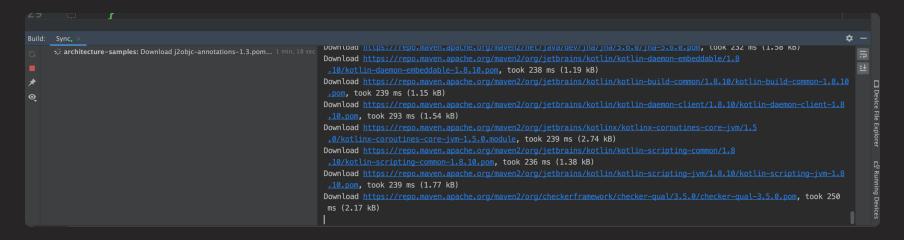
Android 同步优化: 关闭 Jetifier

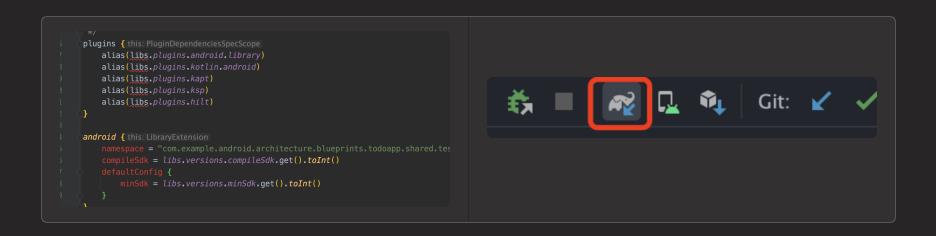


@肖玲通(小灵通)

Ctalks

背景介绍





	同步次数	同步成功率	同步耗时	after configuration
数据组 A	3.7 万	96.34%	10min	5min

Gradle 生命周期

```
gradle.beforeSettings
 Initialization
                             gradle.settingsEvaluated
                                  Projects顺序执行
                                                                gradle.beforeProjec
                                                                                                每个project都
会执行
                                                                 evaluated Project
Configuration :
                                                                gradle.afterProject
                             gradle.projectsEvaluated
                            gradle.taskGraph.whenReady
                                  Tasks顺序执行
                                                                gradle.taskGraph.beforeTask
                                                                       execute Task
  Execution
                                                                  gradle.taskGraph.afterTask
                               gradle.buildFinished
                                                                                          一時哩哔哩技术
```

断点

同步阶段分析

```
Code fragment:
gradle.startParameter.taskNam
ςes
Result:
```

密集恐惧症请闭眼

• • •

同步命令

./gradlew -P android.injected.build.model.only.advanced=true -P android.injected.build.model.only=true -P org.gradle.kotlin.dsl.provider.c id=465291211918416 -P

```
kotlin.mpp.enableIntransitiveMetadataConfiguration=true -P android.injected.invoked.from.ide=true -
P android.injected.build.model.only.versioned=3 -P android.injected.studio.version=2022.2.1 -P
idea.gradle.do.not.build.tasks=false -D idea.active=true -Didea.version=2022.2 -D
java.awt.headless=true -D idea.sync.active=true -D
org.gradle.internal.GradleProjectBuilderOptions=omit_all_tasks --init-script
/private/var/folders/0c/zkqf7nrj6pl_d40jp4m9h3xm0000gn/T/sync.studio.tooling11.gradle --init-script
/private/var/folders/0c/zkqf7nrj6pl_d40jp4m9h3xm0000gn/T/ijmapper.gradle --init-script
/private/var/folders/0c/zkqf7nrj6pl_d40jp4m9h3xm0000gn/T/ijinit.gradle
```

