HZ BOOKS
华章科技

通过众多案例深入解读Android UI设计的方法和技巧，从实用角度出发，诠释以用户为中心的设计方法

以Google设计语言Material Design指导UI设计模式，轻松自信地设计和交付精美的移动App

Android UI设计

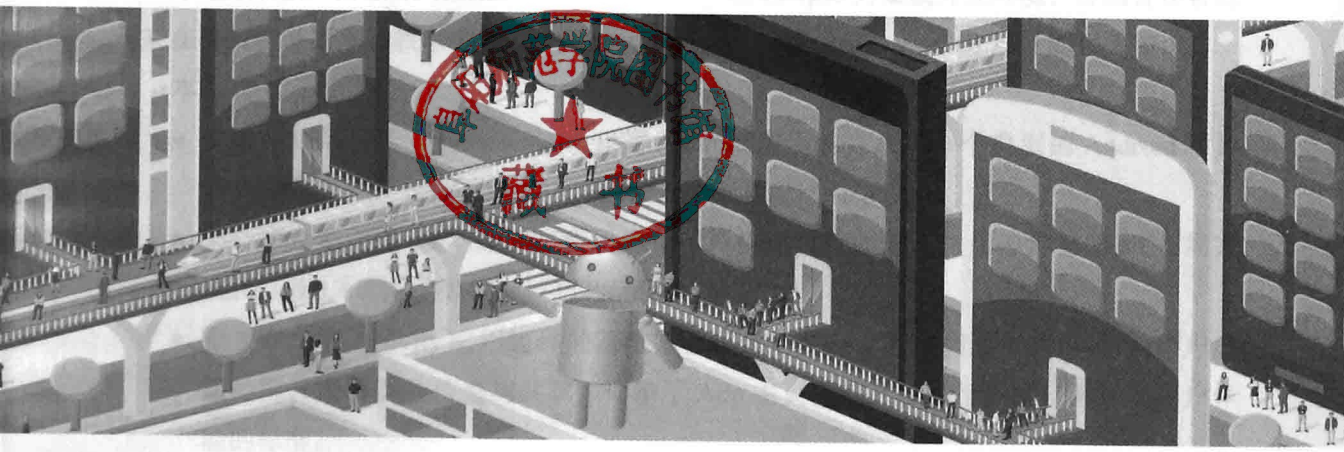
Android User Interface Design

李维勇 主编
杜亚杰 张以利 陈宇 参编

机械工业出版社
China Machine Press

Android UI设计

Android User Interface Design



李维勇 主编
杜亚杰 张以利 陈宇 参编

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Android UI 设计 / 李维勇主编 . —北京：机械工业出版社，2015.2
（UI / UE 系列丛书）

ISBN 978-7-111-48855-2

I. A… II. 李… III. 移动终端 – 应用程序 – 程序设计 IV. TN929.53

中国版本图书馆 CIP 数据核字（2014）第 295631 号

本书是基于 Android KitKat 平台进行移动应用开发的入门级教程，通过众多开源案例项目全面系统地介绍 Android UI 设计的方法、技巧和模式。

全书共 12 章，从 Android 应用设计者的角度系统讲解了从事 Android UI 设计必须要掌握的 Android 平台的所有技术和特性，主要内容包括可视化的 UI 设计与管理、常见 UI 控件设计与事件处理、UI 容器与导航设计、菜单与对话框设计、自定义控件设计、桌面 UI 设计、平板 UI 设计，以及主题样式和动画设计等，全面总结了 Android UI 的设计原理、设计理念和设计模式，最后通过一个综合的案例项目阐述 Android UI 设计的方法和技巧。

本书以案例贯穿全程，知识结构清晰，语言简洁，易于学习和提高，非常适合初学 Android UI 设计的在校大学生和希望系统掌握 Android UI 编程的开发人员阅读。

Android UI 设计

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：李 燕 责任校对：董纪丽
印 刷：北京市荣盛彩色印刷有限公司 版 次：2015 年 3 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：17.75
书 号：ISBN 978-7-111-48855-2 定 价：59.00 元

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

Preface 前　言

Android 是 Google 推出的一款广受移动应用软件开发者追捧的开源操作系统，近年来，Android 手机的市场占有率一直排名第一。

本书以 Android SDK KitKat 4.4 为开发平台，以 Eclipse 为集成开发环境，并结合作者近年来在手机软件研发和教学中积累的经验，详细介绍了 Android 平台 UI 设计的相关知识。

