

Android Developers

配置构建

Android 构建系统编译应用资源和源代码，然后将它们打包成可供您测试、部署、签署和分发的 APK。Android Studio 使用 Gradle (<http://www.gradle.org/>) 这一高级构建工具包来自动化执行和管理构建流程，同时也允许您定义灵活的自定义构建配置。每个构建配置均可自行定义一组代码和资源，同时对所有应用版本共有的部分加以重复利用。Android Plugin for Gradle 与这个构建工具包协作，共同提供专用于构建和测试 Android 应用的流程和可配置设置。

Gradle 和 Android 插件独立于 Android Studio 运行。这意味着，您可以在 Android Studio 内、使用计算机上的命令行工具或在未安装 Android Studio 的计算机（例如持续性集成服务器）上构建 Android 应用。如果您不使用 Android Studio，可以学习如何从命令行构建和运行您的应用 (<https://developer.android.google.cn/studio/build/building-commandline.html>)。无论您是从命令行、在远程计算机上还是使用 Android Studio 构建项目，构建的输出都相同。

注：由于 Gradle 和 Android 插件独立于 Android Studio 运行，您需要单独更新构建工具。请阅读版本说明，了解如何更新 Gradle 和 Android 插件 (<https://developer.android.google.cn/studio/releases/gradle-plugin.html#updating-plugin>)。

Android 构建系统非常灵活，让您能够在不修改应用核心源文件的情况下执行自定义构建配置。本章帮助您了解 Android 构建系统的工作原理，以及它如何帮助您对多个构建配置进行自定义和自动化处理。如果您只想了解有关部署应用的更多信息，请参阅在 Android Studio 中构建和运行项目 (<https://developer.android.google.cn/studio/run/index.html>)。要立即开始使用 Android Studio 创建自定义构建配置，请参阅配置构建变体 (<https://developer.android.google.cn/studio/build/build-variants.html>)。

构建流程

本文内容

[构建流程](#)[自定义构建配置](#)[构建配置文件](#)[Gradle 设置文件](#)[顶级构建文件](#)[模块级构建文件](#)[Gradle 属性文件](#)[将项目与 Gradle 文件同步](#)[源集](#)

另请参阅

[创建和编辑运行/调试配置](#)

视频

新的 Android SDK 构建系统

(<https://www.youtube.com/watch?>

This site uses cookies to store your preferences for site-specific language and display options.

