Executors工厂类

# Executors工厂类

## 继承关系介绍

public class **Executors** extends **Object 存在包java.util.concurrent.Executors。**



since：JDK1.5

## 功能介绍：四大类

**Executors是一个线程的工厂类，方便快速地创建很多线程池，是一个线程池的工具类。**

为该包中的 **ExecutorService**, **ScheduledExecutorService**, **ThreadFactory**, and **Callable** 这些类提供工厂方法和工具方法。因此Executors中的方法也主要是分成**四大类**。

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

### 提供创建**线程池ExecutorService**的方法：Methods that create and return an **ExecutorService** set up with commonly useful configuration settings.

### 提供创建**定时线程池ScheduledExecutorService**的方法：Methods that create and return a **ScheduledExecutorService** set up with commonly useful configuration settings.

### **"wrapped" ExecutorService** ：Methods that create and return **a "wrapped" ExecutorService**, that disables reconfiguration by making implementation-specific methods inaccessible.

**如：DelegatedExecutorService 、DelegatedScheduledExecutorService 、FinalizableDelegatedExecutorService。**

### 返回**ThreadFactory**的方法：Methods that create and return a **ThreadFactory** that sets newly created threads to a known state.

两个：defaultThreadFactory和privilegedThreadFactory。

### 返回**Callable的方法**：Methods that create and return a **Callable** out of other closure-like forms, so they can be used in execution methods requiring Callable.

## 工具方法介绍

### **获取线程池ExecutorService**；(最常用)

### 获取**定时线程池ScheduledExecutorService**；

### 获取**ThreadFactory：defaultThreadFactory和privilegeThreadFactory**

### 获取**Callable**

## 返回ThreadFactory：2个defaultThreadFactory和privilegedThreadFactory

### static ThreadFactory **defaultThreadFactory**()

Returns a default thread factory used to create new threads.

### static ThreadFactory **privilegedThreadFactory**()

Returns a thread factory used to create new threads that have the same permissions as the current thread.

## 获取线程池：返回ExecutorService接口引用

没有ThreadFactory参数的内部缺省使用的是Executors. **defaultThreadFactory();返回的DefaultThreadFactory。**

### 获取单线程线程池**SingleThreadExecutor**

static ExecutorService **newSingleThreadExecutor**()

Creates an Executor that uses a single worker thread operating off an unbounded queue.

static ExecutorService **newSingleThreadExecutor**(ThreadFactory 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.

### 获取**单线程定时周期执行**线程池**SingleThreadScheduledExecutor**

static **ScheduledExecutorService** newSingleThreadScheduledExecutor()

Creates a single-threaded executor **that can schedule commands to run after a given delay, or to execute periodically.**

static **ScheduledExecutorService** newSingleThreadScheduledExecutor(ThreadFactory threadFactory)

Creates a single-threaded executor that can schedule commands to run after a given delay, or to execute periodically.

### **具有缓存池大小可伸缩的线程池CachedThreadPool**

static ExecutorService **newCachedThreadPool**()

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

static ExecutorService **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.**

### 获取线程数量固定的线程池FixedThreadPool

static ExecutorService **newFixedThreadPool**(int nThreads)

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

static ExecutorService **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.

### 获取定时周期执行的线程池ScheduledThreadPool

static ScheduledExecutorService newScheduledThreadPool(int corePoolSize)

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

static ScheduledExecutorService newScheduledThreadPool(int corePoolSize, ThreadFactory threadFactory)

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

### WorkStealingPool

static ExecutorService newWorkStealingPool()

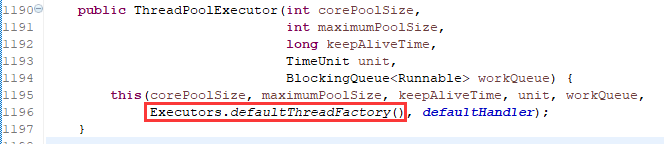
Creates a work-stealing thread pool using all available processors as its target parallelism level.

static ExecutorService newWorkStealingPool(int parallelism)

Creates a thread pool that maintains enough threads to support the given parallelism level, and may use multiple queues to reduce contention.

### 获取线程池的工具方法参数ThreadFactory作用

利用Executors工具类的newXxxThreadPool方法获取线程池的时候，有ThreadFactory和没有ThreadFactory参数这区别：没有ThreadFactory参数的，在方法内部缺省使用的是Executors. **defaultThreadFactory();返回一个Executors内部的一个类DefaultThreadFactory。**



# Executors工具类创建的线程池的源码分析

线程池采用的**BlockingQueue**依次为

newSingleThreadExecutor、newSingleThreadExecutor用的是**LinkedBlockingQueue链表阻塞队列**；

newFixedThreadPool **用的是LinkedBlockingQueue链表阻塞队列**；

newCachedThreadPool用的是**SynchronousQueue同步队列；**

newScheduledThreadPool**用的是DelayedWorkQueue(Executors内部定义的BlockingQueue)；**

## newFixedThreadPool：源代码分析

**Executors中的newXxxThreadPool方法都是直接通过new ThreadPoolExecutor(…)直接创建线程池**的。

### 带有**ThreadFactory**参数：

注意：**BlockingQueue**使用的是**LinkedBlockingQueue**。

public static ExecutorService **newFixedThreadPool**(int nThreads, **ThreadFactory** threadFactory) {

return new ThreadPoolExecutor(nThreads, nThreads,

0L, TimeUnit.MILLISECONDS,

**new LinkedBlockingQueue<Runnable>()**,

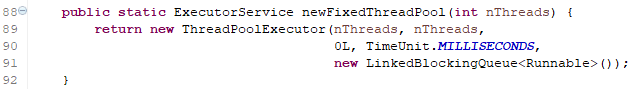
threadFactory);

