Things I wish I knew when I started building Android Libraries VOL. 2

Nishant Srivastava

Soundbrenner

OSS Android libraries developed

Android Libraries/SDK	Github	Description
Sensey	Stars 2k	Android library to make detecting gestures easy
EasyDeviceInfo	Stars 1k	Enabling device information to be at android developers hand like a piece of cake!
RecyclerViewHelper	Stars 598	RecyclerViewHelper provides the most common functions around recycler view like Swipe to dismiss, Drag and Drop, Divider in the ui, events for when item selected and when not selected, on-click listener for items.
QREader	Stars 294	A library that uses google's mobile vision api and simplifies the QR code reading process
ScreenShott	Stars 240	Simple library to take a screenshot of the device screen, programmatically!
Android-Utils	Stars 111	Android library facilitating some very common functionalities in the form of utility classes for Android
PackageHunter	Stars 133	Android library to hunt down package information
Zentone	Stars 70	Easily generate audio tone in android
StackedHorizontalProgressbar	Stars 70	Android Library to implement stacked horizontal progressbar
OptimusHTTP	Stars 31	Android library that simplifies networking in android via an async http client
EvTrack	Stars 7	Android library to make event and exception tracking easy
ShoutOut	Stars 10	Android library for logging information in android
Lantern	Stars 49	Android library handling flashlight for camera and camera2 api. Added support for handling display/screen light.
ValidateTor	Stars 105	Android library for fast and simple string validation.

Side A

1. How does Gradle process Android Libraries
2. Android Archive(AAR)
3. Why doesn't my AAR download transitive dependencies?
4. Maven Artifact
5. POM?
6. Bundled Proguard Configs

Side B

7. Modularization
8. Avoiding Resource Name Conflicts
9. minSdkVersion Restriction
10. Access Visibility vs Code Organization
11. Lifecycle Aware Android Library
12. Auto Init Android Library

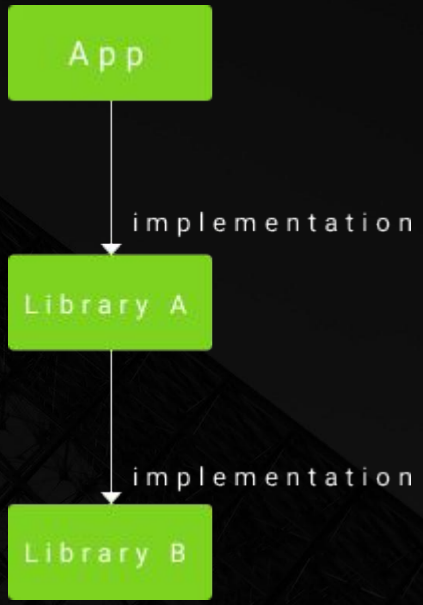
How does Gradle process Android Libraries

How does Gradle process Android Libraries

```
repositories{
  jCenter()
}

dependencies {
   implementation 'com.example.myawesomelib:1.0.0'
   ...
}
```

How does Gradle process Android Libraries

- implementation

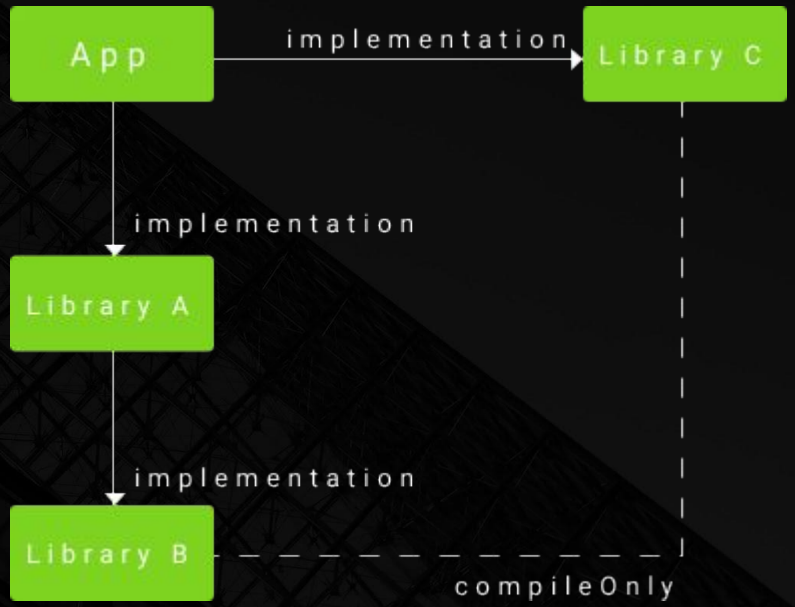


How does Gradle process Android Libraries

- api

How does Gradle process Android Libraries

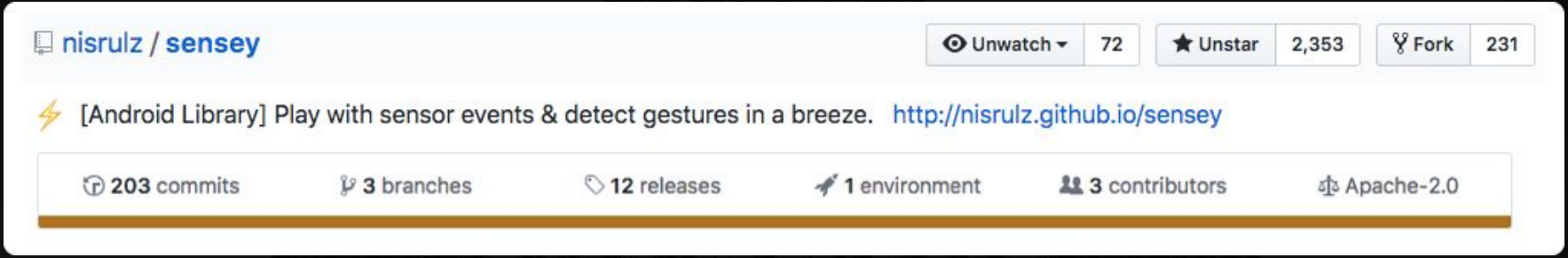
- compileOnly



Android ARchive(AAR)

