链滴

java.util.concurrent 包分析--Executor

作者：oldcaptain
原文链接：https://ld246.com/article/1403943072438
来源网站：链滴

<h2>一、api中的结构</h2>
java.util.concurrent
接口 Executor
所有已知子接口：
<dl><dd>ExecutorService, ScheduledExecutorService</dd></dl><dl><dt>所有已知实现类：</dt><dd>AbstractExecutorService, ScheduledThreadPoolExecutor, ThreadPoolExecutor</dd><dd>该接口中，主要有一个方法 void execute(Runnable command)，主要用来执行Runnable任务。一般情况是通过Executors.newXXX的方式来返回实现该接口的实例。</dd><dd>实现的该接口的实例有如下几种（创建线池）：</dd><dd>1、</dd><dd>
<pre>public static ExecutorService ewCachedThreadPool()</pre>
缓存线程池可以提高程序性能，因为长时间保持空闲的这种类型的线程池不会占用任何资源调用缓存线程池对象将重用以前构造的线程（线程可用状态），若线程没有可用的，则创建一个新线添加到池中，缓存线程池将终止并从池中移除60秒未被使用的线程
2、</dd><dd>
<pre>public static ScheduledExecutorSevice newSingleThreadScheduledExecutor()

创一个单线程执行程序，它可安排在给定延迟后运行命令或者定期地执行。（注意，如果因为在关闭前执行期间出现失败而终止了此单个线程，那么如果需要，一个新线程会代替它执行后续的任务）。<san style="color: #ff0000;">可保证顺序地执行各个任务，并且在任意给定的时间不会有多个线程是动的。与其他等效的 newScheduledThreadPool(1) 不同，可保证无需重新配置此方法所回的执行程序即可使用其他的线程。</pre>
3、
<pre>public static ScheduledExecutorSevice newScheduledThreadPool(int corePoolSize)

创建一个线程池，它可安排在给定延迟后运行命令或者定期地执行。</pre>
4、
<pre>public static ExecutorService ewFixedThreadPool(int nThreads)

创建一个可重用固定线程数的程池，以共享的无界队列方式来运行这些线程。在任意点，在大多数 nThreads 线程会于处理任务的活动状态。如果在所有线程处于活动状态时提交附加任务，则在有可用线程之前，附加务将在队列中等待。如果在关闭前的执行期间由于失败而导致任何线程终止，那么一个新线程将代替执行后续的任务（如果需要）。在某个线程被显式地 <code>关闭</coe>之前，池中的线程将一直存在。</pre>
5、
<pre>public static ExecutorService ewSingleThreadExecutor(ThreadFactory threadFactory)

创建一个使用单个 worker 线程的 Executor，以无界队列式来运行该线程，并在需要时使用提供的 ThreadFactory 创建新线程。与其他等效的 newFixe

ThreadPool(1, threadFactory) 不同，可保证无需重新配置此方法所返回的执行程序即可使用他的线程。

6、</pre>
我们可以通过ThreadPoolExecutor来创建一个线程池。
<pre>new ThreadPoolExecutor(corePoolSize, maximumPoolSize, keepAliveTime, milliseconds runnableTaskQueue, handler);
</pre>
创建一个线程池需要输入几个参数：

corePoolSize（线程池的基本大小）：当提交一个任务到线程池时，线程池会创建一个线程来执任务，即使其他空闲的基本线程能够执行新任务也会创建线程，等到需要执行的任务数大于线程池基大小时就不再创建。如果调用了线程池的prestartAllCoreThreads方法，线程池会提前创建并启动所基本线程。
runnableTaskQueue（任务队列）：用于保存等待执行的任务的阻塞队列。 可以选择以下几个塞队列。

ArrayBlockingQueue：是一个基于数组结构的有界阻塞队列，此队列按 FIFO（先进先出）原则元素进行排序。
LinkedBlockingQueue：一个基于链表结构的阻塞队列，此队列按FIFO （先进先出） 排序元素吞吐量通常要高于ArrayBlockingQueue。静态工厂方法Executors.newFixedThreadPool()使用了这队列。
SynchronousQueue：一个不存储元素的阻塞队列。每个插入操作必须等到另一个线程调用移操作，否则插入操作一直处于阻塞状态，吞吐量通常要高于LinkedBlockingQueue，静态工厂方法Executors.newCachedThreadPool使用了这个队列。
PriorityBlockingQueue：一个具有优先级的无限阻塞队列。

maximumPoolSize（线程池最大大小）：线程池允许创建的最大线程数。如果队列满了，并且创建的线程数小于最大线程数，则线程池会再创建新的线程执行任务。值得注意的是如果使用了无界任务队列这个参数就没什么效果。
ThreadFactory：用于设置创建线程的工厂，可以通过线程工厂给每个创建出来的线程设置更有义的名字。
RejectedExecutionHandler（饱和策略）：当队列和线程池都满了，说明线程池处于饱和状态那么必须采取一种策略处理提交的新任务。这个策略默认情况下是AbortPolicy，表示无法处理新任时抛出异常。以下是JDK1.5提供的四种策略。

AbortPolicy：直接抛出异常。
CallerRunsPolicy：只用调用者所在线程来运行任务。
DiscardOldestPolicy：丢弃队列里最近的一个任务，并执行当前任务。
DiscardPolicy：不处理，丢弃掉。
当然也可以根据应用场景需要来实现RejectedExecutionHandler接口自定义策略。如记录日志持久化不能处理的任务。

keepAliveTime（线程活动保持时间）：线程池的工作线程空闲后，保持存活的时间。所以如果务很多，并且每个任务执行的时间比较短，可以调大这个时间，提高线程的利用率。
TimeUnit（线程活动保持时间的单位）：可选的单位有天（DAYS），小时（HOURS），分钟MINUTES），毫秒(MILLISECONDS)，微秒(MICROSECONDS, 千分之一毫秒)和毫微秒(NANOSECNDS, 千分之一微秒)。

</dd><dd>-----未完待续-------</dd></dl>