OK

构建流程涉及许多将您的项目转换成 Android 应用软件包 (APK) 的工具和流程。构建流程非常灵活，因此了解它的一些底层工作原理会很有帮助。

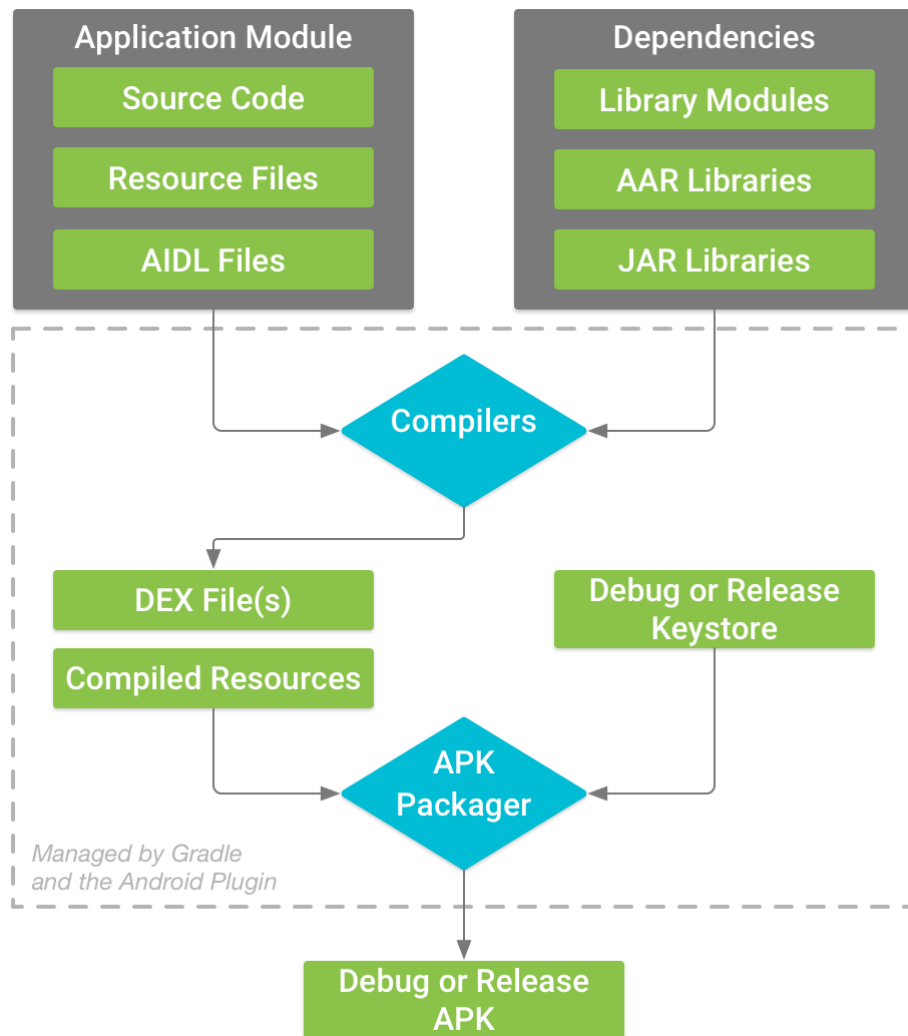


图 1. 典型 Android 应用模块的构建流程。

如图 1 所示，典型 Android 应用模块的构建流程通常依循下列步骤：

2. APK 打包器将 DEX 文件和已编译资源合并成单个 APK。不过，必须先签署 APK，才能将应用安装并部署到 Android 设备上。
3. APK 打包器使用调试或发布密钥库签署您的 APK：
 - a. 如果您构建的是调试版本的应用（即专用于测试和分析的应用），打包器会使用调试密钥库签署您的应用。Android Studio 自动使用调试密钥库配置新项目。
 - b. 如果您构建的是打算向外发布的发布版本应用，打包器会使用发布密钥库签署您的应用。要创建发布密钥库，请阅读在 Android Studio 中签署您的应用 (<https://developer.android.google.cn/studio/publish/app-signing.html#studio>)。
4. 在生成最终 APK 之前，打包器会使用 zipalign (<https://developer.android.google.cn/studio/command-line/zipalign.html>) 工具对应用进行优化，减少其在设备上运行时的内存占用。

构建流程结束时，您将获得可用来进行部署、测试的调试 APK，或者可用来发布给外部用户的发布 APK。

自定义构建配置

Gradle 和 Android 插件可帮助您完成以下方面的构建配置：

构建类型

构建类型定义 Gradle 在构建和打包您的应用时使用的某些属性，通常针对开发生命周期的不同阶段进行配置。例如，调试构建类型支持调试选项，使用调试密钥签署 APK；而发布构建类型则可压缩、混淆 APK 以及使用发布密钥签署 APK 进行分发。您必须至少定义一个构建类型才能构建应用 - Android Studio 默认情况下会创建调试和发布构建类型。要开始为应用自定义打包设置，请学习如何配置构建类型 (<https://developer.android.google.cn/studio/build/build-variants.html#build-types>)。

产品风味

产品风味代表您可以发布给用户的不同应用版本，例如免费和付费的应用版本。您可以将产品风味自定义为使用不同的代码和资源，同时对所有应用版本共有的部分加以共享和重复利用。产品风味是可选项，并且您必须手动创建。要开始创建不同的应用版本，请学习如何配置产品风味

This site uses cookies to store your preferences for site-specific language and display options.

