Weex源码分析系列（五）Weex SDK架构分析

1、前言

经过前面几篇Weex源码分析系列文章，相信大家对Weex是什么、Weex能带来什么、Weex是如何工作的等几个问题已经有了自己的答案。

备注：

- **本文会非常简洁，因为是站在前面几篇源码分析文章的肩膀之上进行概括总结。如果还有疑惑的话强烈建议大家回过头再去看看之前的文章；**
- **基于Weex0.16.0版本；**

2、Weex的核心能力

诚然，Weex为了方便开发者提供了各种方便的组件便于傻瓜式开发，但随着我们对Weex剥丝抽茧，我们更能学习到Weex的精髓：如何以新思路解决老难题？

Weex动态化、跨平台、结合Native的设计不仅解决了需要频繁发版的问题，同时在一定程度上可以节省人力成本。在技术选型上RN与Weex的思路都没有问题，也都坚信这条道路都可以走得通。**对于二者最核心的能力就是JS引擎与Native的交互能力，别的类如Js框架、组件支持、工具链、扩展、生态等实际上都是附属，都是一项技术的附带产物，为了赢得开发者的支持而做的。**

对于Weex而言其实只要保留下Js与Native的交互能力，别的各种产物我们都可以自己实现一份，思路不会有大的差别。

所以大家更应该由一项新技术的学习逐渐过渡到真正理解技术背后的本质，如果是我们自己实现我们的思路是什么样的，新技术有哪些我们可以借鉴的地方！

3、Weex框架分析

大家回忆下关于各个组件源码分析的文章中：

- **首先离不开的就是JSBridge，这是JS与Native交互的桥梁；对应于通信层；**
- **JS发回来的每一个指令都会先经过各种Manager的处理，然后封装成相应的Action；对应框架层；**
- **注意很多操作都是在特定的线程比如WeeXDomThread，与之相对应的有线程的切换；对应于线程通讯层；**
- **最后每一个JS指令执行的时候都会对应于一个组件：Module、Component、Adapter等；对应于组件层；**

4、Weex框架图

我们再对着框架图分析一遍：WXBridge负责JS引擎与Native的交互、框架层负责将JS指令进行处理、通讯层进行各个线程的切换、每一个JS指令都对应了组件层的执行。

5、总结

本文是对前面几篇源码分析文章的一个概括总结，只是尝试去**理解Weex SDK的框架设计及核心原理，毕竟对于Weex整个的生态而言SDK只占了其中的一部分，还有很多类如开发流程、工具链、扩展等很多的面等待我们继续探索。**

欢迎持续关注Weex源码分析项目：Weex-Analysis-Project