一次次的同步后,我们看到了这样的日志

• 嗯,同步是没有任务执行的,那看样子是配置阶段慢了

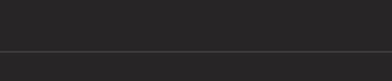
> Transform jetified-default.aar (project :weibo) with JetifyTransform WARNING: [XmlResourcesTransformer] No mapping for: android/support/FILE_PROVIDER_PATHS

```
WARNING: [XmlResourcesTransformer] No mapping for:
 android/support/FILE_PROVIDER_PATHS
 > Transform jetified-default.aar (project :weibo) with <a href="JetifyTransform">JetifyTransform</a>
 WARNING: [XmlResourcesTransformer] No mapping for:
 android/support/FILE PROVIDER PATHS
 WARNING: [XmlResourcesTransformer] No mapping for:
 android/support/FILE_PROVIDER_PATHS
 > Transform jetified-default.aar (project :weibo) with <a href="JetifyTransform">JetifyTransform</a>
 WARNING: [XmlResourcesTransformer] No mapping for:
 android/support/FILE_PROVIDER_PATHS
 WARNING: [XmlResourcesTransformer] No mapping for:
 android/support/FILE_PROVIDER_PATHS
在 gradle.properties 中设置
```

日志消失了......

android.enableJetifier=false

Jetifier



▼ 什么是 TransformAction

在依赖解析阶段进行的产物变换

IDE 中 AAR 的代码索引指向的是 aar 吗?

> Gradle: androidx.activity:activity:1.6.1@aar > Gradle: androidx.activity:activity-compose:1.6.1@aar > Tin Gradle: androidx.activity:activity-ktx:1.6.1@aar > Gradle: androidx.annotation:annotation:1.5.0

> 🖳 < jbr-17 > /Applications/Android Studio.app/Contents/jbr/Contents/ł

> In Gradle: androidx.annotation:annotation-experimental: 1.1.00 aar

> 📭 Gradle: androidx.appcompat:appcompat-resources:1.4.2@aar

> Gradle: androidx.appcompat:appcompat:1.4.2@aar

Transforming dependency artifacts on resolution

Jetifier 的背后

Support

android.enableJetifier

```
@CacheableTransform
abstract class JetifyTransform: TransformAction<JetifyTransform.Parameters> {
```

Jetifier —

```
Transform
                                                       scope
            Artifact transform
                                                       全局
                                                     任务执行期
             AGP transform
无奖竞猜
```

Artifact transform vs AGP4.2 transform

> In Gradle: androidx.arch.core:core-common:2.1.0

gradlew.bat **ILICENSE**

🚯 renovate.json settings.gradle.kts

```
> III Gradle: androidx.arch.core:core-runtime:2.1.0@aar
          > Till Gradle: androidx.arch.core:core-testing:2.1.0@aar
          > 🛅 Gradle: androidx.cardview:cardview:1.0.0@aar
          > Gradle: androidx.collection:collection:1.1.0
          > 🔐 Gradle: androidx.compose.animation:animation:1.3.3@aar
          > 🛅 Gradle: androidx.compose.animation:animation-core:1.3.3@aar
          > Gradle: androidx.compose.compiler:compiler:1.4.3
          > In Gradle: androidx.compose.foundation:foundation:1.3.1@aar
          > 🛅 Gradle: androidx.compose.foundation:foundation-layout:1.3.1@aar
          > Gradle: androidx.compose.material:material:1.3.1@aar
          > Gradle: androidx.compose.material:material-icons-core:1.3.1@aar
        答案
         不是的,指向的是 aar 中的 classes.jar
          ExtractAarTransform 源码
                                                                                                           classes.jar
             drawerlayout-1.0.0
                                                            jars
                                                            annotations.zip
                                                            R.txt
                                                            AndroidManifest.xml
Jetify Transform 浅析
support 与 androidX 的 mapping
     Class 又作
                   Cclass ByteCodeTransformer internal constructor

private val context: TransformationContext
```

val remapper = CoreRemapperImpl(context, writer) reader.accept(remapper.classRemapper, flags: 0 /*

🥷 XmlResourcesTransformer.kt 🗴 🕌 Strings.kt 🗴 🥷 JetifyTransform.kt 🗴 🏰 _Collections.kt 🗴 🥷 Processor.kt 🗴 🧶 ByteCodeTransformer.kt 🗴 🐧 Path.java 🗴 🕷 TransformationResult.kt 🗴 🥷 Archive.kt 🗴 🔩 Archive.kt 🔾 private val patterns = listOf(

