

Android实现多渠道打包，动态替换包名、Icon、图片等资源的方案

目的或需求

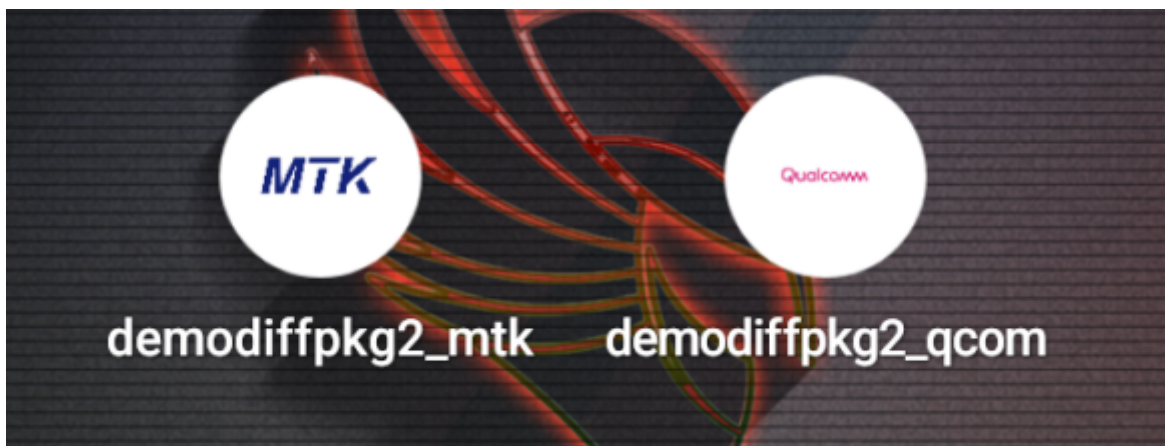
主要是一套代码，编译出二个不同包名的应用，并且可以自动根据不同的配置来动态替换包名，应用名，应用图标，字符串，图片等。

按比较专业的术语描述此需求为：

要实现一个壳工程打出不同样式的包，也就是使用Gradle中的productFlavors，在做定制或适配的时候，不需要建立多个工程、来回切换项目分支、逐个编译apk，使用productFlavors可以帮我们简化这一步操作，快速打包所有项目版本的apk。

实现效果

生成二个不同包名的apk，并且应用图标和应用名称实现动态替换



图一：桌面显示二个不同包名应用动态替换不同应用图标和应用名称的效果图

图标，字符串实现动态替换



图二：显示二个不同应用图标和字符串动态替换的效果图

Demo源码

这个demo源码，非常的简单，主要是显示一个activity界面，此界面包括二个TextView和一个ImageView。如上面的图二。

实现方案

app\build.gradle

```
1 android.buildFeatures.buildConfig true
2
3 flavorDimensions "channel"
4 productFlavors{
5     mtk{
6         manifestPlaceholders = [str:"mtk",
```

```

7             package_name:"com.example.demodiffpkg2_mtk",
8             APP_LOGO_ICON:"@drawable/logo_mtk",
9             APP_NAME:"@string/app_name_mtk"]
10         applicationId "com.example.demodiffpkg2_mtk"
11         versionCode 1
12         versionName "1.0"
13
14         //resValue "string","app_name","demodiffpkg2_mtk"
15         resValue "string","text_name","demodiffpkg2_mtk"
16
17         buildConfigField("String", "HOST", "\"mtk\"")
18         buildConfigField("boolean", "isEnabled", "true")
19     }
20
21     qcom{
22         manifestPlaceholders = [str:"qcom",
23             package_name:"com.example.demodiffpkg2_qcom",
24             APP_LOGO_ICON:"@drawable/logo_qcom",
25             APP_NAME:"@string/app_name_qcom"]
26         applicationId "com.example.demodiffpkg2_qcom"
27         versionCode 2
28         versionName "2.0"
29
30         //resValue "string","app_name","demodiffpkg2_qcom"
31         resValue "string","text_name","demodiffpkg2_qcom"
32
33         buildConfigField("String", "HOST", "\"qcom\"")
34         buildConfigField("boolean", "isEnabled", "false")
35     }
36 }

