NSTimer Class Reference

Cocoa > Events & Other Input

2008-11-19

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSTimer Class Reference 5

Overview 5
Tasks 6
- Creating a Timer 6
- Firing a Timer 6
- Stopping a Timer 7
- Information About a Timer 7

Class Methods 7
- scheduledTimerWithTimeInterval:invocation:repeats: 7
- scheduledTimerWithTimeInterval:target:selector:userInfo:repeats: 8
- timerWithTimeInterval:invocation:repeats: 9
- timerWithTimeInterval:target:selector:userInfo:repeats: 9

Instance Methods 10
- fire 10
- fireDate 11
- initWithFireDate:interval:target:selector:userInfo:repeats: 11
- invalidate 12
- isValid 13
- setFireDate: 13
- timeInterval 13
- userInfo 13

Document Revision History 15

Index 17

NSTimer Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTimer.h
Companion guides	Timer Programming Topics for Cocoa Run Loops
Related sample code	FunkyOverlayWindow ImageClient NSOperationSample ThreadsImporter ThreadsImportMovie

Overview

`NSTimer` creates timer objects or, more simply, timers. A timer waits until a certain time interval has elapsed and then fires, sending a specified message to a specified object. For example, you could create an `NSTimer` object that sends a message to a window, telling it to update itself after a certain time interval.

Timers work in conjunction with run loops. To use a timer effectively, you should be aware of how run loops operate—see `NSRunLoop` and *Run Loops*. Note in particular that run loops retain their timers, so you can release a timer after you have added it to a run loop.

A timer is not a real-time mechanism; it fires only when one of the run loop modes to which the timer has been added is running and able to check if the timer's firing time has passed. Because of the various input sources a typical run loop manages, the effective resolution of the time interval for a timer is limited to on the order of 50-100 milliseconds. If a timer's firing time occurs while the run loop is in a mode that is not monitoring the timer or during a long callout, the timer does not fire until the next time the run loop checks the timer. Therefore, the actual time at which the timer fires potentially can be a significant period of time after the scheduled firing time.

A repeating timer reschedules itself based on the scheduled firing time, not the actual firing time. For example, if a timer is scheduled to fire at a particular time and every 5 seconds after that, the scheduled firing time will always fall on the original 5 second time intervals, even if the actual firing time gets delayed. If the firing time is delayed so far that it passes one or more of the scheduled firing times, the timer is fired only once for that time period; the timer is then rescheduled, after firing, for the next scheduled firing time in the future.

Each run loop timer can be registered in only one run loop at a time, although it can be added to multiple run loop modes within that run loop.

There are three ways to create a timer. The `scheduledTimerWithTimeInterval:invocation:repeats:` (page 7) and `scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:` (page 8) class methods automatically add the new timer to the current `NSRunLoop` object in the default mode (`NSDefaultRunLoopMode`). The `timerWithTimeInterval:invocation:repeats:` (page 9) and `timerWithTimeInterval:target:selector:userInfo:repeats:` (page 9) class methods create timers that you may add to a run loop at a later time by sending the message `addTimer:forMode:` to the `NSRunLoop` object. Finally, you can allocate the timer directly and initialize it with `initWithFireDate:interval:target:selector:userInfo:repeats:` (page 11), which allows you to specify both an initial fire date and a repeating interval. If you specify that the timer should repeat, it automatically reschedules itself after it fires. If you specify that the timer should not repeat, it is automatically invalidated after it fires.

To request the removal of a timer from an `NSRunLoop` object, send the timer the `invalidate` (page 12) message from the same thread on which the timer was installed. This message immediately disables the timer, so it no longer affects the `NSRunLoop` object. The run loop removes and releases the timer, either just before the `invalidate` (page 12) method returns or at some later point.

`NSTimer` is "toll-free bridged" with its Core Foundation counterpart, *CFRunLoopTimer Reference*. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSTimer *` parameter, you can pass a `CFRunLoopTimerRef`, and in a function where you see a `CFRunLoopTimerRef` parameter, you can pass an `NSTimer` instance (you cast one type to the other to suppress compiler warnings). See Interchangeable Data Types for more information on toll-free bridging.

Tasks

Creating a Timer

`+ scheduledTimerWithTimeInterval:invocation:repeats:` (page 7)
Returns a new `NSTimer` object, scheduled with the current `NSRunLoop` object in the default mode.

`+ scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:` (page 8)
Returns a new `NSTimer` object, scheduled the current `NSRunLoop` object in the default mode.

`+ timerWithTimeInterval:invocation:repeats:` (page 9)
Returns a new `NSTimer` that, when added to a run loop, will fire after a given number of seconds.

`+ timerWithTimeInterval:target:selector:userInfo:repeats:` (page 9)
Returns a new `NSTimer` that, when added to a run loop, will fire after a specified number of seconds.

`- initWithFireDate:interval:target:selector:userInfo:repeats:` (page 11)
Initializes a new `NSTimer` that, when added to a run loop, will fire at a given date.

