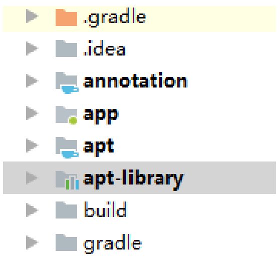
APT

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：



```
▶ .gradle
▶ .idea
▶ annotation
▶ app
▶ apt
▶ apt-library
▶ build
▶ gradle
```

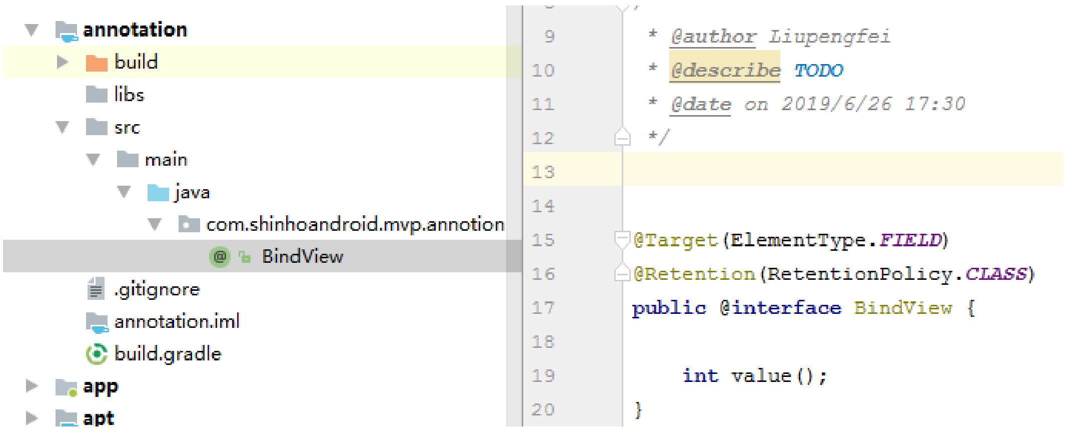
app为项目，调用APT方；

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



```
▼ annotation
    ▶ build
      libs
    ▼ src
        ▼ main
            ▼ java
                ▼ com.shinhoandroid.mvp.annotion
                      BindView
      .gitignore
      annotation.iml
      build.gradle
▶ app
▶ apt
```

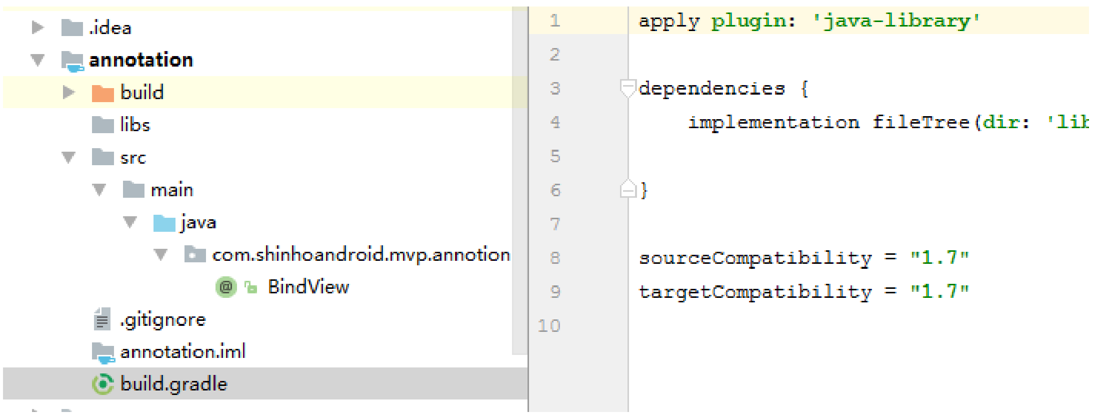
```java
 * @author Liupengfei
 * @describe TODO
 * @date on 2019/6/26 17:30
 */


@Target(ElementType.FIELD)
@Retention(RetentionPolicy.CLASS)
public @interface BindView {

    int value();
}
```