```

其配置项的解释说明：

配置生成不同的包名

对应我们添加二个项目(mtk,qcom)，分别生成包名为(com.example.demodiffpkg2_mtk, com.example.demodiffpkg2_qcom)的二个apk应用。

```

1 productFlavors{
2     mtk{
3         manifestPlaceholders = [str:"mtk",
4             package_name:"com.example.demodiffpkg2_mtk",
5             APP_LOGO_ICON:"@drawable/logo_mtk",

```

```

6             APP_NAME:"@string/app_name_mtk"]
7         applicationId "com.example.demodiffpkg2_mtk"
8         .....
9     }
10
11     qcom{
12         manifestPlaceholders = [str:"qcom",
13             package_name:"com.example.demodiffpkg2_qcom",
14             APP_LOGO_ICON:"@drawable/logo_qcom",
15             APP_NAME:"@string/app_name_qcom"]
16         applicationId "com.example.demodiffpkg2_qcom"
17         .....
18     }
19 }

```

动态替换应用图标和应用名

```

1 productFlavors{
2     mtk{
3         manifestPlaceholders = [str:"mtk",
4             package_name:"com.example.demodiffpkg2_mtk",
5             APP_LOGO_ICON:"@drawable/logo_mtk",
6             APP_NAME:"@string/app_name_mtk"]
7         .....
8     }
9
10    qcom{
11        manifestPlaceholders = [str:"qcom",
12            package_name:"com.example.demodiffpkg2_qcom",
13            APP_LOGO_ICON:"@drawable/logo_qcom",
14            APP_NAME:"@string/app_name_qcom"]
15        .....
16    }
17 }

```

在app\src\main\AndroidManifest.xml中配置对应的动态应用图标和应用名

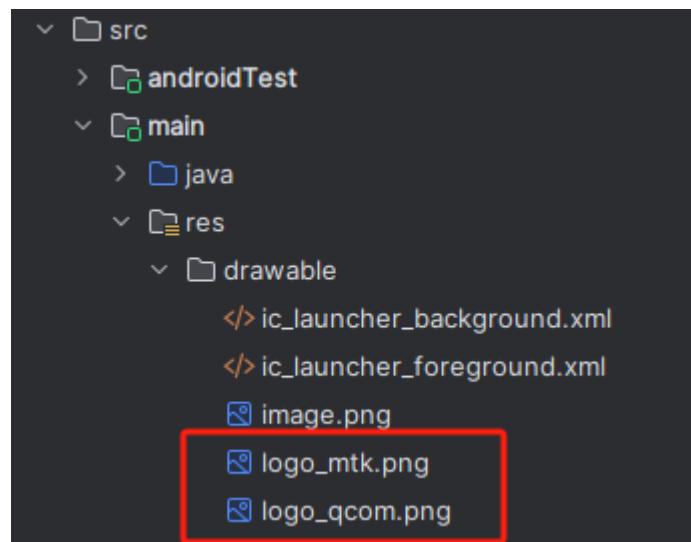
```

1 <application
2     android:allowBackup="true"

```

```
3     android:dataExtractionRules="@xml/data_extraction_rules"
4     android:fullBackupContent="@xml/backup_rules"
5     android:icon="@{APP_LOGO_ICON}"
6     android:label="@{APP_NAME}"
7     android:supportsRtl="true"
8     android:theme="@style/Theme.DemoDiffPkg2"
9     tools:targetApi="31">
```

logo_qcom.png和logo_mtk.png二张对应应用图标的图片：



其不同的应用名对应：

app\src\main\res\values\strings.xml

```
1 <string name="app_name_mtk">demodiffpkg2_mtk</string>
2 <string name="app_name_qcom">demodiffpkg2_qcom</string>
```

资源的overlay（包括图片，字符串），也可以说是动态替换资源文件----特别推荐！！

这部分，主要是activity主界面的图片和字符串：

对于应用mtk，我们将图片显示为mtk，字符串显示为tv1 mtk；

对于应用qcom，我们将图片显示为QualcoMM，字符串显示为tv1 qcom；



图三：资源文件overlay动态替换的效果图

界面布局为：

```
1 <ImageView
2     android:layout_width="100dp"
3     android:layout_height="100dp"
4     android:layout_margin="30dp"
5     android:background="@drawable/image"
6 />
7
8 <TextView
9     android:id="@+id/tv1"
10    android:layout_width="wrap_content"
11    android:layout_height="wrap_content"
12    android:layout_margin="30dp"
13    android:text="@string/text_name_1"
14 />
```

