Android彻底组件化方案

移动开发团队： 曾祥辉

目录

content

什么是组件化?

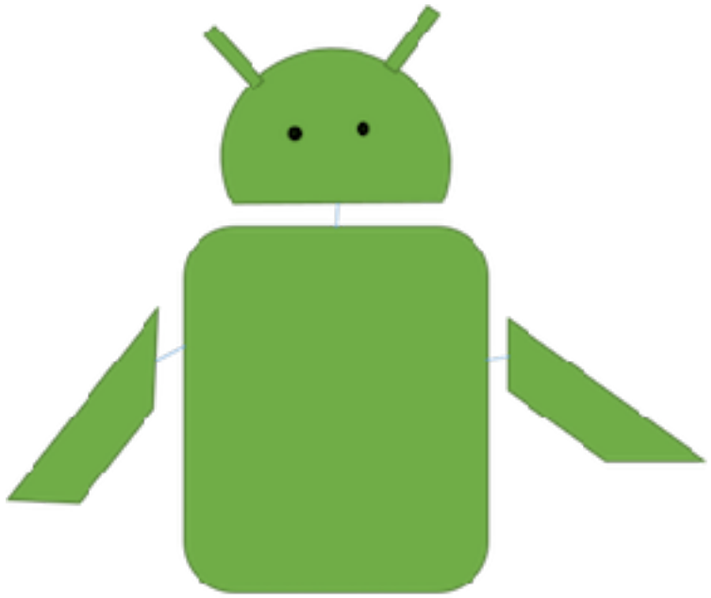
模块化、组件化与插件化的关系

- **模块化**是指解决一个复杂问题时自顶向下逐层把系统划分成若干模块(Module)的过程。

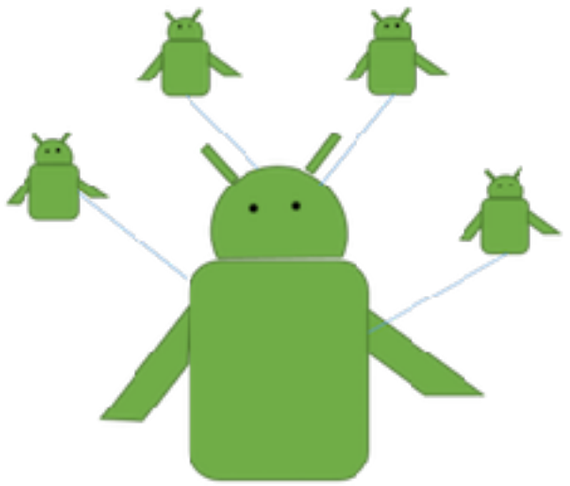
- 在技术开发领域，模块化是指代码分拆，分而治之，解耦分层的一种过程。

- 具体到android开发领域，模块化的具体实现方法，可分为组件化与插件化。

模块化是一种代码分而治之的指导理念，而组件化与插件化则是两种具体的实现途径！



组件化



插件化

- 组件化是一个有机整体，需要完整功能的话，所有器官都应该存在；
- 实际上，只要有个入口，组件化中的每个器官都是可以单独存在的；
- 两者最大的区别在于插件化可以动态增加和修改线上的模块，**组件化的动态能力相对较弱，只能对线上已有模块进行动态的加载和卸载，不能新增和修改**。

组件化还是插件化?

- **重要原则**：是否有动态增加或修改线上模块的强需求？需求弱，则一般不需要考虑插件化。一般电商类或广告类产品这个需求较强烈！

- **三个权衡方面：**

1.目前主流的插件化框架不可避免地会对系统的api进行hook，兼容性方面(android碎片化严重)是个隐患，且需要专门人员去维护；

2.对繁杂业务进行拆分时间成本较大，要考虑开发节奏。

3.插件添加、修改存在时间或失败的风险，对于一些主要功能模块使用插件化，存在一定的风险。

组件化是对app代码根据某些特点的使用场景、功能或业务进行分拆、解耦成相互独立的组件的过程。

为什么要组件化?

之前：

年龄

2-6岁

症状

肥胖

危害

巨大开发包袱(会不会影响其它？？)

拖慢开发节奏(编译一次好久)

