第 1 章 Android 开发基础

Android 是一种以 Linux 为基础的开放源代码操作系统，主要使用于便携设备。目前尚未有统一中文名称，中国大陆地区大多数人使用“安卓”或“安致”。Android 操作系统最初由 Andy Rubin 开发，主要用于支持手机。2005 年由 Google 收购注资，并组建开放手机联盟，进行开发改良，逐渐扩展到平板电脑及其他领域。2008 年，在 Google I/O 大会上，谷歌提出了 Android HAL 架构图，并在同年 9 月正式发布了 Android 1.0 系统，这也是 Android 系统最早的版本。到目前为止，最新的正式版本是 2016 年 8 月发布的 Android 7.0 Nougat(牛轧糖)。截止到 2016 年 4 月的前三个月数据统计，Android 系统的全球市场份额已经达到了 76%，在美国市场该期间内的份额为 67.6%，在中国大陆城市地区该期间内的市场份额为 78.8%。目前 Android 的主要竞争对手是 Apple 的 IOS。

1.1 Android 入门

Android 系统是基于 Linux 平台的智能手机操作系统。Android 是一个开源的平台，它由操作系统、中间件、用户界面和应用软件等部分组成。

Android 系统采用软件堆层(Software Stack，又名软件叠层)的系统架构，由多个程序组合来完成一个共同的任务。Android 的系统架构主要分为三个层次：底层以 Linux 内核为基础，由 C 语言开发，是移动设备的操作系统，只提供基本功能；中间层包括函数库(Library)和虚拟机(Virtual Machine)，由 C++ 开发；最上层是各种应用软件组成，包括通话程序、短信程序和游戏等。

Android 系统作为一个移动开放平台，提供了与 Apple 移动系统不同的生态环境。任何手机厂商都可以免费使用 Android 系统来定制自己的产品，而且 Android 系统提供了标准的 SDK，应用软件可以由第三方开发者独立完成，目前应用的开发语言使用 Java。

1.1.1 Android 简介

从 Andoird 1.5 版本开始，Android 选择使用甜点名称作为系统版本的代号，其版本的名称分别为纸杯蛋糕(Cupcake)、甜甜圈(Donut)、松饼(Eclair)、冻酸奶(Froyo)、姜饼(Gingerbread)、蜂巢(Honeycomb)、冰激凌三明治(Ice Cream Sandwich)、果冻豆(Jelly Bean)、巧克力(KitKat)、棒棒糖(Lollipop)、棉花糖(Marshmallow)、牛轧糖(Nougat)。而且版本名称的英文名首字母按照字母的顺序，从 C 开始到 N。目前最新的版本 Android 7.0 (API level 24) Nougat 的下一个版本，应该是以 O 作为首字母的甜点名称。

表 1.1 是 Android 各版本对应的名称和 API Level 所对应的市场份额，可以在程序开发和发布时作为参考。

表 1.1 Android 的各个版本

版　本	名　称	API Level	比例
2.2	Froyo	8	0.1%
2.3	Gingerbread	10	1.7%
4.0	Ice Cream Sandwich	15	1.6%
4.1	Jelly Bean	16	6.0%
4.2		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%
7.0	Nougat	24	0

Google 以 7 天为一个周期对 Android 版本的使用情况进行统计，表 1.1 中的比例表示了 2016 年 8 月 1 日为止，Android 设备中的不同版本分布比例。根据表 1.1 的数据，图 1.1 显示了目前版本的使用分布。从图 1.1 可以看出，目前手机上用得最多的版本为 4.4 的 KitKat 和 5.0、5.1 的 Lollipop，但 Marshmallow 已经占有了不少的份额，如果要发布应用，需要选择适合的 Android 版本，方便在不同的版本中支持该应用。

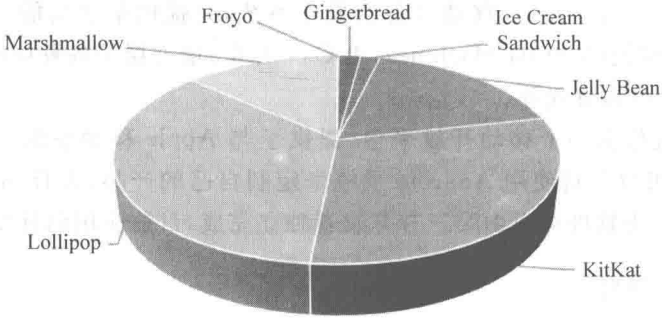

图 1.1 Android 版本分布

1.1.2 Android 技术架构

Android 技术架构见图 1.2。Android 技术架构由五部分组成，从顶层到底层，依次是 Application、Android Framework、Native Libraries、Android Runtime（ART）、

Hardware Abstraction Layer(HAL)、Linux Kernel。如果是 Android 应用开发者，只需要了解 Application 和 Android Framework 上面两个层次；如果是嵌入式和硬件移植的开发者，还需要了解 Native Libraries、Android Runtime(ART)、Hardware Abstraction Layer (HAL)和 Linux Kernel 这几个部分。

