任务 T1

开启学生空间 App 的开发之旅

Android 手机是目前广泛使用的智能手机，如何开发一款心仪的 Android 应用，并发布到应用商店供大家下载使用呢？从现在开始，本书将带领大家一步步开发学生空间 App。本任务将带领大家认识 Android，了解 Android 的发展；并讲解 Android 应用开发所需的开发环境，如何创建、运行和调试一个 Android 应用，以及 Android 应用的目录结构等。

任务 T1-1　什么是 Android

任务目标

- 了解 Android 的历史与发展;
- 了解 Android 的平台架构;
- 熟悉 Android 开发环境的搭建;
- 熟悉 Android 项目的创建与运行;
- 熟悉 Android 模拟器的使用方法;
- 了解 Android 项目打包发布的步骤。

任务分析

本子任务是为学生空间 App 创建欢迎界面，在 Android Studio 创建第一个版本的学生空间 StuSpace 项目工程，并在 Android 模拟器中运行，界面实现效果如图 1-1 所示。

图 1-1　学生空间欢迎界面的运行效果图

1.1.1 Android 系统概述

目前，应用在智能手机的操作系统主要有 Android（谷歌）、iOS（苹果）、Windows phone（微软）、Symbian（诺基亚）等，最主流的当属谷歌的 Android 和苹果的 iOS。

Android 是基于 Linux 内核、开放源码的操作系统，主要应用于便携设备，如智能手机、平板电脑等。第一部 Android 智能手机发布于 2008 年 10 月，此后 Android 逐渐扩展到平板电脑及其他领域上，如电视、数码照相机、游戏机等。2011 年第一季度，Android 在全球的市场份额中首次超过塞班系统，跃居全球第一；2015 年第四季度，Android 平台手机的全球市场份额已经达到 78.1%。

1.1.2 Android 的历史与发展

Android 操作系统最初由安迪·鲁宾（Andy Rubin）开发，主要用于支持手机，2005 年 8 月由谷歌公司收购注资。2007 年 11 月，谷歌与 84 家硬件制造商、软件开发商及电信营运商成立了开放手机联盟（Open Handset Alliance，OHA），共同研发改良了 Android，随后，谷歌以 Apache 免费开放源码的许可证的授权方式，发布了 Android 的源代码。

Android 用甜品的名称作为它们系统版本代号的命名，从 Android 1.5 发布开始，作为每个版本代表的甜品的尺寸越变越大，并按照 26 个字母排序：纸杯蛋糕（Cupcake），甜甜圈（Donut），松饼（Eclair），冻酸奶（Froyo），姜饼（Gingerbread），蜂巢（Honeycomb）等。

至今，Android 已经经历了多个版本的更新，截止到本书完稿之时，最新的版本为 2015 年 5 月 28 日发布的 Android 6.0，代号为 Marshmallow（棉花糖）。根据市场对当前 Android 系统版本的分布报告显示，Android 6.0 Marshmallow 的市场份额还比较低，而 Android 4.4 KitKat 的份额攀升至 33.4%，占据了主导地位。从 Android 4.0 开始，Android 系统有了一个质的飞跃，本书将专注讲解 Android 4.0 及以上版本的开发。

Android 平台的优势如下：

1．平台开放性

Android 的优势首先就是其开放性，开放的平台允许任何移动终端厂商加入到 Android 联盟中。开放性使其拥有更多优秀的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，最大的受益正是其丰富的应用软件资源。

2．硬件丰富性

这一点是与 Android 的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪、功能各异的多种产品。功能的差异和界面的特色并不会影响到数据同步、软件兼容性，如同

从 Android 风格手机改用苹果 iPhone 风格，同时可将 iPhone 中优秀的软件带到 Android 上使用一样，联系人等资料更是可以方便地转移。

3．开发便捷性

Android 提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻挠，可想而知，会有多少新颖的软件诞生。

4．和 iPhone 相比，具有更广泛的开发群体

从技术角度而言，Android 是一种融入所有 Web 应用的平台，随着 Android 版本的更新，从最初的触屏到现在的多点触摸，从普通的联系人到现在的数据同步，从简单的谷歌地图到现在的导航系统，从基本的网页浏览到现在的 HTML 5，这都说明 Android 已经逐渐稳定，功能越来越强大。

另外，Android 不仅支持 Java、C、C++等主流编程语言，还支持 Ruby、Rython 等脚本语言，谷歌甚至专门为 Android 应用开发推出了 Simple 语言，这使得 Android 有着非常广泛的开发群体。

1.1.3 Android 体系架构及 Dalvik

Android 体系架构主要包括 Application、Application Framework、Libraries、Android Runtime 和 Linux Kernel，如图 1-2 所示。Android 的体系架构分为四个层，从高到低分别是应用程序层（Application）、应用程序框架层（Application Framework）、系统运行库层（Libraries）和 Linux 内核层（Linux Kernel）。

