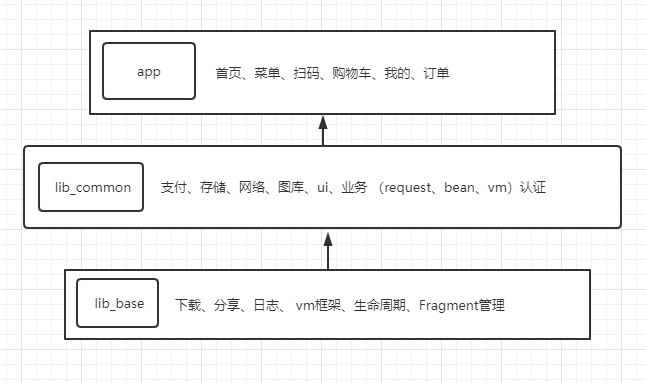
**app组件化架构方案初稿**

**一、项目现状与思考**



**当前问题**

1. 业务代码散落耦合至各处，不便于代码阅读与后续扩展.
2. 所有业务在一个大module，不利于并行开发
3. 功能模块层级不明确，职责不清晰，过多功能耦合在同一层
4. 基础层被完全依赖没有按需依赖

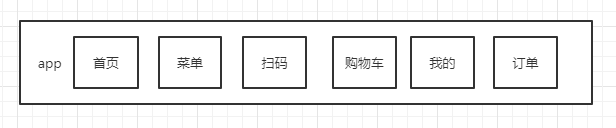
**设计改进**

1. 业务隔离，将项目业务拆分模块化，并且代码隔离
2. 厘清业务与功能，业务层，功能层，基础层分层设计
3. 组件设计松耦合，各组件抽象接口暴露能力，同层代码依赖隔离
4. 组件设计高聚合，各层次各组件单一职责，各组件独立运行能力
5. 按需依赖，依赖最小化原则
6. 支持渐进式随着业务增长可持续优化

