2015年2月5日 星期四

面试题总结

一、OC的理解及特性

- OC作为一门面向对象的语言，自然具有面向对象的语言特性，如：封装、继承、多态。它具有静态语言的特性(如C++),又有动态语言的效率(动态绑定、动态加载等)。整体 来说,确实是一门不错的编程语言。
- Objective-C具有相当多的动态特性，表现为了三个方面:**动态类型**（Dynamic typing）、**动态绑定**（Dynamic binding）、**动态加载**（Dynamic loading）。之所以叫做动态,是因为必须到运行时(run time)才会做 一些事情。
- 1、动态类型：即运行时再决定对象的类型。这类动态特性在日常应用中非常常见，简单说就是id类型。实际上静态类型因为其固定性和可预知性 而使用得更加广泛。静态类型是强类型,而动态类型属于弱类型。运行时决定接收者。
- 2、动态绑定：基于动态类型，在某个实例对象被确定后，其类型便被确定了。该对象对应的属性和响应的消息也被完全确定，这就是动态绑定。
- 3、动态加载：根据需求加载所需要的资源，这点很容易理解，对于iOS开发来说，基本就是根据不同的机型做适配。最经典的例子就是在Retina设备上加载@2x的图片，而在老一些的普通屏设备上加载原图。随着Retina iPad的推出，和之后可能的Retina Mac的出现，这个特性相信会被越来越多地使用。（让程序在运行时添加代码模块以及其他资源。用户可以根据需要加载一些可执行代码和资源,而不是在启动时就加载所有组件。可执行代码中可以含有和程序运行时整合的新类）。

二、简述对内存管理的理解

- OC内存管理遵循“谁创建，谁释放。谁引用，谁管理”的机制，当创建或引用一个对象的时候，需要向它发送alloc copy retain 消息，当释放该对象时需要发送release消息，当该对象引用计数为0时，系统将释放该对象，这是OC的手动内存管理机制；iOS5.0之后OC又提供了自动管理机制，ARC(automatic reference counting)，管理机制跟手动管理机制一样，只是不再需要调用 retain release autorelease；它是编译时特性，当你启用ARC时，在适当的位置插入release和autorelease；它引用了strong和weak 关键字，strong修饰的指针变量指向对象时，当指针指向新值,或者指针不再存在时,相关联的对象就会自动释放，而weak修饰的指针变量指向对象，当对象的拥有者指向新值或者不存在时weak修饰的指针则自动置为nil，这是ARC管理机制。
- OC是通过引用计数来对内存进行管理的，核心思想是遵循“谁创建，谁释放；谁引用，谁管理”的机制。分为两种方式：一种是ARC（自动内存管理）、一种是MRC（手动内存管理）。
- 如果一个对象有一个 __strong 类型的指针指向着，那么这个对象就不会被释放。如果一个指针指向超出了它的作用域，就会被指向nil。如果一个指针被指向nil，那么它原来

指向的对象就被释放了。当一个视图控制器被释放时，它内部的全局的指针会被指向nil。用法：不管是全局变量还是局部变量用__strong描述就可以了。

- 局部变量：出了作用域，指针会被置为nil。
- 全局变量：当视图控制器被释放时，内部的全局变量会被置为nil。
- 方法内部创建对象，外面使用需要添加 __autorelease 。
- 连线的时候，用__weak 描述。
- 代理用unsafe_unretained就相当于assign。
- block 中为了避免循环引用的问题，使用__weak 描述。
- 声明属性时，不要以new开头。如果非要以new开头命名属性的名字，就需要自己制定一个get方法名如：@property (getter=theString)NSString *newString;
- 如果要使用自动释放池，用@autoreleasepool{ }。
- ARC 只能管理Foundation 框架中的变量，如果你的程序中把 foundation中的变量强制转换成了Core Foundation中的变量需要交换管理权。
- 在非ARC的工程中 采用ARC去编译某些类　　-fobjc-arc。
- 在ARC下的工程中 采用非ARC去编译某些类 -fno-fobjc-arc。

三、你如何理解MVC设计模式

- MVC是一种架构模式,M表示数据模型Model V表示视图View C表示控制器Controller。
- 1、Model 负责存储,定义,操作数据。
- 2、View 用来展示数据给用户,和用户进行操作交互。
- 3、Controller 是Model与View的协调者,Controller 把Model中的数据拿过来给View用。Controller 可以直接与Model和View进行通信，而View不能和Controller直接通信。View与Controller通信需要利用代理协议的方式，当有数据的更新时,Model也要与Controller进行通信,这个时候就用notification 和KVO ， 这个方式就像一个广播一样,Model发信号 ,Controller设置监听器接受信号 ,当有数据要更新时,就发信号给Controller。Model和View不能直接进行通信,因为这样就违背了MVC的设计思想。