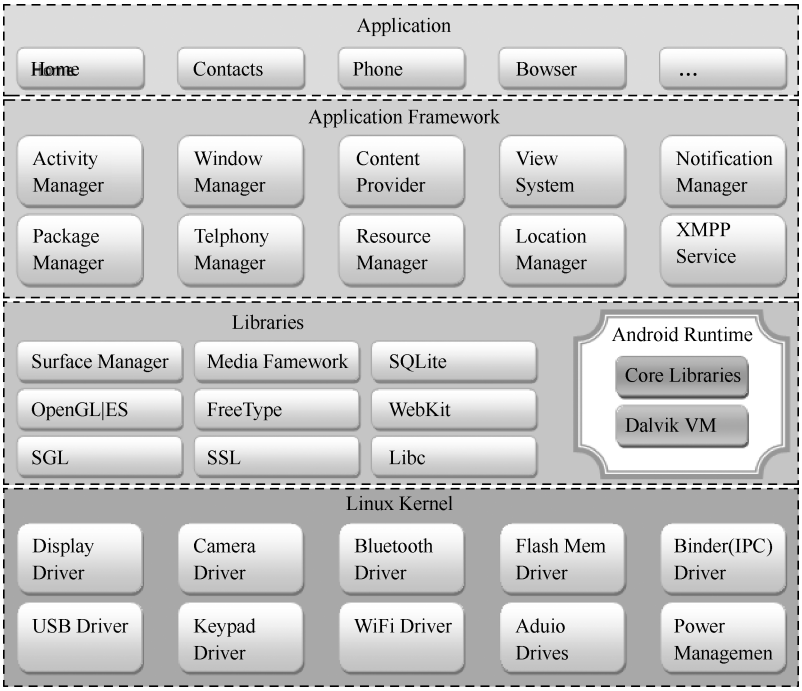

图 1-2　Android 平台架构

Android 操作系统可以在四个主要层面上分为 5 个部分。

（1）应用程序层（Application）：Android 系统包含了一系列核心应用程序，包括电子邮件、短信、日历、拨号器、地图、浏览器、联系人等。本书重点讲解如何编写 Android 操作系统上运行的应用程序，在程序分层上，与系统核心应用程序平级。

（2）应用程序框架层（Application Framework）：Android 应用程序框架提供了大量的 API（Application programing Interface）供开发人员使用，Android 应用的开发就是调用这些 API，实现需求功能。应用程序框架是应用程序的基础，任何一个应用程序都可以开发 Android 系统的功能模块，只要发布的时候遵循应用程序框架的规范，其他应用程序也可以使用这个功能模块。

（3）系统运行库层（Libraries）：Android 系统运行库是用 C/C++语言编写的，是一套被不同组件所使用的函数库组成的集合。一般来说，Android 应用开发者无法直接调用这套函数库，都是通过应用程序框架提供的 API 对这些函数库进行调用的。

（4）Android 运行时：Android 运行时由两部分组成，即 Android 核心库和 Dalvik 虚拟机。其中，核心库集提供了 Java 语言核心库所能使用的绝大部分功能，Dalvik 虚拟机负责运行 Android 应用程序。虽然 Android 应用程序通过 Java 语言编写，但 Java 程序都必须在 Java 虚拟机上运行，由于硬件的限制，Android 应用程序并未使用 Java 的虚拟机来运行程序，而使用了独立的虚拟机——Dalvik 虚拟机，它对多个同时高效运行的虚拟机进行了优化。每个 Android 应用程序都运行在单独的 Dalvik 虚拟机上，因此 Android 系统可以方便地对应用程序进行隔离。

（5）Linux 内核：Android 系统是基于 Linux 内核建立的操作系统，Linux 内核为 Android 系统提供了安全性、内存管理、进程管理、网络协议栈、驱动模型等核心系统服务，帮助 Android 系统实现了底层硬件与上层软件之间的抽象。

Android 应用程序使用了独立的 Dalvik 虚拟机，其与 Java 的虚拟机之间的区别如下。

Java 虚拟机（Java Virtual Machine，JVM）是虚构出来的运行 Java 程序的平台，通过在实际的计算机上仿真模拟各种计算机功能。它具有完善的硬件架构（如处理器、堆栈、寄存器等），还具有相应的指令系统，使用 JVM 使 Java 程序支持与操作系统无关。理论上，在任何操作系统中，只要有对应的 JVM，即可运行 Java 程序。

Dalvik VM 是在 Android 操作系统上运行 Android 程序的虚拟机，其指令集基于寄存器架构的，执行特有的文件格式——dex 字节码，完成对象生命周期管理、堆栈管理、线程管理、安全异常管理、垃圾回收等重要功能。

由于 Android 应用程序的开发语言是 Java，而 Java 程序运行在 JVM 上，因此有些人会把 Android 的虚拟机 Dalvik VM 和 JVM 弄混淆，但是，实际上 Dalvik 并未遵守 JVM 规范，而且两者也是互不兼容的。

JVM 运行的是.class 文件的 Java 字节码，但是 Dalvik VM 运行的是其转换后的 DEX（Dalvik Executable）文件。JVM 字节从.class 文件或者 JAR 包中加载字节码后运行，而 Dalvik VM 无法直接从.class 文件或 JAR 包中加载字节码，它需要通过 DX 工具将应用程序所有的.class 文件编译成一个.dex 文件，Dalvik VM 则运行这个.dex 文件。图 1-3 显示了 Dalvik VM 与 JVM 编译过程的区别。

