从零开始的Android新项目（1）：架构搭建篇

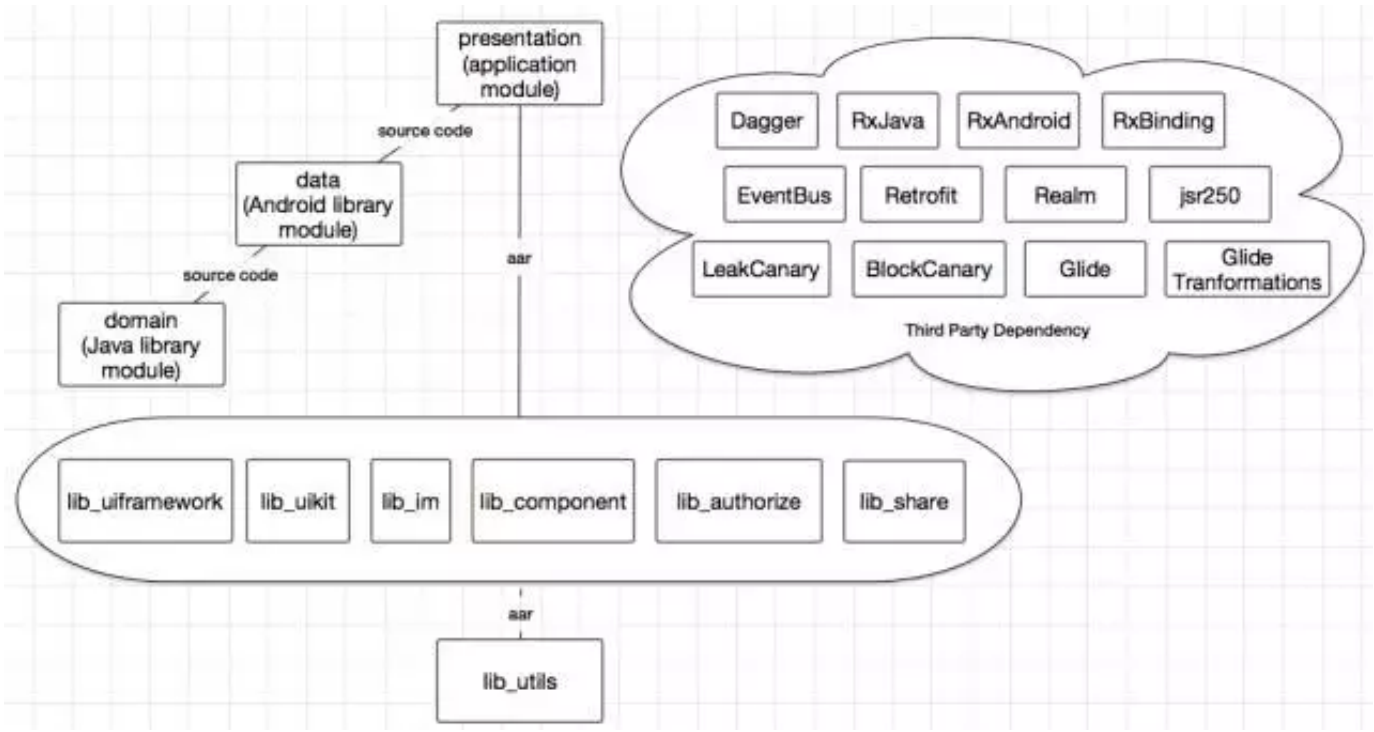
2016-03-22 安卓应用频道

(点击上方公众号，可快速关注)

> 来源：翟一帆（@翟一帆markzhai）
> 链接：http://blog.zhaiyifan.cn/2016/03/14/android-new-project-from-0-p1/

