

The Java Executors Utility Class

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Lesson

- Understand the implementation of key methods in the Executors utility class

Class Executors

`java.lang.Object`
`java.util.concurrent.Executors`

```
public class Executors  
extends Object
```

Factory and utility methods for `Executor`, `ExecutorService`, `ScheduledExecutorService`, `ThreadFactory`, and `Callable` classes defined in this package. This class supports the following kinds of methods:

- Methods that create and return an `ExecutorService` set up with commonly useful configuration settings.
- Methods that create and return a `ScheduledExecutorService` 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` that sets newly created threads to a known state.
- Methods that create and return a `Callable` out of other closure-like forms, so they can be used in execution methods requiring `Callable`.

The Java Executors Utility Class

The Java Executors Utility Class

- Executors is a Java utility class

Class Executors

```
java.lang.Object  
    java.util.concurrent.Executors
```

```
public class Executors  
    extends Object
```

Factory and utility methods for `Executor`, `ExecutorService`, `ScheduledExecutorService`, `ThreadFactory`, and `Callable` classes defined in this package. This class supports the following kinds of methods:

- Methods that create and return an `ExecutorService` set up with commonly useful configuration settings.
- Methods that create and return a `ScheduledExecutorService` 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` that sets newly created threads to a known state.
- Methods that create and return a `Callable` out of other closure-like forms, so they can be used in execution methods requiring `Callable`.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html

The Java Executors Utility Class

- Executors is a Java utility class
 - A utility class is a final class having only static methods, no non-static state, & a private constructor

Class Executors

```
java.lang.Object  
java.util.concurrent.Executors
```

```
public class Executors  
extends Object
```









































Factory and utility methods for `Executor`, `ExecutorService`, `ScheduledExecutorService`, `ThreadFactory`, and `Callable` classes defined in this package. This class supports the following kinds of methods:

- Methods that create and return an `ExecutorService` set up with commonly useful configuration settings.
- Methods that create and return a `ScheduledExecutorService` 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` that sets newly created threads to a known state.
- Methods that create and return a `Callable` out of other closure-like forms, so they can be used in execution methods requiring `Callable`.

See www.quora.com/What-is-the-best-way-to-write-utility-classes-in-Java/answer/Jon-Harley

The Java Executors Utility Class

- It defines utility methods used by Executor framework classes

| <<Java Class>> | |
|---|--|
| Executors | |
|  |  newFixedThreadPool(int):ExecutorService |
|  |  newWorkStealingPool(int):ExecutorService |
|  |  newWorkStealingPool():ExecutorService |
|  |  newFixedThreadPool(int,ThreadFactory):ExecutorService |
|  |  newSingleThreadExecutor():ExecutorService |
|  |  newSingleThreadExecutor(ThreadFactory):ExecutorService |
|  |  newCachedThreadPool():ExecutorService |
|  |  newCachedThreadPool(ThreadFactory):ExecutorService |
|  |  newSingleThreadScheduledExecutor():ScheduledExecutorService |
|  |  newSingleThreadScheduledExecutor(ThreadFactory):ScheduledExecutorService |
|  |  newScheduledThreadPool(int):ScheduledExecutorService |
|  |  newScheduledThreadPool(int,ThreadFactory):ScheduledExecutorService |
|  |  defaultThreadFactory() |
|  |  privilegedThreadFactory() |
|  |  callable(Runnable,T):Callable<T> |
|  |  callable(Runnable):Callable<Object> |
|  |  callable(PrivilegedAction<?>):Callable<Object> |
|  |  callable(PrivilegedExceptionAction<?>):Callable<Object> |
|  |  privilegedCallable(Callable<T>):Callable<T> |
|  |  privilegedCallableUsingCurrentClassLoader(Callable<T>):Callable<T> |

The Java Executors Utility Class

- It defines utility methods used by Executor framework classes, e.g.
 - `defaultThreadFactory()` sets new threads to a known state

```
public class Executors {  
    ...  
    public static ThreadFactory  
        defaultThreadFactory() {  
        return new  
            DefaultThreadFactory();  
    }  
}
```



The Java Executors Utility Class

- It defines utility methods used by Executor framework classes, e.g.
 - `defaultThreadFactory()` sets new threads to a known state
 - The `defaultThreadFactory()` is used by these factory methods

`defaultThreadFactory()`

Returns a default thread factory used to create new threads.

