iOS应用程序一般都是由自己编写的代码和系统框架(system frameworks)组成，系统框架提供一些基本infrastructure给所有app来运行，而你提供自己编写的代码来定制app的外观和行为。因此，了解iOS infrastructure和它们如何工作对编写app是很有帮助的。

Main函数入口

所有基于C编写的app的入口都是main函数，但iOS应用程序有点不同。不同就是你不需要为iOS应用程序而自己编写main函数，当你使用Xcode创建工程的时候就已经提供了。除非一些特殊情况，否则你不应该修改Xcode提供的main函数实现。示例代码如下：

```
#import
#import "AppDelegate.h"
int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]))
    }
}
```

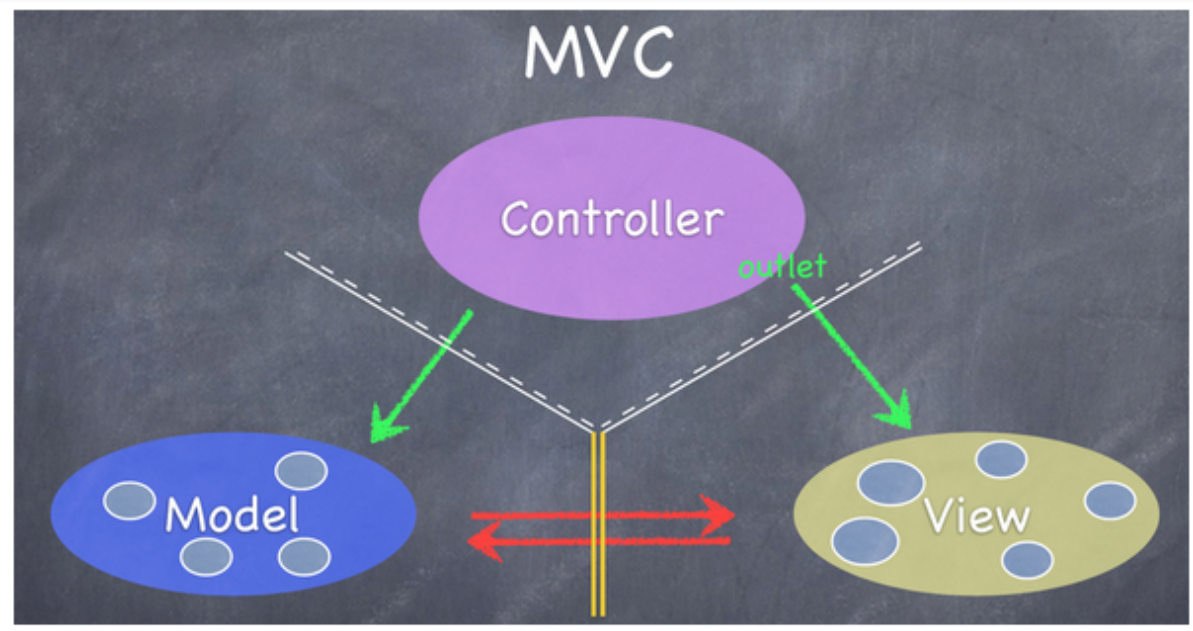
上面实例代码中有一个很重要的函数UIApplicationMain，它主要是创建app的几个核心对象来处理以下过程：

1. 从可用Storyboard文件加载用户界面
2. 调用AppDelegate自定义代码来做一些初始化设置
3. 将app放入Main Run Loop环境中来响应和处理与用户交互产生的事件