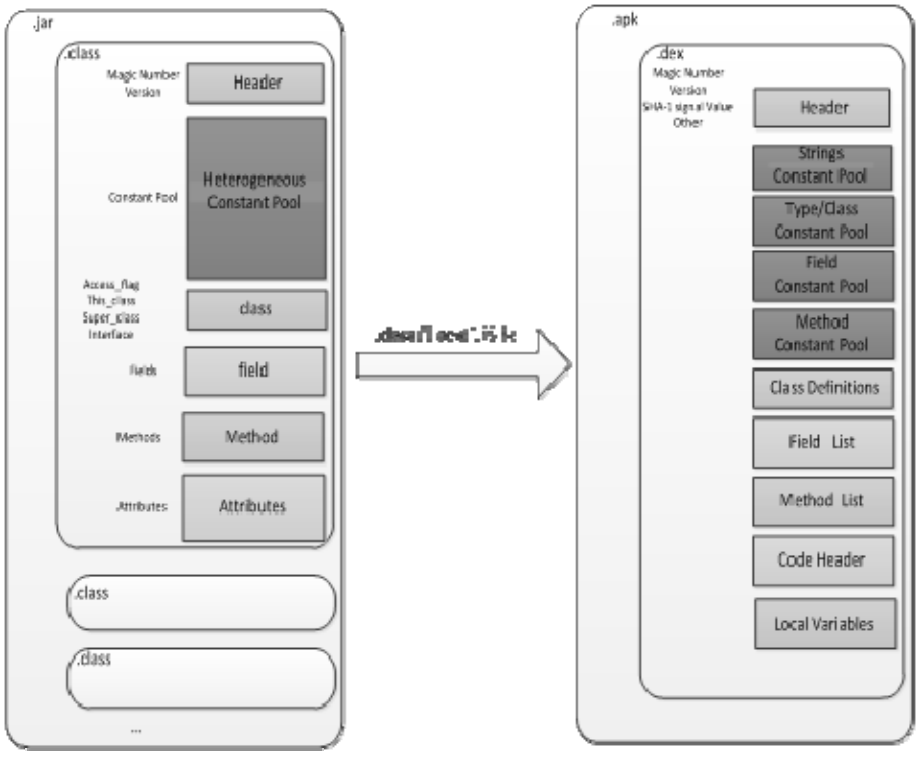

图 1-3　Dalvik VM 与 JVM 编译过程

从图中可以看出，Dalvik VM 把.java 文件编译成.class 后，会对.class 进行重构，整合基本元素（常量池、类定义、数据段），最后压缩写进一个.dex 文件中。其中，常量池描述了所有的常量，包括引用、方法名、数值常量等；类定义包括访问标识、类名等基本信息；数据段中包含各种被 VM 指定的方法代码，以及类、方法的相关信息和实例变量。这种把多个.class 文件进行整合的方法，大大提高了 Android 程序的运行速度，例如，应用程序中多个类定义了字符串常量 TAG，而在 JVM 中，会编译成多个.class 文件，每个.class 文件的常量池中，均包含这个 TAG 常量，但是 Dalvik VM 在编译成.dex 文件之后，其常量池里只有一个 TAG 常量。

JVM 和 Dalvik VM 还有一点非常不同，即基于的架构不同。JVM 是基于栈的架构，而 Dalvik VM 是基于寄存器的架构。相对于基于栈的 JVM 而言，基于寄存器的 Dalvik VM 实现虽然牺牲了一些硬件上的通用性，但是它在代码的执行效率上更胜一筹。一般来讲，VM 中指令的解释执行的时间主要花费在三个方面：分发指令、访问运算数、执行运算。其中，分发指令这个环节对性能的影响最大。在基于寄存器的 Dalvik VM 中，可以更有效地减少冗余指令的分发，减少内存的读写访问。

从 JVM 和 Dalvik VM 的区别上来说，Dalvik VM 主要针对 Android 嵌入式操作系统的特点进行了各种优化，使其更省电、更省内存、运行效率更高，但是牺牲了一些 JVM 的与平台无关的特性。实际上，Dalvik VM 本就是为 Android 设计的，无需考虑其他平台的问题。这里只介绍了 JVM 和 Dalvik VM 的主要区别，毕竟本书并不是讲解 Android 内核的，因此这里只是点明 Dalvik VM 的特点，读者对这部分的内容了解即可。

1.1.4 Android 版本

自 Android 系统首次发布至今，Android 经历了很多的版本更新。表 1-1 列出了 Android 系统的不同版本的发布时间及对应的版本号。

表 1-1　Android 系统版本

Android 版本	发 布 日 期	代　　号
Android 1.1	2009 年	Petit Four （花式小蛋糕）
Android 1.5	2009 年	Cupcake（纸杯蛋糕）
Android 1.6	2009 年	Donut（甜甜圈）
Android 2.0/2.1	2009 年	Eclair（长松饼）
Android 2.2	2010 年	Froyo（冻酸奶）
Android 2.3	2010 年	Gingerbread（姜饼）
Android 3.0/3.1/3.2	2011 年	Honeycomb（蜂巢）
Android 4.0	2011 年	Ice Cream Sandwich（冰淇淋三明治）
Android 4.1/4.2/4.3	2012 年	Jelly Bean（果冻豆）
Android 4.4	2013 年	KitKat（奇巧巧克力棒）
Android 5.0/5.1	2014 年	Lollipop（棒棒糖）
Android 6.0	2015 年	Marshmallow（棉花糖）

比较重要的版本及变化如下。

1．Android 2.3 Gingerbread（姜饼）

该版本增加了对 NFC（近距离通信）和大量其他传感器的支持，包括陀螺仪和气压计。Gingerbread 将成为世界上最多使用的移动操作系统。

2．Android 3.0 Honeycomb（蜂窝）

这是第一个真正的 Android 平板电脑发布系统，摩托罗拉 Xoom 成为第一个真正意义上的 Android 平板。Honeycomb 添加了对多核处理器的支持，是迄今为止唯一一个没有进入开放源代码的 Android 版本。

3．Android 4.0 Ice Cream Sanwich（冰淇淋三明治）

Android 4.0 Ice Cream Sandwich 对操作系统做了一次大的彻底的更改，从 Honeycomb 引入许多元素到智能手机中。这个版本的操作系统包含了大量新的特性，包括 Android Beam、全景式拍照以及通过脸孔进行手机解锁的能力。

4．Android 5.0 Lollipop

2014 年 6 月 25 日于谷歌的 I/O 2014 大会上发布了 Developer 版（Android L），在 2014 年 10 月 15 日正式发布且名称定为 Lollipop。该版本的特点如下。

