

100offer
互联网人才拍卖

告别盲目投简历

让海量好机会主动来找你

了解更多

JPUSH SDK

简单、高并发、毫秒级推送

JPUSH极光推送

WWW.JPUSH.CN

[首页](#) > [iOS开发](#)

这些 iOS 面试基础题目，你都深入了解吗？

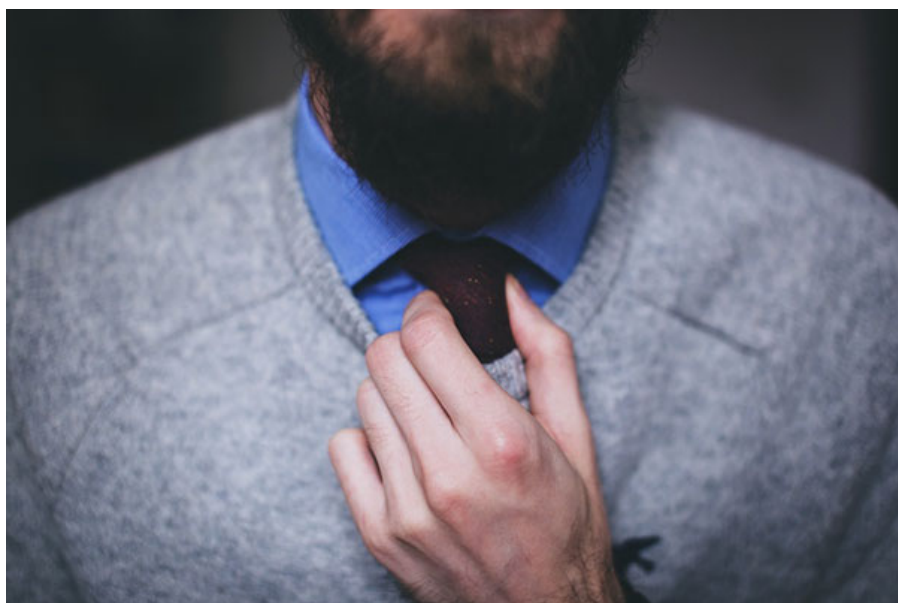
2015-11-06 09:12 编辑: suiling 分类: iOS开发 来源: seedante 的简书

46

12247

[iOS](#) [面试](#) [面试题](#)

招聘信息: PHP开发工程师

作者: [seedante](#) 授权本站转载。题目来自博客: [面试百度的记录](#), 有些问题我能回答一下, 不能回答的或有更好的回答我放个相关链接供参考。

1面

- Objective C runtime library: Objective C的对象模型, block的底层实现结构, 消息发送, 消息转发, 这些都需要背后C一层的描述, 内存管理。
- Core Data: 中多线程中处理大量数据同步时的操作。
- Multithreading: 什么时候处理多线程, 几种方式, 优缺点。
- Delegate, Notification, KVO, other 优缺点

runtime有一点追问, category, method 的实现机制, class的载入过程。1面整体感觉不错, 40分钟不到, 感觉回答的还可以。被通知一会儿二面。

唐巧前辈说这些都是 iOS 的基础问题, 应该对此深入的了解。当初看到时, 大部分回答不上来, 因为平时没有好好思考整理过。这里大部分的概念大多会在学习 OC 的过程中遇到过, 但还是得经过写代码才能有更深的理解。反正我当

热门资讯



这些 iOS 面试基础题目，你都深入了解吗？

点击量 11479



自定义 push 和 pop 实现有趣的相册翻开效果

点击量 8080



Objective-C 的现代语法和新特性

点击量 7944



HTML5 崛起：不再高冷，不再小众

点击量 6463



月薪3万的程序员都避开了哪些坑

点击量 6004



CVP关东升：Swift精华教程汇总（第一期）

点击量 5721



关于 @synchronized，这儿比你想知道的还要

点击量 5545



【译】17个提升iOS开发效率的必用工具

点击量 5266



【译】4个你需要知道的Asset Catalog的秘密

点击量 5256



程序员技术晋升非正式攻略

点击量 4518

综合评论

事实上 老板不傻 挖人他只给一个人涨薪水 如果不换人涨薪水, 全公司其他员工lqcandqq13 评论了 程序员的那些事儿 -- 皆大欢喜的加薪...

事实上 老板不傻 挖人他只给一个人涨薪水 如果不换人涨薪水, 全公司其他员工lqcandqq13 评论了 程序员的那些事儿 -- 皆大欢喜的加薪...

想象是发展的源泉, 不晓得为什么会被这么多人喷
zhouyujangsx 评论了 观点: 原生 app

初看那些设计模式是云里雾里，每个字都认识，就是不知道说的什么。即使现在，有些东西，我也不是很理解。

Objective-C 底层

Objective-C runtime library: Objective-C 的对象模型，Block 的底层实现结构，消息发送，消息转发，category，method 实现，class load。

runtime 我在平时很少涉及到，没有系统学习过，而且很多次看了不久就忘了，所以这里给出一些不错的文章的链接供参考。这几个问题在《iOS 7 Programming Pushing the Limits》都有过深入的解读（我有电子版，是盗版，这里给出这本书在 Github 的地址，工作后我会把去年看过的盗版书全部补偿买回，没有 iOS 8 的版本，不知道是不是由于盗版太多导致的）。另外，唐巧前辈撰文讨论过前两者：

1. 《Objective-C 对象模型及应用》

唐巧在后记中也提到了 iOS 64-bit 带来的变化：

后记

文章发表后，一些同行指出在ARM64的CPU下，isa的内部结构有变化。这点我是知道的，不过希望以后再撰文讨论。感兴趣的同学可以查看苹果今年WWDC2013的视频：《Session 404 Advanced in Objective-C》。

那么就来看看 Session 404 Advanced in Objective-C，从36分起讲相关的东西，喔，看不懂，那还是看看这个吧，在《iOS 7 Programming Pushing the Limits》的 Further Reading: objc_explain_Non-pointer_isa 部分谈论了这个问题。

2. 《谈 Objective-C Block 的实现》

内容非常翔实，特别是关于 Block 类型的部分，强烈建议做下文章开头提到的测试：Objective-C Blocks Quiz。

3. 消息发送和消息转发

消息发送比较好理解，先了解下 runtime 吧，可以查看官方文档《Objective-C Runtime Guide》。之前学习其他语言的时候还没有关注过调用函数的背后发生了什么，在 Objective-C 中，在对象上调用方法称为发送消息，比如 [receiver message];这行代码，编译的时候编译器将之转换为对 底层 C 函数objc_msgSend 的调用：objc_msgSend(receiver, selector); 在运行时，调用哪个方法则完全由 runtime 决定，甚至在运行时可以替换调用的方法，这是 Objective-C 被称为动态语言的根本原因。对于消息转发，说实话我现在还不知道这个的应用场景，看到的大部分博客都是说消息转发给了你补救措施来应对没有没有实现的方法防止 Crash 或者实现类似多继承的机制，我有个疑惑，干嘛不实现那个方法，而要在代价很大的转发机制里处理呢。在《Effective Objective-C 2.0》一书第 12 条 tip 中用 @dynamic 演示了实现动态方法解析的例子来说明消息转发的意义，老实说，我还是没有理解这个的意义。这里有个对官方文档的中文翻译和一些注解。

4. Implement of category and method

找到了来自这位比我厉害得多的90后：《刨根问底Objective – C Runtime（3） – 消息和 Category》（文章原来的链接放进来跟简书的处理有冲突，这里给的是博客地址，而不是这篇文章的具体地址，不过很好找）。

5. Class load

可以看这篇博客：《Objective-C Class Loading and Initialization》，看了下作者的 Github，原来是我以前 follow 过的国外程序员，看人家的 repo 和星星，质量有保障，再看博客文章列表，有很多深入底层的内容，一座宝矿啊。另

正在死亡...

mark

自来也大人 评论了 深入理解RunLoop

啥也不用装，直接用plist文件编辑数据，然后将plist找个XML转json的工具 yuedong56 评论了 如何给App快速搭建虚拟服务器...

