

TURING

图灵原創



第一行代码

Android

第 2 版

郭霖 ◎ 著

人民邮电出版社
北京

图书在版编目 (C I P) 数据

第一行代码——Android / 郭霖著. -- 2版. -- 北京 : 人民邮电出版社, 2016.12
(图灵原创)
ISBN 978-7-115-43978-9

I. ①第… II. ①郭… III. ①移动终端—应用程序—程序设计 IV. ①TN929.53

中国版本图书馆CIP数据核字(2016)第267736号

内 容 提 要

本书被广大 Android 开发者誉为“Android 学习第一书”。全书系统全面、循序渐进地介绍了 Android 软件开发的必备知识、经验和技巧。

第 2 版基于 Android 7.0 对第 1 版进行了全面更新，将所有知识点都在最新的 Android 系统上进行重新适配，使用全新的 Android Studio 开发工具代替之前的 Eclipse，并添加了对 Material Design、运行时权限、Gradle、RecyclerView、百分比布局、OkHttp、Lambda 表达式等全新知识点的详细讲解。

本书内容通俗易懂，由浅入深，既是 Android 初学者的入门必备，也是 Android 开发者的进阶首选。

-
- ◆ 著 郭霖
 - 责任编辑 王军花
 - 执行编辑 张霞
 - 责任印制 彭志环
 - 封面插画 巫俊武
 - 封面设计 潘建永 陈冰
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京鑫正大印刷有限公司印刷
 - ◆ 开本：800×1000 1/16
印张：36.25
字数：856千字 2016年12月第2版
印数：89 001~99 000册 2016年12月北京第2次印刷
-

定价：79.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广字第 8052 号

前　　言

虽然我从事Android开发工作已经很多年了，但是之前从来没有想过自己要去写一本Android技术相关的书。在我看来，写一本书可以算是一个很庞大的工程，写一本好书的难度并不亚于开发一款好的应用程序。

由于我长期坚持在CSDN上发表技术博文，因而得到了大量网友的认可，也积累了一定的名气。很荣幸的是，人民邮电出版社图灵公司的前副总编辑陈冰老师联系上了我，希望我可以写一本关于Android开发技术的书，这着实让我受宠若惊。

在写本书第1版的时候，我可以说是费了相当大的心思。写书和写博客最大的区别在于，书的内容不能像博客那样散乱，不能想到哪里写到哪里，而是一定要系统化，要循序渐进，基本上在写第1章的时候就应该把全书的内容都确定下来了。

令我非常欣慰的是，本书的第1版在推出之后获得了广大读者的强烈认可，在短短两年时间内，已经成为了国内最畅销的Android技术书。各大书店、图书馆都能看到《第一行代码》的身影，许多学校和培训机构也纷纷将《第一行代码》选为Android课程的教材。

不过，在科技高速发展的今天，各种技术的发展都是日新月异的。在两年的时间里，Android操作系统经历了5.0、6.0、7.0的飞速升级。不可否认的是，本书第1版中的不少知识点都已经过时，而且这两年间出现的很多新知识，第1版中也没有涵盖。因此，这让我坚定了写作本书第2版的想法。

刚开始写的时候，我以为只是小修小补，但事实上并没有我想象得那么轻松。除了介绍新知识点以外，书中之前的所有项目都需要重新编写和测试，以保证代码在新老系统上的兼容性。另外，由于Android从5.0系统开始，UI风格变化很大，因此第1版中所有的截图都需要更新。毫不夸张地说，我几乎重写了整本书。

而现在，你手中捧着的正是全新版的《第一行代码》，同时这也是国内第一本基于Android 7.0系统写作的技术书。我真诚地希望你可以用心去阅读这本书，因为每多掌握一份知识，你就会多一份喜悦。Enjoy it!

第2版的变化

由于第2版修改内容繁多，因此这里我只列举出最主要的变化。首先是开发工具上的改变，本书第1版使用的开发工具是Eclipse，而第2版使用了目前最新的Android Studio 2.2版本。另外，

本书第1版是基于Android 4.x系统的，而第2版是基于Android 7.0系统的，其中囊括了新系统中的诸多知识点，包括Android 5.0系统中引入的Material Design、Android 6.0系统中引入的运行时权限和Doze模式、Android 7.0系统中引入的多窗口模式等。

除此之外，第2版还加入了Gradle、RecyclerView、百分比布局、OkHttp、Lambda表达式等全新知识点的讲解，内容将前所未有地充实。

读者对象

本书内容通俗易懂，由浅入深，既适合初学者阅读，也同样适合专业人员。学习本书内容之前，你并不需要有任何的Android基础，但是你需要有一定的Java基础，因为Android开发都是使用Java语言的，而本书并不会去专门介绍Java方面的知识。

阅读本书时，你可以根据自身的情况来决定如何阅读。如果你是初学者的话，建议你从第1章开始循序渐进地阅读，这样理解起来就不会感到吃力。而如果你已经有了一定的Android基础，那么就可以选择某些你感兴趣的章节进行跳跃式的阅读。但请记住，很多章最后的最佳实践部分一定是你不想错过的。

本书内容

正如前面所说，本书的内容是非常系统化的，不仅全面介绍了那些你必须掌握的知识，而且保证了各章的难度都是梯度式上升的。全书一共分为15章，涵盖了四大组件、UI、碎片、数据存储、多媒体、网络、定位服务等方方面面的知识。为了让你在学完所有内容之后还可以有综合运用的能力，本书的尾声部分还会带你一起开发一个天气预报程序，并教会你如何将程序发布到应用商店，以及如何在程序中嵌入广告盈利。

除此之外，本书的第5章、第7章、第11章、第14章中都穿插有对Git的讲解，如果想要掌握它的用法，这几章的内容是绝对不能错过的。

本书中各个章节的内容都相对比较独立，因此除了可以循序渐进地学习之外，你还可以把它当成一本参考手册，随时查阅。

源码下载

首先，我建议你在学习本书的时候将所有项目的源码都亲手敲上一遍，因为只有这样才能加深你对代码的理解。不过为了方便于你的学习，我还是提供了书中所有项目的源码，请仅在需要的时候再去参考（如下载项目中的图片资源）。切勿直接将源码复制粘贴就当成自己的东西了，只有亲手敲过的代码才真正是你自己的。

源码下载地址：<https://github.com/guolindev/booksource>。

致 谢

在这近一年的时间里，我又完成了一项浩大的工程。和写作本书第1版时的感觉类似，当全书完稿之后，回顾整本书，我仍然不敢相信这所有的内容竟然是我一字字地敲出来的。

如今这已经是我写的第二本书了，和写第一本书时的情况不同，现在我有了更广的人脉和资源，有了更多的人愿意帮助和支持我来完成一本更好的技术书。因此，我要在这里对很多人表示感谢。

首先我要感谢我的父母，感谢你们将我抚养长大，感谢你们的付出，让我从小不用为生计、上学而发愁，可以一直做我自己想做的事情，也感谢你们指引我走上了技术这条路。

其次我要感谢我的妻子，感谢你每天为我准备好一日三餐，感谢你对我永远的包容，不管是平日的加班还是没日没夜的写书，你都一直默默地理解和支持我。

我还非常感谢本书第1版的编辑陈冰老师，如果没有你当初在CSDN上找到我，并邀请我写书，就不会有现在的《第一行代码》。另外，你也是当时唯一一个坚信这本书一定会大卖的人，甚至连我自己当时都没有如此的眼光。

我也非常感谢本书第2版的编辑张霞，你全程负责了第2版的出版工作，并且完成得非常出色。你对文字的把控能力让我敬佩，感谢你对书中每一章节的尽心审阅，才能让这本书更趋近于完美。

另外我还要特别感谢一部分人，你们对本书的试读、内容建议、勘误检查、代码纠错，甚至是对我个人的支持等都作出了卓越的贡献。有了你们的帮助，才会有这样一本更加出色的书呈现在所有人面前，这本书上也理应有你们的名字（按姓氏拼音排序，排名不分先后）：

陈建林 陈俊杰 陈雷 陈龙 陈琪 陈秀相 陈逸鸣 代云蛟 董霖轩 段郭森
高鹤泉 高太稳 关爱民 何以诚 胡恩泽 黄楠 赖帆 李济州 李建友 李沛明
李潭 李永鹏 李志云 林火荣 刘萌 刘明渊 刘治国 陆德俊 罗亚超 吕国鑫
马文杰 覃文斌 孙建飞 王柏强 王光东 王杰 王龙 王路路 王鹏 王荣宗
王善昌 韦振南 吴波 吴宏权 吴绍志 徐阳 轩仲宽 杨辉 易静杰 查童
张鸿洋 张英祥 赵翠龙 赵庆元 赵迎超 郑传书 郑敏馨 庄育锋 周苏 朱海丰

目 录

第1章 开始启程——你的第一行

Android 代码	1
1.1 了解全貌——Android 王国简介	2
1.1.1 Android 系统架构	2
1.1.2 Android 已发布的版本	3
1.1.3 Android 应用开发特色	4
1.2 手把手带你搭建开发环境	5
1.2.1 准备所需要的工具	5
1.2.2 搭建开发环境	5
1.3 创建你的第一个 Android 项目	9
1.3.1 创建 HelloWorld 项目	9
1.3.2 启动模拟器	12
1.3.3 运行 HelloWorld	15
1.3.4 分析你的第一个 Android 程序	16
1.3.5 详解项目中的资源	22
1.3.6 详解 build.gradle 文件	23
1.4 前行必备——掌握日志工具的使用	26
1.4.1 使用 Android 的日志工具 Log	26
1.4.2 为什么使用 Log 而不使用 System.out	27
1.5 小结与点评	29

第2章 先从看得到的入手——探究

活动	30
2.1 活动是什么	30
2.2 活动的基本用法	30
2.2.1 手动创建活动	31
2.2.2 创建和加载布局	32

2.2.3 在 AndroidManifest 文件中 注册	35
2.2.4 在活动中使用 Toast	37
2.2.5 在活动中使用 Menu	38
2.2.6 销毁一个活动	40
2.3 使用 Intent 在活动之间穿梭	41
2.3.1 使用显式 Intent	41
2.3.2 使用隐式 Intent	44
2.3.3 更多隐式 Intent 的用法	46
2.3.4 向下一个活动传递数据	50
2.3.5 返回数据给上一个活动	51
2.4 活动的生命周期	53
2.4.1 返回栈	53
2.4.2 活动状态	54
2.4.3 活动的生存期	55
2.4.4 体验活动的生命周期	56
2.4.5 活动被回收了怎么办	62
2.5 活动的启动模式	63
2.5.1 standard	64
2.5.2 singleTop	65
2.5.3 singleTask	67
2.5.4 singleInstance	68
2.6 活动的最佳实践	71
2.6.1 知晓当前是在哪一个活动	71
2.6.2 随时随地退出程序	72
2.6.3 启动活动的最佳写法	74
2.7 小结与点评	75

