动脑学院

做一家受人尊敬的企业，做一位受人尊敬的老师

突破学习瓶颈　掌握核心技术

动脑学院
DONGNAOCOLLEGE

Android Apk瘦身之旅

主讲老师：david

动脑学院
DONGNAOCOLLEGE

SVG

https://developer.android.google.cn/studio/write/vector-asset-studio.html

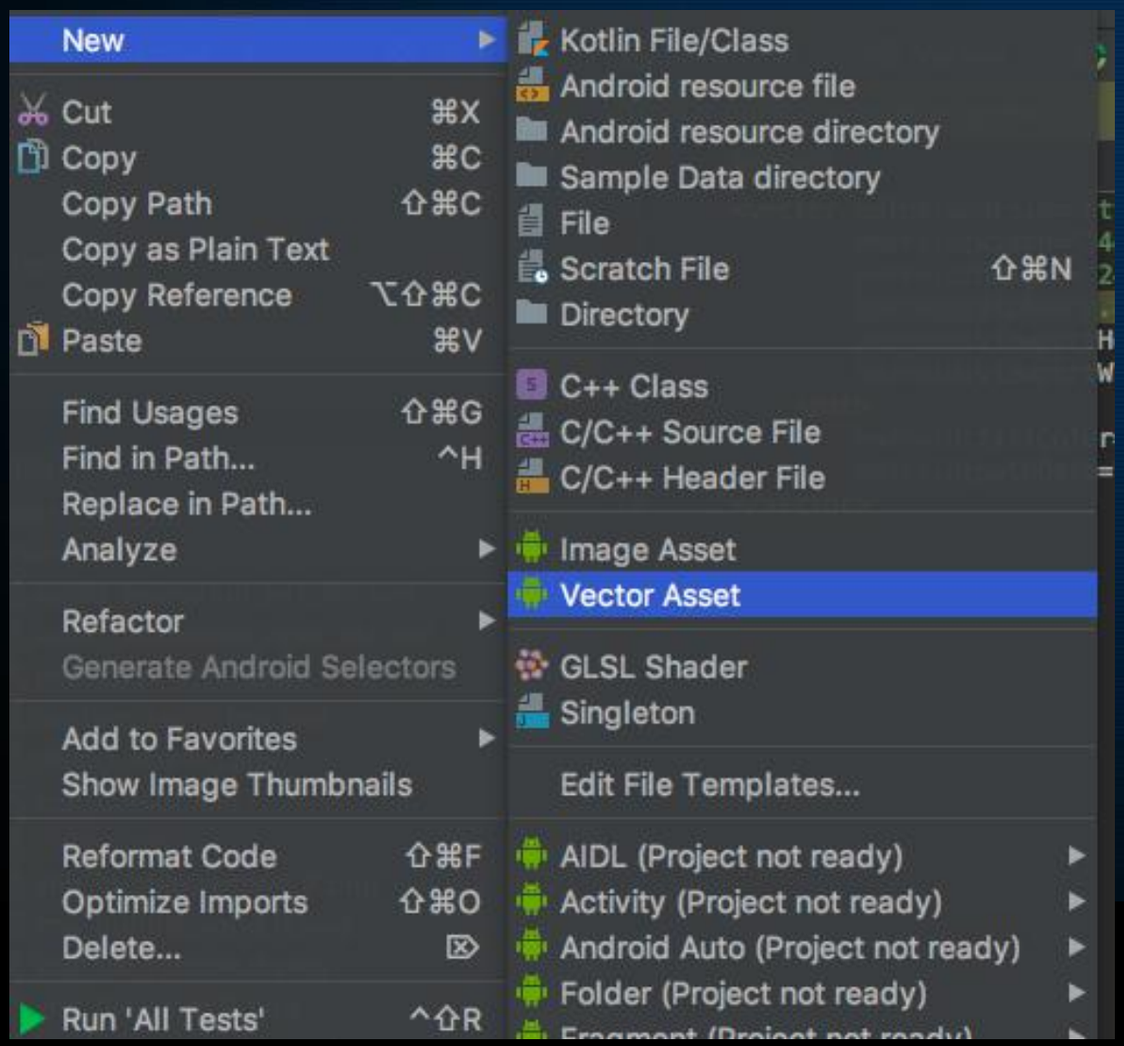
Scalable Vector Graphics，可缩放矢量图。SVG不会像位图一样因为缩放而让图片质量下降。优点在于节约空间与内存。常用于简单小图标。