其主要的思想是：

在对应平行app/src/main此目录，添加我们定义的不同项目，分别添加需要overlay动态替换的资源文件。

这个完全就是android开发中常用的overlay替换的思路，此思路完全可以替换所有的资源文件，而不是仅仅为图片和字符串，并且此思路还有一个好处，就是可以兼顾到资源文件的国际化，各种不同的场景，最是适合我们的客制化开发，特别推荐。

如我们创建mtk和qcom二个目录：

```
1 mtk/java
2 mtk/res/drawable/image.png
3 mtk/res/values/strings.xml
4
5 qcom/java
6 qcom/res/drawable/image.png
7 qcom/res/values/strings.xml
```

字符串资源的动态替换：

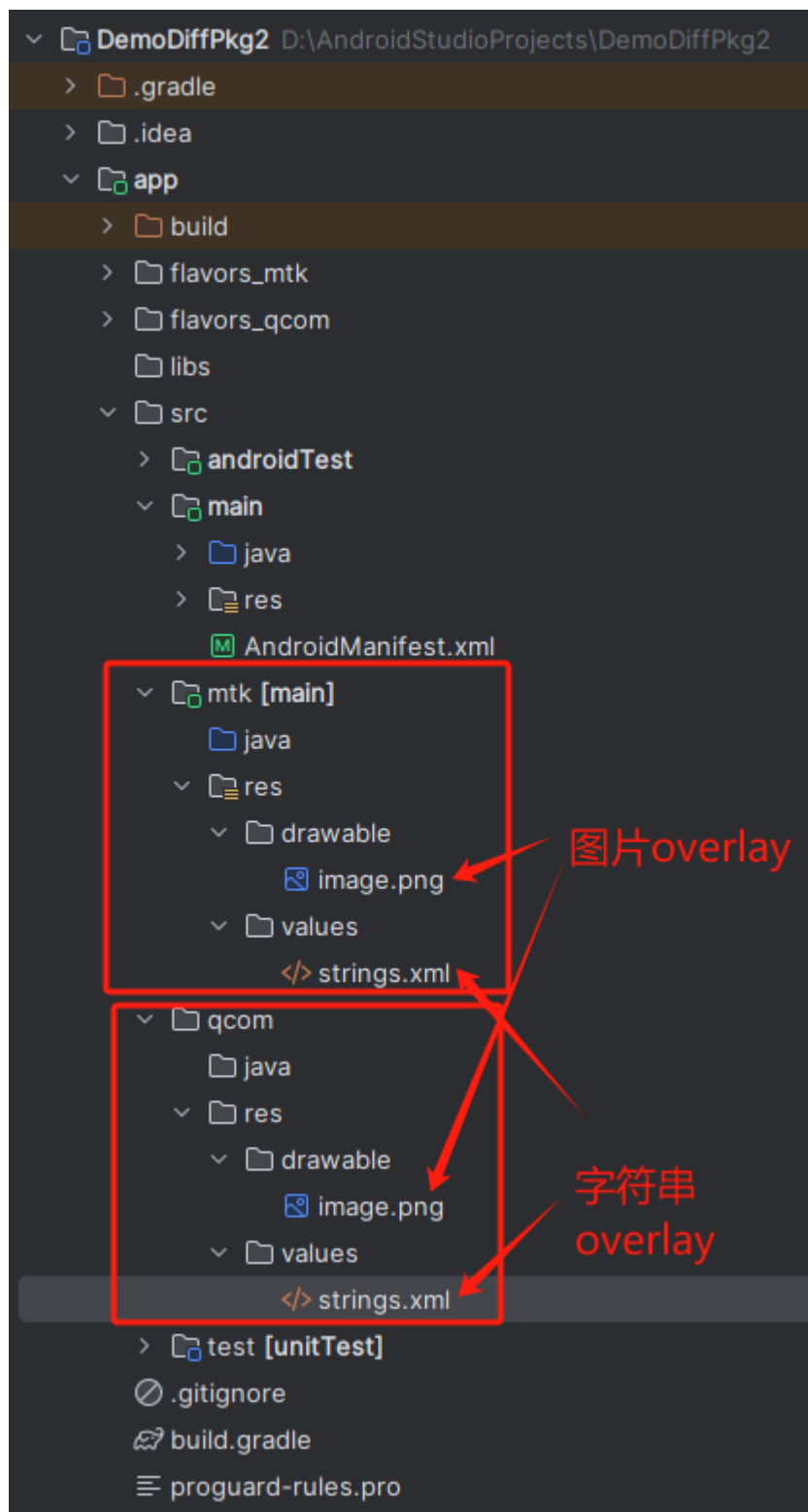
mtk/res/values/strings.xml

```
1 <resources>
2     <string name="text_name_1">tv1 mtk</string>
3 </resources>
```