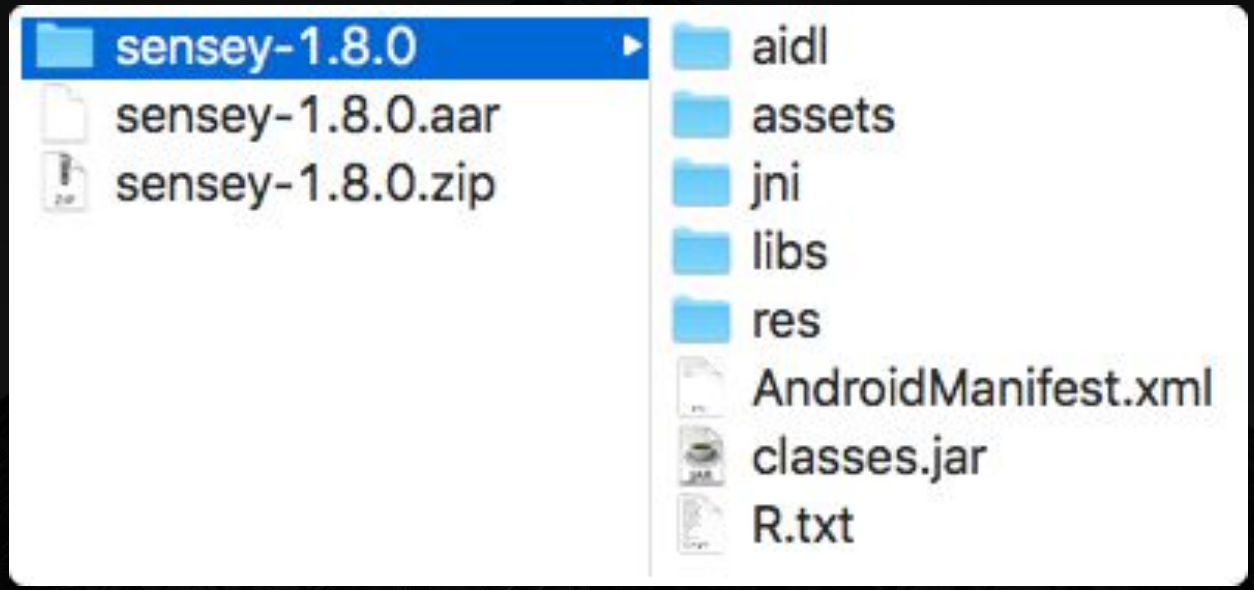
AAR = Java ARchive(JAR) + Resources

AAR = Java ARchive(JAR) + Resources



Sensey: https://github.com/nisrulz/sensey

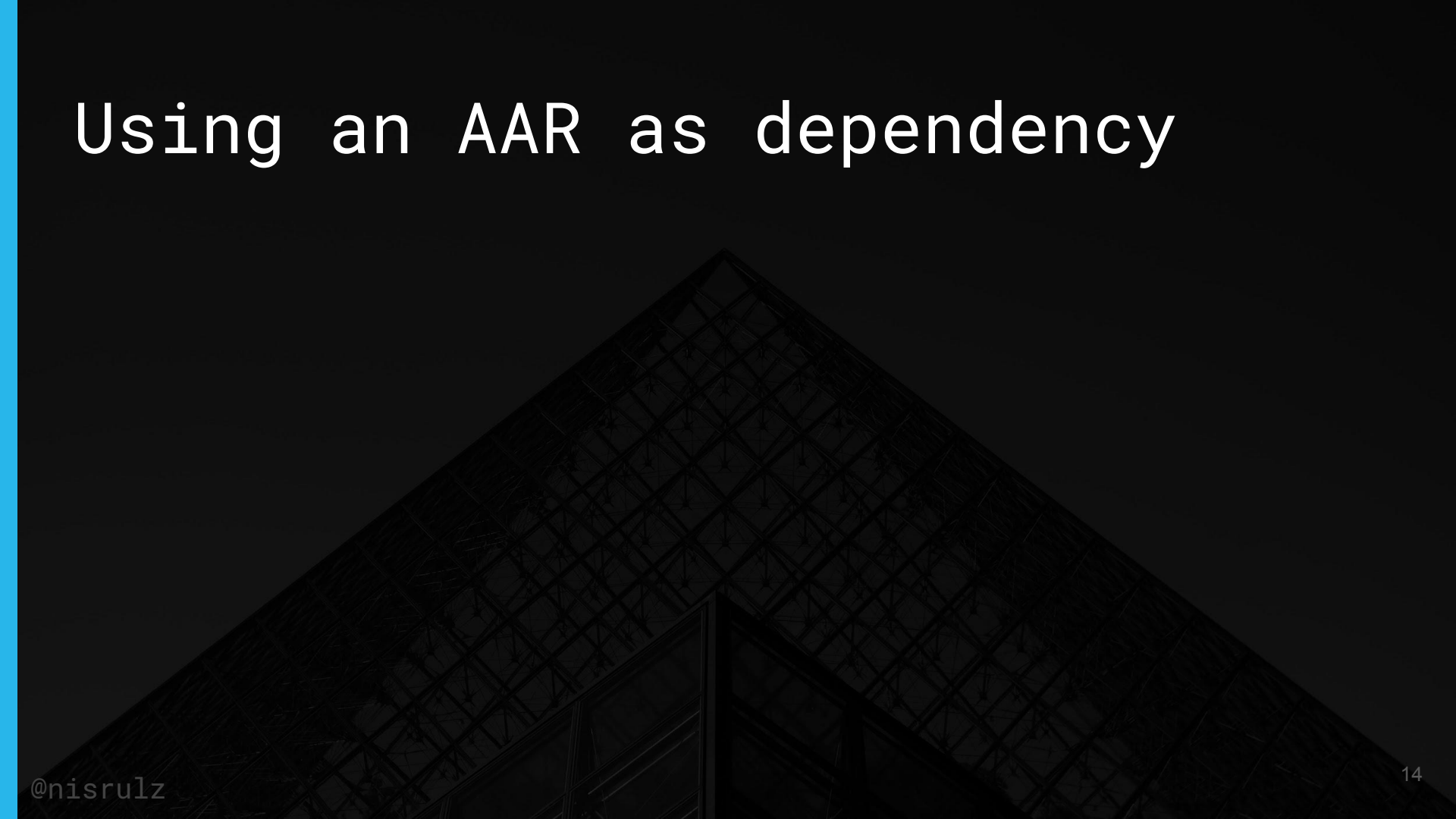
AAR = Java ARchive(JAR) + Resources



Sensey's build.gradle

```
dependencies {
    ...
    // Other testing dependencies

    // Transitive dependency: Support Compat library
    implementation "com.android.support:support-compat:27.0.2"
}
```

Using an AAR as dependency



Using an AAR as dependency

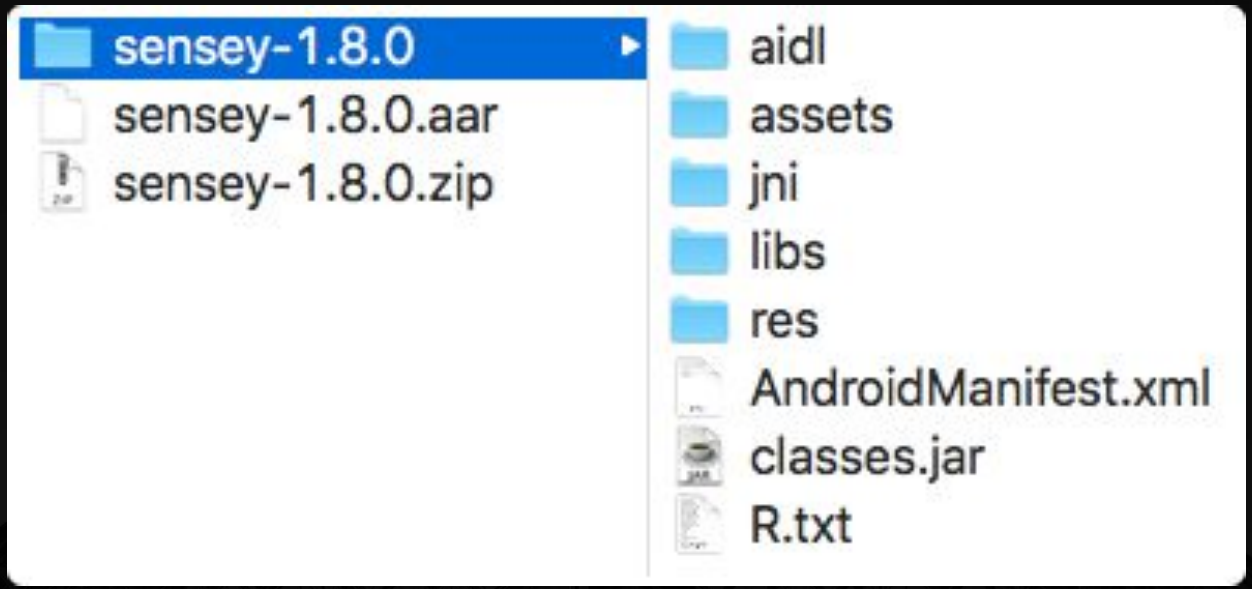
```
repositories{
  flatDir{
     dirs 'libs'
   }
}
dependencies {
   implementation(name:'nameOfYourAARFileWithoutExtension', ext:'aar')
}
```

