android中还支持在Java代码中创建逐帧动画。具体的步骤是：首先创建AnimationDrawable对象，然后调用addFrame()方法向动画中添加帧，每调用一次addFrame()方法，将添加一个帧。

例 17.1 用刷子画出外星人

在 Android Studio 中创建一个 Module，名称为“Draw”。实现本实例的具体步骤如下：

（1）首先在 res/drawable 目录中添加一个名称为 max.xml 的 XML 资源文件，然后在该文件中定义组成动画的图片资源，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@mipmap/max_1" android:duration="150"/>
    <item android:drawable="@mipmap/max_2" android:duration="150"/>
    <item android:drawable="@mipmap/max_3" android:duration="150"/>
    <item android:drawable="@mipmap/max_4" android:duration="150"/>
    <item android:drawable="@mipmap/max_5" android:duration="150"/>
    <item android:drawable="@mipmap/max_6" android:duration="150"/>
    <item android:drawable="@mipmap/max_7" android:duration="150"/>
    <item android:drawable="@mipmap/max_8" android:duration="150"/>
    <item android:drawable="@mipmap/max_9" android:duration="150"/>
    <item android:drawable="@mipmap/max_10" android:duration="150"/>
```

说明 由于动画资源代码较多，这里仅给出部分代码，详细代码请参阅实例源码即可。

（2）修改新建项目的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并且在该布局管理器中将默认添加的 TextView 组件删除，然后添加 1 个 ImageView 控件用于显示指定的动画资源，关键代码如下：

```
<ImageView
    android:id="@+id/max"
    android:layout_centerInParent="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/max"/>
```

（3）在 MainActivity 中，定义图像控件 ImageView 与动画资源对象 AnimationDrawable，代码如下：

```
private ImageView imageView;                    //图像控件
private AnimationDrawable animationDrawable;    //动画资源对象
```

（4）在 MainActivity 中的 onCreate() 方法中获取 ImageView 控件与 AnimationDrawable 对象，代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    imageView = (ImageView) findViewById(R.id.max);//实例化控件对象
    animationDrawable = (AnimationDrawable) imageView.getDrawable();
}
```

（5）重写 onResume() 方法与 onPause() 方法，实现界面获取焦点时开启帧动画，界面暂停时关闭帧动画，代码如下：

```
@Override
```

```
02  protected void onResume() {
03      super.onResume();
04      animationDrawable.start();                  //开启帧动画
05  }
06  @Override
07  protected void onPause() {
08      super.onPause();
09      animationDrawable.stop();                   //关闭帧动画
10  }
```

（6）运行本实例，将显示如图 17.1 所示的效果。

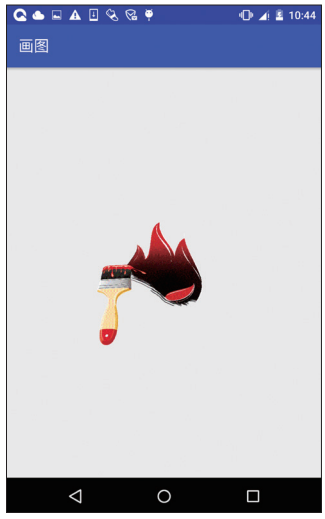


图 17.1　用刷子画外星人

17.2 补间动画

Android 除了支持逐帧动画之外，还支持补间（Tween）动画。在实现补间动画时，只需要定义动画开始和动画结束的关键帧，而动画变化的中间帧由系统自动计算并补齐。对于补间动画而言，开发者只需要指定动画开始和结束的关键帧，并指定动画的持续时间即可。其示意图如图 17.2 所示。

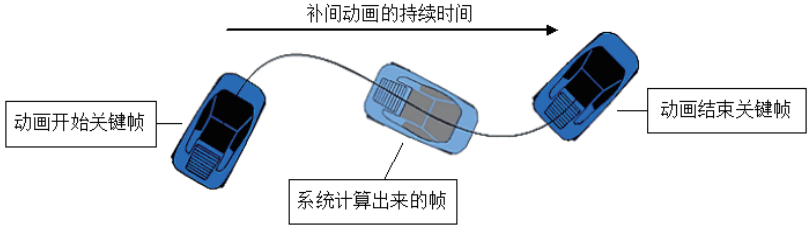


图 17.2　补间动画示意图

在 Android 中，提供了 4 种补间动画。下面分别进行介绍。

17.2.1 旋转动画（RotateAnimation）

旋转动画就是通过为动画指定开始时的旋转角度、结束时的旋转角度以及持续时间来创建动画。在旋转时，还可以通过指定轴心点坐标来改变旋转的中心。同透明度渐变动画一样，也可以在 XML 文件中定义旋转动画资源文件，基本的语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource">
    <rotate
        android:fromDegrees="float"
        android:toDegrees="float"
        android:pivotX="float"
        android:pivotY="float"
        android:repeatMode="reverse|restart"
        android:repeatCount="次数|infinite"
        android:duration="Integer"/>
</set>
```

在上面的语法中，各属性说明如表 17.1 所示。

表 17.1 定义旋转动画时常用的属性

属 性	描 述
android:interpolator	用于控制动画的变化速度，使得动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值见表 17.5
android:fromDegrees	用于指定动画开始时的旋转角度
android:toDegrees	用于指定动画结束时的旋转角度
android:pivotX	用于指定轴心点的 X 坐标
android:pivotY	用于指定轴心点的 Y 坐标
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒

例如，定义一个让图片从 0° 转到 360° 、持续时间为 2 秒钟、中心点在图片中心的动画，可以使用下面的代码：

```
01  <rotate
02      android:fromDegrees="0"
03      android:toDegrees="360"
04      android:pivotX="50%"
05      android:pivotY="50%"
06      android:duration="2000">
07  </rotate>
```

17.2.2 缩放动画（ScaleAnimation）

缩放动画就是通过为动画指定开始时的缩放系数、结束时的缩放系数以及持续时间来创建动画。在缩放时，还可以通过指定轴心点坐标来改变缩放的中心。同透明度渐变动画一样，也可以在XML 文件中定义缩放动画资源文件，基本的语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource">
    <scale
        android:fromXScale="float"
        android:toXScale="float"
        android:fromYScale="float"
        android:toYScale="float"
        android:pivotX="float"
        android:pivotY="float"
        android:repeatMode="reverse|restart"
        android:repeatCount="次数|infinite"
        android:duration="Integer"/>
</set>
```

在上面的语法中，各属性说明如表 17.2 所示。

表 17.2　定义缩放动画时常用的属性

属　性	描　述
android:interpolator	用于控制动画的变化速度，使得动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值见表 17.5
android:fromXScale	用于指定动画开始时水平方向上的缩放系数，值为 1.0 表示不变化
android:toXScale	用于指定动画结束时水平方向上的缩放系数，值为 1.0 表示不变化
android:fromYScale	用于指定动画开始时垂直方向上的缩放系数，值为 1.0 表示不变化
android:toYScale	用于指定动画结束时垂直方向上的缩放系数，值为 1.0 表示不变化
android:pivotX	用于指定轴心点的 X 坐标
android:pivotY	用于指定轴心点的 Y 坐标
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性值可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒

例如，定义一个以图片的中心为轴心点，将图片放大 2 倍、持续时间为 2 秒钟的动画，可以使用下面的代码：

```
01  <scale android:fromXScale="1"
02      android:fromYScale="1"
03      android:toXScale="2.0"
```

```
04      android:toYScale="2.0"
05      android:pivotX="50%"
06      android:pivotY="50%"
07      android:duration="2000"/>
```

17.2.3 平移动画（Translate Animation）

平移动画就是通过为动画指定开始时的位置、结束时的位置以及持续时间来创建动画。同透明度渐变动画一样，也可以在 XML 文件中定义平移动画资源文件，基本的语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource">
    <translate
        android:fromXDelta="float"
        android:toXDelta="float"
        android:fromYDelta="float"
        android:toYDelta="float"
        android:repeatMode="reverse|restart"
        android:repeatCount="次数|infinite"
        android:duration="Integer"/>
</set>
```

在上面的语法中，各属性说明如表 17.3 所示。

表 17.3 定义平移动画时常用的属性

属 性	描 述
android:interpolator	用于控制动画的变化速度，使得动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值见表 17.5
android:fromXDelta	用于指定动画开始时水平方向上的位置
android:toXDelta	用于指定动画结束时水平方向上的位置
android:fromYDelta	用于指定动画开始时垂直方向上的位置
android:toYDelta	用于指定动画结束时垂直方向上的位置
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒

例如，定义一个让图片从（0,0）点到（300,300）点、持续时间为 2 秒钟的动画，可以使用下面的代码：

```
01  <translate
02      android:fromXDelta="0"
03      android:toXDelta="300"
```

```
04      android:fromYDelta="0"
05      android:toYDelta="300"
06      android:duration="2000">
07  </translate>
```

17.2.4 透明度渐变动画（AlphaAnimation）

透明度渐变动画是指通过 View 组件透明度的变化来实现 View 的渐隐渐显效果。它主要通过为动画指定开始时的透明度、结束时的透明度以及持续时间来创建动画。同逐帧动画一样，也可以在 XML 文件中定义透明度渐变动画的资源文件，基本的语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@[package:]anim/interpolator_resource">
    <alpha
        android:repeatMode="reverse|restart"
        android:repeatCount="次数|infinite"
        android:duration="Integer"
        android:fromAlpha="float"
        android:toAlpha="float" />
</set>
```

在上面的语法中，各属性说明如表 17.4 和 17.5 所示。

表 17.4　定义透明度渐变动画时常用的属性

属　性	描　述
android:interpolator	用于控制动画的变化速度，使动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值如表 17.5 所示
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性值可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒
android:fromAlpha	用于指定动画开始时的透明度，值为 0.0 代表完全透明，值为 1.0 代表完全不透明
android:toAlpha	用于指定动画结束时的透明度，值为 0.0 代表完全透明，值为 1.0 代表完全不透明

表 17.5　android:interpolator 属性的常用属性值

属　性　值	描　述
@android:anim/linear_interpolator	动画一直在做匀速改变
@android:anim/accelerate_interpolator	动画在开始的地方改变较慢，然后开始加速
@android:anim/decelerate_interpolator	动画在开始的地方改变速度较快，然后开始减速
@android:anim/accelerate_decelerate_interpolator	动画在开始和结束的地方改变速度较慢，在中间的时候加速

续表

属 性 值	描 述
@android:anim/cycle_interpolator	动画循环播放特定的次数，变化速度按正弦曲线改变
@android:anim/bounce_interpolator	动画结束的地方采用弹球效果
@android:anim/anticipate_overshoot_interpolator	在动画开始的地方先向后退一小步，再开始动画，到结束的地方再超出一小步，最后回到动画结束的地方
@android:anim/overshoot_interpolator	动画快速到达终点并超出一小步，最后回到动画结束的地方
@android:anim/anticipate_interpolator	在动画开始的地方先向后退一小步，再快速到达动画结束的地方

例如，定义一个让 View 组件从完全透明到完全不透明、持续时间为 2 秒钟的动画，可以使用下面的代码：

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="0"
        android:toAlpha="1"
        android:duration="2000"/>
</set>
```

例 17.2 淡入、淡出的补间动画

在 Android Studio 中创建 Module，名称为“Tween Animation”，具体步骤如下：

（1）在新建项目的 res 目录中，创建一个名称为 anim 的目录，并在该目录中创建实现淡入、淡出的动画资源文件。

① 创建名称为 anim_alpha_in.xml 的 XML 资源文件，在该文件中定义一个淡入效果的动画，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="0"
        android:toAlpha="1"
        android:duration="4000"/>
</set>
```

② 创建名称为 anim_alpha_out.xml 的 XML 资源文件，在该文件中定义一个淡出效果的动画，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="1"
        android:toAlpha="0"
        android:duration="2000"/>
</set>
```

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的

布局管理器修改为相对布局管理器，并将 TextView 组件删除，然后添加 ViewFlipper 组件用于实现切换图片，关键代码如下：

```
<ViewFlipper
    android:id="@+id/flipper"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</ViewFlipper>
```

（3）打开主活动 MainActivity，让其实现 GestureDetector.OnGestureListener 接口与相对应的方法。然后定义所需的成员变量与动画数组，关键代码如下：

```
ViewFlipper flipper;                          //定义ViewFlipper
GestureDetector detector;                     //定义手势检测器
Animation[] animation = new Animation[2];      //定义动画数组，为ViewFlipper指定切换动画
final int distance = 50;                      //定义手势动作两点之间最小距离
//定义图片数组
private int[] images = new int[]{
R.drawable.img01, R.drawable.img02, R.drawable.img03,
R.drawable.img04, R.drawable.img05, R.drawable.img06,
R.drawable.img07, R.drawable.img08, R.drawable.img09,
};
```

（4）在 onCreate() 方法中，首先创建手势检测器，然后为 ViewFlipper 组件加载需要切换的图片，最后进行动画数组的初始化工作，关键代码如下：

```
detector = new GestureDetector(this, this);          //创建手势检测器
flipper = (ViewFlipper) findViewById(R.id.flipper);  //获取ViewFlipper
for (int i = 0; i < images.length; i++) {
    ImageView imageView = new ImageView(this);
    imageView.setImageResource(images[i]);
    flipper.addView(imageView);                      //加载图片
}
//初始化动画数组
animation[0] = AnimationUtils.loadAnimation(this, R.anim.anim_alpha_in); //淡入动画
animation[1] = AnimationUtils.loadAnimation(this, R.anim.anim_alpha_out);//淡出动画
```

（5）在重写的 onFling() 方法中，通过判断手指滑动方向指定切换的动画效果，关键代码如下：

```
@Override
public boolean onFling
              (MotionEvent motionEvent, MotionEvent motionEvent1, float v, float v1) {
    //为ViewFlipper设置切换的动画效果
    flipper.setInAnimation(animation[0]);
    flipper.setOutAnimation(animation[1]);
    //如果第一个触点事件的X坐标到第二个触点事件的X坐标的距离超过distance就是从右向左滑动
    if (motionEvent.getX() - motionEvent1.getX() > distance) {
        flipper.showPrevious();
```

```
            return true;
        //如果第二个触点事件的X坐标到第一个触点事件的X坐标的距离超过distance就是从左向右滑动
        } else if (motionEvent1.getX() - motionEvent.getX() > distance) {
            flipper.showNext();
            return true;
        }
        return false;
    }
```

（6）重写 onTouchEvent() 方法，在该方法中实现将当前 Activity 上的触碰事件交给 GestureDetector 处理。具体代码如下：

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    //将当前Activity上的触碰事件交给GestureDetector处理
    return detector.onTouchEvent(event);
}
```

（7）运行本实例，通过手指左右滑动模拟器屏幕，将实现图片的切换，并带有淡入淡出动画效果，如图 17.3 所示。

图 17.3　淡入、淡出动画

17.3 属性动画

属性动画是补间动画的升级版，它允许视图的所有属性都能实现渐变的动画效果，可以完成补间动画所不能完成的任务。例如，背景的颜色、文字的颜色、文字的大小等。

17.3.1 属性动画的使用

ObjectAnimator 类是属性动画的专属类，该类的常用方法如下：

1. ofFloat() 方法

该方法用于构造并返回 1 个在浮点值之间进行动画的属性动画。语法格式有以下 4 种：

语法一：

```
ofFloat (Object target, String propertyName, float... values)
```

语法二：

```
ofFloat (T target, Property<T, Float> property, float... values)
```

语法三：

```
ofFloat (Object target, String xPropertyName, String yPropertyName, Path path)
```

语法四：

```
ofFloat (T target, Property<T, Float> xProperty, Property<T, Float> yProperty, Path path)
```

参数含义如表 17.6 所示。

表 17.6　参数含义

参 数 名 称	描　　述
target	指定该动画使用在哪个控件
propertyName	指定动画的属性名称，如平移动画、旋转动画等
xPropertyName	x 坐标动画的属性名称，如平移动画、旋转动画等
yPropertyName	y 坐标动画的属性名称，如平移动画、旋转动画等
values	可变长的参数，表示动画值的范围
property	动画属性
path	动画的路径
xProperty	动画属性的 x 坐标
yProperty	动画属性的 y 坐标

2. ofInt() 方法

该方法用于构造并返回 1 个在 int 值之间进行动画的属性动画，参数说明参照 ofFloat() 方法即可。语法格式有以下 4 种：

语法一：

```
ofInt (Object target, String propertyName, int... values)
```

语法二：

```
ofInt (T target, Property<T, Integer> property, int... values)
```

语法三：

```
ofInt (Object target, String xPropertyName, String yPropertyName, Path path)
```

语法四：

```
ofInt (T target, Property<T, Integer> xProperty, Property<T, Integer> yProperty, Path path)
```

3. ofObject() 方法

该方法用于构造并返回 1 个在对象值之间进行动画的属性动画，相同的参数说明可以参照 ofFloat() 方法即可。语法格式有以下 4 种：

语法一：

```
ofObject (T target, Property<T, V> property, TypeEvaluator<V> evaluator, V... values)
```

语法二：

```
ofObject (T target, Property<T, V> property, TypeConverter<PointF, V> converter, Path path)
```

语法三：

```
ofObject (Object target, String propertyName, TypeConverter<PointF, ?> converter, Path path)
```

语法四：

```
ofObject (T target, Property<T, P> property, TypeConverter<V, P> converter,
TypeEvaluator<V> evaluator, V... values)
```

参数含义如表 17.7 所示。

表 17.7 参数含义

参 数 名 称	描 述
evaluator	插值器
converter	类型转换

4. ofArgb() 方法

该方法用于构造并返回 1 个在颜色值之间进行动画的属性动画，参数说明可以参照 ofFloat() 方法即可。语法格式有以下两种：

语法一：

```
ofArgb (Object target, String propertyName, int... values)
```

语法二：

```
ofArgb (T target, Property<T, Integer> property, int... values)
```

5. addListener() 方法

该方法用于添加动画监听器，使用该方法需要实现 Animator.AnimatorListener 接口与接口中的 4 个方法。

◆ onAnimationStart：动画开始播放时触发。
◆ onAnimationEnd：动画结束播放时触发。
◆ onAnimationCancel：动画取消播放时触发。
◆ onAnimationRepeat：动画重复播放时触发。

6. removeListener() 方法

该方法用于移除指定的动画监听器。

7. removeAllListeners() 方法

该方法用于移除所有的动画监听器。

ObjectAnimator 常用的属性方法如表 17.8 所示。

表 17.8 属性方法

方 法 名 称	描 述
start()	开始播放动画
end()	结束播放动画
cancel()	取消播放动画
pause()	暂停播放动画
resume()	恢复播放动画
reverse()	反方向播放动画
setEvaluator()	设置估值器
setInterpolator()	设置插值器
setRepeatMode()	设置重复模式，ValueAnimator.RESTART 为重新开始播放、ValueAnimator.REVERSE 为反方向播放
setRepeatCount()	设置重复的次数
setDuration()	设置动画持续的时间，单位为毫秒

下面通过一个实例演示 ObjectAnimator 属性动画的具体应用。

例 17.3 来回翻转的图片

在 Android Studio 中创建 Module，名称为“ObjectAnimator”，具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并将默认添加的 TextView 组件删除，然后添加 1 个 ImageView 组件用于显示来回翻转的图片。具体代码如下：

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

```
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示翻转的图片-->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"
        android:background="@drawable/img1" />
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 实现 View.OnClickListener 与 Animator.AnimatorListener 接口并实现两个接口中相对应的方法，然后定义所需的全局变量。修改后的具体代码如下：

```
public class MainActivity extends AppCompatActivity
        implements View.OnClickListener, Animator.AnimatorListener {
    private ImageView imageView;                    //定义图像组件
    private ObjectAnimator objectAnimator;          //定义翻转动画
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //动画开始播放时触发
    @Override
    public void onAnimationStart(Animator animation) {
    }
    //动画结束播放时触发
    @Override
    public void onAnimationEnd(Animator animation) {
    }
    //动画取消播放时触发
    @Override
    public void onAnimationCancel(Animator animation) {
    }
    //动画重复播放时触发
    @Override
    public void onAnimationRepeat(Animator animation) {
    }
    //图像组件的单击事件
    @Override
    public void onClick(View v) {
    }
}
```

（3）在 onCreate() 方法中，首先获取用于显示翻转图片的 ImageView 组件并为其设置单击事

件监听器，然后设置沿着 X 轴纵向翻转的属性动画并为其添加动画监听器。关键代码如下：

```
imageView = (ImageView) findViewById(R.id.imageView); //获取先翻转的图像组件
imageView.setOnClickListener(this);                   //设置单击事件监听器
//设置沿着X轴纵向先翻转动画
objectAnimator = ObjectAnimator.ofFloat(imageView, "rotationX", 0, -180);
objectAnimator.addListener(this);                     //为先翻转动画设置监听器
```

（4）在 onClick() 方法中，首先通过 switch() 方法判断单击显示翻转图片的 ImageView 组件时，启动先翻转的属性动画。关键代码如下：

```
switch (v.getId()) {    //判断单击组件的id
    case R.id.imageView:
        //启动翻转动画
        objectAnimator.setDuration(1000).start();
}
```

（5）在 onAnimationEnd() 方法中，创建 1 个沿着 X 轴纵向后翻转的属性动画并启动该动画，用于实现先翻转的动画结束播放时启动后翻转的属性动画。关键代码如下：

```
//启动后翻转动画
objectAnimator.ofFloat(imageView, "rotationX", -180, 0).setDuration(1000).start();
```

（6）运行本实例，将显示如图 17.4 所示的效果，单击图片后，图片将执行先翻转动画，如图 17.5 所示。先翻转动画结束后，将自动执行后翻转动画，如图 17.6 所示。

图 17.4　默认的运行效果

图 17.5　先翻转动画

图 17.6　后翻转动画

17.3.2 属性动画组合

在补间动画当中可以通过 AnimationSet 来组合多种动画效果，属性动画同样也可以实现多种动画的组合，它就是 AnimatorSet 类，该类与 ObjectAnimator 都是 Animator 的子类，但是在使用的方法却有些不同。AnimatorSet 提供了 5 种方法用于集合动画的创建，语法格式如下：

语法一：

```
play(Animator anim)
```

该方法是指添加一个动画，并返回 AnimatorSet.Builder 对象，通过该 Builder 对象实现组合动画播放顺序的方法。具体方法如下：

◆ after：指定该动画在当前动画之后播放。例如，play(A 动画).after(B 动画)，该播放顺序为 B 动画先播放。

◆ before：指定该动画在当前动画之前播放。例如，play(A 动画).before(B 动画)，该播放顺序为 A 动画先播放。

◆ with：指定该动画与当前动画同时播放。例如，play(A 动画).with(B 动画)，该播放顺序为两种动画同时播放。

语法二：

```
playSequentially(Animator... items)
```

该方法是指添加一组动画并根据添加的顺序进行动画的播放。

语法三：

```
playSequentially(List<Animator> items)
```

该方法是指添加一组动画并根据添加的顺序进行动画的播放。

语法四：

```
playTogether(Animator... items)
```

该方法是指添加一组动画并同时播放整组动画。

语法五：

```
playTogether(Collection<Animator> items)
```

该方法是指添加一组动画并同时播放整组动画。

下面通过一个实例演示 AnimatorSet 属性动画组合的具体应用。

例 17.4 四处走动的红方块

在 Android Studio 中创建 Module，名称为“AnimatorSet”，具体步骤如下：

（1）修改新建 Module 的 res\layout 目录下布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并将默认添加的 TextView 组件删除，然后添加 1 个 ImageView 组件用于显示红色的方块。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
```

```
    <!--显示红色方块的图像组件-->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:background="#FF0000" />
</RelativeLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 实现 View.OnClickListener 接口并实现接口中相对应的方法，然后定义所需的全局变量。修改后的具体代码如下：

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private ImageView imageView;                //定义图像组件
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //单击事件监听器
    @Override
    public void onClick(View v) {
    }
}
```

（3）在 onCreate() 方法中，首先获取用于显示红色方块的 ImageView 组件并为其设置单击事件监听器。关键代码如下：

```
imageView = (ImageView) findViewById(R.id.imageView); //获取显示红色方块的图像组件
imageView.setOnClickListener(this);          //为显示红色方块的图像组件设置单击事件
```

（4）在 onClick() 方法中，首先通过 switch() 方法判断单击显示红色方块的 ImageView 组件时，启动属性动画组合。关键代码如下：

```
switch (v.getId()) {    //判断单击组件的id
    case R.id.imageView:
        //创建平移动画1
        ObjectAnimator translationAnimator1 = ObjectAnimator.
                ofFloat(imageView, "translationX", 0, 260);
        //创建旋转动画
        ObjectAnimator rotateAnimator = ObjectAnimator.
                ofFloat(imageView, "rotation", 0, 360, 360, 0);
        //创建平移动画2
        ObjectAnimator translationAnimator2 = ObjectAnimator.
```

```
                ofFloat(imageView, "translationY", 0, 340);
        //创建渐变动画
        ObjectAnimator alphaAnimator= ObjectAnimator.
                ofFloat(imageView, "alpha", 1f, 0.1f, 0.1f, 1f);
        //创建平移动画3
        ObjectAnimator translationAnimator3 = ObjectAnimator.
                ofFloat(imageView, "TranslationX", 260, 0);
        //创建缩放动画
        ObjectAnimator scaleAnimator= ObjectAnimator.
                ofFloat(imageView, "scaleY", 1f, 0.5f, 1f);
        //创建平移动画4
        ObjectAnimator translationAnimator4 = ObjectAnimator.
                ofFloat(imageView, "TranslationY", 340, 0);
        //创建动画组合
        AnimatorSet animatorSet = new AnimatorSet();
        //设置动画顺序播放
        animatorSet.playSequentially(translationAnimator1, rotateAnimator,
                translationAnimator2,alphaAnimator,
                translationAnimator3,scaleAnimator,
                translationAnimator4
        );
        animatorSet.setDuration(1000);        //设置每个动画持续1秒
        animatorSet.start();                  //启动属性动画组
}
```

（5）运行本实例，单击红色方块播放向右平移动画，然后播放旋转动画如图 17.7 所示；旋转动画结束后红色方块向下平移；然后播放渐变动画如图 17.8 所示；渐变动画结束后红色方块向左平移；然后播放缩放动画，如图 17.9 所示，缩放动画结束后红色方块向上平移回到起点位置。

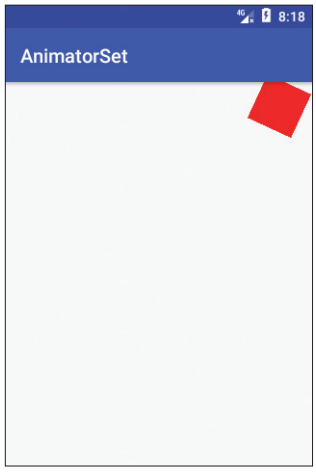


图 17.7　向右平移后旋转动画

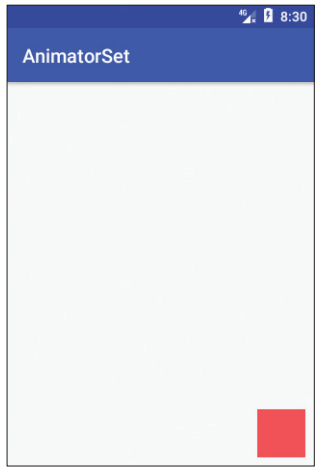


图 17.8　向下平移后渐变动画

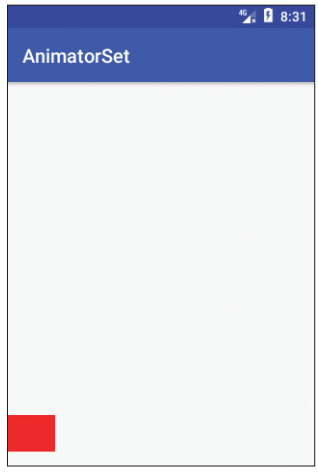
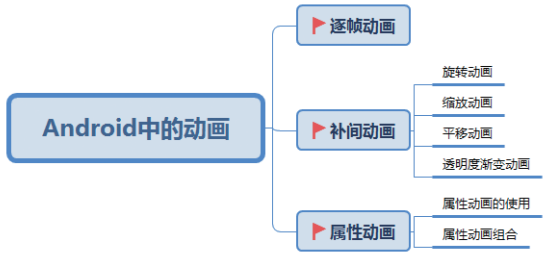


图 17.9　向左平移后缩放动画

17.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 18 章

播放音频与视频

本章介绍 Android 中的多媒体技术，主要包括如何使用 MediaPlayer 与 SoundPool 实现音频的播放以及如何使用视频相关的组件实现视频播放器。并且通过已经掌握的基础技术实现动态显示歌词的音乐播放器与添加视频弹幕的小程序。

18.1 播放音频

Android 提供了对常用音频格式和视频格式的支持，它所支持的音频格式有 MP3（.mp3）、3GPP（.3gp）、Ogg（.ogg）和 WAVE（.wav）等，支持的视频格式有 3GPP（.3gp）和 MPEG-4（.mp4）等。通过 Android API 提供的相关方法，可以在 Android 中实现音频与视频的播放。下面将分别介绍播放音频与视频的方法。

18.1.1 使用 MediaPlayer 播放音频

在 Android 中，提供了 MediaPlayer 类用来播放音频。使用 MediaPlayer 类播放音频比较简单，只需要创建该类的对象，并为其指定要播放的音频文件，然后调用该类的 start() 方法播放即可。MediaPlayer 类中有许多方法，其中比较常用的方法及其描述如表 18.1 所示。

表 18.1　MediaPlayer 类中的常用方法

方　法	描　述
create(Context context, int resid)	根据指定的资源 ID 创建一个 MediaPlayer 对象
create(Context context, Uri uri)	根据指定的 URI 创建一个 MediaPlayer 对象
setDataSource()	指定要装载的资源
prepare()	准备播放（在播放前调用）
start()	开始播放
stop()	停止播放
pause()	暂停播放
reset()	恢复 MediaPlayer 到未初始化状态

下面对如何使用 MediaPlayer 播放音频进行详细介绍。

1. 创建 MediaPlayer 对象，并装载音频文件

创建 MediaPlayer 对象并装载音频文件，可以使用 MediaPayer 类提供的静态方法 create() 来实现，也可以通过其无参的构造方法来创建并实例化该类的对象来实现。

◆ 使用 create() 方法创建 MediaPlayer 对象并装载音频文件

MediaPlayer 类的静态方法 create() 常用的语法格式有以下两种。

◎ create(Context context, int resid)

用于从资源 ID 所对应的资源文件中装载音频，并返回新创建的 MediaPlayer 对象。例如，要创建装载音频资源（res/raw/d.wav）的 MediaPlayer 对象，可以使用下面的代码：

```
MediaPlayer player=MediaPlayer.create(this, R.raw.d);
```

◎ create(Context context, Uri uri)

用于根据指定的 URI 来装载音频，并返回新创建的 MediaPlayer 对象。例如，要创建装载了音频文件（URI 地址为 http://www.mingribook.com/sound/bg.mp3）的 MediaPlayer 对象，可以使用下面的代码：

```
MediaPlayer player=MediaPlayer.create(this, Uri.parse("http://www.mingribook.com/sound/bg.mp3"));
```

说明 在访问网络中的资源时，要在AndroidManifest.xml文件中授予该程序访问网络的权限，具体的授权代码如下：

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

◆ 通过无参的构造方法来创建 MediaPlayer 对象并装载音频文件

使用无参的构造方法来创建 MediaPlayer 对象时，需要单独指定要装载的资源，可以使用 MediaPlayer 类的 setDataSource() 方法实现。

在使用 setDataSource() 方法装载音频文件后，实际上 MediaPlayer 并未真正装载该音频文件，还需要调用 MediaPlayer 的 prepare() 方法去真正装载音频文件。使用无参的构造方法来创建 MediaPlayer 对象并装载指定的音频文件，可以使用下面的代码：

```
01  MediaPlayer player1=new MediaPlayer();
02  try {
03      player1.setDataSource("/sdcard/music.mp3");          //指定要装载的音频文件
04      player1.prepare();                                   //预加载音频
05  } catch (IOException e) {
06      e.printStackTrace();
07  }
```

说明 通过MediaPlayer类的静态方法create()来创建MediaPlayer对象时，已经装载了要播放的音频，所以这种方法适用于播放单独的音频文件时。而通过无参的构造方法来创建MediaPlayer对象并装载音频文件时，可以根据需要来随时改变要加载的文件，所以这种方法适用于连续播放多个文件时。

2. 开始或恢复播放

在获取到 MediaPlayer 对象后，就可以使用 MediaPlayer 类提供的 start() 方法来开始播放音频或恢复播放已经暂停的音频。例如，已经创建了一个名称为 player 的对象，并且装载了要播放的音频，可以使用下面的代码播放该音频：

```
player.start();                    //开始播放
```

3. 停止播放

使用 MediaPlayer 类提供 stop() 方法可以停止正在播放的音频。例如，已经创建了一个名称为 player 的对象，并且已经开始播放装载的音频，可以使用下面的代码停止播放该音频：

```
player.stop();                     //停止播放
```

4. 暂停播放

使用 MediaPlayer 类提供的 pause() 方法可以暂停正在播放的音频。例如，已经创建了一个名称为 player 的对象，并且已经开始播放装载的音频，可以使用下面的代码暂停播放该音频：

```
player.pause();                    //暂停播放
```

例 18.1 简易音乐播放器

在 Android Studio 中创建一个 Module，名称为“Music Player”。实现本实例的具体步骤如下：

（1）将要播放的音频文件上传到 SD 卡的根目录中，这里要播放的音频文件为 music.mp3。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，并将 TextView 组件删除，然后为相对布局管理器添加背景图片，最后添加两个 ImageButton 组件，作为“播放”“暂停”“停止”按钮。

（3）打开 MainActivity 类，该类继承 Activity，然后在该类中，定义所需的成员变量，具体代码如下：

```
01  private MediaPlayer player;              //定义MediaPlayer对象
02  private boolean isPause = false;         //定义是否暂停
03  private File file;                       //定义要播放的音频文件
```

（4）在 MainActivity 类的 onCreate() 方法中，获取布局文件中的相关组件与要播放的音频文件，关键代码如下：

```
01  //设置全屏显示
02  getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
03          WindowManager.LayoutParams.FLAG_FULLSCREEN);
04  //获取"播放/暂停"按钮
05  final ImageButton btn_play = (ImageButton) findViewById(R.id.btn_play);
06  //获取"停止"按钮
07  final ImageButton btn_stop = (ImageButton) findViewById(R.id.btn_stop);
08  file = new File("/sdcard/music.mp3");     //获取要播放的音频文件
```

（5）如果音频文件存在，就会创建一个装载该文件的 MediaPlayer 对象；如果不存在则做出提示。关键代码如下：

```
01  if (file.exists()) {                     //如果音频文件存在
02      //创建MediaPlayer对象,并解析要播放的音频文件
03      player = MediaPlayer.create(this, Uri.parse(file.getAbsolutePath()));
04  } else {
```

```
05      //提示音频文件不存在
06      Toast.makeText(MainActivity.this,
07              "要播放的音频文件不存在！", Toast.LENGTH_SHORT).show();
08      return;
09  }
```

（6）在 MainActivity 类中，创建 play() 方法，实现重新播放音频的功能，具体代码如下：

```
01  private void play() {                                    //实现音频播放功能
02      try {
03          player.reset();                                  //重置MediaPlayer对象
04          player.setDataSource(file.getAbsolutePath());    //重新设置要播放的音频
05          player.prepare();                                //预加载音频
06          player.start();                                  //开始播放
07      } catch (Exception e) {
08          e.printStackTrace();                             //输出异常信息
09      }
10  }
```

（7）在 onCreate() 方法中，为 MediaPlayer 对象添加完成事件监听器，用于当音频播放完毕后，重新开始播放音频，关键代码如下：

```
01  //为MediaPlayer添加完成事件监听器，实现当音频播放完毕后，重新开始播放音频
02  player.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
03      @Override
04      public void onCompletion(MediaPlayer mp) {
05          play();                                  //调用play()方法，实现播放功能
06      }
07  });
```

（8）为播放按钮添加单击事件监听器，实现继续播放与暂停播放，关键代码如下：

```
01  btn_play.setOnClickListener(new View.OnClickListener() {//实现继续播放与暂停播放
02      @Override
03      public void onClick(View v) {
04          if (player.isPlaying() && !isPause) {            //如果音频处于播放状态
05              player.pause();                              //暂停播放
06              isPause = true;                              //设置为暂停状态
07              //更换为播放图标
08              ((ImageButton) v).setImageDrawable(getResources()
09                      .getDrawable(R.drawable.play, null));
10          } else {
11              player.start();                              //继续播放
12              //更换为暂停图标
13              ((ImageButton) v).setImageDrawable(getResources()
14                      .getDrawable(R.drawable.pause, null));
15              isPause = false;                             //设置为播放状态
16          }
17      }
18  });
```

（9）为停止按钮添加单击事件监听器，实现停止播放音频，关键代码如下：

```
btn_stop.setOnClickListener(new View.OnClickListener() {//单击停止按钮，实现停止播放音频
    @Override
    public void onClick(View v) {
        player.stop();                                    //停止播放
        //更换为播放图标
        btn_play.setImageDrawable(getResources()
                .getDrawable(R.drawable.play, null));
    }
});
```

（10）重写 Activity 的 onDestroy() 方法，用于在当前 Activity 销毁时，停止正在播放的音频，并释放 MediaPlayer 所占用的资源，具体代码如下：

```
@Override
protected void onDestroy() {
    if (player.isPlaying()) {                 //如果音频处于播放状态
        player.stop();                        //停止音频的播放
    }
    player.release();                         //释放资源
    super.onDestroy();
}
```

（11）从 Android 4.4.2 开始，如果想要访问 SD 卡的文件，就需要在 AndroidManifest.xml 文件中赋予程序访问 SD 卡的权限，关键代码如下：

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

说明 本章实例第一次运行将会出现空指针异常，无法获取文件路径。用户可以在模拟器中进入→“设置”→“应用和通知”→“你的App应用名称”→“权限”界面，然后开启存储空间权限，即可获取文件指定路径。

（12）运行本实例，将显示如图 18.1 所示的界面。

图 18.1　简易音乐播放器

18.1.2 使用 SoundPool 播放音频

Android 还提供了另一个播放音频的类——SoundPool（即音频池），可以同时播放多个短小的音频，而且占用的资源较少。使用 SoundPool 播放音频，首先需要创建 SoundPool 对象，然后加载所要播放的音频，最后调用 play() 方法播放音频，下面进行详细介绍。

1．创建 SoundPool 对象

SoundPool 类提供了一个构造方法，用来创建 SoundPool 对象，该构造方法的语法格式如下：

```
SoundPool (int maxStreams, int streamType, int srcQuality)
```

参数说明如下：

◆ maxStreams：用于指定可以容纳多少个音频。

◆ streamType：用于指定声音类型，可以通过 AudioManager 类提供的常量进行指定，通常使用 STREAM_MUSIC。

◆ srcQuality：用于指定音频的品质，默认值为 0。

例如，创建一个可以容纳 10 个音频流的 SoundPool 对象，代码如下：

```
01  //创建一个SoundPool对象，该对象可以容纳10个音频流
02          SoundPool soundpool = new SoundPool(10,
03                  AudioManager.STREAM_SYSTEM, 0);
```

2．加载所要播放的音频

创建 SoundPool 对象后，可以调用 load() 方法来加载要播放的音频。load() 方法的语法格式有以下 4 种。

◆ public int load (Context context, int resId, int priority)：用于通过指定的资源 ID 来加载音频。

◆ public int load (String path, int priority)：用于通过音频文件的路径来加载音频。

◆ public int load (AssetFileDescriptor afd, int priority)：用于从 AssetFileDescriptor 所对应的文件中加载音频。

◆ public int load (FileDescriptor fd, long offset, long length, int priority)：用于加载 FileDescriptor 对象中从 offset 开始、长度为 length 的音频。

例如，要通过资源 ID 来加载音频文件 ding.wav，可以使用下面的代码：

```
soundpool.load(this, R.raw.ding, 1);
```

说明 为了更好地管理所加载的每个音频，一般使用HashMap<Integer, Integer>对象来管理这些音频。这时可以先创建一个HashMap<Integer, Integer>对象，然后应用该对象的put()方法将加载的音频保存到该对象中。例如，创建一个HashMap<Integer, Integer>对象，并应用put()方法添加一个音频，可以使用下面的代码:

```
01  //创建一个HashMap对象
02  HashMap<Integer, Integer> soundmap = new HashMap<Integer, Integer>();
03  soundmap.put(1, soundpool.load(this, R.raw.chimes, 1));
```

3．播放音频

调用 SoundPool 对象的 play() 方法可播放指定的音频。play() 方法的语法格式如下：

```
play (int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
```

play() 方法各参数的说明如表 18.2 所示。

表 18.2 play() 方法的参数说明

参　数	描　述
soundID	指定要播放的音频，该音频为通过 load() 方法返回的音频
leftVolume	指定左声道的音量，取值范围为 0.0~1.0
rightVolume	指定右声道的音量，取值范围为 0.0~1.0
priority	指定播放音频的优先级，数值越大，优先级越高
loop	指定循环次数，0 为不循环，-1 为循环
rate	指定速率，正常为 1，最低为 0.5，最高为 2

例如，要播放 raw 资源中保存的音频文件 notify.wav，可以使用下面的代码：

```
01  //播放指定的音频
02  soundPool.play(soundpool.load(MainActivity.this, R.raw.notify, 1), 1, 1, 0, 0, 1);
```

例 18.2 模拟手机选择铃声

在 Android Studio 中创建一个 SDK 最小版本为 21 的 Module，名称为“Select Ringtone”，实现本实例的具体步骤如下：

（1）在 res 目录下创建 raw 资源文件夹，将要播放的音频文件复制到该文件夹当中。

（2）修改布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并将 TextView 组件删除，然后添加一个 ListView 组件，用于显示要选择的铃声。

（3）在 res/layout 目录下，创建一个名称为 main.xml 文件，用于指定 ListView 列表项的布局样式。

（4）打开 MainActivity 类，在 onCreate() 方法中，通过 for 循环将列表项文字保存到 Map 中，并添加到 list 集合中，关键代码如下：

```
01  ListView listview = (ListView) findViewById(R.id.listView);  //获取列表视图
02  String[] title = new String[]{"布谷鸟叫声", "风铃声", "门铃声", "电话声", "鸟叫声",
03          "水流声", "公鸡叫声"};                    //定义并初始化保存列表项文字的数组
04  //创建一个list集合
05  List<Map<String, Object>> listItems = new ArrayList<Map<String, Object>>();
06  //通过for循环将列表项文字保存到Map中，并添加到List集合中
07  for (int i = 0; i < title.length; i++) {
08      Map<String, Object> map = new HashMap<String, Object>();//实例化Map对象
09      map.put("name", title[i]);
10      listItems.add(map);                          //将map对象添加到List集合中
11  }
```

（5）创建 AudioAttributes 对象，然后设置场景与音效的相关属性，关键代码如下：

```
01  AudioAttributes attr = new AudioAttributes.Builder()          //设置音效的相关属性
02          .setUsage(AudioAttributes.USAGE_GAME)                 //设置音效的使用场景
03          .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC) //设置音效的类型
04          .build();
```

```
final SoundPool soundpool = new SoundPool.Builder()          //创建SoundPool对象
        .setAudioAttributes(attr)                            //设置音效池的属性
        .setMaxStreams(10)                                   //设置最多可容纳10个音频流
        .build();
```

（6）创建一个 HashMap 对象，将要播放的音频流保存到 HashMap 对象中，关键代码如下：

```
final HashMap<Integer, Integer> soundmap = new HashMap<Integer, Integer>();
soundmap.put(0, soundpool.load(this, R.raw.cuckoo, 1));
soundmap.put(1, soundpool.load(this, R.raw.chimes, 1));
soundmap.put(2, soundpool.load(this, R.raw.notify, 1));
soundmap.put(3, soundpool.load(this, R.raw.ringout, 1));
soundmap.put(4, soundpool.load(this, R.raw.bird, 1));
soundmap.put(5, soundpool.load(this, R.raw.water, 1));
soundmap.put(6, soundpool.load(this, R.raw.cock, 1));
```

（7）创建 SimpleAdapter 适配器并将适配器与 ListView 关联，关键代码如下：

```
SimpleAdapter adapter = new SimpleAdapter(this, listItems,
        R.layout.main, new String[]{"name",}, new int[]{
        R.id.title});                    //创建SimpleAdapter
listview.setAdapter(adapter);            //将适配器与ListView关联
```

（8）为 ListView 设置事件监听器，为每个选项设置所要播放的音频，关键代码如下：

```
listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        //获取选项的值
        Map<String, Object> map = (Map<String, Object>) parent.getItemAtPosition(position);
        soundpool.play(soundmap.get(position), 1, 1, 0, 0, 1);//播放所选音频
    }
});
```

（9）运行本实例，将显示如图 18.2 所示的界面，单击任一列表项，即可播放对应的铃声。

图 18.2　通过 SoundPool 模拟手机选择铃声

18.1.3 动态显示歌词的音乐播放器

在掌握了 MediaPlayer 的使用方法以后，通过 MediaPlayer 技术实现一个可以动态显示歌词的音乐播放器，例如 QQ 音乐、酷狗音乐等。

其功能可以通过本地扫描的方式将手机中的音乐以列表的形式进行展示，如图 18.3 所示；在列表中单击某音乐进行该音乐的播放，并且动态显示当前音乐的歌词如图 18.4 所示。

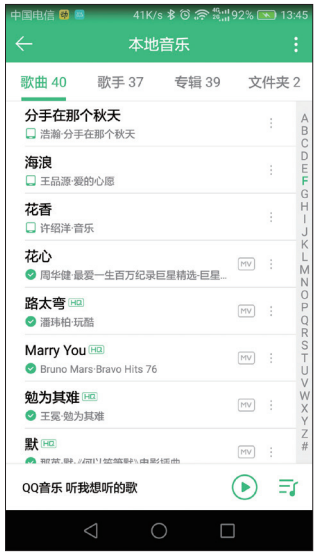


图 18.3 显示本地音乐列表

图 18.4 播放音乐并动态显示歌词

例 18.3 动态显示歌词的音乐播放器

1. 开发准备

打开 Android Studio 开发工具，新建一个名称为“MusicPlayer”。项目创建完成后，打开 build.gradle（Module app）文件，由于该项目需要使用第三方技术的支持，所以需要在 build.gradle（Module app）文件中添加依赖包与相关配置。具体步骤如下：

（1）添加 ButterKnife 依赖包。ButterKnife 是一个专注于 Android 系统 View 注入框架，可以一键生成 findViewById 以及 setOnClickListener 代码，减少大量的绑定控件与绑定事件的代码。该技术不仅可以通过插件的方式安装在开发工具中，也可以通过添加依赖包的形式使用该技术。依赖代码如下：

```
01  compile 'com.jakewharton:butterknife:8.4.0'
02  annotationProcessor 'com.jakewharton:butterknife-compiler:8.4.0'
```

（2）添加 LrcView 依赖包。LrcView 用于实现读取 lrc 歌词文件的控件，支持上下拖动歌词、歌词自动换行、自定义属性。该控件不断完善至今已有多个更新后的版本，由于本项目仅实现歌词跟随音乐自动换行的效果，所以这里使用 LrcView 的 1.3 版本即可。依赖代码如下：

```
compile 'me.wcy:lrcview:1.3.0'
```

（3）添加 EventBus 依赖包，EventBus 用于实现在 Android 的各个控件以及线程之间进行传递

消息，并且将事件的发送者与接收者之间进行解耦。依赖代码如下：

```
compile 'org.greenrobot:eventbus:3.0.0'
```

（4）添加 RecyclerView 依赖包，RecyclerView 用于实现歌曲列表，它并不是第三方开源技术，而是谷歌公司在 Android 5.0 之后推出的 1 个新控件。RecylerView 可以说是 ListView 和 GridView 的增强升级版，既可以实现纵向列表还可以实现网格列表以及瀑布流等列表效果。依赖代码如下：

```
implementation 'com.android.support:recyclerview-v7:26.1.0'
```

（5）添加支持 lambda 表达式的编译选项，由于本项目中使用了 lambda 表达式，并且当前 JDK 为 1.8 版本，所以需要添加编辑选项时标注 Java 版本即可。代码如下：

```
01  compileOptions {
02      sourceCompatibility JavaVersion.VERSION_1_8
03      targetCompatibility JavaVersion.VERSION_1_8
04  }
```

2. 实现过程

完成了基本的开发准备后，需要实现项目开发的过程。具体步骤如下：

（1）在 layout 目录下创建 4 个布局文件，分别为“title_layout”用于显示音乐播放器的标题栏、“list_item_layout”用于显示歌曲列表中的每一个子项布局、“bottom_layout”用于显示音乐播放器底部的控制布局、最后在默认生成的 activity_main 布局文件中添加用于显示歌曲列表的 RecyclerView 控件以及用于显示歌词的 LrcView 控件。关键代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:id="@+id/activity_main"
06      android:layout_width="match_parent"
07      android:layout_height="match_parent"
08      android:orientation="vertical"
09      tools:context="com.mingrisoft.musicplayer.avtivity.MusicActivity">
10      <!--包括标题栏布局-->
11      <include layout="@layout/title_layout" />
12
13      <RelativeLayout
14          android:layout_width="match_parent"
15          android:layout_height="0dp"
16          android:layout_weight="1">
17          <!--显示歌曲列表的控件-->
18          <android.support.v7.widget.RecyclerView
19              android:id="@+id/music_list"
20              android:layout_width="match_parent"
21              android:layout_height="match_parent"
22              android:background="@color/colorList" />
```

```
        <!--显示歌词的控件-->
        <me.wcy.lrcview.LrcView
            android:id="@+id/music_lrc"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/colorWindow"
            android:visibility="gone"
            app:lrcAnimationDuration="1000"
            app:lrcCurrentTextColor="@color/colorYellow"
            app:lrcDividerHeight="24dp"
            app:lrcLabel="找不到歌词(&gt;_&lt;)"
            app:lrcNormalTextColor="@android:color/darker_gray"
            app:lrcPadding="16dp"
            app:lrcTextSize="18sp" />
    </RelativeLayout>
    <!--包括底部控制布局-->
    <include layout="@layout/bottom_layout" />
</LinearLayout>
```

说明 由于页码的限制，这里只给出关键代码，其他代码请参照源码即可。

（2）在 MusicPlayer 项目中的“app\src\main\java”目录下创建 utils 包，该包用于保存本项目中所使用到的工具类文件，该包中包括 3 个类文件，其类名及作用如表 18.3 所示。

表 18.3 util 包中的三个类

类　名	作　用
History	用来读取轻量级数据内容
SearchFile	用来扫描歌曲文件和歌词文件
ToastUtils	弹出吐司的工具类

（3）在项目中的“app\src\main\java”目录下创建 activity 包，用来存放 Activity，将 MainActivity.java 文件放到 activity 包中。在 MainActivity 类中添加 initialization() 方法，用来执行界面的初始化操作。代码如下：

```
01 private void initialization() {
02     //扫描本地音乐
03     musics = SearchFile.searchLocaltion(this);
04     if (musicList != null) {
05         //绑定服务
06         Intent intent = new Intent(this, PlayMusicService.class);
07         intent.putParcelableArrayListExtra("MUSIC", musics);
08         bindService(intent, this, BIND_AUTO_CREATE);
09         initBottomBar();                          //初始化底部功能栏
10         setMusicList();                           //设置音乐列表
11         setBottomBar();                           //设置底部功能栏
12     } else {
```

```
13          ToastUtils.Short(this, "本地没有歌曲！");
14      }
15  }
```

（4）在 SearchFile 类中扫描手机中所有的音频，每扫描出一个音频，就创建一个 Music 对象，并将音频的歌名、歌手和文件路径存入创建的 Music 对象中，最后将 Music 对象存入数据集合中。代码如下：

```
01  public static ArrayList<Music> searchLocaltion(Context context){
02      ArrayList<Music> musicList = null;
03      //扫描本地的歌曲
04      Cursor cursor = context .getApplicationContext()
05                              .getContentResolver()
06                              .query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
07                                      null, null, null,null);
08      if(cursor!=null){
09          musicList=new ArrayList<>();                    //该集合用来存放歌曲的信息
10          while(cursor.moveToNext() ){                    //游标下移
11              Music music;
12              music=new Music();
13              //获取歌曲名称
14              String music_name=cursor.getString(
15                      cursor.getColumnIndex(MediaStore.Audio.Media.TITLE));
16              //获取歌手名称
17              String music_singer=cursor.getString(
18                      cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST));
19              //获取歌曲路径
20              String music_path=cursor.getString(
21                      cursor.getColumnIndex(MediaStore.Audio.Media.DATA));
22              //获取音乐ID
23              long music_id = cursor.getLong(cursor
24                      .getColumnIndex(MediaStore.Audio.Media._ID));
25              //获取专辑
26              String music_album = cursor.getString(cursor
27                      .getColumnIndex(MediaStore.Audio.Media.ALBUM));
28              //获取专辑ID
29              int music_albumId = cursor.getInt(cursor
30                      .getColumnIndex(MediaStore.Audio.Media.ALBUM_ID));
31              //是否为音乐
32              int isMusic = cursor.getInt(cursor
33                      .getColumnIndex(MediaStore.Audio.Media.IS_MUSIC));
34              if (isMusic != 0){                          //如果是音乐保存到集合中
35                  music.setMusic_name(music_name);
36                  music.setMusic_singer(music_singer);
37                  music.setMusic_path(music_path);
38                  music.setMusic_id(music_id);
39                  music.setMusic_album(music_album);
```

```
                music.setMusic_albumId(music_albumId);
                musicList.add(music);
            }
        }
    }
    cursor.close();              //关闭游标
    return musicList;
}
```

（5）创建 PlayMusicService 类用来控制音频的播放和停止。在该类中创建 playMusic() 方法，用来准备播放的资源，然后同步资源，这一步要完全遵循官网 API 的说明来设置，否则是无法播放音频的。代码如下：

```
private void playMusic(){
    try {
        mediaPlayer.reset();     //重置播放器
        //获取歌曲对象
        Music music = musicList.get(nowPlayPosition);
        //获取歌曲的路径
        String musicPath = music.getMusic_path();
        //设置播放歌曲
        mediaPlayer.setDataSource(musicPath);
        //字符串替换
        String lrcPath = musicPath.replace(".mp3",".lrc");
        //查找歌词
        lrcView.loadLrc(SearchFile.getLrcText(lrcPath));
        isPlaying = true;
        //用于更新歌词
        thread = new Thread(runnable);
        thread.start();
        //设置播放模式为播放音乐
        mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        //同步资源
        mediaPlayer.prepare();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（6）调用 MediaPlayer 的 start() 方法，用于实现同步音乐资源后播放音乐。代码如下：

```
@Override
public void onPrepared(MediaPlayer mp) {
    mp.start();
}
```

说明 本项目在运行前需要将lrc歌词文件保存在歌曲同一级目录下。

（7）运行本实例，将显示如图 18.5 所示的音乐列表，单击播放按钮将播放指定的音乐，并且将歌词显示在界面当中，如图 18.6 所示。

图 18.5　显示本地音乐列表

图 18.6　播放音乐并动态显示歌词

18.2 播放视频

18.2.1 使用 VideoView 播放视频

在 Android 中，提供了 VideoView 组件用于播放视频文件。要想使用 VideoView 组件播放视频，首先需要在布局文件中添加该组件，然后在 Activity 中获取该组件，并应用 setVideoPath() 方法或 setVideoURI() 方法加载要播放的视频，最后调用 start() 方法来播放视频。另外，VideoView 组件还提供了 stop() 和 pause() 方法，用于停止或暂停视频的播放。

在布局文件中添加 VideoView 组件的基本语法格式如下：

```
<VideoView
    属性列表>
</VideoView>
```

VideoView 组件支持的 XML 属性如表 18.4 所示。

表 18.4　VideoView 组件支持的 XML 属性

XML 属性	描　述
android:id	设置组件的 ID
android:background	设置背景，可以设置背景图片，也可以设置背景颜色
android:layout_gravity	设置对齐方式