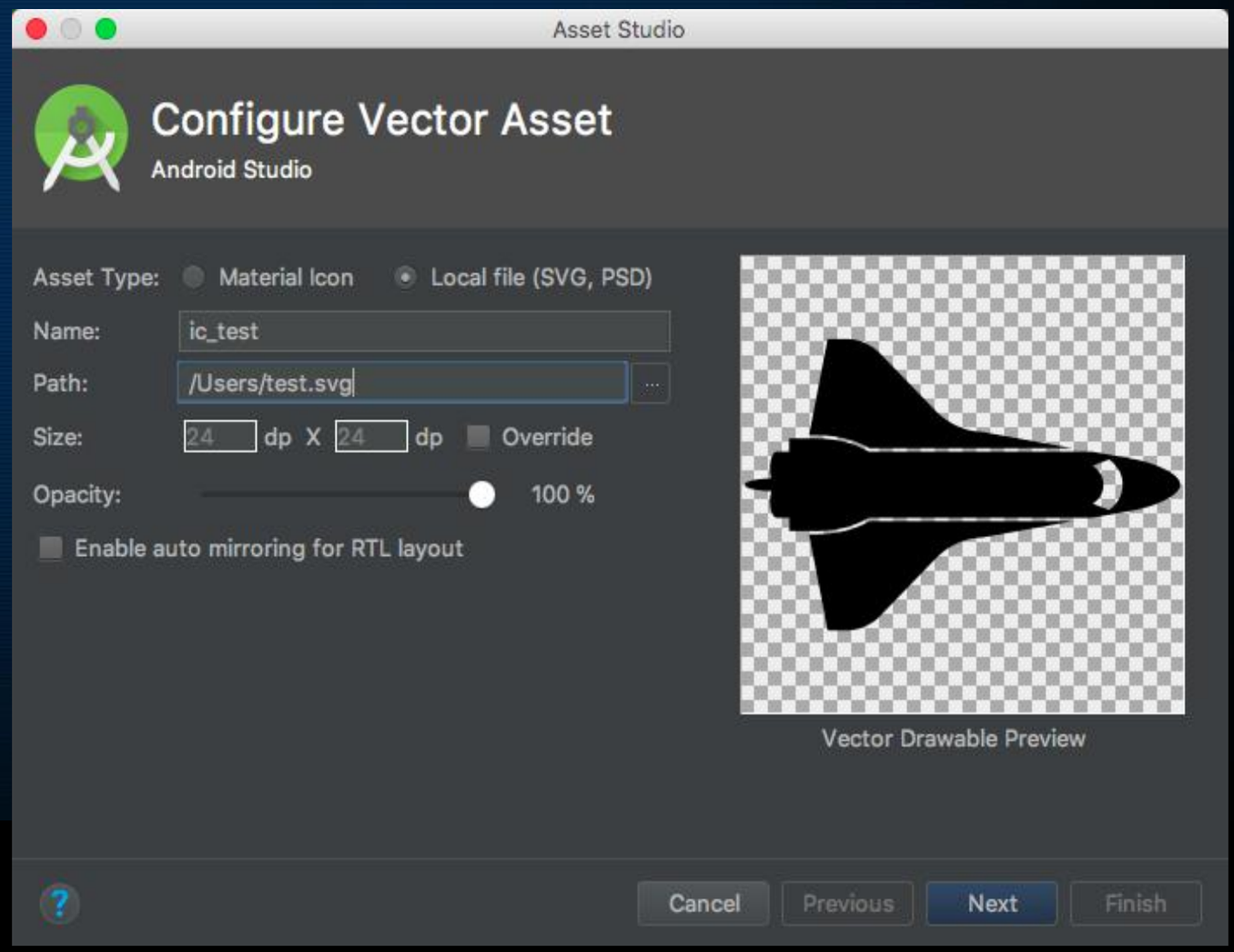
svg是由xml定义的，标准svg根节点为<svg>

在Android中通过 Vector实现对svg的支持，根节点为<vector>。

获得一张svg需要先进行转换才能在android当中使用。

在Android Studio中打开工程，在res目录中点击右键





批量转换svg

as上单个转换svg图像，如果有多个svg需要转换为android的vector，则可以通过第三方工具

https://github.com/MegatronKing/SVG-Android/blob/master/svg-vector-cli/bat/svg2vector-cli-1.0.0.jar

执行转换

```
java -jar svg2vector-cli-1.0.0.jar -d . -o a -h 20 -w 20
```

-d 指定svg文件所在目录
-f 指定当个svg文件
-h 设置转换后svg的高
-w 设置转换后svg的宽
-o 输出android vector图像目录

SVG兼容

https://developer.android.google.cn/studio/write/vector-asset-studio.html

Android 5.0（API 级别 21）及更高版本会提供矢量图支持。如果应用的最低 API 级别低于以上版本

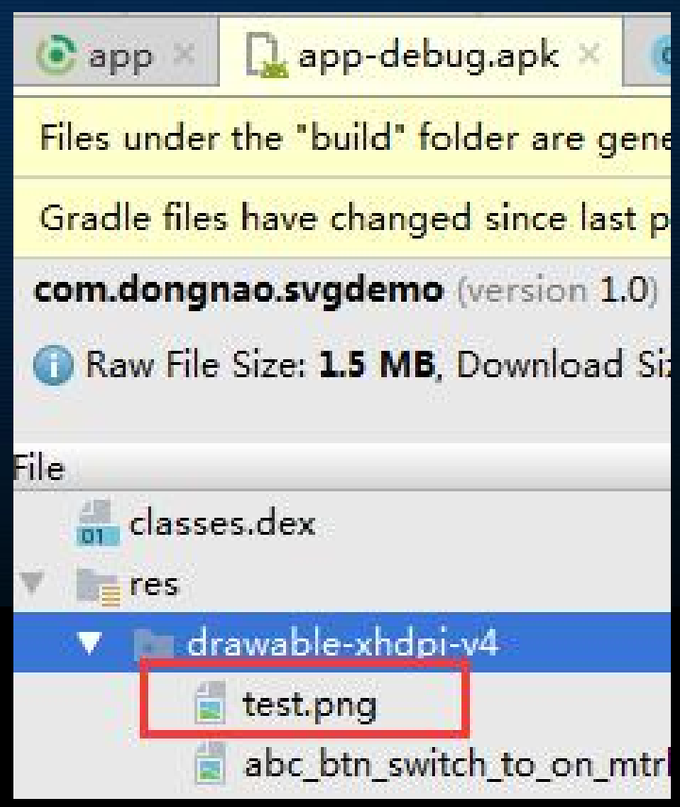
技术	APK 中的可绘制资源	XML VectorDrawable 元素	版本	版本号标志	应用代码
PNG 生成	矢量和光栅	支持的子集	SVG：适用于 Gradle 的 Android 插件 1.5.0 或更高版本 PSD：Android Studio 2.2 或更高版本	默认	支持的各种编码技术
支持库 23.2 或更高版本	矢量	完全支持	适用于 Gradle 的 Android 插件 2.0 或更高版本	需要支持库声明	支持的编码技术子集

png生成

app/build.gradle 中添加：

generatedDensities =['xhdpi','hdpi']添加drawable/test.xml矢量图

```
android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.dongnao.svgdemo"
        minSdkVersion 14
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.suppo
        generatedDensities =['xhdpi']
    }
    buildTypes {
```