四、IOS中的持久化（数据存储）方式，各有什么特点，IOS平台怎么做数据的持久化？CoreData和SQLite有无必然联系？

- 数据存储的核心都是写文件。主要有四种持久化方式：**属性列表**、**对象序列化**、**SQLite数据库**、**CoreData**。

- 属性列表、对象序列化适合小数据量存储和查询操作。
- SQLite、CoreData适合大数据量存储和查询操作。
- plist文件写入：只有 NSString、NSArray、NSDictory、NSData 可以writeToFile ；存储的依旧是plist文件。plist 文件可以存储的7种数据类型：array，dictory，string，bool，data，date，number。
- 对象序列化：对象序列化是通过序列化的形式，键值关系存储到本地，转化成二进制流。实现NSCoding协议必须实现的两个方法。**编码（对象的序列化）**：把不能直接存储到plist文件中得数据，转化为二进制的数据，NSData；可以存储到本地。**解码(对象的反序列化)**:把二进制数据转化为本来的类型。
- SQLite3：大量的有规律的数据使用数据库。
- CoreData：**通过管理对象进行增、删、改、查操作的**。它不是一个数据库，不过可以使用SQLite数据库来保持数据，也可以使用其他的方式来存储数据。如：XML。**CoreData是面向对象的API**；CoreData是iOS中非常中要的一项技术，因为几乎在你编写的所有的应用程序中，CoreData都作为数据存储的基础，**CoreData是苹果官方提供的一套框架，用来解决与对象生命周期管理、对象关系图管理和持久化等方面相关的问题**。大多情况下, 我们引入CoreData作为持久化数据的解决方案,并利用它将持久化数据映射为内存对象。 提供的是对象-关系映射(object-relational mapping)功能(也就是说,CoreData可以将Objective-c对象转换成数据,保存到SQL中,然后也能将保存后的数据还原成OC对象)。
- **CoreData的特征**：
- 1) 通过CoreData管理应用程序的数据模型，可以极大程度减少需编写的代码数量。
- 2) 将对象数据存储在SQLite数据库已获得性能优化。
- 3) 提供NSFetchedResultsController 类用于管理表视图的数据。 即将Core Data的持久化存储显示在表视图中，并对这些数据进行管理：增、删、改。
- 4) 管理undo/redo操作;。
- 5) 检查托管对象的属性值是否正确。

五、线程和进程分别是什么，程序中为什么要使用多线程

- 一个程序至少要有进程，一个进程至少要有一个线程。
- 进程：资源分配的最小独立单位。就是一个应用程序在处理机上的一次执行过程，它可以申请和拥有系统资源；进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动，进程是系统进行资源分配和调度的一个独立单位。
- 线程：进程下的一个分支。是进程的一个实体，是CPU调度和分派的基本单位，它是比进程更小的能独立运行的基本单位。线程自己基本上不拥有系统资源。只拥有一点在运

行中必不可少的资源(如程序计数器、一组寄存器和栈)，但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源。

- 多线程编程是防止主线程堵塞，增加运行效率等的最佳方法。Apple提供了NSOperation这个类，提供了一个优秀的多线程编程方法。一个NSOperationQueue 操作队列，就相当于一个线程管理器，而非一个线程。因为你可以设置这个线程管理器内可以并行运行的的线程数量等等。多线程是一个比较轻量级的方法来实现单个应用程序内多个代码执行路径。iPhone OS下的主线程的堆栈大小是1M，第二个线程开始就是512KB，并且该值不能通过编译器开关或线程API函数来更改，只有主线程有直接修改UI的能力。

六、定时器和线程的区别

- 定时器：可以执行多次，默认在主线程中。
- 线程：只能执行一次 。

七、Apple设备尺寸和编程尺寸

设备	系统	分辨率	屏幕尺寸	倍数
iPhone				
iPhone 2GS	IOS 1	320*480	3.5英寸	1x
iPhone 3G	IOS 2	320*480	3.5英寸	1x
iPhone 3GS	IOS 3	320*480	3.5英寸	1x
iPhone 4	IOS 4	320*480	3.5英寸	2x
iPhone 4s	IOS 5	320*480	3.5英寸	2x
iPhone 5	IOS 6	320*568	4.0英寸	2x
iPhone 5s/c	IOS 7	320*568	4.0英寸	2x
iPhone 6	IOS 8	375*667	4.7英寸	2x
iPhone 6plus	IOS 8	414*736	5.5英寸	3x
IPod Touch				
IPod Touch1G	IOS 1	320*480	3.5英寸	1x
IPod Touch2G	IOS 2	320*480	3.5英寸	1x
IPod Touch3G	IOS 3	320*480	3.5英寸	1x
IPod Touch4G	IOS 4	320*480	3.5英寸	2x

- IPod Touch5G IOS 6 320*568 4.0英寸 2x
- **iPad**
- iPad IOS 3 1024*768 9.7英寸 1x
- iPad2 IOS 4 1024*768 9.7英寸 1x
- iPad3(New iPad)IOS 5 1024*768 9.7英寸 2x
- iPad4 IOS 6 1024*768 9.7英寸 2x
- iPad Air IOS 7 1024*768 9.7英寸 2x
- iPad Air2 IOS 8 1024*768 9.7英寸 2x
- **iPad mini**
- iPad mini IOS6 1024*768 7.9英寸 1x
- iPad mini2 IOS7 1024*768 7.9英寸 2x
- iPad mini3 IOS8 1024*768 7.9英寸 2x

八、Http协议的特点，关于Http请求GET和POST的区别，什么是Https协议

- Http超文本传输协议，是短连接，是客户端主动发送请求，服务器做出响应，服务器响应之后，连接断开。HTTP是一个属于应用层的面向对象的协议。HTTP有两类报文：请求报文和响应报文。
- HTTP请求报文：一个HTTP请求报文由请求行（request line）、请求头部（header）、空行和请求数据4个部分组成。
- HTTP响应报文：HTTP响应由三个部分组成，分别是：状态行、消息报头、响应正文。
- GET请求：参数在地址后拼接，没有请求数据，不安全（因为所有的参数都拼接在地址后面），不适合传输大量数据（长度有限制，限制在1024个字节）
- POST请求：参数在请求数据区放着，相对Get请求更安全，并且数据大小没有限制。
- 1、GET提交，请求的数据会附在URL之后（就是把数据放置在HTTP协议头＜request-line＞中），以?分割URL和传输数据，多个参数用&连接;例如：login.action?name=hyddd&password=idontknow&verify=%E4%BD%A0%E5%A5%BD。如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用BASE64加密，得出如：%E4%BD%A0%E5%A5%BD，其中％XX中的XX为该符号以16进制表示的ASCII。

