常见线程池：

## **newCachedThreadPool**

创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。

这种类型的线程池特点是：

* 工作线程的创建数量几乎没有限制(其实也有限制的,数目为Interger. MAX\_VALUE), 这样可灵活的往线程池中添加线程。
* 如果长时间没有往线程池中提交任务，即如果工作线程空闲了指定的时间(默认为1分钟)，则该工作线程将自动终止。终止后，如果你又提交了新的任务，则线程池重新创建一个工作线程。
* 在使用CachedThreadPool时，一定要注意控制任务的数量，否则，由于大量线程同时运行，很有会造成系统瘫痪

## ****newFixedThreadPool****

创建一个指定工作线程数量的线程池。每当提交一个任务就创建一个工作线程，如果工作线程数量达到线程池初始的最大数，则将提交的任务存入到池队列中。

FixedThreadPool是一个典型且优秀的线程池，它具有线程池提高程序效率和节省创建线程时所耗的开销的优点。但是，在线程池空闲时，即线程池中没有可运行任务时，它不会释放工作线程，还会占用一定的系统资源。

## ****newSingleThreadExecutor****

创建一个单线程化的Executor，即只创建唯一的工作者线程来执行任务，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。如果这个线程异常结束，会有另一个取代它，保证顺序执行。单工作线程最大的特点是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活动的。

## ****newScheduleThreadPool****

创建一个定长的线程池，而且支持定时的以及周期性的任务执行，支持定时及周期性任务执行。

一、核心参数

corePoolSize（核心线程数）

（1）核心线程会一直存在，即使没有任务执行；

（2）当线程数小于核心线程数的时候，即使有空闲线程，也会一直创建线程直到达到核心线程数；

（3）设置allowCoreThreadTimeout=true（默认false）时，核心线程会超时关闭。

queueCapacity（任务队列容量）

也叫阻塞队列，当核心线程都在运行，此时再有任务进来，会进入任务队列，排队等待线程执行。

maxPoolSize（最大线程数）

（1）线程池里允许存在的最大线程数量；

（2）当任务队列已满，且线程数量大于等于核心线程数时，会创建新的线程执行任务；

（3）线程池里允许存在的最大线程数量。当任务队列已满，且线程数量大于等于核心线程数时，会创建新的线程执行任务。

keepAliveTime（线程空闲时间）

（1）当线程空闲时间达到keepAliveTime时，线程会退出（关闭），直到线程数等于核心线程数；

（2）如果设置了allowCoreThreadTimeout=true，则线程会退出直到线程数等于零。

allowCoreThreadTimeout（允许核心线程超时）

rejectedExecutionHandler（任务拒绝处理器）

（1）当线程数量达到最大线程数，且任务队列已满时，会拒绝任务；

（2）调用线程池shutdown()方法后，会等待执行完线程池的任务之后，再shutdown()。如果在调用了shutdown()方法和线程池真正shutdown()之间提交任务，会拒绝新任务。

二、线程池参数默认值

corePoolSize = 1

queueCapacity = Integer.MAX\_VALUE

maxPoolSize = Integer.MAX\_VALUE

keepAliveTime = 60秒

allowCoreThreadTimeout = false

rejectedExecutionHandler = AbortPolicy()

三、ThreadPoolExecutor（线程池）执行顺序

当线程数小于核心线程数时，会一直创建线程直到线程数等于核心线程数；

当线程数等于核心线程数时，新加入的任务会被放到任务队列等待执行；

当任务队列已满，又有新的任务时，会创建线程直到线程数量等于最大线程数；

当线程数等于最大线程数，且任务队列已满时，新加入任务会被拒绝。

————————————————

版权声明：本文为CSDN博主「凉凉的西瓜」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_42815754/article/details/84669545