`newCachedThreadPool()`

Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available.

`newCachedThreadPool(ThreadFactory 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.

`newFixedThreadPool(int nThreads)`

Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue.

`newFixedThreadPool(int nThreads, ThreadFactory 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.

`newScheduledThreadPool(int corePoolSize)`

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

`newScheduledThreadPool(int corePoolSize, ThreadFactory threadFactory)`

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

The Java Executors Utility Class

- It defines utility methods used by Executor framework classes, e.g.
 - `defaultThreadFactory()` sets new threads to a known state
 - User-defined `ThreadFactory` objects can be passed to other factory methods in Executors

`defaultThreadFactory()`

Returns a default thread factory used to create new threads.

`newCachedThreadPool()`

Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available.

`newCachedThreadPool(ThreadFactory 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.

`newFixedThreadPool(int nThreads)`

Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue.

`newFixedThreadPool(int nThreads, ThreadFactory 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.

`newScheduledThreadPool(int corePoolSize)`

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

`newScheduledThreadPool(int corePoolSize, ThreadFactory threadFactory)`

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

The Java Executors Utility Class

- It defines utility methods used by Executor framework classes, e.g.
 - `defaultThreadFactory()` sets new threads to a known state
 - User-defined `ThreadFactory` objects can be passed to other factory methods in Executors
 - e.g., enables apps to create custom thread subclasses, priorities, etc.



`defaultThreadFactory()`

Returns a default thread factory used to create new threads.

`newCachedThreadPool()`

Creates a thread pool that creates new threads as needed, but will reuse previously constructed threads when they are available.

`newCachedThreadPool(ThreadFactory 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.

`newFixedThreadPool(int nThreads)`

Creates a thread pool that reuses a fixed number of threads operating off a shared unbounded queue.

`newFixedThreadPool(int nThreads, ThreadFactory 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.

`newScheduledThreadPool(int corePoolSize)`

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

`newScheduledThreadPool(int corePoolSize, ThreadFactory threadFactory)`

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

The Java Executors Utility Class

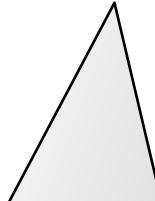
- It defines utility methods used by Executor framework classes, e.g.
 - `defaultThreadFactory()` sets new threads to a known state
 - User-defined `ThreadFactory` objects can be passed to other factory methods in `Executors`
- Create a callable from a runnable

```
public class Executors {  
    ...  
    public static Callable<Object>  
        callable(Runnable task) {  
        ...  
        return new RunnableAdapter  
            <Object>(task, null);  
    }  
}
```

The Java Executors Utility Class

- It defines utility methods used by Executor framework classes, e.g.
 - `defaultThreadFactory()` sets new threads to a known state
 - User-defined `ThreadFactory` objects can be passed to other factory methods in `Executors`
- Create a callable from a runnable

```
public class Executors {  
    ...  
    public static Callable<Object>  
        callable(Runnable task) {  
        ...  
        return new RunnableAdapter  
            <Object>(task, null);  
    }  
}
```



```
class RunnableAdapter<T> implements Callable<T> {  
    final Runnable task; final T result;  
  
    RunnableAdapter(Runnable t, T r) { task = t; result = r; }  
    public T call() { task.run(); return result; }  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools

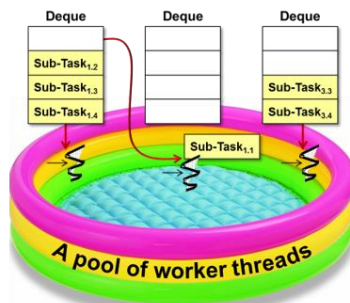
*Cached (Variable-sized)
Thread Pool*



*Fixed-sized
Thread Pool*



*Work-stealing
Thread Pool*



<<Java Class>>

Executors

