“Android工具箱”之

从kapt迁移到ksp

北京理工大学计算机学院
金旭亮

概述

kapt和ksp，都是Android Studio项目中用于处理注解的插件，按照官方推荐，新项目应该使用ksp取代kapt，因为ksp拥有更快的处理速度。

本讲所介绍之内容，适用于左图所示之版本

ksp官网

https://github.com/google/ksp

Kotlin Symbol Processing API

Welcome to KSP!

Kotlin Symbol Processing (KSP) is an API that you can use to develop lightweight compiler plugins. KSP provides a simplified compiler plugin API that leverages the power of Kotlin while keeping the learning curve at a minimum. Compared to KAPT, annotation processors that use KSP can run up to 2x faster.

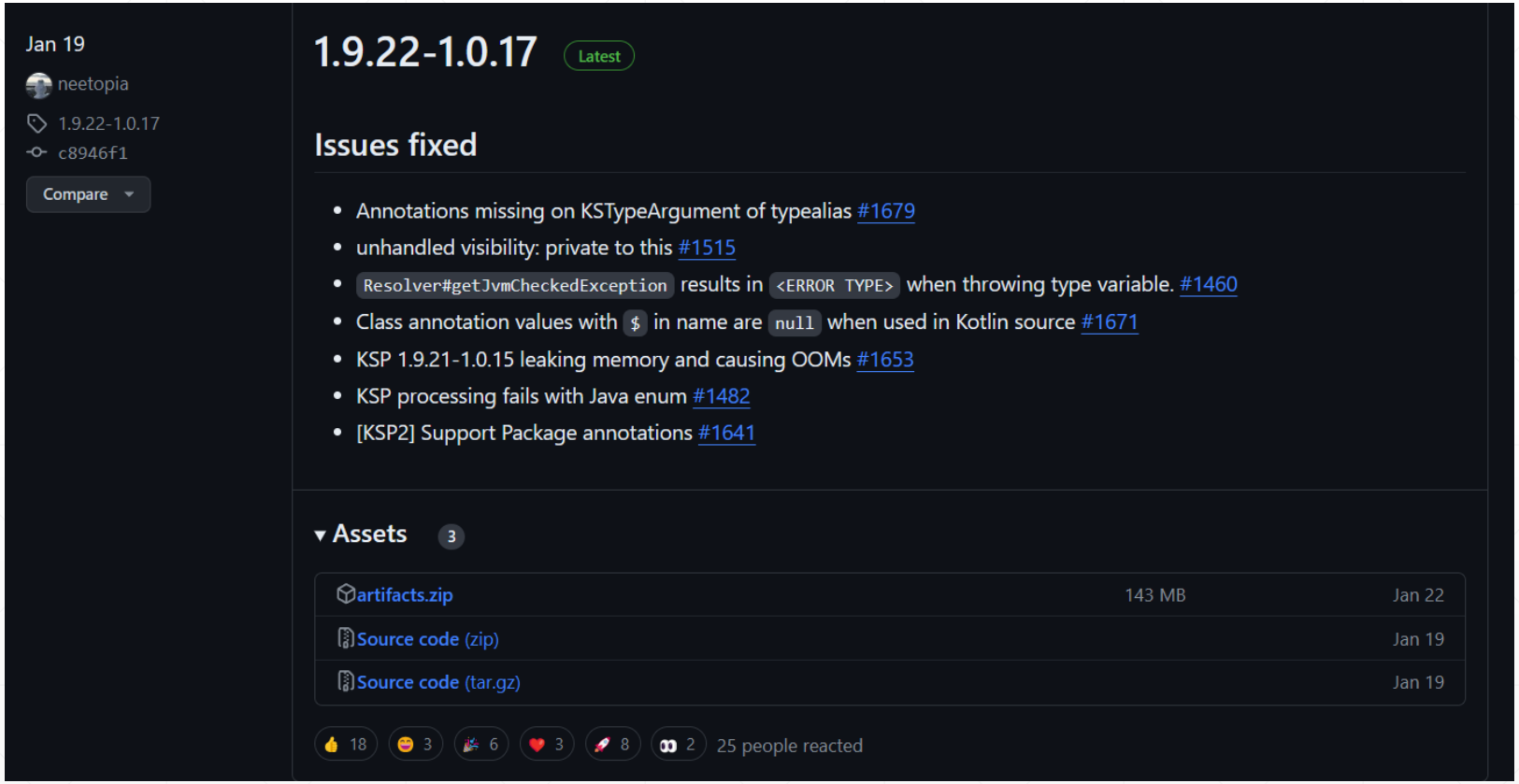
Most of the documentation of KSP can be found on kotlinlang.org. Here are some handy links:

