Android Developers

# 创建 Android 库

Android 库在结构上与 Android 应用模块相同。它可以提供构建应用所需的一切内容，包括源代码、资源文件和 Android 清单。不过，Android 库将编译到您可以用作 Android 应用模块依赖项的 Android 归档 (AAR) 文件，而不是在设备上运行的 APK。与 JAR 文件不同，AAR 文件可以包含 Android 资源和一个清单文件，这样，除了 Java 类与方法外，您还可以捆绑布局和可绘制对象等共享资源。

本文内容



库模块在以下情况下非常有用：

- 构建使用某些相同组件（例如 Activity、服务或 UI 布局）的多个应用。
- 构建存在多个 APK 变体（例如免费版本和付费版本）的应用并且需要在两种版本中使用相同的核心组件。

在任何一种情况下，只需要将您希望重用的文件移动到库模块中，然后以依赖项的形式为每个应用模块添加库。本页面将说明如何执行这两个操作。

## 创建库模块

要在您的项目中创建一个新的库模块，请按以下步骤操作：

1. 点击 **File > New > New Module**。
2. 在出现的 **Create New Module** 窗口中，依次点击 **Android Library** 和 **Next**。

   还存在一个用于创建 **Java** 库的选项，可以构建传统的 JAR 文件。尽管 JAR 文件在大多数项目中都非常实用（尤其在您希望与其他平台共享代码时），但这种文件不允许您包含 Android 资源或清单文件，而后者对于 Android 项目中的代码重用非常有用。因此，本指南将侧重论述创建 Android 库。
3. 为您的库命名，并为库中代码选择一个最低的 SDK 版本，然后点击 **Finish**。

在 Gradle 项目同步完成后，库模块将显示左侧的 **Project** 面板中。如果您未看到新模块文件夹，请确保将视图切换为 Android 视图 (https://developer.android.com/studio/projects/index.html?hl=zh-cn#ProjectFiles)。

## 将应用模块转换为库模块

如果您现有的应用模块包含您希望重用的所有代码，则可以按照以下步骤将其转换为库模块：

1. 打开现有应用模块的 `build.gradle` 文件。您应在顶部看到以下内容：

   ```
   apply plugin: 'com.android.application'
   ```

2. 按照下面所示更改插件分配：

   ```
   apply plugin: 'com.android.library'
   ```

3. 点击 **Sync Project with Gradle Files**。

就这么简单。模块的整个结构仍然相同，但是现在它将作为 Android 库运行，构建也将创建一个 AAR 文件，而不是 APK。

# 以依赖项形式添加您的库

要在另一个应用模块中使用您的 Android 库的代码，请按以下步骤操作：

1. 通过两种方式之一将库添加到您的项目（如果您是在相同项目中创建的库模块，则该模块已经存在，您可以跳过此步骤）：
   - 添加已编译的 AAR（或 JAR）文件：
     1. 点击 **File > New Module**。
     2. 依次点击 **Import .JAR/.AAR Package** 和 **Next**。
     3. 输入 AAR 或 JAR 文件的位置，然后点击 **Finish**。
   - 将库模块导入到您的项目中：
     1. 点击 **File > New > Import Module**。
     2. 输入库模块目录的位置，然后点击 **Finish**。

   库模块将复制到您的项目中，因此您可以尽管编辑库代码。如果您希望维护一个版本的库代码，则此方

法可能不是您想要的，您应按照上文所述导入编译的 AAR 文件。

2. 确保库列在您 `settings.gradle` 文件的顶部，如下面名为“my-library-module”的库所示：

```
include ':app', ':my-library-module'
```

3. 打开应用模块的 `build.gradle` 文件，并向 `dependencies` 块中添加一行新代码，如下面的片段所示：

```
dependencies {
    compile project(":my-library-module")
}
```

