第 3 章 chapter 3

第一个 Android 程序

本章主要介绍开发 Android 应用程序的基础知识和基本方法。通过本章内容的学习，读者可以掌握使用 Android Studio 开发 Android 应用程序的过程和方法，了解 Android 应用程序的目录结构和自动生成文件的作用。

本章学习目标：

- 掌握使用 Android Studio 开发 Android 应用程序的方法；
- 了解 AndroidManifest.xml 文件的用途；
- 了解 Android 的程序结构。

3.1 Android Studio 创建应用程序

本节介绍如何使用 Android Studio 集成开发环境建立第一个 Android 程序 HelloAndroid。首先启动 Android Studio，显示的集成开发环境如图 3.1 所示。

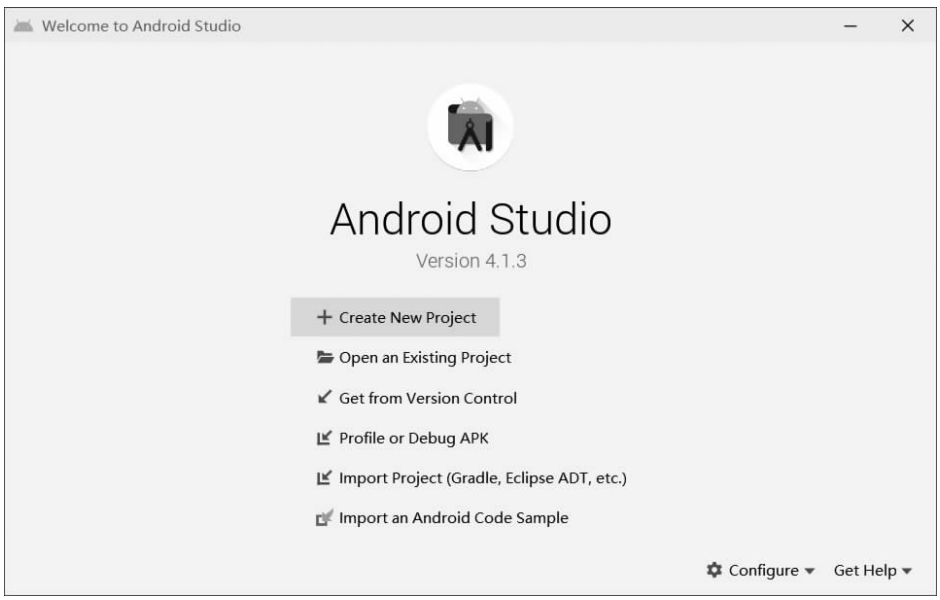

图 3.1　集成开发环境

运用如下方法可以创建一个新的 Android 工程。单击 Create New Project，跳转到选择模板页面。根据需要创建适用手机或平板电脑的 Android 工程，同时还提供创建适用于电视、可穿戴设备、车载应用、智能家居的 Android 工程。为了简化开发工作，选择 Empty Activity 建立一个空白的 Activity 即可，如图 3.2 所示。

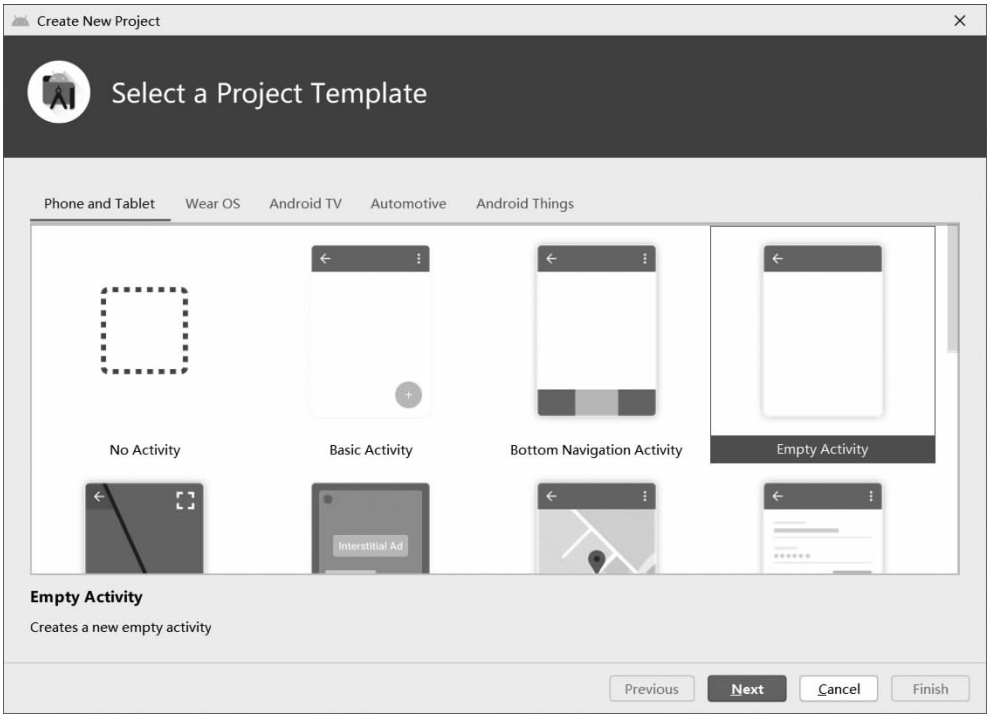

图 3.2　选择页面模板

进入工程配置页面后，需要在 Name 栏中输入应用程序名称，在 Package name 栏中输入包名称，在 Save location 栏中选择项目路径，在 Language 单选框中选择 Java。在 Minimum SDK 栏中选择能运行的最低版本的 SDK，推荐使用 API 16，它可以支持目前大部分的设备，如图 3.3 所示。

包名称(Package Name)是包的命名空间，遵循 Java 包的命名方法。包名称由两个或多个标识符组成，中间用点隔开，例如 edu.hrbeu.HelloAndroid。使用包主要为了避免命名冲突，因此可以使用反写电子邮箱地址的方式保证命名的唯一性。例如，笔者电子邮箱地址是 wangxianghui@hrbeu.edu.cn，则可以将包名称命名为 cn.edu.hrbeu.wangxianghui。为了保证代码的简洁，第一个 Android 程序的包名称使用 ice.hrbeu.helloandroid。

