怎么面试架构师 (http://casatwy.com/zen-yao-mian-shi-jia-gou-shi.html)

Date Sun 14 June 2015 Tags architect (http://casatwy.com/tag/architect.html) / interview (http://casatwy.com/tag/interview.html)

其实本文想说的是：当面试一个架构师的时候，我们应该问什么问题？我觉得，问什么样的问题，体现了team leader更加看重架构师的哪些特点。

我一直认为，做技术就跟练武一样，在练武的不同阶段，分招式和心法。技术也一样，在不同的阶段，也分招式和心法。另外，就我个人而言，经常忘记招式，一方面可以说十二年来，我用过的招式很多，到了现在也不记得几个。另一方面我自己也不会特意去记。事实上，十二年代码写下来，我反而越来越不关注招式，而是越来越关注如何解决问题，也就是心法。所以我作为team leader的时候，我会更加看重这个架构师候选人是不是有一套属于自己的心法。

上面说的听着很玄，下面我就直接回到正题：我们面试架构师候选人时，应该问什么样的问题？

大致会有几种类型的问题：

1. 当前技术领域中的一些技术细节
2. 算法和数据结构
3. 方案设计思路

当前技术领域的技术细节类问题

针对第一类问题，我认为是很有必要问的，架构师对技术细节的理解，是很能够影响他做架构时的设计思路的。毕竟每一个领域都有不同，了解不同领域的差异，以及特定领域的技术细节，很影响架构时的设计思路和实现手段。

Social

RSS (http://casatwy.com/feeds/all.atom.xml)

github (http://github.com/casatwy)

facebook (https://www.facebook.com/taloyum)

google+ (https://plus.google.com/u/0/108264119649922067163)

weibo (http://weibo.com/casatwy)

Tags

(http://casatwy.com/)

Links

casatwy (http://casatwy.com/)

刘坤的技术博客 (http://blog.cnbluebox.com)

齐道长的博客 (http://qitaos.github.io)

然而，这并不是鼓励大家去挖出各种细节的问题，然后去考察架构师候选人，这里需要有一个度。举个例子：

> 你如何去把一个view的所有subview清空?

1. 如果知道NSArray有makeObjectsPerformSelector这个方法的人，他们能够说出直接使用这个方法，然后在selector里面写removeFromSuperView的selector，就好了，而且很省事，一句话就搞定。
2. 如果知道NSArray有enumerator方法的人，他们会说出使用这种方法枚举每一个subview，在block里把removeFromSuperView调用起来，也差不多两三行的事儿。
3. 不知道NSArray有上面这些方法的人，他会说用for...in...的方法遍历，然后取到这每一个subview，让他们执行removeFromSuperView。可能要花费大概四五行。

这几种答案谁的更好？在我看来一样好。为什么？

因为这个问题其实考察的是这个人知不知道某个方法，当然你可以说他知道这个方法是因为他仔细看过文档或者头文件。但除了这个以外，这个问题对判断这个人是不是一个合格的架构师没有任何意义。架构师的任务在于使用合理的手段完成架构的任务，上面三种做法都是合理的手段，只不过是实现技巧上的不同而已。

这样的问题还可以拓展开来：你完全可以问一个架构师候选人某一个领域的这种类似问题，而恰好你比他熟悉，如果候选人答不上来，你会认为他可能在这方面花的时间还不够，这方面的理解不够深，导致减分。但如果答上来了，有可能加分有可能不加分。

然而，这一切并没有什么卵用。如果角色对调，让候选人来面试你，他完全可以问出各种这样类似的问题，一样让你抓耳挠腮百思不得其解。那么该如何考察一个架构师候选人对自己领域中技术细节的理解呢？我们来看下面这些问题：

> 你觉得block当初是为了解决什么样的问题而设计的？你如何区分何时使用block，何时不使用?
> 你觉得ReactiveCocoa当初是为了解决什么样的问题而设计的？你何时会考虑使用RAC，何时不用?
> 你觉得MVVM这样的思想是为了解决什么样的问题而产生的?