`@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`

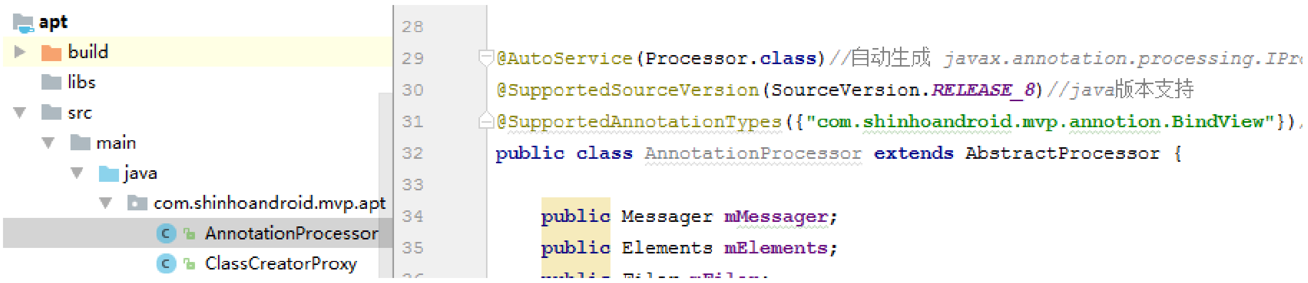
build.gradle内容如下：



```
▶ .idea
▼ annotation
    ▶ build
      libs
    ▼ src
        ▼ main
            ▼ java
                ▼ com.shinhoandroid.mvp.annotion
                      BindView
      .gitignore
      annotation.iml
      build.gradle
```

```groovy
apply plugin: 'java-library'

dependencies {
    implementation fileTree(dir: 'lib

}

sourceCompatibility = "1.7"
targetCompatibility = "1.7"

