HenCoder Plus 第 33 课 讲义

Annotation Processing

用反射实现 ButterKnife

- Bind class
- bind(Activity) method
- 用反射获取 Field[]，然后获取 Annotation BindView

插播：ButterKnife 是依赖注入吗?

- dagger 是依赖注入
- ButterKnife 轻量级依赖注入?
- 什么是依赖注入：把依赖的决定权交给外部，即依赖注入
- ButterKnife：自己决定依赖的的获取，只把执行过程交给 ButterKnife
- 所以：ButterKnife 只是一个 View Binding 库，而不是依赖注入

Annotation Processing

- 理解 Annotation Processing 的原理：编译过程中读源码，然后生成代码，再编译
- 举例：

```
public class MainActivity$Binding {
  public MainActivity$Binding(MainActivity
activity) {
    activity.textView =
activity.findViewById(R.id.textView);
  }
}
```

```
public class Binding {
    public static void bind(Activity activity) {
        try {
            Class bindingClass =
Class.forName(activity.getClass().getCanonicalNa
me() + "$Binding");
            Constructor constructor =
bindingClass.getDeclaredConstructor(Class.forNam
e(activity.getClass().getCanonicalName()));
            constructor.newInstance(activity);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }
    }
}
```

- Annotation Processing 的目的：自动生成这部分代码

用 Annotation Processing 实现 ButterKnife

- Annotation Processing 用法：
    - resources/META-INF/services/javax.annotation.processing.Processor
    - 继承 AbstractProcessor
    - 重写 getSupportedAnnotationTypes() 和 process()
        - annotaions: 程序中出现的已注册的 Annotations；roundEnv：各个 java 文件
    - 依赖：annotationProcessor
    - 先测试生成 java 文件的功能：
        - javapoet
        - 代码：

```
ClassName className =
ClassName.get("com.hencoder.a25", "Test");
TypeSpec builtClass =
TypeSpec.classBuilder(className).build();
JavaFile.builder("com.hencoder.a25",
builtClass)
    .build
    .writeTo(filer);
```

```
ClassName className =
ClassName.get("com.hencoder.a25",
"MainActivity$Binding");
        TypeSpec builtClass =
TypeSpec.classBuilder(className)

.addModifiers(Modifier.PUBLIC)

.addMethod(MethodSpec.constructorBuilder()

.addModifiers(Modifier.PUBLIC)

.addParameter(ClassName.get("com.hencoder.
a25", "MainActivity"), "activity")

.addStatement("activity.textView =
activity.findViewById(R.id.textView)")
                        .build())
                .build();
        try {

 JavaFile.builder("com.hencoder.a25",
builtClass)
            .build().writeTo(filer);
```

```
          } catch (IOException e) {
              e.printStackTrace();
          }
```

- 自动生成代码：
  - 需要把 Annotation 单独拆成一个 java lib module，被主项目和 processor 分别依赖

```
for (Element element :
roundEnv.getRootElements()) {
            String packageStr =
element.getEnclosingElement().toString();
            String classStr =
element.getSimpleName().toString();
            ClassName className =
ClassName.get(packageStr, classStr +
"$Binding");
            MethodSpec.Builder
constructorBuilder =
MethodSpec.constructorBuilder()

.addModifiers(Modifier.PUBLIC)

.addParameter(ClassName.get(packageStr,
classStr), "activity");
            boolean hasBinding = false;

            for (Element enclosedElement :
element.getEnclosedElements()) {
                BindView bindView =
enclosedElement.getAnnotation(BindView.class);
                if (bindView != null) {
                    hasBinding = true;
```

```
 constructorBuilder.addStatement("activity.$N =
activity.findViewById($L)",

 enclosedElement.getSimpleName(),
bindView.value());
                }
            }

            TypeSpec builtClass =
TypeSpec.classBuilder(className)

.addModifiers(Modifier.PUBLIC)

.addMethod(constructorBuilder.build())
                    .build();

            if (hasBinding) {
                try {
                    JavaFile.builder(packageStr,
builtClass)

.build().writeTo(filer);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
```

- 还需要一个 lib module，依赖 annotation，把 bind 那些东西写在这里。主项目依赖 lib，lib 依赖 annotations。最终主项目中有两个依赖：lib 和 processor

- 内部类的问题
  - 使用 getElementsAnnotatedWith() 来做