# 途强Android 5.0

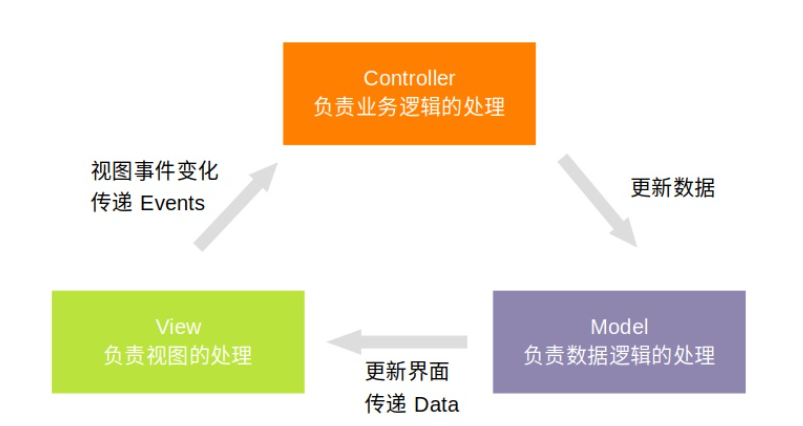
## 为什么要做5.0

1. 在市场需求快节奏变化的今天和未来，为了能达到在用户需求、市场变化、政策变化的时候能够快速迭代，快速交付版本以满足需求，抓住机会，Android整体架构必须能够支撑多人混合开发、多模块多需求同时开发而不会出现较大的困境，减少代码冲突。

1. 协同开发，便于测试。项目代码量越来越大，每次的编译速度越来越慢，哪怕几句代码的修改，都需要等待若干分钟等待编译运行查看执行结果，极大的降低了开发效率；业务模块越来越多，不可避免地产生越来越多且复杂的耦合，哪怕一次小的功能更新，也需要对修改代码耦合的模块进行充分测试；团队人数越来越多，却要求开发人员了解与之业务相关的每一个业务模块，防止出现此开发人员修改代码导致其他模块出现bug的情况，这个要求对于开发人员显然是不友好的；
2. 现有APP功能在架构、业务耦合程度、代码规范、代码安全、使用性能、可扩展性和可维护性都存在可改进的空间，在长期的可持续发展考虑，代码模块应该是长期的持续的进行逐步优化和改进。

## 怎么做

**使用现有的架构模式：MVC**



在 MVC 架构中，View 产生事件，通知到 Controller，Controller 中进行一系列逻辑处理，之后通知给 Model 去更新数据，Model 更新数据后，再将数据结构通知给 View 去更新界面。这就是一个完整 MVC 的数据流向

**为什么使用MVC架构模式**

MVC的架构模式适合快速上手、迭代、适合中小型项目，对于途强项目来说合理的使用MVC模式是学习、开发成本最低的架构模式，虽然也存在C的职责不清晰，VC代码都集中于Activity与Fragmengt中，导致都代码过于臃肿的问题，但是相比MVP和MVVM等其他设计模式的优缺点和重构架构的成本来讲，还是MVC模式更切合途强在线。

