Writing your own Annotation Processors in Android

Hitanshu Dhawan
Android Developer @ UrbanClap

What are Annotations ?

a form of syntactic **metadata** that can be added to classes, interfaces, methods, variables, parameters etc.

e.g., @Override, @Nullable.

What are Annotation Processors ?

a tool for **scanning** and **processing** annotations at compile time.

- ➔ is part of the compilation process
- ➔ can issue notes, warnings and errors
- ➔ can generate 'new' source files

e.g., Room, Dagger.

Why Annotation Processors ?

- ➔ ~~runtime~~ compile time
- ➔ no reflection
- ➔ generate boilerplate code*

How does Annotation Processing work ?

- ➔ annotation processing tool (`apt`)
- ➔ the Mirror API
- ➔ annotation processing rounds

How does Annotation Processing work ?

- ➔ **annotation processing tool (`apt`)**
- ➔ the Mirror API
- ➔ annotation processing rounds

Annotation Processing Tool (`apt`)

- is part of the `javac` compiler
- scans for all the annotations in source files
- invokes the registered annotation processors

How does Annotation Processing work ?

- ➔ annotation processing tool (`apt`)
- ➔ **the Mirror API**
- ➔ annotation processing rounds

The Mirror API

- ➔ is used to model the semantic structure of a program
- ➔ is represented at the language level, rather than at the VM level
- ➔ is a read-only API

How does Annotation Processing work ?

- ➔ annotation processing tool (`apt`)
- ➔ the Mirror API
- ➔ **annotation processing rounds**

Annotation Processing Rounds

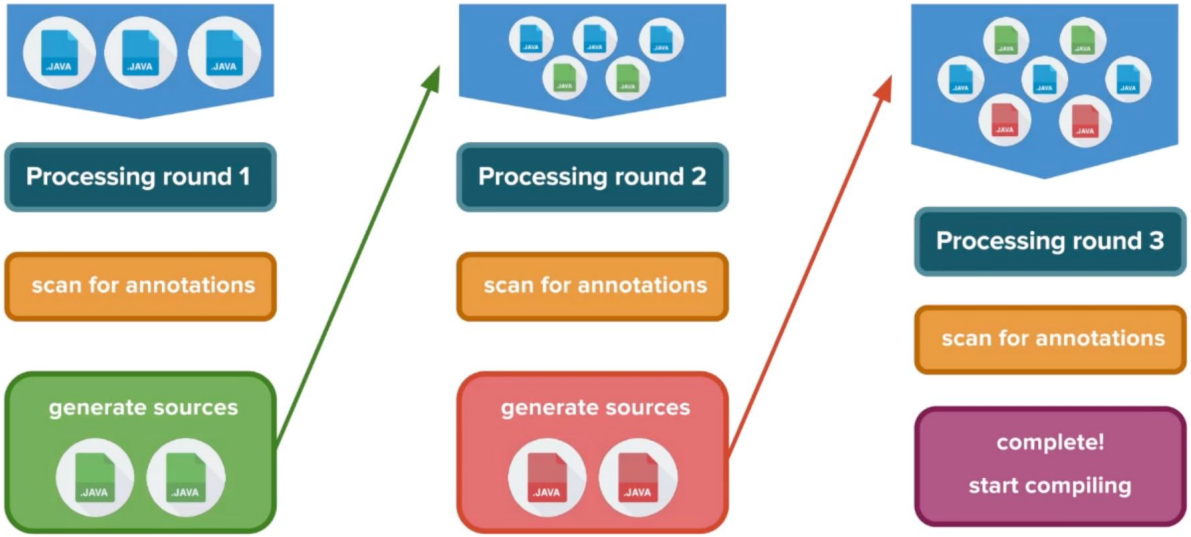

image courtesy: Jorge Castillo

Let's get started...

```
@Singleton
```

Let's get started...

```java
@Singleton
public class LocationUtil {

    private static final LocationUtil INSTANCE = new LocationUtil();

    private LocationUtil() {}

    public static LocationUtil getInstance() {
        return INSTANCE;
    }

}
```

How to write Annotations ?

```
@Retention(RetentionPolicy.SOURCE)
@Target(ElementType.TYPE)
public @interface Singleton {}
```