第3章 软件也要拼脸蛋——UI 开发的点点滴滴	76	第4章 手机平板要兼顾——探究碎片	142
3.1 如何编写程序界面	76	4.1 碎片是什么	142
3.2 常用控件的使用方法	77	4.2 碎片的使用方式	144
3.2.1 TextView	77	4.2.1 碎片的简单用法	144
3.2.2 Button	80	4.2.2 动态添加碎片	147
3.2.3 EditText	82	4.2.3 在碎片中模拟返回栈	150
3.2.4 ImageView	86	4.2.4 碎片和活动之间进行通信	151
3.2.5 ProgressBar	88	4.3 碎片的生命周期	151
3.2.6 AlertDialog	91	4.3.1 碎片的状态和回调	151
3.2.7 ProgressDialog	93	4.3.2 体验碎片的生命周期	153
3.3 详解 4 种基本布局	94	4.4 动态加载布局的技巧	156
3.3.1 线性布局	94	4.4.1 使用限定符	156
3.3.2 相对布局	100	4.4.2 使用最小宽度限定符	159
3.3.3 帧布局	103	4.5 碎片的最佳实践——一个简易版的新闻应用	160
3.3.4 百分比布局	105	4.6 小结与点评	169
3.4 系统控件不够用？创建自定义控件	108		
3.4.1 引入布局	109		
3.4.2 创建自定义控件	111		
3.5 最常用和最难用的控件——ListView	113		
3.5.1 ListView 的简单用法	114		
3.5.2 定制 ListView 的界面	115		
3.5.3 提升 ListView 的运行效率	119		
3.5.4 ListView 的点击事件	120		
3.6 更强大的滚动控件——RecyclerView	122		
3.6.1 RecyclerView 的基本用法	122		
3.6.2 实现横向滚动和瀑布流布局	125		
3.6.3 RecyclerView 的点击事件	130		
3.7 编写界面的最佳实践	132		
3.7.1 制作 Nine-Patch 图片	132		
3.7.2 编写精美的聊天界面	135		
3.8 小结与点评	141		
第5章 全局大喇叭——详解广播机制	170		
5.1 广播机制简介	170		
5.2 接收系统广播	171		
5.2.1 动态注册监听网络变化	171		
5.2.2 静态注册实现开机启动	174		
5.3 发送自定义广播	177		
5.3.1 发送标准广播	177		
5.3.2 发送有序广播	179		
5.4 使用本地广播	183		
5.5 广播的最佳实践——实现强制下线功能	185		
5.6 Git 时间——初识版本控制工具	192		
5.6.1 安装 Git	192		
5.6.2 创建代码仓库	193		
5.6.3 提交本地代码	195		
5.7 小结与点评	195		

第6章 数据存储全方案——详解

持久化技术	196
6.1 持久化技术简介	196
6.2 文件存储	197
6.2.1 将数据存储到文件中	197
6.2.2 从文件中读取数据	201
6.3 SharedPreferences 存储	203
6.3.1 将数据存储到 SharedPreferences 中	203
6.3.2 从 SharedPreferences 中读取数据	206
6.3.3 实现记住密码功能	208
6.4 SQLite 数据库存储	211
6.4.1 创建数据库	211
6.4.2 升级数据库	216
6.4.3 添加数据	219
6.4.4 更新数据	222
6.4.5 删除数据	224
6.4.6 查询数据	225
6.4.7 使用 SQL 操作数据库	228
6.5 使用 LitePal 操作数据库	229
6.5.1 LitePal 简介	229
6.5.2 配置 LitePal	230
6.5.3 创建和升级数据库	231
6.5.4 使用 LitePal 添加数据	236
6.5.5 使用 LitePal 更新数据	237
6.5.6 使用 LitePal 删除数据	240
6.5.7 使用 LitePal 查询数据	241
6.6 小结与点评	243

第7章 跨程序共享数据——探究

内容提供器	244
7.1 内容提供器简介	244
7.2 运行时权限	245
7.2.1 Android 权限机制详解	245
7.2.2 在程序运行时申请权限	249

7.3 访问其他程序中的数据

7.3.1 ContentResolver 的基本用法	254
7.3.2 读取系统联系人	256
7.4 创建自己的内容提供器	260
7.4.1 创建内容提供器的步骤	261
7.4.2 实现跨程序数据共享	265
7.5 Git 时间——版本控制工具进阶	275
7.5.1 忽略文件	275
7.5.2 查看修改内容	276
7.5.3 撤销未提交的修改	278
7.5.4 查看提交记录	279
7.6 小结与点评	280

第8章 丰富你的程序——运用手机**多媒体**

8.1 将程序运行到手机上	281
8.2 使用通知	283
8.2.1 通知的基本用法	283
8.2.2 通知的进阶技巧	289
8.2.3 通知的高级功能	291
8.3 调用摄像头和相册	293
8.3.1 调用摄像头拍照	294
8.3.2 从相册中选择照片	298
8.4 播放多媒体文件	303
8.4.1 播放音频	303
8.4.2 播放视频	307
8.5 小结与点评	311

第9章 看看精彩的世界——使用**网络技术**

9.1 WebView 的用法	312
9.2 使用 HTTP 协议访问网络	314
9.2.1 使用 HttpURLConnection	315
9.2.2 使用 OkHttp	319
9.3 解析 XML 格式数据	321
9.3.1 Pull 解析方式	324

9.3.2 SAX 解析方式.....	326	11.4 使用百度地图.....	395
9.4 解析 JSON 格式数据.....	329	11.4.1 让地图显示出来.....	395
9.4.1 使用 JSONObject	330	11.4.2 移动到我的位置.....	397
9.4.2 使用 GSON	331	11.4.3 让“我”显示在地图上	400
9.5 网络编程的最佳实践.....	334	11.5 Git 时间——版本控制工具的高级	
9.6 小结与点评.....	338	用法	402
第 10 章 后台默默的劳动者——探究服务	339	11.5.1 分支的用法	403
10.1 服务是什么	339	11.5.2 与远程版本库协作	404
10.2 Android 多线程编程	340	11.6 小结与点评	406
10.2.1 线程的基本用法	340		
10.2.2 在子线程中更新 UI.....	341		
10.2.3 解析异步消息处理机制	345		
10.2.4 使用 AsyncTask	347		
10.3 服务的基本用法	349		
10.3.1 定义一个服务	349		
10.3.2 启动和停止服务	352		
10.3.3 活动和服务进行通信	355		
10.4 服务的生命周期	359		
10.5 服务的更多技巧	359		
10.5.1 使用前台服务	359		
10.5.2 使用 IntentService	361		
10.6 服务的最佳实践——完整版的下载示例	365		
10.7 小结与点评	378		
第 11 章 Android 特色开发——基于位置的服务	379		
11.1 基于位置的服务简介	379		
11.2 申请 API Key	380		
11.3 使用百度定位	384		
11.3.1 准备 LBS SDK	384		
11.3.2 确定自己位置的经纬度	386		
11.3.3 选择定位模式	391		
11.3.4 看得懂的位置信息	393		
第 12 章 最佳的 UI 体验——Material Design 实战	407		
12.1 什么是 Material Design	407		
12.2 Toolbar	408		
12.3 滑动菜单	415		
12.3.1 DrawerLayout	415		
12.3.2 NavigationView	418		
12.4 悬浮按钮和可交互提示	423		
12.4.1 FloatingActionButton	424		
12.4.2 Snackbar	427		
12.4.3 CoordinatorLayout	428		
12.5 卡片式布局	430		
12.5.1 CardView	431		
12.5.2 AppBarLayout	437		
12.6 下拉刷新	440		
12.7 可折叠式标题栏	443		
12.7.1 CollapsingToolbarLayout	443		
12.7.2 充分利用系统状态栏空间	453		
12.8 小结与点评	456		
第 13 章 继续进阶——你还应该掌握的高级技巧	457		
13.1 全局获取 Context 的技巧	457		
13.2 使用 Intent 传递对象	461		
13.2.1 Serializable 方式	461		
13.2.2 Parcelable 方式	463		

13.3	定制自己的日志工具	464
13.4	调试 Android 程序	466
13.5	创建定时任务	469
13.5.1	Alarm 机制	469
13.5.2	Doze 模式	471
13.6	多窗口模式编程	472
13.6.1	进入多窗口模式	473
13.6.2	多窗口模式下的生命周期	475
13.6.3	禁用多窗口模式	479
13.7	Lambda 表达式	481
13.8	总结	485
第 14 章 进入实战——开发酷欧天气		
	天气	486
14.1	功能需求及技术可行性分析	486
14.2	Git 时间——将代码托管到 GitHub 上	489
14.3	创建数据库和表	494
14.4	遍历全国省市县数据	499
14.5	显示天气信息	509
14.5.1	定义 GSON 实体类	509
14.5.2	编写天气界面	514
14.5.3	将天气显示到界面上	520
14.5.4	获取必应每日一图	526
14.6	手动更新天气和切换城市	532
14.6.1	手动更新天气	532
14.6.2	切换城市	535
14.7	后台自动更新天气	540
14.8	修改图标和名称	542
14.9	你还可以做的事情	543
第 15 章 最后一步——将应用发布到 360 应用商店		
	545
15.1	生成正式签名的 APK 文件	545
15.1.1	使用 Android Studio 生成	546
15.1.2	使用 Gradle 生成	548
15.1.3	生成多渠道 APK 文件	551
15.2	申请 360 开发者账号	554
15.3	发布应用程序	556
15.4	嵌入广告进行盈利	560
15.4.1	注册腾讯广告联盟账号	560
15.4.2	新建媒体和广告位	562
15.4.3	接入广告 SDK	564
15.4.4	重新发布应用程序	569
15.5	结束语	570

第 1 章

开始启程——你的第一行 Android 代码

欢迎你来到 Android 世界！Android 系统是目前世界上市场占有率最高的移动操作系统，不管你在哪里，都可以看到 Android 手机几乎无处不在。今天的 Android 世界可谓欣欣向荣，可是你知道它的过去是什么样的吗？我们一起来看一看它的发展史吧。

2003 年 10 月，Andy Rubin 等人一起创办了 Android 公司。2005 年 8 月谷歌收购了这家仅仅成立了 22 个月的公司，并让 Andy Rubin 继续负责 Android 项目。在经过了数年的研发之后，谷歌终于在 2008 年推出了 Android 系统的第一个版本。但自那之后，Android 的发展就一直受到重重阻挠。乔布斯自始至终认为 Android 是一个抄袭 iPhone 的产品，里面剽窃了诸多 iPhone 的创意，并声称一定要毁掉 Android。而本身就是基于 Linux 开发的 Android 操作系统，在 2010 年被 Linux 团队从 Linux 内核主线中除名。又由于 Android 中的应用程序都是使用 Java 开发的，甲骨文则针对 Android 侵犯 Java 知识产权一事对谷歌提起了诉讼……

可是，似乎再多的困难也阻挡不了 Android 快速前进的步伐。由于谷歌的开放政策，任何手机厂商和个人都能免费获取到 Android 操作系统的源码，并且可以自由地使用和定制。三星、HTC、摩托罗拉、索爱等公司都推出了各自系列的 Android 手机，Android 市场上百花齐放。仅仅推出两年后，Android 就超过了已经霸占市场逾十年的诺基亚 Symbian，成为了全球第一大智能手机操作系统，并且每天都还会有数百万台新的 Android 设备被激活。而近几年，国内的手机厂商也是大放异彩，小米、华为、魅族等新兴品牌都推出了相当不错的 Android 手机，并且也获得了市场的广泛认可，目前 Android 已经占据了全球智能手机操作系统 70% 以上的份额。