没那效果啊 我 没看出来

大神求助啊 评论了 源码推荐(11.04): swift2.0仿微信聊天...

每个app都放广告，我他妈不用手机了 wesinlove 评论了 出大事了！A商要联合程序员们打淘宝分京东了！ ...

猴赛雷

苏小染 评论了 出大事了！A商要联合程序员们打淘宝分京东了！ ...

很好用，好人赞赞赞~~~

广州_虾 评论了 iOS开发——UI组件（个人整理） ...

但是尼玛盛犬不懂啊【摔】 龙之谷不花钱根本活不下去

山风不多行 评论了 “双11”除了剁手 游戏人还应该学到什么？ ...

相关帖子

xcode7发布的app审核通过了，手机上下载很慢

会变换的贝塞尔曲线各路大神都有什么思路吗？

让你从手写Model文件的繁琐过程解脱出来的工具

如何更改配置信息中NDK的路径

向各位大神求助啊~~本人小白，急。。。。

AFNetworking实现断点下载续传，有哪些方法？

升级XCODE7以后，以前的项目在IOS7老系统上闪退，怎么办才好。。

一个美术小菜鸡问几个问题，

关于 I P A 发布的一些疑问

外在《Effective Objective-C 2.0》书中第51节《精简 initialize 与 load 的实现》中也讨论了这个问题，当初看完一头雾水，如今终于能看懂啦。

Core Data: 大量数据多线程同步

这个问题我已经[单独成篇](#)放到这里了，添加了更多的基础知识和介绍。

第一步：搭建 Core Data 多线程环境

这个问题首先要解决的是搭建 Core Data 多线程环境。NSManagedObjectContext 不是线程安全的，你不能随便地开启一个后台线程访问 managed object context 进行数据操作就管这叫支持多线程了。Core Data 对多线程的支持比较好，NSManagedObjectContext 在初始化时可以指定并发模式，有三种选项：

1.NSConfinementConcurrencyType

这种模式是用于向后兼容的，使用这种模式时你应该保证不能在其他线程使用 context，但这点很难保证，不推荐使用。此模式在 iOS 9中已经被废弃。

2.NSPrivateQueueConcurrencyType

在一个私有队列中创建并管理 context。

3.NSMainQueueConcurrencyType

其实这种模式与第 2 种模式比较相似，只不过 context 与主队列绑定，也因此与应用的 event loop 很亲密。当 context 与 UI 更新相关的话就使用这种模式。

搭建多线程 Core Data 环境的方案一般如下，创建一个 NSMainQueueConcurrencyType 的 context 用于响应 UI 事件，其他涉及大量数据操作可能会阻塞 UI 的就使用 NSPrivateQueueConcurrencyType 的 context。环境搭建好了，如何实现多线程操作？官方文档《[Using a Private Queue to Support Concurrency](#)》为我们做了示范，在 private queue 的 context 中进行操作时，应该使用以下方法：

```
1 func performBlock(_ block: () -> Void) //在私有队列中异步地执行 Block
2 func performBlockAndWait(_ block: () -> Void) //在私有队列中执行 Block 直至操作结束才
```

要在不同线程中使用 managed object context 时，不需要我们创建后台线程然后访问 managed object context 进行操作，而是交给 context 自身绑定的私有队列去处理，我们只需要在上述两个方法的 Block 中执行操作即可。而且，在 NSMainQueueConcurrencyType 的 context 中也应该使用这种方法执行操作，这样可以确保 context 本身在主线程中进行操作。

