从零开始学iOS7开发系列3-我的地盘我做主-Cha7

原文及示例代码来自raywenderlich store中的iOS Apprentice 系列3教程，经过翻译和改编。

期待已久的炉石传说终于在今天-2014年1月24日宣布国服公测了，当然，个人更期待的是iPad和iPhone版本的公测。毕竟这种休闲卡牌游戏还是拿在手上随地随地玩才过瘾啊。不过考虑到炉石传说是用Unity3D开发的，这种跨平台移植的难度很小，主要还是在美术素材和一些交互细节上做一些调整。
说到炉石传说我就想到国内的那个山寨版XX传说，不得不佩服国人在山寨和模仿上已经到了炉火纯青的地步，从UI，图标到数值设定全面copy paste。唯一不同的是技术上采用cocos2d-x引擎开发，开发团队号称2个人，20天，且用2D引擎实现了3D效果，颇引以为豪。
暴雪昨天已经正式起诉了这个游戏土豪，不过估计对他们也没太大伤害。

为什么天朝盛产熟练的代码工和码农、码奴，为什么大多数的天朝技术人员对开源项目和与人分享兴趣不大，为什么天朝程序猿没有开发出流行的编程语言，以及有世界影响力的引擎、框架、工具？？？

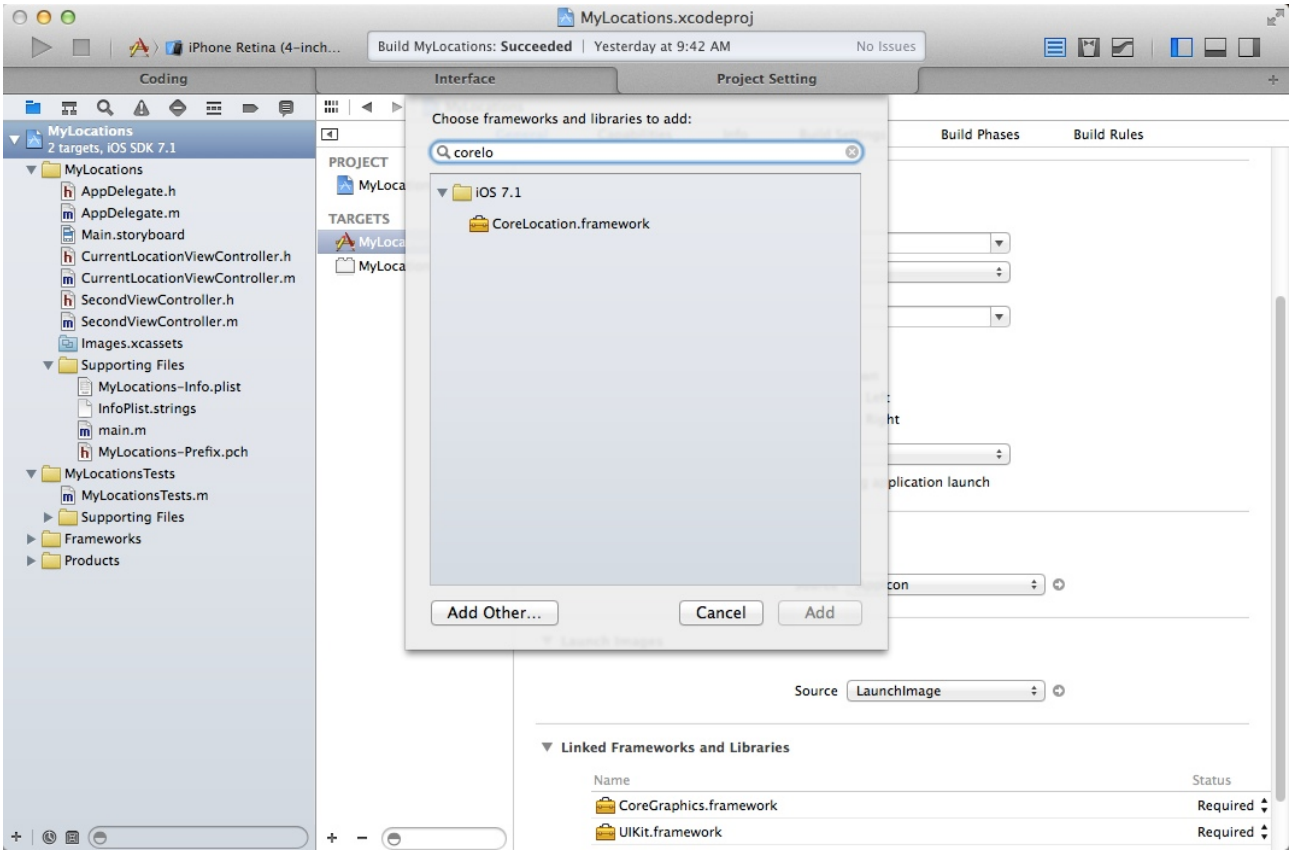
我的个人功力还远远不够，所以这里也给不出答案，大家自己思考吧。

言归正传，这里终于要真正接触Core Location了。

目前的iOS设备基本上都可以通过某种方式对用户定位，比如通过GPS卫星，通过Wi-Fi或者蜂窝网络的三角测量法。关于这些定位技术，限于篇幅，这里不可能给出详细的解释。不过兴趣是最好的老师，谁说程序猿就不能懂点GPS的知识？我们完全可以在维基百科，谷哥和度娘上找到所需要的答案。需要让我教你怎么搜索吗？不好意思，还是你自己来吧~

在iOS开发中，有一个强大的框架可以帮你获取用户的地理位置，这就是传说级卡牌，哦不，传说中的强力卡牌-Core Location。我会告诉你像Core Location这样的紫色卡牌在iOS中有哪些吗？别着急，慢慢看吧。

我们可以在iOS应用中通过Core Location来获取用户的当前经度和纬度信息，对于带compass（罗盘）的设备，还可以提供heading（朝向）信息，不过在这篇教程里暂时用不到。如果你对风水堪舆之类的感兴趣，不妨在学完本系列教程后尝试着做个风水类的应用，貌似此类应用和彩票、星座、相术类应用一样颇受人青睐。我还有个更NB的idea，如果你在给房地产商土豪们开发的应用中嵌入