• Xml 文件

```
override fun canTransform(file: ArchiveFile) = file.isXmlFile() && !file.isPomFile()
             override fun runTransform(file: ArchiveFile) {
                  if (file.isSingleFile) {...}
                  if (file.fileName == "maven-metadata.xml") {...}
                  val charset = getCharset(file)
                  val sb = StringBuilder(file.data.toString(charset))
                  val changesDone = replaceWithPatterns(sb, patterns, file.relativePath)
                  if (changesDone) {...}
                  if (context.isInReversedMode &&
                            changesDone &&
                            file.relativePath.toString().endsWith(suffix: "annotations.xml")) {
重压回 aar/jar
       er.kt 🗴 🐔 Strings.kt 🗴 🥷 jetifyTransform.kt 🗴 🐔 _Collections.kt 🗴 🥷 Processor.kt 🗴 👰 ByteCodeTransformer.kt 🗴 🐧 Path.java 🗴 🥷 TransformationResult.kt 🗴 👰 Archive.kt 💉 🥷 Archive.kt 🔭 🕷 Archive.file.kt
        aThrows(IOException::class)
        override fun writeSelfTo(outputStream: OutputStream) {
```

Pattern.compile(regex: "</?([a-zA-Z0-9.]+)"), // </{candidate} or <{candidate}

Pattern.compile(regex: $[a-zA-Z0-9:]+=\"([^\"]+)\"")$, // any="{candidate}"

Pattern.compile(regex: ">\\s*([a-zA-Z0-9.\\$_]+)<") // >{candidate}<

⑥ 在 Java 中, ASM (Java 学节码操作框架)是一个强大的库,用于在字节码级别上分析、转换 ○ ○ 和生成 Java 类文件。ClassRemapper 是 ASM 库中的一个类,用于在字节码中重新映射类名

ClassRemapper 的主要用途是在字节码转换过程中修改类名和符号引用,以满足特定的需求。一些常见的用途包括:

类重命名:通过使用 ClassRemapper,可以将类名重命名为其他名称,以增加代码的保护性和安全性。这对于代码混淆和防止逆向工程非常有用。

ClassRemapper,可以修改字节码中对外部类的引用,以适应新的依赖关系。 3. 类迁移: 当将代码从一个上下文移植到另一个上下文时,可能需要修改类名和符号。

总之,ClassRemapper 是 ASM 库中的一个工具,用于在字节码级别上重新映射类名和符号引用。它提供了灵活的功能,可以在进行字节码操作和转换时对类名和符号进行修改。

```
Log.v(TAG, message: "Writing file: %s", file.relativePath)
    val path = file.relativePath.toString().replace(oldChar: '\\', newChar: '/')
    val entry = ZipEntry(path)
    entry.lastModifiedTime = FileTime.from(Instant.now()) // b/78249473
   file.writeSelfTo(out)
out.finish()
```

问题解决

改掉同步阶段的开关

- ▼ 修改 startParameter
 - -P: projectProperties
 - 设置 gradle 全局配置参数不得行

-D: systemPropertiesArgs

修改 settings.ext

```
设置 settings.ext 参数同样不行
  settings.gradle.ext["android.enableJetifier"]="false"
```

@AppPlugin (com.android.build.gradle)

(gradle-7.4.0.jar)

o↓ | open class BasePlugin: Plugin<Project> { Choose Implementation of BasePlugin (4 found) private

