高高山顶立，深深海底行

—Android组件化的正确姿势

得到　张明庆

张明庆

毕业于浙江大学、中科院

| 网易

网易贵金属Android负责人

| 得到

得到App客户端架构师

DDComponent作者

掘金 | IT大咖说 知识共享平台

目录
CONTENTS

为什么要组件化

年龄

2-6岁

症状

肥胖

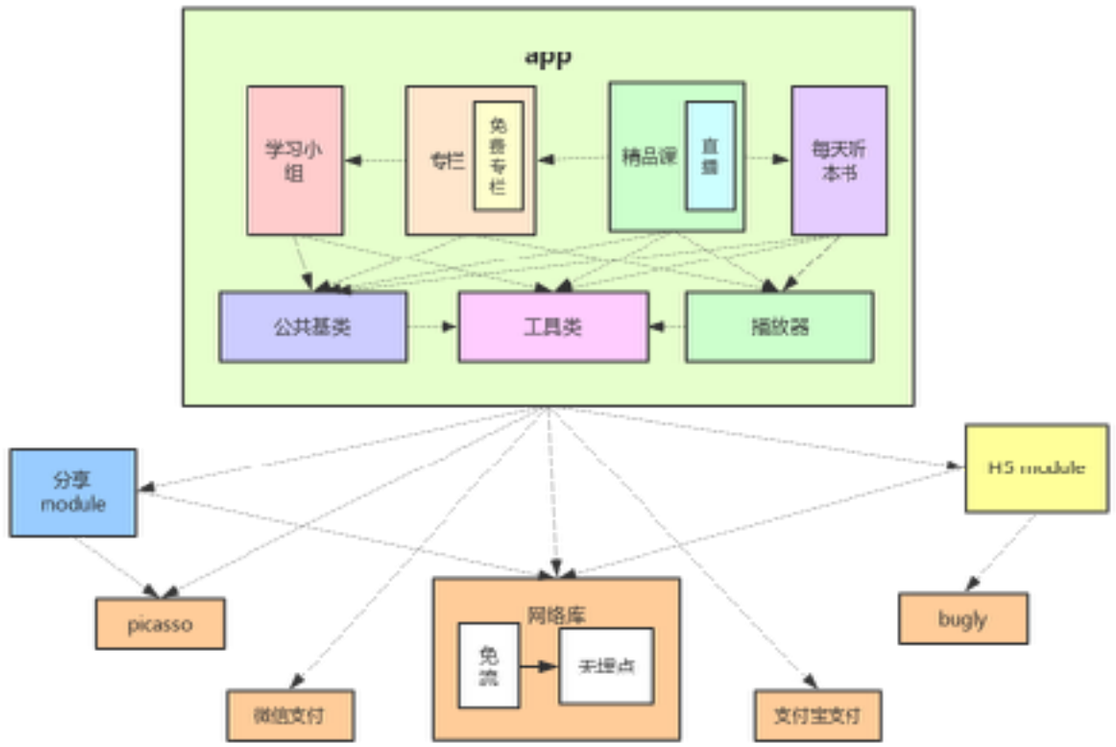
危害

巨大开发包袱

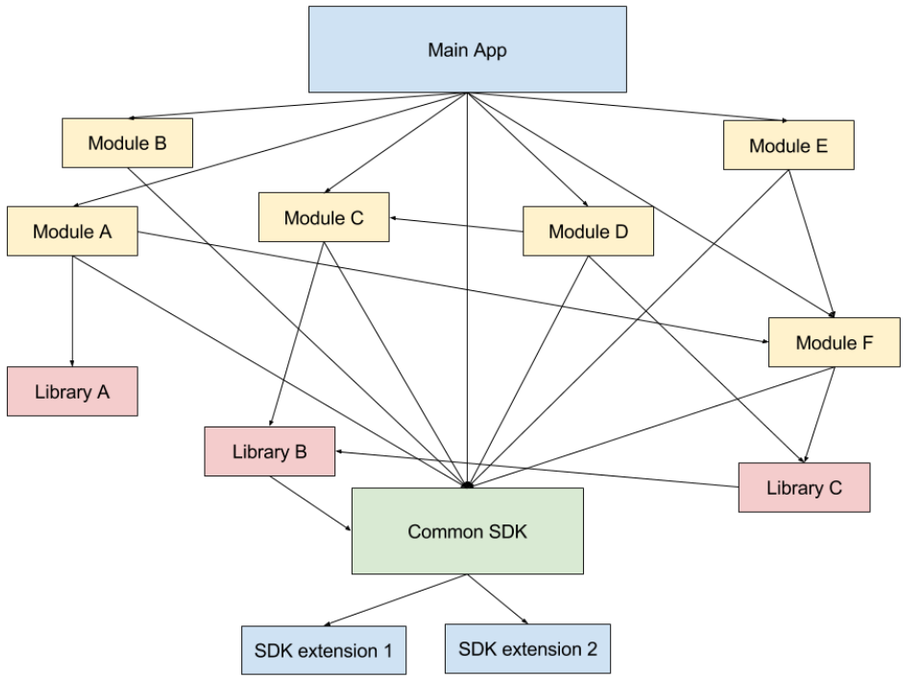
拖慢开发节奏

下班晚

用包来区分模块


app
学习小组
专栏
免费专栏
精品课
直播
每天听本书
公共基类
工具类
播放器
分享module
H5 module
picasso
网络库
免流
无埋点
bugly
微信支付
支付宝支付

用module来区分模块



组件化和插件化阵营

	组件化	插件化
app		
开源方案	DDComponent	Altas RePlugin VirtualAPK small DroidPlugin dynamic-load-apk AndroidDynamicLoader 23Code

为什么不是插件化