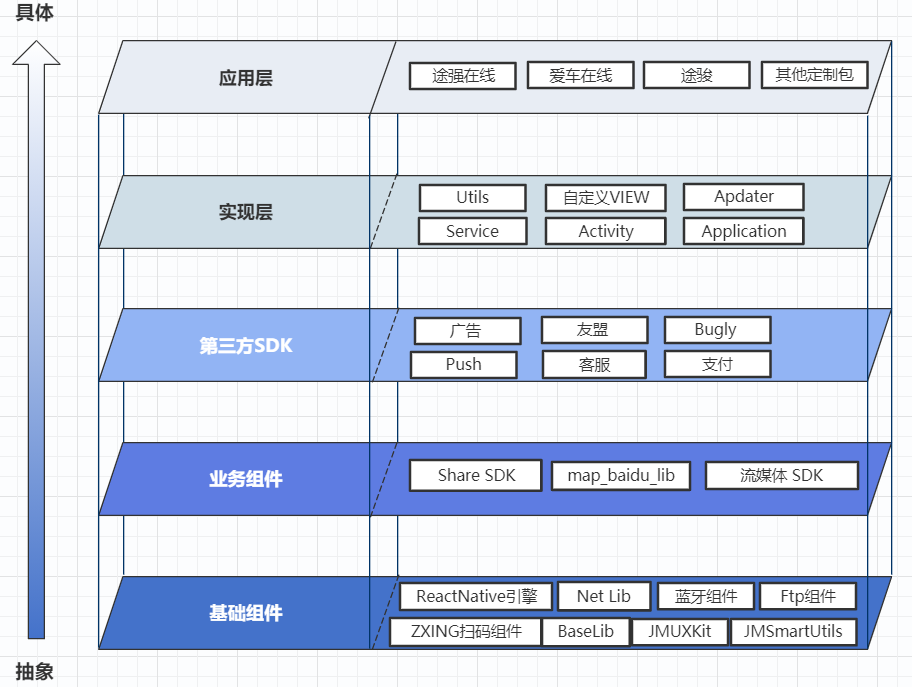
**如何优化现有MVC模式**

清晰的划分VC的的职责，尽量让Activity/Fragment只承担V的角色,把页面中需要实现的逻辑分发处理尽可能的抽离到自定义View.、Utils、Manager中,UI布局、元素的获取使用注解方式。

1. 把列表数据与View绑定的逻辑放置到Adpater中。
2. 涉及到的算法、功能、数据转换代码抽取到Utils中处理，统一接口调用。
3. 把复杂的页面交互组合到自定义View当中，Activity只对单独的引用。

**优化项目架构**

架构图如下：



途强安卓的架构主要是将功能和业务组件化，各个组件能独立拎出来加入到其他工程中也能使用。

基础组件，是最底层的组件库，提供最基本的单一功能模块。统一基础组件的运用也有利于公司产品风格统一化，方便管理。

**BaseLib**当前基础组件缺少Android base组件也就是BaseLib,此组件承载的是所有Android基础控件、适配器、activity等基础运用的封装，具备此组件后能减少上层代码逻辑和重复代码。

**NetLib**组件，网络交互是途强中最重要的也是频次最高的功能之一，将数据的请求、接收和转换封装为一个基础库是为网络交互能够做到“进”，“出”的统一管理，将每次都需要处理的数据格式、请求操作、异常处理进行一个整体的封装，达到页面请求数据只需要简洁的代码，只需要关心请求的参数和接收到回调的数据。

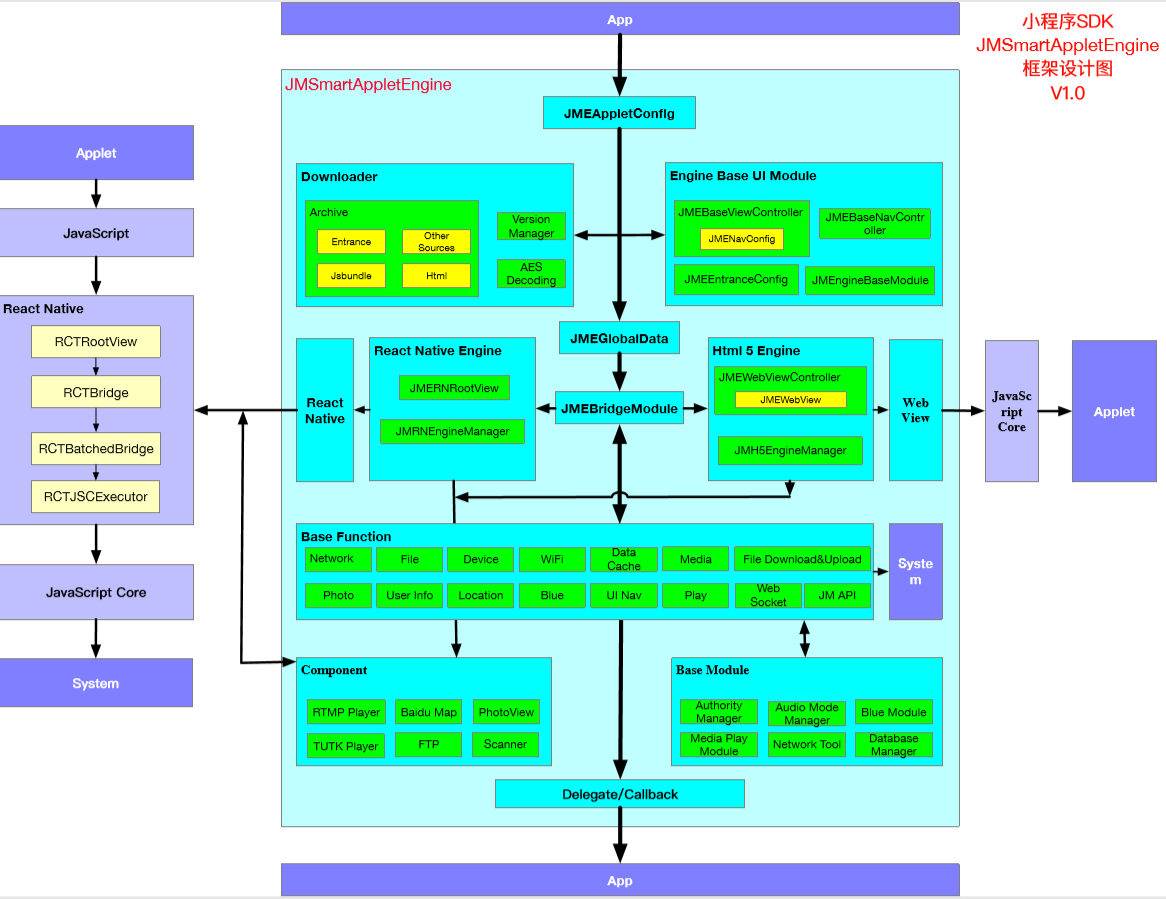
**第三方SDK** 统一由Gradle进行版本的管理和关联，将各个SDK的API分别模块化至各个Manager中进行管理，统一方法的入口，减少响应功能页面暴露的代码。