- POST提交：把提交的数据放置在是HTTP包的包体＜request-body＞中。上文示例中红色字体标明的就是实际的传输数据。
- 因此，GET提交的数据会在地址栏中显示出来，而POST提交，地址栏不会改变。
- 2、传输数据的大小：首先声明,HTTP协议没有对传输的数据大小进行限制，HTTP协议规范也没有对URL长度进行限制。 而在实际开发中存在的限制主要有：GET:特定浏览器和服务器对URL长度有限制，例如IE对URL长度的限制是2083字节(2K+35)。对于其他浏览器，如Netscape、FireFox等，理论上没有长度限制，其限制取决于操作系统的支持。因此对于GET提交时，传输数据就会受到URL长度的限制。POST:由于不是通过URL传值，理论上数据不受限。但实际各个WEB服务器会规定对post提交数据大小进行限制，Apache、IIS6都有各自的配置。
- 3、安全性：POST的安全性要比GET的安全性高。注意：这里所说的安全性和上面GET提到的“安全”不是同个概念。上面“安全”的含义仅仅是不作数据修改，而这里安全的含义是真正的Security的含义，比如：通过GET提交数据，用户名和密码将明文出现在URL上，因为(1)登录页面有可能被浏览器缓存， (2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了。
- Https是：Https（Secure Hypertext Transfer Protocol）安全超文本传输协议，它是一个安全通信通道，它基于HTTP开发，用于在客户计算机和服务器之间交换信息。它使用安全套结字层（SSl）进行信息交换，简单来说它是HTTP的安全版。

九、对NSUserDefaults的理解

- NSUserDefaults：系统提供的一种存储数据的方式。主要用于保存少量数据。默认存储到Library下的Preferences文件夹。

十、类别的作用

- 1、给系统原有的类添加方法，不能扩展属性，如果类别中方法的名字跟系统的方法名一样，在调用的时候类别中的方法优先级更高。
- 2、分散类的实现:如：+ (NSIndexPath *)indexPathForRow:(NSInteger)row inSection:(NSInteger)section；原本是属于NSIndexPath的方法，但是因为这个方法经常在使用表的时候来调用、跟表的关系特别密切。所以把这个方法以类别的形式、声明在了UITableView.h中。
- 3、声明私有方法。 某一个方法只实现，不声明，相当于是一个私有方法。
- 类别中不能够声明变量。类别不可以直接添加属性。property描述写setter方法，就不会报错。

十一、多线程编程

- **1、NSThread**：当需要进行一些耗时间的操作时，会把耗时间的操作放到分线程中。线程同步：多个线程同时访问一个数据会出问题。①、NSLock。②、线程同步块@synchronized（self）{}。

- **2、NSOperationQueue 操作队列**（不需要考虑线程同步的问题）。编码的重点都放在main里面。1、NSInvocationOperation，BSBlockOperation。2、自定义Operation。
- 创建一个操作，绑定相应的方法，当把操作添加到操作队列中时，操作绑定的方法就会自动执行了。当把操作添加到操作队列中时，默认会调用main方法。
- **3、GCD（Grand Central Dispatch ）**宏大的中央调度。①、串行队列。②、并发队列。③、主线程队列。
- 同步和异步：同步和异步是相对于另外一个任务而言的，同步指的是第一个任务不执行完，不会开始第二个，异步是不管第一个有没有执行完，都会开始第二个。
- 串行和并发：串行和并发是相对于多个任务而言的，串行指的是多个任务按照一定的顺序执行，并发指的是多个任务同时进行。
- 代码是在分线程中执行；在主线程队列中刷新UI。

十二、SDWebImage原理

- 调用类别的方法：①、从内存（字典）中找图片（当这个图片在本次使用程序的过程中已经被加载过），找到直接使用。②、从沙盒中找（当这个图片在之前使用程序的过程中被加载过），找到使用，缓存到内存中。③、从网络上获取，使用，缓存到内存，缓存到沙盒。

十三、objective-C中是否有二维数组？怎么实现二维数组

- 没有。数组中嵌套数组可实现二维数组。

十四、对于单元格重用的理解

- 当屏幕上的单元格滑出屏幕时，系统会把这个单元格添加到重用队列中，等待被重用，当有新单元格从屏幕外滑入屏幕内时，从重用队列中找看有没有可以重用的单元格，如果有，就拿过来用，如果没有就创建一个来使用。

十五、对通讯的理解

- 客户端跟服务器之间的数据传递，客户端向服务器发送请求，服务器给客户端响应。

十六、区分2G3G私有API

- 首先定义一个枚举来代表不同的网络类型typedef enum {
- NETWORK_TYPE_NONE= 0，
- NETWORK_TYPE_WIFI= 1，
- NETWORK_TYPE_3G= 2，

```
    NETWORK_TYPE_2G= 3,
} NETWORK_TYPE;
```