说了这些，想必你已经体会到 Android 系统炙手可热的程度，并且迫不及待地想要加入到 Android 开发者的行列当中了吧。试想一下，十个人中有七个人的手机都可以运行你编写的应用程序，还有什么能比这个更诱人的呢？那么从今天起，我就带你踏上学习 Android 的旅途，一步步地引导你成为一名出色的 Android 开发者。

好了，现在我们就来一起初窥一下 Android 世界吧。

1.1 了解全貌——Android 王国简介

Android 从面世以来到现在已经发布了二十几个版本了。在这几年的发展过程中，谷歌为 Android 王国建立了一个完整的生态系统。手机厂商、开发者、用户之间相互依存，共同推进着 Android 的蓬勃发展。开发者在其中扮演着不可或缺的角色，因为如果没有开发者来制作丰富的应用程序，那么不管多么优秀的操作系统，也是难以得到大众用户喜爱的，相信没有多少人能够忍受没有 QQ、微信的手机吧。而且，谷歌推出的 Google Play 更是给开发者带来了大量的机遇，只要你能制作出优秀的产品，在 Google Play 上获得了用户的认可，你就完全可以得到不错的经济回报，从而成为一名独立开发者，甚至是成功创业！

那我们现在就以一个开发者的角度，去了解一下这个操作系统吧。纯理论型的东西也比较无聊，怕你看睡着了，因此我只挑重点介绍，这些东西跟你以后的开发工作都是息息相关的。

1.1.1 Android 系统架构

为了让你能够更好地理解 Android 系统是怎么工作的，我们先来看一下它的系统架构。Android 大致可以分为四层架构：Linux 内核层、系统运行库层、应用框架层和应用层。

1. Linux 内核层

Android 系统是基于 Linux 内核的，这一层为 Android 设备的各种硬件提供了底层的驱动，如显示驱动、音频驱动、照相机驱动、蓝牙驱动、Wi-Fi 驱动、电源管理等。

2. 系统运行库层

这一层通过一些 C/C++ 库来为 Android 系统提供了主要的特性支持。如 SQLite 库提供了数据库的支持，OpenGL|ES 库提供了 3D 绘图的支持，Webkit 库提供了浏览器内核的支持等。

同样在这一层还有 Android 运行时库，它主要提供了一些核心库，能够允许开发者使用 Java 语言来编写 Android 应用。另外，Android 运行时库中还包含了 Dalvik 虚拟机（5.0 系统之后改为 ART 运行环境），它使得每一个 Android 应用都能运行在独立的进程当中，并且拥有一个自己的 Dalvik 虚拟机实例。相较于 Java 虚拟机，Dalvik 是专门为移动设备定制的，它针对手机内存、CPU 性能有限等情况做了优化处理。

3. 应用框架层

这一层主要提供了构建应用程序时可能用到的各种 API，Android 自带的一些核心应用就是使用这些 API 完成的，开发者也可以通过使用这些 API 来构建自己的应用程序。

4. 应用层

所有安装在手机上的应用程序都是属于这一层的，比如系统自带的联系人、短信等程序，或者是你从 Google Play 上下载的小游戏，当然还包括你自己开发的程序。

结合图 1.1 你将会理解得更加深刻，图片源自维基百科。

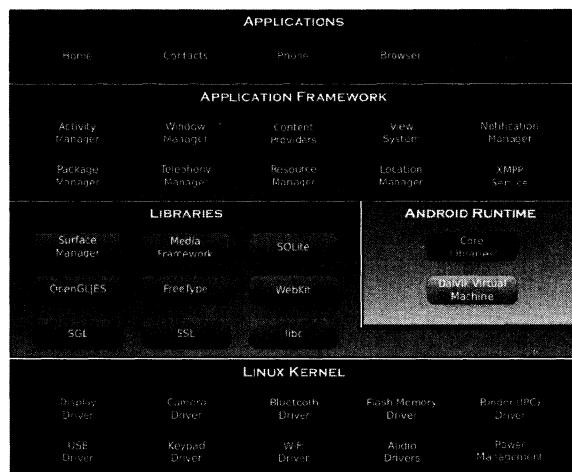


图 1.1 Android 系统架构

1.1.2 Android 已发布的版本

2008年9月，谷歌正式发布了Android 1.0系统，这也是Android系统最早的版本。随后的几年，谷歌以惊人的速度不断地更新Android系统，2.1、2.2、2.3系统的推出使Android占据了大量的市场。2011年2月，谷歌发布了Android 3.0系统，这个系统版本是专门为平板电脑设计的，但也是Android为数不多的比较失败的版本，推出之后一直不见什么起色，市场份额也少得可怜。不过很快，在同年的10月，谷歌又发布了Android 4.0系统，这个版本不再对手机和平板进行差异化区分，既可以应用在手机上，也可以应用在平板上。2014年Google I/O大会上，谷歌推出了号称史上版本改动最大的Android 5.0系统，其中使用ART运行环境替代了Dalvik虚拟机，大大提升了应用的运行速度，还提出了Material Design的概念来优化应用的界面设计。除此之外，还推出了Android Wear、Android Auto、Android TV系统，从而进军可穿戴设备、汽车、电视等全新领域。之后Android的更新速度更加迅速，2015年Google I/O大会上推出了Android 6.0系统，加入运行时权限功能，2016年Google I/O大会上推出了Android 7.0系统，加入多窗口模式功能，这也是目前最新的Android系统版本。

下表中列出了目前市场上主要的Android系统版本及其详细信息。你看到这张表格时，数据很可能已经发生了变化，查看最新的数据可以访问 <http://developer.android.com/about/dashboards/>。

版本号	系统代号	API	市场占有率
2.2	Froyo	8	0.1%
2.3.3 – 2.3.7	Gingerbread	10	1.5%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	1.3%
4.1.x		16	5.6%
4.2.x	Jelly Bean	17	7.7%
4.3		18	2.3%

(续)

版本号	系统代号	API	市场占有率
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%
7.0	Nougat	24	0.1%

从上表中可以看出，目前4.0以上的系统已经占据了超过98%的Android市场份额，因此我们本书中开发的程序也只面向4.0以上的系统，2.x的系统就不再去兼容了。

1.1.3 Android应用开发特色

预告一下，你马上就要开始真正的Android开发旅程了。不过先别急，在开始之前我们再一起看一看，Android系统到底提供了哪些东西，可供我们开发出优秀的应用程序。

1. 四大组件

Android系统四大组件分别是活动（Activity）、服务（Service）、广播接收器（Broadcast Receiver）和内容提供器（Content Provider）。其中活动是所有Android应用程序的门面，凡是在应用中你看得到的东西，都是放在活动中的。而服务就比较低调了，你无法看到它，但它会一直在后台默默地运行，即使用户退出了应用，服务仍然是可以继续运行的。广播接收器允许你的应用接收来自各处的广播消息，比如电话、短信等，当然你的应用同样也可以向外发出广播消息。内容提供器则为应用程序之间共享数据提供了可能，比如你想要读取系统电话簿中的联系人，就需要通过内容提供器来实现。

2. 丰富的系统控件

Android系统为开发者提供了丰富的系统控件，使得我们可以很轻松地编写出漂亮的界面。当然如果你品位比较高，不满足于系统自带的控件效果，也完全可以定制属于自己的控件。

3. SQLite数据库

Android系统还自带了这种轻量级、运算速度极快的嵌入式关系型数据库。它不仅支持标准的SQL语法，还可以通过Android封装好的API进行操作，让存储和读取数据变得非常方便。

4. 强大的多媒体

Android系统还提供了丰富的多媒体服务，如音乐、视频、录音、拍照、闹铃，等等，这一切你都可以在程序中通过代码进行控制，让你的应用变得更加丰富多彩。

5. 地理位置定位

移动设备和PC相比起来，地理位置定位功能应该可以算是很大的一个亮点。现在的Android手机都内置有GPS，走到哪儿都可以定位到自己的位置，发挥你的想象就可以做出创意十足的应

用，如果再结合功能强大的地图功能，LBS 这一领域潜力无限。

既然有 Android 这样出色的系统给我们提供了这么丰富的工具，你还用担心做不出优秀的应用吗？好了，纯理论的东西就介绍到这里，我知道你已经迫不及待想要开始真正的开发之旅了，那我们就开始启程吧！

1.2 手把手带你搭建环境

俗话说得好，“工欲善其事，必先利其器”，开着记事本就想去开发 Android 程序显然不是明智之举，选择一个好的 IDE 可以极大地提高你的开发效率，因此本节我就将手把手带着你把开发环境搭建起来。

1.2.1 准备所需要的工具

我现在对你了解还并不多，但我希望你已经是一个颇有经验的 Java 程序员，这样你理解本书的内容时将会轻而易举，因为 Android 程序都是使用 Java 语言编写的。如果你对 Java 只是略有了解，那阅读本书应该会有一点困难，不过一边阅读一边补充 Java 知识也是可以的。但如果你对 Java 完全没有了解，那么我建议你可以暂时将本书放下，先买本介绍 Java 基础知识的书学上两个星期，把 Java 的基本语法和特性都学会了，再来继续阅读这本书。

好了，既然你已经阅读到这里，说明你已经掌握 Java 的基本用法了，下面我们就来看一看开发 Android 程序需要准备哪些工具。

- **JDK**。JDK 是 Java 语言的软件开发工具包，它包含了 Java 的运行环境、工具集合、基础类库等内容。需要注意的是，本书中的 Android 程序必须要使用 JDK 8 或以上版本才能进行开发。
- **Android SDK**。Android SDK 是谷歌提供的 Android 开发工具包，在开发 Android 程序时，我们需要通过引入该工具包，来使用 Android 相关的 API。
- **Android Studio**。在很早之前，Android 项目都是用 Eclipse 来开发的，相信所有 Java 开发者都一定会对这个工具非常熟悉，它是 Java 开发神器，安装 ADT 插件后就可以用来开发 Android 程序了。而在 2013 年的时候，谷歌推出了一款官方的 IDE 工具 Android Studio，由于不再是以插件的形式存在，Android Studio 在开发 Android 程序方面要远比 Eclipse 强大和方便得多。不过由于 Android Studio 早期的测试版本并不是非常稳定，所以本书的第一版仍然选用的 Eclipse 来作为开发工具。而如今，Android Studio 已经推出了 2.2 版本，稳定性完全不再是问题，普及程度方面也远超 Eclipse，没有比现在更适合的时机来换用 Android Studio 了，因此本书中所有的代码都将在 Android Studio 上进行开发。

1.2.2 搭建开发环境

当然，上述软件并不需要你去一个个地下载，因为谷歌为了简化搭建开发环境的过程，将所

有需要用到的工具都帮我们集成好了，到Android官网就可以下载最新的开发工具，下载地址是：<https://developer.android.com/studio/index.html>。不过，Android官网通常都需要科学上网才能访问，如果你无法访问的话，也可以直接到我的百度网盘去下载，下载地址是：<https://pan.baidu.com/s/1nuABMDb>。（注意网址中是阿拉伯数字1，而不是英文字母1。）

