| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Executors.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/ExecutorCompletionService.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/Executors.html)    [**NO FRAMES**](http://docs.google.com/Executors.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **java.util.concurrent**

Class Executors

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.concurrent.Executors**

public class **Executors**extends [Object](http://docs.google.com/java/lang/Object.html)

Factory and utility methods for [Executor](http://docs.google.com/java/util/concurrent/Executor.html), [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html), [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html), [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html), and [Callable](http://docs.google.com/java/util/concurrent/Callable.html) classes defined in this package. This class supports the following kinds of methods:

* Methods that create and return an [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) set up with commonly useful configuration settings.
* Methods that create and return a [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) set up with commonly useful configuration settings.
* Methods that create and return a "wrapped" ExecutorService, that disables reconfiguration by making implementation-specific methods inaccessible.
* Methods that create and return a [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) that sets newly created threads to a known state.
* Methods that create and return a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) out of other closure-like forms, so they can be used in execution methods requiring Callable.

**Since:** 1.5

| **Method Summary** | |
| --- | --- |
| static [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[Object](http://docs.google.com/java/lang/Object.html)> | [**callable**](http://docs.google.com/java/util/concurrent/Executors.html#callable(java.security.PrivilegedAction))([PrivilegedAction](http://docs.google.com/java/security/PrivilegedAction.html)<?> action)            Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given privileged action and returns its result. |
| static [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[Object](http://docs.google.com/java/lang/Object.html)> | [**callable**](http://docs.google.com/java/util/concurrent/Executors.html#callable(java.security.PrivilegedExceptionAction))([PrivilegedExceptionAction](http://docs.google.com/java/security/PrivilegedExceptionAction.html)<?> action)            Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given privileged exception action and returns its result. |
| static [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[Object](http://docs.google.com/java/lang/Object.html)> | [**callable**](http://docs.google.com/java/util/concurrent/Executors.html#callable(java.lang.Runnable))([Runnable](http://docs.google.com/java/lang/Runnable.html) task)            Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given task and returns null. |
| static   | <T> [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> | | --- | | [**callable**](http://docs.google.com/java/util/concurrent/Executors.html#callable(java.lang.Runnable,%20T))([Runnable](http://docs.google.com/java/lang/Runnable.html) task, T result)            Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given task and returns the given result. |
| static [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) | [**defaultThreadFactory**](http://docs.google.com/java/util/concurrent/Executors.html#defaultThreadFactory())()            Returns a default thread factory used to create new threads. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**newCachedThreadPool**](http://docs.google.com/java/util/concurrent/Executors.html#newCachedThreadPool())()            Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**newCachedThreadPool**](http://docs.google.com/java/util/concurrent/Executors.html#newCachedThreadPool(java.util.concurrent.ThreadFactory))([ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)            Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available, and uses the provided ThreadFactory to create new threads when needed. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**newFixedThreadPool**](http://docs.google.com/java/util/concurrent/Executors.html#newFixedThreadPool(int))(int nThreads)            Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**newFixedThreadPool**](http://docs.google.com/java/util/concurrent/Executors.html#newFixedThreadPool(int,%20java.util.concurrent.ThreadFactory))(int nThreads, [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)            Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue, using the provided ThreadFactory to create new threads when needed. |
| static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) | [**newScheduledThreadPool**](http://docs.google.com/java/util/concurrent/Executors.html#newScheduledThreadPool(int))(int corePoolSize)            Creates a thread pool that can schedule commands to run after a given delay, or to execute periodically. |
| static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) | [**newScheduledThreadPool**](http://docs.google.com/java/util/concurrent/Executors.html#newScheduledThreadPool(int,%20java.util.concurrent.ThreadFactory))(int corePoolSize, [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)            Creates a thread pool that can schedule commands to run after a given delay, or to execute periodically. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**newSingleThreadExecutor**](http://docs.google.com/java/util/concurrent/Executors.html#newSingleThreadExecutor())()            Creates an Executor that uses a single worker thread operating off an unbounded queue. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**newSingleThreadExecutor**](http://docs.google.com/java/util/concurrent/Executors.html#newSingleThreadExecutor(java.util.concurrent.ThreadFactory))([ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)            Creates an Executor that uses a single worker thread operating off an unbounded queue, and uses the provided ThreadFactory to create a new thread when needed. |
| static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) | [**newSingleThreadScheduledExecutor**](http://docs.google.com/java/util/concurrent/Executors.html#newSingleThreadScheduledExecutor())()            Creates a single-threaded executor that can schedule commands to run after a given delay, or to execute periodically. |
| static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) | [**newSingleThreadScheduledExecutor**](http://docs.google.com/java/util/concurrent/Executors.html#newSingleThreadScheduledExecutor(java.util.concurrent.ThreadFactory))([ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)            Creates a single-threaded executor that can schedule commands to run after a given delay, or to execute periodically. |
| static   | <T> [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> | | --- | | [**privilegedCallable**](http://docs.google.com/java/util/concurrent/Executors.html#privilegedCallable(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> callable)            Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that will, when called, execute the given callable under the current access control context. |
| static   | <T> [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> | | --- | | [**privilegedCallableUsingCurrentClassLoader**](http://docs.google.com/java/util/concurrent/Executors.html#privilegedCallableUsingCurrentClassLoader(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> callable)            Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that will, when called, execute the given callable under the current access control context, with the current context class loader as the context class loader. |
| static [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) | [**privilegedThreadFactory**](http://docs.google.com/java/util/concurrent/Executors.html#privilegedThreadFactory())()            Returns a thread factory used to create new threads that have the same permissions as the current thread. |
| static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**unconfigurableExecutorService**](http://docs.google.com/java/util/concurrent/Executors.html#unconfigurableExecutorService(java.util.concurrent.ExecutorService))([ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) executor)            Returns an object that delegates all defined [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) methods to the given executor, but not any other methods that might otherwise be accessible using casts. |
| static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) | [**unconfigurableScheduledExecutorService**](http://docs.google.com/java/util/concurrent/Executors.html#unconfigurableScheduledExecutorService(java.util.concurrent.ScheduledExecutorService))([ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) executor)            Returns an object that delegates all defined [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) methods to the given executor, but not any other methods that might otherwise be accessible using casts. |

