阅读感受：这本书只有130页，价格49元，略贵，为什么说贵？因为内容读起来像看天书，没有结果，只有过程，看了内容不知道干什么用，很多都是在讲解Gradle知识点，就像汽车，给你讲汽车各种配件知识，最终你不会开车一样。

# 第1章：Gradle和Android Studio入门

## 1.1 AndroidStudio

保持版本更新，有4个不同的更新途径：

* 金丝雀版为最新版，但是可能包含Bug。
* 开发版每个月或多或少都需要更新。
* 测试版有完整的更新，但是仍然可能包含bug。
* android studio默认为稳定版，作为功能全面测试的版本，它应该没有bug。

## 1.2 理解Gradle基础

如果你想用Gradle构建你的Android项目，那么你需要创建一个叫build.gradle的脚本。build.gradle脚本的书写没有基于传统的XML，而是基于Groovy的领域专用语言（DSL）。Gradle团队认为，基于动态语言的DSL语言与Ant或者任何基于XML的构建系统相比，优势都十分明显。

### 1.2.1 项目和任务

在Gradle中，最重要的两个概念是项目和任务。每一次构建都包含至少一个项目，每个项目又包括一个或多个任务。每个build.gradle文件都代表着一个项目，任务定义在构建脚本里。当初始化构建过程时，Gradle会基于build文件组装项目和任务对象。一个任务对象包含一系列动作对象，这些动作对象之后会按顺序执行。一个单独的动作对象就是一个待执行的代码块，它和Java中的方法类似。

### 1.2.2 构建生命周期

执行一个Gradle构建的最简单的形式是，只执行任务中的动作，而这些任务又依赖于其他任务。为了简化构建过程，构建工具会新建一个动态的模型流，叫作Directed Acyclic Graph(DAG)。这意味着所有的任务都会被一个接一个地执行，循环是不可能的。一旦一个任务被执行，其就不会被再次执行。那些没有依赖的任务通常会被优先执行。在构建的配置阶段会生成依赖关系图。一个Gradle的构建通常有如下三个阶段。

* 初始化：项目实例会在该阶段被创建。如果一个项目有多个模块，并且每一个模块都有其对应的build.gradle文件，那么就会创建多个项目实例。
* 配置：在该阶段，构建脚本会被执行，并为每个项目实例创建和配置任务。
* 执行：在该阶段，Gradle将决定哪个任务会被执行。哪些任务被执行取决于开始该次构建的参数配置和该Gradle文件的当前目录。

### 1.2.3 构建配置文件

每一个基于Gradle构建的项目，都应该至少有一个build.gradle文件。Android的构建文件中，有一些元素是必需的

buildscript {

repositories {

jcenter()

}

dependencies {

classpath 'com. android.tools.build:gradle:1. 2. 3'

}

}

这就是实际构建配置的地方。在repositories代码块中，JCenter库被配置为整个构建过程的依赖仓库。JCenter是一个预配置的Maven仓库，不需要额外的配置。当然在Gradle中有多个仓库可供选择时，你可以很容易地添加自己的本地或远程仓库。

构建脚本代码块在Android构建工具上定义了一个依赖，就像Maven的artifact。这就是Android插件的来源，Android插件提供了构建和测试应用所需要的一切。每一个Android项目都应该申请该插件：

apply plugin: 'com.android.application'

插件用于扩展Gradle构建脚本的能力。在一个项目中应用一个插件，该项目就可以使用该插件预定义的一些属性和任务。

如果你正在构建一个依赖库，那么你需要声明'com.android.library'，而不是'com.android.application'。你不能在一个模块中同时使用它们，因为这会导致构建错误。一个模块可以是一个Android应用模块，或者是一个Android依赖模块，但不能二者都是。

当你使用Android插件时，不仅可以配置针对Android的特殊约定，还可以生成只应用于Android的任务。下面的Android代码片段是由插件来定义的，可以配置在每个项目中：

android {

compileSdkVersion 22

buildToolsVersion "22. 0. 1"