本书共 12 章。

- 第 1 章介绍基于 Eclipse + ADT 开发 Android 应用的方法及一个典型的 Android 项目的架构组成，并分析了移动 App 的设计原则和设计风格。
- 第 2 章介绍通过 ADT 插件实现图形化用户界面设计的方法、几种常见的 UI 布局方式，以及 UI 布局的原则、技巧和优化方法。
- 第 3 章介绍 Activity 应用组件的基础知识，包括创建、管理和退出 Activity，用户界面的跳转及数据的传递与共享，列举了 App 主页面的几种常见模式，阐述了用户体验的标准。
- 第 4 章主要介绍 Android 平台 Service、BroadcastReceiver 和 ContentProvider 等应用组件的核心知识，使用 Intent 在组件之间传递消息的机制，以及基于 Mashup 模式的应用模型。
- 第 5 章介绍常用表单控件的设计、适配器控件的设计，以及用户界面常见事件的触发与响应方法，分析了移动 App 表单 UI 的设计、大数据的加载模式以及提高搜索用户体验的方法。
- 第 6 章介绍 Toast、Notification 和 AlertDialog 这 3 种用户信息提示的方法，选项菜单和内容菜单的设计方法，以及动作栏和用户界面导航的设计，分析了用户通知设计的策略和原则。
- 第 7 章介绍常见容器 UI 的设计，包括导航类容器设计、特定容器设计，以及广泛使

用的第三方容器控件的设计，分析了用户引导页的设计技巧。

- 第 8 章介绍自定义控件设计的知识，包括定制一个基于 View 的控件、重构一个 View 子类，以及使用 Skia 绘制用户界面，并列举了几种常见的开源 UI 工具。
- 第 9 章介绍主题和样式的设计方法、系统主题资源的应用，以及设计帧动画、补间动画和属性动画的方法，分析了应用风格设计的 8 个技巧，阐述了用户界面动态设计的原则和技巧。
- 第 10 章介绍桌面 UI 设计方法，包括桌面组件的布局与属性描述、桌面组件的广播响应，以及基于集合的桌面应用组件的设计，并介绍了桌面组件的设计规范。
- 第 11 章介绍平板 UI 设计的知识，包括使用 Fragment 灵活构建 UI 界面的方法、管理 Fragment 之间的通信，以及设计平板设置界面的方法，分析了 Google 关于平板设计的原则和几种常见的平板布局模式。
- 第 12 章通过 Apollo 音乐播放器案例的用户界面设计，系统阐述了移动 App 开发中 UI 设计的知识、技巧和模式应用。

本书紧密结合初学者的学习习惯和认知规律，采用了大量简单而又实用的设计案例，使得读者在阅读时不会有障碍，并可通过简单的代码移植生成新的应用。书中采用的开源案例项目把与 Android 开发相关的技术和设计完美结合，别具一格，弥补了 Android 设计人员知识的不足。

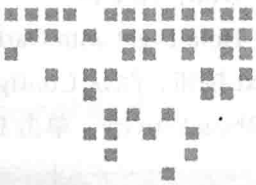
本书由李维勇担任主编，杜亚杰、张以利、陈宇参与编写。南京信息职业技术学院软件学院移动互联网应用技术教研室全体同仁共同参与了本书的校对和文稿的审核。本书的编写得到了南京信息职业技术学院、南京工业职业技术学院、南京审计学院金审学院等兄弟院校的大力支持和帮助，上海尚强信息科技有限公司对教材案例项目的策划、开发和测试提供了大量信息，机械工业出版社的编辑为本书的策划和出版提供了宝贵的经验和支持，在此表示衷心感谢。同时，本书在编写过程中参考了大量的相关资料，吸取了许多同仁的宝贵经验，在此一并致谢。

由于作者水平有限，难免存在疏漏，恳请广大读者批评指正，并欢迎提出宝贵意见和建议。另本书的配套课件、习题答案及源代码均可从华章公司网站（www.hzbook.com）下载。