（1）采用全新 Material Design 界面。

（2）支持 64 位处理器。

（3）全面由 Dalvik 转用 ART 编译，性能可提升四倍。

（4）改良的通知界面及新增的优先模式。

（5）预载省电及充电预测功能。

（6）新增自动内容加密功能。

（7）新增多人设备分享功能，可在其他设备登录自己的账号，并获取用户的联系人、日历等谷歌云数据。

（8）强化网络及传输连接性，包括 Wi-Fi、蓝牙及 NFC。

（9）强化多媒体功能，如支持 RAW 格式拍摄。

（10）强化“OK Google”功能。

（11）改善 Android TV 的支持。

（12）提供低视力的设置，以协助色弱人士。

（13）改善了 Google Now 功能。

5．Android 6.0 Marshmallow

这是 2015 年 5 月 28 日于谷歌 I/O 2015 大会上发布的代号为“Marshmallow（棉花糖）”的 Android 6.0 系统。它提供了很多新的特性，如锁屏下语音搜索、Doze 电量管理等。

Android 的各个版本之间大部分 API 是向下兼容的，对于一些少部分的 API，也提供了向下兼容包。

1.1.5 Android 开发环境搭建

目前主流的 Android 开发环境有两种：一种是 Android Studio，另一种是 Eclipse、Android SDK 和 ADT 插件的组合。经过几次更新之后，谷歌推出的 Android Studio 已经成为非常强大的集成开发环境，谷歌也宣布 2015 年底中止对 Eclipse 的官方支持，目前 Eclipse 支持 Android 的最高版本为 Android 6.0，因此，本书的 Android 继承开发环境采用了 Android Studio。

Android Studio 是一款全新的基于 IntelliJ IDEA 的 Android IDE，类似于 Eclipse ADT 插件，Android Studio 提供了集成的 Android 开发工具用于开发和调试。

下面介绍如何使用 Android Studio 来搭建 Android 开发环境。

首先下载和安装 JDK，到 Oracle 官网进行下载，下载地址为 http://www.oracle.com/technetwork/java/javase/downloads/index.html，建议下载 jdk1.7 以上版本，下载后双击安装即可。

其次，配置 JDK 的变量环境，右击“我的电脑”选项，选择“属性”选项，在打开的窗口中双击“高级系统设置”选项，弹出“系统属性”对话框，如图 1-4 所示，在“高级”选项卡中单击“环境变量”按钮。

JDK 需要配置三个系统变量环境，分别是 JAVA_HOME、path 和 classpath。下面是这三个变量的设置方法。

1．JAVA_HOME

JAVA_HOME 的变量值为 JDK 在计算机中的安装路径，如 C:\Program Files\Java\jdk1.8.0_20，创建好后可以利用%JAVA_HOME%作为 JDK 安装目录的统一引用路径。

图 1-4 “系统属性”对话框

2．path

可以直接编辑 path 属性，在原有值后追加%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin 即可。

3．classpath

classpath 变量值为.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar。

下载 Android Studio，网址为 http://developer.android.com/sdk/index.html。可以根据自己的实际需要，选择一个合适的版本下载。一般来说，推荐下载最新版的 android-studio-bundle-×××.×××××××-windows.exe，因为在这个版本中已经包含了 Android SDK，无需再单独下载，比较方便。Android Studio 下载完成后，双击并逐步安装即可，如图 1-5 所示。

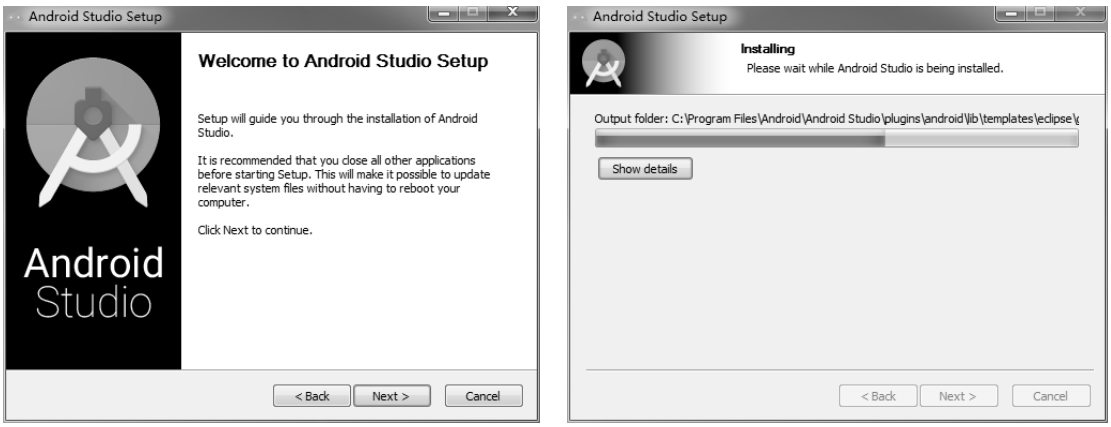

图 1-5　Android Studio 安装过程

运行 Android Studio，在每一次安装后，都会进入如图 1-6 所示的界面，这时需选择导入 Android Studio 的配置文件。如果以前使用过 Android Studio，可以选择使用以前的配置文件。如果是第一次使用，则可以选中第二个单选按钮。

