# **闲鱼 Flutter 架构演进和实践**

11 月 23 日， **字节跳动技术沙龙 | Flutter 技术专场**在北京后山艺术空间圆满结束。我们邀请到字节跳动移动平台部 Flutter 架构师袁辉辉，Google Flutter 团队工程师 Justin McCandless，字节跳动移动平台部 Flutter 资深工程师李梦云，阿里巴巴高级技术专家王树彬和大家进行分享交流。

以下是 阿里巴巴高级技术专家王树彬 的分享主题沉淀，《闲鱼 Flutter 架构演进和实践》。

演讲内容大纲：

1. 闲鱼为什么选择 Flutter
2. 闲鱼的架构及其演进
3. 回顾及展望

今天主要分享的内容是闲鱼一年半在 Flutter 的演进，希望有某个阶段某个场景和大家契合，给大家一些启发和讨论。
**个人介绍**
先做一个自我介绍，我是闲鱼的架构负责人，在 2009 年就加入阿里了，2009 年是淘宝首次实现营收平衡的那一年，也是第一次双十一的那一年，在那以后迎来了淘宝快速发展的时期，在那个快速发展的背景下，我做了商家百亿级数据的在线分析系统以及营销系统，再后来就是用到无线，在无线时代时去做手淘上 0 到 1 的地理围栏和 LBS 的无线特性的项目。
闲鱼是 2014 年创立的，2015 年我来到闲鱼，经历闲鱼从早期到现在超过 2000 万 DAU 的发展阶段，这 4 年时间看到闲鱼这样涨起来。中间到了一定团队规模以后，我就去负责闲鱼架构，以适应闲鱼的快速发展。
**闲鱼为什么选择 Flutter？**
Flutter 也并不是最完美的一种框架，没有最好的只有最适合的，到底是什么场景让闲鱼选择了 Flutter？
**闲鱼背后是高速发展的 C2C 市场**
大的业务背景是这样的。首先，闲鱼是 C2C 的交易市场，随着中国网购商品不断丰富，社会上存在越来越多大量的闲置物品，出现了这样一个存量市场，这个市场规模有多大？从闲鱼角度看，在 2019 年财年，就有 6000 万以上的用户发布了自己闲置的商品，这个数字还在增长，到现在每天已经有 100 万人发布超过 200 万件商品。然后，因为它是 C2C 对等交易，这样的用户会形成更多互动和社区属性，可以看到闲鱼里已经沉淀 130 多万个鱼塘，形成这样的社区。这是一个大的背景，就是高速发展的存量市场。
**闲鱼客户端侧的业务特点**
端侧业务特点是什么样的呢？有几个特点：

1. 业务心智。闲鱼是独立的业务心智，促使我们把主要开发力量投在闲鱼 APP 上，有一些外投的场景，像投到手淘、支付宝或者外面的场景，那是少量的页面。
2. 高苹果用户占比。闲鱼的特点是苹果用户占比高。这个特点导致早期时我们在设计上对设计的要求非常极致，两端的设计要求一致且要以苹果设计为主，这时候如果做安卓开发的同学就可以想象到安卓这边的痛苦了。
3. C2C、社区业务背景。导致有很多互动、商品图片及视频的需求，另外快速发展期有很多大的业务尝试。

**闲鱼的端技术选择**
这样几个特点就导致我们下大决心迁到用 Flutter 来做，那是到了 2017 年底的时候。在中间也曾经尝试过用 Weex。
现在闲鱼的技术栈是这样的：活动类和外投类的产品页面用 H5、小程序、Weex 开发；主链路、产品化的页面都用 Flutter 开发，这部分在闲鱼的覆盖率有 80%以上。原来 Native 开发同学都转型到用 Flutter 开发了。这些是技术因素，此外还有团队也是很重要的因素，Native 同学写 Flutter 会转换更快一些，闲鱼有 Native 人员基础，所以才有决心深入去做 Flutter。
**闲鱼的架构及其演进**



这是发展到现在闲鱼的架构：下面是从项目管理、打包平台、持续集成到自动化测试；中间的包括页面组件、Native 混合、音视频组件等；再往上一点是承载业务的容器，让业务能够把自己业务流程下沉的容器（Dynamic 和 Serverless）。

### **闲鱼架构的演进**

这个架构怎么演进来的？大概有三个阶段：第一，混合开发，引入期。第二，规模化，发展期，越来越多页面和人员投入时，会面对一些新的问题。第三，一体化，新探索，成熟以后考虑下一步怎么探索，原来只是从跨两端角度提高效率，现在把云（Server）也考虑进去，从云端一体的角度考虑效能的进一步提升空间。