SVG兼容

支持库

需要Android 支持库 23.2 或更高版本、适用于 Gradle 的 Android 插件 2.0 或更高版本，且仅使用矢量图。利用支持库中的 VectorDrawableCompat 类，可实现在 Android 2.1（API 级别 7）及更高版本中支持 VectorDrawable。

支持库

此技术需要 Android 支持库 23.2 或更高版本、适用于 Gradle 的 Android 插件 2.0 或更高版本，且仅使用矢量图。利用支持库中的 `VectorDrawableCompat` 类，可实现在 Android 2.1（API 级别 7）及更高版本中支持 `VectorDrawable`。

在使用 Vector Asset Studio 之前，您必须向 `build.gradle` 文件添加一条声明：

```
android {
  defaultConfig {
    vectorDrawables.useSupportLibrary = true
  }
}

dependencies {
  compile 'com.android.support:appcompat-v7:23.2.0'
}
```

您还必须使用与支持库兼容的编码技术，例如对矢量图使用 `app:srcCompat` 属性，而不是 `android:src` 属性。如需了解详细信息，请参阅 Android 支持库 23.2。

```
<ImageView
    app:srcCompat="@drawable/test"
```

Tint着色器

tint能够实现图片变色，利用tint显示不同颜色的图片，原本需要多张相同图片不同颜色的情况，能够减少apk的体积。

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/tint" />

<ImageView
    android:tint="@color/colorAccent"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/tint" />
```

原图

着色

selector 点击效果

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:clickable="true"
    android:src="@drawable/tint_src_selector"
    android:tint="@color/tint_color_selector" />
```

```
Drawable drawable = getResources().getDrawable(R.drawable.tint);
DrawableCompat.setTint(drawable,getResources().getColor(R.color.colorAccent));
```

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/tint" android:state_pressed="true" />
    <item android:drawable="@drawable/tint" />
</selector>
```

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/colorPrimary" android:state_pressed="true"
    <item android:color="@color/colorAccent" />
</selector>
```

使用仿微信开源项目完成apk的瘦身

第一步：启用webp转换插件(jcenter已通过)

classpath 'com.dongnao.optimizer:optimizer-picture:2.0.0'

apply plugin: 'com.dongnao.optimizer'

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.1'
        classpath 'com.dongnao.optimizer:optimizer-picture:2.0.0'
        // NOTE: Do not place your application dependencies here; they
        // in the individual module build.gradle files
    }
}
```

```
apply plugin: 'com.dongnao.optimizer'
```

第二步：资源打包配置

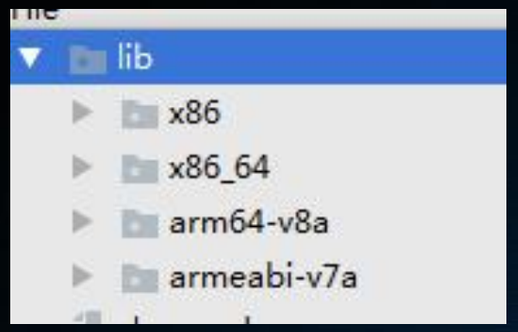
resource.asrc

由于第三方库，如appcompat-v7的引入，库中包含了大量的国际化资源，根据情况通过配置删除。

```
defaultConfig {
    applicationId "com.lqr.wecha
    minSdkVersion 18
    targetSdkVersion 26
    versionCode 1
    versionName "1.0"
    testInstrumentationRunner "a
    只保留英文资源
    resConfigs 'en'
```

ID	Name	default
0x7f070000	abc_action_bar_home_description	Navigate ...
0x7f070001	abc_action_bar_home_description_format	%1$s, %2...
0x7f070002	abc_action_bar_home_subtitle_descripti...	%1$s, %2...
0x7f070003	abc_action_bar_up_description	Navigate ...
0x7f070004	abc_action_menu_overflow_description	More opt...

第三步：动态库打包配置



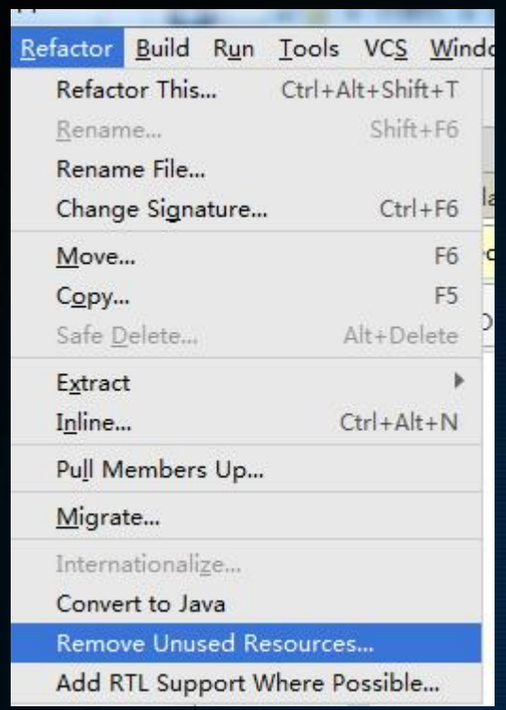
如果项目中包含第三方SDK或者自己使用了ndk，如果不进行配置会打包全cpu架构的动态库进入apk。对于真机，只需要保留一个armeabi(armeabi-v7a)就可以了。

所以可修改配置