OK

构建变体

构建变体是构建类型与产品风味的交叉产物，是 Gradle 在构建应用时使用的配置。您可以利用构建变体在开发时构建产品风味的调试版本，或者构建已签署的产品风味发布版本进行分发。您并不直接配置构建变体，而是配置组成变体的构建类型和产品风味。创建附加构建类型或产品风味也会创建附加构建变体。要了解如何创建和管理构建变体，请阅读配置构建变体 (<https://developer.android.google.cn/studio/build/build-variants.html>) 概览。

清单条目

您可以为构建变体配置中清单文件的一些属性指定值。这些构建值会替换清单文件中的现有值。如果您想为模块生成多个 APK，让每一个 APK 文件都具有不同的应用名称、最低 SDK 版本或目标 SDK 版本，便可运用这一技巧。存在多个清单时，Gradle 会合并清单设置 (<https://developer.android.google.cn/studio/build/manifest-merge.html>)。

依赖项

构建系统管理来自您的本地文件系统以及来自远程存储区的项目依赖项。这样一来，您就不必手动搜索、下载依赖项的二进制文件包以及将它们复制到项目目录内。要了解更多信息，请学习如何声明依赖项 (<https://developer.android.google.cn/studio/build/build-variants.html#dependencies>)。

签署

构建系统让您能够在构建配置中指定签署设置，并可在构建过程中自动签署您的 APK。构建系统通过使用已知凭据的默认密钥和证书签署调试版本，以避免在构建时提示密码。除非您为此构建显式定义签署配置，否则，构建系统不会签署发布版本。如果您没有发布密钥，可以按签署您的应用 (<https://developer.android.google.cn/studio/publish/app-signing.html>) 中所述生成一个。

ProGuard

构建系统让您能够为每个构建变体指定不同的 ProGuard (<https://developer.android.google.cn/studio/build/shrink-code.html>) 规则文件。构建系统可在构建过程中运行 ProGuard 对类进行压缩和混淆处理。

APK 拆分

构建系统让您能够自动构建不同的 APK，并且每个 APK 只包含特定屏幕密度或应用二进制界面 (ABI) 所需的代码和资源。如需了解详细信息，请

This site uses cookies to store your preferences for site-specific language and display options.

OK

构建配置文件

创建自定义构建配置需要您对一个或多个构建配置文件（或 `build.gradle` 文件）进行更改。这些纯文本文件使用域特定语言 (DSL) 以 Groovy (<http://groovy.codehaus.org/>) 语言描述和操作构建逻辑，后者是一种适用于 Java 虚拟机 (JVM) 的动态语言。您无需了解 Groovy 便可开始配置构建，因为 Android Plugin for Gradle 引入了您需要的大多数 DSL 元素。如需了解有关 Android 插件 DSL 的更多信息，请阅读 DSL 参考文档 (<http://google.github.io/android-gradle-dsl/current/index.html>)。