续表

XML 属性	描　述
android:layout_width	设置宽度
android:layout_height	设置高度

在 Android 中还提供了一个可以与 VideoView 组件结合使用的 MediaController 组件。MediaController 组件用于通过图形控制界面来控制视频的播放。

下面通过一个具体的实例来说明如何使用 VideoView 和 MediaController 来播放视频。

例 18.4 使用 VideoView 和 MediaController 播放视频

在 Android Studio 中创建 Module，名称为“VideoView And MediaController”，实现本实例的具体步骤如下：

（1）将要播放的视频文件上传到 SD 卡的根目录中，这里要播放的视频文件为 video.mp4。

（2）修改布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，并将 TextView 组件删除，然后在默认的相对布局管理器中添加一个 VideoView 组件，用于播放视频文件，关键代码如下：

```
<VideoView
    android:id="@+id/video"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

（3）打开 MainActivity 类，该类继承 Activity，在 onCreate() 方法中，指定播放模拟器 SD 卡上的视频文件，并创建一个 android.widget.MediaController 对象，控制视频的播放。关键代码如下：

```
//设置全屏显示
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
VideoView video = (VideoView) findViewById(R.id.video); //获取VideoView组件
//指定播放模拟器SD卡上的视频文件
File file = new File(Environment.getExternalStorageDirectory() + "/video.mp4");
//创建android.widget.MediaController对象，控制视频的播放
MediaController mc = new MediaController(MainActivity.this);
```

（4）实现视频的播放功能，关键代码如下：

```
if (file.exists()) {                                    //判断要播放的视频文件是否存在
    video.setVideoPath(file.getAbsolutePath());         //指定要播放的视频
    video.setMediaController(mc);                  //设置VideoView与MediaController相关联
    video.requestFocus();                               //让VideoView获得焦点
    try {
        video.start();                                  //开始播放视频
    } catch (Exception e) {
        e.printStackTrace();                            //输出异常信息
    }
```

```
    //为VideoView添加完成事件监听器，实现视频播放结束后的提示信息
    video.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
        //弹出消息提示框显示播放完毕
        Toast.makeText(MainActivity.this, "视频播放完毕！", Toast.LENGTH_SHORT).show();
        }
    });
} else {
    //弹出消息提示框提示文件不存在
    Toast.makeText(this, "要播放的视频文件不存在", Toast.LENGTH_SHORT).show();
}
```

（5）从 Android 4.4.2 开始，如果想要访问 SD 卡上的文件，就需要在 AndroidManifest.xml 文件中赋予程序访问 SD 卡的权限，关键代码如下：

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

（6）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置为横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

说明 读者需要在模拟器中开启存储空间权限，方可获取sd卡指定路径。

（7）运行本实例，如图 18.7 所示。

图 18.7　使用 VideoView 和 MediaController 组件播放视频

18.2.2 使用 MediaPlayer 和 SurfaceView 播放视频

使用 MediaPlayer 除可以播放音频外，还可以播放视频文件，只不过使用 MediaPlayer 播放视频时，没有提供图像输出界面。这时，可以使用 SurfaceView 组件来显示视频图像。使用 MediaPlayer 和 SurfaceView 来播放视频，大致可以分为以下 4 个步骤：

（1）定义 SurfaceView 组件。定义 SurfaceView 组件可以在布局管理器中实现，也可以直接在 Java 代码中创建，不过推荐在布局管理器中定义 SurfaceView 组件，其基本语法格式如下：

```
<SurfaceView
    android:id="@+id/ID号"
    android:background="背景"
    android:keepScreenOn="true|false"
    android:layout_width="宽度"
    android:layout_height="高度"/>
```

在上面的语法中，android:keepScreenOn 属性用于指定在播放视频时，是否打开屏幕。

例如，在布局管理器中，添加一个 ID 号为 surfaceView1、设置了背景的 SurfaceView 组件，可以使用下面的代码：

```
<SurfaceView
    android:id="@+id/surfaceView1"
    android:background="@drawable/bg"
    android:keepScreenOn="true"
    android:layout_width="320dp"
    android:layout_height="36dp"/>
```

（2）创建 MediaPlayer 对象，并为其加载要播放的视频。与播放音频时创建 MediaPlayer 对象一样，也可以使用 MediaPlayer 类的静态方法 create() 和无参的构造方法两种方式创建 MediaPlayer 对象，具体方法请参见 18.1.1 小节。

（3）将所播放的视频画面输出到 SurfaceView。使用 MediaPlayer 对象的 setDisplay() 方法，可以将所播放的视频画面输出到 SurfaceView。setDisplay() 方法的语法格式如下：

```
setDisplay(SurfaceHolder sh)
```

其中，参数 sh 用于指定 SurfaceHolder 对象，可以通过 SurfaceView 对象的 getHolder() 方法获得。例如，为 MediaPlayer 对象指定输出视频画面的 SurfaceView，可以使用下面的代码：

```
mediaPlayer.setDisplay(surfaceview.getHolder());    //设置将视频画面输出到SurfaceView
```

（4）调用 MediaPlayer 对象的相应方法控制视频的播放。使用 MediaPlayer 对象提供的 play()、pause() 和 stop() 方法，可以控制视频的播放、暂停和停止。

下面通过一个具体的实例来说明如何使用 MediaPlayer 和 SurfaceView 来播放视频。

例 18.5 通过 MeidaPlayer 和 SurfaceView 播放视频

在 Android Studio 中创建 Module，名称为“MeidaPlayer And SurfaceView”，实现本实例的具体步骤如下：

（1）将要播放的视频文件上传到 SD 卡的根目录中，这里要播放的视频文件为 video.mp4。

（2）修改布局文件 activity_main.xml，首先将默认添加的相对布局管理器修改为垂直线性布局管理器，然后将默认添加的 TextView 组件，再添加一个 SurfaceView 组件，用于显示视频图像，并添加一个水平线性布局管理器，最后在水平线性布局管理器中分别添加播放按钮、暂停按钮和停止按钮。具体代码请参见云盘。

（3）打开 MainActivity 类，该类继承 Activity，然后在该类中，定义所需的成员变量，具体代码如下：

```
01  private ImageButton play, pause, stop;  //定义播放、暂停和停止按钮
02  private MediaPlayer mediaPlayer;        //定义MediaPlayer对象
03  private SurfaceHolder surfaceHolder;    //定义SurfaceHolder对象
04  private boolean noPlay = true;          //定义播放状态
```

（4）在 MainActivity 类的 onCreate() 方法中，首先获取 SurfaceView 组件与 SurfaceHolder 对象，然后创建 MediaPlayer 对象并设置多媒体类型，关键代码如下：

```
01  //设置全屏显示
02  getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
03          WindowManager.LayoutParams.FLAG_FULLSCREEN);
04  play = (ImageButton) findViewById(R.id.play);                    //获取播放按钮对象
05  pause = (ImageButton) findViewById(R.id.pause);                  //获取暂停按钮对象
06  stop = (ImageButton) findViewById(R.id.stop);                    //获取停止按钮对象
07  //获取SurfaceView组件
08  SurfaceView surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
09  surfaceHolder = surfaceView.getHolder();                         //获取SurfaceHolder
10  pause.setEnabled(false);                                         //设置暂停按钮不可用
11  stop.setEnabled(false);                                          //设置停止按钮不可用
12  mediaPlayer = new MediaPlayer();                                 //创建MediaPlayer对象
13  mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);       //设置多媒体的类型
```

（5）在 MainActivity 类中，创建 play() 方法，在该方法中实现视频的播放功能，具体代码如下：

```
01  public void play() {  //创建play()方法，在该方法中实现视频的播放功能
02      mediaPlayer.reset();                        //重置MediaPlayer
03      mediaPlayer.setDisplay(surfaceHolder);      //把视频画面输出到SurfaceView
04      try {
05          //模拟器的SD卡上的视频文件
06          mediaPlayer.setDataSource(Environment.getExternalStorageDirectory()
07                  + "/video.mp4");
08          mediaPlayer.prepare();                  //预加载
09      } catch (Exception e) {                     //输出异常信息
10          e.printStackTrace();
11      }
12      mediaPlayer.start();                        //开始播放
13      pause.setEnabled(true);                     //设置"暂停"按钮可用
14      stop.setEnabled(true);                      //设置"停止"按钮可用
15  }
```

（6）在 onCreate() 方法中，单击播放按钮，实现视频的播放与继续播放功能，关键代码如下：

```
01  play.setOnClickListener(new View.OnClickListener() {          //实现播放与继续播放功能
02      @Override
03      public void onClick(View v) {
```

```
            if (noPlay) {                    //如果没有播放视频
                play();                      //调用play()方法实现播放功能
                noPlay = false;              //设置播放状态为正在播放
            } else {
                mediaPlayer.start();         //继续播放视频
            }
        }
    });
```

（7）单击暂停按钮，实现视频的暂停功能，关键代码如下：

```
pause.setOnClickListener(new View.OnClickListener() {  //实现暂停功能
    @Override
    public void onClick(View v) {
        if (mediaPlayer.isPlaying()) { //如果视频处于播放状态
            mediaPlayer.pause();       //暂停视频的播放
        }
    }
});
```

（8）单击停止按钮，实现视频的停止功能，关键代码如下：

```
stop.setOnClickListener(new View.OnClickListener() {  //实现停止功能
    @Override
    public void onClick(View v) {
        if (mediaPlayer.isPlaying()) { //如果视频处于播放状态
            mediaPlayer.stop();        //停止播放
            noPlay = true;             //设置播放状态为没有播放
            pause.setEnabled(false);   //设置"暂停"按钮不可用
            stop.setEnabled(false);    //设置"停止"按钮不可用
        }
    }
});
```

（9）为 MediaPlayer 对象添加完成事件监听器，实现当视频播放完以后给出相应提示，关键代码如下：

```
//为MediaPlayer对象添加完成事件监听器
mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        Toast.makeText(MainActivity.this, "视频播放完毕！", Toast.LENGTH_SHORT).show();
    }
});
```

（10）重写 Activity 的 onDestroy() 方法，用于在当前 Activity 销毁时，停止正在播放的视频，并释放 MediaPlayer 所占用的资源，具体代码如下：

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (mediaPlayer != null) {          //如果MediaPlayer不为空
        if (mediaPlayer.isPlaying()) {  //如果处于播放状态
            mediaPlayer.stop();         //停止播放视频
        }
       //Activity销毁时停止播放，释放资源。不做这个操作，即使退出还是能听到视频播放的声音
        mediaPlayer.release();
    }
}
```

（11）如果需要访问 SD 卡的文件，需要在 AndroidManifest.xml 文件中赋予程序访问 SD 卡的权限，关键代码如下：

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

（12）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（13）运行本实例，将显示如图 18.8 所示。

图 18.8　使用 MediaPlayer 和 SurfaceView 播放视频

说明　读者需要在模拟器中开启存储空间权限，即可获取sd卡指定路径。

从上面的开发过程可以看出，使用 MediaPlayer 播放视频文件要复杂一些，其优点是灵活性高，用户可以自定义开发控制按钮来控制视频播放；而使用 VideoView 来播放视频文件相对比较简单，但缺点是灵活性不高。因此，开发人员可以根据自己的需要来选择使用哪一种方式播放视频文件。

18.2.3 添加视频弹幕

通过上面的学习，已经掌握 Android 中可以播放视频的两种技术。本节将通过第三方技术实现

一个可以添加弹幕的视频播放器。例如哔哩哔哩 App 在播放视频时，顶部所显示的弹幕，如图 18.9 所示。并且在观看视频时，通过手指在屏幕两侧上下滑动来实现控制声音的大小与屏幕的亮度。

图 18.9　播放视频所显示的弹幕

例 18.6　添加视频弹幕

1.　开发准备

打开 Android Studio 开发工具，新建一个名称为“VideoBarrage”的 Android 项目。由于该项目需要使用哔哩哔哩开源弹幕框架 DanmakuFlameMaster，所以需要在 build.gradle（Module app）文件中添加依赖包，然后下载第三方视频播放器框架 Vitamio。具体步骤如下：

（1）添加 DanmakuFlameMaster 依赖包，通过该依赖包实现视频中所显示的弹幕，代码如下：

```
01  compile 'com.github.ctiao:DanmakuFlameMaster:0.9.16'
02  compile 'com.github.ctiao:ndkbitmap-armv7a:0.9.16'
```

注意 弹幕框架DanmakuFlameMaster（项目地址：https://github.com/Bilibili/DanmakuFlameMaster），根据该地址可以下载弹幕框架的项目源码，然后根据代码分析框架内方法与功能的应用。

（2）在 Vitamio 官方网站（https://www.vitamio.org/）中下载 Vitamio 演示代码，然后将代码中的 vitamio 模块导入开发环境中，再参考 build.gradle（Module app）文件修改 build.gradle（Module vitamio）文件如图 18.10 所示，最后将需要使用该框架的项目依赖于 vitamio 模块。

图 18.10　修改 build.gradle（Module vitamio）文件

说明 在修改build.gradle（Module vitamio）文件时，版本号需要根据当前开发环境与Android版本而修改。由于Vitamio演示代码的版本不断更新，vitamio模块可以在源码中进行复制。

2. 实现过程

完成了基本的开发准备后，需要实现项目开发的过程。具体步骤如下：

（1）在 layout 目录下创建 3 个布局文件，分别为“mine_mediacontroller_top”用于显示视频控制器顶部、“mine_mediacontroller_bottom”用于显示视频控制器底部、最后在默认生成的 activity_main 布局文件中添加用于播放视频的 Vitamio 第三方控件以及显示弹幕的控件。关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.mingrisoft.videobarrage.MainActivity">
    <FrameLayout
        android:id="@+id/home"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <!--第三方Vitamio播放视频控件-->
        <io.vov.vitamio.widget.VideoView
            android:id="@+id/video"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <!--显示弹幕控件-->
        <master.flame.danmaku.ui.widget.DanmakuView
            android:id="@+id/mdanmaku"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:layout_gravity="center_vertical"/>
        <!--显示声音与屏幕亮度的图标-->
        <ImageView
            android:id="@+id/hint"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@mipmap/sound"
            android:layout_gravity="center"
            android:background="@drawable/hint_shape"
            android:visibility="gone"/>
    </FrameLayout>
</LinearLayout>
```

（2）在 MainActivity 类的 onCreate() 方法中，实例化 VideoView 控件和 MediaController 控制器对象，然后设置 VideoView 的播放资源和 VideoView 的控制器，并让 MediaController 绑定 VideoView。最后播放视频。关键代码如下：

```
videoView = (VideoView) findViewById(R.id.video);
controllerHome = (FrameLayout) findViewById(R.id.home);
hint = (ImageView) findViewById(R.id.hint);
if (!Vitamio.isInitialized(this)) {
    Log.e("Vitamio", "没有被初始化！");
    return;
}
//播放控制器
controller = new MyController(this, true, controllerHome);
barragge = new MyBarrageSwitch();                    //弹幕开关
controller.setBarrageSwitchListener(barragge);    //弹幕开关监听
//播放的视频地址
videoView.setVideoURI(Uri.parse
  ("http://videos.mingrisoft.com/tx/Android/76/104/1115351991313104111_fpzsk.mp4"));
videoView.setMediaController(controller);          //设置控制器
videoView.requestFocus();                          //获取焦点
//准备监听
videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mp) {
        videoView.start();                         //播放视频
    }
});
```

（3）在 MainActivity 类中增加 addDanmaku() 方法，用来添加弹幕信息，并设置弹幕的效果。代码如下：

```
private void addDanmaku(String content) {
    BaseDanmaku danmaku = danmakuContext.mDanmakuFactory.
            createDanmaku(BaseDanmaku.TYPE_SCROLL_RL);
    danmaku.text = content;
    danmaku.padding = 5;                           //弹幕内容
    danmaku.textSize = sp2px(20);                  //字体大小
    danmaku.textColor = Color.BLACK;               //字体颜色
    danmaku.setTime(danmakuView.getCurrentTime());//发布时间
    danmakuView.addDanmaku(danmaku);
}
```

（4）在 MainAvtivity 类中添加 changeSound() 和 changeBright() 方法，用来根据手势调节屏幕的亮度和播放视频时的音量大小。代码如下：

```
private void changeSound(float values) {
    //获取当前音量
    nowSoundValues = manager.getStreamVolume(AudioManager.STREAM_MUSIC);
    if (nowSoundValues < 0) {
```

```
            nowSoundValues = 0;
        }
        //定义变化后的音量
        int index = (int) (values * maxSoundValues) + nowSoundValues;
        if (index > maxSoundValues) {       //判断最大音量
            index = maxSoundValues;         //定义最大音量
        } else if (index < 0) {
            index = 0;
        }
        //设置音量
        manager.setStreamVolume(AudioManager.STREAM_MUSIC, index, 0);
    }

    private void changeBright(float values) {
        //获取当前屏幕亮度
        nowScreenBright = getWindow().getAttributes().screenBrightness;
        if (nowScreenBright <= 0.00f || nowScreenBright < 0.01f) {
            nowScreenBright = 0.50f;
        }
        //获取窗口属性
        WindowManager.LayoutParams lpa = getWindow().getAttributes();
        //变化后的亮度
        lpa.screenBrightness = nowScreenBright + values;
        if (lpa.screenBrightness > 1.0f)
            lpa.screenBrightness = 1.0f;
        else if (lpa.screenBrightness < 0.01f)
            lpa.screenBrightness = 0.01f;
        //设置屏幕亮度
        getWindow().setAttributes(lpa);
    }
```

（5）运行本实例，将显示如图 18.11 所示的视频播放界面，单击右上角的“弹幕”，按钮将显示弹幕文字。

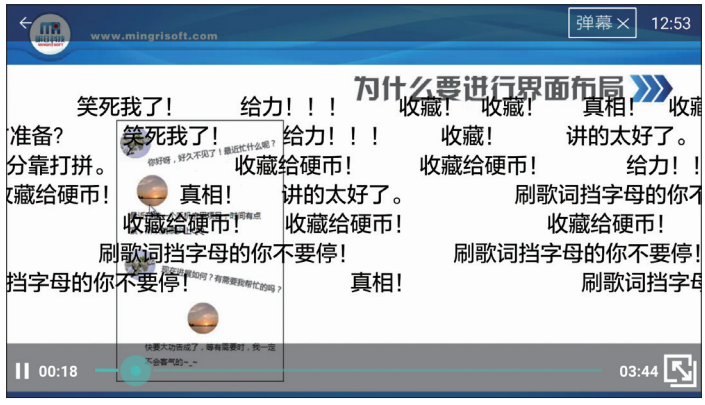
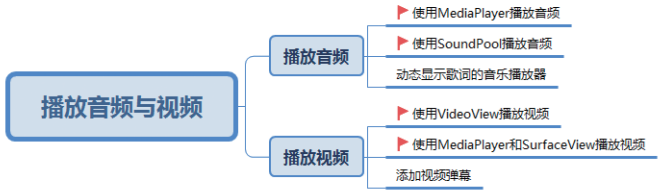


图 18.11　视频弹幕

18.3 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 19 章

拍照与显示图片

本章主要介绍 Android 中最常用的摄像头技术，主要包括使用摄像头拍照、通过摄像头录制视频以及如何调用系统下的相机与浏览图库中的图片。

19.1 控制摄像头

19.1.1 拍照

现在的智能手机和平板电脑一般都会提供摄像头拍照功能。在 Android 中提供了专门用于处理摄像头相关事件的类，即 android.hardware 包中的 Camera 类。Camera 类没有构造方法，可以通过其提供的 open() 方法打开摄像头。打开摄像头后，可以通过 Camera.Parameters 类处理摄像头的拍照参数。拍照参数设置完成后，可以调用 startPreview() 方法预览拍照画面，也可以调用 takePicture() 方法进行拍照。结束程序时，可以调用 Camera 类的 stopPreview() 方法结束预览，并调用 release() 方法释放摄像头资源。Camera 类常用的方法如表 19.1 所示。

表 19.1　Camera 类常用的方法

方　法	描　述
getParameters()	用于获取摄像头参数
Camera.open()	用于打开摄像头
release()	用于释放摄像头资源
setParameters(Camera.Parameters params)	用于设置摄像头的拍照参数
setPreviewDisplay(SurfaceHolder holder)	用于为摄像头指定一个用来显示预览画面的 SurfaceView
startPreview()	用于开始预览画面
takePicture(Camera.ShutterCallback shutter,Camera.PictureCallback raw, Camera.PictureCallback jpeg)	用于进行拍照
stopPreview()	用于停止预览

下面通过一个实例来说明控制摄像头拍照的具体过程。

例 19.1 实现控制摄像头拍照功能

在 Android Studio 中创建一个 Module，名称为“Camera”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为帧布局管理器，然后将 TextView 组件删除，再添加一个 SurfaceView 组件（用于显示摄像头预览画面），最后添加一个预览按钮和一个拍照按钮。具体代码请参见云盘。

（2）打开 MainActivity 类，该类继承 Activity，然后在该类中，定义所需的成员变量，关键代码如下：

```
01  private Camera camera;                                  //定义相机对象
02  private boolean isPreview = false;                      //定义非预览状态
```

（3）在 MainActivity 类的 onCreate() 方法中，首先设置全屏显示，然后判断手机是否安装 SD 卡，关键代码如下：

```
01  //设置全屏显示
02  getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
03          WindowManager.LayoutParams.FLAG_FULLSCREEN);
04  if (!Environment.getExternalStorageState().equals(    //判断手机是否安装SD卡
05          Environment.MEDIA_MOUNTED)) {
06      Toast.makeText(this, "请安装SD卡！", Toast.LENGTH_SHORT).show(); //提示安装SD卡
07  }
```

（4）获取 SurfaceView 组件与 SurfaceHolder 对象，用于显示摄像头预览，关键代码如下：

```
01  //获取SurfaceView组件，用于显示摄像头预览
02  SurfaceView sv = (SurfaceView) findViewById(R.id.surfaceView);
03  final SurfaceHolder sh = sv.getHolder();                //获取SurfaceHolder对象
04  //设置该SurfaceHolder自己不维护缓冲
05  sh.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
06  ImageButton preview = (ImageButton) findViewById(R.id.preview);       //获取"预览"按钮
07  ImageButton takePicture = (ImageButton) findViewById(R.id.takephoto);//获取"拍照"按钮
```

（5）为预览按钮添加单击事件监听器，实现摄像头的预览功能，关键代码如下：

```
01  preview.setOnClickListener(new View.OnClickListener() {  //实现摄像头预览功能
02      @Override
03      public void onClick(View v) {
04          //如果摄像头为非预览模式，则打开相机
05          if (!isPreview) {
06              camera = Camera.open();                     //打开相机
07                      isPreview = true;                   //设置为预览状态
08          }
09          try {
10              camera.setPreviewDisplay(sh);               //设置用于显示预览的SurfaceView
11              Camera.Parameters parameters = camera.getParameters();  //获取相机参数
```

```
            parameters.setPictureFormat(PixelFormat.JPEG);     //指定图片为JPEG图片
            parameters.set("jpeg-quality", 80);   //设置图片的质量
            camera.setParameters(parameters);     //重新设置相机参数
            camera.startPreview();                //开始预览
            camera.autoFocus(null);               //设置自动对焦
        } catch (IOException e) {                 //输出异常信息
            e.printStackTrace();
        }
    }
});
```

（6）在 MainActivity 中，创建实现重新预览的方法 resetCamera()，在该方法中，当 isPreview 变量的值为真时，调用摄像头的 startPreview() 方法开启预览，具体代码如下：

```
private void resetCamera() {          //创建resetCamera()方法，实现重新预览功能
    if (!isPreview) {                 //如果为非预览模式
        camera.startPreview();        //开启预览
        isPreview = true;
    }
}
```

（7）实现拍照的回调接口，在重写的 onPictureTaken() 方法中，首先根据拍照所得的数据创建位图，然后保存所拍摄的图片，再把保存的图片文件插入到系统图库，最后通知图库更新，具体代码如下：

```
//实现将照片保存到系统图库中
final Camera.PictureCallback jpeg = new Camera.PictureCallback() {  //照片回调函数
    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        //根据拍照所得的数据创建位图
        final Bitmap bm = BitmapFactory.decodeByteArray(data, 0,
                data.length);
        camera.stopPreview();                     //停止预览
        isPreview = false;                        //设置为非预览状态
        //获取sd卡根目录
        File appDir = new File(Environment.getExternalStorageDirectory(), "/DCIM/Camera/");
        if (!appDir.exists()) {                   //如果该目录不存在
            appDir.mkdir();                       //创建该目录
        }
        //将获取的当前系统时间设置为照片名称
        String fileName = System.currentTimeMillis() + ".jpg";
        File file = new File(appDir, fileName);     //创建文件对象
        try {  //保存拍到的图片
            FileOutputStream fos = new FileOutputStream(file);//创建一个文件输出流对象
            //将图片内容压缩为JPEG格式输出到输出流对象中
            bm.compress(Bitmap.CompressFormat.JPEG, 100, fos);
            //将缓冲区中的数据全部写出到输出流中
```

```
            fos.flush();
            fos.close();                                    //关闭文件输出流对象
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        //将照片插入到系统图库
        try {
            MediaStore.Images.Media.insertImage(MainActivity.this.getContentResolver(),
                    file.getAbsolutePath(), fileName, null);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        //最后通知图库更新
        Intent intent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
        Uri uri = Uri.fromFile(file);
        intent.setData(uri);
        MainActivity.this.sendBroadcast(intent);    //这个广播的目的就是更新图库
        Toast.makeText(MainActivity.this, "照片保存至: " + file, Toast.LENGTH_LONG).show();
        resetCamera();                                  //调用重新预览resetCamera()方法
    }
};
```

（8）在 onCreate() 方法中，为拍照按钮添加单击事件监听器，实现摄像头的拍照功能，关键代码如下：

```
takePicture.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (camera != null) {                           //相机不为空
            camera.takePicture(null, null, jpeg);  //进行拍照
        }
    }
});
```

（9）重写 Activity 的 onPause() 方法，用于当暂停 Activity 时，停止预览并释放摄像头资源，具体代码如下：

```
@Override
protected void onPause() {
    if (camera != null) {                               //如果摄像头不为空
        camera.stopPreview();                           //停止预览
        camera.release();                               //释放资源
    }
    super.onPause();
}
```

（10）由于本程序需要访问 SD 卡和控制摄像头，所以需要在 AndroidManifest.xml 文件中赋予程序访问 SD 卡和控制摄像头的权限，关键代码如下：

```
<!-- 授予程序可以向SD卡中保存文件的权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- 授予程序使用摄像头的权限 -->
<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature android:name="android.hardware.camera"/>
```

（11）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（12）运行本实例，在显示的界面中，单击预览按钮，启动摄像头，单击拍照按钮进行拍照，如图 19.1 所示。

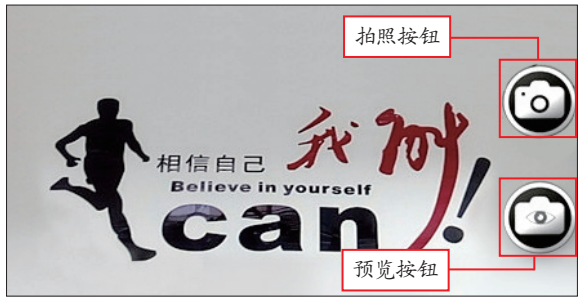


图 19.1　预览与拍照

说明　本实例需要摄像头硬件的支持，这里我们使用真机测试。读者需要在手机中手动开启摄像头权限与sd卡读写权限。

19.1.2 使用 Camera2 进行拍照

在 Android 5.0 版本中安卓的官方推出了 Camera 的升级版 Camera2，它不仅大幅提高了 Android 系统的拍照功能，还可以支持以下 5 点新特性：

（1）支持 30 帧的高清连拍功能。

（2）支持每帧之间的手动设置。

（3）支持 RAW 原始图像的拍摄。

（4）支持快门零延迟以及电影速拍。

（5）支持相机其它方面的手动控制，包括噪音消除。

Camera2 在 Camera 的基础上进行了大幅的改造，通过多个管理类来实现相机拍照时的工作过程，通过 Camera2 实现相机拍照的常用管理类与方法如下：

◆ CameraManager 类：相机管理器，主要用于检测系统摄像头、打开系统摄像头等。常用方法如下：

◎ getCameraCharacteristics() 方法：获取指定摄像头特性。

◎ getCameraIdList() 方法：获取相机列表。通常返回后置摄像头或前置摄像头信息。

◎ openCamera() 方法：打开指定的摄像头，第一个参数是指定摄像头的 ID，第二个参数是相机的回调方法，相机打开 1 次该方法被调用 1 次，第三个参数是通过指定的 Handler 处理程序。

◎ setTorchMode() 方法：用于打开或关闭指定摄像头 ID 的闪光灯。

◆ CameraDevice 类：用于实现添加预览界面、拍照请求等。常用方法如下：

◎ close() 方法：用于关闭相机。

◎ createCaptureSession() 方法：创建相机会话，第一个参数是捕获数据的 Surface 列表，第二个参数是 CameraCaptureSession 的状态回调接口，创建完成后回调 onConfigured 方法。第三个参数是确定 Callback 在哪个线程执行，将该参数设置为 null 的话就是在当前线程执行。

◎ createCaptureRequest() 方法：创建拍照请求，该方法支持 TEMPLATE_PREVIEW（预览请求）、TEMPLATE_STILL_CAPTURE（拍照请求）等参数，通过该方法创建 CaptureRequest.Builder 对象。

◆ ImageReader 类：读取图像数据，该类需要通过 setOnImageAvailableListener() 方法，设置图像数据监听器。该监听器需要实现 ImageReader.OnImageAvailableListener 接口与该接口中的 onImageAvailable() 方法。

说明 由于Camera2是在Android 5.0系统版本中推出的新技术，所以很多手机还无法使用Camera2中的方法。因此需要通过CameraCharacteristics类进行手机摄像头支持级别的判断，代码如下：

```
01  //获取相机服务
02  CameraManager cameraManager= (CameraManager) getSystemService(Context.CAMERA_SERVICE);
03  //获取指定摄像头特性
04  CameraCharacteristics characteristics =
05   cameraManager.getCameraCharacteristics(cameraId);
06  //获取手机摄像头支持的级别
07  characteristics.get(CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL);
```

返回值具体含义如下：

◆ INFO_SUPPORTED_HARDWARE_LEVEL_LEGACY：不建议使用 Camera2 中的相关技术。

◆ INFO_SUPPORTED_HARDWARE_LEVEL_LIMITED：有限支持 Camera2 中的相关技术。

◆ INFO_SUPPORTED_HARDWARE_LEVEL_FULL：完全支持 Camera2 中的相关技术。

下面通过一个具体的实例演示使用 Camera2 进行拍照的具体应用。

例 19.2 使用 Camera2 进行拍照

在 Android Studio 中创建 Module，名称为“Camera2 TakePicture”，该 Module 最小 sdk 版本为 21，实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为帧布局管理器并将 TextView 组件删除，然后添加 1 个 TextureView 组件用于显示预览图像，最后添加 1 个 ImageView 组件用于实现拍照按钮。具体代码如下：

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
03      xmlns:app="http://schemas.android.com/apk/res-auto"
04      xmlns:tools="http://schemas.android.com/tools"
05      android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示预览图像-->
    <TextureView
        android:id="@+id/textureView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <!--拍照按钮-->
    <ImageView
        android:id="@+id/take_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|bottom"
        android:src="@mipmap/ic_launcher_round"
        android:onClick="onTakePicture"/>
</FrameLayout>
```

（2）打开主活动 MainActivity.java 文件，首先定义所需要的全局变量，然后在 onCreate() 方法中获取用于显示预览图像的 TextureView 组件并为其设置监听器。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    private TextureView textureView;          //显示预览图像的组件
    private String cameraId = "0";            //摄像头id，0为后置摄像头，1为前置摄像头
    private CameraManager cameraManager;      //定义相机管理
    private CameraDevice cameraDevice;        //相机装置
    private CameraCaptureSession cameraCaptureSession;    //相机拍照会话
    //预览请求与拍照请求
    private CaptureRequest.Builder previewRequest, takePictureRequest;
    private CaptureRequest startPreview;                  //开始预览
    private ImageReader imageReader;                      //图像读取
    private Size previewDimension;                        //预览尺寸
    //定义旋转方向
    private static final SparseIntArray ORIENTATIONS = new SparseIntArray();
    static {
        ORIENTATIONS.append(Surface.ROTATION_0, 90);
        ORIENTATIONS.append(Surface.ROTATION_90, 0);
        ORIENTATIONS.append(Surface.ROTATION_180, 270);
        ORIENTATIONS.append(Surface.ROTATION_270, 180);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取显示预览图像组件
        textureView = (TextureView) findViewById(R.id.textureView);
        //设置TextureView监听
        textureView.setSurfaceTextureListener(surfaceTextureListener);
```

```
    }
}
```

说明 代码中的红色代码并不是错误代码，而是还没有通过代码创建该参数，在接下来的步骤当中该参数被创建后，代码颜色将显示正常。

（2）在 onCreate() 方法下创建 TextureView 组件的监听事件并实现其 4 个方法，具体代码如下：

```
TextureView.SurfaceTextureListener surfaceTextureListener =
        new TextureView.SurfaceTextureListener() {
    //可用时调用该方法
    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture surface,
                                          int width, int height) {
        openCamera();                       //调用打开摄像头方法
    }
    //TextureView组件尺寸变化时调用该方法
    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture surface,
                                            int width, int height) {

    }
    //销毁时调用该方法
    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
        if (cameraDevice != null) {         //相机装置存在时
            cameraDevice.close();           //关闭相机装置
            cameraDevice = null;            //设置相机装置为空
        }
        return true;
    }
    //更新时调用该方法
    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surface) {
    }
};
```

（3）创建 openCamera() 方法，在该方法中首先获取相机的管理服务，然后设置摄像头参数，最后判断相机权限是否开启，如果没有开启就添加相机与内存卡写入权限，否则打开摄像头。具体代码如下：

```
private void openCamera(){
    //获取相机服务
    cameraManager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    //调用设置摄像头参数方法
    setCameraParameters(cameraManager);
    try {
        //判断相机权限是否开启
```

```
        if (ActivityCompat.checkSelfPermission(this,
                Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
            //添加相机权限与内存卡写入权限
            ActivityCompat.requestPermissions(MainActivity.this,
                    new String[]{"android.permission.CAMERA",
                            "android.permission.WRITE_EXTERNAL_STORAGE"}, 1);
        } else {
            //否则打开摄像头
            cameraManager.openCamera(cameraId, stateCallback, null);
        }
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

（4）重写 onRequestPermissionsResult() 方法，该方法是权限授权的回调方法，在该方法中判断如果已经授权就调用打开摄像头的方法，否则弹出该权限的重要性的对话框。具体代码如下：

```
@Override
public void onRequestPermissionsResult(int permsRequestCode,
                                       String[] permissions, int[] grantResults) {
    switch (permsRequestCode) {
        case 1:
            //判断授权结果是否已经授权
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    openCamera();          //调用打开摄像头方法
            } else {
                //用户没有授权为用户提示该权限的重要性
                new AlertDialog.Builder(this)
                     .setTitle("提示信息")
                     .setMessage("当前应用缺少必要权限，该功能暂时无法使用。" +
                             "如若需要，请单击【确定】按钮前往设置中心进行权限授权。")
                     .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                            }
                       })
                      .setPositiveButton("确定", new DialogInterface.OnClickListener() {
                          @Override
                          public void onClick(DialogInterface dialog, int which) {
                              //通过Intent跳转设置中心界面
                              Intent intent = new Intent(
                                      Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                              intent.setData(Uri.parse("package:" + getPackageName()));
                              startActivity(intent);
                            }
```

```
                        }).show();
                }
                break;
        }
    }
```

（5）创建 setCameraParameters() 方法，在该方法中首先获取制定的摄像头，然后获取摄像头配置属性，最后设置预览尺寸，该方法主要用于设置摄像头参数。具体代码如下：

```
01  private void setCameraParameters(CameraManager cameraManager) {
02      try {
03          //获取指定的摄像头
04          CameraCharacteristics cameraCharacteristics =
05                      cameraManager.getCameraCharacteristics(cameraId);
06          //获取摄像头配置属性
07          StreamConfigurationMap streamConfigurationMap = cameraCharacteristics.get(
08                   CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
09          //获取摄像头支持的最大尺寸
10          Size largestDimension = Collections.max(
11                  Arrays.asList(streamConfigurationMap.getOutputSizes
12                          (ImageFormat.JPEG)), new CompareDimension());
13          //创建一个ImageReader对象，用于获取摄像头的图像数据
14          imageReader = ImageReader.newInstance(largestDimension.getWidth(),
15                  largestDimension.getHeight(),ImageFormat.JPEG, 2);
16          //设置获取图片的监听
17          imageReader.setOnImageAvailableListener(imageAvailableListener, null);
18          //设置预览尺寸
19          previewDimension = streamConfigurationMap.getOutputSizes
20  (SurfaceTexture.class)[0];
21      } catch (CameraAccessException e) {
22          e.printStackTrace();
23      } catch (NullPointerException e) {
24      }
25  }
26  //尺寸比较器
27  static class CompareDimension implements Comparator<Size> {
28      @Override
29      public int compare(Size lhs, Size rhs) {
30          //强制转换为long保证不会发生溢出
31          return Long.signum((long) lhs.getWidth() * lhs.getHeight() -
32                  (long) rhs.getWidth() * rhs.getHeight());
33      }
34  }
```

（6）通过关键字 New 创建 CameraDevice.StateCallback 对象并实现其 3 个方法，该对象为摄像头状态的监听器。具体代码如下：

```
01 private CameraDevice.StateCallback stateCallback = new CameraDevice.StateCallback() {
02     //摄像头被打开时触发该方法
03     @Override
04     public void onOpened(CameraDevice cameraDevice) {
05         MainActivity.this.cameraDevice = cameraDevice;
06         //开始预览
07         beginPreview();
08     }
09     //摄像头断开连接时触发该方法
10     @Override
11     public void onDisconnected(CameraDevice cameraDevice) {
12         MainActivity.this.cameraDevice.close();
13         MainActivity.this.cameraDevice = null;
14     }
15     //打开摄像头出现错误时触发该方法
16     @Override
17     public void onError(CameraDevice cameraDevice, int error) {
18         cameraDevice.close();
19     }
20 };
```

（7）创建 beginPreview 方法，在方法中首先创建预览请求，然后创建相机捕获会话，最后进行预览。具体代码如下：

```
01 private void beginPreview() {
02     SurfaceTexture mSurfaceTexture = textureView.getSurfaceTexture();
03     //设置TextureView的缓冲区大小
04     mSurfaceTexture.setDefaultBufferSize(
05             previewDimension.getWidth(), previewDimension.getHeight());
06     //获取Surface显示预览数据
07     Surface mSurface = new Surface(mSurfaceTexture);
08     try {
09         //创建预览请求
10         previewRequest = cameraDevice.createCaptureRequest
11 (CameraDevice.TEMPLATE_PREVIEW);
12         //设置自动对焦模式
13         previewRequest.set(CaptureRequest.CONTROL_AF_MODE,
14                 CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
15         //设置Surface作为预览数据的显示界面
16         previewRequest.addTarget(mSurface);
17         //创建相机捕获会话
18         cameraDevice.createCaptureSession(Arrays.asList(mSurface,
19                 imageReader.getSurface()), new CameraCaptureSession.StateCallback() {
20             @Override
21             public void onConfigured(CameraCaptureSession session) {
22                 try {
23                     //开始预览
```

```
                    startPreview = previewRequest.build();
                    cameraCaptureSession = session;
                    //设置反复捕获数据的请求，这样预览界面就会一直有数据显示
                    cameraCaptureSession.setRepeatingRequest(startPreview, null, null);
                } catch (CameraAccessException e) {
                    e.printStackTrace();
                }
            }
            @Override
            public void onConfigureFailed(CameraCaptureSession session) {
            }
        }, null);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

（8）创建 onTakePicture() 方法，该方法是拍照按钮的单击事件，在该方法中首先需要创建拍照请求，然后设置拍照需要的相关属性，最后进行拍照。具体代码如下：

```
public void onTakePicture(View view) {
    try {
        if (cameraDevice == null) {
            return;
        }
        //创建拍照请求
        takePictureRequest = cameraDevice.
createCaptureRequest(CameraDevice.TEMPLATE_STILL_CAPTURE);
        //设置自动对焦模式
        takePictureRequest.set(CaptureRequest.CONTROL_AF_MODE,
CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
        //将imageReader的surface设为目标
        takePictureRequest.addTarget(imageReader.getSurface());
        //获取设备方向
        int rotation = getWindowManager().getDefaultDisplay().getRotation();
        //根据设备方向计算设置照片的方向
        takePictureRequest.set(CaptureRequest.JPEG_ORIENTATION,
                ORIENTATIONS.get(rotation));
        //停止连续取景
        cameraCaptureSession.stopRepeating();
        //拍照
        CaptureRequest captureRequest = takePictureRequest.build();
        //设置拍照监听
        cameraCaptureSession.capture(captureRequest, captureCallback, null);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

（9）通过关键字 New 创建 CameraCaptureSession.CaptureCallback 对象并实现其 2 个方法，在 onCaptureCompleted() 方法中实现拍照完成后重新预览的功能。具体代码如下：

```
01  private CameraCaptureSession.CaptureCallback captureCallback =
02          new CameraCaptureSession.CaptureCallback() {
03              //拍照成功
04              @Override
05              public void onCaptureCompleted(CameraCaptureSession session,
06                                             CaptureRequest request, TotalCaptureResult result) {
07                  //自动对焦模式
08                  takePictureRequest.set(CaptureRequest.CONTROL_AF_TRIGGER,
09                          CameraMetadata.CONTROL_AF_TRIGGER_CANCEL);
10                  //自动曝光模式
11                  takePictureRequest.set(CaptureRequest.CONTROL_AE_MODE,
12                          CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);
13                  try {
14                      //重新预览
15                      cameraCaptureSession.setRepeatingRequest(startPreview, null, null);
16                  } catch (CameraAccessException e) {
17                      e.printStackTrace();
18                  }
19              }
20              @Override
21              public void onCaptureFailed(CameraCaptureSession session,
22                                          CaptureRequest request, CaptureFailure failure) {
23                  super.onCaptureFailed(session, request, failure);
24              }
25          };
```

（10）通过关键字 New 创建 ImageReader.OnImageAvailableListener 对象并实现 onImageAvailable() 方法，在该方法中首先获取照片的数据，然后创建并指定照片保存的位置，再将照片插入系统图库中，最后通知图库更新。具体代码如下：

```
01  private ImageReader.OnImageAvailableListener imageAvailableListener =
02          new ImageReader.OnImageAvailableListener() {
03              //当照片数据可用时激发该方法
04              @Override
05              public void onImageAvailable(ImageReader reader) {
06              if (!Environment.getExternalStorageState().equals( //判断手机是否安装SD卡
07                          Environment.MEDIA_MOUNTED)) {
08                      Toast.makeText(getApplicationContext(),
09                              "请安装SD卡！", Toast.LENGTH_SHORT).show(); //提示安装SD卡
10                  }
11                  //获取捕获的照片数据
12                  Image image = reader.acquireNextImage();
13                  ByteBuffer byteBuffer = image.getPlanes()[0].getBuffer();
14                  byte[] data = new byte[byteBuffer.remaining()];
15                  byteBuffer.get(data);
```

```
                //获取手机拍照后保存路径
                String filePath = Environment.
                        getExternalStorageDirectory().getPath() + "/DCIM/Camera/";
                String fileName = System.currentTimeMillis() + ".jpg";   //照片名称
                File file = new File(filePath, fileName);
                try {
                    //保存拍到的图片
                    FileOutputStream fileOutputStream = new FileOutputStream(file);
                    fileOutputStream.write(data);        //写入图片数据
                    fileOutputStream.close();            //关闭文件输出流对象
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    image.close();
                }
                //将照片插入到系统图库
                try {
                    MediaStore.Images.Media.insertImage(MainActivity.this
                        .getContentResolver(),file.getAbsolutePath(), fileName, null);
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                }
                //最后通知图库更新
                Intent intent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
                Uri uri = Uri.fromFile(file);
                intent.setData(uri);
                MainActivity.this.sendBroadcast(intent);//这个广播的目的就是更新图库
                Toast.makeText(MainActivity.this,
                        "照片保存至: " + file, Toast.LENGTH_LONG).show();
            }
        };
```

（11）重写 onRestart() 方法，在该方法中实现重返界面时启动相机的功能，然后重写 onPause() 方法，在该方法中实现界面停止时释放相机资源的功能。具体代码如下：

```
//重返界面启动相机
@Override
protected void onRestart() {
    super.onRestart();
    if (textureView.isAvailable()) {    //显示预览图像组件可用时
        if (cameraDevice == null) {     //相机装置为空时
            openCamera();               //调用打开摄像头方法
        }
    } else {
        //否则为显示预览图像组件设置监听器
```

```
            textureView.setSurfaceTextureListener(surfaceTextureListener);
        }
    }
    //界面停止时释放相机资源
    @Override
    protected void onPause() {
        super.onPause();
        if (cameraDevice != null) {          //相机装置存在时
            cameraDevice.close();            //关闭相机装置
            cameraDevice = null;             //设置相机装置为空
        }
    }
```

（12）打开 AndroidManifest.xml 文件，在该文件中添加相机与内存卡写入权限。具体代码如下：

```
<!--相机权限-->
<uses-permission android:name="android.permission.CAMERA" />
<!--内存卡写入权限-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

（13）运行本实例，将显示允许相机与内存卡写入权限的提示对话框如图 19.2 所示。单击底部中间位置的拍照按钮进行拍照，如图 19.3 所示。

图 19.2　允许相机与内存卡写入权限

图 19.3　拍照效果图

说明　由于该实例需要使用手机的摄像头硬件进行图像的获取与捕捉，所以需要使用真机来运行该实例。

19.1.3　录制视频

Android 系统提供了 MediaRecorder 类，用于录制音频和视频。使用 MediaRecorder 类录制视频的过程比较简单，基本步骤如下：

（1）创建 MediaRecorder 对象。

（2）调用 MediaRecorder 对象的 setAudioSource() 方法设置声音来源，一般需要传入 MediaRecorder.AudioSource.MIC 参数指定录制来自麦克风的声音。因为在录制视频时不仅需要采集声音，还需要采集图像，所以在调用 setAudioSource() 方法时需要调用 setVideoSource() 方法来设置图像来源。

（3）调用 MediaRecorder 对象的 setOutputFormat() 方法设置输出文件的格式。

（4）设置所录制的音频和视频的编码格式、编码位率等，常用方法如表 19.2 所示。

表 19.2　音频和视频的常用设置方法

方　法	描　述
setAudioEncoder(int audio_encoder)	设置声音的编码格式
setAudioEncodingBitRate(int bitRate)	设置声音的编码位率
setAudioSamplingRate(int samplingRate)	设置声音的采样率
setVideoEncoder(int video_encoder)	设置视频的编码格式
setVideoEncodingBitRate(int bitRate)	设置视频的编码位率
setVideoFrameRate(int rate)	设置视频的帧速率
setVideoSize(int width, int height)	设置视频的宽度和高度

说明 表19.2中各个方法的参数可以控制所录制的声音、视频的品质以及文件的大小。一般来说，声音、视频的品质越好，文件越大。

（5）调用 MediaRecorder 对象的 setOutputFile(String path) 方法设置所录制视频文件的保存位置。

（6）调用 MediaRecorder 对象的 setPreviewDisplay(Surface sv) 方法设置使用哪个 SurfaceView 来显示视频预览。

（7）调用 MediaRecorder 对象的 prepare() 方法准备录制视频。

（8）调用 MediaRecorder 对象的 start() 方法开始录制视频。

（9）录制完成，调用 MediaRecorder 对象的 stop() 方法停止录制，并调用 release() 方法释放资源。

注意 在执行上述步骤的时候，必须在设置视频文件的输出格式之后再设置音频和视频的编码格式，否则程序将会抛出异常。

在录制视频的时候需要使用麦克风录制声音以及使用摄像头采集图像，这些都需要授予相应的权限。而且由于录制视频时视频文件所占用的存储空间不断增大，可能需要使用外部存储器（外部 SD 卡），因此需要授予程序向外部存储设备写入数据的权限。具体方法是在 AndroidManifest.xml 文件中增加如下授权配置：

```
<!-- 授予程序录制声音的权限 -->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<!-- 授予程序使用摄像头的权限 -->
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<!-- 授予使用外部存储器的权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

说明 Android官方API中指出，目前最新版的MediaRecorder不能在模拟器上运行，因此用户需要在真机上进行测试。

下面通过一个具体的实例来说明控制摄像头录制视频的具体过程。

例 19.3 实现摄像头录制视频功能

在 Android Studio 中创建 Module，名称为“Record Video”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加一个 SurfaceView 组件，用于显示视频图像，再添加一个水平线性布局管理器，最后在水平线性布局管理器中分别添加“播放”按钮、“录制”按钮和“停止”按钮。具体代码请参见云盘。

（2）创建一个 Empty Activity 界面，名称为 PlayVideoActivity。该界面代码与实例 17.2.2 小节中的实例代码相同，SD 卡上的视频路径不同。该界面用于播放录制的视频文件。

（3）打开 MainActivity 类，该类继承 Activity，然后在该类中，定义所需的成员变量，关键代码如下：

```
01 private ImageButton play, stop, record;          //定义播放、停止、录制、三种图片按钮
02 private MediaRecorder mediaRecorder;             //定义用于实现录制视频的MediaRecorder类
03 private SurfaceView surfaceView;                 //定义用于显示图像的SurfaceView类
04 private boolean isRecord = false;                //定义录制状态
05 private File videoFile;                          //定义录制视频的文件夹
06 private android.hardware.Camera camera;          //定义摄像头
07 private File path;                               //定义录制视频保存的路径
```

（4）在 MainActivity 类的 onCreate() 方法中，判断手机是否安装 SD 卡，关键代码如下：

```
01 //设置全屏显示
02 getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
03         WindowManager.LayoutParams.FLAG_FULLSCREEN);
04 //判断SD卡是否存在，如果不存在给出相应提示
05 if (!android.os.Environment.getExternalStorageState().equals(
06         android.os.Environment.MEDIA_MOUNTED)) {
07     //弹出消息提示框显示提示信息
08     Toast.makeText(MainActivity.this, "请安装SD卡！", Toast.LENGTH_SHORT).show();
09 }
```

（5）获取布局文件中的相关组件，关键代码如下：

```
01 stop = (ImageButton) findViewById(R.id.stop);       //获取停止录制的按钮
02 record = (ImageButton) findViewById(R.id.record);  //获取录制按钮
03 play = (ImageButton) findViewById(R.id.play);       //获取播放已经录制好视频的按钮
04 stop.setEnabled(false);                             //设置停止按钮不可用
05 play.setEnabled(false);                             //设置播放按钮不可用
06 //获取显示图像的SurfaceView组件
07 surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
08 surfaceView.getHolder().setFixedSize(1920, 1080); //设置SurfaceView的分辨率
```

（6）在 MainActivity 中，创建 record() 方法，在该方法中实现录制功能。具体代码如下：

```
01 private void record() {                             //创建record()方法，实现录制功能
```

```
    //设置录制视频保存的文件夹
    videoFile = new File(Environment.getExternalStorageDirectory() + "/Myvideo/");
    if (!videoFile.exists()) {                                    //如果该目录不存在
        videoFile.mkdir();                                        //就创建该目录
    }
    String fileName = "video.mp4";                                //视频文件的名称
    path = new File(videoFile, fileName);                         //视频文件的路径
    //创建MediaRecorder对象
    mediaRecorder = new MediaRecorder();
    camera.setDisplayOrientation(90);                             //调整摄像头角度
    camera.unlock();                                              //解锁摄像头
    mediaRecorder.setCamera(camera);                              //使用摄像头
    mediaRecorder.reset();                                        //重置MediaRecorder
    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);//设置麦克风获取声音
    mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);//设置摄像头获取图像
    //设置视频输出格式为MP4
    mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);//设置声音编码格式
    //设置视频编码格式为MP4
    mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
    mediaRecorder.setVideoEncodingBitRate(1920 * 1080);           //设置清晰度
    mediaRecorder.setVideoSize(1920, 1080);                       //设置视频的尺寸
    mediaRecorder.setVideoFrameRate(10);                          //设置为每秒10帧
    mediaRecorder.setOutputFile(path.getAbsolutePath());          //设置视频输出路径
    //设置使用SurfaceView预览视频
    mediaRecorder.setPreviewDisplay(surfaceView.getHolder().getSurface());
    mediaRecorder.setOrientationHint(90);                         //调整播放视频角度
    try {
        mediaRecorder.prepare();                                  //准备录像
    } catch (Exception e) {
        e.printStackTrace();
    }
    mediaRecorder.start();      //开始录制
    Toast.makeText(MainActivity.this, "开始录像", Toast.LENGTH_SHORT).show();
    record.setEnabled(false);   //设置录制按钮不可用
    stop.setEnabled(true);      //设置停止按钮可用
    play.setEnabled(false);     //设置播放按钮不可用
    isRecord = true;            //设置录像状态为正在录制
}
```

（7）在 onCreate() 方法中，单击录制按钮，实现录制功能。关键代码如下：

```
record.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        record();                                  //调用record()方法实现录制功能
    }
});
```

（8）单击停止按钮，实现停止录制并提示视频的保存位置。关键代码如下：

```
stop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isRecord) {                 //如果是正在录制
            mediaRecorder.stop();       //停止录制
            mediaRecorder.release();    //释放资源
            record.setEnabled(true);    //设置录制按钮可用
            stop.setEnabled(false);     //设置停止按钮不可用
            play.setEnabled(true);      //设置播放按钮可用
            Toast.makeText(MainActivity.this, "录像保存在：" +
                    path, Toast.LENGTH_SHORT).show();
        }
    }
});
```

（9）单击播放按钮，实现播放录制好的视频。关键代码如下：

```
//为播放按钮设置单击事件，实现播放录制好的视频
play.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //通过Intent跳转播放视频界面
        Intent intent = new Intent(MainActivity.this, PlayVideoActivity.class);
        startActivity(intent);
    }
});
```

（10）重写 Activity 的 onResume() 方法，用于 Activity 获取焦点时，开启摄像头。onPause() 方法，用于 Activity 失去焦点时，停止预览并释放资源。具体代码如下：

```
@Override
protected void onResume() {
    camera = android.hardware.Camera.open(); //开启摄像头
    super.onResume();
}

@Override
protected void onPause() {
    camera.stopPreview();                    //停止预览
    camera.release();                        //释放资源
    super.onPause();
}
```

（11）打开 AndroidManifest.xml 文件，在其中设置相关权限，具体代码如下：

```
<!--授予程序录制声音的权限-->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

```
<!--授予程序使用摄像头的权限-->
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<!--授予使用外部存储器的权限-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

（12）运行本实例，将显示如图 19.4 所示。

图 19.4　录制视频

19.2 调用系统相机与图库

19.2.1 使用 Intent 启动系统相机

在生活中经常会通过各种 App 与朋友分享自己的照片，这些照片可以在相册中获取也可以通过调用手机的摄像头进行拍照，如果在应用程序中拍照，那么最简单的方法就是通过 Intent 来调用系统下的相机程序进行拍照。

例如，单击名称为“btn”的按钮时，调用系统下的相机程序就可以使用下面的代码：

```
Button btn= (Button) findViewById(R.id.btn);        //获取普通按钮
//设置按钮单击事件
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //创建Intent并设置动作为图像捕捉
        Intent intent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(intent,100);        //启动Intent并要求返回数据
    }
});
```

将 Intent 动作设置为 MediaStore.ACTION_IMAGE_CAPTURE 用于调用系统下的相机程序，然后通过 startActivityForResult() 方法来启动 Intent，当相机程序拍完照片后可以将返回的数据进行处理，代码如下：

```
//重写onActivityResult()方法
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK && requestCode == 100) {//判断如果拍到照片后
        //此处进行数据处理data为数据
    }else {
        //否则提示没有拍到照片
        Toast.makeText(this, "没有拍到照片！", Toast.LENGTH_SHORT).show();
    }
}
```