SDK 最低版本(Minimum SDK)是指 Android 程序能够运行的最低 API 等级，如果手机中的 Android 系统的 API 等级低于程序的 SDK 最低版本，则程序不能在该 Android 系统中运行。选择低版本的 API 可以提高程序的兼容性，但是为了兼容低版本 API，在工程中就无法使用新版本 API 中加入的新功能。为此，一般选择使用 API 16，可以在兼

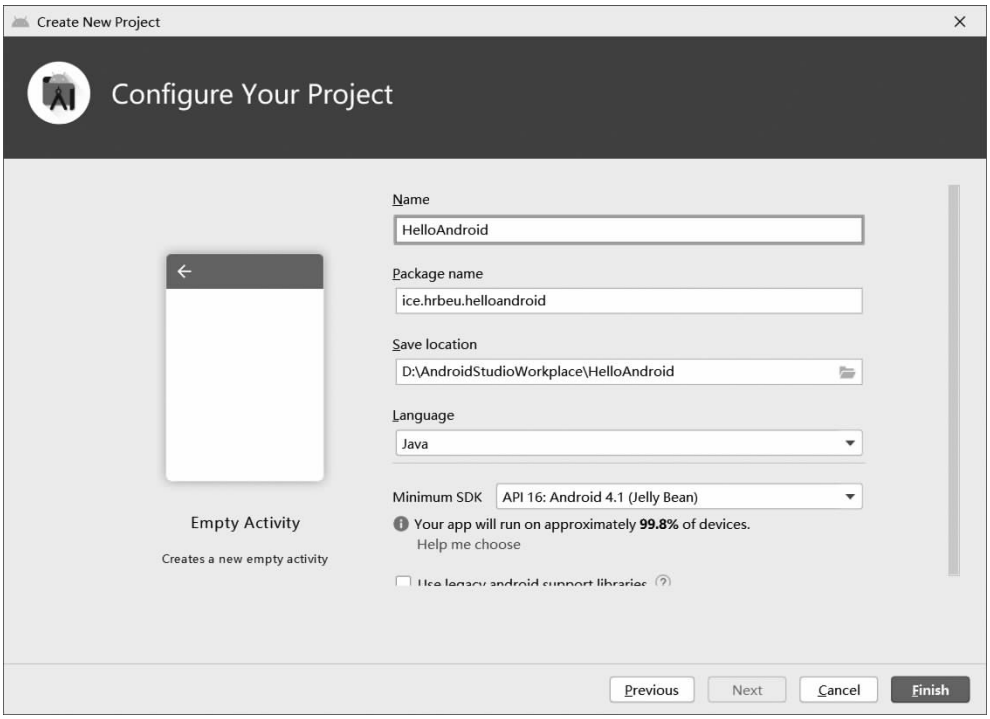

图 3.3 新建工程

容绝大部分手机的情况下，使用一些流行的新功能。

API 等级是 Android 系统中用来标识 API 框架版本的一个整数，用来识别 Android 程序的可运行性。如果 Android 程序标识的 API 等级高于 Android 系统所支持的 API 等级，程序则无法在该 Android 系统中运行。API 等级与系统版本之间的对照关系可参考表 3.1。

表 3.1 API 等级与系统版本对照表

系 统 版 本	API 等级	版 本 代 号	支持设备类型
Android 11.0	30	R	智能手机、平板电脑、智能家居
Android 10.0	29	Q	智能手机、平板电脑
Android 9.0	28	Pie(P)	智能手机、平板电脑
Android 8.1	27	Oreo(O)	智能手机、平板电脑
Android 8.0	26	Oreo(O)	智能手机、平板电脑

续表

系统版本	API等级	版本代号	支持设备类型
Android 7.1	25	Nouget(N)	智能手机、平板电脑
Android 7.0	24	Nouget(N)	智能手机、平板电脑
Android 6.0	23	Marshmallow(M)	智能手机、平板电脑
Android 5.1	22	Lollipop MR1(L)	智能手机、平板电脑
Android 5.0.1	21	Lollipop	智能手机、平板电脑
Android 4.4W	20	KitKat Wear	可穿戴设备
Android 4.4	19	KitKat	智能手机、平板电脑
Android 4.3	18	Jelly Bean	智能手机、平板电脑
Android 4.2	17	Jelly Bean	智能手机、平板电脑
Android 4.1	16	Jelly Bean	智能手机、平板电脑
Android 4.0.3-4.0.4	15	Ice Cream Sandwich	智能手机、平板电脑
Android 4.0	14	Ice Cream Sandwich	智能手机、平板电脑
Android 3.2	13	HONEYCOMB_MR2	平板电脑
Android 3.1.x	12	HONEYCOMB_MR1	平板电脑
Android 3.0.x	11	HONEYCOMB	平板电脑
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1	智能手机
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD	智能手机
Android 2.2.x	8	FROYO	智能手机
Android 2.1.x	7	ECLAIR_MR1	智能手机
Android 2.0.1	6	ECLAIR_0_1	智能手机
Android 2.0	5	ECLAIR	智能手机
Android 1.6	4	DONUT	智能手机
Android 1.5	3	CUPCAKE	智能手机
Android 1.1	2	BASE_1_1	智能手机
Android 1.0	1	BASE	智能手机

最后单击 Finish 按钮，工程向导会根据用户所填写的 Android 工程信息，自动在后台创建 Android 工程所需要的基础文件和目录结构。当创建过程结束，用户可以看到如图 3.4 所示的内容。

