Java线程池

线程是一个操作系统概念。操作系统负责这个线程的创建、挂起、运行、阻塞和终结操作。而操作系统创建线程、切换线程状态、终结线程都要进行CPU调度——这是一个耗费时间和系统资源的事情



Java语言提供两种基础线程池：ScheduledThreadPoolExecutor和ThreadPoolExecutor，可以实现线程资源的复用，都实现了ExecutorService接口，该接口本身和线程池没有直接关系，它的定义更接近执行器，而使用线程管理的方式进行实现只是一种实现方式。

# 1.ThreadPoolExeucotr

其逻辑结构如下图所示：



下面的代码是ThreadPoolExecutor的构造函数：

*ThreadPoolExecutor(*

*int corePoolSize,*

*int maximumPoolSize,*

*long keepAliveTime,*

*TimeUnit unit,*

*BlockingQueue<Runnable> workQueue,*

*ThreadFactory threadFactory,*

*RejectedExecutionHandler handler)*

下面介绍其参数：

* workQueue，调用线程池对象的submit方法或者execute方法，要求线程池运行的任务，但是出于某些原因线程池并没有马上运行这些任务，而是送入一个队列等待执行
* 核心线程（corePoolSize），用于执行任务的核心线程数量，如果不进行特别的设定，线程池中始终保持该数量的线程数
* 非核心线程（maximumPoolSize），一旦任务数量过多，由等待队列的特性决定，线程池将创建非核心线程临时帮助运行任务，[corePoolSize , maximumPoolSize]区间的数量为非核心线程的最大数量。如果某个线程没有运行任何任务，在等待keepAliveTime时间后，这个线程会被销毁
* keepAliveTime和TimeUnit，配合使用，空闲线程等待回收的时间

ThreadPoolExecutor的执行流程如下：

1. 通过线程池提供的submit或者execute方法，要求线程池执行某个任务
   1. 如果当前线程池中运行的线程数量还没有达到corePoolSize大小时，线程池会创建一个新的线程运行该任务，无论之前创建的线程是否处于空闲状态
   2. 如果线程池中的运行的线程数量达到corePoolSize，将该任务加入到等待队列中，直到某一个线程空闲，线程池根据设置的等待队列规则，从队列中取出一个新的任务执行
   3. 根据队列规则，如果无法加入队列，则线程池创建一个非核心线程直接运行这个任务
   4. 其他情况时，线程池拒绝接受该任务，抛出RejectedExecutionException
2. 线程池中某个线程完成任务的执行，会试图到等到任务队列中取出下一个等待任务，调用等待队列的poll方法
3. 线程池中的线程数目超过corePoolSize，当某个线程处理完成后，等待keepAliveTime没有新任务分配给它，则回收该线程

## 1.1 prestartAllCoreThreads

当线程池中的线程还没有达到你设置的corePoolSize参数值的时候，如果有新的任务到来，线程池将创建新的线程运行这个任务，无论之前已经创建的线程是否处于空闲状态。

prestartAllCoreThreads方法，在还没有接收到任何任务的情况下，先创建符合corePoolSize参数值的线程数，执行示例：

*ThreadPoolExecutor poolExecutor =new*

*ThreadPoolExecutor(5,10,1,TimeUnit.MINUTES,*

*new ArrayBlockingQueue<Runnable>(1));*

*poolExecutor.prestartAllCoreThreads();*

## 1.2 线程池的等待队列

使用ThreadPoolExecutor线程池的时候，指定一个实现BlockingQueue接口的任务等待队列，有三种队列：SynchronousQueue、LinkedBlockingQueue和ArrayBlockingQueue。

DelayQueue是一个无界的BlockingQueue，用于放置实现Delay接口的对象，其中对象只能在到期时才能从队列中取走。该队列有序，对头对象的延迟到期时间最长。典型的场景：

* 淘宝订单业务，下单之后如果三十分钟之内没有付款就自动取消订单
* 饿了吗订餐通知，下单成功后60s之后给用户发送短信通知

这些业务的特点是需要延迟工作。

## 1.3 RejectedExecutionHandler

在ThreadPoolExecutor线程池中还有一个重要接口：RejectExecutionHandler，当提交给线程池中的某一个新任务无法直接被线程池直接处理，又无法加入等待队列，也无法创建新的线程执行。又或者线程池已经调用shutdown()方法停止了工作；又或者线程池不是处于正常的工作状态；这时候ThreadPoolExecutor线程池会拒绝处理这个任务，触发创建ThreadPoolExecutor线程池时定义的RejectedExecutionHandler接口的实现。其提供四种可以直接使用的RejectedExecutionHandler接口的实现：

* CallerRunsPolicy，这个拒绝处理器，将直接运行这个任务的run方法。但是，请注意并不是在ThreadPoolExecutor线程池中的线程中运行，而是直接调用这个任务实现的run方法。
* AbortPolicy，这个处理器，在任务被拒绝后会创建一个RejectedExecutionException异常并抛出。这个处理过程也是ThreadPoolExecutor线程池默认的RejectedExecutionHandler实现
* DiscardPolicy，DiscardPolicy处理器，将会默默丢弃这个被拒绝的任务，不会抛出异常，也不会通过其他方式执行这个任务的任何一个方法，更不会出现任何的日志提示。
* DiscardOldestPolicy，它会检查当前ThreadPoolExecutor线程池的等待队列。并调用队列的poll()方法，将当前处于等待队列列头的等待任务强行取出，然后再试图将当前被拒绝的任务提交到线程池执行

https://www.journaldev.com/1069/threadpoolexecutor-java-thread-pool-example-executorservice

https://blog.csdn.net/lipc\_/article/details/52025993