下面通过一个实例演示通过 Intent 调用系统相机程序的具体应用。

例 19.4 调用系统相机进行拍照

在 Android Studio 中创建 Module，名称为“Intent Camera”，实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器并将 TextView 组件删除，然后先添加 1 个 Button 组件用于启动相机的按钮，再添加 1 个 ImageView 组件用于显示所拍的照片。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.mingrisoft.MainActivity">
    <!--启动相机按钮-->
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="启动系统相机" />
    <!--显示所拍的照片-->
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_gravity="center_horizontal"/>
</LinearLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中首先获取 Button 组件并为其设置单击事件监听器用于启动系统相机程序。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.button);//获取普通按钮
        //设置按钮单击事件
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //创建Intent并设置动作为图像捕捉
                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(intent, 100);      //启动Intent并要求返回数据
            }
        });
    }
}
```

（3）重写 onActivityResult() 方法，判断如果拍到照片后将照片显示在 ImageView 组件当中否则通过 Toast 提示“没有拍到照片！”。具体代码如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK && requestCode == 100) {//判断如果拍到照片后
        Bundle bundle = data.getExtras();                    //获取Intent的附加数据
        Bitmap bitmap= (Bitmap) bundle.get("data");          //获取数据并转换为Bitmap
        //获取显示拍到图片的组件
        ImageView imageView= (ImageView) findViewById(R.id.imageView);
        imageView.setImageBitmap(bitmap);                    //将Bitmap数据显示
    } else {
        //否则提示没有拍到照片
        Toast.makeText(this, "没有拍到照片！", Toast.LENGTH_SHORT).show();
    }
}
```

（4）运行本实例，如图 19.5 所示。

图 19.5　调用系统相机拍照

19.2.2 使用 Intent 浏览图库并选取图片

虽然通过 Intent 来调用系统下的相机程序非常方便，但我们不需要每次都调用相机进行拍照。有时手机的图库中已经存在我们所需要的图片，此时也可以通过 Intent 调用系统中的图库进行图片的浏览或者是选取图片并显示在其他的位置。

例如，单击名称为“button”的按钮时，调用系统下的图库就可以使用下面的代码：

```
Button button = (Button) findViewById(R.id.button); //获取普通按钮
//设置按钮单击事件
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //创建Intent并设置动作为调用系统图库
        Intent intent = new Intent(Intent.ACTION_PICK,
                android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(intent, 101);        //启动Intent并要求返回数据
    }
});
```

下面通过一个实例演示通过 Intent 调用系统图库的具体应用。

例 19.5 调用系统图库并选取图片

在 Android Studio 中创建 Module，名称为“Select Picture”，实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器并将 TextView 组件删除，然后先添加 1 个 Button 组件用于调用系统图库的按钮，再添加 1 个 ImageView 组件用于显示选择的图片。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.mingrisoft.MainActivity">
    <!--选择图片按钮-->
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="点击选择图片" />
    <!--显示选择的图片-->
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

（2）打开主活动 MainActivity.java 文件，修改默认生成的代码，让 MainActivity 直接继承 Activity，并导入 android.app.Activity 类，然后在 onCreate() 方法中首先获取 Button 组件并为其设置单击事件监听器，在 onClick() 方法中判断是否开启了 SD 卡读取权限，没有开启进行开启否则调用系统图库。修改后的具体代码如下：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.button);      //获取普通按钮
        //设置按钮单击事件
        button.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                //判断是否开启了SD卡读取权限，没有开启进行开启否则调用系统图库
                if (ContextCompat.checkSelfPermission(MainActivity.this,
                        Manifest.permission.READ_EXTERNAL_STORAGE) !=
                        PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(MainActivity.this,
                        new String[]{Manifest.permission.READ_EXTERNAL_STORAGE}, 1);
                } else {
                    //创建Intent并设置动作为调用系统图库
                    Intent intent = new Intent(Intent.ACTION_PICK,
                    android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                    startActivityForResult(intent, 101);  //启动Intent并要求返回数据
                }
            }
        });
    }
}
```

（3）重写 onRequestPermissionsResult() 方法，该方法是添加权限的回调方法用于实现开启 SD 卡读取权限后调用系统图库。具体代码如下：

```
@Override
public void onRequestPermissionsResult(int requestCode,
                @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    //创建Intent并设置动作为调用系统图库
    Intent intent = new Intent(Intent.ACTION_PICK,
             android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, 101);    //启动Intent并要求返回数据
}
```

（4）重写 onActivityResult() 方法，判断如果选择图片后将图片显示在 ImageView 组件当中。具体代码如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == Activity.RESULT_OK && requestCode == 101 && data != null) {
        Uri selectPicture = data.getData();                    //获取图片Uri数据
        String[] picturePath = {MediaStore.Images.Media.DATA};//图库路径
        //查询选择的图片
        Cursor cursor = getContentResolver().
```

```
                query(selectPicture, picturePath, null, null, null);
        cursor.moveToFirst();                                    //指向查询结果的第一个位置
        int columnIndex = cursor.getColumnIndex(picturePath[0]);//获取列的索引
        String imagePath = cursor.getString(columnIndex);        //根据索引找到选择的图片
        Bitmap bitmap = BitmapFactory.decodeFile(imagePath);     //加载图片
        //获取图像组件
        ImageView imageView = (ImageView) findViewById(R.id.image);
        imageView.setImageBitmap(bitmap);                        //显示选择的图片
        cursor.close();                                          //关闭游标
    }
}
```

（5）打开 AndroidManifest.xml 文件，在该文件中添加 SD 卡读取权限的代码。关键代码如下：

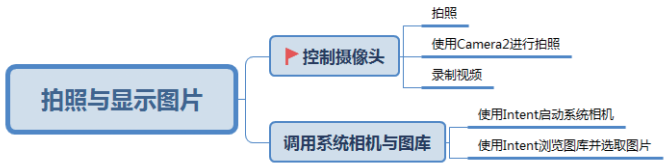
```
<!--SD卡读取权限代码-->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

（6）运行本实例，将显示如图 19.6 所示。

图 19.6　浏览系统图库

19.3 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 20 章

利用 OpenGL 实现 3D 图形

在现在这个网络游戏逐渐盛行的时代，2D 游戏已经不能完全满足用户的需求，3D 技术已经被广泛的应用在 PC 游戏中，3D 技术下一步将会向手机平台发展，而 Android 系统作为当前最流行的手机操作系统，完全内置 3D 技术——OpenGL 的支持，本章将对 Android 中的 3D 技术——OpenGL 进行详细讲解。

20.1 OpenGL 简介

OpenGL（Open Graphics Library）是由 SGI 公司于 1992 年发布的，一个功能强大、调用方便的底层图形库，它为编程人员提供了统一的操作，以便充分利用任何制造商提供的硬件。OpenGL 的核心实现了视区和光照等我们熟知的概念，并试图向开发人员隐藏大部分硬件层。

由于 OpenGL 是专门为工作站设计的，它太大了，无法安装在移动设备上。所以 Khronos Group 为 OpenGL 提供了一个子集 OpenGL ES（OpenGL for Embedded System）。OpenGL ES 是免费的、跨平台的、功能完善的 2D/3D 图形库接口 API，它是专门针对多种嵌入式系统（包括手机、PDA 和游戏主机等）而设计的，提供一种标准方法来描述在图形处理器或主 CPU 上渲染这些图像的底层硬件。

说明 | Khronos Group是一个图形软硬件行业协会，该协会主要关注图形和多媒体方向的开放标准。

OpenGL ES 去除了 OpenGL 中的 glBegin/glEnd、四边形（GL_QUADS）、多边形（GL_POLYGONS）等复杂图元的许多非绝对必要的特性。经过多年发展，目前的 OpenGL ES 主要有 OpenGL ES 1.x（针对固定管线硬件）和 OpenGL ES 2.x（针对可编程管线硬件）两个版本。OpenGL ES 1.0 是以 OpenGL 1.3 规范为基础的，OpenGL ES 1.1 是以 OpenGL 1.5 规范为基础的，OpenGL ES 2.0 则是参照 OpenGL 2.0 规范定义的，它补充和修改了 OpenGL ES 1.1 标准着色器语言及 API，将 OpenGL ES 1.1 中所有可以用着色器程序替换的功能全部删除了，这样可以节约移动设备的开销及电力消耗。

说明 | OpenGL ES可以应用于很多主流移动平台上，包括Android、Symbian和ios等。

Android 为 OpenGL 提供了相应的支持，它专门为支持 OpenGL 提供了 android.opengl 包。在该包中，GLES10 类是为支持 OpenGL ES 1.0 而提供的；GLES11 类是为支持 OpenGL ES 1.1 而提供的，GLES20 类是为支持 OpenGL ES 2.0 而提供的。其中，OpenGL ES 2.0 是从 Android 2.2（API Level 8）版本才开始使用的。

说明 | 如果你的应用只支持OpenGL ES 2.0，你必须在该项目的AndroidManifest.xml文件中添加下列设置：

```
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

20.2 绘制 3D 图形

OpenGL ES 一个最常用的功能就是绘制 3D 图形。要绘制 3D 图形，大致可以分为两个步骤，下面分别进行讲解。

20.2.1 构建 3D 开发的基本框架

构建一个 3D 开发的基本框架大致可以分为以下几个步骤：

（1）创建一个 Activity，并指定该 Activity 显示的内容是一个指定了 Renderer 对象的 GLSurfaceView 对象。例如，创建一个名称为 MainActivity 的 Activity，在重写的 onCreate() 方法中，创建一个 GLSurfaceView 对象用于显示 OpenGL 渲染后的 3D 图像，然后为其指定使用的 Renderer 对象，该对象用于实现 3D 图形的绘制。最后将其设置为 Activity 要显示的内容，可以使用下面的代码：

```
01 @Override
02 protected void onCreate(Bundle savedInstanceState) {
03     super.onCreate(savedInstanceState);
04     GLSurfaceView mGLView = new GLSurfaceView(this); //创建一个GLSurfaceView对象
05     mGLView.setRenderer(new CubeRenderer());//为GLSurfaceView指定使用的Renderer对象
06     setContentView(mGLView);                //设置Activity显示的内容为GLSurfaceView对象
07 }
```

通常情况下，考虑到当 Activity 恢复和暂停时，GLSurfaceView 对象也恢复或者暂停，还要重写 Activity 的 onResume() 方法和 onPause() 方法。例如，如果一个 Activity 使用的 GLSurfaceView 对象为 mGLView，那么，可以使用以下的重写 onResume() 和 onPause() 方法的代码：

```
01 @Override
02 protected void onResume() {
03     super.onResume();
04     mGLView.onResume();
05 }
06 @Override
07 protected void onPause() {
08     super.onPause();
09     mGLView.onPause();
10 }
```

（2）创建实现 GLSurfaceView.Renderer 接口的类。在创建该类时，需要实现接口中的以下 3 个方法：

☑ public void onSurfaceCreated(GL10 gl, EGLConfig config)：当 GLSurfaceView 被创建时回调该方法。

☑ public void onDrawFrame(GL10 gl)：Renderer 对象调用该方法绘制 GLSurfaceView 的当前帧。

☑ public void onSurfaceChanged(GL10 gl, int width, int height)：当 GLSurfaceView 的大小改变时回调该方法。

例如，创建一个实现 GLSurfaceView.Renderer 接口的类 EmptyRenderer，并实现 onSurfaceCreated()、onDrawFrame() 和 onSurfaceChanged() 方法，为窗体设置背景颜色，具体代码如下：

```
01 import javax.microedition.khronos.egl.EGLConfig;
02 import javax.microedition.khronos.opengles.GL10;
03 import android.opengl.GLSurfaceView;
04 public class EmptyRenderer implements GLSurfaceView.Renderer {
05     public void onSurfaceCreated(GL10 gl, EGLConfig config) {
06         //设置窗体的背景颜色
07         gl.glClearColor(0.7f, 0.7f, 0.9f, 1.0f);
08     }
09     public void onDrawFrame(GL10 gl) {
10         //重设背景颜色
11         gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
12     }
13     public void onSurfaceChanged(GL10 gl, int width, int height) {
14         gl.glViewport(0, 0, width, height);
15     }
16 }
```

当窗口被创建时，需要调用 onSurfaceCreated() 方法，进行一些初始化操作。onSurfaceCreated() 方法有一个 GL10 类型的参数 gl，gl 相当于 OpenGL ES 的画笔。通过它提供的方法不仅可以绘制 3D 图形，也可以对 OpenGL 进行初始化。下面将以表格的形式给出 GL10 提供的用于进行初始化的方法，对于 GL10 提供的用于绘制 3D 图形的方法将在 20.2.2 小节进行介绍。GL10 提供的用于进行初始化的方法如表 20.1 所示。

表 20.1　GL10 提供的用于进行初始化的方法

方　法	描　述
glClearColor(float red, float green, float blue, float alpha)	用于指定清除屏幕时使用的颜色，4 个参数分别用于设置红、绿、蓝和透明度的值，值的范围是 0.0f~1.0f
glDisable(int cap)	用于禁用 OpenGL ES 某个方面的特性。例如，要关闭抗抖动功能，可以使用“gl.glDisable(GL10.GL_DITHER);”语句
glEnable(int cap)	用于启用 OpenGL ES 某个方面的特性
glFrustumf(float left, float right, float bottom, float top, float zNear, float zFar)	用于设置透视视窗的空间大小
glHint(int target, int mode)	用于对 OpenGL ES 某个方面进行修正
glLoadIdentity()	用于初始化单位矩阵
glMatrixMode(int mode)	用于设置视图的矩阵模式。通常可以使用 GL10.GL_MODELVIEW 和 GL10.GL_PROJECTION 两个常量值

续表

方　法	描　述
glShadeModel(int mode)	用于设置 OpenGL ES 的阴影模式。例如，要设置为平滑模式，可以使用“gl.glShadeModel(GL10.GL_SMOOTH);”语句
glViewport(int x, int y, int width, int height)	用于设置 3D 场景的大小

20.2.2 绘制一个模型

在基本框架构建完成后，我们就可以在该框架的基础上绘制 3D 模型了。在 OpenGL ES 中，任何模型都会被分解为三角形。下面将以绘制一个 2D 的三角形为例介绍绘制 3D 模型的基本步骤。

（1）在 onSurfaceCreated() 方法中，定义顶点坐标数组。例如，要绘制一个 2D 的三角形，可以使用以下代码定义顶点坐标数组：

```
01  private final IntBuffer mVertexBuffer;
02      public GLTriangle() {
03      int one = 65536;
04      int vertices[] = {
05              0, one, 0,          //上顶点
06              -one, -one, 0,      //左下点
07              one, -one, 0        //右下点
08      };
09      ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
10      vbb.order(ByteOrder.nativeOrder());
11      mVertexBuffer = vbb.asIntBuffer();
12      mVertexBuffer.put(vertices);
13      mVertexBuffer.position(0);
14  }
```

说明 在默认的情况下，OpenGL ES采取的坐标是[0,0,0]（X,Y,Z）表示GLSurfaceView的中心；[1,1,0]表示GLSurfaceView的右上角；[-1,-1,0]表示GLSurfaceView的左下角。

（2）在 onSurfaceCreated() 方法中，应用以下代码启用顶点坐标数组。

```
gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);          //启用顶点坐标数组
```

（3）在 onDrawFrame() 方法中，应用步骤（1）定义的顶点坐标数组绘制图形。例如，要绘制一个三角形可以使用下面的代码。

```
01  gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);    //为画笔指定顶点坐标数据
02  gl.glColor4f(1, 0, 0, 0.5f);                               //设置画笔颜色
03  gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 3);             //绘制图形
```

在了解了应用 OpenGL ES 绘制 3D 图形的基本步骤后，下面将介绍一个具体的实例来介绍如何绘制一个立方体。

例 20.1　实现绘制一个 6 个面采用不同颜色的立方体

在 Android Studio 中创建一个 Module，名称为“Color Cube”。实现本实例的具体步骤如下：

（1）在默认创建的 MainActivity 中，创建一个 GLSurfaceView 类型的成员变量，关键代码如下：

```
private GLSurfaceView mGLView;                    //定义GLSurfaceView
```

（2）在重写的 onCreate() 方法中，首先创建一个 GLSurfaceView 对象，然后为 GLSurfaceView 指定使用的 Renderer 对象，最后再设置 Activity 显示的内容为 GLSurfaceView 对象，关键代码如下：

```
01 @Override
02 protected void onCreate(Bundle savedInstanceState) {
03     super.onCreate(savedInstanceState);
04     mGLView = new GLSurfaceView(this);       //创建一个GLSurfaceView对象
05     mGLView.setRenderer(new CubeRenderer());//为GLSurfaceView指定使用的Renderer对象
06     setContentView(mGLView);                 //设置Activity显示的内容为GLSurfaceView对象
07 }
```

（3）重写 onResume() 和 onPause() 方法，具体代码如下：

```
01 @Override
02 protected void onResume() {
03     super.onResume();
04     mGLView.onResume();
05 }
06 @Override
07 protected void onPause() {
08     super.onPause();
09     mGLView.onPause();
10 }
```

（4）创建 Utility 类，在该类中创建一个名称为 FloatBuffer 的静态方法，该方法用于创建顶点数据的缓冲，具体代码如下：

```
01 public class Utility {
02     //创建顶点数据缓冲
03     public static FloatBuffer createFloatBuffer(float data[])
04     {
05         ByteBuffer vbb=ByteBuffer.allocateDirect(data.length*4);    //创建数据缓冲
06         vbb.order(ByteOrder.nativeOrder());                         //设置字节顺序
07         FloatBuffer outBuffer=vbb.asFloatBuffer();                  //转换为float型缓冲
08         outBuffer.put(data).position(0);//向缓冲中放入顶点坐标数据并设置缓冲区的起始位置
09         return outBuffer;
10     }
11 }
```

（5）创建一个实现 GLSurfaceView.Renderer 接口的 CubeRenderer 类，并实现 onSurfaceCreated()、onDrawFrame() 和 onSurfaceChanged() 方法，具体代码如下：

```
01 import javax.microedition.khronos.egl.EGLConfig;
02 import javax.microedition.khronos.opengles.GL10;
03 import android.opengl.GLSurfaceView;
```

```
public class CubeRenderer implements GLSurfaceView.Renderer {
    @Override
    public void onDrawFrame(GL10 gl) {
    }
    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
    }
    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    }
}
```

（6）在 CubeRenderer 类中，首先定义一个顶点坐标的数据缓冲，然后创建一个记录顶点位置的数组。具体代码如下：

```
 FloatBuffer mVertexBuffer;          //顶点坐标数据缓冲
//立方体顶点坐标
private float box[] = {
        //前
        -0.5f, -0.5f,  0.5f, 0.5f, -0.5f,  0.5f,
        -0.5f,  0.5f,  0.5f, 0.5f,  0.5f,  0.5f,
        //后
        -0.5f, -0.5f, -0.5f, -0.5f,  0.5f, -0.5f,
        0.5f, -0.5f, -0.5f, 0.5f,  0.5f, -0.5f,
        //左
        -0.5f, -0.5f,  0.5f, -0.5f,  0.5f,  0.5f,
        -0.5f, -0.5f, -0.5f, -0.5f,  0.5f, -0.5f,
        //右
        0.5f, -0.5f, -0.5f, 0.5f,  0.5f, -0.5f,
        0.5f, -0.5f,  0.5f, 0.5f,  0.5f,  0.5f,
        //上
        -0.5f,  0.5f,  0.5f, 0.5f,  0.5f,  0.5f,
        -0.5f,  0.5f, -0.5f, 0.5f,  0.5f, -0.5f,
        //下
        -0.5f, -0.5f,  0.5f, -0.5f, -0.5f, -0.5f,
        0.5f, -0.5f,  0.5f, 0.5f, -0.5f, -0.5f
};
```

（7）在 onSurfaceCreated() 方法中，应用以下代码进行初始化操作，主要包括转换顶点坐标的数据类型、设置窗体背景颜色、设置阴影平滑模式、设置深度缓存、启用深度测试及设置深度测试的类型、最后设置系统对透视进行修正。具体代码如下：

```
@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    //将顶点坐标转换为float类型的数据
    mVertexBuffer= Utility.createFloatBuffer(box);
    gl.glClearColor(0.7f, 0.9f, 0.9f, 1.0f); //设置窗体背景颜色
```

```
06      //设置阴影平滑模式
07      gl.glShadeModel(GL10.GL_SMOOTH);
08      //设置深度缓存
09      gl.glClearDepthf(1.0f);
10      //启动深度测试
11      gl.glEnable(GL10.GL_DEPTH_TEST);
12      //深度测试的类型
13      gl.glDepthFunc(GL10.GL_LEQUAL);
14      //告诉系统对透视进行修正
15      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_NICEST);
16  }
```

说明 深度测试就是让OpenGL ES负责跟踪每个物体在Z轴上的深度，这样可避免后面的物体遮挡前面的物体。

（8）在 onSurfaceChanged 方法中，首先设置 OpenGL 场景的大小，然后将当前矩阵模式设为投影矩阵，再初始化单位矩阵并设置窗口比例和透视图，最后设置模型矩阵与重置模型矩阵，具体代码如下：

```
01  @Override
02  public void onSurfaceChanged(GL10 gl, int width, int height) {
03      //设置场景大小
04      gl.glViewport(0, 0, width, height);
05      //设置投影矩阵
06      gl.glMatrixMode(GL10.GL_PROJECTION);
07      //初始化单位矩阵
08      gl.glLoadIdentity();
09      //设置窗口比例和透视图
10      GLU.gluPerspective(gl, 45.0f, (float)width/(float)height, 1f, 100.0f);
11      //设置模型矩阵
12      gl.glMatrixMode(GL10.GL_MODELVIEW);
13      //重置模型矩阵
14      gl.glLoadIdentity();
15  }
```

（9）在 onDrawFrame() 方法中，首先清除颜色缓存和深度缓存并重置模型矩阵，然后开启顶点坐标功能并设置顶点坐标，再设置视点，并旋转总坐标系，最后绘制立方体。具体代码如下：

```
01    @Override
02    public void onDrawFrame(GL10 gl) {
03        //清除颜色缓存和深度缓存
04        gl.glClear(GL10.GL_COLOR_BUFFER_BIT|GL10.GL_DEPTH_BUFFER_BIT);
05        gl.glMatrixMode(GL10.GL_MODELVIEW);          //设置使用模型矩阵进行变换
06        //重置模型矩阵
07        gl.glLoadIdentity();
08        //开启顶点坐标功能
09        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
```

```
10          //设置顶点坐标
11          gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mVertexBuffer);
12          //当使用GL_MODELVIEW模式时，必须设置视点，也就是观察点
13  GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
14          gl.glRotatef(1000, -0.1f, -0.1f, 0.05f); //旋转总坐标系
15          //绘制FRONT和BACK两个面
16          gl.glColor4f(1, 0, 0, 1);
17          gl.glNormal3f(0, 0, 1);
18          //绘制第一个立方体面
19          gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
20          gl.glColor4f(1, 0, 0.5f, 1);
21          gl.glNormal3f(0, 0, -1);
22          //绘制第二个立方体面
23          gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 4);
24          //绘制LEFT和RIGHT两个面
25          gl.glColor4f(0, 1, 0, 1);
26          gl.glNormal3f(-1, 0, 0);
27          //绘制第三个立方体面
28          gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 8, 4);
29          gl.glColor4f(0, 1, 0.5f, 1);
30          gl.glNormal3f(1, 0, 0);
31          //绘制第四个立方体面
32          gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 12, 4);
33          //绘制TOP和BOTTOM两个面
34          gl.glColor4f(0, 0, 1, 1);
35          gl.glNormal3f(0, 1, 0);
36          //绘制第五个立方体面
37          gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 16, 4);
38          gl.glColor4f(0, 0, 0.5f, 1);
39          gl.glNormal3f(0, -1, 0);
40          //绘制第六个立方体面
41          gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 20, 4);
42      }
```

（10）运行本实例，如图 20.1 所示。

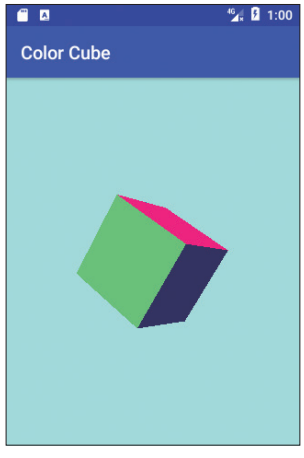


图 20.1　绘制一个立方体

20.3 添加效果

在上一节中我们已经介绍了如何绘制3D模型，在实际应用开发时，经常需要为其添加纹理贴图、光照、旋转等效果。这节我们将介绍如何为 3D 模型添加纹理贴图、光照、旋转以及透明效果等。

20.3.1 应用纹理贴图

为了让 3D 图形更加逼真，我们需要为这些 3D 图形应用纹理贴图。例如，要在场景中放置一个木箱，那么就需要为场景中绘制的立方体应用木材纹理进行贴图。为 3D 模型添加纹理贴图大致可以分为以下 3 个步骤：

（1）设置贴图坐标的数组信息，这与设置顶点坐标数组类似。

（2）设置启用贴图坐标数组。

（3）调用 GL10 的 texImage2D() 方法生成纹理。

说明 在使用纹理贴图时，需要准备一张纹理图片，建议该图片的长宽是2的N次方，例如，可以是256*256的图片，也可以是512*512的图片。

下面通过一个实例来演示纹理贴图的应用。

例 20.2 对立方体进行纹理贴图

在实例 20.1 的基础上为绘制的立方体进行纹理贴图，具体步骤如下：

（1）打开 CubeRenderer 类文件，首先定义用于保存纹理贴图数据缓冲的成员变量，然后定义贴图坐标数组。再创建一个用于传递上下文的构造方法。具体代码如下：

```
01   FloatBuffer mTextureBuffer;          //纹理贴图数据缓冲
02  //立方体纹理坐标
03  private float texCoords[] = {
04      //前
05      0.0f, 0.0f, 1.0f, 0.0f,
06      0.0f, 1.0f, 1.0f, 1.0f,
07      //后
08      1.0f, 0.0f, 1.0f, 1.0f,
09      0.0f, 0.0f, 0.0f, 1.0f,
10      //左
11      1.0f, 0.0f, 1.0f, 1.0f,
12      0.0f, 0.0f, 0.0f, 1.0f,
13      //右
14      1.0f, 0.0f, 1.0f, 1.0f,
15      0.0f, 0.0f, 0.0f, 1.0f,
16      //上
17      0.0f, 0.0f, 1.0f, 0.0f,
18      0.0f, 1.0f, 1.0f, 1.0f,
```

```
19      //下
20      1.0f, 0.0f, 1.0f, 1.0f,
21      0.0f, 0.0f, 0.0f, 1.0f
22  };
23
24  Context context;     //定义上下文
25  //创建传递上下文的构造方法
26  public CubeRenderer(Context context) {
27      this.context = context;
28  }
```

（2）创建 loadTexture() 方法，在该方法中首先加载位图文件，然后使用图片生成纹理，再设置纹理过滤器，最后释放资源。具体代码如下：

```
01  private void loadTexture(GL10 gl, Context context, int resource) {
02       Bitmap bmp = BitmapFactory.decodeResource(context.getResources(),
03               resource);     //加载位图
04       GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);     //使用图片生成纹理
05       gl.glTexParameterf(GL10.GL_TEXTURE_2D,
06               GL10.GL_TEXTURE_MIN_FILTER,GL10.GL_NEAREST);
07       gl.glTexParameterf(GL10.GL_TEXTURE_2D,
08               GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_NEAREST);
09       bmp.recycle();     //释放资源
10   }
```

（3）在 onSurfaceCreated() 方法中，添加以下代码，首先将纹理贴图的顶点坐标转换为 float 类型的数据，然后开启纹理功能，最后调用 loadTexture() 方法加载纹理图片。关键代码如下：

```
01  //将纹理贴图的顶点坐标转换为float类型的数据
02  mTextureBuffer=Utility.createFloatBuffer(texCoords);
03  //开启纹理功能
04  gl.glEnable(GL10.GL_TEXTURE_2D);
05  //加载纹理
06  loadTexture(gl,context,R.mipmap.mr);
```

（4）在 onDrawFrame() 方法中，添加以下代码，首先开启纹理顶点坐标，然后设置纹理顶点坐标。关键代码如下：

```
01  //开启纹理顶点坐标
02  gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
03  //设置纹理顶点坐标
04  gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
```

（5）打开 MainActivity 类，在 onCreate() 方法中将为 GLSurfaceView 指定使用的 Renderer 对象添加上下文参数 this，关键代码如下：

```
mGLView.setRenderer(new CubeRenderer(this));    //为GLSurfaceView指定使用的Renderer对象
```

（6）运行本实例，将显示如图 20.2 所示的运行结果。

图 20.2　为立方体进行纹理贴图

说明　在运行本实例之前，需要在CubeRenderer类的onDrawFrame ()方法中将例20.1中立方体每一面的颜色绘制代码删除。

20.3.2　旋转

到目前为止，我们绘制的 3D 物体还是静止的，为了更好地看到 3D 效果，还可以为其添加旋转效果，这样就可达到动画效果了。要实现旋转比较简单，只需要使用 GL10 的 glRotatef() 方法不断旋转要放置的对象即可。glRotatef() 方法的语法格式如下：

```
glRotatef(float angle, float x, float y, float z)
```

其中，参数 angle 通常是一个变量，表示对象转过的角度；x 表示 X 轴的旋转方向（值为 1 表示顺时针、-1 表示逆时针方向、0 表示不旋转）；y 表示 Y 轴的旋转方向（值为 1 表示顺时针、-1 表示逆时针方向、0 表示不旋转）；z 表示 Z 轴的旋转方向（值为 1 表示顺时针、-1 表示逆时针方向、0 表示不旋转）。

例如，要将对象经过 X 轴旋转 n 角度，可以使用下面的代码：

```
gl.glRotatef(n, 1, 0, 0);
```

下面通过一个实例来演示 3D 物体旋转的应用。

例 20.3　实现一个不断旋转的立方体

在实例 20.2 的基础上实现一个不断旋转的立方体，具体步骤如下：

（1）打开 CubeRenderer 类文件，在该类中定义，用于保存开始时间的成员变量，具体代码如下：

```
private long startTime;                    //保存开始时间
```

（2）在构造方法中，为成员变量 startTime 赋初始值为当前时间，具体代码如下：

```
startTime=System.currentTimeMillis();      //系统当前时间
```

（3）在 onDrawFrame() 方法的绘制立方体的代码之前，添加以下代码，完成旋转立方体的操作。

```
//旋转
long elapsed = System.currentTimeMillis() - startTime;        //计算逝去的时间
gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);               //在y轴上旋转30度
gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);               //在x轴上旋转15度
```

（4）运行本实例，将显示如图 20.3 所示的运行结果。

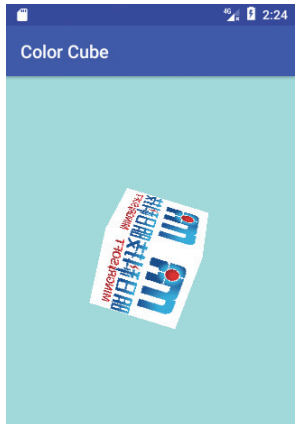


图 20.3　旋转的立方体

20.3.3 光照效果

为了使用程序效果更加美观、逼真，还可以让其模拟光照效果。在为物体添加光照效果前，我们先来了解一下 3D 图形支持的光照类型。所有的 3D 图形都支持以下 3 种光照类型：

☑ 环境光：一种普通的光线，光线会照亮整个场景，即使对象背对着光线也可以。

☑ 散射光：柔和的方向性光线，例如，荧光板上发出的光线就是这种散射光。场景中的大部分光线通常来源于散射光源。

☑ 镜面高光：耀眼的光线，通常来源于明亮的点光源。与有光泽的材料结合使用时，这种光会带来高光效果，增加场景的真实感。

在 OpenGL 中，添加光照效果通常分为以下两个步骤进行。

1．光线

在定义光照效果时，通常需要定义光线，也就是为场景添加光源。这可以通过 GL10 提供的 glLightfv() 方法实现。glLightfv() 方法的语法格式如下：

```
glLightfv(int light, int pname, float[] params, int offset)
```

其中，light 表示光源的 ID，当程序中包含多个光源时，可以通过这个 ID 来区分光源；pname 表示光源的类型（参数值为 GL10.GL_AMBIENT 表示环境光、参数值为 GL10.GL_DIFFUSE 表示散射光）；params：表示光源数组；offset：表示偏移量。

例如，我们要定义一个发出白色的全方向的光源，可以使用下面的代码：

```
float lightAmbient[]=new float[]{0.2f,0.2f,0.2f,1};           //定义环境光
```

```
02  float lightDiffuse[]=new float[]{1,1,1,1};          //定义散射光
03  float lightPos[]=new float[]{1,1,1,1};              //定义光源的位置
04  gl.glEnable(GL10.GL_LIGHTING);                      //启用光源
05  gl.glEnable(GL10.GL_LIGHT0);                        //启用0号光源
06  gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient,0); //设置环境光
07  gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);//设置散射光
08  gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);   //设置光源的位置
```

注意 在定义和设置光源后，还需要使用glEnable()方法启用光源，否则，设置的光源将不起作用。

2. 被照射的物体

在定义光照效果时，通常需要定义被照射物体的制作材料，因为不同材料的光线反射情况是不同的。使用 GL10 提供的 glMaterialfv() 方法可以设置材质的环境光和散射光。glMaterialfv() 方法的语法格式如下：

```
glMaterialfv(int face, int pname, float[] params, int offset)
```

其中，face 表示为正面还是背面材质设置光源；pname 表示光源的类型（参数值为 GL10.GL_AMBIENT 表示环境光、参数值为 GL10.GL_DIFFUSE 表示散射光）；params：表示光源数组；offset：表示偏移量。

例如，定义一个不是很亮的纸质的物体，可以使用下面的代码：

```
01  float matAmbient[]=new float[]{1,1,1,1};            //定义材质的环境光
02  float matDiffuse[]=new float[]{1,1,1,1};            //定义材质的散射光
03  //设置材质的环境光
04  gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, matAmbient,0);
05  //设置材质的散射光
06  gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, matDiffuse,0);
```

下面通过一个具体的实例来演示为物体添加光照效果。

例 20.4 实现为旋转的立方体添加光照效果的功能

在实例 20.3 的基础上实现为旋转的立方体添加光照效果的功能，具体步骤如下：

（1）打开 CubeRenderer 类文件，在 onSurfaceCreated() 方法中为被照射的物体设置材质。首先定义材质的环境光和散射光，然后设置材质的环境光和散射光，具体代码如下：

```
01  float matAmbient[]=new float[]{1,1,1,1};            //定义材质的环境光
02  float matDiffuse[]=new float[]{1,1,1,1};            //定义材质的散射光
03  //设置材质的环境光
04  gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, matAmbient,0);
05  //设置材质的散射光
06  gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, matDiffuse,0);
```

（2）在 onSurfaceCreated() 方法中添加场景光线。首先定义环境光和散射光，并定义光源的位置，然后启用光源和 0 号光源，最后设置环境光、散射光和光源的位置，具体代码如下：

```
01  float lightAmbient[]=new float[]{0.2f,0.2f,0.2f,1};          //定义环境光
02  float lightDiffuse[]=new float[]{1,1,1,1};                   //定义散射光
03  float lightPos[]=new float[]{1,1,1,1};                       //定义光源的位置
04  gl.glEnable(GL10.GL_LIGHTING);                               //启用光源
05  gl.glEnable(GL10.GL_LIGHT0);                                 //启用0号光源
06  gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient,0); //设置环境光
07  gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);//设置散射光
08  gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);   //设置光源的位置
```

（3）运行本实例，将显示如图 20.4 所示的运行结果。

图 20.4　为立方体添加光照效果

20.3.4 透明效果

在游戏中，经常需要应用透明效果，使用 OpenGL ES 实现简单的透明效果也比较简单，只需要应用以下代码就可以实现。

```
01  gl.glDisable(GL10.GL_DEPTH_TEST);              //关闭深度测试
02  gl.glEnable(GL10.GL_BLEND);                    //打开混合
03  //使用alpha通道值进行混色，从而达到透明效果
04  gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
```

说明 实现透明效果时，需要关闭深度测试，并且打开混合效果，然后才能使用GL10类的glBlendFunc()方法进行混色，从而达到透明效果。

下面通过一个具体的实例来演示实现透明效果。

例 20.5　实现透明旋转的立方体

在实例 20.4 的基础上实现一个透明旋转的立方体，具体步骤如下：

打开 CubeRenderer 类文件，在 onSurfaceCreated() 方法中为立方体添加透明效果。首先修改背景颜色为蓝色，然后关闭深度测试，再打开混合效果，最后再使用 alpha 通道值进行混色，从而达到透明效果，具体代码如下：

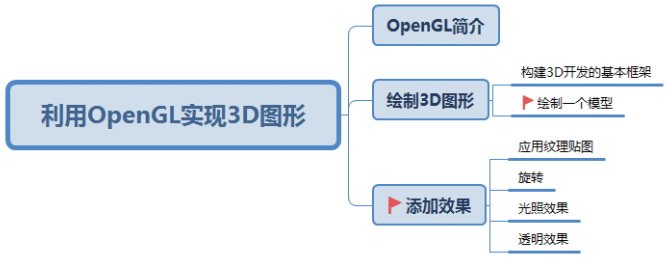
```
01  gl.glClearColor(0.08f, 0.16f, 0.39f, 1.0f); //设置窗体背景颜色
02  gl.glDisable(GL10.GL_DEPTH_TEST);           //关闭深度测试
03  gl.glEnable(GL10.GL_BLEND);                 //打开混合
04  //使用alpha通道值进行混色，从而达到透明效果
05  gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
```

运行本实例，将显示如图 20.5 所示的运行结果。

图 20.5　透明且旋转的立方体

20.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 21 章

数据存储技术

Android 系统提供了多种数据存储方法。例如，使用 SharedPreferences 进行简单存储、文件存储、SQLite 数据库存储以及使用 Content Provider 共享数据等。本章将对这几种常用的数据存储方法进行详细介绍。

21.1 SharedPreferences 存储

Android 系统提供了轻量级的数据存储方式——SharedPreferences 存储。它屏蔽了对底层文件的操作，通过为程序开发人员提供简单的编程接口，实现以最简单的方式对数据进行永久保存。这种方式主要对少量的数据进行保存，比如对应用程序的配置信息、手机应用的主题、游戏的玩家积分等进行保存。例如，对微信进行通用设置后可以对相关配置信息进行保存，进行通用设置的界面如图 21.1 所示；对手机微博客户端设置应用主题后就可以对该主题进行保存，设置应用主题的界面如图 21.2 所示。

图 21.1　微信通用设置

图 21.2　设置微博主题

下面将对 SharedPreferences 进行详细介绍。

21.1.1 获得 SharedPreferences 对象

SharedPreferences 接口位于 android.content 包中，用于使用键值（key-value）对的方式来存储数据。该类主要用于基本类型，例如，boolean、float、int、long 和 String。在应用程序结束后，数据仍旧会保存。数据是以 XML 文件格式保存在 Android 手机系统下的“/data/data/< 应用程序包名 >/shared_prefs”目录中，该文件被称为 Shared Preference（共享的首选项）文件。

通常情况下，可以通过以下两种方式获得 SharedPreferences 对象。

1. 使用 getSharedPreferences() 方法获取

如果需要多个使用名称来区分的 SharedPreferences 文件，则可以使用 getSharedPreferences() 方法获取，该方法的基本语法格式如下：

```
getSharedPreferences(String name, int mode)
```

参数说明如下：

◆ name：共享文件的名称（不包括扩展名），该文件为 XML 格式。对于使用同一个名称获得的多个 SharedPreferences 引用，其指向同一个对象。

◆ mode：用于指定访问权限，它的参数值可以是 MODE_PRIVATE（表示只能被本应用程序读和写，其中写入的内容会覆盖原文件的内容）、MODE_MULTI_PROCESS（表示可以跨进程、跨应用读取）。

2. 使用 getPreferences() 方法获取

如果 Activity 仅需要一个 SharedPreferences 文件，则可以使用 getPreferences() 方法获取。因为只有一个文件，所以不需要提供名称。该方法的语法格式如下：

```
getPreferences(int mode)
```

其中，参数 mode 的取值同 getSharedPreferences() 方法相同。

21.1.2 向 SharedPreferences 文件存储数据

完成向 SharedPreferences 文件中存储数据的步骤如下：

（1）调用 SharedPreferences 类的 edit() 方法获得 SharedPreferences.Editor 对象。例如，可以使用下面的代码获得私有类型的 SharedPreferences.Editor 对象。

```
SharedPreferences.Editor editor=getSharedPreferences("mr",MODE_PRIVATE).edit();
```

（2）向 SharedPreferences.Editor 对象中添加数据。例如，调用 putBoolean() 方法添加布尔型数据、调用 putString() 方法添加字符串数据、调用 putInt() 方法添加整型数据，可以使用下面的代码：

```
01  editor.putString("username", username);
02  editor.putBoolean("status", false);
03  editor.putInt("age", 20);
```

（3）使用 commit() 方法提交数据，从而完成数据存储操作。例如，提交步骤（1）获得的 SharedPreferences.Editor 对象，可以使用下面的代码：

```
editor.commit();
```

21.1.3 读取 SharedPreferences 文件中存储的数据

从 SharedPreferences 文件中读取数据时，主要使用 SharedPreferences 类的 getXxx() 方法。例如，下面的代码可以实现分别获取 String、Boolean 和 int 类型的值。

```
SharedPreferences sp = getSharedPreferences("mr", MODE_PRIVATE);
String username = sp.getString("username", "mr");      //获得用户名
Boolean status = sp.getBoolean("status", false);
int age = sp.getInt("age", 18);
```

例 21.1 使用 SharedPreferences 保存用户信息

在 Android Studio 中创建一个 Module，名称为“SharedPreferences”，实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，在该文件中通过嵌套布局的方式进行界面的布局，具体代码请参见云盘。

（2）打开主活动 MainActivity，该类继承 Activity，在该类中，定义所需的成员变量，关键代码如下：

```
private EditText et_password;          //声明输入密码控件
private EditText et_username;          //声明输入用户名控件
private String username, password;     //声明用户名密码
SharedPreferences mySharedPreferences;//声明信息保存类
```

（3）创建 onLogin() 方法，该方法为保存信息按钮的单击事件，在该事件中首先获取控件上输入的内容是否为空，并做出相应的提示信息，然后实例化 SharedPreferences 对象，并通过该对象保存用户信息，保存成功后弹出提示。代码如下：

```
public void onLogin(View view){
    //获取控件输入的内容
    username = et_username.getText().toString().trim();
    password = et_password.getText().toString().trim();
    if (TextUtils.isEmpty(username) || TextUtils.isEmpty(password)) {
        Toast.makeText(this, "用户名或密码不可为空", Toast.LENGTH_SHORT).show();
    }
    else {
        //实例化SharedPreferences对象
        mySharedPreferences= getSharedPreferences("myuserinfo",
                MainActivity.MODE_PRIVATE);
        //实例化SharedPreferences.Editor对象（第二步）
        SharedPreferences.Editor editor = mySharedPreferences.edit();
        //用putString的方法保存数据
        editor.putString("username", username);
        editor.putString("pwd", password);
        //提交当前数据
        editor.commit();
        Toast.makeText(this, "信息保存成功！", Toast.LENGTH_SHORT).show();
```

```
20      }
21  }
```

（4）创建 onShow() 方法，该方法为显示保存信息按钮的单击事件，在该事件中首先实例化 SharedPreferences 对象，并通过该对象获取保存的用户信息，然后创建 AlertDialog 对话框，将保存的用户信息通过该对话框显示出来，代码如下：

```
01  public void onShow(View v) {
02      //实例化SharedPreferences对象
03      mySharedPreferences = getSharedPreferences("myuserinfo",
04              MainActivity.MODE_PRIVATE);
05      //使用getString方法获得value，注意第2个参数是value的默认值
06      String username = mySharedPreferences.getString("username", "没有保存的数据");
07      String pwd = mySharedPreferences.getString("pwd", "没有保存的数据");
08      AlertDialog.Builder builder = new AlertDialog.Builder(
09              MainActivity.this);
10      builder.setTitle("显示保存的用户信息");
11      builder.setMessage("用户名：" +username+"\n"+"密码：" +pwd);
12      builder.setCancelable(false);
13      builder.setPositiveButton("知道了！", new DialogInterface.OnClickListener() {
14          public void onClick(DialogInterface dialog, int which) {
15              dialog.cancel();
16          }
17      });
18      builder.create().show();
19  }
```

（5）运行本实例，将显示如图 21.3 所示的登录界面，输入账号“明日学院”、密码“1234567890”，单击“保存用户信息”按钮将显示如图 21.4 所示的界面。信息保存完成后单击“显示保存信息”按钮显示如图 21.5 所示的用户信息。

图 21.3　登录界面

图 21.4　输入用户信息

图 21.5　显示保存的用户信息

说明 运行本实例，信息保存成功后，保存账号与密码的myuserinfo.xml文件将被保存在“data\data\<项目资源包名>\shared_prefs\”目录下。由于系统安全问题该文件无法查看。该文件只能在本应用中读取。

21.2 文件存储

学习过 Java SE 的读者都知道，在 Java 中提供了一套完整的 IO 流体系，通过这些 IO 流可以很方便地访问磁盘上的文件内容。在 Android 中也同样支持以这种方式来访问手机存储器上的文件。例如，对游戏需要使用的资源文件进行下载并存储在手机中的指定位置，如图 21.6 所示；再如，将下载的歌曲存储在手机的指定路径下，如图 21.7 所示。

图 21.6 下载并保存资源文件

图 21.7 已下载的歌曲文件

在 Android 中，主要提供了以下两种方式用于访问手机存储器上的文件。

（1）内部存储：使用 FileOutputStream 类提供的 openFileOutput 方法和 FileInputStream 类提供的 openFileInput() 方法访问设备内部存储器上的文件。

（2）外部存储：使用 Environment 类的 getExternalStorageDirectory() 方法对外部存储上的文件进行数据读写。

本节将对这两种方式进行详细讲解。

21.2.1 内部存储

内部存储位于 Android 手机系统下的“/data/data/< 包名 >/files”目录中。使用 Java 提供的 IO 流体系可以很方便的对内部存储的数据进行读写操作。其中，FileOutputStream 类的 openFileOutput() 方法用来打开相应的输出流；而 FileInputStream 类的 openFileInput() 方法用来打开相应的输入流。默认情况下，使用 IO 流保存的文件仅对当前应用程序可见，对于其他应用程序（包括用户）是不可见的（即不能访问其中的数据）。

说明 如果用户卸载了应用程序，则保存数据的文件也会一起被删除。

1. 写入文件

要实现向内部存储器中写入文件，首先需要获取文件输出流对象 FileOutputStream，可以通过 FileOutputStream 类的 openFileOutput() 方法来实现，然后再调用 FileOutputStream 对象的 write() 方法写入文件内容，再调用 flush() 方法清空缓存，最后调用 close() 方法关闭文件输出流对象。

FileOutputStream 类的 openFileOutput() 方法的基本语法格式如下：

```
FileOutputStream openFileOutput(String name, int mode) throws FileNotFoundException
```

◆ 参数 name：用于指定文件名称，该参数不能包含描述路径的斜杠“/”。

◆ 参数 mode：用于指定访问权限，可以使用如下取值：

◎ MODE_PRIVATE：表示文件只能被创建它的程序访问。

◎ MODE_APPEND：表示追加模式，如果文件存在，则在文件的结尾处添加新数据，否则创建文件。

◎ MODE_WORLD_READABLE：表示可以被其他应用程序读，但不能写。

◎ MODE_WORLD_WRITEABLE：表示可以被其他应用程序读和写。

openFileOutput() 方法的返回值为 FileOutputStream 对象。

注意 openFileOutput()方法需要抛出FileNotFoundException异常。

例如，创建一个只能被创建它的程序访问的文件 mr.txt，代码如下：

```
01  try {
02      FileOutputStream fos = openFileOutput("mr.txt", MODE_PRIVATE);  //获取文件输出流
03      fos.write("www.mingrisoft.com".getBytes());                     //保存网址
04      fos.flush();                                                    //清空缓存
05      fos.close();                                                    //关闭文件输出流
06  } catch (FileNotFoundException e) {
07      e.printStackTrace();
08  }
```

在上面的代码中，FileOutputStream 对象的 write() 方法用于将数据写入文件；flush() 方法用于将缓存中的数据写入文件，清空缓存；close() 方法用于关闭 FileOutputStream 对象。

2. 读取文件

要实现读取内部存储器中的文件，首先需要获取文件输入流对象 FileInputStream，可以通过 FileInputStream 类的 openFileInput() 方法来实现，然后再调用 FileInputStream 对象的 read() 方法读取文件内容，最后调用 close() 方法关闭文件输入流对象。

FileInputStream 类的 openFileInput() 方法的基本语法格式如下：

```
FileInputStream openFileInput(String name) throws FileNotFoundException
```

该方法只有一个参数，用于指定文件名称，同样不可以包含描述路径的斜杠“/”，而且也需要抛出 FileNotFoundException 异常，返回值为 FileInputStream 对象。

例如，读取文件 mr.txt 的内容，可以使用下面的代码：

```
01  FileInputStream fis = openFileInput("mr.txt");        //获取文件输入流
02  byte[ ] buffer = new byte[fis.available()];           //定义保存数据的数组
03  fis.read(buffer);                                     //从输入流中读取数据
```

下面通过一个实例演示如何使用 Java 提供的 IO 流体系对内部存储文件进行操作。

例 21.2 使用内部存储保存备忘录信息

在 Android Studio 中创建 Module，名称为“Internal Storage Notepad”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后删除 TextView 组件，再添加一个 EditText 组件用于填写备忘录信息，最后添加两个图片按钮分别用于保存与取消信息。

（2）打开主活动 MainActivity，让其继承自 Activity，然后在该类中，定义所需的成员变量，关键代码如下：

```
byte[] buffer = null;           //定义保存数据的数组
```

（3）在重写的 onCreate() 方法中，首先获取用于填写备忘录信息的编辑框组件，然后获取“保存”按钮与“取消”按钮，关键代码如下：

```
01 //获取用于填写备忘录信息的编辑框组件
02 final EditText etext = (EditText) findViewById(R.id.editText);
03 ImageButton btn_save = (ImageButton) findViewById(R.id.btn_save);       //获取保存按钮
04 ImageButton btn_cancel = (ImageButton) findViewById(R.id.btn_cancel);  //获取取消按钮
```

（4）单击保存按钮，使用内部存储将填写的备忘信息保存，关键代码如下：

```
01 btn_save.setOnClickListener(new View.OnClickListener() {
02     @Override
03     public void onClick(View v) {
04         FileOutputStream fos = null;                        //定义文件输出流
05         String text = etext.getText().toString();           //获取文本信息
06         try {
07             //获取文件输出流，并指定文件保存的位置
08             fos = openFileOutput("memo", MODE_PRIVATE);
09             fos.write(text.getBytes());                     //保存文本信息
10             fos.flush();                                    //清除缓存
11         } catch (FileNotFoundException e) {
12             e.printStackTrace();
13         } catch (IOException e) {
14             e.printStackTrace();
15         } finally {
16             if (fos != null) {                              //输出流不为空时
17                 try {
18                     fos.close();                            //关闭文件输出流
19                     Toast.makeText(MainActivity.this,
20                             "保存成功", Toast.LENGTH_SHORT).show();
21                 } catch (IOException e) {
```

```
                    e.printStackTrace();
                }
            }
        }
    });
```

（5）实现第二次打开应用时显示上一次所保存的文本信息，关键代码如下：

```
FileInputStream fis = null;                          //定义文件输入流
try {
    fis = openFileInput("memo");                     //获取文件输入流
    buffer = new byte[fis.available()];              //保存数据的数组
    fis.read(buffer);                                //从输入流中读取数据
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (fis != null) {                               //输入流不为空时
        try {
            fis.close();                             //关闭输入流
            String data = new String(buffer);        //获取数组中保存的数据
            etext.setText(data);                     //将读取的数据显示到编辑框中
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

（6）单击“取消”按钮，实现退出应用，关键代码如下：

```
btn_cancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();                                    //退出应用
    }
});
```

（7）运行本实例，将显示填写备忘录信息的界面，如图 21.8 所示，输入备忘录信息后，单击“保存”按钮将提示保存成功，当第二次开启备忘录时将显示已保存的信息，单击“取消”按钮将退出备忘录。

说明 运行本实例，保存备忘录信息后，memo文件将被保存在“data\data\<项目资源包名>\files\”目录下。由于系统安全问题该文件无法查看，只能在本应用中读取。

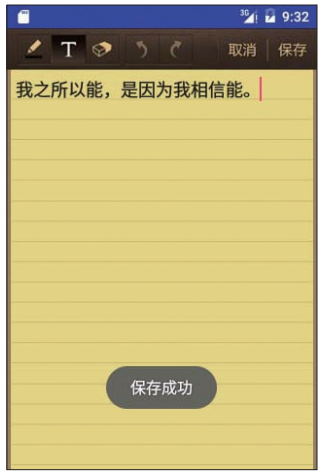


图 21.8　内部存储保存备忘录信息

21.2.2 外部存储

每个 Android 设备都支持共享的外部存储用来保存文件，它也是手机中的存储介质。保存在外部存储的文件都是全局可读的，而且在用户使用 USB 连接电脑后，可以修改这些文件。在 Android 程序中，对外部存储的文件进行操作时，需要使用 Environment 类的 getExternalStorageDirectory() 方法，该方法用来获取外部存储器的目录。

说明　为了读、写外部存储上的数据，必须在应用程序的全局配置文件（AndroidManifest.xml）中添加读、写外部存储的权限。配置如下：

```
01  <!--开启在外部存储中创建与删除文件权限-->
02  <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
03  <!--开启向外部存储写入数据权限 -->
04  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

下面将通过一个实例来演示如何在外部存储上创建文件。

例 21.3　保存信息到 SD 卡

在 Android Studio 中创建 Module，名称为“External Storage”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，将默认添加的布局管理器修改为垂直线性布局管理器，然后通过嵌套布局的方式进行界面的布局，具体代码请参见云盘。

（2）创建 MPermissionsActivity 类，该类继承自 AppCompatActivity 类，用于动态获取 SD 读写权限，具体代码请参见云盘。

（3）打开主活动 MainActivity，让其继承 MPermissionsActivity，然后在该类中，定义所需的成员变量，然后在 onCreate() 方法中，使用 File 类在 SD 卡根目录中创建文件保存的位置，关键代码如下：

```
file = new File(Environment.getExternalStorageDirectory(), FILENAME);
```

（4）创建 writesd() 方法，该方法用于实现将输入的信息保存在 SD 卡的指定路径下，代码如下：

```
01  private void writesd() {
02      if (Environment.getExternalStorageState().equals(
03              Environment.MEDIA_MOUNTED)) {
04          try {
05              //写入文件流
06              FileOutputStream fos = new FileOutputStream(file);
07              fos.write(et1.getText().toString().getBytes());
08              //关闭写入流
09              fos.close();
10              //弹出提示信息
11              Toast.makeText(MainActivity.this, "写入文件成功",
12                      Toast.LENGTH_LONG).show();
13          } catch (Exception e) {
14              //弹出提示信息
15              Toast.makeText(MainActivity.this, "写入文件失败",
16                      Toast.LENGTH_SHORT).show();
17          }
18      } else {
19          //此时SDcard不存在或者不能进行读写操作的
20          Toast.makeText(MainActivity.this,
21                  "此时SDcard不存在或者不能进行读写操作", Toast.LENGTH_SHORT).show();
22      }
23  }
```

