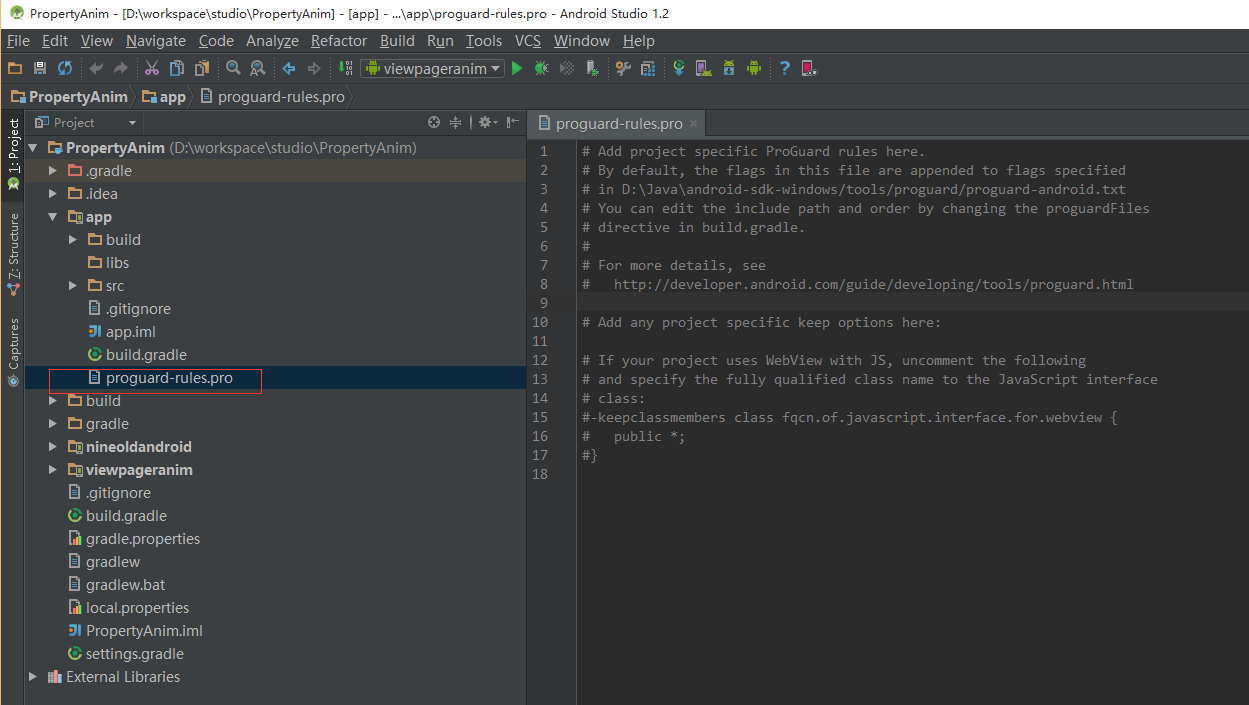
修改自已工程下的proguard-project.txt文件

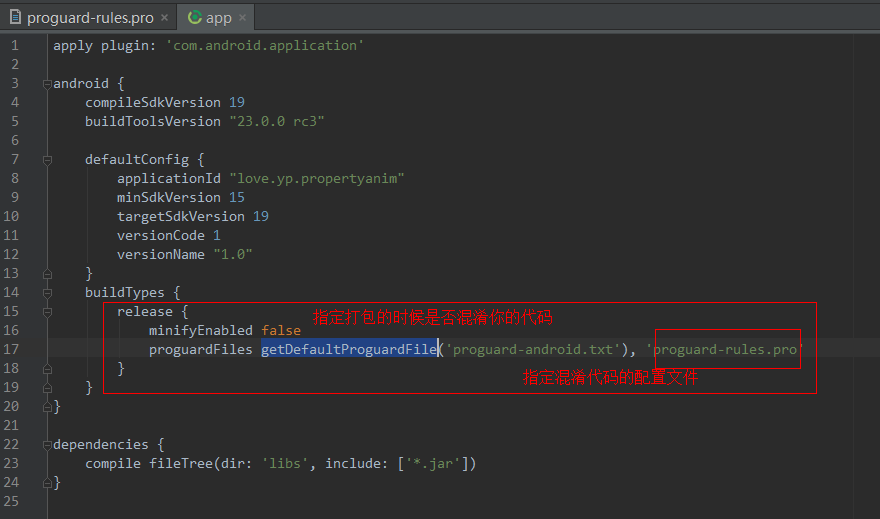
参考：${sdk.dir}/tools/proguard/proguard-android.txt

