| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/FutureTask.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/concurrent/Future.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/FutureTask.html)    [**NO FRAMES**](http://docs.google.com/FutureTask.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#17dp8vu) |

## **java.util.concurrent**

Class FutureTask<V>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.concurrent.FutureTask<V>**

**Type Parameters:**V - The result type returned by this FutureTask's get method **All Implemented Interfaces:** [Runnable](http://docs.google.com/java/lang/Runnable.html), [Future](http://docs.google.com/java/util/concurrent/Future.html)<V>, [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<V>

public class **FutureTask<V>**extends [Object](http://docs.google.com/java/lang/Object.html)implements [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<V>

A cancellable asynchronous computation. This class provides a base implementation of [Future](http://docs.google.com/java/util/concurrent/Future.html), with methods to start and cancel a computation, query to see if the computation is complete, and retrieve the result of the computation. The result can only be retrieved when the computation has completed; the get method will block if the computation has not yet completed. Once the computation has completed, the computation cannot be restarted or cancelled.

A FutureTask can be used to wrap a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) or [Runnable](http://docs.google.com/java/lang/Runnable.html) object. Because FutureTask implements Runnable, a FutureTask can be submitted to an [Executor](http://docs.google.com/java/util/concurrent/Executor.html) for execution.

In addition to serving as a standalone class, this class provides protected functionality that may be useful when creating customized task classes.

**Since:** 1.5

| **Constructor Summary** | |
| --- | --- |
| [**FutureTask**](http://docs.google.com/java/util/concurrent/FutureTask.html#FutureTask(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> callable)            Creates a FutureTask that will upon running, execute the given Callable. |
| [**FutureTask**](http://docs.google.com/java/util/concurrent/FutureTask.html#FutureTask(java.lang.Runnable,%20V))([Runnable](http://docs.google.com/java/lang/Runnable.html) runnable, [V](http://docs.google.com/java/util/concurrent/FutureTask.html) result)            Creates a FutureTask that will upon running, execute the given Runnable, and arrange that get will return the given result on successful completion. |

| **Method Summary** | |
| --- | --- |
| boolean | [**cancel**](http://docs.google.com/java/util/concurrent/FutureTask.html#cancel(boolean))(boolean mayInterruptIfRunning)            Attempts to cancel execution of this task. |
| protected  void | [**done**](http://docs.google.com/java/util/concurrent/FutureTask.html#done())()            Protected method invoked when this task transitions to state isDone (whether normally or via cancellation). |
| [V](http://docs.google.com/java/util/concurrent/FutureTask.html) | [**get**](http://docs.google.com/java/util/concurrent/FutureTask.html#get())()            Waits if necessary for the computation to complete, and then retrieves its result. |
| [V](http://docs.google.com/java/util/concurrent/FutureTask.html) | [**get**](http://docs.google.com/java/util/concurrent/FutureTask.html#get(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available. |
| boolean | [**isCancelled**](http://docs.google.com/java/util/concurrent/FutureTask.html#isCancelled())()            Returns true if this task was cancelled before it completed normally. |
| boolean | [**isDone**](http://docs.google.com/java/util/concurrent/FutureTask.html#isDone())()            Returns true if this task completed. |
| void | [**run**](http://docs.google.com/java/util/concurrent/FutureTask.html#run())()            Sets this Future to the result of its computation unless it has been cancelled. |
| protected  boolean | [**runAndReset**](http://docs.google.com/java/util/concurrent/FutureTask.html#runAndReset())()            Executes the computation without setting its result, and then resets this Future to initial state, failing to do so if the computation encounters an exception or is cancelled. |
| protected  void | [**set**](http://docs.google.com/java/util/concurrent/FutureTask.html#set(V))([V](http://docs.google.com/java/util/concurrent/FutureTask.html) v)            Sets the result of this Future to the given value unless this future has already been set or has been cancelled. |
| protected  void | [**setException**](http://docs.google.com/java/util/concurrent/FutureTask.html#setException(java.lang.Throwable))([Throwable](http://docs.google.com/java/lang/Throwable.html) t)            Causes this future to report an ExecutionException with the given throwable as its cause, unless this Future has already been set or has been cancelled. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### FutureTask

public **FutureTask**([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> callable)

Creates a FutureTask that will upon running, execute the given Callable.

**Parameters:**callable - the callable task **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if callable is null

### FutureTask

public **FutureTask**([Runnable](http://docs.google.com/java/lang/Runnable.html) runnable,  
 [V](http://docs.google.com/java/util/concurrent/FutureTask.html) result)

Creates a FutureTask that will upon running, execute the given Runnable, and arrange that get will return the given result on successful completion.

**Parameters:**runnable - the runnable taskresult - the result to return on successful completion. If you don't need a particular result, consider using constructions of the form: Future<?> f = new FutureTask<Object>(runnable, null) **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if runnable is null

| **Method Detail** |
| --- |

### isCancelled

public boolean **isCancelled**()

**Description copied from interface:** [**Future**](http://docs.google.com/java/util/concurrent/Future.html#isCancelled()) Returns true if this task was cancelled before it completed normally.

