# **[Android业务组件化之现状分析与探讨](http://www.cnblogs.com/whoislcj/p/5853393.html)**

### **前言：**

      从个人经历来说的话，从事APP开发这么多年来，所接触的APP的体积变得越来越大，业务的也变得越来越复杂，总来来说只有一句话：这是一个APP臃肿的时代！所以为了告别APP臃肿的时代，让我们进入一个U盘时代，每个业务模块都是一个具备独立运行的U盘，插在哪里都可以完美运行，这就是推进业务组件化的初衷也是一个美好的愿景。

   业务组件化相关文章地址：

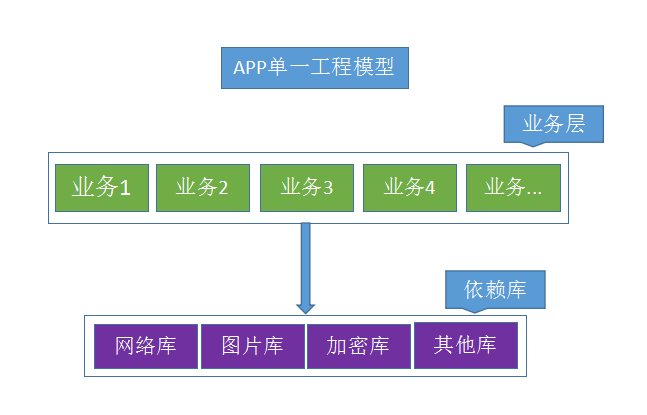
* [Android业务组件化之现状分析与探讨](http://www.cnblogs.com/whoislcj/p/5853393.html)
* [Android业务组件化之URL Scheme使用](http://www.cnblogs.com/whoislcj/p/5825333.html)
* [Android业务组件化之子模块SubModule的拆分以及它们之间的路由Router实现](http://www.cnblogs.com/whoislcj/p/5860138.html)

### **需求背景：**

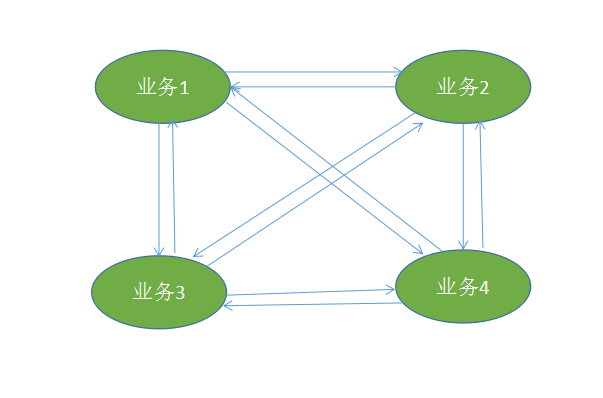
     随着公司的快速发展，版本不断的迭代，业务变得也越来越复杂，业务模块的数量有可能还会继续增加，而且每个模块的代码也会越来越多，这样下去势必影响开发效率，增加项目的维护成本，每个工程师都要熟悉如此之多的代码，而且每次编译如此之多的代码，电脑卡死了有木有？慢的像蜗牛先生有木有？这样以来估计工程师直接疯掉了！然后跳楼go die 了！所以推进公司业务组件化迫在眉睫，这也是实现业务组件化的大背景。

### **现状分析：**

      只有知道自己问题出在了哪里？才好寻找解决问题的办法，我们先来看下目前大部分app的单一项目结构原型。大致如下图所示：



一眼望去这结构不是挺清晰的么？每个业务都放在单独的包名下，网络库、图片加载库等第三方库与上层业务都完美的剥离了，我们再来看下他们的直接的依赖关系图：



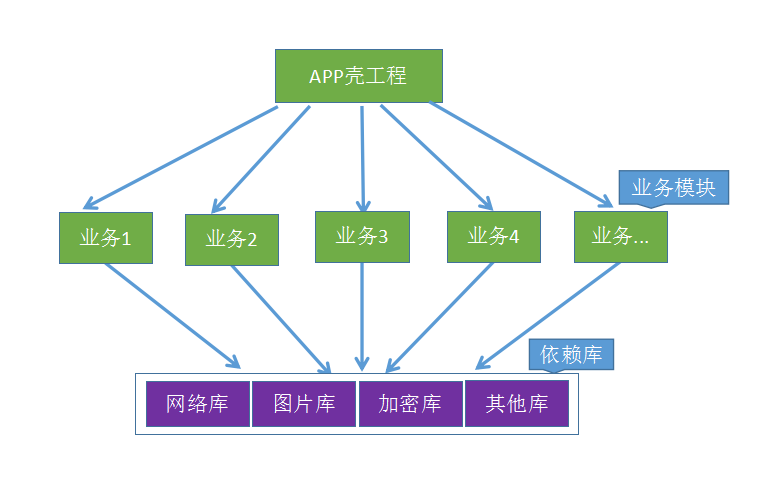
虽然上面的依赖关系举例有点太过于极端，但是真实场景中是存在的。各个业务之间代码互相引用，所以这种结构也是架构整改的根本动机，也是当务之急应该考虑的事情。为了更好的满足各个业务的迭代而彼此不受影响，更好的解决上面这种让人头疼的依赖关系，开始着手app架构整改。

