# 『RunLoop』

1. iOS里面有两套API可以访问和使用RunLoop：

Foundation: NSRunLoop --- 对CFRunLoopRef封装，面向对象

Core Foundation: CFRunLoopRef --- c,开源，更加底层

#### 2.RunLoop与线程：

#### 每条线程都有唯一的与之对应的RunLoop对象；

#### 主线程的RunLoop已经创建好了，而子线程的需要手动创建。（也就是说子线程的RunLoop默认是关闭的，因为有时候开了个线程但却没有必要开一个RunLoop，不然反而浪费了资源。 ）

RunLoop在第一次获取时创建，在线程结束时销毁。（这就相当于 线程是一个类，RunLoop是类里的实例变量，这样便于理解）

3. 获取RunLoop对象

Foundation：

[NSRunLoop currentRunLoop]; // 获得当前线程的RunLoop对象

[NSRunLoop mainRunLoop]; // 获得主线程的RunLoop对象

Core Foundation：

CFRunLoopGetCurrent(); // 获得当前线程的RunLoop对象

CFRunLoopGetMain(); // 获得主线程的RunLoop对象

4. RunLoop 相关类

UIApplicationMain这个函数的内部就已经启动了一个RunLoop，所以函数一直没有返回，这才使得程序保持运行。

NSTimer的使用：

1.[NSTimer scheduledTimerWithTimeInterval:1.0 target:self selector:@selector(timerTest) userInfo:nil repeats:YES];

2. NSTimer \*timer = [NSTimer timerWithTimeInterval:1.0 target:self selector:@selector(timerTest) userInfo:nil repeats:YES];

[[NSRunLoop currentRunLoop] addTimer:timer forMode:NSDefaultRunLoopMode];

注意：2比1要多写一句代码，那是因为scheduledTimer方法会自动添加到当前的runloop里面去，而且runloop的运行模式kCFRunLoopDefaultMode，

有scrollView的情况下使用Timer：

NSTimer \*timer = [NSTimer timerWithTimeInterval:1.0 target:self selector:@selector(timerTest) userInfo:nil repeats:YES];

[[NSRunLoop currentRunLoop] addTimer:timer forMode:NSRunLoopCommonModes];