NSTimer

Inherits From:	NSObject
Conforms To:	NSObject (NSObject)
Declared In:	Foundation/NSTimer.h

Class Description

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 that sends a message to a window, telling it to update itself after a certain time interval.

Timers work in conjunction with NSRunLoop objects. NSRunLoops control loops that wait for input, and they use timers to help determine the maximum amount of time they should wait. When the timer's time limit has elapsed, the NSRunLoop fires the timer (causing its message to be sent), then checks for new input.

There are several ways to create a timer. The **scheduledTimerWithTimeInterval**... class methods automatically register the new timer with the current NSRunLoop object in the default mode (NSDefaultRunLoopMode). The **timerWithTimeInterval**... class methods create timers that you may register at a later time by sending the message **addTimer:forMode:** to the NSRunLoop. If you specify that the timer should repeat, it will automatically reschedule itself after it fires.

There is no method that removes the association of a timer from an NSRunLoop—send the timer the **invalidate** message instead. **invalidate** disables the timer, so it will no longer affect the NSRunLoop.

See the NSRunLoop class description for more information on NSRunLoops.

Method Types

Creating a timer	+ scheduledTimerWithTimeInterval: invocation:repeats: + scheduledTimerWithTimeInterval:target:selector: userInfo:repeats: + timerWithTimeInterval:invocation:repeats: + timerWithTimeInterval:target:selector: userInfo:repeats:
Firing a timer	– fire
Stopping a timer	– invalidate

Getting information about a timer	– isValid – fireDate – timeInterval – userInfo

Class Methods

scheduledTimerWithTimeInterval:invocation:repeats:

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

Returns a new NSTimer object and registers it with the current NSRunLoop in the default mode. After *seconds* have elapsed, the timer fires, sending *invocation*'s message to its target. If seconds is less than or equal to 0.0, this method chooses a nonnegative interval. If *repeats* is YES, the timer will repeatedly reschedule itself.

scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:

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

Returns a new NSTimer object and registers it with the current NSRunLoop in the default mode. After *seconds* have elapsed, the timer fires, sending the message *aSelector* to *target*. The *aSelector* method must take only one argument, an NSTimer object. The timer passes itself as the argument to *aSelector*. To pass more information to the target, use *userInfo*. The target gets *userInfo* by sending **userInfo** to the timer.

If *seconds* is less than or equal to 0.0, this method chooses a nonnegative interval. If *repeats* is YES, the timer will repeatedly reschedule itself.

timerWithTimeInterval:invocation:repeats:

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

Returns a new NSTimer that, when registered, will fire after *seconds*. If *seconds* is less than or equal to 0.0, this method chooses a nonnegative interval. Upon firing, the timer sends *invocation*'s message to its target. If *repeats* is YES, the timer will repeatedly reschedule itself after firing.

timerWithTimeInterval:target:selector:userInfo:repeats:

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

Returns a new NSTimer that, when registered, will fire after *seconds*. Upon firing, the timer sends *aSelector* to *target*. The *aSelector* method must take only one argument, an NSTimer object. The timer passes itself as the argument to *aSelector*. To pass more information to the target, use *userInfo*. The target gets *userInfo* by sending **userInfo** to the timer.

If *seconds* is less than or equal to 0.0, this method chooses a nonnegative interval. If *repeats* is YES, the timer will repeatedly reschedule itself.

nstance Methods

fire

– (void)**fire**

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

fireDate

– (NSDate *)**fireDate**

Returns 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. Use **isValid** to verify that the timer is valid.

See also: – **isValid**

invalidate

– (void)**invalidate**

Stops the receiver from ever firing again. This is the only way to remove a timer from an NSRunLoop.

isValid

– (BOOL)**isValid**

Returns YES if the timer is currently valid, no otherwise.

timeInterval

– (NSTimeInterval)**timeInterval**

Returns the time interval associated with the receiver.

userInfo

– (id)**userInfo**

Additional data the target may use when the receiver is fired.

See also: + **scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:**,
+ **timerWithTimeinterval:target:selector:userInfo:repeats:**