**Android 混淆**

使用Proguard，Proguard是混淆代码的一个开源项目。

原理

Java 是一种跨平台的、解释型语言，Java 源代码编译成中间”字节码”存储于 class 文件中。由于跨平台的需要，Java 字节码中包括了很多源代码信息，如变量名、方法名，并且通过这些名称来访问变量和方法，这些符号带有许多语义信息，很容易被反编译成 Java 源代码。为了防止这种现象，我们可以使用 Java 混淆器对 Java 字节码进行混淆。

混淆就是对发布出去的程序进行重新组织和处理，使得处理后的代码与处理前代码完成相同的功能，而混淆后的代码很难被反编译，即使反编译成功也很难得出程序的真正语义。被混淆过的程序代码，仍然遵照原来的档案格式和指令集，执行结果也与混淆前一样，只是混淆器将代码中的所有变量、函数、类的名称变为简短的英文字母代号，在缺乏相应的函数名和程序注释的况下，即使被反编译，也将难以阅读。同时混淆是不可逆的，在混淆的过程中一些不影响正常运行的信息将永久丢失，这些信息的丢失使程序变得更加难以理解。

混淆器的作用不仅仅是保护代码，它也有精简编译后程序大小的作用。由于以上介绍的缩短变量和函数名以及丢失部分信息的原因， 编译后 jar 文件体积大约能减少25%，这对当前费用较贵的无线网络传输是有一定意义的。

混淆的具体使用

1. 模块（Module）下的 build.gradle 的配置

android{

buildTypes{

release {

// 是否进行混淆

minifyEnabled false

// 混淆文件的位置，其中'proguard-android.txt'为sdk默认的混淆配置，

//'proguard-rules.pro' 是该模块下的混淆配置

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

proguard-android.txt为sdk默认的混淆配置，proguard-rules.pro是在默认配置的基础上针对本模块做出的针对性混淆处理。

注：proguard-android.txt的位置位于android-sdk/tools/proguard/proguard-android.txt

混淆的基本语法

这个语法的作用是定义出不需要混淆的源代码，那么编译时会自动将未定义的部分全都混淆。至于为什么要保留类名或方法名，主要有三个原因：

1. 让C/C++程序可以通过jni使用对应的java方法。
2. 四大组件在AndroidManifest.xml里面注册了，所以要保留。
3. R文件混淆会导致引用错误。
4. **保留类名**
5. **保留方法名**
6. **保留类名和方法名**

|  |  |  |
| --- | --- | --- |
| 保留 | 防止被移除或被重命名 | 防止被重命名 |
| 类和类成员 | -keep | -keepnames |
| 仅类成员 | -keepclassmembers | -keepclassmemebernames |
| 如果拥有某成员，保留类和类成员 | -keepclasseswithmembers | -keepclasseswithmembernames |

#保留位于 View 类中的 get 和 set 方法

-keepclassmembers public class \* extends android.view.View{

void set\*(\*\*\*);

\*\*\* get\*();

}

#保留在Activity中以View为参数的方法不变

-keepclassmembers class \* extends android.app.Activity{

public void \*(android.view.View);

}

#保留实现了 Parcelable的类名不变

-keep class \* implements android.os.Parcelable{

public static final android.os.Parcelable$Creator \*;

}

#保留 R$\*类中静态成员的变量名

-keepclassmembers class \*\*.R$\*{

public static <fields>;

}

去除日志信息

通过配置 proguard，将 android.util.Log置为无效代码，则可以去除 apk中打印日志的代码。

1. 将build.gradle添加'proguard-android-optimize.txt'文件，该文件中默认打开了优化开关。

android{

buildTypes{

release {

// 是否进行混淆

minifyEnabled false

// 混淆文件的位置，其中'proguard-android-optimize.txt'为sdk默认的混淆配置，

//'proguard-rules.pro' 是该模块下的混淆配置

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

}

1. 配置语法

-assumenosideeffects class android.util.Log{

public static boolean isLoggable(java.lang.String,int);

public static int v(...);

public static int i(...);

public static int w(...);

public static int d(...);

public static int e(...);

}

-assumenosideeffects class com.example.Log.Logger{

public static int v(...);

public static int i(...);

public static int w(...);

public static int d(...);

public static int e(...);