作者

2014 年 12 月

Contents 目　录



第 1 章 Chapter 1

Android 开发基础

1.1 Eclipse 中的 Android 开发

Eclipse 是著名的跨平台开源集成开发环境，对开发 Android 应用提供了良好的支持。

1.1.1 创建项目

在 Eclipse 中创建 Android 项目的步骤如下：

①启动 Eclipse 集成开发环境[⊖]。

②运行 File → New → Android Application Project 菜单命令，打开 New Android Application 向导，显示如图 1-1 所示界面。

在 New Android Application 向导中输入如下信息：

- Application Name：HelloWorld
- Project Name：HelloWorld
- Package Name：com.liweiyong.helloworld（包的名称必须和所有安装在 Android 系统中的应用程序的包名不相同）

其他默认选择如下：

- Minimum Required SDK：API 14
- Target SDK：API 18

⊖ 如果 Eclipse 没有安装 ADT 插件，请参阅官方指导文档 http://developer.android.com/sdk/installing/installing-adt.html。

- Compile With：API 19
- Theme：Holo Light with Dark Action Bar

③单击 Next 按钮，默认 Configure Project 设置和 Configure the attribute of the icon set 界面设置，选择 BlankActivity，单击 Finish 按钮，完成 Hello World 项目的创建。

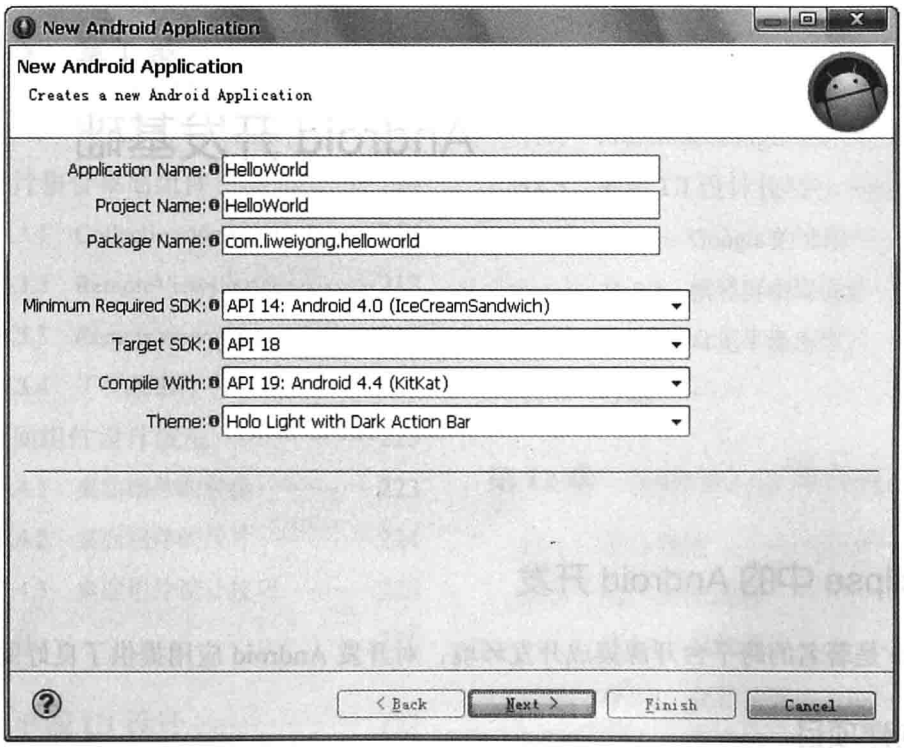


图 1-1 New Android Application 向导

1.1.2 创建 AVD

Android Virtual Device（简称 AVD）是运行 Android 项目的虚拟设备。AVD 通过对硬件和软件的配置进行定义来模拟一个实际的设备。

1. 创建 SD Card 映像文件

Android 模拟器自身已经具备了一个持久化存储空间，但这并不够大，有时需要为应用程序和文件提供更大的存储空间。为了在模拟器上开发使用扩展存储空间的程序，需要在 PC 上模拟一个 SD Card（Secure Digital Memory Card，一种基于半导体快闪记忆器的新一代记忆设备）的虚拟文件，然后加载到模拟器中。

创建 SD Card 虚拟文件的步骤如下：

①在 Windows 中，运行“开始”→“运行”菜单命令，在打开的“运行”窗口中输入“cmd”并单击“确定”按钮，打开命令行窗口。

②在窗口中输入如下命令：

```
mksdcard -l mycard 500M C:\\mysdcard.img
```

该命令的含义是在本地磁盘 C 盘创建一个 500 MB 大小的映像文件 mysdcard.img。此时，查看 C 盘可以看到一个名称为 mysdcard.img 的文件。