（5）创建 readsd() 方法，该方法用于实现读取 SD 卡上所保存的信息文件，并通过提示框显示所保存的信息，代码如下：

```
01  private void readsd() {
02      if (Environment.getExternalStorageState().equals(
03              Environment.MEDIA_MOUNTED)) {
04          try {
05              FileInputStream inputStream = new FileInputStream(file);
06              byte[] b = new byte[inputStream.available()];
07              //读取文件流
08              inputStream.read(b);
09              String nr = new String(b);
10              if (!TextUtils.isEmpty(nr)) {
11                  AlertDialog.Builder builder = new AlertDialog.Builder(
12                          MainActivity.this);
13                  builder.setTitle("读取文档信息");
14                  builder.setMessage("文档内容:  " + nr);
15                  builder.setCancelable(false);
16                  builder.setPositiveButton("知道了！",
17                                                  new DialogInterface.OnClickListener() {
18                      public void onClick(DialogInterface dialog, int which) {
19                          dialog.cancel();
20                      }
21                  });
22                  builder.create().show();
```

```
            } else {
                //弹出提示信息
                Toast.makeText(MainActivity.this, "文档内容为空,请输入要保存信息！",
                        Toast.LENGTH_SHORT).show();
            }

        } catch (Exception e) {
            //弹出提示信息
            Toast.makeText(MainActivity.this, "读取失败",
                    Toast.LENGTH_SHORT).show();
        }
    } else {
        //此时SDcard不存在或者不能进行读写操作的
        Toast.makeText(MainActivity.this,
                "此时SDcard不存在或者不能进行读写操作", Toast.LENGTH_SHORT).show();
    }
}
```

（6）打开 AndroidManifest.xml 文件，在其中设置外部存储的读取与写入权限，具体代码如下：

```
<!--读取sd卡权限-->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<!--写入sd卡权限-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

（7）运行本实例，输入需要保存的文本内容，单击内容写入 SD 卡文档按钮将显示如图 21.9 所示的写入文件成功的提示信息。然后单击右上角的显示保存文本内容的按钮将显示如图 21.10 所示的对话框。

图 21.9　写入成功的提示信息

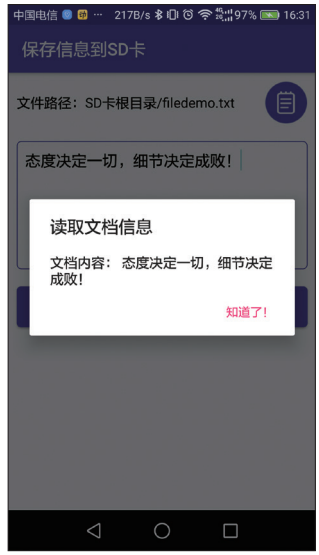


图 21.10　显示读取文档信息的对话框

说明　运行本实例，保存备忘录信息后，filedemo.text文件将被保存在“\storage\emulated\0\”目录下。该文件可通过文件管理器查看。

21.3 数据库存储

Android 系统集成了一个轻量级的关系数据库——SQLite。它不像 Oracle、MySQL 和 SQL Server 等那样专业，但是因为它占用资源少，运行效率高、安全可靠、可移植性强，并且提供零配置运行模式，适用于在资源有限的设备（如手机和平板电脑等）上进行数据存取。

在开发手机应用时，一般会通过代码来动态创建数据库，即在程序运行时，首先尝试打开数据库，如果数据库不存在，则自动创建该数据库，然后再打开数据库。下面介绍如何通过代码来创建以及操作数据库。

21.3.1 创建数据库

在 Android 的 SQLite 数据库中，提供了两种创建数据库的方法，下面分别进行介绍：

◆ 使用 openOrCreateDatabase 方法创建数据库

Android 提供了 SQLiteDatabase，用于表示一个数据库，应用程序只要获得了代表数据库的 SQLiteDatabase 对象，就可以通过 SQLiteDatabase 对象来创建数据库。SQLiteDatabase 提供了 openOrCreateDatabase() 方法来打开或创建一个数据库，语法如下：

```
static SQLiteDatabase openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)
```

path：用于指定数据库文件；

factory：用于实例化一个游标。

说明 游标提供了一种从表中检索数据并进行操作的灵活手段，通过游标可以一次处理查询结果集中的一行，并可以对该行数据执行特定操作。

例如，使用 openOrCreateDatabase() 方法创建一个名称为 user.db 的数据库的代码如下：

```
SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase("user.db", null);
```

◆ 通过 SQLiteOpenHelper 类创建数据库

在 Android 中，提供了一个数据库辅助类 SQLiteOpenHelper。在该类的构造器中，调用 Context 中的方法创建并打开一个指定名称的数据库。我们在应用这个类时，需要编写继承自 SQLiteOpenHelper 类的子类，并且重写 onCreate() 和 onUpgrade() 方法。

21.3.2 数据操作

最常用的数据操作是指添加、删除、更新和查询。对于这些操作程序开发人员完全可以通过执行 SQL 语句来完成。但是这里推荐使用 Android 提供的专用类和方法来实现。SQLiteDatabase 类提供了 insert()、update()、delete() 和 query() 方法，这些方法封装了执行添加、更新、删除和查询操作的 SQL 命令，所以我们可以使用这些方法来完成对应的操作，而不用去编写 SQL 语句了。

（1）添加操作

SQLiteDatabase 类提供了 insert() 方法用于向表中插入数据。insert() 方法的基本语法格式如下：

```
public long insert (String table, String nullColumnHack, ContentValues values)
```

◆ table：用于指定表名。

◆ nullColumnHack：可选的，用于指定当 values 参数为空时，将哪个字段设置为 null，如果 values 不为空，则该参数值可以设置为 null。

◆ values：用于指定具体的字段值，它相当于 Map 集合，也是通过键值对的形式存储值的。

（2）更新操作

SQLiteDatabase 类提供了 update() 方法用于更新表中的数据。update() 方法的基本语法格式如下：

```
public int update(String table, ContentValues values, String whereClause, String[] whereArgs)
```

◆ table：用于指定表名。

◆ values：用于指定要更新的字段及对应的字段值，它相当于 Map 集合，也是通过键值对的形式存储值的。

◆ whereClause：用于指定条件语句，可以使用占位符（?）。

◆ whereArgs：当条件表达式中包含占位符（?）时，该参数用于指定各占位参数的值。如果不包括占位符，该参数值可以设置为 null。

（3）删除操作

SQLiteDatabase 类提供了 delete() 方法用于从表中删除数据。delete() 方法的基本语法格式如下：

```
public int delete(String table, String whereClause, String[] whereArgs)
```

◆ table：用于指定表名。

◆ whereClause：用于指定条件语句，可以使用占位符（?）。

◆ whereArgs：当条件表达式中包含占位符（?）时，该参数用于指定各占位参数的值。如果不包括占位符，该参数值可以设置为 null。

（4）查询操作

SQLiteDatabase 类提供了 query() 方法用于查询表中的数据。query() 方法的基本语法格式如下：

```
public Cursor query(boolean distinct, String table, String[] columns,
        String selection, String[] selectionArgs, String groupBy,
        String having, String orderBy, String limit)
```

◆ table：用于指定表名。

◆ columns：用于指定要查询的列。若为空，则返回所有列。

◆ selection：用于指定 where 子句，即指定查询条件，可以使用占位符（?）。

◆ selectionArgs：where 子句对应的条件值，当条件表达式中包含占位符（?）时，该参数用于指定各占位参数的值。如果不包括占位符，该参数值可以设置为 null。

◆ groupBy：用于指定分组方式。

◆ having：用于指定 having 条件。

◆ orderBy：用于指定排序方式，为空表示采用默认排序方式。

◆ limit：用于限制返回的记录条数，为空表示不限制。

query() 方法的返回值为 Cursor 对象。该对象中保存着查询结果，但是这个结果并不是数据集合的完整复制，而是数据集的指针。通过它提供的多种移动方式，我们可以获取数据集合中的数据。Cursor 类提供的常用方法如表 21.1 所示。

表 21.1 Cursor 类提供的常用方法

方 法	说 明
moveToFirst()	将指针移动到第一条记录上
moveToNext()	将指针移动到下一条记录上
moveToPrevious()	将指针移动到上一条记录上
getCount()	获取集合的记录数量
getColumnIndexOrThrow()	返回指定字段名称的序号，如果字段不存在，则产生异常
getColumnName()	返回指定序号的字段名称
getColumnNames()	返回字段名称的字符串数组
getColumnIndex()	根据字段名称返回序号
moveToPosition()	将指针移动到指定的记录上
getPosition()	返回当前指针的位置

下面通过一个实例来演示如何通过代码创建和操作数据库。

例 21.4 使用数据库模拟中英文词典

在 Android Studio 中创建 Module，名称为“Database Dictionary”，实现本实例的具体步骤如下：

（1）在 com.mingrisoft 包中，创建一个名称为 DBOpenHelper 的 Java 类，让它继承 SQLiteOpenHelper 类，并且重写 onCreate() 方法、onUpgrade() 方法和 DBOpenHelper() 构造方法，用于创建一个 SQLite3 数据库，具体代码如下：

```
01  public class DBOpenHelper extends SQLiteOpenHelper {
02      //定义创建数据表dict的SQL语句
03      final String CREATE_TABLE_SQL =
04              "create table dict(_id integer primary " +
05                      "key autoincrement , word , detail)";
06      public DBOpenHelper(Context context, String name,
07                          SQLiteDatabase.CursorFactory factory, int version) {
08          super(context, name, null, version);      //重写构造方法并设置工厂为null
09      }
10      @Override
11      public void onCreate(SQLiteDatabase db) {
12          db.execSQL(CREATE_TABLE_SQL);             //创建单词信息表
13      }
14      @Override
15      //重写基类的onUpgrade()方法，以便数据库版本更新
16      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
17          //提示版本更新并输出旧版本信息与新版本信息
18          System.out.println("---版本更新-----" + oldVersion + "--->" + newVersion);
19      }
20  }
```

（2）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，并删除 TextView 组件；然后在该布局管理器中添加一个编辑框组件，用于填写要翻译的单词；再添加一个图片按钮用于翻译；最后添加一个 ListView 组件，用于显示翻译结果。

（3）创建一个名称为 result_main.xml 的布局文件，在该布局文件中采用垂直线性布局管理器，并添加 4 个 TextView 组件，用于显示翻译单词的结果。

（4）打开主活动 MainActivity，在该类中定义所需的成员变量，关键代码如下：

```
private DBOpenHelper dbOpenHelper;            //定义DBOpenHelper
```

（5）在重写的 onCreate() 方法中，创建 DBOpenHelper 对象，指定名称、版本号并保存在“databases 目录下”，关键代码如下：

```
01  dbOpenHelper = new DBOpenHelper(MainActivity.this, "dict.db", null, 1);
02  //获取显示结果的ListView
03  final ListView listView = (ListView) findViewById(R.id.result_listView);
04  //获取查询内容的编辑框
05  final EditText etSearch = (EditText) findViewById(R.id.search_et);
06  //获取查询按钮
07  ImageButton btnSearch = (ImageButton) findViewById(R.id.search_btn);
08  //获取跳转添加生词界面的按钮
09  Button btn_add = (Button) findViewById(R.id.btn_add);
```

（6）单击“添加生词”按钮，实现跳转到添加生词的界面，关键代码如下：

```
01  btn_add.setOnClickListener(new View.OnClickListener() {
02      @Override
03      public void onClick(View v) {
04          //通过Intent跳转添加生词界面
05          Intent intent = new Intent(MainActivity.this, AddActivity.class);
06          startActivity(intent);
07      }
08  });
```

（7）创建一个 Empty Activity 界面，名称为 AddActivity，修改布局文件 activity_add.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器；然后在该布局管理器中添加两个编辑框组件，分别用于填写添加词库中的单词与解释；再添加一个水平线性布局管理器，在该布局管理器中添加两个图片按钮，用于保存和取消。

（8）打开 AddActivity 类，在该类中定义所需的成员变量，关键代码如下：

```
private DBOpenHelper dbOpenHelper;                    //定义DBOpenHelper,用于与数据库连接
```

（9）在重写的 onCreate() 方法中，创建 DBOpenHelper 对象，指定名称、版本号并保存在 databases 目录下，关键代码如下：

```
01  dbOpenHelper = new DBOpenHelper(AddActivity.this, "dict.db", null, 1);
02  //获取添加单词的编辑框
03  final EditText etWord = (EditText) findViewById(R.id.add_word);
04  //获取添加解释的编辑框
```

```
final EditText etExplain = (EditText) findViewById(R.id.add_interpret);
ImageButton btn_Save = (ImageButton) findViewById(R.id.save_btn);       //获取保存按钮
ImageButton btn_Cancel = (ImageButton) findViewById(R.id.cancel_btn1); //获取取消按钮
```

（10）在 AddActivity 中，创建 insertData() 方法，在该方法中实现插入数据功能，具体代码如下：

```
private void insertData(SQLiteDatabase readableDatabase, String word, String explain) {
    ContentValues values = new ContentValues();
    values.put("word", word);                          //保存单词
    values.put("detail", explain);                     //保存解释
    readableDatabase.insert("dict", null, values);  //执行插入操作
}
```

（11）在重写的 onCreate() 方法中，单击“保存”按钮，实现将添加的单词解释保存在数据库中，关键代码如下：

```
btn_Save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String word = etWord.getText().toString();       //获取填写的生词
        String explain = etExplain.getText().toString();//获取填写的解释
        if (word.equals("") || explain.equals("")) {     //如果填写的单词或者解释为空时
            Toast.makeText(AddActivity.this,
                    "填写的单词或解释为空", Toast.LENGTH_SHORT).show();
        } else {
            //调用insertData()方法，实现插入生词数据
            insertData(dbOpenHelper.getReadableDatabase(), word, explain);
            //显示提示信息
            Toast.makeText(AddActivity.this,
                    "添加生词成功！", Toast.LENGTH_LONG).show();
        }

    }
});
```

（12）单击“取消”按钮，实现跳转到查询单词界面，关键代码如下：

```
btn_Cancel.setOnClickListener(new View.OnClickListener() {  //实现返回查询单词界面
    @Override
    public void onClick(View v) {
        //通过Intent跳转到查询单词界面
        Intent intent = new Intent(AddActivity.this, MainActivity.class);
        startActivity(intent);
    }
});
```

（13）打开主活动 MainActivity，在重写的 onCreate() 方法中，单击“翻译”按钮，实现查询词库中的单词，关键代码如下：

```
btnSearch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String key = etSearch.getText().toString();          //获取要查询的单词
        //查询单词
        Cursor cursor=dbOpenHelper.getReadableDatabase().query("dict",null
                ,"word = ?",new String[]{key},null,null,null);
        //创建ArrayList对象，用于保存查询结果
        ArrayList<Map<String, String>> resultList = new ArrayList<Map<String, String>>();
        while (cursor.moveToNext()) {                        //遍历Cursor结果集
            Map<String, String> map = new HashMap<>();//将结果集中的数据存入HashMap中
            //取出查询记录中第2列、第3列的值
            map.put("word", cursor.getString(1));
            map.put("interpret", cursor.getString(2));
            resultList.add(map);                        //将查询出的数据存入ArrayList中
        }
        if (resultList == null || resultList.size() == 0) { //如果数据库中没有数据
            //显示提示信息，没有相关记录
            Toast.makeText(MainActivity.this,
                    "很遗憾，没有相关记录！", Toast.LENGTH_LONG).show();
        } else {
            //否则将查询的结果显示到ListView列表中
            SimpleAdapter simpleAdapter = new SimpleAdapter(MainActivity.this, resultList,
                    R.layout.result_main,
                    new String[]{"word", "interpret"}, new int[]{
                    R.id.result_word, R.id.result_interpret});
            listView.setAdapter(simpleAdapter);
        }
    }
});
```

（14）重写 Activity 的 onDestroy() 方法，实现退出应用时，关闭数据库连接，关键代码如下：

```
@Override
protected void onDestroy() {            //实现退出应用时，关闭数据库连接
    super.onDestroy();
    if (dbOpenHelper != null) {         //如果数据库不为空时
        dbOpenHelper.close();           //关闭数据库连接
    }
}
```

（15）运行本实例，首先向数据库中插入数据如图 21.11 所示，然后从数据库中查询数据如图 21.12 所示。

图 21.11　添加生词并创建数据库

图 21.12　翻译数据库中单词

注意　首次运行本实例，数据库为空，进入添加生词界面，向数据库中插入数据方可查询。

21.4　使用 Content Provider 实现数据共享

Content Provider 主要用于在不同的应用程序之间实现数据共享。它提供了一套完整的机制，允许一个程序访问另一个程序中的数据，同时还能保证被访问数据的安全性。

在 Android 程序中，共享数据的实现需要继承自 ContentProvider 基类，该基类为其他应用程序使用和存储数据实现了一套标准方法，然而应用程序并不直接调用这些方法，而是使用一个 ContentResolver 对象去操作指定数据。

21.4.1　Content Provider 概述

Content Provider 内部如何保存数据由其设计者决定，但是所有的 Content Provider 都实现一组通用的方法，用来提供数据的增、删、改、查功能。

客户端通常不会直接使用这些方法，而是通过 ContentResolver 对象实现对 Content Provider 的操作。开发人员可以通过调用 Activity 或者其他应用程序组件的实现类中的 getContentResolver() 方法来获得 ContentResolver 对象，例如：

```
ContentResolver cr = getContentResolver();
```

使用 ContentResolver 提供的方法可以获得 Content Provider 中任何想要的数据。

当开始查询时，Android 系统确认查询的目标 Content Provider 并确保它正在运行。系统会初始化所有 ContentProvider 类的对象，开发人员不必完成此类操作，实际上，开发人员根本不会直接使用 ContentProvider 类的对象。通常，每个类型的 ContentProvider 仅有一个单独的实例。但是该实例能与位于不同应用程序和进程的多个 ContentResolver 类的对象通信。不同进程之间的通信由 ContentProvider 类和 ContentResolver 类处理。

使用 Content Provider 时，通常会用到以下两个概念。

1．数据模型

Content Provider 使用基于数据库模型的简单表格来提供其中的数据，这里每行代表一条记录，每列代表特定类型和含义的数据。例如，联系人的信息可能以表 21.2 所示的方式提供。

表 21.2　联系方式

_ID	NAME	NUMBER	EMAIL
001	张 ××	136*****	136**@163.com
002	王 ××	137*****	137**@google.com
003	李 ××	153*****	153**@qq.com
004	赵 ××	189*****	189**@126.com

每条记录包含一个数值型的 _ID 字段，用于在表格中唯一标识该记录。_ID 能用于匹配相关表格中的记录，例如，在一个表格中查询联系人的电话，在另一表格中查询其照片。

注意 _ID字段前还包含了一条下划线，在编写代码时不要忘记。

查询返回一个 Cursor 对象，它能遍历各行各列来读取各个字段的值。对于各个类型的数据，它都提供了专用的方法。因此，为了读取字段的数据，开发人员必须知道当前字段包含的数据类型。

2．URI 的用法

每个 Content Provider 提供公共的 URI（使用 Uri 类包装）来唯一标识其数据集。管理多个数据集（多个表格）的 Content Provider 为每个数据集提供了单独的 URI。所有为 Content Provider 提供的 URI 都以“content://”作为前缀，它表示数据由 Content Provider 来管理。

如果自定义 Content Provider，则需要为其 URI 也定义一个常量，来简化客户端代码并让日后更新更加简洁。Android 为当前平台提供的 Content Provider 定义了 CONTENT_URI 常量。例如，匹配电话号码到联系人表格的 URI 和匹配保存联系人照片表格的 URI 分别如下：

```
01  android.provider.Contacts.Phones.CONTENT_URI
02  android.provider.Contacts.Photos. CONTENT_URI
```

URI 常量用于所有与 Content Provider 的交互中。每个 ContentResolver 方法使用 URI 作为其第一个参数。它标识 ContentResolver 应该使用哪个 Content Provider 及其中的哪个表格。

Content URI 重要部分的总结如图 21.13 所示。

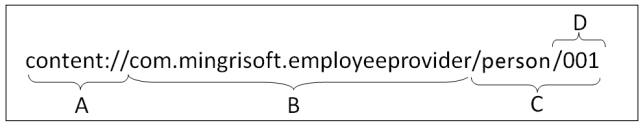


图 21.13　Content URI 重要部分的总结

◆ A：标准的前缀，用于标识该数据由 Content Provider 管理，不需修改。

◆ B：URI 的权限（authority）部分，用于对不同的应用程序做区分，一般会采用完整的类名（使用小写形式）来保证其唯一性。例如，一个包名为 com.mingrisoft 的应用，对应的权限就可以命名为 com.mingrisoft.provider。

◆ C：Content Provider 的路径（path）部分，用于指定要操作的数据，可以是数据表、文件、XML 等。例如，要访问数据表 person 中的所有记录，可以使用“/person”；而要访问 person 中的 ID 为 001 的记录的 name 字段，则需要使用“/person/001/name”。

◆ D：被请求的特定记录的 ID。这是被请求记录的 _ID 值。如果请求不仅限于单条记录，该部分及其前面的斜杠“/”应该删除。

21.4.2 创建 Content Provider

程序开发人员可以通过继承 ContentProvider 类，创建一个新的数据提供者。通常情况下，需要完成以下操作。

◆ 继承 ContentProvider 类来提供数据访问方式。

◆ 在应用程序的 AndroidManifest.xml 文件中声明 Content Provider。

下面分别进行介绍。

1. 继承 ContentProvider 类

开发人员定义 ContentProvider 类的子类以便使用 ContentResolver 和 Cursor 类来共享数据。原则上，这意味着需要实现 ContentProvider 类定义的以下 6 个抽象方法：

```
public boolean onCreate()
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder)
public Uri insert(Uri uri, ContentValues values)
public int update(Uri uri, ContentValues values,String selection, String[] selectionArgs)
public int delete(Uri uri, String selection, String[] selectionArgs)
public String getType(Uri uri)
```

各个方法的说明如表 21.3 所示。

表 21.3 Content Provider 中抽象方法说明

方　法	说　明
onCreate()	用于初始化 Content Provider
query()	返回数据给调用者
insert()	插入新数据到 Content Provider
update()	更新 Content Provider 中已经存在的数据
delete()	从 Content Provider 中删除数据
getType()	返回 Content Provider 数据的 MIME 类型

query() 方法必须返回 Cursor 对象，它用于遍历查询结果。Cursor 自身是一个接口，但是 Android 提供了一些该接口的实现类，例如，SQLiteCursor 能遍历存储在 SQLite 数据库中的数据。通过调用 SQLiteDatabase 类的 query() 方法可以获得 Cursor 对象。它们都位于 android.database 包中，其继承关系如图 21.14 所示。

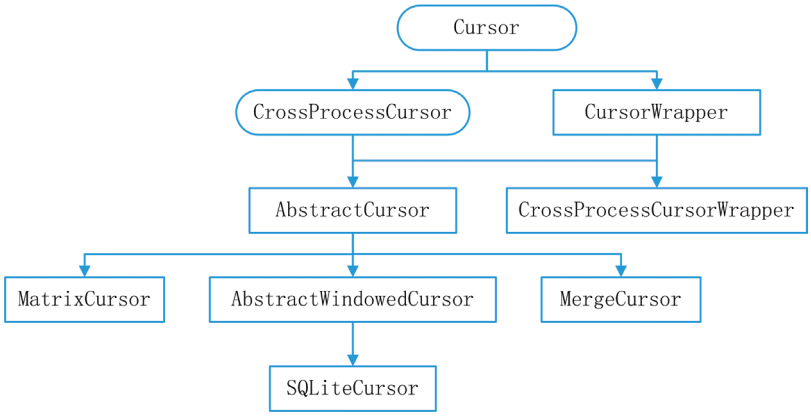


图 21.14　Cursor 接口继承关系

说明　圆角矩形表示接口，非圆角矩形表示类。

由于这些 ContentProvider 方法能被不同进程和线程的不同 ContentResolver 对象调用，所以它们必须以线程安全的方式实现。

此外，开发人员可能也想调用 ContentResolver.notifyChange() 方法以便在数据修改时通知监听器。

除了定义子类自身，还应采取下面的措施，以便简化客户端工作并让类更加易用。

（1）定义 public static final Uri CONTENT_URI 变量（CONTENT_URI 是变量名称）。该字符串表示自定义的Content Provider处理的完整“content:URI”。开发人员必须为该值定义唯一的字符串。最佳的解决方式是使用 Content Provider 的完整类名（小写）。例如，EmployeeProvider 的 URI 可能按以下格式定义：

```
public static final Uri CONTENT_URI = Uri.parse("content://com.mingrisoft.employeeprovider");
```

如果 Content Provider 包含子表，也应该为各个子表定义 URI。这些 URI 应该有相同的权限（因为它标志 Content Provider），然后使用路径进行区分，例如：

```
content://com.mingrisoft.employeeprovider/dba
content://com.mingrisoft.employeeprovider/programmer
content://com.mingrisoft.employeeprovider/ceo
```

（2）定义 Content Provider 将返回给客户端的列名。如果

开发人员使用底层数据库，这些列名通常与 SQL 数据库列名相同。同样定义 public static String 常量，客户端用它们来指定查询中的列和其他指令。确保包含名为“_ID”的整数列用来作为记录的 ID 值。无论记录中其他字段是否唯一，如 URL，开发人员都应该包含该字段。如果打算使用 SQLite 数据库，_ID 字段应该是如下类型：

```
INTEGER PRIMARY KEY AUTOINCREMENT
```

（3）仔细注释每列的数据类型，客户端需要使用这些信息来读取数据。

（4）如果开发人员正在处理新数据类型，则必须定义新的 MIME 类型以便在 ContentProvider.getType() 方法中实现返回。

（5）如果开发人员提供的 byte 数据太大而不能放到表格中，如 Bitmap 文件，则提供给客户端的字段应该包含 content:URI 字符串。

2. 声明 Content Provider

为了让 Android 系统知道开发人员编写的 Content Provider，需要在应用程序的 AndroidManifest.xml 文件中定义 <provider> 元素。没有在配置文件中声明的自定义 Content Provider 对于 Android 系统不可见。

name 属性的值是 ContentProvider 类的子类的完整名称。authorities 属性是 Content Provider 定义的 content:URI 中 authority 部分。ContentProvider 的子类是 EmployeeProvider，<provider> 元素应该如下：

```
<provider android:name="com.mingrisoft.EmployeeProvider"
        android:authorities="com.mingrisoft.employeeprovider"
    . . . />
</provider>
```

注意 authorities属性删除了content:URI中的路径部分。

其他 <provider> 属性能设置读写数据的权限，提供显示给用户的图标或文本，启用或禁用 Content Provider 等。如果数据不需要在多个运行中的 Content Provider 间同步，则设置 multiprocess 为 true。这允许在各个客户端进程创建一个 Content Provider 实例，从而避免执行 IPC（进程间通信）。

21.4.3 使用 Content Provider

Android 系统为常用数据类型提供了很多预定义的 Content Provider（声音、视频、图片、联系人等），它们大都位于 android.provider 包中。开发人员可以查询这些 provider 以获得其中包含的信息（有些需要适当的权限来读取数据）。Android 系统提供的常见 Content Provider，如表 21.4 所示。

表 21.4 常见 ContentProvider 说明

名　称	说　明
Browser	用于管理浏览器相关信息（例如，书签、浏览历史或网络搜索）
CallLog	用于管理通话历史信息
Contacts	用于管理联系人信息
LiveFolders	用于管理由 ContentProvider 提供内容的特定文件夹
MediaStore	用于管理多媒体信息（例如声音、视频和图片等）
Setting	用于管理系统设置和偏好设置（例如蓝牙设置、铃声和其他设备偏好）
SearchRecentSuggestions	用于为应用程序创建简单的查询建议
SyncStateContract	用于使用数据数组账号关联数据的 ContentProvider 约束
UserDictionary	用于在可预测文本输入时，提供用户定义的单词给输入法使用

1. 查询数据

要查询 Content Provider 中的数据，需要以下 3 个信息：

◆ 标识该 Content Provider 的 URI。

◆ 需要查询的数据字段名称。

◆ 字段中数据的类型。

为了查询 Content Provider 中的数据，开发人员需要使用 ContentResolver.query() 方法，该方法返回 Cursor 对象。query() 方法的语法格式如下：

```
public final Cursor query (Uri uri, String[] projection,
                           String selection, String[] selectionArgs, String sortOrder)
```

◆ uri：provider 的 URI，用于标识特定 ContentProvider 和数据集的 CONTENT_URI 常量。如果仅需要返回一条记录，可以在 URI 结尾增加该记录的 _ID 值，即将匹配 ID 值的字符串作为 URI 路径部分的结尾片段。

◆ projection：想要返回的数据列名称。null 值表示返回全部列；否则仅返回指定的列。全部预定义 Content Provider 为其列都定义了常量。例如，android.provider.Contacts.Phones 类定义了 _ID、NUMBER、NUMBER_KEY、NAME 等常量。

◆ selection：决定哪些行被返回的过滤器，格式类似 SQL 的 WHERE 语句（但是不包含 WHERE 自身）。null 值表示返回全部行（除非 URI 限制查询结果为单行记录）。

◆ selectionArgs：选择参数。

◆ sortOrder：返回记录的排序器，格式类似 SQL 的 ORDER BY 语句（但是不包含 ORDER BY 自身）。null 值表示以默认顺序返回记录，这可能是无序的。

获得数据使用 Cursor 对象处理，它能向前或向后遍历整个结果集。开发人员可以使用 Cursor 对象来读取数据，而增加、修改和删除数据则必须使用 ContentResolver 对象。

2．增加记录

为了向 Content Provider 中增加新数据，首先需要在 ContentValues 对象中建立键值对映射，这里每个键匹配 Content Provider 中列名，每个值是该列中希望增加的值。然后调用 ContentResolver. insert() 方法并传递给它 Content Provider 的 URI 参数和 ContentValues 映射。该方法返回新记录的完整 URI，即增加了新记录 ID 的 URI。开发人员可以使用该 URI 来查询并获取该记录的 Cursor，以便修改该记录。

insert() 方法的语法格式如下：

```
public abstract Uri insert (Uri uri, ContentValues values)
```

◆ uri：Content Provider 的 URI。

◆ values：要插入记录的列名和值组成的键 / 值对，不能为空。

3．增加新值

一旦记录存在，开发人员可以向其中增加新信息或者修改已经存在的信息。增加记录到 Contacts 数据库的最佳方式是增加保存新数据的表名到代表记录的 URI，然后使用组装好的 URI 来增加新数据。每个 Contacts 表格以 CONTENT_DIRECTORY 常量的方式提供名称。

开发人员可以调用使用 byte 数组作为参数的 ContentValues.put() 方法向表格中增加少量二进制数据，这适用于类似小图标的图片、短音频片段等。然而，如果需要增加大量二进制数据，如图片或者完整的歌曲等，则需要保存代表数据的 content:URI 到表格，然后使用文件 URI 调用 ContentResolver. openOutputStream() 方法。这样就可以将 Content Provider 的数据保存到文件并在记录的隐藏字段中保存文件路径。

4．批量更新记录

要批量更新数据（例如，将全部字段中“NY”替换成“New York”），可使用 ContentResolver.update() 方法并提供需要修改的列名和值。

update() 方法的语法格式如下：

```
public final int update (Uri uri, ContentValues values,String where, String[] selectionArgs)
```

- uri：Content Provider 的 URI。
- values：要修改记录的列名和值对，不能为空。
- where：决定哪些行被更新的过滤器，格式类似 SQL 的 WHERE 语句（但是不包含 WHERE 自身）。
- selectionArgs：选择参数。

5．删除记录

如果需要删除单条记录，可调用 ContentResolver.delete() 方法并提供特定行的 URI。如果需要删除多条记录，可调用 ContentResolver.delete() 方法并提供删除记录类型的 URI（如 android.provider.Contacts.People.CONTENT_URI）和一个 SQL WHERE 语句，它定义哪些行需要删除。

delete() 方法的语法格式如下：

```
public final int delete (Uri url, String where, String[] selectionArgs)
```

- uri：Content Provider 的 URI。
- where：SQL 条件语句，用于定义哪些行要删除。
- selectionArgs：选择参数。

注意 请确保提供了一个合适的WHERE语句，否则可能删除全部数据。

例 21.5 模拟微信电话本

在 Android Studio 中创建 Module，名称为“WeChat Phone Book”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后为默认添加的布局管理器设置背景图片和 TextView 属性。

（2）打开主活动 MainActivity，该类继承 Activity，在该类中定义所需的成员变量，关键代码如下：

```
private String columns = ContactsContract.Contacts.DISPLAY_NAME;//获取姓名
```

（3）创建 getQueryData() 方法，在该方法中实现获取通讯录信息，关键代码如下：

```
01  private CharSequence getQueryData() {
02      StringBuilder sb = new StringBuilder();                 //用于保存字符串
03      ContentResolver resolver = getContentResolver();        //获取ContentResolver对象
04      //查询记录
05      Cursor cursor = resolver.query(ContactsContract.Contacts.CONTENT_URI,
06              null, null, null, null);
07      int displayNameIndex = cursor.getColumnIndex(columns);//获取姓名记录的索引值
08      //迭代全部记录
09      for (cursor.moveToFirst(); !cursor.isAfterLast(); cursor.moveToNext()) {
10          String displayName = cursor.getString(displayNameIndex);
11          sb.append(displayName + "\n");
12      }
```

```
13      cursor.close();                                    //关闭Cursor
14      return sb.toString();                              //返回查询结果
15  }
```

（4）在 onCreate() 方法中获得布局文件中用于显示查询联系人姓名的 TextView 组件，然后将获取的通讯录信息显示在界面中，关键代码如下：

```
01  TextView tv = (TextView) findViewById(R.id.result);    //获取布局文件中的TextView组件
02  tv.setText(getQueryData());                            //为TextView设置数据
```

（5）在 AndroidManifest.xml 文件中增加读取联系人记录的权限，代码如下：

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

（6）运行本实例，将显示如图 21.15 所示。

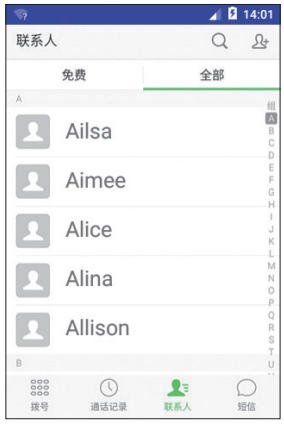
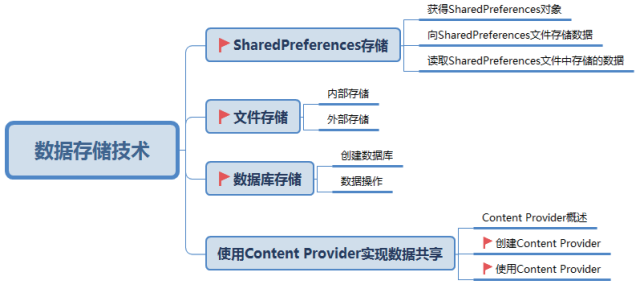


图 21.15　显示联系人

说明　读者需要在模拟器中开启访问通讯录的权限。另外，还需要在通讯录中创建一些联系人信息。

21.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 22 章

默默付出的后台工作者

在进行 Android 程序开发时，有三种默默付出的后台工作者，分别是多线程、消息处理传递机制的 Handler、以及后台服务 Service。在 Android 中，对于一些比较耗时的操作，通常会为其开辟一个单独的线程来执行，以尽可能减少用户的等待时间。默认情况下，所有的操作都在主线程中进行，主线程负责管理与 UI 相关的事件，而在用户自己创建的子线程中，不能对 UI 组件进行操作。因此，Android 提供了消息处理传递机制 Handler 来解决这一问题。而 Service 是用于在后台完成用户指定的操作，它可以用于播放音乐、文件下载和检查新消息推送等。用户可以使用其他组件来与 Service 进行通信。本章将对这三种后台工作者进行详细介绍。

22.1 多 线 程

在现实生活中，很多事情都是同时进行的，例如，我们可以一边看书，一边喝咖啡；而计算机则可以一边播放音乐，一边打印文档。对于这种可以同时进行的任务，可以用线程来表示，每个线程完成一个任务，并与其他线程同时执行，这种机制被称为多线程。下面就来介绍如何创建线程、开启线程、让线程休眠和中断线程。

22.1.1 创建线程

在 Android 中，提供了两种创建线程的方法：一种是通过 Thread 类的构造方法创建线程对象，并重写 run() 方法实现；另一种是通过实现 Runnable 接口实现，下面分别进行介绍。

1. 通过 Thread 类的构造方法创建线程

在 Android 中，可以使用 Thread 类提供的以下构造方法来创建线程。

```
Thread(Runnable runnable)
```

该构造方法的参数 runnable 可以通过创建一个 Runnable 类的对象并重写其 run() 方法来实现，例如，要创建一个名称为 thread 的线程，可以使用下面的代码：

```
01  Thread thread=new Thread(new Runnable(){
02      //重写run()方法
```

```
03      @Override
04      public void run() {
05          //要执行的操作
06      }
07  });
```

说明 在run()方法中，可以编写要执行的操作代码，当线程被开启时，run()方法将被执行。

2. 通过实现 Runnable 接口创建线程

在 Android 中，还可以通过实现 Runnable 接口来创建线程。实现 Runnable 接口的语法格式如下：

```
public class ClassName extends Object implements Runnable
```

当一个类实现 Runnable 接口后，还需要实现其 run() 方法，在 run() 方法中，可以编写要执行的操作的代码。

例如，要创建一个实现了 Runnable 接口的 Activity，可以使用下面的代码：

```
01  public class MainActivity extends Activity implements Runnable {
02      @Override
03      public void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.main);
06      }
07      @Override
08      public void run() {
09          //要执行的操作
10      }
11  }
```

22.1.2 开启线程

创建线程对象后，还需要开启线程，线程才能执行。Thread 类提供了 start() 方法用于开启线程，其语法格式如下：

```
start()
```

例如，存在一个名称为 thread 的线程，如果想开启该线程，可以使用下面的代码：

```
thread.start();                    //开启线程
```

22.1.3 线程的休眠

线程的休眠就是让线程暂停一段时间后再次执行。同 Java 一样，在 Android 中，也可以使用 Thread 类的 sleep() 方法让线程休眠指定的时间。sleep() 方法的语法格式如下：

```
sleep(long time)
```

其中参数 time 用于指定休眠的时间，单位为毫秒。

例如，想要线程休眠 1 秒钟，可以使用下面的代码：

```
Thread.sleep(1000);
```

22.1.4 中断线程

当需要中断指定的线程时，可以使用 Thread 类提供的 interrupt() 方法来实现。使用 interrupt() 方法可以向指定的线程发送一个中断请求，并将该线程标记为中断状态。interrupt() 方法的语法格式如下：

```
interrupt()
```

例如，存在一个名称为 thread 的线程，如果想中断该线程，可以使用下面的代码：

```
01  …                          //省略部分代码
02          thread.interrupt();
03          …                  //省略部分代码
04  public void run() {
05          while(!Thread.currentThread().isInterrupted()){
06          …                  //省略部分代码
07          }
08    }
```

另外，由于当线程执行 wait()、join() 或 sleep() 方法时，线程的中断状态将被清除并抛出 InterruptedException，所以，如果想在线程中执行了 wait()、join() 或 sleep() 方法时中断线程，就需要使用一个 boolean 型的标记变量来记录线程的中断状态，并通过该标记变量来控制循环的执行与停止。例如，通过名称为 isInterrupt 的 boolean 型变量来标记线程的中断，关键代码如下：

```
01  private boolean isInterrupt=false; //定义标记变量
02      …                      //省略部分代码
03      …                      //在需要中断线程时，将isInterrupt的值设置为true
04  public void run() {
05       while(!isInterrupt){
06       …                     //省略部分代码
07          }
08     }
```

下面通过一个具体的实例演示线程的开启与中断。

例 22.1 线程的开启与中断

在 Android Studio 中创建一个 Module，名称为“Thread”。实现本实例的具体步骤如下：

（1）修改新建项目的 res/layout 目录下的布局文件 main.xml，将默认添加的相对布局管理器和 TextView 组件删除，然后添加一个线性布局管理器，在其中添加两个按钮，一个用于开启线程，另一个用于中断线程，具体代码请参见云盘。

（2）打开默认添加的 MainActivity，让该类实现 Runnable 接口，修改后的创建类的代码如下：

```
public class MainActivity extends Activity implements Runnable {}
```

（3）实现 Runnable 接口中的 run() 方法，在该方法中，判断当前线程是否被中断，如果没有被中断，则将循环变量值加 1，并在日志中输出循环变量的值，具体代码如下：

```
@Override
public void run() {
    while (!Thread.currentThread().isInterrupted() && thread!=null) {
        try {
            Thread.sleep(1000);                          //休眠1秒
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        i++;
        Log.i("循环变量：", String.valueOf(i));
    }
}
```

（4）在该 MainActivity 中，创建两个成员变量，具体代码如下：

```
private Thread thread; //声明线程对象
int i;                 //循环变量
```

（5）在 onCreate() 方法中，首先获取布局管理器中添加的“开始”按钮，然后为该按钮添加单击事件监听器，在重写的 onCreate() 方法中，根据当前 Activity 创建一个线程，并开启该线程，具体代码如下：

```
Button startButton = (Button) findViewById(R.id.button1);  //获取"开始"按钮
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        i = 0;
        thread = new Thread(MainActivity.this);          //创建一个线程
        thread.start();                                  //开启线程
    }
});
```

（6）获取布局管理器中添加的“停止”按钮，并为其添加单击事件监听器，在重写的 onCreate() 方法中，如果 thread 对象不为空，则中断线程，并向日志中输出提示信息，具体代码如下：

```
Button stopButton = (Button) findViewById(R.id.button2);  //获取"停止"按钮
stopButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (thread != null) {
            thread.interrupt();                          //中断线程
            thread = null;
        }
        Log.i("提示：", "中断线程");
    }
});
```

（7）重写 MainActivity 的 onDestroy() 方法，在该方法中中断线程，具体代码如下：

```
@Override
protected void onDestroy() {
    if (thread != null) {
        thread.interrupt();                    //中断线程
        thread = null;
    }
    super.onDestroy();
}
```

（8）运行本实例，将显示如图 22.1 所示的界面效果。

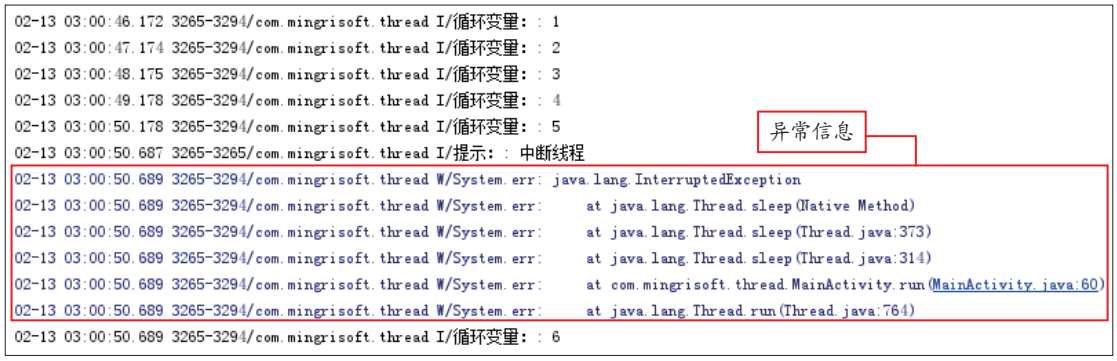


图 22.1　在日志面板中输出的内容

说明 由于本实例调用了interrupt()方法，所以抛出了InterruptedException异常信息。

22.2 Handler 消息传递机制

在 Java 中，对于一些周期性的或者是耗时的操作通常由多线程来实现，而在 Android 中，也可以使用 Java 中的多线程技术。例如，在手机淘宝主界面的上方，对广告进行轮播显示（如图 22.2 所示），以及某些游戏中的计时进度条（如图 22.3 所示），都需要应用到多线程。

图 22.2　手机淘宝广告轮换

图 22.3　找茬时间进度条

在 Android 中使用多线程，有一点需要注意：不能在子线程中动态改变主线程中的 UI 组件的属性。

说明 当一个程序第一次启动时，Android会启动一条主线程，用于负责接收用户的输入，将运行的结果反馈给用户，也称为UI线程；而子线程是指为了执行一些可能产生阻塞操作而新启动的线程，也称为Worker线程。

例如，实现单击按钮时创建新线程，用于改变文本框的显示文本，代码如下：

```
01  public class MainActivity extends AppCompatActivity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          final TextView textView= (TextView) findViewById(R.id.tv);  //获取文本框组件
07          Button button= (Button) findViewById(R.id.button);          //获取按钮组件
08          button.setOnClickListener(new View.OnClickListener() {
09              @Override
10              public void onClick(View v) {
11                  //创建新线程
12                  Thread thread = new Thread(new Runnable() {
13                      @Override
14                      public void run() {
15                   //要执行的操作
16                   textView.setText("你今天的努力，是幸运的伏笔；当下的付出，是明日的花开");
17                      }
18                  });
19                  thread.start();                                     //开启线程
20              }
21          });
22      }
23  }
```

运行时，将产生“抱歉，XXX 已停止运行”的对话框，并且单击状态栏中的“Android Monitor”选项，在 LogCat 面板中，输出如图 22.4 所示的异常信息。

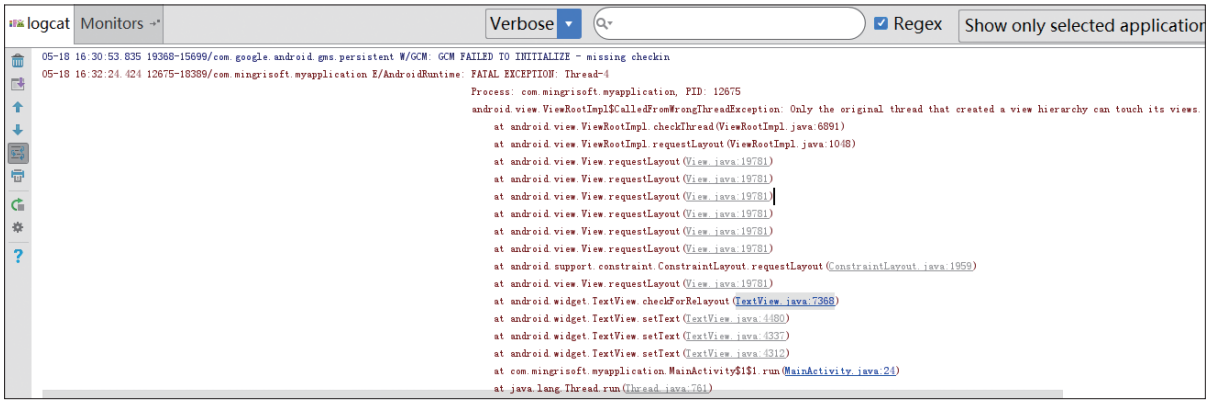


图 22.4　在子线程中更新 UI 组件产生的异常

为此，Android 中引入了 Handler 消息传递机制，来实现在新创建的线程中操作 UI 界面。下面将对 Handler 消息传递机制进行介绍。

22.2.1 Handler 类简介

Handler 是 Android 提供的一个用来更新 UI 的机制，也是一个消息处理的机制。通过 Handler 类（消息处理类）可以发送和处理 Message 对象到其所在线程的 MessageQueue 中。Handler 类主要有以下两个作用。

（1）在任意线程中发送消息。

将 Message 应用 sendMessage() 方法发送到 MessageQueue 中，在发送时可以指定延迟时间、发送时间以及要携带的 Bundle 数据。当 Looper 循环到该 Message 时，调用相应的 Handler 对象的 handlerMessage() 方法对其进行处理。

（2）在主线程中获取并处理消息。

为了让主线程能在适当的时候处理 Handler 所发送的消息，必须通过回调方法来实现。开发者只需要重写 Handler 类中处理消息的方法。这样当新启动的线程发送消息时，Handler 类中处理消息的方法就会被自动回调。

22.2.2 Handler 类中的常用方法

在 Handler 类中包含了一些用于发送和处理消息的常用方法，这些方法如表 22.1 所示。

表 22.1 Handler 类提供的常用方法

方　法	描　述
handleMessage(Message msg)	处理消息的方法。通常重写该方法来处理消息，在发送消息时，该方法会自动回调
hasMessages(int what)	检查消息队列中是否包含 what 属性为指定值的消息
hasMessages(int what, Object object)	检查消息队列中是否包含 what 属性为指定值且 object 属性为指定对象的消息
post(Runnable r)	立即发送 Runnable 对象，该 Runnable 对象最后将被封装成 Message 对象
postAtTime(Runnable r, long uptimeMillis)	定时发送 Runnable 对象，该 Runnable 对象最后将被封装成 Message 对象
postDelayed(Runnable r, long delayMillis)	延迟发送 Runnable 对象，该 Runnable 对象最后将被封装成 Message 对象
sendEmptyMessage(int what)	发送空消息
sendEmptyMessageDelayed(int what, long delayMillis)	指定多少毫秒之后发送空消息
sendMessage(Message msg)	立即发送消息
sendMessageAtTime(Message msg, long uptimeMillis)	定时发送消息

续表

方　法	描　述
sendMessageDelayed(Message msg, long delayMillis)	延迟发送消息
obtainMessage()	获取消息

通过这些方法，应用程序就可以方便地使用 Handler 来进行消息传递。

例 22.2 模拟找茬游戏的时间进度条

在 Android Studio 中创建一个 Module，名称为“Time Progress Bar”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后删除 TextView 组件，再为布局管理器添加背景图片，最后在该布局管理器中添加一个 ProgressBar 组件并设置进度条样式。

（2）打开默认创建的主活动 MainActivity，让其继承自 Activity，在该类中定义所需的成员变量，具体代码如下：

```
final int TIME = 60;                          //定义时间长度
final int TIMER_MSG = 0x001;                  //定义消息代码
private ProgressBar timer;                    //声明水平进度条
private int mProgressStatus = 0;              //定义完成进度
```

（3）在 MainActivity 中创建 android.os.Handler 对象，并重写 handleMessage() 方法，在该方法中判断当前时间进度大于 0 时，更新进度条，然后每隔 1 秒更新一次进度条，关键代码如下：

```
Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        //当前进度大于0
        if (TIME - mProgressStatus > 0) {
            mProgressStatus++;                                  //进度+1
            timer.setProgress(TIME - mProgressStatus);          //更新进度条的显示进度
            handler.sendEmptyMessageDelayed(TIMER_MSG, 1000); //延迟一秒发送消息
        } else {
            //提示时间已到
            Toast.makeText(MainActivity.this,
                    "时间到！游戏结束！", Toast.LENGTH_SHORT).show();
        }
    }
};
```

（4）在 onCreate() 方法中，获取进度条组件，并启动进度条，关键代码如下：

```
timer = (ProgressBar) findViewById(R.id.timer);       //获取进度条组件
handler.sendEmptyMessage(TIMER_MSG);                  //发送消息，启动进度条
```

（5）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（6）运行本实例，将显示如图 22.5 所示。

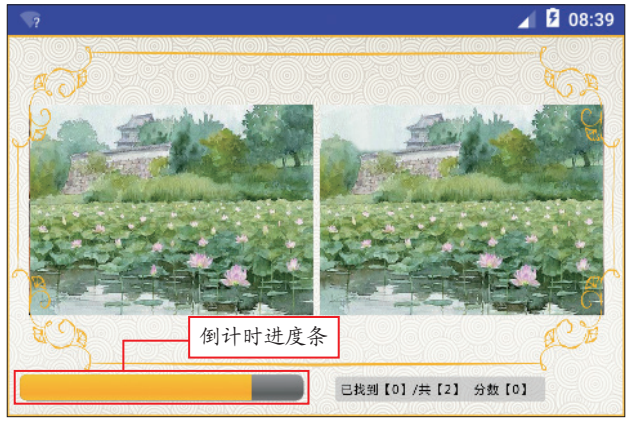


图 22.5　找茬游戏的倒计时进度条

说明　单击模拟器右侧菜单栏中的旋转按钮，将模拟器屏幕切换为横屏状态。

22.2.3 Handler 与 Looper、MessageQueue 的关系

Handler 并不是单独工作，与 Handler 共同工作的有几个重要的组件，主要有 Message、Looper 和 MessageQueue。

◆ Message：通过 Handler 发送、接收和处理的消息对象。

◆ Looper：负责管理 MessageQueue。每个线程只能有一个 Looper，它的 loop() 方法负责读取 MessageQueue 中的消息，读取到消息之后就把消息回传给 Handler 进行处理。

◆ MessageQueue：消息队列，可以看作是一个存储消息的容器。它采用 FIFO（先进先出）的原则来管理消息。在创建 Looper 对象时，会在它的构造器中创建 MessageQueue 对象。

在 Android 中，一个线程对应一个 Looper 对象，而一个 Looper 对象又对应一个 MessageQueue，MessageQueue 用于存放 Message。Handler 发送 Message 给 Looper 管理的 MessageQueue，然后 Looper 又从 MessageQueue 中取出消息，并分配给 Handler 进行处理，如图 22.6 所示。

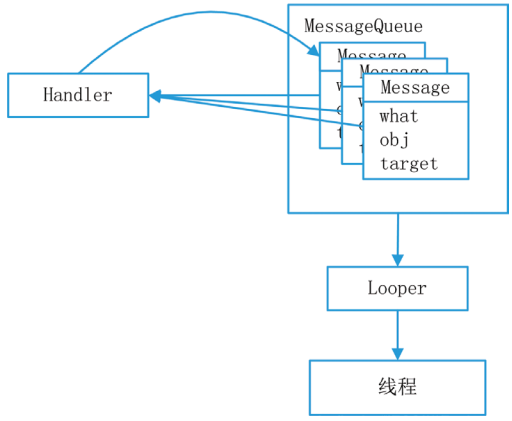


图 22.6　Handler 与 Looper、MessageQueue、Message 的关系图

因此，要在程序中使用 Handler，必须在当前线程中有一个 Looper 对象。线程中的 Looper 对象有以下两种创建方式：

◆ 在主 UI 线程中，系统已经初始化了一个 Looper 对象，因此在程序中可以直接创建 Handler，然后就可以通过 Handler 进行发送消息和处理消息。

◆ 在子线程中，必须手动创建一个 Looper 对象，并通过 loop() 方法启动 Looper。

在子线程中使用 Handler 的步骤如下：

（1）调用 Looper 的 prepare() 方法为当前的线程创建 Looper 对象，在创建 Looper 对象的构造器中会创建与之配套的 MessageQueue。

（2）创建 Handler 子类的实例，重写 handlerMessage() 方法用来处理来自于其他线程的消息。

（3）调用 Looper 的 loop() 方法启动 Looper。

说明 在一个线程中，只能有一个Looper和MessageQueue，但是可以有多个Handler，而且这些Handler可以共享同一个Looper和MessageQueue。

22.2.4 消息类（Message）

消息类（Message）被存放在 MessageQueue 中，一个 MessageQueue 中可以包含多个 Message 对象。每个 Message 对象可以通过 Message.obtain() 或 Handler.obtainMessage() 方法获得。一个 Message 对象具有如表 22.2 所示的 5 个属性。

表 22.2 Message 对象的属性

属　性	类　型	描　述
arg1	int	用来存放整型数据
arg2	int	用来存放整型数据
obj	Object	用来存放发送给接收器的 Object 类型的任意对象
replyTo	Messenger	用来指定此 Message 发送到何处的可选 Messenger 对象
what	int	用于指定用户自定义的消息代码，这样接收者可以了解这个消息的信息

说明 使用Message类的属性可以携带int型数据，如果要携带其他类型的数据，可以先将要携带的数据保存到Bundle对象中，然后通过Message类的setDate()方法将其添加到Message中。

总之，Message 类的使用方法比较简单，在使用时，需注意以下 3 点：

◆ 尽管 Message 有 public 的默认构造方法，但是通常情况下，需要使用 Message.obtain() 或 Handler.obtainMessage() 方法来从消息池中获得空消息对象，以节省资源。

◆ 如果一个 Message 只需要携带简单的 int 型信息，应优先使用 Message.arg1 和 Message.arg2 属性来传递信息，这比用 Bundle 更节省内存。

◆ 尽可能使用 Message.what 来标识信息，以便用不同方式处理 Message。

例 22.3 模拟手机淘宝的轮播广告

在 Android Studio 中创建 Module，名称为“Carousel Advertising”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器，然后删除 TextView 组件，再为布局管理器添加背景图片，最后在该布局管理器中添加一个 ViewFlipper 组件用于切换图片。

（2）在新建 Module 的 res 目录下创建 anim 文件夹，在该文件夹中创建实现平移从右进入与

从左退出的动画资源文件。

（3）打开默认创建的主活动 MainActivity，让其继承自 Activity，在该类中定义所需的成员变量，关键代码如下：

