UIApplication 类

在IOS开发过程中，苹果已经为广大的程序员做好了很多事情，比如尽量简化编程上的IOS应用的启动流程，使程序员不必关心程序从何而来，但是了解IOS应用的启动机制，对于我们更好的掌握IOS编程是很有必要的。本文将详细介绍于IOS的生命周期相关的重要概念。

1,程序的启动函数UIApplicationMain详解

在IOS应用程序中一般都会看到如下代码:

#import <UIKit/UIKit.h>

#import "AppDelegate.h"

int main(int argc, char \*argv[])

{

@autoreleasepool {

return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));

}

}

UIApplicationMain函数原型如下:

int UIApplicationMain(

int argc,

char \*argv[],

NSString \*principalClassName,

NSString \*delegateClassName

);

第一、二个参数是 main() 函数传递的参数;

第三个参数是UIApplication[或者继承自UIApplication]类的名字，如果为空程序会自动创建一个UIApplication的实例。

第四个参数是应用程序代理类UIApplicationDelegate类的名字，如果为nil,系统会从Main nib[主要nib文件在info.plist文件中指定:Main nib file base name字段对应的value]文件寻找Application对象并连接它的delegate。

UIApplicationMain主要创建一个UIApplication对象和一个UIApplicationDelegate对象并设置事件循环，然后开始处理事件。

2,UIApplication/UIApplicationDelegate

在开发过程中我们需要一些全局对象来将程序的各个部分连接起来，这些全局对象中最重要的就是UIApplication对象。但在实际编程中我们并不直接和UIApplication对象打交道，而是和其代理打交道。

UIApplication是应用程序的开始，负责初始化并显示UIWindow，并负责加载应用程序的第一个UIView到UIWindow窗体中。

UIApplication的另一个任务是帮助管理应用程序的生命周期，它是通过 UIApplicationDelegate的代理类来履行这个任务。

UIApplication会负责接收事件，而 UIApplicationDelegate则决定应用程序如何去响应这些事件，UIApplicationDelegate可以处理的事件包括应用程序的生命周期事件（比如程序启动和关闭）、系统事件（比如来电、记事项警告）等。

通常编程人员是没必要修改UIApplication的，只需要知道UIApplication接收系统事件即可，需要关系的是UIApplicationDelegate接口提供生命周期函数来处理应用程序以及应用程序的系统事件。

每一个程序在运行期必须有且仅有一个UIApplication（或则其子类）的一个实例。在程序开始运行的时候，UIApplicationMain函数是程序进入点，这个函数做了很多工作，其中一个重要的工作就是创建一个UIApplication的单例实例。

在代码中，可以通过[UIApplication sharedApplication]来得到这个单例实例的指针。UIApplication的一个主要工作是处理用户事件，它会起一个队列作用，把所有用户事件都放入队列，逐个处理，在处理的时候，它会发送当前事件 到一个合适的处理事件的目标控件。

此外，UIApplication实例还维护一个在本应用中打开的window列表（UIWindow实例），这样它就可以接触应用中的任何一个UIView对象。

UIApplication实例会被赋予一个代理对象，以处理应用程序的生命周期事件（比如程序启动和关闭）、系统事件（比如来电、记事项警告）等等。

UIApplication相关的方法:

+sharedApplication类方法获取当前程序的UIApplication实例

delegate属性获取UIApplicationDelegate实现类的实例

windows属性获取当前程序涉及到窗口类数组(注意是数组)

keyWindow属性获取当前程序关键窗口

如何编写代码来处理系统事件是程序员的工作。处理系统事件需要编写一个继承自UIApplicationDelegate接口的类，而UIApplicationDelegate接口提供生命周期函数来处理应用程序以及应用程序的系统事件，生命周期函数如下所示

1、- (void)applicationWillResignActive:(UIApplication \*)application｛

/ /从主动到非活动状态的应用程序时发送。这可导致产生某些类型的临时中断（如传入电话呼叫或SMS消息），或者当用户退出应用程序，它开始过渡到的背景状态。使用此方法暂停正在进行的任务，禁用定时器，踩下油门， OpenGL ES的帧速率。游戏应该使用这种方法来暂停游戏。