你下载下来的将是一个安装包，安装的过程也很简单，一直点击Next就可以了。其中选择安装组件时建议全部勾上，如图1.2所示。

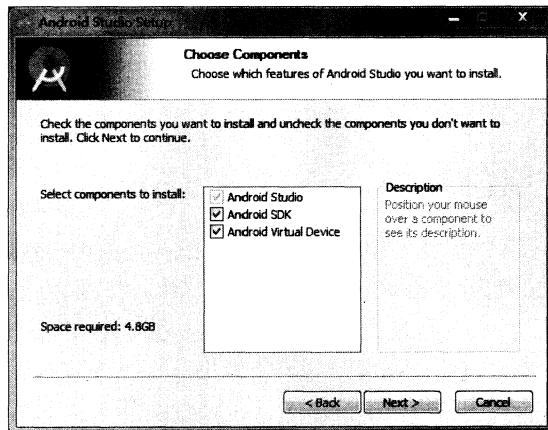


图1.2 选择安装组件

接下来还会让你选择Android Studio的安装地址以及Android SDK的安装地址，这些根据你自己电脑的实际情况选择就行了，不想改动的话就保持默认，如图1.3所示。

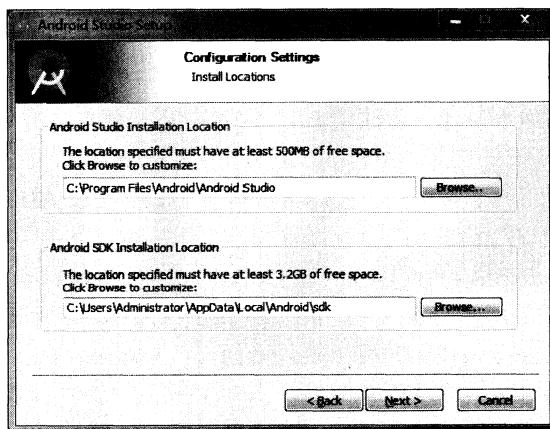


图1.3 选择安装地址

后面就没什么需要注意的了，全部保持默认项，一直点击Next即可完成安装，如图1.4所示。



图 1.4 安装完成

现在点击 Finish 按钮来启动 Android Studio，一开始会让你选择是否导入之前 Android Studio 版本的配置，由于这是我们首次安装，这里选择不导入就可以了，如图 1.5 所示。

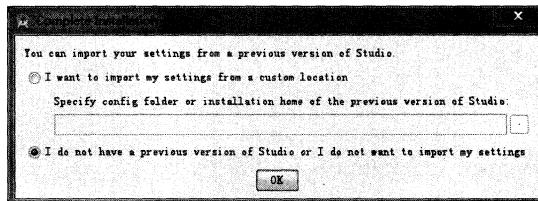


图 1.5 选择不导入配置

点击 OK 按钮会进入到 Android Studio 的配置界面，如图 1.6 所示。

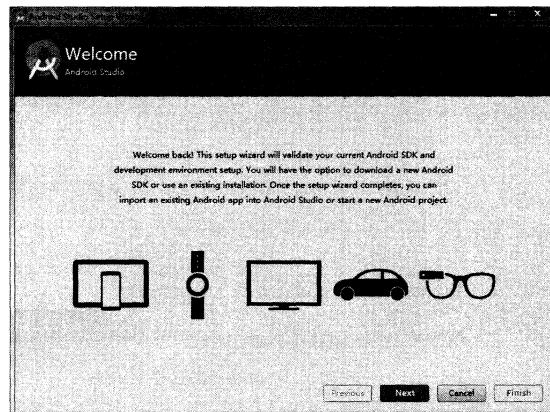


图 1.6 Android Studio 的配置界面

然后点击 Next 开始进行具体的配置，如图 1.7 所示。

这里我们可以选择Android Studio的安装类型，有Standard和Custom两种。Standard表示一切都使用默认的配置，比较方便；Custom则可以根据用户的特殊需求进行自定义。简单起见，这里我们就选择Standard类型了，继续点击Next完成配置工作，如图1.8所示。

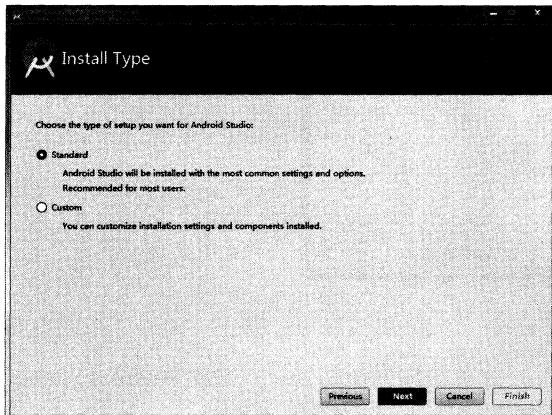


图1.7 选择安装类型

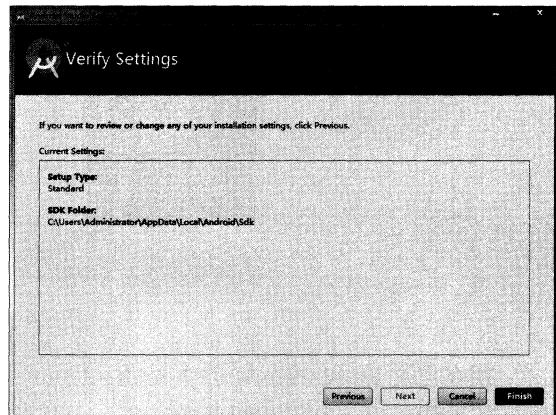


图1.8 完成Android Studio配置

现在点击Finish按钮，配置工作就全部完成了。然后Android Studio会尝试联网下载一些更新，等待更新完成后再点击Finish按钮就会进入Android Studio的欢迎界面，如图1.9所示。

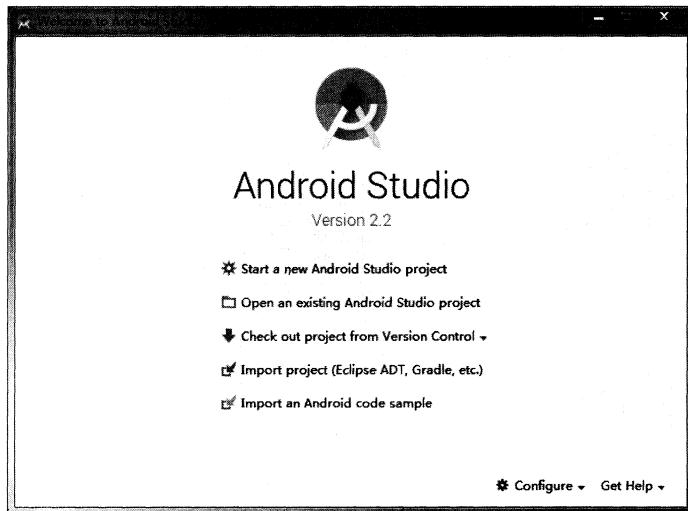


图1.9 Android Studio的欢迎界面

目前为止，Android开发环境就已经全部搭建完成了。那现在应该做什么？当然是写下你的第一行Android代码了，让我们快点开始吧。

1.3 创建你的第一个 Android 项目

任何一个编程语言写出的第一个程序毫无疑问都会是 Hello World，这已经是自 20 世纪 70 年代一直流传下来的传统，在编程界已成为永恒的经典，那我们当然也不会搞例外了。

1.3.1 创建 HelloWorld 项目

在 Android Studio 的欢迎界面点击 Start a new Android Studio project，会打开一个创建新项目的界面，如图 1.10 所示。

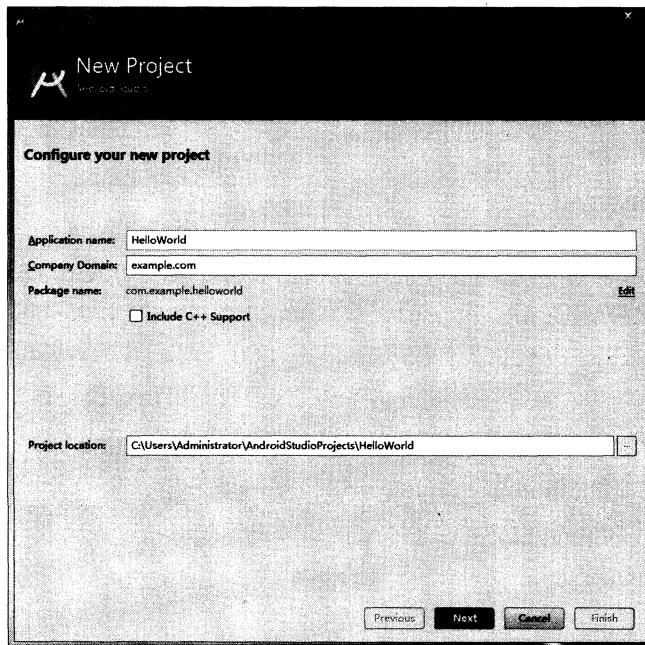


图 1.10 创建新项目

其中 Application name 表示应用名称，此应用安装到手机之后会在手机上显示该名称，这里我们填入 HelloWorld。Company Domain 表示公司域名，如果是个人开发者，没有公司域名的话，那么就像我一样填 example.com 就可以了。Package name 表示项目的包名，Android 系统就是通过包名来区分不同应用程序的，因此包名一定要具有唯一性。Android Studio 会根据应用名称和公司域名来自动帮我们生成合适的包名，如果你不想使用默认生成的包名，也可以点击右侧的 Edit 按钮自行修改。最后，Project location 表示项目代码存放的位置，如果没有特殊要求的话，这里也保持默认就可以了。

接下来点击 Next 可以对项目的最低兼容版本进行设置，如图 1.11 所示。

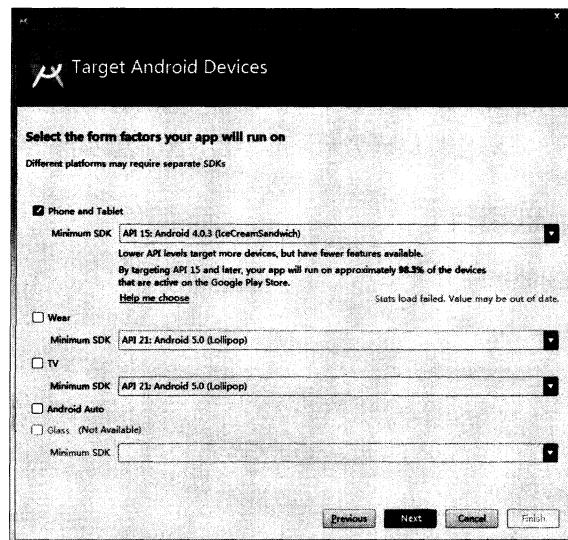


图 1.11 设置项目的最低兼容版本

前面已经说过，Android 4.0 以上的系统已经占据了超过 98% 的 Android 市场份额，因此这里我们将 Minimum SDK 指定成 API 15 就可以了。另外，Wear、TV 和 Android Auto 这几个选项分别是用于开发可穿戴设备、电视和汽车程序的，目前这几个领域在国内还没有普及，我们暂时就先忽略吧。接着点击 Next 会跳转到创建活动界面，这里我们可以选择一种模板，如图 1.12 所示。