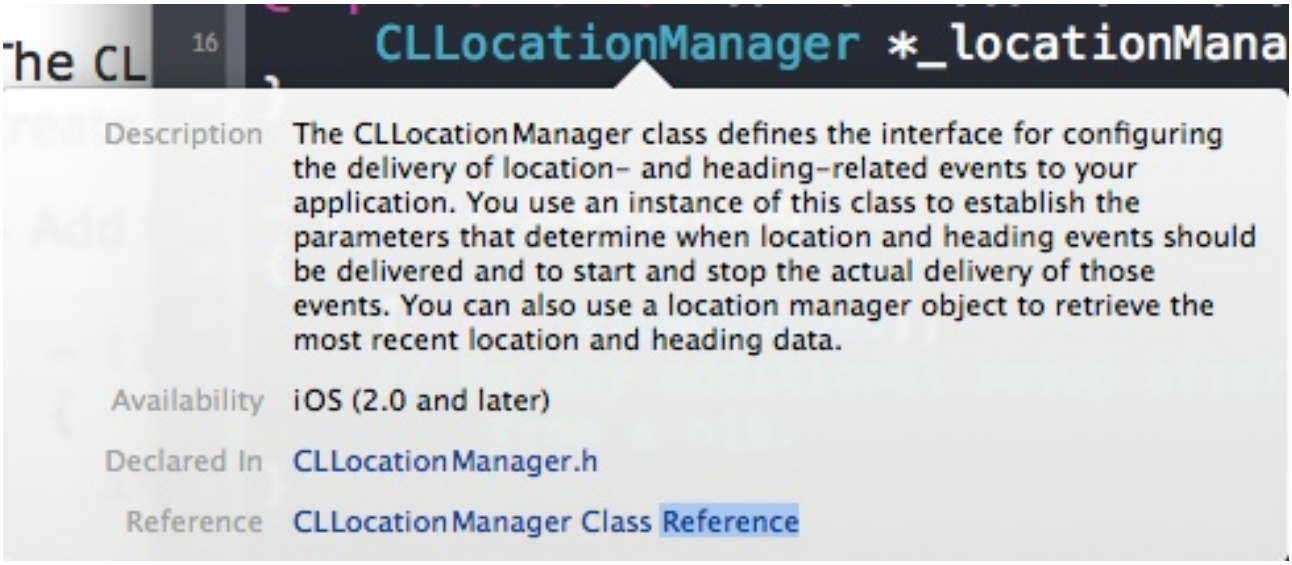
风水功能，显然就会立马升级为大师了。如果你靠这个成了土豪，苟富贵勿相忘，起码像那个发了千万的地产公司那样，给我包个红包，送个iphone6,MBP,mac pro的苹果全系列产品吧。
更NB的是，Core Location还可以在你移动身体的同时持续更新你的位置。千万不要把这个神技告诉你老爹老妈或者GF,BF，不然他们肯定会自学成才，搞一个定制版的追踪应用来监视你了。

虽然说使用Core Location来获取用户的地理位置超级简单，不过现实中总是有无数的坑等着你去踩，然后我在一边乐见所成。不过Don't panic。现在让我们先通过最简单的最有效的方式在项目中添加Core Location框架，然后使用它来获取你的当前坐标。

打开Xcode，切换到项目设置界面的General选项卡，然后滚到（不是你，是鼠标）Linked Frameworks and Libraries这部分。点击+加号按钮，然后选择CoreLocation.framework,然后点击Add。

然后切换到CurrentLocationViewController.h，更改其中的代码如下：

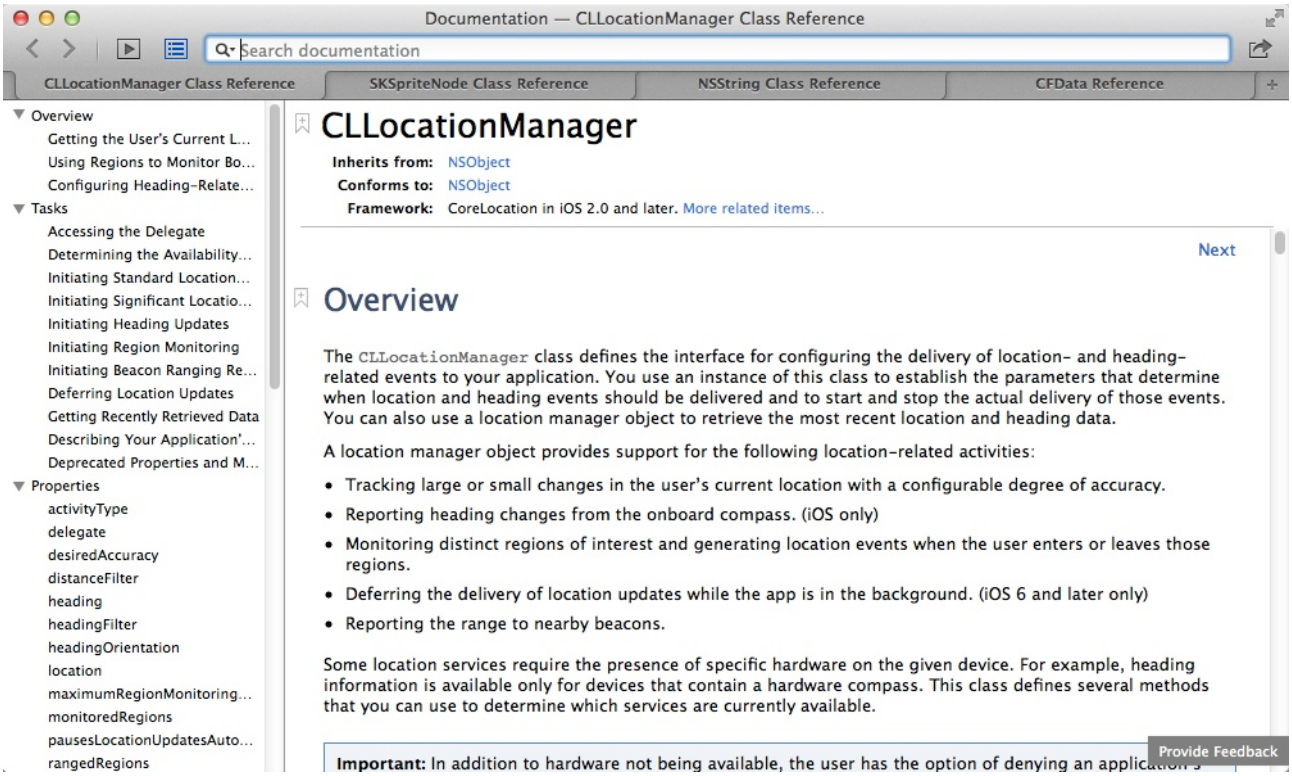
```
#import <CoreLocation/CoreLocation.h>
```



