Groovy language support for Android

[Build Status]

This plugin adds Groovy Language support for Android applications and libraries.

Updates

As of 2.0.0 of this plugin only will work with the Android Gradle Plugin 3.0.0 and above. For support of lower version use 1.2.0.

There is an issues when using build tool 26+ and the Groovy jar. The current work around is to use proguard or to use JarJar to create a jar file that does not have invoke dynamic classes. See ZarZaring the Groovy Jar for Android 26+ to create your own jar and avoid having to run proguard. See Github Issue for more details.

Quick Start

Use a lazybones template from grooid-template

Usage

Edit your `build.gradle` file to contain the following:

```
buildscript {
  repositories {
    jcenter()
  }

  dependencies {
    classpath 'com.android.tools.build:gradle:3.0.0'
    classpath 'org.codehaus.groovy:groovy-android-gradle-plugin:2.0.0'
  }
}

apply plugin: 'com.android.application'
apply plugin: 'groovyx.android'
```

The latest version of the Groovy Android Plugin can be found here

You must choose which version of Groovy you use. Android support is available in starting at the 2.4.x releases. You will need to add the following repository to your `build.gradle` file:

```
repositories {
  jcenter()
}
```

Then you can start using Groovy by adding the groovy dependency with the `grooid` classifier:

```
dependencies {
  compile 'org.codehaus.groovy:groovy:2.4.12:grooid'
}
```

Full list of releases can be found here here. Then use the `assembleDebug` gradle task to test out your build and make sure everything compiles.

Should you want to test development versions of the plugin, you can add the snapshot repository and depend on a SNAPSHOT:

```
buildscript {
  repositories {
    jcenter()
    maven { url 'http://oss.jfrog.org/artifactory/oss-snapshot-local' }
  }
}
  dependencies {
    classpath 'com.android.tools.build:gradle:3.0.0'
    classpath 'org.codehaus.groovy:groovy-android-gradle-plugin:2.1.0-SNAPSHOT'
  }
}
```

Go here to see what the latest SNAPSHOT version is.

Where to put sources?

Groovy sources may be placed in `src/main/groovy`, `src/test/groovy`, `src/androidTest/groovy` and any `src/${buildVariant}/groovy` configured by default. A default project will have the `release` and `debug` variants but these can be configured with build types and flavors. See the android plugin docs for more about configuring different build variants.

Extra groovy sources may be added in a similar fashion as the android plugin using the `androidGroovy.sourceSets` block. This is especially useful for sharing code between the different test types, and also allows you to add Groovy to an existing project. For example

```
androidGroovy {
  sourceSets {
    main {
      groovy {
        srcDirs += 'src/main/java'
      }
    }
  }
}
```

would add all of the Java files in `src/main/java` directory to the Groovy compile task. These files will be removed from the Java compile task, instead being compiled by GroovyC, and will allow for the Java sources to be referenced in the Groovy sources (joint compilation). Please note, that you may need to also add these extra directories to the Java source sets in the android plugin for Android Studio to recognize the Groovy files as source.

Writing Groovy Code

This plugin has been successfully tested with Android Studio and will make no attempts to add support for other IDEs. This plugin will let you write an application in Groovy but it is recommended, for performance, memory and battery life, that you use `@CompileStatic` wherever possible.

Details can be found on Melix's blog and here for more technical details

Including Groovy Libraries

In order to include libraries written in groovy that include the groovy or groovy-all jars, you will need to exclude the groovy dependency allowing the grooid jar to be the one to be compiled against.

For example to use the groovy-xml library you would simply need to do exclude the group `org.codehaus.groovy`.

```
compile ('org.codehaus.groovy:groovy-xml:2.4.3') {
    exclude group: 'org.codehaus.groovy'
}
```

Skipping Groovy Compile

As of version 1.2.0 only build types/build flavors with groovy sources included in them will have the groovy compile task added. If you would like to skip the groovy compilation tasks on older versions or on newer version wish to skip them in build types that have groovy sources you can use the following to disable the groovy compiler task.

```
tasks.whenTaskAdded { task ->
  if (task.name == 'compileDebugGroovyWithGroovyc') { ①
    task.enabled = false
  }
}
```

① Disables groovy compilation only for the debug build type, simply replace `compileDebugGroovyWithGroovyc` with whichever compilation task you would like skip to disable it.

Configuring the Groovy compilation options

The Groovy compilation tasks can be configured in the `androidGroovy` block using the `options` block:

```
androidGroovy {
  options {
    configure(groovyOptions) {
      encoding = 'UTF-8'
      forkOptions.jvmArgs = ['-noverify'] // maybe necessary if you use Google Play
Services
    }
  }
}
```

See GroovyCompile for more options. See Example Application for an example of using these settings to enable custom compilation options.

Only Use GroovyC

For integration with plain java projects or for working with generated files (such as BuildConfig) it may be desirable to only have GroovyC run in order to have Java files reference Groovy files. This is roughly the equivalent of placing all java source files into the groovy source directory (including auto generated files like BuildConfig). In order to only have GroovyC run simply set the flag `skipJavaC` in the `androidGroovy` block to true.

```
androidGroovy {
  skipJavaC = true
}
```

Annotation Processing

As of 1.2.0 Release annotation processing is configured by default.

Previous versions would require `javaAnnotationProcessing` to be set to true.

```
androidGroovy {
  options {
    configure(groovyOptions) {
      javaAnnotationProcessing = true
    }
  }
}
```

Android `packagingOptions`

Groovy Extension Modules and Global transformations both need a file descriptor in order to work. Android packaging has a restriction related to files having the same name located in the same path.

If you are using several Groovy libraries containing extension modules and/or global transformations, Android may complain about those files.

You can simply add the following rule:

```
android {
  packagingOptions {
      exclude 'META-INF/services/org.codehaus.groovy.transform.ASTTransformation'
      exclude 'META-INF/services/org.codehaus.groovy.runtime.ExtensionModule'
  }
}
```

There are no problems excluding global transformation descriptors because those are only used at compile time, never at runtime.

The problem comes with module extensions. Unless you statically compile classes using extension modules with `@CompileStatic` they won't be available at runtime and you'll get a runtime exception.

There is an alternative. The emerger gradle plugin will add excludes for you and merges all extension module descriptors into a single file which will be available at runtime.

Groovy Community

Have questions, want to learn more!? Come ask questions or help others in #android in the Groovy Community Slack groovycommunity.com[Sign-Up Here]