```
ndk {
    //设置支持的SO库架构
    //原配置
      abiFilters "armeabi-v7a", "x86", "arm64-v8a", "x86_64"
    //使用模拟器
    abiFilters "x86"
}
```

第四步：移除无用资源

1、一键移除,如果出现使用动态id使用资源会出现问题(不建议)



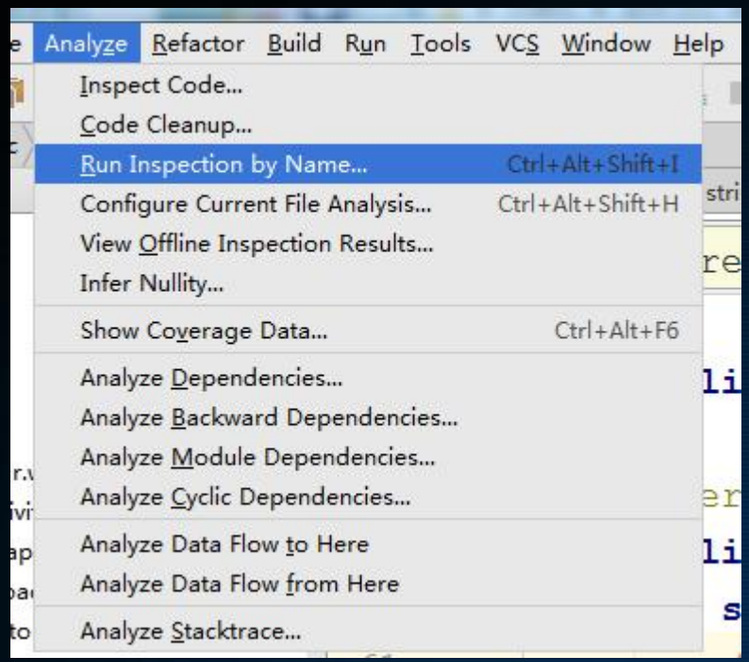
动态获取资源id,未直接使用R.xx.xx ， 则这个id代表的资源会被认为没有使用过(类似不能混淆反射类)

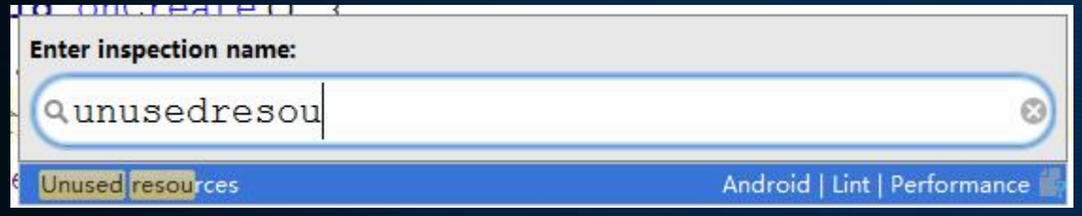
动态获取资源id:

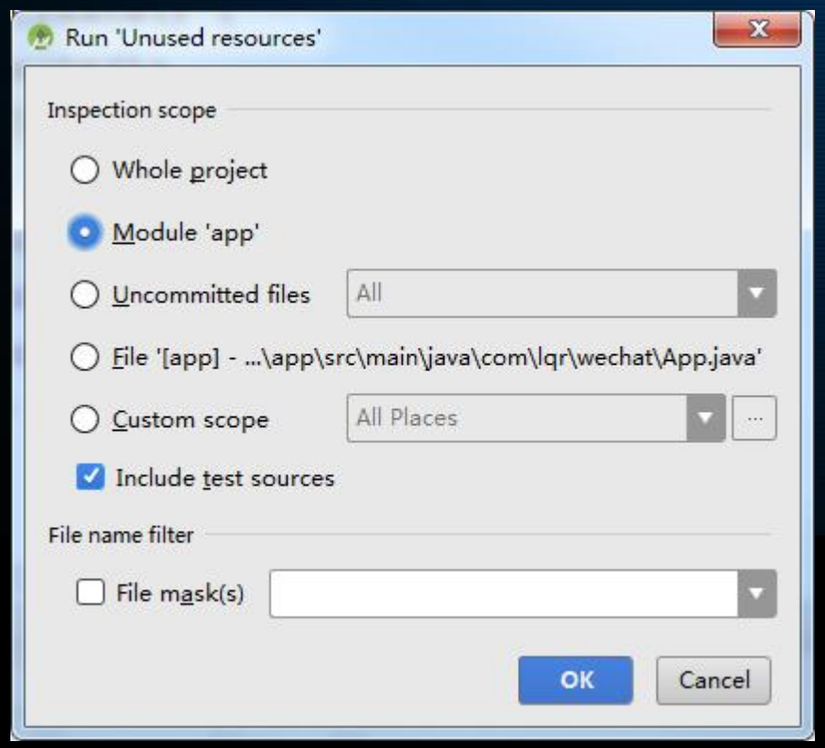
```
//获得字符串 app_name的id
int identifier = getResources().getIdentifier( name: "app_name",  defType: "string", getPackageName());
String name = getResources().getString(identifier);
```

第四步：移除无用资源

2、使用Lint检查

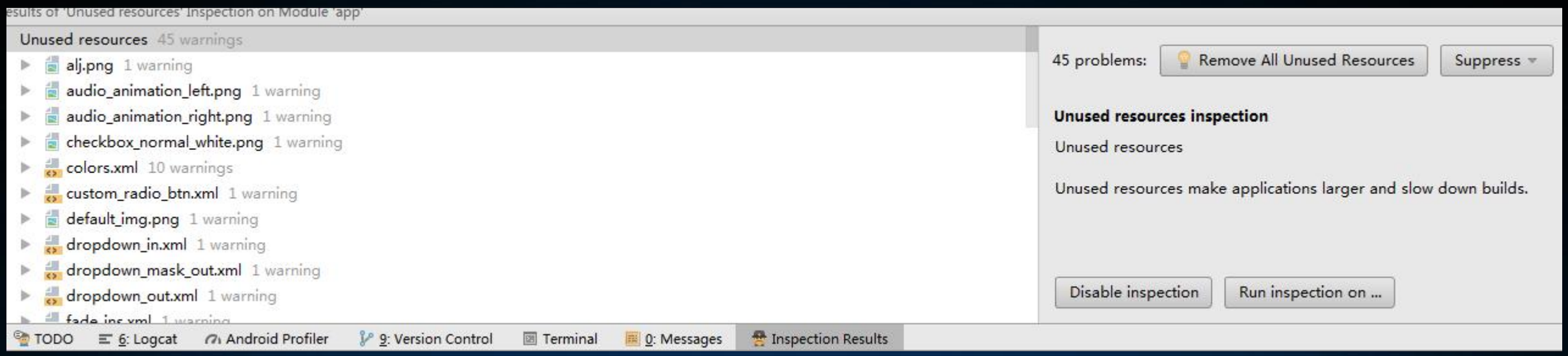






第四步：移除无用资源

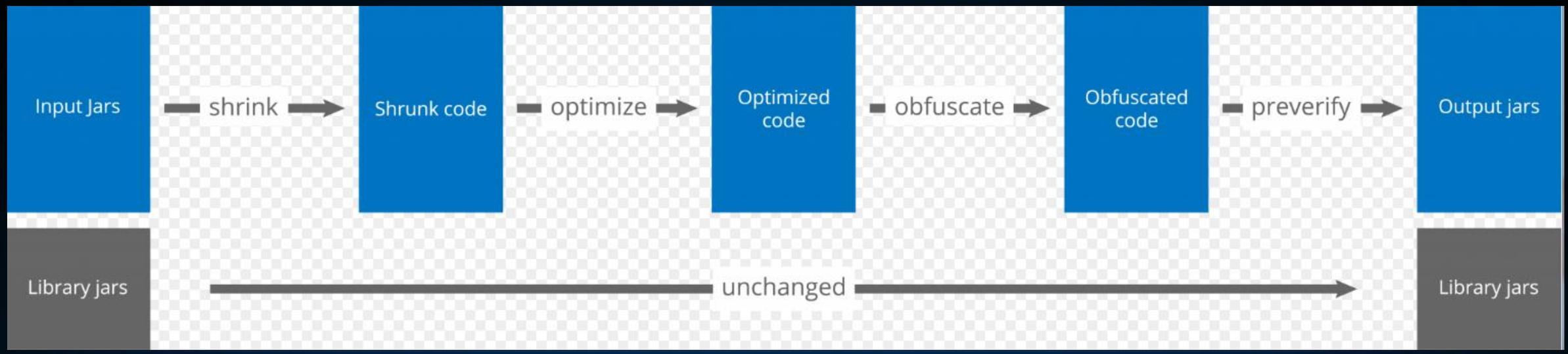
2、使用Lint检查



检查出的无用资源，自行进行排查手动删除

第五步：开启Proguard

进行压缩（Shrink），优化（Optimize），混淆（Obfuscate），预检（Preveirfy）。



1. 压缩（Shrink）:在压缩处理这一步中，用于检测和删除没有使用的类，字段，方法和属性。

2. 优化（Optimize）:在优化处理这一步中，对字节码进行优化，并且移除无用指令。

3. 混淆（Obfuscate）:在混淆处理这一步中，使用a,b,c等无意义的名称，对类，字段和方法进行重命名。

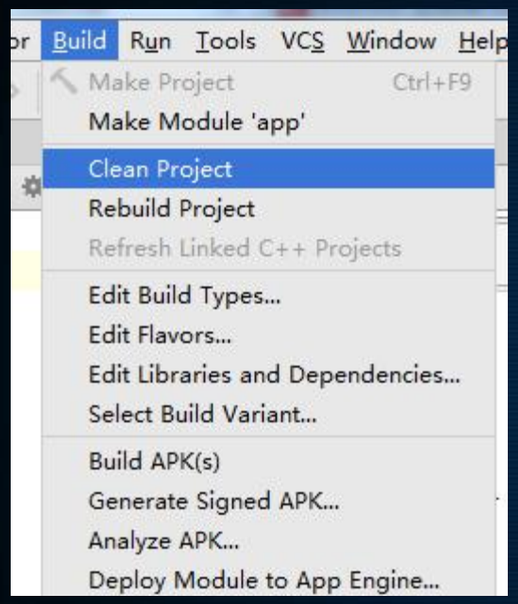
4. 预检（Preveirfy）:在预检这一步中，主要是在Java平台上对处理后的代码进行预检。

第五步：开启代码压缩

仿微信使用较低版本的android gradle插件

