开发者所需要知道的 iOS 9 SDK 新特性

jopen (/news/user/profile?uid=37924) 2015-06-11 10:41:23 • 发布

摘要：年年岁岁花相似，岁岁年年人不同。今年的 WWDC 一如既往的热闹，但是因为要照顾家里刚出生的宝宝以及宝宝的娘，就只能在家里的“窝里蹲”家庭影院来关注这一全球 Apple 开发者的盛会了。

年年岁岁花相似，岁岁年年人不同。今年的 WWDC 一如既往的热闹，但是因为要照顾家里刚出生的宝宝以及宝宝的娘，就只能在家里的“窝里蹲”家庭影院来关注这一全球 Apple 开发者的盛会了。

生命不息，学习不止。一如以往几年，我会陆续写一些关于 WWDC 和新的 SDK 里我觉得有意思和我自己重点关注和学习的内容。现在回头看前几年写的东西，愈来愈感觉到以前青葱岁月的自己真是傻得可爱。不过一路以来的成长轨迹倒是很明 显，也希望自己能就这样淡然地将这段旅程继续下去。

矫情结束，该干活了。让我们来看看今年的 WWDC 中我认为的开发者需要关注的一些内容吧。

总览

iOS 9 时代开发者面临的最大的挑战和最急切的任务可能有两个方面，首先是如何利用和适配 iPad 的新的分屏多任务特性，其次是如何面对和利用 watchOS 2 来构建原生的手表 app。另外的新课题基本就都是现有框架的衍生和扩展，包括从单元测试扩展到 UI 测试，如何进一步占领和使用系统的通知中心及搜索页面，以及 Swift 2 的使用等。

可以说，经过了 iOS 7 和 iOS 8 连续两次重量级的变革和更新，对普通的 app 开发者来说，iOS 9 SDK 略归于缓和和平静，新的 SDK 在 API 和整体设计上并没有发生什么非常巨大的改变。开发者们也正可以利用这个机会喘息一下，尽快进一步熟悉和至少过渡到使用 iOS 8 SDK 的内容来构筑自己的 app (比如尝试使用 Size Class 和 Presentation Controller (http://onevcat.com/2014/07/ios-ui-unique/) 等)，尽快提升自己的职业技能和制作的 app 的水平，并保证能跟上滚滚向前的 Apple 车轮，应该是今年 Cocoa 开发者们的主要任务。

Multitasking

这可以说是 iOS 9 最大的卖点了。多任务特性，特别是分屏多任务使得 iPad 真正变得像一个堪当重任的个人电脑。虽然在很早以前就已经有越狱插件能让 iPad 同时运行多个程序，但是 Apple 还是很谨慎地到 2015 年才在自己性能最为强劲的移动设备上实装这个功能。iOS 9 中的多任务分为三种表现形式，分别是临时调出的滑动覆盖 (Slide Over)，视频播放的画中画模式 (Picture in Picture) 以及真正的同时使用两个 app 的分割视图 (Split View)。现在能运行 iOS 9 的设

备中只有最新的 iPad Air 2 支持分割视图方式，但是相信随着设备的更新，分割视图的使用方式很可能成为人们日常使用 iPad 的一种主流方式，因此提早进行准备是开发者们的必修功课。

虽然第一眼看上去感觉要支持多任务的视图会是一件非常复杂的事情，但是实际上如果你在前一年就紧跟 Apple 步伐的话，就很简单了。滑动覆盖和分割视图的 app 会使用 iOS 8 引入的 Size Class 中的 Compact Width 和 Regular Height 的设定，配合上 AutoLayout 来进行布局。也就是说，如果你的 app 之前就是 iPhone 和 iPad 通用的，并且已经使用了 Size Class 进行布局的话，基本上你不需要再额外做什么事儿就已经能支持 iOS 9 的多任务视图了。但是如果不幸你还没有使用这些技术的话，可能你会需要尽快迁移到这套布局方式中，才能完美支持了。

视频 app 的画中画模式相对简单一些，如果你使用 `AVPlayerViewController` 或者 `AVPlayerLayer` 来播放视频的话，那什么都不用做就已经支持了。但如果你之前选择的方案是 `MPMoviePlayerController` 或者 `MPMoviePlayerViewController` 的话，你可能也需要今早迁移到 AVKit 的框架下来，因为 Media Player 将在 iOS 9 被标记为 deprecated 并不再继续维护。

watchOS 2

在新的 watchOS 2 中，Watch App 的架构发生了巨大改变。新系统中 Watch App 的 extension 将不像现在这样存在于 iPhone 中，而是会直接安装到手表里去，Apple Watch 从一个单纯的界面显示器进化为了可执行开发者代码的设备。得益于此，开发者们也可以在 extension 中访问到像数字表冠和 (虽然都只是很初级的访问，但是聊胜于无) 心跳计数这样的情报。虽然有所进步，但是其实 Apple 在 watchOS 2 里表现出来的态度还是十分谨慎，这可能和初代 Apple Watch 的设备限制有很大关系，所以实际上留给 app 开发者的电量和性能空间并不是十分广阔。但是相比起现在的 WatchKit 来说，可以脱离 iPhone 运行本身就是了不起的进步了。而为了和 iPhone 进行通讯，现在还添加了 WatchConnectivity 这个新框架。我们有足够的理由期待 Apple Watch 和 WatchKit 在接下来两三年里的表现。