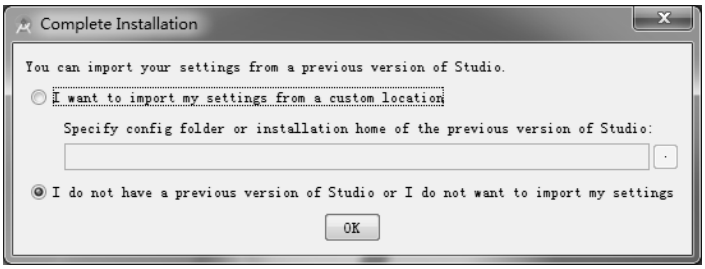

图 1-6　Android Studio 导入配置文件

进入如图 1-7 所示的界面，到此为止，Android Studio 已经安装好并可以使用了。

图 1-7　Android Studio 界面

因为 Android 系统存在多个版本，在实际使用过程中，还需要根据自己的需求下载相应的版本，可通过 SDK Manager 进行下载。在 Android Studio 工具栏中单击“SDK Manager”图标，如图 1-8 所示，打开 SDK Manager，然后选择需要的版本进行下载及安装即可。

图 1-8　Android Studio 工具栏中的“SDK Manager”图标

1.1.6　Android 模拟器及其使用

在成功安装 Android 的开发环境之后，还不能马上进行 Android 的开发。因为 Android 应用程序需要在 Android 的系统上运行，虽然现在 Android 设备越来越便宜，但是并不能要求所有学习者都去购买一部 Android 设备才能开始学习，因此 Android 提供了一个模拟器（Android

Virtual Device，AVD）来模拟一台 Android 手机，本小节将讲解如何创建一个 Android 模拟器。

AVD 可以通过 Android 模拟器管理器进行创建。

首先，在 Android Studio 工具栏中单击“AVD Manager”图标，如图 1-9 所示，打开 AVD Manager，AVD Manager 主界面如图 1-10 所示。

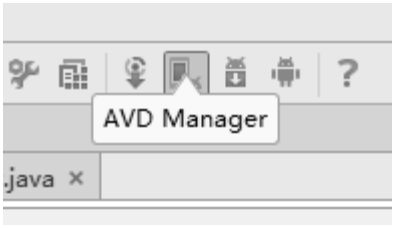

图 1-9　Android Studio 工具栏中的“AVD Manager”图标

图 1-10　AVD Manager 主界面

单击界面中的“Create Virtual Device”按钮，进入创建 Android 模拟器的界面，如图 1-11 所示。根据需要选择相应的设备，单击“Next”按钮，选择一个系统版本，进入模拟器配置界面，如图 1-12 所示，在此界面中输入所创建模拟器的名称，并对相关选项进行配置即可。

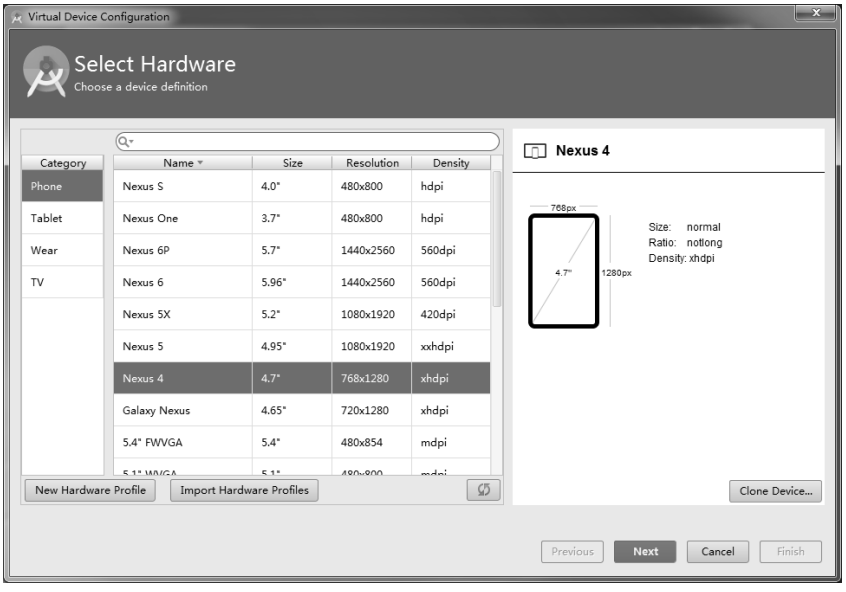

图 1-11 模拟器创建界面

图 1-12　模拟器配置界面

单击“Finish”按钮完成 AVD 的创建，可以在 AVD Manager 中看到新创建的模拟器，如图 1-13 所示，单击右侧的绿色箭头，即可以启动该模拟器。

图 1-13　新创建的模拟器

1.1.7 Android Market

谷歌官方的 Android Market 在国内访问起来速度比较慢，而且它所定位的是全球市场，里面很多的应用都是英文的，不符合国内用户的使用习惯，加上 Android 系统的开放性，使

得越来越多的第三方应用商店相继诞生，市场竞争越来越激烈。这里选取几个比较有代表性的应用商店，分别从应用介绍、下载量、用户评价、特色功能和应用推荐机制等角度进行简单的对比分析，以了解国内 Android 应用商店概况。

1．百度应用商店

百度应用商店实际上是一个聚合，它从众多的第三方商店（包括下面列出的一些商店）里挑选出应用。但因为它来自国内领先的搜索引擎——百度，所以在市场中占据重要位置。移动版百度首页在显著位置添加了百度应用的链接。百度应用同样有自己的 App。

2．腾讯应用宝

国内另一个互联网巨头腾讯也提供了 Android 应用平台。它由原先的“腾讯应用中心”更名为“应用宝”，并修改了 URL 和外观。腾讯的应用宝也有 App 的形式，而且为 Android 平板电脑提供了独立版本。

3．豌豆荚

讲到桌面同步应用，豌豆荚第一个指出了中国智能手机用户不太喜欢管理云内容，所以它创建了一个广义上的应用商店，包括一个移动应用商店和一个 PC 同步应用。实际上，豌豆荚应用商店是一个应用的聚合器，自身并不存储应用。

