Android Development Tools

Table of Contents

[Elements of Android Projects 2](#_Toc100898134)

[Android Project Components 2](#_Toc100898135)

[Manifests 2](#_Toc100898136)

[Java Source Code 3](#_Toc100898137)

[Resources 4](#_Toc100898138)

[Gradle 5](#_Toc100898139)

[Android Emulators 7](#_Toc100898140)

[Dalvik VM 7](#_Toc100898141)

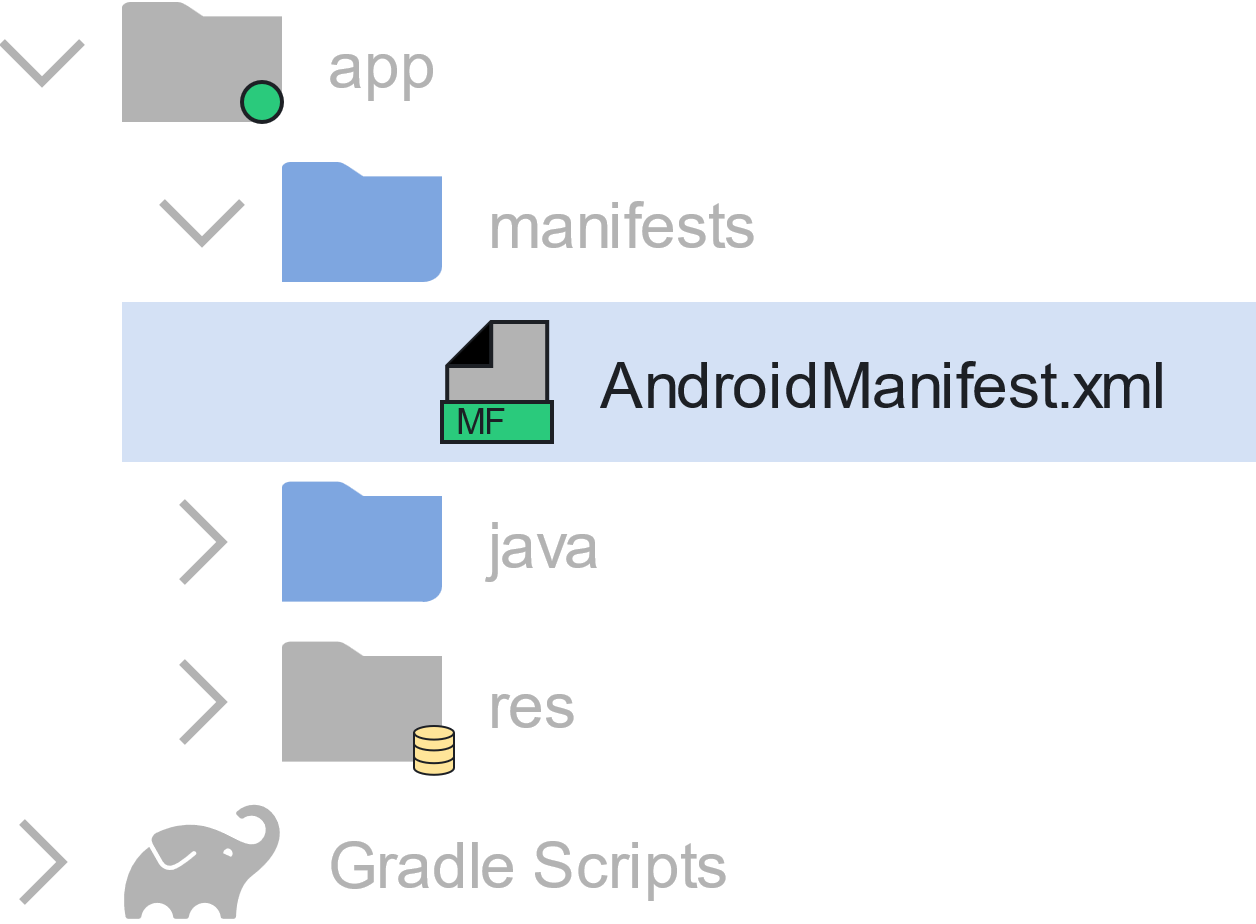
[Producing an Android Application 7](#_Toc100898142)