UI Test

在开发领域里，测试一直是保障产品质量关键。从 Xcode 4 以来，测试在 app 开发中的地位可谓是逐年上升。从 XCT 框架的引入，到测试 target 成为新建项目时的默认，再到去年加入的异步代码测试和性能测试。可以说现在 Xcode 自带的测试框架已经能满足绝大部分单元测试的需求了。

但是这并不够。开发一个 iOS app 从来都是更注重 UI 和用户体验的工作，而简单地单元测试可以很容易地保证 model 层的正确，却很难在 UI 方面有所作为。如何为一个 app 编写 UI 测试一直是 Cocoa 社区的难题之一。之前的话有像是 KIF (https://github.com/kif-framework/KIF)，Automating (https://developer.apple.com/library/ios/documentation/DeveloperTools/Concept
甚至是 FBSnapshotTestCase (https://github.com/facebook/ios-snapshot-test-case) 这种脑洞大开的方案。今年 Apple 给出了一个更加诱人的选项，那就是 Xcode 自带的 XCUITest 的一系列工具。

和大部分已有的 UI 测试工具类似，XCUI 使用 Accessibility 标记来确定 view，但因为是 Apple 自家的东西，它可以自动记录你的操作流程，所以你只需要书写最后的验证部分就可以了，比其他的 UI 测试工具方便很多。

Swift 2

Swift 经过了一年的改善和进步，现在已经可以很好地担任 app 开发的工作了。笔者自己也已经使用 Swift 作为日常工作的主要语言有半年多时间了，这半年里的总体感觉是越写越舒畅。Swift 2 里主要的改动是错误处理方面的变化，Apple 从 Cocoa 传统的基于 `NSError` 错误处理方式变为了 throw catch 的异常处理机制。这个转变确实可以让程序更加安全，新增的 ErrorType 也很好地将错误描述进行了统一。但是在实际接触了一两天之后，在语法上感觉要比原来的处理写的代码多一些。可能是长久以来使用 NSError 的习惯导致吧，笔者还并没有能很好地全面接受 Swift 2 中的异常机制。不过这次 Apple 做的相对激进，把 Cocoa API 中的 error 全数替换成了 throw。所以不管情不情愿，转型到异常处理是 Swift 开发者必须面对的了。

另外 Apple 新加了一些像是 `guard` 和 `defer` 这样的控制流关键字，这在其他一些语言里也是很实用的特性，这让 Swift 的书写更加简化，阅读起来更流畅。为了解决在运行时的不同 SDK 的可用性的问题，Apple 还在 Swift 2 里加入了 avaliable 块，以前我们需要自己去记忆 API 的可用性，并通过检查系统版本并进行对比来做这件事情。现在有了 avaliable 检测，编译器将会检查出那些可能出现版本不匹配的 API 调用，app 开发的安全性得到了进一步的保障。为了让整个 SDK 更适合 Swift 的语法习惯，Apple 终于在 Objective-C 中引入了泛型。这看似是 Objective-C 的加强，但是实际上却实实在在地是为 Swift 一统 Apple 开发开路。有了 Objective-C 泛型以后，用 Swift 访问 Cocoa API 基本不会再得到 `AnyObject` 类型了，这使得 Swift 的安全特性又上了一层台阶。

最后是 Swift 2 开源的消息。Swift 的编译器和标准库将在今年年底开源，对于一般的 app 开发者来说可能并不会带来什么巨变，但这确实意味着 Swift 将从一门 app 制作的专用语言转型为一门通用语言。最容易想到的就是基于 Swift 的后端开发，也许我们会在看到 Javascript 一统天下之前就能先感受一下 Swift 全栈的力量?

App Thinning

笔者在日本工作，因为这边大家流量都是包月且溢出的，所以基本不会有人对 app 的尺寸介意，无非就是下载 5 秒还是 10 秒的区别。但是在和国内同行交流的时候，发现国内 app 开发对尺寸的要求近乎苛刻。因为 iOS app 为了后向兼容，现在都同时包含了 32 bit 和 64 bit 两个 slice。另外在图片资源方面，更是 1x 2x 3x 的图像一应俱全 (好吧现在 1x 应该不太需要了)。而用户使用 app 时，因为设备是特定的，其实只需要其中的一套资源。但是现在在购买和下载的时候却是把整个 app 包都下载了。