然后通过获取手机信号栏上面的网络类型的标志。

```
+ (int)dataNetworkTypeFromStatusBar {
    UIApplication *app = [UIApplication sharedApplication];
    NSArray *subviews = [[[app valueForKey:@"statusBar"] valueForKey:@"foregroundView"] subviews];
    NSNumber *dataNetworkItemView = nil;
    for (id subview in subviews) {
        if([subview isKindOfClass:[NSClassFromString(@"UIStatusBarDataNetworkItemView") class]]) {
            dataNetworkItemView = subview;
            break;
          }
       }
int netType = NETWORK_TYPE_NONE;
NSNumber * num = [dataNetworkItemView valueForKey:@“dataNetworkType"];
if (num == nil) {
        netType = NETWORK_TYPE_NONE;
    }else{
int n = [num intValue];
if (n == 0) {
            netType = NETWORK_TYPE_NONE;
       }elseif (n == 1){
            netType = NETWORK_TYPE_2G;
       }elseif (n == 2){
            netType = NETWORK_TYPE_3G;
       }else{
```

```
            netType = NETWORK_TYPE_WIFI;
        }
    }
return netType;
}
```

十七、如何解决Cell重用问题

- UITableView通过重用单元格来达到节省内存的目的:通过为每个单元格指定一个重用标识符(reuseIdentifier)，即指定了单元格的种类，以及当单元格滚出屏幕时，允许恢复单元格以便重用。对于不同种类的单元格使用不同的ID。对于简单的表格，一个标识符就够了。
- 假如一个TableView中有10个单元格，但是屏幕上最多能显示4个，那么实际上iPhone只是为其分配了4个单元格的内存,没有分配10个，当滚动单元格时，屏幕内显示的单元格重复使用这4个内存。实际上分配的Cell个数为屏幕最大显示数，当有新的Cell进入屏幕时，会随机调用已经滚出屏幕的Cell所占的内存，这就是Cell的重用。
- 对于多变的自定义cell，这种重用机制会导致内容出错。为了解决这种出错适合的方法就是把原来的UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:defineString];修改为:UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexPath];这样就能解决掉cell重用机制导致的问题。

十八、KVC和KVO的理解

- Key Value Coding 是cocoa的一个标准组成部分,它能让我们可以通过 name(key)的方法访问property, 不必调用明确的property accuser ;
- KVC（键-值编码）是一个用于间接访问对象属性的机制(一种使用字符串而而不是访问器方法去访问一个对象实例变量的机制。)，使用该机制不需要调用 set或者get方法以及-> 来访问成员变量,它是通过 setValue:forKey: 和 valueForKey:方法 。
- KVC的机制是啥样的呢？它是以字符串的形式向对象发送消息,字符串是要关注属性的关键;
- 是否存在setter、getter方法,如果不存在,它将在内部查找名为_key 或key的实例变量,如果没有报错;
- 注意:如果是基本数据类型,则需要封装一下;
- KVC的使用环境：无论是 property 还是普通的全局属性变量，都可以用 KVC;

- KVC的优缺点：优点:①、主要的好处就是来减少代码量 ②、没有property的变量(private)也能通过KVC 来设置;
- KVC的缺点:如果key写错时,编写时不会报错,运行时 会报错;
- KVO：KVO是一个对象能够观察另外一个对象的属性的值，并且能够发现值的变化。前面两种模式更加适合一个controller与任何其他的对象进行通信，而KVO更加适合任何类型的对象侦听另外一个任意对象的改变（这里也可以是controller，但一般不是controller）。这是一个对象与另外一个对象保持同步的一种方法，即当另外一种对象的状态发生改变时，观察对象马上作出反应。它只能用来对属性作出反应，而不会用来对方法或者动作作出反应。
- KVO的优点: ①、能够提供一种简单的方法实现两个对象间的同步。例如:model和view之间同步；②、能够对非我们创建的对象，即内部对象的状态改变作出响应，而且不需要改变内部对象（SKD对象）的实 现;；③、能够提供观察的属性的最新值以及先前值；④、用key paths来观察属性，因此也可以观察嵌套对象；⑤、完成了对观察对象的抽象，因为不需要额外的代码来允许观察值能够被观察。
- KVO的缺点:①、我们观察的属性必须使用strings来定义。因此在编译器不会出现警告以及检查；②、对属性重构将导致我们的观察代码不再可用；③、复杂的“if”语句要求对象正在观察多个值。这是因为所有的观察代码通过一个方法来指向；④、当释放观察者时不需要移除观察者。
- KVO的使用：被观察者发出 addObserver:forKeyPath:options:context: 方法来添加观察者。 然后只要被观察者的keyPath值变化（注意:单纯改变其值不会调用此方法,只有通过getters和setters来改变值才会触发KVO），就会在观察者里调用方法 observeValueForKeyPath:ofObject:change:context: 因此观察者需要实现方法 observeValueForKeyPath:ofObject:change:context: 来对KVO发出的通知做出响应。这些代码都只需在观察者里进行实现，被观察者不用添加任何代码，所以谁要监听谁注册，然后对响应进行处理即可，使得观察者与被观察者完全解耦，运用很灵活很简便；但是KVO只能检测类中的属性，并且属性名都是通过NSString来查找，编译器不会帮你检错和补全，纯手敲所以比较容易出错。

十九、NSNotification、block、delegate和KVO的区别

- 代理是一种回调机制，且是一对一的关系，而通知是一对多的关系，一个中对象向所有的观察者提供变更通知（KVO是被观察者向观察者直接发送通知，这是通知和KVO的区别）
- 效率肯定是delegate比NSNotification高。
- delegate 和 block 一般是用于对象1对1 的通信交互。
- delegate: delegate需要定义协议方法，代理对象实现协议方法，并且需要建立代理关系才能实现通信。