```

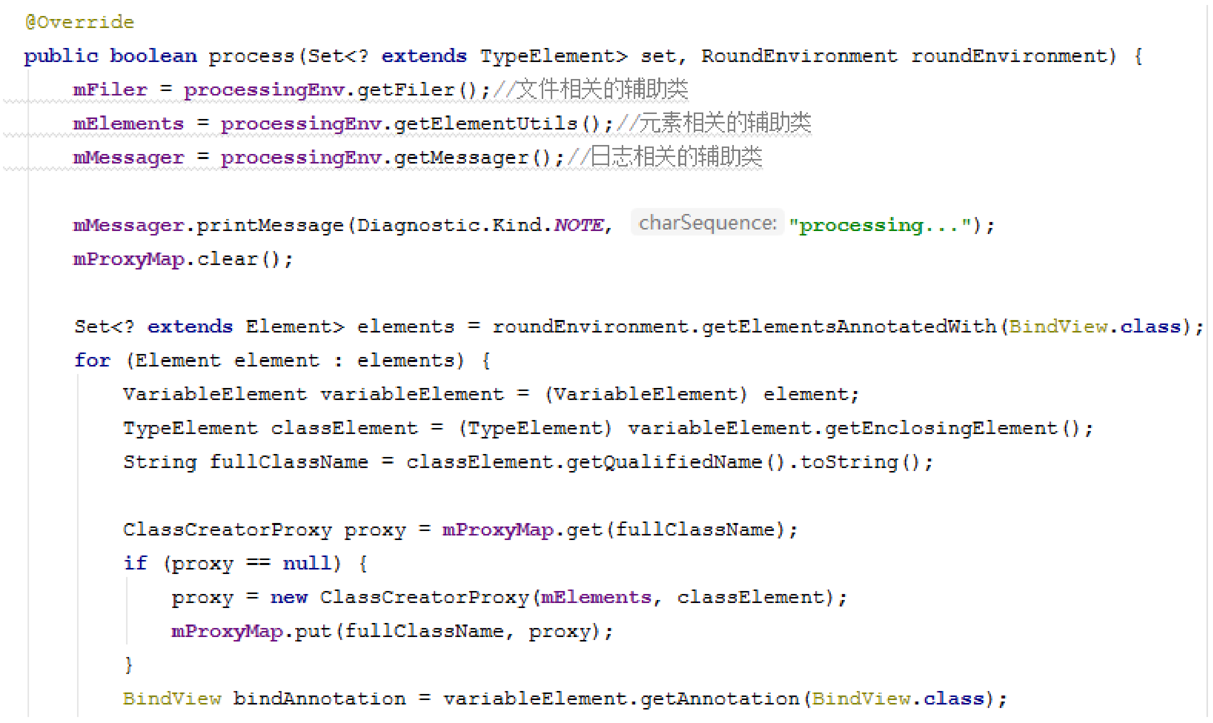
apt 动态生成代码



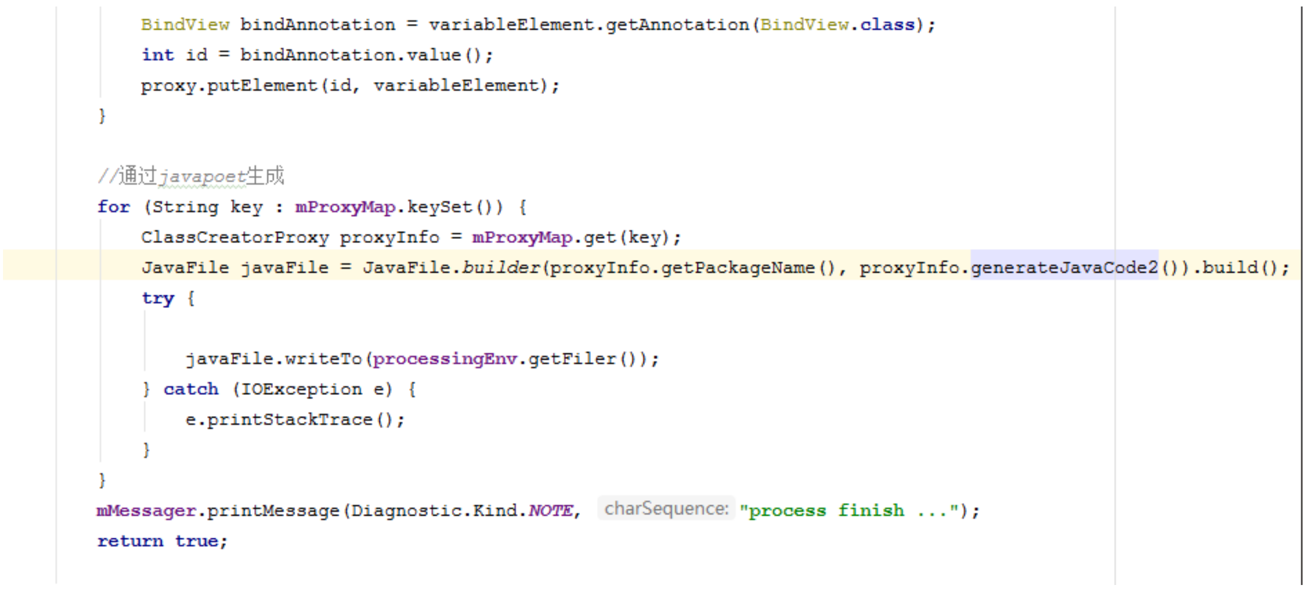
```
@AutoService(Processor.class)//自动生成 javax.annotation.processing.IPr
@SupportedSourceVersion(SourceVersion.RELEASE_8)//java版本支持
@SupportedAnnotationTypes({"com.shinhoandroid.mvp.annotion.BindView"}),
public class AnnotationProcessor extends AbstractProcessor {

    public Messager mMessager;
    public Elements mElements;
```

```
@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



```
@Override
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.getQualifiedName().toString();

        ClassCreatorProxy proxy = mProxyMap.get(fullClassName);
        if (proxy == null) {
            proxy = new ClassCreatorProxy(mElements, classElement);
            mProxyMap.put(fullClassName, proxy);
        }
        BindView bindAnnotation = variableElement.getAnnotation(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();
        try {