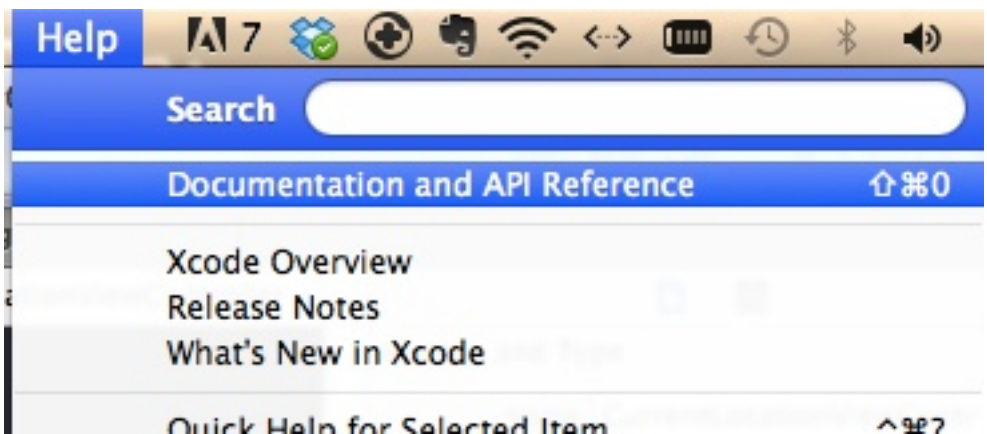
@interface CurrentLocationViewController : UIViewController<CLLocationManagerDelegate>

注意上面黄色高亮的部分才是有改动的部分，其它的代码保持不变。

然后切换到CurrentLocationViewController.m，在@implementation部分添加一个实例变量声明：



@implementation CurrentLocationViewController{



```
    CLLocationManager *_locationManager;
}
```

CLLocationManager是一个对象，它可以向我们提供GPS坐标的信息。如果我们想了解更多关于这个对象的信息该怎么办?
还记得吗？只需要按住option键，然后用鼠标点击CLLocationManager，就可以弹出下面的这个小提示：

如果你e文好，很容易就看懂这个对象是干嘛的。如果e文不好？哼哼，你不知道现在的主流编程语言、框架、引擎、工具都是老外发明的吗？先把e文学好吧。神马？高考不考e文了，喜大普奔~
不过相信我，很快你就会内牛满面的。

这里只是个简单的描述，如果想了解更加详细的内容，可以点击Reference部分的CLLocationManager Class Reference，就会自动打开Xcode里面附带的帮助文档。

还有一种方式打开帮助文档，在Xcode的主菜单中点击Help ,然后点击Documentation and API Reference就好了。
这里多废话几句，对于初学者，遇到别人的源代码看不懂的时候该咋办?
首先当然是看对方有没有提供教程了，如果不是教程是实际项目，那么首先是找对方要项目开发文档和程序说明文档，神马？是个二货公司竟然连开发文档和程序说明文档都不给？那么只好先看看项目结构和里面的注释了。神马？极品前任竟然连注释都懒得写？那我们只好根据方法和变量名称来猜了。神马？极品前任竟然用a,b,c,d,e这样的方法和变量名称？那我们就只好硬着头皮一行行的看了。
如果看到不懂的系统对象该怎么办？这个简单啊，刚才告诉你了的，option点击，或者直接在官方帮助文档里面搜索就行了。

既然定义了CLLocationManager这个对象，那么在哪里初始化它呢？既然我们一开始就打算用它，看来可以考虑放在视图控制器的initWithCoder:这个初始化方法里面。

在CurrentLocationViewController.m中添加initWithCoder:方法的实现代码如下：

```
-(id)initWithCoder:(NSCoder *)aDecoder{

    if((self = [super initWithCoder:aDecoder])){
        _locationManager = [[CLLocationManager alloc]init];
    }
    return self;
}
```

当然，仅仅创建这个新的CLLocationManager对象并不会直接就提供GPS坐标信息。为了开始接收坐标信息，我们需要首先调用startUpdatingLocation方法。

除非是在使用turn-by-turn导航的情况下，我们才需要让应用提供连续的GPS定位信息。连续导航定位很耗电，不信你可以在iPhone上开下自己的高德或者地图导航走两步，很快电池就耗得一干二净了。对我们这款应用来说，我们只需要在进行位置修正的时候开启这个location manager，然后就把它关掉。

