java.util.Timer类介绍

# java.util.Timer类介绍

java.lang.Object

java.util.Timer

public class **Timer** extends Object

A facility for threads to schedule tasks for future execution **in a background thread**. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each Timer object is a single background thread that is used to execute all of the timer's tasks, sequentially. Timer tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

After the last live reference to a Timer object goes away and all outstanding tasks have completed execution, the timer's task execution thread terminates gracefully (and becomes subject to garbage collection). However, this can take arbitrarily long to occur. **By default**, the task execution thread does not run as a daemon thread, so it is capable of keeping an application from terminating. If a caller wants to terminate a timer's task execution thread rapidly, the caller should invoke the timer's cancel method.

If the timer's task execution thread terminates unexpectedly, for example, because its stop method is invoked, any further attempt to schedule a task on the timer will result in an IllegalStateException, as if the timer's cancel method had been invoked.

This class is thread-safe: multiple threads can share a single Timer object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the Object.wait(long) method.

Java 5.0 introduced the java.util.concurrent package and one of the concurrency utilities therein is the ScheduledThreadPoolExecutor which is a thread pool for repeatedly executing tasks at a given rate or delay. It is effectively a more versatile replacement for the Timer/TimerTask combination, as it allows multiple service threads, accepts various time units, and doesn't require subclassing TimerTask (just implement Runnable). Configuring ScheduledThreadPoolExecutor with one thread makes it equivalent to Timer.

Implementation note: This class scales to large numbers of concurrently scheduled tasks (thousands should present no problem). Internally, it uses a binary heap to represent its task queue, so the cost to schedule a task is O(log n), where n is the number of concurrently scheduled tasks.

Implementation note: All constructors start a timer thread.

Since:

1.3

# 构造方法

**Timer()**

Creates a new timer.

**Timer(boolean isDaemon)**

Creates a new timer whose associated thread may be specified to run as a daemon.

**Timer(String name)**

Creates a new timer whose associated thread has the specified name.

**Timer(String name, boolean isDaemon)**

Creates a new timer whose associated thread has the specified name, and may be specified to run as a daemon.

# 一般方法

## void cancel()

Terminates this timer, discarding any currently scheduled tasks.

## int purge()

Removes all cancelled tasks from this timer's task queue.

## schedule方法

void schedule(TimerTask task, Date time)

Schedules the specified task for execution at the specified time.

void schedule(TimerTask task, Date firstTime, long period)

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

void **schedule**(TimerTask task, long delay)

Schedules the specified task for execution after the specified delay.

void schedule(TimerTask task, long delay, long period)

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

## scheduleAtFixedRate

void **scheduleAtFixedRate**(TimerTask task, Date firstTime, long period)

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

void **scheduleAtFixedRate**(TimerTask task, long delay, long period)

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.