            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

```
     */
    public TypeSpec generateJavaCode2() {
        TypeSpec bindingClass = TypeSpec.classBuilder(mBindingClassName)
                .addModifiers(Modifier.PUBLIC)
                .addMethod(generateMethods2())
                .build();
        return bindingClass;

    }
```

build.gradle内容如下:

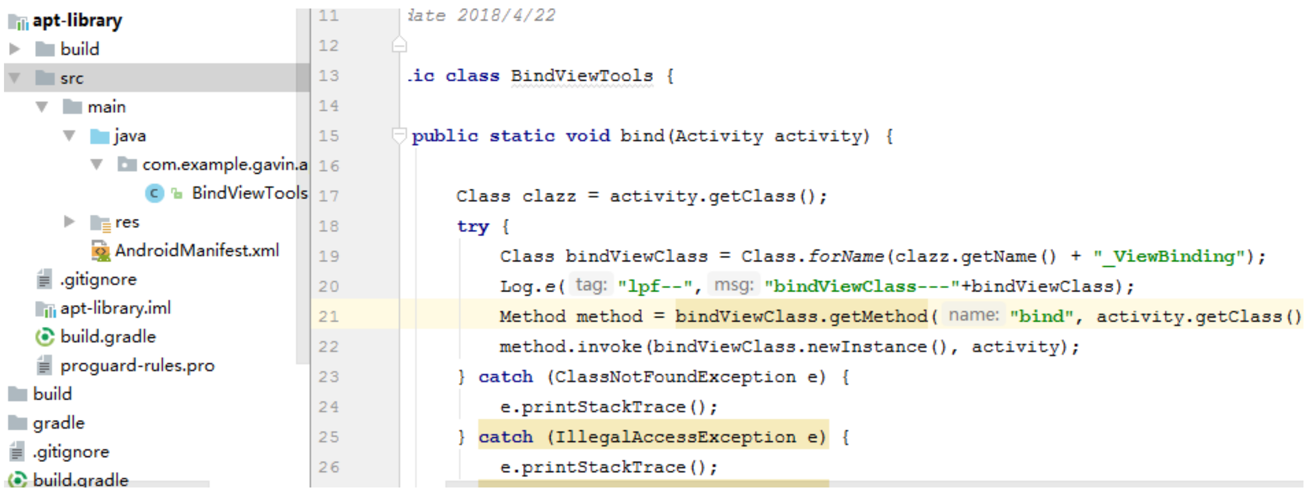
```
apply plugin: 'java'

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')
}
tasks.withType(JavaCompile) {
    options.encoding = "UTF-8"
}

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

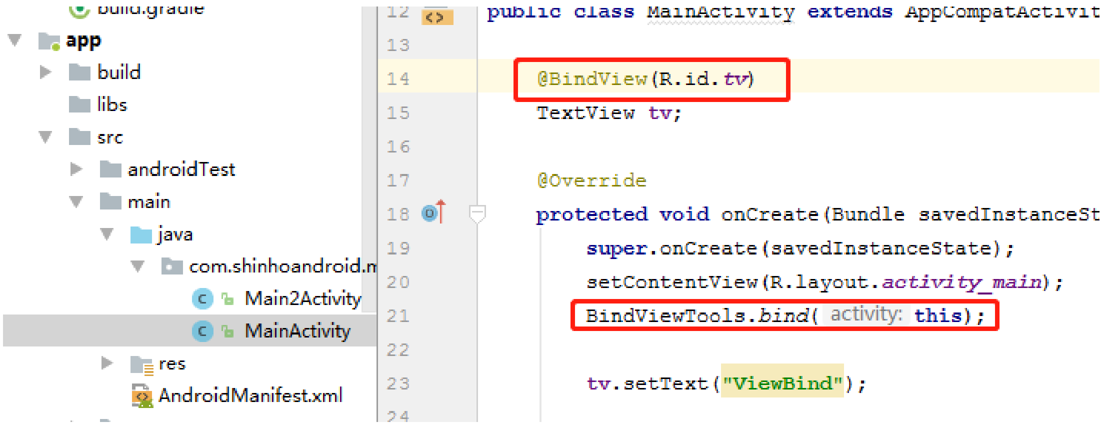
```
tasks.withType(JavaCompile) {
    options.encoding = "UTF-8"
}
```
使用UTF-8格式，一般不需要

apt-library 工具包，提供给app调用生成代码的方法



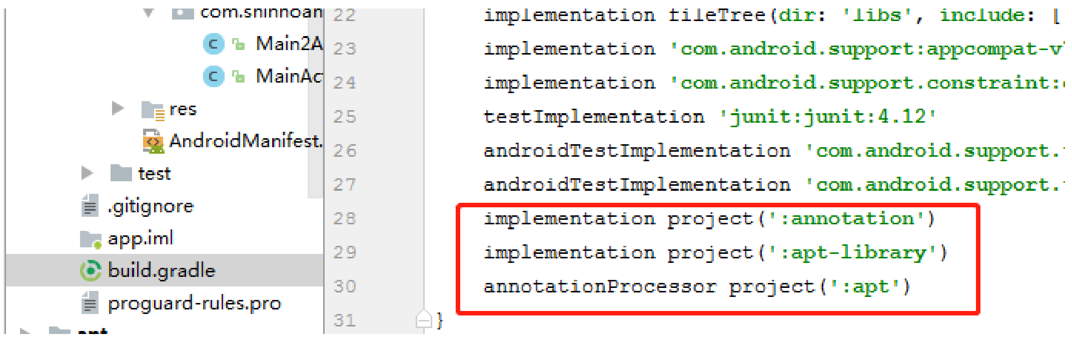
```
date 2018/4/22