这句话写下来和说出来都很简单，不过用代码实现还不是那么简单。不过别害怕，现在最重要的是实现第一个里程碑，就是用Core Location来接受信息。

更改CurrentLocationViewController.m中的getLocation:方法：

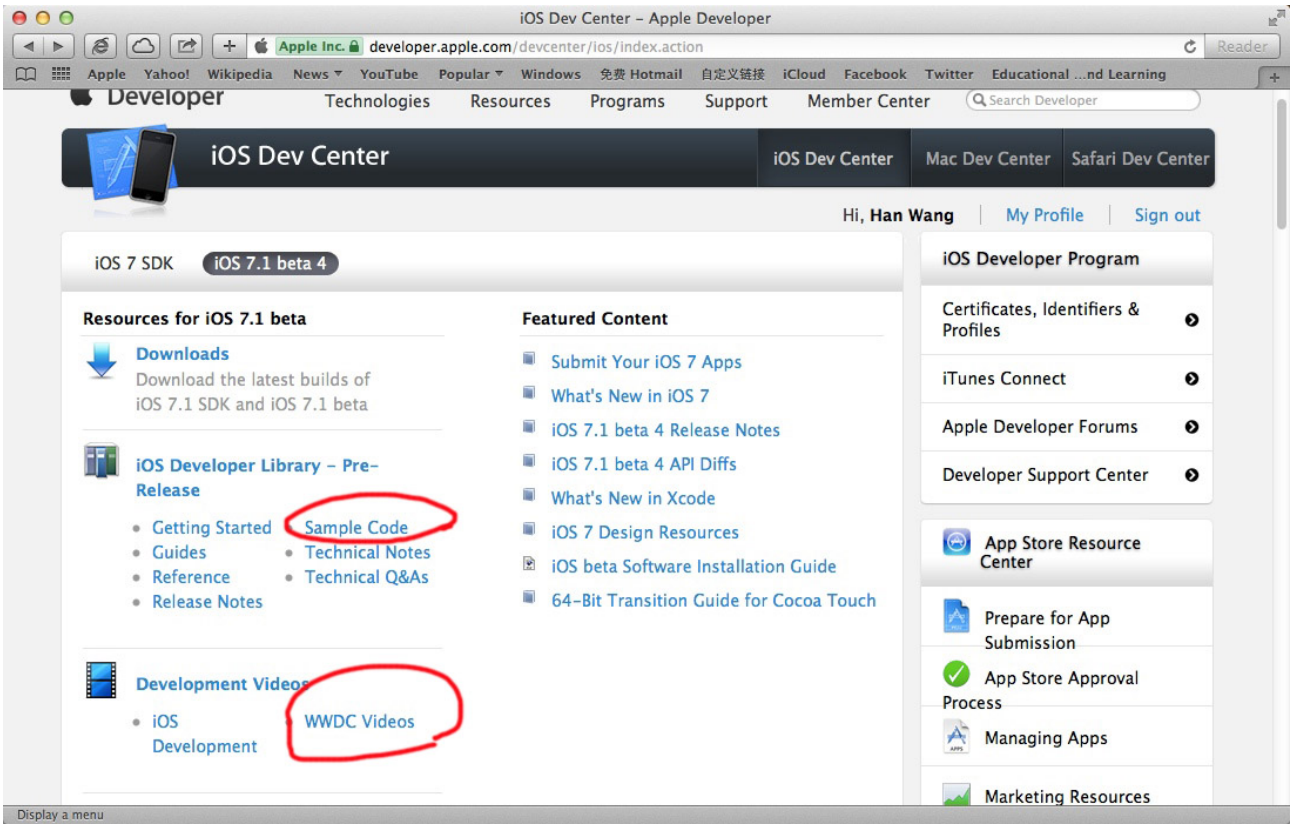
```
-(IBAction)getLocation:(id)sender{


    _locationManager.delegate =self;
    _locationManager.desiredAccuracy = kCLLocationAccuracyNearestTenMeters;
    [_locationManager startUpdatingLocation];
}
```

这个动作方法是和Get My Location按钮关联在一起的。使用该方法可以让location manager知道当前的视图控制器是它的代理对象，而且我们需要设置的坐标经度是10米。接着我们开启了location manager，从此刻开始就会把位置信息的更新发送给代理对象（这里是视图控制器）。

既然是代理对象，那么显然需要实现某些协议。

在CurrentLocationViewController.m中添加以下方法：

```
#pragma mark -CLLocationManagerDelegate
```

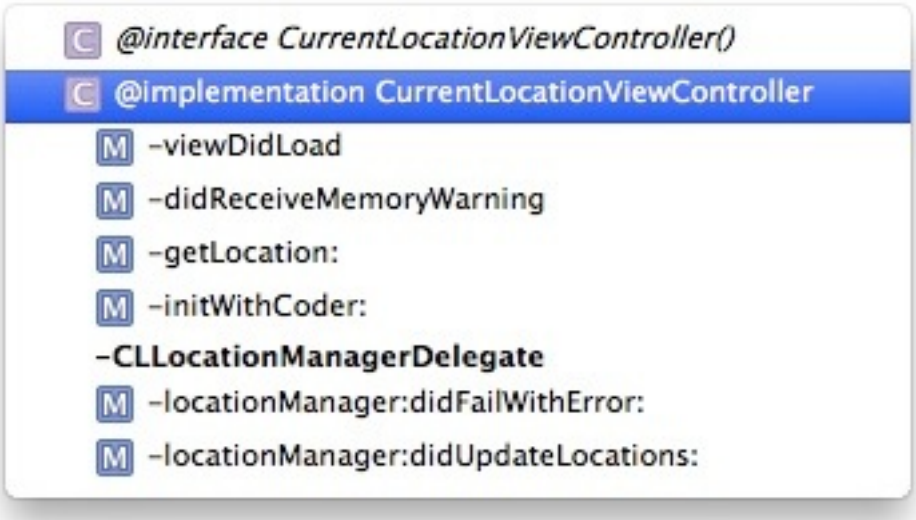



```
-(void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *)error{

    NSLog(@"定位失败：%@",error);
}

-(void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations{

    CLLocation *newLocation = [locations lastObject];
    NSLog(@"已更新坐标，当前位置：%@",newLocation);
```

}

这里首先要思考一个问题。你怎么知道CLLocationManagerDelegate这个协议有哪些方法？这些方法各自有神马作用？什么时候该用哪个方法？

装B的程序猿会告诉你，这你都不懂还搞个P的iOS开发啊？这简直就是生而知之的事情啊，就和太阳东升西落一样自然啊。对这种程序猿，你要回敬一句，我去年买了个表。
没有谁生而知之，菜鸟的菜主要是经验少，并不代表没有潜力。刚才说过了，如果碰到一个没见过的东西，首先可以在官方自带的文档里面是搜索。如果实在不懂，可以在stackoverflow里面问。一回生二回熟，可能刚开始你不知道用什么，怎么用，但看的多了，接触的多了，做的项目多了，自然就会知道该怎么用了。
所以学习编程一个很重要的内容就是要掌握自学的方法，和快速学习新知识的技巧。
平生最敬畏两种人，一种是初出茅庐但极有潜力的所谓“菜鸟”，一种是项目经验丰富阅码无数但仍然云淡风轻与时俱进的真正高手。平生最瞧不起的就是刚刚入门离高手还有十万八千米就在那里对