qcom/res/values/strings.xml

```
1 <resources>
2     <string name="text_name_1">tv1 qcom</string>
3 </resources>
```

其drawable下的图片资源，values下的字符串，color等其他的资源，都是可以根据不同的项目来overlay动态替换的。



图四：资源文件overlay动态替换的代码结构

resValue简单的字符串动态替换

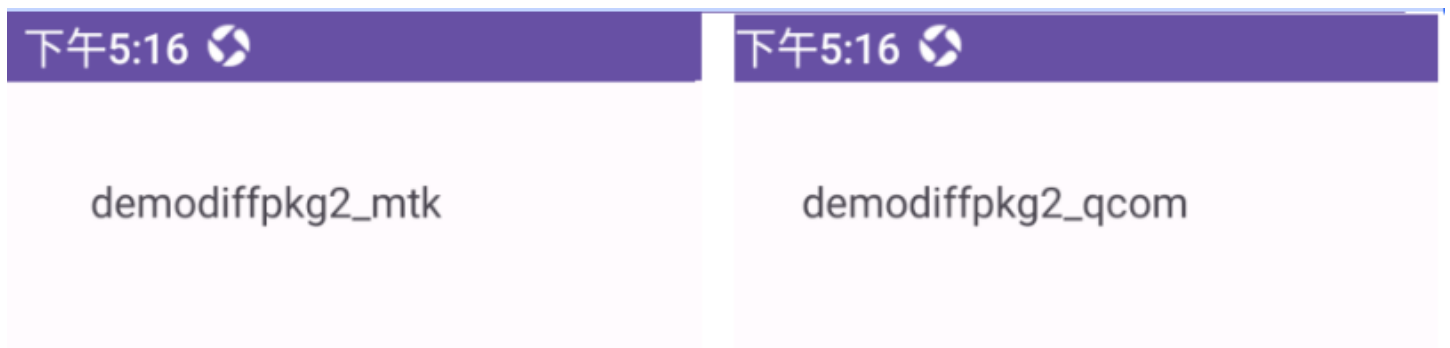
上面的资源overlay动态替换作用非常大，并且还可以实现字符串的国符化，当然缺点是需要创建文件，resValue可以非常方便的实现简单的字符串动态替换


```

1 flavorDimensions "channel"
2 productFlavors{
3     mtk{
4         .....
5         resValue "string","text_name","demodiffpkg2_mtk"
6     }
7
8     qcom{
9         .....
10        resValue "string","text_name","demodiffpkg2_qcom"
11    }
12 }

```

这个主要是实现主界面的第一个TextView的动态替换：



其布局文件为：

```

1 <TextView
2     android:layout_width="wrap_content"
3     android:layout_height="wrap_content"
4     android:layout_margin="30dp"
5     android:text="@string/text_name" />

```

text_name的字符串分别定义在：

其会自动生成为：

DemoDiffPkg2\app\build\generated\res\resValues\mtk\debug\values\gradleResValues.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>

```

