Android跨平台开发

## 1、混合开发概述

Hybrid App主要以JS+Native两者相互调用为主，从开发层面实现“一次开发，多处运行”的机制，成为真正适合跨平台的开发。Hybrid App兼具了Native App良好用户体验的优势，也兼具了Web App使用HTML5跨平台开发低成本的优势。

目前已经有众多Hybrid App开发成功应用，比如美团、爱奇艺、微信等知名移动应用，都是采用Hybrid App开发模式。

## 2、移动应用开发的三种方式比较

移动应用开发的方式，目前主要有三种：

Native App： 本地应用程序（原生App）

Web App：网页应用程序（移动web）

Hybrid App：混合应用程序（混合App）

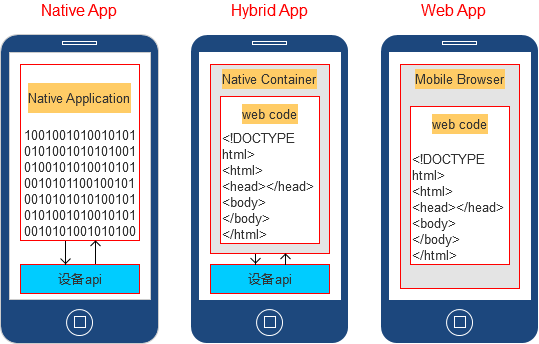


表2：三种移动应用开发方式比较



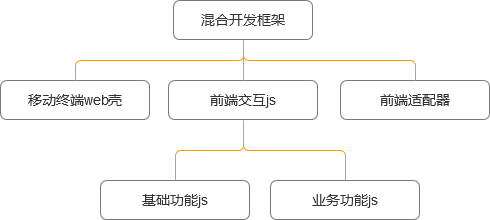
3、混合开发应用场景

（1）折中考虑——如果企业使用 Hybrid 开发方法，就能集Native 和web两者之所长。一方面，Native 让开发者可以充分利用现代移动设备所提供的全部不同的特性和功能。另一方面，使用 Web 语言编写的所有代码都可以在不同的移动平台之间共享，使得开发和日常维护过程变得集中式、更简短、更经济高效。

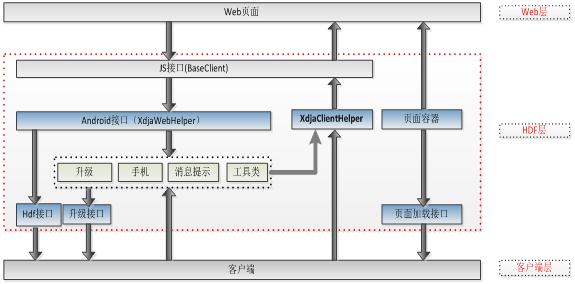
（2）内部技能——许多企业都拥有Web 开发技能。如果选择 Hybrid 开发方法，在合适解决方案的支持下，Web 开发者只要仅仅运用 HTML、CSS 和 JavaScript 等 Web 技能，就能构建 App，同时提供 Native 用户体验。

（3）考虑未来——HTML5的可用性和功能都在迅速改进。许多分析师预测，它可能会成为开发前端 App 的默认技术。如果用 HTML 来编写 App 的大部分代码，并且只有在需要时才使用 Native 代码，公司就能确保他们今天的投入在明天不会变得过时，因为 HTML 功能变得更丰富，可以满足现代企业一系列更广泛的移动要求。

4、混合开发框架和层次结构图



混合开发结构图



1） 页面加载

页面容器（XdjaWebView）类，是整个框架的核心和基础，主要用来实现页面的加载，以及对页面加载完成后的后续操作提供支持，例如：文件下载、js支持、文件上传，数据缓存、进度条等；

页面加载接口：对页面的加载过程进行跟踪；例如：页面加载进度百分比，页面开始加载、页面加载出错、页面加载完成等

2） JS调用Android功能

网页：页面调用js接口中的具体方法；

JS接口：调用android接口中一一对应的具体方法；

android接口：直接调用框架中集成的功能，或者通过框架接口在应用系统中自定义功能（例如，退出、返回键响应等)；其中升级功能的返回结果或者过程信息，可以在客户端中通过升级接口获取。

XdjaClientHelper：如果需要将框架中的方法返回值通知给js方法，你们可以通过XdjaClientHelper类来实现；

3）应用系统调用JS功能

应用系统通过XdjaClientHelper来实现对js功能的调用；

4） 应用系统调用HDF功能

应用系统可以调用框架集成的工具类、消息提示框、升级模块以及手机上常见的打电话发短信等功能。

## 1.为什么16年H5大火

16年，几乎所有的设备都是4.0以上了,而且4.4以上已经超过70%,特别是5.0以上都已经超过40%了,而Android 4.4以上对H5的支持就已经很不错了,所以我们几乎以及可以肆无忌惮的使用H5了

在Html5没有兴盛之前,加载的Html往往只能用来做一些简单的静态资源显示,但是H5大行其道以后,Html5中有很多新增的功能,炫酷的效果,特别是iOS中H5支持一直都很良好,Android 4.4以上支持也足够,所以这时候发现可以将一些主要的逻辑都用H5页面来编写,然后原生直接用webview加载显示,这样大大提高了开发效率,而且体验也很不错

## Hybrid定义

前面有提到Hybrid这种模式,那么它是怎么样定义的呢？怎么样的开发模式才算是Hybrid模式呢?

