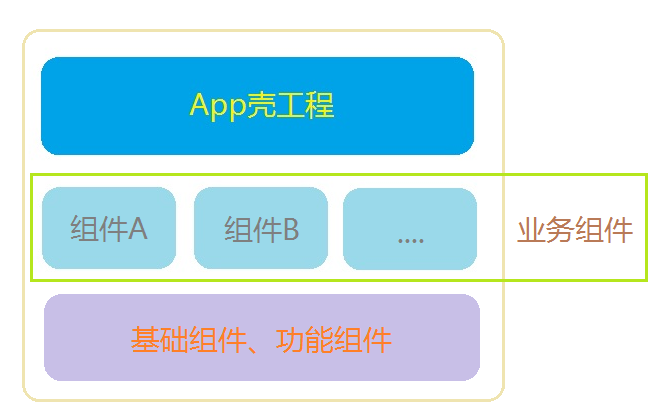
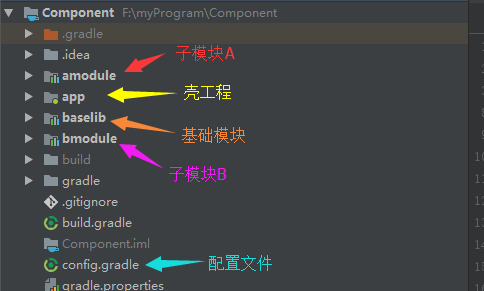
ComponentDemo说明

1. 设计思想

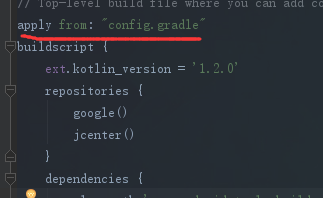


1. 模块说明

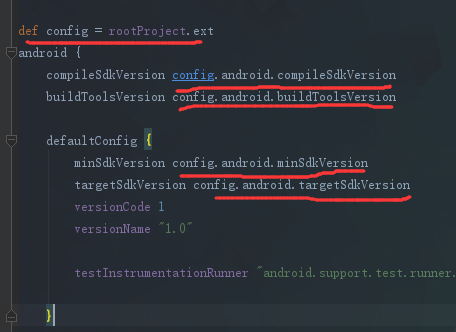


1. 创建流程：
2. 添加config.gradle

拷贝config.gradle到project目录下，并在project的bulid.gradle添加依赖。如图



1. 导入基础库baselib(可选择kotlin版或java版，此文档仅介绍kotlin版，java版大同小异)
2. 创建各个module，并添加依赖baselib（如project中的AModule和BModule）,App壳工程添加各module依赖（无需再次添加baselib，因为其他module已经添加过）
3. 统一各module版本配置，参考各个module的build.gradle,如图：



因baselib已添加appcompat包，所以可将各个组件及壳工程的implementation 'com.android.support:appcompat-v7:\*\*’删除

1. 添加dagger2依赖

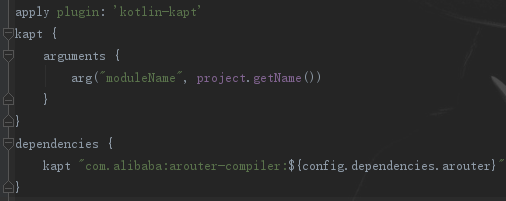
在各module（app壳工程除外）的build.gradle的头部添加

apply plugin: 'kotlin-kapt'， **dependencies**添加

kapt "com.google.dagger:dagger-compiler:${config.dependencies.dagger}"

1. 添加Arouter

在各module的build.gradle的添加



初始化sdk：

if (isDebug()) { // 这两行必须写在init之前，否则这些配置在init过程中将无效

ARouter.openLog(); // 打印日志

ARouter.openDebug(); // 开启调试模式(如果在InstantRun模式下运行，必须开启调试模式！线上版本需要关闭,否则有安全风险)

}

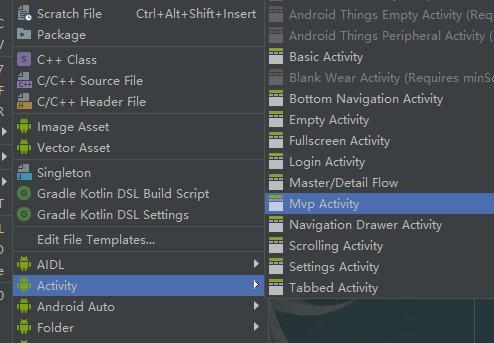
ARouter.init(mApplication); // 尽可能早，推荐在Application中初始化

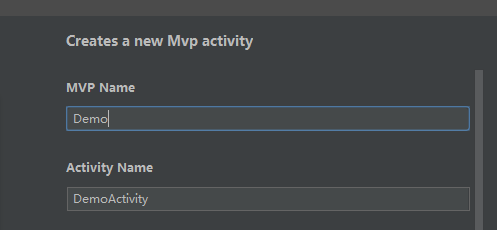
详见：<https://github.com/alibaba/ARouter>

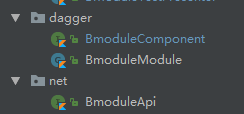
1. Application初始化调用问题

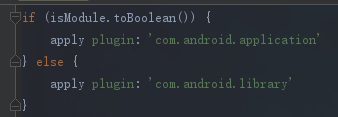
子module实现ApplicationImpl方法，通过反射方法在baselib工程的Application调用各module的moduleApplicationImpl().onCreate()方法 (需添加类路径)

1. 创建Activity模板：
2. 找到Android studio的安装路径，并打开Activity模板文件夹，如我的路径为D:\android-studio\plugins\android\lib\templates\activities
3. 解压附带的压缩包，并拷贝MVP模板到该文件夹下
4. 重启Android studio
5. 右键点击目录选择创建Activity即可按照模板生成代码，如图





1. 实现dagger2注入
2. 实现配置：
3. 点击AS工具类Build->make project生成代码
4. 在所需Activity中添加注入代码
5. 多人开发协作问题：
6. Application 与 Library相互切换
7. 在project的gradle.properties添加isModule = false
8. 在组件module的build进行判断，如下图



1. 匹配对应的AndroidManifest.xml

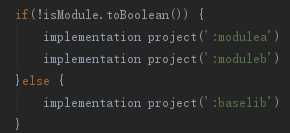
在组件module设置路径，如图：



并在src/main路径下创建module文件夹，添加AndroidManifes.xml

1. App壳工程配置

在build.gradle的dependencies修改依赖，如图：



1. 当单独开发业务模块时，将gradle.properties下的isModule 值改成true，合并工程时改成false
2. 注意事项：
3. 混淆时需忽略ApplicationImpl的实现类，因用到反射获取方法
4. 添加业务module时，需在baselib的config/ModuleConfig配置ApplicationImpl的具体路径