Apple 终于意识到了这件事情有多傻，iOS 9 中终于可以仅选择需要的内容 (Slicing) 下载了。这对用户来说是很大的利好，因为只需要升级到 iOS 9，就可以节省很多流量。对于开发者来说，并没有太多要做的事情，只需要使用 asset catalog 来管理素材标记 2x 3x 就可以了。

给 App 瘦身的另一个手段是提交 Bitcode 给 Apple，而不是最终的二进制。Bitcode 是 LLVM 的中间码，在编译器更新时，Apple 可以用你之前提交的 Bitcode 进行优化，这样你就不必在编译器更新后再次提交你的 app，也能享受到编译器改进所带来的好处。Bitcode 支持在新项目中是默认开启的，没有特别理由的话，你也不需要将它特意关掉。

最后就是按需加载的资源。这可能在游戏中应用场景会多一些。你可以用 tag 来组织像图像或者声音这样的资源，比如把它们标记为 level1，level2 这样。然后一开始只需要下载 level1 的内容，在玩的过程中再去下载 level2。或者也可以通过这个来推后下载那些需要内购才能获得的资源文件。在一些大型游戏里这是很常见的优化方法，现在在 iOS 9 里也可以方便地使用了。

人工智能和搜索 API

如果说这届 WWDC Keynote 上还有什么留给我印象深刻的内容的话，我会给更加智能的手机助理投上一票。虽然看起来还很初级，比如就是插入耳机时播放你喜欢的音乐，推荐你可能会联系的 人和打开的 app 等，但是这确实是很有意义的一步。现在的 Siri 只是一个问答系统，如果上下文中断，“她”甚至不记得前面两句话说了些什么。一个不会记住 Boss 习惯的秘书一定不是一个好护士，而 Apple 正在让 iPhone 向这方面努力。好消息是我们大概暂时还不用担心会碰到故意不通过图灵测试的机器，所以在人工智能上还有很大的空间可以发挥。

而搜索 API (https://developer.apple.com/library/prerelease/ios/releasenotes/General/WhatsNDontLinkElementID_1) 实质上让 app 多了一个可能的入口。有些用户会非常频繁地使用搜索界面，这是一个绝好的展示你的 app 和提高打开率的机会。如果 app 类型合适的话，这是非常值得一做的追加特性。

游戏相关

游戏类的 app 因为在不同的移动平台上的用户体验并没有鸿沟似的差异，所以是最容易跨平台的 - 毕竟现在无论哪个开发商都无法忽视安卓的份额。这也是 Apple 自家的 SpriteKit 和 SceneKit 这样的游戏框架一直不温不火的原因。比起被局限在 Apple 平台，更多的开发商选择像是 Unity 或者 Cocos2d-x 这样的跨平台方案。但是今年 Apple 还是持续加强了游戏方面的开发工具支持，包括负责状态机维护和寻路等的 GameplayKit 框架，负责录像和回放游戏过程的 ReplayKit 框架，以及物理建模的 Model I/O 框架。

这些其实都是在 Apple 的游戏开发体系中补充了一些游戏业界已经很成熟的算法和工具，为开发者节省了不少时间。对于个人开发者自制的游戏来说，Apple 的工具提供了相对低的门槛，易于上手。但是在现在大部分游戏开发都需要跨平台的年代，总感觉 Apple 体系是否能顺利走下去还需要进一步观察。

其他

HomeKit，CloudKit，HealthKit 等等杂七杂八的框架。如果是 iOS Only 的 app 的话，使用 CloudKit 做 BaaS (http://en.wikipedia.org/wiki/Mobile_Backend_as_a_service) 也许是不错的选择，但是也要面临今后跨平台数据难以共享的风险。其他几个框架专业性相对较

强，大部分需要配合硬件支援，其实一直说智能硬件是下一个爆点， 但是至少现在为止还没能爆出大的声响，更多的却已经进入到廉价竞争 (手环什么的你懂的)，只能说期待这些设备的后续表现吧。

最后是一个对于刚入门或者打算投身到 Apple 开发中的朋友的福利。现在你可以不需要加入付费的开发者计划就能将 app 部署到自己的设备上了，而在以前这至少需要你加入 99 美金每年的开发者计划，这可以说进一步降低了进行 Apple 开发的门槛。

总结

正如上面提到的，对开发者来说，今年的 WWDC 并没有像 13 年和 14 年那样颠覆性的变化，大多是对已有特性的加强补充和对开发工具链的增强。今年可以说是一个 Cocoa 开发者们沉淀之前知识，增进自己技能的好机会。现在 WWDC 15 还在如火如荼的进行之中。如果你打算尽早拥抱新 SDK 的变化的话，请不要犹豫，直接访问 Apple 的开发者网站 (https://developer.apple.com/)，去寻找和观看自己感兴趣的话题吧。

来自：http://www.infoq.com/cn/news/2015/06/ios9-sdk (http://www.infoq.com/cn/news/2015/06/ios9-sdk)

iOS 9 (/news/tags/iOS 9)