newFixedThreadPool()方法:该方法返回一个固定线程数量的线程池,该线程池中的线程数量始终不变.

newSingleTHreadExecutor()方法:该方法返回一个只有一个线程的线程池

newCachedThreadPool()方法:该方法返回一个可根据实际情况调整线程数量的线程池

newSingleThreadScheduledExecutor()方法:该方法返回一个ScheduledExecutorService对象,线程池大小为1;

newScheduledThreadPool()方法:该方法也返回一个ScheduledExecutorService对象,但是该线程池可以指定线程数量

计划任务

newScheduledThreadPool(),它返回一个ScheduledExecutorService对象,可以根据时间需要对线程进行调度,方法scheduleAtFixedRate()方法和scheduleWithFixedDelay()会对任务进行周期性的调度,但是有区别

scheduleAtFixedRate:创建一个周期任务,任务开始于给定的初始延时.后续的任务按照给定的周期进行,后续第一个任务将在initialDelay+period时执行,后续第二个任务会在initialDelay+2*period时执行

scheduleWithFixedDelay:创建一个周期性任务,任务开始于初始延时时间,后续任务会按照给定的延时进行,即上一个任务的结束时间到下一个任务的开始时间的时间差;

创建线程池的关键参数:

```
new ThreadPoolExecutor(int corePoolSize,
                       int maximumPoolSize,
                       long keepAliveTime,
                      TimeUnit unit,
                      BlockingQueue<Runnable> workQueue,
                      ThreadFactory threadFactory,
                      RejectedExecutionHandler handler)
```

corePoolSize:指定了线程池中的线程数量

maximumPoolSize:制定了线程中的最大线程数量

keepAliveTime:当线程池数量超过corePoolSize时,多余的空闲线程的存活时间,即超过corePoolSize的空闲线程,在多长时间内会被销毁

unit:keepAliveTime的时间单位

workQueue:任务队列,被提交但尚未被执行的任务

threadFactory:线程工厂,用于创建线程,一般用默认的即可

handler:拒绝策略,当任务太多来不及处理时,如何拒绝任务.

JDK内置的拒绝策略:

AbortPolicy策略:该策略会直接抛出异常,阻止系统正常工作

CallerRunsPolicy策略:只要线程池未关闭,该策略直接在调用者线程中,运行当前被丢弃的任务,显然这样不会真的丢弃任务,但是,任务提交线程的性能极有可能会急剧下降

DiscardOldestPolicy策略:该策略将丢弃最老的一个请求,也就是即将被执行的一个任务,并尝试再次提交当前任务.

DiscardPolicy策略:该策略默默地丢弃无法处理地任务,不予任何处理.

线程工厂:

Fork/Join框架

ForkJoinPool线程池,可以向ForkJoin线程池,提交一个ForkJoin任务.所谓ForkJoinTask任务就是RecursiveAction类和RecursiveTask类.它们分别表示没有返回值的任务和可以携带返回值的任务