下班晚

目前财产险app采取的方案：

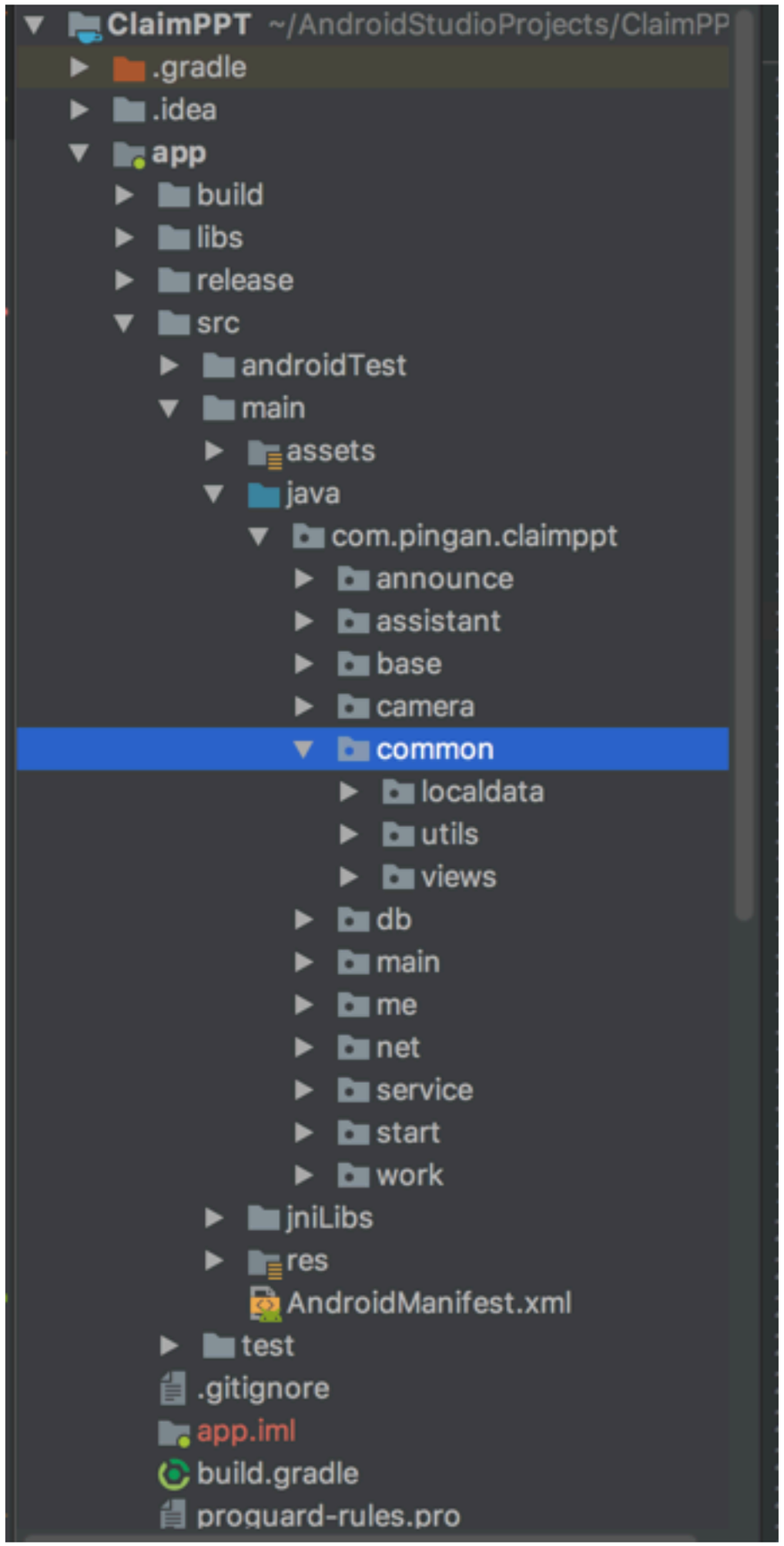
- 按功能分包，可以在一定程序隔离代码，独立业务、功能模块

存在的问题：

- 代码并没有做到完全的隔离，各业务模块之间代码还可以相互引用、跳转；

- 业务功能复杂时，编译打包时间会增长，拖慢开发节奏；

- 通用功能未独立成组件，app间资源共享不便；



组件化带来哪些好处?

开发

① 减轻开发包袱，再也不用担心所改的代码会影响到其它模块了；
② 单独调试，不用每次编译都耗费时间，拖慢开发节奏；
③ 利于团队协同开发，不用怕同事改到代码，影响到自己了；