Using an AAR as dependency

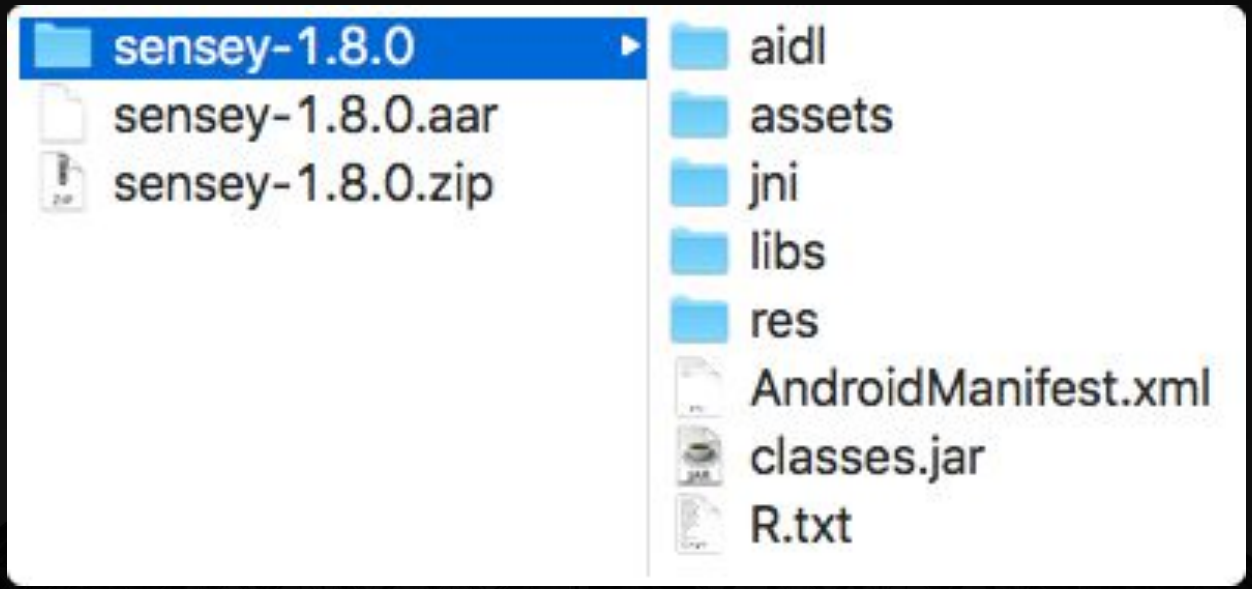
```
repositories{
  flatDir{
     dirs 'libs'
   }
}
dependencies {
   implementation(name:'nameOfYourAARFileWithoutExtension', ext:'aar')
}
// No transitive dependencies of the library are downloaded
```

Why doesn’t my AAR download transitive dependencies?

Sensey's AAR


sensey-1.8.0
sensey-1.8.0.aar
sensey-1.8.0.zip
aidl
assets
jni
libs
res
AndroidManifest.xml
classes.jar
R.txt

Sensey's AAR



No **build.gradle** file 😢

Maven Artifact

Maven Artifact

POM?

POM

- Project Object Model
- XML file
- Configuration details used by Maven to build the project

POM

POM of Sensey Android Library

```
<?xml version="1.0" encoding="UTF-8"?>
<project xsi:schemaLocation="..." xmlns="..."xmlns:xsi="...">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.github.nisrulz</groupId><artifactId>sensey</artifactId><version>1.8.0</version>
  <packaging>aar</packaging><name>sensey</name>
  <description>Android library which makes playing with sensor events &amp;amp; detecting gestures a breeze.</description>
  <url>https://github.com/nisrulz/sensey</url>
  <licenses>
    <license>
      <name>The Apache Software License, Version 2.0</name><url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
    </license>
  </licenses>
  <developers>
    <developer><id>nisrulz</id><name>Nishant Srivastava</name><email>nisrulz@gmail.com</email></developer>
  </developers>
  <scm>
    <connection>https://github.com/nisrulz/sensey.git</connection>
    <developerConnection>https://github.com/nisrulz/sensey.git</developerConnection>
    <url>https://github.com/nisrulz/sensey</url>
  </scm>
  <dependencies>
    <dependency>
      <groupId>com.android.support</groupId>
      <artifactId>support-compat</artifactId><version>27.0.2</version>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      ...
    </dependency>
  </dependencies>
</project>
```

POM

POM of Sensey Android Library

```
<?xml version="1.0" encoding="UTF-8"?>
<project xsi:schemaLocation="..." xmlns="..."xmlns:xsi="...">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.github.nisrulz</groupId><artifactId>sensey</artifactId><version>1.8.0</version>
  <packaging>aar</packaging><name>sensey</name>
  <description>Android library which makes playing with sensor events &amp;amp; detecting gestures a breeze.</description>
  <url>https://github.com/nisrulz/sensey</url>
  <licenses>
    <license>
      <name>The Apache Software License, Version 2.0</name><url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
    </license>
  </licenses>
  <developers>
    <developer><id>nisrulz</id><name>Nishant Srivastava</name><email>nisrulz@gmail.com</email></developer>
  </developers>
  <scm>
    <connection>https://github.com/nisrulz/sensey.git</connection>
    <developerConnection>https://github.com/nisrulz/sensey.git</developerConnection>
    <url>https://github.com/nisrulz/sensey</url>
  </scm>
  <dependencies>
    <dependency>
      <groupId>com.android.support</groupId>
      <artifactId>support-compat</artifactId><version>27.0.2</version>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      ...
    </dependency>
  </dependencies>
</project>
```

POM

POM of Sensey Android Library

```
<?xml version="1.0" encoding="UTF-8"?>
<project xsi:schemaLocation="..." xmlns="..."xmlns:xsi="...">
  ...
  <dependencies>
    <dependency>
      <groupId>com.android.support</groupId>
      <artifactId>support-compat</artifactId><version>27.0.2</version>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      ...
    </dependency>
  </dependencies>
</project>
```