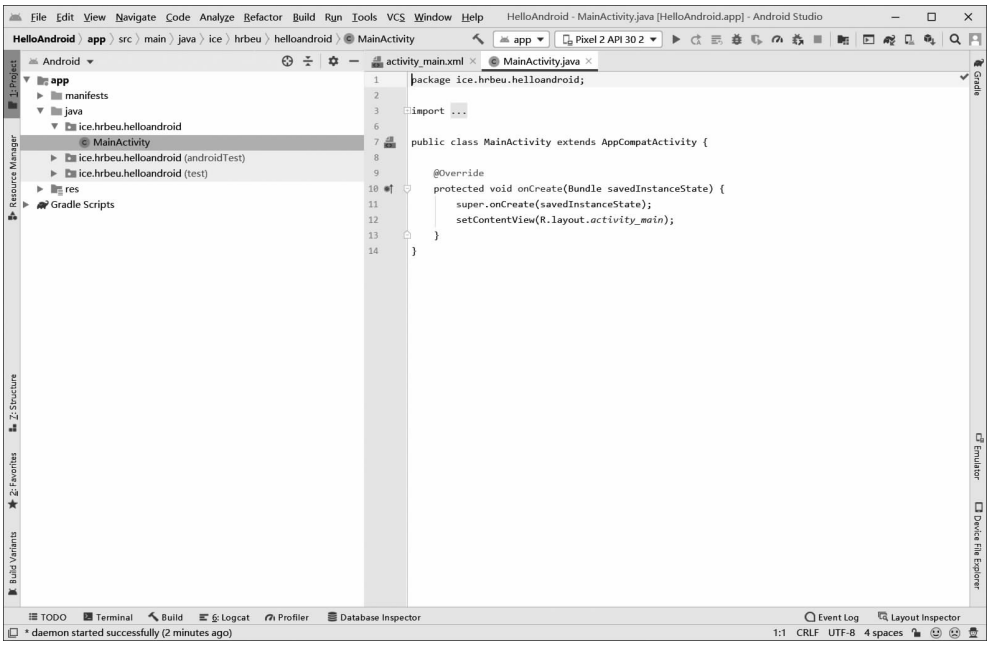

图 3.4 HelloAndroid 工程的文件和目录结构

用户无须在 HelloAndroid 工程中添加任何代码，即可运行 HelloAndroid 程序。但为了让 Android 程序能够正常运行，必须首先建立 Android 虚拟设备(Android Virtual Device，AVD)。

3.2 建立 Android 虚拟设备

AVD 是对 Android 模拟器进行自定义的配置清单，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、支持的附件 SDK 库和存储设置等信息。在用户配置好 AVD 后，Android Studio 就可以按照用户的要求启动特定版本和硬件特征的 Android 模拟器。配置 AVD 最简单的方式是选择 Android Studio 的 Tools→AVD Manager 命令启动 AVD 管理器，也可以单击工具栏中机器人与手机的图标打开 AVD 管理器。AVD 管理器如图 3.5 所示。

在 AVD 管理器中单击 Create Virtual Device 按钮，打开 AVD 创建界面，如图 3.6 所示。选择一个分辨率合适的设备，单击 Next 按钮。选择一个 Android SDK 版本，单击 Download 按钮下载该 SDK 版本，如图 3.7 所示。

在 AVD Name 输入框中输入 AVD 名称后，单击 Finish 按钮保存 AVD 的配置信息，如图 3.8 所示。然后在 AVD 管理器中单击启动按钮启动 Android 模拟器，如图 3.9

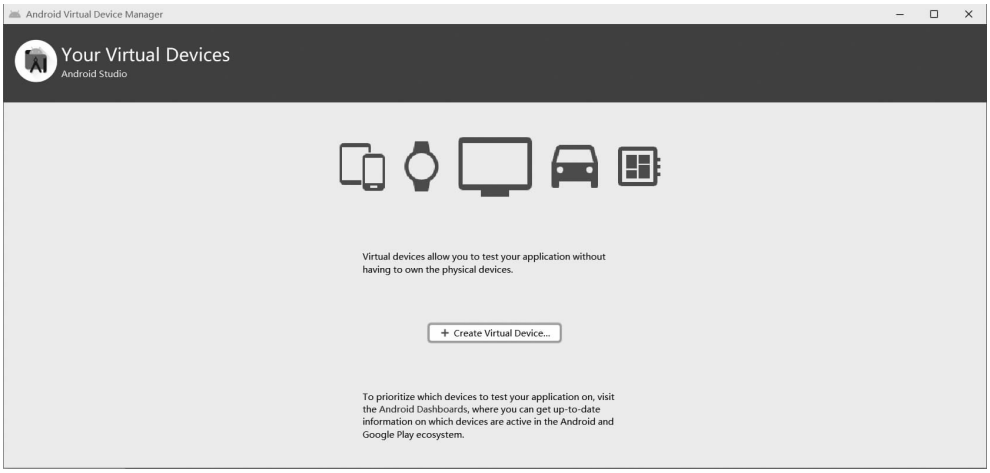

图 3.5　AVD 管理器

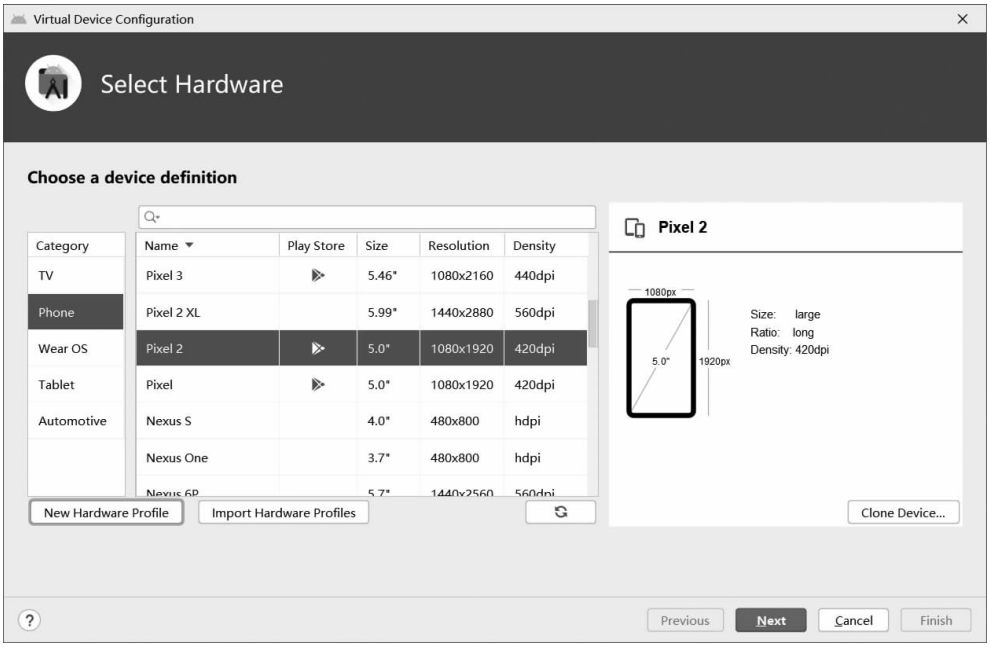

图 3.6　AVD 创建界面

所示。

启动 Android 模拟器是一个缓慢的过程，程序调试完毕后，不必关闭 Android 模拟器，可以节约下次程序调试时启动模拟器的时间。

在运行、调试或测试 Android 程序代码之前，需要创建运行/调试配置信息。首先选择工具栏中 Add Configuration 下拉列表①，然后单击 Run/Debug Configurations 对话框左上方的“＋”号按钮②，选择 Android App 模板创建运行配置，如图 3.10 所示。在 Name 输入框中输入配置模板名称，Module 下拉栏选择需要运行调试的项目程序，其他

