NSTimer

fire

我们先用 NSTimer 来做个简单的计时器，每隔5秒钟在控制台输出 Fire 。比较想当然的做法是这样的：

```
@interface DetailViewController ()
@property (nonatomic, weak) NSTimer *timer;
@end
@implementation DetailViewController
- (IBAction)fireButtonPressed:(id)sender {
    _timer = [NSTimer scheduledTimerWithTimeInterval:3.0f
                                              target:self
                                            selector:@selector(timerFire:)
                                            userInfo:nil
                                             repeats:YES];
    [_timer fire];
}
-(void)timerFire:(id)userinfo {
    NSLog(@"Fire");
}
@end
```

运行之后确实在控制台每隔3秒钟输出一次 Fire ，然而当我们从这个界面跳转到其他界面的时候却发现：控制台还在源源不断的输出着 Fire 。看来 Timer 并没有停止。

invalidate

既然没有停止，那我们在 DemoViewController 的 dealloc 里加上 invalidate 的方法：

```
-(void)dealloc {
    [_timer invalidate];
    NSLog(@"%@ dealloc", NSStringFromClass([self class]));
}
```

再次运行，还是没有停止。原因是 Timer 添加到 Runloop 的时候，会被 Runloop 强引用：

```
Note in particular that run loops maintain strong references to their timers, so you dor
```

然后 Timer 又会有一个对 Target 的强引用（也就是 self ）：

```
Target is the object to which to send the message specified by aSelector when the timer
```

也就是说 NSTimer 强引用了 self ，导致 self 一直不能被释放掉，所以也就走不到 self 的 dealloc 里。

既然如此，那我们可以再加个 invalidate 按钮：

```
- (IBAction)invalidateButtonPressed:(id)sender {
    [_timer invalidate];
}
```

嗯这样就可以了。（在 SOF 上有人说该在 invalidate 之后执行 _timer = nil ，未能理解为什么，如果你知道原因可以告诉我：）

在 invalidate 方法的文档里还有这这样一段话：

You must send this message from the thread on which the timer was installed. If you send this message from another thread, the input source associated with the timer may not be removed from its run loop, which could prevent the thread from exiting properly.

NSTimer 在哪个线程创建就要在哪个线程停止，否则会导致资源不能被正确的释放。看起来各种坑还不少。

dealloc

那么问题来了：如果我就是想让这个 NSTimer 一直输出，直到 DemoViewController 销毁了才停止，我该如何让它停止呢?

- NSTimer 被 Runloop 强引用了，如果要释放就要调用 invalidate 方法。
- 但是我想在 DemoViewController 的 dealloc 里调用 invalidate 方法，但是 self 被 NSTimer 强引用了。
- 所以我还是要释放 NSTimer 先，然而不调用 invalidate 方法就不能释放它。
- 然而你不进入到 dealloc 方法里我又不能调用 invalidate 方法。
- 嗯...

HWWeakTimer

weakSelf

问题的关键就在于 self 被 NSTimer 强引用了，如果我们能打破这个强引用问题自然而然就解决了。所以一个很简单的想法就是：weakSelf：

```
__weak typeof(self) weakSelf = self;
_timer = [NSTimer scheduledTimerWithTimeInterval:3.0f
                                          target:weakSelf
                                        selector:@selector(timerFire:)
                                        userInfo:nil
                                         repeats:YES];
```

然而这并没有什么卵用，这里的 __weak 和 __strong 唯一的区别就是：如果在这两行代码执行的期间 self 被释放了， NSTimer 的 target 会变成 nil 。

target

既然没办法通过 __weak 把 self 抽离出来，我们可以造个假的 target 给 NSTimer 。这个假的 target 类似于一个中间的代理人，它做的唯一的工作就是挺身而出接下了 NSTimer 的强引用。类声明如下：

```
@interface HWWeakTimerTarget : NSObject
@property (nonatomic, weak) id target;
@property (nonatomic, assign) SEL selector;
@property (nonatomic, weak) NSTimer* timer;
@end
@implementation HWWeakTimerTarget
```

```
- (void) fire:(NSTimer *)timer {
    if(self.target) {
        [self.target performSelector:self.selector withObject:timer.userInfo];
    } else {
        [self.timer invalidate];
    }
}
@end
```

然后我们再封装个假的 scheduledTimerWithTimeInterval 方法，但是在调用的时候已经偷梁换柱了：

```
+ (NSTimer *) scheduledTimerWithTimeInterval:(NSTimeInterval)interval
                                      target:(id)aTarget
                                    selector:(SEL)aSelector
                                    userInfo:(id)userInfo
                                     repeats:(BOOL)repeats {
    HWWeakTimerTarget* timerTarget = [[HWWeakTimerTarget alloc] init];
    timerTarget.target = aTarget;
    timerTarget.selector = aSelector;
    timerTarget.timer = [NSTimer scheduledTimerWithTimeInterval:interval
                                                         target:timerTarget
                                                       selector:@selector(fire:)
                                                       userInfo:userInfo
                                                        repeats:repeats];
    return timerTarget.timer;
}
```

再次运行，问题解决。

block

如果能用 block 来调用 NSTimer 那岂不是更好了。我们可以这样来实现：

```
+ (NSTimer *)scheduledTimerWithTimeInterval:(NSTimeInterval)interval
                                      block:(HWTimerHandler)block
                                   userInfo:(id)userInfo
                                    repeats:(BOOL)repeats {
    return [self scheduledTimerWithTimeInterval:interval
                                         target:self
                                       selector:@selector(_timerBlockInvoke:)
                                       userInfo:@[[block copy], userInfo]
                                        repeats:repeats];
}
+ (void)_timerBlockInvoke:(NSArray*)userInfo {
    HWTimerHandler block = userInfo[0];
    id info = userInfo[1];
    // or `!block ?: block();` @sunnyxx
    if (block) {
        block(info);
    }
}
```

这样我们就可以直接在 block 里写相关逻辑了：

```
- (IBAction)fireButtonPressed:(id)sender {
    _timer = [HWWeakTimer scheduledTimerWithTimeInterval:3.0f block:^(id userInfo) {
        NSLog(@"%@", userInfo);
    } userInfo:@"Fire" repeats:YES];
    [_timer fire];
}
```

嗯就是这样。