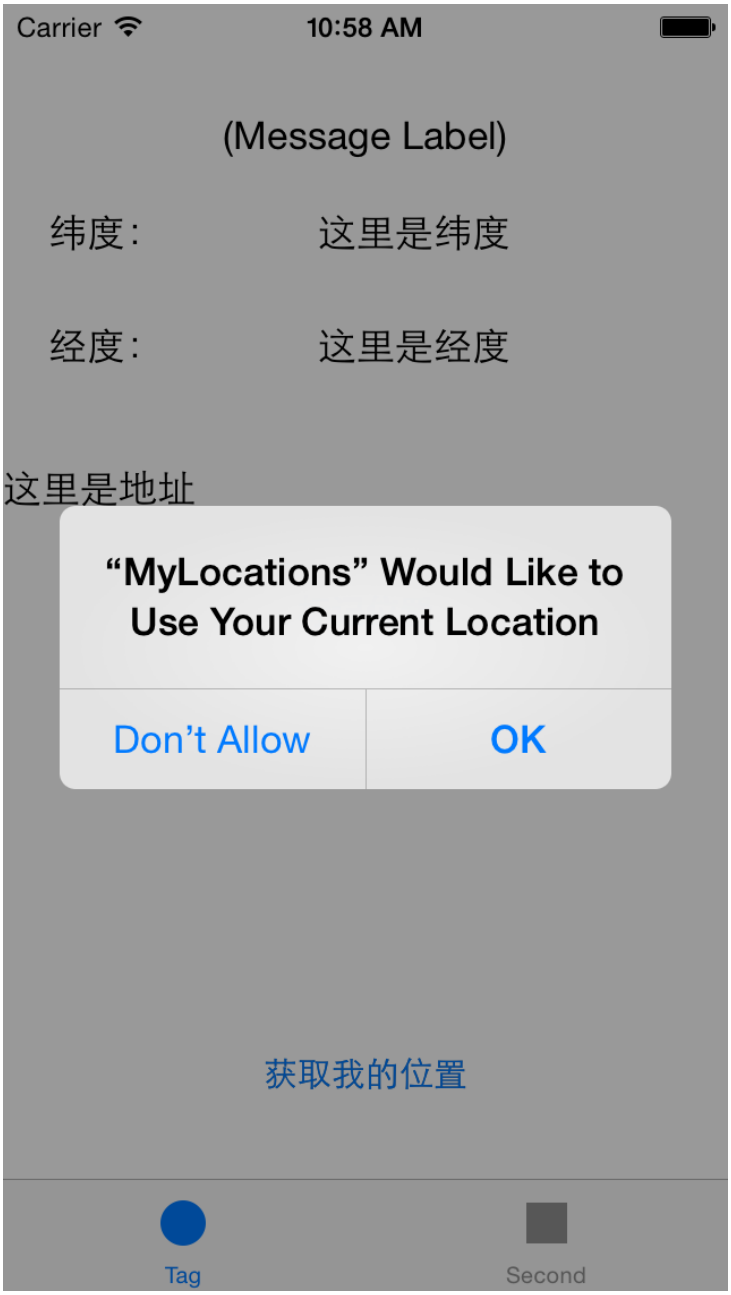


新人各种炫耀BS的SB和2B。

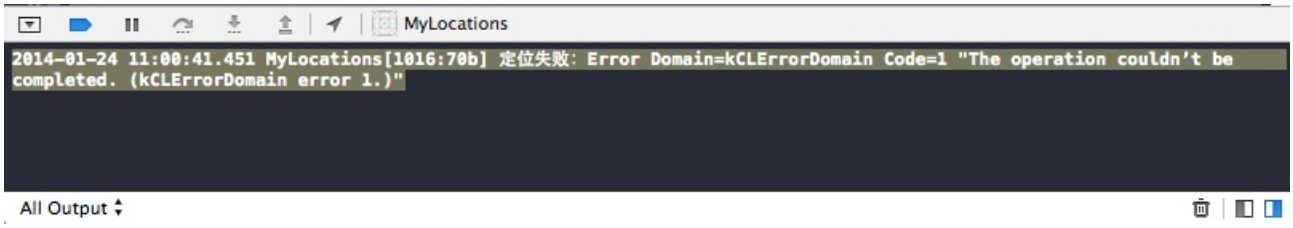
对于iOS开发，要善用苹果的帮助文档和官方的示例代码，还有WWDC 视频也非常值得看。
帮助文档大家都知道在哪儿了，那么官方示例代码和WWDC视频在哪儿有？
很简单，通过developer.apple.com进入开发者网站，然后进入iOS Dev Center,通过红色圈起来的链接就可以找到需要的资源了。

别忘了右侧还有Apple Developer Forums也值得去看看。个人经验，几乎95%的问题可以通过官方开发论坛和stackoverflow得到解决。作为补充可以去quora和google 的discussion groups。
为什么要废这些话？直接告诉我问题怎么解决不就好了吗？
还是那句话，我们这个系列教程的主要目的是培养程序猿的思维方式，授人以鱼不如授人以渔，所以宁可多废话几句。如果你希望天朝那种直接给结论的教学方式，对不起这里概不奉送！
而且对于此类解决问题的思路，我们会重复说明，不厌其烦。所以，非喜勿入了~

通过苹果的帮助文档，我们可以看到，刚才的协议方法didFailWithError:作用是通知代理对象location manager无法获取坐标信息，而错误原因就在附带的error参数中。



还要注意到这个方法是optional的，也就是可选非强制实现的，不过苹果官方也说了，这个方法是recommended的。世上没有完美的事情，因为硬件设备的种种原因或许无法获得地理位置信息，这是极有可能的。在iOS中没有java的那种try catch机制，所以类似这样的协议方法还是非常有用的。

