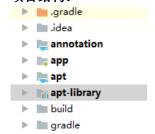
APT动态生成代码,绑定View

通过注解,绑定内容,通过APT生成代码,最后由APP调用;

项目结构:



app为项目,调用APT方;

annotation 注解:

apt 动态生成代码的类;

apt-library 工具包,提供给app调用生成代码的方法;

annotation 注解



@Target (ElementType. FIELD) 表示注解作用的地方,类,方法,成员变量,这里FIELD表示成员变量; @Retention(RetentionPolicy. CLASS) 表示注解保留的生命周期,这里是指保留到生成CLASS文件时;

int value(); 默认的一个参数,用于获取注解时参数,比如: @BindView(R.id.tv)中获取R.id.tv

build.gradle内容如下:

```
apply plugin: 'java-library'
 annotation
build
                                                   dependencies {
                                                       implementation fileTree(dir: 'lik
     libs
   ▼ 🖿 src
                                            5
      ▼ 🖿 main
        ▼ 📄 java
          ▼ 🖿 com.shinhoandroid.mvp.annotion
                                            8
                                                   sourceCompatibility = "1.7"
                @ 🔓 BindView
                                                   targetCompatibility = "1.7"
                                            9
      gitignore .gitignore
      annotation.iml
     build.gradle
```

apt 动态生成代码

```
apt apt
                                   | @AutoService (Processor.class) / 自动生成 javax.annotation.processing.IPro
  libs
                                   @SupportedSourceVersion(SourceVersion.RELEASE 8) // java版本支持
 ▼ 📄 src
                                   {\tt @SupportedAnnotationTypes(\{"com.shinhoandroid.mvp.annotion.BindView}) \\
                                  public class AnnotationProcessor extends AbstractProcessor {
    ▼ iava
      ▼ 🛅 com.shinhoandroid.mvp.ap
                                      public Messager mMessager;
         © № AnnotationProcessor
                                      public Elements mElements;
           ClassCreatorProxy
@AutoService(Processor.class) //自动生成Java文件
@SupportedSourceVersion(SourceVersion. RELEASE 8) //java版本支持
@SupportedAnnotationTypes({"com.shinhoandroid.mvp.annotion.BindView"})/标注注解处理器
支持的注解类型,就是我们刚才定义的接口BindView,可以写多个接口
process 生成代码部分
 public boolean process(Set<? extends TypeElement> set, RoundEnvironment roundEnvironment) {
     mFiler = processingEnv.getFiler();//文件相关的辅助类
     mElements = processingEnv.getElementUtils();//元素相关的辅助类
   mMessager = processingEnv.getMessager();//日志相关的辅助类
     mMessager.printMessage(Diagnostic.Kind.NOTE, charSequence: "processing...");
    mProxyMap.clear();
     Set<? extends Element> elements = roundEnvironment.getElementsAnnotatedWith(BindView.class);
     for (Element element : elements) {
         VariableElement variableElement = (VariableElement) element;
         TypeElement classElement = (TypeElement) variableElement.getEnclosingElement();
         String fullClassName = classElement.getOualifiedName().toString();
        ClassCreatorProxy proxy = mProxyMap.get(fullClassName);
         if (proxy == null) {
            proxy = new ClassCreatorProxy(mElements, classElement);
            mProxyMap.put(fullClassName, proxy);
         BindView bindAnnotation = variableElement.qetAnnotation(BindView.class);
         BindView bindAnnotation = variableElement.getAnnotation(BindView.class);
         int id = bindAnnotation.value();
         proxy.putElement(id, variableElement);
      //通过javapoet生成
      for (String key : mProxyMap.keySet()) {
         ClassCreatorProxy proxyInfo = mProxyMap.get(key);
        JavaFile javaFile = JavaFile.builder(proxyInfo.getPackageName(), proxyInfo.generateJavaCode2()).build();
             javaFile.writeTo(processingEnv.getFiler());
         } catch (IOException e) {
            e.printStackTrace();
      mMessager.printMessage(Diagnostic.Kind.NOTE, charSequence: "process finish ...");
      return true;
```

ClassCreatorProxy 代码生成辅助类

build.gradle内容如下:

```
apply plugin: 'java'
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.google.auto.service:auto-service:1.0-rc2'
 implementation 'com.squareup:javapoet:1.7.0'
    implementation project(':annotation')
1}
tasks.withType(JavaCompile) {
    options.encoding = "UTF-8"
11
sourceCompatibility = "1.7"
targetCompatibility = "1.7"
com.squareup:javapoet:1.7.0 代码生成类
com.google.auto.service:auto-service:1.0-rc2 APT注解,自动执行类
如果gradle版本大于 Gradle Version 4.10.2,需要添加:
annotationProcessor 'com.google.auto.service:auto-service:1.0-rc2'
tasks.withType(JavaCompile) {
    options.encoding = "UTF-8"
    使用UTF-8格式,一般不需要
```

apt-library 工具包,提供给app调用生成代码的方法

```
📷 apt-library
                                    || iate 2018/4/22
▶ ■ build
▼ 🖿 src
                                     .ic class BindViewTools {
   ▼ Im main
                                    public static void bind(Activity activity) {
       ▼ 🛅 com.example.gavin.a
            © № BindViewTools 17
                                         Class clazz = activity.getClass();
                                          Class bindViewClass = Class.forName(clazz.getName() + "_ViewBinding");
  gitignore
                                             Log.e( tag: "lpf--", msg: "bindViewClass---"+bindViewClass);
                                      Method method = bindViewClass.getMethod ( name: "bind", activity.getClass()
  build.gradle
                                             method.invoke(bindViewClass.newInstance(), activity);
  proguard-rules.pro
                                         } catch (ClassNotFoundException e) {
build
                                             e.printStackTrace();
gradle
                                         } catch (IllegalAccessException e) {
gitignore build.gradle
                                             e.printStackTrace();
```

使用反射调用ViewBinding类中的方法

app为项目,调用APT方;



build.gradle内容如下:

```
▼ com.sninnoan 22
                                 implementation fileTree(dir: 'libs', include: [
          C Main2A 23
                                 implementation 'com.android.support:appcompat-v'
         C MainAc 24
                                 implementation 'com.android.support.constraint:
               25
                                 testImplementation 'junit:junit:4.12'
     AndroidManifest. 26
                                 androidTestImplementation 'com.android.support.
▶ ■ test
                                 androidTestImplementation 'com.android.support.
gitignore ...
                                 implementation project(':annotation')
                    28
📭 app.iml
                    29
                                 implementation project(':apt-library')
📀 build.gradle
                   30
                                 annotationProcessor project(':apt')
proguard-rules.pro
```

引入APT

最后生成的代码目录:

