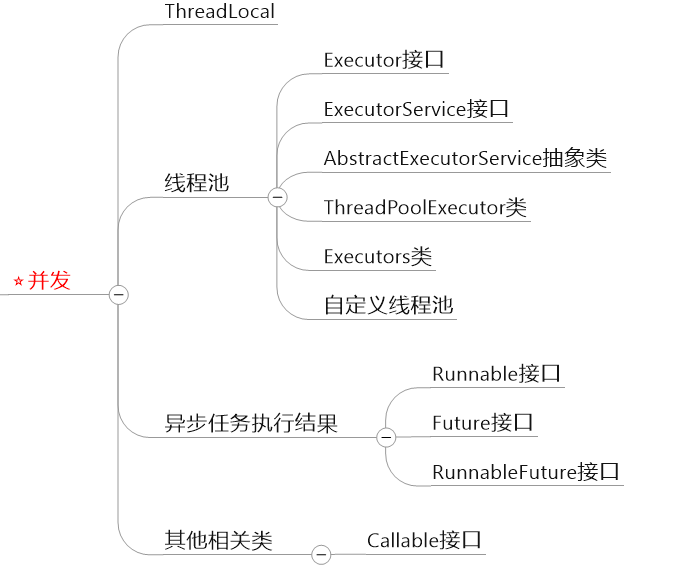
# 自定义线程池

## 前言



花了几天时间去学习和了解上图中展示的线程池相关的类，通过查看Executors类，我明白了为什么《阿里巴巴Java开发规范》中不让开发者不要使用Executors类中提供的工厂方法获取线程池，而是让开发者自己定义线程池。

## 正文

### Executors类中工厂方法概述

查看Executors类的源码，能够发现Executors类中的工厂方法最终通过ThreadPoolExecutor类的构造函数创建线程池：

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

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

### ThreadPoolExecutor类构造函数概述

ThreadPoolExecutor类的构造函数总共有4个：

**Constructor：**

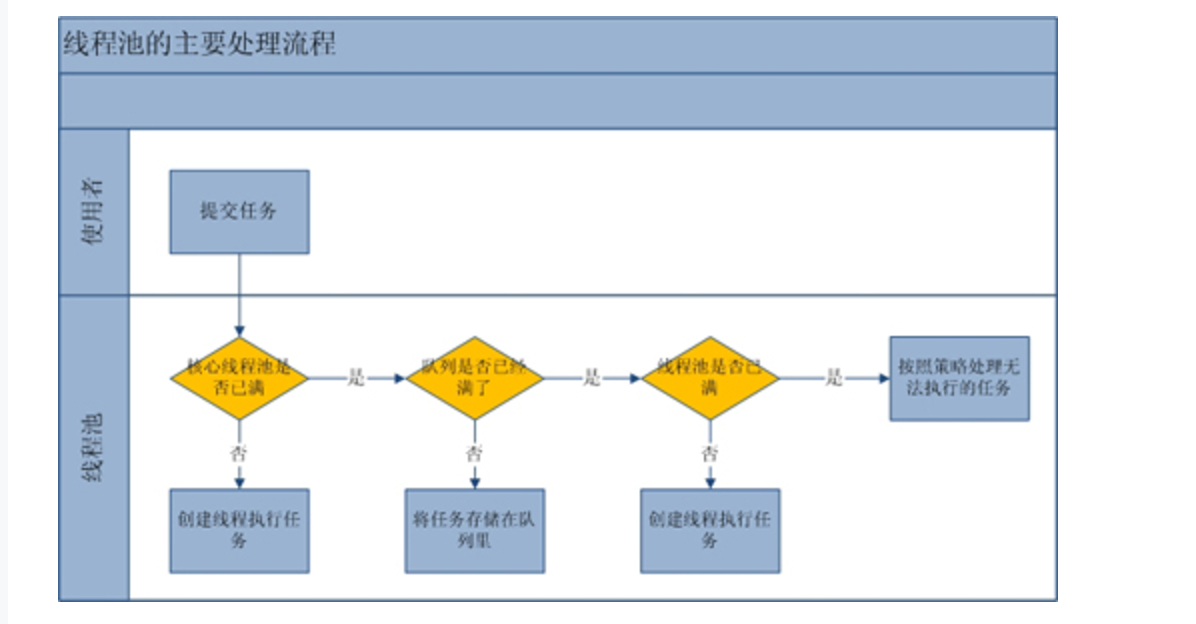
|  |
| --- |
| 1  public ThreadPoolExecutor*(*int corePoolSize,int maximumPoolSize,long keepAliveTime,TimeUnit unit,BlockingQueue*<*Runnable*>* workQueue*)* |
| 2  public ThreadPoolExecutor*(*int corePoolSize,int maximumPoolSize,long keepAliveTime,TimeUnit unit,BlockingQueue*<*Runnable*>* workQueue,  ThreadFactory threadFactory*)* |
| 3  public ThreadPoolExecutor*(*int corePoolSize,int maximumPoolSize,long keepAliveTime,TimeUnit unit,BlockingQueue*<*Runnable*>* workQueue,  RejectedExecutionHandler handler*)* |
| 4  public ThreadPoolExecutor*(*int corePoolSize,int maximumPoolSize,long keepAliveTime,TimeUnit unit,BlockingQueue*<*Runnable*>* workQueue,  ThreadFactory threadFactory,RejectedExecutionHandler handler*)* |