Bundled Proguard Configs

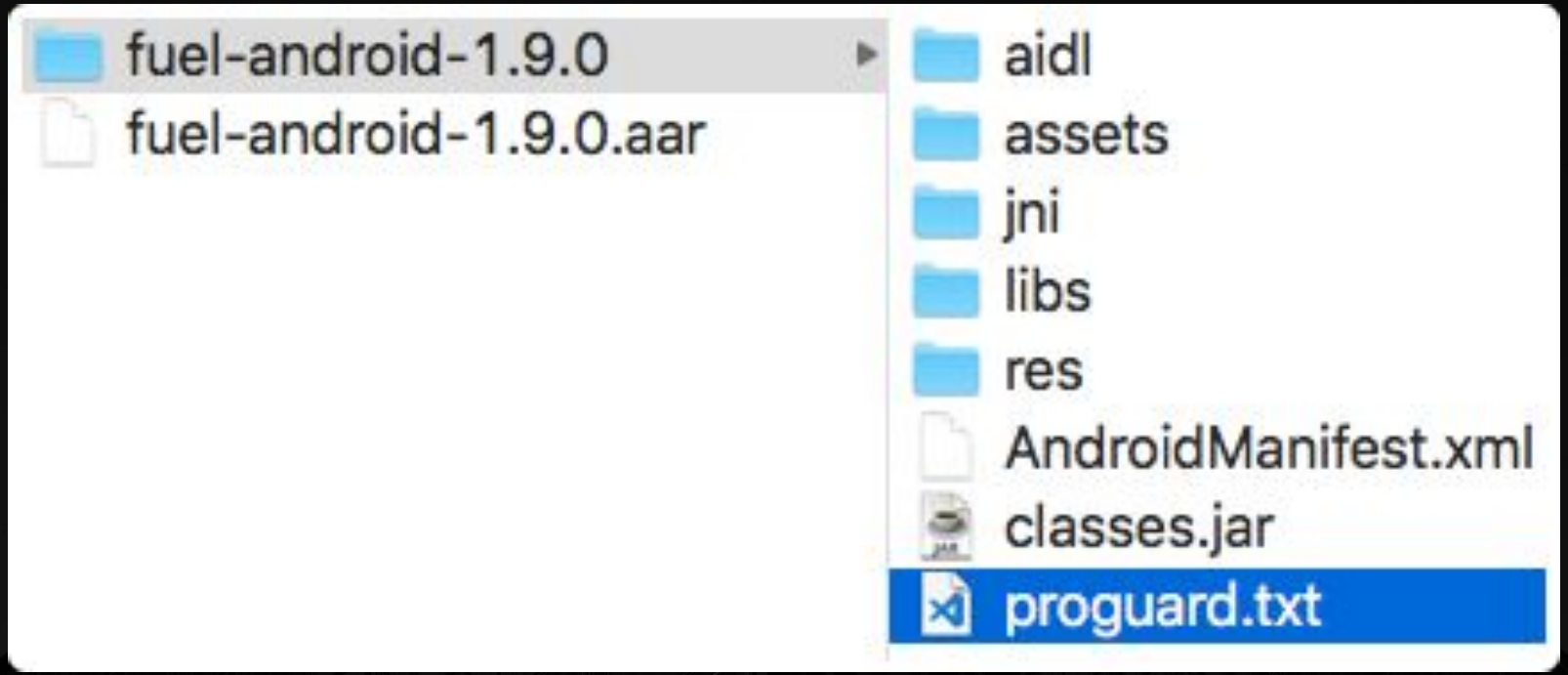
Bundled Proguard Configs

```
android {
    release {
        minifyEnabled true

        // Rules to be used during the AAR generation
        proguardFiles 'proguard-rules-for-building-library.pro'

        // Rules appended to the integrating app
        consumerProguardFiles 'proguard-rules-for-using-library.pro'
    }
    ...
}
```

Bundled Proguard Configs

Fuel: https://github.com/kittinunf/Fuel

Bundled Proguard Configs



Bundled Proguard Configs

```
# Fuel’s bundled Proguard file
# Without specifically keeping this class,
# callbacks on android don't function properly.
-keep class com.github.kittinunf.fuel.android.util.AndroidEnvironment
```

Bundled Proguard Configs

```
# To check the merged configuration
# Add the below to your current config
-printconfiguration proguard-merged-config.txt
```

Bundled Proguard Configs

```
# DON'T DO THIS
-dontobfuscate
-optimizations !code/allocation/variable

# Effectively no optimizations
-keep public class * {
    public protected *;
}
```

Bundled Proguard Configs

```
# DON'T DO THIS
# Adding the below in library proguard rules disables
# the optimizations in the Android app
-dontoptimize
```

Modularization

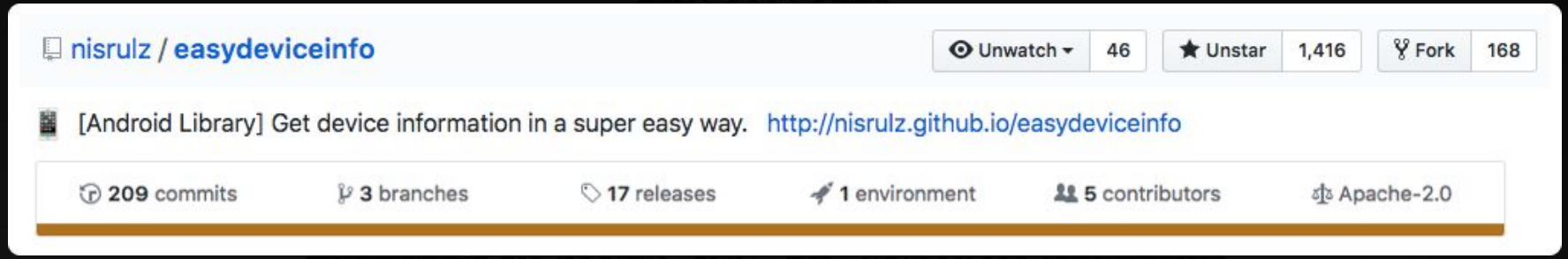
Modularization

★ **Note:** Don't use the combined **`play-services`** target. It brings in dozens of libraries, bloating your application. Instead, specify only the specific Google Play services APIs your app uses.

Table 1. Individual APIs and corresponding `build.gradle` descriptions.

API	Description in **`build.gradle`**
Google+	**`com.google.android.gms:play-services-plus:16.0.0`**
Google Account Login	**`com.google.android.gms:play-services-auth:16.0.1`**
Google Actions, Base Client Library	**`com.google.android.gms:play-services-base:16.0.1`**
Google Sign In	**`com.google.android.gms:play-services-identity:16.0.0`**
Google Analytics	**`com.google.android.gms:play-services-analytics:16.0.4`**

Modularization



EasyDeviceInfo: https://github.com/nisrulz/easydeviceinfo

Modularization

```
dependencies {
    def libVer = {latest_version}

    // Base + Ads Bundled Library
    implementation "com.github.nisrulz:easydeviceinfo:$libVer"

    // Base Library
    implementation "com.github.nisrulz:easydeviceinfo-base:$libVer"

    // Ads Library
    implementation "com.github.nisrulz:easydeviceinfo-ads:$libVer"
}
```

Modularization

```
<!-- POM file for com.github.nisrulz:easydeviceinfo -->
<project xsi:schemaLocation="..." xmlns="..." xmlns:xsi="...">
  ...
  <dependencies>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-ads</artifactId> <version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-base</artifactId> <version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    ...
  </dependencies>
</project>
```