代码混淆

首先，在你的工程目录下，找到proguard-rules.pro文件，它就是你要进行编写混淆配置的文件：



光编写该文件还不够哦，你还需要在你module的build.gradle文件中引用该混淆文件：



好了，知道在哪配置混淆文件后，下面开始讲讲如何配置混淆：

混淆文件 proguard-rules.pro 参数详解：

**[plain]** [view plain](http://blog.csdn.net/zuiwuyuan/article/details/48552701#) [copy](http://blog.csdn.net/zuiwuyuan/article/details/48552701#)

[print?](http://blog.csdn.net/zuiwuyuan/article/details/48552701#)

1. -optimizationpasses 5                                                           # 指定代码的压缩级别
2. -dontusemixedcaseclassnames                                                     # 是否使用大小写混合
3. -dontskipnonpubliclibraryclasses                                                # 是否混淆第三方jar
4. -dontpreverify                                                                  # 混淆时是否做预校验
5. -verbose                                                                        # 混淆时是否记录日志
6. -optimizations !code/simplification/arithmetic,!field/\*,!class/merging/\*        # 混淆时所采用的算法
8. -keep public class \* extends android.app.Activity                               # 保持哪些类不被混淆
9. -keep public class \* extends android.app.Application                            # 保持哪些类不被混淆
10. -keep public class \* extends android.app.Service                                # 保持哪些类不被混淆
11. -keep public class \* extends android.content.BroadcastReceiver                  # 保持哪些类不被混淆
12. -keep public class \* extends android.content.ContentProvider                    # 保持哪些类不被混淆
13. -keep public class \* extends android.app.backup.BackupAgentHelper               # 保持哪些类不被混淆
14. -keep public class \* extends android.preference.Preference                      # 保持哪些类不被混淆
15. -keep public class com.android.vending.licensing.ILicensingService              # 保持哪些类不被混淆
17. -keepclasseswithmembernames class \* {                                           # 保持 native 方法不被混淆
18. native <methods>;
19. }
21. -keepclasseswithmembers class \* {                                               # 保持自定义控件类不被混淆
22. public <init>(android.content.Context, android.util.AttributeSet);
23. }
25. -keepclasseswithmembers class \* {
26. public <init>(android.content.Context, android.util.AttributeSet, int);     # 保持自定义控件类不被混淆
27. }
29. -keepclassmembers class \* extends android.app.Activity {                        # 保持自定义控件类不被混淆
30. public void \*(android.view.View);
31. }
33. -keepclassmembers enum \* {                                                      # 保持枚举 enum 类不被混淆
34. public static \*\*[] values();
35. public static \*\* valueOf(java.lang.String);
36. }
38. -keep class \* implements android.os.Parcelable {                                # 保持 Parcelable 不被混淆
39. public static final android.os.Parcelable$Creator \*;
40. }
42. -keep class MyClass;                                                            # 保持自己定义的类不被混淆

以上是最基础的配置，几乎每个项目都需要进行这些地方的混淆（或保持不混淆）。

如果你仔细看过上方的注释，就会了解一些基本代码混淆策略了。

只是，这还远远不够，因为你在项目中，总会不可避免的引用第三方的library库或是jar包,那，如果你不能够正确的混淆第三方的资源，可能会导致你的应用无法使用。（我就因为忘了配置Gson相关的混淆，导致页面一直没有数据显示）。

贴出关于第三方的混淆部分：

**[plain]** [view plain](http://blog.csdn.net/zuiwuyuan/article/details/48552701#) [copy](http://blog.csdn.net/zuiwuyuan/article/details/48552701#)

[print?](http://blog.csdn.net/zuiwuyuan/article/details/48552701#)

1. #如果有引用v4包可以添加下面这行
2. -keep class android.support.v4.\*\* { \*; }
3. -keep public class \* extends android.support.v4.\*\*
4. -keep public class \* extends android.app.Fragment

7. #如果引用了v4或者v7包，可以忽略警告，因为用不到android.support
8. -dontwarn android.support.\*\*


12. #保持自定义组件不被混淆
13. -keep public class \* extends android.view.View {
14. public <init>(android.content.Context);
15. public <init>(android.content.Context, android.util.AttributeSet);
16. public <init>(android.content.Context, android.util.AttributeSet, int);
17. public void set\*(...);
18. }

