UIApplication的核心作用是提供了iOS程序运行期间的控制和协作工作。

iPhone应用程序是由主函数main启动，它负责调用UIApplicationMain函数，该函数的形式如下所示：  int UIApplicationMain (  int argc,  char \*argv[],  NSString \*principalClassName,  NSString \*delegateClassName  );  那么UIApplicationMain函数到底做了哪些事情呢？这个函数主要负责三件 事情：

 1）从给定的类名初始化应用程序对象，也就是初始化UIApplication或者子类对象的一个实例，如果你在这里给定的是nil，那么 系统会默认UIApplication类，也就主要是这个类来控制以及协调应用程序的运行。在后续的工作中，你可以用静态方法sharedApplication 来获取应用程序的句柄。

2）从给定的应用程序委托类，初始化一个应用程序委托。并把该委托设置为应用程序的委托，这里就有如果传入参数为nil，会调用函数访问 Info.plist文件来寻找主nib文件，获取应用程序委托。

3）启动主事件循环，并开始接收事件。

上面是UIApplicationMain函数的工作，接下来一个问题是应用程序视图的显示、消息的控制怎么办？下面就是UIApplication（或 者子类）对象的职责，这个对象主要做下面几件事：

1）负责处理到来的用户事件，并分发事件消息到应该处理该消息的目标对象（sender,  action)。

2）管理以及控制视图，包括呈现、控制行为、当前显示视图等。

3）该对象有一个应用程序委托对象，当一些生命周期内重要事件（可以包括系统事件或者生命周期控制事件）发生时，应用程序通知该对象。例如，应用程序启 动、内存不够了或者应用程序结束等，让这些事件发生时，应用程序委托去响应。

通 过上面的分析，可以知道UIApplication对开发者来说，是一个黑箱，因为所有的操作，都可以由它的委托来帮我们完成，它只需要在 后面维护一些不可更改的东西，如事件消息分发和传递、给委托发送事件处理请求等等，如，应用程序加载处理完毕，它会发送消息给委托，然后委托可以在 applicationDidFinishLanching委托函数中去实现开发者想要的动作。利用XCODE在创建应用程序时，会默认实现一个应用程序 委托类。而对于加载的视图，则有视图相关的委托类来处理视图加载过程的生命事件。下面说明委托主要可以办哪些事情：  控制应用程序的行为

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

          应用程序启动完毕。

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

        当由于其它方法打开应用程序（如URL指定或者连接），通知委托启动完毕

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

         通知委托，应用程序将在关闭 退出，请做一些清理工作。

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

        通知委托，应用程序收到了为来自系统的内存不足警告。-(void)applicationSignificantTimeChange:(UIApplication \*)application

      通知委托系统时间发生改变（主要是指时间属性，而不是具体的时间值）

打开URL

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

           打开指定的URL

控制状态栏方位变化

– application:willChangeStatusBarOrientation:duration:

        设备方向将要发生改变

– application:didChangeStatusBarOrientation:

活动状态改变

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

   通知委托应用程序将进入非活动状态，在此期间，应用程序不接收消息或事件。-(void)applicationDidBecomeActive:(UIApplication \*)application

      通知委托应用程序进入活动状态，请恢复数据

可以看到UIApplication的头文件实现

@interface UIApplication :UIResponder {

@package

id \_delegate ;  //这就是应用程序委托。

NSTimer .......

}

因此，在UIApplication中处理的系统事件时，只需转到\_delegate这个类去处理， 这个类对象就是应用程序委托对象。我们可以从应用程序的单例类对象中得到应用程序委托的对象

UIApplicationDelegate\* myDelegate = [[UIApplication sharedApplication] delegate];

UIApplication 接收到所有的系统事件和生命周期事件时，都会把事件传递给UIApplicationDelegate进行处理，对于用户输入事件，则传递给相应的目标对象去处理。比如我们在应用程序被来电等消息后，可以调用应用程序委托类的 applicationWillResignActive（）方法，这个方法在用户锁住屏幕时，也会调用，与之相适应的是应用程序重新被用户打开时的委托 方法。另外常用的就是内存不足的系统警告，此时会调用应用程序委托类的applicationDidReceiveMemoryWarning()方法， 然后我们就可以试着释放一些内存了。