```
final int FLAG_MSG = 0x001;     //定义要发送的消息代码
private ViewFlipper flipper;    //定义ViewFlipper
private Message message;        //声明消息对象
//定义图片数组
private int[] images = new int[]{R.drawable.img1, R.drawable.img2, R.drawable.img3,
        R.drawable.img4, R.drawable.img5, R.drawable.img6,
        R.drawable.img7, R.drawable.img8};
//定义动画数组，为ViewFlipper指定切换动画
private Animation[] animation = new Animation[2];
```

（4）在 onCreate() 方法中，首先获取用于切换图像的 ViewFlipper 组件，然后获取数组中的图片并加载，再初始化动画数组，并设置采用动画效果，关键代码如下：

```
flipper = (ViewFlipper) findViewById(R.id.viewFlipper); //获取ViewFlipper
for (int i = 0; i < images.length; i++) {               //遍历图片数组中的图片
    ImageView imageView = new ImageView(this);          //创建ImageView对象
    imageView.setImageResource(images[i]);              //将遍历的图片保存在ImageView中
    flipper.addView(imageView);                         //加载图片
}
//右侧平移进入动画
animation[0] = AnimationUtils.loadAnimation(this, R.anim.slide_in_right);
//左侧平移退出动画
animation[1] = AnimationUtils.loadAnimation(this, R.anim.slide_out_left);
flipper.setInAnimation(animation[0]);                   //为flipper设置图片进入动画效果
flipper.setOutAnimation(animation[1]);                  //为flipper设置图片退出动画效果
```

（5）在 MainActivity 类中，创建 android.os.Handler 对象，并重写 handleMessage() 方法，在重写的 handleMessage() 方法中，首先判断是否为发送的标记，如果是，则显示下一个动画和图片，然后再延迟 3 秒发送消息，关键代码如下：

```
Handler handler = new Handler() {                       //创建android.os.Handler对象
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == FLAG_MSG) {                     //如果接收到的是发送的消息标记
            flipper.showPrevious();                     //显示下一张图片
        }
        message=handler.obtainMessage(FLAG_MSG);        //获取要发送的消息
        handler.sendMessageDelayed(message, 3000);      //延迟3秒发送消息
    }
};
```

（6）在 onCreate() 方法中，设置发送 handler 消息，用于启动 Handler 对象中的延迟消息，关键代码如下：

```
message=Message.obtain();                               //获得消息对象
message.what=FLAG_MSG;                                  //设置消息代码
handler.sendMessage(message);                           //发送消息
```

（7）运行本实例，将显示如图 22.7 所示的效果。

图 22.7　手机淘宝轮播广告

22.3 Service 的应用

Service（服务）是能够在后台长时间运行，并且不提供用户界面的应用程序组件。其他应用程序组件能启动 Service，并且即便用户切换到另一个应用程序，Service 还可以在后台运行。此外，组件能够绑定到 Service 并与之交互，甚至执行进程间通信（IPC）。例如，Service 能在后台处理网络事务、播放音乐、执行文件操作或者与 Content Provider 通信。

例如，通过 Service 可以实现在手机后台播放音乐，手机锁屏后的播放音乐界面如图 22.8 所示；通过 Service 还可以实现在手机后台监控地理位置的改变，手机地图中记录地理位置的界面如图 22.9 所示。

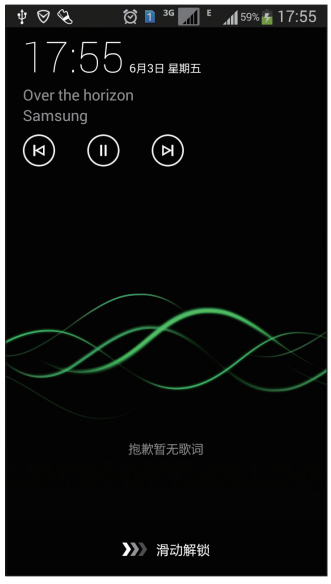


图 22.8　后台播放音乐

图 22.9　记录地理位置

22.3.1 Service 的分类

Service 按照启动方式可以分为以下两种类型。

◆ Started Service：当应用程序组件（如 Activity）通过调用 startService() 方法启动 Service 时，Service 处于启动状态。一旦启动，Service 能在后台无限期运行。

◆ Bound Service：当应用程序组件通过调用 bindService() 方法绑定到 Service 时，Service 处于绑定状态。多个组件可以同时绑定到一个 Service 上，当它们都解除绑定时，Service 被销毁。

Started Service 与 Bound Service 的区别如表 22.3 所示。

表 22.3 Started Service 与 Bound Service 的区别

Started Service	Bound Service
使用 startService() 方法启动	调用 bindService() 方法绑定
通常只启动，不返回值	发送请求，得到返回值
启动 Service 的组件与 Service 之间没有关联，即使关闭该组件，Service 也会一直运行	启动 Service 的组件与 Service 绑定在一起，如果关闭该组件，Service 就会停止
回调 onStartCommand() 方法，允许组件启动 Service	回调 onBind() 方法，允许组件绑定 Service

尽管本章将两种类型的 Service 分开讨论，不过 Service 也可以同时属于这两种类型，既可以启动（无限期运行）也能绑定。不管应用程序是否为启动状态、绑定状态或者两者兼有，都能通过 Intent 使用 Service，就像使用 Activity 那样。然而，开发人员可以在配置文件中将 Service 声明为私有的，从而阻止其他应用程序访问。

22.3.2 Service 的生命周期

Service 的生命周期比 Activity 简单很多，但是却需要开发人员更加关注 Service 如何创建和销毁，因为 Service 可能在用户不知情的情况下在后台运行。图 22.10 演示了 Service 的生命周期。

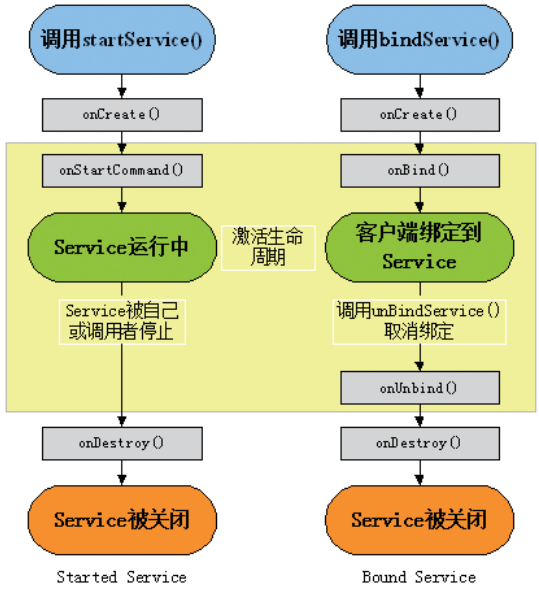


图 22.10 Service 的生命周期

由图 22.10 可以看出，Service 的生命周期可以分成两个不同的路径：

◆ 通过 startService() 方法启动 Service

当其他组件调用 startService() 方法时，Service 被创建，并且无限期运行，其自身必须调用 stopSelf() 方法或者其他组件调用 stopService() 方法来停止 Service。当 Service 停止时，系统将其销毁。

◆ 通过 bindService() 方法启动 Service

当其他组件调用 bindService() 方法时，Service 被创建。接着客户端通过 IBinder 接口与 Service 通信。客户端通过 unbindService() 方法关闭连接。多个客户端能绑定到同一个 Service 并且当它们都解绑定时，系统销毁 Service（Service 不需要被停止）。

这两条路径并非完全独立，即开发人员可以绑定已经使用 startService() 方法启动的 Service。例如，后台音乐 Service 能使用包含音乐信息的 Intent 通过调用 startService() 方法启动。当用户需要控制播放器或者获得当前音乐信息时，可以调用 bindService() 方法绑定 Activity 到 Service。此时，只有 stopService() 和 stopSelf() 方法全部被客户端解绑定时才能停止 Service。

为了创建 Service，开发人员需要创建 Service 类或其子类的子类。在实现类中，需要重写一些处理 Service 生命周期重要方面的回调方法，并根据需要提供组件绑定到 Service 的机制。需要重写的重要回调方法如表 22.4 所示。

表 22.4　Service 生命周期中的回调方法

方　法　名	描　　述
void onCreate()	当 Service 第一次创建时，系统调用该方法执行一次性建立过程（在系统调用 onStartCommand() 或 onBind() 方法前）。如果 Service 已经运行，该方法不被调用
void onStartCommand(Intent intent, int flags, int startId)	当其他组件（如 Activity）调用 startService() 方法请求 Service 启动时，系统调用该方法。一旦该方法执行，Service 就启动并在后台无限期运行
IBinder onBind(Intent intent)	该方法是 Service 子类必须实现的方法，该方法返回一个 IBinder 对象，应用程序可以通过该对象与 Service 组件进行通信
void onDestroy()	当 Service 不再使用并即将销毁时，系统调用该方法

22.3.3 创建与配置 Service

应用程序组件（如 Activity）可以通过调用 startService() 方法和传递 Intent 对象来启动 Service。在 Intent 对象中指定了 Service 并且包含 Service 需要使用的全部数据。Service 使用 onStartCommand() 方法接收 Intent。

Android 提供了两个类供开发人员继承用于创建和启动 Service。

◆ Service：这是所有 Service 的基类。当继承该类时，创建新线程来执行 Service 的全部工作是非常重要的。因为 Service 默认使用应用程序主线程，这可能降低应用程序 Activity 的运行性能。

◆ IntentService：这是 Service 类的子类，它每次使用一个 Worker 线程来处理全部启动请求。在不必同时处理多个请求时，这是最佳选择。开发人员仅需要实现 onHandleIntent() 方法，该方法接收每次启动请求的 Intent 以便完成后台任务。

使用 Android Studio 可以很方便地创建并配置 Service，方法步骤如下：

（1）在 Module 的包名（如 com.mingrisoft）节点上单击鼠标右键，然后依次选择 New → Service → Service 菜单项，如图 22.11 所示。

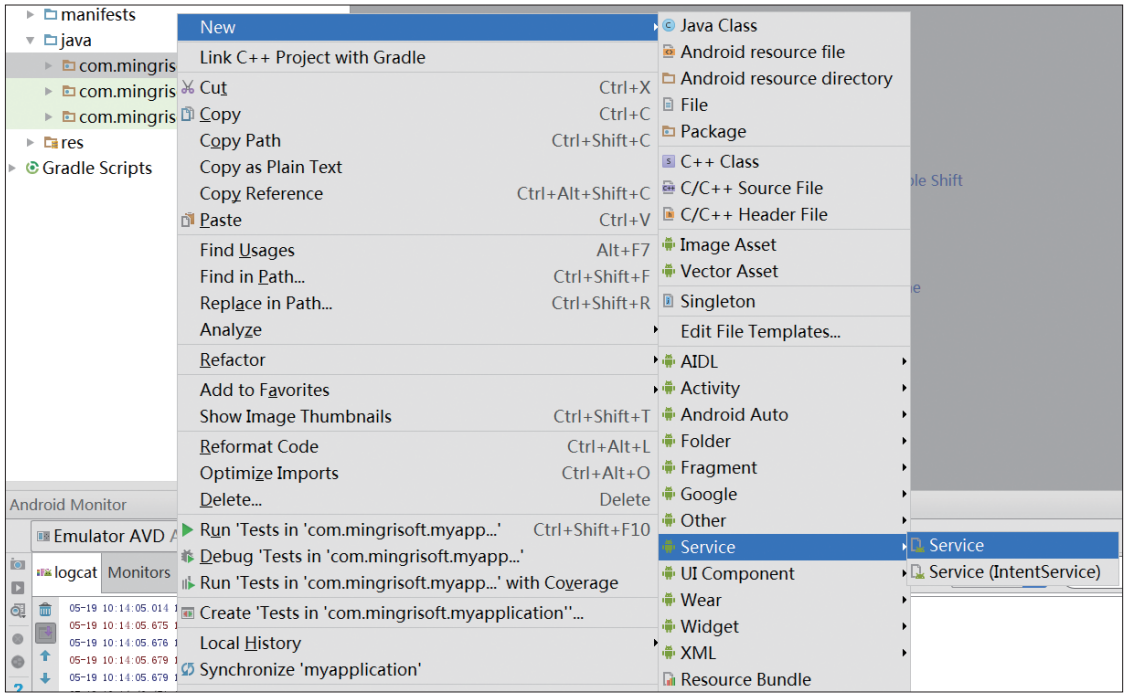


图 22.11　选择“Service”选项

说明　在图22.11中，如果选择New → Service → Service（IntentService）菜单项，可以创建继承自IntentService的Service。

（2）在弹出的对话框中的 Class Name 文本框中输入 Service 的名称（如 MyService），如图 22.12 所示。

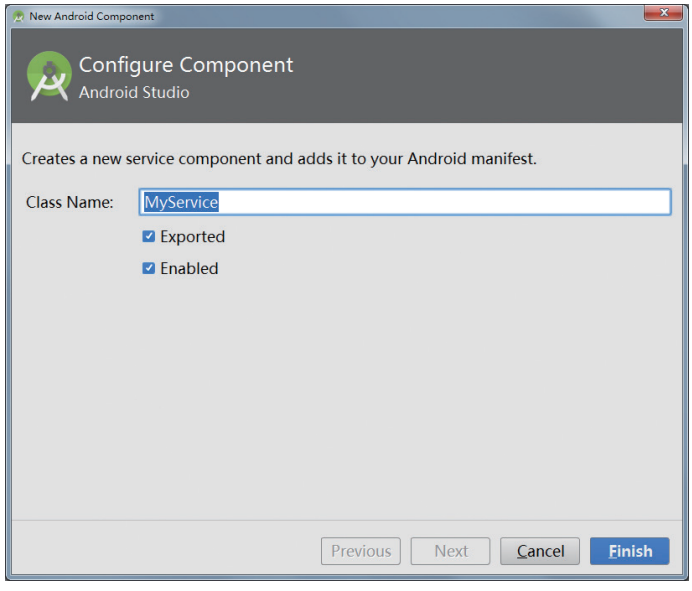


图 22.12　修改创建的 Service 名称

（3）单击 Finish 按钮即可创建一个 Service，然后就可以在类中重写需要的回调方法。通常情况下，经常会重写以下 3 个方法。

◆ onCreate()：在 Service 创建时调用；
◆ onStartCommand()：在每次启动 Service 时调用；
◆ onDestroy()：在 Service 销毁时调用。

例如，在刚刚创建的 MyService 中重写了 onCreate()、onStartCommand()、onDestroy() 这 3 个方法，实现开启新线程模拟一段耗时操作，同时监控 Service 的状态。具体代码如下：

```
01  public class MyService extends Service {
02      public MyService() {
03      }
04      @Override
05      public IBinder onBind(Intent intent) {
06          //TODO: Return the communication channel to the service.
07          throw new UnsupportedOperationException("Not yet implemented");
08      }
09      @Override
10      public void onDestroy() {
11          Log.i("Service: ", "Service已停止");
12          super.onDestroy();
13      }
14      @Override
15      public void onCreate() {
16          Log.i("Service: ", "Service已创建");
17          super.onCreate();
18      }
19      @Override
20      public int onStartCommand(Intent intent, int flags, int startId) {
21          new Thread(new Runnable() {
22              @Override
23              public void run() {
24                  Log.i("Service: ", "Service已开启");
25                  //模拟一段耗时任务
26                  long endTime = System.currentTimeMillis() + 5 * 1000;
27                  while (System.currentTimeMillis() < endTime) {
28                      synchronized (this) {
29                          try {
30                              wait(endTime - System.currentTimeMillis());
31                          } catch (Exception e) {
32                              e.printStackTrace();
33                          }
34                      }
35                  }
36                  stopSelf();                    //停止Service
37              }
38          }).start();
39          return super.onStartCommand(intent, flags, startId);
40      }
41  }
```

在创建 Service 之后，系统会自动在 AndroidManifest.xml 文件中配置 Service，配置 Service 使用 <service…/> 标记，如图 22.13 所示。

```
<service
    android:name=".MyService"
    android:enabled="true"
    android:exported="true"></service>
```

图 22.13　自动配置 Service

其中，enabled、exported 两个属性的说明如下：

◆ android:enabled

用于指定 Service 能否被实例化，true 表示能，false 表示不能，默认值是 true。<application> 标记也有自己的 enabled 属性，适用于应用中所有的组件。当 Service 被启用时，只有 <application> 和 <service> 标记的 enabled 属性同时设置为 true（两者的默认值都是 true）时，才能让 Service 可用，并且能被实例化。任何一个是 false，Service 都将被禁用。

◆ android:exported

用于指定其他应用程序组件能否调用 Service 或者与其交互，true 表示能，false 表示不能。当该值是 false 时，只有同一个应用程序的组件或者具有相同用户 ID 的应用程序能启动或者绑定到 Service。

android:exported 属性的默认值依赖于 Service 是否包含 Intent 过滤器。若没有过滤器，说明 Service 仅能通过精确类名调用，这意味着 Service 仅用于应用程序内部（因为其他程序可能不知道类名）。此时，默认值是 false；若存在至少一个过滤器，暗示 Service 可以用于外部，因此默认值是 true。

22.3.4　启动和停止 Service

1．启动 Service

开发人员可以通过 Activity 或者其他应用程序组件将 Intent 对象（指定要启动的 Service）传递到 startService() 方法中来启动 Service。Android 系统调用 Service 的 onStartCommand() 方法并将 Intent 传递给它。

例如，Activity 能使用显式 Intent 和 startService() 方法启动 22.3.3 小节创建的 Service（MyService），其代码如下：

```
01  Intent intent =new Intent(this,MyService.class);
02  startService(intent);
```

启动 MyService 后，在 LogCat 中会输出如图 22.14 所示的日志信息。

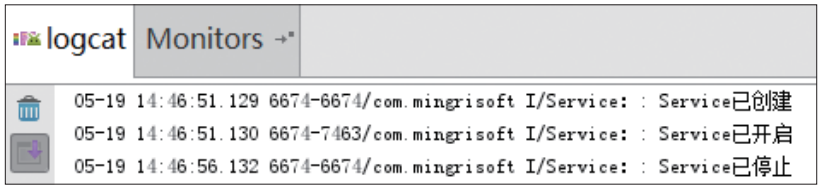


图 22.14　输出的 Service 启动状态的日志信息

在执行 startService() 方法后，Android 系统调用 Service 的 onStartCommand() 方法。如果 Service 还没有运行，系统首先调用 onCreate() 方法，接着调用 onStartCommand() 方法。

如果 Service 没有提供绑定，startService() 方法发送的 Intent 是应用程序组件和 Service 之间唯一的通信模式。然而，如果开发人员需要 Service 返回结果，则启动该 Service 的客户端能为广播创建 PendingIntent（使用 getBroadcast() 方法），并通过启动 Service 的 Intent 进行发送。Service 接下来便能使用广播来发送结果。

多次启动 Service 的请求会导致调用多次 Service 的 onStartCommand() 方法。

2. 停止 Service

已启动的 Service 必须管理自己的生命周期，即系统不会停止或销毁 Service，除非系统必须回收系统内存而且在 onStartCommand() 方法返回后 Service 继续运行。因此，Service 必须调用 stopSelf() 方法停止自身，或者其他组件调用 stopService() 方法停止 Service。当多次启动 Service 后，仅需要一个停止方法来停止 Service。

当使用 stopSelf() 或 stopService() 方法请求停止时，系统会尽快销毁 Service。

注意 应用程序应该在任务完成后停止Service，来避免系统资源浪费和电池消耗。即便是绑定Service，如果调用了onStartCommand()方法也必须停止Service。

例 22.4 使用 Service 控制游戏的背景音乐

在 Android Studio 中创建一个 Module，名称为“Background Music Service”，实现本实例的具体步骤如下：

（1）在 Android Studio 中创建一个最小 SDK 版本为 21 的 Module，然后在 res 目录下创建 raw 子目录，并将音乐文件 music.mp3 复制到 raw 子目录中，作为播放音乐的资源文件。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后删除 TextView 组件，再为布局管理器添加背景图片，最后在该布局管理器中添加一个 ImageButton 组件，用于启动 Service 与停止 Service，具体代码请参见云盘。

（3）在 com.mingrisoft 包中，创建一个名称为 MusicService 的 Service 类，然后在该类中定义当前播放状态的变量值与 MediaPlayer 对象，具体代码如下：

```
01  public class MusicService extends Service {
02      public MusicService() {
03      }
04      static boolean isplay;                          //定义当前播放状态
05      MediaPlayer player;                             //声明MediaPlayer对象
06      @Override
07      public IBinder onBind(Intent intent) {          //必须实现的绑定方法
08          throw new UnsupportedOperationException("Not yet implemented");
09      }
10  }
```

（4）在 MusicService 类中，重写 onCreate() 方法，创建 MediaPlayer 对象并加载播放的音乐文件。关键代码如下：

```
01  @Override
02  public void onCreate() {
03      //创建MediaPlayer对象并加载播放的音乐文件
```

```
    player = MediaPlayer.create(this, R.raw.music);
}
```

（5）重写 onStartCommand() 方法，在该方法中实现音乐的播放，关键代码如下：

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) { //实现音乐的播放
    if (!player.isPlaying()) {                                      //如果没有播放音乐
        player.start();                                             //播放音乐
        isplay = player.isPlaying();                               //当前状态为正在播放音乐
    }
    return super.onStartCommand(intent, flags, startId);
}
```

（6）重写 onDestroy() 方法，在该方法中实现停止音乐的播放，关键代码如下：

```
@Override
public void onDestroy() {                  //Activity销毁时
    player.stop();                         //停止音频的播放
    isplay = player.isPlaying();           //当前状态没有播放音乐
    player.release();                      //释放资源
    super.onDestroy();
}
```

（7）打开 MainActivity 类，该类继承 Activity，在 onCreate() 方法中，单击按钮实现启动 Service 并播放背景音乐，再次单击按钮实现停止 Service 并停止播放背景音乐，关键代码如下：

```
//设置全屏显示
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
//获取"播放/停止"按钮
ImageButton btn_play = (ImageButton) findViewById(R.id.btn_play);
//启动服务与停止服务，实现播放背景音乐与停止播放背景音乐
btn_play.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (MusicService.isplay == false) {          //判断音乐播放的状态
            //启动服务，从而实现播放背景音乐
            startService(new Intent(MainActivity.this, MusicService.class));
            //更换播放背景音乐图标
            ((ImageButton) v).setImageDrawable
                    (getResources().getDrawable(R.drawable.play, null));
        } else {
            //停止服务，从而实现停止播放背景音乐
            stopService(new Intent(MainActivity.this, MusicService.class));
            //更换停止背景音乐图标
            ((ImageButton) v).setImageDrawable
                    (getResources().getDrawable(R.drawable.stop, null));
```

```
22          }
23      }
24  });
```

说明 如果没有停止Service，关闭当前应用，音乐将继续播放。

（8）重写 onStart() 方法，在该方法中实现进入界面时启动背景音乐 Service，关键代码如下：

```
01  @Override
02  protected void onStart() {  //实现进入界面时，启动背景音乐服务
03      //启动服务，从而实现播放背景音乐
04      startService(new Intent(MainActivity.this, MusicService.class));
05      super.onStart();
06  }
```

（9）运行本实例，将显示如图 22.15 所示。

图 22.15　控制游戏的背景音乐的播放

22.3.5 Bound Service

当应用程序组件通过调用 bindService() 方法绑定到 Service 时，Service 处于绑定状态。多个组件可以一次绑定到一个 Service 上，当它们都解绑定时，Service 被销毁。

如果 Service 仅用于本地应用程序并且不必跨进程工作，则开发人员可以实现自己的 Binder 类来为客户端提供访问 Service 公共方法的方式。

注意 这仅当客户端与Service位于同一个应用程序和进程时才有效，这也是最常见的情况。例如，音乐播放器需要绑定Activity到自己的Service来在后台播放音乐。

应用程序组件（客户端）能调用 bindService() 方法绑定到 Service，该方法的语法格式如下：

```
bindService(Intent service, ServiceConnection conn, int flags)
```

参数说明如下：

◆ service：通过 Intent 指定要启动的 Service。

◆ conn：一个 ServiceConnection 对象，该对象用于监听访问者与 Service 之间的连接情况。

◆ flags：指定绑定时是否自动创建 Service。该值设置为 0 时表示不自动创建，设置为 BIND_AUTO_CREATE 时表示自动创建。

接下来 Android 系统调用 Service 的 onBind() 方法，返回 IBinder 对象来与 Service 通信。

注意 只有Activity、Service和ContentProvider能绑定到Service，BroadcastReceiver不能绑定到Service。

例 22.5 模拟双色球彩票的随机选号

在 Android Studio 中创建 Module，名称为“Random Selection Number”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，并为布局管理器设置背景图片，然后添加 7 个 TextView 组件用于显示双色球的七组号码，最后添加一个用于选择随机号码的 Button 按钮。

（2）在 com.mingrisoft 包中，创建一个名称为 BinderService 的 Service 类，然后在该类中创建一个 MyBinder 内部类，用于获取 Service 对象与 Service 状态，关键代码如下：

```
01 public class MyBinder extends Binder {  //创建MyBinder内部类并获取服务对象与Service状态
02     public BinderService getService() { //创建获取Service的方法
03         return BinderService.this;      //返回当前Service类
04     }
05 }
```

（3）在必须实现的 onBind() 方法中返回 MyBinder 服务对象，关键代码如下：

```
01 @Override
02 public IBinder onBind(Intent intent) {  //必须实现的绑定方法
03     return new MyBinder();              //返回MyBinder服务对象
04 }
```

（4）创建 getRandomNumber() 方法，用于获取随机数字并将其转换为字符串保存在 ArrayList 数组中，关键代码如下：

```
01 public List getRandomNumber() {          //创建获取随机号码的方法
02     List resArr = new ArrayList();       //创建ArrayList数组
03     String strNumber="";
04     for (int i = 0; i < 7; i++) {  //将随机获取的数字转换为字符串添加到ArrayList数组中
05         int number = new Random().nextInt(33) + 1;
06         //把生成的随机数格式化为两位的字符串
07         if (number<10) {                 //在数字1~9前加0
08             strNumber = "0" + String.valueOf(number);
09         } else {
```

```
            strNumber=String.valueOf(number);
        }
        resArr.add(strNumber);
    }
    return resArr;              //将数组返回
}
```

（5）重写 onDestroy() 方法，用于销毁该 Service，具体代码如下：

```
@Override
public void onDestroy() {       //销毁该Service
    super.onDestroy();
}
```

（6）打开默认创建的 MainActivity 类，该类继承 Activity，在该类中定义 Service 类与文本框组件 ID，关键代码如下：

```
BinderService binderService;    //声明BinderService
//文本框组件ID
int[] tvid = {R.id.textView1, R.id.textView2, R.id.textView3, R.id.textView4,
        R.id.textView5, R.id.textView6, R.id.textView7};
```

（7）在 onCreate() 方法中，实现单击按钮获取随机的彩票号码，关键代码如下：

```
Button btn_random = (Button) findViewById(R.id.btn);        //获取随机选号按钮
btn_random.setOnClickListener(new View.OnClickListener() {//单击按钮，获取随机彩票号码
    @Override
    public void onClick(View v) {
        List number = binderService.getRandomNumber(); //获取BinderService类中的随机数数组
        for (int i = 0; i < number.size(); i++) {          //遍历数组并显示
            TextView tv = (TextView) findViewById(tvid[i]);//获取文本框组件对象
            String strNumber = number.get(i).toString();    //将获取的号码转为String类型
            tv.setText(strNumber);                          //显示生成的随机号码
        }
    }
});
```

（8）在 MainActivity 中，创建 ServiceConnection 对象并实现相应的方法，然后在重写的 onServiceConnected() 方法中获取后台 Service，具体代码如下：

```
private ServiceConnection conn = new ServiceConnection() {//设置与后台Service进行通讯
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        //获取后台Service信息
        binderService = ((BinderService.MyBinder) service).getService();
```

```
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
    }
};
```

说明 当Service与绑定它的组件连接成功时将回调ServiceConnection对象的onServiceConnected()方法；当Service与绑定它的组件断开连接时将回调ServiceConnection对象的onServiceDisconnected()方法。

（9）重写 onStart() 与 onStop() 方法，用于实现启动 Activity 时与后台 Service 进行绑定、关闭 Activity 时解除与后台 Service 的绑定，关键代码如下：

```
@Override
protected void onStart() { //设置启动Activity时与后台Service进行绑定
    super.onStart();
    Intent intent = new Intent(this, BinderService.class);  //创建启动Service的Intent
    bindService(intent, conn, BIND_AUTO_CREATE);            //绑定指定Service
}
@Override
protected void onStop() {  //设置关闭Activity时解除与后台Service的绑定
    super.onStop();
    unbindService(conn);   //解除绑定Service
}
```

（10）运行本实例，将显示如图 22.16 所示。

图 22.16　双色球随机选号

22.3.6 IntentService

IntentService 是 Service 的子类。在介绍 IntentService 之前，先来了解使用 Service 时需要注意

的两个问题：

◆ Service 不会专门启动一个线程来执行耗时操作，所有的操作都是在主线程中进行的，以至于容易出现 ANR（Application Not Responding）的情况。所以需要手动开启一个子线程。

◆ Service 不会自动停止，需要调用 stopSelf() 方法或者是 stopService() 方法来停止。

而使用 IntentService，则不会出现这两个问题。因为 IntentService 在开启 Service 时，会自动开启一个新的线程来执行它。另外，当 Service 运行结束后会自动停止。

例如，如果把 22.3.3 小节创建的 MyService 修改为继承 IntentService，则可以使用下面的代码来模拟执行一段耗时任务，并测试其开启和停止。

```
01  public class MyIntentService extends IntentService {
02      public MyIntentService() {
03          super("MyIntentService");
04      }
05      @Override
06      protected void onHandleIntent(Intent intent) {
07          Log.i("IntentService: ", "Service已启动");
08          //模拟一段耗时任务
09          long endTime = System.currentTimeMillis() + 5 * 1000;
10          while (System.currentTimeMillis() < endTime) {
11              synchronized (this) {
12                  try {
13                      wait(endTime - System.currentTimeMillis());
14                  } catch (Exception e) {
15                      e.printStackTrace();
16                  }
17              }
18          }
19      }
20      @Override
21      public void onDestroy() {
22          Log.i("IntentService", "Service已停止");
23      }
24  }
```

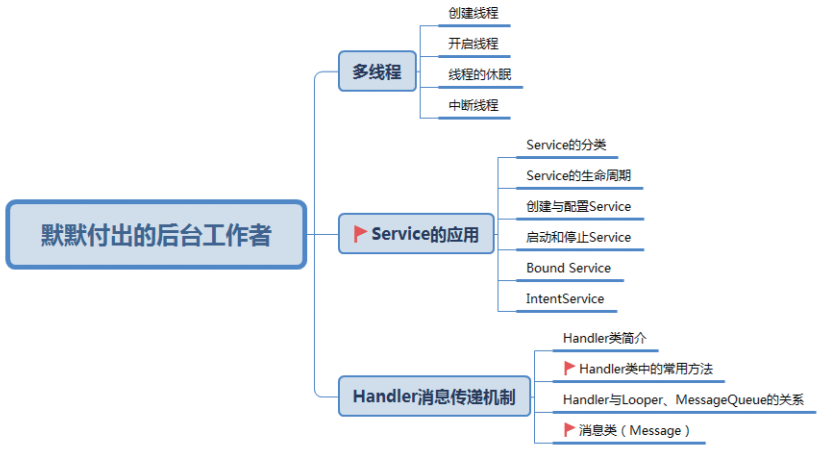
启动应用上面代码创建的 IntentService，在 LogCat 面板中将显示如图 22.17 所示的日志信息。

图 22.17　输出 Service 启动状态日志

从上面的代码中，可以看出使用 IntentService 执行耗时操作时不需要手动开启线程和停止 Service。

22.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 23 章

传感器应用

现在的 Android 手机中，都会内置一些传感器。通过这些传感器可以监测手机上发生的物理事件，而我们只要灵活的运用这些事件，就可以开发出很多方便、实用的 APP。本章将对 Android 中的传感器进行详细介绍。

23.1 Android 传感器概述

传感器是一种微型的物理设备，能够探测、感受到外界信号，并按一定规律转换成我们需要的信息。在 Android 系统中，提供了用于接收这些信息并传递给我们的 API。利用这些 API 就可以开发出想要的功能。

Android 系统中的传感器可用于监视设备的移动和位置以及周围环境的变化。例如，实现类似微信摇一摇的功能时，如图 23.1 所示，可以使用加速度传感器来监听各个方向的加速度值；实现类似神庙逃亡 2 游戏时，如图 23.2 所示，可以使用方向传感器来实现倾斜设备变道功能。

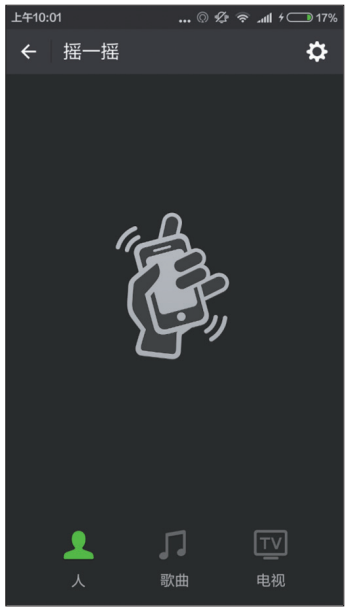


图 23.1　微信摇一摇

图 23.2　神庙逃亡 2

23.1.1 Android 的常用传感器

目前市场上很多 App 都使用到传感器。比如在一些 App 中可以自动识别屏幕的横屏或竖屏方向来改变屏幕布局，这是因为手机硬件支持重力感应和方向判断等功能。实际上 Android 系统对所有类型的传感器的处理都是一样的，只是传感器的类型有所区别。

与传感器硬件进行交互需要使用 Sensor 对象。Sensor 对象描述了它们代表的硬件传感器的属性，其中包括传感器的类型、名称、制造商以及与精确度和范围有关的详细信息。

Sensor 类包含了一组常量，这些常量描述了一个特定的 Sensor 对象所表示的硬件传感器的类型。形式为 Sensor.TYPE_<TYPE>。在 Android 中支持的传感器的类型如表 23.1 所示。

表 23.1　Android 中支持的传感器类型

名　　称	传感器类型常量	描　　述
加速度传感器	Sensor.TYPE_ACCELEROMETER	用于获取 Android 设备在 X、Y、Z 三个方向上的加速度，单位为 m/s²
重力传感器	Sensor.TYPE_GRAVITY	返回一个三维向量，这个三维向量可显示重力的方向和强度，单位为 m/s² 。其坐标系统与加速度传感器的坐标系统相同
线性加速度传感器	Sensor.TYPE_LINEAR_ACCELEROMETER	用于获取 Android 设备在 X、Y、Z 三个方向上不包括重力的加速度，单位为 m/s² 。加速度传感器、重力传感器和线性加速度传感器这三者输出值的计算公式如下：加速度传感器 = 重力传感器 + 线性加速度传感器
陀螺仪传感器	Sensor.TYPE_GYROSCOPE	用于获取 Android 设备在 X、Y、Z 这三个方向上的旋转速度，单位是弧度 / 秒。该值为正值时代表逆时针旋转，该值为负值时代表顺时针旋转
光线传感器	Sensor.TYPE_LIGHT	用于获取 Android 设备所处外界环境的光线强度，单位是勒克斯（Lux 简称 lx）
磁场传感器	Sensor.TYPE_MAGNETIC_FIELD	用于获取 Android 设备在 X、Y、Z 三个方向上的磁场数据，单位是微特斯拉（μT）
方向传感器	Sensor.TYPE_ORIENTATION	返回三个角度，这三个角度可以确定设备的摆放状态
压力传感器	Sensor.TYPE_PRESSURE	用于获取 Android 设备所处环境的压力的大小，单位为毫巴（millibars）
距离传感器	Sensor.TYPE_PROXIMITY	用于检测物体与 Android 设备的距离，单位是厘米。一些距离传感器只能返回“远”和“近”两个状态，“远”表示传感器的最大工作范围，而“近”是指比该范围小的任何值

续表

名　称	传感器类型常量	描　述
温度传感器	Sensor.TYPE_AMBIENT_TEMPERATURE	用于获取 Android 设备所处环境的温度，单位是摄氏度。这个传感器是在 Android 4.0 中引入的，用于代替已被弃用的 Sensor.TYPE _ TEMPERATURE
相对湿度传感器	Sensor.TYPE_RELATIVE_HUMIDITY	用于获取 Android 设备所处环境的相对湿度，以百分比的形式表示。这个传感器是在 Android 4.0 中引入的
旋转矢量传感器	Sensor.TYPE_ROTATION_VECTOR	返回设备的方向，它表示为 X、Y、Z 三个轴的角度的组合，是一个将坐标轴和角度混合计算得到的数据

说明 虽然Android系统中支持多种传感器类型，但并不是每个Android设备都完全支持这些传感器。

23.1.2 开发步骤

开发传感器应用大致需要经过以下 3 个步骤：

（1）调用 Context 的 getSystemService(Context.SENSOR_SERVICE) 方法来获取 SensorManager 对象。SensorManager 是所有传感器的一个综合管理类，包括传感器的种类、采样率、精准度等。调用 Context 的 getSystemService() 方法的代码如下：

```
SensorManager sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

（2）调用 SensorManager 的 getDefaultSensor(int type) 方法来获取指定类型的传感器。例如，返回默认的压力传感器的代码如下：

```
Sensor defaultPressure = sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);
```

（3）在 Activity 的 onResume() 方法中调用 SensorManager 的 registerListener() 方法为指定传感器注册监听器。程序通过实现监听器即可获取传感器传回来的数据。调用 registerListener() 方法的语法格式如下：

```
sensorManager.registerListener(SensorEventListener listener, Sensor sensor, int rate)
```

参数说明如下：

- listener：监听传感器事件的监听器。该监听器需要实现 SensorEventListener 接口。
- sensor：传感器对象。
- rate：指定获取传感器数据的频率，它支持的频率值如表 23.2 所示。

表 23.2　获取传感器数据的频率值

频　率　值	描　述
SensorManager.SENSOR_DELAY_FASTEST	尽可能快地获得传感器数据，延迟最小
SensorManager.SENSOR_DELAY_GAME	适合游戏的频率

续表

频 率 值	描 述
SensorManager.SENSOR_DELAY_NORMAL	正常频率
SensorManager.SENSOR_DELAY_UI	适合普通用户界面的频率，延迟较大

例如，使用正常频率为默认的压力传感器注册监听器的代码如下：

```
sensorManager.registerListener(this, defaultPressure, SensorManager.SENSOR_DELAY_NORMAL);
```

◆ SensorEventListener 是使用传感器的核心，其中需要实现的两个方法如下：

◎ onSensorChanged(SensorEvent event) 方法

该方法在传感器的值发生改变时调用。其参数是一个 SensorEvent 对象，通过该对象的 values 属性可以获取传感器的值，该值是一个包含了已检测到的新值的浮点型数组。不同传感器所返回的值的个数及其含义是不同的。不同传感器的返回值的详细信息如表 23.3 所示。

表 23.3 传感器的返回值

传感器名称	值 的 数 量	值 的 构 成	注 释
重力传感器	3	value[0]：X 轴 value[1]：Y 轴 value[2]：Z 轴	沿着三个坐标轴以 m/s² 为单位的重力
加速度传感器	3	value[0]：X 轴 value[1]：Y 轴 value[2]：Z 轴	沿着三个坐标轴以 m/s² 为单位的加速度
线性加速度传感器	3	value[0]：X 轴 value[1]：Y 轴 value[2]：Z 轴	沿着三个坐标轴以 m/s² 为单位的加速度，不包含重力
陀螺仪传感器	3	value[0]：X 轴 value[1]：Y 轴 value[2]：Z 轴	绕三个坐标轴的旋转速率，单位是弧度 / 秒
光线传感器	1	value[0]：照度	以勒克斯（Lux）为单位测量的外界光线强度
磁场传感器	3	value[0]：X 轴 value[1]：Y 轴 value[2]：Z 轴	以微特斯拉为单位表示的环境磁场
方向传感器	3	value[0]：X 轴 value[1]：Y 轴 value[2]：Z 轴	以角度确定设备的摆放状态
压力传感器	1	value[0]：气压	以毫巴为单位测量的气压
距离传感器	1	value[0]：距离	以厘米为单位测量的设备与目标的距离

续表

传感器名称	值 的 数 量	值 的 构 成	注　　释
温度传感器	1	value[0]：温度	以摄氏度为单位测量的环境温度
相对湿度传感器	1	value[0]：相对湿度	以百分比形式表示的相对湿度
旋转矢量传感器	3（还有一个可选参数）	value[0]：x*sin(θ/2) value[1]：y*sin(θ/2) value[2]：z*sin(θ/2) value[3]：cos(θ/2)（可选）	设备方向，以绕坐标轴的旋转角度表示

传感器的坐标系统和 Android 设备屏幕的坐标系统不同。对于大多数传感器来说，其坐标系统的 X 轴方向沿屏幕向右，Y 轴方向沿屏幕向上，Z 轴方向是垂直屏幕向上。传感器的坐标系统如图 23.3 所示。

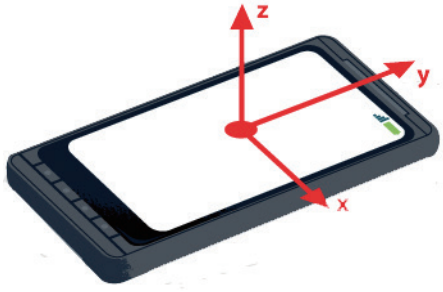


图 23.3　传感器的坐标系统

注意 在Android设备屏幕的方向发生改变时，传感器坐标系统的各坐标轴不会发生变化，即传感器的坐标系统不会因设备的移动而改变。

◎ onAccuracyChanged(Sensor sensor, int accuracy) 方法

该方法在传感器的精度发生改变时调用。参数 sensor 表示传感器对象，参数 accuracy 表示该传感器新的精度值。

以上就是开发传感器的 3 个步骤。除此之外，当应用程序不再需要接收更新时，需要注销其传感器事件监听器，代码如下：

```
sensorManager.unregisterListener(this);
```

说明 Android模拟器本身并没有提供传感器的功能，开发者需要把程序部署到具有传感器的物理设备上运行。

下面通过一个实例来演示传感器的开发步骤。

例 23.1　实时输出重力传感器和光线传感器的值

在 Android Studio 中创建一个 Module，名称为“Sensor Test”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res//layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直的线性布局管理器，然后在布局管理器中添加用于显示传感器名称的文本框组件与用于显示传感器输出信息的编辑框组件。

（2）打开默认添加的 MainActivity，然后实现 SensorEventListener 接口，再重写相应的方法，并定义所需的成员变量，最后在 onCreate() 方法中，获取布局管理器中添加的编辑框组件，并获取传感器管理对象，具体代码如下：

```
01 public class MainActivity extends AppCompatActivity implements SensorEventListener {
02     EditText textGRAVITY, textLIGHT;                    //传感器输出信息的编辑框
03     private SensorManager sensorManager;                //定义传感器管理器
04     @Override
05     protected void onCreate(Bundle savedInstanceState) {
06         super.onCreate(savedInstanceState);
07         setContentView(R.layout.activity_main);
08         //获取重力传感器输出信息的编辑框
09         textGRAVITY= (EditText) findViewById(R.id.textGRAVITY);
10         //获取光线传感器输出信息的编辑框
11         textLIGHT= (EditText) findViewById(R.id.textLIGHT);
12         //获取传感器管理
13         sensorManager= (SensorManager) getSystemService(SENSOR_SERVICE);
14     }
15     @Override
16     public void onSensorChanged(SensorEvent event) {
17     }
18     @Override
19     public void onAccuracyChanged(Sensor sensor, int accuracy) {
20     }
21 }
```

（3）重写 onResume() 方法，实现当界面获取焦点时为传感器注册监听器，具体代码如下：

```
01 @Override
02 protected void onResume() {
03     super.onResume();
04     //为重力传感器注册监听器
05     sensorManager.registerListener(this,
06             sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY),
07             SensorManager.SENSOR_DELAY_GAME);
08     //为光线传感器注册监听器
09     sensorManager.registerListener(this,
10             sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
11             SensorManager.SENSOR_DELAY_GAME);
12 }
```

（4）重写 onPause() 与 onStop() 方法，并且在这两个方法中取消注册的监听器，具体代码如下：

```
01 @Override
02 protected void onPause() {  //取消注册监听器
03     sensorManager.unregisterListener(this);
```

```
    super.onPause();
}
@Override07       protected void onStop() {  //取消注册监听器
    sensorManager.unregisterListener(this);
    super.onStop();
}
```

（5）重写 onSensorChanged() 方法，在该方法中首先获取传感器 X、Y、Z 三个轴的输出信息，然后获取传感器类型，并输出相应传感器的信息，关键代码如下：

```
float[] values = event.values;              //获取X、Y、Z三轴的输出信息
int sensorType = event.sensor.getType();    //获取传感器类型
switch (sensorType) {
    case Sensor.TYPE_GRAVITY:
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("X轴横向重力值:");
        stringBuilder.append(values[0]);
        stringBuilder.append("\nY轴纵向重力值:");
        stringBuilder.append(values[1]);
        stringBuilder.append("\nZ轴向上重力值:");
        stringBuilder.append(values[2]);
        textGRAVITY.setText(stringBuilder.toString());
        break;
    case Sensor.TYPE_LIGHT:
        stringBuilder = new StringBuilder();
        stringBuilder.append("光的强度值:");
        stringBuilder.append(values[0]);
        textLIGHT.setText(stringBuilder.toString());
        break;
}
```

（6）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其竖屏显示，关键代码如下：

```
android:screenOrientation="portrait"
```

（7）运行本实例，将显示如图 23.4 所示。

图 23.4　获取传感器输出信息

23.2 方向传感器

方向传感器简称为 O-sensor，它用于感应 Android 设备的摆放状态。方向传感器可以返回三个角度，第 1 个代表在 Z 轴上旋转的角度、第 2 个代表在 X 轴上旋转的角度、第 3 个代表在 Y 轴上旋转的角度。

在以前的 Android SDK 中，我们可以通过 SensorManager 对象的 getDefaultSensor(Sensor.TYPE_ORIENTATION) 方法可以获取到方向传感器，但是在最新版本的 SDK 中提示这种方式已过期，不建议使用，因此 Google 建议使用加速度传感器和磁场传感器组合计算出方向和角度值。其步骤如下：

（1）获得加速度传感器和磁场传感器的实例，并为它们注册监听器。

（2）在 onSensorChanged() 方法中，分别获取加速度传感器和磁场传感器的值，并传入到 getRotationMatrix() 方法中，从而得出一个包含旋转矩阵的 R 数组。该数组中保存着磁场和加速度的数据。 getRotationMatrix() 方法的语法如下：

```
public static boolean getRotationMatrix
        (float[] R, float[] I, float[] gravity, float[] geomagnetic)
```

getRotationMatrix() 方法的参数如表 23.4 所示。

表 23.4 getRotationMatrix() 方法的参数

参　数	描　述
R	需要填充的 float 型数组，大小是 9
I	一个转换矩阵，将磁场数据转换进实际的重力坐标中，一般情况下可以设置为 null
gravity	一个大小为 3 的 float 型数组，表示从加速度传感器获取来的数据
geomagnetic	一个大小为 3float 型的数组，表示从磁场传感器获取来的数据

（3）通过 SensorManager.getOrientation() 方法来获得所需的旋转数据。getOrientation() 方法的语法如下：

```
public static float[] getOrientation (float[] R, float[] values)
```

☑ 参数 R 是步骤（2）得到的旋转矩阵，通过该值求出方位角；

☑ 参数 values 是一个包括 3 个元素的 float 类型的数组，手机在各个方向上的旋转数据都会被保存到这个数组中。每个数组元素代表的值如表 23.5 所示。

表 23.5 values 数组的数组元素描述

数组元素	描　述
values[0]	手机在 Z 轴上旋转时，手机顶部朝向与正北方的夹角。如果用“磁场 + 加速度”的方式得到的数据范围是 -180 ～ 180 度，也就是说，0 度表示正北，90 度表示正东，180/-180 度表示正南，-90 度表示正西；而直接通过方向传感器得到的数据范围是 0 ～ 359 度，0 度表示正北，90 度表示正东，180 度表示正南，270 度表示正西

续表

数 组 元 素	描　　述
values[1]	手机在 X 轴上旋转时（即手机前后翻转时）手机与水平面形成的夹角，手机顶部向上抬起时，该角度的范围是 0~-90 度，手机尾部向上抬起时，该角度的范围是 0~90 度
values[2]	手机在 Y 轴上旋转时（即手机左右翻转时）手机与水平面形成的夹角，手机左侧抬起时，该角度的范围是 0~90 度，手机右侧抬起时，该角度的范围是 0~-90 度

表 18.5 中的 values[0]、values[1] 和 values[2] 代表的旋转方向如图 23.5 所示。

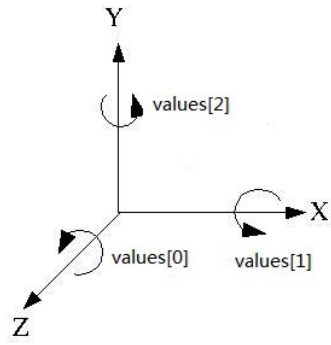


图 23.5　values 数组各元素代表的旋转方向

注意 通过getOrientation()方法计算得到的数据是以弧度为单位的，一般情况下，我们都会使用角度为旋转角度的单位，所以需要使用Math.toDegrees()方法进行转换。例如，将values[0]转换为角度可以使用下面的代码:

```
Math.toDegrees(values[0]);
```

通过使用方向传感器，应用程序就可以检测到设备的摆放状态，比如手机顶部或尾部的朝向、倾斜角度等。因此，借助于方向传感器可以开发出水平仪等应用。

下面通过一个实例来演示方向传感器的应用。

例 23.2　通过方向传感器实现一个水平仪

在 Android Studio 中创建 Module，名称为“Level”，实现本实例的具体步骤如下：

（1）创建一个名称为 SpiritlevelView 的类，该类继承自 android.view.View 类并且实现 SensorEventListener 接口，再重写相应的方法。具体代码如下：

```
01  public class SpiritlevelView extends View implements SensorEventListener {
02          public SpiritlevelView(Context context, @Nullable AttributeSet attrs) {
03          super(context, attrs);
04      }
05      @Override
06      public void onSensorChanged(SensorEvent event) {
07      }
08      @Override
09      public void onAccuracyChanged(Sensor sensor, int accuracy) {
10      }
11  }
```

（2）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为帧布局管理器，然后将内边距与默认添加的 TextView 组件删除并设置背景图片。最后在帧布局管理器中添加步骤 (1) 中创建的自定义 View。

（3）打开 SpiritlevelView 类，在 SpiritlevelView 类中，定义所需的成员变量。关键代码如下：

```
01 private Bitmap bubble;          //定义水平仪中的小蓝球位图
02 private int MAX_ANGLE = 30;     //定义水平仪最大倾斜角，超过该角度，小蓝球将直接位于边界
03 private int bubbleX, bubbleY;   //定义水平仪中小蓝球的X、Y坐标
```

（4）在 SpiritlevelView 类的构造方法中，首先获取要绘制的小蓝球位图与传感器管理，然后为磁场传感器和加速度传感器注册监听器。具体代码如下：

```
01 public SpiritlevelView(Context context, AttributeSet attrs) {
02     super(context, attrs);
03     bubble = BitmapFactory                                  //加载小蓝球图片
04             .decodeResource(getResources(), R.drawable.bubble);
05     SensorManager sensorManager = (SensorManager) context
06             .getSystemService(Context.SENSOR_SERVICE);      //获取传感器管理
07     sensorManager.registerListener(this,                    //为磁场传感器注册监听器
08             sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
09             SensorManager.SENSOR_DELAY_GAME);
10     sensorManager.registerListener(this,                    //为加速度传感器注册监听器
11             sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
12             SensorManager.SENSOR_DELAY_GAME);
13 }
```

（5）在 SpiritlevelView 类中，创建传感器的取值数组，关键代码如下：

```
01 float[] accelerometerValues = new float[3];    //创建加速度传感器Z轴、X轴、Y轴取值数组
02 float[] magneticValues = new float[3];         //创建磁场传感器Z轴、X轴、Y轴取值数组
```

（6）重写 onSensorChanged() 方法，在该方法中首先获取方向信息，然后调用 getPosition() 方法计算小篮球的动态位置。关键代码如下：

```
01 @Override
02 public void onSensorChanged(SensorEvent event) {
03     //如果当前为加速度传感器
04     if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
05         //将取出的值放到加速度传感器取值数组中
06         accelerometerValues = event.values.clone();
07         //如果当前为磁场传感器
08     } else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
09         magneticValues = event.values.clone(); //将取出的值放到磁场传感器取值数组中
10     }
11     float[] R = new float[9];                          //创建存放旋转数据的取值数组
12     float[] values = new float[3];                     //创建存放方向数据的取值数组
13     SensorManager.getRotationMatrix(R, null, accelerometerValues, magneticValues);
14     SensorManager.getOrientation(R, values);    //获取方向Z轴、X轴、Y轴信息值
```

```
15      float xAngle = (float) Math.toDegrees(values[1]);    //获取与X轴的夹角
16      float yAngle = (float) Math.toDegrees(values[2]);    //获取与Y轴的夹角
17      getPosition(xAngle,yAngle);                          //获取小蓝球的位置坐标
18      super.postInvalidate();                              //刷新界面
19  }
```

（7）编写自定义方法 getPosition()，用于根据 X 轴和 Y 轴的旋转角度确定小蓝球的位置，具体代码如下：

```
01. private  void getPosition(float xAngle,float yAngle){
02.     //小蓝球位于中间时（水平仪完全水平），小蓝球的X、Y坐标
03.     int x = (super.getWidth() - bubble.getWidth()) / 2;
04.     int y = (super.getHeight() - bubble.getHeight()) / 2;
05.     /********控制小球的X轴位置******/
06.     if (Math.abs(yAngle) <= MAX_ANGLE) {    //如果Y轴的倾斜角度还在最大角度之内
07.         //根据Y轴的倾斜角度计算X坐标的变化值（倾斜角度越大，X坐标变化越大）
08.         int deltaX = (int)
09.                 ((super.getWidth() - bubble.getWidth()) / 2 * yAngle / MAX_ANGLE);
10.         x -= deltaX;
11.         //如果Y轴的倾斜角度已经大于MAX_ANGLE，小蓝球在最左边
12.     } else if (yAngle > MAX_ANGLE) {
13.         x = 0;
14.     } else {  //如果与Y轴的倾斜角已经小于负的MAX_ANGLE，小蓝球在最右边
15.         x = super.getWidth() - bubble.getWidth();
16.     }
17.     /********控制小球的Y轴位置******/
18.     if (Math.abs(xAngle) <= MAX_ANGLE) {    //如果X轴的倾斜角度还在最大角度之内
19.         //根据X轴的倾斜角度计算Y坐标的变化值（倾斜角度越大，Y坐标变化越大）
20.         int deltaY = (int)
21.                 ((super.getHeight() - bubble.getHeight()) / 2 * xAngle / MAX_ANGLE);
22.         y += deltaY;
23.         //如果与X轴的倾斜角度已经大于MAX_ANGLE，小蓝球在最下边
24.     } else if (xAngle > MAX_ANGLE) {
25.         y = super.getHeight() - bubble.getHeight();
26.     } else {  //如果X轴的倾斜角已经小于负的MAX_ANGLE，小蓝球在最上边
27.         y = 0;
28.     }
29.     //更新小蓝球的坐标
30.     bubbleX = x;
31.     bubbleY = y;
32. }
```

（8）重写 onDraw() 方法，在该方法中首先根据方向传感器的坐标绘制小蓝球的位置。关键代码如下：