```
Exception while processing task java.io.FileNotFoundException:
D:\lance\vip\LQRWeChat\app\build\intermediates\proguard-rules\debug\aapt_rules.txt
(系统找不到指定的路径。)
Execution failed for task ':app:transformClassesAndResourcesWithProguardForDebug'.
> Job failed, see logs for details
```

先clear 再make



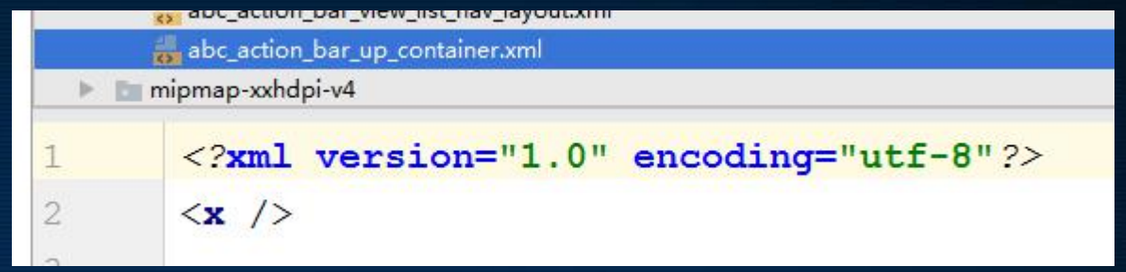