## Elements of Android Projects

* Application Name
* Project Name
* Package Name
* Minimum SDK Version
* Compile SDK Version
* Target SDK Version
* Theme

## Android Project Components

### Manifests



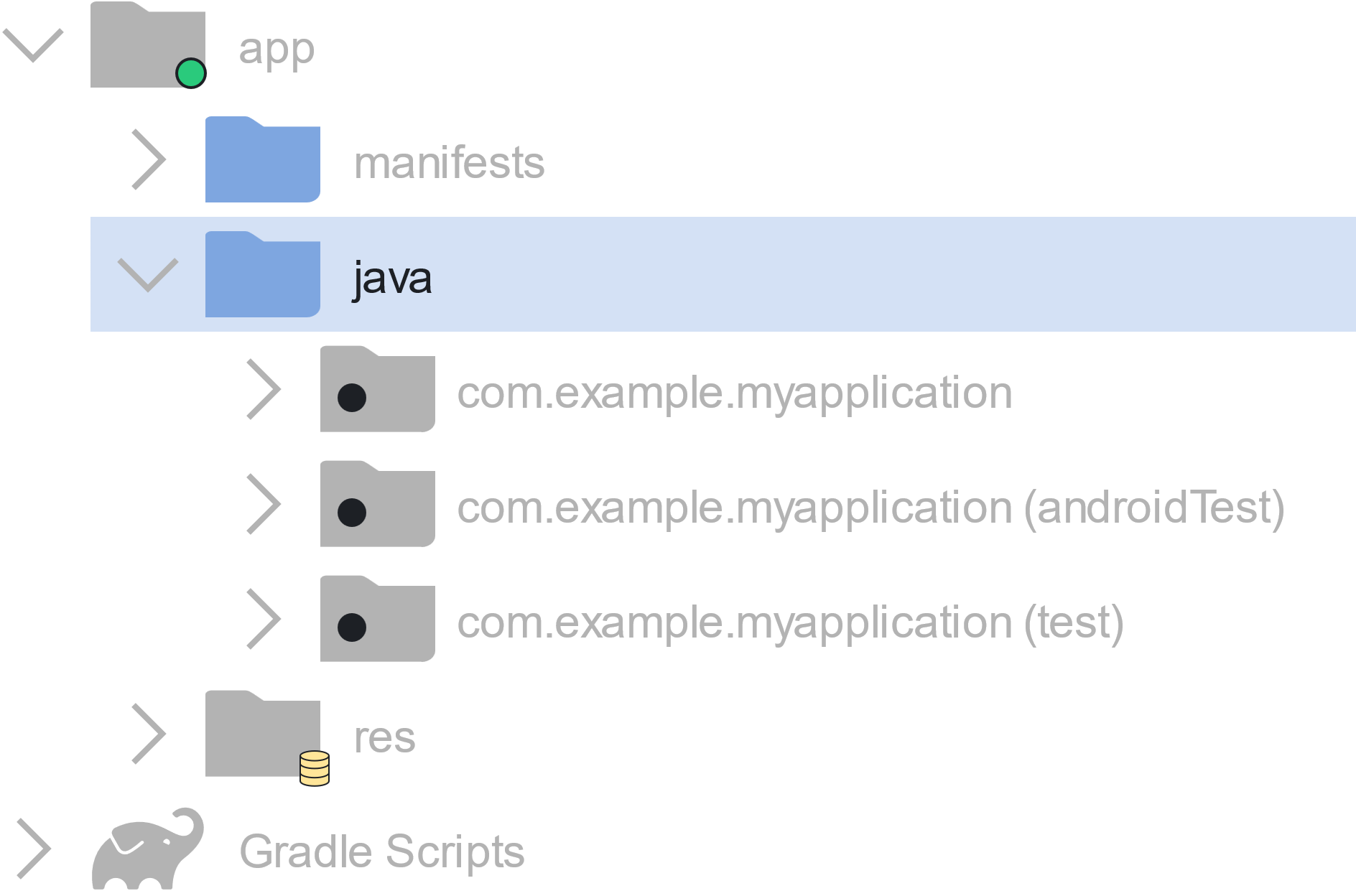
The AndroidManifest.xml file performs several functions:

* It contains the **Main Activity** on which the application starts.
* It declares explicit and implicit **Intent Filters**. We will learn what intent filters are later on.
* It declares all the different parts of the application like the package name, the different activities, services being used, broadcast receivers, icons, themes, etc.
* It declares **permissions** that will be requested by the application, such as network permissions and location permissions.

<application  
 android:allowBackup="true"  
 android:icon="@drawable/ic\_launcher"  
 android:label="@string/app\_name"  
 android:theme="@style/AppTheme">  
 <activity  
 android:name=".MainActivity"  
 android:label="My Application">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
</application>

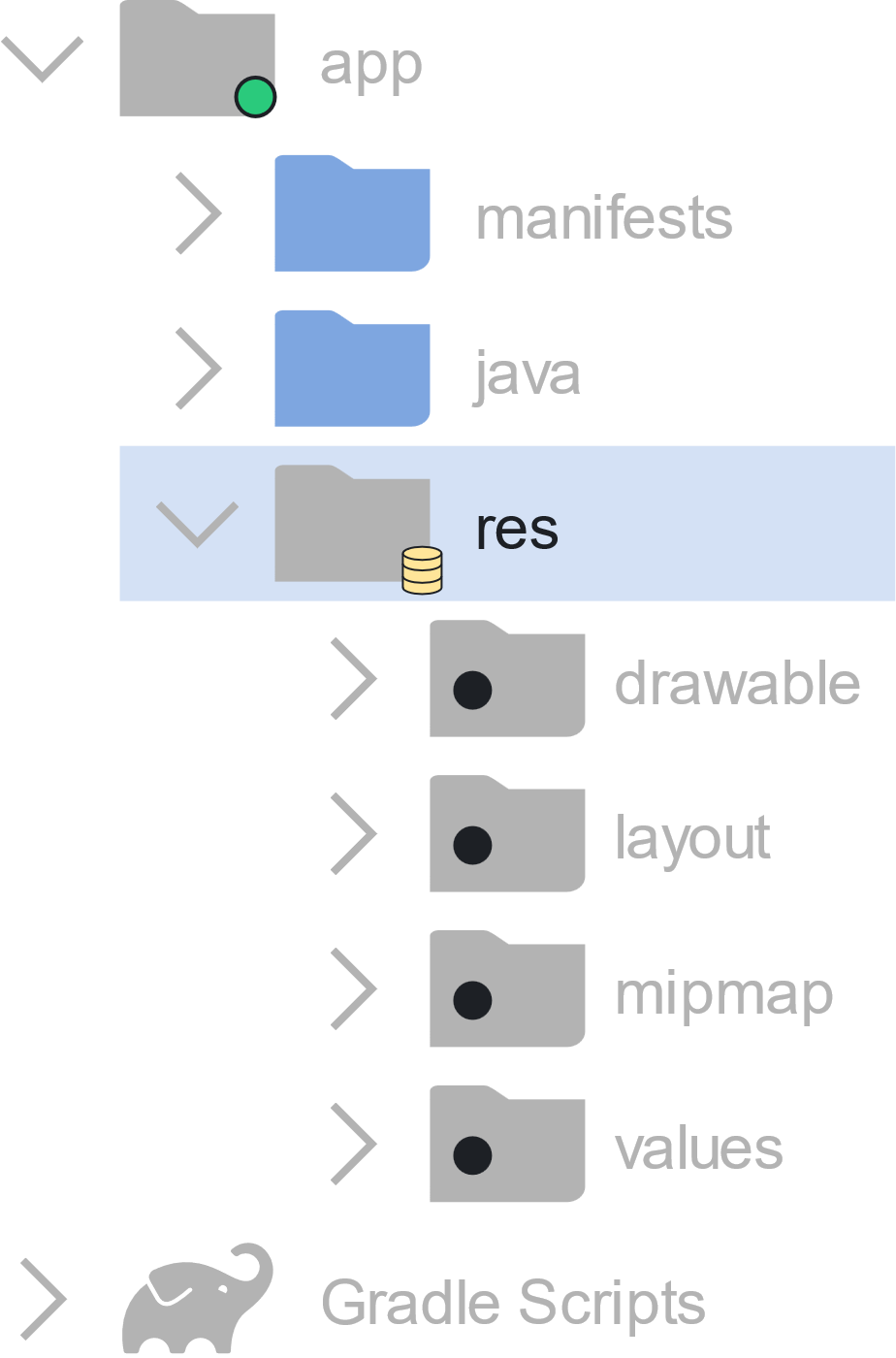
XML

### Java Source Code



This contains all of our **source code** in .java files.