任务实战

在 Android Studio 中新建了 Android 工程 T1_1_HelloWorld，并在模拟器上运行，具体实现过程如下。

步骤 1　打开创建向导。

在 Android Studio 界面的菜单中依次选择“File -> New -> New Project”选项打开创建向导，如图 1-14 所示。

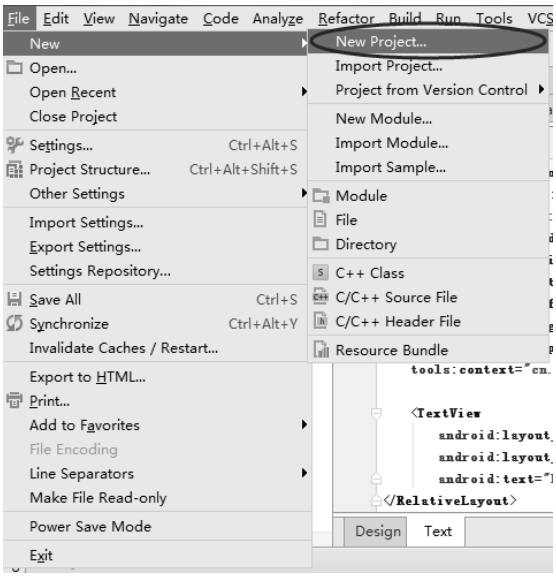

图 1-14　新建 Android Project

步骤 2　设置新建的向导。

在 Android 项目创建向导界面中，依次输入应用程序的名称、公司域名，并选择工程所在位置，然后单击“Next”按钮，如图 1-15 所示。

步骤 3　选择 App 将要运行的因素。

根据需求，选择最低 SDK 版本，然后单击“Next”按钮，如图 1-16 所示。再按提示选择相应的 Activity，并输入 Activity 名称、布局名称等内容，最后单击“Finish”按钮即可完成 T1_1_HelloWorld Android 项目的创建，并自动完成了 HelloWord 应用所需要的代码。

步骤 4　运行项目。

运行 T1_1_HelloWorld 项目，在 Android Studio 工具栏中单击图 1-17 中的运行图标，即可根据需求选择对应的模拟器或者真机运行该项目。