｝

2、- (void)applicationDidBecomeActive:(UIApplication \*)application

说明：当应用程序入活动状态执行，这个刚好跟上面那个方法相反

3、- (void)applicationDidEnterBackground:(UIApplication \*)application｛

说明：当程序被推送到后台的时候调用。所以要设置后台继续运行，则在这个函数里面设置即可

//使用这个方法来释放共享资源，保存用户数据，废止定时器，并存储足够的应用程序状态信息的情况下被终止后，将应用程序恢复到目前的状态。

//如果你的应用程序支持后台运行，这种方法被调用，而不是applicationWillTerminate ：当用户退出。

｝

4、- (void)applicationWillEnterForeground:(UIApplication \*)application

说明：当程序从后台将要重新回到前台时候调用，这个刚好跟上面的那个方法相反。

5、- (void)applicationWillTerminate:(UIApplication \*)application

//不支持多任务的时候调用

说明：当程序将要退出是被调用，通常是用来保存数据和一些退出前的清理工作。这个需要要设置UIApplicationExitsOnSuspend的键值（自动设置）。

6、- (void)applicationDidReceiveMemoryWarning:(UIApplication \*)application

说明：iPhone设备只有有限的内存，如果为应用程序分配了太多内存操作系统会终止应用程序的运行，在终止前会执行这个方法，通常可以在这里进行内存清理工作防止程序被终止

7、- (void)applicationSignificantTimeChange:(UIApplication\*)application

说明：当系统时间发生改变时执行

8、- (void)applicationDidFinishLaunching:(UIApplication\*)application

说明：当程序载入后执行

9、- (void)application:(UIApplication)application willChangeStatusBarFrame:(CGRect)newStatusBarFrame

说明：当StatusBar框将要变化时执行

10、- (void)application:(UIApplication\*)application willChangeStatusBarOrientation:

(UIInterfaceOrientation)newStatusBarOrientation

duration:(NSTimeInterval)duration

说明：当StatusBar框方向将要变化时执行

11、- (BOOL)application:(UIApplication\*)application handleOpenURL:(NSURL\*)url

说明：当通过url执行

12、- (void)application:(UIApplication\*)application didChangeStatusBarOrientation:(UIInterfaceOrientation)oldStatusBarOrientation

说明：当StatusBar框方向变化完成后执行

13、- (void)application:(UIApplication\*)application didChangeSetStatusBarFrame:(CGRect)oldStatusBarFrame

说明：当StatusBar框变化完成后执行

3,UIWindow

UIWindow是所有视图的根，每个程序里只有一个UIWindow实例。它定义了管理并协调窗口在屏幕上显示的对象。它的两个主要功能是分发事件给UIView对象和提供显示视图的区域。UIView通常充满整个屏幕,示例代码如下:

self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];

4,IOS应用程序的状态

IOS系统的资源是有限的，应用程序在前台和在后台的状态是不一样的。在后台时，程序会受到系统的很多限制，这样可以提高电池的使用和用户体验。

//开发app,我们要遵循apple公司的一些指导原则，原则如下：

应用程序的状态

Not running 未运行 程序没启动

Inactive 未激活 程序在前台运行，不过没有接收到事件。在没有事件处理情况下程序通常停留在这个状态

Active 激活 程序在前台运行而且接收到了事件。这也是前台的一个正常的模式

Backgroud 后台 程序在后台而且能执行代码，大多数程序进入这个状态后会在在这个状态上停留一会。时间到之后会进入挂起状态(Suspended)。有的程序经过特殊的请求后可以长期处于Backgroud状态

Suspended 挂起 程序在后台不能执行代码。系统会自动把程序变成这个状态而且不会发出通知。当挂起时，程序还是停留在内存中的，当系统内存低时，系统就把挂起的程序清除掉，为前台程序提供更多的内存。