```
01  @Override
02  protected void onDraw(Canvas canvas) {
03      super.onDraw(canvas);
```

```
04        //根据小蓝球坐标绘制小蓝球
05        canvas.drawBitmap(bubble, bubbleX, bubbleY, null);
06    }
```

（9）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其竖屏显示，关键代码如下：

```
android:screenOrientation="portrait"
```

（10）运行本实例，将显示如图 23.6 所示。

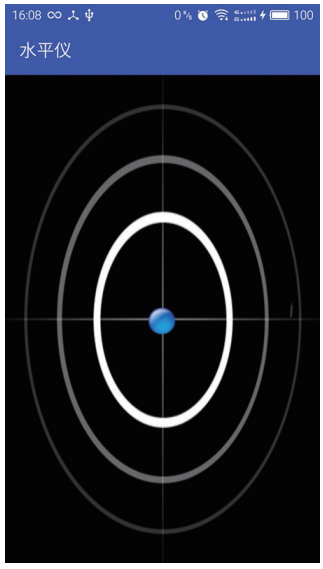


图 23.6　方向传感器的水平仪

23.3　磁场传感器

磁场传感器简称为 M-sensor，主要用于读取 Android 设备外部的磁场强度。随着 Android 设备位置移动和摆放状态的改变，周围的磁场在设备 X、Y、Z 三个坐标方向上的影响也会发生改变。

磁场传感器会返回 3 个数据，这 3 个数据分别代表 X、Y、Z 三个坐标方向上的磁场数据。该数值的单位是微特斯拉（μT）。

通过使用磁场传感器，应用程序就可以检测到设备周围的磁场强度，因此，借助于磁场传感器可以开发出指南针等应用。

下面通过一个实例来演示磁场传感器的应用。

例 23.3　使用磁场传感器实现指南针

在 Android Studio 中创建 Module，名称为“Compass”。实现本实例的具体步骤如下：

（1）创建一个名称为 PointerView 类，该类继承自 android.view.View 类并且实现 SensorEventListener 接口，再重写相应的方法，最后定义所需的成员变量，具体代码如下：

```
01 public class PointerView extends View implements SensorEventListener {
02     private Bitmap pointer = null;              //定义指针位图
03     private float[] allValue;                   //定义传感器三轴的输出信息
04     private SensorManager sensorManager;        //定义传感器管理器
05     public PointerView(Context context, AttributeSet attrs) {
06         super(context, attrs);
07     }
08     @Override
09     public void onSensorChanged(SensorEvent event) {
10     }
11     @Override
12     public void onAccuracyChanged(Sensor sensor, int accuracy) {
13     }
14     @Override
15     protected void onDraw(Canvas canvas) {
16         super.onDraw(canvas);
17     }
18 }
```

（2）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为帧布局管理器，然后将默认添加的 TextView 组件删除。并且在帧布局管理器中添加一个 ImageView 组件，用于显示背景图，最后添加步骤（1）中创建的自定义 View。修改后的代码如下：

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <FrameLayout
03     xmlns:android="http://schemas.android.com/apk/res/android"
04     xmlns:tools="http://schemas.android.com/tools"
05     android:layout_width="match_parent"
06     android:layout_height="match_parent"
07     tools:context="com.mingrisoft.MainActivity">
08     <ImageView
09         android:id="@+id/background"
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:layout_gravity="center"
13         android:src="@drawable/background"
14         />
15     <!--添加自定义View-->
16     <com.mingrisoft.PointerView
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content" />
19 </FrameLayout>
```

（3）打开 PointerView 类，在 PointerView 类的构造方法中，首先获取要绘制的指针位图与传感器管理器，然后为磁场传感器注册监听器。关键代码如下：

```
01 pointer = BitmapFactory.decodeResource(super.getResources(),
```

```
        R.drawable.pointer); //获取要绘制的指针位图
//获取传感器管理器
sensorManager = (SensorManager) context
        .getSystemService(Context.SENSOR_SERVICE);
//为磁场传感器注册监听器
sensorManager.registerListener(this,
        sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
        SensorManager.SENSOR_DELAY_GAME);
```

（4）重写 onSensorChanged() 方法，在该方法中首先判断获取到的是否是磁场传感器，然后获取磁场传感器 X、Y、Z 三个坐标轴的输出信息并保存信息，最后通过 super.postInvalidate() 方法刷新界面。关键代码如下：

```
if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) { //如果是磁场传感器
    float value[] = event.values;                          //获取磁场传感器三轴的输出信息
    allValue = value;                                      //保存输出信息
    super.postInvalidate();                                //刷新界面
}
```

（5）重写 onDraw() 方法，在该方法中首先根据磁场传感器的坐标计算指针的角度，然后绘制指针。关键代码如下：

```
Paint p = new Paint();              //创建画笔
if (allValue != null) {             //传感器三轴输出信息不为空
    float x = allValue[0];          //获取X轴坐标
    float y = allValue[1];          //获取Y轴坐标
    canvas.save();                  //保存Canvas的状态
    canvas.restore();               //重置绘图对象
    //以屏幕中心点作为旋转中心
    canvas.translate(super.getWidth() / 2, super.getHeight() / 2);
    //判断Y轴为0时的旋转角度
    if (y == 0 && x > 0) {
        canvas.rotate(90);          //旋转角度为90度
    } else if (y == 0 && x < 0) {
        canvas.rotate(270);         //旋转角度为270度
    } else {
        //通过三角函数tanh()方法计算旋转角度
        if (y >= 0) {
            canvas.rotate((float) Math.tanh(x / y) * 90);
        } else {
            canvas.rotate(180 + (float) Math.tanh(x / y) * 90);
        }
    }
}
//绘制指针
canvas.drawBitmap(this.pointer, -this.pointer.getWidth() / 2,
        -this.pointer.getHeight() / 2, p);
```

（6）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其竖屏显示，关键代码如下：

```
android:screenOrientation="portrait"
```

（7）运行本实例，将显示如图 23.7 所示的界面效果。

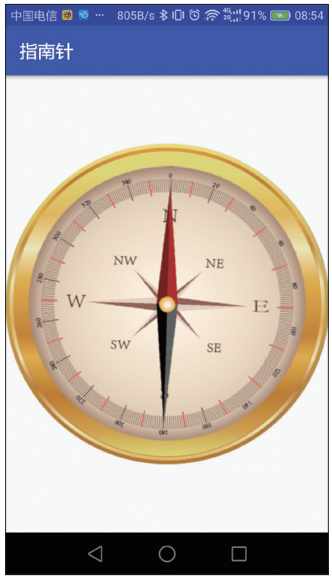


图 23.7　指南针

23.4　加速度传感器

加速度传感器是用于检测设备加速度的传感器。对于加速度传感器来说，SensorEvent 对象的 values 属性将返回 3 个值，分别代表 Android 设备在 X、Y、Z 三个坐标方向上的加速度，单位为 m/s^2。当 Android 设备横向左右移动时，可能产生 X 轴上的加速度，当 Android 设备前后移动时，可能产生 Y 轴上的加速度，当 Android 设备垂直上下移动时，可能产生 Z 轴上的加速度。

通过使用加速度传感器，可以开发出类似微信摇一摇以及运动 App 的计步功能。

下面通过一个实例来演示加速度传感器的应用。

例 23.4　使用加速度传感器实现摇红包

在 Android Studio 中创建 Module，名称为“Shake Red Packet”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后将 TextView 组件删除，再为布局管理器添加背景。

（2）创建一个名称为 packet.xml 的布局文件，在该布局文件中添加一个 ImageView 组件，用于显示红包图片。

（3）打开默认添加的 MainActivity，让 MainActivity 实现 SensorEventListener 接口，再重写相应的方法，最后在该类中，定义所需的成员变量，关键代码如下：

```
01 private SensorManager sensorManager;      //定义传感器管理器
02 private Vibrator vibrator;                //定义振动器
```

（4）在 onCreate() 方法中，获取传感器管理器与振动器服务。关键代码如下：

```
01 sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);  //获取传感器管理器
02 vibrator = (Vibrator) getSystemService(Service.VIBRATOR_SERVICE);  //获取振动器服务
```

（5）重写 onResume() 方法，并在该方法中为传感器注册监听器。具体代码如下：

```
01 @Override
02 protected void onResume() {
03     super.onResume();
04     //为加速度传感器注册监听器
05     sensorManager.registerListener(this, sensorManager.getDefaultSensor
06             (Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_GAME);
07 }
```

（6）重写 onSensorChanged() 方法，实现摇动手机，显示红包的功能。具体代码如下：

```
01 @Override
02 public void onSensorChanged(SensorEvent event) {
03     float[] values = event.values;                  //获取传感器X、Y、Z三个坐标轴的输出信息
04     int sensorType = event.sensor.getType();        //获取传感器类型
05     if (sensorType == Sensor.TYPE_ACCELEROMETER) {  //如果是加速度传感器
06         //X轴输出信息>15,Y轴输出信息>15,Z轴输出信息>20
07         if (values[0] > 15 || values[1] > 15 || values[2] > 20) {
08             Toast.makeText(MainActivity.this, "摇一摇", Toast.LENGTH_SHORT).show();
09             //创建AlertDialog.Builder对象
10             AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
11             alertDialog.setView(R.layout.packet);        //添加布局文件
12             alertDialog.show();                          //显示alertDialog
13             vibrator.vibrate(500);                       //设置振动器频率
14             sensorManager.unregisterListener(this);      //取消注册的监听器
15         }
16     }
17 }
```

（7）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，设置其竖屏显示，关键代码如下：

```
android:screenOrientation="portrait"
```

（8）打开 AndroidManifest.xml 文件，在其中设置振动器的使用权限，具体代码如下：

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

（9）运行本实例，将显示如图 23.8 所示的界面，摇动手机后显示如图 23.9 所示的红包。

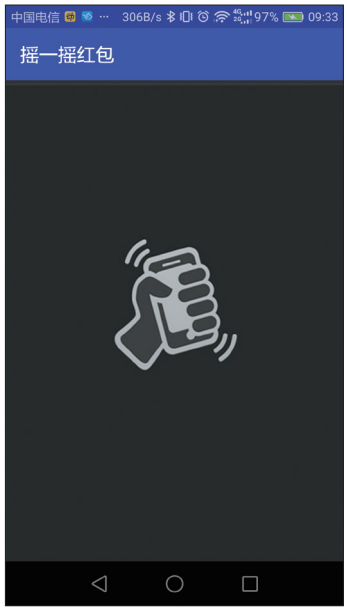


图 23.8　摇一摇界面

图 23.9　显示红包

23.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。

第 24 章

位置服务与地图应用

由于移动设备比电脑更容易随身携带，所以在移动设备上使用位置服务与地图应用，就更加实用了。本章将对 Android 中的位置服务与百度地图应用进行详细的介绍。

24.1 位 置 服 务

在开发 Android 位置服务相关应用时，可以从 GPS（全球定位系统）或者网络获得用户位置。通过 GPS 能获得最精确的信息。例如，在某些外卖软件的“选择收货地址”页面中，不仅定位了当前所在城市，而且附近地址也被列出，界面如图 24.1 所示；再如，某些天气软件能够自动定位当前所在城市，界面如图 24.2 所示。

图 24.1　定位当前位置

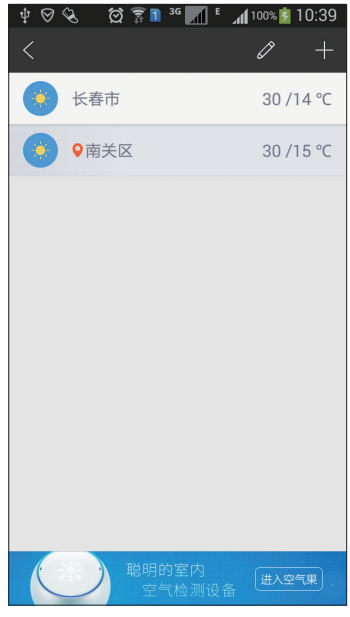


图 24.2　自动定位当前所在城市

对于开发 Android 应用的程序员来说，开发 GPS 功能的应用程序非常简单。在 Android 系统中，开发人员需要使用以下类访问定位服务。

◆ LocationManager：该类提供系统定位服务访问功能。

LocationManager 提供的常用方法如表 24.1 所示。

表 24.1 LocationManager 提供的常用方法

方 法 名	描 述
List<String> getAllProviders()	获取所有的 LocationProvider 列表
Location getLastKnownLocation(String provider)	根据 LocationProvider 获取最近一次已知的 Location
LocationProvider getProvider(String name)	根据名称来获取 LocationProvider
void requestLocationUpdates (String provider, long minTime, float minDistance, PendingIntent intent)	通过指定的 LocationProvider 周期性地获取定位信息，并通过 intent 启动相应的组件
void requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)	通过指定的 LocationProvider 周期性地获取定位信息，并触发 listener 所对应的触发器

☑ LocationProvider：定位组件的抽象表示，通过该类可以获取该定位组件的相关信息。

LocationProvider 提供的常用方法如表 24.2 所示。

表 24.2 LocationProvider 提供的常用方法

方 法 名	描 述
int getAccuracy()	返回 LocationProvider 的精度
String getName()	返回 LocationProvider 的名称
int getPowerRequirement()	获取 LocationProvider 的电源需求

◆ Location：该类表示特定时间的地理位置信息，位置由经度、纬度、UTC 时间戳以及可选的高度、速度、方向等组成。

Location 提供的常用方法如表 24.3 所示。

表 24.3 Location 提供的常用方法

方 法 名	描 述	方 法 名	描 述
float getAccuracy()	获取定位信息的精度	double getLongitude()	获取定位信息的经度
double getAltitude()	获取定位信息的高度	String getProvider()	获取提供该定位信息的 LocationProvider
float getBearing()	获取定位信息的方向	float getSpeed()	获取定位信息的速度
double getLatitude()	获取定位信息的纬度		

上面的 3 个 API 就是 GPS 定位支持的 3 个核心 API，使用它们获取 GPS 定位信息的一般步骤如下：

（1）获取系统的 LocationManager 对象。

（2）使用 LocationManager，通过指定 LocationProvider 来获取定位信息，定位信息由 Location 对象来表示。

（3）从 Location 对象中获取定位信息。

24.1.1 获取 LocationProvider

LocationProvider 是位置源的意思，用来提供定位信息。在获取定位信息之前，需要先获得 Location Provider 对象。常用的 LocationProvider 如表 24.4 所示。

表 24.4 常用的 LocationProvider

方 法 名	描 述
passive	被动定位方式，即利用其他应用程序使用定位更新了定位信息时，系统会保存下来，该应用接收到消息后直接读取就可以了
gps	通过手机里的 GPS 芯片利用卫星获取定位信息
network	通过网络获取定位信息。通常利用手机基站和 WIFI 节点的地址来大致定位

下面将介绍如何获得 Location Provider。

1．获取所有可用的 LocationProvider

在 LocationManager 中，提供了一个 getAllProviders() 方法用来获取系统所有可用的 LocationProvider。下面通过一个例子来演示如何获得当前支持的全部 LocationProvider。

例 24.1 获取所有可用的 LocationProvider 名称

在 Android Studio 中创建一个 Module，名称为“Location Provider”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并添加背景图片，然后为默认添加的 TextView 组件设置属性值，最后再添加一个 TextView 组件，用于显示获取的 LocationProvider 名称。

（2）打开默认添加的 MainActivity 类，让该类继承自 Activity，然后在 onCreate() 方法中完成获取及显示 LocationProvider 名称。具体代码如下：

```
01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         //设置全屏显示
07         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                 WindowManager.LayoutParams.FLAG_FULLSCREEN);
09         //获取显示LocationProvider名称的TextView组件
10         TextView textView = (TextView) findViewById(R.id.provider);
11         //获取location服务
12         LocationManager locationManager = (LocationManager)
13                 getSystemService(LOCATION_SERVICE);
14         //获取系统所有的LocationProvider名称
15         List<String> providersNames = locationManager.getAllProviders();
16         StringBuilder stringBuilder = new StringBuilder();//使用StringBuilder保存数据
17         //遍历获取到的全部LocationProvider名称
18         for (Iterator<String> iterator = providersNames.iterator();
19                                                         iterator.hasNext(); ) {
```

```
20              stringBuilder.append(iterator.next() + "\n");
21          }
22          textView.setText(stringBuilder.toString());  //显示LocationProvider名称
23      }
24  }
```

（3）运行本实例，将显示如图 24.3 所示的界面效果。

图 24.3　获取所有可用的 LocationProvider 名称

2．通过名称获得 LocationProvider

在通过调用 LocationManager 的 getAllProviders() 方法获取所有的 LocationProvider 时返回的是 List<String> 集合，集合中的元素为 LocationProvider 的名称。为了获取实际的 LocationProvider 对象，可以通过 LocationManager 的 getProvider() 方法。例如下面的代码可以获取基于 GPS 的 LocationProvider。

```
LocationProvider lp = lm.getProvider(LocationManager.GPS_PROVIDER);
```

3．通过 Criteria 类获得 LocationProvider

通过 getAllProviders() 方法可以获取系统所有可用的 LocationProvider。但有些时候，应用程序可能希望得到符合指定条件的 LocationProvider，这就需要使用 LocationManager 的 getBestProvider(Criteria criteria, boolean enabledOnly) 方法来获取。在该方法中需要借助于 Criteria 类。

对于位置源而言，有两种用户十分关心的属性：精度和耗电量。在 Criteria 类中，保存了关于精度和耗电量的信息，其说明如表 24.5 所示。

表 24.5　Criteria 类定义的精度和耗电信息

常　量	说　明	常　量	说　明
ACCURACY_COARSE	近似的精度	NO_REQUIREMENT	无要求
ACCURACY_FINE	更精细的精度	POWER_HIGH	高耗电量
ACCURACY_HIGH	高等精度	POWER_MEDIUM	中耗电量

续表

常　量	说　明	常　量	说　明
ACCURACY_MEDIUM	中等精度	POWER_LOW	低耗电量
ACCURACY_LOW	低等精度		

下面通过一个实例来演示通过 Criteria 类获得 LocationProvider 名称。

例 24.2 通过 Criteria 类获得 LocationProvider 名称

在 Android Studio 中创建 Module，名称为“Criteria Location Provider”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后为布局管理器添加背景图片，再为默认添加的 TextView 组件与新添加的 TextView 组件设置属性值，用于显示获取的 LocationProvider 名称。

（2）打开默认添加的 MainActivity 类，该类继承 Activity，在 onCreate() 方法中，完成通过 Criteria 类过滤条件来获取及显示 LocationProvider 名称。具体代码如下：

```
01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         //设置全屏显示
07         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                 WindowManager.LayoutParams.FLAG_FULLSCREEN);
09         //获取显示最佳LocationProvider的TextView组件
10         TextView textView = (TextView) findViewById(R.id.provider);
11         //获取location服务
12         LocationManager locationManager =
13                                 (LocationManager) getSystemService(LOCATION_SERVICE);
14         Criteria criteria=new Criteria();                  //创建过滤条件
15         criteria.setCostAllowed(false);                    //使用不收费的
16         criteria.setAccuracy(Criteria.ACCURACY_FINE);      //使用精度最准确的
17         criteria.setPowerRequirement(Criteria.POWER_LOW);//使用耗电量最低的
18         //获取最佳的LocationProvider名称
19         String provider=locationManager.getBestProvider(criteria,true);
20         textView.setText(provider);  //显示最佳的LocationProvider名称
21     }
22 }
```

（3）打开 AndroidManifest.xml 文件，在其中设置获取最佳的 LocationProvider 权限，具体代码如下：

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

（4）运行本实例，将显示如图 24.4 所示的界面效果。

图 24.4　通过 Criteria 类获得 LocationProvider 名称

24.1.2　获取定位信息

通过手机可以实时获取定位信息，包括用户所在位置的经度、纬度、高度以及方向等。对于位置发生变化的用户，可以在变化后接收到相关的通知。在 LocationManager 类中，定义了多个 requestLocationUpdates() 方法，用来为当前 Activity 注册位置变化通知事件。该方法的声明如下：

```
public void requestLocationUpdates (String provider, long minTime, float minDistance,
LocationListener listener)
```

- provider：注册的 provider 的名称，可以是 GPS_PROVIDER 等。
- minTime：通知间隔的最小时间，单位是毫秒。系统可能为了省电而延长该时间。
- minDistance：更新通知的最小变化距离，单位是米。
- listener：用于处理通知的监听器。

在 LocationListener 接口中，定义了 4 个方法，其说明如表 24.6 所示。

表 24.6　LocationListener 接口中方法说明

方　法	说　明
onLocationChanged	当位置发生变化时调用该方法
onProviderDisabled	当 provider 禁用时调用该方法
onProviderEnabled	当 provider 启用时调用该方法
onStatusChanged	当状态发生变化时调用该方法

例 24.3　获取动态定位信息

在 Android Studio 中创建 Module，名称为“Get Location Information”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并添加背景图片，然后为默认添加的 TextView 组件设置属性值，用于显示获取定位的信息。

（2）打开 MainActivity 类，该类继承 Activity，定义所需的成员变量，并在 onCreate() 方法中，获取系统的 LocationManager 对象，具体代码如下：

```
01  public class MainActivity extends Activity {
02      private TextView text;  //定义用于显示LocationProvider的TextView组件
03      @Override
04      public void onCreate(Bundle savedInstanceState) {
05          super.onCreate(savedInstanceState);
06          setContentView(R.layout.activity_main);
07          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                  WindowManager.LayoutParams.FLAG_FULLSCREEN);    //设置全屏显示
09          //获取显示Location信息的TextView组件
10          text = (TextView) findViewById(R.id.location);
11          //获取系统的LocationManager对象
12          LocationManager locationManager =
13                  (LocationManager) getSystemService(LOCATION_SERVICE);
14      }
15  }
```

（3）在 MainActivity 中，创建 locationUpdates() 方法，用于获取指定的查询信息并显示。具体代码如下：

```
01  public void locationUpdates(Location location) {          //获取指定的查询信息
02      //如果location不为空时
03      if (location != null) {
04          StringBuilder stringBuilder = new StringBuilder();//使用StringBuilder保存数据
05          //获取经度、纬度、等属性值
06          stringBuilder.append("您的位置信息：\n");
07          stringBuilder.append("经度：");
08          stringBuilder.append(location.getLongitude());
09          stringBuilder.append("\n纬度：");
10          stringBuilder.append(location.getLatitude());
11          stringBuilder.append("\n精确度：");
12          stringBuilder.append(location.getAccuracy());
13          stringBuilder.append("\n高度：");
14          stringBuilder.append(location.getAltitude());
15          stringBuilder.append("\n方向：");
16          stringBuilder.append(location.getBearing());
17          stringBuilder.append("\n速度：");
18          stringBuilder.append(location.getSpeed());
19          stringBuilder.append("\n时间：");
20          //设置日期时间格式
21          SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH mm ss");
22          stringBuilder.append(dateFormat.format(new Date(location.getTime())));
23          text.setText(stringBuilder);                       //显示获取的信息
24      } else {
25          //否则输出空信息
26          text.setText("没有获取到GPS信息");
```

```
    }
}
```

（4）在 onCreate() 方法中，通过 locationManager.requestLocationUpdates() 方法与位置监听器来设置每秒获取一次 location 信息，并将最新的定位信息传递给创建的 locationUpdates() 方法中。关键代码如下：

```
//设置每秒获取一次location信息
locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,       //GPS定位提供者
        1000,                               //更新数据时间为1000毫秒
        1,                                  //位置间隔为1米
        //位置监听器
        new LocationListener() {//GPS定位信息发生改变时触发，用于更新位置信息
            @Override
            public void onLocationChanged(Location location) {
                //GPS信息发生改变时，更新位置
                locationUpdates(location);
            }

            @Override
            //位置状态发生改变时触发
            public void onStatusChanged(String provider, int status, Bundle extras) {
            }
            @Override
            //定位提供者启动时触发
            public void onProviderEnabled(String provider) {
            }
            @Override
            //定位提供者关闭时触发
            public void onProviderDisabled(String provider) {
            }
        });
//从GPS获取最新的定位信息
Location location = locationManager.getLastKnownLocation
        (LocationManager.GPS_PROVIDER);
locationUpdates(location);      //将最新的定位信息传递给创建的locationUpdates()方法中
```

说明 添加上面的代码后，在代码的下方将出现红色浪线，此时按下〈Alt+Enter〉键，选择Add Permission ACCESS_FINE_LOCATION选项，然后再次按下〈Alt+Enter〉键，选择Add permission check 选项，解决红色浪线问题。此时，Android Studio将自动添加以下权限检查的代码。

```
if (ActivityCompat.checkSelfPermission(this,
   Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
   && ActivityCompat.checkSelfPermission(this,
   Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
   return;
}
```

（5）在 AndroidManifest.xml 文件中，添加以下代码设置访问 LocationProvider 的相关权限，具体代码如下：

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

（6）运行本实例，将显示如图 24.5 所示。

图 24.5　获取动态定位信息

说明　在运行本实例时，需要开启手机的GPS功能，同时需要开启该应用的定位权限。本实例在测试的过程中建议在室外进行测试，保证GPS的信号质量。

24.2　百度地图服务

百度地图服务是目前很常用的地图服务。为了便于 Android 开发者使用百度地图，百度提供了百度地图 Android SDK。它是一套基于 Android 2.1 及以上版本设备的应用程序接口，适用于 Android 系统移动设备的地图应用开发。通过调用地图 SDK 接口，可以轻松访问百度地图服务和数据，构建功能丰富、交互性强的地图类应用程序。

24.2.1　获得地图 API 密钥

在使用百度地图 Android SDK 时，首先需要申请密钥（API Key）。如果没有 API 密钥，就不会向应用程序返回任何地图图块（map tile）。由于该密钥与百度账号相关联，而且还与创建的过程名称有关，因此，在申请密钥前，必须先注册百度账号。获取地图 API 密钥的。具体步骤如下：

（1）在浏览器的地址栏中输入密钥的申请地址 http://lbsyun.baidu.com/apiconsole/key，百度账号注册完成后，输入账号和密码进行百度账号的登录，如果您没有注册为百度开发者，还需要注册为百度开发者。这时，页面将自动跳转到百度地图开放平台开发者激活页面。

（2）百度地图开放平台开发者激活后，单击“申请密钥”按钮，将进入到如图 24.6 所示的创建应用页面。

图 24.6　创建应用页面

安全码的组成规则为：Android 签名证书的 SHA1 值 + “;” +packagename（即：数字签名 + 英文状态下的分号 + 包名）。在安全码的组成规则中，主要有以下两部分内容需要获取。

◆ 获取 Android 签名证书的 SHA1 值

Android 签名证书的 SHA1 值可以在 Android Studio 的 signingReport 中获取，具体操作方法如图 24.7 所示。

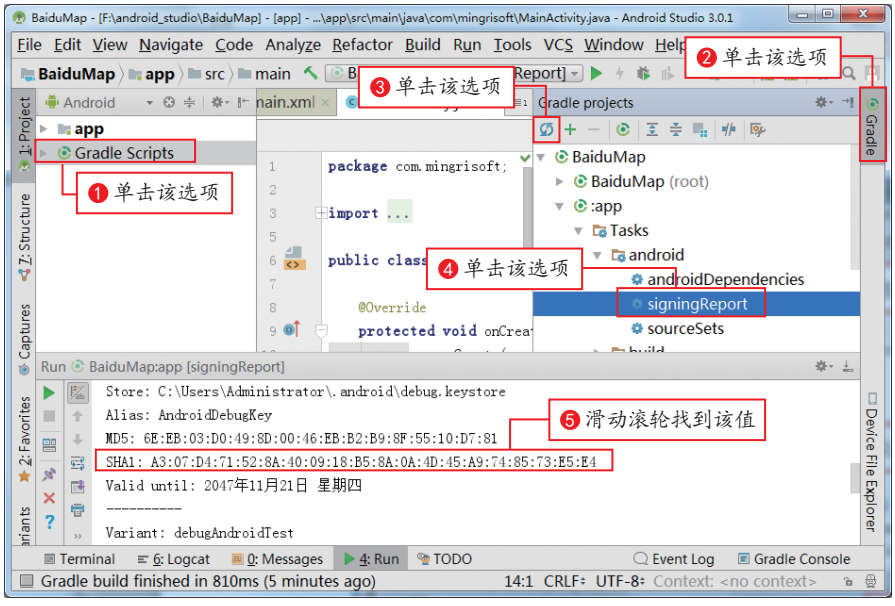


图 24.7　获取 Android 签名证书的 SHA1 值

◆ 获取包名

包名是在 AndroidManifest.xml 中，通过 package 属性定义的名称，如图 24.8 所示。

```
<?xml version="1.0" encoding="utf
<manifest xmlns:android="http://s...android.com/apk/res/android"
    package="com.mingrisoft" >
```

包名

图 24.8　定义的包名

在图 24.6 所示的页面中输入“发布版 SHA1”的值和包名后会自动生成安全码，如图 24.9 所示。

图 24.9　自动生成安全码

（3）单击“提交”按钮，返回到如图 24.10 所示的应用列表页面，在该页面中将显示刚刚创建的应用。

图 24.10　显示申请到的密钥

24.2.2 下载 SDK 开发包

要开发百度地图应用，需要下载百度地图 SDK 开发包，该包可以到百度地图 API 网站下载，具体下载步骤如下：

（1）在浏览器的地址栏中输入网址 http://lbsyun.baidu.com，进入到百度地图 API 首页，将鼠标移动到“开发文档”超链接上将显示对应子菜单，单击“Android 地图 SDK”超链接进入到“Android 地图 SDK”页面，单击该页面左侧的“产品下载”超链接进入到如图 24.11 所示的相关下载页面。

图 24.11　产品下载页面

（2）单击“自定义下载”按钮，进入到如图 24.12 所示的页面，在该页面中根据自己的项目需要勾选相应的功能。

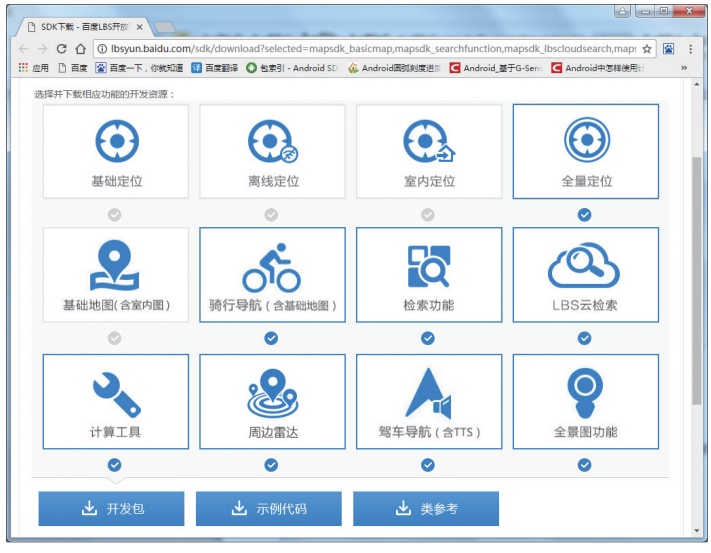


图 24.12　选择要下载的资源

（3）选中如图 24.12 所示的资源，单击“开发包”按钮开始下载。下载完成后，将得到一个名称为 BaiduLBS_AndroidSDK_Lib.zip 的文件。该文件就是百度地图 SDK 开发包。

24.2.3 创建使用百度地图 API 的项目

例 24.4 应用百度地图 API 实现显示百度地图

在 Android Studio 中创建 Module，名称为“Baidu Maps”。实现本实例的具体步骤如下：

（1）将项目结构类型切换为 Project，再解压缩下载的 BaiduLBS_AndroidSDK_Lib.zip 文件，将其中的 libs 目录下 3 个 Jar 文件复制到该 Module 的 libs 文件夹中，如图 24.13 所示。

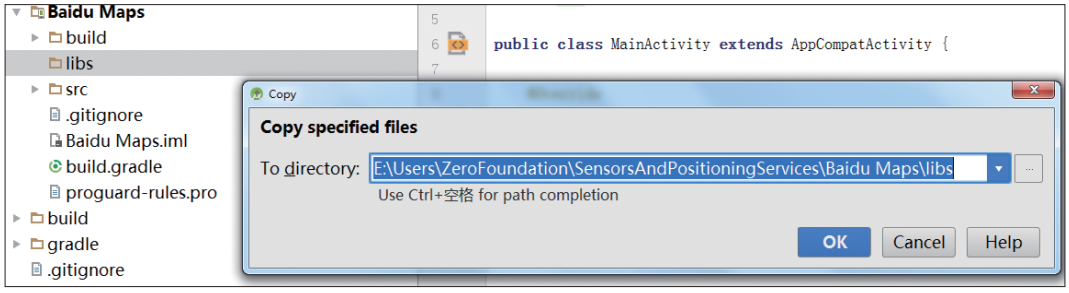


图 24.13 复制 Jar 文件

（2）同时选中 BaiduLBS_Android.jar、httpmime-4.1.2.jar 和 IndoorscapeAlbumPlugin.jar 文件，单击右键，在弹出的快捷菜单中选择“Add As Library…”菜单项，添加百度类库，如图 24.14 所示。

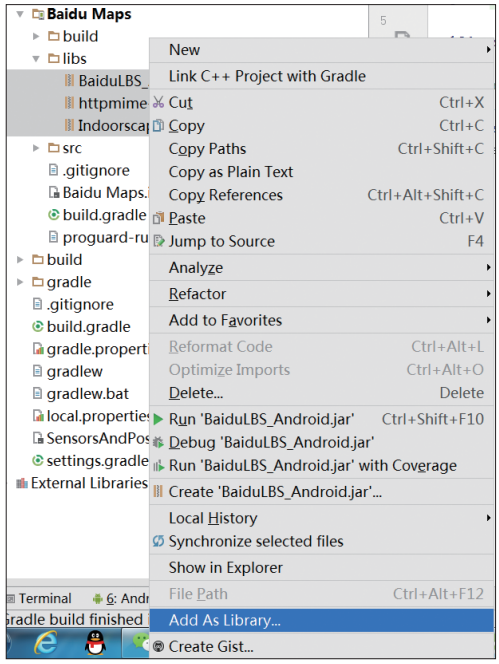


图 24.14 选择 Add As Library 菜单项

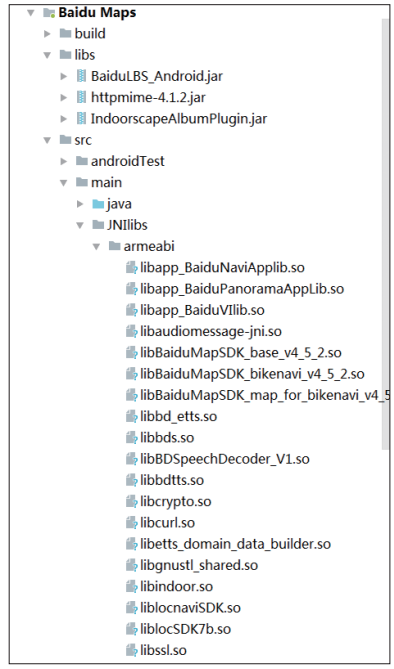


图 24.15 将 armeabi 整个目录复制到 JNIlibs 目录中

（3）在 src/main 目录中新建目录 JNIlibs，并将整个 armeabi 目录复制到 JNIlibs 目录中。复制后的效果如图 24.15 所示。

（4）在 AndroidManifest.xml 文件的 <application> 标签中添加子标签 <meta-data> 用于指定开发密钥，格式如图 24.16 所示。

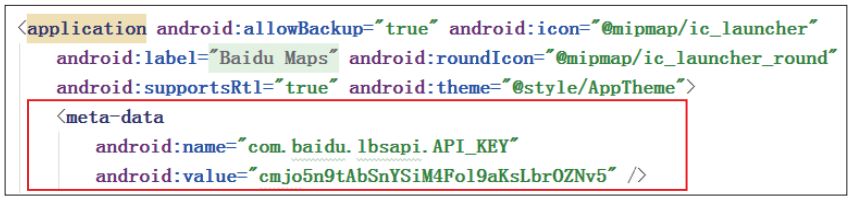


图 24.16　指定开发密钥

例如，申请的密钥为 cmjo5n9tAbSnYSiM4Fol9aKsLbrOZNv5，可以使用下面的代码：

```
<meta-data
    android:name="com.baidu.lbsapi.API_KEY"
    android:value="cmjo5n9tAbSnYSiM4Fol9aKsLbrOZNv5" />
```

（5）在 AndroidManifest.xml 文件的 <manifest> 标签中添加子标签 <uses-permission> 允许所需权限。通常情况下，使用百度 API 需要允许以下权限。

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="com.android.launcher.permission.READ_SETTINGS" />
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
```

（6）在布局文件 activity_main.xml 中，首先将默认添加的布局管理器修改为相对布局管理器，然后添加地图组件，并删除 TextView 组件，关键代码如下：

```
<com.baidu.mapapi.map.MapView
    android:id="@+id/bmapview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true" />
```

（7）打开主活动 MainActivity，在该文件中声明所需成员变量，关键代码如下：

```
private MapView mMapView;  //百度地图组件
```

（8）在 MainActivity 的 onCreate() 方法中，初始化 SDK 引用的 Context 全局变量，然后获取布局文件中添加的百度地图组件，以及百度地图对象，关键代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    SDKInitializer.initialize(getApplicationContext());     //初始化地图SDK
    setContentView(R.layout.activity_main);
```

```
06      mMapView = (MapView) findViewById(R.id.bmapview);      //获取地图组件
07  }
```

注意 在initialize()方法中必须传入ApplicationContext，如果传入this，或者MainActivity.this都会在运行时报错，所以建议把该方法放到Application的初始化方法中。

（9）重写 activity 的生命周期的几个方法来管理地图的生命周期。关键代码如下：

```
01  //实现地图生命周期管理
02  @Override
03  protected void onResume() {
04      super.onResume();
05      mMapView.onResume();
06  }
07
08  @Override
09  protected void onPause() {
10      super.onPause();
11      mMapView.onPause();
12  }
13
14  @Override
15  protected void onDestroy() {
16      mMapView.onDestroy();
17      mMapView = null;
18      super.onDestroy();
19  }
```

（10）运行本实例，将显示如图 24.17 所示的界面效果。

图 24.17　显示百度地图

说明 在运行使用百度地图API的程序时，需要连接互联网。

24.2.4 定位到我的位置

在实现定位到我的位置时，首先要开启定位图层，可以使用 com.baidu.mapapi.map.BaiduMap 对象的 setMyLocationEnabled() 方法实现。该方法的语法格式如下：

```
Public final void set MyLocationEnabled(boolean enabled)
```

其中，enabled 参数用于指定是否允许定位图层，值为 true 时表示允许，否则为不允许。开启定位图层的代码如下：

```
mBaiduMap.setMyLocationEnabled(true);
```

创建 MyLocationData 对象，用于构造定位数据，包括 GPS 定位时方向角度、纬度坐标、经度坐标、定位精度和时速等。例如，构造定位数据，指定 GPS 定位时方向角度为 100、坐标位置为当前位置，可以使用下面的代码。

```
01  MyLocationData locData = new MyLocationData.Builder()
02          .accuracy(location.getAccuracy())    //设置精度
03          .direction(100)                      //此处设置开发者获取到的方向信息，顺时针0-360
04          .latitude(location.getLatitude())    //设置纬度坐标
05          .longitude(location.getLongitude()) //设置经度坐标
06          .build();
```

说明 此处代码将使用Android原有定位方法。

设置定位数据，并配置定位图层的一些信息。代码如下：

```
01  //设置定位数据
02  mBaiduMap.setMyLocationData(locData);
03  //设置自定义定位图标
04  BitmapDescriptor mCurrentMarker = BitmapDescriptorFactory
05          .fromResource(R.drawable.icon_geo);
06  mCurrentMode = MyLocationConfiguration.LocationMode.NORMAL;//设置定位模式
07  //位置构造方式，将定位模式，定义图标添加到该位置
08  MyLocationConfiguration config = new
09          MyLocationConfiguration(mCurrentMode, true, mCurrentMarker);
10  mBaiduMap.setMyLocationConfiguration(config);              //地图显示定位图标
```

最后，在不需要定位图层时关闭定位图层。代码如下：

```
mBaiduMap.setMyLocationEnabled(false);                      //关闭定位图层
```

下面通过一个实例来演示如何在百度地图上定位我的位置。

例 24.5 百度地图定位我的位置

说明 该实例在本章实例24.4基础上进行修改。

在 Android Studio 中创建 Module，名称为“My Location”。实现本实例的具体步骤如下：

（1）打开主活动 MainActivity，在该文件中声明所需成员变量，关键代码如下：

```
01 private MapView mMapView;              //百度地图组件
02 private BaiduMap mBaiduMap;            //定义百度地图对象
03 private boolean isFirstLoc = true;     //定义第一次启动
04 private MyLocationConfiguration.LocationMode mCurrentMode;  //定义当前定位模式
```

（2）在 MainActivity 的 onCreate() 方法中，首先获取百度地图对象，然后设置地图缩放级别，再获取系统的 LocationManager 对象，关键代码如下：

```
01 mBaiduMap = mMapView.getMap();                                    //获取百度地图对象
02 mBaiduMap.setMapStatus(MapStatusUpdateFactory.newMapStatus
03         (new MapStatus.Builder().zoom(13).build()));              //设置缩放级别
04 //获取系统的LocationManager对象
05 LocationManager locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
```

（3）在 onCreate() 方法中，通过 locationManager.requestLocationUpdates() 方法与位置监听器来设置每秒获取一次 location 信息，并将最新的定位信息传递给创建的 locationUpdates() 方法中。关键代码如下：

```
01 //设置每一秒获取一次location信息
02 locationManager.requestLocationUpdates(
03         LocationManager.GPS_PROVIDER,          //GPS定位提供者
04         1000,                                  //更新数据时间为1000毫秒
05         1,                                     //位置间隔为1米
06         //位置监听器
07         new LocationListener() {  //GPS定位信息发生改变时触发，用于更新位置信息
08             @Override
09             public void onLocationChanged(Location location) {
10                 //GPS信息发生改变时，更新位置
11                 locationUpdates(location);
12             }
13             @Override
14             //位置状态发生改变时触发
15             public void onStatusChanged(String provider, int status, Bundle extras) {
16             }
17             @Override
18             //定位提供者启动时触发
19             public void onProviderEnabled(String provider) {
20             }
21             @Override
22             //定位提供者关闭时触发
23             public void onProviderDisabled(String provider) {
24             }
25         });
26 //从GPS获取最新的定位信息
27 Location location = locationManager.getLastKnownLocation
28                                                 (LocationManager.GPS_PROVIDER);
29 locationUpdates(location);     //将最新的定位信息传递给创建的locationUpdates()方法中
```

说明 添加上的代码后，在代码的下方将出现红色浪线，此时按下〈Alt+Enter〉键，然后选择Add Permission ACCESS_FINE_LOCATION选项，添加该权限解决红色浪线问题。此时，Android Studio将自动添加以下权限检查的代码：

```
//添加权限检查
if (ActivityCompat.checkSelfPermission(this,
      Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
      && ActivityCompat.checkSelfPermission(this,
      Manifest.permission.ACCESS_COARSE_LOCATION)
      != PackageManager.PERMISSION_GRANTED) {
    return;
}
```

（4）在 MainActivity 中，创建 locationUpdates() 方法，用于获取当前的经纬度，并设置定位。当获取的信息不为空时，首先设置第一次定位的中心点为当前位置，然后构造和设置定位数据，最后在地图上显示定位图标。关键代码如下：

```
public void locationUpdates(Location location) {  //获取指定的查询信息
    //如果location不为空时
    if (location != null) {
        //获取用户当前经纬度
        LatLng ll = new LatLng(location.getLatitude(), location.getLongitude());
        Log.i("Location","纬度: "+
                        location.getLatitude()+" | 经度: "+location.getLongitude());
        if (isFirstLoc) {  //如果是第一次定位,就定位到以自己为中心
            MapStatusUpdate u = MapStatusUpdateFactory.newLatLng(ll);//更新坐标位置
            mBaiduMap.animateMapStatus(u);                          //设置地图位置
            isFirstLoc = false;                                     //取消第一次定位
        }
        //构造定位数据
        MyLocationData locData = new MyLocationData.Builder().
                accuracy(location.getAccuracy())                    //设置精度
                .direction(100)        //此处设置开发者获取到的方向信息，顺时针0-360
                .latitude(location.getLatitude())                   //设置纬度坐标
                .longitude(location.getLongitude())                 //设置经度坐标
                .build();

        //设置定位数据
        mBaiduMap.setMyLocationData(locData);
        //设置自定义定位图标
        BitmapDescriptor mCurrentMarker = BitmapDescriptorFactory
                .fromResource(R.drawable.icon_geo);
        mCurrentMode = MyLocationConfiguration.LocationMode.NORMAL;   //设置定位模式
        //位置构造方式，将定位模式，定义图标添加到该位置
        MyLocationConfiguration config = new
                MyLocationConfiguration(mCurrentMode, true, mCurrentMarker);
        mBaiduMap.setMyLocationConfiguration(config);               //地图显示定位图标
```

```
31      } else {
32          //否则输出空信息
33          Log.i("Location","没有获取到GPS信息");
34      }
35  }
```

（5）重写 Activity 的 onStart() 方法和 onStop() 方法，在 onStart() 方法中启动地图定位，在 onStop() 方法中停止地图定位。关键代码如下：

```
01  @Override
02  protected void onStart() {                          //启动地图定位
03      super.onStart();
04      mBaiduMap.setMyLocationEnabled(true);           //启动定位图层
05  }
06  @Override
07  protected void onStop() {                           //停止地图定位
08      super.onStop();
09      mBaiduMap.setMyLocationEnabled(false);          //关闭定位图层
10  }
```

（6）在 AndroidManifest.xml 文件中，添加以下代码设置访问 LocationProvider 的相关权限，具体代码如下：

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

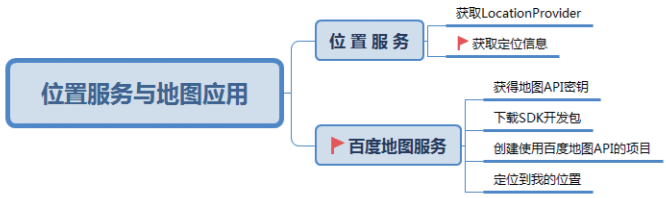
（7）运行本实例，将显示如图 24.18 所示。

图 24.18　在百度地图上显示我的位置

说明　运行本实例需要在手机中进入“设置”→“应用”→你的App应用名称→“权限”开启位置权限，即可进行定位。

24.3 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 25 章

网络编程及 Internet 应用

智能手机的一个主要功能就是可以访问互联网，大多数 App 都需要通过互联网执行某种网络通信，因此网络支持对于手机 App 来说是尤为重要的。Android 平台在网络编程和 Internet 应用上也是非常优秀的。本章将对 Android 中的网络编程和 Internet 应用的相关知识进行详细介绍。

25.1 通过 HTTP 访问网络

在 Android 中也可以使用 HTTP 协议访问网络。例如，在使用应用宝 App 下载游戏时（如图 25.1 所示），或者刷新朋友圈时（如图 25.2 所示），都需要通过 HTTP 协议访问网络。

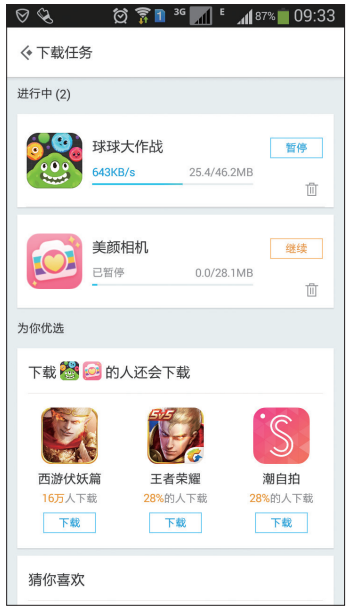


图 25.1　下载游戏

图 25.2　刷新朋友圈

在 Android 中提供了两个用于 HTTP 通信的 API，即 HttpURLConnection 和 Apache 的 HttpClient。由于 Android 6.0 版本已经基本将 HttpClient 从 SDK 中移除了。所以这里主要介绍 HttpURLConnection。

HttpURLConnection 类位于 java.net 包中，用于发送 HTTP 请求和获取 HTTP 响应。由于该类是抽象类，不能直接实例化对象，则需要使用 URL 的 openConnection() 方法来获得。例如，要创建一个“http://www.mingribook.com”网站对应的 HttpURLConnection 对象，可以使用下面的代码：

```
01  URL url = new URL("http://www.mingribook.com/");
02  HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
```

HttpURLConnection 是 URLConnection 的一个子类，它在 URLConnection 的基础上提供了如表 25.1 所示的方法，从而方便发送和响应 HTTP 请求。

表 25.1　HttpURLConnection 常用的方法

方　法	描　述
int getResponseCode()	获取服务器的响应代码
String getResponseMessage()	获取服务器的响应消息
String getRequestMethod()	获取发送请求的方法
void setRequestMethod(String method)	设置发送请求的方法

创建了 HttpURLConnection 对象后，就可以使用该对象发送 HTTP 请求了。

25.1.1 发送 GET 请求

使用 HttpURLConnection 对象发送请求时，默认发送的就是 GET 请求。因此，发送 GET 请求比较简单，只需要在指定连接地址时，先将要传递的参数通过“?参数名=参数值”的形式进行传递（多个参数间使用英文半角的“&”符号分隔。例如，要传递用户名和 E-mail 地址这两个参数，可以使用 ?user=wgh&email= wgh717@sohu.com 实现），然后获取流中的数据，并关闭连接即可。

说明 使用HTTP协议访问网络就是客户端与服务器的通信，所以运行本章实例不仅需要创建客户端App实例，还需要创建简单的后台服务器。

注意 （1）永远不要在主线程上执行网络调用。（2）在Service而不是Activity中执行网络操作。

下面通过一个实例来说明如何使用 HttpURLConnection 发送 GET 请求。

例 25.1 使用 GET 方式发表并显示微博信息

在 Android Studio 中创建一个 Module，名称为“GET Request”。实现本实例的具体步骤如下：

（1）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器并为其设置背景图片，删除默认添加的 TextView 组件，然后添加一个编辑框（用于输入微博内容）以及一个“发表”按钮，再添加一个滚动视图，在该视图中添加一个文本框，用于显示从服务器上读取的微博内容。

（2）打开主活动 MainActivity，该类继承 Activity，定义所需的成员变量，具体代码如下：

```
01  private EditText content;                    //定义一个输入文本内容的编辑框对象
02  private Handler handler;                     //定义一个android.os.Handler对象
03  private String result = "";                  //定义一个代表显示内容的字符串
```

（3）创建 base64() 方法，对传递的参数进行 Base64 编码，用于解决乱码问题。具体代码如下：

```
01 public String base64(String content) {                    //对字符串进行Base64编码
02     try {
03         //对字符串进行Base64编码
04         content = Base64.encodeToString(content.getBytes("utf-8"), Base64.DEFAULT);
05         content = URLEncoder.encode(content, "utf-8");//对字符串进行URL编码
06     } catch (UnsupportedEncodingException e) {
07         e.printStackTrace();
08     }
09     return content;
10 }
```

说明 要解决应用GET方法传递中文参数时产生乱码的问题，也可以使用Java提供的URLEncoder类来实现。

（4）创建 send() 方法，用于建立一个 HTTP 连接，并将输入的内容发送到 Web 服务器，再读取服务器的处理结果，具体代码如下：

```
01 public void send() {
02     String target = "";
03     target = "http://192.168.1.198:8080/example/get.jsp?content="
04             + base64(content.getText().toString().trim()); //要访问的URL地址
05     URL url;
06     try {
07         url = new URL(target);
08         HttpURLConnection urlConn = (HttpURLConnection) url
09                 .openConnection();                          //创建一个HTTP连接
10         InputStreamReader in = new InputStreamReader(
11                 urlConn.getInputStream());                  //获得读取的内容
12         BufferedReader buffer = new BufferedReader(in);     //获取输入流对象
13         String inputLine = null;
14         //通过循环逐行读取输入流中的内容
15         while ((inputLine = buffer.readLine()) != null) {
16             result += inputLine + "\n";
17         }
18         in.close();                                         //关闭字符输入流对象
19         urlConn.disconnect();                               //断开连接
20     } catch (MalformedURLException e) {
21         e.printStackTrace();
22     } catch (IOException e) {
23         e.printStackTrace();
24     }
25 }
```

说明 根据当前计算机的IP和Tomcat服务器的端口号设置要访问的URL地址。上面代码中的192.168.1.198是当前计算机的IP地址，8080是Tomcat服务器的端口号。

（5）在 onCreate() 方法中，首先在发表按钮的单击事件中，判断输入内容是否为空，然后创

建 Handler 对象并重写 handleMessage() 方法，用于更新 UI 界面，最后创建新的线程，用于从服务器中获取相关数据，具体代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //获取输入文本内容的EditText组件
    content = (EditText) findViewById(R.id.content);
    //获取显示结果的TextView组件
    final TextView resultTV = (TextView) findViewById(R.id.result);
    Button button = (Button) findViewById(R.id.button);    //获取"发表"按钮组件
    //单击"发表"按钮，实现读取服务器微博信息
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //判断输入内容是否为空，为空时给出提示消息，否则访问服务器
            if ("".equals(content.getText().toString())) {
                Toast.makeText(MainActivity.this, "请输入要发表的内容！",
                        Toast.LENGTH_SHORT).show();       //显示消息提示
                return;
            }
            handler = new Handler() {                //将服务器中的数据，显示在UI界面中
                @Override
                public void handleMessage(Message msg) {
                    super.handleMessage(msg);
                }
            };
            new Thread(new Runnable() { //创建一个新线程，用于发送并读取微博信息
                public void run() {
                    send();              //调用send()方法，用于发送文本内容到Web服务器
                    Message m = handler.obtainMessage();  //获取一个Message
                    handler.sendMessage(m);               //发送消息
                }
            }).start();                                   //开启线程
        }
    });
}
```

（6）重写 Handler 对象中的 handleMessage() 方法，在该方法中实现将服务器中的数据，显示在 UI 界面中，关键代码如下：

```
if (result != null) {          //如果服务器返回结果不为空
    resultTV.setText(result);  //显示获得的结果
    content.setText("");       //清空文本框
}
```

（7）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

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

（8）创建 Web 应用，用于接收 Android 客户端发送的请求，并做出响应。这里在 Tomcat 安装路径下的 webapps 目录中创建一个名称为 example 的文件夹，再在该文件夹中创建一个名称为 get.jsp 的文件，用于获取参数 content 指定的微博信息，并输出转码后的 content 变量的值，具体代码如下：

```
<%@page contentType="text/html; charset=utf-8" language="java" import="sun.misc.BASE64Decoder"%>
<%
String content=request.getParameter("content");                    //获取输入的微博信息
 if(content!=null){
    BASE64Decoder decoder=new BASE64Decoder();
    content=new String(decoder.decodeBuffer(content),"utf-8");     //进行base64解码
String date=new java.util.Date().toLocaleString();                 //获取系统时间
%>
<%="[马 云]于 "+date+" 发表一条微博，内容如下："%>
<%=content%>
<% }%>
```

说明 可以将云盘需要部署到Tomcat下的文件\example文件夹放到Tomcat安装路径下的webapps目录中，并启动Tomcat服务器，然后运行本实例。

（9）运行本实例，将显示如图 25.3 所示。

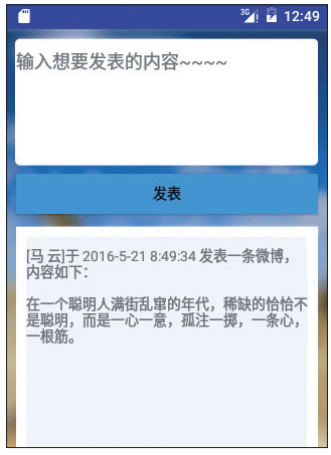


图 25.3 使用 GET 方式发表并显示微博信息

25.1.2 发送 POST 请求

在 Android 中，使用 HttpURLConnection 类发送请求时，默认采用的是 GET 请求，如果要发送 POST 请求，需要通过其 setRequestMethod() 方法进行指定。例如，创建一个 HTTP 连接，并为该连接指定请求的发送方式为 POST，可以使用下面的代码：