Firing a Timer

`- fire` (page 10)
Causes the receiver's message to be sent to its target.

Stopping a Timer

- `invalidate` (page 12)
  Stops the receiver from ever firing again and requests its removal from its `NSRunLoop` object.

Information About a Timer

- `isValid` (page 13)
  Returns a Boolean value that indicates whether the receiver is currently valid.
- `fireDate` (page 11)
  Returns the date at which the receiver will fire.
- `setFireDate:` (page 13)
  Resets the receiver to fire next at a given date.
- `timeInterval` (page 13)
  Returns the receiver's time interval.
- `userInfo` (page 13)
  Returns the receiver's *`userInfo`* object.

Class Methods

scheduledTimerWithTimeInterval:invocation:repeats:

Returns a new `NSTimer` object, scheduled with the current `NSRunLoop` object in the default mode.

```
+ (NSTimer *)scheduledTimerWithTimeInterval:(NSTimeInterval)seconds
    invocation:(NSInvocation *)invocation  repeats:(BOOL)repeats
```

Parameters

`seconds`

The number of seconds between firings of the timer.

If *`seconds`* is less than or equal to `0.0`, this method chooses a nonnegative interval.

`invocation`

The invocation to use when the timer fires.

The timer instructs the invocation object to retain its arguments.

`repeats`

If `YES`, the timer will repeatedly reschedule itself until invalidated. If `NO`, the timer will be invalidated after it fires.

Return Value
A new `NSTimer` object, configured according to the specified parameters.

Discussion
After *`seconds`* seconds have elapsed, the timer fires, invoking *`invocation`*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTimer.h

scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:

Returns a new NSTimer object, scheduled the current NSRunLoop object in the default mode.

```
+ (NSTimer *)scheduledTimerWithTimeInterval:(NSTimeInterval)seconds
    target:(id)target selector:(SEL)aSelector  userInfo:(id)userInfo
    repeats:(BOOL)repeats
```

Parameters

seconds

The number of seconds between firings of the timer.

If *seconds* is less than or equal to 0.0, this method chooses a nonnegative interval.

target

The object to which to send the message specified by *aSelector* when the timer fires.

aSelector

The message to send to *target* when the timer fires.

The selector must have the following signature:

```
- (void)timerFireMethod:(NSTimer*)theTimer
```

The timer passes itself as the argument to this method.

userInfo

The user info the new timer.

This parameter may be nil.

repeats

If YES, the timer will repeatedly reschedule itself until invalidated. If NO, the timer will be invalidated after it fires.

Return Value
A new NSTimer object, configured according to the specified parameters.

Discussion
After *seconds* seconds have elapsed, the timer fires, sending the message *aSelector* to *target*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Draw Pixels
FunkyOverlayWindow
ImageClient
TextureRange
UIElementInspector

Declared In
NSTimer.h

timerWithTimeInterval:invocation:repeats:

Returns a new `NSTimer` that, when added to a run loop, will fire after a given number of seconds.

```
+ (NSTimer *)timerWithTimeInterval:(NSTimeInterval)seconds  invocation:(NSInvocation
     *)invocation  repeats:(BOOL)repeats
```

Parameters

seconds

The number of seconds between firings of the timer.

If *seconds* is less than or equal to `0.0`, this method chooses a nonnegative interval.

invocation

The invocation to use when the timer fires.

The timer instructs the invocation object to retain its arguments.

repeats

If `YES`, the timer will repeatedly reschedule itself until invalidated. If `NO`, the timer will be invalidated after it fires.

Return Value

A new `NSTimer` object, configured according to the specified parameters.

Discussion

You must add the new timer to a run loop, using `addTimer:forMode:`. Then, after *seconds* have elapsed, the timer fires, invoking *invocation*. (If the timer is configured to repeat, there is no need to subsequently re-add the timer to the run loop.)

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTimer.h`

timerWithTimeInterval:target:selector:userInfo:repeats:

Returns a new `NSTimer` that, when added to a run loop, will fire after a specified number of seconds.

```
+ (NSTimer *)timerWithTimeInterval:(NSTimeInterval)seconds  target:(id)target
    selector:(SEL)aSelector  userInfo:(id)userInfo  repeats:(BOOL)repeats
```

Parameters

seconds

The number of seconds between firings of the timer.

If *seconds* is less than or equal to `0.0`, this method chooses a nonnegative interval.

target

The object to which to send the message specified by *aSelector* when the timer fires.

aSelector

The message to send to *target* when the timer fires.

The selector must have the following signature:

```
- (void)timerFireMethod:(NSTimer*)theTimer
```

The timer passes itself as the argument to this method.

userInfo

The user info for the new timer.

This parameter may be `nil`.

repeats

If `YES`, the timer will repeatedly reschedule itself until invalidated. If `NO`, the timer will be invalidated after it fires.

Return Value
A new `NSTimer` object, configured according to the specified parameters.

Discussion
You must add the new timer to a run loop, using `addTimer:forMode:`. Then, after *seconds* seconds have elapsed, the timer fires, sending the message *aSelector* to *target*. (If the timer is configured to repeat, there is no need to subsequently re-add the timer to the run loop.)

