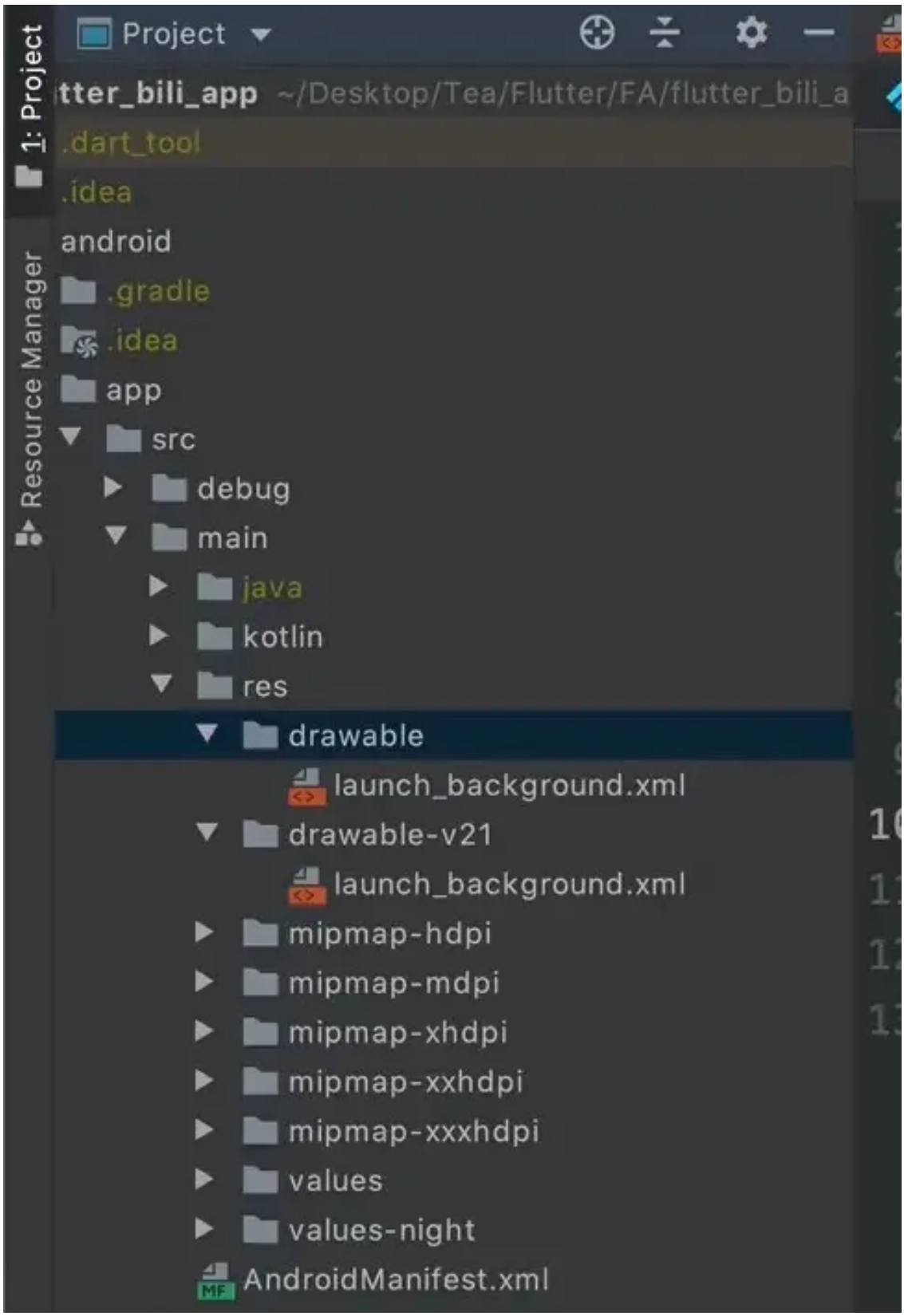
Flutter集成与打包Android应用

一个半吊子工程师

添加启动屏

添加启动背景图

修改文件目录`drawable-v21\launch_background.xml`与`drawable\launch_background.xml`


Project
tter_bili_app ~/Desktop/Tea/Flutter/FA/flutter_bili_a
.dart_tool
.idea
android
.gradle
.idea
app
src
debug
main
java
kotlin
res
drawable
launch_background.xml
drawable-v21
launch_background.xml
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-night
AndroidManifest.xml
1: Project
Resource Manager

```
<?xml version="1.0" encoding="utf-8"?><layer-list
xmlns:android="http://schemas.android.com/apk/res/android">
//添加 splash_screen_background这个是图片
    <item android:drawable="@drawable/
splash_screen_background" /></layer-list>
```