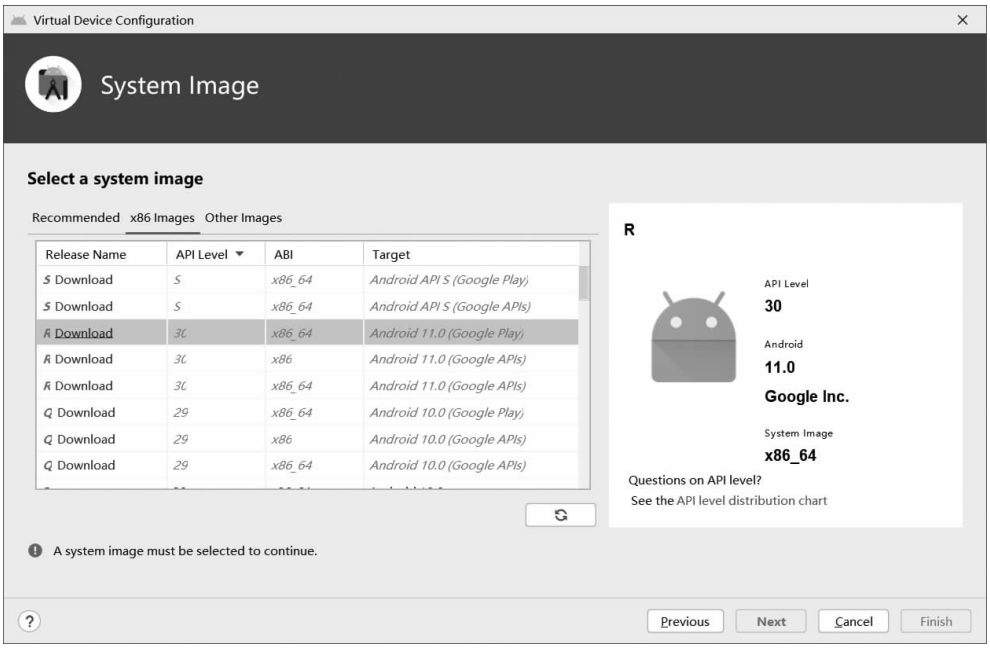

图 3.7 下载 Android SDK 版本

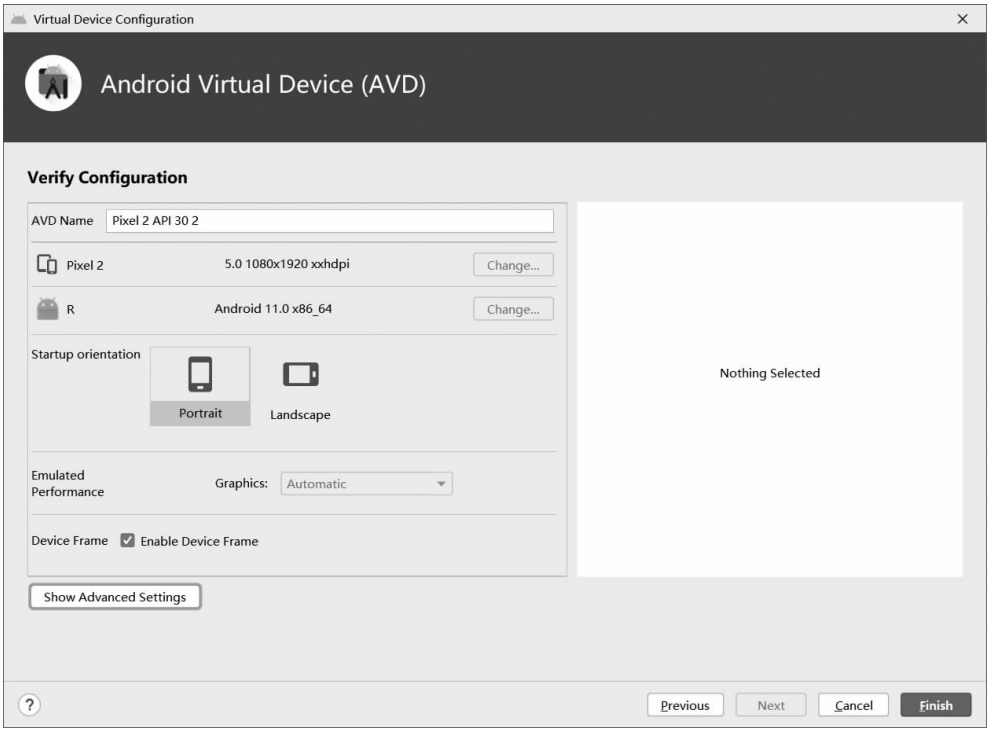

图 3.8 AVD 配置界面

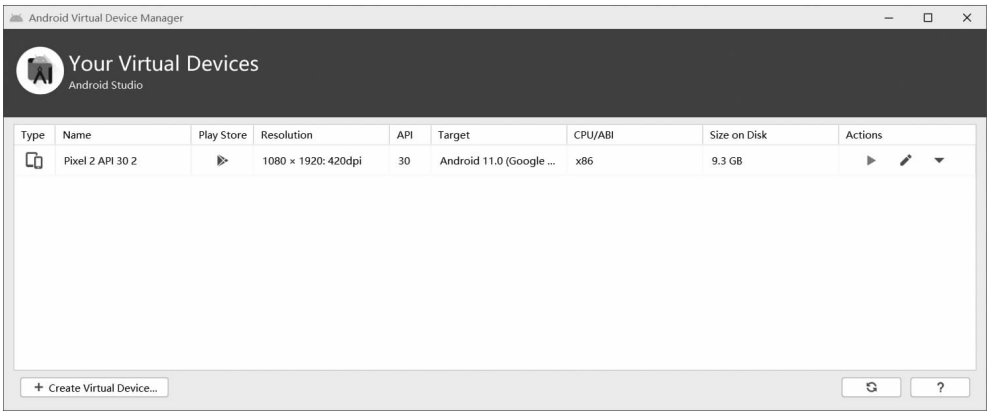

图 3.9　虚拟设备管理界面

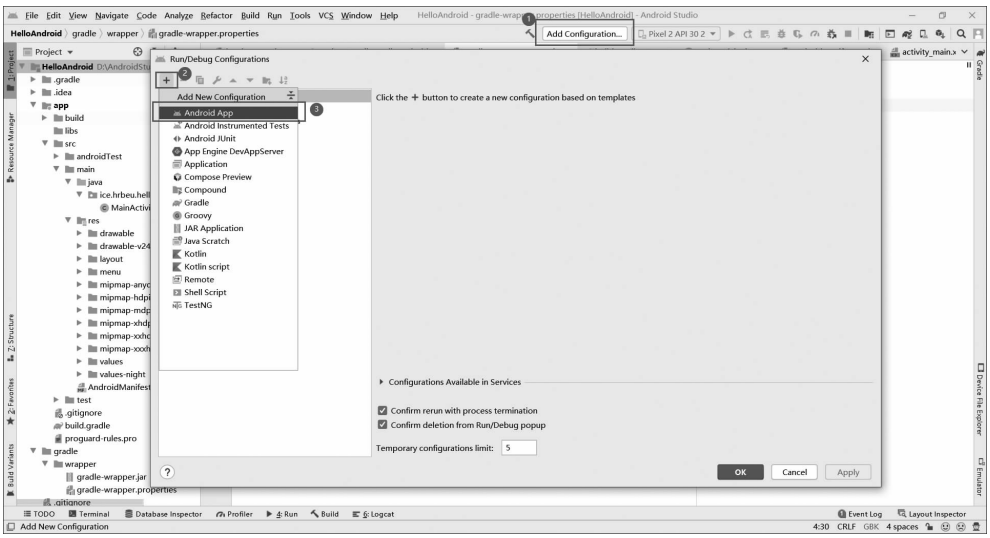

图 3.10　创建运行配置模板向导

选项保持默认设置，最后单击 OK 按钮保存配置，如图 3.11 所示。

使用 Android Studio 运行 Android 程序非常简单，只要选择 Run→Run app 命令就可以运行 Android 程序。Android Studio 会自动完成 Android 程序编译、打包和上传等过程，并将程序的运行结果显示在模拟器中。HelloAndroid 程序的运行结果如图 3.12 所示。