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLChildWindowDemo
OpenGLCompositorLab
Quartz Composer QCTV
Quartz Composer Texture
ThreadsImporter

Declared In
`NSTimer.h`

Instance Methods

fire

Causes the receiver's message to be sent to its target.

```
- (void)fire
```

Discussion
You can use this method to fire a repeating timer without interrupting its regular firing schedule.

If the timer is non-repeating, it is automatically invalidated after firing, even if its scheduled fire date has not arrived.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- invalidate` (page 12)

Declared In
`NSTimer.h`

fireDate

Returns the date at which the receiver will fire.

```
- (NSDate *)fireDate
```

Return Value
The date at which the receiver will fire. If the timer is no longer valid, this method returns the last date at which the timer fired.

Discussion
Use `isValid` (page 13) to verify that the timer is valid.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setFireDate:` (page 13)

Declared In
`NSTimer.h`

initWithFireDate:interval:target:selector:userInfo:repeats:

Initializes a new `NSTimer` that, when added to a run loop, will fire at a given date.

```
- (id)initWithFireDate:(NSDate *)date interval:(NSTimeInterval)seconds
    target:(id)target  selector:(SEL)aSelector  userInfo:(id)userInfo
    repeats:(BOOL)repeats
```

Parameters

date
The time at which the timer should first fire.

seconds
The number of seconds between firings of the timer.

If *seconds* is less than or equal to `0.0`, this method chooses a nonnegative interval.

target
The object to which to send the message specified by *aSelector* when the timer fires.

aSelector
The message to send to *target* when the timer fires.

The selector must have the following signature:

```
- (void)timerFireMethod:(NSTimer*)theTimer
```

The timer passes itself as the argument to this method.

userInfo
The user info for the new timer.

This parameter may be `nil`.

repeats
If `YES`, the timer will repeatedly reschedule itself until invalidated. If `NO`, the timer will be invalidated after it fires.

Return Value
The receiver, initialized such that, when added to a run loop, it will fire at *date* and then, if *repeats* is YES, every *seconds* after that.

Discussion
You must add the new timer to a run loop, using `addTimer:forMode:`. Upon firing, the timer sends the message *aSelector* to *target*. (If the timer is configured to repeat, there is no need to subsequently re-add the timer to the run loop.)

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
WhackedTV

Declared In
NSTimer.h

invalidate

Stops the receiver from ever firing again and requests its removal from its `NSRunLoop` object.

```
- (void)invalidate
```

Discussion
This is the only way to remove a timer from an `NSRunLoop` object. The `NSRunLoop` object removes and releases the timer, either just before the `invalidate` (page 12) method returns or at some later point.

If it was configured with a target and user info, the receiver releases its references to the them at the point of invalidation.

Special Considerations

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.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- fire` (page 10)

Related Sample Code
DockTile
FunkyOverlayWindow
LiveVideoMixer2
NSOperationSample
WhackedTV

Declared In
NSTimer.h

isValid

Returns a Boolean value that indicates whether the receiver is currently valid.

```
- (BOOL)isValid
```

Return Value
`YES` if the receiver is currently valid, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTimer.h`

setFireDate:

Resets the receiver to fire next at a given date.

```
- (void)setFireDate:(NSDate *)date
```

Parameters
date
The date at which to fire the receiver.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `fireDate` (page 11)

Declared In
`NSTimer.h`

timeInterval

Returns the receiver's time interval.

```
- (NSTimeInterval)timeInterval
```

Return Value
The receiver's time interval. If the receiver is a non-repeating timer, returns `0` (even if a time interval was set).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTimer.h`

userInfo

Returns the receiver's *userInfo* object.

```
- (id)userInfo
```

Return Value
The receiver's *userInfo* object.

Discussion
Do not invoke this method after the timer is invalidated. Use `isValid` (page 13) to test whether the timer is valid.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:` (page 8)
`+ timerWithTimeInterval:target:selector:userInfo:repeats:` (page 9)
`- invalidate` (page 12)

Declared In
`NSTimer.h`

Document Revision History

This table describes the changes to *NSTimer Class Reference*.

Date	Notes
2008-11-19	Augmented parameter descriptions.
2006-09-20	Clarified "repeat interval" for non-repeating timers.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index

F

`fire` instance method 10
`fireDate` instance method 11

I

`initWithFireDate:interval:target:selector:`
`userInfo:repeats:` instance method 11
`invalidate` instance method 12
`isValid` instance method 13

S

`scheduledTimerWithTimeInterval:invocation:repeats:`
class method 7
`scheduledTimerWithTimeInterval:target:selector:`
`userInfo:repeats:` class method 8
`setFireDate:` instance method 13

T

`timeInterval` instance method 13
`timerWithTimeInterval:invocation:repeats:`
class method 9
`timerWithTimeInterval:target:selector:userInfo:`
`repeats:` class method 9

U

`userInfo` instance method 13