测试

① 可以很牛逼的对测试同事说，这期就测修改的某个模块就可以了；

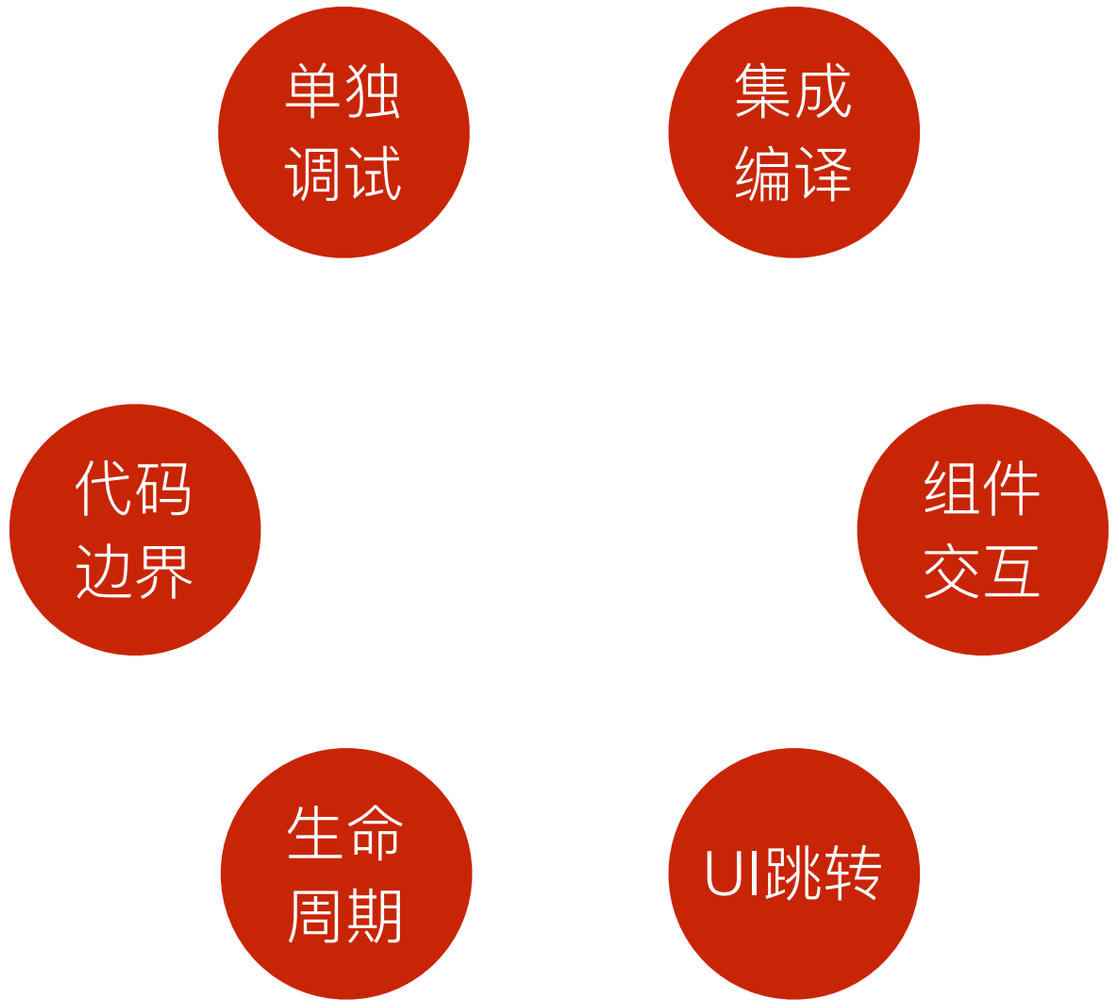
复用

① 其它app要用到某个功能组件时，可以很方便地分享给他们；
② 后面也可以建立与维护一个公共的组件库，资源共享，避免重复造轮子

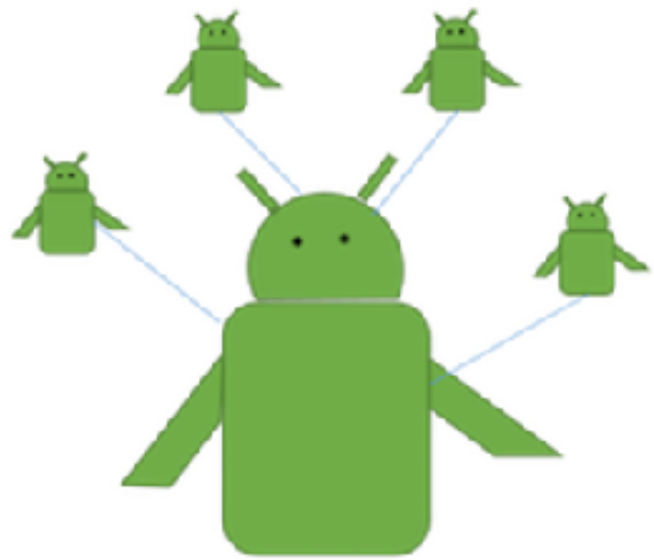
3

组件化的核心是什么?

彻底组件化方案要做到哪些?



单独调试



1. 组件需要怎么单独调试?

```
if(isRunAlone.toBoolean()){
     apply plugin: 'com.android.application'
}else{
     apply plugin: 'com.android.library'
}
```

```
combuild {
    applicationName = ‘com.pingan.claimppt.application.AppApplication'
    isRegisterCompoAuto = true
}
```

2. 如何避免资源冲突?

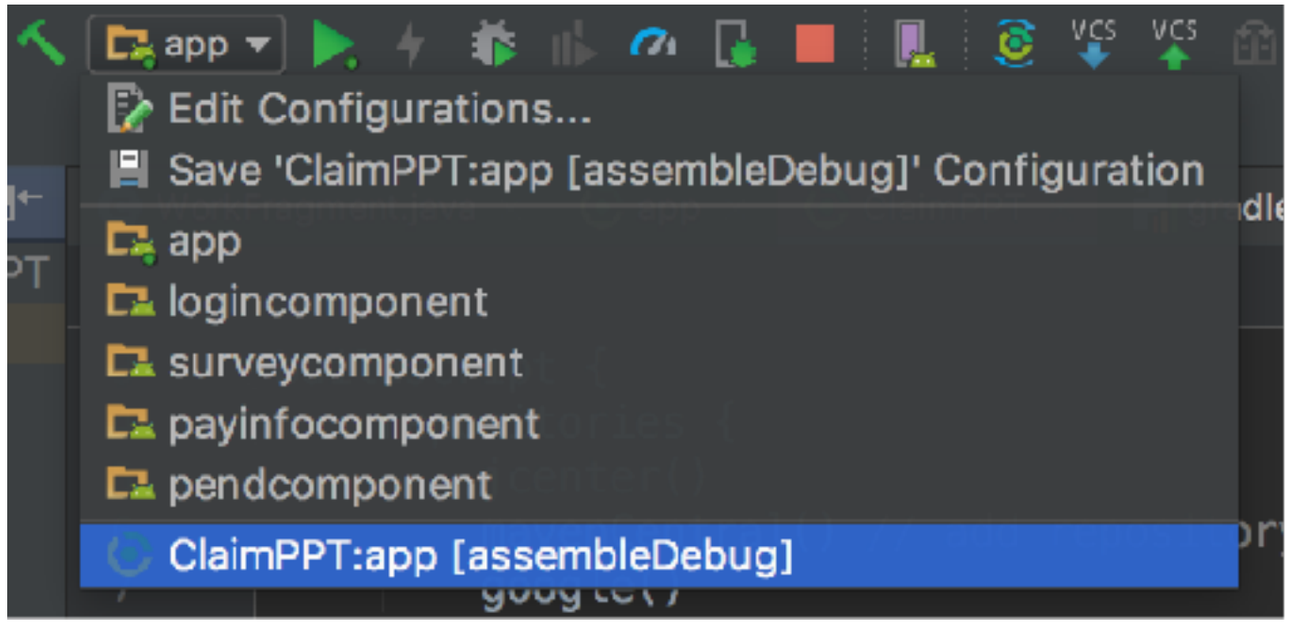
```
android {
   resourcePrefix “survey_"
}
```

集成编译

1.任何一个组件能否充当host?

2.组件由host切换到library是否可以无感知的完成?

3.是否可手动选择要集成的组件?