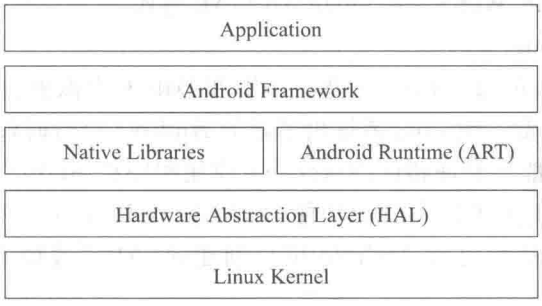

图 1.2　Android 技术架构

1. Application

Android 配置一个核心应用程序集合，包括电子邮件客户端、SMS 程序、日历、地图、浏览器、联系人、照相机、E-mail、时钟、拨号和其他设置。所有应用程序都是用 Java 编程语言写的。用户自己开发的 App 应用也在这个层次。

2. Android Framework

Android 提供开放的应用框架 Android Framework，应用开发者可以方便地设计丰富和新颖的应用程序。通过应用框架，开发者能够自由地利用设备硬件、使用访问位置信息、运行后台服务、设置闹钟，并且向状态栏添加通知等功能。由于这个应用开发框架是完全开放的，应用开发者能够使用框架的核心功能。

设计应用框架的目的在于方便组件的重用，简化应用程序的开发，而且应用程序都可以发布和使用应用框架中的功能。应用框架主要是由下面的功能组成的：

- 视图系统(View System)——包括丰富的、可扩展的视图集合，可用于构建一个应用程序，包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器。
- 内容提供者(ContentProviders)——使应用程序能访问其他应用程序(如通讯录)的数据，或共享自己的数据。
- 管理器(Manager)——包括许多管理器。其中资源管理器(Resource Manager)为应用程序提供可访问的非代码资源，如本地化字符串、图形和布局文件等；通知管理器(Notification Manager)支持所有的应用程序能在状态栏显示自定义警告；活动管理器(Activity Manager)管理用户界面应用程序的生命周期，提供通用的导航回退功能；位置管理器(Location Manager)支持移动设备上基于位置和地图的应用，可以和 Google Location Services API 配合使用，实现更强大的框架。

3. Native Libraries

Native Libraries(本地库)是由 Android 提供的一系列本地头文件和共享库文件，应用程序通过 Android Framework 调用。Native Libraries 支持使用硬件传感器、访问存储

器、处理用户输入、配置信息设置等功能。随着 Android 版本和 Android API Level 的不断更新，Native Libraries 的功能逐渐增长。到目前为止，包括 C/C++ 库、ZLib 压缩库、动态链接库、嵌入式 3D 图形加速标准 OpenGL ES 库、分配和管理 OpenGL ES 的 EGL 库、嵌入式音频加速标准 OpenSL ES 库、FreeType 位图和矢量字体处理库、强大而轻量级的关系数据库引擎 SQLite、WebKit 和 OpenMAX AL 等库。

4. Android Runtime

Android Runtime(ART)包括核心库、ART 和 Dalvik 虚拟机。

从 Android 4.4 开始，Adnroid 就推出了新的 Android 运行时(ART)，替代之前版本的 Dalvik。ART 的功能是管理和运行 Android 应用程序和 Android 的一部分系统服务。与 Dalvik 相比，ART 提供了许多新的功能，来改善 Android 平台和应用的性能。ART 和它的前身 Dalvik 都是专门为 Android 项目创建的，ART 遵循 Dalvik 可执行格式和 Dex 字节码规范。

ART 和 Dalvik 运行 Dex 字节码时是兼容的，因此基于 Dalvik 开发的应用也可以在 ART 上运行。但有时候，基于 Dalvik 的技术无法在 ART 上实现。

5. Hardware Abstraction Layer

硬件抽象层(Hardware Abstraction Layer，HAL)为硬件厂商定义了一个标准接口，Android 系统通过这个标准接口使用硬件实现的功能，底层的硬件驱动实现对于 Android 系统透明。也就是说，Android 的上层应用只需调用这些标准接口来实现软件的功能，不必了解具体是哪家厂商的产品，如何实现。

HAL 使得 Android 系统的上层软件功能与硬件实现隔离开，不同硬件的支持，不会影响也不需要修改上层软件系统。

当然，针对不同的厂商产品，需要开发对应的 HAL。HAL 的实现部分以. so 的文件模式存储在共享库模块中。

6. Linux Kernel

Android 的 Linux Kernel 提供操作系统的核心系统服务，包括安全管理、内存管理、进程管理、网络堆栈、电源管理和驱动模型等。例如声音、显示、相机、蓝牙、Wi-Fi 等设备的驱动，都在这一层实现。

1.2 Android Studio 环境搭建

在进行 Android 应用程序开发之前，需要搭建 Android 应用程序开发环境。本书中采用开源的 Android Studio 2.1.2 集成开发环境开发工具。有关 Eclipse 平台开发 Android 的开发环境搭建，参看附录 A。

Android Studio 是 Android 开发的官方集成开发环境(IDE)，提供开发 Android App 所需要的各种支持工具和软件包。安装完的 Android Studio，具有下面的功能：