4. 点击 **Sync Project with Gradle Files**。

在上面的示例中，名为 `my-library- module` 的 Android 库模块成为 `build.gradle` 文件所在模块的构建依赖项。

您的应用模块现在可以访问 Android 库中的任何代码和资源，库 AAR 文件在构建时已捆绑到您的 APK 中。

不过，如果希望单独共享 AAR 文件，则可以在 *项目名称*/*模块名称*/build/outputs/aar/ 中找到它，也可以通过点击 **Build > Make Project** 的方式重新生成此文件。

# 选择要设为公开的资源

库中的所有资源在默认情况下均处于公开状态。要将所有资源隐式设为私有，您必须至少将一个特定的属性定义为公开。资源包括您项目的 `res/` 目录中的所有文件，例如图像。要阻止您的库用户访问仅供内部使用的资源，您应通过声明一个或多个公开资源的方式来使用这种自动私有标识机制。

要删除某个公开资源，请将一个 声明添加到您的库的 `public.xml` 文件中。如果您之前尚未添加公开资源，则需要在您的库的 `res/values/` 目录中创建 `public.xml` 文件。

下面的示例代码可以创建两个名称分别为 `mylib_app_name` 和 `mylib_public_string` 的公开字符串资源：

```
<resources>
    <public name="mylib_app_name" type="string"/>
    <public name="mylib_public_string" type="string"/>
</resources>
```

如果希望任何资源保持对使用您的库的开发者可见，您应当将其设为公开。例如，尽管 v7 appcompat 库 (https://developer.android.com/topic/libraries/support- library/features.html?hl=zh-cn#v7-appcompat)中的大多数资源都是私有

资源，但是为了支持 Material Design (https://developer.android.com/design/material/index.html?hl=zh-cn)，控制工具栏小部件的属性应当公开。

将属性隐式设为私有不仅可以阻止您的库用户从内部库资源获得代码自动完成建议，还让您能够在不中断您的库客户端的情况下重命名或移除私有资源。私有资源不在代码自动完成和 Theme Editor (https://developer.android.com/studio/write/theme-editor.html?hl=zh-cn) 的作用范围内，并且如果您尝试引用私有资源，Lint (https://developer.android.com/studio/write/lint.html?hl=zh-cn) 将显示警告。

# 开发注意事项

在开发您的库模块和相关应用时，请注意以下行为和限制。

将库模块引用添加至您的 Android 应用模块后，您可以设置它们的相对优先级。构建时，库会按照一次一个的方式与应用合并，并按照从低到高的优先级顺序。

- 资源合并冲突

  构建工具会将库模块中的资源与相关应用模块的资源合并。如果在两个模块中均定义了给定资源 ID，将使用应用中的资源。

  如果多个 AAR 库之间发生冲突，将使用依赖项列表首先列出（位于 `dependencies` 块顶部）的库中的资源。

  为了避免常用资源 ID 的资源冲突，请使用在模块（或在所有项目模块）中具有唯一性的前缀或其他一致的命名方案。

- 库模块可以包含 **JAR** 库

  您可以开发一个自身包含 JAR 库的库模块；不过，您需要手动编辑相关应用模块的构建路径，并添加 JAR 文件的路径。

- 库模块可以依赖外部 **JAR** 库

  您可以开发一个依赖于外部库（例如 Maps 外部库）的库模块。在这种情况下，相关应用必须针对包含外部库（例如 Google API 插件）的目标构建。另外也要注意，库模块和相关应用都必须在其清单文件的 `<uses- library>` (https://developer.android.com/guide/topics/manifest/uses-library-element.html?hl=zh-cn) 元素中声明外部库。

- 库模块不得包含原始资源

  工具不支持在库模块中使用原始资源文件（保存在 `assets/` 目录中）。应用使用的任何原始资源都必须存储在应用模块自身的 `assets/` 目录中。

- 应用模块的 `minSdkVersion` 必须大于或等于库定义的版本

  库作为相关应用模块的一部分编译，因此，库模块中使用的 API 必须与应用模块支持的平台版本兼容。

- 每个库模块都会创建自己的 **R** 类

  在您构建相关应用模块时，库模块将先编译到 AAR 文件中，然后再添加到应用模块中。因此，每个库都有其自己的 `R` 类，并根据库的软件包名称命名。从主模块和库模块生成的 `R` 类会在所需的所有软件包（包括主模块的软件包和库的软件包）中创建。