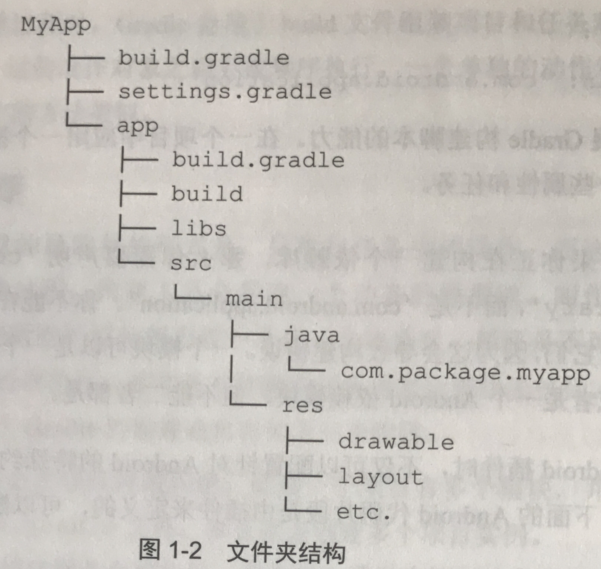
}

这是配置Android特殊约定参数的一部分。Android插件为Android的开发需求提供了一整套DSL。唯一需要的参数属性是编译目标和构建工具。编译目标由compileSdkVersion规定，compileSdkVersion是用来编译应用的SDK版本。使用最新的Android API版本作为编译目标。

在build.gradle文件中还有很多自定义特性。我们将在第2章中讲解最重要的特性，其他特性将在本书的其余部分介绍。

### 1.2.4 项目结构

和Eclipse项目相比，Android项目的文件夹结构的改变相当大。正如前面所提到的，Gradle的优点是约定优于配置，其同样适用于文件夹结构。

一个简单的应用的文件夹结构如图1-2所示。  
 

Gradle项目通常会在根目录下创建一个gradle脚本文件，这使得其在后续阶段新增模块变得更加简单。所有的应用源代码都在app文件夹下。该文件夹名称默认情况下也是模块的名称，但是其不必被命名为app。例如，当你使用Android Studio创建一个手机应用和一个Android智能可穿戴手表应用项目时，默认情况下模块会被叫作application和wearable.

## 1.3 创建新项目

触发Android Studio构建任务的方式有以下几种：

* 在Build菜单里，你可以单击Make Project，或者使用快捷键Ctrl+F9
* 单击工具栏上的Make Project按钮。
* Gradle工具窗列出了所有可用的Gradle任务。

## 1.4 Gradle Wrapper入门

Gradle是一个不断发展的工具，新版本可能会打破向后兼容性，而使用Gradle Wrapper可以避免这个问题，并能确保构建是可重复的。

Gradle Wrapper为微软的Windows操作系统提供了一个batch文件，为其他操作系统提供了一个shell脚本。当你运行这段脚本时，需要的Gradle版本会被自动下载(如果它还不存在)和使用。其原理是，每个需要构建应用的开发者或自构建系统可以仅仅运行Wrapper，然后由Wrapper搞定剩余部分。因此，我们不需在开发者机器或构建服务器上手动安装正确的Gradle版本，此外，Gradle还建议把Wrapper文件添加到你的版本控制系统中。

运行Gradle Wrapper和直接运行Gradle没什么不同，运行gradlew.bat来代替常规gradle命令即可。

## 1.5 获取Gradle Wrapper

项目根目录中有：

* gradlew：Linux和Mac OS X上shell脚本
* gradlew.bat：Windows的批处理文件
* gradle/wrapper/gradle-wrapper.jar：shell脚本和批处理文件需要用到的jar文件
* gradle/wrapper/gradle-wrapper.properties：包含了配置参数，并能决定使用哪一个Gradle版本，如果你想使用内部分发的自定义Gradle版本，那么可以修改distributionUrl属性，这也意味着，任何你使用的应用或者依赖库都可以有一个不同的Gradle URL，所以在运行wrapper之前，一定要检查它是否是可任何的属性。

当项目中使用的不是最新的Gradle版本时，AndroidStudio将会显示一个提醒，提示并建议你自动更新Gradle版本。其原理是，AS更改了gradle-wrapper.properties中的Url，然后执行了一次构建，这样最新版本的Gradle就会被下载。

## 1.6 运行基本构建任务

在Teminal中， 在项目根目录中执行：gradlew task ，显示所有可用任务，加--all参数，显示每个任务对应依赖的详细介绍。

执行gradlew assembleDebug 任务，它会为这个应用创建一个debug版本的apk。

为了避免大量的终端输入，Gradle提供了一种快捷输入，即驼峰式缩写任务名称，如gradlew assembleDebug任务可缩写为：

* gradlew assDeb
* gradlew aD

注意：当驼峰缩写是独一无二的时候才可以，如果另一个任务具有相同的缩写则不行。

其它基本任务：

Check：运行所有的检查，这通常意味着在一个连接的设备或模拟器上运行测试

Build：触发assemble和check

Clean：清除项目的输出

# 第2章 基本自定义构建

## 2.1 理解Gradle文件

使用AS创建一个新项目时，会默认生成3个Gradle文件：项目根目录中的build.gradle和settings.gradle和app中的build.gradle