- IntelliJ IDE+Android Studio plugin；
- Android SDK 工具包；
- Android 平台工具；

- Android开发平台；
- 包含Google Play Service、带有Android系统图像的Android模拟器。

由于Android Studio把Android应用程序开发所需的运行环境、库、开发工具和界面都集成为一个包，因此安装比较简单。具体安装过程见1.2.1节。如果已经熟悉Android Studio开发环境的安装，可以跳过1.2节，直接阅读后面的内容。

Android Studio针对Windows、Mac、Linux不同的操作系统，提供相应的安装包。具体的下载网址为https://developer.android.com/studio/index.html。

在下载完成Android Studio安装包之后，首先需要确认操作系统已经安装了JDK，并且版本在1.8以上。下面针对不同操作系统，介绍Android Studio具体的安装步骤。

1.2.1 基于Windows的安装

具体的安装步骤如下：

(1) 打开命令行窗口，输入javac -version命令，如果JDK不存在，或JDK的版本低于1.8，从网址http://www.oracle.com/technetwork/java/javase/downloads/ jdk8-downloads-2133151.html下载Java SE Development Kit 8，并进行安装。

(2) 运行所下载的Android Studio安装包.exe文件。

(3) 根据安装提示安装Android Studio和SDK工具。

(4) 在安装完成界面，选中Start Android Studio选项，单击Finish按钮，启动Android Studio设置，进行初始设置。

(5) 根据需要选择Android Studio预定的设置，或者选择自定制的设置，然后单击OK按钮，见图1.3。建议选择默认选项，即位于下面的Android Studio预定设置选项，继续下面的配置。这个过程包括下载Android的组件和工具包，并进行安装，需要的时间会有点长，请耐心等待，直到完成配置过程，出现Android Studio启动界面，见图1.4。关闭Android Studio启动界面，就完成了安装。

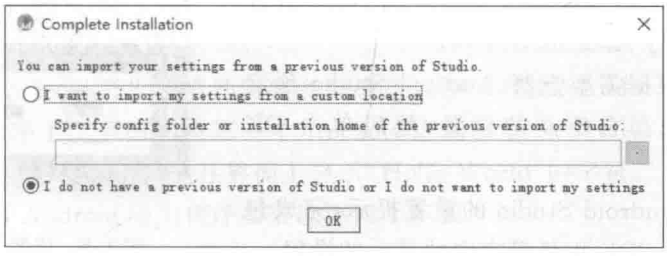

图1.3 配置选择

(6) 设置Windows系统环境变量。注意，在某些Windows系统中，Android Studio安装脚本找不到JDK的路径，如果遇到这种情况，需要在环境变量中设置JDK路径。具体操作如下：选择Start→Computer→System Properties→Advanced System Properties，然后选择Advanced→Environment Variables，增加一个新的系统变量JAVA_HOME指定JDK路径，例如，C:\Program Files\Java\jdk1.8.0_77。

到此，基于Windows的Android Studio安装完成，下一步可以进行创建Android项

图 1.4　Android Studio 启动界面

目的工作了。

1.2.2　基于 Mac 的安装

具体的安装步骤如下：

（1）打开命令行窗口，输入 javac -version 命令，如果 JDK 不存在，或 JDK 的版本低于 1.8，从网址 http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html 下载 Java SE Development Kit 8，并进行安装，见图 1.5。

（2）运行所下载的 Android Studio DMG 文件。

（3）把 Android Studio 拖曳到应用 Application 文件夹，然后运行 Android Studio。

（4）自己根据需要选择 Android Studio 预定的设置，或者选择自定制的设置，然后单击 OK 按钮。

（5）根据 Android Studio 的设置提示，完成包括下载 Android SDK 组件等在内的其余的设置。

图 1.5　安装基于 Mac 的 Android Studio

安装过程中的一些细节，可以参考 Windows 安装的过程。到此，基于 Mac 的 Android Studio 安装完成，下一步可以进行创建 Android 项目的工作了。

1.2.3　基于 Linux 的安装

具体的安装步骤如下：

（1）打开命令行窗口，输入 javac -version 命令，如果 JDK 不存在，或 JDK 的版本低

于 1.8，从网址 http://www.oracle.com/technetwork/java/javase/downloads/ jdk8-downloads-2133151.html 下载 Java SE Development Kit 8，并进行安装。

(2) 在合适的目录下，例如/usr/local 目录，将所下载的 Android Studio 安装包.zip 文件解压缩。

(3) 运行 Android Studio，打开 Terminal，进入 android-studio/bin/目录，执行 studio.sh，见图 1.6。建议把 android-studio/bin/的路径加入 PATH 环境变量，以便在任何目录下都可以启动 Android Studio。

