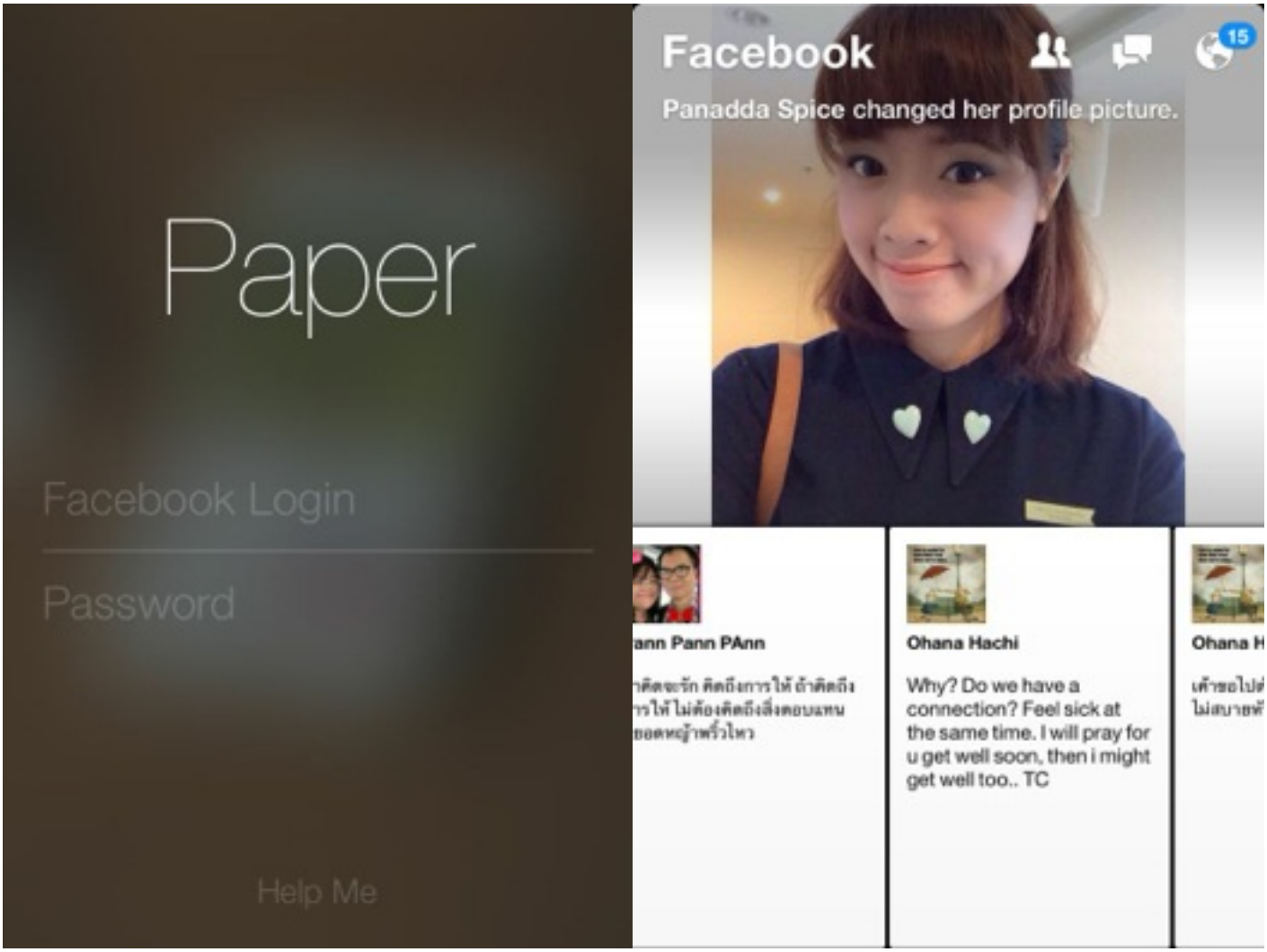
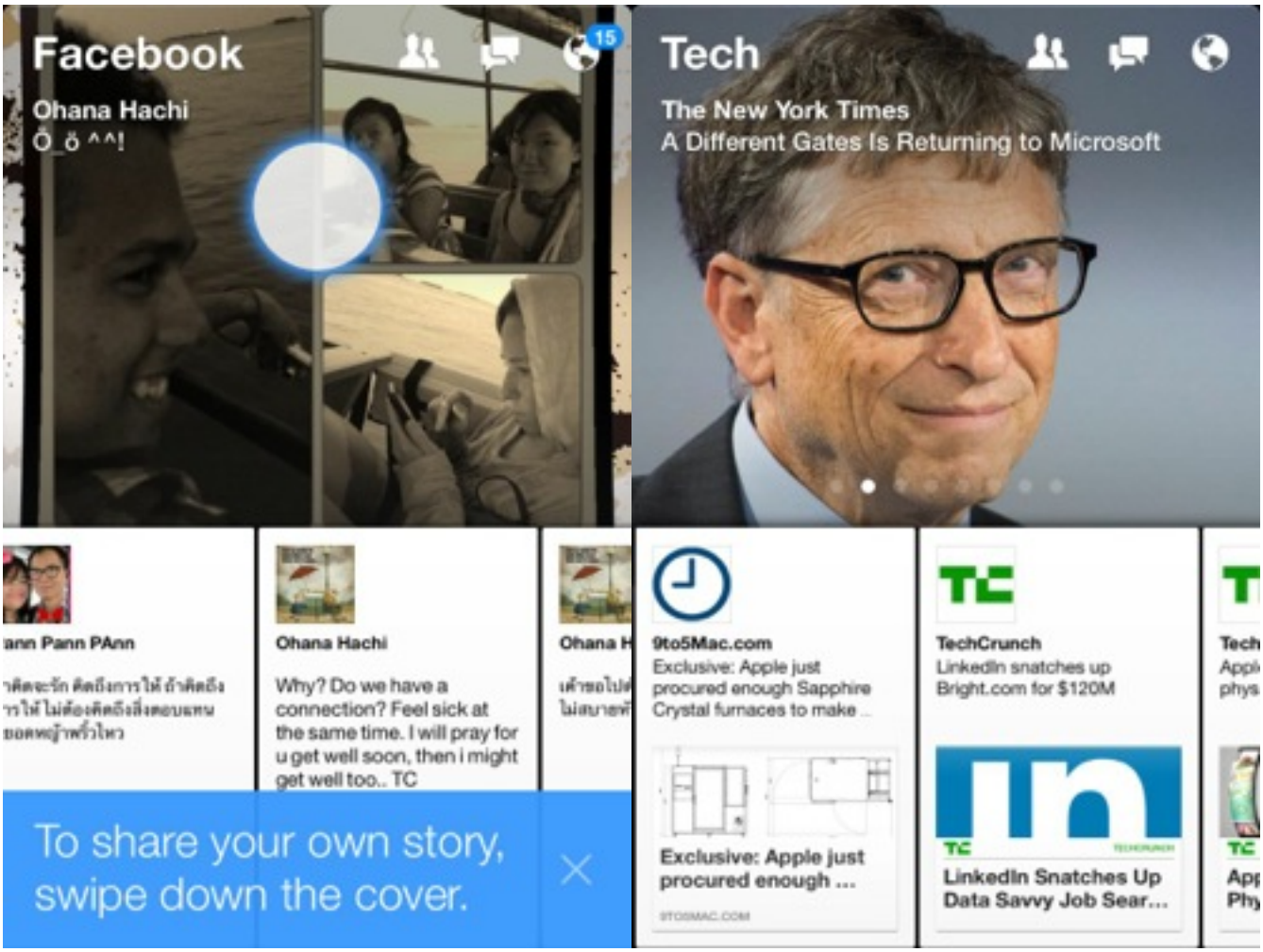
从零开始学iOS7开发系列3-我的地盘我做主-Cha10