21. #保持 Serializable 不被混淆
22. -keepnames class \* implements java.io.Serializable
24. #保持 Serializable 不被混淆并且enum 类也不被混淆
25. -keepclassmembers class \* implements java.io.Serializable {
26. static final long serialVersionUID;
27. private static final java.io.ObjectStreamField[] serialPersistentFields;
28. private void writeObject(java.io.ObjectOutputStream);
29. private void readObject(java.io.ObjectInputStream);
30. java.lang.Object writeReplace();
31. java.lang.Object readResolve();
32. }
34. #保持枚举 enum 类不被混淆 如果混淆报错，建议直接使用上面的 -keepclassmembers class \* implements java.io.Serializable即可
35. -keepclassmembers enum \* {
36. public static \*\*[] values();
37. public static \*\* valueOf(java.lang.String);
38. }
40. -keepclassmembers class \* {
41. public void \*ButtonClicked(android.view.View);
42. }
44. #不混淆资源类
45. #-keepclassmembers class \*\*.R$\* {
46. #    public static <fields>;
47. #}

50. #xUtils(保持注解，及使用注解的Activity不被混淆，不然会影响Activity中你使用注解相关的代码无法使用)
51. -keep class \* extends java.lang.annotation.Annotation {\*;}
52. -keep class com.otb.designerassist.activity.\*\* {\*;}

55. #自己项目特殊处理代码（这些地方我使用了Gson类库和注解，所以不希望被混淆，以免影响程序）
56. -keep class com.otb.designerassist.entity.\*\* {\*;}
57. -keep class com.otb.designerassist.http.rspdata.\*\* {\*;}
58. -keep class com.otb.designerassist.service.\*\* {\*;}

61. ##混淆保护自己项目的部分代码以及引用的第三方jar包library（想混淆去掉"#"）
62. #-libraryjars libs/umeng-analytics-v5.2.4.jar
63. #-libraryjars libs/alipaysecsdk.jar
64. #-libraryjars libs/alipayutdid.jar
65. #-libraryjars libs/weibosdkcore.jar

68. # 以libaray的形式引用的图片加载框架,不想混淆（注意，此处不是jar包形式，想混淆去掉"#"）
69. #-keep class com.nostra13.universalimageloader.\*\* { \*; }

72. ###-------- Gson 相关的混淆配置--------
73. -keepattributes Signature
74. -keepattributes \*Annotation\*
75. -keep class sun.misc.Unsafe { \*; }



80. ###-------- pulltorefresh 相关的混淆配置---------
81. -dontwarn com.handmark.pulltorefresh.library.\*\*
82. -keep class com.handmark.pulltorefresh.library.\*\* { \*;}
83. -dontwarn com.handmark.pulltorefresh.library.extras.\*\*
84. -keep class com.handmark.pulltorefresh.library.extras.\*\* { \*;}
85. -dontwarn com.handmark.pulltorefresh.library.internal.\*\*
86. -keep class com.handmark.pulltorefresh.library.internal.\*\* { \*;}

89. ###---------  reservoir 相关的混淆配置-------
90. -keep class com.anupcowkur.reservoir.\*\* { \*;}

93. ###-------- ShareSDK 相关的混淆配置---------
94. -keep class cn.sharesdk.\*\* { \*; }
95. -keep class com.sina.sso.\*\* { \*; }

98. ###--------------umeng 相关的混淆配置-----------
99. -keep class com.umeng.\*\* { \*; }
100. -keep class com.umeng.analytics.\*\* { \*; }
101. -keep class com.umeng.common.\*\* { \*; }
102. -keep class com.umeng.newxp.\*\* { \*; }

105. ###-----------MPAndroidChart图库相关的混淆配置------------
106. -keep class com.github.mikephil.charting.\*\* { \*; }

以上的配置，即是对一个项目的混淆配置了，相对比较完整，大家可以依葫芦画瓢，写更多的配置，对于一些第三方项目的使用，一般官方会给出如何配置混淆，大家需要小心，别忘了配置。

好啦，如果你已经写好自己的混淆配置文件，不要忘了在build.gradle文件中再次配置下，打开混淆文件：

**[plain]** [view plain](http://blog.csdn.net/zuiwuyuan/article/details/48552701#) [copy](http://blog.csdn.net/zuiwuyuan/article/details/48552701#)

[print?](http://blog.csdn.net/zuiwuyuan/article/details/48552701#)

1. buildTypes {
2. debug {
3. // 显示Log
4. buildConfigField "boolean", "LOG\_DEBUG", "true"