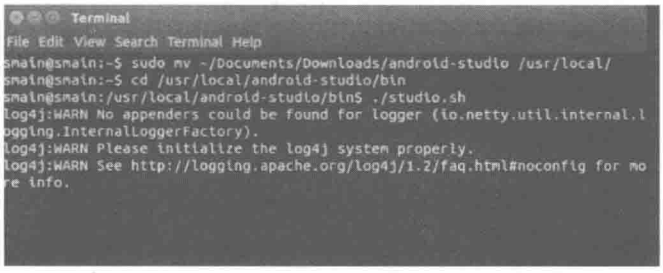

图 1.6　Linux Terminal 中的命令

(4) 自己根据需要选择 Android Studio 预定的设置，或者选择自定制的设置，然后单击 OK 按钮。

(5) 根据 Android Studio 的设置提示，完成包括下载 Android SDK 组件等在内的其余的设置。

(6) 安装过程中的一些细节，可以参考 Windows 安装的过程。到此，基于 Linux 的 Android Studio 安装完成，下一步可以进行创建 Android 项目的工作了。

1.3　第一个 Android 应用程序

1.3.1　创建 Android 项目

在完成了 Android 开发环境的安装和配置后，就可以使用 Android Studio 开始开发 Android 应用程序了。下面介绍第一个 Android 应用程序——Hello Mobile World——的开发和运行过程，它的功能是在界面上显示“Hello World”的字符。

第一次编写 Android 应用程序需要以下四个步骤：

- 创建一个新的 Android 项目。
- 运行应用程序。
- 定义简单的用户界面。
- 启动另一个 Activity。

(1) 在 Windows 下运行 Android Studio，出现图 1.7 所示的 Android Studio 启动界面后，选择第一项 Start a new Android Studio project。

(2) 在新项目的配置界面，填写 Application name、Company Domain 和 Project location 对应的文本框，见图 1.8。

图 1.7 Android Studio 启动界面

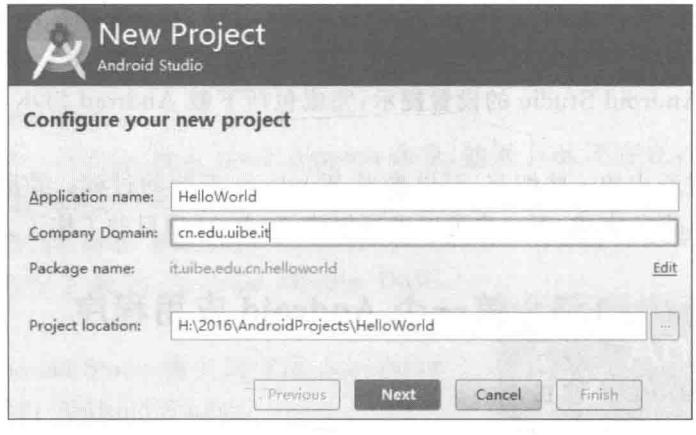

图 1.8 项目配置界面

其中，应用名称 Application name 指计划要开发的 App 名称，这里的 App 名称为“HelloWorld”；公司域名 Company Domain 会体现在 App 的 Java 包的配置上，Package 的名称刚好与其相反，并反向对应 Java 源代码存储的目录；项目位置 Project location 是指整个项目所在的路径，可以通过文本框后面的按钮，在计算机中选择合适的存储位置。

(3) 在选择 Android 运行的设备的界面上，选择 Phone and Tablet 复选框，让 App 的运行环境设置为手机和平板电脑，见图 1.9。

如果在后面的开发中，要开发可穿戴设备、电视、音频和眼镜方面的应用，可以根据需要选择对应的复选框，来运行和调试 App。

(4) 在 Add an Activity to Mobile 界面，选择默认选项 Empty Activity，单击 Next 按钮。图形用户界面是通过类 Activity 的子类来实现的，这些子类统称为 Activities。

图 1.9　选择 Android 运行的设备

(5) 在 Activity 配置界面，填写 Activity Name、Layout Name 对应的文本框，见图 1.10。

图 1.10　新 Activity 配置界面

其中，Activity Name 对应 Java 源代码中 Activity 子类的名称；Layout Name 指 XML 布局文件的名称，也就是说，Android 系统可以通过 XML 文件，设置 Android 手机和平板电脑的用户界面具体有哪些组件，如何显示。在这里，选用 Android Studio 默认的名称。

(6) 单击 Finish 按钮，完成 Android Studio 新项目创建的设置，就可以看到 Android Studio 开发界面了。

(7) 关闭提示对话框，单击开发界面左边界的 Project 标签，打开 Project 的内容，在左边的窗口可以看到刚才所创建项目的目录结构。展开相应的目录，双击 java 目录下的