```

3      <!-- Automatically generated file. DO NOT MODIFY -->
4
5      <!-- Value from product flavor: mtk -->
6      <string name="text_name" translatable="false">demodiffpkg2_mtk</string>
7
8  </resources>

```

buildConfigField动态配置

这个主要是对应自动生成BuildConfig类，方便项目来客制化：

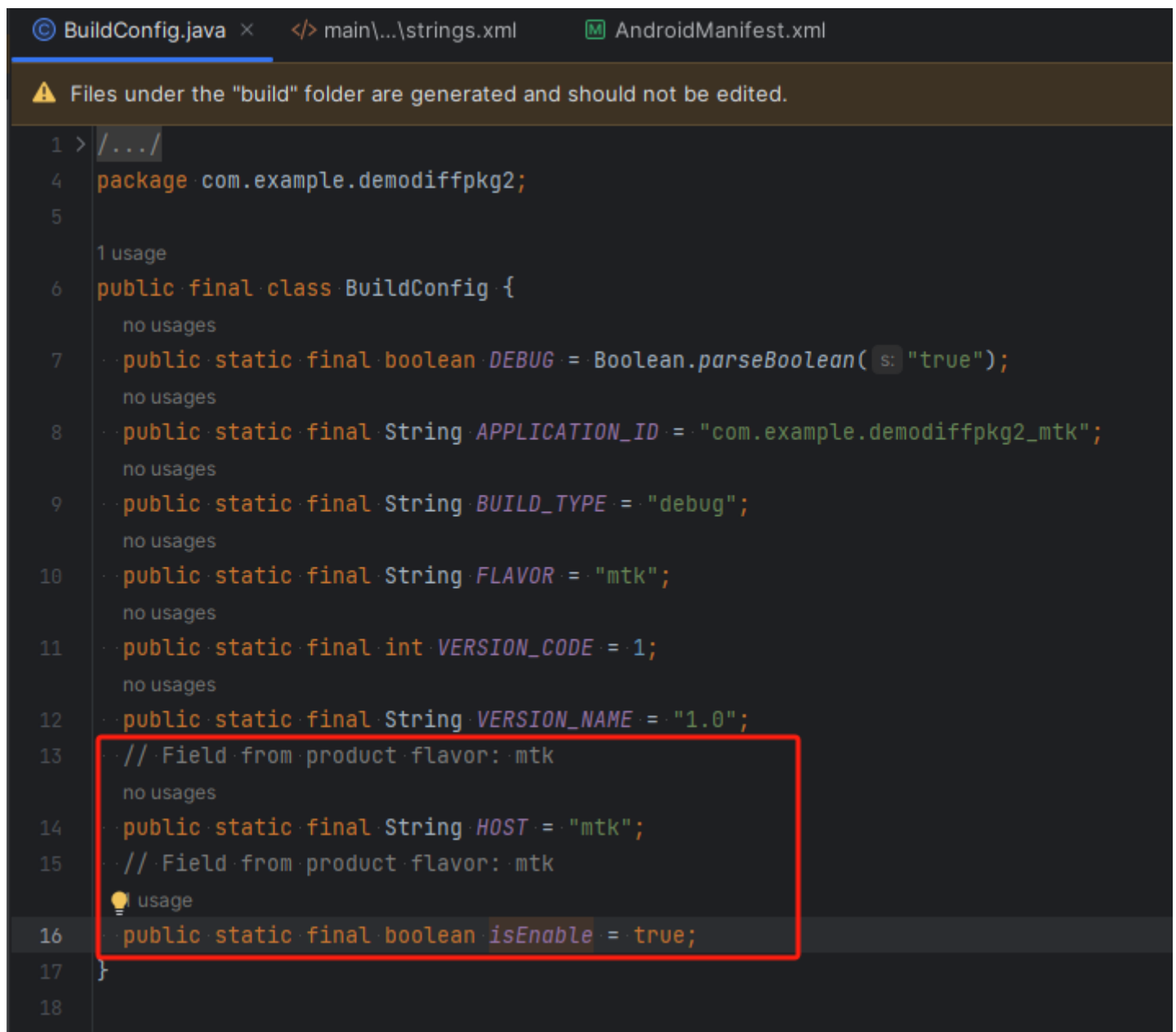
```