应用程序的架构

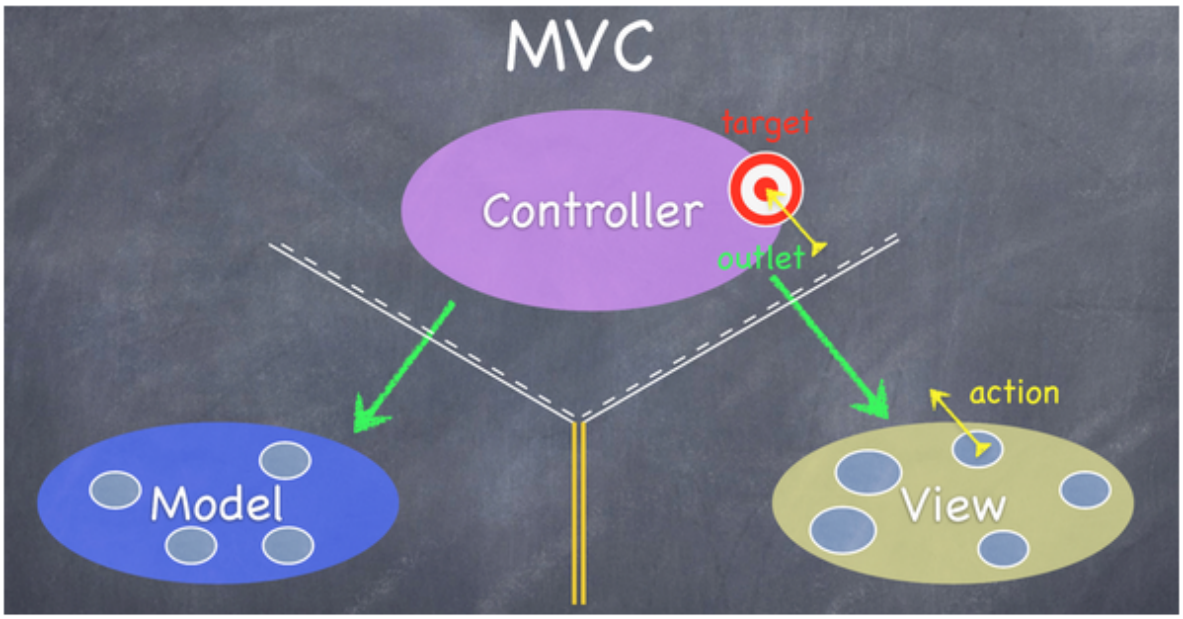
iOS应用程序都遵循Model-View-Controller的架构，Model负责存储数据和处理业务逻辑，View负责显示数据和与用户交互，Controller是两者的中介，协调Model和View相互协作。它们的通讯规则如下：

1. Controller能够访问Model和View，Model和View不能互相访问



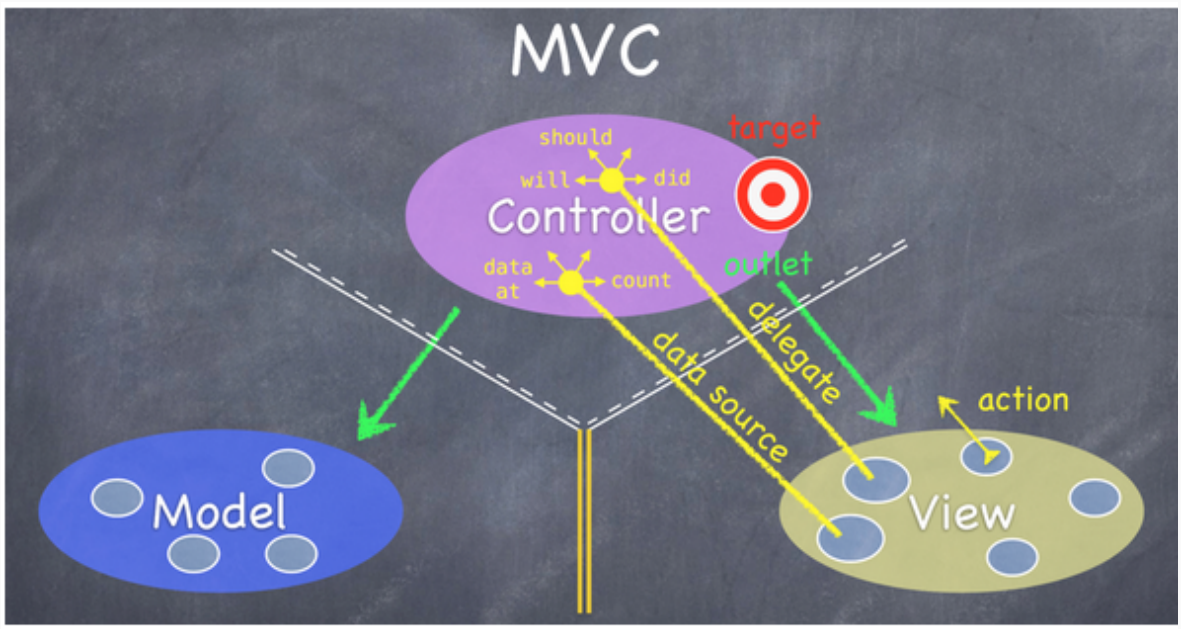
MVC Communication – Reference from Stanford University.png