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。

新年好，在这个欢乐祥和的马年春节里面，一定没少收各种红包吧？我初略统计了下，各种份子钱、微信红包发了不下3000元，而收到的只有累积不超过100元的微信红包。。。这，这让人情何以堪啊？！
各种土豪们，不管你们在哪个微信群，以后一定要拉我入伙啊，聚沙成塔，哪怕是0.01元的红包我也不嫌少的。记得我的微信号：iseedo
在开始今天的课程前，首先推荐一款应用，Facebook昨天刚发布的新应用Paper。额，这个Paper和之前的某个知名应用不同，是Facebook推出的一款媒体聚合应用。如果你用过Zaker，姑且可以有个基本的概念。
不过这个Paper应用的设计和体验实在是太NB了，是我目前为止见过的最高大上的Facebook推出的产品（包括网站，应用，游戏），同时也是2013年6月以来唯一一款令人耳目一新眼前一亮的应用。且慢，我们不是来学iOS开发的吗，在这里打这个广告貌似有点不妥吧？谬矣。昨天刚刚看了一篇文章，是对当下美利坚火爆的全民coding运动的批判。里面提到一个观点，现在的全民编程运动更看重的是学习coding，而不是programming。神马是coding,神马是programming？这里就不展开说了，你可以大略理解为码农码奴和Neo的区别。一个只写代码，一个用代码解决问题。
看到此文后，我很想把这个专栏的后缀从kidscoding改为kidsprogramming，不过考虑到用户体验（programming这个单词太长，还不好记），还是就这样吧。