至此，已使用 Android Studio 创建了第一个 Android 程序，并得到程序的运行结果，对如何建立和运行 Android 程序已经有了基本的了解。后面的内容仍然以 HelloAndroid 为例，介绍 Android Studio 创建的 Android 的程序目录结构和文件用途。

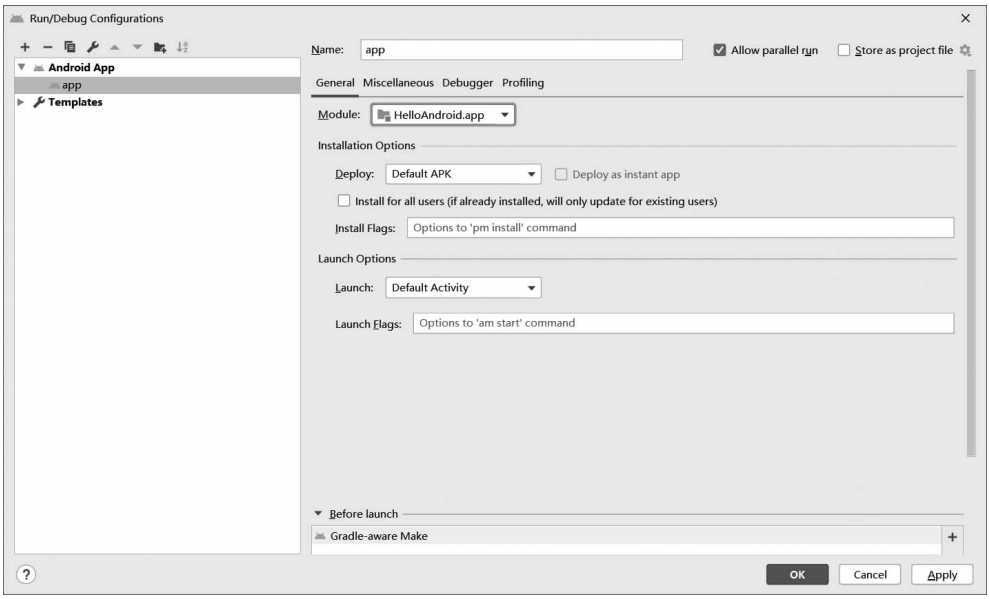

图 3.11 运行配置模板创建界面

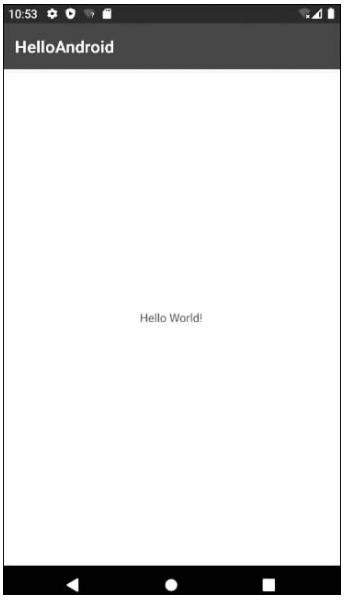

图 3.12 HelloAndroid 的运行结果

3.3 Android 程序结构

在 Android Studio 中，一个程序项目有 3 种视图，分为 Android 视图、Project 视图和 Packages 视图。

Android 视图是通过类型来组织项目的资产文件。例如，AndroidManifest 文件和 XML 文件在 manifests 文件中，所有的 Java 类都在 java 文件夹中，所有的资源文件都在 res 文件夹下，如图 3.13 所示。