- Host的gradle.properties文件中，动态添加想要集成的组件

```
debugComponent=logincomponent,surveycomponent,pendcomponent
compileComponent=logincomponent,surveycomponent,pendcomponent
```

- 代码添加或删除相应的组件(反射)

```
Router.registerComponent(“com.pingan.claimppt.applike.SurveyApplike");
Router.unregisterComponent("com.pingan.claimppt.applike.SurveyApplike");
```

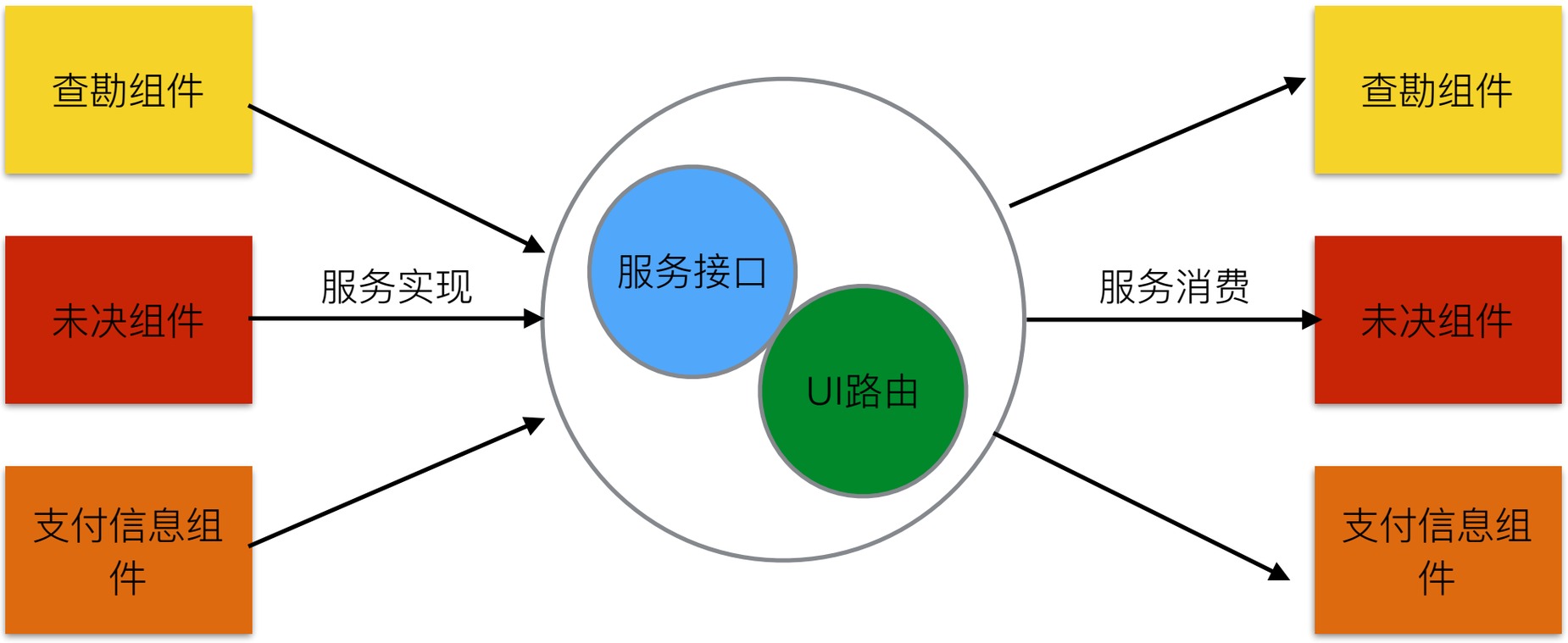
组件交互

1. 每个组件怎么提供服务

2. 怎么样做到更方便的服务发现?

3. 服务接口如何自动与其他代码剥离?

componentservice,暴露接口，组件实现，提供实现类，避免组件之间直接交互



UI跳转

1. 是否支持scheme跳转？三种跳转(bundle,uri,startActivityForResult)

```
<scheme>://<host>/<path>?<query>
```

```
DDComp://share/shareBook?bookName=Gone with the Wind
```

2. 路由和传递参数是否支持自动注解生成?

```
@RouteNode(path = "/shareBook", desc = "分享书籍页面")
public class ShareActivity extends AppCompatActivity {
```

3. 是否可以生成清晰的路由表?

```
auto generated, do not change !!!!

HOST : share

分享杂志页面
/shareMagazine
author:com.luojilab.componentservice.share.bean.Author
bookName:String

分享书籍页面
/shareBook
author:com.luojilab.componentservice.share.bean.Author
bookName:String
```

```
android {
    compileSdkVersion 26
    buildToolsVersion "26.0.1"
    defaultConfig {
        javaCompileOptions {
            annotationProcessorOptions {
                arguments = [host: "app"]
            }
        }
    }
```

```
annotationProcessor
'com.luojilab.ddcomponent:router-anno-compiler:
1.0.0'
```