在继续之前，先上几张Paper的靓图！





当然，遗憾的是从这几张截图完全体现不出它的高大上，因为它真正的亮点必须在实际使用的时候才能感受到。比如视频intro，游戏般的语音提示，全屏界面，取消所有的导航栏，只需要不断的向下滑动！好久没有看到这种让人耳目一新眼前一亮的应用了！用Paper的时候，你就知道什么叫丝一般柔滑的感觉了！

当然，作为一个开发者，从专业（装B）的角度来看，这个应用的交互体验实现肯定借用了iOS7的一些新特性，比如Dynamic。可惜时间不够用，不然我真有一种把它“山寨”一下来练手的冲动。

说到“山寨”，无论是学设计还是学开发（注意我用的是开发，不是coding)的XD，其实在刚入门的时候可以考虑从“山寨”这些优秀的产品开始。当然，“山寨”的目的不是为了放到appstore上去欺名盗世和欺骗小朋友捞金，而是为了在模仿的过程中练手。同时因为有一个完整的产品作参考，作为实战练手项目是非常好的。不过如果你最后还放到appstore上去卖，那就是节操问题了。

不知不觉，竟然花了这么长时间来八卦和摆龙门阵，实在有点对不起良心啊。还是来继续之前的学习吧。

神马？你已经忘了上一课的内容了？彻底无语了，只能小提示下，在上一课中我们已经学习了如何终止无休止的导航定位，免得让你的用户一夜醒来发现房子得卖掉！

接下来我们要学习一个新的名词Reverse geocoding(反向地理编码）

地理编码是神马意思造吗？就是把我们日常习惯用的地址转化成GPS的经纬度坐标信息。比如苹果教的圣殿所在地，1 Infinite Loop ,Cupertino,California，它的地理编码就是37.33240904,-122.03051218。你造这个数字的含义吗？想象你和GF约好去看电影，告诉小美要去28.11009832,-135.23232355这个地方见面。。。请问你是007或者周星驰童鞋吗？何弃疗？

所以当我们通过iPhone获取到GPS坐标信息后，还必须通过反向地理编码把这些人类理解不能的数字转换成实际生活中常用的地址。这就是reverse geocoding,coding,coding…
当然，既然知道reverse geocoding的意思了，那么geocoding的意思也就不用多解释了的吧。

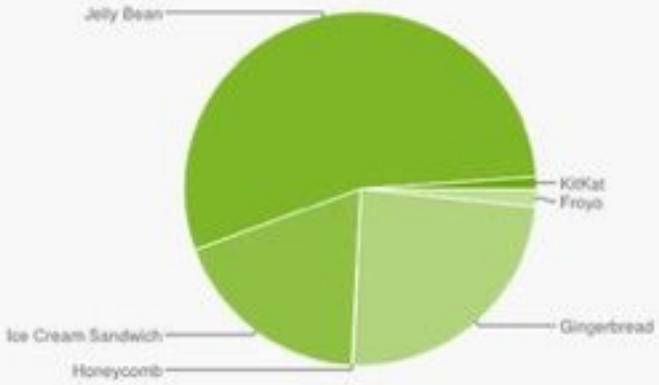
在iOS 中不管是geocoding还是reverse geocoding都是很轻松的事情，不过这篇教程中我们主要解决的是人类的需求。

注意：

假如你的手边还有远古时代的iOS项目要维护，可能里面会用到MKReverseGeocoder对象，不过从iOS5就把它删除了。也就是说在新的项目中不能使用该对象，新的替代对象是CLGeocoder。

这里多废话一句，在学习和使用iOS开发的过程中，一定要把握一个基本原则，那就是-一直向前看，别回头!
这一点和Android开发有本质的区别，目前为止Android Jelly Bean的占有率仍然最高（4.1,4.2,4.3），