2. 当View与用户交互产生事件时，使用target-action方式来处理



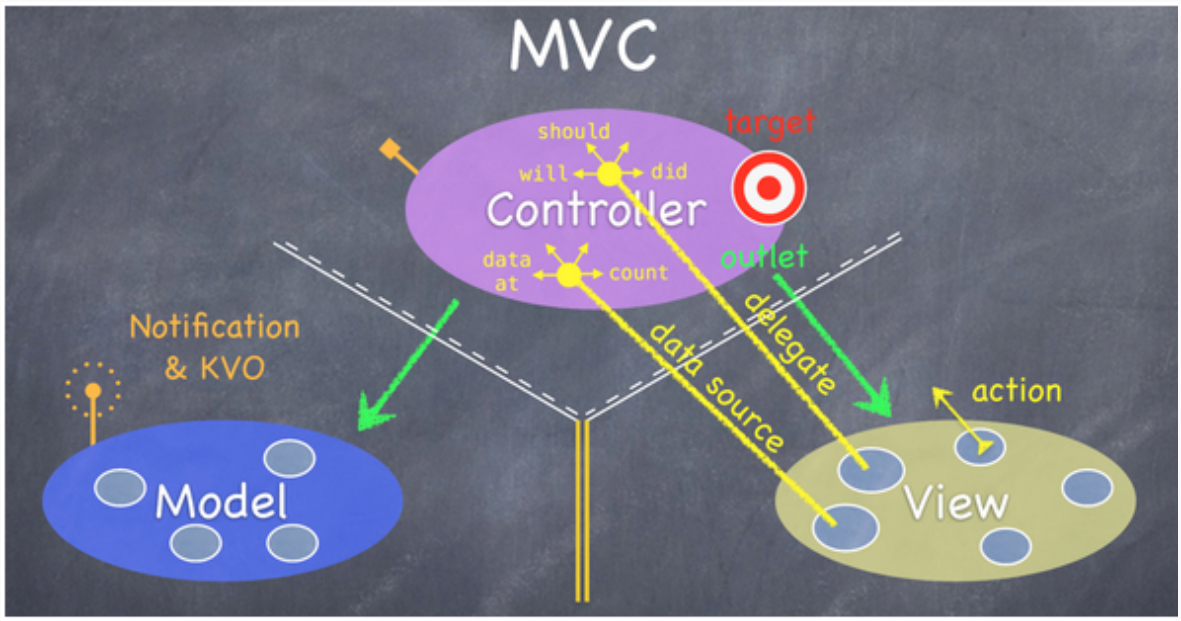
MVC Communication – Reference from Stanford University.png

3. 当View需要处理一些特殊UI逻辑或获取数据源时，通过delegate或data source方式交给Controller来处理



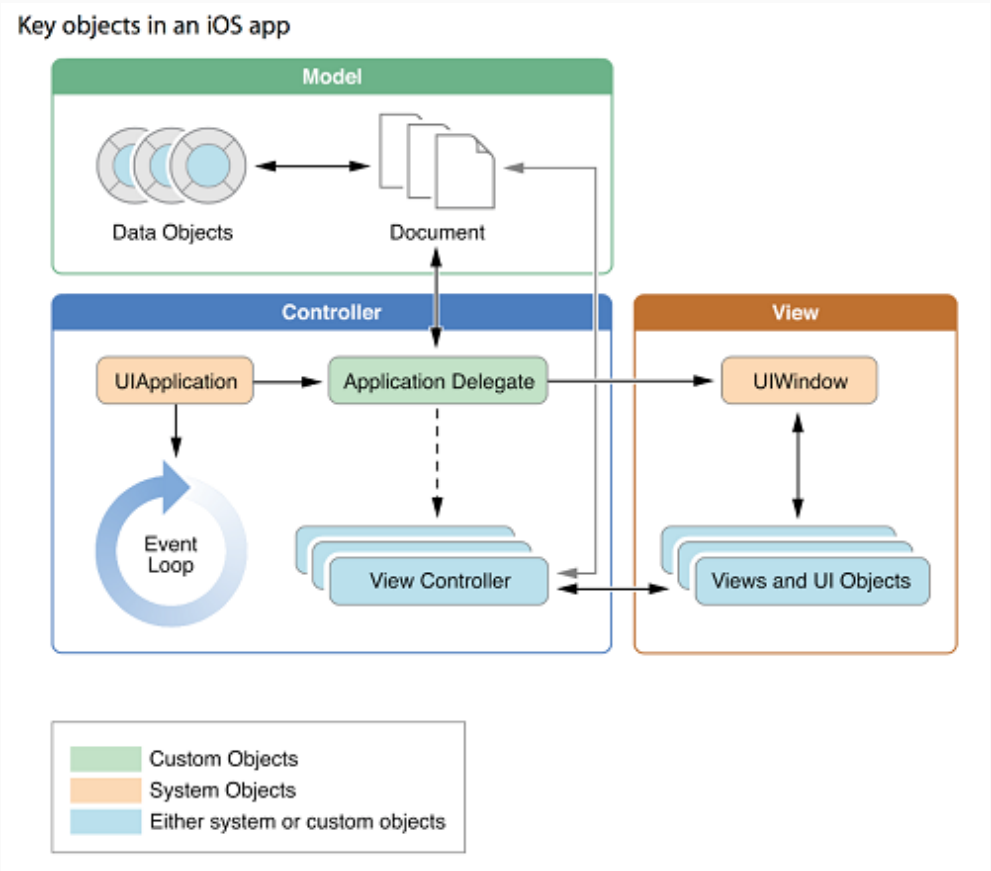
MVC Communication – Reference from Stanford University.png