## **一、混合开发、引入期**

引入期通常做些什么事情？闲鱼从最开始要用 Flutter，到最后上线用了 3－4 个月时间，因为的前车之鉴少。但是到今年，杭州有一家初创公司，它从决定开始用 Flutter，到最后上线，只用了大概 1 个月，可以看出现在这个引入期成本已经降下来了。虽然降了，但是引入期最大的工作内容还是在混合开发这一块，也就是怎么把 Flutter 混合到自己已有 APP，除非你是开发一个全新的纯 Flutter 应用。

混合开发大概包括几种：

* 第一种，混合栈，这是躲不掉的，怎样把页面混合起来，混合栈是这个架构图里的 Boost，已经开源了。
* 第二种，工程化，从代码管理到持续集成、打包、监控、高可用，这一串的链路。
* 第三种，音视频和资源一体化，这块主要是考虑高性能图片、视频组件与 Native 复用。

### **1. 混合栈的关注点**



（1）单页面栈和多页面栈。从 Native 角度看，当新开一个 Flutter 页面，这个页面要在新的 Activity（以 Android 为例）中打开，还是在同一个 Activity 里面面来回切换 Flutter 页面？如果每一个 Flutter 页面都挂在一个新的 Activity 下，就算多页面栈的设计。反之就是单页面栈。哪种更好？如果想跟 Native 本身的栈能够很好结合，逻辑和顺序都能完全一致的话，那多页面栈是最合适的。那单页面栈适合用在什么场景？比如在 Flutter 页面弹出浮层或者 Dialog 的场景，这时直接在 Flutter 内部用 Navigator.push 就可以达到想要的效果，但是除此之外，大多数场景都需要多页面栈的方式。flutter\_boost 这两种都支持。

（2）单引擎和多引擎。上面说的多页面栈涉及一个问题，每个 Flutter 页面都挂在一个新的 Activity（以 Android 为例）下，那这些页面的 Flutter 引擎到底是用同一个引擎还是多个引擎呢？这也是混合栈要考虑的重要问题。直观来看，要适应多页面栈，那就用多引擎，每个页面都起一个 Flutter 引擎最简单了，但最大的问题是内存会暴涨，虽然现在官方也想去深度复用这个引擎，让引擎能够很轻量化，但是目前这个还没有 Ready。所以从单、多引擎来讲，目前我们使用的方案还是单引擎的方案。单引擎的方案的优点就是内存省了，所有的 Flutter 都在一个引擎里。

（3）单引擎虽然解决了内存暴涨问题，但由于引擎内的栈跟外面 Native 栈的页面生命周期不一样，用单个 Flutter 引擎相当于把这个 Flutter 页面在 Native 页面间来回移动，Native 的生命周期跟 Flutter 生命周期要完全对上，如果对不好的话就会出现转场白屏、埋点丢失、页面销毁不当等各种奇怪的问题。对单引擎设计来讲，实现基本流程通常没问题，但是最重要的是里面细节问题特别多，需要仔细处理各个细节问题。这方面大家可以参考 f lutter \_b oost 。

### **2. 混合开发的工程化问题**

这块问题通常是因为原来有 Native 同学，现在有 Flutter 的同学，要一起去开发。这个问题刚才有一个同学也提问到了，这时工程到底怎么做？是用 Flutter 包 Native，还是用 Native 包 Flutter？这是两种思维方式。标准的 Flutter 是在 Flutter 下面挂了安卓、iOS 的工程，用这样 Flutter 包 Native 的方式。还有一种方式是可以反过来，可以把 Flutter add to Native 的思路，闲鱼用这个思路做了一个 flutter\_boot 工具，已经开源，大家有兴趣也可以参考。

### **3. 音视频和资源一体化**

闲鱼原来在 Native 时期，有成熟的图片库，也有成熟的视频组件做滤镜、编解码优化之类优化的，这些组件在 Flutter 上要复用起来，如果用官方默认的方式在生产环境会发现性能有问题。



这是关于视频复用的方案，第一个官方直接给的默认实现，在没有格式转换的情况下问题不大，但需要格式转换时，性能就会有问题。以打开摄像头做预览的场景为例，默认的第一种方案是把 Texture 提取到 Java 或者 Flutter 运行期的 Buffer 对象里，每一帧通过 Channel 传递到 Flutter，再贴到 Texture 上，这个办法缺点是有一次读、一次写的过程，性能不是最优的。从我们的实测看，iPhone 7 上 720P 的视频，这种方案平均每帧多消耗 5-6ms。