MainActivity，以及 res/layout 目录下的 activity_main. xml 文件，在右边的窗口可以看到文件的内容，见图 1. 11。

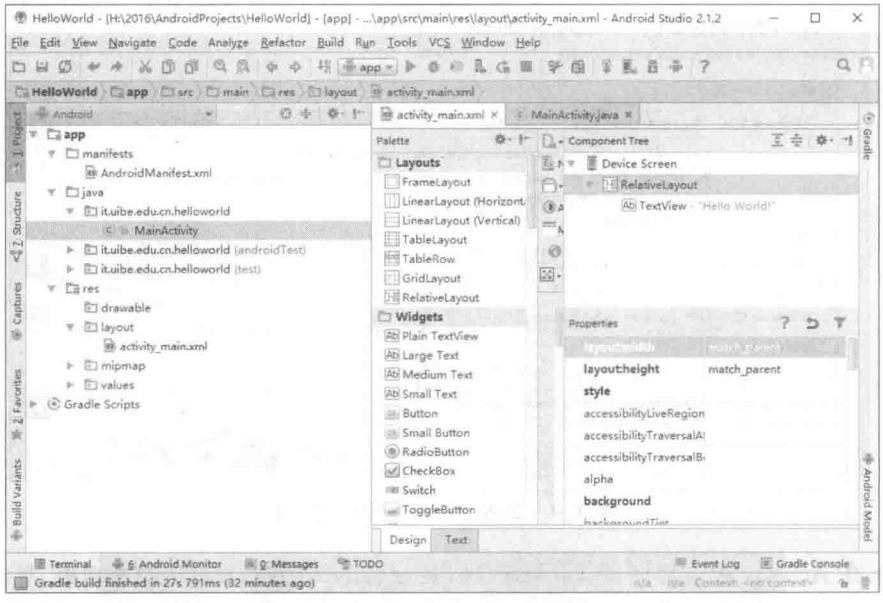

图 1. 11　Android Studio 开发界面

到这里，就完成了一个新项目的创建。在这个新创建的 HelloWorld App 项目中，默认创建了一些缺省文件，分别存放在不同的目录下，见图 1. 11。下面对其中两个重要的文件 MainActivity. java 和 activity_main. xml 进行一下说明。

代码 1. 1 是 java 目录下的 MainActivity. java 文件的具体内容。

代码 1. 1　MainActivity. java 文件

```
package it.uibe.edu.cn.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

从代码 1. 1 中可以看出 MainActivity 是 Activity 的子类。Activity 是 Android 系统中用于实现用户图形界面的类。

protected void onCreate()方法是Activity所定义的方法，当Activity启动时由Android系统调用，首先在这个方法中执行初始化和用户界面设置工作，Java源程序的setContentView()方法，通过R.layout.activity_main类调用了activity_main.xml定义的用户界面，运行时在屏幕上显示“Hello World!”。一个Activity并不一定要有用户界面，但通常都会有。Activity概念可以参照Applet来理解。

代码1.2是activity_main.xml文件的具体内容。

代码1.2 activity_main.xml文件

```
<?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"
    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="it.uibe.edu.cn.helloworld.MainActivity">

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

在这个XML布局文件中，元素TextView下的android:text="Hello World!"语句定义了屏幕显示的内容为“Hello World!”。该布局文件的其他功能和具体代码的含义在后续章节解释。

进行完上一步的工作后，就可以运行App了。

Android App的运行可以在两个不同的环境中进行选择：真正的Android设备和Android模拟器。接下来详述Android设备和Android模拟器上安装和运行Android App的具体过程。

1.3.2 在手机上运行HelloWorld App

下面以Samsung SM-A8000型号的手机为例，进行环境配置，运行App，其他的手机配置过程类似。

(1) 通过USB将手机连接到计算机。

将用于测试的手机用USB连线连接到计算机。

(2) 设置手机的USB调试选项。

打开手机的设置界面，打开“开发者选项”，见图1.12左图；然后在打开后的“开发者选项”配置界面中，将第一项设置为“开”的状态，“USB调试”设置为“开”的状态，见图1.12

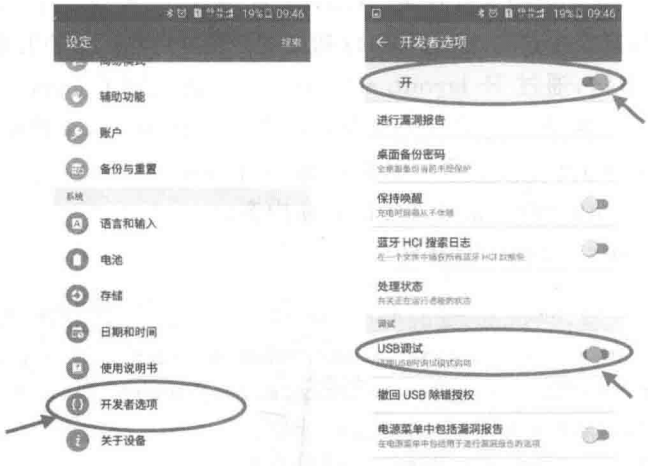

图 1.12　手机的 USB 调试选项设置

右图。

（3）运行 Android Studio 中的应用程序 HelloWorld App。

在 Android Studio 中，单击图 1.13 中所示的 App 运行按钮 ▶，尝试运行 HelloWorld App 程序。因为这是第一次运行 App，所以平台会给出运行设备选择的对话框，请先进行运行设备的选择或配置。