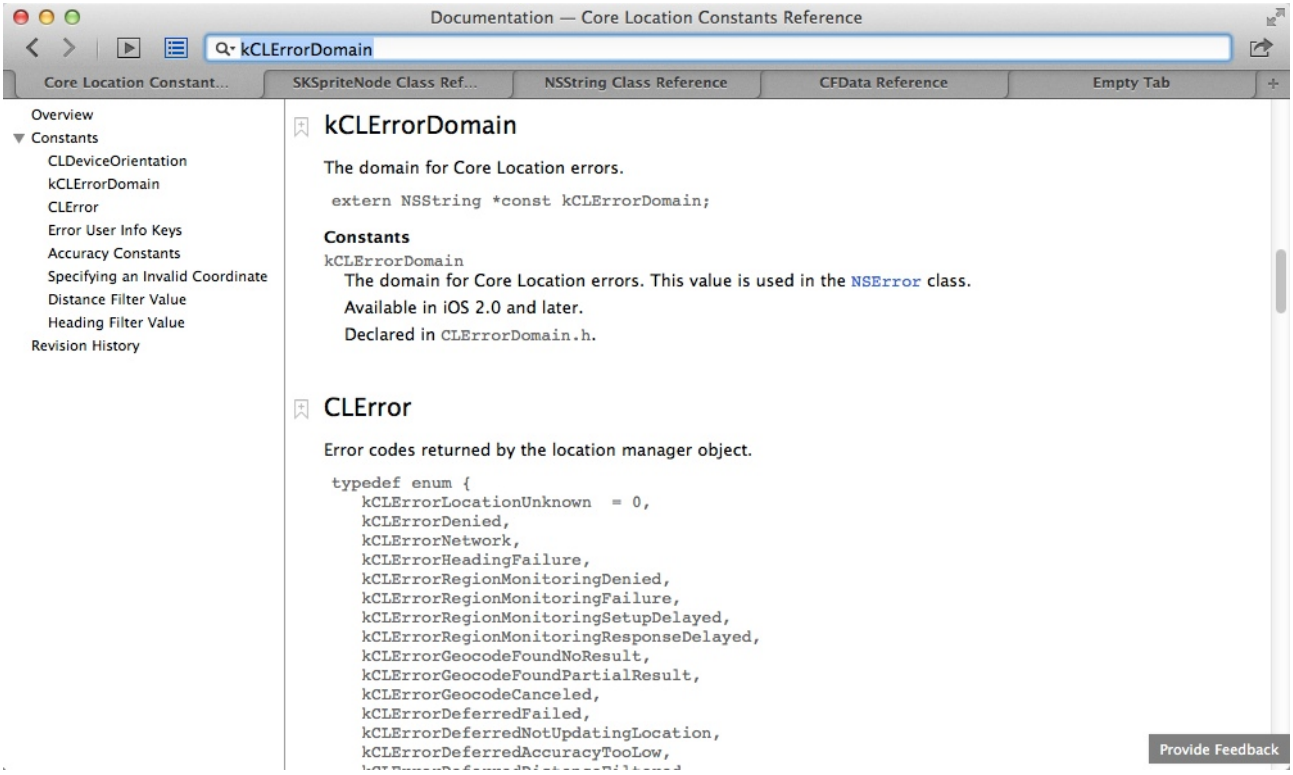


然后就是didUpdateLocations:这个方法。通过官方文档我们看到它是optional的，同时也是recommended的。其作用是通知代理对象有新的地理位置信息可用，而信息就在NSArray类型的参数中。该数组是由一系列的CLLocation类型的对象组成的，任何时候都会有1个或多个地理位置对

象。这些CLLocation对象的顺序是按照获取的顺序来排列的，因此数组的最后就是最新的坐标信息。
所以你会看到我们调用了lastObject方法来获取最新的坐标信息。并通过NSLog在调试区输出。

这里还要再补充说明一点，pragma mark的作用。

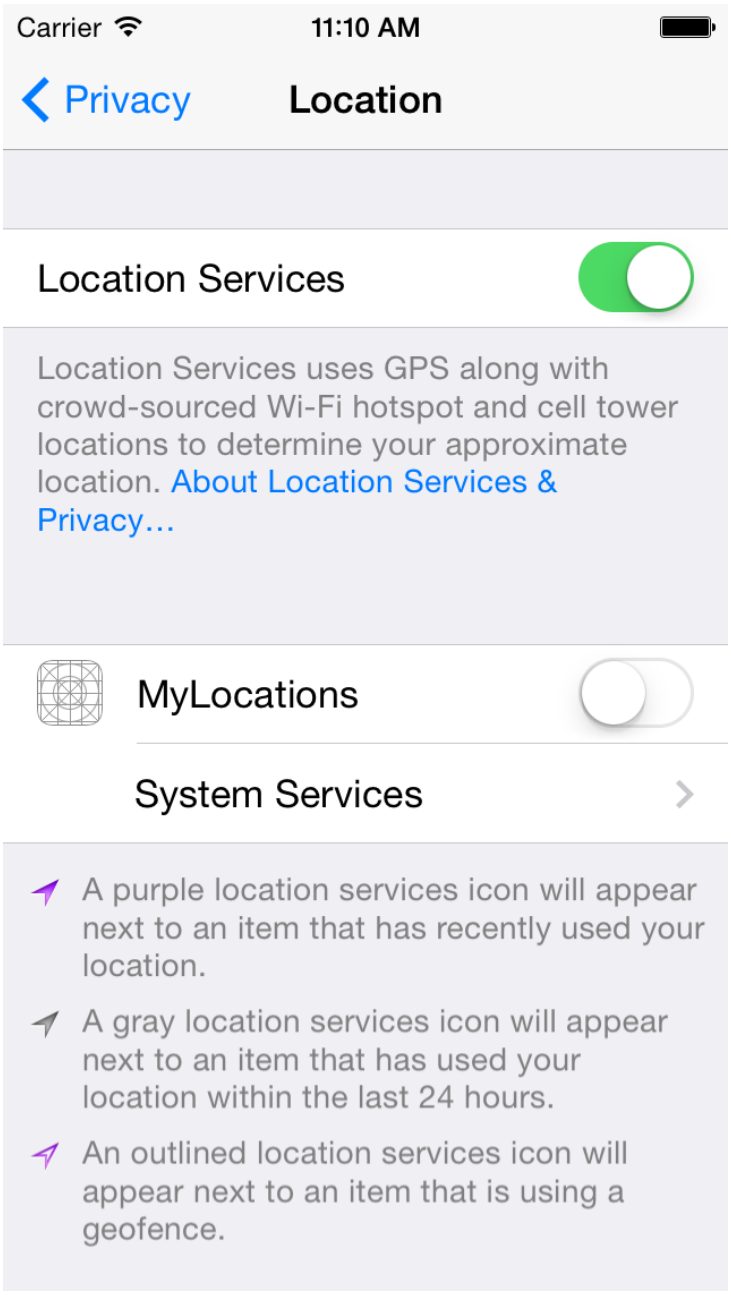
其实在之前的教程中有提到过，以#pragma mark开头的代码并非真正的可执行代码。它的作用类似于注释，不过注释最主要是针对人类的，方便人类读懂代码。而pragma(编译指示）除了可以有效组