```
newFixedThreadPool(int):ExecutorService
newWorkStealingPool(int):ExecutorService
newWorkStealingPool():ExecutorService
newFixedThreadPool(int,ThreadFactory):ExecutorService
newSingleThreadExecutor():ExecutorService
newSingleThreadExecutor(ThreadFactory):ExecutorService
newCachedThreadPool():ExecutorService
newCachedThreadPool(ThreadFactory):ExecutorService
newSingleThreadScheduledExecutor():ScheduledExecutorService
newSingleThreadScheduledExecutor(ThreadFactory):ScheduledExecutorService
newScheduledThreadPool(int):ScheduledExecutorService
newScheduledThreadPool(int,ThreadFactory):ScheduledExecutorService
defaultThreadFactory()
privilegedThreadFactory()
callable(Runnable,T):Callable<T>
callable(Runnable):Callable<Object>
callable(PrivilegedAction<?>):Callable<Object>
callable(PrivilegedExceptionAction<?>):Callable<Object>
privilegedCallable(Callable<T>):Callable<T>
privilegedCallableUsingCurrentClassLoader(Callable<T>):Callable<T>
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools



| <<Java Class>> Executors | |
|--|--|
| newFixedThreadPool(int):ExecutorService | |
| newWorkStealingPool(int):ExecutorService | |
| newWorkStealingPool():ExecutorService | |
| newFixedThreadPool(int,ThreadFactory):ExecutorService | |
| newSingleThreadExecutor():ExecutorService | |
| newSingleThreadExecutor(ThreadFactory):ExecutorService | |
| newCachedThreadPool():ExecutorService | |
| newCachedThreadPool(ThreadFactory):ExecutorService | |
| newSingleThreadScheduledExecutor():ScheduledExecutorService | |
| newSingleThreadScheduledExecutor(ThreadFactory):ScheduledExecutorService | |
| newScheduledThreadPool(int):ScheduledExecutorService | |
| newScheduledThreadPool(int,ThreadFactory):ScheduledExecutorService | |
| defaultThreadFactory() | |
| privilegedThreadFactory() | |
| callable(Runnable,T):Callable<T> | |
| callable(Runnable):Callable<Object> | |
| callable(PrivilegedAction<?>):Callable<Object> | |
| callable(PrivilegedExceptionAction<?>):Callable<Object> | |
| privilegedCallable(Callable<T>):Callable<T> | |
| privilegedCallableUsingCurrentClassLoader(Callable<T>):Callable<T> | |

It can also create a thread pool with just one thread!

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools

```
public class Executors {  
    ...  
    public static ExecutorService  
newFixedThreadPool(int nThreads,  
ThreadFactory threadFactory){  
    return new ThreadPoolExecutor  
        (nThreads, nThreads,  
         0L, TimeUnit.MILLISECONDS,  
         new LinkedBlockingQueue  
             <Runnable>(),  
         threadFactory);  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Uses ThreadPoolExecutor class

```
public class Executors {  
    ...  
    public static ExecutorService  
    newFixedThreadPool(int nThreads,  
        ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (nThreads, nThreads,  
                0L, TimeUnit.MILLISECONDS,  
                new LinkedBlockingQueue  
                    <Runnable>(),  
                threadFactory);  
    }  
}
```

See earlier lesson on "*Overview of Java ThreadPoolExecutor*"

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Uses `ThreadPoolExecutor` class
 - Core pool size & maximum pool size are the same