| **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)) |

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

### newFixedThreadPool

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **newFixedThreadPool**(int nThreads)

Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue. At any point, at most nThreads threads will be active processing tasks. If additional tasks are submitted when all threads are active, they will wait in the queue until a thread is available. If any thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks. The threads in the pool will exist until it is explicitly [shutdown](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdown()).

**Parameters:**nThreads - the number of threads in the pool **Returns:**the newly created thread pool **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if nThreads <= 0

### newFixedThreadPool

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **newFixedThreadPool**(int nThreads,  
 [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)

Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue, using the provided ThreadFactory to create new threads when needed. At any point, at most nThreads threads will be active processing tasks. If additional tasks are submitted when all threads are active, they will wait in the queue until a thread is available. If any thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks. The threads in the pool will exist until it is explicitly [shutdown](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdown()).

**Parameters:**nThreads - the number of threads in the poolthreadFactory - the factory to use when creating new threads **Returns:**the newly created thread pool **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if nThreads <= 0

### newSingleThreadExecutor

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **newSingleThreadExecutor**()

Creates an Executor that uses a single worker thread operating off an unbounded queue. (Note however that if this single thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks.) Tasks are guaranteed to execute sequentially, and no more than one task will be active at any given time. Unlike the otherwise equivalent newFixedThreadPool(1) the returned executor is guaranteed not to be reconfigurable to use additional threads.

**Returns:**the newly created single-threaded Executor

### newSingleThreadExecutor

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **newSingleThreadExecutor**([ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)

Creates an Executor that uses a single worker thread operating off an unbounded queue, and uses the provided ThreadFactory to create a new thread when needed. Unlike the otherwise equivalent newFixedThreadPool(1, threadFactory) the returned executor is guaranteed not to be reconfigurable to use additional threads.

