Executors.newCachedThreadPool()

Executors.newSingleThreadPool() newfixedThreadPool()

Executors.newScheduleThreadPool() 这个会周期性的执行任务线程池的任务。

ScheduledExecutorService scheduledThreadPool = Executors.newScheduledThreadPool(5);

 scheduledThreadPool.scheduleAtFixedRate(new Runnable() {

public void run() {

System.out.println("delay 1 seconds, and excute every 3 seconds");

   }

  }, 1, 3, TimeUnit.SECONDS);

这个在run方法里加时间，表示延迟一秒后每3秒后执行一次。

上面这几个线程池的创建都会调用到这个方法，

return new ThreadPoolExecutor(0, Integer.MAX\_VALUE,  
 60L, TimeUnit.SECONDS,  
 new SynchronousQueue<Runnable>()，xx，xxx);

也就是会返回一个ThreadPoolExecutor()对象，通过构造函数创建的，也就是线程池的东西都在ThreadPoolExecutor

**注意不要忘了线程工厂也是7剑之一**

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

七个参数其实指的是标黄的这7个

corePollSize：核心线程数。在创建了线程池后，线程中没有任何线程，等到有任务到来时才创建线程去执行任务。默认情况下，在创建了线程池后，线程池中的线程数为0，当有任务来之后，就会创建一个线程去执行任务，当线程池中的线程数目达到corePoolSize后，就会把到达的任务放到缓存队列当中。

每个变量的作用都已经标明出来了，这里要重点解释一下corePoolSize、maximumPoolSize、largestPoolSize三个变量。

　　corePoolSize在很多地方被翻译成核心池大小，其实我的理解这个就是线程池的大小。举个简单的例子：

　　假如有一个工厂，工厂里面有10个工人，每个工人同时只能做一件任务。

　　因此只要当10个工人中有工人是空闲的，来了任务就分配给空闲的工人做；

　　当10个工人都有任务在做时，如果还来了任务，就把任务进行排队等待；

　　如果说新任务数目增长的速度远远大于工人做任务的速度，那么此时工厂主管可能会想补救措施，比如重新招4个临时工人进来；

　　然后就将任务也分配给这4个临时工人做；

　　如果说着14个工人做任务的速度还是不够，此时工厂主管可能就要考虑不再接收新的任务或者抛弃前面的一些任务了。

　　当这14个工人当中有人空闲时，而新任务增长的速度又比较缓慢，工厂主管可能就考虑辞掉4个临时工了，只保持原来的10个工人，毕竟请额外的工人是要花钱的。

　　这个例子中的corePoolSize就是10，而maximumPoolSize就是14（10+4）。

　　也就是说corePoolSize就是线程池大小，maximumPoolSize在我看来是线程池的一种补救措施，即任务量突然过大时的一种补救措施。

　　不过为了方便理解，在本文后面还是将corePoolSize翻译成核心池大小。

　　largestPoolSize只是一个用来起记录作用的变量，用来记录线程池中曾经有过的最大线程数目，跟线程池的容量没有任何关系。

maximumPoolSize：最大线程数。表明线程中最多能够创建的线程数量。

keepAliveTime：空闲的线程保留的时间。

TimeUnit：空闲线程的保留时间单位。

TimeUnit.DAYS; //天

TimeUnit.HOURS; //小时

TimeUnit.MINUTES; //分钟

TimeUnit.SECONDS; //秒

TimeUnit.MILLISECONDS; //毫秒

TimeUnit.MICROSECONDS; //微妙

TimeUnit.NANOSECONDS; //纳秒

BlockingQueue<Runnable>：阻塞队列，存储等待执行的任务。参数有ArrayBlockingQueue、LinkedBlockingQueue、SynchronousQueue可选。

ThreadFactory：线程工厂，用来创建线程

RejectedExecutionHandler：队列已满，而且任务量大于最大线程的异常处理策略。有以下取值