设置启动屏全屏

styles.xml

```
    <style name="LaunchTheme" parent="@android:style/
Theme.Light.NoTitleBar">

        <item name="android:windowBackground">@drawable/
launch_background</item>

        <item name="android:windowFullscreen">true</item>

    </style>
```

支持刘海屏

找到 AndroidManifest.xml

```
<activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|
keyboardHidden|keyboard|screenSize|smallestScreenSize|
locale|layoutDirection|fontScale|screenLayout|density|
uiMode"
```

```
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <!-- 支持刘海屏-->
            <meta-data android:name="android.notch_support"
android:value="true"/>
```

设置状态栏为透明色

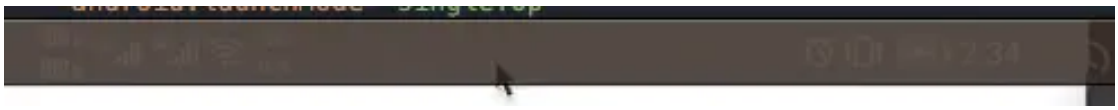


image.png

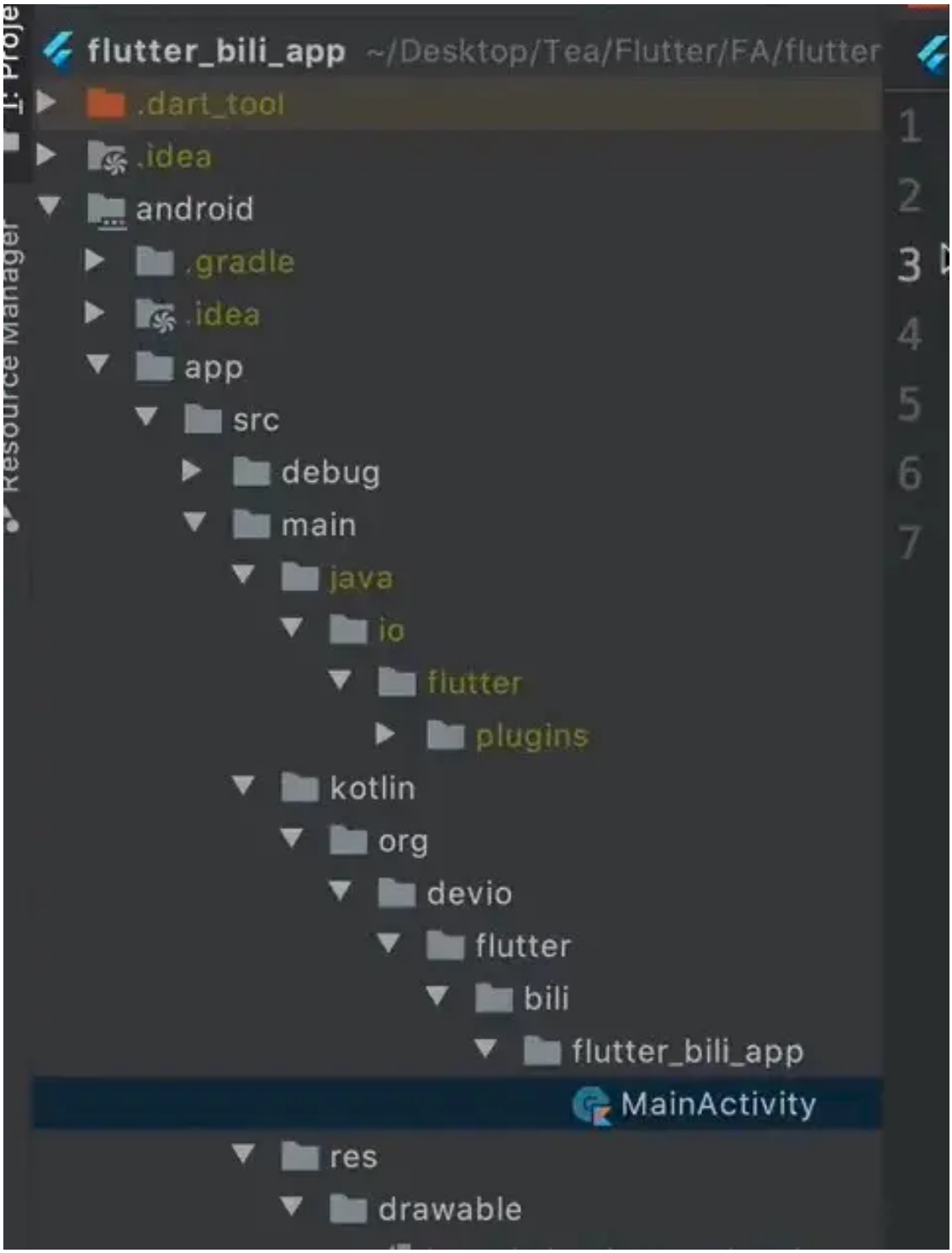


image.png

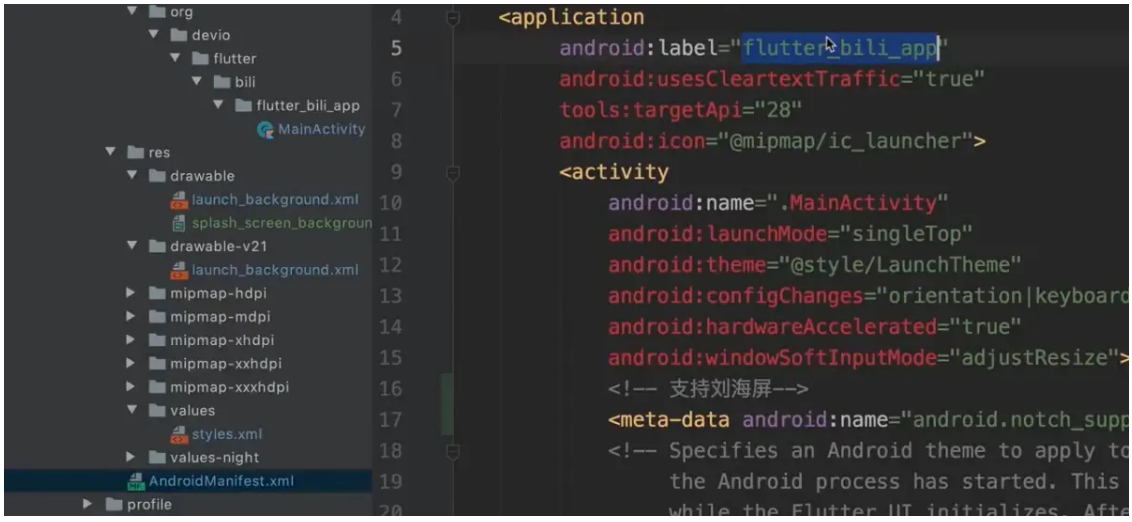
```
package org.devio.flutter.bili.flutter_bili_app
```

```
 import android.graphics.Color
 import android.os.Build
 import android.os.Bundleimport
io.flutter.embedding.android.FlutterActivity

classMainActivity : FlutterActivity() {
     override fun onCreate(savedInstanceState: Bundle?) {
         super.onCreate(savedInstanceState)
         if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {
             valwindow = activity.window
             //设置状态栏为透明色，fix 启动时状态栏会灰色闪一下
             window.statusBarColor = Color.TRANSPARENT
         }
     }
}
```

修改后需要重新运行

APP名字



快照名

```
return MultiProvider(
    providers: topProviders,
    child: Consumer<ThemeProvider>(builder: (BuildContext c
        ThemeProvider themeProvider, Widget child) {
      return MaterialApp(
        home: widget,
        theme: themeProvider.getTheme(),
        darkTheme: themeProvider.getTheme(isDarkMode: tru
        themeMode: themeProvider.getThemeMode(),
       title: 'Flutter',

      );  // MaterialApp

    }));  // Consumer, MultiProvider
});  // FutureBuilder
```