}

**更详细的语法**

-include {filename} 从给定的文件中读取配置参数

-basedirectory {directoryname} 指定基础目录为以后相对的档案名称

-injars {class\_path} 指定要处理的应用程序jar,war,ear和目录

-outjars {class\_path} 指定处理完后要输出的jar,war,ear和目录的名称

-libraryjars {classpath} 指定要处理的应用程序jar,war,ear和目录所需要的程序库文件

-dontskipnonpubliclibraryclasses 指定不去忽略非公共的库类。

-dontskipnonpubliclibraryclassmembers 指定不去忽略包可见的库类的成员。

保留选项

-keep {Modifier} {class\_specification} 保护指定的类文件和类的成员

-keepclassmembers {modifier} {class\_specification} 保护指定类的成员，如果此类受到保护他们会保护的更好

-keepclasseswithmembers {class\_specification} 保护指定的类和类的成员，但条件是所有指定的类和类成员是要存在。

-keepnames {class\_specification} 保护指定的类和类的成员的名称（如果他们不会压缩步骤中删除）

-keepclassmembernames {class\_specification} 保护指定的类的成员的名称（如果他们不会压缩步骤中删除）

-keepclasseswithmembernames {class\_specification} 保护指定的类和类的成员的名称，如果所有指定的类成员出席（在压缩步骤之后）

-printseeds {filename} 列出类和类的成员-keep选项的清单，标准输出到给定的文件

压缩

-dontshrink 不压缩输入的类文件

-printusage {filename}

-whyareyoukeeping {class\_specification}

优化

-dontoptimize 不优化输入的类文件

-assumenosideeffects {class\_specification} 优化时假设指定的方法，没有任何副作用

-allowaccessmodification 优化时允许访问并修改有修饰符的类和类的成员

混淆

-dontobfuscate 不混淆输入的类文件

-printmapping {filename}

-applymapping {filename} 重用映射增加混淆

-obfuscationdictionary {filename} 使用给定文件中的关键字作为要混淆方法的名称

-overloadaggressively 混淆时应用侵入式重载

-useuniqueclassmembernames 确定统一的混淆类的成员名称来增加混淆

-flattenpackagehierarchy {package\_name} 重新包装所有重命名的包并放在给定的单一包中

-repackageclass {package\_name} 重新包装所有重命名的类文件中放在给定的单一包中

-dontusemixedcaseclassnames 混淆时不会产生形形色色的类名

-keepattributes {attribute\_name,...} 保护给定的可选属性，例如LineNumberTable, LocalVariableTable, SourceFile, Deprecated, Synthetic, Signature, and

InnerClasses.

-renamesourcefileattribute {string} 设置源文件中给定的字符串常量

**Demo1**

-ignorewarnings # 忽略警告，避免打包时某些警告出现

-optimizationpasses 5 # 指定代码的压缩级别，表示proguard对代码进行迭代优化的次数，Android一般为5

-dontusemixedcaseclassnames # 是否使用大小写混合

-dontskipnonpubliclibraryclasses # 是否混淆第三方jar

-dontpreverify # 混淆时是否做预校验

-verbose # 混淆时是否记录日志

-optimizations !code/simplification/arithmetic,!field/\*,!class/merging/\* # 混淆时所采用的算法

-libraryjars libs/treecore.jar

-dontwarn android.support.v4.\*\* #缺省proguard 会检查每一个引用是否正确，但是第三方库里面往往有些不会用到的类，没有正确引用。如果不配置的话，系统就会报错。

-dontwarn android.os.\*\*

-keep class android.support.v4.\*\* { \*; } # 保持哪些类不被混淆

-keep class com.baidu.\*\* { \*; }

-keep class vi.com.gdi.bgl.android.\*\*{\*;}

-keep class android.os.\*\*{\*;}

-keep interface android.support.v4.app.\*\* { \*; }

-keep public class \* extends android.support.v4.\*\*

-keep public class \* extends android.app.Fragment

-keep public class \* extends android.app.Activity

-keep public class \* extends android.app.Application

-keep public class \* extends android.app.Service

-keep public class \* extends android.content.BroadcastReceiver

-keep public class \* extends android.content.ContentProvider

-keep public class \* extends android.support.v4.widget

-keep public class \* extends com.sqlcrypt.database

-keep public class \* extends com.sqlcrypt.database.sqlite

-keep public class \* extends com.treecore.\*\*

-keep public class \* extends de.greenrobot.dao.\*\*

-keepclasseswithmembernames class \* { # 保持 native 方法不被混淆

native <methods>;

}

-keepclasseswithmembers class \* { # 保持自定义控件类不被混淆

public <init>(android.content.Context, android.util.AttributeSet);

}

-keepclasseswithmembers class \* { # 保持自定义控件类不被混淆

public <init>(android.content.Context, android.util.AttributeSet, int);

}

-keepclassmembers class \* extends android.app.Activity { //保持类成员

public void \*(android.view.View);

}

-keepclassmembers enum \* { # 保持枚举 enum 类不被混淆

public static \*\*[] values();

public static \*\* valueOf(java.lang.String);

}

-keep class \* implements android.os.Parcelable { # 保持 Parcelable 不被混淆

public static final android.os.Parcelable$Creator \*;