- block: block更加简洁,不需要定义繁琐的协议方法，但是如果通信事件比较多的话，建议用delegate 。
- 通知（NSNotification）: 通知主要用于一对多情况下通信，通信对象之间不需要建立关系!
- 和delegate一样，KVO和NSNotification的作用也是类与类之 间的通信，与delegate不同的是：①、这两个都是负责发出通知，剩下的事情就不管了，所以没有返回值; ②、代理是一种回调机制，且是一对一的关系，而且通知是一对多的关系，一个中心对象向所有的观察者提供变更通知。KVO是被观察者向观察者直接发送通知，这是通知和KVO的区别。

二十、对代理的理解

- 代理又叫委托，是一种设计模式，代理是对象与对象之间的通信交互，代理解除了对象与对象之间的耦合性。代理需要制订协议方法，代理对象实现协议方法，而且必须建立代理关系才可以实现通信。代理一般是一对一的通信，一般用于视图和对象之间的交互。

二十一、TCP和UDP的区别与联系

- **1**、**TCP**为传输控制层协议。这种协议可以提供面向连接的、可靠的、点到点的通信。
- **2**、**UDP**为用户数据报协议。它可以根据提供非连接的不可靠的点到多点的通信。
- **3**、用TCP还是UDP，要看程序注重哪一个方面、可靠还是快速。

二十二、scoket连接和Http连接的区别

- Http协议是基于TCP连接的，TCP协议：对应于传输层。主要解决数据如何在网络中传输；而Http是应用层协议，主要解决如何包装数据。Socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口（API），通过Socket，我们才能使用TCP/IP协议。
- **1**、**Http连接**：Http连接就是所谓的短连接，是客户端用Http协议进行请求，客户端向服务器发送一次请求，服务器端响应后连接即会断掉，以节省资源。服务器不能主动给客户端响应（除非采用Http长连接技术）。iPhone主要使用类是NSURLConnection。
- **2**、**socket连接**：socket连接就是所谓的长连接，是客户端跟服务器端直接使用Socket“套接字”进行连接，并没有规定连接后断开，所以客户端和服务器端可以保持连接通道，双方都可以主动发送数据。一般多用于游戏开发或股票开发这种要求即时性很强并且保持发送数据量比较大的场合使用。主要用的类是CFSocketRef。

二十三、TCP连接的三次握手

- **1、第一次握手**：客户端发送syn包（syn=j）到服务器，并进入SYN_SEND状态，等待服务器确认；
- **2、第二次握手**：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN+RECV状态；
- **3、第三次握手**：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK（ack=k+1），此时发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

二十四、XML数据的解析方式，各有什么不同，JSON解析有哪些框架

- 1、XML数据解析有两种解析方式：DOM解析与SAX解析。
- 2、DOM解析必须先完成DOM树的构造，在处理规模较大的XML文档时就很耗内存，占用资源较多
- 3、与DOM不同的是，它是用事件驱动模型，解析XML文档时每遇到一个开始或者结束标签、或者属性、或者一条指令时，程序就产生一个事件来进行相应的处理，因此，SAX相对于DOM来说更适合操作大的XML文档。
- 4、JSON解析：性能比较好的主要是第三方的JSONKIT和 ios自带的JSON解析类，自带的JSON解析性能是最高的，但是只能是IOS5之后可以使用。

二十五、XML数据解析的理解

- 解析 XML 通常有两种方式，DOM 和 SAX：DOM解析XML时，读入整个XML文档并构建一个驻留内存的树结构（节点 树），通过遍历树结构可以检索任意XML节点，读取它的属性和值。而且通常情况下，可以借助XPath，直接查询XML节点。SAX解析XML，是基于事件通知的模式，一边读取XML文档一边处理，不必等整个文档加载完之后才采取操作，当在读取解析过程中遇到需要处理的对象，会发出通知对其进行处理。

二十六、对Socket的通信方式的理解

- Socket的通信是通过TCP/IP协议，实现客户端与服务器端之间的通信方式。客户端通过三次握手与服务器建立可靠的连接，然后进行数据传输。
- Socket中的默认连接超时时间是30秒，默认大小是8k（可以理解为一个数据包的大小）。

二十七、如何进行真机调试

- **1**、首先需要用“钥匙串”创建一个钥匙（key）。

- 2、将钥匙串上传到官网，获取IOS Development 证书。
- 3、创建app ID 也就我们应用程序中的Boundle ID。
- 4、添加Device ID 也就是UDID。
- 5、通过勾选前面所创建的证书： app ID 、 Device ID，
- 6、生成mobileprovision文件。
- 7、先决条件 ——》 你需要一个账号 99$

二十八、APP发布的上架流程

- 1、通过https：//itueconnect.apple.com添加应用信息。
- 2、下载安装发布证书。
- 3、选择发布证书，使用Archive编译发布包。用Xcode将代码（发布包）上传到服务器。
- 4、等待审核通过。

二十九、如何生成IPA

- 菜单栏 Product —> Archive

三十、什么是SVN？为什么要用SVN？及SVN、Git协作开发，怎么防止代码文件冲突

- 简单说 **SVN** = 版本控制 + 备份服务器 。也就是说可以把SVN当成您的备份服务器，更好的是，它可以帮助您记住每次上传到这个服务器的档案内容。并且自动的赋予每次的变更一个版本。
- **为什么要用SVN**：①、备份工作档案的重要性。②、版本控管的重要性。③、伙伴间的数据同步的重要性。④、如果没有一个好的办法，备份不同版本是很耗费硬盘空间的。
- SVN有很棒的版本控管机制，所有上传的版本都会帮您记录下来，也有版本分支及合并等好用的功能。SVN可以让不同的开发者存取同样的档案，并且利用SVN Server作为档案同步的机制。就是说，您有档案更新时，无需将档案寄给您的开发成员。只需要告诉他新的版本已经在SVN Server上面，请他自己去SVN Server上面就可以取得最新版本。而且SVN Server也可以做到当您上传新版本后，自动发信给相关的成员。SVN的存放档案方式是采用差异备份的方式。也就是说，他只会备份有不同的地方。所以很省硬盘空间。此外，他也可以这对所谓的非文字文件进行差异备份。
- **防止冲突：**