4. Model不能直接与Controller通信，当Model有数据更新时，可以通过Notification或KVO (Key Value Observing)来通知Controller更新View



MVC Communication – Reference from Stanford University.png

了解iOS的MVC设计模式之后，我们从下图来了解在MVC模式下iOS应用程序有哪些关键对象以及它们职责主要是什么?



The Structure of an App.png

- UIApplication对象

  用户与iOS设备交互时产生的事件(Multitouch Events，Motion Event，Remote Control Event)交由UIApplication对象来分发给control objects(UIControl)对应的target objects来处理并且管理整个事件循环，而一

些关于app运行时重要事件委托给app delegate来处理。

- App delegate对象

  App delegate对象遵循UIApplicationDelegate协议，响应app运行时重要事件(app启动、app内存不足、app终止、切换到另一个app、切回app)，主要用于app在启动时初始化一些重要数据结构；例如，初始化UIWindow，设置一些属性，为window添加rootViewController。

- View controller对象

  View Controller有一个view属性是view层次结构中的根view，你可以添加子view来构建复杂的view；controller有一些viewDidLoad、viewWillAppear等方法来管理view的生命周期；由于它继承UIResponder，所有还会响应和处理用户事件。

- Documents和data model对象

  data model对象主要用来存储数据。例如，饿了么app在搜索切换地址后，有历史记录搜索地址历史，当app下次启动时，读取和显示搜索地址历史。

  document对象(继承UIDocument)用来管理一些或所有的data model对象。document对象并不是必须的，但提供一种方便的方式来分组属于单个文件或多个文件的数据。

- UIWindow对象

  UIWindow对象位于view层次结构中的最顶层，它充当一个基本容器而不显示内容，如果想显示内容，添加一个content view到window。

  它也是继承UIResponder，所以它也是会响应和处理用户事件。

- View、control、layer对象

  View对象可以通过addSubview和removeFromSuperview 等方法管理view的层次结构，使用layoutSubviews、layoutIfNeeded和setNeedsLayout等方法布局view的层次结构，当你发现系统提供view已经满足不了你想要的外观需求时，可以重写drawRect方法或通过layer属性来构造复杂的图形外观和动画。还有一点，UIView也是继承UIResponder，所以也能够处理用户事件。