组件的生命周期

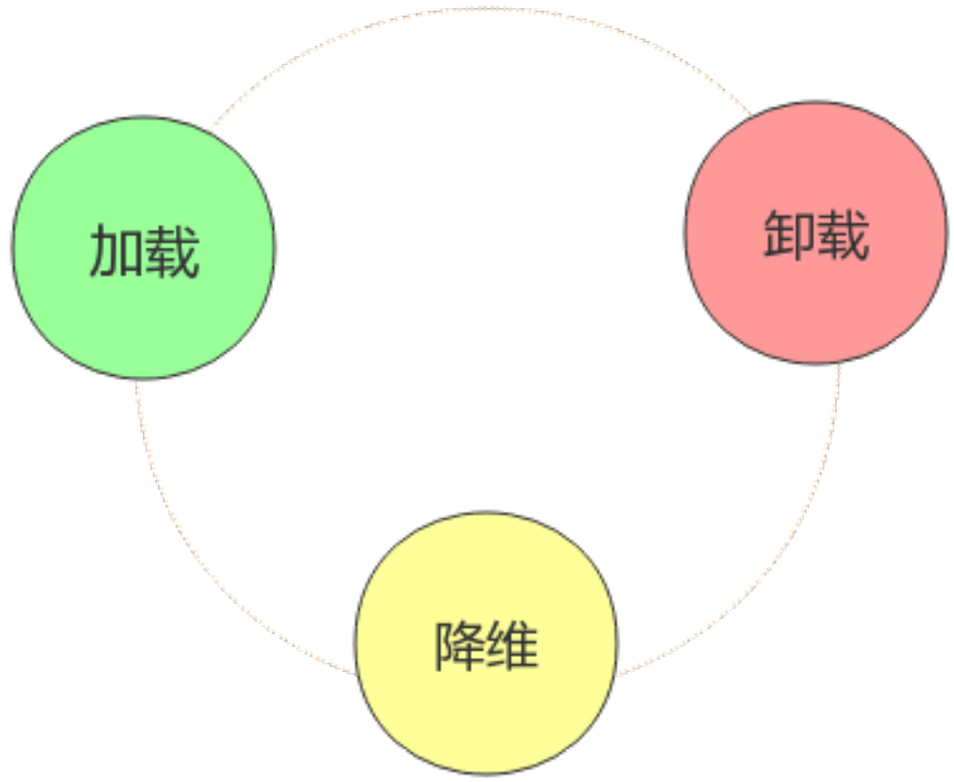
- 每个组件增加一个ApplicationLike类，里面定义了onCreate和onStop两个生命周期函数
- 由于主项目和组件之间是隔离的，那么主项目如何调用组件ApplicationLike的生命周期方法呢，目前我们采用的是基于编译期字节码插入的方式，扫描所有的ApplicationLike类（其有一个共同的父类），然后通过**javassist**在主项目的onCreate中插入调用ApplicationLike.onCreate的代码

Javassist是一个开源的分析、编辑和创建Java字节码的类库，直接使用java编码的形式，而不需要了解虚拟机指令，就能动态改变类的结构，或者动态生成类。

1. 运行时动态打开组件?

2. 运行时动态关闭组件?

3. 运行时将某个组件降维为H5?



代码边界

1. 如何做到代码隔离？

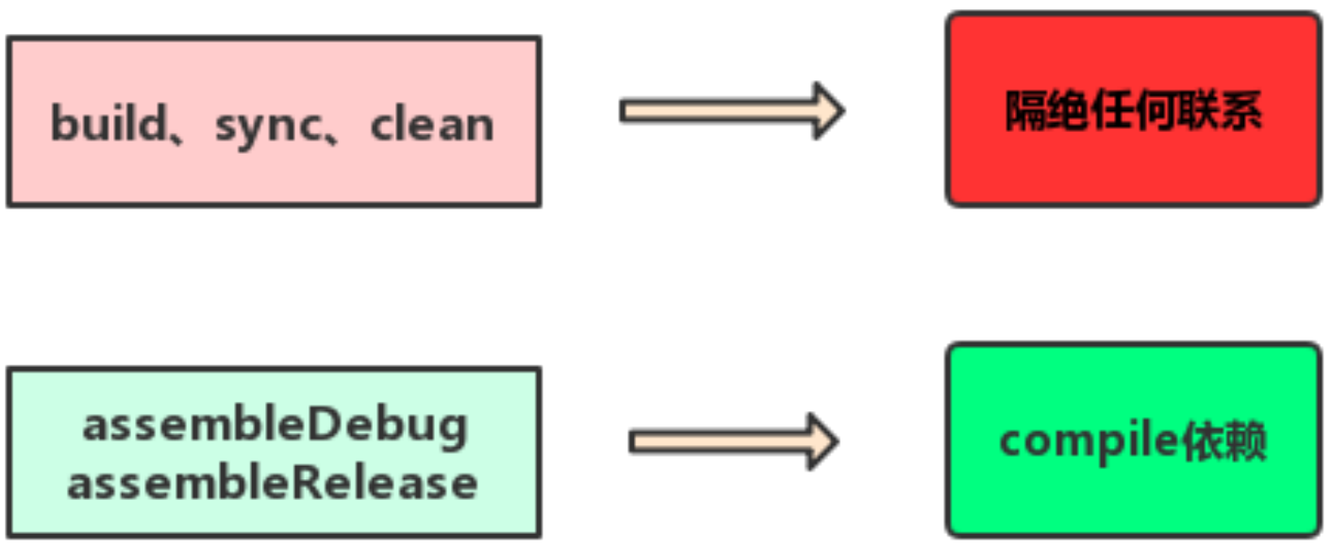
语法	旧语法	功能	支持类型	代码隔离效果
implementation	compile	编译期间对其他组件不可见，运行期间可见	jar aar	“隔代”编译期间隔离
api	compile	编译和运行期间都可见	jar aar	没有隔离
compileOnly	provided	只参与编译	jar	没有隔离
runtimeOnly	apk	编译期间不可见，运行期间可见	jar aar	编译期间隔离