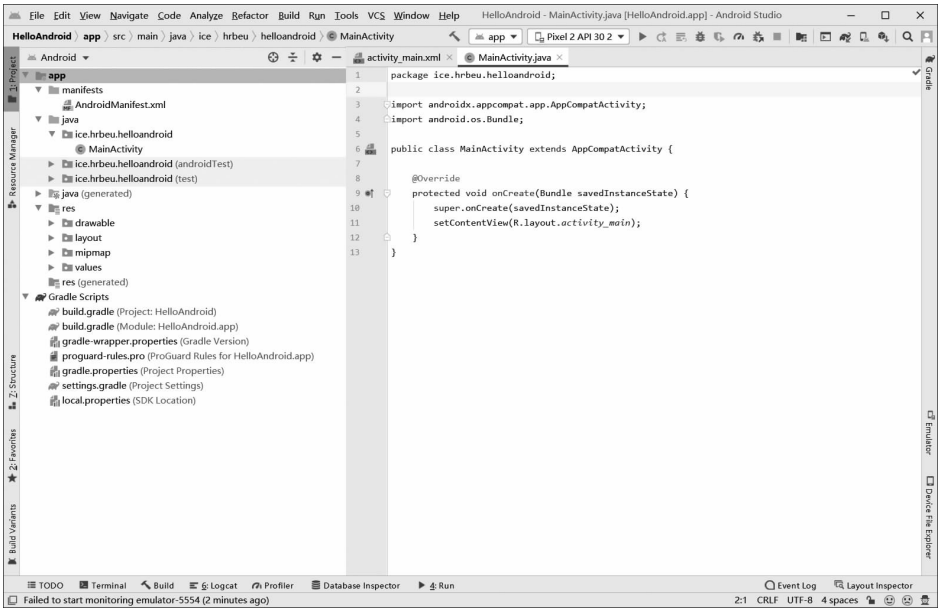

图 3.13 Android 视图

在默认的 Android 视图结构中，不能反映项目在磁盘上的实际物理组织。如果要查看项目的实际结构，就要切换到 Project 视图结构，如图 3.14 所示。

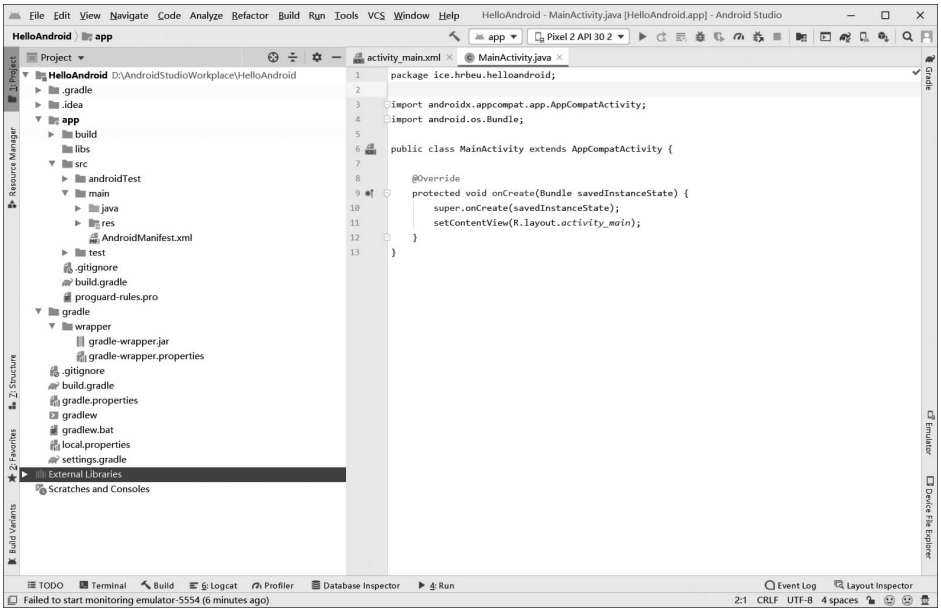

图 3.14 Project 视图

Package 视图与 Project 视图相比，最大的区别就是隐藏了相关的配置文件、属性文件和系统自身的目录，只显示当前的 Module 列表和 Module 下面的目录和文件，如图 3.15 所示。

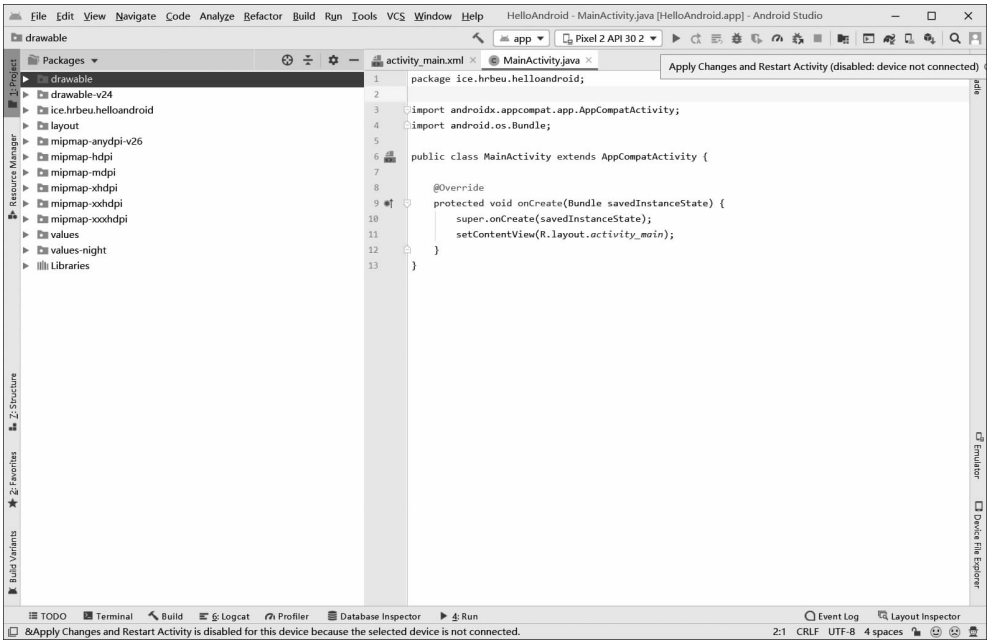

图 3.15 Package 视图

在建立 HelloAndroid 程序的过程中，Android Studio 会自动建立一些目录和文件，如图 3.16 所示。这些目录和文件有其固定的作用，有的允许修改，有的则不能进行修改，了解这些文件和目录对 Android 程序开发有着非常重要的作用。