Modularization

```
<!-- POM file for com.github.nisrulz:easydeviceinfo-->
<project xsi:schemaLocation="..." xmlns="..." xmlns:xsi="...">
  ...
  <dependencies>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-ads</artifactId> <version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-base</artifactId> <version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    ...
  </dependencies>
</project>
```

Modularization

```
<!-- POM file for com.github.nisrulz:easydeviceinfo-->
<project xsi:schemaLocation="..." xmlns="..." xmlns:xsi="...">
  ...
  <dependencies>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-ads</artifactId> <version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-base</artifactId> <version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    ...
  </dependencies>
</project>
```

Modularization

```
<!-- POM file for com.github.nisrulz:easydeviceinfo-ads -->
<project xsi:schemaLocation="..." xmlns="..." xmlns:xsi="...">
  ...
  <dependencies>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-common</artifactId><version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>com.google.android.gms</groupId>
      <artifactId>play-services-ads-identifier</artifactId><version>16.0.0</version>
      <scope>runtime</scope>
    </dependency>
    ...
  </dependencies>
</project>
```

Modularization

```
<!-- POM file for com.github.nisrulz:easydeviceinfo-ads -->
<project xsi:schemaLocation="..." xmlns="..." xmlns:xsi="...">
  ...
  <dependencies>
    <dependency>
      <groupId>com.github.nisrulz</groupId>
      <artifactId>easydeviceinfo-common</artifactId><version>2.5.0</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>com.google.android.gms</groupId>
      <artifactId>play-services-ads-identifier</artifactId><version>16.0.0</version>
      <scope>runtime</scope>
    </dependency>
    ...
  </dependencies>
</project>
```

Avoiding Resource Name Conflicts

What happens when...

Conflict occurs between a library & app resource

> Project will not compile

What happens when...

Conflict occurs between 2 libraries integrated in the app

> Resources from library defined first in *build.gradle* gets included

Solution?

Solution?

Add a prefix to all your resources.

Solution?

Add a prefix to all your resources.

How?

Solution?

Add a prefix to all your resources.

How?

- Enforce this in Android Studio

Solution?

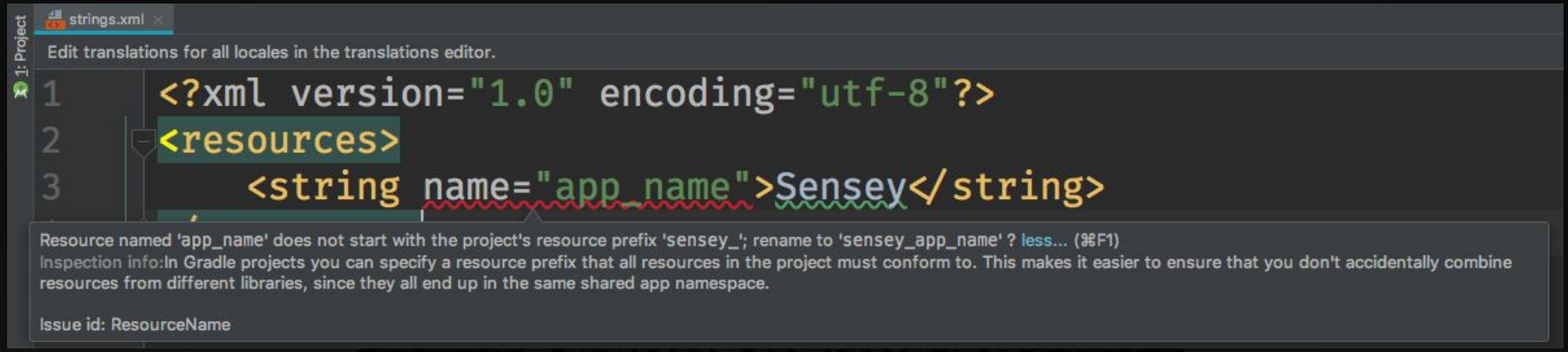
Add a prefix to all your resources.

How?

- Enforce this in Android Studio
- Declare in *build.gradle* of library

```
android {
    resourcePrefix 'YOUR_PREFIX_' // i.e 'sensey_'
}
```

Solution?



Resource named ‘app_name’ does not start with the project’s resource prefix ‘sensey_’;

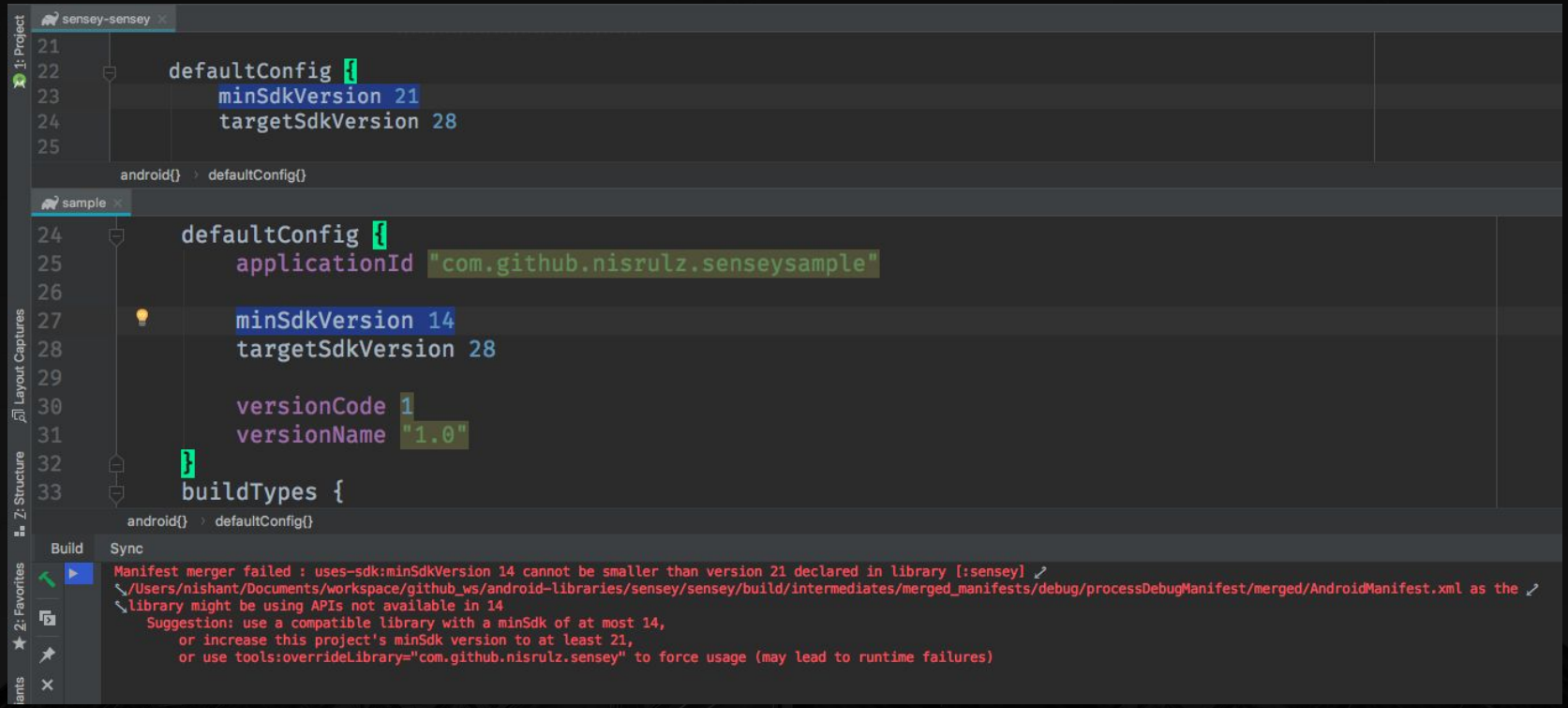
Rename to `sensey_app_name`?