**应用层** 将多个APP应用都生成一个空壳，不同的参数、页面配置都可以在各自的壳中进行配置后打包，可应用在不同的定制包中。

**优化项**

1. **基础组件-ReactNative引擎 2.0**

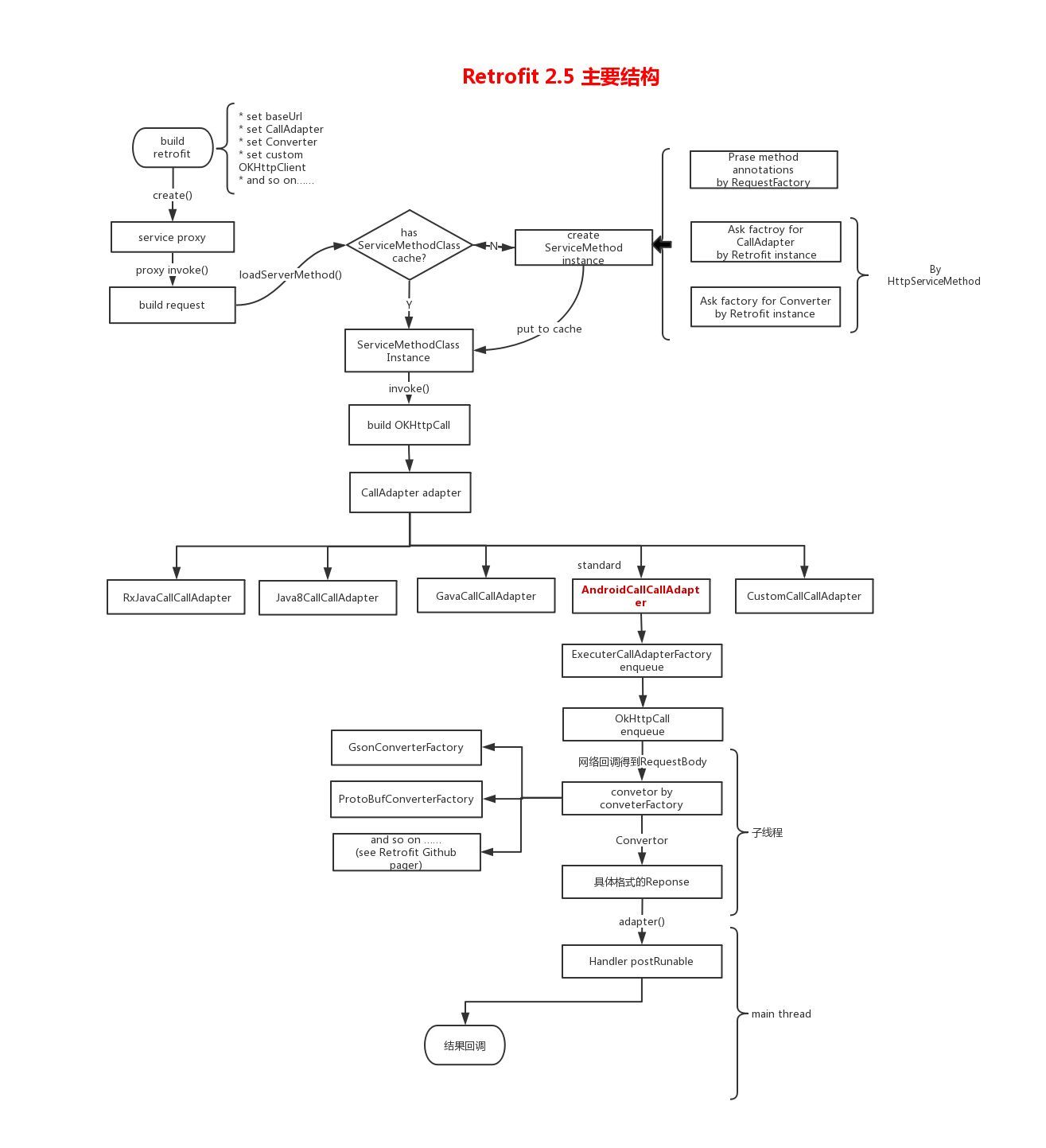
途强目前已集成Rn引擎SDK(JMSmartAppletEngine),该SDK提供支撑APP与前端小程序之间交互和通信能力，引擎中的API提供了原生APP的能力和视图组件，其中途强只引用了少部分API，大部分API需要进一步的验证和调整，包括通用接口的增加、接口回调数据/格式的调整。