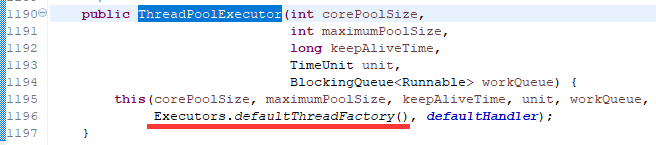
}

**采用链表阻塞队列**；**核心线程数和最大线程数相等**；keepAliveTime失效，无意义，设为0。

预热之后，线程池中的线程数固定为nThreads；阻塞队列长度无限制。

### 不带有**ThreadFactory**参数：





## newSingleThreadExecutor

与newFixedThreadPool相比，只是**nThreads=1**即可。

线程池中固定只有1个线程。

public static ExecutorService **newSingleThreadExecutor**(ThreadFactory threadFactory) {

return new FinalizableDelegatedExecutorService

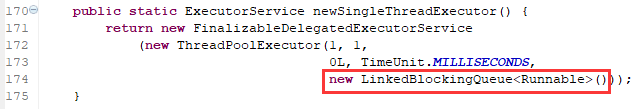
(new ThreadPoolExecutor(**1, 1,**

0L, TimeUnit.MILLISECONDS,

new LinkedBlockingQueue<Runnable>(),

threadFactory));

}



public static ExecutorService **newSingleThreadExecutor**() {

return new FinalizableDelegatedExecutorService

(new ThreadPoolExecutor(1, 1,

0L, TimeUnit.MILLISECONDS,

new LinkedBlockingQueue<Runnable>()));

}

## newCachedThreadPool

阻塞队列采用的是**SynchronousQueue同步队列**；

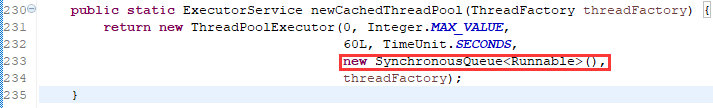
核心线程数为0，最大线程数无界(Integer的最大值)；

keepAliveTime空闲线程存活时间是60秒；

**在请求数大于处理数，线程无限扩增，容易由于线程数过多导致内存溢出；**

**在没有请求的时候，线程数为0，不占用系统资源；**

**来一个请求，若没有空余线程，则直接创建一个新的线程处理任务，因此吞吐量更高。**



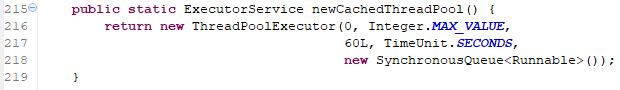
public static ExecutorService newCachedThreadPool(ThreadFactory threadFactory) {

return new ThreadPoolExecutor(0, Integer.MAX\_VALUE,

**60L, TimeUnit.SECONDS**,

**new SynchronousQueue<Runnable>(),**

threadFactory);

}

public static ExecutorService newCachedThreadPool() {

return new ThreadPoolExecutor(0, Integer.MAX\_VALUE,

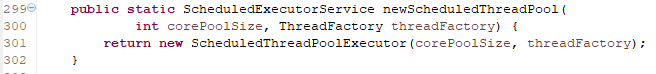
60L, TimeUnit.SECONDS,

new SynchronousQueue<Runnable>());

}

## newScheduledThreadPool

**DelayedWorkQueue**



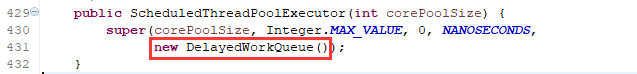
public ScheduledThreadPoolExecutor(int **corePoolSize**,

ThreadFactory threadFactory) {

super(corePoolSize, Integer.MAX\_VALUE, 0, NANOSECONDS,

**new DelayedWorkQueue()**, threadFactory);

}





public ScheduledThreadPoolExecutor(int corePoolSize) {  
 super(corePoolSize, Integer.*MAX\_VALUE*, 0, *NANOSECONDS*,  
 new DelayedWorkQueue());  
}

上述的super方法就是ThreadPoolExecutor的构造方法：

public ThreadPoolExecutor(int corePoolSize,  
 int maximumPoolSize,  
 long keepAliveTime,  
 TimeUnit unit,  
 BlockingQueue<Runnable> workQueue) {  
 this(corePoolSize, maximumPoolSize, keepAliveTime, unit, workQueue,  
 Executors.*defaultThreadFactory*(), *defaultHandler*);  
}

在**ScheduledThreadPoolExecutor**中，创建了一个阻塞队列BlockingQueue。

static class **DelayedWorkQueue** extends AbstractQueue<Runnable>  
 implements BlockingQueue<Runnable> {…}

普通的ThreadPoolExecutor与定时的ScheduledThreadPoolExecutor相比较：

核心线程数、最大线程数、空闲线程存活时间、饱和策略都一致；

不同的是**阻塞队列和处理的Runnable任务。**

**在**ScheduledThreadPoolExecutor中有两个比较重要的子类：

一是延迟队列DelayedWorkQueue；

一是定时任务ScheduledFutureTask；

这两个类的具体实现见ScheduledThreadPoolExecutor类的分析。