第二步：数据的同步操作

多 context 同步最简单的方案如下：

```
1 NotificationCenter.defaultCenter().addObserver(self,
2 selector: "backgroundContextDidSave",
3 name: NSManagedObjectContextDidSave,
4 object: backgroundContext)
5 func backgroundContextDidSave(notification: NSNotification){
6     mainContext.performBlock(){
7         mainContext.mergeChangesFromContextDidSaveNotification(notification)
8     }
9 }
```

NSManagedObjectContext 在执行保存操作后会发出 NSManagedObjectContextDidSaveNotification，包含了 context 所有的变化信息，包括新增的、更新的以及删除的对象的信息；而

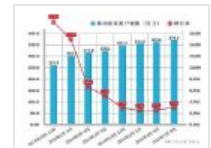
微博



CocoaChina

加关注

【“双11”除了剁手 游戏人还应该学点什么？】双11已经从草根的“光棍节”成为全民的“购物节”，甚至是全民的“剁手节”。而反观中国游戏产业，虽历来不乏让用户掏钱的手段，但却有能让用户心甘情愿的，更提上不上达到欲罢不能的“剁手”境界。详见：<http://t.cn/RUOiRUd>



11月11日 21:02

转发 | 评论

【COC设计师：吸取玩家意见 持续对游戏改进】芬兰手游大厂 Supercell 的代表作《部落冲突》已经在世界规模



mergeChangesFromContextDidSaveNotification(_ notification) 方法则用于合并其他 context 中发生的变化。

如果 context 并未观察其他 context 的 NSManagedObjectContextDidSaveNotification 通知，且保存时，persistent store 已经被其他 context 更改过，那么很可能存在差异，此时同步就有了以下几种选择：选择保存 context 中的版本或者使用 persistent store 的版本替换 context 的版本，又或是将两者的版本融合。这种同步方式由 NSManagedObjectContext 的 mergePolicy 属性决定。

1.NSErrorMergePolicy

默认策略，有冲突时保存失败，persistent store 和 context 都维持原样，并返回错误信息，是唯一反馈错误信息的合并策略。

2.NSMergeByPropertyStoreTrumpMergePolicy

当 persistent store 和 context 里的版本有冲突，persistent store 里的版本有优先权，context 里使用 persistent store 里的版本替换，但是 context 里没有冲突的变化则不会受到影响。

3.NSMergeByPropertyObjectTrumpMergePolicy

与上面相反，context 里的版本有优先权，persistent store 里使用 context 里的版本替换，但是 persistent store 里没有冲突的变化不受影响。

4.NSOverwriteMergePolicy

用 context 里的版本强制覆盖 persistent store 里的版本。

5.NSRollbackMergePolicy

放弃 context 中的所有变化并使用 persistent store 中的版本进行替换。

同步是件很复杂的事情，实际上还是需要根据实际需要来选择同步方案。上面两种方案中第一种概念简单实现容易，第二种比较复杂相对危险，需要谨慎选择同步策略。还有一点需要注意，如果需要跨线程使用 managed object，那么不要直接在其他 context 里使用该 managed object，而应该通过该对象的 objectId 将该对象 fetch 到 context 里。

最后，搞定大量数据

多线程和同步问题解决，最后的难点：大量数据。大量数据意味着需要我们关注内存占用和性能，写代码时需要记得以下规则：

- 1.尽可能缓存需要的数据，不相关的数据保持 faults 状态。
- 2.fetch 时尽可能精准，少引入不相关的数据。
- 3.构建多 context 时尽量将同类 managed object 集中，最大限度减少合并需求。
- 4.提升操作效率，对 Asynchronous Fetch, Batch Update, Batch Delete 等新特性尽可能利用。

多线程编程

在 iOS 编程中，这几种情况下需要处理多线程：UI 事件必须在主线程里进行，其他的可以放在后台进行；而进行一些耗时长或阻塞线程的任务，最后放进后台线程里进行。iOS 的多线程技术有这么几种：线程，GCD 和