图 1.12 选择模板

可以看到，Android Studio 提供了很多种内置模板，不过由于我们才刚刚开始学习，用不着这么多复杂的模板，这里直接选择 Empty Activity 来创建一个空的活动就可以了。

继续点击 Next，可以给创建的活动和布局命名，如图 1.13 所示。

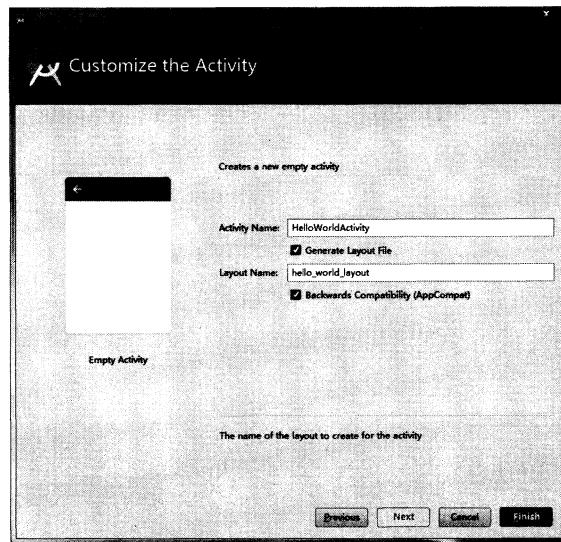


图 1.13 给活动和布局命名

其中，Activity Name 表示活动的名字，这里填入 HelloWorldActivity，Layout Name 表示布局的命名，这里填入 hello_world_layout。然后点击 Finish 按钮，并耐心等待一会儿，项目就会创建成功了，如图 1.14 所示。

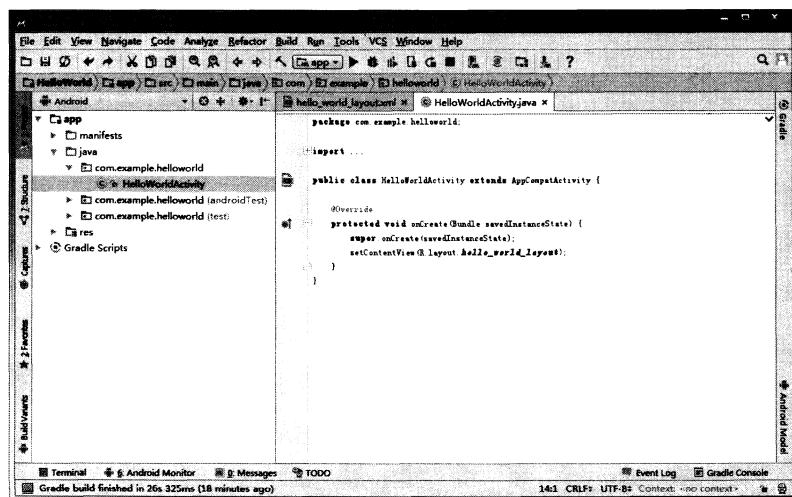


图 1.14 项目创建成功

1.3.2 启动模拟器

由于Android Studio自动为我们生成了很多东西，你现在不需要编写任何代码，HelloWorld项目就已经可以运行了。但是在此之前还必须要有一个运行的载体，可以是一部Android手机，也可以是Android模拟器。这里我们暂时先使用模拟器来运行程序，如果你想立刻就将程序运行到手机上的话，可以参考8.1节的内容。

那么我们现在就来创建一个Android模拟器，观察Android Studio顶部工具栏中的图标，如图1.15所示。



图1.15 顶部工具栏中的图标

其中，最左边的按钮就是用于创建和启动模拟器的，点击该按钮，会弹出如图1.16所示的窗口。

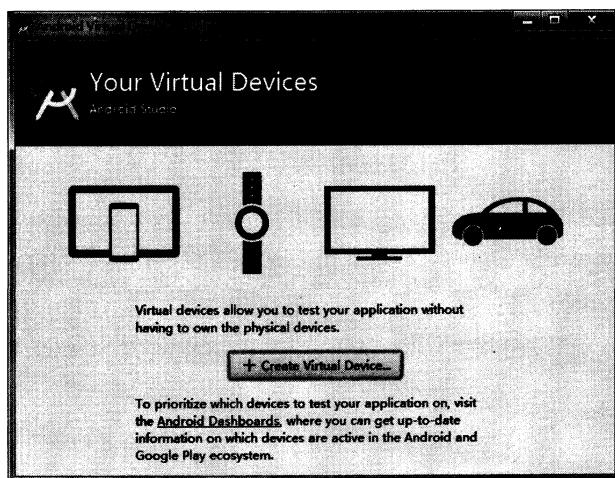


图1.16 创建模拟器

可以看到，目前我们的模拟器列表中还是空的，点击Create Virtual Device按钮就可以立刻开始创建了，如图1.17所示。

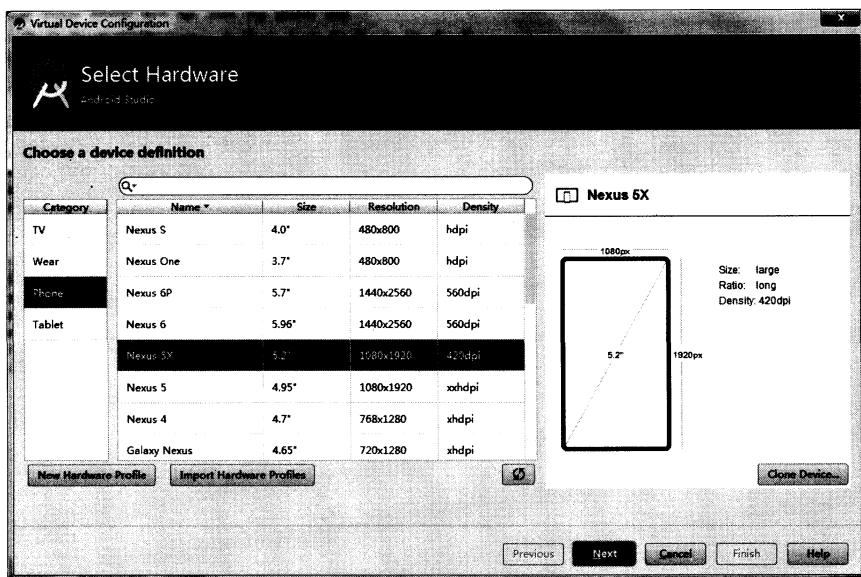


图 1.17 选择要创建的模拟器设备

这里有很多种设备可供我们选择，不仅能创建手机模拟器，还可以创建平板、手表、电视等模拟器。

那么我就选择创建 Nexus 5X 这台设备的模拟器了，点击 Next，如图 1.18 所示。

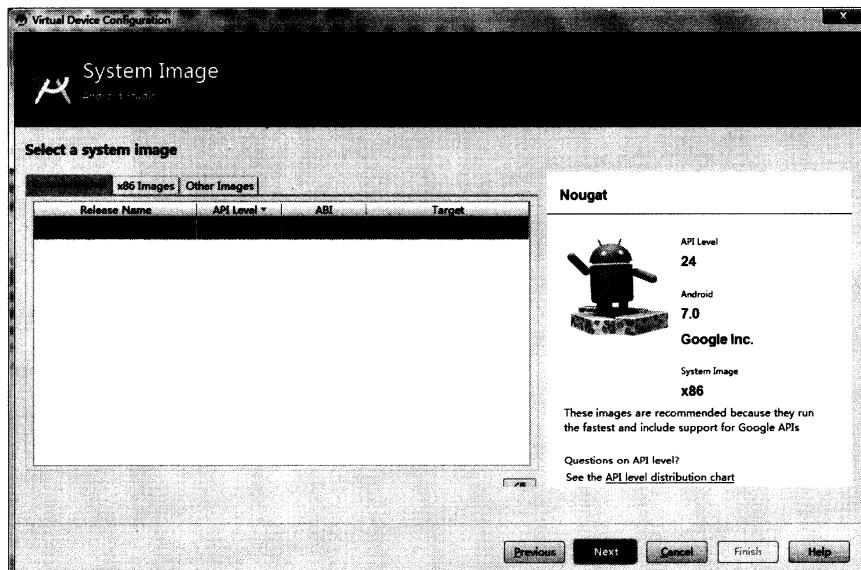


图 1.18 选择模拟器的操作系统版本

这里可以选择模拟器所使用的操作系统版本，毫无疑问，我们肯定要选择最新的Android 7.0系统。继续点击Next，如图1.19所示。

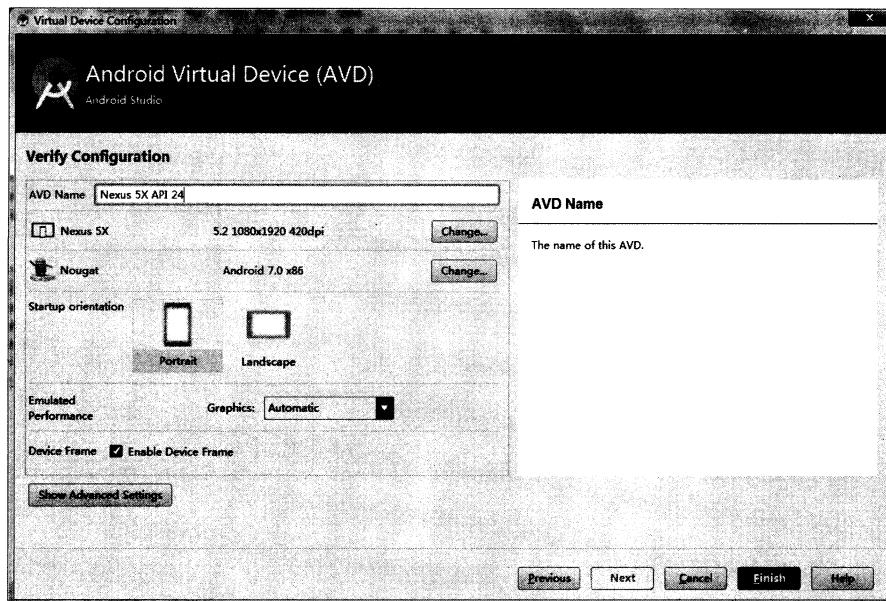


图1.19 确认模拟器配置

在这里我们可以对模拟器的一些配置进行确认，比如说指定模拟器的名字、分辨率、横竖屏等信息，如果没有特殊需求的话，全部保持默认就可以了。点击Finish完成模拟器的创建，然后会弹出如图1.20所示的窗口。



图1.20 模拟器列表

可以看到，现在模拟器列表中已经存在一个创建好的模拟器设备了，点击 Actions 栏目中最左边的三角形按钮即可启动模拟器。模拟器会像手机一样，有一个开机过程，启动完成之后的界面如图 1.21 所示。