Version	Codename	API	Distribution
2.2	Froyo	8	1.6%
2.3.3 - 2.3.7	Gingerbread	10	24.1%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	18.6%
4.1.x	Jelly Bean	16	37.4%
4.2.x		17	12.9%
4.3		18	4.2%
4.4	KitKat	19	1.1%



如果你是从Android开发转过来的，恭喜你，以后再也不用担心妈妈问你旧版本兼容的问题了。
因为苹果已经下了强制命令，从2月份开始所有的应用都必须支持最新的iOS7！！！

这也是我一直以来更喜欢苹果开发的原因，始终追求不懈的创新，一直向前，别回头！

我们将使用CLGeocoder对象将GPS坐标数据转换成人类可以理解的地址，然后在界面的addressLabel标签上显示。当然，实现这一点并不难，不过要注意几个准则。首先，我们不能同时发出N个反向地理编码请求。reverse geocoding的过程是通过Apple的服务器来完成的，因此每一个反向地理编码都会占用苹果服务器的带宽和处理器时间。如果你应用中的此类请求把苹果服务器阻塞了，那么Apple绝逼饶不了你。

不过说到这里，嘿嘿，我们岂不是多了一种黑苹果服务器的方式？！
好吧，就当我没说，相信Apple不会这么傻的放任你去执行阻塞攻击的，不过，架不住人海战术吗，嘿嘿。

因为MyLocations应用的设计定位，用户只会偶尔使用下，所以不太可能会黑掉苹果的服务器。不过我们仍然应该限制每次的地理编码请求数量。reverse geocoding需要靠网络连接来实现（因为需要借用苹果服务器），而我们在设计和开发产品的时候应该尽可能为用户着想，帮他们节约流量。否则，下一个被删的应用就是你哦！

打开久违的Xcode,切换到CurrentLocationViewController.m，然后添加几个实例变量的声明如下：

```
    CLGeocoder *_geocoder;
    CLPlacemark *_placemark;
    BOOL _performingReverseGeocoding;
    NSError *_lastGeocodingError;
```

这几行代码和之前声明location manager相关的代码类似。CLGeocoder这个对象的作用是进行地理编码，而CLPlacemark这个对象则包含了地址结果。当进行某个geocoding操作的时候，我们会将_performingReverseGeocoding设置为YES。如果出现了某种错误，_lastGeocodingError中会包含一个NSError对象。

在CurrentLocationViewController.m中，在initWithCoder方法中创建一个geocoder对象：

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

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

上面的黄色高亮部分创建并初始化了用于地理编码的对象。
接下来就是在didUpdateLocations方法中让其投入工作了：

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

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

    if([newLocation.timestamp timeIntervalSinceNow] < -5.0){
        return;
    }

    if(newLocation.horizontalAccuracy <0){
        return;
    }
```

```
    if(_location == nil || _location.horizontalAccuracy > newLocation.horizontalAccuracy){

        _lastLocationError = nil;
        _location = newLocation;
        [self updateLabels];

        if(newLocation.horizontalAccuracy <= _locationManager.desiredAccuracy){
            NSLog(@"***目标诺森德！成功完成定位");
            [self stopLocationManager];
            [self configureGetButton];
        }

        if(!_performingReverseGeocoding){
            NSLog(@"*** Going to geocode");
            _performingReverseGeocoding = YES;

            [_geocoder reverseGeocodeLocation:_location completionHandler:^(NSArray
*placemarks,NSError *error){

                NSLog(@"*** Found placemarks: %@, error: %@",placemarks,error);

                _lastGeocodingError = error;
                if(error == nil && [placemarks count] >0){
                    _placemark = [placemarks lastObject];
                }else{
                    _placemark = nil;
                }

                _performingReverseGeocoding = NO;
                [self updateLabels];
            }];