**Parameters:**threadFactory - the factory to use when creating new threads **Returns:**the newly created single-threaded Executor **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory is null

### newCachedThreadPool

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **newCachedThreadPool**()

Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available. These pools will typically improve the performance of programs that execute many short-lived asynchronous tasks. Calls to execute will reuse previously constructed threads if available. If no existing thread is available, a new thread will be created and added to the pool. Threads that have not been used for sixty seconds are terminated and removed from the cache. Thus, a pool that remains idle for long enough will not consume any resources. Note that pools with similar properties but different details (for example, timeout parameters) may be created using [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) constructors.

**Returns:**the newly created thread pool

### newCachedThreadPool

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **newCachedThreadPool**([ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)

Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available, and uses the provided ThreadFactory to create new threads when needed.

**Parameters:**threadFactory - the factory to use when creating new threads **Returns:**the newly created thread pool **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory is null

### newSingleThreadScheduledExecutor

public static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **newSingleThreadScheduledExecutor**()

Creates a single-threaded executor that can schedule commands to run after a given delay, or to execute periodically. (Note however that if this single thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks.) Tasks are guaranteed to execute sequentially, and no more than one task will be active at any given time. Unlike the otherwise equivalent newScheduledThreadPool(1) the returned executor is guaranteed not to be reconfigurable to use additional threads.

**Returns:**the newly created scheduled executor

### newSingleThreadScheduledExecutor

public static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **newSingleThreadScheduledExecutor**([ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)

Creates a single-threaded executor that can schedule commands to run after a given delay, or to execute periodically. (Note however that if this single thread terminates due to a failure during execution prior to shutdown, a new one will take its place if needed to execute subsequent tasks.) Tasks are guaranteed to execute sequentially, and no more than one task will be active at any given time. Unlike the otherwise equivalent newScheduledThreadPool(1, threadFactory) the returned executor is guaranteed not to be reconfigurable to use additional threads.

**Parameters:**threadFactory - the factory to use when creating new threads **Returns:**a newly created scheduled executor **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory is null

### newScheduledThreadPool

public static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **newScheduledThreadPool**(int corePoolSize)

Creates a thread pool that can schedule commands to run after a given delay, or to execute periodically.

**Parameters:**corePoolSize - the number of threads to keep in the pool, even if they are idle. **Returns:**a newly created scheduled thread pool **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if corePoolSize < 0

### newScheduledThreadPool

public static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **newScheduledThreadPool**(int corePoolSize,  
 [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)

Creates a thread pool that can schedule commands to run after a given delay, or to execute periodically.

**Parameters:**corePoolSize - the number of threads to keep in the pool, even if they are idle.threadFactory - the factory to use when the executor creates a new thread. **Returns:**a newly created scheduled thread pool **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if corePoolSize < 0 [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory is null

### unconfigurableExecutorService

public static [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **unconfigurableExecutorService**([ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) executor)

Returns an object that delegates all defined [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) methods to the given executor, but not any other methods that might otherwise be accessible using casts. This provides a way to safely "freeze" configuration and disallow tuning of a given concrete implementation.

**Parameters:**executor - the underlying implementation **Returns:**an ExecutorService instance **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if executor null

### unconfigurableScheduledExecutorService

public static [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **unconfigurableScheduledExecutorService**([ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) executor)

Returns an object that delegates all defined [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) methods to the given executor, but not any other methods that might otherwise be accessible using casts. This provides a way to safely "freeze" configuration and disallow tuning of a given concrete implementation.

**Parameters:**executor - the underlying implementation **Returns:**a ScheduledExecutorService instance **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if executor null

### defaultThreadFactory

public static [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) **defaultThreadFactory**()

Returns a default thread factory used to create new threads. This factory creates all new threads used by an Executor in the same [ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html). If there is a [SecurityManager](http://docs.google.com/java/lang/SecurityManager.html), it uses the group of [System.getSecurityManager()](http://docs.google.com/java/lang/System.html#getSecurityManager()), else the group of the thread invoking this defaultThreadFactory method. Each new thread is created as a non-daemon thread with priority set to the smaller of Thread.NORM\_PRIORITY and the maximum priority permitted in the thread group. New threads have names accessible via [Thread.getName()](http://docs.google.com/java/lang/Thread.html#getName()) of *pool-N-thread-M*, where *N* is the sequence number of this factory, and *M* is the sequence number of the thread created by this factory.