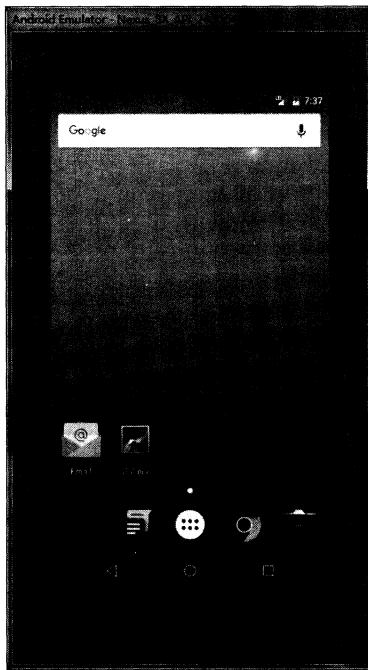


图 1.21 启动后的模拟器界面

很清新的 Android 界面出来了！看上去还挺不错吧，你几乎可以像使用手机一样使用它，Android 模拟器对手机的模仿度非常高，快去体验一下吧。

1.3.3 运行 HelloWorld

现在模拟器已经启动起来了，那么下面我们就开始将 HelloWorld 项目运行到模拟器上。观察 Android Studio 顶部工具栏中的图标，如图 1.22 所示。其中左边的锤子按钮是用来编译项目的，中间的下拉列表是用来选择运行哪一个项目的，通常 app 就是当前的主项目，右边的三角形按钮是用来运行项目的。

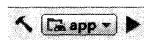


图 1.22 顶部工具栏中的图标

现在点击右边的运行按钮，会弹出一个选择运行设备的对话框，如图 1.23 所示。

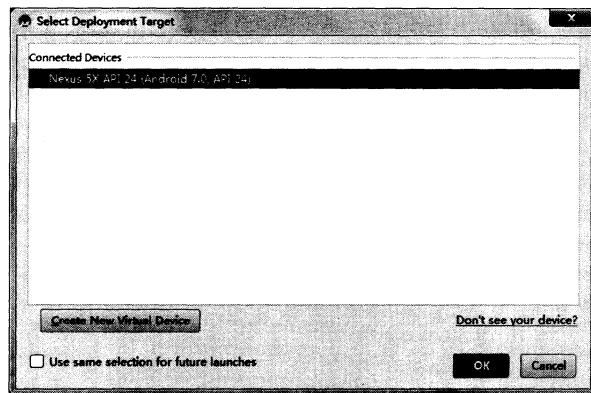


图 1.23 选择运行设备对话框

可以看到，我们刚刚创建的模拟器现在是在线的，点击 OK 按钮，稍微等待一会儿，HelloWorld 项目就会运行到模拟器上了，结果应该和图 1.24 中显示的是一样的。

HelloWorld 项目运行成功！并且你会发现，模拟器上已经安装上 HelloWorld 这个应用了。打开启动器列表，如图 1.25 所示。

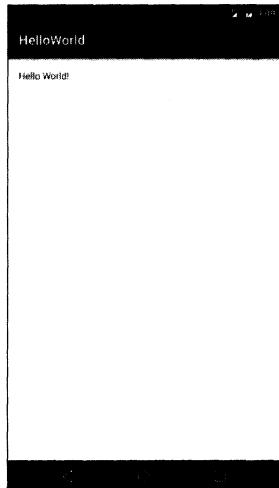


图 1.24 运行 HelloWorld 项目

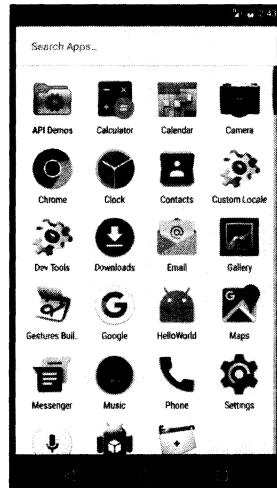


图 1.25 查看启动器列表

这个时候你可能会说我坑你了，说好的第一行代码呢？怎么一行还没写，项目就已经运行起来了？这个只能说是因为 Android Studio 太智能了，已经帮我们把一些简单内容都自动生成了。你也别心急，后面写代码的机会多着呢，我们先来分析一下 HelloWorld 这个项目吧。

1.3.4 分析你的第一个 Android 程序

回到 Android Studio 当中，首先展开 HelloWorld 项目，你会看到如图 1.26 所示的项目结构。

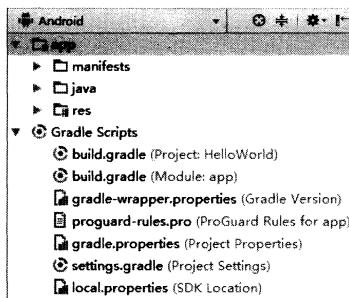


图 1.26 Android 模式的项目结构

任何一个新建的项目都会默认使用 Android 模式的项目结构，但这并不是项目真实的目录结构，而是被 Android Studio 转换过的。这种项目结构简洁明了，适合进行快速开发，但是对于新手来说可能并不易于理解。点击图 1.26 当中的 Android 区域可以切换项目结构模式，如图 1.27 所示。

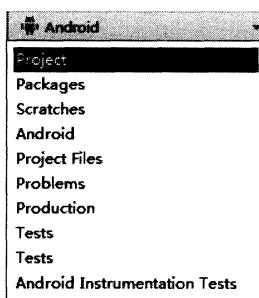


图 1.27 切换项目结构模式

这里我们将项目结构模式切换成 Project，这就是项目真实的目录结构了，如图 1.28 所示。

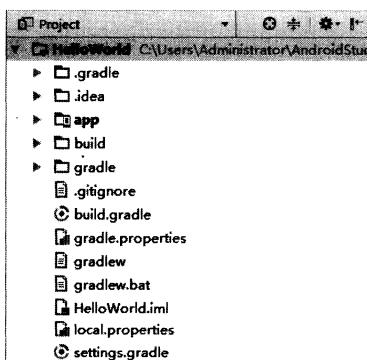


图 1.28 Project 模式的项目结构

一开始看到这么多陌生的东西，你一定会感到有点头昏吧。别担心，我现在就对图 1.28 中的内容进行一一讲解，之后你再看这张图就不会感到那么吃力了。

1. .gradle 和.idea

这两个目录下放置的都是 Android Studio 自动生成的一些文件，我们无须关心，也不要去做手动编辑。

2. app

项目中的代码、资源等内容几乎都是放置在这个目录下的，我们后面的开发工作也基本都是在这个目录下进行的，待会儿还会对这个目录单独展开进行讲解。

3. build

这个目录你也不需要过多关心，它主要包含了一些在编译时自动生成的文件。

4. gradle

这个目录下包含了 gradle wrapper 的配置文件，使用 gradle wrapper 的方式不需要提前将 gradle 下载好，而是会自动根据本地的缓存情况决定是否需要联网下载 gradle。Android Studio 默认没有启用 gradle wrapper 的方式，如果需要打开，可以点击 Android Studio 导航栏→File→Settings→Build, Execution, Deployment→Gradle，进行配置更改。

5. .gitignore

这个文件是用来将指定的目录或文件排除在版本控制之外的，关于版本控制我们将在第 5 章中开始正式的学习。

6. build.gradle

这是项目全局的 gradle 构建脚本，通常这个文件中的内容是不需要修改的。稍后我们将会详细分析 gradle 构建脚本中的具体内容。

7. gradle.properties

这个文件是全局的 gradle 配置文件，在这里配置的属性将会影响到项目中所有的 gradle 编译脚本。

8. gradlew 和 gradlew.bat

这两个文件是用来在命令行界面中执行 gradle 命令的，其中 gradlew 是在 Linux 或 Mac 系统中使用的，gradlew.bat 是在 Windows 系统中使用的。

9. HelloWorld.iml

iml 文件是所有 IntelliJ IDEA 项目都会自动生成的一个文件（Android Studio 是基于 IntelliJ IDEA 开发的），用于标识这是一个 IntelliJ IDEA 项目，我们不需要修改这个文件中的任何内容。

10. local.properties

这个文件用于指定本机中的 Android SDK 路径，通常内容都是自动生成的，我们并不需要修改。除非你本机中的 Android SDK 位置发生了变化，那么就将这个文件中的路径改成新的位置即可。

11. settings.gradle

这个文件用于指定项目中所有引入的模块。由于 HelloWorld 项目中就只有一个 app 模块，因此该文件中也就只引入了 app 这一个模块。通常情况下模块的引入都是自动完成的，需要我们手动去修改这个文件的场景可能比较少。

现在整个项目的外层目录结构已经介绍完了。你会发现，除了 app 目录之外，大多数的文件和目录都是自动生成的，我们并不需要进行修改。想必你已经猜到了，app 目录下的内容才是我们以后的工作重点，展开之后结构如图 1.29 所示。

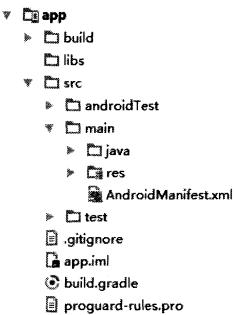


图 1.29 app 目录下的结构

那么下面我们就来对 app 目录下的内容进行更为详细的分析。

1. build

这个目录和外层的 build 目录类似，主要也是包含了一些在编译时自动生成的文件，不过它里面的内容会更多更杂，我们不需要过多关心。

2. libs

如果你的项目中使用到了第三方 jar 包，就需要把这些 jar 包都放在 libs 目录下，放在这个目录下的 jar 包都会被自动添加到构建路径里去。

3. androidTest

此处是用来编写 Android Test 测试用例的，可以对项目进行一些自动化测试。

4. java

毫无疑问，java 目录是放置我们所有 Java 代码的地方，展开该目录，你将看到我们刚才创建的 HelloWorldActivity 文件就在里面。

5. res

这个目录下的内容就有点多了。简单点说，就是你在项目中使用到的所有图片、布局、字符串等资源都要存放在这个目录下。当然这个目录下还有很多子目录，图片放在 drawable 目录下，布局放在 layout 目录下，字符串放在 values 目录下，所以你不用担心会把整个 res 目录弄得乱糟糟的。

6. AndroidManifest.xml

这是你整个Android项目的配置文件，你在程序中定义的所有四大组件都需要在这个文件里注册，另外还可以在这个文件中给应用程序添加权限声明。由于这个文件以后会经常用到，我们用到的时候再做详细说明。

7. test

此处是用来编写Unit Test测试用例的，是对项目进行自动化测试的另一种方式。

8. .gitignore

这个文件用于将app模块内的指定的目录或文件排除在版本控制之外，作用和外层的.gitignore文件类似。

9. app.iml

IntelliJ IDEA项目自动生成的文件，我们不需要关心或修改这个文件中的内容。

10. build.gradle

这是app模块的gradle构建脚本，这个文件中会指定很多项目构建相关的配置，我们稍后将会详细分析gradle构建脚本中的具体内容。

11. proguard-rules.pro

这个文件用于指定项目代码的混淆规则，当代码开发完成后打成安装包文件，如果不希望代码被别人破解，通常会将代码进行混淆，从而让破解者难以阅读。

这样整个项目的目录结构就都介绍完了，如果你还不能完全理解的话也很正常，毕竟里面有太多的东西你都还没接触过。不过不用担心，这并不会影响到你后面的学习。等你学完整本书再回来看这个目录结构图时，你会觉得特别地清晰和简单。