- 除了动态添加功能，组件化能实现所有插件化的功能
- 组件化的学习曲线更平滑
- 插件化不可避免进行系统Hook，9.0以上前途未卜

目录
CONTENTS

从代码搬家到隔离解耦

case1 组件需要怎么单独调试?

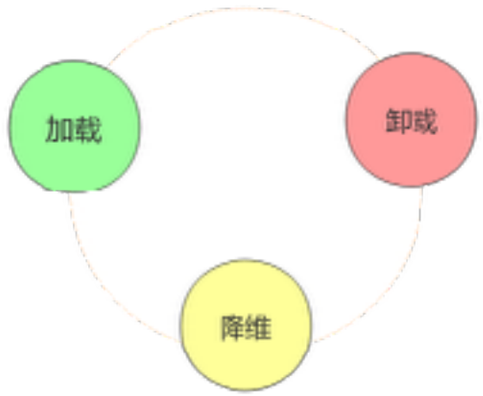
case2 如何避免资源冲突?

组件的生命周期

case1 运行时动态打开组件?

case2 运行时动态关闭组件?

case3 运行时将某个组件降维为H5?

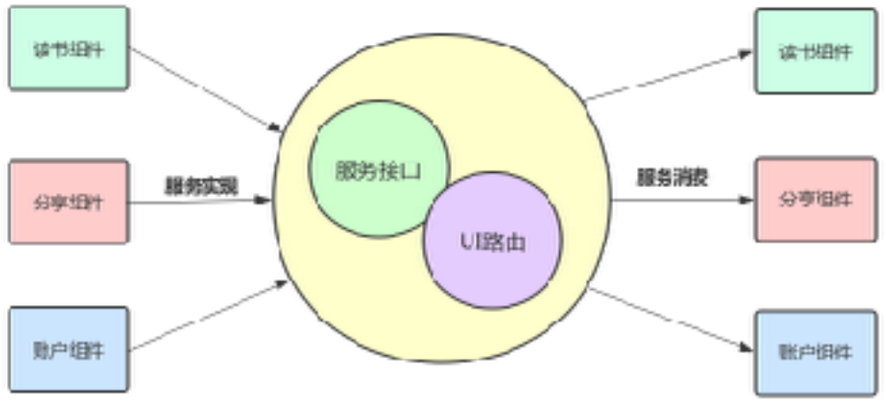


从鸡犬不闻到互通有无

case1 每个组件怎么提供服务

case2 怎么样做到更方便的服务发现?

case3 服务接口如何自动与其他代码剥离?



UI跳转

case1 是否支持scheme跳转?

case2 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 {
```

case3 是否可以生成清晰的路由表?

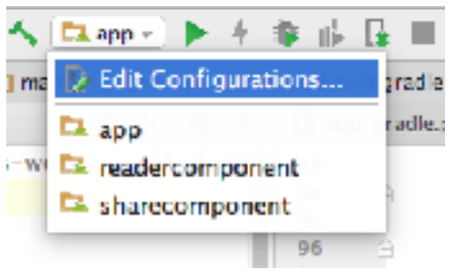
```
auto generated, do not change !!!!