minSdkVersion Restriction

minSdk Restriction

minSdkVersion of app >= minSdkVersion of library

minSdk Restriction



minSdk Restriction

Manifest merger failed : uses-sdk:minSdkVersion 14 cannot be smaller than version 21 declared in library [:sensey]

Suggestion:

+ Use a compatible library with a minSdk of at most 14
+ Increase this project's minSdk version to at least 21
+ Use tools:overrideLibrary="com.github.nisrulz.sensey" to force usage (may lead to runtime failures)

minSdk Restriction

Manifest merger failed : uses-sdk:minSdkVersion 14 cannot be smaller than version 21 declared in library [:sensey]

Suggestion:

+ Use a compatible library with a minSdk of at most 14
+ Increase this project's minSdk version to at least 21
+ Use tools:overrideLibrary="com.github.nisrulz.sensey" to force usage (may lead to runtime failures)

minSdk Restriction

Manifest merger failed : uses-sdk:minSdkVersion 14 cannot be smaller than version 21 declared in library [:sensey]

Suggestion:

+ Use a compatible library with a minSdk of at most 14
+ Increase this project's minSdk version to at least 21
+ Use tools:overrideLibrary="com.github.nisrulz.sensey" to force usage (may lead to runtime failures)

minSdk Restriction

```
<manifest xmlns:android="..."
    xmlns:tools="http://schemas.android.com/tools"
    package="com.github.nisrulz.senseysample">

    <uses-sdk tools:overrideLibrary="com.github.nisrulz.sensey"/>

    <application ...>
        ...
    </application>
</manifest>
```

minSdk Restriction

```
<manifest xmlns:android="..."
    xmlns:tools="http://schemas.android.com/tools"
    package="com.github.nisrulz.senseysample">

    <uses-sdk tools:overrideLibrary="com.github.nisrulz.sensey"/>

    <application ...>
        ...
    </application>
</manifest>
```

Access Visibility

vs

Code Organization

Visibility vs Organization

Visibility vs Organization

- Code organized in individual packages; everything is public

Visibility vs Organization

- Code organized inside one package; everything is package private and only public on demand

Visibility vs Organization

- Code organized inside the module i.e individual packages; everything is internal and only public on demand (in Kotlin land)

- Drawback
  - Need dependency on kotlin std library

Visibility vs Organization

```
// file name: exampleLibrary.kt
// module name: example

package com.example.library
```

Visibility vs Organization

```
// file name: exampleLibrary.kt
// module name: example

package com.example.library

// visible inside exampleLibrary.kt
private fun setup() { ... }
```

Visibility vs Organization

```
// file name: exampleLibrary.kt
// module name: example

package com.example.library

// visible inside exampleLibrary.kt
private fun setup() { ... }

// property is visible everywhere
public var name: String = "ExampleLib"
    // setter is visible only in exampleLibrary.kt
    private set{...}
```

Visibility vs Organization

```
// file name: exampleLibrary.kt
// module name: example

package com.example.library

// visible inside exampleLibrary.kt
private fun setup() { ... }

// property is visible everywhere
public var name: String = "ExampleLib"
    // setter is visible only in exampleLibrary.kt
    private set{...}

// visible inside the module i.e example
internal val debugTag = "Example-Debug"
```

Lifecycle-Aware

Android Library

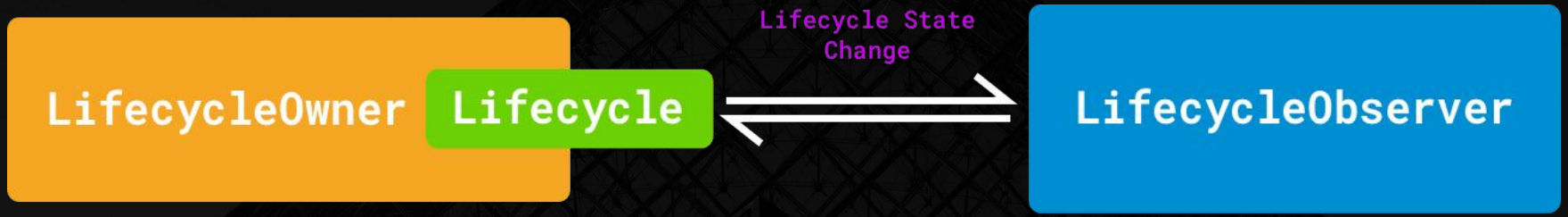
Lifecycle Components

Classes designed to help deal with Android lifecycle

- Lifecycle
- LifecycleOwner
- LifecycleObserver

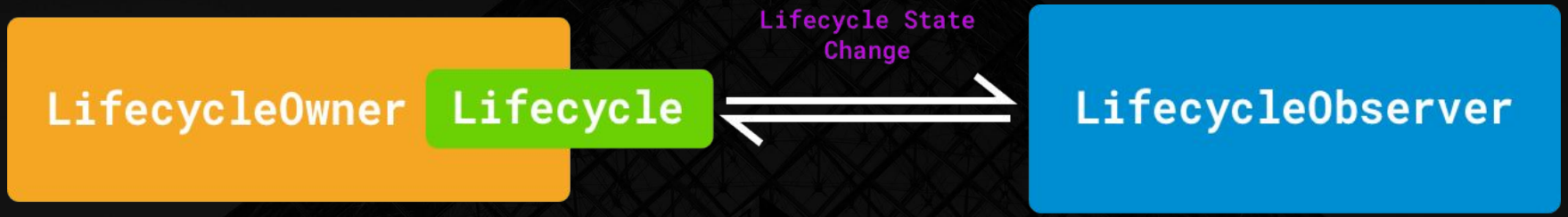
Lifecycle Components

Classes designed to help deal with Android lifecycle



Lifecycle Components

Classes designed to help deal with Android lifecycle



i.e Activity, Fragment, Service, Custom

Lifecycle

```
class MainActivity extends AppCompatActivity() {...}
```

Lifecycle

```
class MainActivity extends AppCompatActivity() {...}

public class AppCompatActivity extends FragmentActivity{...}
```

Lifecycle

```
class MainActivity extends AppCompatActivity() {...}

public class AppCompatActivity extends FragmentActivity{...}

public class FragmentActivity extends SupportActivity {
    ...
    public Lifecycle getLifecycle() {
        return super.getLifecycle();
    }
    ...
}
```

LifecycleObserver

```
dependencies {
    def lifecycleVer = "2.0.0"

    // Runtime
    implementation "androidx.lifecycle:lifecycle-runtime:$lifecycleVer"

    // Annotation Support
    annotationProcessor "androidx.lifecycle:lifecycle-compiler:$lifecycleVer"
    ...
}
```