How to write Annotation Processors ?

```
public class Processor extends AbstractProcessor {

    //…

}
```

```java
public class Processor extends AbstractProcessor {

    public synchronized void init(ProcessingEnvironment processingEnv) {
        super.init(processingEnvironment);
        // get the utilities…
    }

    public boolean process(Set<? extends TypeElement> annotations,
                           RoundEnvironment roundEnv) {
        // do the processing…
        return true;
    }

    public Set<String> getSupportedAnnotationTypes() {
        return new HashSet<String>() {{
            add(Singleton.class.getCanonicalName());
        }};
    }

    public SourceVersion getSupportedSourceVersion() {
        return SourceVersion.latestSupported();
    }
}
```

init()

```
public synchronized void init(ProcessingEnvironment processingEnv) {
    super.init(processingEnv);
    mMessager = processingEnv.getMessager();
    mFiler    = processingEnv.getFiler();
    mElements = processingEnv.getElementUtils();
    mTypes    = processingEnv.getTypeUtils();
}
```

process()

```
public boolean process(Set<? extends TypeElement> annotations,
                       RoundEnvironment roundEnv) {

    for (TypeElement typeElement :
            ElementFilter.typesIn(roundEnv.getElementsAnnotatedWith(Singleton.class))) {

        // check for private constructors
        // …

        // check for getInstance method
        // …
    }

    return true;
}
```

```java
// check for private constructors

List<ExecutableElement> constructors =
        ElementFilter.constructorsIn(typeElement.getEnclosedElements());
for (ExecutableElement constructor : constructors) {
    if (constructor.getModifiers().isEmpty()
            || !constructor.getModifiers().contains(Modifier.PRIVATE)) {
        mMessager.printMessage(Kind.ERROR,
                "constructor of a singleton class must be private", constructor);
        return true;
    }
}
```

```java
// check for getInstance method

List<ExecutableElement> methods =
        ElementFilter.methodsIn(typeElement.getEnclosedElements());
for (ExecutableElement method : methods) {
    // check for name
    if (method.getSimpleName().contentEquals("getInstance")) {
        // check for return type
        if (mTypes.isSameType(method.getReturnType(), typeElement.asType())) {
            // check for modifiers
            if (method.getModifiers().contains(Modifier.PRIVATE)) {
                mMessager.printMessage(Kind.ERROR,
                        "getInstance method can't have a private modifier", method);
                return true;
            }
            if (!method.getModifiers().contains(Modifier.STATIC)) {
                mMessager.printMessage(Kind.ERROR,
                        "getInstance method should have a static modifier", method);
                return true;
            }
        }
    }
}
```

How to register Annotation Processors ?

How to register Annotation Processors ?

- ➔ old way

sing › singleton-compiler › src › main › resources › META-INF › services › javax.annotation.processing.Processor

app

Git:

Project

AnnotationProcessing ~/Desktop/home/AnnotationProcessing

- .gradle
- .idea
- app
- build
- butterknife
- butterknife-annotations
- butterknife-compiler
- gradle
- images
- ksingleton-annotations
- ksingleton-compiler
- singleton-annotations
- singleton-compiler
  - build
  - src
    - main
      - java
      - resources
        - META-INF.services
          - javax.annotation.processing.Processor
    - build.gradle
    - singleton-compiler.iml
  - .gitignore
  - AnnotationProcessing.iml
  - build.gradle
  - gradle.properties
  - gradlew
  - gradlew.bat
  - LICENSE
  - local.properties
  - README.md
  - settings.gradle
- External Libraries
- Scratches and Consoles

javax.annotation.processing.Processor

```
1  com.hitanshudhawan.singleton_compiler.Processor
```

1: Project | Resource Manager | Layout Captures | 7: Structure | 2: Favorites | Build Variants | Gradle | Device File Explorer

9: Version Control | TODO | Terminal | Build | 6: Logcat | 2 Event Log

Gradle build finished in 926 ms (a minute ago)

1:48 n/a UTF-8 4 spaces Git: master

How to register Annotation Processors ?

➔ new way

```java
@AutoService(Processor.class)
public class Processor extends AbstractProcessor {

    //…

}
```

app > src > main > java > com > hitanshudhawan > annotationprocessingexample > LocationUtil

Singleton.java | LocationUtil.java | Processor.java

```java
package com.hitanshudhawan.annotationprocessingexample;

import com.hitanshudhawan.singleton_annotations.Singleton;

@Singleton
public class LocationUtil {

    private static final LocationUtil INSTANCE = new LocationUtil();

    private LocationUtil() {}

    public static LocationUtil getInstance() {
        return INSTANCE;
    }

}
```