        }

    }

}
```

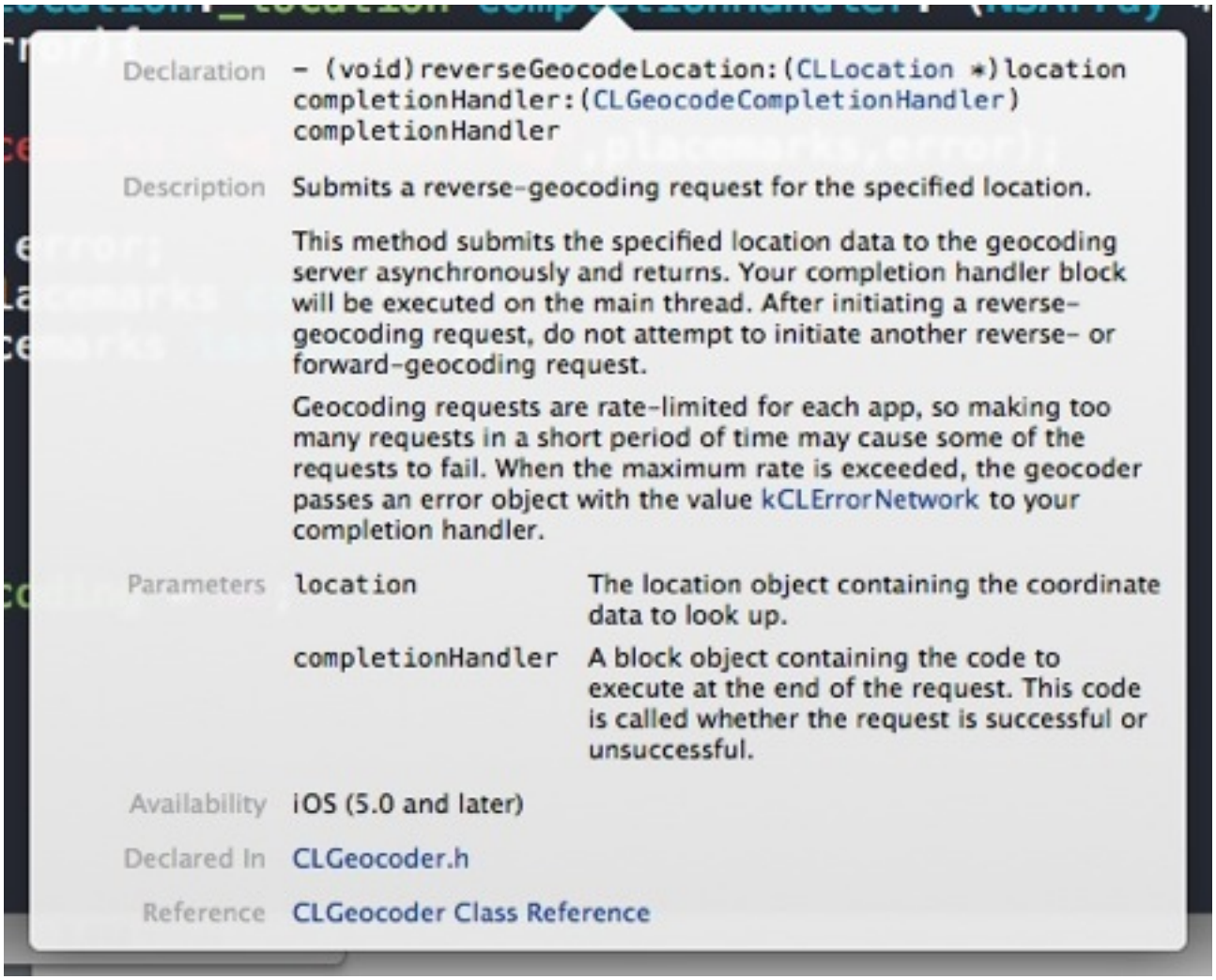
好吧，看到上面的一大堆新代码是不是有一种崩溃的赶脚？这尼玛还是人看的代码吗？
和之前的location manager不同，CLGeocoder没有用代理对象来通知相关结果，而是用了一种叫block（块语句）的可怕而神秘的东西。Blocks对Objective-C语言来说也是个新事物（iOS4才引入的）。使用块语句可以更方便的写某些特定的代码块。从iOS4之后，很多新API都采用块语句而非代理对象的方式。

那么既然有代理，怎么还会出现block这种奇怪的代码生物体呢？原因很简单，使用代理的问题在于，我们需要多写一个或几个方法。比如对CLLocationManager来说，就需要实现didUpdateLocations和didFailWithError方法。通过实现不同的方法，我们让代码来响应不同的请求。而使用blocks则不同，处理响应的代码是在内部实现的。
比如刚才的块语句：

```
[_geocoder reverseGeocodeLocation:_location completionHandler:^(NSArray *placemarks,
NSError *error) {
// put your statements here