```
public class Executors {  
    ...  
    public static ExecutorService  
    newFixedThreadPool(int nThreads,  
        ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (nThreads, nThreads,  
                0L, TimeUnit.MILLISECONDS,  
                new LinkedBlockingQueue  
                    <Runnable>(),  
                threadFactory);  
    }  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Uses `ThreadPoolExecutor` class
 - Core pool size & maximum pool size are the same
 - Idle threads don't timeout

```
public class Executors {  
    ...  
    public static ExecutorService  
    newFixedThreadPool(int nThreads,  
        ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (nThreads, nThreads,  
                0L, TimeUnit.MILLISECONDS,  
                new LinkedBlockingQueue  
                    <Runnable>(),  
                threadFactory);  
    }  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Uses `ThreadPoolExecutor` class
 - Core pool size & maximum pool size are the same
 - Idle threads don't timeout
 - Threads can block on a shared unbounded queue

```
public class Executors {  
    ...  
    public static ExecutorService  
    newFixedThreadPool(int nThreads,  
        ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (nThreads, nThreads,  
                0L, TimeUnit.MILLISECONDS,  
                new LinkedBlockingQueue  
                    <Runnable>(),  
                threadFactory);  
    }  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Uses ThreadPoolExecutor class
 - Core pool size & maximum pool size are the same
 - Idle threads don't timeout
 - Threads can block on a shared unbounded queue
 - Threads can be created via a custom ThreadFactory

```
public class Executors {  
    ...  
    public static ExecutorService  
    newFixedThreadPool(int nThreads,  
        ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (nThreads, nThreads,  
                0L, TimeUnit.MILLISECONDS,  
                new LinkedBlockingQueue  
                    <Runnable>(),  
                    threadFactory);  
    }  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Uses ThreadPoolExecutor class
 - Core pool size & maximum pool size are the same
 - Idle threads don't timeout
 - Threads can block on a shared unbounded queue
 - Threads can be created via a custom ThreadFactory

```
public class Executors {  
    ...  
    public static ExecutorService  
        newFixedThreadPool(int  
                                nThreads) {  
        return new ThreadPoolExecutor  
            (nThreads, nThreads,  
             0L, TimeUnit.MILLISECONDS,  
             new LinkedBlockingQueue  
                 <Runnable>());  
    }  
}
```



A variant of `newFixedThreadPool()` uses `DefaultThreadFactory`

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools

```
public class Executors {  
    ...  
    public static ExecutorService  
newCachedThreadPool  
    (ThreadFactory threadFactory){  
    return new ThreadPoolExecutor  
        (0, Integer.MAX_VALUE,  
         60L, TimeUnit.SECONDS,  
         new SynchronousQueue  
             <Runnable>(),  
         threadFactory);  
    }  
    ...  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Uses ThreadPoolExecutor class

```
public class Executors {  
    ...  
    public static ExecutorService  
    newCachedThreadPool  
        (ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (0, Integer.MAX_VALUE,  
             60L, TimeUnit.SECONDS,  
             new SynchronousQueue  
                 <Runnable>(),  
             threadFactory);  
    }  
    ...  
}
```

See earlier lesson on "*Overview of Java ThreadPoolExecutor*"

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Uses ThreadPoolExecutor class
 - New threads started as needed, but existing threads are reused

```
public class Executors {  
    ...  
    public static ExecutorService  
    newCachedThreadPool  
        (ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (0, Integer.MAX_VALUE,  
             60L, TimeUnit.SECONDS,  
             new SynchronousQueue  
                 <Runnable>(),  
             threadFactory);  
    }  
    ...  
}
```


The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Uses ThreadPoolExecutor class
 - New threads started as needed, but existing threads are reused
 - Terminate & remove threads from cache if unused for 60 seconds

```
public class Executors {  
    ...  
    public static ExecutorService  
    newCachedThreadPool  
        (ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (0, Integer.MAX_VALUE,  
             60L, TimeUnit.SECONDS,  
             new SynchronousQueue  
                 <Runnable>(),  
             threadFactory);  
    }  
    ...  
}
```