开始新项目时，Android Studio 会自动为您创建其中的部分文件（如图 2 所示），并为它们填充合理的默认值。

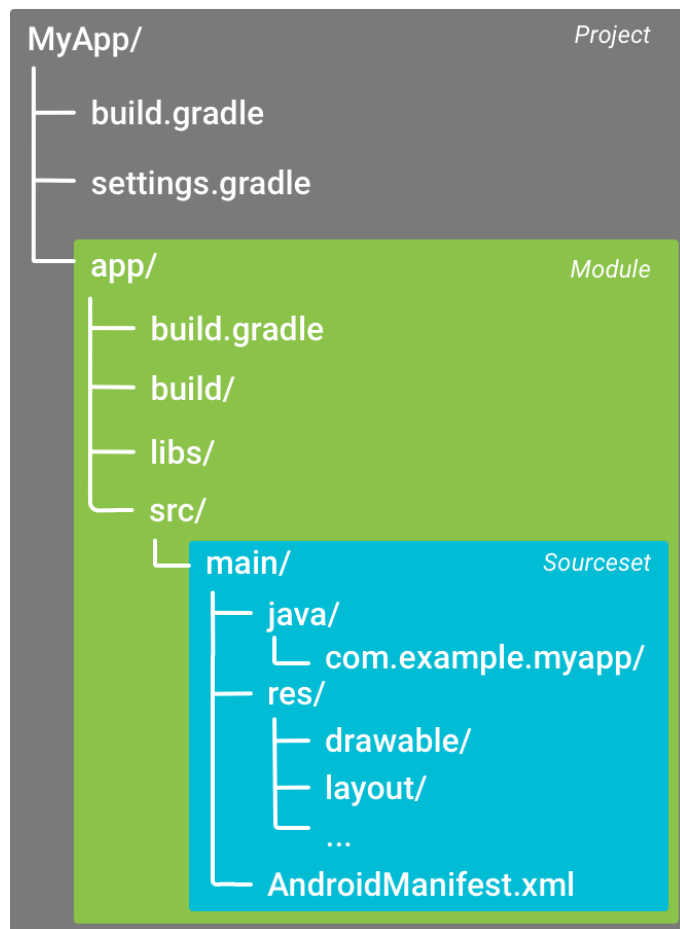


图 2. Android 应用模块的默认项目结构。

有几个 Gradle 构建配置文件是 Android 应用标准项目结构的组成部分。您必须了解其中每一个文件的范围和用途及其应定义的基本 DSL 元素，才能着手配置构建。

Gradle 设置文件

`settings.gradle` 文件位于项目根目录，用于指示 Gradle 在构建应用时应将哪些模块包括在内。对大多数项目而言，该文件很简单，只包括以下内容：

```
include ':app'
```

This site uses cookies to store your preferences for site-specific language and display options.

OK

不过，多模块项目需要指定应包括在最终构建之中的每个模块。

顶级构建文件

顶级 `build.gradle` 文件位于项目根目录，用于定义适用于项目中所有模块的构建配置。默认情况下，这个顶级构建文件使用 `buildscript {}` 代码块来定义项目中所有模块共用的 Gradle 存储区和依赖项。以下代码示例描述的默认设置和 DSL 元素可在新建项目后的顶级 `build.gradle` 文件中找到。

```
/**
 * The buildscript {} block is where you configure the repositories and
 * dependencies for Gradle itself--meaning, you should not include dependencies
 * for your modules here. For example, this block includes the Android plugin for
 * Gradle as a dependency because it provides the additional instructions Gradle
 * needs to build Android app modules.
 */

buildscript {

    /**
     * The repositories {} block configures the repositories Gradle uses to
     * search or download the dependencies. Gradle pre-configures support for remote
     * repositories such as JCenter, Maven Central, and Ivy. You can also use local
     * repositories or define your own remote repositories. The code below defines
     * JCenter as the repository Gradle should use to look for its dependencies.
     */

    repositories {
        jcenter()
    }

    /**
     * The dependencies {} block configures the dependencies Gradle needs to use
     * to build your project. The following line adds Android Plugin for Gradle
     * version 3.0.1 as a classpath dependency.
     */
}
```

This site uses cookies to store your preferences for site-specific language and display options.

OK

```
        classpath 'com.android.tools.build:gradle:3.0.1'
    }
}

/**
 * The allprojects {} block is where you configure the repositories and
 * dependencies used by all modules in your project, such as third-party plugins
 * or libraries. Dependencies that are not required by all the modules in the
 * project should be configured in module-level build.gradle files. For new
 * projects, Android Studio configures JCenter as the default repository, but it
 * does not configure any dependencies.
 */

allprojects {
    repositories {
        jcenter()
    }
}
```

模块级构建文件

模块级 `build.gradle` 文件位于每个 `<project>/<module>/` 目录，用于配置适用于其所在模块的构建设置。您可以通过配置这些构建设置来提供自定义打包选项（例如附加构建类型和产品风味），以及替换 `main/` 应用清单或顶级 `build.gradle` 文件中的设置。

以下这个示例 Android 应用模块 `build.gradle` 文件概述了您应该了解的部分基本 DSL 元素和设置。

```
/**
 * The first line in the build configuration applies the Android plugin for
 * Gradle to this build and makes the android {} block available to specify
 * Android-specific build options.
 */

apply plugin: 'com.android.application'

/**
```

This site uses cookies to store your preferences for site-specific language and display options.