2. 创建 AVD 并关联 SD Card

在 Eclipse 中创建 AVD 并关联 SD Card 的步骤如下：

①在 Eclipse 中，运行 Window → Android Virtual Device Manager 菜单命令，打开 Android Virtual Device Manager 对话框。单击对话框中的 New 按钮，弹出如图 1-2 所示的创建 AVD 对话框。

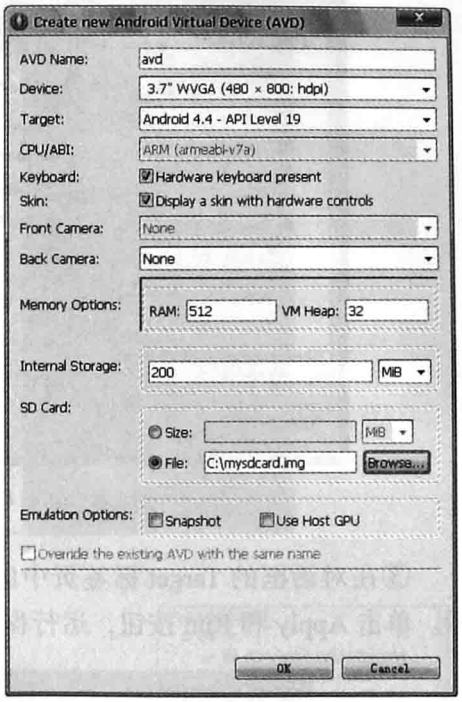


图 1-2　创建 AVD 对话框

- 在 AVD Name 文本框中输入 AVD 的名称（可以自定义）。
- Device 用于设置模拟器的尺寸和分辨率。SDK 提供的常见分辨率（sdk\platforms\android-#\skins\）包括：HVGA（320×480）、QVGA（320×240）、WQVGA400（400×240）、WQVGA432（432×240）、WVGA800（800×480）、WVGA854（854×480）和 WXGA（1280×800）等。
- 在 Target 中选择需要的 SDK 版本（平板电脑开发的最低版本是 Android API 11）。
- SD Card 的大小可以自定义输入数值；也可以选择 File 单选按钮后，单击 Browse 按钮，在打开的对话框中选择前面创建的 SD Card 映像文件。
- 其他默认设置即可。

②单击 OK 按钮，完成 AVD 的创建。

1.1.3　运行项目

下面介绍在 Eclipse 中运行 Android 项目的方法，步骤如下：

①在 Eclipse 的项目窗口中，右击项目节点名称 HelloWorld，运行 Run as → Run Configurations 菜单命令，打开 Run Configurations 对话框。

②在对话框的左侧选择 Android Application，并单击上方的 New launch configuration 按钮，在右侧的 Android 标签页中单击 Browse 按钮，打开 Project Selection 对话框，选择 HelloWorld 项目，如图 1-3 所示。