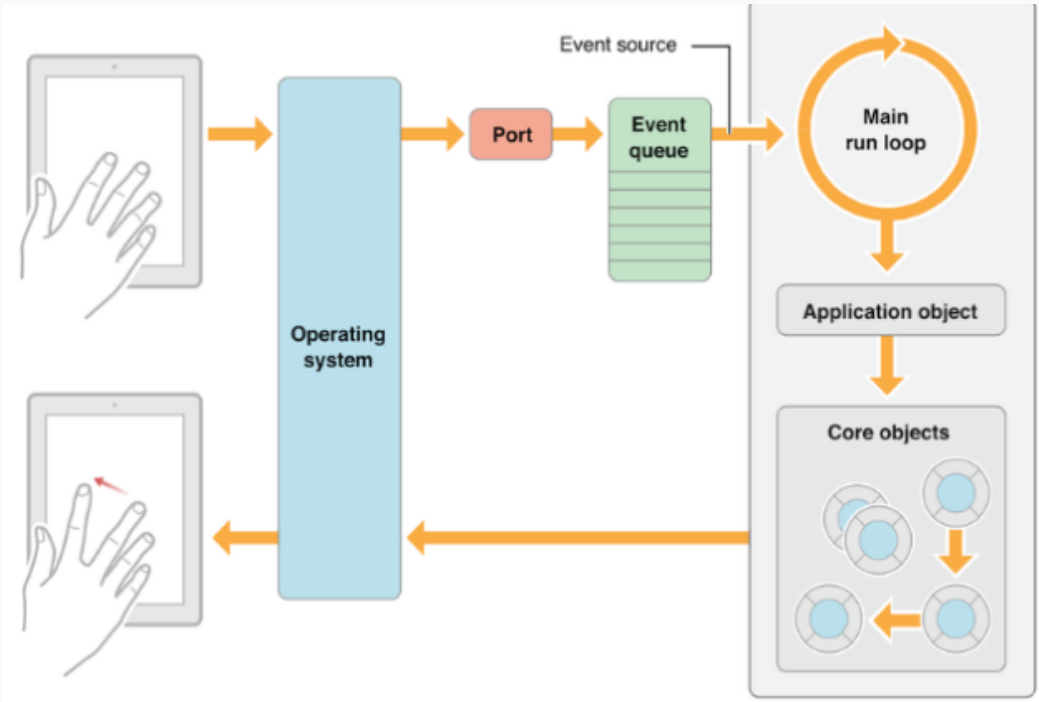
  Control对象通常就是处理特定类型用户交互的View，常用的有button、switch、text field等。

  除了使用View和Control来构建view层次结构来影响app外观之外，还可以使用Core Animation框架的Layer对象来渲染view外观和构建复杂的动画。

Main Run Loop

一个iOS应用程序的main run loop主要作用是处理所有与用户相关的事件。UIApplication对象在启动时就设置main run loop和使用它来处理事件和更新基于view的界面。正如它的名字显示，main run loop是运行在应用程序的主线程。这样就确保与接收到用户相关的事件被有序地处理。

下图显示main run loop的架构和用户事件最终是怎样被应用程序处理。当用户与设备交互时，系统就会生成与交互关联的事件，然后被应用程序的UIKit通过一个特殊的端口来分发。应用程序把事件放入队列，然后逐个分发到main run loop来执行。UIApplication对象是第一个对象接收到事件，然后决定怎样处理它。一个touch event通常都被分发到main window对象，然后依次分发到发生触碰的view。其他event的接收事件对象路径可能有点不同。