}];
```

来看看苹果官方的文档解释：



通过这个块语句，程序通知CLGeocoder对象，我们希望对当前地理位置坐标进行反向地理编码。而当这个编码工作完成之后，就会立即执行块语句中completionHandler之后大括号中的预计。

这里的block是：

```
^(NSArray *placemarks, NSError *error) {
// put your statements here
}];
```

通过这个可怕的^符号，我们就知道了这里用的是块语句。而^后面（）中的参数则是块语句的参数，它们的用法和方法或函数中的参数相同。当geocoder通过你所提供的位置对象查找到了结果（或是遇到了错误）时，就会触发块语句，并执行其中的代码行。
这里的placemarks变量包含了CLPlacemark对象的数组，而CLPlacemark对象则描述了地址信息。error变量当然就是出错时的信息。

这说明了神马?

注意，这就说明了块语句中的代码不会在调用didUpdateLocations方法的同时执行。反之，块语句及其中所有的代码都会当CLGeocoder对象完成了反向地理编码（或出错）时才会执行。只有在此时块语句中的代码才会被执行。换句话说，实际上我们定义了一个内联的代理方法。

好吧，正如代理和协议让大家很恼火一样，块语句blocks很可能让你今天的午餐难以下咽。不过没关系，学习一种知识和技能是需要不断重复的。因此我们会在下一个教程中不断让你接触块语句，直到你做梦都会梦到它（噩梦啊，放过我吧么么哒）。如果你之前学过其它编程语言，你可以把它看做closures（闭包）。

让我们回到reverse geocoding的那大段代码。刚才说过我们需要对应用进行限制，让它每次只执行一次编码请求，因此我们首先检查geocoder是否正在繁忙之中：

if (!_performingReverseGeocoding) { NSLog(@"*** Going to geocode");

接下来我们会启动geocoder，然后让它执行块语句。

```
_performingReverseGeocoding = YES;
[_geocoder reverseGeocodeLocation:_location
completionHandler:^(NSArray *placemarks, NSError *error) {
NSLog(@"*** Found placemarks: %@, error: %@",
placemarks, error);
```

正如对location manager的处理那样，我们需要保存错误对象以便后续可以参考，只不过这里用到了另外一个实例变量来保存它：

```
 _lastGeocodingError = error;
            if(error == nil && [placemarks count] >0){
                _placemark = [placemarks lastObject];
            }else{
                _placemark = nil;
            }

            _performingReverseGeocoding = NO;
            [self updateLabels];
        }];
```

如果一切正常，而且在placemarks数组中有可用的对象，就从中取出最后一个对象。通常情况下在数组中只应该有一个CLPlacemark对象，不过有时候可能会出现奇特的现象，那就是一个坐标对应多个地址（平行时空？）因此我们的应用只能处理一个地址，所以就要选择最后一个。

注意到这里我们用到了一点点所谓的defensive programming（防御型编程）的概念。
我们特别检查了数组中是否存在对象，如果没有error错误的话应该是有的，不过我们不要盲目相信机器之心，所以还是要防范于未然。

如果存在错误，那么就将_placemark设为nil。咦，为何之前对location对象没有这么做呢？如果发生某个错误的话，我们仍然有之前的地址对象，因为它仍然有可能是正确的（或足够精确的），至少比没有好。不过对地址来说就不是这么回事了。我们不想显示一个旧的地址，因为只有对应当前位置的地址才是有效的，否则还不如不要。

小伙伴们，再次提醒大家。在移动应用开发的时候，没有任何事情是完美的。我们有可能会获取到位置信息，也有可能获取不到。即便获取到了，也可能不太精确。如果网络连接正常，那么reverse geocoding可能会顺利完成，不过在实际中当然也可能没那么顺利。最后，并非所有的GPS坐标都可以对应真实的地址（在撒哈拉沙漠中会有xxx路52号的概念吗？）

好吧，废话不多话，还是来编译运行跑跑看吧。

选择一个location看看效果。当发现首个位置的时候，你会看到debug区的reverse geocoder开始启动了。如果你选择的是Apple总舵所在，那么就会看到重复提示相同的位置。而只有当读数的精度达到一定程度时才会再次启用reverse geocode。

当然，这么做对我们的用户有点不公平，所以还是要在界面中显示结果。

更改updateLabels方法的代码如下：

```
-(void)updateLabels{

    if(_location !=nil){
        self.latitudeLabel.text = [NSString stringWithFormat:@"%.8f",_location.coordinate.latitude];
        self.longtitudeLabel.text = [NSString stringWithFormat:@"%.
8f",_location.coordinate.longitude];
```