**Specified by:**[isCancelled](http://docs.google.com/java/util/concurrent/Future.html#isCancelled()) in interface [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> **Returns:**true if this task was cancelled before it completed

### isDone

public boolean **isDone**()

**Description copied from interface:** [**Future**](http://docs.google.com/java/util/concurrent/Future.html#isDone()) Returns true if this task completed. Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

**Specified by:**[isDone](http://docs.google.com/java/util/concurrent/Future.html#isDone()) in interface [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> **Returns:**true if this task completed

### cancel

public boolean **cancel**(boolean mayInterruptIfRunning)

**Description copied from interface:** [**Future**](http://docs.google.com/java/util/concurrent/Future.html#cancel(boolean)) Attempts to cancel execution of this task. This attempt will fail if the task has already completed, has already been cancelled, or could not be cancelled for some other reason. If successful, and this task has not started when cancel is called, this task should never run. If the task has already started, then the mayInterruptIfRunning parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to [Future.isDone()](http://docs.google.com/java/util/concurrent/Future.html#isDone()) will always return true. Subsequent calls to [Future.isCancelled()](http://docs.google.com/java/util/concurrent/Future.html#isCancelled()) will always return true if this method returned true.

**Specified by:**[cancel](http://docs.google.com/java/util/concurrent/Future.html#cancel(boolean)) in interface [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> **Parameters:**mayInterruptIfRunning - true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete **Returns:**false if the task could not be cancelled, typically because it has already completed normally; true otherwise

### get

public [V](http://docs.google.com/java/util/concurrent/FutureTask.html) **get**()  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html),  
 [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html)

**Description copied from interface:** [**Future**](http://docs.google.com/java/util/concurrent/Future.html#get()) Waits if necessary for the computation to complete, and then retrieves its result.

**Specified by:**[get](http://docs.google.com/java/util/concurrent/Future.html#get()) in interface [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> **Returns:**the computed result **Throws:** [CancellationException](http://docs.google.com/java/util/concurrent/CancellationException.html) - if the computation was cancelled [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread was interrupted while waiting [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html) - if the computation threw an exception

### get

public [V](http://docs.google.com/java/util/concurrent/FutureTask.html) **get**(long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html),  
 [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html),  
 [TimeoutException](http://docs.google.com/java/util/concurrent/TimeoutException.html)

**Description copied from interface:** [**Future**](http://docs.google.com/java/util/concurrent/Future.html#get(long,%20java.util.concurrent.TimeUnit)) Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

**Specified by:**[get](http://docs.google.com/java/util/concurrent/Future.html#get(long,%20java.util.concurrent.TimeUnit)) in interface [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> **Parameters:**timeout - the maximum time to waitunit - the time unit of the timeout argument **Returns:**the computed result **Throws:** [CancellationException](http://docs.google.com/java/util/concurrent/CancellationException.html) - if the computation was cancelled [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if the current thread was interrupted while waiting [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html) - if the computation threw an exception [TimeoutException](http://docs.google.com/java/util/concurrent/TimeoutException.html) - if the wait timed out

### done

protected void **done**()

Protected method invoked when this task transitions to state isDone (whether normally or via cancellation). The default implementation does nothing. Subclasses may override this method to invoke completion callbacks or perform bookkeeping. Note that you can query status inside the implementation of this method to determine whether this task has been cancelled.

### set

protected void **set**([V](http://docs.google.com/java/util/concurrent/FutureTask.html) v)

Sets the result of this Future to the given value unless this future has already been set or has been cancelled. This method is invoked internally by the run method upon successful completion of the computation.

**Parameters:**v - the value

### setException

protected void **setException**([Throwable](http://docs.google.com/java/lang/Throwable.html) t)

Causes this future to report an ExecutionException with the given throwable as its cause, unless this Future has already been set or has been cancelled. This method is invoked internally by the run method upon failure of the computation.

**Parameters:**t - the cause of failure

### run

public void **run**()

Sets this Future to the result of its computation unless it has been cancelled.

**Specified by:**[run](http://docs.google.com/java/lang/Runnable.html#run()) in interface [Runnable](http://docs.google.com/java/lang/Runnable.html)**Specified by:**[run](http://docs.google.com/java/util/concurrent/RunnableFuture.html#run()) in interface [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<[V](http://docs.google.com/java/util/concurrent/FutureTask.html)> **See Also:**[Thread.run()](http://docs.google.com/java/lang/Thread.html#run())

### runAndReset

protected boolean **runAndReset**()

Executes the computation without setting its result, and then resets this Future to initial state, failing to do so if the computation encounters an exception or is cancelled. This is designed for use with tasks that intrinsically execute more than once.

**Returns:**true if successfully run and reset

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/FutureTask.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/concurrent/Future.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/FutureTask.html)    [**NO FRAMES**](http://docs.google.com/FutureTask.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#17dp8vu) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).