答案在本文不是重点，当然如果各位对答案感兴趣，可以在评论区问一下，我在评论区回答。在我遇到的各种面试官中，我从来没遇到过能问出这样类似问题的面试官 。我面试别人的时候，我问过这种比较侧重对某一项技术的理解的问题，有人能答好有人答不好，然后从招进来的人看，当初答好这种问题的人，后来都在团队中起到了顶梁柱的作用。答不好这样问题的人，但是他们因为知道很多技术细节，也还是招进来了，虽然也能很好地完成需求和任务，但是代码结构、设计思路都会有或多或少的缺陷，写出来的组件在使用上也会感觉怪怪。

所以，考察一个架构师候选人在某一领域的技术时，通用的技术细节的问题可以问一下，偏门的技术细节问出来就很没有意义。一个架构师最关键的是他对技术的理解深度，理解深刻的人，才能写出简单易用易拓展的架构。然后面试官需要区分好问题，有

些问题是属于“知道、不知道”，有些问题是属于“理解、不理解”，对于面试一个高级工程师来说，可能会比较偏向前者，因为他需要知道足够多，然后完成需求的速度才快，不需要总是去Google。但对于面试一个架构师来说，其实大部分基础知识应该是已经具备了的，不至于写个TableView还要去翻Google。但在做SDK的时候，是会遇到一些偏门问题的，是需要去Google的。但架构师跟高级工程师的区别就在于，架构师知道该往哪个方向去Google，能够把握住问题的实质去解决问题。所以对于考察架构师而言，应该更加偏向后者，理解和不理解。

回想一下，其实有很多类似 知道、不知道 的问题，你是在code review中，其他人的博客中，文档中就能学到的。但是那些 理解、不理解 的问题，其实大部分都是你多年代码的经验思考出来的，即便你去看了博客看了文档，该不理解的还是不理解。而作为一名架构师，真正要考察的就是 理解、不理解 的问题。所以你明白，为什么当初那些技术细节答不上来的人，但是对技术理解很深刻的人能成为顶梁柱，成为架构师。而技术细节知道很多，但技术理解不深刻的人还是只能做高级工程师的原因了吧?

算法和数据结构类问题

第二类问题，算法和数据结构相关的问题。这种问题也是很需要问的，但似乎现在在社招的时候会问这种问题的面试官不太多，只有在面试比较初级的人或者应届生的时候才会拿来问。

我觉得面试官即便在面试架构师的时候，还是要问这样的问题的，只是要注意考察侧重点。一个架构师如果不了解数据结构和算法，那他真的很难做出靠谱的架构，毕竟很多SDK底下充斥着各种各样的数据结构，而且有经验的人都很清楚，对于一类数据而言，不同的结构设计或表达方式，很影响最终实现的方案的优雅程度。所以我们面试架构师时，侧重点在于，对于某个问题，你如何去选择合适的数据结构，合适的算法来解决这样的问题。

但是，在面试应届生时，我们问算法和数据结构问题时，其实更加关注的是他的动手能力，给一个很简单的问题，然后让他把代码写出来，或白板，或IDE。就国内大部分公司招聘的情况和其公司自身的情况来看，如果你学facebook / google那样出算法题，你基本上招不到人。因为会这些题目的人，都在facebook / google那儿排队呢。

然后算法和数据结构相关的问题第二个考察点在于，候选人的思考是否足够细密。这个不管是对架构师候选人，还是对应届生还是对社招的高级工程师而言，重要程度都是一样的。这个就不多说了。

你让一名架构师候选人在面试的时候做一个华容道算法，在你而言其实是对他的一种鄙视，在他而言他也很有可能写不出。但如果你让一名架构师候选人在面试时候展示他对各数据结构的理解，不同场景下如何设计合理的数据结构和算法，如何权衡时间与空间的取舍，这才是对他的一种重视。

方案设计思路类问题

第三类问题，方案设计思路。大概一年以前我在面试携程的时候，遇到过面试官问我这种问题，其它我就没有遇到过了，一般都是我在自我介绍的时候主动挑一个去讲。我在面试别人的时候，我也会问这样的问题，比如说：

对于一个app的网络层，你在设计时，你会考虑哪些问题?
对于一个app的持久层，如果让你直接用sqlite，你如何设计版本迁移方案?
工作中，你会采用哪些手段来做解耦?