### Resources

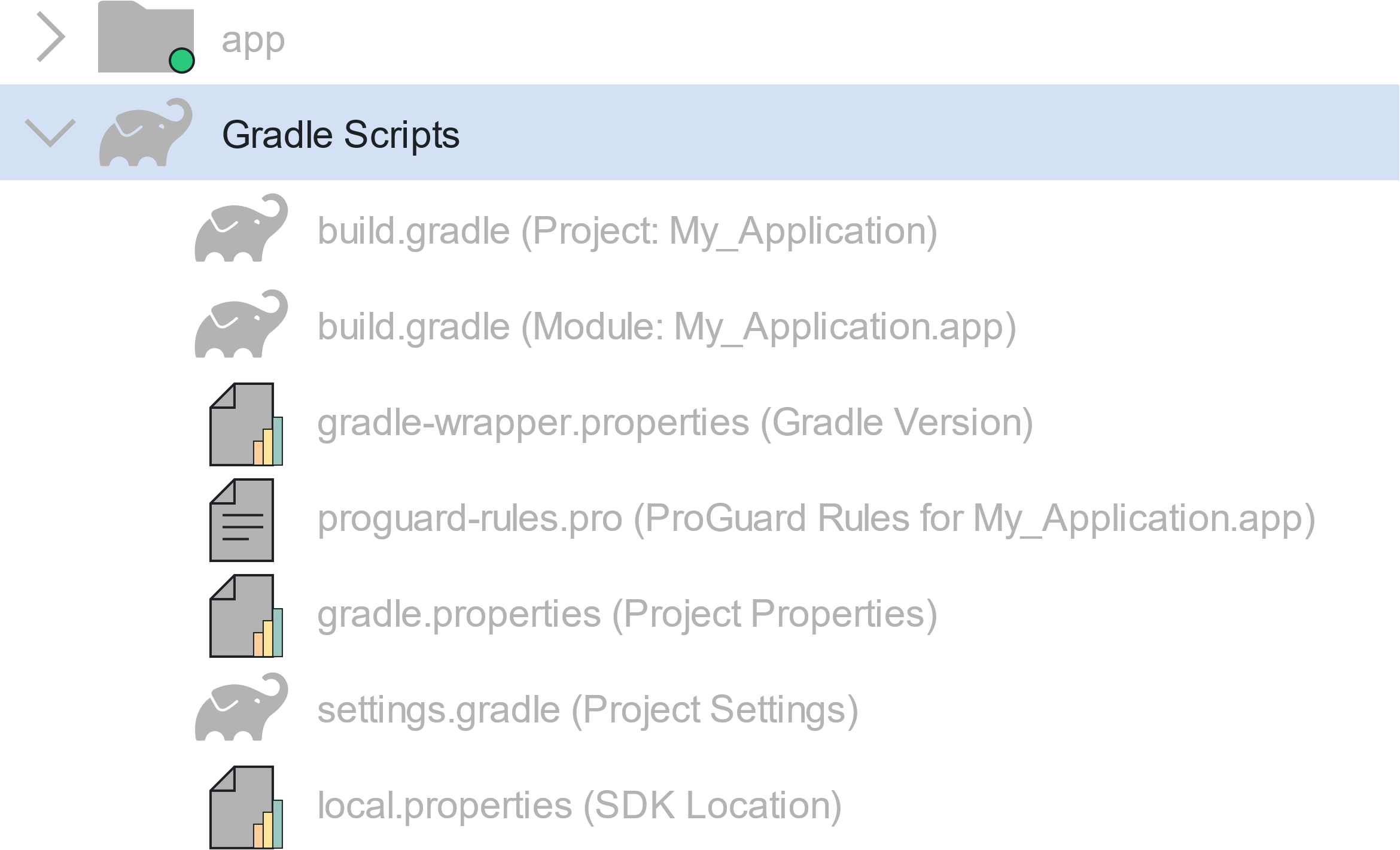


**Resources** contain additional files like images, layout definitions, user interface settings, animation instructions, etc.

There are several resource directories that perform different functions:

* res/drawable – Images
* res/layout – .xml files with user-interface layout definitions
* res/menu – XML based menu specifications; deprecated
* res/values – Lists of constants like strings, dimensions, colours, etc.
* res/raw – Raw files like audio or images; deprecated
* res/xml – Other general-purpose .xml files; deprecated
* res/mipmap – Drawable files for different launcher icon densities

### Gradle



The **Gradle Scripts** are used to generate the application from the .java and .xml files. There are two types of Gradle scripts, a **top-level Gradle Script**, which contains top-level configurations and a **module-level Gradle Script**, which contains dependencies and SDK versions. Both are confusingly called build.gradle.

Top-Level Gradle Script:

// Top-level build file where you can add configuration options common to all

// sub-projects/modules.  
buildscript {  
 repositories {  
 jcenter()  
 }  
 dependencies {  
 classpath 'com.android.tools.build:gradle:1.0.0'  
 // Note: Do not place your application dependencies in the individual

// module build.gradle files  
 }  
}  
allprojects {  
 repositories {  
 jcenter()  
 }  
}

XML

Module-Level Gradle Script:

apply plugin: 'com.android.application'  
android {  
 compileSdkVersion 32  
 buildToolsVersion "19.1.0"  
  
 defaultConfig {  
 applicationId "com.example.myapplication"  
 minSdkVersion 23  
 targetSdkVersion 32  
 versionCode 1  
 versionName "1.0"  
 }  
 buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
 }  
 }  
}

XML

It is best not to touch the top-level Gradle script. The module level Gradle script however will need to be modified at different times for many different reasons.

## Android Emulators

The Android Studio application comes with emulators called **Android Virtual Devices** (AVDs). We can create an AVD for one of many different types of devices and using any API version using the **AVD Manager**. They work very similarly to real devices, even including things like RAM, SD cards, location, etc., but they are not a true substitute for a real device.

## Dalvik VM

The Java language uses a **virtual machine** (VM) to produce .class files from .java files. This is called the **Java Virtual Machine** (JVM). Android has a similar thing called the **Dalvik Virtual Machine** (DVM), which creates .dex files from .class files. These are better optimized for mobile devices, with things like better memory management and battery utilization. They also introduce new libraries, but do not support some Java libraries like AWT or Swing.

## Producing an Android Application