LifecycleObserver

```
public class AwesomeLib implements LifecycleObserver {

    @OnLifecycleEvent(Lifecycle.Event.ON_CREATE)
    public void init() { ... }

    @OnLifecycleEvent(Lifecycle.Event.ON_START)
    public void libOnStart() { ... }

    ...
    @OnLifecycleEvent(Lifecycle.Event.ON_DESTROY)
    public void cleanup() { ... }
}
```

LifecycleOwner

```
class MainActivity : AppCompatActivity() {

    val awesomeLib = AwesomeLib()

    override fun onResume() {
        ...
        // Add lifecycle observer
        lifecycle.addObserver(awesomeLib)
    }

    override fun onStop() {
        ...
        // Remove lifecycle observer
        lifecycle.removeObserver(awesomeLib)
    }
}
```

LifecycleAware Library

ProcessLifecycleOwner

Class that tracks the lifecycle of whole application process

ProcessLifecycleOwner

```
dependencies {

    // For ProcessLifecycleOwner
    implementation "androidx.lifecycle:lifecycle-extensions:2.0.0"
    ...
}
```

ProcessLifecycleOwner

Does something weird…

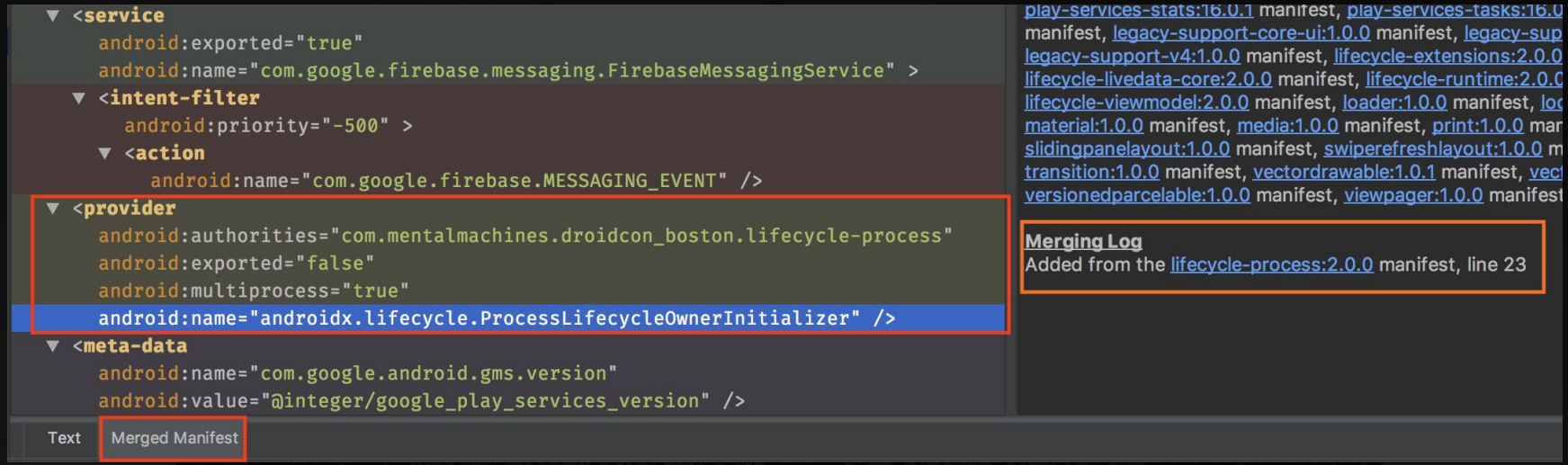
ProcessLifecycleOwner

Does something weird…

Adding lifecycle-extensions artifact

> automatically adds <provider> element to the merged manifest

ProcessLifecycleOwner



ProcessLifecycleOwner

```
<manifest >
    <application>
        ...
        <provider
            android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
            android:authorities="com.example.app.lifecycle-process"
            android:exported="false"
            android:multiprocess="true" />
    </application>
</manifest>
```

ProcessLifecycleOwner

```
<manifest >
    <application>
        ...
        <provider
            android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
            android:authorities="com.example.app.lifecycle-process"
            android:exported="false"
            android:multiprocess="true" />
    </application>
</manifest>
```

ProcessLifecycleOwner

Pre-AndroidX

```
<manifest >
    <application>
        ...
        <provider
            android:name="android.arch.lifecycle.ProcessLifecycleOwnerInitializer"
            android:authorities="com.example.app.lifecycle-trojan"
            android:exported="false"
            android:multiprocess="true" />
    </application>
</manifest>
```

ProcessLifecycleOwner

Pre-AndroidX

```
<manifest >
    <application>
        ...
        <provider
            android:name="android.arch.lifecycle.ProcessLifecycleOwnerInitializer"
            android:authorities="com.example.app.lifecycle-trojan"
            android:exported="false"
            android:multiprocess="true" />
    </application>
</manifest>
```

ProcessLifecycleOwner

Pre-AndroidX

lifecycle-trojan

One of our customers is looking at releasing an app with our library. It's just gone through their security testing and they have flagged a certain content provider that is listed in the manifest of the generated apk. It's called <app-packagename>.lifecycle-trojan. This seems to have been generated automatically and it takes the package name of the host-app. We are using lifecycle components so I speculate it's to do with that.

I can verify that I get the same content provider in my app when I build a dummy app which contains the library.

Can someone explain what lifecycle-trojan does?

9 Comments Share Save Give Award Hide Report

ProcessLifecycleOwner

AndroidX

```
<manifest >
    <application>
        ...
        <provider
            android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
            android:authorities="com.example.app.lifecycle-process"
            android:exported="false"
            android:multiprocess="true" />
    </application>
</manifest>
```

ProcessLifecycleOwner

AndroidX

```
<manifest >
    <application>
        ...
        <provider
            android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
            android:authorities="com.example.app.lifecycle-process"
            android:exported="false"
            android:multiprocess="true" />
    </application>
</manifest>
```

ProcessLifecycleOwner

```
// Internal class to initialize Lifecycles.
public class ProcessLifecycleOwnerInitializer extends ContentProvider {
    @Override
    public boolean onCreate() {
        ...
        ProcessLifecycleOwner.init(getContext());
        return true;
    }
    ...
}
```

ProcessLifecycleOwner

Why?

To invoke ProcessLifecycleOwner as soon as process starts

ProcessLifecycleOwner

Why?

To invoke ProcessLifecycleOwner as soon as process starts

Drawback?

Initializes ProcessLifecycleOwner even if your app does not use it!

ProcessLifecycleOwner

How to get rid of it in app?

ProcessLifecycleOwner

How to get rid of it in app?