严格来说，大部分面试官也会问这样的问题，但是是看到你简历上写过你有这个经验，然后直接问这个方案你是怎么做的，而不是问这个方案你是怎么设计的。在我看来，大部分方案的实现其实没有什么技术含量，真正有技术含量的地方在于，拿到这个问题时，你是如何思考的。就比如数据库版本迁移方案，设计的过程是很艰苦的，但设计完毕实现的时候，就是码代码，不能说完全没有技术含量，只能说实现的时候所需要耗费的脑力跟设计时候比，差太远了，在我看来属于没有什么技术含量。

说到技术含量的事情，我也遇到过特别多的面试官喜欢问这个问题：过去你解决了哪些比较有技术含量的问题？我一般不会拿这个问题去问候选人，因为我觉得真的到了代码层面，是基本上不存在技术含量的概念的，码代码这个工作本身，就是用计算机能懂的方式告诉计算机应该怎么做事，其实就是一件很没技术含量的事情。

所以我认为的技术含量是，你如何去设计一个靠谱的解决方案，这个解决方案足够周密，思考足够长远，提供的API很好看，代码很容易阅读，很好维护。

还有就是逃不掉的23种设计模式。设计模式这种东西早年被业界说了很多，都说烂了，但我不否认的是，这种对设计方法的总结，是每个架构师的起步和入门。如果一个架构师连什么场合使用设么设计模式都分不清楚，各种设计模式他的设计初衷和希望解决的问题都不知道，那他算是不合格的架构师。然而面试官也很少会去问这样的问题，一方面可能觉得问这种问题很low，另一方面其实也有少部分面试官对设计模式仅仅处在了解和知道的情况，不敢随便拿出来问。

总结

面试架构师其实是一件不容易的事情，能考察架构师候选人实力的面试官，首先自己就已经对架构本身有了很好的理解，就应该是一个合格的架构师，其次是需要足够务实，有合理的手段合理的问题，通过面试来了解候选人是不是一个适合做架构师的人。最后，要有足够识人的眼光以及合适的判断标准，通过候选人的回答，对候选人进行筛选。从我对目前面试的情况来看，对这个我持悲观态度。大部分面试官给候选人的感觉更多的是：我问你一个这个问题，看你知不知道？而不是：我问你一个这个问题，看你怎么去思考?

架构师和更高级的高级工程师之间，还是有区别的。所以各家公司如果要想找到合理靠谱的架构师，还是很不容易的。

评论系统我用的是Disqus，不定期被墙。所以如果你看到文章下面没有加载出评论列表，翻个墙就有了。

我的博客拒绝挂任何广告，如果您觉得文章有价值，可以通过支付宝扫描下面的二维码捐助我。

Comments

69 条评论 Casa Taloyum 1 登录

Recommend 9 分享 按从新到旧排序

Join the discussion…

杨晓霞 · 2个月前
看了博主的文章，对自己有了新的认识和追求，非常感谢!
回复 · 分享 ›

气定神闲 · 5个月前
试着回答一下吧，只是自己的看法。
1.block的设计是为了解决方法中方法可做为参数传入当前上下文中，将方法具体的实现进行抽象，一种策略设计模式的具体表现。
2.ReactiveCocoa通过信号将层与层之前的通信统一，方便层之间的通信，和状态同步，然而冷热信号之类的也看的头大，认知曲线也相当陡峭。在viewController业务逻辑非常复杂，类极大的情况，一般如果设计的好也不太会发生，MVC的根本问题是C层装啥，C变成C+VM也不见得解决了多少问题，关键是服务层service或VM是不是设计的颗粒恰当。不可否认ReactiveCocoa会使代码减少，VM可测试，但相对反应就是排错会变的复杂，一个signal很难定位问题出现的层。简单来说，小项目不行，人水平低也不适合。Controller比较简单也最好不用。
3.MVVM是想解决MVC中Controller中包含业务逻辑的问题。Controller一般是不会重用的啊，但VM（业务逻辑）一般是重用的问题。VM通过一种统一信息手段，达到可测试，可重用。
回复 · 分享 ›

刘小壮 · 7个月前
请指教文中提到的问题：你觉得block当初是为了解决什么样的问题而设计的?
1 回复 · 分享 ›

xiang li · 7个月前
虽然还没有架构师的级别，但是还是说说自己的看法：
1. block的设计可以代替很多以往用代理模式的场景，因为用block可以很方便的传递上下文数据，而不用还用额外的属性变量去存储值。
2. ReactiveCocoa 用的不多，可以代替以往通知，delegate，KVO几种模式状态同步的问题。

3. MVVM 解决MVC中Controller过于臃肿，代码可复用性差，单元测试难等问题。
回复 · 分享 ›

正中 赵 · 8个月前
也可以问问他看过哪些项目的代码XD
回复 · 分享 ›

OrlAnd0 · 10个月前
有个关于MethodSwizzling的疑问.
"Swizzling should always be done in a dispatch_once."

↑ Back to top