AppID 和版本号

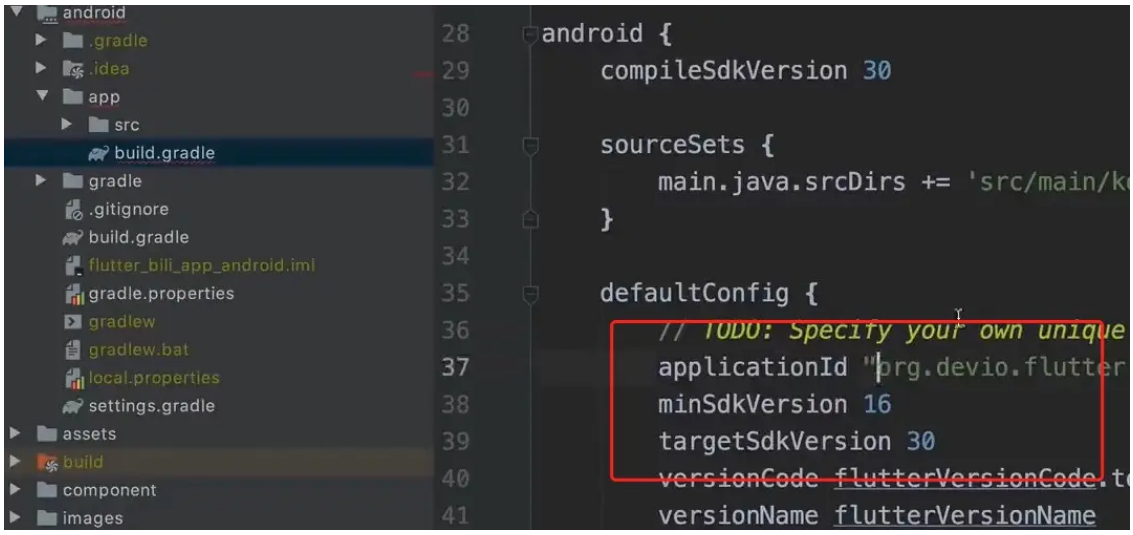
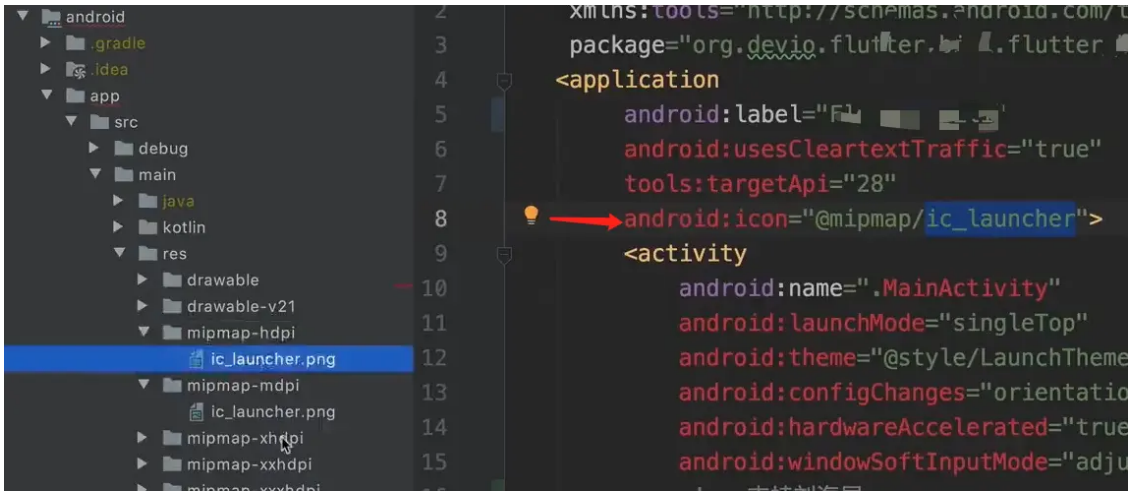


image.png

启动图标



配置网络权限

在`AndroidManifest.xml`中添加

```
<uses-permission
android:name="android.permission.INTERNET"/>
```