后来我们尝试两种方案把性能提升，第一种方案的思路是“新方案 1”这样的，想办法把 Native 侧的 Texture 跟 Flutter 侧的 Texture 复用，直接传递 Texture ID，这个好处是没有往上 Dump 内存的过程。这种方式不好的地方是需要改 Flutter 的代码，打 Patch。还有一种更好的方案，这是我们后来发现的，这种方案不用打 Patch，可以用 Surface Texture 直接复用的机制，在 Native 和 Flutter 侧是同样的 Surface Texture，就可以复用了，建议大家用这种方式更好一些。

## **二、规模化**

工程变复杂了，页面变多了，这时候重要关注的点是怎样让人员之间、模块之间解耦。团队大了并行开发更多，有几个小组需要并行开发，不再是一个大组了。另外，也要关注代码标准化。

刚才说到解耦，大概有两个层面之间的解耦：一个是“Card”的解耦，长页面中每一行是一个业务组件 Card， Card 之间让它们尽量有少的互相依赖。还有一部分是 Card 的内部解耦，主要是解 UI、逻辑和数据分离的问题，这个问题业界有一种 Redux 方案，闲鱼做的 fish\_redux 是把组件 Card 间的解耦能力加到了 Redux 上，即把一个大页面拆成多个组件，组件间也可以解耦解了。

规模化期另外一个考虑的问题是高可用。Flutter 有一个特点，由 Flutter 引起的整个 APP 的 Crash 很少，更有效的方式是监控它的异常，对异常的监控就需要在这个阶段把异常做聚类、信息裁剪。另外，还需要对页面 FPS，可交互时长等的监控。

此外，CI 和自动化测试方面，在规模期也有一些事情要做，原来的自动化测试有一些没有办法用在 Flutter 上，例如基于 Instrumentation 和 UiAutomator 的测试方案。只有基于 Monkey 的可以直接用。

这里详细介绍下解耦时用到的 AOP 技术，AOP 是解侵入的手段， JAVA 和 OC 里这块都容易，但是在 Dart 里没有成熟方案。AOP 的一种解决方案是基于代码生成，另外一种解决方案是在 Dill 里修改，Dill 是 Dart 到最终产物中间的语言。



这块的思路是什么样？原来有一个 APP 工程，你想对它加一些功能，这个用 AOP 方式切进来，让原 APP 工程感知不到。解决方案是让 aop-project 的 Main 引用 app-project 的 Main。打包时选生成 aop.dill，包含这两个工程所有的代码，再经过 aspectd 的处理，把两个工程的切面编织到一起，成为最终产出物。

怎么去改这个 aop.dill？Flutter 在编译中有一个很好的设计，它提供转换的能力，可以让你在编译流程中拿到语法对象以后，对这个对象做自己的转换。基于这样的原理，读取语法代码的树，然后对它做遍历，把工程里的 AOP 注解解析出来，编织到一起就可以了。做 AOP 时要考虑，比如以 foo() 函数为例，是想在 foo 外面做界面插入，还是在 foo 里面第一行和末行插入，甚至还有一种可能是在 foo 函数中间某一行做插入，这些都是 AOP 不同的编织动作，对应几个操作分别有不同的注解。

## **三、一体化**

第一个概念，动态模板是在业务层想做业务流程下沉时用到。比如一个交易链路里有可定制的优惠模块、物流模块、商品信息模块等，可能有非常多组件，可以把这个流程沉淀，变成中台，让上层多个业务能够复用组装，在这个主干的流程上做自己的扩展点。我们的一个解法是用动态模板的方式，让 Flutter 支持模板化。

第二个概念是 Serverless 框架，可以把服务端的业务层代码跟客户端代码统一。

上面讲的是一体化的端侧的概念，下面这部分是服务端的容器，很重要的基础设施是 FaaS 平台和 ServiceMesh。标准的 Faas 不需要我们自己去建，云上和阿里内部都有，我们要做的是让 Dart 跑在 Faas 上，做一个符合 Faas 规范的 Dart 语言的 Runtime 容器，核心要解决什么问题？一个是让 Runtime 能适应不同的 Faas 平台，它有足够的解耦性。另外一个重要的问题是解决语言无关性问题， Dart 语言肯定碰到原来生态如 Java 或者某种语言的，像闲鱼的是 Java 生态，那怎么去调这些异构服务，要解决语言无关性的问题。大概思路跟 Servicemesh 差不多，用 Sidecar 解决语言无关性的问题。