The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Uses ThreadPoolExecutor class
 - New threads started as needed, but existing threads are reused
 - Terminate & remove threads from cache if unused for 60 seconds
 - execute() does a “rendezvous” with a new worker thread

```
public class Executors {  
    ...  
    public static ExecutorService  
    newCachedThreadPool  
        (ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (0, Integer.MAX_VALUE,  
             60L, TimeUnit.SECONDS,  
             new SynchronousQueue  
                 <Runnable>(),  
             threadFactory);  
    }  
    ...  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Uses ThreadPoolExecutor class
 - New threads started as needed, but existing threads are reused
 - Terminate & remove threads from cache if unused for 60 seconds
 - execute() does a “rendezvous” with a new worker thread
 - Threads can be created via custom ThreadFactory

```
public class Executors {  
    ...  
    public static ExecutorService  
    newCachedThreadPool  
        (ThreadFactory threadFactory){  
        return new ThreadPoolExecutor  
            (0, Integer.MAX_VALUE,  
             60L, TimeUnit.SECONDS,  
             new SynchronousQueue  
                 <Runnable>(),  
             threadFactory);  
    }  
    ...  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Uses ThreadPoolExecutor class
 - New threads started as needed, but existing threads are reused
 - Terminate & remove threads from cache if unused for 60 seconds
 - execute() does a “rendezvous” with a new worker thread
 - Threads can be created via custom ThreadFactory

```
public class Executors {  
    ...  
    public static ExecutorService  
    newCachedThreadPool () {  
        return new ThreadPoolExecutor  
            (0, Integer.MAX_VALUE,  
             60L, TimeUnit.SECONDS,  
             new SynchronousQueue  
                 <Runnable> () ) ;  
    }  
    ...  
}
```



A variant of `newCachedThreadPool()` uses `DefaultThreadFactory`

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Create work-stealing thread pools

```
public class Executors {  
    ...  
    public static ExecutorService  
        newWorkStealingPool  
        (int parallelism) {  
        return new ForkJoinPool  
            (parallelism,  
             ForkJoinPool  
             .defaultForkJoin  
             WorkerThreadFactory,  
             null,  
             true);  
    }  
    ...  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Create work-stealing thread pools
 - Implemented via ForkJoinPool

```
public class Executors {  
    ...  
    public static ExecutorService  
        newWorkStealingPool  
        (int parallelism) {  
        return new ForkJoinPool  
            (parallelism,  
             ForkJoinPool  
                .defaultForkJoin  
                WorkerThreadFactory,  
             null,  
             true);  
    }  
    ...  
}
```

See lessons on "*Java ForkJoinPool*"

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Create work-stealing thread pools
 - Implemented via ForkJoinPool
 - Set the target parallelism level

```
public class Executors {  
    ...  
    public static ExecutorService  
        newWorkStealingPool  
        (int parallelism) {  
        return new ForkJoinPool  
            (parallelism,  
             ForkJoinPool  
                .defaultForkJoin  
                WorkerThreadFactory,  
             null,  
             true);  
    }  
    ...  
}
```

The Java Executors Utility Class

- It also defines factory methods to make Executor thread pools, e.g.
 - Create fixed-sized thread pools
 - Create variable-sized thread pools
 - Create work-stealing thread pools
 - Implemented via ForkJoinPool
 - Set the target parallelism level
 - etc.

```
public class Executors {  
    ...  
    public static ExecutorService  
        newWorkStealingPool  
            (int parallelism) {  
        return new ForkJoinPool  
            (parallelism,  
             ForkJoinPool  
                .defaultForkJoin  
                WorkerThreadFactory,  
             null,  
             true) ;  
    }  
    ...  
}
```

End of the Java Executors Utility Class