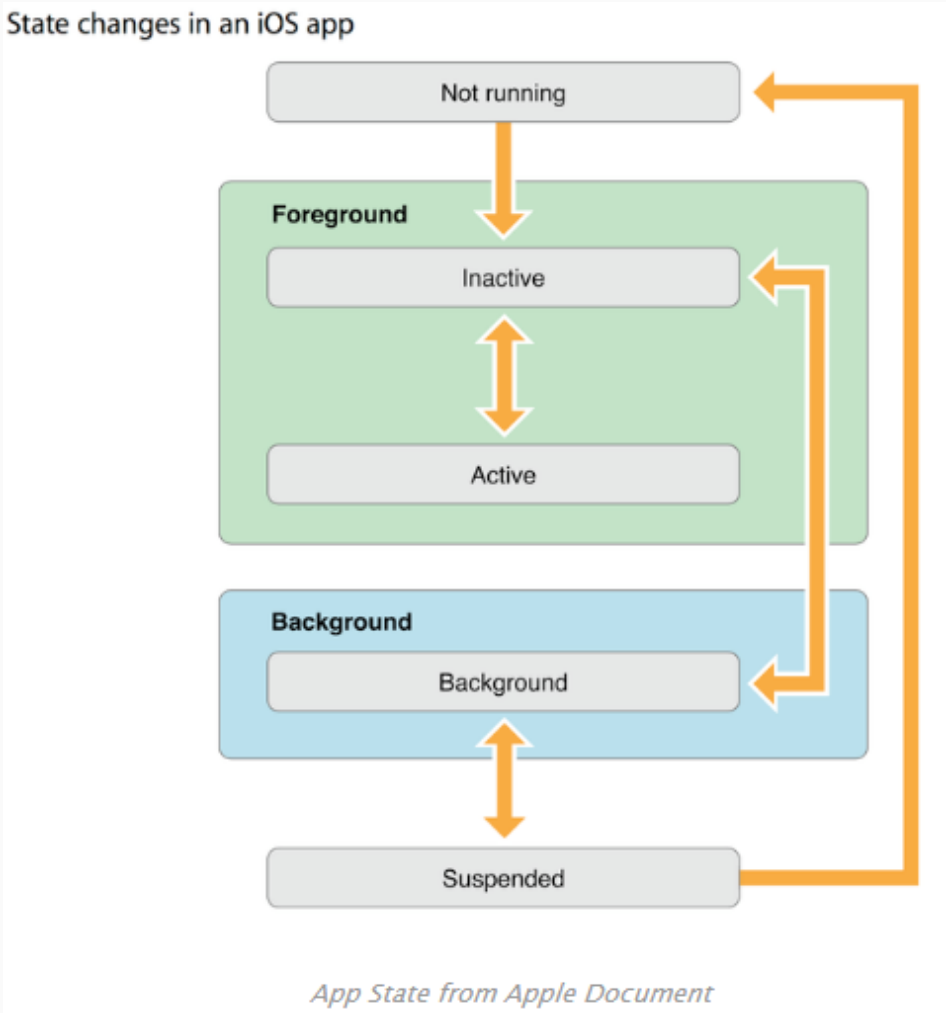


Main Run Loop from Apple Document

大多数的事件通过使用main run loop来分发，但有些不是。有些事件被发送到一个delegate对象或传递到你提供的block中。想了解更多如何处理大多数类型的事件，其中包括touch、remote control、motion、accelerometer和gyroscopic等事件，请查阅Event Handle Guide for iOS。

应用程序的状态和多任务

有时系统会从app一种状态切换另一种状态来响应系统发生的事件。例如，当用户按下home键、电话打入、或其他中断发生时，当前运行的应用程序会切换状态来响应。应用程序的状态有以下几种：



App State from Apple Document

- Not running：app还没运行
- Inactive：app运行在foreground但没有接收事件
- Active：app运行在foreground和正在接收事件
- Background：运行在background和正在执行代码
- Suspended：运行在background但没有执行代码

大多数发生状态转换时都会调用delegate对象对应的方法来响应app的状态改变。下面汇总了delegate对象的所有方法，当app状态发生转换时，你可能会使用到它们。

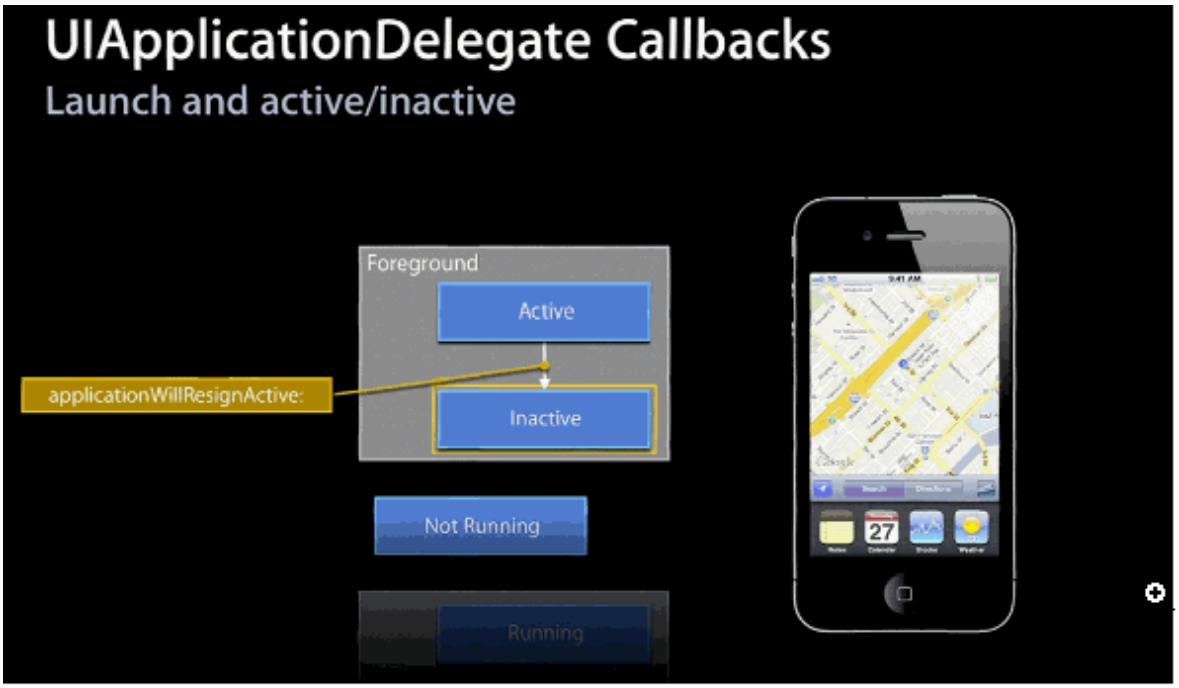
- application:willFinishLaunchingWithOptions: - 这个方法是你在启动时的第一次机会来执行代码
- application:didFinishLaunchingWithOptions: - 这个方法允许你在显示app给用户之前执行最后的初始化操作
- applicationDidBecomeActive: - app已经切换到active状态后需要执行的操作
- applicationWillResignActive: - app将要从前台切换到后台时需要执行的操作
- applicationDidEnterBackground: - app已经进入后台后需要执行的操作
- applicationWillEnterForeground: - app将要从后台切换到前台需要执行的操作，但app还不是active状态
- applicationWillTerminate: - app将要结束时需要执行的操作