接下来我们一起分析一下HelloWorld项目究竟是怎么运行起来的吧。首先打开AndroidManifest.xml文件，从中可以找到如下代码：

```
<activity android:name=".HelloWorldActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

这段代码表示对HelloWorldActivity这个活动进行注册，没有在AndroidManifest.xml里注册的活动是不能使用的。其中intent-filter里的两行代码非常重要，<action android:name="android.intent.action.MAIN" />和<category android:name="android.intent.category.LAUNCHER" />表示HelloWorldActivity是这个项目的主活动，在手机上点击应用图标，首先启动的就是这个活动。

那HelloWorldActivity具体又有什么作用呢？我在介绍Android四大组件的时候说过，活动

是 Android 应用程序的门面，凡是在应用中你看得到的东西，都是放在活动中的。因此你在图 1.24 中看到的界面，其实就是 HelloWorldActivity 这个活动。那我们快去看一下它的代码吧，打开 HelloWorldActivity，代码如下所示：

```
public class HelloWorldActivity extends AppCompatActivity {

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

}
```

首先我们可以看到，HelloWorldActivity 是继承自 AppCompatActivity 的，这是一种向下兼容的 Activity，可以将 Activity 在各个系统版本中增加的特性和功能最低兼容到 Android 2.1 系统。Activity 是 Android 系统提供的一个活动基类，我们项目中所有的活动都必须继承它或者它的子类才能拥有活动的特性（AppCompatActivity 是 Activity 的子类）。然后可以看到 HelloWorldActivity 中有一个 onCreate() 方法，这个方法是一个活动被创建时必定要执行的方法，其中只有两行代码，并且没有 Hello World! 的字样。那么图 1.24 中显示的 Hello World! 是在哪里定义的呢？

其实 Android 程序的设计讲究逻辑和视图分离，因此是不推荐在活动中直接编写界面的，更加通用的一种做法是，在布局文件中编写界面，然后在活动中引入进来。可以看到，在 onCreate() 方法的第二行调用了 setContentView() 方法，就是这个方法给当前的活动引入了一个 hello_world_layout 布局，那 Hello World! 一定就是在里面定义的了！我们快打开这个文件看一看。

布局文件都是定义在 res/layout 目录下的，当你展开 layout 目录，你会看到 hello_world_layout.xml 这个文件。打开该文件并切换到 Text 视图，代码如下所示：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/hello_world_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.helloworld.HelloWorldActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

现在还看不懂？没关系，后面我会对布局进行详细讲解的，你现在只需要看到上面代码中有

一个 TextView，这是 Android 系统提供的一个控件，用于在布局中显示文字的。然后你终于在 TextView 中看到了 Hello World! 的字样！哈哈！终于找到了，原来就是通过 `android:text="Hello World!"` 这句代码定义的。

这样我们就将 HelloWorld 项目的目录结构以及基本的执行过程都分析完了，相信你对 Android 项目已经有了一个初步的认识，下一小节中我们就来学习一下项目中所包含的资源。

1.3.5 详解项目中的资源

如果你展开 res 目录看一下，其实里面的东西还是挺多的，很容易让人看得眼花缭乱，如图 1.30 所示。

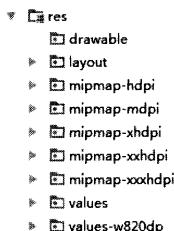


图 1.30 res 目录下的结构

看到这么多的文件夹也不用害怕，其实归纳一下，res 目录就变得非常简单了。所有以 `drawable` 开头的文件夹都是用来放图片的，所有以 `mipmap` 开头的文件夹都是用来放应用图标的，所有以 `values` 开头的文件夹都是用来放字符串、样式、颜色等配置的，`layout` 文件夹是用来放布局文件的。怎么样，是不是突然感觉清晰了很多？

之所以有这么多 `mipmap` 开头的文件夹，其实主要是为了让程序能够更好地兼容各种设备。`drawable` 文件夹也是相同的道理，虽然 Android Studio 没有帮我们自动生成，但是我们应该自己创建 `drawable-hdpi`、`drawable-xhdpi`、`drawable-xxhdpi` 等文件夹。在制作程序的时候最好能够给同一张图片提供几个不同分辨率的版本，分别放在这些文件夹下，然后当程序运行的时候，会自动根据当前运行设备分辨率的高低选择加载哪个文件夹下的图片。当然这只是理想情况，更多的时候美工只会提供给我们一份图片，这时你就把所有图片都放在 `drawable-xxhdpi` 文件夹下就好了。

知道了 res 目录下每个文件夹的含义，我们再来看一下如何去使用这些资源吧。打开 `res/values/strings.xml` 文件，内容如下所示：

```

<resources>
    <string name="app_name">HelloWorld</string>
</resources>
  
```

可以看到，这里定义了一个应用程序名的字符串，我们有以下两种方式来引用它。

- 在代码中通过 `R.string.hello_world` 可以获得该字符串的引用。
- 在 XML 中通过 `@string/hello_world` 可以获得该字符串的引用。

基本的语法就是上面这两种方式，其中 `string` 部分是可以替换的，如果是引用的图片资源就可以替换成 `drawable`，如果是引用的应用图标就可以替换成 `mipmap`，如果是引用的布局文件就可以替换成 `layout`，以此类推。

下面举一个简单的例子来帮助你理解，打开 `AndroidManifest.xml` 文件，找到如下代码：

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    ...
</application>
```

其中，`HelloWorld` 项目应用图标就是通过 `android:icon` 属性来指定的，应用的名称则是通过 `android:label` 属性指定的。可以看到，这里对资源引用的方式正是我们刚刚学过的在 XML 中引用资源的语法。

经过本小节的学习，如果你想修改应用的图标或者名称，相信已经知道该怎么办了吧。

1.3.6 详解 `build.gradle` 文件

不同于 `Eclipse`，`Android Studio` 是采用 `Gradle` 来构建项目的。`Gradle` 是一个非常先进的项目构建工具，它使用了一种基于 `Groovy` 的领域特定语言（DSL）来声明项目设置，摒弃了传统基于 `XML`（如 `Ant` 和 `Maven`）的各种烦琐配置。

在 1.3.4 小节中我们已经看到，`HelloWorld` 项目中有两个 `build.gradle` 文件，一个是在最外层目录下的，一个是在 `app` 目录下的。这两个文件对构建 `Android Studio` 项目都起到了至关重要的作用，下面我们就来对这两个文件中的内容进行详细的分析。

先来看一下最外层目录下的 `build.gradle` 文件，代码如下所示：

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}
```

这些代码都是自动生成的，虽然语法结构看上去可能有点难以理解，但是如果我们忽略语法结构，只看最关键的部分，其实还是很好懂的。

首先，两处 `repositories` 的闭包中都声明了 `jcenter()` 这行配置，那么这个 `jcenter` 是什么意思呢？其实它是一个代码托管仓库，很多 Android 开源项目都会选择将代码托管到 `jcenter` 上，声明了这行配置之后，我们就可以在项目中轻松引用任何 `jcenter` 上的开源项目了。

接下来，`dependencies` 闭包中使用 `classpath` 声明了一个 Gradle 插件。为什么要声明这个插件呢？因为 Gradle 并不是专门为构建 Android 项目而开发的，Java、C++ 等很多种项目都可以使用 Gradle 来构建。因此如果我们要想使用它来构建 Android 项目，则需要声明 `com.android.tools.build:gradle:2.2.0` 这个插件。其中，最后面的部分是插件的版本号，我在写作本书时最新的插件版本是 2.2.0。

这样我们就将最外层目录下的 `build.gradle` 文件分析完了，通常情况下你并不需要修改这个文件中的内容，除非你想添加一些全局的项目构建配置。

下面我们再来看一下 `app` 目录下的 `build.gradle` 文件，代码如下所示：

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.2"
    defaultConfig {
        applicationId "com.example.helloworld"
        minSdkVersion 15
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:24.2.1'
    testCompile 'junit:junit:4.12'
}
```

这个文件中的内容就要相对复杂一些了，下面我们一行行地进行分析。首先第一行应用了一个插件，一般有两种值可选：`com.android.application` 表示这是一个应用程序模块，`com.android.library` 表示这是一个库模块。应用程序模块和库模块的最大区别在于，一个是可以直接运行的，一个只能作为代码库依附于别的应用程序模块来运行。

接下来是一个大的 `android` 闭包，在这个闭包中我们可以配置项目构建的各种属性。其中，`compileSdkVersion` 用于指定项目的编译版本，这里指定成 24 表示使用 Android 7.0 系统的 SDK 编译。`buildToolsVersion` 用于指定项目构建工具的版本，目前最新的版本就是 24.0.2，如果

有更新的版本时，Android Studio 会进行提示。

然后我们看到，这里在 `android` 闭包中又嵌套了一个 `defaultConfig` 闭包，`defaultConfig` 闭包中可以对项目的更多细节进行配置。其中，`applicationId` 用于指定项目的包名，前面我们在创建项目的时候其实已经指定过包名了，如果你想在后面对其进行修改，那么就是在这里修改的。`minSdkVersion` 用于指定项目最低兼容的 Android 系统版本，这里指定成 15 表示最低兼容到 Android 4.0 系统。`targetSdkVersion` 指定的值表示你在该目标版本上已经做过了充分的测试，系统将会为你的应用程序启用一些最新的功能和特性。比如说 Android 6.0 系统中引入了运行时权限这个功能，如果你将 `targetSdkVersion` 指定成 23 或者更高，那么系统就会为你的程序启用运行时权限功能，而如果你将 `targetSdkVersion` 指定成 22，那么就说明你的程序最高只在 Android 5.1 系统上做过充分的测试，Android 6.0 系统中引入的新功能自然就不会启用了。剩下的两个属性都比较简单，`versionCode` 用于指定项目的版本号，`versionName` 用于指定项目的版本名，这两个属性在生成安装文件的时候非常重要，我们在后面都会学到。

分析完了 `defaultConfig` 闭包，接下来我们看一下 `buildTypes` 闭包。`buildTypes` 闭包中用于指定生成安装文件的相关配置，通常只会有两个子闭包，一个是 `debug`，一个是 `release`。`debug` 闭包用于指定生成测试版安装文件的配置，`release` 闭包用于指定生成正式版安装文件的配置。另外，`debug` 闭包是可以忽略不写的，因此我们看到上面的代码中就只有一个 `release` 闭包。下面来看一下 `release` 闭包中的具体内容吧，`minifyEnabled` 用于指定是否对项目的代码进行混淆，`true` 表示混淆，`false` 表示不混淆。`proguardFiles` 用于指定混淆时使用的规则文件，这里指定了两个文件，第一个 `proguard-android.txt` 是在 Android SDK 目录下的，里面是所有项目通用的混淆规则，第二个 `proguard-rules.pro` 是在当前项目的根目录下的，里面可以编写当前项目特有的混淆规则。需要注意的是，通过 Android Studio 直接运行项目生成的都是测试版安装文件，关于如何生成正式版安装文件我们将会在第 15 章中学习。