1  android.buildFeatures.buildConfig true
2
3  flavorDimensions "channel"
4  productFlavors{
5      mtk{
6          .....
7          buildConfigField("String", "HOST", "\"mtk\"")
8          buildConfigField("boolean", "isEnabled", "true")
9      }
10
11     qcom{
12         .....
13         buildConfigField("String", "HOST", "\"qcom\"")
14         buildConfigField("boolean", "isEnabled", "false")
15     }
16 }

```

对应会自动生成类：

app\build\generated\source\buildConfig\mtk\debug\com\example\demodiffpkg2\BuildConfig.java



```
BuildConfig.java x </> main\...\strings.xml AndroidManifest.xml
⚠ Files under the "build" folder are generated and should not be edited.

1 > /.../
4 package com.example.demodiffpkg2;
5
6 public final class BuildConfig {
7     public static final boolean DEBUG = Boolean.parseBoolean(s: "true");
8     public static final String APPLICATION_ID = "com.example.demodiffpkg2_mtk";
9     public static final String BUILD_TYPE = "debug";
10    public static final String FLAVOR = "mtk";
11    public static final int VERSION_CODE = 1;
12    public static final String VERSION_NAME = "1.0";
13    // Field from product flavor: mtk
14    public static final String HOST = "mtk";
15    // Field from product flavor: mtk
16    public static final boolean isEnabled = true;
17 }
18
```

java文件的调用方式为：

```
1 boolean isOK = BuildConfig.isEnabled;
```

动态替换版本号

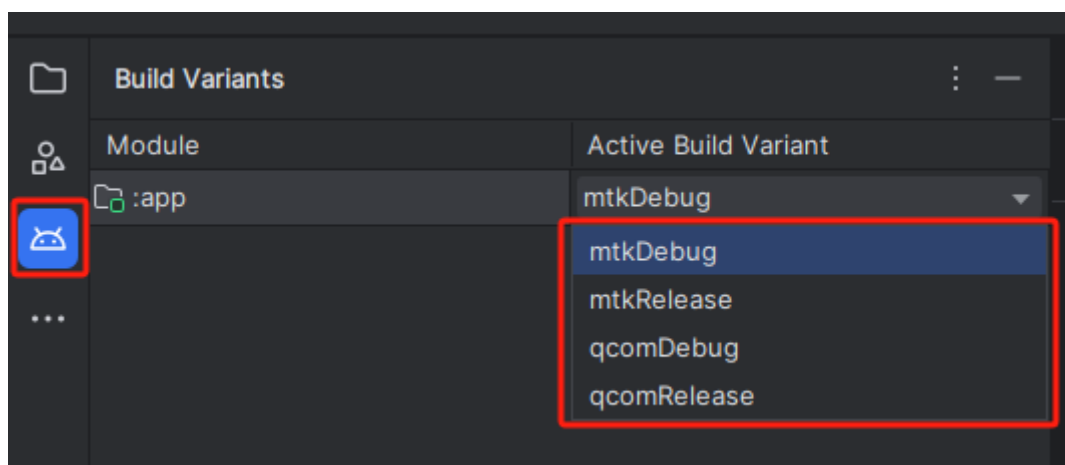
这个非常明显，只是动态设置一下：versionCode，versionName

```
1 flavorDimensions "channel"
2 productFlavors{
3     mtk{
```

```
4      .....
5      versionCode 1
6      versionName "1.0"
7  }
8
9  qcom{
10     .....
11     versionCode 2
12     versionName "2.0"
13 }
14 }
```

编译apk

在Build Variants----Active Build Variant中选择我们对应的项目来编译：



参考资料

<https://blog.csdn.net/woshizisezise/article/details/96303750>

Android实现多渠道打包，动态替换包名、Icon、图片等资源，解决因applicationId和BuildConfig路径不匹配的问题-CSDN博客

文章浏览阅读6.7w次，点赞6次，收藏19次。前言:Android实现多渠道打包，这个问题并不新鲜，解决方案是固定的那么几种，...

公司相关信息

com.retroidpocket.gameassistant

com.odin2.gameassistant

com.odin2.odinlauncher

com.retroidpocket.rp3launcher

com.retroidpocket.rp3settings

com.odin2.rp3settings