}

-keep class MyClass; # 保持自己定义的类不被混淆

**Demo2**

#默认的proguard-android.txt已经增加了Annotation、native、view的setget方法、Activity参数为view的方法、Enum枚举、Parcelable、R，此处不再写

#------------------------------------------通用区域----------------------------------------------------

#----------------------基本指令------------------------

#代码混淆压缩比，在0和7之间，默认为5，一般不需要改

-optimizationpasses 5

#混淆时不使用大小写混合，混淆后的类名为小写

-dontusemixedcaseclassnames

#指定不去忽略非公共的库的类

-dontskipnonpubliclibraryclasses

#指定不去忽略非公共的库的类的成员

-dontskipnonpubliclibraryclassmembers

#不做预校验，preverify是proguard的4个步骤之一

#Android不需要preverify，去掉这一步可加快混淆速度

-dontpreverify

#有了verbose这句话，混淆后就会生成映射文件

#包含有类名->混淆后类名的映射关系

#然后使用printmapping指定映射文件的名称

-verbose

-printmapping proguardMapping.txt

#指定混淆时采用的算法，后面的参数是一个过滤器

#这个过滤器是谷歌推荐的算法，一般不改变

-optimizations !code/simplification/cast,!field/\*,!class/merging/\*

#保护代码中的Annotation不被混淆，这在JSON实体映射时非常重要，比如fastJson

-keepattributes \*Annotation\*,InnerClasses

#避免混淆泛型，这在JSON实体映射时非常重要，比如fastJson

-keepattributes Signature

#抛出异常时保留代码行号，在异常分析中可以方便定位

-keepattributes SourceFile,LineNumberTable

#-----------------------全局混淆-----------------------

#除了项目目录，其他都不混淆②

-keep class !com.glela.\*\* {\*;}

-dontwarn \*\*

#---------------------默认保留-------------------------

#基础保留

-keep class \* extends android.app.Activity

-keep class \* extends android.app.Application

-keep class \* extends android.app.Service

-keep class \* extends android.content.BroadcastReceiver

-keep class \* extends android.content.ContentProvider

-keep class \* extends android.app.backup.BackupAgentHelper

-keep class \* extends android.preference.Preference

-keep class \* extends android.view.View {

<init>(...);

}

#序列化

-keepclassmembers class \* implements java.io.Serializable {

static final long serialVersionUID;

private static final java.io.ObjectStreamField[] serialPersistentFields;

private void writeObject(java.io.ObjectOutputStream);

private void readObject(java.io.ObjectInputStream);

java.lang.Object writeReplace();

java.lang.Object readResolve();

}

#EventBus的注解

-keepclassmembers class \* {

@org.greenrobot.eventbus.Subscribe <methods>;

}

#WebView

-keepclassmembers class \* extends android.webkit.WebView {\*;}

-keepclassmembers class \* extends android.webkit.WebViewClient {\*;}

-keepclassmembers class \* extends android.webkit.WebChromeClient {\*;}

-keepclassmembers class \* {

@android.webkit.JavascriptInterface <methods>;

}

#-------------------------------------------项目定义区-------------------------------------------------

#com.glela.\*是为了适配其他项目③

#bean

-keep class com.glela.\*.bean\*\* {\*;}

#eventbus传递的类

-keep class com.glela.\*.eventbus.\*\* {\*;}

#zxing相关，目前没动过

-keep class com.glela.\*.zxingjava.\*\* {\*;}

#反射相关

**保持第三方jar**

#在proguard.cfg中加入不混淆的jar包 和 keep不混淆的class：   
-libraryjars libs/HcPaysdk.jar   
-keep class com.hc.pay.\*\*{\*;}

**具体操作详解**

压缩(Shrinking)：默认开启，用以减小应用体积，移除未被使用的类和成员，并且会在优化动作执行之后再次执行（因为优化后可能会再次暴露一些未被使用的类和成员）。

-dontshrink #关闭压缩

优化（Optimization）：默认开启，在字节码级别执行优化，让应用运行的更快。

-dontoptimize #关闭优化

-optimizationpasses n #表示proguard对代码进行迭代优化的次数，Android一般为5

混淆（Obfuscation）：默认开启，增大反编译难度，类和类成员会被随机命名，除非用keep保护。

-dontobfuscate #关闭混淆

一颗星表示只是保持该包下的类名，而子包下的类名还是会被混淆；

-keep class com.thc.test.\*

两颗星表示把本包和所含子包下的类名都保持；

-keep class com.thc.test.\*\*

（上面两种方式保持类后，会发现类名虽然未混淆，但里面的具体方法和变量命名还是变了）

如果我们要保留一个类中的内部类不被混淆则需要用$符号。

-keepclassmembers class cc.ninty.chat.ui.fragment.ScriptFragment$JavaScriptInterface {

public \*;