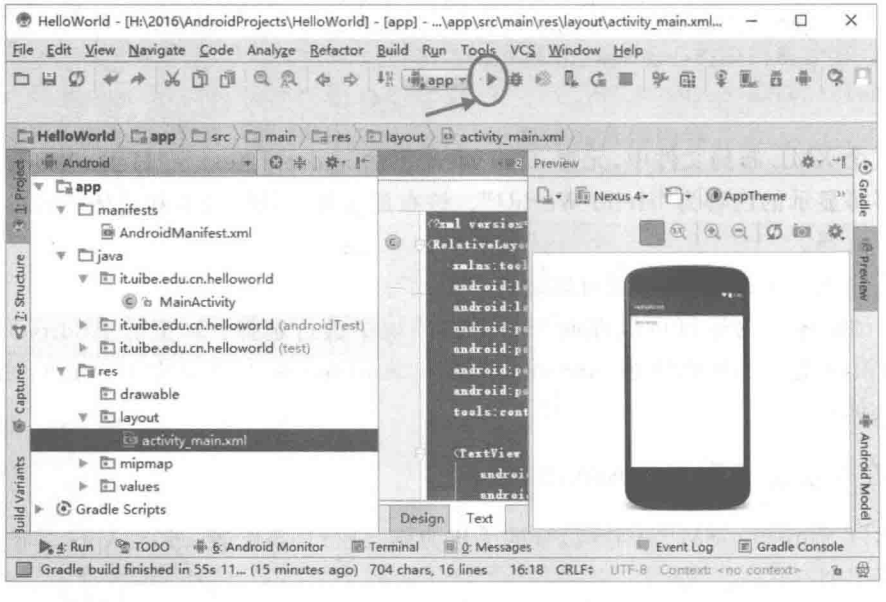

图 1.13　Android Studio 开发界面

（4）选择运行设备。

因为前面已经设置了手机的开发者 USB 调试选项，成功与设备连接后，在运行设备选择对话框中会出现所连接的手机选项，见图 1.14。

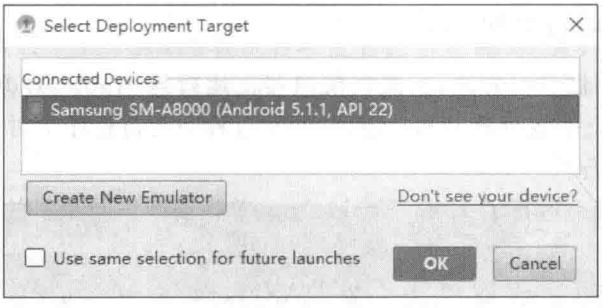

图 1.14　选择设备对话框

选择手机设备作为运行设备，单击 OK 按钮，在手机上运行 HelloWorld App。

(5) App 在手机上安装运行。

在手机上运行 App 的速度很快，几乎马上就可以从手机上看到图 1.15 左图所示的结果。从 App 返回手机的主界面，可以看到屏幕上出现一个绿色小机器人，表明 HelloWorld App 已经装载到了手机上，可以单击其图标。

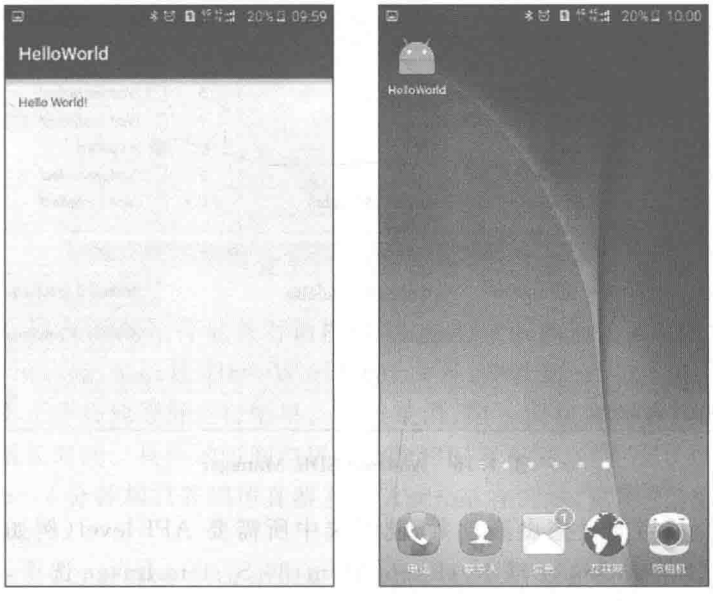

图 1.15　手机运行结果

在手机上执行 App，实际上是把 Android Studio 编译后的 apk 文件装载到手机并执行的过程，相当于从网上下载应用，并安装后执行的过程。在开发过程中，可以一直使用手机来进行开发 App 的调试和测试，但大多数情况下，会使用 Android 手机模拟器，即 AVD 来进行调试、运行和测试。因为 AVD 可以根据需要设置成不同 Android 版本、不同显示模式、不同性能、不同厂商的模拟运行环境，对 App 进行多方面的测试，有利于所开发 App 的兼容性、健壮性和适应性。

1.3.3 在 AVD 上运行 HelloWorld App

不对上面的程序做任何修改，下面在模拟器上运行该 App。Android 模拟器能够模拟 App 在真实手机上的运行效果和功能，在开发过程中支持程序员对 Android App 进行代码测试、调试和运行。