### 2.1.1 settings文件

settings.gradle用于配置构建时需要包含的模块，如果是单模块可以不需要settings.gradle。

在这背后，Gradle会为每个settings文件创建一个Settings对象，并调用该对象的相关方法，你不必知道Settings类的相关细节，但是意识到这一点是非常好的。

### 2.1.2 顶层构建文件

在项目中，所有模块的配置参数都应在顶层build.gradle文件中配置。默认情况下其包含如下两个代码块：

buildscript {

repositories {

jcenter()

}

dependencies {

classpath 'com.android.tools.build:gradle:2.3.0'

}

}

allprojects {

repositories {

jcenter()

}

}

实际构建配置在buildscript代码块内，repositories代码块将JCenter配置成一个仓库，这个仓库就是一系列的依赖包，或者说，在我们应用和依赖项目中可使用一系列可下载的函数库。JCenter是一个很有名的Maven库。

dependencies代码块用于配置构建过程中的依赖包。这也意味着你不能将你的应用或依赖项目所需要的依赖包包含在顶层构建文件中。默认情况下，唯一被定义的依赖包是Gradle的Android插件。每个Android模块都需要有Android插件，因为该插件可使其执行Android相关的任务。

allprojects代码块可用来声明那些需要被用于所有模块的属性。你甚至可以在allproj ects中创建任务，这些任务最终会被运用到所有模块。

只要你使用了allprojects，模块就会被耦合到项目。这意味着，其很可能在没有主项目构建文件的情况下，无法独立构建模块。最初，这看起来可能不是一个问题，但是如果你后面想要分离一个内部依赖库到自己的项目，那么你将要重构你的构建文件。

### 2.1.3 模块的构建文件

模块层的build.gradle文件的属性只能应用在模块，它可以覆盖顶层的build.gradle文件任何属性。该文件示例如下：

apply plugin: 'com.android.application'

android {

compileSdkVersion 25

buildToolsVersion "25.0.2"

defaultConfig {

applicationId "com.example.dzl.helloworld"

minSdkVersion 15

targetSdkVersion 25

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-v7:25.3.1'

}

上面代码分为3个主要代码块：插件、Android、dependencies（依赖）

**1、插件**

第一行用到了Android应用插件，该插件在顶层构建文件中被配置成了依赖。谷歌的工具团队负责Android插件的编写和维护，并提供构建、测试和打包Android应用以及依赖项目的所有任务。

**2、Android**

该代码块包含了全部的Android特有配置，这些配置可以被使用是因为我们之前使用了Android插件。

complieSdkVersion，是用来编译应用的Android API版本

buildToolsVersion，是构建工具和编译器使用的版本号，构建工具包含命令行应用，如：aapt、zipalign、dx和renderscript，这些都被用来在打包应用时生成各种中间产物。

defaultConfig代码块用于配置应用的核心属性，这里面的属性可覆盖AndroidManifest.xml中的对应属性，如果defaultConfig中没有配置，则会使用AndroidManifest.xml中的配置。

applicationId：用于app的进程名称，应用的唯一标志。 它对应AndroidManifest.xml中的package属性，这个属性有两个功能，一是作为应用的唯一标识（进程名称），以及作为生成的R文件的包名，当有applicationId属性时，则package属性只作为生成R文件的包名功能。这样的好处是：可以使用构建variants任务来构建一个免费版和付费版，只需要把applicationId改一下即可生成两个应用，而且对程序没有任何影响。

targetSdkVersion用于通知系统，该应用已经在某特定Android版本通过测试，从而操作系统不必启用向前兼容的行为。这和complieSdkVerion没有任何关系。

