HenCoder Plus 讲义

Annotation Processing

用反射实现 ButterKnife

- 创建 `@BindView` 注解
- 用反射获取 Field[]，然后找到加了 `@BindView` 的字段，自动调用 `findViewById()` 来绑定对象

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
- Dagger：外部的依赖图来决定依赖的值，对象自己只负责「索要」，而不负责指定值，所以 Dagger 是依赖注入
- ButterKnife：自己决定依赖的的获取，只把执行过程交给 ButterKnife，所以只是一个视图绑定库，而不是依赖注入

Annotation Processing

- 理解 Annotation Processing 的原理：编译过程中读源码，然后生成新的代码文件，再放在一起进行编译
- 例如：

```
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().getCanonicalName()
+ "$Binding");
            Constructor constructor =
bindingClass.getDeclaredConstructor(Class.forName(act
ivity.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.apt", "Test");
TypeSpec builtClass =
TypeSpec.classBuilder(className).build();
JavaFile.builder("com.hencoder.apt",
builtClass)
    .build
    .writeTo(filer);
```

```
ClassName className =
ClassName.get("com.hencoder.apt",
"MainActivity$Binding");
        TypeSpec builtClass =
TypeSpec.classBuilder(className)
                .addModifiers(Modifier.PUBLIC)

.addMethod(MethodSpec.constructorBuilder()

.addModifiers(Modifier.PUBLIC)

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

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

 JavaFile.builder("com.hencoder.apt",
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

问题和建议?

课上技术相关的问题，都可以去群里和大家讨论，对于比较通用的、有价值的问题，可以去我们的知识星球提问。

具体技术之外的问题和建议，都可以找丢物线（微信：diuwuxian），丢丢会为你解答技术以外的一切。

觉得好?

如果你觉得课程很棒，欢迎给我们好评呀！https://ke.qq.com/comment/index.html?cid=381952

> 一定要是你真的觉得好，再给我们好评。不要仅仅因为对扔物线的支持而好评（报名课程已经是你最大的支持了，再不够的话 B 站多来点三连我也很开心），另外我们也坚决不做好评返现等任何的交易。我们只希望，在课程对你有帮助的前提下，可以看到你温暖的评价。

更多内容：

- 网站：https://hencoder.com；https://kaixue.io
- 各大搜索引擎、微信公众号、微博、知乎、掘金、哔哩哔哩、YouTube、西瓜视频、抖音、快手、微视：统一账号「**扔物线**」，我会持续输出优质的技术内容，欢迎大家关注。
- 哔哩哔哩快捷传送门：https://space.bilibili.com/27559447

大家如果喜欢我们的课程，还请去扔物线的哔哩哔哩，帮我素质三连，感谢大家！