在 Project 视图下，Android Studio 以工程名称 HelloAndroid 和 External Libraries 作为根目录，将所有自动生成的和非自动生成的文件都保存在这两个根目录下。Hello Android 根目录下包含 4 个子目录.gradle、.idea、app 和 gradle，以及 7 个工程文件.gitignore、build.gradle、gradle.properties、gradlew、gradlew.bat、local.properties 和 settings.gradle。External Libraries 根目录存放项目所依赖的所有类库。

.gradle 目录和.idea 目录是用来存放 Android Studio 自动编译工具生成的文件和开发工具产生的文件。

app 目录是用来存放程序中的代码资源文件。其中，build 目录在编译时生成，主要包含编译时自动生成的内容，在 outputs 目录下存放打包好的 apk 文件，libs 目录存放第三方 jar 包，然后 jar 包会被自动添加到构建路径（如集成地图 sdk，把 jar 包放到 libs 目录，可以在 build.gradle 文件中查看当前项目依赖）。

src 目录是源代码目录。androidTest 目录是用来编写 android test 测试用例的，可以对项目进行自动化测试；main 目录下的 java 目录是存放 Java 代码的地方，res 是存放资源的目录。Android 程序所有的图像、颜色、风格、主题、界面布局和字符串等资源都保

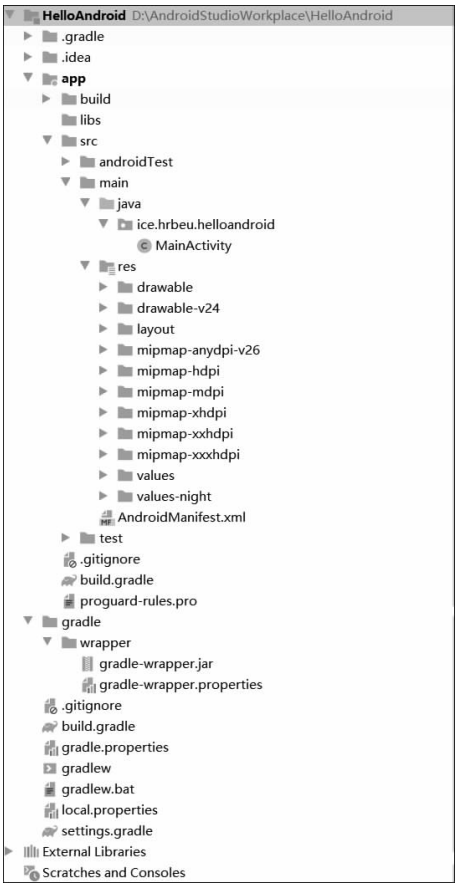

图 3.16　HelloAndroid 工程的目录和文件

存在 res 目录下的几个子目录中。其中，drawable 目录用来保存图像文件，layout 目录用来保存与用户界面相关的布局文件，mipmap-hdpi、mipmap-mdpi、mipmap-xhdpi、mipmap-xxhdpi 和 mipmap-xxxhdpi目录用来保存同一个程序中针对不同屏幕尺寸需要显示的不同大小的图标文件，引导页的图片也建议放在这里。values 目录保存颜色、风格、主题和字符串等资源；AndroidManifest.xml 是整个项目的配置文件，四大组件都需要在这里注册才能正常运行；test 目录用来编写 Unit Test 测试用例的目录，是对项目进行自动化测试的另一种方式；.gitignore 文件是为 git 源码管理的配置文件；build.gradle 文件是 Android 项目的 Gradle 构建脚本文件，用于配置 Android 构建过程所需要的参数和引用依赖；proguard-rules.pro 用来指定项目代码的混淆规则，帮助代码打包成混淆过的安装包文件。

gradle 目录下包含 gradle-wrapper 的配置文件，使用 gradle-wrapper 的方式不需要提前将 gradle 下载好，只需要根据本地的缓存情况决定是否需要联网下载 gradle。Android Studio 默认没有启动 gradle-wrapper 的方式，如果需要打开，可以在 Android Studio 导航栏选择 File→Settings→Build，Execution，Deployment→Gradle 命令进行配

置更改。

.gitignore(外层)文件用于将指定的目录或文件排除在版本控制之外,作用和内层的.gitignore 文件类似。

build.gradle(外层)文件是项目全局编译环境配置。

gradle.properties 是全局的 gradle 配置文件。这里配置的属性将会影响到项目中所有的 gradle 编译脚本。

gradlew 和 gradlew.bat 用来在命令行界面执行 gradle 命令,其中 gradlew 是在 Linux 或 Mac 系统中使用,gradlew.bat 是在 Windows 系统中使用。

local.properties 配置文件用来指定本机中 Android SDK 的路径,一般是自动生成,除非 SDK 位置发生变化,否则无须修改该文件的路径。

settings.gradle 用于指定项目中所有引入的模块。由于项目中只有一个 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,需要手动修改这个文件的场景比较少。

AndroidManifest.xml 是 XML 格式的 Android 程序声明文件,包含了 Android 系统运行 Android 程序前所必须掌握的重要信息。这些信息包括应用程序名称、图标、包名称、模块组成、授权和 SDK 最低版本等,而且每个 Android 程序必须在根目录下包含一个 AndroidManifest.xml 文件。XML 是一种可扩展标记语言,本身独立于任何编程语言,能够对复杂的数据进行编码,且易于理解。Android 工程中多处使用了 XML 文件,使应用程序开发更加具有弹性,且易于后期的维护和理解。

