**RunLoop**

什么是RunLoop，顾名思义，RunLoop就是在‘跑圈’，其本质是一个do while循环。RunLoop提供了这么一种机制，当有任务处理时，线程的RunLoop会保持忙碌，而在没有任何任务处理时，会让线程休眠，从而让出CPU。当再次有任务需要处理时，RunLoop会被唤醒，来处理事件，直到任务处理完毕，再次进入休眠。

**#pragma mark-- RunLoop应用场景(一):RunLoop保证线程的长时间存活**

/\*

1,ios当子线程中的任务执行完毕后，子线程就会被销毁掉。

2,如果程序中，需要经常在子线程中执行任务，频繁的创建和销毁线程，会造成资源的浪费。这时候我们就可以使用RunLoop来让该线程长时间存活而不被销毁

3,每个线程都会有自己的一个runLoop;

// 注意!!!!

1.获取RunLoop只能使用 [NSRunLoop currentRunLoop] 或 [NSRunLoop mainRunLoop];

2.即使RunLoop开始运行，如果RunLoop 中的 modes 为空，或者要执行的mode里没有item，那么RunLoop会直接在当前loop中返回，并进入睡眠状态。

3.自己创建的Thread中的任务是在kCFRunLoopDefaultMode这个mode中执行的。

4.在子线程创建好后，最好所有的任务都放在AutoreleasePool中。

\*/

**#pragma mark --- RunLoop应用场景(二):RunLoop 让Timer正常运转**

/\*

然后，我们在滑动tableView的时候timerUpdate方法，并不会调用。

\* 原因是啥呢？\*

原因是当我们滑动scrollView时，主线程的RunLoop 会切换到UITrackingRunLoopMode这个Mode，执行的也是UITrackingRunLoopMode下的任务（Mode中的item），而timer 是添加在NSDefaultRunLoopMode下的，所以timer任务并不会执行，只有当UITrackingRunLoopMode的任务执行完毕，runloop切换到NSDefaultRunLoopMode后，才会继续执行timer。

\* 要如何解决这一问题呢？\*

解决方法很简单，我们只需要在添加timer 时，将mode 设置为NSRunLoopCommonModes即可。

添加到NSRunLoopCommonModes中的还没有执行的任务，会在mode切换时，再次添加到当前的mode中，这样就能保证不管当前runloop切换到哪一个mode，任务都能正常执行

并且被添加到NSRunLoopCommonModes中的任务会存储在runloop 的commonModeItems中。

这就是多线程与runloop的关系了，每一个线程都有一个与之关联的RunLoop，而每一个RunLoop可能会有多个Mode。CPU会在多个线程间切换来执行任务，呈现出多个线程同时执行的效果。执行的任务其实就是RunLoop去各个Mode里执行各个item。因为RunLoop是独立的两个，相互不会影响，所以在子线程添加timer，滑动视图时，timer能正常运行。

1、如果是在主线程中运行timer，想要timer在某界面有视图滚动时，依然能正常运转，那么将timer添加到RunLoop中时，就需要设置mode 为NSRunLoopCommonModes。

2、如果是在子线程中运行timer,那么将timer添加到RunLoop中后，Mode设置为NSDefaultRunLoopMode或NSRunLoopCommonModes均可，但是需要保证RunLoop在运行，且其中有任务。

\*/

**#pragma mark-- RunLoop应用场景(三):当RunLoop处于NSDefaultRunLoopMode模式下kCFRunLoopBeforeWaiting渲染页面;**

/\*

//让UITableView、UICollectionView等延迟加载图片

// 当RunLoop处于defalut模式时,发生回掉, 执行[YYRunLoopImageVC task\_2:cell indexPath:indexPath];

进行图片加载和页面渲染;

一次RunLoop循环:BeforeWaiting->BeforeSource->BeforeWaiting

利用RunLoop进行延迟加载图片本质上是利用主线程在页面滚动的时候 RunLoop的模式切换到Traking模式,而这个时候我们设置图片加载的模式是defalut模式下的BeforeWaiting状态,所以在滚动的时候不会加载图片;

[imageView3 performSelectorOnMainThread:@selector(setImage:) withObject:image3 waitUntilDone:NO modes:@[NSDefaultRunLoopMode]];

\*/

**#pragma mark-- RunLoop应用场景(四):用分线程来监控主线程的耗时操作;**

/\*

主要测试从BeforeSource到BeforeWaiting之间的时间;

\*/

#pragma mark-- RunLoop应用场景(五): RunLoop Crash处理

/\*

我们在HandleException回调函数中，可以获取到当前的RunLoop，然后获取该RunLoop中的所有Mode，手动运行一遍;

CFRunLoopRef runLoop = CFRunLoopGetCurrent();

CFArrayRef allModes = CFRunLoopCopyAllModes(runLoop);

while (!ignore) {

for (NSString \*mode in (\_\_bridge NSArray \*)allModes) {

CFRunLoopRunInMode((CFStringRef)mode, 0.001, false);

}

NSLog(@"ignoreignoreignoreignoreignore");

}

\*/