- 库模块可能包含自己的 **ProGuard** 配置文件

  通过将 ProGuard (https://developer.android.com/studio/build/shrink-code.html?hl=zh-cn) 配置文件添加到包含其 ProGuard 指令的库，您可以在自己的库上启用代码压缩。构建工具会为库模块将此文件嵌入到生成的 AAR 文件中。在您将库添加到应用模块时，库的 ProGuard 文件将附加至应用模块的 ProGuard 配置文件 (`proguard.txt`)。

  通过将 ProGuard 文件嵌入到您的库模块中，您可以确保依赖于此库的应用模块不必手动更新其 ProGuard 文件即可使用库。当 ProGuard 在 Android 应用模块上运行时，它会同时使用来自应用模块和库的指令，因此您不应当只在库上运行 ProGuard。

  要指定您的库的配置文件名称，请将其添加到 `consumerProguardFiles` 方法中，此方法位于您的库的 `build.gradle` 文件的 `defaultConfig` 块内。例如，以下片段会将 `lib-proguard-rules.txt` 设置为库的 ProGuard 配置文件：

  ```
  android {
      defaultConfig {
          consumerProguardFiles 'lib-proguard-rules.txt'
      }
      ...
  }
  ```

  默认情况下，应用模块会使用库的发布构建，即使在使用应用模块的调试构建类型时亦是如此。要使用库中不同的构建类型，您必须将依赖项添加到应用的 `build.gradle` 文件的 `dependencies` 块中，并在库的 `build.gradle` 文件中将 `publishNonDefault` 设置为 `true`。例如，您应用的 `build.gradle` 文件中的以下代码段会使应用在应用模块于调试模式下构建时使用库的调试构建类型，以及在应用模块于发布模式下构建时使用库的发布构建类型：

  ```
  dependencies {
      debugCompile project(path: ':library', configuration: 'debug')
      releaseCompile project(path: ':library', configuration: 'release')
  }
  ```

  您还必须在自己库的 `build.gradle` 文件的 `android` 块内添加以下代码行，以便将此库的非发布配置展示

给使用它的项目：

```
android {
    ...
    publishNonDefault true
}
```

不过请注意，设置 `publishNonDefault` 会增加构建时间。

为了确保您的库的 ProGuard 规则不会将意外的压缩副作用施加到应用模块，请仅包含适当规则，停用不适用于此库的 ProGuard 功能。尝试协助开发者的规则可能会与应用模块或它的其他库中的现有代码冲突，因此不应包含这些规则。例如，您的库的 ProGuard 文件可以指定在应用模块的压缩期间需要保留的代码 (https://developer.android.com/studio/build/shrink-code.html?hl=zh-cn#keep-code)。

> 注：Jack 工具链 (https://source.android.com/source/jack.html?hl=zh-cn#shrinking_and_obfuscation)仅支持 ProGuard 的部分压缩和模糊选项。

# AAR 文件详解

AAR 文件的文件扩展名为 `.aar`，Maven 工件类型也应当是 `aar`。文件本身是一个包含以下强制性条目的 `zip` 文件：

- `/AndroidManifest.xml`
- `/classes.jar`
- `/res/`
- `/R.txt`

此外，AAR 文件可能包含以下可选条目中的一个或多个：

- `/assets/`
- `/libs/`*`名称`*`.jar`
- `/jni/`***`abi 名称`***`/`*`名称`*`.so`（其中 ***`abi 名称`*** 是 Android 支持的 ABI (https://developer.android.com/ndk/guides/abis.html?hl=zh-cn#sa) 之一）
- `/proguard.txt`
- `/lint.jar`

Follow @AndroidDev on Twitter

Follow Android Developers on Google+

Check out Android Developers on YouTube