OK


```
*/

android {

    /**
     * compileSdkVersion specifies the Android API level Gradle should use to
     * compile your app. This means your app can use the API features included in
     * this API level and lower.
     *
     * buildToolsVersion specifies the version of the SDK build tools, command-line
     * utilities, and compiler that Gradle should use to build your app. You need to
     * download the build tools using the SDK Manager.
     */

    compileSdkVersion 26
    buildToolsVersion "26.0.2"

    /**
     * The defaultConfig {} block encapsulates default settings and entries for all
     * build variants, and can override some attributes in main/AndroidManifest.xml
     * dynamically from the build system. You can configure product flavors to override
     * these values for different versions of your app.
     */

    defaultConfig {

        /**
         * applicationId uniquely identifies the package for publishing.
         * However, your source code should still reference the package name
         * defined by the package attribute in the main/AndroidManifest.xml file.
         */

        applicationId 'com.example.myapp'

        // Defines the minimum API level required to run the app.
        minSdkVersion 15

        // Specifies the APT level used to test the app
    }
}
```

This site uses cookies to store your preferences for site-specific language and display options.

OK

```
// Defines the version number of your app.
versionCode 1

// Defines a user-friendly version name for your app.
versionName "1.0"
}

/**
 * The buildTypes {} block is where you can configure multiple build types.
 * By default, the build system defines two build types: debug and release. The
 * debug build type is not explicitly shown in the default build configuration,
 * but it includes debugging tools and is signed with the debug key. The release
 * build type applies Proguard settings and is not signed by default.
 */

buildTypes {

    /**
     * By default, Android Studio configures the release build type to enable code
     * shrinking, using minifyEnabled, and specifies the Proguard settings file.
     */

    release {
        minifyEnabled true // Enables code shrinking for the release build type.
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}

/**
 * The productFlavors {} block is where you can configure multiple product
 * flavors. This allows you to create different versions of your app that can
 * override defaultConfig {} with their own settings. Product flavors are
 * optional, and the build system does not create them by default. This example
 * creates a free and paid product flavor. Each product flavor then specifies
 * its own application ID, so that they can exist on the Google Play Store, or
 * an Android device, simultaneously.
 */
```

```
productFlavors {
    free {
        applicationId 'com.example.myapp.free'
    }

    paid {
        applicationId 'com.example.myapp.paid'
    }
}

/**
 * The splits {} block is where you can configure different APK builds that
 * each contain only code and resources for a supported screen density or
 * ABI. You'll also need to configure your build so that each APK has a
 * different versionCode.
 */

splits {
    // Screen density split settings
    density {

        // Enable or disable the density split mechanism
        enable false

        // Exclude these densities from splits
        exclude "ldpi", "tvdpi", "xxxhdpi", "400dpi", "560dpi"
    }
}

/**
 * The dependencies {} block in the module-level build configuration file
 * only specifies dependencies required to build the module itself.
 */

dependencies {
    compile project(":lib")
    compile 'com.android.support:appcompat-v7:27.0.2'
```

This site uses cookies to store your preferences for site-specific language and display options.