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
```

集成调试



case1 任何一个组件能否充当host?

case2 组件由host切换到library是否可以无感知的完成?

代码和资源隔离

case1 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

case2 上面可以做到资源隔离?

NO！！

代码隔离和资源隔离

case3 可否做到编译期组件不可见，但同时全部组件参与打包?

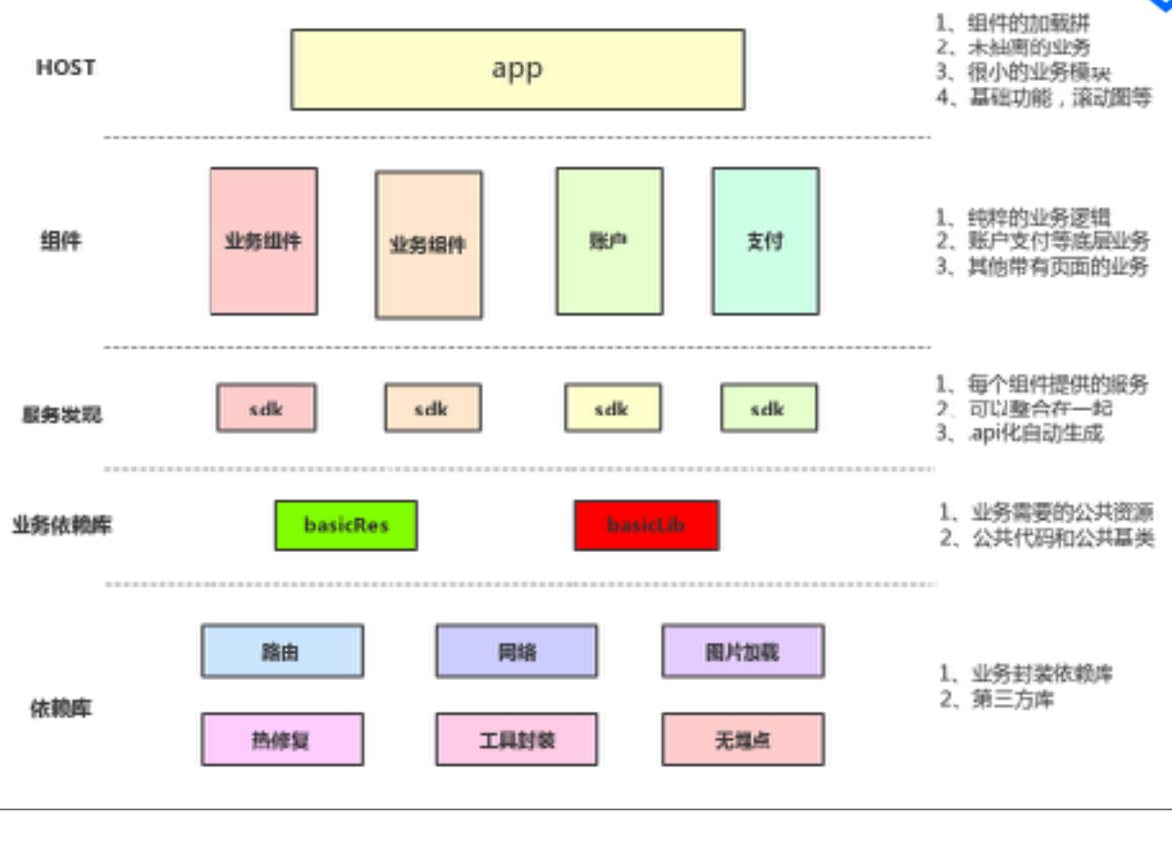
目录
CONTENTS

“组件化一直这么痛苦，还是只有设计的时候这样？”

Always

–Jean Reno

划分项目


HOST
app
1、组件的加载拼
2、未拆离的业务
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

怎么开始-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，例如读书、直播

3、将造成组件依赖主项目的模块继续抽离，例如账户、支付

4、讲主模块抽离成一个Host壳子，只包含小的业务模块或者组件的拼装逻辑

万事开头难

1、走出舒适区，做好充足的准备

Showing 489 changed files with 597 additions and 458 deletions

Showing 336 changed files with 202 additions and 207 deletions

Showing 208 changed files with 5038 additions and 0 deletions