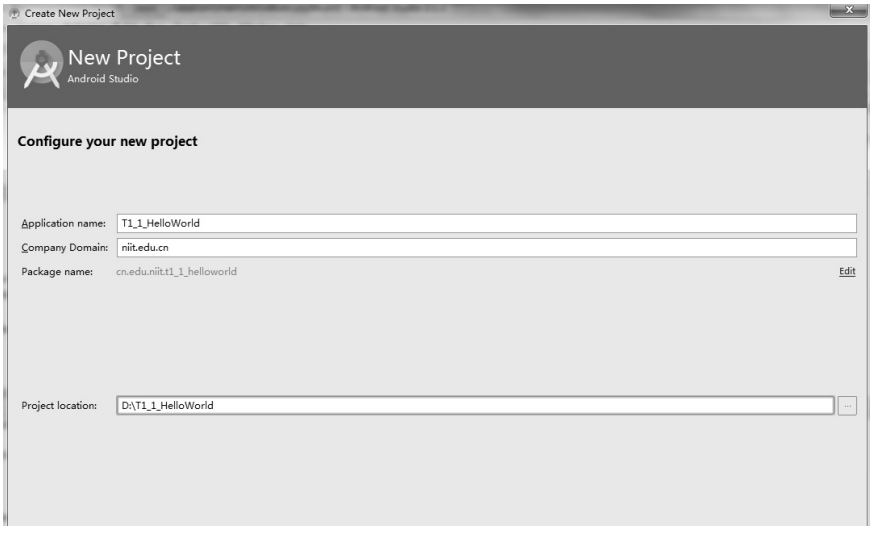

图 1-15　新建 Android Project 向导界面（一）

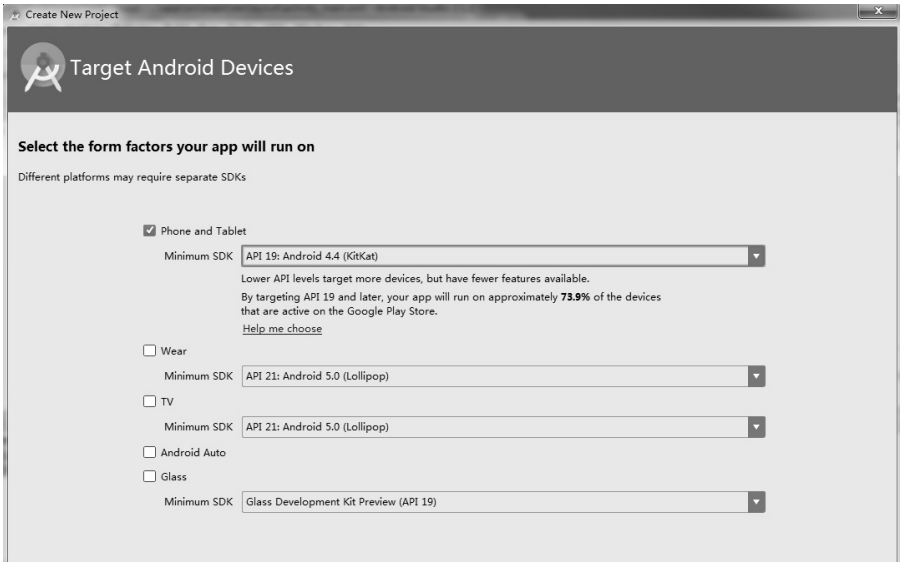

图 1-16　新建 Android Project 向导界面（二）

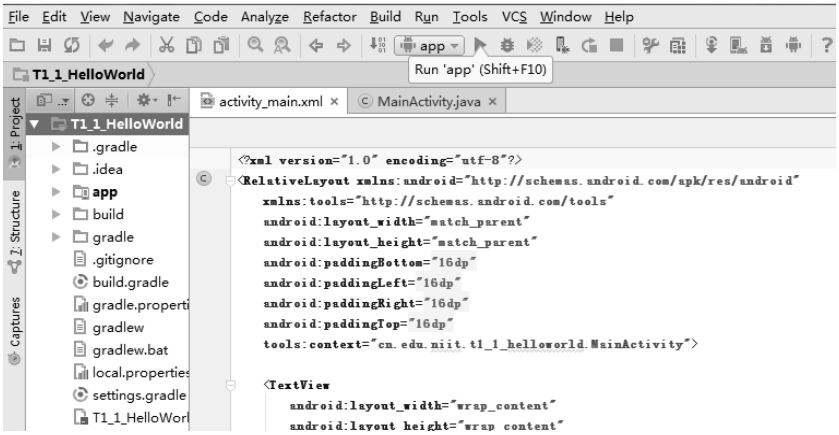

图 1-17　运行 Android Project

自此，就完成了一个简单的 HelloWorld Android 项目的创建。虽然全部依靠工具自动生成，一行代码也没有编写，但是它表示了一个完整 Android 应用从创建到运行的过程。之后将学习如何在这个过程中添加需要的代码逻辑。

步骤 5　Android 项目的打包与发布。

做完一个 Android 项目之后，如何才能把项目发布到 Internet 上供别人使用呢？我们需要将自己的程序打包成 Android 安装包文件——APK（Android Package），其扩展名为“.apk”。将 APK 文件直接上传到 Android 模拟器或 Android 手机中执行即可进行安装。Android 系统要求具有其开发者签名的私人密钥的应用程序才能够被安装。生成数字签名以及打包项目成 APK 都可以采用命令行的方式，但是通过 Android Studio 中的向导会更加方便地完成整个流程。下面以前面开发的“T1_1_HelloWorld”为例，演示如何生成 APK 文件。

当开发完一个项目后，在 Android Studio 菜单中依次选择“Build->Generate Signed APK…”选项，如图 1-18 所示。

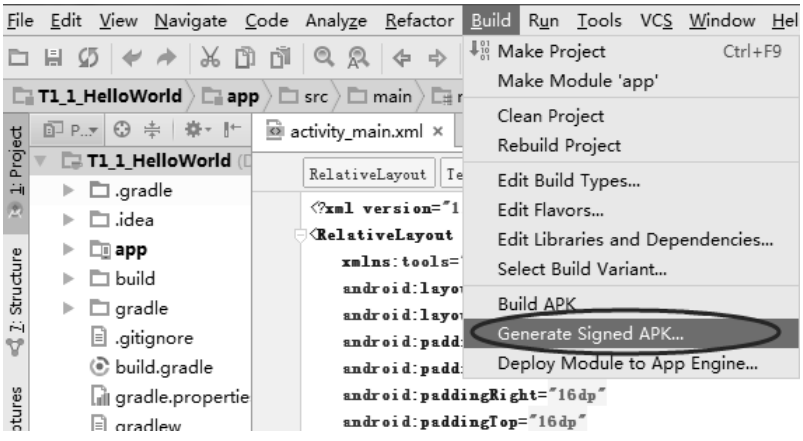

图 1-18　生成 APK 的操作流程

弹出的对话框如图 1-19 所示。单击“Create new…”按钮创建密钥库，如图 1-20 所示，如果已有密钥库则忽略这一步，可单击“Choose existing…”按钮选择相应的密钥库文件。

图 1-19 “Generate Signed APK”对话框

图 1-20 “New Key Store”对话框

最后，单击“Next”按钮，选择保存路径后，单击“Finish”按钮即可完成打包签名操作。

本任务主要介绍了 Android 系统的概况，具体讲解了 Android 系统的发展历史及其平台架构、Dalvik 虚拟机、Android 历史版本、Android 开发环境的搭建、Android 模拟器的使用方法等内容，并通过学生空间项目的第一个工程的创建与运行，熟悉了在 Android Studio 中创建 Android 工程的具体方法。

1．思考题

（1）简述各种手机操作系统的特点。

（2）Android 体系架构自上而下可分为哪些层？

（3）表述 Android 的体系架构及层次划分，并说明各个层次的主要内容。

（4）分析 Dalvik VM 和 JVM 的区别。

（5）搭建 Android Studio 开发环境需要哪些软件及步骤？

2．实操练习

（1）搭建 Android 开发环境。

（2）创建一个 HelloWorld Android 应用程序，并在模拟器上运行。

3．扩展阅读

（1）了解 Android 运行时，阅读维基百科对 ADT 的介绍，参考网址为 https://zh.wikipedia.org/wiki/Android_Runtime

（2）整理书中提到的专业词汇的英文，了解它们的含义，并记录下来。

任务 T1-2　认识 Android 应用的结构

任务目标

- 熟悉 Android 项目的目录结构；
- 学会 ADT 常用窗口的使用方法；
- 了解 DDMS、LogCat 的使用方法。

知识准备

1.2.1　Android 应用的目录结构

根据 Android 项目向导创建的 Android 应用程序，虽然没有编写代码，但是它也具有一个完整的 Android 项目的结构，Android Studio 提供的项目结构类型如图 1-21 所示。