如果刚才在手机上运行了 HelloWorld App，首先要断开与手机的 USB 连接，然后再进行下面的步骤。

（1）在创建 AVD 之前，需要先安装 HAXM，即 Hardware Accelerated Execution Manager，Intel 的硬件加速执行管理器。

选择 Tool→Android→SDK Manager，单击对话框中的蓝色链接 Launch Standalone SDK Manager，打开独立的 Android SDK Manager，将右边的滚动条拖曳到最底部，可以看到最后一个选项 Intel x86 Emulator Accelerator（HAXM installer），见图 1.16。

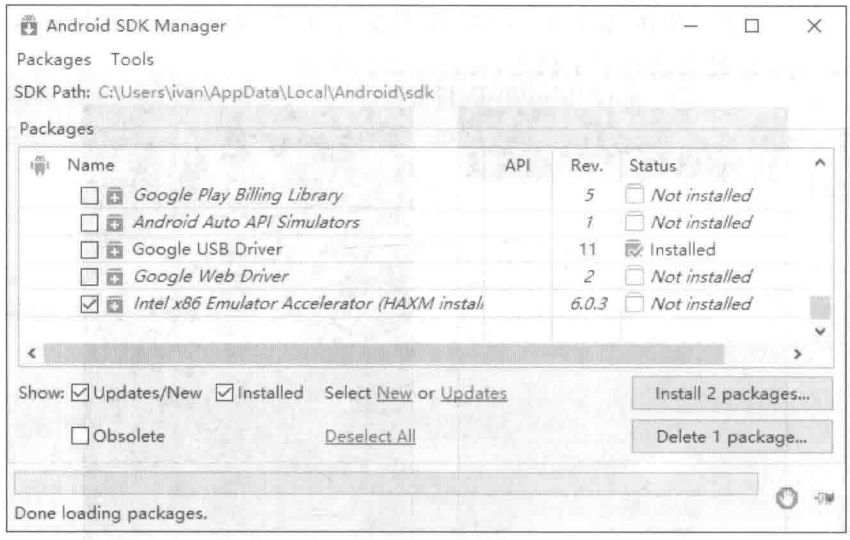

图 1.16 Android SDK Manager

选中该选项，然后向上拖曳滚动条，同时选中所需要 API level（例如 API 23）中的 Intel x86 Atom System Image 或 Intel x86 Atom_86 System Image 选项，单击右下的按钮 Install 2 packages，安装这两个包。

要注意的是，虽然安装完成，在 Status 栏显示 Installed，但 HAXM 并没有真正安装到系统，还需要进入 HAXM 安装包所在的目录手动安装 HAXM。具体路径是 Windows 用户目录下的 AppData\Local\Android\sdk\extras\intel\Hardware_Accelerated_Execution_Manager。

（2）单击 Android Studio 的运行按钮，运行 Android Studio 中的应用程序 HelloWorld App。

（3）在图 1.14 所示的选择设备对话框中单击 Create New Emulator 按钮，创建运行 App 的模拟设备 AVD（Android Virtual Device）。

(4) 在接下来的对话框中选择模拟的手机型号、Android 的版本及 API level，进行模拟器配置，然后在最后一个对话框中检查各项配置是否正确，在 AVD Name 对应的文本框中给出 AVD 的名称，单击 Finish 按钮，完成 AVD 创建，见图 1.17。第一次配置可以采用默认设置。

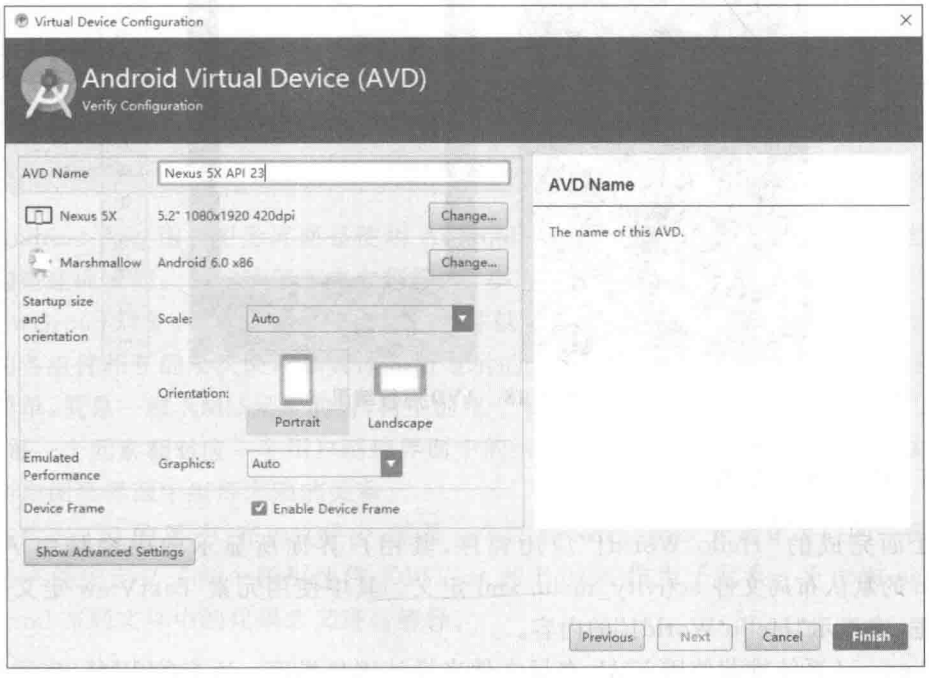

图 1.17　AVD 配置

(5) 在图 1.14 所示的选择设备对话框中，双击刚才所创建的 AVD，Android Studio 会把 HelloWorld App，也就是 HelloWorld.apk 文件，安装到 AVD 上并运行。