NSOperationQueue。线程这种技术比较复杂，而多线程编程向来是「复杂必死」，推荐尽可能使用后二者，但线程有个后二者没有的优势：能够精确保证任务执行的时间。GCD 全称是 Grand Central Dispatch, 是 libdispatch 这个库的外部代号，基于 C 的底层来实现；而NSOperationQueue，通称操作队列，是基于 GCD 实现的。GCD 能做的 NSOperationQueue 基本上都能做，而且还有些 GCD 中不易实现的特性，如挂起、取消任务，虽然在 iOS 8 中，GCD 也提供了取消任务的功能，但在 GCD 中任务的挂起和取消都有较大的局限性；虽然大多数情况下应该使用抽象级别更高的 API，也就是 NSOperationQueue，但处理一般的后台任务我偏爱 GCD，主要是 GCD 搭配 Block 使用简单，非常方便。如何选择，下面两个链接对此问题的讨论值得一看：

[StackOverflow: NSOperation vs. Grand Central Dispatch](#)

[Blog: When to use NSOperation vs. GCD](#)

另外，还推荐这些文章：objc 的并发编程专题《[Concurrent Programming](#)》及[中文翻译版本](#)；雷纯锋的博客《[iOS 并发编程之 Operation Queues](#)》；NSHipster 的《[NSOperation](#)》。

设计模式

评价 Delegate, Notification, KVO 几种设计模式的优缺点

我不觉得这个问题是个好问题，与其比较这几个设计模式的优缺点，不如谈它们各自的特点比较好，因为它们是为了解决某类问题才设计出来的，有各自适合的使用场景。另外，给个 iOS 中设计模式的介绍：[iOS Design Patterns](#)。

为什么出题都喜欢把这三个设计模式拿来对比呢？Notification 和 KVO 都是用于协助对象间的通信：某个对象监听某个事件的发生，当某个事件发生时，该对象会得到通知然后做出响应。这几句话大概是以前看过的书本上说的。如果你以前没接触过设计模式，第一次学习时总是能够看到事件、响应这类模糊的词汇，看得你云里雾里，好吧，我说的是我。但 delegate，应该说没有监听的功能，而是当事件发生或时机到了，要求 delegate 对象做点什么。刚开始学习 OC 的时候，一书中将 delegate 比喻为助手，那时候不怎么理解，现在觉得这个比喻十分恰当。虽然 delegate 模式在 OC 中随处可见，在 UIViewController 类中广泛存在，但在开发 FaceAlbum 的过程中只遇到过一次自定义 protocol/delegate 的情况，后来还是用 KVO 取代了。相对于 Notification 和 KVO 模式，使用 delegate 模式你会明确知道对象的 delegate 能干什么，因为要成为某个对象的 delegate，该对象得遵守指定的 protocol，protocol 指定了 delegate 对象需要实现的方法。

Notification 和 KVO 两者都需要监听事件的对象(早期看见事件就犯晕，如今写来觉得用这个词挺顺手的)去注册，delegate 则需要 delegate 对象遵守指定的 protocol；Notification 中监听者向一个单例对象 NSNotificationCenter 注册，NSNotificationCenter 类似一个广播中心，接受任何对象的注册，后者则向要监听的对象注册，一对一，这两者都不需要对象之间有联系，而 delegate 则需要通信的对象通过变量联系；NSNotificationCenter 模式里监听的对象与被监听的对象通信是通过 NSNotificationCenter 这个中介，而 KVO 里，不能说两者是直接通信的，我没有了解过 KVO 是如何实现通信的，从表面上看两者就那么心灵感应一般，这是系统替我们实现的，而 delegate，由于通过变量连接，直接向 delegate 发送消息即可，在这点上，NSNotification 不需要通信双方知道对方，而后两者则不然；在响应事件时，NSNotification 和 KVO 模式里都是在注册时指定响应方法，而 delegate 则在 protocol 里预定义了响应方法。