```
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();//创建一个HTTP连接
urlConn.setRequestMethod("POST");                                  //指定请求方式为POST
```

发送 POST 请求要比发送 GET 请求复杂一些，它需要通过 HttpURLConnection 类及其父类 URLConnection 提供的方法设置相关内容，常用的方法如表 25.2 所示。

表 25.2　发送 POST 请求时常用的方法

方　法	描　述
setDoInput(boolean newValue)	用于设置是否向连接中写入数据，如果参数值为 true，表示写入数据；否则不写入数据
setDoOutput(boolean newValue)	用于设置是否从连接中读取数据，如果参数值为 true，表示读取数据；否则不读取数据
setUseCaches(boolean newValue)	用于设置是否缓存数据，如果参数值为 true，表示缓存数据；否则表示禁用缓存
setInstanceFollowRedirects(boolean followRedirects)	用于设置是否应该自动执行 HTTP 重定向，参数值为 true 时，表示自动执行；否则不自动执行
setRequestProperty(String field, String newValue)	用于设置一般请求属性，例如，要设置内容类型为表单数据，可以进行以下设置 setRequestProperty(“Content-Type”,” application/x-www-form-urlencoded”)

下面通过一个具体的实例来介绍如何使用 HttpURLConnection 类发送 POST 请求。

例 25.2　使用 POST 方式登录 QQ

在 Android Studio 中创建 Module，名称为“POST Request”。实现本实例的具体步骤如下：

（1）打开主活动 MainActivity，该类继承 Activity，定义所需的成员变量，具体代码如下：

```
01  private EditText edit_Username;          //定义一个输入用户名的编辑框组件
02  private EditText edit_Password;          //定义一个输入密码的编辑框组件
03  private Handler handler;                 //定义一个android.os.Handler对象
04  private String result = "";              //定义一个代表显示内容的字符串
```

（2）创建 send() 方法，用于建立一个 HTTP 连接，并将输入的内容发送到 Web 服务器，再读取服务器的处理结果，具体代码如下：

```
01  public void send() {
02      String target = "http://192.168.1.198:8080/example/post.jsp";//要提交的服务器地址
03      URL url;
04      try {
05          url = new URL(target);                                    //创建URL对象
06          //创建一个HTTP连接
07          HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
08          urlConn.setRequestMethod("POST");                         //指定使用POST请求方式
09          urlConn.setDoInput(true);                                 //向连接中写入数据
10          urlConn.setDoOutput(true);                                //从连接中读取数据
11          urlConn.setUseCaches(false);                              //禁止缓存
12          urlConn.setInstanceFollowRedirects(true);                 //自动执行HTTP重定向
13          urlConn.setRequestProperty("Content-Type",
14                  "application/x-www-form-urlencoded");             //设置内容类型
```

```
        DataOutputStream out = new DataOutputStream(
                urlConn.getOutputStream());                          //获取输出流
        //连接要提交的数据
        String param = "username="
                + URLEncoder.encode(edit_Username.getText().toString(), "utf-8")
                + "&password="
                + URLEncoder.encode(edit_Password.getText().toString(), "utf-8");
        out.writeBytes(param);          //将要传递的数据写入数据输出流
        out.flush();                    //输出缓存
        out.close();                    //关闭数据输出流
        if (urlConn.getResponseCode() == HttpURLConnection.HTTP_OK) {//判断是否响应成功
            InputStreamReader in = new InputStreamReader(
                    urlConn.getInputStream());                       //获得读取的内容
            BufferedReader buffer = new BufferedReader(in);          //获取输入流对象
            String inputLine = null;
            //通过循环逐行读取输入流中的内容
            while ((inputLine = buffer.readLine()) != null) {
                result += inputLine;
            }
            in.close();                                              //关闭字符输入流
        }
        urlConn.disconnect();                                        //断开连接
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（3）在 onCreate() 方法中，首先在登录按钮的单击事件中，判断账号和密码是否为空，然后创建 Handler 对象并重写 handleMessage() 方法，用于更新 UI 界面，最后创建新的线程，用于从服务器中获取相关数据，关键代码如下：

```
edit_Username = (EditText) findViewById(R.id.username); //获取用于输入账号的编辑框组件
edit_Password = (EditText) findViewById(R.id.password); //获取用于输入密码的编辑框组件
ImageButton btn_Login = (ImageButton) findViewById(R.id.login);//获取用于登录的按钮控件
//单击"登录"按钮，发送信息与服务器交互
btn_Login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //当用户名、密码为空时给出相应提示
        if ("".equals(edit_Username.getText().toString())
                || "".equals(edit_Password.getText().toString())) {
            Toast.makeText(MainActivity.this, "请填写账号或密码！",
                    Toast.LENGTH_SHORT).show();
            return;
        }
```

```
        handler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
            }
        };
        new Thread(new Runnable() {//创建一个新线程，用于从网络上获取数据
            public void run() {
                send();             //调用send()方法，用于将账号和密码发送到Web服务器
                Message m = handler.obtainMessage();     //获取一个Message对象
                handler.sendMessage(m);                  //发送消息
            }
        }).start();                                      //开启线程
    }
});
```

（4）重写 Handler 对象中的 handleMessage() 方法，在该方法中实现通过服务器中返回的数据判断是否显示登录后界面，关键代码如下：

```
//如果服务器返回值为"ok"，则表示账号和密码输入正确
if ("ok".equals(result)) {
    //跳转登录后界面
    Intent in = new Intent(MainActivity.this, MessageActivity.class);
    startActivity(in);
}else {
    //账号、密码错误的提示信息
    Toast.makeText(MainActivity.this, "请填写正确的账号和密码！",
            Toast.LENGTH_SHORT).show();
}
```

（5）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

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

（6）创建 Web 应用，用于接收 Android 客户端发送的请求，并做出响应。这里编写一个名称为 post.jsp 的文件。在该文件中首先获取客户端填写的账号和密码，然后判断账号和密码是否正确，如果账号和密码正确，则向客户端传递通过指令“ok”。具体代码如下：

```
<%@ page contentType="text/html; charset=utf-8" language="java" %>
<%String password=request.getParameter("password");           //获取输入的密码
String username=request.getParameter("username");             //获取输入的用户名
if(password!=null && username!=null){
username=new String(username.getBytes("iso-8859-1"),"utf-8");  //对用户名进行转码
password=new String(password.getBytes("iso-8859-1"),"utf-8");  //对密码进行转码
if("mr".equals(username)&&"mrsoft".equals(password)){
%>
```

```
09  <%="ok"%>
10  <%}%>
11  <%}%>
```

说明 将post.jsp文件放到Tomcat安装路径下的“webapps\example”目录中，并启动Tomcat服务器，然后运行本实例。

（7）运行本实例，在显示的界面中输入账号“mr”与密码“mrsoft”，如图 25.4 所示，单击“登录”按钮，通过服务器判断账号和密码正确后显示登录后界面，如图 25.5 所示。

图 25.4 登录界面

图 25.5 登录后的界面

25.2 通过 OkHttp3 访问网络

25.2.1 OkHttp3 简介

除了可以使用 HttpURLConnection 进行网络访问以外，还可以使用一种更加方便、简单的方式来进行网络的访问，那就是 OkHttp3，它是由 Square 公司开发的开源网络通讯库。OkHttp3 的基本特点有：

- 支持 HTTP 协议 2.0，并允许连接到同一个主机地址的所有请求共享套接字。
- 如果 HTTP 协议 2.0 不可用，连接池也可以减少请求延迟。
- 支持 GZIP, 可以压缩下载体积。
- 响应缓存可以完全避免重复请求的网络。

在使用 OkHttp3 开源网络通讯库之前，需要在 gradle 文件的 dependencies 节点中添加依赖库代码。具体代码如下：

```
compile 'com.squareup.okhttp3:okhttp:插入新的版本'
```

在 AndroidManifest.xml 文件中添加网络权限的代码，具体代码如下：

```
01  <!--网络权限-->
02  <uses-permission android:name="android.permission.INTERNET"/>
```

25.2.2 OkHttp3 的基本用法

OkHttp3 的请求方式有以下两种：

◆ 使用 GET 方式进行网络请求可以使用以下代码：

```
//创建OkHttpClient对象
OkHttpClient okHttpClient = new OkHttpClient();
//创建请求对象
Request request = new Request.Builder()
        .url("请求地址")
        .build();
try {
    //创建请求响应
    Response response = okHttpClient.newCall(request).execute();
    response.body().string();     //获取响应资源
} catch (IOException e) {
    e.printStackTrace();
}
```

◆ 使用 POST 方式进行网络请求可以使用以下代码：

```
//创建OkHttpClient 对象
OkHttpClient okHttpClient = new OkHttpClient();
//Form表单格式的参数传递
FormBody formBody = new FormBody.Builder()
        .add("参数名","参数值").build();
//创建请求对象
Request request = new Request.Builder()
        .url("请求地址")
        .post(formBody)
        .build();
try {
    //创建请求响应
    Response response = okHttpClient.newCall(request).execute();
    response.body().string();     //获取响应资源
} catch (IOException e) {
    e.printStackTrace();
}
```

POST 网络请求与 GET 网络请求类似，只是多了 1 个 FormBody 用于添加请求的参数，然后传递给 Request。

下面通过一个具体的实例演示 OkHttp3 的具体应用。

例 25.3 使用 OkHttp3 下载网络图片

在 Android Studio 中创建 Module，名称为“OkHttp3”，实现本实例的具体步骤如下：

（1）打开 build.gradle（Module：OkHttp3）文件，在 dependencies 节点中添加 OkHttp3 的依赖

库代码，然后在 AndroidManifest.xml 文件中添加网络权限的代码。

（2）修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并将 TextView 组件删除，然后添加 1 个用于显示下载图片的 ImageView 组件，再添加 1 个 Button 组件用于下载按钮。具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.MainActivity">
    <!--显示下载的图片-->
    <ImageView
        android:id="@+id/image1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:src="@mipmap/bbs_new" />
    <!--下载按钮-->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="down"
        android:text="下载图片" />
</RelativeLayout>
```

（3）打开主活动 MainActivity.java 文件，定义所需要的全局变量并创建 1 个 Handler 对象用于显示下载的图片，然后在 onCreate() 方法中创建进度对话框用于显示下载图片时的等待进度，然后获取用于显示图片的 ImageView 组件。具体代码如下：

```
public class MainActivity extends AppCompatActivity {
    private ProgressDialog dialog;          //下载等待对话框
    private ImageView image;                //显示下载图片的组件
    private Handler handler=new Handler(){
        @Override
        public void handleMessage(Message msg) {
            dialog.dismiss();               //关闭弹窗
            //显示下载的图片
            image.setImageBitmap((Bitmap) msg.obj);
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        dialog = new ProgressDialog(this);              //创建进度对话框
        dialog.setTitle("提示");                         //设置弹窗的标题
        dialog.setMessage("下载图片中,请稍等！");          //设置对话框提示内容
        image = (ImageView) findViewById(R.id.image1); //获取显示图片的组件
    }
}
```

（4）创建 1 个名称为 down 的方法，该方法是下载按钮的单击事件，在该方法中首先创建 OkHttpClient 对象，然后创建网络请求，再创建请求呼叫，最后在 onResponse() 方法中将响应数据转换为 Bitmap 位图数据并通过 Handler 发送该数据。具体代码如下：

```
public void down(View view) {
    dialog.show();//显示进度对话框
    //创建OkHttpClient对象
    OkHttpClient okHttpClient = new OkHttpClient();
    //创建网络请求
    Request request = new Request.Builder()
            .get()
            .url("http://test.mingrisoft.com//Public/" +
                    "uploads/book_pic/590af7401f562.png")
            .build();
    //创建请求呼叫
    Call call = okHttpClient.newCall(request);
    call.enqueue(new Callback() {
        //无法执行请求时调用
        @Override
        public void onFailure(Call call, IOException e) {
        }
        //返回HTTP响应时调用
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            //将响应数据转换为输入流数据
            InputStream inputStream = response.body().byteStream();
            //将输入流数据转换为Bitmap位图数据
            Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
            //创建Message消息并发送
            Message msg = Message.obtain();
            msg.obj = bitmap;
            handler.sendMessage(msg);
        }
    });
}
```

（5）运行本实例，将显示如图 25.6 所示。单击下载图片按钮将显示如图 25.7 所示的下载等待对话框。下载完成后将自动显示如图 25.8 所示下载完成的网络图片。

图 25.6 显示默认图片

图 25.7 显示下载等待对话框

图 25.8 显示下载完成的网络图片

25.3 解析 JSON 格式数据

25.3.1 JSON 简介

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式，语法简洁，不仅易于阅读和编写，而且也易于机器的解析和生成。

JSON 通常由两种数据结构组成，一种是对象（“名称 / 值”形式的映射），另一种是数组（值的有序列表）。JSON 没有变量或其他控制，只用于数据传输。

（1）对象

在 JSON 中，可以使用下面的语法格式来定义对象。

```
{"属性1":属性值1,"属性2":属性值2……"属性n":属性值n}
```

◆ 属性 1~ 属性 n：用于指定对象拥有的属性名。

◆ 属性值 1~ 属性值 n：用于指定各属性对应的属性值，其值可以是字符串、数字、布尔值（true/false）、null、对象和数组。

例如，定义一个保存名人信息的对象，可以使用下面的代码：

```
01 {
02     "name":"扎克伯格",
03         "address":"United States New York",
04         "wellknownsaying":"当你有使命，它会让你更专注"
05 }
```

（2）数组

在 JSON 中，可以使用下面的语法格式来定义对象。

```
{"数组名":[
```

```
            对象1,对象2……,对象n
]}
```

◆ 数组名：用于指定当前数组名。

◆ 对象 1~ 对象 n：用于指定各数组元素，它的值为合法的 JSON 对象。

例如，定义一个保存名人信息的数组，可以使用下面的代码：

```
{"famousPerson":[
  {"name":"扎克伯格","address":" 美国 "," wellknownsaying ":"当你有使命，它会让你更专注"},
  {"name":"马云","address":"中国"," wellknownsaying ":"心中无敌者，无敌于天下"}
]}
```

25.3.2 解析 JSON 数据

在 Android 的官网中提供了解析 JSON 数据的 JSONObject 和 JSONArray 对象。其中，JSONObject 用于解析 JSON 对象；JSONArray 用于解析 JSON 数组。下面将通过一个具体的实例说明如何解析 JSON 数据。

例 25.4 获取 JSON 数据，显示计步器的个人信息

在 Android Studio 中创建 Module，名称为“Analysis Of JSON Data”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为垂直线性布局管理器，然后添加 8 个 TextView 组件，用于显示计步器的 8 个信息值。

（2）打开主活动 MainActivity，该类继承 Activity，定义所需的成员变量，关键代码如下：

```
private Handler handler;          //定义一个android.os.Handler对象
private String result = "";       //定义一个代表显示内容的字符串
```

（3）创建 send() 方法，实现发送请求并获取 JSON 数据，关键代码如下：

```
public void send() {                    //创建send()方法，实现发送请求并获取JSON数据
    //要发送请求的服务器地址
    String target = "http://192.168.1.198:8080/example/index.json";
    URL url;
    try {
        url = new URL(target);                              //创建URL对象
        //创建一个HTTP连接
        HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
        urlConn.setRequestMethod("POST");                   //指定使用POST请求方式
        urlConn.setDoOutput(true);                          //从连接中读取数据
        urlConn.setUseCaches(false);                        //禁止缓存
        urlConn.setInstanceFollowRedirects(true);           //自动执行HTTP重定向
        InputStreamReader in = new InputStreamReader(
                urlConn.getInputStream());                  //获得读取的内容
        BufferedReader buffer = new BufferedReader(in); //获取输入流对象
        String inputLine = null;
```

```
        //通过循环逐行读取输入流中的内容
        while ((inputLine = buffer.readLine()) != null) {
            result += inputLine;
        }
        in.close();                                          //关闭输入流
        urlConn.disconnect();                                //断开连接
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（4）在 onCreate() 方法中，首先创建 Handler 对象并重写 handleMessage() 方法，用于更新 UI 界面，然后创建新的线程，用于从服务器中获取 JSON 数据，关键代码如下：

```
final TextView step = (TextView) findViewById(R.id.text1); //获取TextView显示单日步数
final TextView time = (TextView) findViewById(R.id.text2); //获取TextView显示单日时间
final TextView heat = (TextView) findViewById(R.id.text3); //获取TextView显示单日热量
final TextView km = (TextView) findViewById(R.id.text4);   //获取TextView显示单日公里数
final TextView step1 = (TextView) findViewById(R.id.text5);//获取TextView显示每周步数
final TextView time1 = (TextView) findViewById(R.id.text6);//获取TextView显示每周时间
final TextView heat1 = (TextView) findViewById(R.id.text7);//获取TextView显示每周热量
final TextView km1 = (TextView) findViewById(R.id.text8);  //获取TextView显示每周公里数
handler = new Handler() {                                  //解析返回的JSON串数据并显示
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
    }
};
new Thread(new Runnable() {                 //创建一个新线程，用于从服务器中获取JSON数据
    public void run() {
        send();                             //调用send()方法，用于发送请求并获取JSON数据
        Message m = handler.obtainMessage();            //获取一个Message对象
        handler.sendMessage(m);                         //发送消息
    }
}).start();                                             //开启线程
```

（5）重写 Handler 对象中的 handleMessage() 方法，在该方法中实现解析返回的 JSON 串数据并显示，关键代码如下：

```
//创建TextView二维数组
TextView[][] tv = {{step, time, heat, km}, {step1, time1, heat1, km1}};
try {
    JSONArray jsonArray = new JSONArray(result);   //将获取的数据保存在JSONArray数组中
    for (int i = 0; i < jsonArray.length(); i++) { //通过for循环遍历JSON数据
        JSONObject jsonObject = jsonArray.getJSONObject(i);  //解析JSON数据
```

```
            tv[i][0].setText(jsonObject.getString("step"));       //获取JSON中的步数值
            tv[i][1].setText(jsonObject.getString("time"));       //获取JSON中的时间值
            tv[i][2].setText(jsonObject.getString("heat"));       //获取JSON中的热量值
            tv[i][3].setText(jsonObject.getString("km"));         //获取JSON中的公里数
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

（6）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

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

（7）创建 Web 服务器，用于接收 Android 客户端发送的请求，并做出响应。这里编写一个名称为 index.json 的文件。在该文件中编写要返回的 JSON 数据，具体代码如下：

```
[{"step":"12,672","time":"1h 58m","heat":"306","km":"8.3"},
{"step":"73,885","time":"11h 41m","heat":"1,771","km":"48.7"}]
```

说明 将index.json文件放到Tomcat安装路径下的webapps\example目录中，并启动Tomcat服务器，然后运行本实例。

（8）运行本实例，将显示如图 25.9 所示。

图 25.9 显示计步器的个人信息

25.3.3 使用 GSON 解析数据

GSON 是 Google 提供的一个 Java 类库，用于将 Java 对象转换为 JSON 数据，也可以将一个 JSON 字符串转换为对应的 Java 对象。不过 GSON 并没有被 Android 官方添加在 API 当中，所以在使用 GSON 类库时需要在 gradle 文件的 dependencies 节点中添加 GSON 库的依赖代码。具体代码如下：

```
compile'com.google.code.gson:gson:插入新的版本'
```

也可以在 GSON 的官方网站中下载 GSON 的 jar 包，来使用 GSON 类库。

GSON 类库在解析 JSON 数据时更加的方便、简单，它可以将一段 JSON 格式的字符串自动映射成一个对象，所以再也不需要手动去编写代码进行解析了。例如，一段 JSON 格式的数据如下所示：

```
{
        "id":101,
        "name":"Summer",
        "age":28,
        "height":1.75
    }
```

解析单个对象首先需要创建 1 个 Information 类，在该类中定义 ID、名字、年龄、身高，并且为这些属性设置 get() 方法。具体代码如下：

```
public class Information {
    public int id;                  //定义ID
    public String name;             //定义名字
    public int age;                 //定义年龄
    public float height;            //定义身高
    public int getId() {            //获取ID
        return id;
    }
    public String getName() {       //获取姓名
        return name;
    }
    public int getAge() {           //获取年龄
        return age;
    }
    public float getHeight() {      //获取身高
        return height;
    }
}
```

然后创建 Gson 对象，再通过 fromJson() 方法进行 json 数据的解析，最后通过 Information 类中的 getId() 方法获取 json 数据中的 ID 值。具体代码如下：

```
Gson gson=new Gson();                                          //创建Gson对象
Information info=gson.fromJson(json数据,Information.class);    //解析json数据
info.getId();                                                  //获取json数据中的ID
```

通常情况下 JSON 数据包含多个对象，如果解析一段 JSON 数组就需要使用 TypeToken 类来帮忙，将需要解析的数据类型传入到 fromJson() 方法中。

下面通过一个具体的实例演示 GSON 解析数据的应用。

例 25.5 将 json 序列变为 list 对象

在 Android Studio 中创建 Module，名称为“GSON”，实现本实例的具体步骤如下（本实例在 25.3.2 小节中的实例基础上进行修改）：

（1）打开 build.gradle（Module：GSON）文件，在 dependencies 节点中添加 GSON 与 OkHttp3 的依赖库代码。然后在 AndroidManifest.xml 文件中添加网络权限的代码。

（2）在“java/com.mingrisoft”包中创建 Information.java 类，在该类中定义步数、时间、热量、公里数，，并且为这些属性设置 get() 方法。具体代码如下：

```
01  public class Information {
02      public String step;             //定义步数
03      public String time;             //定义时间
04      public String heat;             //定义热量
05      public String km;               //定义公里数
06      public String getStep() {       //获取步数
07          return step;
08      }
09      public String getTime() {       //获取时间
10          return time;
11      }
12      public String getHeat() {       //获取热量
13          return heat;
14      }
15      public String getKm() {         //获取公里数
16          return km;
17      }
18  }
```

（3）打开主活动 MainActivity.java 文件，修改 send() 方法中的代码，在该方法中通过 OkHttp3 实现向服务器发送获取 JSON 信息的网络请求。关键代码如下：

```
01  public void send() {
02      //要发送请求的服务器地址
03      String target = "http://192.168.1.198:8080/example/index.json";
04      //创建OkHttpClient对象
05      OkHttpClient okHttpClient = new OkHttpClient();
06      //创建网络请求
07      Request request = new Request.Builder()
08              .get()
09              .url(target)
10              .build();
11      try {
12          //创建请求响应
13          Response response = okHttpClient.newCall(request).execute();
14          result = response.body().string();   //获取json数据
15      } catch (IOException e) {
16          e.printStackTrace();
17      }
18  }
```

（4）修改 handleMessage() 方法中的代码，在该方法中首先创建 1 个用于解析 JSON 数据的类型，然后通过 fromJson() 方法进行 JSON 数据的解析，最后将解析的数据显示在 TextView 文本框组件当中。关键代码如下：

```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        //创建TextView二维数组
        TextView[][] tv = {{step, time, heat, km}, {step1, time1, heat1, km1}};
        //创建解析JSON的类型
        Type listType = new TypeToken<ArrayList<Information>>(){}.getType();
        //解析JSON数据
        ArrayList<Information> foos = new Gson().fromJson(result,listType);
        for (int i = 0; i < foos.size(); i++) {          //通过for循环遍历JSON数据
            tv[i][0].setText(foos.get(i).getStep());     //获取JSON中的步数值
            tv[i][1].setText(foos.get(i).getTime());     //获取JSON中的时间值
            tv[i][2].setText(foos.get(i).getHeat());     //获取JSON中的热量值
            tv[i][3].setText(foos.get(i).getKm());       //获取JSON中的公里数
        }
        super.handleMessage(msg);
    }
};
```

（5）运行本实例，如图 25.10 所示，显示解析后的 JSON 数据。

图 25.10　显示解析后的 JSON 数据

说明　由于该实例是在25.3.2小节中的实例基础上进行修改，所以同样需要启动Tomcat服务器，然后运行本实例。

25.4 使用 WebView 显示网页

Android 提供了内置的浏览器，该浏览器使用了开源的 WebKit 引擎。WebKit 不仅能够搜索网址、查看电子邮件，而且能够播放视频节目。在 Android 中，要使用内置的浏览器，需要通过 WebView 组件来实现。通过 WebView 组件可以轻松实现显示网页功能。例如，通过 WebView 来实现上网功能，通过 QQ 浏览器显示明日学院主页，效果如图 25.11 所示。

图 25.11　通过 QQ 浏览器显示网页

25.4.1　使用 WebView 组件浏览网页

WebView 组件是专门用来浏览网页的，其使用方法与其他组件一样，既可以在 XML 布局文件中使用 <WebView> 标记添加，又可以在 Java 文件中通过 new 关键字创建。推荐采用 <WebView> 标记在 XML 布局文件中添加。在 XML 布局文件中添加一个 WebView 组件可以使用下面的代码：

```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

添加 WebView 组件后，就可以应用该组件提供的方法执行浏览器操作。WebView 组件提供的常用方法如表 25.3 所示。

表 25.3　WebView 组件提供的常用方法

方　法	描　述
loadUrl(String url)	用于加载指定 URL 对应的网页
loadData(String data, String mimeType, String encoding)	用于将指定的字符串数据加载到浏览器中
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl)	用于基于 URL 加载指定的数据
capturePicture()	用于创建当前屏幕的快照
goBack()	执行后退操作，相当于浏览器上的后退按钮的功能
goForward()	执行前进操作，相当于浏览器上的前进按钮的功能

续表

方　法	描　述
stopLoading()	用于停止加载当前页面
reload()	用于刷新当前页面

下面通过一个实例来说明如何使用 WebView 组件浏览网页。

例 25.6 使用 WebView 组件浏览网页

在 Android Studio 中创建 Module，名称为“WebView”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，将默认添加的布局管理器修改为相对布局管理器并删除默认添加的 TextView 组件，然后添加一个 WebView 组件，关键代码如下：

```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

（2）打开主活动 MainActivity，该类继承 Activity，在 onCreate() 方法中，首先获取布局管理器中添加的 WebView 组件，然后为 WebView 指定要加载网页的 URL 地址，最后设置加载内容自适应屏幕，具体代码如下：

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //设置全屏显示
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        //获取布局管理器中添加的WebView组件
        WebView webView = (WebView) findViewById(R.id.webView1);
        webView.loadUrl("http://www.mingrisoft.com/Bbs.html");//指定要加载的网页
        webView.getSettings().setUseWideViewPort(true);  //设置此属性，可任意比例缩放
        webView.getSettings().setLoadWithOverviewMode(true);//设置加载内容自适应屏幕
    }
}
```

（3）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

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

说明 如果想让WebView组件具有放大和缩小网页的功能，则要进行以下设置：

```
webView.getSettings().setSupportZoom(true);
webView.getSettings().setBuiltInZoomControls(true);
```

（4）运行本实例，将显示如图 25.12 所示。

图 25.12 使用 WebView 组件浏览网页

25.4.2 使用 WebView 加载 HTML 代码

在进行 Android 开发时，对于一些游戏的帮助信息，使用 HTML 代码进行显示比较实用，不仅可以让界面更加美观，而且可以让开发更加简单、快捷。WebView 组件提供了 loadData() 和 loadDataWithBaseURL() 方法来加载 HTML 代码。loadData() 方法一般很少使用，因为使用该方法加载带中文的 HTML 内容时会产生乱码，而使用 loadDataWithBaseURL() 方法就不会出现这种情况。loadDataWithBaseURL() 方法的基本语法格式如下：

```
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding,
            String historyUrl)
```

loadDataWithBaseURL() 方法各参数的说明如表 25.4 所示。

表 25.4 loadDataWithBaseURL() 方法的参数说明

参　数	描　述
baseUrl	用于指定当前页使用的基本 URL。如果为 null，则使用默认的 about:blank，即空白页
data	用于指定要显示的字符串数据
mimeType	用于指定要显示内容的 MIME 类型。如果为 null，则默认使用 text/html
encoding	用于指定数据的编码方式
historyUrl	用于指定当前页的历史 URL，也就是进入该页前显示页的 URL。如果为 null，则使用默认的 about:blank

下面通过一个具体的实例来说明如何使用 WebView 组件加载 HTML 代码。

例 25.7 实现使用 WebView 组件加载 HTML 游戏指南界面

在 Android Studio 中创建 Module，名称为“WebView And HTML”，实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后将 TextView 组件删除，再添加一个 Button 组件，用于单击后跳转到游戏指南界面。

（2）创建一个名称为 HelpActivity 的 Activity，修改布局文件 activity_help.xml，将默认添加的布局管理器修改为相对布局管理器，然后添加一个 WebView 组件，用于加载 HTML 代码编写的游戏指南界面。

（3）打开 HelpActivity 类，该类继承 Activity，在 onCreate() 方法中，首先获取布局管理器中添加的 WebView 组件，然后创建一个字符串构建器，将要显示的 HTML 内容放置在该构建器中，最后通过 loadDataWithBaseURL() 方法加载数据，具体代码如下：

```
01  public class HelpActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_help);
06          //设置全屏显示
07          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
09          //获取布局管理器中添加的WebView组件
10          WebView webview = (WebView) findViewById(R.id. webView1);
11          webview.setBackgroundResource(R.drawable. bg_help);//设置WebView背景图片
12          webview.setBackgroundColor(0);                    //设置WebView背景色为透明
13          //创建一个字符串构建器，将要显示的HTML内容放置在该构建器中
14          StringBuilder sb = new StringBuilder();
15          sb.append("<br>");
16          sb.append("<br>");
17          sb.append("<span style=font-size:20px>" +
18                  "<div>疯狂动物来找茬操作指南：</div></span>");
19          sb.append("<ul>");
20          sb.append("<span style=font-size:20px><li>一共三关。</li></span>");
21          sb.append("<br>");
22          sb.append("<span style=font-size:20px>" +
23                  "<li>找出两张图片的5处不同点。</li></span>");
24          sb.append("<br>");
25          sb.append("<span style=font-size:20px>" +
26                  "<li>剩余时间越长，分数越高。</li></span>");
27          sb.append("<br>");
28          sb.append("<span style=font-size:20px>" +
29                  "<li>每过完一个关卡将询问是否进入</li></span>");
30          sb.append("<span style=font-size:20px>下一关。</span>");
31          sb.append("</ul>");
32          //加载数据
33          webview.loadDataWithBaseURL(null, sb.toString(), "text/html", "utf-8", null);
34      }
35  }
```

（4）打开 MainActivity 类，该类继承 Activity，在 onCreate() 方法中，实现单击“游戏玩法”按钮后跳转到游戏操作指南页面，具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      public void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //设置全屏显示
07          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
09          Button btn= (Button) findViewById(R.id.btn_help);//获取布局文件中的游戏玩法按钮
10          //实现单击按钮跳转游戏指南页面
11          btn.setOnClickListener(new View.OnClickListener() {
12              @Override
13              public void onClick(View v) {
14                  //设置通过Intent跳转游戏指南页面
15                  Intent intent = new Intent(MainActivity.this, HelpActivity.class);
16                  startActivity(intent);
17              }
18          });
19      }
20  }
```

（5）在 AndroidManifest.xml 文件的 <activity> 标记中添加 screenOrientation 属性，分别设置 MainActivity 与 HelpActivity 横屏显示，关键代码如下：

```
android:screenOrientation="landscape"
```

（6）运行本实例，将显示如图 25.13 所示。单击“游戏玩法”按钮，将显示如图 25.14 所示的运行结果。

图 25.13　游戏初始界面

图 25.14　使用 WebView 加载 HTML 界面

25.4.3　让 WebView 支持 JavaScript

在默认的情况下，WebView 组件是不支持 JavaScript 的，但是在运行某些不得不使用 JavaScript 代码的网站时，需要让 WebView 支持 JavaScript。例如，在图 25.15 中显示的是明日图书网的登录页面，

如果在填写登录信息时只输入昵称，此时单击“登录”按钮，就会弹出如图 25.16 所示的提示框，该提示框就是网页中通过 JavaScript 代码实现的。

图 25.15 网站登录页面

图 25.16 弹出 JavaScript 提示框

实际上，让 WebView 组件支持 JavaScript 比较简单，只需要以下两个步骤就可以实现：

（1）使用 WebView 组件的 WebSettings 对象提供的 setJavaScriptEnabled() 方法让 JavaScript 可用。例如，存在一个名称为 webview 的 WebView 组件，要设置在该组件中允许使用 JavaScript，可以使用的代码如下：

```
webView.getSettings().setJavaScriptEnabled(true);          //设置JavaScript可用
```

（2）经过以上设置后，网页中的大部分 JavaScript 代码均可用。但是，对于通过 window.alert() 方法弹出的对话框并不可用。要想显示弹出的对话框，需要使用 WebView 组件的 setWebChromeClient() 方法来处理 JavaScript 的对话框，具体代码如下：

```
webView.setWebChromeClient(new WebChromeClient());        //设置处理JavaScript中的对话框
```

这样设置后，再使用 WebView 显示带弹出 JavaScript 对话框的网页时，网页中弹出的对话框将不会被屏蔽掉。下面通过一个具体的实例来说明如何让 WebView 支持 JavaScript。

例 25.8 WebView 加载 QQ 空间“写说说”界面

在 Android Studio 中创建 Module，名称为“WebView And JavaScript”。实现本实例的具体步骤如下：

（1）修改布局文件 activity_main.xml，首先将默认添加布局管理器修改为相对布局管理器，然后将 TextView 组件删除，再添加一个 WebView 组件，用于加载 JavaScript 页面。

（2）打开主活动 MainActivity，该类继承 Activity，在 onCreate() 方法中，首先获取布局管理器中添加的 WebView 组件，然后设置 JavaScript 可用，再设置处理 JavaScript 中的对话框，最后指定要加载的网页，具体代码如下：

```
01  public class MainActivity extends Activity {
02      @Override
03      protected void onCreate(Bundle savedInstanceState) {
04          super.onCreate(savedInstanceState);
05          setContentView(R.layout.activity_main);
06          //获取布局文件中的WebView组件
07          WebView webView = (WebView) findViewById(R.id.webView1);
08          //设置JavaScript可用
09          webView.getSettings().setJavaScriptEnabled(true);
10          //设置处理JavaScript中的对话框
11          webView.setWebChromeClient(new WebChromeClient());
12          //指定要加载的网页
13          webView.loadUrl("http://192.168.1.198:8080/example/javascript.jsp");
14      }
15  }
```

（3）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限，具体代码如下：

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

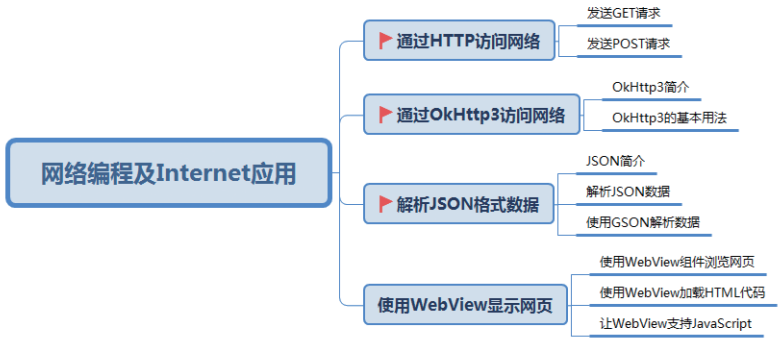
说明 将javascript.jsp文件放到Tomcat安装路径下的“webapps\example”目录中，并启动Tomcat服务器，然后运行本实例。

（4）运行本实例，如果未输入任何内容，直接单击“发表”按钮后将弹出一个提示对话框，如图 25.17 所示。

图 25.17　让 WebView 支持 JavaScript

25.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 26 章

Android 中的小部件

Widget 是一种可以被嵌入到其他程序的视图，并可周期性进行更新。通常情况下，用于开发主屏幕上的信息显示程序。本章将对 Widget 的概念、设计原理、开发步骤、应用 Activity 配置 Widget 以及通过 Service 定时更新 Widget 进行详细介绍。

26.1 Widget 简介

Widget 可以直接译作小部件，它是一个具有特定功能的视图，通常是指放在智能手机、平板电脑等屏幕上的桌面小组件应用，例如，在模拟器的主屏上添加的时钟小部件，如图 26.1 所示。再例如我们手机屏幕上经常看到的音乐、日历、天气预报、信息提醒等组件都是 Widget。它与一般的应用程序有所不同，一般应用虽然也可以以图标的形式（快捷方式）放置在桌面上，但是它们必须点击运行和查看，而 Widget 一般不需要点击运行，就可以直观地呈现其主要内容。当然，Widget 也可以设置为单击打开其他页面或应用。

说明 在Android中，把所有的组件（包括TextView、Button、ImageView和ListView等）都称为Widget，这里介绍的Widget实际上是指App Widget。

在 Android 系统中，自带了多个 Widget 程序，可以在系统的 Widget 选项卡中看到，如图 26.2 所示，主要包括地图、日历、时钟、音乐播放器等。这些组件可以被定时更新，例如，日历组件每天更新；时钟组件每分钟更新等。

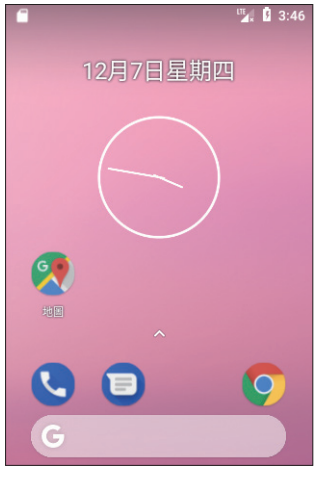


图 26.1　模拟器主屏幕中的 Widget

图 26.2　Android 提供的 Widget

在 Android 系统中，默认情况下，并不是所有的 Widget 组件都会被添加到主屏幕上。用户可以根据自己的需要来有选择的添加，具体的添加方法如下：

在主屏幕长按后显示如图 26.3 所示的三个选项卡，单击“微件”将显示如图 26.2 所示的 Widget 列表，在这里选择要添加的 Widget，长按并选择要放置的位置，便可以将其添加到主屏幕上。

图 26.3　微件选项卡

说明　要删除已经添加到主屏幕上的Widget组件，只需要长按并拖动到“移除”按钮上即可。

26.2 Widget 基础

在进行 Widget 开发时，首先需要了解其设计原则，然后再进行具体的开发，最后将其安装到主屏幕上。如果不需要安装到主屏幕上的 Widget 小程序时，也可以将其从主屏幕上删除。

26.2.1 设计原则

在设计 Widget 时，需要了解以下两点原则，一个是 Widget 的重要组成部分，另一个是 Widget 尺寸的确定公式，下面分别进行介绍。

1. 标准 Widget 剖析

典型的 Widget 有 3 个重要的部分组成，分别是一个限位框（单元格边界）、一个框架和 Widget Controls（Widget 的界面元素）。设计周全的 Widget 会在限位框的边缘和框架之间，以及框架的内边缘和 Widget Controls 之间都保留一些间隙，如图 26.4 所示。其中，在限位框的边缘和框架之间的间隙称为 Widget Margins（外边距）；在框架的内边缘和 Widget Controls 之间的间隙称为 Widget Padding（内边距）。

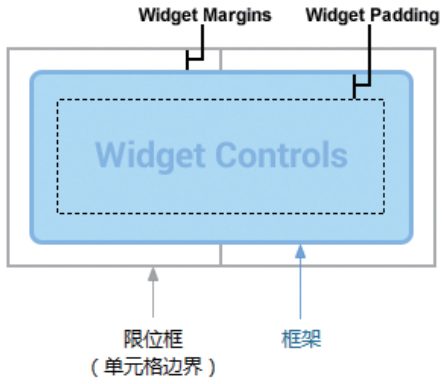


图 26.4　Widget 的组成

为了保证多个 Widget 显示时不会离得太近，一般都会设置 Widget Margins，即限位框边界与框架边界的距离。如果 Widget Margins 的值为 0，则两个 Widget 就会连在一起。所以在 Android 4.0 系统后，会自动添加 Widget Margins，用来保证两个 Widget 之间留有一定的距离。不过要应用这一功能，需要设置项目的 minSdkVersion（最小 SDK 版本）为 14，否则不会自动留有一定的距离。

为了让开发的 Widget 兼容 Android 4.0 以及较早版本，可以按下面的步骤操作。

（1）依次单击 File → Project structure → Flavors 中，设置 TargetSdkVersion（目标 SDK 版本）为 14 或者更高。

（2）创建布局文件时，使用尺寸资源，代码如下：

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android: layout_margin="@dimen/widget_margin">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:background="@drawable/my_widget_background">
        …
    </LinearLayout>
</FrameLayout>
```

（3）创建两个尺寸资源文件，一个放置在 res/values/ 目录下，用于为 Android 4.0 以前的系统提供 margins，另一个放置在 res/values-v14/ 目录下，用于为 Android 4.0 系统提供 margins。

在 res/values/dimens.xml 文件中，设置外边距为 8dp，代码如下：

```
<dimen name="widget_margin">8dp</dimen>
```

在 res/values-v14/dimens.xml 文件中，设置外边距为 0dp，代码如下：

```
<dimen name="widget_margin">0dp</dimen>
```

2. 确定 Widget 的尺寸

在 Android 系统中，将主屏幕划分为多个单元格，其数量和大小会根据设置的不同而不同，一

般将智能手机的屏幕划分为 4×4 个单元格，而平板电脑的屏幕划分为 8×7 个单元格。每个 Widget 必须定义一个 minWidth 和 minHeight，用于确定在默认的情况下占用的最低单元格的数量，具体的计算方法如表 26.1 所示。

表 26.1　Widget 最小尺寸与单元格数量对应关系

Widget 尺寸（minWidth 和 minHeight）	单元格数量
40dp	1
110dp	2
180dp	3
250dp	4
…	…
70n-30	n

说明 表26.1列出的是Android 4.0以后版本的计算原则，而在Android 4.0以前，采用的是74n-2的原则。

在设定 minWidth 和 minHeight 时，最基本的原则是使 Widget 处于最佳的显示状态。

当 Widget 的尺寸不够填满所应占用的单元格时，Widget 会在横向和纵向拉伸，以填充所有应占据的单元格。所以为了增加 Widget 对不同屏幕尺寸和单元格尺寸的适应性，建议尽量使用有自适应用能力的布局，推荐使用 FrameLayout、LinearLayout、RelativeLayout 和 GridLayout。而在设计界面的背景图片时，最好使用 9-Patch 文件。

说明 9-Patch图片是使用Android SDK中提供的工具Draw 9-patch生成的，它的扩展名为.9.png。它是一个可以伸缩的标准PNG图像，Android会自动调整大小来容纳显示的内容，如图26.5为一个9-Patch图片的可拉伸区域和内容填充框。

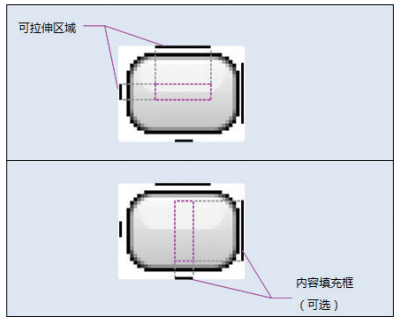


图 26.5　9-Patch 图片的可拉伸区域和内容填充框

26.2.2 开发步骤

开发一个简单的 Widget 组件通常需要经过以下几个步骤。

（1）设计 Widget 的布局。

（2）定义 Widget 的元数据。

（3）实现 Widget 的添加、删除和更新等。

（4）在 AndroidManifest.xml 文件中声明 Widget。

下面通过一个具体的实例来演示开发 Widget 组件的具体步骤。

例 26.1 开发第一个 Widget 组件

在 Android Studio 中创建一个 Module，名称为“First Widget”。实现本实例的具体步骤如下：

（1）在新建 Module 的 res/layout 目录下创建 Widget 布局文件 widget_layout.xml，在该文件中将默认添加的布局管理器修改为相对布局管理器，然后在该布局管理器中添加一个文本框组件和一个图像视图组件，作为 Widget 上显示的内容，关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
    <TextView
        android:id="@+id/msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView1"
        android:layout_centerHorizontal="true"
        android:text="第一个Widget组件"/>
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:src="@mipmap/ic_launcher"/>
</RelativeLayout>
```

（3）在 res 目录下创建一个名称为 xml 的子目录，并且在该目录中再创建一个名称为 widget_template.xml 的 Widget 元数据文件，在该文件中，指定 minWidth 和 minHeight 属性为 180dp 和 110dp，表示这个 Widget 占 2×2 个单元格，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="110dp"
    android:minHeight="110dp"
    android:updatePeriodMillis="86400000"
    android:initialLayout="@layout/widget_layout"
    >
</appwidget-provider>
```

在上面的代码中，设置 updatePeriodMillis 属性用于指定更新周期，这里设置为 86400000 毫秒，表示 1 天；设置 initialLayout 属性用于指定 Widget 使用的布局文件。

（4）在 com.mingrisoft 包中创建一个名称为 FirstWidget 的类，让其继承 android.appwidget.AppWidgetProvider，用来接收与 Widget 相关的删除、失效、生效和更新等消息，然后重写它的 onDeleted()、onDisabled()、onEnabled() 和 onUpdate() 方法，关键代码如下：

```
public class FirstWidget extends AppWidgetProvider {
```

```
    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
    }
    @Override
    public void onDisabled(Context context) {
        super.onDisabled(context);
    }
    @Override
    public void onEnabled(Context context) {
        super.onEnabled(context);
    }
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
                         int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
    }
}
```

（5）在 AndroidManifest.xml 文件中声明 Widget，主要包括指定创建的 AppWidgetProvider 的子类、Widget 的元数据以及元数据的资源路径，关键代码如下：

```
<receiver android:name=".FirstWidget">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE"/>
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/widget_template"/>
</receiver>
```

（6）运行本实例，将显示如图 26.6 所示的默认界面。退出应用后在主屏幕中长按将显示如图 26.7 所示的界面，单击微件，在列表中将添加如图 26.8 所示的 Widget 组件。

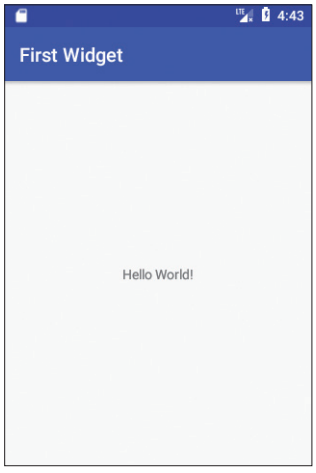


图 26.6　默认运行效果

图 26.7　微件选项卡

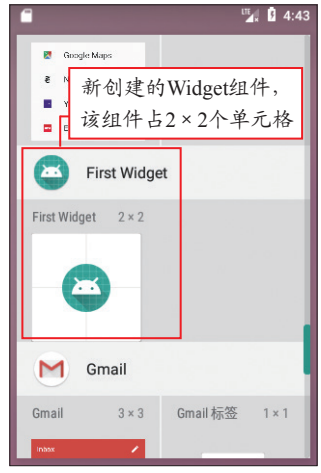


图 26.8　新创建的 Widget 组件

26.2.3 安装及删除

在完成了创建 Widget 示例的所有代码并运行后，将在 Widget 列表中添加一个新的 Widget，但是，此时的 Widget 还没有显示在主屏幕上。如果想添加到主屏幕上，需要在 Widget 列表中，找到该 Widget，如图 26.8 所示，然后在该 Widget 上长按（超过 2 秒），将显示如图 26.9 所示的界面，选择合适的位置，松开手指，即可将该 Widget 添加到主屏幕上，如图 26.10 所示。

图 26.9　选择添加位置的效果

图 26.10　添加到桌面的效果

将 Widget 安装到主屏幕上后，如果想要将其删除，可以通过在主屏幕上的该 Widget 上长按，当桌面上方出现“移除”按钮时，如图 26.11 所示，拖动该 Widget 到“移除”按钮上，当 Widget 变红时，松开手指，就可以将其从主屏幕上删除。

图 26.11　从主屏幕上删除 Widget

26.3 Widget 配置

在使用 Widget 时，有时需要根据个人喜欢设置 Widget 的不同特征，例如，Widget 的外观风格、

字体颜色、字体大小、显示文字或背景图案等。这时可以通过以下知识实现在将 Widget 添加到主屏幕时，启动一个用于配置 Widget 的 Activity，通过这个 Activity 设置 Widget 的特征。

26.3.1 在 Widget 元数据文件中声明 Activity

例 26.2 实现一个配置 Widget 的示例

要通过 Activity 来配置 Widget 特征，首先需要在 Widget 元数据文件中声明该 Activity，具体方法是为 <appwidget-provider> 标记添加 android:configure 属性，其属性值设置为 Activity 所在的类。

首先在例 26.1 的 Module 中创建一个名称为“Main2Activity”的 Activity，然后打开 widget_template.xml 文件，为 <appwidget-provider> 标记添加 android:configure 属性，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="180dp"
    android:minHeight="110dp"
    android:updatePeriodMillis="86400000"
    android:initialLayout="@layout/widget_layout"
    android:configure="com.mingrisoft.Main2Activity"
    >
</appwidget-provider>
```

说明 在上面的代码中，声明的Activity为com.mingrisoft包中的Main2Activity。

26.3.2 创建配置 Widget 的 Activity

创建用于配置 Widget 特征的 Activity，并设置其布局文件。在布局文件中，需要通过添加单选按钮组来实现让用户选择自己所需的特征信息。

例如，在实例 26.2 中，创建配置 Widget 的 Activity 的步骤如下：

（1）打开 activity_main2.xml，在该文件中首先将默认添加的布局管理器修改为相对布局管理器，然后添加一个文本框组件、一个包含 3 个单选按钮的单选按钮组和一个按钮，关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.mingrisoft.Main2Activity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
13          android:text="选择要显示的文字: " />
14      <RadioGroup
15          android:id="@+id/textGroup"
16          android:layout_width="wrap_content"
17          android:layout_height="wrap_content"
18          android:layout_alignParentLeft="true"
19          android:layout_below="@+id/textView1" >
20          <RadioButton
21              android:id="@+id/text0"
22              android:layout_width="wrap_content"
23              android:layout_height="wrap_content"
24              android:checked="true"
25              android:text="明日科技" />
26          <RadioButton
27              android:id="@+id/text1"
28              android:layout_width="wrap_content"
29              android:layout_height="wrap_content"
30              android:text="明日图书" />
31          <RadioButton
32              android:id="@+id/text2"
33              android:layout_width="wrap_content"
34              android:layout_height="wrap_content"
35              android:text="www.mingrisoft.com" />
36      </RadioGroup>
37      <Button
38          android:id="@+id/ok"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:layout_alignLeft="@+id/textGroup"
42          android:layout_below="@+id/textGroup"
43          android:text="确定" />
44  </RelativeLayout>
```

（2）重写 Main2Activity 类中的 onCreate() 方法，在该方法中首先获取布局文件中添加的按钮，并为其指定单击事件监听器，然后在重写的 onClick() 方法中，获取单选按钮组的值。关键代码如下：

```
01  Button btn_ok=(Button)findViewById(R.id.ok);    //获取Activity上添加的"确定"按钮
02  final RadioGroup textGroup=(RadioGroup)findViewById(R.id.textGroup);
03  btn_ok.setOnClickListener(new View.OnClickListener() {
04      @Override
05      public void onClick(View v) {
06          String text="";
07          /******************获取单选按钮组的值*******************/
08          for(int i=0;i<textGroup.getChildCount();i++){
09              RadioButton r=(RadioButton)textGroup.getChildAt(i);
10              if(r.isChecked()){
11                  text=(String)r.getText();
```

```
                break;
            }
        }
        /*******************************************************/
    }
});
```

26.3.3 获取 Widget 的 ID

Widget 的宿主在启动 Activity 时，将 Widget 的 ID 保存在 Intent 中，通过调用 Bundle 对象的 getInt() 方法，可以获取 Widget 的 ID。

例如，要获取例 26.2 Module 中的 Widge 的 ID，可以通过下面的代码实现：

```
Intent intent=getIntent();          //获取Intent对象
Bundle extras=intent.getExtras();   //获取Bundle对象
int mAppWidgetId=0;                 //定义保存Widget ID的变量
if(extras!=null){
    mAppWidgetId=extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
            AppWidgetManager.INVALID_APPWIDGET_ID);
}
if(mAppWidgetId==AppWidgetManager.INVALID_APPWIDGET_ID){
    finish();
}
```

说明 如果AppWidgetManager.INVALID_APPWIDGET_ID的值为0，表示没有获取Widget的ID。

26.3.4 更新 Widget

在实现更新 Widget 时，首先需要通过调用 getInstance() 方法获取 AppWidgetManager 实例，然后创建一个 RemoteViews 对象，通过其更改 Widget 的界面元素，最后调用 updateAppWidget() 方法完成 Widget 的更新。

例如，在例 26.2 Module 中，添加更新 Widget 上显示文字为单选按钮组的值，可以使用下面的代码。

```
AppWidgetManager appWidgetManager=AppWidgetManager.getInstance(Main2Activity.this);
RemoteViews views=new RemoteViews(Main2Activity.this.getPackageName(),
        R.layout.widget_layout);
views.setTextViewText(R.id.msg, text);
appWidgetManager.updateAppWidget(mAppWidgetId, views);
```

26.3.5 设置返回信息并关闭 Activity

要实现设置返回信息，并关闭 Activity，可以通过调用 setResult() 方法实现。如果返回码为 RESULT_OK 表示设置成功，宿主会将 Widget 实例加载到主屏幕上；如果返回码为 RESULT_CANCELED 表示取消，这时宿主会取消 Widget 实例的加载过程，Widget 将不会显示到主屏幕上。例如，在实例 02 Module

中设置返回信息为获取的 Widget ID，并且使用 AppWidgetManager.EXTRA_APPWIDGET_ID 作为关键字，可以使用下面的代码实现。

```
01  Intent resultValue=new Intent();
02  resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId);
03  setResult(RESULT_OK,resultValue);
04  finish();
```

运行本实例，将创建的 Widget 向主屏幕上添加时，将显示如图 26.12 所示的选择显示文字的 Activity，选择想要显示的文字（如明日图书），单击“确定”按钮，即可在主屏幕上显示如图 26.13 所示的 Widget。

图 26.12　选择显示文字的 Activity

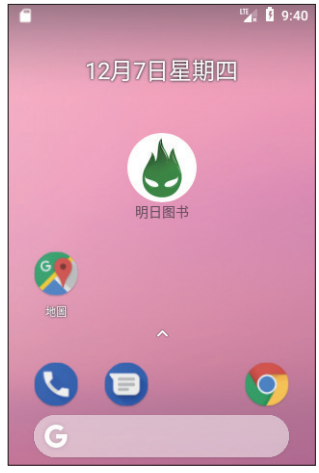


图 26.13　在主屏幕上显示的 Widget

26.4 Widget 与 Service

在开发 Widget 时，在 Widget 元数据文件中，通过 updatePeriodMillis 属性可以指定更新时间，但是该属性值必须大于 30 分钟，如果低于这个时间，该属性将不生效。另外，如果在 onUpdate() 方法中更新时，必须保证要运行的代码可以在 5 秒钟内执行完毕，否则将会产生无响应（Application Not Responding,ANR）错误。所以如果需要在 Widget 中进行频繁更新，一般需要采用 Service 周期性地更新 Widget 的方法。下面将通过一个具体的实例来介绍如何实现通过 Service 更新 Widget。

例 26.3　实现一个定时更新的 Widget

在 Android Studio 中创建 Module，名称为“Widget And Service”，在该 Module 中实现一个定时更新的 Widget，在该 Widget 中，将每隔一分钟更新一次显示时间。实现本实例的具体步骤如下：