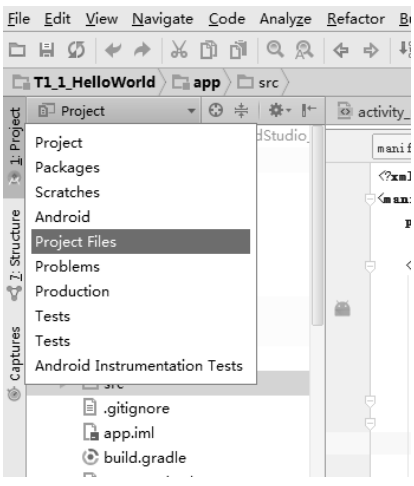

图 1-21　Android Studio 项目结构类型

在这几种项目结构类型中，Project 结构类型的所有视图都是真实的目录，project 和 module 结构显示清晰，而 Android 结构类型最大的优点就是隐藏了一些自动生成的文件和目录，并且把一些资源文件、源文件清晰地合并在一起，让开发者对于比较关心的文件一目了然，因此，常用的就是这两种结构。下面以 T1_1_HelloWorld 为例，分别介绍这两种结构类型。

1．Project 结构类型

Project 结构类型的目录如图 1-22 所示。

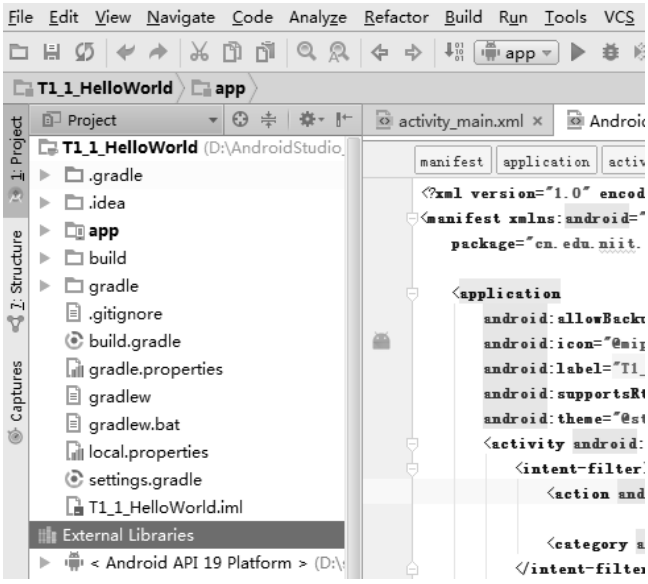

图 1-22　Project 结构类型的目录

从图 1-22 可以看出，在 Project 结构类型的目录中包含很多不同的文件与文件夹，下面对目录中的主要文件及文件夹进行说明。

（1）app/build/：编译输出的目录。

（2）app/build.gradle：app 模块的 gradle 编译文件。

（3）app/app.iml：app 模块的配置文件。

（4）app/proguard-rules.pro：app 模块的 proguard 文件。

（5）build.gradle：项目的 gradle 编译文件。

（6）gradlew：编译脚本，可以在命令行执行打包。

（7）local.properties：配置 SDK/NDK。

（8）settings.gradle：定义项目包含哪些模块。

（9）T1_1_HelloWorld.iml：项目的配置文件。

（10）External Libraries：项目依赖的 Lib，编译时自动下载。

2．Android 结构类型

Android 结构类型的目录如图 1-23 所示。

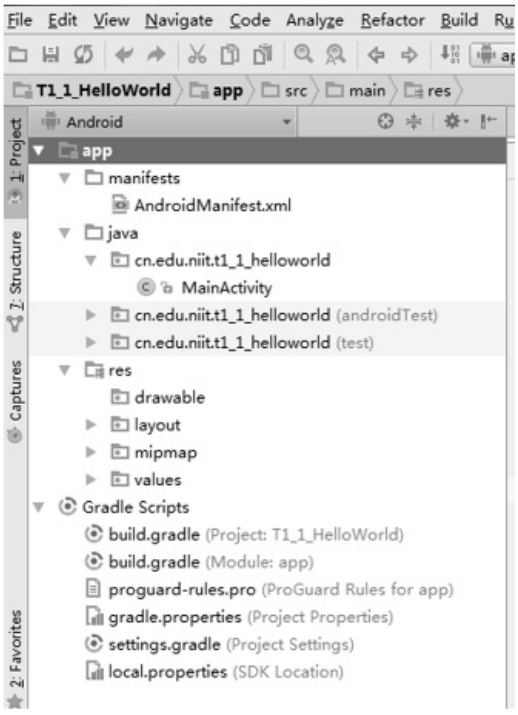

图 1-23　Android 结构类型的目录

从图 1-23 可以看出，在 Android 结构类型的目录中同样包含很多不同的文件与文件夹，下面对该目录中的主要文件及文件夹进行说明。

（1）app/manifests/AndroidManifest.xml：Android 项目的清单文件。

（2）app/java：项目的源代码及测试代码。

（3）app/res：项目的资源目录，存储所有的项目资源。

① app/res/drawable：存放一些自定义形状和按钮切换颜色之类的 XML。

② app/res/layout：存放布局文件。

③ app/res/mipmap：存放原生图片资源。

④ app/res/values：存放 App 引用的一些值，如 colors.xml、dimens.xml、strings.xml、styles.xml。

（4）Gradle Scripts：与 gradle 编译相关的脚本。

介绍完 Android 项目的目录结构之后，单独对其中的 R.java 文件以及 AndroidManifest.xml 清单文件进行具体说明，其余内容将会在本书的其他章节进行讲解。

3．R.java 文件

将 Android Studio 的项目结构切换成 Project 结构类型，依次打开 app->build->generated->source->r->debug，在 debug 的两个子文件中分别有一个 R 文件，这就是 R.java 文件。R.java 文件是编译器自动生成的，它无需开发人员对其进行维护。R.java 会自动收录当前应用中所有的资源，并根据这些资源建立对应的 ID，包括布局资源、控件资源、String 资源、Drawable 资源等。我们可以简单地把 R.java 理解成当前 Android 应用的资源字典。