织代码，同时也可以让编译器清楚的知道它该如何处理源代码。我们接触最多的就是#pragma 这个编译指示标记了，它会告诉Xcode，开发者把源代码组织的很漂亮。
要知道，按照苹果的产品设计理念，无论是软件还是硬件，美都是由内到外的。

好了，在Simulator中编译运行应用，然后按下Get My Location按钮（或者你随便取的其它名字）。
别告诉我你找不到Get My Location按钮，你不是在学校应付考试，别学傻了！

当我们首次运行应用按下Get My Location按钮的时候，会看到下面的提示：

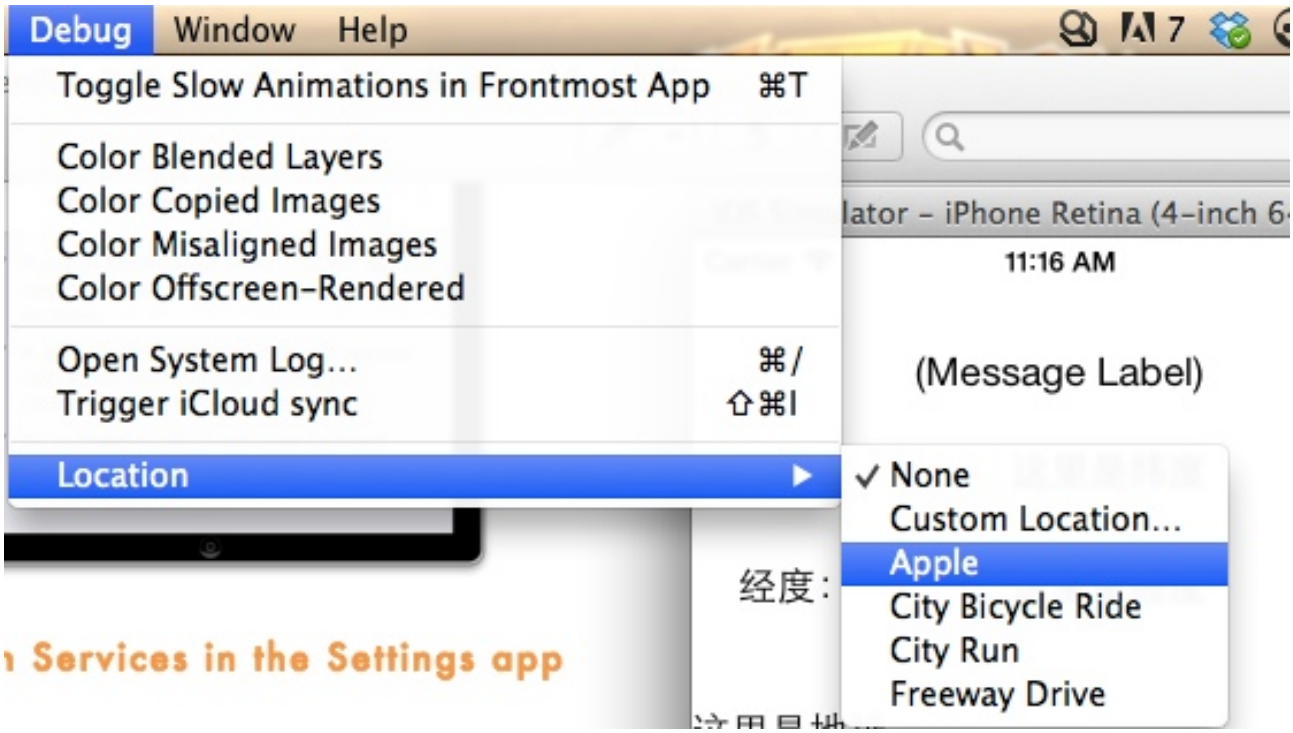


如果你选择了Don't Allow按钮，那么Core Location框架就无法向应用提供地理位置信息。

可以试试看，此时在Xcode的debug区会看到下面的信息“

这就是locationManager:didFailWithError:这个协议方法的功劳了。它告诉我们location manager无法获取地理位置信息。原因在NSError对象中。NSError是iOS SDK提供的标准对象，其作用就是提供程序运行中的各种错误信息。如果你学过java，或许了解其中的exception对象，作用和这个类似。
在后面的学习中，我们还将接触这个对象，因为这个世界太不完美了~
想了解NSError的详细信息吗？直接看苹果官方文档吧。
一个NSErro对象有一个domain和一个code属性。这里的domain是kCLErrorDomain，也就是说错误来源是Core Location(CL)。而code是1（或者说是kCLErrorDenied），也就是说用户不允许当前应用获取自己的位置信息。

你可能要问了，你怎么知道的？废话，你不会在苹果文档里面搜吗？



看到了吧，其中的枚举变量中，为1的值就是kCLErrorDenied.而在文档里面也解释了：

kCLErrorDenied
Access to the location service was denied by the user.
Available in iOS 2.0 and later.
Declared in CLError.h.

好了，现在我们可以停止这个应用，然后再次编译运行。

好吧，人生杯具再现。这次当你按Get My Location按钮的时候，什么都没发生，除了debug区和之前一样的错误提示。

换句话说，你的应用已经被用户列在了黑名单中，除非他/她哪天心情好改变了主意。

怎么改?

在Simulator中进入Setting,然后进入Privacy-Location。这里有个MyLocations应用，可以开启这个开关，这样就算是重新启用地理位置服务了。

好吧，这样或许有戏。
那么好，重新编译运行应用。
再次按下Get My Locations按钮，oops….