各个程序运行状态时代理的回调：

- (BOOL)application:(UIApplication \*)application willFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

告诉代理进程启动但还没进入状态保存

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

告诉代理启动基本完成程序准备开始运行

- (void)applicationWillResignActive:(UIApplication \*)application

当应用程序将要入非活动状态执行，在此期间，应用程序不接收消息或事件，比如来电话了

- (void)applicationDidBecomeActive:(UIApplication \*)application

当应用程序入活动状态执行，这个刚好跟上面那个方法相反

- (void)applicationDidEnterBackground:(UIApplication \*)application

当程序被推送到后台的时候调用。所以要设置后台继续运行，则在这个函数里面设置即可

- (void)applicationWillEnterForeground:(UIApplication \*)application

当程序从后台将要重新回到前台时候调用，这个刚好跟上面的那个方法相反。

- (void)applicationWillTerminate:(UIApplication \*)application

当程序将要退出是被调用，通常是用来保存数据和一些退出前的清理工作。这个需要要设置UIApplicationExitsOnSuspend的键值。

- (void)applicationDidFinishLaunching:(UIApplication\*)application

当程序载入后执行

在上面8个方法对应的方法中键入NSLog打印。

现在启动程序看看执行的顺序：

启动程序

willFinishLaunchingWithOptions

didFinishLaunchingWithOptions

applicationDidBecomeActive

按下home键

applicationWillResignActive

applicationDidEnterBackground

双击home键，再打开程序

applicationWillEnterForeground

applicationDidBecomeActive

1、调用 自带mail

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"mailto://admin@hzlzh.com"]];

2、调用 电话phone

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"tel://8008808888"]];

iOS应用内拨打电话结束后返回应用

一般在应用中拨打电话的方式是：

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"tel://123456789"]];

使用这种方式拨打电话时，当用户结束通话后，iphone界面会停留在电话界面。

用如下方式，可以使得用户结束通话后自动返回到应用：

UIWebView\*callWebview =[[UIWebView alloc] init];

NSURL \*telURL =[NSURL URLWithString:@"tel:10086"];// 貌似tel:// 或者 tel: 都行

[callWebview loadRequest:[NSURLRequest requestWithURL:telURL]];

//记得添加到view上

[self.view addSubview:callWebview];

　还有一种私有方法：（可能不能通过审核）

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"telprompt://10086"]];

3、调用 SMS

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"sms://800888"]];

4、调用自带 浏览器 safari

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"http://www.hzlzh.com"]];

启动得到界面状态栏高度的方法

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

{

self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];

// Override point for customization after application launch.

self.window.backgroundColor = [UIColor whiteColor];

/////////////////////////////////////

//为 XYZAppView.xib 类创建代理关联

self.myview = [[XYZAppViewController alloc]initWithNibName:@"XYZAppView" bundle:nil];

UIView \*myvv = self.myview.view;

CGRect viewFrame = myvv.frame;

float a = [UIApplication sharedApplication].statusBarFrame.size.height;

NSLog(@"%f",a);

NSLog(@"%f",viewFrame.origin.y);

viewFrame.origin.y += [UIApplication sharedApplication].statusBarFrame.size.height;

myvv.frame = viewFrame;

NSLog(@"viewFrame :%d",viewFrame);

[self.window addSubview:myview.view];

/////////////////////////////////////

//在屏幕上显示窗口

[self.window makeKeyAndVisible];

return YES;

}

启动判断当前何种设备，根据设备读取 RootView

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

{

self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];

// Override point for customization after application launch.

//判断当前是何种设备，自动选择 iPhone 或 iPad 对应的 xib 文件

if ([[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPhone) {

self.viewController = [[[ViewController alloc] initWithNibName:@"ViewController\_iPhone" bundle:nil] autorelease];

} else {

self.viewController = [[[ViewController alloc] initWithNibName:@"ViewController\_iPad" bundle:nil] autorelease];

}

self.window.rootViewController = self.viewController;

[self.window makeKeyAndVisible];

return YES;

}