.ic class BindViewTools {

 public static void bind(Activity activity) {

     Class clazz = activity.getClass();
     try {
         Class bindViewClass = Class.forName(clazz.getName() + "_ViewBinding");
         Log.e( tag: "lpf--", msg: "bindViewClass---"+bindViewClass);
         Method method = bindViewClass.getMethod( name: "bind", activity.getClass()
         method.invoke(bindViewClass.newInstance(), activity);
     } catch (ClassNotFoundException e) {
         e.printStackTrace();
     } catch (IllegalAccessException e) {
         e.printStackTrace();
```
使用反射调用ViewBinding类中的方法

app为项目，调用APT方;



```
public class MainActivity extends AppCompatActivit

    @BindView(R.id.tv)
    TextView tv;

    @Override
    protected void onCreate(Bundle savedInstanceSt
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        BindViewTools.bind( activity: this);

        tv.setText("ViewBind");
```

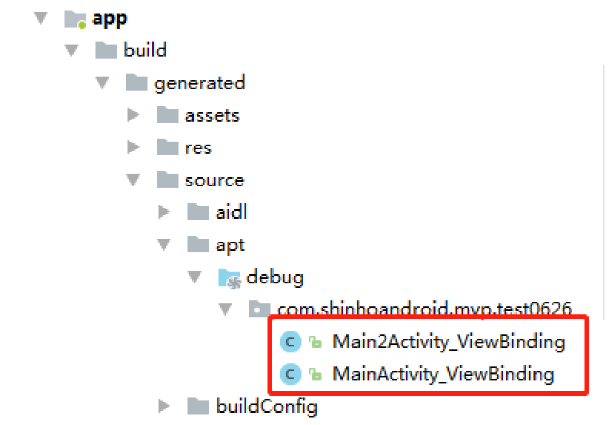
build.gradle内容如下：



```
    implementation fileTree(dir: 'libs', include: [
    implementation 'com.android.support:appcompat-v
    implementation 'com.android.support.constraint:
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.
    androidTestImplementation 'com.android.support.
    implementation project(':annotation')
    implementation project(':apt-library')
    annotationProcessor project(':apt')
}
```

引入APT

最后生成的代码目录：



app
build
generated
assets
res
source
aidl
apt
debug
com.shinhoandroid.mvp.test0626
Main2Activity_ViewBinding
MainActivity_ViewBinding
buildConfig