7. versionNameSuffix "-debug"
8. minifyEnabled false
9. zipAlignEnabled false
10. shrinkResources false
11. signingConfig signingConfigs.assist
12. }

15. release {
16. // 不显示Log
17. buildConfigField "boolean", "LOG\_DEBUG", "false"
19. //混淆
20. minifyEnabled true
22. //Zipalign优化
23. zipAlignEnabled true

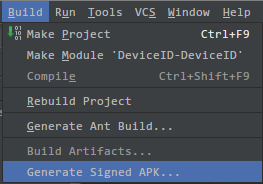
26. // 移除无用的resource文件
27. shrinkResources true
28. //加载默认混淆配置文件
29. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
30. //签名
31. signingConfig signingConfigs.<span style="font-family: Arial, Helvetica, sans-serif;">assist</span>
33. }
34. }

release节点下，minifyEnabled设置为true。

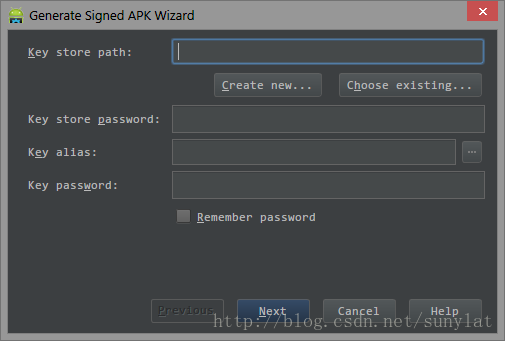
导出APK文件

  学会了如何混淆你的项目代码之后，接下来，我们看下，如何使用Android Studio导出APK文件吧。

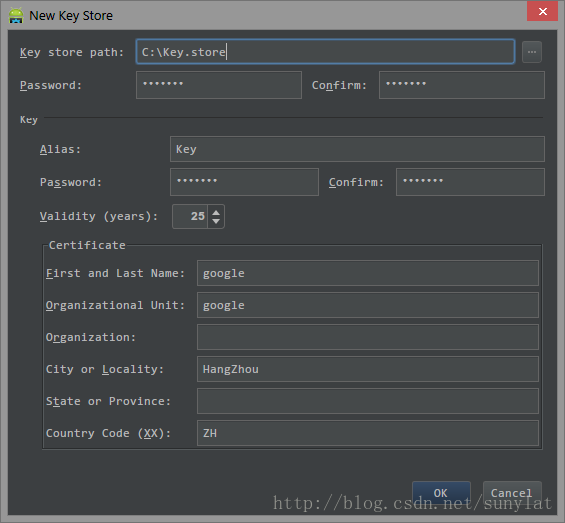
（1）Android Studio菜单Build->Generate Signed APK



（2）弹出签名选择、创建窗口



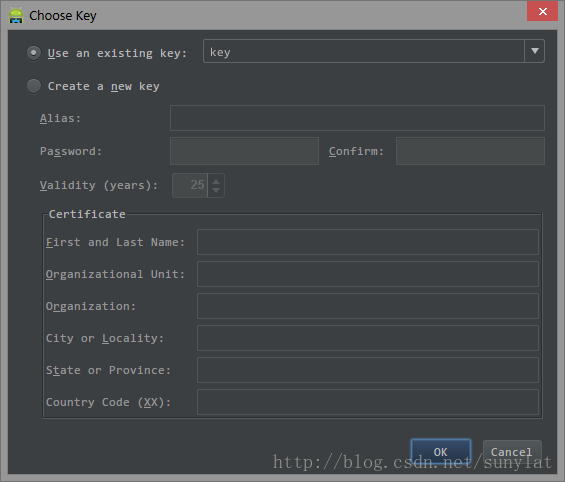
（3）创建密钥库及密钥，创建后会自动选择刚创建的密钥库和密钥（已拥有密钥库跳过）      点击“Create new...”按钮创建密钥库



Key store path：密钥库文件的地址  Password/Confirm：密钥库的密码  Alias：密钥名称  Password/Confirm：密钥密码  Validity(years)：密钥有效时间  First and Last Name：密钥颁发者姓名  Organizational Unit：密钥颁发组织  City or Locality：城市  Country Code(XX)：国家

（4）选择已存在密钥库及密钥（在（3）中创建密钥库后跳过此步骤）

    点击“Choose existing...”按钮找到密钥库文件      Key store password输入已选择的密钥库文件的密码      点击Key alias后的“...”按钮，选择或者创建一个密钥



（5）点击“Next”按钮，选择保存路径后，点击“Finish”按钮完成

