线程池

什么是线程池?

- 1.线程的创建需要开辟内存空间：比如本地方法栈、虚拟机栈、程序计数器等线程私有变量的内存频繁的创建和消耗会带来一定的性能开销，大量的同类线程还会导致内存消耗完或者过度切换。
- 2.而使用池化技术来管理和使用线程，叫做线程池

意义

- 1.线程池避免了线程频繁的创建和消耗带来的性能开销，
- 2.可以很好的管理任务和友好的拒绝任务，以及执行一些定时任务。

方法

执行

1.submit

- 只能执行 Runnable 任务（无返回值）

2.execute

- 既能执行 Runnable 任务，也能执行 Callable 有返回值任务

注意

- execute 执行任务如果有 OOM 异常会将异常打印到控制台； submit 执行任务出现了 OOM 异常时不会打印异常

关闭

1.shutdown

- 拒绝执行新任务加入，等待线程池中的任务队列执行完之后，再停止线程池

2.shutdownNow

- 拒绝执行新任务，不会等待任务队列中的任务执行完成，就会停止线程池

- 子主题 3

状态

- RUNNING：线程池创建之后的初始状态，这种状态下可以执行任务
- SHUTDOWN：该状态下线程池不再接受新任务，但是会处理工作队列种的任务
- STOP：调用了 shutdownNow 时，该状态下线程池不再接受新任务，并且不会处理工作队列种的任务，并且会中断线程
- TIDYING：该状态下所有任务都已终止，将会执行 terminated() 钩子方法
- TERMINATED: 销毁状态，执行完 terminated() 钩子方法

- TERMINATED: 销毁状态，执行完 terminated() 钩子方法

创建

1. 创建固定的线程池（任务数取向无穷大、不建议使用）
   - ExecutorService service =
     Executors.newFixedThreadPool();
2. 创建带缓存的线程池（根据任务数量生成对应的线程数，所以它适合用于短期大量任务）
   - ExecutorService service =
     Executors.newCachedThreadPool();
3. 创建可以执行定时任务的线程池
   - ScheduledExecutorService service =
     Executors.newScheduledThreadPool()
4. 创建单个执行定时任务的线程池
   - ScheduledExecutorService service =
     Executors.newSingleThreadScheduledExecutor();
5. 创建单个线程池
   - ExecutorService service =
     Executors.newSingleThreadExecutor();
6. 根据当前的工作环境(cup核心数，任务量)创建线程，异步线程池)
   - ExecutorService service =
     Executors.newWorkStealingPool();
7. 原始创建方法
   - ThreadPoolExecutor

ThreadPoolExecutor 七大参数

- 核心线程数（正式员工） int corePoolSize
- 最大线程数（正式员工和临时工） int maximumPoolSize
- 生存周期 （临时工） long keepAliveTime
- 时间单位 TimeUnit unit
- 任务队列 BlockingQueue<Runnable> workQueue
- 线程工厂 ThreadFactory threadFactory
- 拒绝策略 RejectedExecutionHandler handler

拒绝策略

1. 默认拒绝策略（拒绝并抛出异常）

- 1.默认拒绝策略（拒绝并抛出异常）
  - new ThreadPoolExecutor.AbortPolicy()
- 2.使用调用线程池的线程来执行任务
  （使用主线程来执行任务）
  - new ThreadPoolExecutor.CallerRunsPolicy()
- 3.忽略新任务
  - new ThreadPoolExecutor.DiscardPolicy()
- 4.忽略老任务
  - new ThreadPoolExecutor.DiscardOldestPolicy()

- 单个线程的线程池有什么意义？
  - 1.无需频繁的创建和销毁线程
  - 2.因为线程池有任务队列，可以更好的分配和管理以及存储任务

线程池的执行流程

- 1.当任务数小于核心线程数时，创建线程
- 2.当任务数大于等于核心线程数，且任务队列未满时，将任务放入任务队列
- 3.当任务数大于等于核心线程数，且任务队列已满
  - 若任务数小于最大线程数，创建线程
  - 若任务数等于最大线程数，拒绝任务