说了这么多，不直观，说个实际场景，比如在 UICollectionView 里选择 cell 的时候，希望 title 能够跟踪选中 cell 的数量。这里用 NSNotification 和 KVO 都能实现，但是我更喜欢 KVO，感觉更优雅，因为使用 NSNotification 模式的话，选中一个 cell 的时候要在选择的方法里手动发布通知，而 KVO，只要对观察的属性实现 KVO 兼容的方法就可以了；而 delegate，自己做自己的 delegate，呃。而面对一些系统里的事件，比如键盘的出现与消失，图片库的变化，使用 NSNotification 更加自然，因为 KVO 限于对对象属性的跟踪。

暂时写这么多，推荐博客《[When to use Delegation, Notification, or Observation in iOS](#)》，可能需要翻墙。



微信扫一扫
订阅每日移动开发及APP推广热点资讯
公众号：CocoaChina

我要投稿 收藏文章 分享到：

70

上一篇：iOS开发——UI组件（个人整理）
下一篇：iOS 蓝牙开发（三）app作为外设被连接的实现

相关资讯

【译】17个提升iOS开发效率的必用工具
iOS 蓝牙开发（四）BabyBluetooth蓝牙库介绍
iOS开发——UI组件（个人整理）
苹果向开发者发布 iOS 9.2 第二个测试版
源码推荐(10.29)：cell下拉点击，高仿百度糯米iOS

初窥iOS 9联系人框架
iOS 蓝牙开发（三）app作为外设被连接的实现
[XTPaster] iOS 贴纸功能实现
【投稿】iOS自定义键盘
【译】iOS Xcode部署配置



我来说两句



您还没有登录！请 [登录](#) 或 [注册](#)

所有评论（46）



10711996352015-11-11 05:02:19

路还很长，却想一下子到达终点

0 0 回复



自来也大人2015-11-11 03:27:24













mark

0 0 回复



david_mobile2015-11-10 03:56:10

看了这个清单之后觉得路还是很长的

		0	0	回复
	blueSky1991 mark	2015-11-09 09:03:41		
		0	0	回复
	lgypaopao mark	2015-11-09 08:57:09		
		0	0	回复
	CocoaKun mark	2015-11-09 08:09:52		
		0	0	回复
	lzm470445107 你的二面呢？	2015-11-09 08:00:33		
		0	0	回复
	冬冬_phone mark	2015-11-09 06:41:10		
		0	0	回复
	chihuobing mark	2015-11-09 03:19:36		
		0	0	回复
	琼华羲和 mark	2015-11-09 02:24:33		
		0	0	回复
	zuofei5566 mark	2015-11-09 00:57:07		
		0	0	回复
	missen_xue mark	2015-11-09 00:06:20		
		0	0	回复
	yunyuchen 写的真乱 不知道写的什么	2015-11-08 02:02:58		
		1	0	回复
	NoWhere_Man 接着学习	2015-11-07 16:18:18		
		0	0	回复
	新晋菜鸟 看=晕了	2015-11-07 08:35:13		

👍 0 💬 0 回复



新晋菜鸟

看=晕了

2015-11-07 08:35:12

👍 0 💬 0 回复



杨芳学

mark

2015-11-07 06:02:41

👍 0 💬 0 回复



jiemuyu

mark

2015-11-07 05:45:32

👍 0 💬 0 回复



cfl2005

mark了

2015-11-06 14:50:19

👍 0 💬 0 回复



galaxyfanfan

mark

2015-11-06 12:51:54

👍 0 💬 0 回复

更多评论

[关于我们](#) [商务合作](#) [联系我们](#) [合作伙伴](#)

北京融控科技有限公司版权所有

©2015 Chukong Technologies, Inc.

京ICP备 11006519号 京ICP证 100954号 京公网安备11010502020289  京网文[2012]0426-138号