现在讲下app启动、来回切换app和锁屏时状态的切换和调用对应哪些delegate对象的方法：

- app启动和active/inactive

Launch and active/inactive from Apple WWDC 2011 Session

如图所示，当app启动时，首先由not running状态切换到inactive状态，此时调用application:didFinishLaunchingWithOptions:方法；然后由inactive状态切换到active状态，此时调用applicationDidBecomeActive:方法。



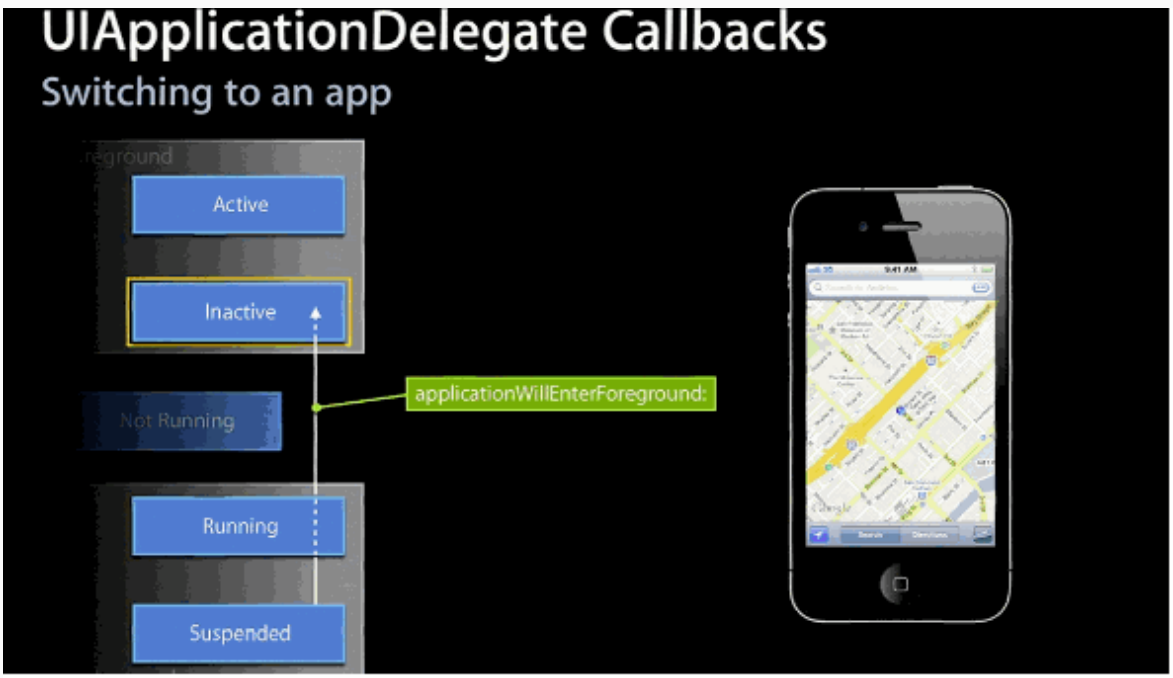
Launch and active/inactive 2 from Apple WWDC 2011 Session