buildTypes代码块用来定义如何构建和打包不同构建类型的应用。

**3、依赖包**

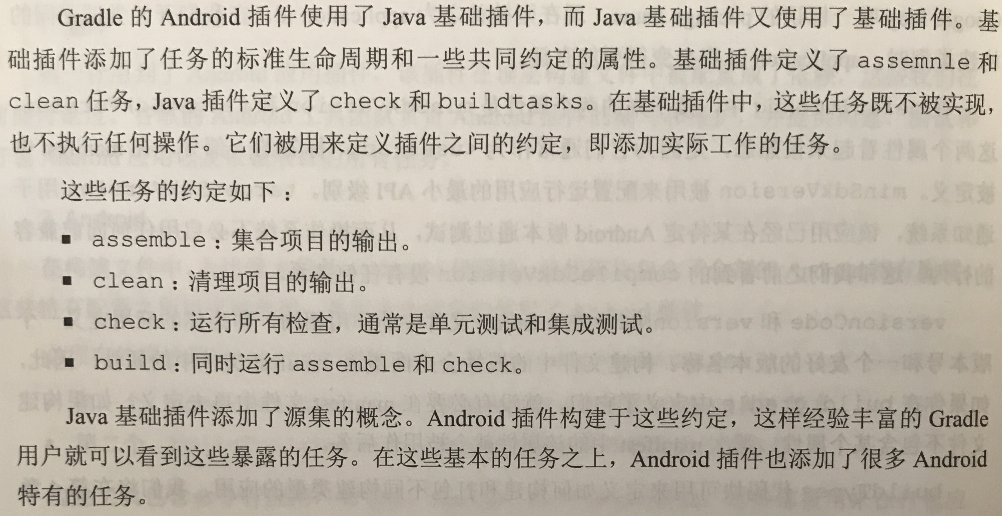
依赖代码块是标准Gradle配置的一部分，所以不必把这些配置放在android代码块内

## 2.2 任务入门

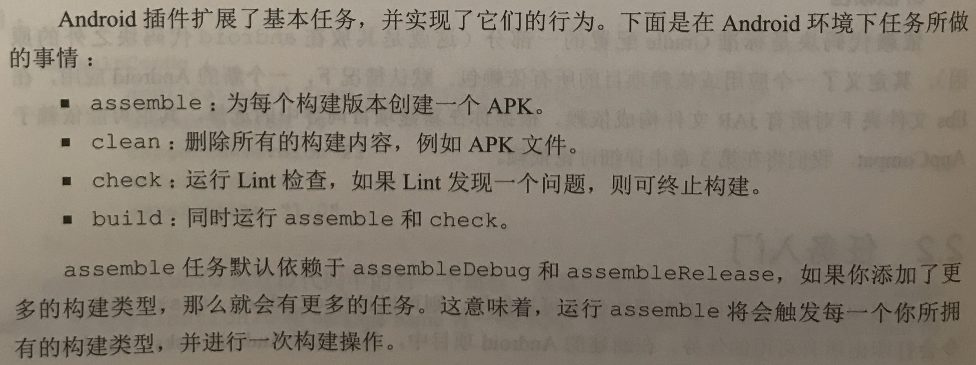
运行gradlew tasks任务可显示当前项目中有哪些可用任务，在新建的Android项目中，其会包括Android tasks、build tasks、build setup tasks、help tasks、install tasks、verification tasks和其它tasks。

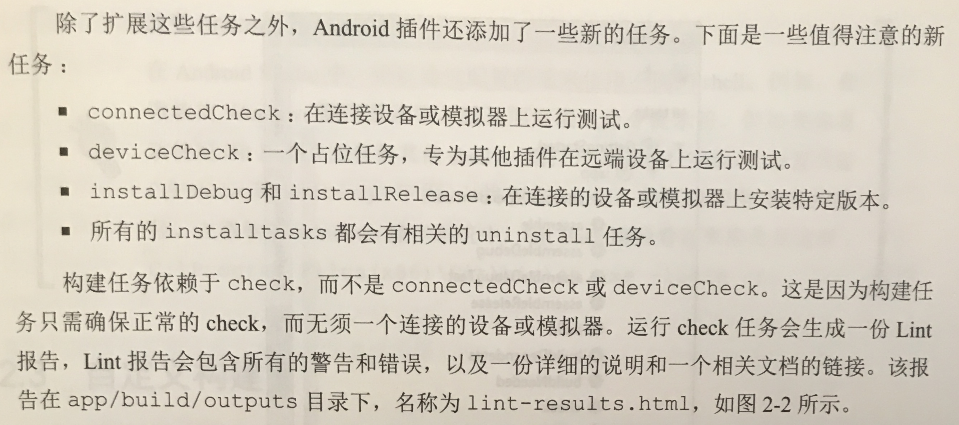
gradlew tasks --all，除了查看有哪些任何，还可看到它们之间的依赖。

### 2.2.1 基础任务



### 2.2.2 Android任务







### 2.2.3 AndroidStudio

你不必经常在命令行界面运行Gradle任务，可在Gradle工具窗口中执行，双击需要执行任务，在Gradle Console窗口中可查看正在运行的任务的进度。

## 2.3.2 BuildConfig 和资源