2. 上面可以做到资源隔离？

NO！

代码边界

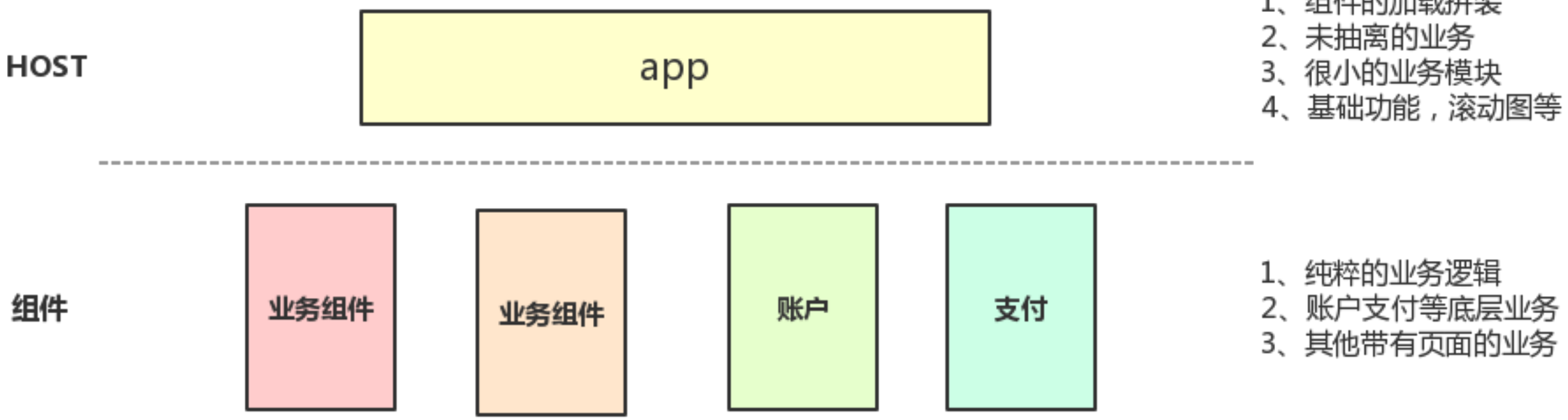
3. 可否做到编译期组件不可见，但同时全部组件 参与打包?



4

财产险app组件化尝试

划分项目


HOST
app
1、组件的加载拼装
2、未抽离的业务
3、很小的业务模块
4、基础功能，滚动图等
组件
业务组件
业务组件
账户
支付
1、纯粹的业务逻辑
2、账户支付等底层业务
3、其他带有页面的业务
服务发现
sdk
sdk
sdk
sdk
1、每个组件提供的服务
2、可以整合在一起
3、.api化自动生成
业务依赖库
basicRes
basicLib
1、业务需要的公共资源
2、公共代码和公共基类
依赖库
路由
网络
图片加载
热修复
工具封装
无埋点
1、业务封装依赖库
2、第三方库