修改 AGP#BasePlugin#ProjectServices

```
thi @LibraryPlugin (com.android.build.gradle)
                    @TestPlugin (com.android.build.gradle)
 //com.android.build.gradle.internal.DependencyConfigurator#configureDependencyChecks
 fun configureDependencyChecks(): DependencyConfigurator {
    val useAndroidX = projectServices.projectOptions.get(BooleanOption.USE_ANDROID_X)
    val enableJetifier = projectServices.projectOptions.get(BooleanOption.ENABLE_JETIFIER)
▼ demo 测试:在 android library 模块下测试
    class TurnOffJetifierPlugin : Plugin<Project> {
           override fun apply(project: Project) {
                    val service = it.getProjectService() ?: return@withType
                    val service = it.getProjectService() ?: return@withType
                    val projectOptions = service.projectOptions
                    val projectOptionsReflect = Reflect.on(projectOptions)
                    val optionValueReflect = Reflect.onClass(
                        "com.android.build.gradle.options.ProjectOptions\$OptionValue",
                    val defaultProvider = DefaultProvider() { false }
                    val optionValueObj = optionValueReflect.create(projectOptions,
                            .set("valueForUseAtConfiguration", defaultProvider)
                            .set("valueForUseAtExecution", defaultProvider)
                    val map = getNewMap(projectOptionsReflect, optionValueObj)
                    projectOptionsReflect.set("booleanOptionValues", map)
          private fun BasePlugin<*, *, *>?.getProjectService() =
                       .field("projectServices")
        在 demo 中验证是 work 的,移植到项目中
    // 项目的根 build.gradle 应用插件
    allProjects {
     afterEvaluate {
      apply plugin: TurnOffJetifierPlugin.class
    // Include builded project 应用插件
    settings.gradle.startParameter.initScripts = [initFile]
    # initFile
    allprojects { Project project ->
     project.afterEvaluate {
      apply plugin: TurnOffJetifierPlugin.class
  可这时代码在项目中却跪了,推测修改晚于了读取
```

1. 黑魔法依赖于 AGP 对应实现,增加了维护成本 2. 关闭同步阶段的 Jetifier 操作,并没有解决编译打包阶段的 transform

思索

▼ 最终诉求是?

▼ 改了值会有什么问题

- 迁移所有 Support 包从而关闭 Jetifier

Support 依赖迁移

▼ 二方库

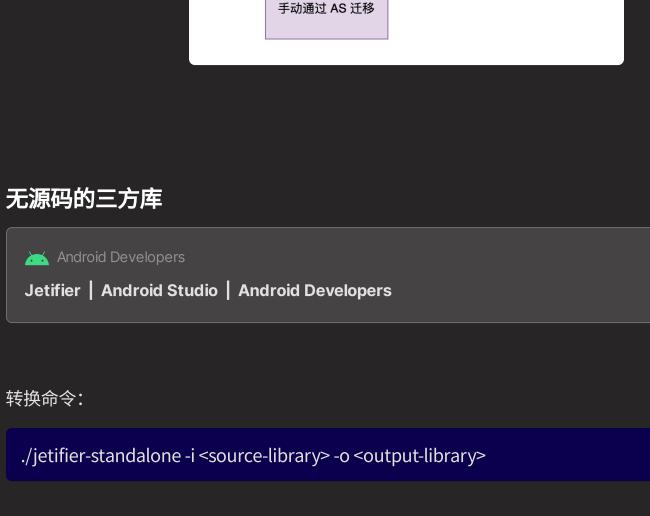
```
Convert to @CompileStatic
Add Right-to-Left (RTL) Support..
```

```
有源码的二方库: AS → Refactor → Migrate to AndroidX 进行迁移发布
▼ 三方库
   开源的三方库
```

是否已迁移 androidx

待迁移三方库

无 break change



pom 依赖的同步修改, 参考 <u>androidx-class-mapping.csv</u>

直接使用对应版本

迁移过程的坑

▼ 包大小差异过大

▼ R.class

▼ so 大小差异

转换命令:

```
R 文件传递导致的
解决办法: 开启非传递 R
```

解决办法: 用线上的 so 替换掉编译生成的 so

安利 JakeWharton 的 diffuse

ndk 版本不一致导致的

修复 600KB diff 降至 1KB

```
▼ nexus 上传
              Android
               Browse Index Browse Storage
               Select Artifact(s) for Upload
               Select Artifact(s) to Upload...
                Filename:
                Classifier:
               Extension:
                Add Artifact
```

Filename:

Classifier:

Extension:

Add Artifact

▼ AndroidX 日志检查

```
Select GAV Definition Source
                                    Artifact Upload
                                                                       GAV Definition: 🔅 GAV Parameters
                                                                                                         v 0
                                                                        Auto Guess:
                                                                                      ✓ ②
                                                                        Group:
                                                                                    com.facebook.fresco
                        C:\fakepath\imagepipeline-1.13.0-bili4.aar
                                                                        Artifact:
                                                                                      imagepipeline
                                                                        Version:
                                                                                     1.13.0
                                                                        Packaging:
                                                                                                       × 0
                                                                                      aar
对应依赖: com.facebook.fresco:imagepipeline:1.13.0:bili4
正确的依赖: com.facebook.fresco:imagepipeline:1.13.0-bili4
解决方法:
                                                                           Select GAV Definition Source
    Select Artifact(s) for Upload
                                                                           GAV Definition: • GAV Parameters
    Select Artifact(s) to Upload...
```

C:\fakepath\imagepipeline-1.13.0-bili4.aar

温馨提示:nexus 上传时,POM, .module 文件也要上传

× (3)

× 0

✓ ②

imagepipeline

1.13.0-bili4

com.facebook.fresco

Auto Guess:

Group:

Artifact:

Version:

Packaging:

```
WARNING:Your project has set `android.useAndroidX=true`, but jetified-xxx-10.10.0.aar still
 contains legacy support libraries, which may cause runtime issues.
 This behavior will not be allowed in Android Gradle plugin 8.0.
 Please use only AndroidX dependencies or set `android.enableJetifier=true` in the
  `gradle.properties` file to migrate your project to AndroidX (see
 https://developer.android.com/jetpack/androidx/migrate for more info).
反编译工具:<u>Jadx</u>
   class AndroidXEnabledJetifierDisabled(...) : Action<ResolvableDependencies> {
```

"\${AndroidXEnabledJetifierDisabled::class.java.name}_issue_reported"

```
override fun execute(resolvableDependencies: ResolvableDependencies) {
                                      // Report only once
                                      if (project.extensions.extraProperties.has(issueReported)) {
                                                     return
                                      // long long check logic
com. and roid. build. gradle. internal. dependency. And roid XDependency Check \$Android XE nabled Jetifick and roid Android 
erDisabled_issue_reported=true
```

private val issueReported =

```
▼ 防劣化
   ▼ 移除 support 依赖
        allprojects {
         configurations.all { Configuration c ->
           if (c.state == Configuration.State.UNRESOLVED) {
              exclude group: 'com.android.support'
               exclude group: 'android.arch.core'
               exclude group: 'android.arch.lifecycle'
              exclude group: 'android.arch.persistence.room'
               exclude group: 'android.arch.persistence'
               exclude group: 'com.squareup.leakcanary', module: "leakcanary-object-watcher-
        android-support-fragments"
```

▼ A8 检查阻塞合码 2023-03-27 19:06:43,078 - root - ERROR - execute gradle task FAILURE 2023-03-27 19:06:43,078 - root - ERROR - 快编二进制检查错误信息如下 Error : not found Class android.support.v7.widget.RecyclerView\$ViewHolder from void com.zhihu.matisse.internal.ui.adapter.AlbumPhotosAdapter\$PhotoViewHolder.<init> ${\tt Error: not found Class and roid.support. v4.app.} Fragment from {\tt com.zhihu.matisse.internal.ui.PhotoSelectionFragment from the com.zhihu.matisse.internal.ui.PhotoSelectionFragment$

Error : not found Method void com.zhihu.matisse.internal.ui.PhotoSelectionFragment.setArguments(android.os.Bundle) from com.zhihu.matisse.internal.ui.PhotoSelectionFragment.setArguments(android.os.Bundle)

Error : not found Method android.os.Bundle com.zhihu.matisse.internal.ui.PhotoSelectionFragment.getArguments() from void com.zhihu.matisse.internal.ui.PhotoSelecti

最后

2018 2022

androidX 库出现在 Android 的舞台上 AGP 7.2.2 宣告开启 Jetifier 的 support 库不再迭代 warning 标志

2011 2021

Android 引入 support 库 yexpunication and support 库 来实现向后兼容 Jetpack Compose 出现 完全基于 AndroidX 之上



参考链接

KOGE

artifact transform

<u>jetifier cli</u>

<u>jetifier warning</u>

classifier 相关

一切基于 POM

<u>哔哩哔哩 Android 同步优化•Jetifier</u>

广告位

广告位招租

哔哩哔哩技术



二分电台



个人微信