这样整个 `android` 闭包中的内容就都分析完了，接下来还剩一个 `dependencies` 闭包。这个闭包的功能非常强大，它可以指定当前项目所有的依赖关系。通常 Android Studio 项目一共有 3 种依赖方式：本地依赖、库依赖和远程依赖。本地依赖可以对本地的 Jar 包或目录添加依赖关系，库依赖可以对项目中的库模块添加依赖关系，远程依赖则可以对 jcenter 库上的开源项目添加依赖关系。观察一下 `dependencies` 闭包中的配置，第一行的 `compile fileTree` 就是一个本地依赖声明，它表示将 `libs` 目录下所有 `.jar` 后缀的文件都添加到项目的构建路径当中。而第二行的 `compile` 则是远程依赖声明，`com.android.support:appcompat-v7:24.2.1` 就是一个标准的远程依赖库格式，其中 `com.android.support` 是域名部分，用于和其他公司的库做区分；`appcompat-v7` 是组名称，用于和同一个公司中不同的库做区分；`24.2.1` 是版本号，用于和同一个库不同的版本做区分。加上这句声明后，Gradle 在构建项目时会首先检查一下本地是否已经有这个库的缓存，如果没有的话则会去自动联网下载，然后再添加到项目的构建路径当中。至于库依赖声明这里没有用到，它的基本格式是 `compile project` 后面加上要依赖的库名称，比如说有一个库模块的名字叫 `helper`，那么添加这个库的依赖关系只需要加入 `compile project(':helper')` 这句声明即可。另外剩下

的一句 `testCompile` 是用于声明测试用例库的，这个我们暂时用不到，先忽略它就可以了。

1.4 前行必备——掌握日志工具的使用

通过上一节的学习，你已经成功创建了你的第一个Android程序，并且对Android项目的目录结构和运行流程都有了一定的了解。现在本应该是你继续前行的时候，不过我想在这里给你穿插一点内容，讲解一下Android中日志工具的使用方法，这对你以后的Android开发之旅会有极大的帮助。

1.4.1 使用Android的日志工具Log

Android中的日志工具类是 `Log` (`android.util.Log`)，这个类中提供了如下5个方法来供我们打印日志。

- `Log.v()`。用于打印那些最为琐碎的、意义最小的日志信息。对应级别 `verbose`，是Android日志里面级别最低的一种。
- `Log.d()`。用于打印一些调试信息，这些信息对你调试程序和分析问题应该是有帮助的。对应级别 `debug`，比 `verbose` 高一级。
- `Log.i()`。用于打印一些比较重要的数据，这些数据应该是你非常想看到的、可以帮你分析用户行为数据。对应级别 `info`，比 `debug` 高一级。
- `Log.w()`。用于打印一些警告信息，提示程序在这个地方可能会有潜在的风险，最好去修复一下这些出现警告的地方。对应级别 `warn`，比 `info` 高一级。
- `Log.e()`。用于打印程序中的错误信息，比如程序进入到了 `catch` 语句当中。当有错误信息打印出来的时候，一般都代表你的程序出现严重问题了，必须尽快修复。对应级别 `error`，比 `warn` 高一级。

其实很简单，一共就5个方法，当然每个方法还会有不同的重载，但那对你来说肯定不是什么难理解的地方了。我们现在就在HelloWorld项目中试一试日志工具好不好用吧。

打开HelloWorldActivity，在`onCreate()`方法中添加一行打印日志的语句，如下所示：

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.hello_world_layout);  
    Log.d("HelloWorldActivity", "onCreate execute");  
}
```

`Log.d()`方法中传入了两个参数：第一个参数是 `tag`，一般传入当前的类名就好，主要用于对打印信息进行过滤；第二个参数是 `msg`，即想要打印的具体的内容。

现在可以重新运行一下HelloWorld这个项目了，点击顶部工具栏上的运行按钮，或者使用快捷键 Shift + F10 (Mac系统是 control + R)，等程序运行完毕，点击Android Studio底部工具栏的Android Monitor，在logcat中就可以看到打印信息了，如图1.31所示。

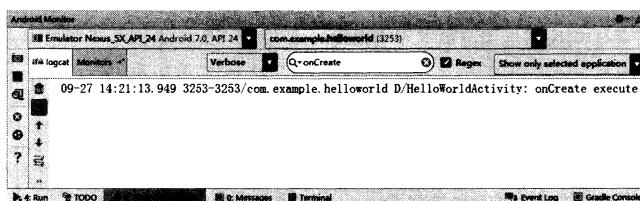


图 1.31 logcat 中的打印信息

其中，你不仅可以看到打印日志的内容和 tag 名，就连程序的包名、打印的时间以及应用程序的进程号都可以看到。

另外，不知道你有没有注意到，你的第一行代码已经在不知不觉中写出来了，我也总算是交差了。

1.4.2 为什么使用 Log 而不使用 System.out

我相信很多的 Java 新手都非常喜欢使用 `System.out.println()` 方法来打印日志，不知道你是不是也喜欢这么做。不过在真正的项目开发中，是极度不建议使用 `System.out.println()` 方法的！如果你在公司的项目中经常使用这个方法，就很有可能要挨骂了。

为什么 `System.out.println()` 方法会这么遭大家唾弃呢？经过我仔细分析之后，发现这个方法除了使用方便一点之外，其他就一无是处了。方便在哪儿呢？在 Eclipse 中你只需要输入 `sys`，然后按下代码提示键，这个方法就会自动出来了，相信这也是很多 Java 新手对它钟情的原因，不过遗憾的是，Android Studio 中已经不支持这种快捷输入了。那缺点又在哪儿了呢？这个就太多了，比如日志打印不可控制、打印时间无法确定、不能添加过滤器、日志没有级别区分……

听我说了这些，你可能已经不太想用 `System.out.println()` 方法了，那么 Log 就把上面所说的缺点全部都改好了吗？虽然谈不上全部，但我觉得 Log 已经做得相当不错了。我现在就来带你看看 Log 和 logcat 配合的强大之处。

首先刚才提到的快捷输入，在 Android Studio 当中也是有的，比如你想打印一条 `debug` 级别的日志，那么只需要输入 `logd`，然后按下 Tab 键，就会帮你自动补全一条完整的打印语句。输入 `logi`，然后按下 Tab 键，会自动补全一条 `info` 级别的打印日志。输入 `logw`，按下 Tab 键，会自动补全一条 `warn` 级别的打印日志，以此类推。另外，由于 Log 的所有打印方法都要求传入一个 tag 参数，每次写一遍显然太过麻烦。这里还有一个小技巧，我们在 `onCreate()` 方法的外面输入 `logt`，然后按下 Tab 键，这时就会以当前的类名作为值自动生成一个 TAG 常量，如下所示：

```
public class HelloWorldActivity extends AppCompatActivity {

    private static final String TAG = "HelloWorldActivity";

    ...
}
```

除了快捷输入之外，logcat中还能很轻松地添加过滤器，你可以在图1.32中看到我们目前所有的过滤器。

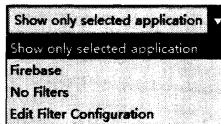


图1.32 logcat中的过滤器

目前只有3个过滤器，Show only selected application表示只显示当前选中程序的日志，Firebase是谷歌提供的一个分析工具，我们可以不用管它，No Filters相当于没有过滤器，会把所有的日志都显示出来。那可不可以自定义过滤器呢？当然可以，我们现在就来添加一个过滤器试试。

点击图1.32中的Edit Filter Configuration，会弹出一个过滤器配置界面。我们给过滤器起名叫data，并且让它对名为data的tag进行过滤，如图1.33所示。

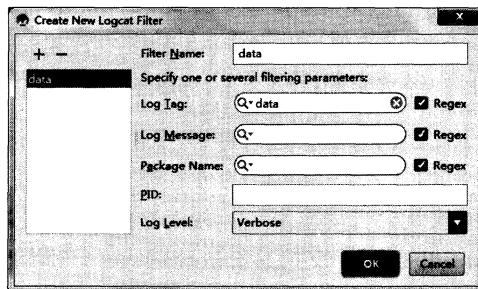


图1.33 过滤器配置界面

点击OK，你就会发现你已经多出了一个data过滤器。当你点击这个过滤器的时候，你会发现刚才在onCreate()方法里打印的日志没了，这是因为data这个过滤器只会显示tag名称为data的日志。你可以尝试在onCreate()方法中把打印日志的语句改成Log.d("data", "onCreate execute")，然后再次运行程序，你就会在data过滤器下看到这行日志了。

不知道你有没有体会到使用过滤器的好处，可能现在还没有吧。不过当你的程序打印出成百上千行日志的时候，你就会迫切地需要过滤器了。

看完了过滤器，再来看一下logcat中的日志级别控制吧。logcat中主要有5个级别，分别对应着上一节介绍的5个方法，如图1.34所示。

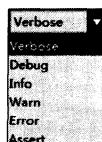


图1.34 logcat中的日志级别

当前我们选中的级别是 `verbose`，也就是最低等级。这意味着不管我们使用哪一个方法打印日志，这条日志都一定会显示出来。而如果我们将级别选中为 `debug`，这时只有我们使用 `debug` 及以上级别方法打印的日志才会显示出来，以此类推。你可以做一下试验，当你把 `logcat` 中的级别选中为 `info`、`warn` 或者 `error` 时，我们在 `onCreate()` 方法中打印的语句是不会显示的，因为我们打印日志时使用的是 `Log.d()` 方法。

日志级别控制的好处就是，你可以很快地找到你所关心的那些日志。相信如果让你从上千行日志中查找一条崩溃信息，你一定会抓狂的吧。而现在你只需要将日志级别选中为 `error`，那些不相干的琐碎信息就不会再干扰你的视线了。

最后我们再来看一下关键字过滤。如果使用过滤器加日志级别控制还是不能锁定到你想查看的日志内容的话，那么还可以通过关键字进行进一步的过滤，如图 1.35 所示。

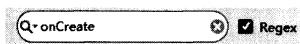


图 1.35 关键字输入框

我们可以在输入框里输入关键字的内容，这样只有符合关键字条件的日志才会显示出来，从而能够快速定位到任何你想查看的日志。另外还有一点需要注意，关键字过滤是支持正则表达式的，有了这个特性，我们就可以构建出更加丰富的过滤条件。

关于 Android 中日志工具的使用我就准备讲到这里，`logcat` 中其他的一些使用技巧就要靠你自己去摸索了。今天你已经学到了足够多的东西，我们来总结和梳理一下吧。

1.5 小结与点评

你现在一定会觉得很充实，甚至有点沾沾自喜。确实应该如此，因为你已经成为一名真正的 Android 开发者了。通过本章的学习，你首先对 Android 系统有了更加充足的认识，然后成功将 Android 开发环境搭建了起来，接着创建了你自己的第一个 Android 项目，并对 Android 项目的目录结构和执行过程有了一定的认识，在本章的最后还学习了 Android 日志工具的使用，这难道还不够充实吗？

不过你也别太过于满足，相信你很清楚，Android 开发者和出色的 Android 开发者还是有很大的区别的，你还需要付出更多的努力才行。即使你目前在 Java 领域已经有了不错的成绩，我也希望在 Android 的世界你可以放下身段，以一只萌级小菜鸟的身份起飞，在后面的旅途中你会不断地成长。

现在你可以非常安心地休息一段时间，因为今天你已经做得非常不错了。储备好能量，准备进入到下一章的旅程当中。