查看源码发现前3个构造函数最终调用的是第4个构造函数，因此我们先看看构造函数各参数的意义：

**Parameters and Descriptions:**

|  |  |
| --- | --- |
| corePoolSize | the number of threads to keep in the pool, even if they are idle, unless allowCoreThreadTimeOut is set  除非设置了allowCoreThreadTimeOut参数，否则线程池中将维持corePoolSize个线程，即使有些线程是空闲的。 |
| maximumPoolSize | the maximum number of threads to allow in the pool  池中最多允许maximum个线程存在。 |
| keepAliveTime | when the number of threads is greater than the core, this is the maximum time that excess idle threads will wait for new tasks before terminating.  当池中的线程数超过了corePoolSize,线程的空闲时间不能超过keepAliveTime。 |
| unit | the time unit for the keepAliveTime argument  keepAliveTime参数的单位。 |
| workQueue | the queue to use for holding tasks before they are executed. This queue will hold only the Runnable tasks submitted by the execute method.  暂存待执行任务的队列，该队列只能暂存通过execute方法提交的任务。 |
| threadFactory | the factory to use when the executor creates a new thread.  executor方法通过threadFactory创建线程。 |
| handler | the handler to use when execution is blocked because the thread bounds and queue capacities are reached  因线程边界和队列容量问题而拒绝执行任务时的处理函数。 |

### 线程池主要处理流程



1. 当客户端向线程池提交一个任务，如果此时池中线程数不足corePoolSize，则创建一个新的线程处理该任务，即使此时池中存在空闲线程。
2. 如果池中线程数超过corePoolSize，则尝试将任务暂存到workQueue，如果暂存成功，待排在前面的任务开始执行且池中存在空闲线程，将分配一个空闲线程执行该任务
3. 如果暂存失败，则进一步判断线程池中线程数是否达到maximumPoolSize，如果尚未达到，则创建一个新线程执行任务，如果已经达到，则拒绝执行该任务，并且启动设置的处理程序。
4. 如果线程池中线程数量超过corePoolSize，则清除空闲时间超过keepAliveTime的线程，直到线程中的线程数不大于corePoolSize为止。

以下为网上看到的一个例子：

公司要设立一个项目组来处理某些任务，hr部门给的人员编制是10个人（corePoolSize），同时安排了一间有15个座位（maximumPoolSize）的办公室。最开始的时候来一个任务，就招聘一个人，就这样一个一个的招聘，招满十个人。招满10个人后先暂停招人，后续提交的任务如果如果没空做就先积攒着，当任务积攒到一定程度，10个人可能做不完，于是开始招一些临时工，