Build: | Build Output | Sync

- **Build:** completed successfully at 2019-06-15 15:34 — 823 ms
  - Run build /Users/hitanshudhawan/Desktop/home/AnnotationProcessing — 787 ms
    - Load build — 2 ms
    - Configure build — 329 ms
    - Calculate task graph — 54 ms
    - Run tasks — 401 ms

TODO | Terminal | Build | 6: Logcat | Event Log

Gradle build finished in 824 ms (moments ago)

1:56 LF UTF-8 4 spaces

app > src > main > java > com > hitanshudhawan > annotationprocessingexample > LocationUtil

Singleton.java | LocationUtil.java | Processor.java

```java
package com.hitanshudhawan.annotationprocessingexample;

import com.hitanshudhawan.singleton_annotations.Singleton;

@Singleton
public class LocationUtil {

    private static final LocationUtil INSTANCE = new LocationUtil();

    public LocationUtil() {}

    public static LocationUtil getInstance() {
        return INSTANCE;
    }

}
```

Build: Build Output | Sync

- **Build:** build failed at 2019-06-15 15:34 with 1 error — 1 s 22 ms
  - Run build /Users/hitanshudhawan/Desktop/home/AnnotationProcessing — 976 ms
    - Load build — 2 ms
    - Configure build — 342 ms
    - Calculate task graph — 50 ms
    - Run tasks — 580 ms
  - **Java compiler:** (1 error)
    - /Users/hitanshudhawan/Desktop/home/AnnotationProcessing (1 error)
      - app/src/main/java (1 error)
        - com/hitanshudhawan/annotationprocessingexample/LocationUtil.java (1 error)
          - error: constructor of a singleton class must be private

```
error: constructor of a singleton class must be private
```

TODO | Terminal | Build | 6: Logcat | Event Log

Gradle build finished in 1 s 22 ms (moments ago)

1:56 LF UTF-8 4 spaces

app > src > main > java > com > hitanshudhawan > annotationprocessingexample > LocationUtil

Singleton.java | LocationUtil.java | Processor.java

```java
package com.hitanshudhawan.annotationprocessingexample;

import com.hitanshudhawan.singleton_annotations.Singleton;

@Singleton
public class LocationUtil {

    private static final LocationUtil INSTANCE = new LocationUtil();

    private LocationUtil() {}

    public LocationUtil getInstance() {
        return INSTANCE;
    }

}
```

Build: Build Output | Sync

- **Build:** build failed at 2019-06-15 15:35 with 1 error — 885 ms
  - Run build /Users/hitanshudhawan/Desktop/home/AnnotationProcessing — 817 ms
    - Load build — 2 ms
    - Configure build — 290 ms
    - Calculate task graph — 46 ms
    - Run tasks — 473 ms
  - **Java compiler:** (1 error)
    - /Users/hitanshudhawan/Desktop/home/AnnotationProcessing (1 error)
      - app/src/main/java (1 error)
        - com/hitanshudhawan/annotationprocessingexample/LocationUtil.java (1 error)
          - error: getInstance method should have a static modifier

```
error: getInstance method should have a static modifier
```

TODO | Terminal | Build | 6: Logcat | Event Log

Gradle build finished in 885 ms (moments ago)

1:56 LF UTF-8 4 spaces

How to generate `.java` files ?

JavaPoet

```java
package com.example.helloworld;

public final class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, JavaPoet!");
    }
}
```

```java
MethodSpec main = MethodSpec.methodBuilder("main")
    .addModifiers(Modifier.PUBLIC, Modifier.STATIC)
    .returns(void.class)
    .addParameter(String[].class, "args")
    .addStatement("$T.out.println($S)", System.class, "Hello, JavaPoet!")
    .build();

TypeSpec helloWorld = TypeSpec.classBuilder("HelloWorld")
    .addModifiers(Modifier.PUBLIC, Modifier.FINAL)
    .addMethod(main)
    .build();

JavaFile javaFile = JavaFile.builder("com.example.helloworld", helloWorld)
    .build();

javaFile.writeTo(System.out);
```

ButterKnife

```
@BindView & @OnClick
```

Resources

- Writing your own Annotation Processors in Android
  https://medium.com/androidiots/writing-your-own-annotation-processors-in-android-1fa0cd96ef11
- Annotation Processing Samples
  https://github.com/hitanshu-dhawan/AnnotationProcessing
- PojoPreferences
  https://github.com/hitanshu-dhawan/PojoPreferences

@ThankYou

/in/hitanshu-dhawan

/hitanshu-dhawan

/@hitanshudhawan