**2.Net Lib 网络组件库**

途强现有的网络框架使用的是Retrofit+反射+evenBus方式，该方式的弊端是接口的请求编写复杂，并且调试比较困难，更难接受的是数据的回调方式编写代码很笨重，重复代码多，判断多，维护起来困难，Activity的代码越显得臃肿。

因此有必要进行网络组件库的重构，采用Rxjava2+Retrofit模式进行封装.



1. 接口的请求发起调用使用单例模式获取请求对象。
2. 服务器响应数据的基类BasicResponse，与服务器端统一数据返回格式，一般组成字段为{int code,String message,T data},统一所有接口的返回数据，方便拦截处理以及数据转换和错误抛出。
3. 设置日志拦截器，拦截服务器返回的json数据。Retrofit将请求到json数据直接转换成了实体类，但有时候我们需要查看json数据，Retrofit并没有提供直接获取json数据的功能。因此我们需要自定义一个日志拦截器拦截json数据，并输入到控制台以方便开发调试。
4. 设置Http请求头。给OkHttp 添加请求头拦截器，配置请求头信息。还可以为接口统一添加请求头数据。例如，把用户名、密码（或者token、安全签名）统一添加到请求头。后续每个接口的请求头中都会携带用户名、密码（或者token）数据，避免了为每个接口单独添加。
5. 为OkHttp配置缓存。同样可以同过拦截器实现缓存处理。包括控制缓存的最大生命值，控制缓存的过期时间。
6. 采用https，我们还可以在此处理证书校验以及服务器校验。
7. 数据处理类GsonResponseBodyConverter，将返回的数据转换为对应的实体Bean.
8. 定义DefaultObserver类在获取到数据后进行错误异常的处理，该类负责检查错误类型，返回对应的错误提示语。并且加入等待框dialog的封装。
9. 结合rxlifecycle管理Retrofit的生命周期。

**2.集成JMSmartUtils基础组件**

JMSmartUtils是公司内部开发的Android工具类组件，组件集合了大多数的包括字符处理、数据转换、日期、屏幕、图像、加密等方法.

**3.项目中工具类整合优化**

包括不限于：图片处理、字符串处理、加密、db工具、log工具、地图坐标转换、屏幕计算、系统信息获取、文件处理、日期、国际化等等，清除冗余重复的工具、代码，将功能类别一致的封装到统一的类中，提供公共静态方法给予外部调用。

**4.代码优化**

（1）冗余代码、无效过期的代码、无用资源文件的清理，提供代码有效率。

（2）梳理所有SDK引用和使用目的的清单，清理无用SDK。

（3）每两周进行一次代码走查，对日常迭代的代码质量进行把关。

（4）在不低于两周的周期时间内对线上项目BUGLY上报的崩溃问题进行查看和修复，发版后必须观察两日内的项目稳定情况。

（5）现有项目目录结构混乱，划分规则不一，需要重新划分目录结构。

**5.应用安全**

（1）代码混淆：安卓项目打包需要进行代码混淆，不能将核心代码无任何保护性的发布出去，针对该问题，除了进行代码混淆意外，安装包的发布还需要经过包的加固（360加固等）。

（2）数据加密：通信数据与本地保存的数据进行加密处理。

**6.应用性能**

（1）应用占用内存大小的优化，减少内存泄漏问题。

（2）APK包体积优化，降低APK包大小。

（3）电量优化，降低应用运行时所耗电量大小。

（4）页面卡顿，响应时间的优化。