```
//开启debug的混淆
debug {
    minifyEnabled true
    proguardFiles getDefaultProguar
}
```

第六步：启用资源缩减

移除app中未使用的资源，包括代码库中未使用的资源。
只与代码压缩协同工作，未使用的代码被移除，任何不再被引用的资源也会被移除。

```
//开启debug的混淆
debug {
    shrinkResources true
    minifyEnabled true
    proguardFiles getDefaultPr
}
```

未使用的support包中的layout



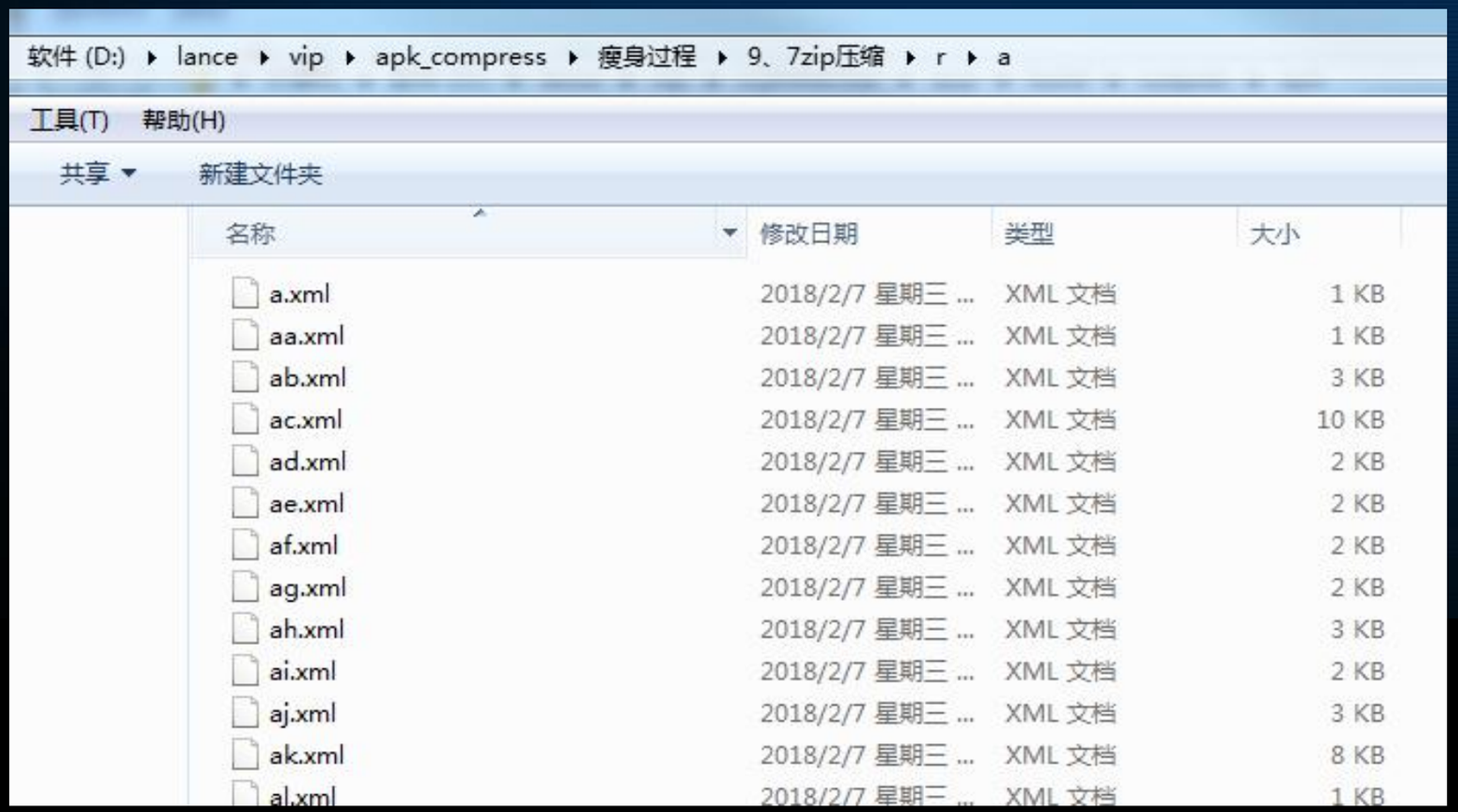
压缩配置 res/raw/keep.xml

https://developer.android.google.cn/studio/build/shrink-code.html#shrink-resources

第七步：资源混淆与7zip压缩

资源混淆配合7zip压缩　减小apk大小，增加破解难度(微信)

效果：



资源混淆

构建流程：

https://developer.android.google.cn/studio/build/index.html?hl=zh-cn

R文件