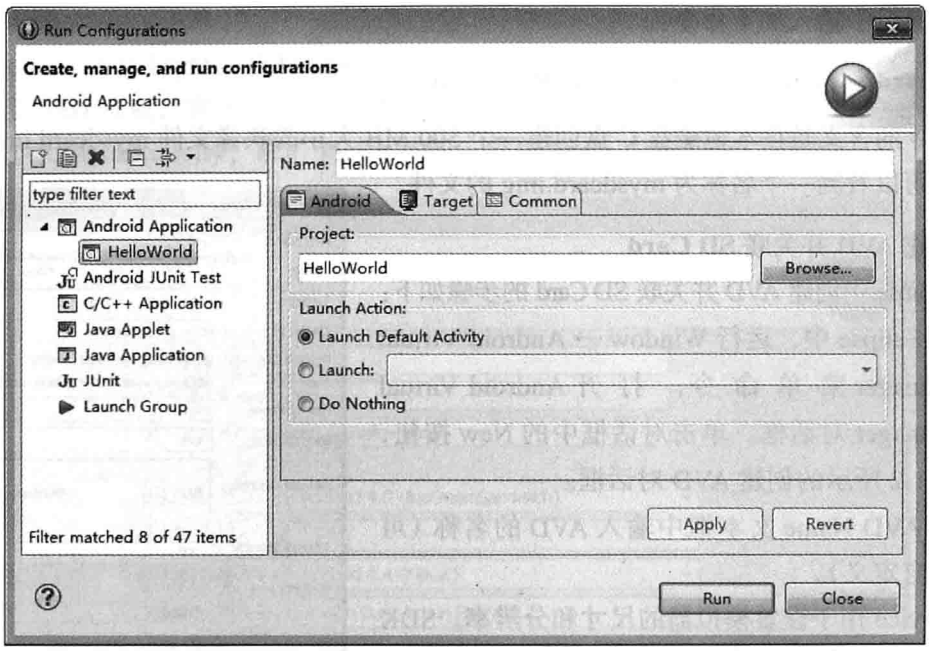


图 1-3 Run Configurations 对话框——选择项目

③在对话框的 Target 标签页中的 AVD 列表中勾选合适的 Android 模拟器，如图 1-4 所示。单击 Apply 和 Run 按钮，运行程序[⊖]。

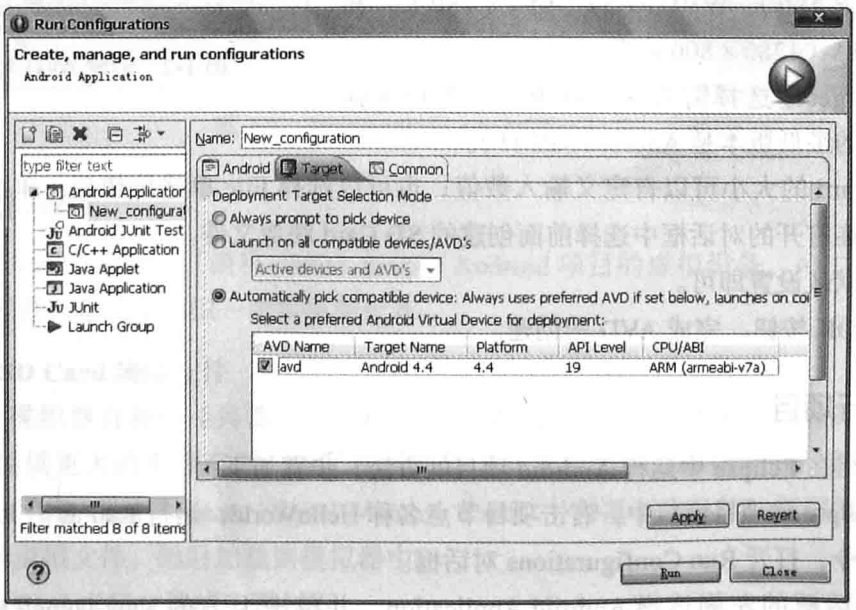


图 1-4 Run Configurations 对话框

⊖ 如果需要对 AVD 进行缩放，特别是使用平板模拟器时，可以在 Run Configurations 对话框的 Additional Emulator Command Line Options 中输入类似“-scale 0.8”这样的命令，这样模拟器的显示尺寸将缩小到 80%。