- 1、防止代码冲突：不要多人同时修改同一个文件。例如：A、B都修改同一个文件，先让A修改，然后提交到服务器，然后B更新下来，在进行修改。
- 2、服务器上的项目文件xcodeproj，仅让一个人管理提交，其他人只更新。防止此文件产生冲突。

三十一、如何进行网络推送

- 一种是Apple自己提供的通知服务（APNS服务器）、一种是用第三方推送机制。
- 首先应用发送通知；系统弹出提示框询问用户是否允许，当用户允许后向苹果服务器（APNS）请求deviceToken，并由苹果服务器发送给自己的应用；自己的应用将DeviceToken发送给自己的服务器；自己服务器想要发送网络推送时将deviceToken以及想要推送的信息发送给苹果服务器，苹果服务器将信发送给应用。
- 推送信息内容 —> 总容量不超过256个字节。
- iOS SDK本身提供的APNS服务器推送，它可以直接推送给目标用户并根据您的方式弹出提示。优点：不论应用是否开启，都会发送到手机端。缺点：消息推送的机制是苹果服务端控制，个别时候可能会有延迟，因为苹果服务器也会有队列来处理所有的消息请求。
- 第三方推送机制，普遍使用Socket机制来实现的，它几乎可以达到即时的发送到目标用户手机端，适用于即时通讯类应用。优点：它几乎是实时的，主要取决于它心跳包的节奏。缺点：因为iOS系统的限制，应用不能长时间的后台的运行，所以在应用关闭的情况下这种推送机制不可用。

三十二、网络七层协议

- 应用层
- 1.主要功能：用户接口、应用程序
- 2.application典型设备：网关
- 3.典型协议、标准和应用：TELNET, FTP, HTTP
- 我们做应用层，比如我们做软件，一个视频播放器，这个就是指一个应用层。
- 表示层
- 1.主要功能：数据的表示、压缩和加密 presentation
- 2.典型设备：网关
- 3.典型协议、标准和应用：ASCLL、PICT、TIFF、JPEG、 MIDI、MPEG
- 表示层相当于一个东西怎么表示，表示的一些协议，像图片：JEPG、声音：MIDI、视频：MPEG

- 表示层就是定义这个层的协议的。比如：某个人说说自己做表示层，可能这个人就是在做MPEG4.
- 会话层
- 1.主要功能：会话的建立和结束 session
- 2.典型设备：网关
- 3.典型协议、标准和应用：RPC、SQL、NFS 、X WINDOWS、ASP
- 传输层
- 1.主要功能：端到端控制 transport
- 2.典型设备：网关
- 3.典型协议、标准和应用：TCP、UDP、SPX
- 网络层
- 1.主要功能：路由，寻址 network
- 2.典型设备：路由器
- 3.典型协议、标准和应用：IP、IPX、APPLETALK、ICMP
- 数据链路层
- 1.主要功能：保证误差错的数据链路 data link
- 2.典型设备：交换机、网桥、网卡
- 3.典型协议、标准和应用：802.2、802.3ATM、HDLC、FRAME RELAY
- 物理层
- 1.主要功能：传输比特流 physical
- 2.典型设备：集线器、中继器
- 3.典型协议、标准和应用：V.35、EIA/TIA-232

三十三、LayoutSubViews在什么时候会被调用?

- 当View本身的frame改变时，会调用的这个方法。

三十四、Core Data的6成员对象

- 1、**NSManagedObject**

- 被管理的数据记录 Managed Object Model 是描述应用程序的数据模型,这个模型包含实体（Entity），特性（Property），读取请求（Fetch Request）等。
- **2、NSManagedObjectContext**
- 管理对象上下文，持久性存储模型对象，参与对数据对象进行各种操作的全过程，并监测数据对象的变化，以提供对 undo/redo 的支持及更新绑定到数据的 UI。
- **3、NSPersistentStoreCoordinator**
- 连接数据库的 Persistent Store Coordinator 相当于数据文件管理器，处理底层的对数据文件的读取与写入。一般我们无需与它打交道。
- **4、NSManagedObjectModel**
- 被管理的数据模型，数据结构。
- **5、NSFetchRequest**
- 数据请求
- **6、NSEntityDescription**
- 表格实体结构
- 此外还需要知道.xcdatamodel文件编译后为.momd或者.mom文件

三十五、为何要使用CoreData？有哪些功能？

- **为何要使用CoreData：**
- 使用Core Data有很多原因，其中最简单的一条就是：它能让你为Model层写的代码的行数减少为原来的50%到70%。 这归功于之前提到的Core Data的特性。更妙的是，对于上述特性你也既不用去测试，也不用花功夫去优化。
- Core Data拥有成熟的代码，这些代码通过单元测试来保证品质。应用Core Data的程序每天被世界上几百万用户使用。通过了几个版本的发布，已经被高度优化。 它能利用Model层的信息和运行时的特性，而不通过程序层的代码实现。 除了提供强大的安全支持和错误处理外，它还提供了最优的内存扩展性，可实现有竞争力的解决方案。不使用Core Data的话，你需要花很长时间来起草自己的方案，解决各种问题，这样做效率不高。
- 除了Core Data本身的优点之外，使用它还有其他的好处： 它很容易和Mac OS X系统的Tool chain集成；利用Model设计工具可以按图形化方式轻松创建数据库的结构；你可以用Instruments的相关模板来测试Core Data的效率并debug。 在Mac OS X的桌面程序中，Core Data还和Interface Builder集成（打开Inspector可以看到有binding的选项，这个东东iPhone上木有。。。），按照model来创建UI变的更简单了。 这些功能能更进一步的帮助你缩短设计、开发、测试程序的周期。