```
public final class R {
    public static final class anim {
        public static final int abc_fade_in=0x7f050000;
        public static final int abc_fade_out=0x7f050001;
        public static final int abc_grow_fade_in_from_bottom=0x7
        public static final int abc_popup_enter=0x7f050003;
        public static final int abc_popup_exit=0x7f050004;
```

每个资源类型都有对应的 R子类（例如，R.anim 对应于所有动画资源），而该类型的每个资源都有对应的静态整型数（例如，R.anim.abc_fade_in）。这个整型数就是可用来检索资源的资源 ID。

格式为：0xpptteeee（p代表的是package，t代表的是type，e代表的是entry）

Package ID 包ID。系统为0x01，应用程序资源为0x7f。

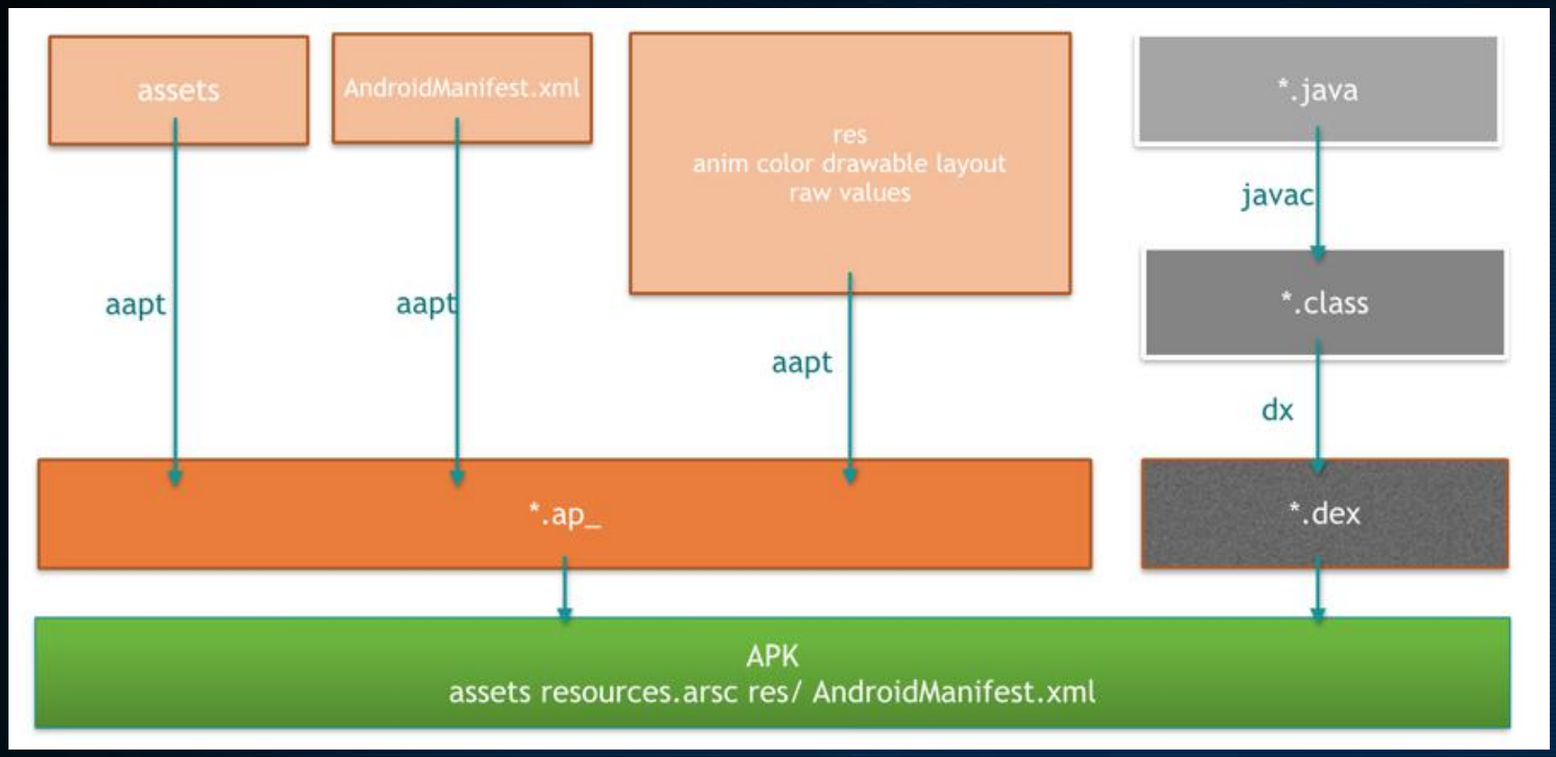
Type ID 资源的类型ID。资源的类型有animator、anim、color等等，每一种都会被赋予一个ID。

Entry ID 资源在其所属的资源类型中所出现的次序。

AAPT

Android Asset Packaging Tool

apk中的manifest、bitmap xml 、layout xml等等这些资源全是二进制的。由aapt进行编译后的资源



1、把"assets"和"res/raw"目录下的所有资源进行打包（根据不同的文件后缀选择压缩或不压缩），而"res/"目录下的其他资源进行编译或者其他处理（具体处理方式视文件后缀不同而不同，例如：".xml"会编译成二进制文件）；

2、对除了assets资源之外所有的资源赋予一个资源ID常量，并且会生成一个资源索引表resources.arsc；

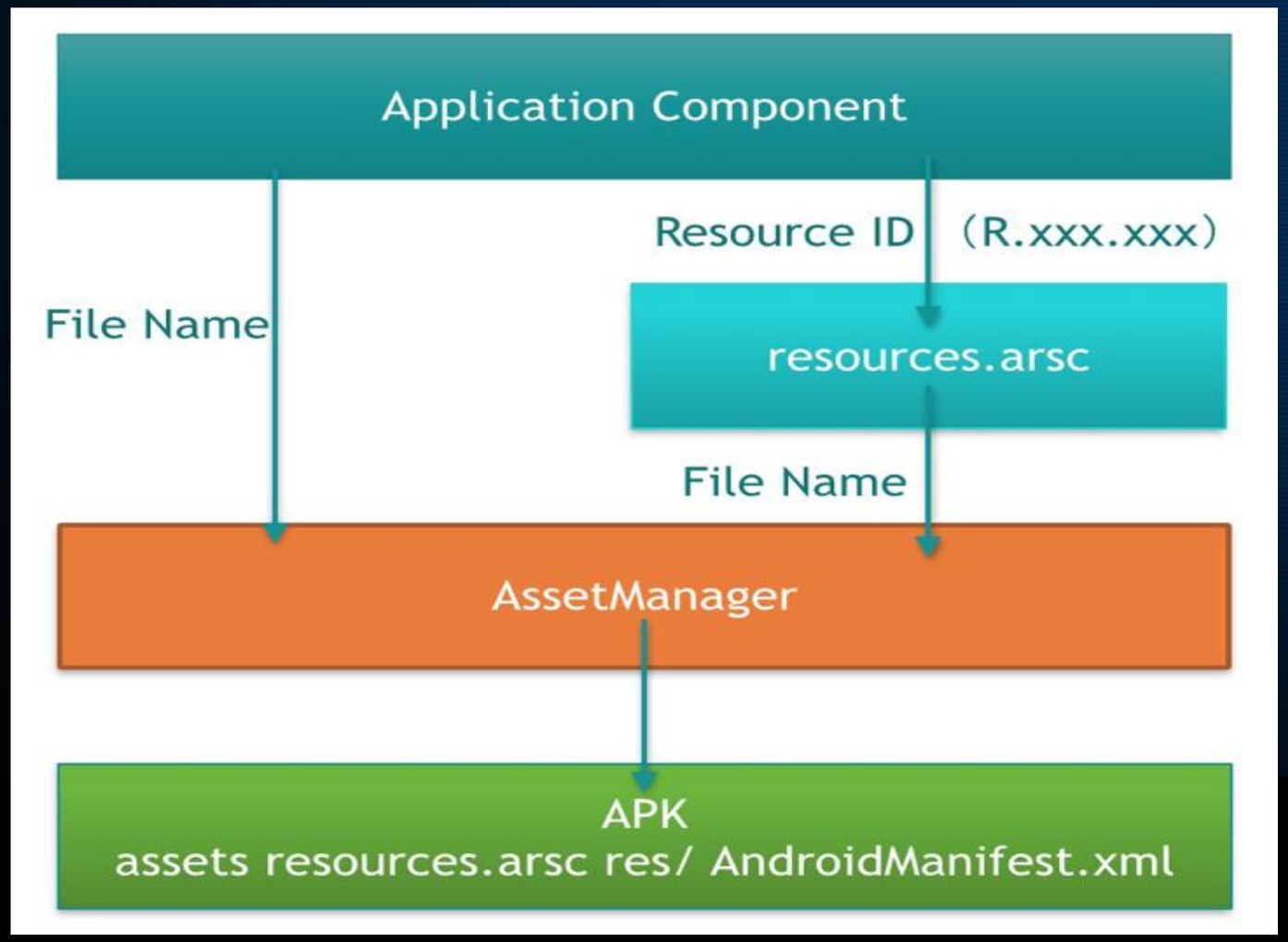