图 1-5 显示了项目的运行结果。

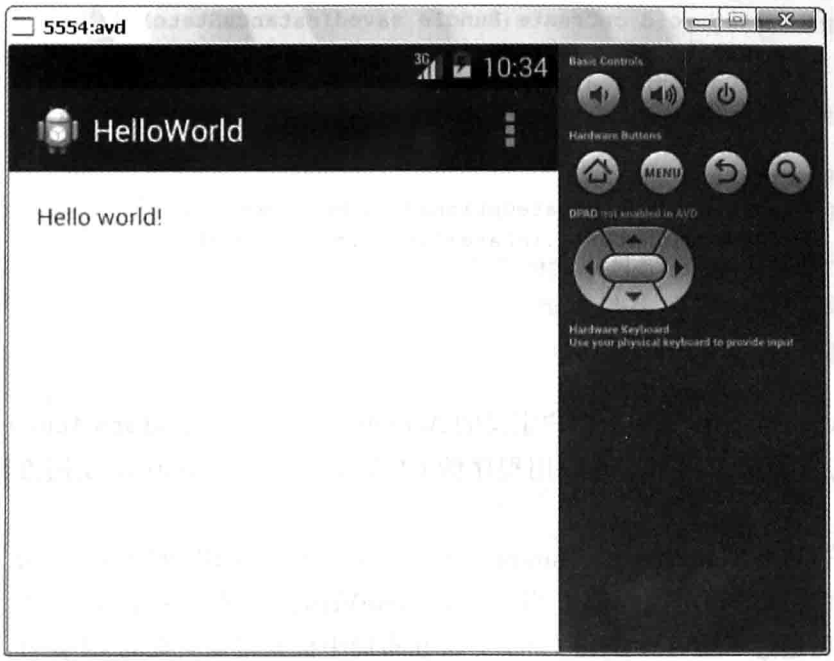


图 1-5　项目运行结果

1.2　Android 项目架构

在 Eclipse 的 Package Explorer 窗口中展开 HelloWorld 项目，项目架构如图 1-6 所示。

一个 Android 工程包含了组成 Android 应用的所有源代码文件。Android 工程主要由 src、gen、assets、res 文件夹和 AndroidManifest.xml 等文件组成，下面分别对其进行解析。

图 1-6　HelloWorld 项目架构

1.2.1　Java 代码解析

Android 项目的 Java 代码主要存放在 src 文件夹和 gen 文件夹的包文件夹下。

1. MainActivity 解析

src 文件夹用来存放项目的源代码。

在 Package Explorer 窗口中，展开项目的 src 文件夹，打开通过向导生成的 MainActivity.java，核心代码如下：

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

```
02
03      @Override
04      protected void onCreate(Bundle savedInstanceState) {
05          super.onCreate(savedInstanceState);
06          setContentView(R.layout.activity_main);
07      }
08
09      @Override
10      public boolean onCreateOptionsMenu(Menu menu) {
11          getMenuInflater().inflate(R.menu.main, menu);
12          return true;
13      }
14
15  }
```

01 行的 MainActivity 是一个用户定义的 Activity，继承自 android.app.Activity 类。Activity 是应用程序的表示层，用于构建应用程序的 UI 界面。关于 Activity 的使用方法将在第 3 章介绍。

03 ~ 07 行重载的 onCreate（Bundle）方法是 Activity 生命周期的组成部分，用于初始化 Activity。例如界面的显示内容通过调用 setContentView() 方法来指定显示布局，如 activity_main.xml。然后通过 findViewById（int）方法在布局中检索需要交互的 UI 控件。

04 行的 Bundle 类用于 Activity 之间传递数据。该类提供了公有方法 containKey（String key），如果给定的 key 包含在 Bundle 的映射中，则返回 true，否则返回 false。该类实现了 Parceable 和 Cloneable 接口，所以它具有这两者的特性。

05 行的 super.onCreate（savedInstanceState）方法的作用是调用父类中的 onCreate() 方法来实现对界面的绘制工作。注意，从 savedInstanceState 中读取保存到存储设备中的数据时，需要判断 savedInstanceState 是否为 null，因为 Activity 第一次启动时并没有数据被存储在设备中。有关 savedInstanceState 的介绍参见 3.4.4 节。

09 ~ 13 行的 onCreateOptionsMenu（Menu）方法是 Activity 中提供的用于创建菜单项的回调方法。通过其中的 getMenuInflater().inflate（int，Menu）方法加载 res/menu 中定义的菜单资源。onCreateOptionsMenu（Menu）方法通常和 OnOptionsMenuSelected（MenuItem）回调方法配合使用，以响应菜单的选择事件。

2. R. java 解析

打开 gen/com.liweiyong.helloworld 中的 R.java 文件，部分代码如下：

```
01  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
02   *
03   * This class was automatically generated by the
04   * aapt tool from the resource data it found.  It
05   * should not be modified by hand.
06   */
07
08  public final class R {
```

```
09      public static final class array {
10          public static final int action_file=0x7f040005;
11          ...
12      }
13      public static final class layout {
14          public static final int activity_main=0x7f030000;
15      }
16      ...
17  }
```

程序的第 01 ~ 06 行为注释说明，R.java 文件由 aapt 工具根据 res 中的资源自动生成，不要手动修改该文件。R.java 由 ADT 根据 res 中的资源自动生成 drawable、layout、string 等静态匿名内部类。不同的静态内部类分别根据其 res 中的资源定义一系列资源标识符，如“public static final int activity_main =0x7f030000;”对应的是 layout 目录下的 activity_main.xml 文件。

