MiscTimedEntry

Inherits From:	MiscThreadedObject
Declared In:	MiscTimedEntry.h

Class Description

This class defines a thread-safe timer that will call back a target object with a selectable message.   The timer may be set up to go off once, repeat a N times, or repeat indefinately. The timer may be run in another thread.

A timed entry blocks, waiting for a timer return, then sends the action message to the target. A timed entry may be run in it's own   thread, providing an asyncronous interupt, or run, providing a thread safe blocking timer.   Of course, when run in it's own thread, the call back will be run in the timer's thread.

Time is measured in milliseconds.   Action methods *must* be of the form " - methodName:sender";   when sent, 'sender' will be the TimedEntry instance.

An example:

```
{
    BOOL async = (random() & 1);
    id te = [[MiscTimedEntry alloc] initWithTarget:self
          action:@selector(someAction:)
```

```
                 interval:5*1000
                 data:somePtr];
if (async)
                 [te runInNewThread];   /* returns immediately with async callbacks */
        else
        [te run];                       /* sender blocked until timer goes off */
}
```

Instance Variables

void ***userdata**;
id **target**;
SEL **action**;
int **interval**;
port_t **port**;
int **repeatCount**;

userdata	A handle to data that may be set to provide a context for the target.
target	The receipient of the action method.
action	Message sent to target after the timer goes off.
interval	The time in milliseconds to sleep before messaging target.
port	Used to sleep.
repeatCount	The number of times to sleep & wake up.

Method Types

- action
- free
- init
- initWithTarget:action:interval:data:
- interval
- port
- repeatCount
- run
- setAction:
- setInterval:
- setPort:
- setRepeatCount:
- setTarget:
- setUserData:
- stop
- target
- userData

Instance Methods

action

- (SEL)**action**

Returns the timer's action.

See also:

free

- **free**

Stops the timer if necessary and frees the instance. This message may be sent from any thread.

See also: stop

init

- **init**

Initializes a timer with the default values by sending *[self initWithTarget:NULL action:(SEL)0 interval:500 data:NULL]*

See also: initWithTarget:action:interval:data:

initWithTarget:action:interval:data:

- **initWithTarget:***aTarget*
  action:(SEL)*anAction*
  interval:(int)*ival*
  data:(void *)*data*

Initializes a timer instance with the given values. RepeatCount is initialized to -1: repeat forever.

See also: setAction:, setInterval, setData:, setTarget:, setRepeatCount:

interval

- (int)**interval**

Returns the sleep interval.

See also: setInterval:

port

- (int)**port**

Returns the mach port that the instance waits on.

repeatCount
- (int)**repeatCount**

Returns the repeat count, the number of times the timer will interupt *target*.   A repeat count of -1 indicates forever.

See also: setRepeatCount

run
- **run**

Starts the timer waiting.

This method causes the caller to block until sleeping *repeatCount* times.   If a non blocking interupt is desired, use runInNewThread.

The run method, coupled with a repeatCount of 1 provides a simple thread-safe alternative to sleep(3).

Returns **nil** if the target object is nil or does not respond to **action**.

See also: runInNewThread, stop.

runInNewThread
- **runInNewThread**

Causes the timer to be run in a separate thread and control returned immediately back to the caller.   The target's action method will be dispatched in the timer's thread.

See also: stop, run, runInNewThread (MiscThreadedObject)

setAction:

- (void)**setAction:**(SEL)*anAction*

Set the action message. ***action*** is the message sent to the target object when the interval has elapsed. **action** has the form of an InterfaceBuilder tm action method, i.e. *- actionMethod:sender*.

If the timer is running in a different thread, the action method will be dispatched from that thread.

setInterval:

- (void)**setInterval:**(int)*anInterval*

The interval, in milliseconds, between messaging target.

setRepeatCount:

- (void)**setRepeatCount:**(int)*count*

The number of times to perform the wait/message cycle. A repeat count of -1 will cause the timer to repeat until a stop message is received.

See also: run, stop

setTarget:

- (void)**setTarget:***sender*

Sets the target object .

See also: setAction:

setUserData:

- (void)**setUserData:**(void *)*userData*

Allows the sender to place data in the timer.   Upon receipt of the *action* message, the target object may retrieve this data using the **userData** method.

See also: userData

stop

- **stop**

Stops the timer but does not free it.

target

- **target**

Returns the receipient object of the action message.

userData

- (void *)**userData**

Returns the data given in the **setUserData**: method.

See also: setUserData: