HenCoder Plus 第 28 课 讲义

Annotation Processing、简历与面试

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().getCanonicalName() + "$Binding");
            Constructor constructor =
bindingClass.getDeclaredConstructor(Class.forName(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

简历

技能：要足够具体，让面试官对你真正快速了解，也让面试官有内容可问

- 反例：熟悉 HTTP、熟悉自定义 View、熟悉多线程
- 正例：
    - 掌握 HTTP 报文格式，了解请求行、状态行、Header、Body 的格式和作用，了解 GET、POST、PUT、DELETE、HEAD 方法的作用和区别，了解常见状态码如 200、301、302、307、404、500 的含义
    - 了解 HTTPS 的工作方式，熟悉对称加密、非对称加密、数字签名的含义和区别，了解 HTTPS 的连接建立过程
    - 了解 Cookie 和 Authorization Header 两种登录方式的工作原理、使用方式、历史背景和区别
    - 熟悉 OAuth2 的工作原理、安全原理和几种不同的工作方式，以及 OAuth2 在 Android 上的实现方式
    - 阅读过并掌握 Retrofit 和 OkHttp 的源码，熟悉它们的核心结构、工作原理，清楚了解这二者各自和 HTTP 的关系，以及它们在 Android 开发中的角色和它们之间的关系

面试前

准备一份可见的面试作品

- 最早：写博客加分
- 博客泛滥后：有 GitHub 加分
- GitHub 泛滥后：有星星多的开源项目加分；为著名项目做过贡献加分
- 有已发布的个人作品的永远加分

回顾自己做过的项目

- 项目大致架构怎样
- 有哪些难点，这些难点的解决方案（重要）
    - 万年难题：你在项目中遇到过印象最深的问题是什么？你是怎么解决的？

提前了解面试的公司和面试官

- 下他们的软件用用

- 针对对方的应用中的优点提一些问题
    - 这个是怎么做的?
    - 这个是不是怎么做的?

面试

太宽泛的问题?

你说你擅长 XXX，讲一下吧

- 先讲框架，然后抓住具体的点来讲细节

如果问到了不会的问题

- 这个我不会 / 了解的不多，能换一个吗
- 我比较擅长 XXX 和 YYY 和 ZZZ，可以问一下这些吗

面试官的想法（不知所措的时候就想想）

- 他的水平是不是和简历的描述符合？我要验证
- 简历上一些比较模糊的描述，具体是怎样的？我要考察
- 他的能力是否符合我们的需求？我要问一下

面试之外

- 如果面试情况很差，记得别影响心情，因为有些面试官并没有给你足够好的展示机会
- 如果多次面试都情况很差？你确定你学完我的课程了吗?