每当 res 中的资源发生变化，aapt 工具都会自动在 R.java 对应的内部类中生成一个静态 int 类型的常量，以对新添加的资源进行索引。如果从 res 中删除一个资源，R.java 中对应的索引也会自动删除。

通过 R.java 可以很快地查找需要的资源，另外编译器也会检查 R.java 列表中的资源是否被使用，没有被使用到的资源不会编译进 apk 中，这样可以减少应用在手机中占用的空间。

3. BuildConfig. java 解析

打开 gen/com.liweiyong.helloworld 中的 BuildConfig.java 文件，代码如下：

```
01  /** Automatically generated file. DO NOT MODIFY */
02  package cn.liweiyong.helloworld;
03
04  public final class BuildConfig {
05      public final static boolean DEBUG = true;
06  }
```

ADT 允许开发者只在调试模式下运行某些代码。BuildConfig 类包含一个 DEBUG 常量，该常量会根据 Build 类型自动设置值。可以通过 BuildConfig.DEBUG 常量来编写只在 Debug 模式下运行的代码。如果有些代码不想在应用发布后执行，也可以使用该功能。

例如，在调试日志时，不想在软件发布后被其他开发者看到，过去的方式是设置一个全局变量，标记软件为调试模式还是发布模式。现在，有了 BuildConfig.DEBUG 之后，在代码中可以直接写入：

```
01  if (BuildConfig.DEBUG) {
02      Log.d(TAG, "output something");
03  }
```

在发布前，BuildConfig.DEBUG 的值自动为 true。当通过 Android Tools → Export Signed Application Package 命令发布包时，BuildConfig.DEBUG 的值自动变为 false。

1.2.2 项目资源解析

Android 项目的资源组织在 res 文件夹中，资源包括 drawable、layout、values、anim、xml、raw、color 以及 menu 等。下面分别对最常用的 drawable、layout、values 资源的定义和使用进行解析。

1. drawable 解析

drawable 用于存放图片文件资源，或者能被编译为 drawable 类型的 XML 文件。包括 drawable-hdpi、drawable-mdpi、drawable-ldpi 和 drawable-xhdpi 等多个存放图片的文件夹。这些文件夹的作用如下：

- drawable-hdpi 里面存放高分辨率[㊀]的图片，如 WVGA(480×800)、FWVGA(480×854)。
- drawable-mdpi 里面存放中等分辨率的图片，如 HVGA（320×480）。
- drawable-ldpi 里面存放低分辨率的图片，如 QVGA（240×320）。
- drawable-xhdpi 里面存放至少 960×720 分辨率的图片。

在设计应用的图标时，对于五种主流的像素密度（MDPI、HDPI、XHDPI、XXHDPI 和 XXXHDPI）应按照 2∶3∶4∶6∶8 的比例进行缩放。例如，一个启动图标的尺寸为 48×48dp[㊁]，这表示在 MDPI 的屏幕上其实际尺寸应为 48×48px[㊂]，在 HDPI 的屏幕上其实际

㊀ 可以通过如下的方法获取屏幕分辨率信息：
getWindowManager().getDefaultDisplay().getMetrics(new DisplayMetrics());

㊁ dp（即 dip，设备独立像素）是 Android 布局时最常用的尺寸单位，它与像素密度（dpi）密切相关。简单地说，dp 就是一种基本上和设备无关的单位，它可以保证一套 UI 在不同机器上面的适配，而显示效果不会出现很大的偏差。例如，可以使得同一个图片在不同分辨率下的屏幕上保持基本相同的物理大小。dp 与 px 之间的关系可以通过下面的工具类转换：

```
public class DensityUtil {

    /**
     * 根据手机的分辨率从 dp 的单位转成为 px(像素)
     */
    public static int dip2px(Context context, float dpValue) {
        final float scale = context.getResources().getDisplayMetrics().density;
        return (int) (dpValue * scale + 0.5f);
    }

    /**
     * 根据手机的分辨率从 px(像素) 的单位转成为 dp
     */
    public static int px2dip(Context context, float pxValue) {
        final float scale = context.getResources().getDisplayMetrics().density;
        return (int) (pxValue / scale + 0.5f);
    }
}
```

㊂ px 即像素，1px 代表屏幕上一个物理的像素点。px 单位不被建议使用，因为同样 px 的图片，在不同手机上显示的实际大小可能不同。