java並發庫之Executors常用的創建ExecutorService的幾個方法說明 - 文星的個人空間 - 開源中國社區

一、線程池的創建

我們可以通過ThreadPoolExecutor來創建一個線程池。

```
new  ThreadPoolExecutor(corePoolSize, maximumPoolSize, keepAliveTime,
milliseconds,runnableTaskQueue, handler);
```

創建一個線程池需要輸入幾個參數：

- corePoolSize（線程池的基本大小）：當提交一個任務到線程池時，線程池會創建一個線程來執行任務，即使其他空閒的基本線程能夠執行新任務也會創建線程，等到需要執行的任務數大於線程池基本大小時就不再創建。如果調用了線程池的prestartAllCoreThreads方法，線程池會提前創建並啟動所有基本線程。
- runnableTaskQueue（任務隊列）：用於保存等待執行的任務的阻塞隊列。 可以選擇以下幾個阻塞隊列。
    - ArrayBlockingQueue：是一個基於數組結構的有界阻塞隊列，此隊列按 FIFO（先進先出）原則對元素進行排序。
    - LinkedBlockingQueue：一個基於鏈表結構的阻塞隊列，此隊列按FIFO （先進先出） 排序元素，吞吐量通常要高於ArrayBlockingQueue。靜態工廠方法Executors.newFixedThreadPool()使用了這個隊列。
    - SynchronousQueue：一個不存儲元素的阻塞隊列。每個插入操作必須等到另一個線程調用移除操作，否則插入操作一直處於阻塞狀態，吞吐量通常要高於LinkedBlockingQueue，靜態工廠方法Executors.newCachedThreadPool使用了這個隊列。
    - PriorityBlockingQueue：一個具有優先級的無限阻塞隊列。
- maximumPoolSize（線程池最大大小）：線程池允許創建的最大線程數。如果隊列滿了，並且已創建的線程數小於最大線程數，則線程池會再創建新的線程執行任務。值得注意的是如果使用了無界的任務隊列這個參數就沒什麼效果。
- ThreadFactory：用於設置創建線程的工廠，可以通過線程工廠給每個創建出來的線程設置更有意義的名字。
- RejectedExecutionHandler（飽和策略）：當隊列和線程池都滿了，說明線程池處於飽和狀態，那麼必須採取一種策略處理提交的新任務。這個策略默認情況下是AbortPolicy，表示無法處理新任務時拋出異常。以下是JDK1.5提供的四種策略。
    - AbortPolicy：直接拋出異常。
    - CallerRunsPolicy：只用調用者所在線程來運行任務。
    - DiscardOldestPolicy：丟棄隊列裡最近的一個任務，並執行當前任務。
    - DiscardPolicy：不處理，丟棄掉。
    - 當然也可以根據應用場景需要來實現RejectedExecutionHandler接口自定義策略。如記錄日誌或持久化不能處理的任務。
- keepAliveTime（線程活動保持時間）：線程池的工作線程空閒後，保持存活的時間。所以如果任務很多，並且每個任務執行的時間比較短，可以調大這個時間，提高線程的利用率。

- TimeUnit（線程活動保持時間的單位）：可選的單位有天（DAYS），小時（HOURS），分鐘（MINUTES），毫秒(MILLISECONDS)，微秒(MICROSECONDS, 千分之一毫秒)和毫微秒(NANOSECONDS, 千分之一微秒)。

二、Executors提供了一些方便創建ThreadPoolExecutor的常用方法，主要有以下幾個：

1、 Executors.newFixedThreadPool(int nThreads);創建固定大小(nThreads,大小不能超過int的最大值)的線程池

//線程數量

int nThreads = 20;

//創建executor 服務

ExecutorService executor = Executors.newFixedThreadPool(nThreads) ;

重載後的版本，需要多傳入實現了ThreadFactory接口的對象。

ExecutorService executor = Executors. newFixedThreadPool(nThreads,threadFactory);

說明：創建固定大小(nThreads,大小不能超過int的最大值)的線程池，緩衝任務的隊列為LinkedBlockingQueue,大小為整型的最大數，當使用此線程池時，在同執行的任務數量超過傳入的線程池大小值後，將會放入LinkedBlockingQueue，在LinkedBlockingQueue中的任務需要等待線程空閒後再執行，如果放入LinkedBlockingQueue中的任務超過整型的最大數時，拋出RejectedExecutionException。

2、Executors.newSingleThreadExecutor()：創建大小為1的固定線程池。

ExecutorService executor = Executors.newSingleThreadExecutor();

重載後的版本，需要多傳入實現了ThreadFactory接口的對象。

ExecutorService executor = Executors. newSingleThreadScheduledExecutor(ThreadFactory threadFactory)

說明：創建大小為1的固定線程池，同時執行任務(task)的只有一個,其它的（任務）task都放在LinkedBlockingQueue中排隊等待執行。

3、**Executors**.**newCachedThreadPool**()；創建corePoolSize為0，最大線程數為整型的最大數，線程keepAliveTime為1分鐘，緩存任務的隊列為SynchronousQueue的線程池。

ExecutorService executor = Executors.newCachedThreadPool()；

當然也可以以下面的方式創建，重載後的版本，需要多傳入實現了ThreadFactory接口的對象。

ExecutorService executor = Executors.newCachedThreadPool(ThreadFactory threadFactory) ;

說明：使用時，放入線程池的task任務會復用線程或啟動新線程來執行，注意事項：啟動的線程數如果超過整型最大值後會拋出RejectedExecutionException異常，啟動後的線程存活時間為一分鐘。

4、Executors.newScheduledThreadPool(int corePoolSize):創建corePoolSize大小的線程池。

//線程數量

int corePoolSize= 20;

//創建executor 服務

ExecutorService executor = Executors.newScheduledThreadPool(corePoolSize) ;

重載後的版本，需要多傳入實現了ThreadFactory接口的對象。

ExecutorService executor = Executors.newScheduledThreadPool(corePoolSize, threadFactory) ；

說明：線程keepAliveTime為0，緩存任務的隊列為DelayedWorkQueue，注意不要超過整型的最大值。