这时需要真正耐心地等待运行结果。一般来说，第一次在仿真器上运行 Android 应用程序需要花很长时间。具体的时间与机器性能和配置有关。

如果运行 3～4 分钟却只看到仿真器上的 Android 在闪烁，见图 1.18 左图，此时不要着急，请耐心等待。

几分钟之后，可以看到图 1.18 中图所示的运行结果显示在屏幕上。图 1.18 右图的工具条是对模拟器手机进行操作的各种功能按钮，可以实现关机、调节声音大小、摇动、照相等功能。如果单击 O，就回到手机 Home 的状态，通过单击应用程序图标，可以看到手机上装载的应用程序，从中找到刚才所运行的 HelloWorld App 的白底绿色小机器人图标。

(6) AVD 可以创建多个，如果要在后续的 App 开发过程中默认使用某个 AVD，在双击它之前，选中图 1.14 中的 Use same selection for future launches 复选框。

Android Studio 的 AVD 除了可以在运行 App 时在选择设备配置时创建，还可以在其提供的 AVD 管理工具里进行创建、删除、修改等操作，具体的操作通过选择 Tool→Android→AVD Manager，在弹出的对话框中进行。

图 1.18　AVD 运行结果

1.3.4　定义简单的用户界面

上面完成的“Hello World!”应用程序，其用户界面所显示的内容是由 Android Studio 的默认布局文件 activity_main. xml 定义。其中使用元素 TextView 定义了一个文本框，来显示“Hello World!”的内容。

Android 系统推荐使用 XML 布局文件来设计用户界面。所有的 XML 布局文件都位于 App 项目的 res/layout/目录中。使用这种方式，可以很方便地定义结构化的用户图形界面，图形组件的布局和相互之间的关系可以很容易、清楚地设定。同时也可以避免直接在源代码中构建应用程序的用户界面，导致一些小的布局变化引起 Java 源代码的大量修改和重新编译。

在 Android Studio 中提供两种定义 XML 布局文件的方式：图形化构建和编写 XML 代码。在这里，采用直接编写 XML 代码的方式，定义一个自己简单的用户界面，替换前面的“Hello World!”用户界面。

1. 编写布局文件

在 Project 的 app 目录中，展开 res/layout/文件夹，双击打开 activity_main. xml 文件，清除原来的所有内容，替换成代码 1.3 中的内容，编写自己的用户界面布局。

代码 1.3　自己定义的 activity_main. xml

```
<?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:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

Android App用户图形界面是使用Android系统的View和ViewGroup子类，按层次结构搭建而成的。例如按钮和文本框等UI小组件是View对象，而列表和网格等容器是ViewGroup对象。Android XML布局文件就是通过树形结构，来说明在屏幕上所要显示的各组件相互的层次关系和具体如何显示的。一个Android XML文件的总体结构比较简单，就是一棵XML元素的树，树中的每个节点都使用View类的子类名作为元素名称，每一个元素都对应一个用户图形界面中的组件。通过树的根和分支之间的关系，定义了用户图形界面中组件之间的关系。

在代码1.3的XML布局文件中，就是一个以LinearLayout元素作为根的一棵树，在该布局文件中定义了两个图形组件EditText和Button作为子元素。下面对acitvity_main.xml布局文件中的代码含义进行解释。

1) android:id

这个属性为子元素所定义的View对象提供了唯一的标识，应用程序可以通过这个ID识别、操作或获取这个对象。

另外，在XML文件中引用任何一个其他XML文件中的资源时，，需要以@开头，后面跟随资源的类型、一个斜线和资源名称。例如，"@string/edit_message"表示应用string类型的资源，这个资源的名称是edit_message。string资源在/res/strings中定义。

如果在@后、资源类型前添加一个+，表示定义一个资源的ID。

2) android:layout_width和android:layout_height

这两个属性定义View对象的可用宽度和长度。值match_parent表示占据其父节点ViewGroup所占据的整个空间；值wrap_content表示根据View的内容大小来设定对象的大小。通过设定不同的值，View对象的外形有所变化，执行性能也会有所不同。例如在这里EditText的属性中layout_weight="1"和layout_width="0dp"配合使用，会减少系统计算量，改善应用的性能。

3) android:hint

这个属性设定EditText中默认显示的字符串。如果没有预先设定这个属性，EditText初始显示为空。这里没有直接给这个属性赋一个字符串常量值，而是使用了一个string资源引用@string/edit_message，指向字符串资源文件中定义的名称为edit_message的资源。字符串定义在res/values/strings.xml文件中。这是一种推荐的方式，