签名 APP

构建 release 包需要用到 keystore，如果你还没有 keystore 可以通过一下方式来创建：

创建 keystore

Mac：

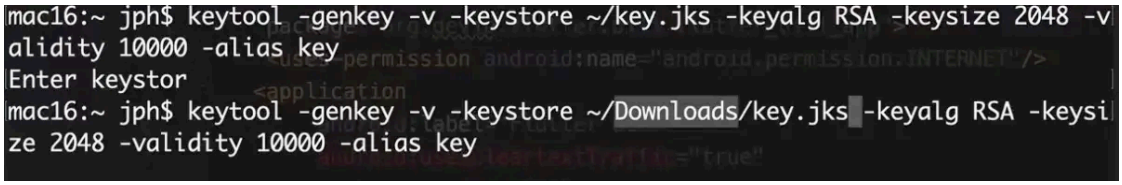
```
keytool -genkey -v -keystore ~/key.jks -keyalg RSA -keysize
2048 -validity 10000 -alias key
```

Windows：

```
keytool -genkey -v -keystore c:\Users\USER_NAME\key.jks
-storetype JKS -keyalg RSA -keysize 2048 -validity 10000
-alias key
```

-keystore：用户指定存储路径

然后按照提示进行输入，完成所有输入之后会在上述路径中创建一个key.jks文件。然后将该文件复制到Flutter项目下的android目录下。



配置 keystore

在你的flutter应用/android目录下创建key.properties文件，然后添加：

```
storePassword=<password from previous step>

keyPassword=<password from previous step>

keyAlias=key

storeFile=../key.jks
```

配置打包签名

用AS打开你的flutter应用/android/app/build.gradle文件然后在android代码块上面添加：

```
   def keystoreProperties = new Properties()
```

```
   def keystorePropertiesFile =
rootProject.file('key.properties')

   if (keystorePropertiesFile.exists()) {

       keystoreProperties.load(new
FileInputStream(keystorePropertiesFile))

   }

   android {

         ...

   }
```

然后在 android 代码块中添加 signingConfigs：

```
    signingConfigs {

        release {

            keyAlias keystoreProperties['keyAlias']

            keyPassword keystoreProperties['keyPassword']

            storeFile file(keystoreProperties['storeFile'])
```

```
            storePassword
keystoreProperties['storePassword']

        }

    }
```

最后在 buildTypes 代码块中添加 release 配置：

```
    buildTypes {

        release {

            signingConfig signingConfigs.release

        }

    }
```

构建 release 包

构建全部架构的安装包

```
./gradlew assembleRelease
```

构建出来的 Release 包是包含所有 ABI 架构的。

构建单一架构的安装包

```
cd <flutter应用的android目录>
```

```
flutter build apk --split-per-abi

 flutter build: 命令默认会构建出release包

 --split-per-abi: 表示构建单一架构
```

上传应用

安装包构建好之后发布到 Android 各大应用市场，无论是长传到那个应用市场首先需要注册该平台的开发者通过开发者认证后便可进行应用上传了。

开发者注册和应用上传各大平台都有详细的说明教程和问题，下面分享国内比较大的应用市场：

· 华为：https://consumer.huawei.com/cn/

· 小米：https://dev.mi.com/console/

· OPPO：https://open.oppomobile.com/

· vivo：https://dev.vivo.com.cn/home

· 百度：https://app.baidu.com/apps/

有需要的小伙伴可以按照官方的说明进行注册和上传应用。

FAQ

启用代码压缩后无法打包

很多同学会发现当启用代码压缩 (minifyEnabled true) 后，在打 release 包时报错，无法打出 release 包，那么只需将

targetSdkVersion 和 compileSdkVersion 升级到 28。

关联问题 @https://github.com/flutter/flutter/issues/26860#issuecomment-469751224

安装 release 包运行 crash 报 so 包找不到

flutter build apk 会构建出包含 x86_64、arm64-v8a、armeabi-v7a 架构的安装包，但如何项目中所依赖的某个库和 Flutter 所支持的架构不一致就会出现 so 找不到的 crash，比如：XX 三方库仅有 armeabi-v7a 的 so，当 APP 被安装到支持 arm64-v8a 的手机上时，手机发现 APP 中包含 arm64-v8a 的目录，于是就向这个目录中查找某某三方库的 so 发现找不到就报错了。 解决方案是只会打 armeabi-v7a 架构的 flutter so，所以在 release 包前需要添加如下配置：

```
ndk {
//只打包flutter所支持的架构，flutter没有armeabi架构的so，加x86的原因是为了能够兼容模拟器
//abiFilters "armeabi-v7a","arm64-v8a","x86_64","x86"
//release 时只打"armeabi-v7包
abiFilters "armeabi-v7a"
}
```