2、 你见过凌晨四点钟的北京吗?

组件化会长期停留在中间状态

1、你的App会长期很胖，指望一次成功是不可能的

2、基于组件完全平行，集成交给app集成调试的方案是不可行的

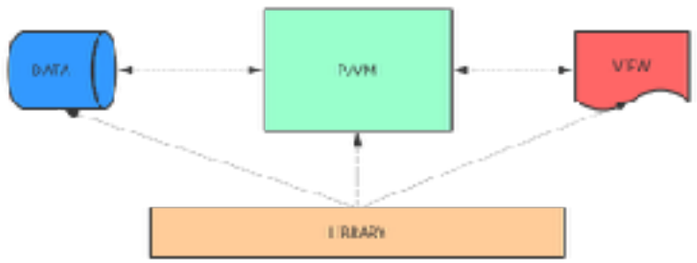
3、但这不是停滞的理由，要抓紧一起时机

无止境的优化

1、四个维度优化：工程 组件 页面 类

2、组件内部：pin分包结构，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

看好护城河，防止地道战

1、避免下沉：Event事件、实现类、公共资源

2、地道战：广播、SharedPreferences、DB

3、我们的原则：接受冗余，不知道是否下沉的就不要下沉

康威法则的威力

组织架构和代码架构之间存在着映射关系

第一条：尽量贴近组织结构和产品业务

第二条：尽力去反向改变组织结构和产品业务

第三条：如果做不到请回到第一条

第四条：不要逆行，你不是英雄

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

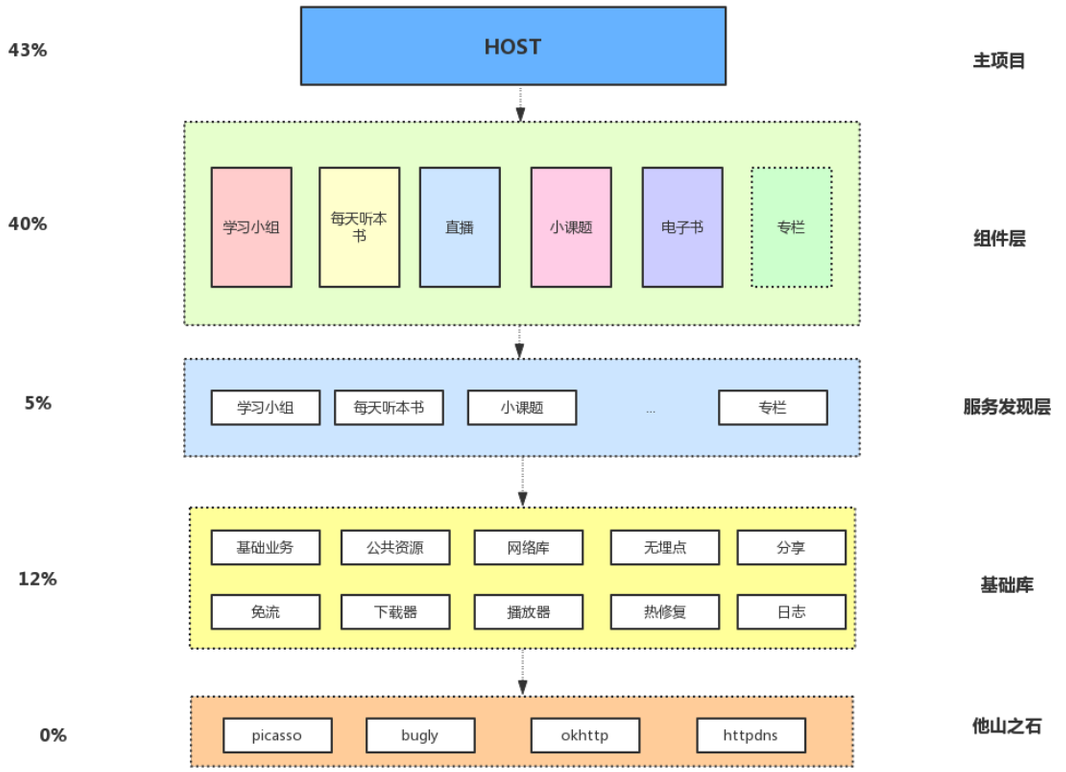
2、持续的输出正向结果

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

目录
CONTENTS

得到目前的组件化成果


43%
HOST
主项目
40%
学习小组
每天听本书
直播
小课题
电子书
专栏
组件层
5%
学习小组
每天听本书
小课题
...
专栏
服务发现层
12%
基础业务
公共资源
网络库
无埋点
分享
免流
下载器
播放器
热修复
日志
基础库
0%
picasso
bugly
okhttp
httpdns
他山之石

QA