**APT在mvp中的实践**

|  |  |
| --- | --- |
| **DOCUMENT ID: APT** | |
| **PROJECT:** | **RELEASE:** |
| **FEATURE:** | **SUBSYSTEM: Scenarios** |
| **DISTRIBUTE TO:** | |

目录

Section 1: 背景 1-4

Section 2: 如何使用APT 2-6

1 新建两个Java Lib 2-6

2 注解处理器 2-6

3 集成APT 2-7

4 生成及调试 2-7

Section 3: 经验总结 3-7

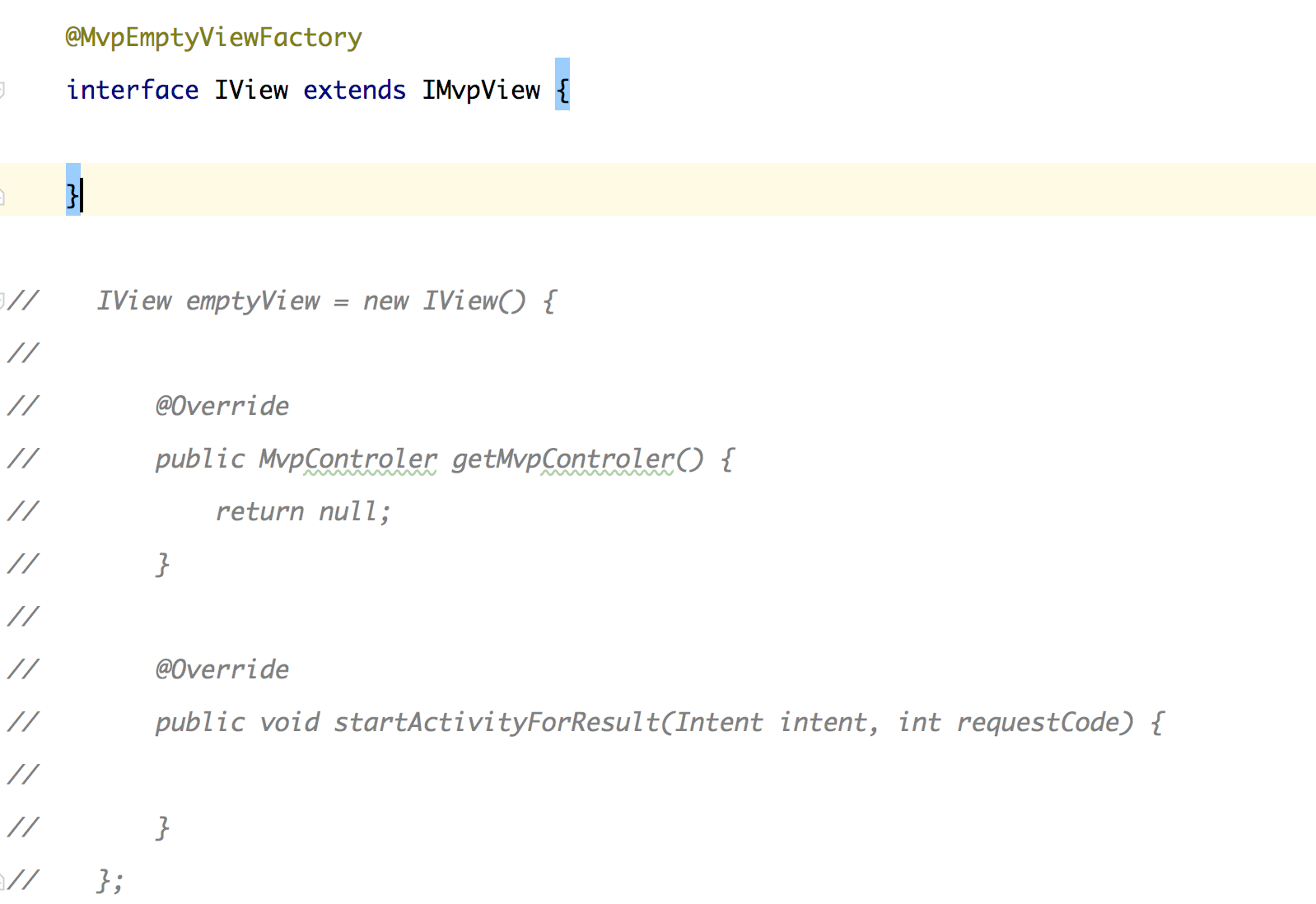
1 适用场景 3-7

2 建议 3-7

# 背景

主要为了解决客户端MVP架构中，V层和P层生命周期不同步时，生成空View保护性逻辑。

之前需要手写空View的代码，现在通过注解配置可自动生成，同理于黄油刀的@ BindView 注解。



通过配置该注解，在编译期 （compileDebugJavaWithJavac）会自动生成Java类。

# 如何使用APT

## 1 新建两个Java Lib

第一个annotation 专门存放编译期注解。

第二个 apt 专门存放生成这个Java代码的注解处理器，并在Gradle添加两个外部包和自定义注解依赖。

implementation **'com.squareup:javapoet:1.9.0'**implementation **'com.google.auto.service:auto-service:1.0-rc3'**implementation project(**':annotation'**)

分别是配合apt便捷生成java文件的工具及特定路径下生成配置文件。

## 2 注解处理器

继承AbstractProcessor并复写process方法，同时添加下图三个注解。



@AutoService(Processor.class)注解，这个注解会自动在指定路径下生成一个配置文件：

@SupportedAnnotationTypes注解，配置这个类所要处理的注解类型。(传入String类型参数，格式为：包名+类名);

@SupportedSourceVersion(SourceVersion.RELEASE\_8)注解，JDK版本

**MvpEmptyViewProcessor MVP空View Java代码生成器**

这里的代码主要是用javapoet 这个库的语法通过该注解拿到的信息生成Java类。

大致逻辑：

1. 获取被编译期注解修饰的类信息。
2. 获取该类及父类结构里方法、参数、返回值信息。
3. 根据这些信息生成该类的匿名对象。

<https://github.com/square/javapoet>

## 3 集成APT

项目Gradle 文件 依赖上述两个Java Lib。

implementation project(':annotation')

kapt/ annotationProcessor project(':apt') (要看项目中是否有用 apply plugin: 'kotlin-kapt')

## 4 生成及调试

在想要生成的地方用编译期注解进行修饰，然后Rebuild项目。

APT 调试 <https://blog.csdn.net/jonch_hzc/article/details/78796592>

# 经验总结

## 适用场景

逻辑比较简单的场景，如逻辑较为复杂定位问题比较麻烦 其次 会影响编译速度。

## 建议

可以看看黄油刀的源码，对更深层次的理解APT有很大帮助。