**二、改进方案**

**基于业务的模块化**

为减少现有业务的入侵，减少项目稳定性与发版节奏的风险，现阶段业务仅模块化拆分，不同业务划分不同模块。实现代码隔离。

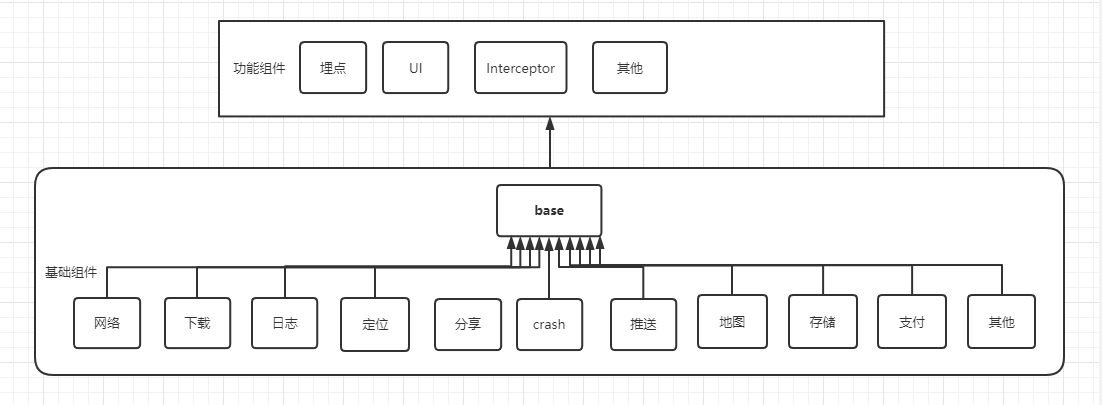


页面跳转：ARouter路由解耦

数据通信：ViewModel+LiveData

**基于功能的组件化**

功能组件划分与分层



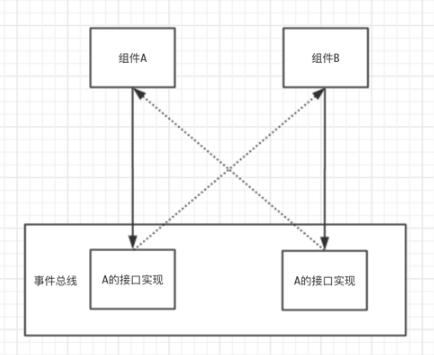
通过功能划分组件，组件单一职责内聚，提供接口导出能力，解耦交互。

**三、组件化实现方案**

**组件通信**

组件化目的，在于优化调用，在于通信解耦，须使用一种致力于事件驱动的组件化模型。

将**组件能力注册至总线**，由总线统一提供对外的访问接口，组件通过接口下沉的方式，对外提供的能力，接口下沉到base层，实现组件间的通信。

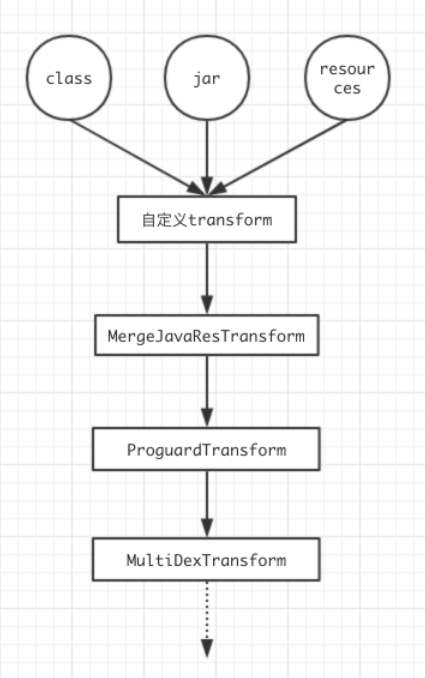


**接口注册**

为解决新增组件手动注册的问题，动态接口注册

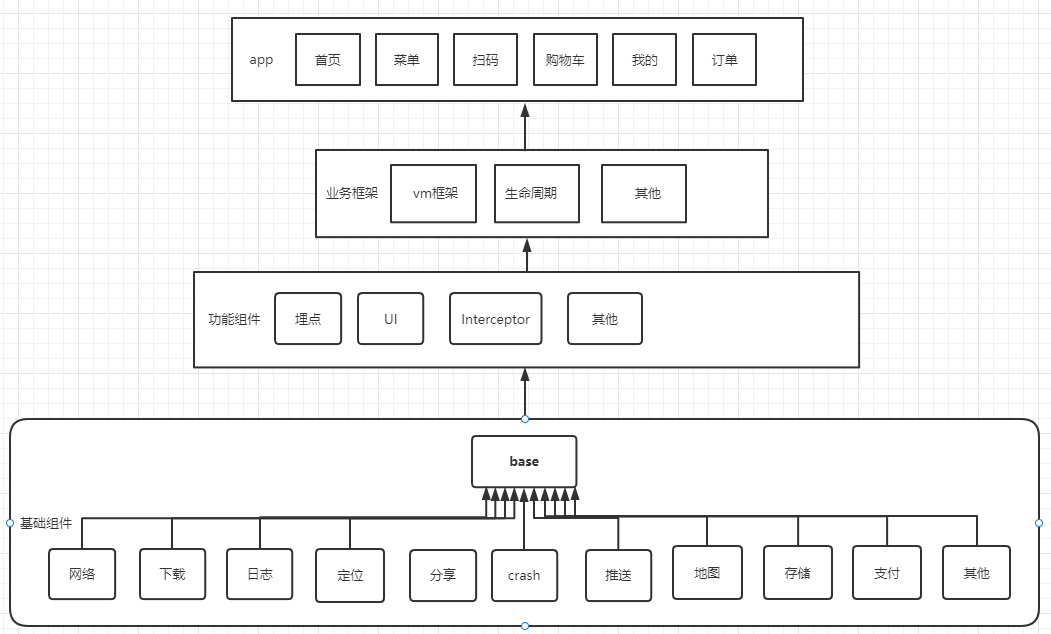
技术思想：在编译期间，扫描即将打包到apk中的所有类，将所有组件类收集起来，通 过字节码操作的方式生成注册代码到组件管理类中，从而实现编译时自动注册的功能

特点：不需要注解，不会增加新的类；性能高，不需要反射，运行时直接调用组件的构 造方法；能扫描到所有类，不会出现遗漏。

技术方案：ASM编写字节码 + Transform API编译期扫描 实现Gradle Plugin 

成熟框架CC ComponentCaller：基于组件总线的、支持渐进式改造的、支持跨进程调用的、跨进程调用、完整的Android组件化框架

**四、整体结构**



**五、执行落地**

**落地次序**

1、调整业务代码目录，为组件化做准备，**进展除bean目录外已完成**

（业务bean与common组件的网络请求耦合，需改造下层的代码方可迁移）

2、剥离耦合较少的组件：下载、

3、完成组件化架构搭建，验证可行性，并输出参考文档

5、遴选项目功能，提取组件，完成验证

剥离下载组件、完成验证

6、上线验证

7、重复3与4

**问题**

兼容已有项目、并行开发、上线验证

业务组件的提取