

;J\_ToDo\_Subclass3.rtfld;linkMarkername ;↪ Previous Section  
;L\_ToDo\_BuildRunExtend.rtfld;linkMarkername ;↪ Next Section

## 4. To Do Tutorial

# Setting Up Timers for Notification Messages

The To Do application includes as a feature the capability for notifying users of items with impending due times. Users can specify various intervals before the due time for these notifications, which take the form of a message in an attention panel. In this section you will implement the notification feature of To Do. In the process you'll learn the basics of creating, setting, and responding to timers.

Here's how it works: Each `ToDoItem` with a `When to Notify` switch (other than `Do not notify`) selected in the inspector panel—and hence has a positive `secsUntilNotif` value—has a timer set for it. If a user cancels a notification by selecting `Do not notify`, the document controller invalidates the timer. When a timer fires, it invokes a method that displays the attention panel, selects the `Do not notify` switch, and sets `secsUntilNotif` to zero.

Implementing the timer feature takes place entirely in Project Builder, but extends across several classes.

### 1 Add the timer as an instance variable to `ToDoItem`.

Open `ToDoItem.h`.

Add the instance variable `itemTimer` of class `NSTimer`.

Write accessor methods to get and set this instance variable.

### 2 Create and set the timer, or invalidate it.

Open `ToDoDoc.m`.

Implement the **setTimerForItem:** method, which is shown below.

```
- (void) setTimerForItem: (ToDoItem *) anItem
{
    NSDate *notifDate;
    NSTimer *aTimer;
    if ([anItem secsUntilNotif]) { /* 1 */
        notifDate = [[anItem day] addTimeInterval:[anItem
            secsUntilNotif]];
        aTimer = [NSTimer scheduledTimerWithTimeInterval: /* 2 */
            [notifDate timeIntervalSinceNow]
            target:self
            selector:@selector(itemTimerFired:)
            userInfo:anItem
            repeats:NO];
        [anItem setItemTimer:aTimer];
    } else
        [[anItem itemTimer] invalidate]; /* 3 */
}
```

This method sets or invalidates a timer, depending on whether the `ToDoItem` passed in has a positive **secsUntilNotif** value.

1. Tests the `ToDoItem` to see if it has a positive **secsUntilNotif** value and, if it has, composes the time the notification should be sent.
2. Creates a timer and schedules it to fire at the notification time, and instructs it to invoke **itemTimerFired:** when it fires. It also sets the timer in the `ToDoItem`.

3. If the **secsUntilNotif** variable is zero, invalidates the item's timer.

### 3 Respond to timers firing.

Implement **itemTimerFired:** as shown at right.

```
- (void)itemTimerFired:(id)timer
{
    id anItem = [timer userInfo];
    ToDoInspector *inspController = [[[NSApp delegate] /* 1 */
        inspector] delegate];
    NSDate *dueDate = [[anItem day] addTimeInterval: /* 2 */
        [anItem secsUntilDue]];
    NSBeep();
    NSRunAlertPanel(@"To Do", @"%@ on %@", nil, nil, nil,
        [anItem itemName], [dueDate
        descriptionWithCalendarFormat:@"%b %d, %Y at %I:%M %p"
        timeZone:[NSTimeZone defaultTimeZone] locale:nil]);
    [anItem setSecsUntilNotif:0];
    [inspController resetNotifSwitch];
}
```

When a **ToDoItem**'s timer goes off, it invokes the **itemTimerFired:** method (remember, you designated this method when you scheduled the timer).

1. This method communicates with **ToDoInspector** in a more direct manner than notification. It gets the **ToDoInspector** object through this chain of association: the delegate of the application object is **ToDoController**, which holds the **id** of the inspector panel as an instance variable, and the delegate of

the inspector panel is `ToDoInspector`.

2. Composes the notification time (as an `NSDate`), beeps, and displays an attention panel specifying the name of a `ToDoItem` and the time it is due. It then sets the `ToDoItem`'s `secsUntilNotif` instance variable to zero, and sends `resetNotifSwitch` to `ToDoInspector` to have it reset the <sup>a</sup>When to Notify<sup>o</sup> switches to <sup>a</sup>Do not Notify.<sup>o</sup>

### ***TableRule.eps –Before You Go On***

**Implement `resetNotifSwitch`:** You haven't written `ToDoInspector`'s `resetNotifSwitch` method yet, so do it now as an exercise. It should select the <sup>a</sup>Do not Notify<sup>o</sup> switch after turning off all switches in the matrix, and then force a redisplay of the switch matrix.

758816\_TableRule.eps –

Next you must send `setTimerForItem:` at the right place and time, which is `ToDoInspector`, when the user alters a <sup>a</sup>When to Notify<sup>o</sup> value.

## **4 Send the message that sets the timer at the right times**

Open `ToDoInspector.m`.

In `switchChecked:`, insert the `setTimerForItem:` message below *after* the switch statement evaluating which <sup>a</sup>When to Notify<sup>o</sup> switch was checked.

In `controlTextDidEndEditing:`, insert the same message at the end of the block related to the `inspNotifOtherHours` variable.

```
[[[NSApp mainWindow] delegate] setTimerForItem:currentItem];
```

Instead of archiving an item's `NSTimer`, To Do re-creates and resets it when the application is launched.

## **5 When the application is launched, reset item timers.**

Add the code below to **ToDoDoc's initWithFile:** method.

```
if ([self activeDays]) {
    dayenum = [[self activeDays] keyEnumerator];
    while (itemDate = [dayenum nextObject]) {
        NSEnumerator *itemenum;
        ToDoItem *anItem=nil;
        NSArray *itemArray = [[self activeDays]
            objectForKey:itemDate];
        itemenum = [itemArray objectEnumerator];
        while ((anItem = [itemenum nextObject]) &&
            [anItem isKindOfClass:[ToDoItem class]] &&
            [anItem secsUntilNotif]) {
            [self setTimerForItem:anItem];
        }
    }
}
```

This block of code traverses the **activeDays** dictionary, evaluating each **ToDoItem** within the dictionary. If the **ToDoItem** has a positive **secsUntilNotif** value, it invokes **setTimerForItem:** to have a timer set for it.

**Related Concept:** ;ToDoConcepts.rtf;linkMarkername TickTockBrrring:RunLoopsandTimers;; Tick Tock Brrring: Run Loops and Timers