中间是通道服务响应框架，这个框架主要是通道层的抽象，为什么要有通道层的抽象？原来是基于无线网关接口，现在基于 Faas，可以把这个 Faas 看成一个本地接口去调，甚至可以把 Faas 上跑的函数看成本地端的函数，这样去调用。

一体化的演进过程：第一步，刚才讲的是一个逻辑下移的过程，把原来端侧的逻辑写到了 Faas 层，这就是逻辑下移的过程。第二步刚才讲的模板是属于 SSR 范畴，属于把渲染下移的过程。最后是远景，把 UI 自动化生成掉。一体化的设想在没用之前业务可能有很多疑惑，到底这种开发模式大家会不会接受、能不能真的带来效率的提升，但是把这个推出来以后，我们有几个业务基于这个去做，效果还是非常不错的，把原来非常重的云端逻辑归一化了，例如下单页的逻辑，客户端下单要算优惠，组件联动，端侧有很重的逻辑，服务端也很重的逻辑，现在下单的业务同学用一体化把这两个逻辑归一在一起，都到 Faas 上解这个逻辑时，就变得可维护性好了很多。

有的同学可能会疑惑一体化后，端上原来可能是有状态的，在 FaaS 上怎么解决端上状态的问题？

这个问题的解法有几个思路，首先对于客户端编程来讲，我们用框架把本地函数和 FaaS 函数统一封装路由，需要访问远程的情况才会路由到远程。另外，基于 Redux 的 Action 方式调用 FaaS，Action 中可以先把客户端完整 State 状态传到 Faas 上，Faas 函数可以知道客户端的完整状态。有一些副作用是请求次数变多，对于这种特殊情况需要注意。

一体化还有一个重要的问题是工程化归一，使开发 FaaS 像在本地工程在写客户端代码一样。工程化归一业界有开源软件 Serverless Framework 可以复用，亚马逊、Google 云在 Faas 层都有 CLI 工具组件，基于开源产品去打通工程一体化就容易一些。

## **回顾及展望**

大概回顾一下今天要讲的内容：

第一，闲鱼选择 Flutter 选择的因素，给大家做参考，闲鱼是在这样几个业务特点：独立 APP、双端一致性要求，苹果用户占比高；APP 中以产品化页面为主。另外，团队组成也是一个重要因素。

第二，闲鱼架构演进过程，从引入期、到发展期、到一体化的探索，每个阶段要去考虑的问题和关键的技术点。

最后，展望接下来要做的：

Flutter 和 Faas 刚刚讲的一体化这块刚刚起步没多久，有一些典型的业务场景落地了，但是还没有大规模的做更多铺开。UI2code 尝试。另外，我们希望把 Flutter 基础设施、基础组件能力开放，能让超大型的 APP 使用。最后，Flutter 的社区生态，要靠大家一起共建和完善。

## **Q&A**

提问：我是一名 iOS 开发者，目前在做跨平台工作。跨平台开发者都知道阿里系有一个相当知名的 Weex，您也提到闲鱼在 2016 年主要业务都是由 Weex 去承载的，我们也有做 Weex 的相关工作，它有能够做到多端同构，你可以自定义一些驱动，甚至部署到小程序上。从 Weex 更符合大前端、一体化的趋势，Weex 在跨平台以及原生平台上都有相当的表现。我的问题是，作为阿里嫡系跨平台优秀的引擎，是什么契机决定让你们闲鱼团队全面拥抱 Flutter？能够给我们一个更好的参考。

回答：这个问题是场景的问题。Weex 跟 Flutter 在跨端、动态性、性能方面的表现，是一个三角形。目前的技术框架都很难把这三要素都解决非常好，比如你刚才说 Weex 很好，能够很完美的解决这三个问题，但其实 Flutter 的关键优势是在性能更好，比 Weex 好。Weex 的优势是它前端的生态和容器的广泛性以及它的动态性。说到底是场景问题，闲鱼 80%是主链路场景，更多是性能这块的考虑，只有 20%是属于外投活动类的，这部分我们用 Weex、H5、小程序做，主链路还是更高性能的选择。杭州那个创业公司例子也是这样的，在 RN 和 Flutter 中选择，最后敲定到了 Flutter 上。确实大家都有这个考虑，还是看自己的场景想取哪方面。