- **CoreData功能初窥：**

- 1) 对于key-value coding 和key-value observing完整且自动化的支持除了为属性整合KVC和KVO的访问方法外，Core Data还整合了适当的集合访问方法来处理多值关系。

- 2) 自动验证属性(property)值
  Core Data中的managed object扩展了标准的KVC验证方法，以保证单个的数值在可接受的范围之内，从而使组合的值有意义。（需校准翻译）

- 3) 支持跟踪修改和撤销操作
  对于撤销和重做的功能，除过用基本的文本编辑外，Core Data还提供内置的管理方式。

- 4) 关系的维护
  Core Data管理数据的变化传播，包括维护对象间关系的一致性。

- 5) 在内存中和界面上分组、过滤、组织数据

- 6) 自动支持对象存储在外部数据仓库的功能

- 7) 创建复杂请求
  你不需要动手去写复杂的SQL语句，就可以创建复杂的数据请求。方法是在“获取请求”(fetch request)中关联NSPredicate（又看到这个东东了，之前用它做过正则）。NSPrdicate支持基本的功能、相关子查询和其他高级的 SQL特性。它还支持正确的Unicode编码（不太懂，请高人指点）， 区域感知查询（据说就是根据区域、语言设置调整查询的行为）、排序和正则表达式。

- 8) 延迟操作（原文为Futures(faulting）直译为期货，这里个人感觉就是延迟操作的形象说法。请高人指教)。
  Core Data 使用延迟加载(lazy loading)的方式减少内存负载。 它还支持部分实体化延迟加载，和“写时拷贝”的数据共享机制。（写时拷贝，说的是在复制对象的时候，实际上不生成新的空间，而是让对象共享一块存储区域，在其内容发生改变的时候再分配）。

- 9) 合并的策略
  Core Data 内置了版本跟踪和乐观锁定(optimistic locking)来支持多用户写入冲突的解决。
  注：乐观锁，假定数据一般不出现冲突，所以在数据提交更新的时候，才对数据的冲突进行检测，如果冲突了，就返回冲突信息。

- 10) 数据迁移
  就开发工作和运行时资源来说，处理数据库架构的改变总是很复杂。Core Data的schema migration工具可以简化应对数据库结构变化的任务， 而且在某些情况下，允许你执行高效率的数据库原地迁移工作。

- 11) 可选择针对程序Controller层的集成，来支持UI的显示同步Core Data在iPhone OS之上 提供NSFetchedResultsController对象来做相关工作，在Mac OS X上，我们用Cocoa提供的绑定(Binding)机制来完成。

三十六、深拷贝和浅拷贝

- 使用自定义对象copy需要实现NSCopying（NSMutableCopying）。具体深浅拷贝取决与协议的实现。
- 容器类，深拷贝拷贝容易类，不拷贝容器内部对象。想要实现，用对象序列化or For循环。

三十七、ASIHttpRequest 、AFNetWorking之间的区别

- ASIHttpRequest功能很强大，主要是在MRC下实现的，是对系统CFNetwork API进行了封装，支持HTTP协议的CFHTTP。配置比较复杂，并且ASI框架默认不会帮你监听网络改变，如果需要让ASI帮你监听网络状态改变，需要手动开始这个功能。
- AFNetworking构建于NSURLConnection, NSOperation, 以及其他熟悉的Foundation技术之上。拥有良好的架构，丰富的API，以及模块化构建方式，所以使用起来非常轻松。它基于NSOperation封装的，AFURLConnectionOperation的子类。
- 更新状态：ASIHttp 2012年10月份，已经停止更新；AFNetworking 持续更新中，目前已更新至2.3.1版。
- 介绍：ASI直接操作对象ASIHttpRequest，是一个实现了NSCoping协议的NSOperation子类；AFNetworking直接操作对象的AFHttpClient，是一个实现了NSCoding和NSCopying协议的NSObject子类。
- 同步请求：ASI是直接通过调用一个startSynchronous方法；AFNetworking默认没有封装同步请求，如果开发者需要使用同步请求，则需要重写getPath：paramenters：success：failures方法，对AFHttpRequestOperation进行同步处理。
- 性能对比：AFNetworking请求优于ASIHttpRequest。

三十八、什么是单例模式？什么时候使用单例模式？

- 单例模式是iOS中常用的一种设计模式。单例模式是一个类在系统中只有一个实例对象。通过全局的一个入口点对这个实例对象进行访问。
- 在程序中，单例模式经常用于只希望一个类中有一个实例，而不运行一个类还有两个以上的实例。当然，在iOS SDK中，根据特定的需求，有些类不仅提供了单例访问的接口，还为开发者提供了实例化一个新的对象接口，例如，NSFileManager可以通过defaultManager方法返回相同的一个NSFileManager对象。如果需要新的一个NSFileManager实例对象，可以通过init方法。
- iOS中单例模式的实现方式一般分为两种：Non-ARC(非ARC)和ARC+GCD。