当app发生中断时，由active状态切换到inactive状态，此时调用applicationWillResignActive:方法。

- 来回切换app

Switch from an app from Apple WWDC 2011 Session

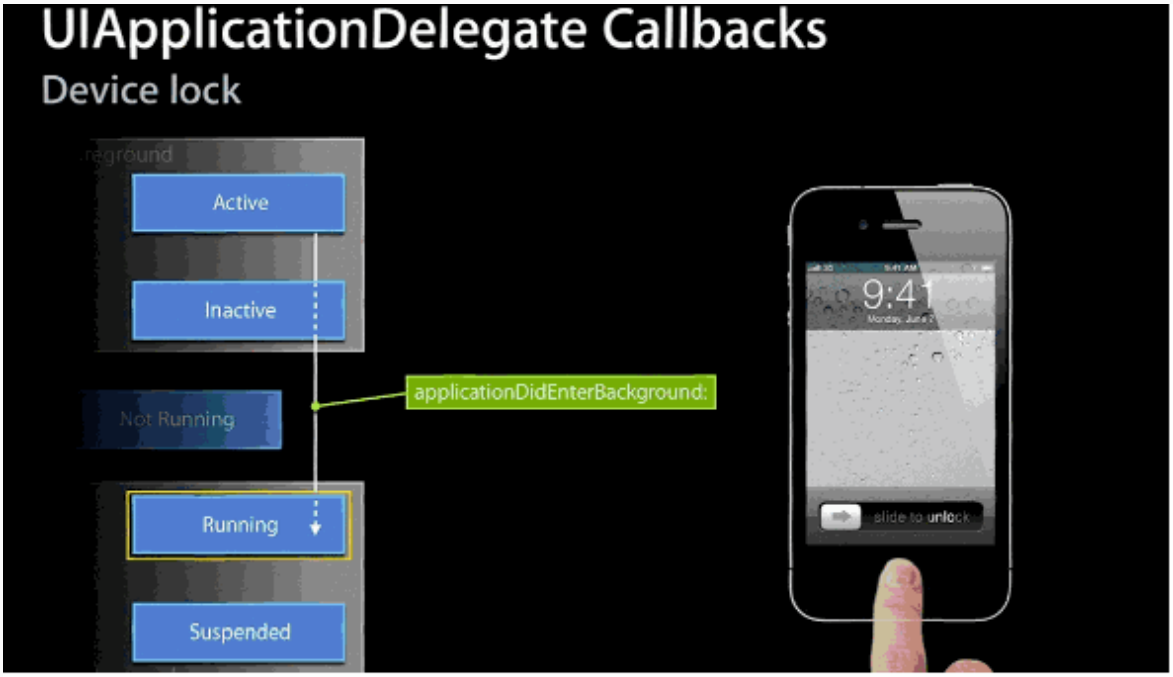
如图所示，当切换到另一个app时，由状态active切换到inactive，此时调用applicationWillResignActive:方法；然后从inactive状态切换到running状态，此时调用applicationDidEnterBackground:方法。



Switch to an app from Apple WWDC 2011 Session

而当切换回本来的app时，由running状态切换到inactive状态，此时调用applicationWillEnterForeground:方法，然后由inactive状态切换到active状态，调用applicationDidBecomeActive:方法。

- 锁屏



Device lock from Apple WWDC 2011 Session

如何所示，当手机锁屏时，由状态active切换到inactive，此时调用applicationWillResignActive:；然后再由inactive状态切换到running状态，此时调用applicationDidEnterBackground:方法。

更多关于app状态切换以及调用app delegate哪些方法，请观看WWDC 2011 Session的session_320__adopting_multitasking_in_your_app视频。

应用程序的终止

系统常常是为其他app启动时由于内存不足而回收内存最后需要终止应用程序，但有时也会是由于app很长时间才响应而终止。如果app当时运行在后台并且没有暂停，系统会在应用程序终止之前调用applicationWillTerminate:来保存用户的一些重要数据以便下次启动时恢复到app原来的状态。

总结

本文总结了iOS应用程序从启动到结束过程中有哪些关键对象在参与，以及当用户与系统交互时产生事件时，系统利用main run loop来管理事件循环，决定将事件交给系统哪些对象处理和如何处理。而当app启动、来回切换app和锁屏时，app的状态如何切换和调用对应的哪些app delegate对象来处理。