Use Merge rule marker in app's AndroidManifest.xml:

```
// Remove marked element from the merged manifest
tools:node="remove"
```

ProcessLifecycleOwner

App's AndroidManifest.xml

```
<application>
    ...
    <provider
       android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
       android:authorities="${applicationId}.lifecycle-process"

       xmlns:tools="http://schemas.android.com/tools"
       tools:node="remove"
       />
</application>
```

ProcessLifecycleOwner

App's AndroidManifest.xml

```
<application>
    ...
    <provider
       android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
       android:authorities="${applicationId}.lifecycle-process"

       xmlns:tools="http://schemas.android.com/tools"
       tools:node="remove"
       />
</application>
```

ProcessLifecycleOwner

App's AndroidManifest.xml

```
<application>
    ...
    <provider
       android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
       android:authorities="${applicationId}.lifecycle-process"

       xmlns:tools="http://schemas.android.com/tools"
       tools:node="remove"
       />
</application>
```

ProcessLifecycleOwner

App's AndroidManifest.xml

```
<application>
    ...
    <provider
       android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
       android:authorities="${applicationId}.lifecycle-process"

       xmlns:tools="http://schemas.android.com/tools"
       tools:node="remove"
       />
</application>
```

Auto Initialize

Android Library

AutoInit Android Library

Android Libraries need Android context to handle simple tasks such as

- Hook into Android Runtime
- Access app resources
- Use System Services
- Register BroadcastReceiver

AutoInit Android Library

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // Init android library
        MyAwesomeLibrary.init(this);
    }
}
```

AutoInit Android Library

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // Init android library
        MyAwesomeLibrary.init(this);
    }
}
```

AutoInit Android Library

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // Init android library
        MyAwesomeLibrary.init(this);
    }
}
```

AutoInit Android Library

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:name=".MyApplication"
    ... >
```

AutoInit Android Library

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:name=".MyApplication"
    ... >
```

AutoInit Android Library

`ContentProvider` can be used to simplify the process.

AutoInit Android Library

`ContentProvider` can be used to simplify the process.

AutoInit Android Library

ContentProvider can be used to simplify the process.

Simply because ContentProvider

- Is created and initialized (on the main thread) before *all* other components
- Participate in manifest merging at build time.

AutoInit Android Library

```
public class AwesomeLibInitProvider extends ContentProvider {
    ...
    @Override
    public boolean onCreate() {

        // get the context (Application context)
        Context context = getContext();

        // initialize AwesomeLib  here
        AwesomeLib.getInstance().init(context);

        return false;
    }
    ...
}
```

AutoInit Android Library

```
public class AwesomeLibInitProvider extends ContentProvider {
    ...
    @Override
    public boolean onCreate() {

        // get the context (Application context)
        Context context = getContext();

        // initialize AwesomeLib  here
        AwesomeLib.getInstance().init(context);

        return false;
    }
    ...
}
```

AutoInit Android Library

```
<manifest xmlns:android=".."
    package="github.nisrulz.sample.awesomelib">

    <application>
        <provider
            android:name=".AwesomeLibInitProvider"
            android:authorities="${applicationId}.awesomelibinitprovider"
            android:enabled="true"
            android:exported="false"/>
    </application>

</manifest>
```

AutoInit Android Library

```
<manifest xmlns:android=".."
    package="github.nisrulz.sample.awesomelib">

    <application>
        <provider
            android:name=".AwesomeLibInitProvider"
            android:authorities="${applicationId}.awesomelibinitprovider"
            android:enabled="true"
            android:exported="false"/>
    </application>

</manifest>
```

AutoInit Android Library

```
<manifest xmlns:android=".."
    package="github.nisrulz.sample.awesomelib">

    <application>
        <provider
            android:name=".AwesomeLibInitProvider"
            android:authorities="${applicationId}.awesomelibinitprovider"
            android:enabled="true"
            android:exported="false"/>
    </application>

</manifest>
```

AutoInit Android Library

```
<manifest xmlns:android=".."
    package="github.nisrulz.sample.awesomelib">

    <application>
        <provider
            android:name=".AwesomeLibInitProvider"
            android:authorities="${applicationId}.awesomelibinitprovider"
            android:enabled="true"
            android:exported="false"/>
    </application>

</manifest>
```

AutoInit Android Library

`ContentProvider` can be used to simplify the process.

AutoInit Android Library

ContentProvider can be used to simplify the process.

Challenges:

- There can be only one Content Provider with a given “authority” string
- Only run on main thread

AutoInit Android Library

`ContentProvider` can be used to simplify the process.

AutoInit Android Library

~~ContentProvider can be used to simplify the process.~~

Why this is a bad idea:

- Increases startup time
- Bloats applications even when not used
- Abusing functionality of ContentProvider

AutoInit Android Library

~~ContentProvider can be used to simplify the process.~~

Why this is a bad idea:

- Increases startup time (Solution: async initialize)
- Bloats applications even when not used
- Abusing functionality of ContentProvider

AutoInit Android Library

Some android libraries that use this...

AutoInit Android Library

ProcessLifecycleOwner
[androidx.lifecycle:lifecycle-extensions:2.0.0]

```
<application>
    <provider
        android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"
        android:authorities="${applicationId}.lifecycle-process"
        android:exported="false"
        android:multiprocess="true" />
</application>
```

AutoInit Android Library

Firebase
[com.google.firebase:firebase-common:16.0.5]

```
<application>
    <provider
        android:name="com.google.firebase.provider.FirebaseInitProvider"
        android:authorities="${applicationId}.firebaseinitprovider"
        android:exported="false"
        android:initOrder="100" />
</application>
```

AutoInit Android Library

Facebook-Core

[com.facebook.android:facebook-core:4.34.0]

```
<application>
    <provider
        android:name="com.facebook.internal.FacebookInitProvider"
        android:authorities="${applicationId}.FacebookInitProvider"
        android:exported="false" />
</application>
```

AutoInit Android Library

Facebook-Marketing

[com.facebook.android:facebook-marketing:4.34.0]

```
<application>
    <provider
        android:name="com.facebook.marketing.internal.MarketingInitProvider"
        android:authorities="${applicationId}.MarketingInitProvider"
        android:exported="false" />
</application>
```

AutoInit Android Library

Crashlytics

[com.crashlytics.sdk.android:crashlytics:2.9.5]

```
<application>
    <provider
        android:name="com.crashlytics.android.CrashlyticsInitProvider"
        android:authorities="${applicationId}.crashlyticsinitprovider"
        android:exported="false"
        android:initOrder="90" />
</application>
```

AutoInit Android Library

Picasso

```
// Pre - v2.71828
Picasso.with(this).load("...<url>...").into(imageView);

// After - v2.71828
Picasso.get().load("...<url>...").into(imageView);
```

AutoInit Android Library

Picasso

```
// Pre - v2.71828
Picasso.with(this).load("...<url>...").into(imageView);

// After - v2.71828
Picasso.get().load("...<url>...").into(imageView);
```

AutoInit Android Library

Picasso

[com.squareup.picasso:picasso:2.71828]

```
<application>
    <provider
        android:name=".PicassoContentProvider"
        android:authorities="${applicationId}.com.squareup.picasso3"
        android:exported="false"/>
 </application>
```

Links/References

Android Libraries I have built:
https://github.com/nisrulz/nisrulz.github.io#open-source-contributions

Auto initialize android library example:
https://github.com/nisrulz/android-examples/tree/develop/AutoInitLibrary

Lifecycle Aware android library example:
https://github.com/nisrulz/android-examples/tree/develop/LifeCycleCompForLib


Thank You
Q & A
twitter.com/nisrulz
github.com/nisrulz
www.nisrulz.com


Things I wish I knew
when I started building
Android Libraries
VOL. 2
Nishant Srivastava
twitter.com/nisrulz
github.com/nisrulz
www.nisrulz.com