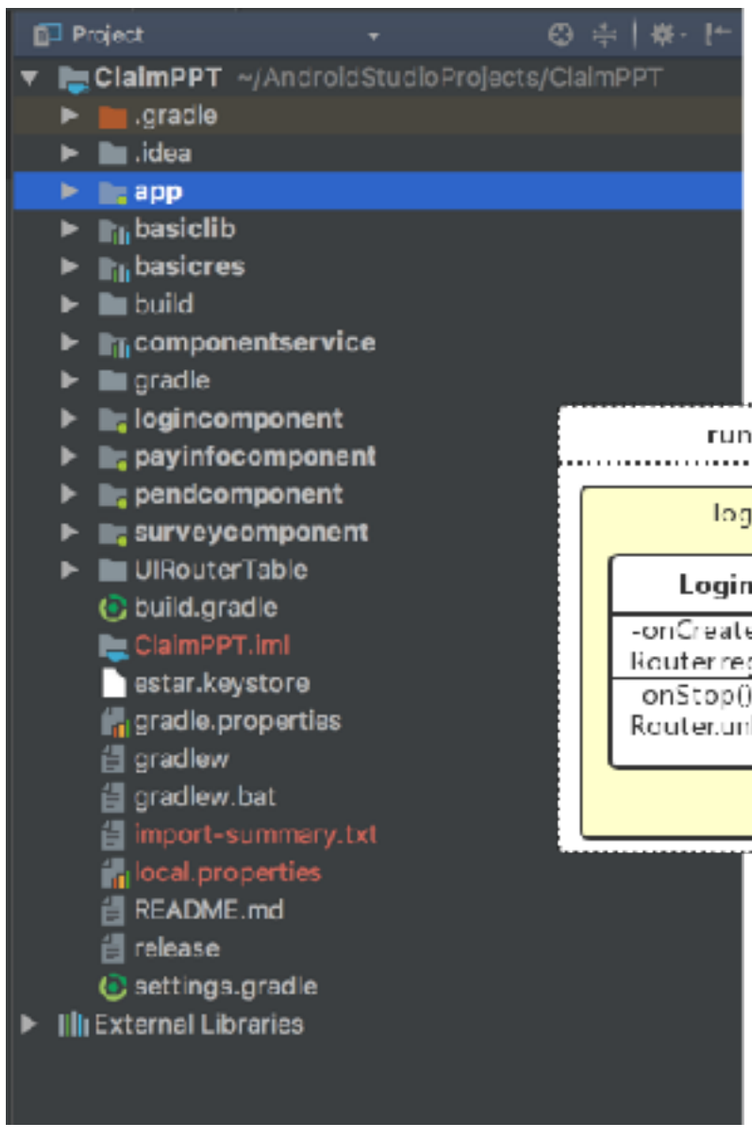
组件化四部曲

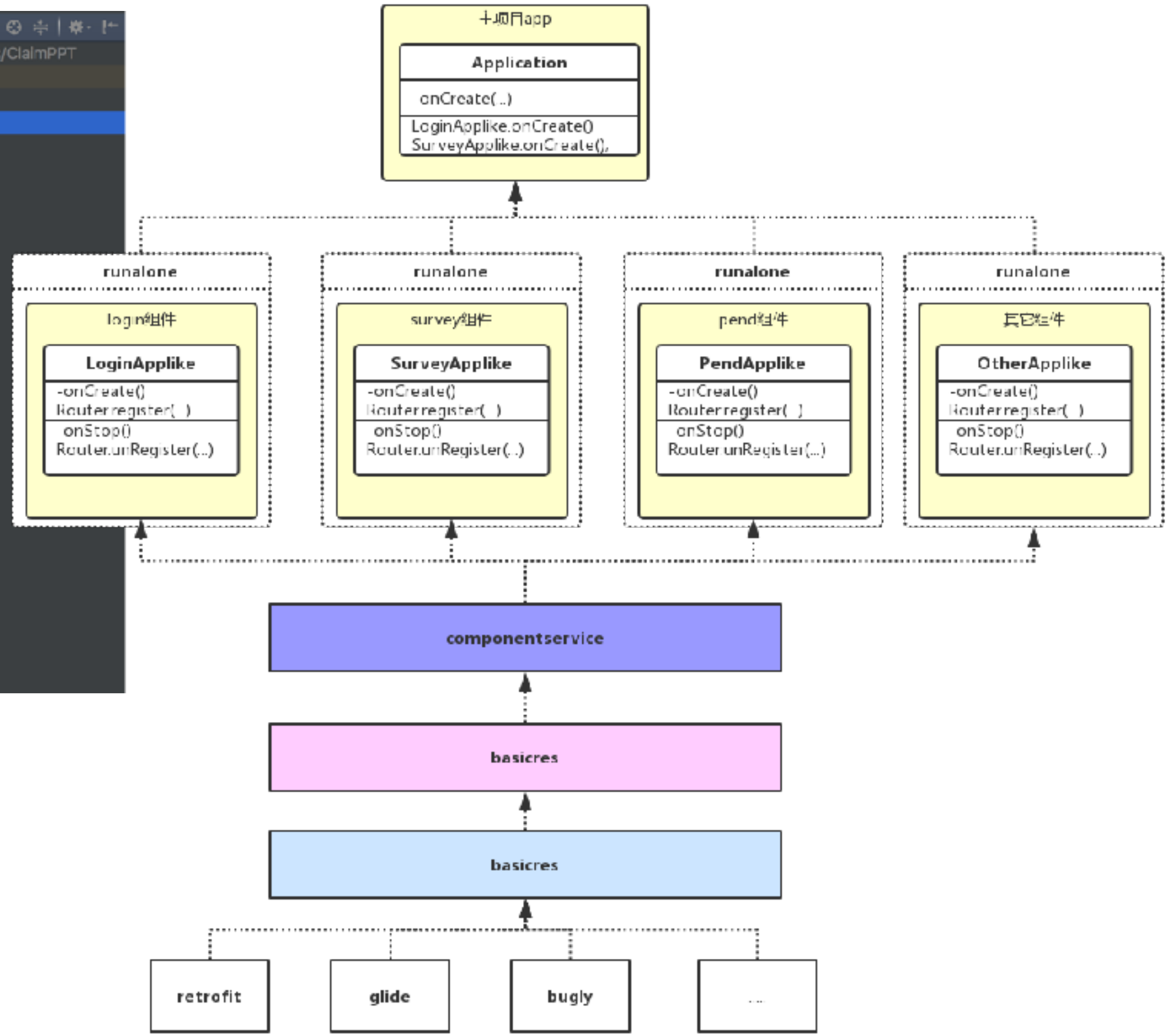
1、依赖库先行，业务依赖库初步抽离

2、寻找业务的边界，抽离边界清晰的业务模块，例如待勘、未决管理

3、将造成组件依赖主项目的模块继续抽离，例如登录、拍照

4、将主模块抽离成一个Host壳子，只包含小的业务模块或者组件的拼装逻辑


Project
ClaimPPT ~/AndroidStudioProjects/ClaimPPT
.gradle
.idea
app
basiclib
basicres
build
componentservice
gradle
logincomponent
payinfocomponent
pendcomponent
surveycomponent
UIRouterTable
build.gradle
ClaimPPT.iml
astar.keystore
gradle.properties
gradlew
gradlew.bat
import-summary.txt
local.properties
README.md
release
settings.gradle
External Libraries


主项目app
Application
onCreate(..)
LoginApplike.onCreate()
SurveyApplike.onCreate()..
runalone
login组件
LoginApplike
-onCreate()
Routerregister(..)
onStop()
Router.unRegister(..)
runalone
survey组件
SurveyApplike
-onCreate()
Routerregister(..)
onStop()
Router.unRegister(..)
runalone
pend组件
PendApplike
-onCreate()
Routerregister(..)
onStop()
Router.unRegister(..)
runalone
其它组件
OtherApplike
-onCreate()
Routerregister(..)
onStop()
Router.unRegister(..)
componentservice
basicres
basicres
retrofit
glide
bugly
...