最近一直在忙新项目的事情，所以有的坑一直没填。。现在看来可能一时半会儿还填不了，倒不如记录一下新项目的搭建。

试想一下，如果没有历史负担，没有KPI压力，去新搭建一个项目，你会怎么设计和实现呢？



Application specific

类似clean architecture，分为三层 presentation – data – domain。

关于Clean Architecture由于国内的一些文章和项目都多少有偏离和错误，建议直接看

- http://fernandocejas.com/2014/09/03/architecting-android-the-clean-way/
- http://fernandocejas.com/2015/07/18/architecting-android-the-evolution/

两篇文章。

domain为纯领域模型，是Java library，包含了use case(Interactor)、DO、repository接口等domain package，目标是在任何Java语言的平台上都能直接使用，所以必须是平台无关，对平台没有任何依赖，能使用Java的方法（JUnit + Mockito）来直接进行测试。

data为domain的实现，是Android library，从MVP的角度来说，即是M层，内部隐藏所有数据细节，cache、数据库、网络、PO、exception（根据业务特点自定义的exception）、repository的具体实现（内部屏蔽数据细节，可能来自网络、数据库、缓存等）。使用Robolectric 3 + JUnit + Mockito进行集成测试。

presentation即展示层，是Android application module，对data不存在实际感知，依赖仅仅是Dagger注入的实例化，所有访问都通过接口进行，可见的是domain里的接口。包含了error message factory(所有exception都会被factory生成对应的message)、DI、mapper（vo转换）、VO（data binding进行MVVM）、navigation、presenter（不强制，仅在一些特别复杂的场景引入）、usersystem、utils（业务相关）、view（activity、fragment、adapter等）、application实现、业务常量等。可以使用Espresso和Android Instrumentation进行UI测试。

整个项目类似MVP+MVVM的混合（谁跟你说的MVP和MVVM是互斥的？），不过P层不一定存在，以避免为了模式而模式所导致的开发压力。

Common libraries

aar依赖以避免对编译速度造成影响，不过这里比较巧妙的是依赖作为module引入，所以当需要源码依赖的时候在build.gradle里进行注释/反注释就能迅速切换，十分方便。

lib_uiframework: UI framework，包含了各种Base类，如BaseActivity、BaseFragment、Navigator（应用中的一切通过scheme跳转）、ActivityManager、FragmentBackHelper等。

lib_uikit: 各种自定义view、第三方view的gradle或者源码依赖。

lib_im: 即时通讯库。

lib_component: 组件库，如cache、gif、ClassLoaderInjecter、Log、SP、Web等。

lib_authorize: 第三方认证(登陆)，目前包括QQ、微信、微博、LinkedIn。

lib_share: 第三方分享，目前包括QQ、微信、微博。

…等等

Third party libraries

Dagger, jsr250: 依赖注入框架让我们省了很多代码，jsr250则是Dagger注入时的一个annotation根据，我们的domain依赖了jsr250。

Rx大家族: RxJava, RxAndroid, RxBinding。新时代Android开发必备，还用说吗？

Retrofit: 装逼，喔，不对，高效率开发必备的网络框架。使用注解生成API，方便极了。我这里的实践是根据业务划分多个API接口，然后通过Dagger module进行实例化注入。

EventBus: 有了RxJava还要EventBus？当然，事件并不总是一对一的，也不总是流式的（可能是持续被动的）。举个简单的例子，feed详情信息更新后（比如点赞数据），外部可能有2个timeline页面也需要更新这个数据，这个时候EventBus就可以优雅地进行事件广播。

Realm：一个不依赖于SqlLite的ORM库，特点就是…快。还有同时有多个平台的方案: ReactNative, OC, Swift, Java…

Google Support库, data binding: Google家的，不用说了吧。

LeakCanary, BlockCanary: 开发阶段必备工具，前者是square家检测内存泄露的，后者是我自己做的卡顿检测工具。

Glide, Glide Transformation: 不折腾fresco，乖乖用Glide。

安卓应用频道

专注分享安卓应用相关内容

微信号：AndroidPD

长按识别二维码关注

伯乐在线 旗下微信公众号

商务合作QQ：2302462408