AndroidManifest.xml 文件的代码如下。

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="ice.hrbeu.helloandroid"
4      android:versionCode="1"
5      android:versionName="1.0">
6      <uses-sdk
7          android:minSdkVersion="16"
8          android:maxSdkVersion="30"/>
9      <application
10         android:allowBackup="true"
11         android:icon="@mipmap/ic_launcher"
12         android:label="@string/app_name"
13         android:roundIcon="@mipmap/ic_launcher_round"
14         android:supportsRtl="true"
15         android:theme="@style/Theme.HelloAndroid">
16         <activity
17             android:name=".MainActivity"
18             android:label="@string/app_name">
19             <intent-filter>
20                 <action android:name="android.intent.action.MAIN" />
```

```
21                <category android:name="android.intent.category.LAUNCHER" />
22            </intent-filter>
23        </activity>
24    </application>
25 </manifest>
```

在 AndroidManifest.xml 文件中，根元素是 manifest，其包含 xmlns:android、package、android:versionCode 和 android:versionName 4 个属性。其中，第 2 行属性 xmlns:android 定义 Android 的命名空间，值为 http://schemas.android.com/apk/res/android；第 3 行属性 package 定义应用程序的包名称；第 4 行属性 android:versionCode 定义应用程序的版本号，是一个整数值，数值越大说明版本越新，但仅在程序内部使用，并不提供给应用程序使用者；第 5 行属性 android:versionName 定义应用程序的版本名称，是一个字符串，仅限于为用户提供一个版本标识。

manifest 元素仅能包含一个 application 元素，application 元素中能够声明 Android 程序中最重要的 4 个组成部分，包括 Activity、Service、BroadcastReceiver 和 ContentProvider，所定义的属性将影响所有组成部分。第 11 行属性 android:icon 定义 Android 应用程序的图标，其中@mipmap/ic_launcher 是一种资源引用方式，表示资源类型是图像，资源名称为 ic_launcher，对应的资源文件为 ic_launcher.png，目录是 res/mipmap-hdpi、res/mipmap-mdpi、res/mipmap-xhdpi、res/mipmap-xxhdpi 和 res/mipmap-xxxhdpi，这 5 个目录中的资源仍通过@mipmap 进行调用；第 12 行属性 android:label 则定义了 Android 应用程序的标签名称。第 14 行属性 android: supportSRtl 用作地区适配，RTL 即从右向左布局，为了支持阿拉伯语和波斯语，即阅读习惯从右向左的语言，在 application 标签里添加 android: supportSRtl="true"，从而改变布局。

activity 元素是对 Activity 子类的声明，不在 AndroidManifest.xml 文件中声明的 Activity 将不能够在用户界面中显示。第 17 行属性 android:name 定义了实现 Activity 类的名称，可以是完整的类名称，如 ice.hrbeu.helloandroid.MainActivity，也可以是简化后的类名称，如.MainActivity；第 18 行属性 android:label 则定义了 Activity 的标签名称，标签名称将在用户界面的 Activity 上部显示，@string/app_name 同样属于资源引用，表示资源类型是字符串，资源名称为 app_name，资源保存在 res/values 目录下的 strings.xml 文件中。

intent-filter 中声明两个子元素 action 和 category，在这里不详细讨论两个字元素的用途，但可以肯定的是，intent-filter 使 HelloAndroid 程序在启动时，将.MainActivity 这个 Activity 作为默认启动模块。

activity_main.xml 文件是界面布局文件，可利用 XML 描述用户界面，界面布局的相关内容将在第 5 章中进行详细介绍。activity_main.xml 代码的第 7 行说明在界面中使用 TextView 控件，TextView 控件主要用来显示字符串文本。代码第 10 行说明 TextView 控件需要显示的字符串，非常明显，@string/hello_world 是对资源的引用。通过 strings.xml 文件的第 5 行代码分析，在 TextView 控件中显示的字符串应是“Hello World, HelloAndroidActivity!”。如果读者修改 strings.xml 文件的第 5 行代码的内容，重新编

译、运行后，模拟器中显示的结果也应随之更改。

activity_main.xml 文件的代码如下。

```
<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"
    tools:context="ice.hrbeu.helloandroid.MainActivity" >

    <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

strings.xml 文件的代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">HelloAndroid</string>
    <string name="hello_world">Hello world! </string>
    <string name="action_settings">Settings</string>

</resources>
```

MainActivity.java 是 Android 工程向导根据 Activity 名称创建的 java 文件，这个文件完全可以手工修改。为了在 Android 系统上显示图形界面，需要使用代码继承 Activity 类，并在 onCreate()函数中声明需要显示的内容。

MainActivity.java 文件的代码如下。

```
package ice.hrbeu.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
13          setContentView(R.layout.activity_main);
14      }
15
16      @Override
17      public boolean onCreateOptionsMenu(Menu menu) {
18          // Inflate the menu; this adds items to the action bar if it is present.
19          getMenuInflater().inflate(R.menu.main, menu);
20          return true;
21      }
22
23      @Override
24      public boolean onOptionsItemSelected(MenuItem item) {
25          // Handle action bar item clicks here. The action bar will
26          // automatically handle clicks on the Home/Up button, so long
27          // as you specify a parent activity in AndroidManifest.xml.
28          int id =item.getItemId();
29          if (id ==R.id.action_settings) {
30              return true;
31          }
32          return super.onOptionsItemSelected(item);
33      }
34  }
```

第3行和第4行代码通过android.jar从Android SDK中引入了Activity和Bundle两个重要的包，用于子类继承和信息传递；第8行代码声明MainActivity类继承Activity类；第10行代码表明需要重写onCreate()函数；第11行代码的onCreate()会在Activity首次启动时被调用，为了便于理解，可以认为onCreate()是HelloAndroid程序的主入口函数；第12行代码调用父类的onCreate()函数，并将savedInstanceState传递给父类，savedInstanceState是Activity的状态信息；第13行代码声明了需要显示的用户界面，此界面是用XML描述的界面布局，保存在scr/layout/activity_main.xml资源文件中。

到这里分析了Android程序的目录结构和文件的用途，对AndroidManifest.xml文件、Java代码文件、资源引用等内容有了初步的了解。

习　　题

1. 简述AndroidManefiest.xml文件的用途。

2. 简述res目录下的各种资源类型。

3. 使用Android Studio建立名为MyAndroidStudio的工程，包名称为edu.hrbeu.MyAndroidStudio，程序运行时显示Hello MyAndroidStudio。