（1）在 res/drawable 目录中创建一个背景资源，用于作为 Widget 的背景。

（2）在新建 Module 的 res/layout 目录下创建 Widget 布局文件 widget_layout.xml，在该文件中首先将默认添加的布局管理器修改为相对布局管理器，并且为其设置背景资源，然后在该布局管理器中添加一个文本框组件，作为 Widget 上显示的内容，关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/solidline_drawable"
    android:gravity="center"
    >
    <TextView
        android:id="@+id/msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:text="正在获取时间..."
        android:textColor="#000000"/>
</RelativeLayout>
```

（3）在 res 目录下创建一个名称为 xml 的子目录，并且在该目录中再创建一个名称为 widget_template.xml 的 Widget 元数据文件，在该文件中，指定 minWidth 和 minHeight 属性为 110dp 和 40dp，表示这个 Widget 占 2×1 个单元格，具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="110dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:initialLayout="@layout/widget_layout"
    >
</appwidget-provider>
```

（4）在 com.mingrisoft 包中，创建一个名称为 TimerWidget 的类，让其继承 android.appwidget.AppWidgetProvider，用来接收与 Widget 相关的删除、失效、生效和更新等消息，然后声明一个保存所有 Widget 实例的 ID 值的队列，并且重写它的 onDeleted()、onDisabled()、onEnabled() 和 onUpdate() 方法，最后再创建 updateAppWidget() 方法，用于更新 Widget，关键代码如下：

```
public class TimerWidget extends AppWidgetProvider {
    //声明一个保存所有Widget实例的ID值的队列
    private static Queue<Integer> widgetIds=new LinkedList<Integer>();
    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
    }
    @Override
    public void onDisabled(Context context) {
        super.onDisabled(context);
    }
```

```
    @Override
    public void onEnabled(Context context) {
        super.onEnabled(context);
    }
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
                         int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
    }
    public static void updateAppWidget(Context context, String time) {
        //获取AppWidgetManager对象
        AppWidgetManager appWidgetManager=AppWidgetManager.getInstance(context);
        //获取Widget视图对象
      RemoteViewsviews=newRemoteViews(context.getPackageName(),R.layout.widget_layout);
        //更改Widget的msg文本框的显示文本为获取的系统时间
        views.setTextViewText(R.id.msg,time);
        final int N=widgetIds.size();          //获取队列中元素的个数
        for(int i=0;i<N;i++){
            int appWidgetId=widgetIds.poll(); //移除并返回队列头部的元素
            appWidgetManager.updateAppWidget(appWidgetId, views);  //更新Widget
            widgetIds.add(appWidgetId);        //将Widget ID添加到队列中
        }
    }
}
```

（5）创建名称为 TimerService 的 Service，让其实现 Runnable 接口，然后重写 run() 方法，在该方法中每隔一分钟获取一次系统时间，并且更新 Widget 界面，最后再重写 onStart() 方法，在该方法中，如果没有开启线程，则创建并开启线程，关键代码如下：

```
public class TimerService extends Service implements Runnable {
    boolean threadRunning=false;              //标记线程是否开启
    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
    @Override
    public void run() {
        while(!Thread.interrupted()){         //当线程没有中断
            SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd HH:mm");//设置日期格式
            String time=df.format(new Date());//获取当前系统时间
            TimerWidget.updateAppWidget(this,time);//更新Widget界面
            try {
                Thread.sleep(60000);          //让线程休眠1分钟
            } catch (InterruptedException e) {
                e.printStackTrace();
```

```
            }
        }
    }
    @Override
    public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
        if(!threadRunning){
            threadRunning=true;
            new Thread(this).start();                    //创建并开启线程
        }
    }
}
```

（6）在 TimerWidget 类的 onUpdate() 方法中将 Widgest ID 添加到队列中，并且开启 TimerService 服务，具体代码如下：

```
for(int i=0;i<appWidgetIds.length;i++){
    widgetIds.add(appWidgetIds[i]);                      //将Widget ID添加到队列中
}
context.startService(new Intent(context,TimerService.class));//开启服务
```

（7）在 TimerWidget 类的 onDeleted() 方法中将 Widget 的 ID 从 WidgetIds 队列中移除，具体代码如下：

```
for(int i=0;i<appWidgetIds.length;i++){
    if(widgetIds.contains(appWidgetIds[i])){             //如果在队列中
        widgetIds.remove(appWidgetIds[i]);               //从队列中移除
    }
}
```

（8）在 TimerWidget 类的 onDisabled() 方法中停止 TimerService 服务，具体代码如下：

```
context.stopService(new Intent(context,TimerService.class));   //停止服务
```

（9）在 AndroidManifest.xml 文件的 <application> 标记中声明 Widget 和 Service，关键代码如下：

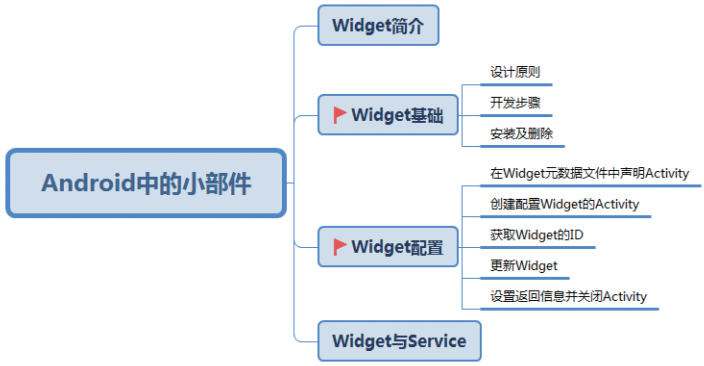
```
<receiver android:name=".TimerWidget">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE"/>
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/widget_template"/>
</receiver>
<service android:name=".TimerService" />
```

运行本实例，将创建的 Widget 添加到主屏幕上，然后每隔一分钟，会自动更新一次显示的时间，如图 26.14 所示。

图 26.14　定时更新的 Widget

26.5 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 27 章

常用的第三方技术

在开发 Android 项目的过程中，除了谷歌对 Android 系统自身不断的优化与升级，还需要使用众多服务商所提供的开发包。本章将介绍 App 常用的第三方技术，以及第三方技术的使用方法，主要包括第三方登录、分享与支付。

27.1 第三方登录

常见的 App 应用多数都会借助第三方平台实现登录，例如，迅雷 App 中的第三方登录界面（如图 27.1 所示）以及今日头条 App 中的第三方登录界面（如图 27.2 所示）。友盟社会化组件，可以让移动应用快速具备社会化登录、分享、评论、喜欢等功能，并提供实时、全面的社会化数据统计分析服务。

图 27.1　迅雷 App 第三方登录

图 27.2　今日头条 App 第三方分享

27.1.1 申请第三方账号

大部分 App 应用进行登录、分享操作时都需要在第三方平台创建应用并提交审核，创建应用后，登录与分享操作时所显示的应用图标、名称需要在对应的开放平台进行设置，例如，QQ、微信、新浪、Facebook、钉钉等。

1. QQ 及 Qzone

QQ 及 Qzone 使用同一个 AppID 及 Appkey，登录腾讯开放平台 (http://open.qq.com/)，选择 Android 或 iOS 应用，填写相关应用信息并提交审核，未审核前通过只能使用测试账号。

2. 微信

登录微信开放平台 (https://open.weixin.qq.com/)，填写相关应用信息，审核通过后获取到微信 AppID 及 AppSecret，如果需要微信登录功能，需要申请微信登录权限，注意微信登录有效期为一年，需要按时在微信平台认证。

注意 应用包名、签名设置必须和打包后的apk一致。

3. 新浪微博

登录新浪微博开放平台 (http://open.weibo.com/)，填写相关应用信息并上传 icon 图片，即可获取到 AppID 及 AppSecret。同时确保授权回调页与代码中配置一致授权回调页、取消授权回调页设置在应用信息 → 高级信息，具体位置如图 27.3 所示。

图 27.3 授权设置

说明 由于第三方平台种类较多，本节主要提供以上三种平台作为参考，如需其它平台申请账号方法可以参考友盟三方账号申请页面中进行查询（http://dev.umeng.com/social/android/operation）。

27.1.2 实现第三方登录

1. 申请友盟 AppKey

（1）注册友盟开发者账号并登陆，然后展开全部产品，单击立即使用 U-Share 分享，如图 27.4 所示。

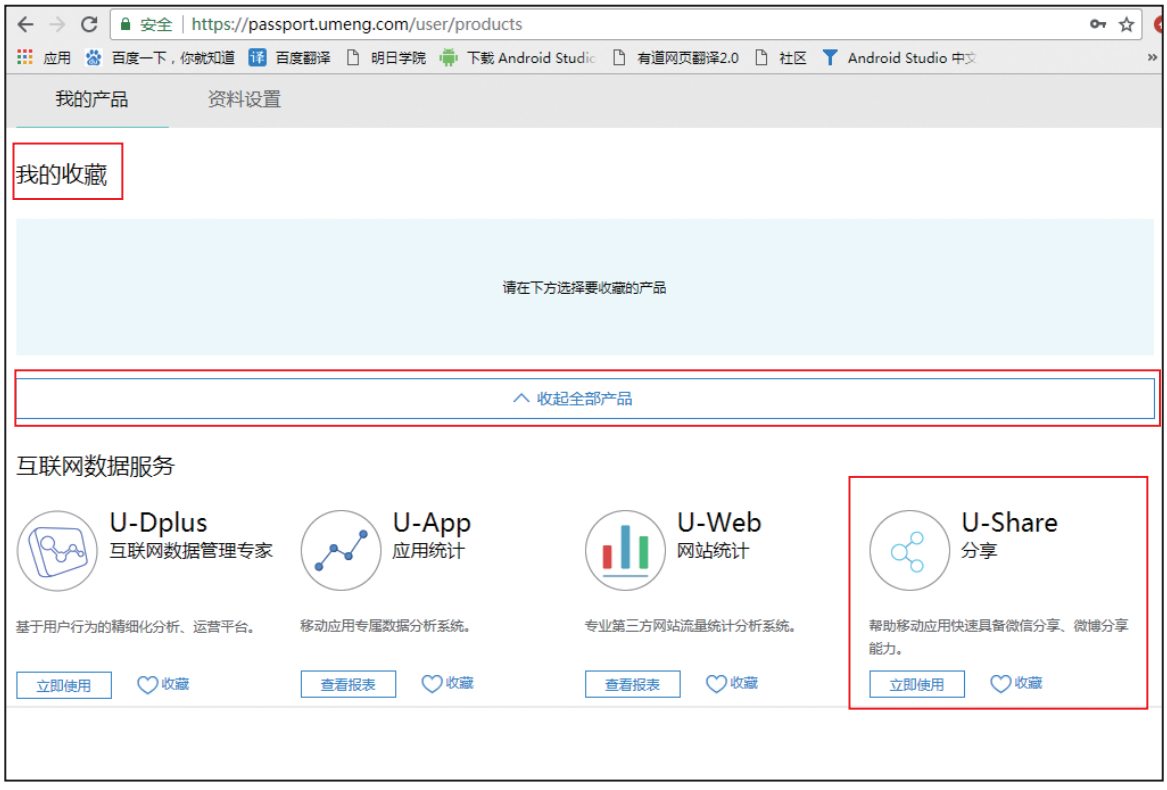


图 27.4　使用 U-Share 分享

（2）打开友盟我的产品页面，在该页面中单击左下方添加新应用如图 27.5 所示。

图 27.5　打开友盟我的产品页面

（3）将显示如图 27.6 所示的应用管理页面，在该页面添加应用基本信息，单击提交并获取 AppKey 按钮即可获取 AppKey。

图 27.6　获取友盟 AppKey

（4）登录友盟网站，个人中心 → 展开全部产品 → 立即使用 U-Share → 左上角选择应用 → 组件 → 社会化分享 → 设置 → 填写自己申请的 QQ 平台 App ID 与 Appkey 如图 27.7 所示。

图 27.7　配置社交平台

2. 下载集成 Demo

打开（http://mobile.umeng.com/custom_sdk）友盟组件下载中心，下载官方写好的集成 Demo 进行开发的参考，也可以根据需求勾选相应的模块进行 SDK 的下载，如图 27.8 所示。

图 27.8 组件下载中心

说明 由于SDK版本不断更新，这里建议读者直接下载集成Demo，集成Demo不仅可以进行使用方法的参考，还可以使用Demo中配置的SDK，这样可以提高开发效率。

3. 工程配置

（1）集成 Demo 下载完成后，将 ShareSDK 模块导入自己编写的项目当中，该项目依赖于 ShareSDK 模块，参考如图 27.9 所示的界面。然后将集成 Demo 中 app 模块下 libs 文件夹中的资源库与所需 jar 包手动导入至自己编写的项目中，再选中 jar 包，右键菜单中选择 Add As Library 选项添加该 jar 包如图 27.10 所示。

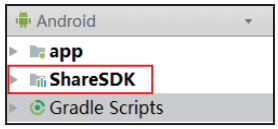


图 27.9 导入集成 Demo 中的 ShareSDK 模块

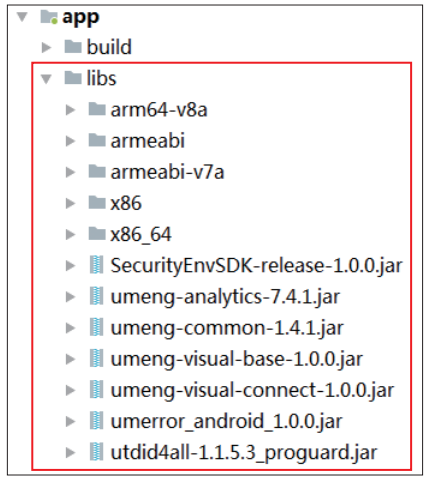


图 27.10　导入资源库与所需的 jar 包

注意　选中其中一个jar包添加后其它jar包将自动添加在自己编写的项目当中。

（2）打开 Android Manifest.XML 文件，添加三方登录所需要的权限，代码如下：

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name=
    "android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
```

（3）添加 QQ 平台所需要的 Activity，代码如下：

```
<activity
    android:name="com.tencent.tauth.AuthActivity"
    android:launchMode="singleTask"
    android:noHistory="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="tencent3921700954" />
    </intent-filter>
</activity>
<activity
    android:name="com.tencent.connect.common.AssistActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
```

（4）添加友盟 AppKey，代码如下：

```
<meta-data
    android:name="UMENG_APPKEY"
    android:value="5a72a170a40fa32be2000216" >
</meta-data>
```

（5）创建 BaseApplication.Java 类，该类继承自 Application 类，然后在该类中实现友盟的初始化工作并配置第三方平台，这里以手机 QQ 为例。代码如下：

```
public class BaseApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        /**
         * 开启debug模式，方便定位错误，具体错误检查方式可以查看
         * http://dev.umeng.com/social/android/quick-integration的报错必看，
         * 正式发布，请关闭该模式
         */
        Config.DEBUG = true;
        UMShareAPI.get(this);  //初始化友盟
    }
    //各个平台的配置，建议放在全局Application或者程序入口
    {
        //QQ登录
        PlatformConfig.setQQZone("1106719830", "z97X5Uf93WY12bdb");
    }
}
```

（6）打开 AndroidManifest.xml 文件，在该文件的 application 中配置第三方 AppKey。代码如下：

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:name=".BaseApplication"
    android:theme="@style/AppTheme">
```

（7）打开 MainActivity.Java 类，在该类中首先定义保存第三方平台信息的集合与三种平台的登录图标。关键代码如下：

```
//保存平台信息的集合
public ArrayList<SnsPlatform> platforms = new ArrayList<SnsPlatform>();
//分享平台集合
private SHARE_MEDIA[] list = {SHARE_MEDIA.QQ, SHARE_MEDIA.SINA, SHARE_MEDIA.WEIXIN};
private ImageView QQ, WeiXin, Sina;   //三种平台的图标
```

（8）创建 initPlatforms() 方法，用于初始化平台信息；创建 init() 方法，用于初始化控件与设置单击事件；在 onCreate() 方法中调用 initPlatforms() 方法和 init() 方法实现初始化功能。关键代码如下：

```
//初始化平台信息
private void initPlatforms() {
    platforms.clear();                                          //清空平台信息
    for (SHARE_MEDIA e : list) {
        if (!e.toString().equals(SHARE_MEDIA.GENERIC.toString())) {
            platforms.add(e.toSnsPlatform());                   //添加平台信息
        }
    }
}
//初始化控件
private void init() {
    QQ = (ImageView) findViewById(R.id.iv_qq_login);            //获取QQ登录图标
    WeiXin = (ImageView) findViewById(R.id.iv_weixin_login);//获取微信登录图标
    Sina = (ImageView) findViewById(R.id.iv_sina_login);        //获取新浪登录图标
    //设置单击事件
    QQ.setOnClickListener(this);
    WeiXin.setOnClickListener(this);
    Sina.setOnClickListener(this);
}
```

（9）实现登录图标单击事件监听方法，然后创建 UmEnter() 方法，用于实现登录事件的处理，最后在登录图标的单击事件监听方法中调用 UmEnter() 方法即可。关键代码如下：

```
private void UmEnter(int pos) {
    //获取登录状态
    final boolean isauth = UMShareAPI.get(this).
            isAuthorize((Activity) this, platforms.get(pos).mPlatform);
    if (isauth) {
        //如果登录状态就移除登录
        UMShareAPI.get(this).deleteOauth((Activity) this,
                platforms.get(pos).mPlatform, authListener);
        Toast.makeText(this, "移除登录！",
                Toast.LENGTH_SHORT).show();
    } else {
        //否则进行登录
        UMShareAPI.get(this).doOauthVerify((Activity) this,
                platforms.get(pos).mPlatform, authListener);
    }
}
```

（10）创建友盟登录监听对象，该监听对象为三方登录授权的回调，其中授权成功会回调 onComplete，取消授权回调 onCancel，授权错误回调 onError，对应的错误信息可以通过 onError 的 Throwable 参数来打印。关键代码如下：

```
UMAuthListener authListener = new UMAuthListener() {
    @Override
    public void onStart(SHARE_MEDIA share_media) {
    }
    //登录成功回调该方法，该方法中的data为反馈信息其中包含QQ号码等等信息
    @Override
```

```
07      public void onComplete(SHARE_MEDIA platform,
08                             int action, Map<String, String> data) {
09          Toast.makeText(MainActivity.this,
10                  "成功了", Toast.LENGTH_LONG).show();
11      }
12      //登录失败回调该方法
13      @Override
14      public void onError(SHARE_MEDIA platform, int action, Throwable t) {
15          Toast.makeText(MainActivity.this,
16                  "失败: " + t.getMessage(), Toast.LENGTH_LONG).show();
17      }
18      //取消登录回调该方法
19      @Override
20      public void onCancel(SHARE_MEDIA platform, int action) {
21          Toast.makeText(MainActivity.this,
22                  "取消了", Toast.LENGTH_LONG).show();
23      }
24  };
```


（11）最后在登录所在的 Activity 里复写 onActivityResult 方法，注意不可在 fragment 中实现，如果在 fragment 中调用登录，在 fragment 依赖的 Activity 中实现，如果不实现 onActivityResult 方法，会导致登录或回调无法正常进行。代码如下：

```
01  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
02      super.onActivityResult(requestCode, resultCode, data);
03      UMShareAPI.get(this).onActivityResult(requestCode, resultCode, data);
04  }
```

（12）运行本项目，将显示如图 27.11 所示的界面，单击左下角的 QQ 登录图标，将显示如图 27.12 所示的登录授权界面。单击登录按钮将返回图 27.11 所示的界面并提示登录成功，同时电脑中的 QQ 将显示如图 27.13 所示的三方登录提示。

图 27.11　默认显示的界面

图 27.12　登录授权界面

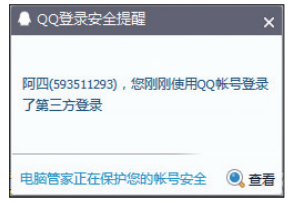


图 27.13　电脑 QQ 三方登录提示

注意 本实例仅实现QQ三方登录，其他三方登录按照以上方式申请三方平台的AppKey然后进行友盟社交平台进行配置即可。

27.2 社会化分享

社会化分享是指用户通过网络把文本、图片等信息分享到用户的交际圈当中，从而实现让朋友了解自己的分享内容。对于 App 来说，分享的社交平台有很多，目前最常用的就是 QQ 分享与微信分享，如图 27.14 所示和图 27.15 所示。本节将介绍 QQ 与微信 SDK 中的分享功能。

图 27.14 QQ 分享界面

图 27.15 微信分享界面

27.2.1 QQ 分享

QQ 分享可以实现分享消息到 QQ 或者是分享到 QQ 空间，分享消息到 QQ 的接口时，可将新闻、图片、文字、应用等分享给 QQ 好友、群和讨论组。Tencent 类的 shareToQQ 函数可直接调用，不需要用户授权（使用手机 QQ 当前的登录态）。调用将打开分享的界面，用户选择好友、群或讨论组之后，点击确定即可完成分享，并进入与该好友进行对话的窗口。

例如，分享图文消息可以通过以下代码实现：

```
01  private void onClickShare() {
02      final Bundle params = new Bundle();
03      params.putInt(QQShare.SHARE_TO_QQ_KEY_TYPE, Tencent.SHARE_TO_QQ_TYPE_DEFAULT);
04      params.putString(QQShare.SHARE_TO_QQ_TITLE, "要分享的标题");
05      params.putString(QQShare.SHARE_TO_QQ_SUMMARY,  "要分享的摘要");
06      params.putString(QQShare.SHARE_TO_QQ_TARGET_URL,
07              "http://www.qq.com/news/1.html");
08      params.putString(QQShare.SHARE_TO_QQ_IMAGE_URL,
09              "http://imgcache.qq.com/qzone/space_item/pre/0/66768.gif");
10      params.putString(QQShare.SHARE_TO_QQ_APP_NAME,  "测试应用222222");
11      params.putInt(QQShare.SHARE_TO_QQ_EXT_INT,  "其他附加功能");
12      mTencent.shareToQQ(MainActivity.this, params, new BaseUiListener());
13  }
```

调用分享接口的 params 参数说明如表 27.1 所示。

表 27.1　分享接口的 params 参数

参　数	是否必填	类　型	参数说明
QQShare.SHARE_TO_QQ_KEY_TYPE	必填	Int	分享的类型。图文分享 (普通分享) 填 Tencent.SHARE_TO_QQ_TYPE_DEFAULT
QQShare.PARAM_TARGET_URL	必填	String	这条分享消息被好友点击后的跳转 URL
QQShare.PARAM_TITLE	必填	String	分享的标题 , 最长 30 个字符
QQShare.PARAM_SUMMARY	可选	String	分享的消息摘要，最长 40 个字
QQShare.SHARE_TO_QQ_IMAGE_URL	可选	String	分享图片的 URL 或者本地路径
QQShare.SHARE_TO_QQ_APP_NAME	可选	String	手 Q 客户端顶部，替换“返回”按钮文字，如果为空，用返回代替
QQShare.SHARE_TO_QQ_EXT_INT	可选	Int	分享额外选项，两种类型可选（默认是不隐藏分享到 QZone 按钮且不自动打开分享到 QZone 的对话框）：Tencent.SHARE_TO_QQ_FLAG_QZONE_AUTO_OPEN，分享时自动打开分享到 QZone 的对话框 Tencent.SHARE_TO_QQ_FLAG_QZONE_ITEM_HIDE，分享时隐藏分享到 QZone 按钮

1.　配置工程

（1）打开（http://wiki.connect.qq.com）QQ 互联文档资料页，在左侧的导航列表中依次单击 SDK 及资源下载 → SDK 下载，在 SDK for 移动应用接入中下载 Android SDK，如图 27.16 所示。

图 27.16　下载 QQ 互联 SDK

（2）SDK 下载完成后，解压 SDK 压缩包，将 libs 文件夹中的 jar 包手动导入到自己编写的项目当中，然后选中导入的 jar 包，右键菜单中选择 Add As Library 选项将 jar 包添加在项目当中，如图 27.17 所示。

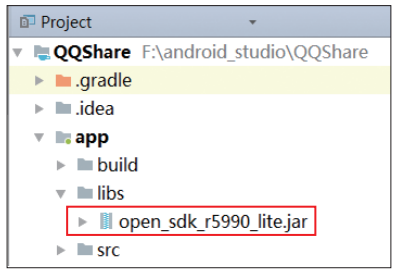


图 27.17　添加 jar 包

（3）打开 AndroidManifest.xml 文件，在该文件中首先添加 QQ 分享所需要的网络权限，代码如下：

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

（4）在 <application> 节点下增加以下配置（注：不配置将会导致无法调用 API），代码如下：

```
<activity
    android:name="com.tencent.tauth.AuthActivity"
    android:launchMode="singleTask"
    android:noHistory="true" >
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="tencent100330589" />
        <!--tencent100330589用于添加自己在QQ开发平台申请的AppKey即可 -->
    </intent-filter>
</activity>
<activity
    android:name="com.tencent.connect.common.AssistActivity"
    android:configChanges="orientation|keyboardHidden"
    android:screenOrientation="behind"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
```

2. 实现 QQ 好友分享功能

（1）在 MainActivity.Java 文件中，定义相关控件与 QQ 互联平台申请的 AppID，代码如下：

```
private Tencent mTencent;     //定义QQ平台
private EditText tv_title;    //填写分享标题的控件
private EditText tv_content; //填写分享内容的控件
//这里替换为开发者在QQ互联平台申请的AppID
private final String QQ_APPID = "100330589";
```

（2）在 onCreate() 方法中创建 QQ 分享实例并获取相关控件，关键代码如下：

```
//创建一个QQ分享实例，用于QQ分享、QQ空间分享、腾讯微博分享
mTencent = Tencent.createInstance(QQ_APPID, this);
tv_title=findViewById(R.id.tv_title);        //获取填写标题的控件
tv_content=findViewById(R.id.tv_content);    //获取填写内容的控件
```

（3）创建 onShare() 方法，该方法为分享按钮的单击事件处理，代码如下：

```
public void onShare(View view) {
    Bundle params = new Bundle();
    //设置分享标题
    params.putString(QQShare.SHARE_TO_QQ_TITLE,
            tv_title.getText().toString());
    //设置分享内容
    params.putString(QQShare.SHARE_TO_QQ_SUMMARY,
            tv_content.getText().toString());
    //好友单击后跳转的网页
    params.putString(QQShare.SHARE_TO_QQ_TARGET_URL,
            "http://www.mingrisoft.com");
    //分享显示的图标地址
    params.putString(QQShare.SHARE_TO_QQ_IMAGE_URL,
            "http://www.mingrisoft.com/Public/images/code_news.jpg");
    //分享qq好友并绑定分享监听器
    mTencent.shareToQQ(MainActivity.this, params,qqShareListener);
}
```

（4）创建分享监听器实例，并实现其中的三个方法。代码如下：

```
IUiListener qqShareListener = new IUiListener() {
    @Override
    public void onCancel() {                    //分享取消时调用该方法
        Toast.makeText(MainActivity.this,
                "分享取消！", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onComplete(Object response) {   //分享成功时调用该方法
        Toast.makeText(MainActivity.this,
                "分享成功！", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onError(UiError e) {            //分享错误时调用该方法
        Toast.makeText(MainActivity.this,
                "分享错误！", Toast.LENGTH_SHORT).show();
    }
};
```

（5）重写 onActivityResult() 方法，用于成功接收回调，否则监听器无效。代码如下：

```
01  @Override
02  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
03      Tencent.onActivityResultData(requestCode,resultCode,data,qqShareListener);
04  }
```

（6）运行本项目，填写分享标题与内容，将显示如图 27.18 所示的界面；单击分享 QQ 好友按钮，然后选择需要分享的好友将显示如图 27.19 所示的界面；发送分享后将显示如图 27.20 所示的界面。好友单击分享内容将显示指定的网页。

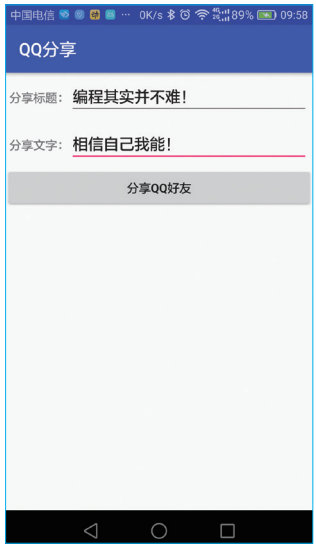


图 27.18　填写分享标题与内容

图 27.19　选择分享好友

图 27.20　发送分享内容

说明　由于QQ分享内容较多，这里仅介绍了QQ好友分享图文相关的介绍与示例。在下载的SDK压缩包中可以查看接口调用说明来学习QQ分享接口的其他内容。

27.3　微信分享

微信分享是指第三方 App 通过接入该功能，让用户可以从 App 分享文字、图片、音乐、视频、网页至微信好友会话、朋友圈。微信的好友分享与朋友圈分享主要通过 IWXAPI、WXMediaMessage、SendMessageToWX.Req 三个类来实现。

IWXAPI 类主要用于注册微信的 AppID 与发送数据至微信，常用方法如表 27.2 所示。

表 27.2　IWXAPI 类常用方法

方法	描　述
createWXAPI	通过 WXAPIFactory 工厂，获取 IWXAPI 的实例
registerApp	注册指定的 AppID
sendReq	调用 API 接口发送数据到微信。该方法的参数为 SendMessageToWX.Req 对象

WXMediaMessage 类主要用于设置媒体对象与分享内容，常用方法如表 27.3 所示。

表 27.3 WXMediaMessage 类常用方法

方法	描 述
title	分享的标题
description	分享的内容
mediaObject	分享的媒体信息，WXTextObject 为文本、WXImageObject 为图片、WXMusicObject 为音乐、WXVideoObject 为视频、WXWebpageObject 为网页类型
thumbData	分享的缩略图

SendMessageToWX.Req 类主要用于发送分享请求，常用方法如表 27.4 所示。

表 27.4 SendMessageToWX.Req 类常用方法

方法	描 述
thansaction	用于标识每次请求的唯一性
message	设置请求信息，参数为 WXMediaMessage 对象
scene	设置请求场景，SendMessageToWX.Req.WXSceneTimeline 表示分享至朋友圈、WXSceneSession 分享给微信好友、WXSceneFavorite 添加至微信收藏。

实现微信分享的具体步骤如下：

1. 配置工程

（1）打开微信开放平台网址（https://open.weixin.qq.com/），在顶部导航栏中选择“资源中心”，然后在资源中心页面左侧的导航栏中依次选择“资源下载”→“Android 资源下载”如图 27.21 所示。

图 27.21 Android 资源下载

说明 向下滑动网页可以在范例代码处下载官方提供的范例工程。

（2）为了方便测试微信分享功能，在创建 Android 项目时需要将包名设置为“net.sourceforge.simcpux”，如图 27.22 所示，这样可以使用范例工程中的 AppID，否则需要开发者在微信开发平台自行申请 AppID，并需要审核通过才会生效。

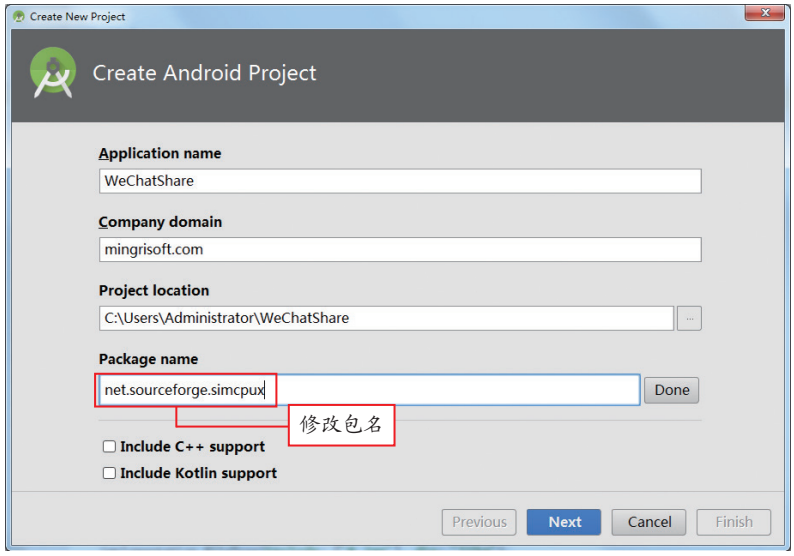


图 27.22　修改项目包名

（3）打开 build.gradle（Module: app）文件，在该文件中的 android 节点中进行签名配置，代码如下：

```
signingConfigs {
    debug {
        storeFile file("debug.keystore")
    }
}
```

（4）将范例工程中的 debug.keystore 文件复制在自己编写项目的 app 模块中，如图 27.23 所示。

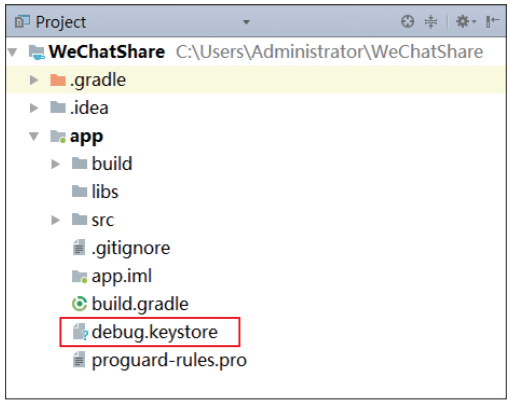


图 27.23　复制范例工程中的签名文件

（5）在 build.gradle 文件中，添加如下依赖代码即可：

```
compile 'com.tencent.mm.opensdk:wechat-sdk-android-with-mta:+'
```

2. 实现微信朋友圈分享功能

（1）在 MainActivity.Java 文件中，定义微信分享接口与 AppID，代码如下：

```
//IWXAPI 是第三方app和微信通信的openapi接口
private IWXAPI api;
//APP_ID 替换为你的应用从官方网站申请到的合法appId
public static final String APP_ID = "wxd930ea5d5a258f4f";
private EditText text;                    //填写分享内容的编辑框
```

（2）在 onCreate() 方法中，创建微信分享接口并注册 AppID，代码如下：

```
//通过WXAPIFactory工厂，获取IWXAPI的实例
api = WXAPIFactory.createWXAPI(this, APP_ID, false);
//将该app注册到微信
api.registerApp(APP_ID);
text = findViewById(R.id.tv_text);      //获取填写分享内容的编辑框
```

（3）创建 onShare() 与 buildTransaction() 方法，用于实现分享文本信息至朋友圈当中，代码如下：

```
public void onShare(View view) {
    //分享文本消息
    WXTextObject textObj = new WXTextObject();
    //设置文本信息
    textObj.text = text.getText().toString();
    //创建媒体对象
    WXMediaMessage msg = new WXMediaMessage();
    //设置媒体对象为文本
    msg.mediaObject = textObj;
    msg.description = text.getText().toString();
    //构造一个req
    SendMessageToWX.Req req = new SendMessageToWX.Req();
    //transaction字段用于唯一标识一个请求
    req.transaction = buildTransaction("text");
    //设置发送的消息
    req.message = msg;
    //发送信息至朋友圈
    req.scene = SendMessageToWX.Req.WXSceneTimeline;
    api.sendReq(req);       //调用api接口发送数据到微信
    //发送到聊天界面——WXSceneSession
    //发送到朋友圈——WXSceneTimeline
    //添加到微信收藏——WXSceneFavorite
}
//标识符
private String buildTransaction(final String type) {
    return (type == null) ? String.valueOf(System.currentTimeMillis())
```

```
27              : type + System.currentTimeMillis();
28  }
```

（4）在 net.sourceforge.simcpux 包下添加 wxapi 包，并且在该包中创建名称为 WXEntryActivity 的 Java 类，该类继承自 AppCompatActivity，然后实现 IWXAPIEventHandler 接口并重写相应的方法，用于实现微信分享后的回调界面，关键代码如下：

```
01  @Override
02  public void onResp(BaseResp resp) {
03     String result = "";
04     switch (resp.errCode) {
05     case BaseResp.ErrCode.ERR_OK:
06        result = "发送成功";
07        break;
08     case BaseResp.ErrCode.ERR_USER_CANCEL:
09        result = "发送取消";
10        break;
11     case BaseResp.ErrCode.ERR_AUTH_DENIED:
12        result = "发送拒绝";
13        break;
14     default:
15        result = "发送未知异常";
16        break;
17     }
18     Toast.makeText(this, result, Toast.LENGTH_LONG).show();
19     finish();
20  }
```

（5）为了接收到内容，需要在 WXEntryActivity 类的 onCreate() 方法中添加以下代码：

```
01  try {
02      api.handleIntent(getIntent(), this);
03  } catch (Exception e) {
04      e.printStackTrace();
05  }
```

（6）在 AndroidManifest.xml 文件中注册微信分享的回调界面 WXEntryActivity，代码如下：

```
01  <!--微信回调页-->
02  <activity
03      android:name=".wxapi.WXEntryActivity"
04      android:label="@string/app_name"
05      android:exported="true"
06      android:screenOrientation="portrait"/>
```

（7）运行本项目，填写需要分享的文本信息，将显示如图 27.24 所示的界面，单击分享微信朋友圈按钮，将显示如图 27.25 所示的界面。发送分享后，在微信朋友圈中将显示如图 27.26 所示的界面。

图 27.24　填写分享内容

图 27.25　朋友圈发送界面

图 27.26　朋友圈显示分享内容

27

27.3　第三方支付

第三方支付是指具备一定实力和信誉保障的独立机构，采用与各大银行签约的方式，通过与银行支付结算系统接口对接而促成交易双方进行交易的网络支付模式。例如微信支付（如图 27.27 所示）和支付宝支付（如图 27.28 所示）。

图 27.27　微信支付

图 27.28　支付宝支付

27.3.1. 微信支付

微信支付是集成在微信 App 中的支付功能，用户可以通过手机完成快速的支付流程。微信支付以绑定银行卡的快捷支付为基础，向用户提供安全、快捷、高效的支付服务。

1. 申请微信支付

（1）打开微信支付官方网站（https://pay.weixin.qq.com），注册账号。

（2）开放平台需进行开发者资质认证后才可申请微信支付，认证费：300 元 / 次；提交 App 基本信息，通过开放平台应用审核，以获得 AppID。

（3）登录开放平台，单击“管理中心”，选择需要申请支付功能对应的 App，开始填写资料等待审核，审核时间为 1-5 个工作日内。

（4）资料审核通过后，请登录联系人邮箱查收商户号和密码，完成账户验证。

（5）本协议为线上电子协议，签署后方可进行交易及资金结算，签署完立即生效。

（6）进行数据传输时，请遵循以下规则：

传输方式：为保证交易安全性，采用 HTTPS 传输。

提交方式：采用 POST 方法提交。

数据格式：提交和返回数据都为 XML 格式，根节点名为 xml。

字符编码：统一采用 UTF-8 字符编码。

签名算法：MD5，后续会兼容 SHA1、SHA256、HMAC 等。

签名要求：请求和接收数据均需要校验签名

证书要求：调用申请退款、撤销订单接口需要商户证书

判断逻辑：先判断协议字段返回，再判断业务返回，最后判断交易状态。

2. 使用官方示例实现支付功能

（1）创建一个 Android 项目，命名为“WeiXinPay”。

（2）在官网下载支付的 Demo，将 Demo 用开发工具打开。

（3）导入相关的 Jar 包到“app/libs”目录中。

（4）导入 .So 文件到“app/java/jniLibs”目录中。

（5）在 Constants 类中将 API_ID 换成自己申请的 APP_ID，关键代码如下：

```
public static final String APP_ID = "wxd930ea5d5a258f4f";  //更换成自己的ID
```

（6）AppRegister 广播类用于验证记录的 APPID，该类继承自 BroadcastReceiver，实现 onReceiver() 方法，关键代码如下：

```
01  final IWXAPI msgApi = WXAPIFactory.createWXAPI(context, null);
02  //支付平台记录AppId
03  msgApi.registerApp(Constants.APP_ID);
```

（7）支付的回调类为 PayActivity，支付后判断成功或者失败的代码均在此类中。

注意 由于官方数据不断更新，以上内容仅供参考，实际开发应以官方文档为主。

27.3.2 支付宝支付

支付宝是国内领先的第三方支付平台，具有简单、安全、快速的支付解决方法。手机支付宝 App 是支付宝官方推出的集手机支付和生活应用为一体的手机软件，通过加密传输、手机认证等安全保障体系，让用户随时可以通过该支付功能进行购物或手机充值、转账等功能。

1. 申请支付宝应用

（1）打开蚂蚁金服官方网站（https://openhome.alipay.com/platform/home.htm），注册账号。

（2）注册账号之后，进入开发者中心，在开发者中心单击创建项目，将显示创建名称应用页面，在该页面中输入应用名称，为“明日科技”，如图 27.29 所示。

图 27.29 创建应用

（3）添加好名称之后，单击“创建”按钮，可以看到如图 27.30 所示的界面，红框区域为以后要用的 APPID。

图 27.30 应用信息

（4）在进行支付宝开发时，除了需要 APPID 外还需要一些其他相关的信息，如图 27.31 所示。

图 27.31 设置相关信息

2. 使用官方示例实现支付功能

（1）创建一个 Android 项目，命名为“PayZhiFuBao”。

（2）在官网下载支付宝的 Demo，将 Demo 用开发工具 Android Stidio 打开。

（3）导入相关的 Jar 包到“app/libs”目录中

（4）导入 .So 文件到“app/java/jniLibs”目录中。

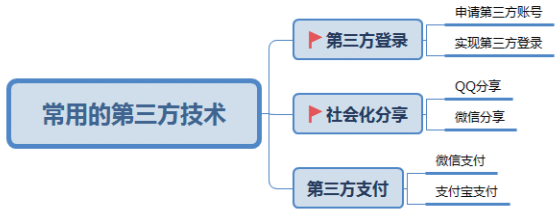
（5）其中最重要的类就是 PayDemoActivity，在该类中重点要找到以下代码，并根据注释进行设置。

```
01     //TODO/** 支付宝支付业务：入参app_id */
02     public static final String APPID = "";
03     //TODO/** 支付宝账户登录授权业务：入参pid值 */
04     public static final String PID = "";
05     //TODO/** 支付宝账户登录授权业务：入参target_id值 */
06     public static final String TARGET_ID = "";
07  //TODO/** 商户私钥，pkcs8格式 */
08  //TODO/** 如下私钥，RSA2_PRIVATE 或者 RSA_PRIVATE 只需要填入一个 */
09  //TODO/** 如果商户两个都设置了，优先使用 RSA2_PRIVATE */
10  //TODO/** RSA2_PRIVATE 可以保证商户交易在更加安全的环境下进行，建议使用 RSA2_PRIVATE */
11  //TODO/** 获取 RSA2_PRIVATE，建议使用支付宝提供的公私钥生成工具生成， */
12  //TODO/** 工具地址：*/
13  //TODO/**https://doc.open.alipay.com/docs/doc.
                                        htm?treeId=291&articleId=106097&docType=1 */
14     public static final String RSA2_PRIVATE = "";
15     public static final String RSA_PRIVATE = "";
16     private static final int SDK_PAY_FLAG = 1;
17     private static final int SDK_AUTH_FLAG = 2;
```

注意 由于官方数据不断更新，以上内容仅供参考，实际开发应以官方文档为主。

27.4 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。



第 28 章

发布你的 App

App 在上线以前还需要导出已签名的 APK 安装包并将 App 发布在应用商店中才可以供用户下载并使用。本章将介绍 App 如何发布在应用商店的整个流程。

28.1 导出 APK 安装包

通过 Android Studio 开发工具导出 APK 安装包时，通常有两种类型一种是 Debug 调试版本，另一种类型是 Release 发布版，调试版本可以设置断点、单步执行、使用 TRACE/ASSERT 等调试输出语句。而发布版不包含任何调试信息，所以发布版体积小运行速度要比调试版快。除了有两种类型的区别以外，在导出 APK 安装包的过程中还需要进行签名，目的是确保应用开发商的合法 ID，防止其他开发商使用相同包名来混淆替换已经安装的程序。

使用 Android Studio 导出 APK 安装包的具体步骤如下：

（1）在顶部的菜单栏中依次选择 Build → Generate Singned APK 将显示如图 28.1 所示，在该界面中选择需要生成 APK 安装包的模块名称，然后单击 Next 按钮将显示如图 28.2 所示的 APK 签名界面。在该界面中如果原来有密钥文件，可以单击“Choose existing…”按钮，选择已有的签名文件的路径，填写其他信息单击 Next 按钮，然后选择生成调试版或发布版既可在模块目录下找到对应生成的 APK 安装包。

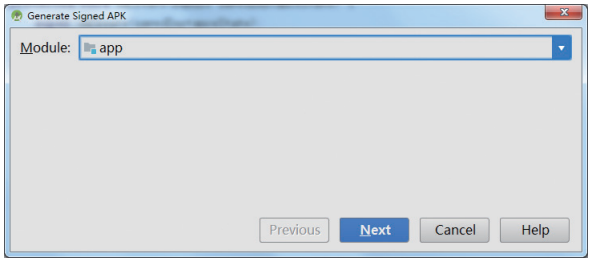


图 28.1　选择生成 APK 安装包的模块

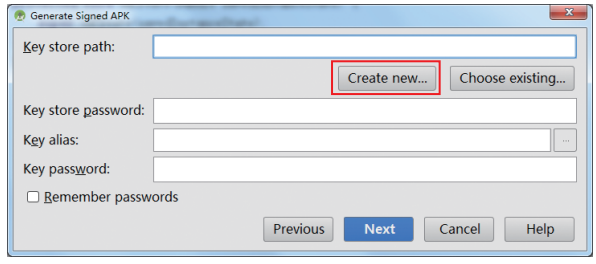


图 28.2　APK 签名界面

（2）如果第一次生成 APK 安装包并没有密钥文件，即可单击“Create new…”按钮，将显示如图 28.3 所示的界面。

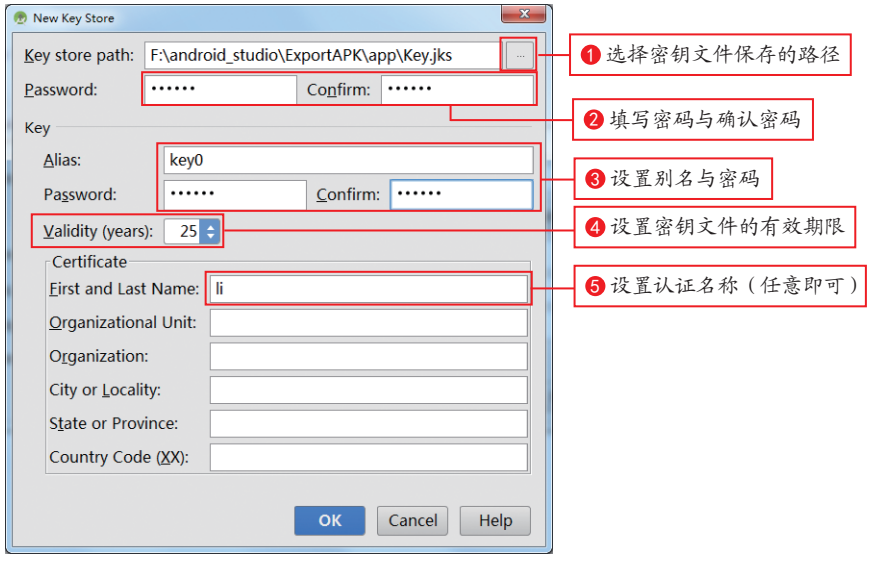


图 28.3　填写创建密钥的界面

（3）单击 OK 按钮将回到 APK 签名界面，并自动填写密码和别名填写如图 28.4 所示。单击 Next 按钮，将显示如图 28.5 所示的 APK 保存界面，在该界面中选择签名版本后单击 Finish 按钮，等待 Android Studio 生成 APK 即可。签名后的 APK 将被保存在指定的目录当中，如图 28.6 所示。

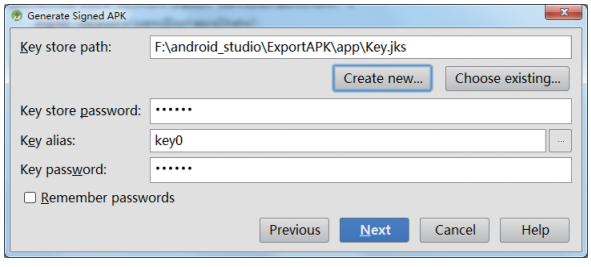


图 28.4　返回 APK 签名界面

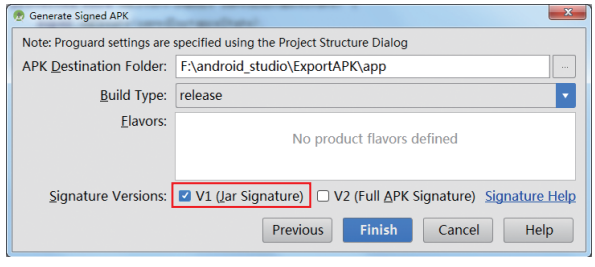


图 28.5　APK 保存界面

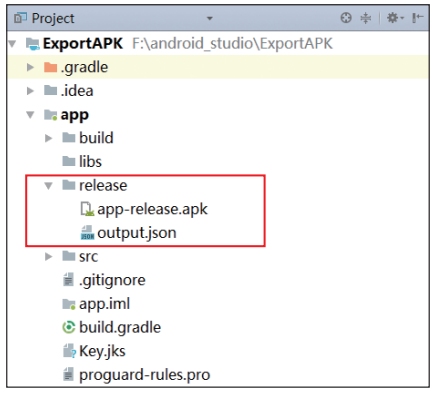


图 28.6　APK 保存位置

说明　为了让别人不能反编译自己所开发的应用程序，可以使用Android代码混淆。在Android Studio开发工具中实现代码混淆的步骤如下：

（1）在混淆配置文件 proguard-rules.pro 中配置混淆规则，具体规则请参考 ProGuard 官网（https://www.guardsquare.com）。

（2）在 buid.gradle（Module:app）文件中开启混淆开关，将 minifyEnabled 设置为 true，代码如下：

```
buildTypes {
    //release包配置
    release {
        //开启混淆开关
        minifyEnabled true
        //指定混淆文件（这里默认即可）
        proguardFiles getDefaultProguardFile
        ('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

（3）混淆后签名并导出 APK 安装包即可。

28.2 发布 App

导出已经签名的 APK 安装包以后，需要将 APK 发布到应用商店中，常用的应用商店有很多，例如腾讯的应用宝、360 手机助手、小米应用商店、华为应用商店等。在上传应用之前需要在指定的应用商店创建开发者账号并提交应用，然后应用商店会进行审核，审核完成后即可上线，最后用户才可以在已经上线的应用商店中下载并安装。

28.2.1 开发者实名认证

由于 App 应用商店较多，这里以华为应用商店为例，首先打开华为开发者联盟的官方网站（http://developer.huawei.com/consumer/cn）单击右上角注册将显示如图 28.7 所示的开发者账号注册页面。

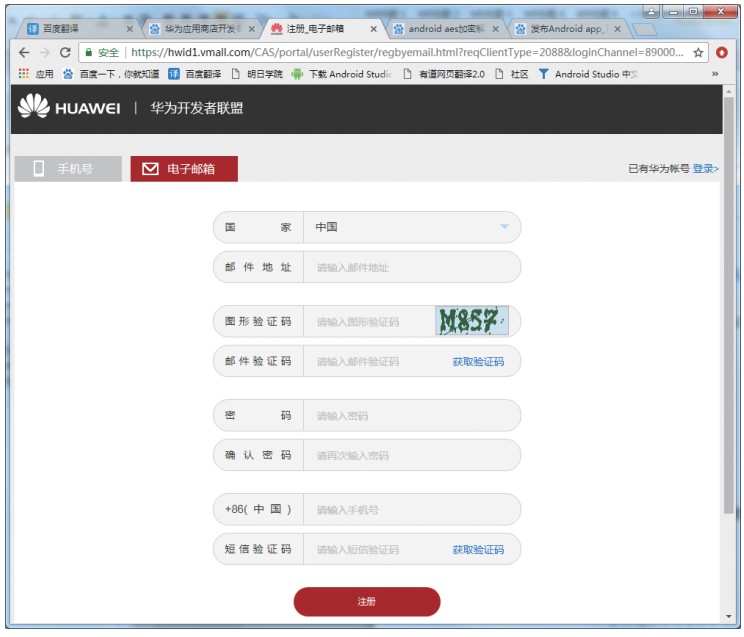


图 28.7 开发者账号注册页面

注册信息填写完成单击注册按钮即可登录华为应用市场，然后在顶部导航栏中依次选择分发→应用市场→上传应用，如图 28.8 所示。

图 28.8　上传应用

单击上传应用后将进行开发者实名认证，然后在认证页面中根据自己的需求进行选择，这里以个人开发者为例，如图 28.9 所示。

图 28.9　开发者实名认证

单击个人开发者，然后将显示选择验证方式页面。此页面有两种验证方式，分别是芝麻信用认证与人工认证，由于人工认证需要 1 ~ 2 个工作日。而芝麻信用认证操作简单认证速度较快，建议使用该认证方式进行认证，如图 28.10 所示。

图 28.10　选择开发者认证方式

认证完成后，网页将自动跳转到开发者实名认证页面，如图 28.11 所示，在该页面中填写必要的信息并接受相关协议后即可提交。

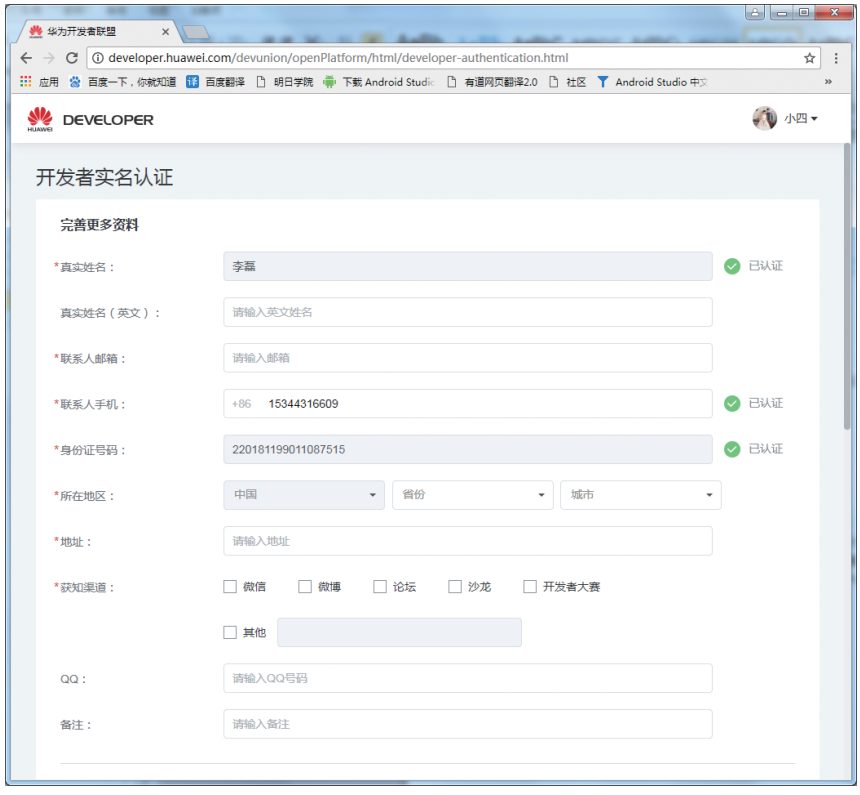


图 28.11　开发者实名认证

28.2.2 上传应用

完成开发者实名认证以后，再次回到主页面中单击上传应用将显示如图 28.12 所示的开发者联盟使用向导，根据该向导即可完成应用的上传，应用上传完成后等待审核即可。

图 28.12 开发者联盟使用向导

审核时间一般需要 1~3 个工作日，审核通过后会以邮件的方式进行提醒，此时登录华为开发者平台账号，在会员中心 → 我的产品中即可看到已经上架的应用记录，如图 28.13 所示，此时用户即可在华为应用市场中下载该应用。

图 28.13 查看上架应用记录

28.3 知识回顾

下面通过一个思维导图对本章所讲内容进行总结，如下图所示。