提问：我之前看闲鱼公众号上有 UI2code，想看看你们最新的进展是怎样的，UI2code 在闲鱼 APP 上页面实际使用的占比率大概是怎样的？

回答：我们有个自己内部创业 APP，一个全新的 APP，是用 UI2Code 做的，效果就很好。我个人觉得 UI2code 目前技术水平适合做全新的 APP 或者全新的页面。做老页面最大的问题是修改、合并的问题。所以 UI2code 对全新 APP 页面作用很大，因为那个全部是用这个生成的。目前这个技术更适用这样的场景。

提问：你们用 Weex 差不多是最早的，我们的 APP 也有 Weex。我想问的是你们对 Flutter 兼容 Weex 有相关的技术吗？比如我之前有些业务是 Weex 做的，现在想在 Flutter 中去做，有相关的技术调研吗？

回答：这两个调用的话，其实它们两个是独立的，Weex 如果渲染到 Native 控件，就像标准 Native 跟 Flutter 是一样的。但是 Weex 如果最后降级成 Web View 就变成 Web View 跟 Flutter 的问题，它们其实也是兼容的，现在 Flutter 上用 Web View 也是没问题的，Web View 本质上也是 Native 的，所以最后都会落到 Native 和 Flutter 的关系问题。如果你想做控件极的关联，比如 Weex 写的控件跟 Flutter 写的控件做交互，它们两个可以在同一个页面出现，这样就会比较难，目前无法实现。此外，如果是想让 Flutter 像 Weex 一样有那么灵活的动态性，目前也没有太好的办法。

提问：Faas 那一层是不是相当于把 Server 端数据库的模型转换成前端 View Model 的过程？另外一个问题，Flutter 客户端协同开发是有组件化方案？还是有其他的？比如开发过程中调试的方案是什么样的？Flutter 多业务线同时开发一个 APP，有很多独立的组件，在开发过程中是怎样调试的？比如运行在闲鱼壳 APP 里跑起来，通过什么样的方式能够跑起来？

回答：Faas 是不是指把服务端 DB 数据透出到客户端的这个问题。大概可以这样理解，从业务开发分层角度看的话，有客户端、中间业务胶水层，领域层。胶水层就是把各个领域的接口拿过来组装，组装成页面要的数据，返回给客户端。FaaS 做的是这一层，胶水层。

提问（续）：相当于领域那一端接口还是 Server 来维护，最后输出更偏向于 Server 输出的数据，客户端需要配置东西的话需要在这层上再封装一层、组装一层客户端需要的形式？

回答：FaaS 这一层就是上面说的胶水层，以前没有 FaaS 时这一层是服务端同学写的，现在由客户端同学用 FaaS 来写。Faas 这一层可以理解为前端通常说的 BFF 这一层，但实际上会比 BFF 的内容更多些，FaaS 解决的是烟囱式的逻辑，FaaS 的横向间依赖比较少，一个请求进来，向下层拿数据把它组装返回。领域层的区别是它有很多领域与领域间的横向依赖，用的书据存储也非常多，模型复杂，就不适合 FaaS，目前还没有用在 FaaS 上。

提问（续）：怎样判断 Faas 具体运行在 Server 端还是客户端？

回答：其实 Faas 都是在 Server 端，只不过有一个基于 Action 的客户端框架，让客户端开发时，本地逻辑和 Faas 逻辑都像写一个函数一样。我们的想法是在 Faas 函数上加一个注解，剩下就都是一样的了，是这样的方向。

提问：我经常关注闲鱼的一些技术文章，您刚才提出两个观点，noUI 和动态模板下发。在 Flutter 中写页面时，有参考 Flutter Design 网站，把所有的组件进行动态拖拽生成静态模板，Flutter 趋势是否可以发展成只关注数据集和业务逻辑，剩下都交给工具集或者工具去做，这样就可以很大一部分解放了 Flutter 写页面的生产力，闲鱼有没有这样的考虑？或者提供这样的编译工具给社区来使用？

回答：这个其实在内部是有用的，但是目前还没有开放出来。你讲的那种方式可能是更多偏工具化的，用工具去生成这样一个页面，它的代码还是 Flutter 的代码。这种方式闲鱼有交集的是 UI2code，确实是直接生成 Flutter 代码，倒也不是拿一个图片完全一下子就生成了，也有你说的拖动过程，要做些修改、属性简单的设置，然后生成 Flutte 页面代码，这块是可以的，但是这块也没有开源。