携程软件架构分析

1、携程最初版本的架构很简单，主要采用了MVC模式，即模型、视图、控制器模式。这种模式将交互式的应用程序分为三个部分，将内部信息表示，用户信息呈现，用户操作处理分开，实现了高效代码重用。而携程的最初版本架构在传统的MVC架构上封装了一个数据服务层，基本架构如下图所示

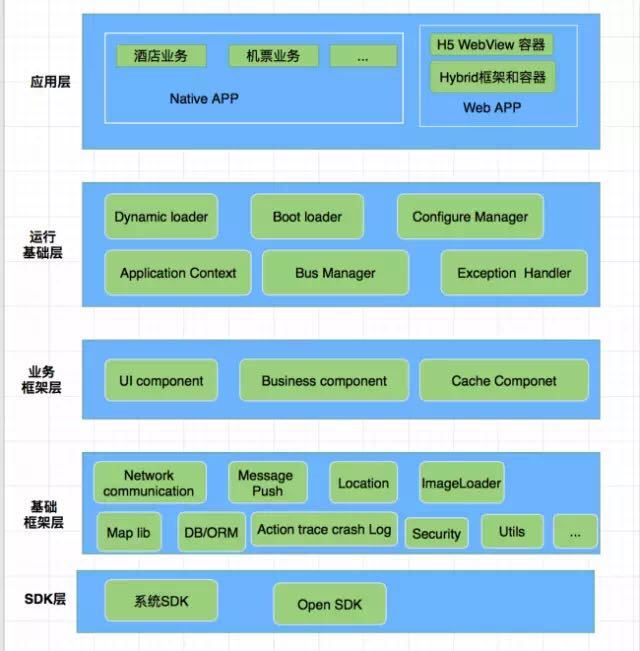
移动端界面UI

业务控制层

数据服务层

业务逻辑层

2、携程后期迅速发展，发现一些功能使用频率低，为了不占用程序资源，将架构进行纵向的分离，改进后的架构图如下所示



3、APP架构3.0

最初应用层的酒店，机票，火车票等都是独立的apk，他们之间独立开发，互不影响，最终统一以插件的方式集成到统一的携程APK里面。酒店和机票之间通迅方式采取两种方式，BUS数据总线跳转 和 URL Scheme跳转。

4、MVC架构的优点与缺点

Ⅰ.优点：

①可以为一个模型在运行时同时建立和使用多个视图。变化-传播机制可以确保所有相关的视图及时得到模型数据变化，从而使所有关联的视图和控制器做到行为同步。

②视图与控制器的可接插性，允许更换视图和控制器对象，而且可以根据需求动态的打开或关闭、甚至在运行期间进行对象替换。

③模型的可移植性。因为模型是独立于视图的，所以可以把一个模型独立地移植到新的平台工作。需要做的只是在新平台上对视图和控制器进行新的修改。

④潜在的框架结构。可以基于此模型建立应用程序框架，不仅仅是用在设计界面的设计中。

Ⅱ.缺点

①增加了系统结构和实现的复杂性。对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。

②视图与控制器间的过于紧密的连接。视图与控制器是相互分离，但确实联系紧密的部件，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用。

③视图对模型数据的低效率访问。依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损害操作性能。

④一般高级的界面工具或构造器不支持模式。改造这些工具以适应MVC需要和建立分离的部件的代价是很高的，从而造成MVC使用的困难。