3、编译AndroidManifest.xml成二进制的XML文件；

resource.arsc

提供资源ID到资源文件路径的映射关系，如 R.layout.activity_main（ID:0x7f030000）到res/layout/activity_main.xml的映射关系，应用开发过程中所使用的资源ID使用0x7f030000查找资源。

android通过AssetManager和Resources获得一个图片或者xml等资源。

其中，Resources类可以根据ID来查找资源，而AssetManager类根据文件名来查找资源。Resources类先根据ID来找到资源文件名称，然后再将该文件名称交给AssetManager类通过查找arsc文件(资源映射文件)来打开对应的文件的。



resource.arsc

混淆就是修改映射表(arsc文件)

通过arsc文件格式，混淆步骤为：
1、解析arsc文件(主要为全局与资源名字符串池)
2、修改字符串池中字符串(以无意义的a/b替换)
3、修改apk中res目录资源文件名
4、打包(7zip)、对齐、签名

解析参考资料：
https://github.com/google/android-arscblamer
格式参考资料：
http://androidxref.com/6.0.1_r10/xref/frameworks/base/include/androidfw/ResourceTypes.h

运行工程需要配置 7z、apksigner、zipalign环境变量

混淆工程扩展

白名单配置
mapping文件生成
.....

进一步减小apk大小：
v1/v2签名？
https://source.android.com/security/apksigning/
全版本允许：　只使用 v1 方案
全版本允许：　同时使用 v1 和 v2 方案
>=7.0版本：　只使用 v2 方案

使用v1签名会在 META-INF/生成签名校验文件
使用v2签名会向zip文件按照格式插入字节数据
同时使用两种签名会增加两份数据(文件+数据),提供配置v1/v2签名开关
v1签名：
　　jarsigner.jar
https://docs.oracle.com/javase/6/docs/technotes/tools/windows/jarsigner.html