**Returns:**a thread factory

### privilegedThreadFactory

public static [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) **privilegedThreadFactory**()

Returns a thread factory used to create new threads that have the same permissions as the current thread. This factory creates threads with the same settings as [defaultThreadFactory()](http://docs.google.com/java/util/concurrent/Executors.html#defaultThreadFactory()), additionally setting the AccessControlContext and contextClassLoader of new threads to be the same as the thread invoking this privilegedThreadFactory method. A new privilegedThreadFactory can be created within an [AccessController.doPrivileged(java.security.PrivilegedAction)](http://docs.google.com/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)) action setting the current thread's access control context to create threads with the selected permission settings holding within that action.

Note that while tasks running within such threads will have the same access control and class loader settings as the current thread, they need not have the same [ThreadLocal](http://docs.google.com/java/lang/ThreadLocal.html) or [InheritableThreadLocal](http://docs.google.com/java/lang/InheritableThreadLocal.html) values. If necessary, particular values of thread locals can be set or reset before any task runs in [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) subclasses using [ThreadPoolExecutor.beforeExecute(java.lang.Thread, java.lang.Runnable)](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#beforeExecute(java.lang.Thread,%20java.lang.Runnable)). Also, if it is necessary to initialize worker threads to have the same InheritableThreadLocal settings as some other designated thread, you can create a custom ThreadFactory in which that thread waits for and services requests to create others that will inherit its values.

**Returns:**a thread factory **Throws:** [AccessControlException](http://docs.google.com/java/security/AccessControlException.html) - if the current access control context does not have permission to both get and set context class loader.

### callable

public static <T> [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> **callable**([Runnable](http://docs.google.com/java/lang/Runnable.html) task,  
 T result)

Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given task and returns the given result. This can be useful when applying methods requiring a Callable to an otherwise resultless action.

**Parameters:**task - the task to runresult - the result to return **Returns:**a callable object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if task null

### callable

public static [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[Object](http://docs.google.com/java/lang/Object.html)> **callable**([Runnable](http://docs.google.com/java/lang/Runnable.html) task)

Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given task and returns null.

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

### callable

public static [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[Object](http://docs.google.com/java/lang/Object.html)> **callable**([PrivilegedAction](http://docs.google.com/java/security/PrivilegedAction.html)<?> action)

Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given privileged action and returns its result.

**Parameters:**action - the privileged action to run **Returns:**a callable object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if action null

### callable

public static [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[Object](http://docs.google.com/java/lang/Object.html)> **callable**([PrivilegedExceptionAction](http://docs.google.com/java/security/PrivilegedExceptionAction.html)<?> action)

Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that, when called, runs the given privileged exception action and returns its result.

**Parameters:**action - the privileged exception action to run **Returns:**a callable object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if action null

### privilegedCallable

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

Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that will, when called, execute the given callable under the current access control context. This method should normally be invoked within an [AccessController.doPrivileged(java.security.PrivilegedAction)](http://docs.google.com/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)) action to create callables that will, if possible, execute under the selected permission settings holding within that action; or if not possible, throw an associated [AccessControlException](http://docs.google.com/java/security/AccessControlException.html).

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

### privilegedCallableUsingCurrentClassLoader

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

Returns a [Callable](http://docs.google.com/java/util/concurrent/Callable.html) object that will, when called, execute the given callable under the current access control context, with the current context class loader as the context class loader. This method should normally be invoked within an [AccessController.doPrivileged(java.security.PrivilegedAction)](http://docs.google.com/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)) action to create callables that will, if possible, execute under the selected permission settings holding within that action; or if not possible, throw an associated [AccessControlException](http://docs.google.com/java/security/AccessControlException.html).

**Parameters:**callable - the underlying task **Returns:**a callable object **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if callable null [AccessControlException](http://docs.google.com/java/security/AccessControlException.html) - if the current access control context does not have permission to both set and get context class loader.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Executors.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/ExecutorCompletionService.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/ExecutorService.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/Executors.html)    [**NO FRAMES**](http://docs.google.com/Executors.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[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).