三十九、对沙盒的理解

- **1、 每个iOS应用都被限制在“沙盒”中，“沙盒”相当于一个加了仅主人可见权限的文件夹，就是应用程序在安装过程中，系统为每个单独的应用程序生成它的主目录和一些关键的子目录—> 文件夹，苹果对沙盒有以下几条限制。**
- （1）、应用程序可以在自己的沙盒里运作，但是不能访问任何其他应用程序的沙盒。
- （2）、应用程序间不能共享数据，沙盒里的文件不能被复制到其他应用程序文件夹中，也不能把其他应用程序文件夹中的文件复制到沙盒里。
- （3）、苹果禁止任何读、写沙盒以外的文件，禁止应用程序将内容写到沙盒以外的文件夹中。
- （4）、沙盒根目录里有三个文件夹：
- Documents，一般应该把应用程序的数据文件存到这个文件夹里，用于存储用户数据或其他应该定期备份的信息。
- Library，下有两个文件夹，Caches存储应用程序再次启动所需的信息，Preferences包含应用程序偏好设置文件，不过不要在这里修改偏好设置。
- temp，存放临时文件，即应用程序再次启动不需要的文件。
- **2、获取沙盒路径**
- （1）、获取沙盒根目录的方法，有以下几种：用NSHomeDirectory获取。
- （2）、获取Document路径 NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,NSUserDomainMask,YES)。

四十、自动释放池是什么？如何工作？

- 自动释放池是NSAutorelease类的一个实例，当向一个对象发送autorelease消息，该对象会自动入池，将会向池中所有对象发送一条release消息，释放对象。

四十一、doGet和doPost的区别

- 在Http通讯协议中默认就是doGet，doGet是在请求地址后加上了要提交的信息，以问号标记，问号之后追加参数名=参数值，各参数之间用&隔开。doGet适合数据量比较小，格式简单的，不私密的数据。doPost适合于比较私密的数据比如用户名密码，可以提交二进制数据，或大量数据。

四十二、对瀑布流的理解

- 首先图片的宽度都是一样的，第一步、将图片等比例的压缩，让图片不变形。第二步、计算图片最低应该摆放的位置，哪一列低就放在哪。第三步、进行最优的排列，在

scrollView的基础上添加两个tableView，然后将之前所计算的scrollView的高度通过tableView展示出来。

- 问题:
- 1、为什么服务器请求的时候将图片的宽高发送给你?
- 因为我们下载好之前就要对图片进行布局。
- 2、你如何使用两个TableView产生联动?
- 我将两个tableView的滚动事件禁止掉了，当最外层scrollView滚动的时候，我将两个TableView跟着滚动，并且更改contentOffset（偏移量），这样产生效果滚动两个tableView。

四十三、OC的消息机制

- 在Objective-C中，message与方法的真正实现是在执行阶段绑定的，而非编译阶段。编译器会将消息发送转换成对objc_msgSend方法的调用。objc_msgSend方法含两个必要参数：receiver、方法名（即：selector）。如：[receiver message]; 将被转换为：objc_msgSend(receiver, selector);objc_msgSend方法也能hold住message的参数，如：objc_msgSend(receiver, selector, arg1, arg2, ...)；objc_msgSend方法会做按照顺序进行以下操作，以完成动态绑定：查找selector所指代的程序（方法的真正实现）。因为不同类对同一方法有不同的实现，所以对方法的真正实现的查找依赖于receiver的类调用该实现，并将一系列参数传递过去将该实现的返回值作为自己的返回值，返回之消息传递的关键是，编译器构建每个类和对象时所采用的数据结构。每个类都包含以下两个必要元素：一个指向父类的指针一个调度表（dispatch table）。该调度表将类的selector与方法的实际内存地址关联起来。每个对象都有一个指向所属类的指针isa。通过该指针，对象可以找到它所属的类，也就找到了其全部父类。

四十四、OC的Run loop

- Run loop是线程相关的基础框架的一部分。一个run loop就是一个事件处理的循环，用来不停的调度工作以及处理输入事件。使用run loop的目的是让你线程在有工作的时候忙于工作，而没工作的时候处于休眠状态。

四十五、如何将产品进行多语言发布，做国际化开发

- 1、新建String File文件，命名为Localizable.strings，往里面添加你想要的语言支持。
- 2、在不同语言的Localizable.strings文件中添加对应的文本。
- 3、XIB文件国际化。在需要国际化的XIB文件上get Info添加多语言版本，修改各语言版本相对应的界面文字及图片。
- 4、程序名称国际化。新建一个strings文件，然后国际化它，get Info...。

四十五、核心动画CALayer

- CALayer和UIView的区别，以及创建核心动画的几个子类,以及它们之间的关系?
- CALayer 是整个图层类的基础，它是所有核心动画图层类的父类。
- 和视图类(NSView 或 UIView)一样,CALayer 有自己的父图层类,同时也拥有自己子图层类的集合，它们构成了一个图层树的层次结构。

四十六、手势

- CA在iOS开发中处理触摸屏幕的操作，在3.2之前都是 由UIResponder而来的的如下四种方法。
- touchesBegan,touchesCancelled, touchesEnded,touchesMove。
- 但是这种方式甄别不同的手势操作很麻烦,需要自己计算做不同的手势分辨。iOS3.2中苹果给出了一个简便的方式,就是使用 UIGestureRecognizer。
- UIGestureRecognizer基类是一个抽象类,我们主要使用它 的子类。
  UITapGestureRecognizer(点击手势)。 UIPinchGestureRecognizer(捏合手势)。
  UIRotationGestureRecognizer(旋转手势)。 UISwipeGestureRecognizer(轻扫手势)。
  UIPanGestureRecognizer(滑动手势)。 UILongPressGestureRecognizer(长按手势)。
- 手势的使用: 创建手势,设置回调事件,添加到view 上，响应事件。