TNND，怎么还是error,哦，不对，这次的错误代码是0.
查查看刚才的文档是啥意思?

kCLErrorLocationUnknown
The location manager was unable to obtain a location value right now.
Available in iOS 2.0 and later.
Declared in CLError.h.

好吧，也就是说location unknown，无法获取坐标信息。

为啥?
很简单啊，因为我们是在Simulator中运行应用，貌似你的Mac中还没用装GPS吧?
虽然Mac可以通过Wi-Fi来获取地理位置信息，不过在simulator中没有内置这个功能。
肿么办?

办法1，既然simulator不行，就用真的设备呗。
方法2，让我们试着骗一下simulator，毕竟它还没有高级AI那么聪明。

方法1就不废话了，看看方法2怎么骗?
在应用运行的时候，从Simulator的菜单栏中选择Debug-location-Apple

好了，现在在debug区你就可以看到下面的信息了。

上面的信息显示持续进行，平均每秒就会向应用提供一个新的位置信息。不过过了一会儿后，其中的经度和纬度信息就保持不变了。这个地址在哪里?
这就是我大苹果教圣地-位于加州Cupertino的苹果总部啊!
只是，我们再也见不到这个熟悉的身影了。

注意看这里的坐标信息细节。最初是”+/-500.00m”，第二个是”+/-65.00m”，接下来是”+/-50.00m”，然后是”+/-30.00m”，最后越变越小，稳定在”+/-5.00m”。这个变动的数据其实就是测量精度，用米来衡量。我们在这里看到的是对真实设备上的模拟。

当我们使用iPhone或其它智能手机（比如android手机）获取位置信息时，通常采用三种不同的方式来获取坐标。这三种方式是蜂窝网络定位、Wi-Fi定位和GPS无线通讯定位：

1.蜂窝网络

在任何时候都有效，只要有手机信号就行，不过它的精度最低。
这里不废话了，感兴趣的朋友可以参考这篇老文章：
http://labs.chinamobile.com/mblog/712208_82886

2.Wi-Fi定位

这种定位方式更加精准，不过仅当附近有可用的Wi-Fi路由器时蔡康永。该系统基于一个庞大的数据库，其中包含了无线通讯设备的地理位置。

3.GPS定位（Global Positioning System)

传说中高大上NB的卫星定位。不过既然是卫星定位，就需要获取一个卫星通讯，因此其速度最慢，而且在室内的时候通常表现不佳。

因此我们的智能设备（不光是现在的iPhone和Android智能手机，还包括今后的Google glass,智能手环之类的东东）可以通过以上的不同方式来获取位置信息，既有快速但精度不高的蜂窝定位和Wi-Fi定位，又有高精度但速度缓慢的GPS定位。更可怕的，以上三种方式都有可能在特定的情况下失效。比如有时候手机没信号，有时候接收不到GPS信号，有时候没有Wi-Fi信号（貌似最后这种情况最普遍啊）。因此如何获取位置信息并非那么简单。考虑到我们没有学过无线通讯原理，要自主判断这些复杂的情况有点勉为其难啊。

幸运的是，Core Location可以帮我们搞定最困难的那部分工作。比如将不同渠道所获取的坐标信息读取成一个有用的数据信息。为了避免让用户等待GPS定位的精确地址太久，Core Location会在第一时间将其获取到的位置信息发送给应用，然后再逐渐提高定位的精度。

小练习

如果作为土豪的你身边有一个iPhone,iPod touch或者iPad，尝试在设备上编译运行这个应用，然后看看结果是怎样的。如果你有多个设备，还可以注意观察在不同设备上的数据差异。

理论充电-异步操作(asynchronous operations)

获取地理位置信息就是传说中的asynchronous（异步）操作。
有些时候应用需要做一些耗时的工作，比如上传图片，下载更新数据，等等。按照正常的思维方式，我们开启了一个任务，然后等它完成并给出反馈。如果运气不好的话，或许这个等待是永恒的。。。比如断网了。对Core Location来说，在获取第一个位置信息前通常需要1到2秒时间，同时需要更长的时间来获取更精确的数据。

这个时候我们可以考虑所谓的异步操作。也就是说当我们开启一个任务后，应用会继续执行其他的任务。最最重要的是，用户界面仍然是可以互动的，我们可以发送和处理新的事件，而用户也很开心，因为他们可以继续戳戳点点扮演上帝。异步操作又被称为所谓的后台操作（in the background)。一旦操作完成，应用会通过一个代理对象获得通知，然后就可以处理操作的结果了。

和异步(asynchronous)相反的是同步（synchronous)。如果我们开启了一个同步任务，那么应用就必须等待这个操作完成，才能继续其它的工作。
比如对我们这个应用的CLLocationManager来说就是一个很可怕的场面：有好几秒的时间应用完全没有任何反应，这种所谓的”僵直blocking”操作会给用户体验带来毁灭性和灾难性的破坏。

比如说MyLocations在底部有个tab bar。如果应用在获取位置信息的时候被锁死了，那么此时触碰tab上的其它选项卡就没有任何反应。用户本来还指望在等待获取位置的时候看看其它信息，结果现在应用直接被冰冻了，或者更可怕的是干脆崩溃了。

iOS的设计者，苹果的大牛们认为这种局面是不可接受的，因此对于此类耗时超过一秒的操作最好使用异步方式。

在下一系列的教程中我们将多次用到异步操作，因为下一系列的教程会教你如何建立和使用网络连接，并从互联网上下载东西。

顺便提一下，iOS有一个”watchdog timer看门狗计时器“的稀奇玩意儿。如果你的应用长时间没有反应，那么在某些特定的情况下watchdog看门狗就会毫不留情的干掉你的应用！

提醒自己，任何可能被用户注意到的长耗时操作（即便只是执行运算）都需要采用后台执行的异步模式！

晕，不知不觉今天的内容超标了，希望你45分钟内能看完~话说一个人自言自语有点超级无聊，从下一课开始我也来引入一个虚拟的菜鸟徒弟来对话吧。

送上今日福利，美女看多了也觉得没意思，换点美食吧。话说到底是23还是24过小年啊？为了这个差点和土豪兄吵起来。