- Overview
- Quickstart
- Libraries that support KSP
- Why KSP?
- Examples
- How KSP models Kotlin code
- Reference for Java annotation processor authors
- Incremental processing notes
- Multiple round processing notes
- KSP on multiplatform projects
- Running KSP from command line
- FAQ

For debugging and testing processors, as well as KSP itself, please check DEVELOPMENT.md

查询可用的KSP版本

https://github.com/google/ksp/releases



其中，1.9.22，是适用的Kotlin版本。

哪些库现在支持ksp?

完整列表见:

```
https://kotlinlang.org/docs/ksp-overview.html#supported-libraries
```

当在Android Studio中使用kapt时，如果有特定的ksp可用，会给出提示。

Supported libraries

The table includes a list of popular libraries on Android and their various stages of support for KSP:

Library	Status
Room	Officially supported ↗
Moshi	Officially supported ↗
RxHttp	Officially supported ↗
Kotshi	Officially supported ↗
Lyricist	Officially supported ↗
Lich SavedState	Officially supported ↗
gRPC Dekorator	Officially supported ↗
EasyAdapter	Officially supported ↗
Koin Annotations	Officially supported ↗
Glide	Officially supported ↗

在Android Studio中应用ksp-1

在项目的build.gradle.kts中，添加对ksp插件的引用：

build.gradle.kts (My Application)

```
1  // Top-level build file where you can add configuration options common
2  plugins { this: PluginDependenciesSpecScope
3      alias(libs.plugins.androidApplication) apply false
4      alias(libs.plugins.jetbrainsKotlinAndroid) apply false
5      id("com.google.devtools.ksp") version "1.9.22-1.0.17" apply false
6  }
```

注意，需要“Ctrl+点击”jetbrainsKotlinAndroid以了解当前使用的Kotlin版本，然后，到前页PPT所示之网页上查询可用的ksp版本，两者必须一致。

在Android Studio中应用ksp-2

在模块的build.gradle.kts中启用插件：

build.gradle.kts (:app)

```
1  plugins { this: PluginDependenciesSpecScope
2      alias(libs.plugins.androidApplication)
3      alias(libs.plugins.jetbrainsKotlinAndroid)
4      id("com.google.devtools.ksp")
5  }
```

现在，就可以使用ksp取代kapt了，以下是Jetpack Room的示例：

```
val roomVersion = "2.6.1"
implementation("androidx.room:room-runtime:$roomVersion")
ksp("androidx.room:room-compiler:$roomVersion")
implementation("androidx.room:room-ktx:$roomVersion")
testImplementation("androidx.room:room-testing:$roomVersion")
```

切换到新格式-1

如果愿意的话，可以在Android Studio的帮助下，切换到新的“library catalog declaration”格式：

```
val roomVersion = "2.6.1"
implementation("androidx.room:room-runtime:$roomVersion")
ksp("androidx.room:
implementation("and
testImplementation("androidx.room:room-testing:$roomVersion")
```

Use version catalog instead More... (Ctrl+F1)

Replace with new library catalog declaration for androidx-room-runtime Alt+Shift+Enter More actions... Alt+Enter

切换到新格式-2

build.gradle.kts

```
dependencies { this: DependencyHandlerScope

    implementation(libs.androidx.room.runtime)
    ksp(libs.androidx.room.compiler)
    implementation(libs.androidx.room.ktx)
    testImplementation(libs.androidx.room.testing)
```

libs.versions.toml

```
[versions]
roomVersion = "2.6.1"

[libraries]
androidx-room-testing = { module = "androidx.room:room-testing", version.ref = "roomVersion" }
androidx-room-ktx = { module = "androidx.room:room-ktx", version.ref = "roomVersion" }
androidx-room-compiler = { module = "androidx.room:room-compiler", version.ref = "roomVersion" }
androidx-room-runtime = { module = "androidx.room:room-runtime", version.ref = "roomVersion" }
```