```
        self.tagButton.hidden = NO;
        self.messageLabel.text = @"";

        if(_placemark !=nil){
            self.adderssLabel.text = [self stringFromPlacemark:_placemark];
        }else if(_performingReverseGeocoding){
            self.adderssLabel.text = @"寻找中...";
        }else if(_lastGeocodingError !=nil){
            self.adderssLabel.text =@"对不住了姐，出错了";
        }else{
            self.adderssLabel.text =@"啥都没找到";
        }

    }else{

        self.latitudeLabel.text = @"";
        self.longtitudeLabel.text = @"";
        self.adderssLabel.text = @"";
        self.tagButton.hidden = YES;

        NSString *statusMessage;
        if(_lastLocationError != nil){

            if([_lastLocationError.domain isEqualToString:kCLErrorDomain] &&_lastLocationError.code
== kCLErrorDenied)
            {
                statusMessage = @"对不起，用户禁用了定位功能";
            }else{
                statusMessage =@"对不起，获取位置信息错误";
            }
        }else if(![CLLocationManager locationServicesEnabled]){
            statusMessage =@"对不起，用户禁用了定位功能";

        }else if(_updatingLocation){
            statusMessage = @"定位中...";
        }else{
            statusMessage = @"请触碰按钮开始定位";
        }
        self.messageLabel.text = statusMessage;

    }


}
```

因为我们只想对一个位置信息进行一次地址查询，因此只需要在第一个if中更改代码。如果找到了地址，就展示给用户，否则就显示一个状态信息。
为了保证代码的可读性，用了一个独立的stringFromPlacemark:方法来生成字符串：

```
-(NSString*)stringFromPlacemark:(CLPlacemark*)thePlacemark{

    return [NSString stringWithFormat:@"%@ %@\n%@ %@
%@",thePlacemark.subThoroughfare,thePlacemark.thoroughfare,thePlacemark.locality,thePlacem
ark.administrativeArea, thePlacemark.postalCode];
}
```

如果你英文还过得去的话，或许知道subThoroughfare是房间号，thoroughfare是街道名称，locality就是所在城市，adminstrativeArea是省份，而postalCode就是邮编。

接下来在getLocation方法中将_placemark和_lastGeocodingError变量设置成nil，以便开启一个新状态：

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

    if(_updatingLocation){
        [self stopLocationManager];
    }else{
        _location = nil;
        _lastLocationError = nil;
        _placemark = nil;
        _lastGeocodingError = nil;

        [self startLocationManager];
    }

    [self updateLabels];
    [self configureGetButton];
}
```

好了，再次编译运行应用，当找到位置信息几秒后会在地址标签中显示对应的地址。

注意：对于有些位置，地址标签会显示(null)。这是因为CLPlacemark中的信息不完整，比如街道名称不完整。在本教程的后面我们会进行改善。

练习：如果你在Simulator里面选择了City Bicycle Ride或者City Run，那么会在debug区看到一堆不同的坐标（它在模拟某个人不断运动）。但界面上的坐标和地址却不会这么频繁变化，这是为毛?

答案：MyLocations应用的设计理念是找到某个静止位置的最精确坐标集。当某个更精确位置信息到达的时候，我们只更新了_location变量。而具有更高或相同horizontalAccuracy精度的信息会被忽略，不管实际坐标是多少。

而使用City Bicycle Ride或City Run的时候，应用并没有接收到相同坐标的不断增加精度的信息，而是一组完全不同的坐标信息。
换句话说，我们这款应用不适合运动状态下的定位。

最后提示大家：
我在Simulator上做的测试结果是null，而在设备(iphone)上是可以的。

如果你是在Simulator上跑，那么debug区会看到：

所以，不要担心代码出错了，而是因为GPS定位本身就会存在种种实际问题。

好了，今天的课程到此结束，到了送福利的时候了。

顺便附上我的微信二维码 iseedo，大家有问题的话可以扫一扫，或者发邮件给我: eseedo@gmail.com

欢迎中二病深度患者、萌妹子、大爷大妈们来交流，不管你是小学三年级的红领巾，还是80+的退休老干部，只要对编程感兴趣，特别是iOS和游戏开发感兴趣的，都可以加我。

另外下个月打算开通个微信订阅号，放一些和k12学编程的文章在上面。