RunLoop知识点总结

#iOS笔记/复习笔记

RunLoop基本作用

- 保持程序持续运行；
- 处理App各种事件；
- 节省CPU资源，提高程序性能；

RunLoop和线程的关系

- 每条线程都有唯一的一个与之对应的RunLoop对象；
- RunLoop保存在一个全局的Dictionary里，线程作为key,RunLoop作为value；
- 主线程的RunLoop已经自动创建好了，子线程的RunLoop需要主动创建；
- RunLoop在第一次获取时创建，在线程结束时销毁；

RunLoop的mode（运行模式）

一个 RunLoop 包含若干个 Mode，每个Mode又包含若干个Source、Timer、Observer
每次RunLoop启动时，只能指定其中一个 Mode，这个Mode被称作 CurrentMode
如果需要切换Mode，只能退出RunLoop，再重新指定一个Mode进入，这样做主要是为了分隔开不同组的Source、Timer、Observer，让其互不影响。如果Mode里没有任何Source0/Source1/Timer/Observer，RunLoop会立马退出

系统默认注册的5个Mode:

- kCFRunLoopDefaultMode：App的默认Mode，通常主线程是在这个Mode下运行
- UITrackingRunLoopMode：界面跟踪 Mode，用于 ScrollView 追踪触摸滑动，保证界面滑动时不受其他 Mode 影响
- UIInitializationRunLoopMode: 在刚启动 App 时第进入的第一个 Mode，启动完成后就不再使用，会切换到kCFRunLoopDefaultMode
- GSEventReceiveRunLoopMode: 接受系统事件的内部 Mode，通常用不到
- kCFRunLoopCommonModes: 这是一个占位用的Mode，作为标记kCFRunLoopDefaultMode和UITrackingRunLoopMode用，并不是一种真正的Mode

Source1/Source0/Timers/Observer分别代表什么

- Source1 : 基于Port的线程间通信
- Source0 : 触摸事件，PerformSelectors
- Timers : 定时器，NSTimer
- Observer : 监听器，用于监听RunLoop的状态

RunLoop的的运行状态

- kCFRunLoopEntry： 进入RunLoop
- kCFRunLoopBeforeTimers: 即将处理timers
- kCFRunLoopBeforeSources: 即将处理Sources
- kCFRunLoopBeforeWaiting: 即将休息
- kCFRunLoopAfterWaiting:即将被唤醒
- kCFRunLoopExit: 退出RunLoop

RunLoop处理逻辑流程图