从上面的分析我们可以得出适合业务组件化的几种情况：

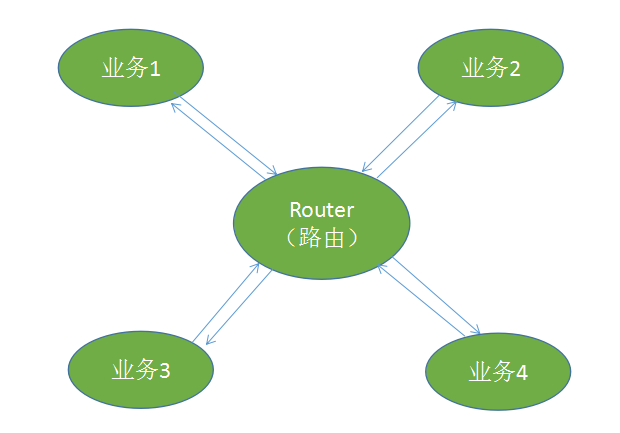
* 业务较多、而且复杂
* 业务之间的依赖需要解耦
* 团队成员较多，需要各自开发相对独立的业务

### **业务组件化方向：**

     APP业务组件化架构整改的方向就是告别结构臃肿的时代，让各个业务变得相对独立。模块工程和类库工程之间遵循向下依赖关系，各个模块之间的遵循平级依赖关系。先看下整改后的各个独立业务模块与类库工程之间的向下依赖关系图：



    整改的方向由一个项目工程拆分成若干个模块工程，由app壳工程提供统一的入口，每个业务独立的模块module共享项目的依赖库。由壳工程集成需要引入的业务模块，至于各个独立的业务模块之间的调用依赖关系，我们借助一个中间层充当路由功能，这个路由我们放在各个业务模块共同引用的依赖库那一层。由路由统一调度他们之间的依赖关系，路由调度解决平级依赖问题示意图：



通过APP壳工程提供的路由功能，各个模块之间调用不再采用传统的显式调用，而是采用隐式调用的方式。从而使各个模块之间不再存在依赖关系。

### **APP业务组件化架构整改技术方案：**

   业务组件化大致需要解决以下几个问题。

#### **1.）业务组件的生命周期**

    按照理想状态的来看待的话，各个业务组件之间没有任何依赖关系，这时我们可以把每个独立的业务组件看成一个可运行的app，所以业务组件的生命周期和应与独立的app保持一致。

#### **2.）业务**组件之间通信****

    在各个单独业务组件内，基本上保持原来的调用方式，组件之间通信采用URL Schema方式实现页面跳转，格式为 schema://host/action?param1=value1¶m2=value2 。关于schema映射表维护由上面提到的Router（路由）这么一个角色来维护，

schema解析由系统Framework来维护。其中schema映射表可以做成后台配置文件形式。也可以代码维护，不过组件需要预先注册。以上说的对于传递基本参数是没啥问题的。如果要传递对象的话，转成Json 字符串就可以了。

关于schema可以参考我的这篇博客：[Android业务组件化之URL Schema使用](http://www.cnblogs.com/whoislcj/p/5825333.html)

### **APP 业务组件化架构整改带来的好处：**

 如此规模大的架构整改需要付出更高的成本，还会涉及一些潜在的风险，那么整改后的架构能够带来哪些好处呢？

加快迭代速度，各个业务模块组件更加独立，不再因为业务耦合情况，在发版时候，由于互相等待而迟迟不能发布版本。

稳定的公共模块采用依赖库方式，提供给各个业务线使用，减少重复开发和维护工作量。

迭代频繁的业务模块采用组件方式，各业务线研发可以互不干扰、提升协作效率，并控制产品质量。

为新业务随时集成提供了基础，所有业务可上可下，灵活多变。

降低团队成员熟悉项目的成本，降低项目的维护难度。

加快编译速度，提高开发效率

### **总结：**

     本篇主要来分析为何要实现业务组件化、组件化的方向、组件化的技术方案简单探讨、组件化带来的好处，由于目前还处于实施初期，很多观点有可能是错误的，很多技术坑还没涉及，后续会跟进更多相关实现方案。