OK

```
compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Gradle 属性文件

Gradle 还包括两个属性文件，位于项目根目录，可用于指定适用于 Gradle 构建工具包本身的设置：

gradle.properties


您可以在其中配置项目范围 Gradle 设置，例如 Gradle 后台进程的最大堆大小。如需了解详细信息，请参阅构建环境 (https://docs.gradle.org/current/userguide/build_environment.html)。

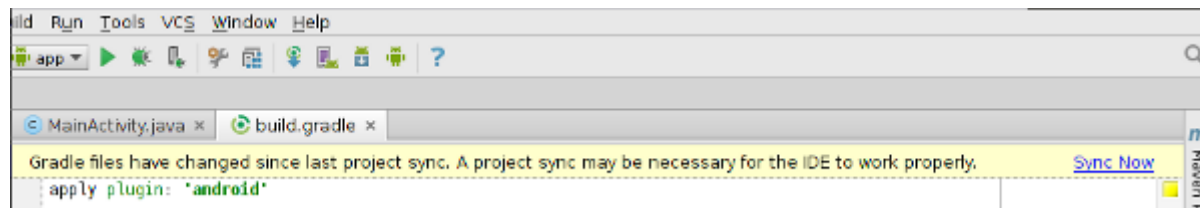
local.properties

为构建系统配置本地环境属性，例如 SDK 安装路径。由于该文件的内容由 Android Studio 自动生成并且专用于本地开发者环境，因此您不应手动修改该文件，或将其纳入您的版本控制系统。

将项目与 Gradle 文件同步

当您在项目中对构建配置文件进行更改时，Android Studio 会要求您同步项目文件，以便其导入您的构建配置更改并执行一些检查来确保您的配置不会造成构建错误。

要同步项目文件，您可以点击做出更改后出现的通知栏中的 **Sync Now**（如图 3 所示），或者点击菜单栏中的 **Sync Project** 。如果 Android Studio 通知配置出现错误，例如：您的源代码使用了只有在 `compileSdkVersion` 以上的 API 级别中才会提供的 API 功能，会显示 **Messages** 窗口，具体描述该问题。



This site uses cookies to store your preferences for site-specific language and display options.

OK

源集

Android Studio 按逻辑关系将每个模块的源代码和资源分组为源集。模块的 `main/` 源集包括其所有构建变体共用的代码和资源。其他源集目录为可选项，在您配置新的构建变体时，Android Studio 不会自动为您创建这些目录。不过，创建类似于 `main/` 的源集有助于让 Gradle 只应在构建特定应用版本时使用的文件和资源井然有序：

`src/main/`

此源集包括所有构建变体共用的代码和资源。

`src/<buildType>/`

创建此源集可加入特定构建类型专用的代码和资源。

`src/<productFlavor>/`

创建此源集可加入特定产品风味专用的代码和资源。

`src/<productFlavorBuildType>/`

创建此源集可加入特定构建变体专用的代码和资源。

例如，要生成应用的“完整调试”版本，构建系统需要合并来自以下源集的代码、设置和资源：

- `src/fullDebug/`（构建变体源集）
- `src/debug/`（构建类型源集）
- `src/full/`（产品风味源集）

注：当您在 Android Studio 中使用 **File > New** 菜单选项新建文件或目录时，可以针对特定源集进行创建。可供您选择的源集取决于您的构建配置，如果所需目录尚不存在，Android Studio 会自动创建。

如果不同源集包含同一文件的不同版本，Gradle 将按以下优先顺序决定使用哪一个文件（左侧源集替换右侧源集的文件和设置）：

构建变体 > 构建类型 > 产品风味 > 主源集 > 库依赖项

这样一来，Gradle 便可使用专用于您试图构建的构建变体的文件，同时对与其他应用版本共用的 Activity、应用逻辑和资源加以重复利用。在合并多个清单 (<https://developer.android.google.cn/studio/build/manifest-merge.html>) 时，Gradle 使用同一优先顺序，这样每个构建变体都能在最终清单中定义不同的组件或权限。如需了解有关创建自定义源集的更多信息，请转至创建用于构建变体的源集 (<https://developer.android.google.cn/studio/build/build-variants.html#sourcesets>)。



Follow Google Developers on
WeChat



Follow @AndroidDev on
Twitter



Follow Android Developers on
Google+



Check out Android Developers
on YouTube