}

既可以保持该包下的类名，又可以保持类里面的内容不被混淆;

-keep class com.thc.test.\*{\*;}

既可以保持该包及子包下的类名，又可以保持类里面的内容不被混淆;

-keep class com.thc.test.\*\*{\*;}

保持某个类名不被混淆（但是内部内容会被混淆）

-keep class com.xlpay.sqlite.cache.BaseDaoImpl

保持某个类的 类名及内部的所有内容不会混淆

-keep class com.xlpay.sqlite.cache.BaseDaoImpl{\*;}

保持类中特定内容，而不是所有的内容可以使用如下：

-keep class com.thc.gradlestudy.MyProguardBean{

<init>; #匹配所有构造器

<fields>;#匹配所有域

<methods>;#匹配所有方法

}

上面就保持住了MyProguardBean这个类中的所有的构造方法、变量、和方法

可以在或前面加上private 、public、native等来进一步指定不被混淆的内容

-keep class com.xlpay.sqlite.cache.BaseDaoImpl{

public <methods>;#保持该类下所有的共有方法不被混淆

public \*;#保持该类下所有的共有内容不被混淆

private <methods>;#保持该类下所有的私有方法不被混淆

private \*;#保持该类下所有的私有内容不被混淆

public <init>(java.lang.String);#保持该类的String类型的构造方法

}

在方法后加入参数，限制特定的方法(经测试：仅限于构造方法可以混淆)

-keep class com.thc.gradlestudy.MyProguardBean{

public <init>(String);

}

要保留一个类中的内部类不被混淆需要用 $ 符号

#保持ProguardTest中的MyClass不被混淆

-keep class com.xlpay.sqlite.cache.ProguardTest$MyClass{\*;}

使用Java的基本规则来保护特定类不被混淆，比如用extends，implement等这些Java规则，如下：保持Android底层组件和类不要混淆

-keep public class \* extends android.app.Activity

-keep public class \* extends android.app.Application

-keep public class \* extends android.app.Service

-keep public class \* extends android.content.BroadcastReceiver

-keep public class \* extends android.content.ContentProvider

-keep public class \* extends android.view.View

如果不需要保持类名，只需要保持该类下的特定方法保持不被混淆，需要使用keepclassmembers，而不是keep，因为keep方法会保持类名。

#保持ProguardTest类下test(String)方法不被混淆

-keepclassmembernames class com.xlpay.sqlite.cache.ProguardTest{

public void test(java.lang.String);

}

如果拥有某成员，保留类和类成员

-keepclasseswithmembernames class com.xlpay.sqlite.cache.ProguardTest

**注意事项**

1. jni方法不可混淆，因为native方法是要完整的包名\_类名\_方法名来定义的，不能修改，否则找不到；

#保持native方法不被混淆

-keepclasseswithmembernames class \* {

native <methods>;

}

1. 反射用到的类混淆时需要注意：只要保持反射用到的类名和方法即可，并不需要将整个被反射到的类都进行保持
2. AndroidMainfest中的类不混淆，所以四大组件和Application的子类和Framework层下所有的类默认不要进行混淆。自定义的View默认也不会被混淆
3. 与服务端交互时，使用GSON、fastjson等框架解析服务端数据时，所写的JSON对象类不混淆，否则无法将JSON解析成对应的对象；
4. 使用第三方开源库或者引用其他第三方的SDK包时，如果有特别要求，也需要在混淆文件中加入对应的混淆规则；
5. 有用到WebView的JS调用也需要保证写的接口方法不混淆，原因和第一条一样；

7. Parcelable的子类和Creator静态成员变量不混淆，否则会产生Android.os.BadParcelableException异常；

-keep class \* implements Android.os.Parcelable {

# 保持Parcelable不被混淆

public static final Android.os.Parcelable$Creator \*;

}

8. 使用enum类型时需要注意避免以下两个方法混淆，因为enum类的特殊性，以下两个方法会被反射调用，见第二条规则。

-keepclassmembers enum \* {

public static \*\*[] values();

public static \*\* valueOf(java.lang.String);

}

**建议**：

发布一款应用除了设minifyEnabled为ture，你也应该设置zipAlignEnabled为true，像Google Play强制要求开发者上传的应用必须是经过zipAlign的，zipAlign可以让安装包中的资源按4字节对齐，这样可以减少应用在运行时的内存消耗。

**附**

混淆后默认会在工程目录app/build/outputs/mapping/release（debug）下生成一个mapping.txt文件，这就是混淆规则，我们可以根据这个文件把混淆后的代码反推回源本的代码。