万事开头难

1、走出舒适区，做好充足的准备

2、 组件化会长期停留在中间状态

3、 App会长期很胖，指望一次瘦身是不可能的

4、 这并不是停滞的理由，做好长期战斗准备

团队共识很重要

共识：我知道，你知道，你知道我知道，我知道你知道。。。。。

1、要得到大多数人的支持

2、持续的输出正向结果

3、给更多人赋能，让更多人入坑

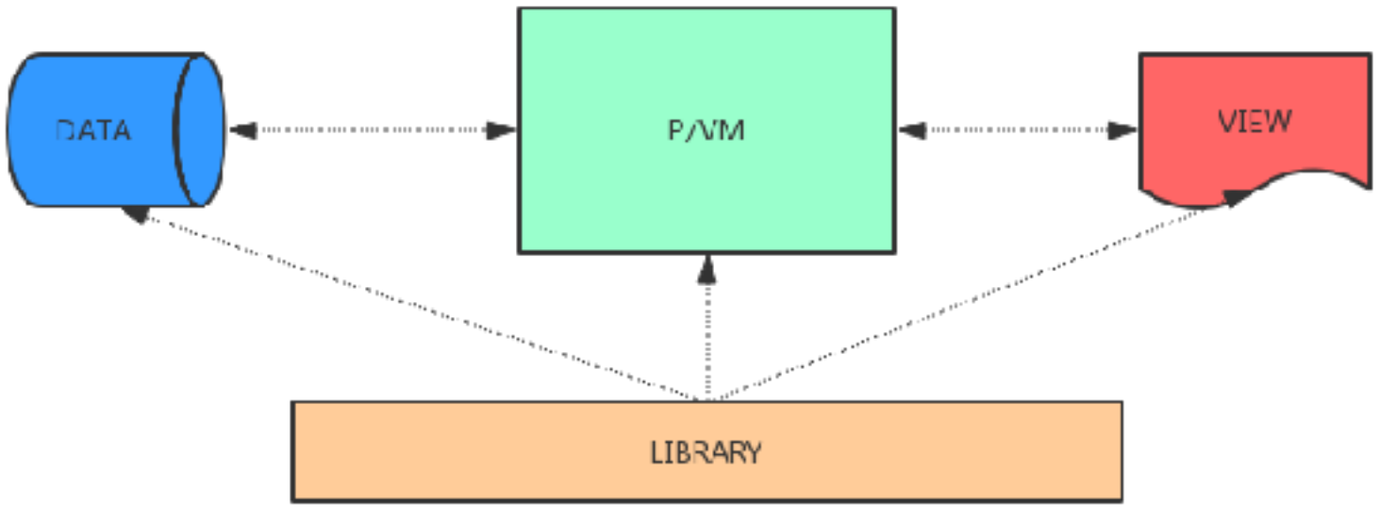
4、建立配套的模块负责人制度+代码审核制度

优化永无止境

1、四个维度优化：工程 组件 页面 类

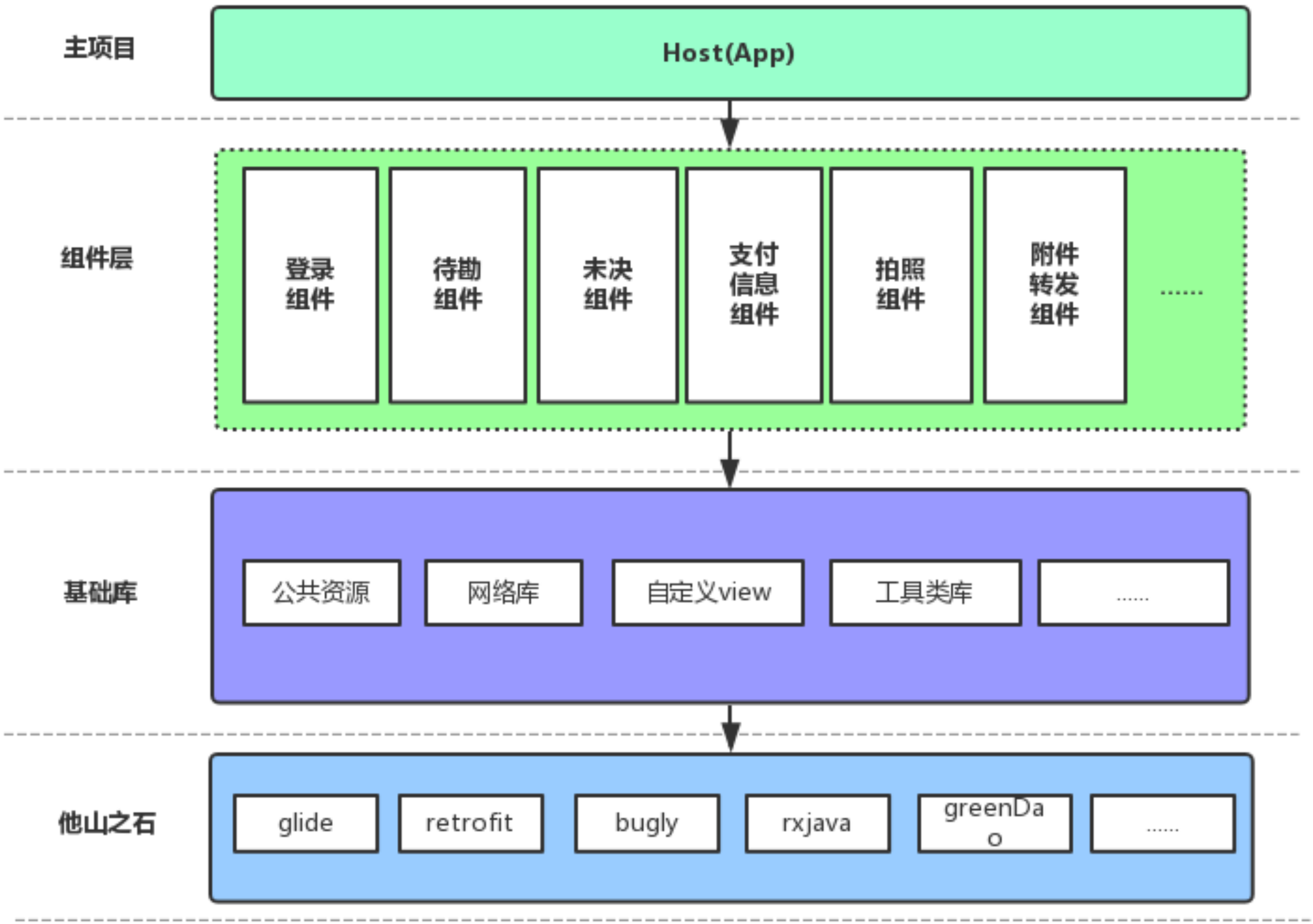
2、组件内部：功能分包，页面级别隔离甚至内聚?

3、页面内部：MVP/MVVM拆分



4、类：单一职责拆分，代码规范

财产险app组件化预期成果



谢谢聆听！