* **Hybrid是半Native半web开发模式**

Hybrid模式中,底层功能API均由原生容器通过某种方式提供,然后业务逻辑由H5页面完成,最终原生容器加载H5页面,完成整个App

* **成熟的Hybrid模式意味着业务逻辑均由H5实现**

一款成熟的Hybrid框架,意味着各种类型的api都很完善,那么这时候几乎所有与业务相关的逻辑都是放在H5页面中的,原生只作为容器存在

* **成熟的Hybrid模式可复用性非常高,可以跨平台开发**

成熟的Hybrid框架,那么原生只会提供底层API,也就是说所有的业务是H5完成,不管是什么项目,业务只由H5实现,这时候就可以发现,业务代码是可以跨平台的,也就是说,开发一次,就可以和各自原生容器结合,组成两种原生安装包了,达到了跨平台开发效果

1）移动终端web壳（以下简称“壳”）：壳是使用操作系统的 API 来创建嵌入式 HTML的渲染引擎。壳主要功能是定义Android应用程序与网页之间的接口，允许网页中的JavaScript调用Android应用程序，提供基于web的应用程序的Android API，将Web嵌入到Android应用程序中。

## Hybrid App的类型划分

上面提到过Hybrid的定义,但实际上,根据Native和web的混合程度,Hybrid也可以再次细分为多种类型(参考百科上的说法)

* **多View混合型**

这种模式主要特点是将webview作为Native中的一个view组件,当需要的时候在独立运行显示,也就是说主体是Native,web技术只是起来一些补充作用（只是用来暂时，不进行和原生api的任何交互）

这种模式几乎就是原生开发,没有降低什么难度,到了16年几乎已经没人使用了

* **单View混合型**

这种模式是在同一个view内,同时包括Native view和webview(互相之间是层叠的关系),比如一些应用会用H5来加载百度地图作为整个页面的主体内容,然后再webview之上覆盖一些原生的view,比如搜索什么的

这种模式开发完成后体验较好,但是开发成本较大,一般适合一些原生人员使用

* **Web主体型**

这种模式算是传统意义上的Hybrid开发,很多Hybrid框架都是基于这种模式的,比如PhoneGap,AppCan,Html5+等

这种模式的一个最大特点是,Hybrid框架已经提供各种api,打包工具,调试工具,然后实际开发时不会使用到任何原生技术,实际上只会使用H5和js来编写,然后js可以调用原生提供的api来实现一些拓展功能。往往程序从入口页面,到每一个功能都是h5和js完成的

理论上来说,这种模式应该是最佳的一种模式(因为用H5和js编写最为快速,能够调用原生api,功能够完善),但是由于一些webview自身的限制,导致了这种模式在性能上损耗不小,包括在一些内存控制上的不足,所以导致体验要逊色于原生不少

当然了,如果能解决体验差问题,这种模式应当是最优的(比如由于iOS对H5支持很好,iOS上的体验就很不错)

* **多主体共存型（灵活型）**

这种模式的存在是为了解决web主体型的不足,这种模式的一个最大特点是,原生开发和h5开发共存,也就是说,对于一些性能要求很高的页面模块,用原生来完成,对于一些通用型模块,用h5和js来完成

这种模式通用有跨平台特性,而且用户体验号,性能高,不逊色与原生,但是有一个很大的限制就是,采用这种模式需要一定的技术前提

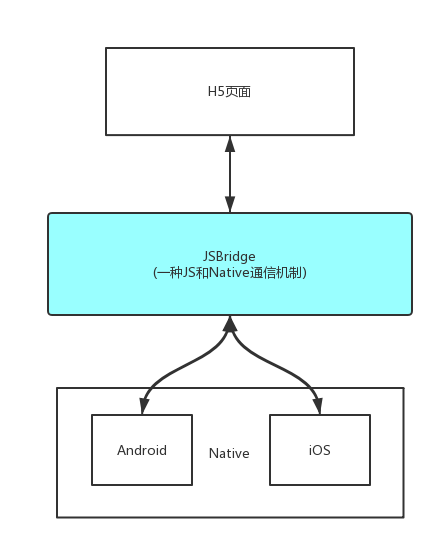
也就是说这种模式不同于web主体型可以直接用第三方框架,这种模式一般是一些有技术支持的公司自己实现的,包括H5和原生的通信,原生API提供,容器的一些处理全部由原生人员来完成,所以说,使用这种技术的前提是得有专业的原生人员(包括Android,iOS)以及业务开发人员(原生开发负责功能,前端解决简单通用h5功能)

当然了,如果技术上没有问题,用这种方案开发出来的App体验是很好的,而且性能也不逊色原生,所以是一种很优的方案

## Hybrid架构

### 基本原理

如下图,痛过JSBridge,H5页面可以调用Native的api,Native也可调用H5页面的方法或者通知H5页面回调



### 内部的实现原理流程

知道了Hybrid的基本原理,那么Hybrid模式内部是如何实现的呢?H5和Native直接的通信又是如何实现的呢?

请参考 后续系列文章

## Hybrid的未来

### 现行多种App开发模式以及分析比较

现在的App开发,除了Hybrid,还有Native,纯web,React Native等方案,下面介绍下各种方案的分析对比

参考 [Native、Hybrid、React Native、Web App方案的分析比较](http://www.cnblogs.com/dailc/p/5930238.html)

### Hybrid面临的挑战

比如Facebook推出的React Native方案,这是Facebook在放弃h5后自行推出一个'反H5方案',一句话总结就是:里面可以用JS来完整的写一个原生应用

Google的flutter

比如微信推出的小程序(16年9月份内测),这也是一个微信主导的'反H5方案',一句话总结就是:里面可以同JS+微信自制的UI方案来写一个类似于原生的应用,只不过这个应用不是发布到App Store中,而是发布到微信中

像以上技术都不断的在冲击着Hybrid模式(当然Native也会有影响),不过都很推崇JS(话说很多前端猿一直希望JS一统天下)