为什么要用线程池

网易云课堂

线程是不是越多越好?

1、线程在java中是一个对象，更是操作系统的资源，线程创建、销毁需要时间。如果创建时间+ 销毁时间 > 执行任务时间 就很不合算。

2、java对象占用堆内存，操作系统线程占用系统内存，根据jvm规范，一个线程默认最大栈大小1M，这个栈空间是需要从系统内存中分配的。线程过多，会消耗很多的内存。

3、操作系统需要频繁切换线程上下文（大家都想被运行），影响性能。

线程池的推出，就是为了方便的控制线程数量。

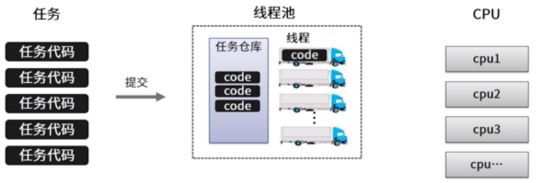
注：本节课代码参考Demo9.java

网易云课堂 × 微专业

线程池原理 – 概念

网易云课堂

1、线程池管理器：用于创建并管理线程池，包括创建线程池，销毁线程池，添加新任务；
2、工作线程：线程池中线程，在没有任务时处于等待状态，可以循环的执行任务；
3、任务接口：每个任务必须实现的接口，以供工作线程调度任务的执行，它主要规定了任务的入口，任务执行完后的收尾工作，任务的执行状态等；
4、任务队列：用于存放没有处理的任务。提供一种缓冲机制。



04:23 / 53:08

线程池API – 接口定义和实现类

网易云课堂

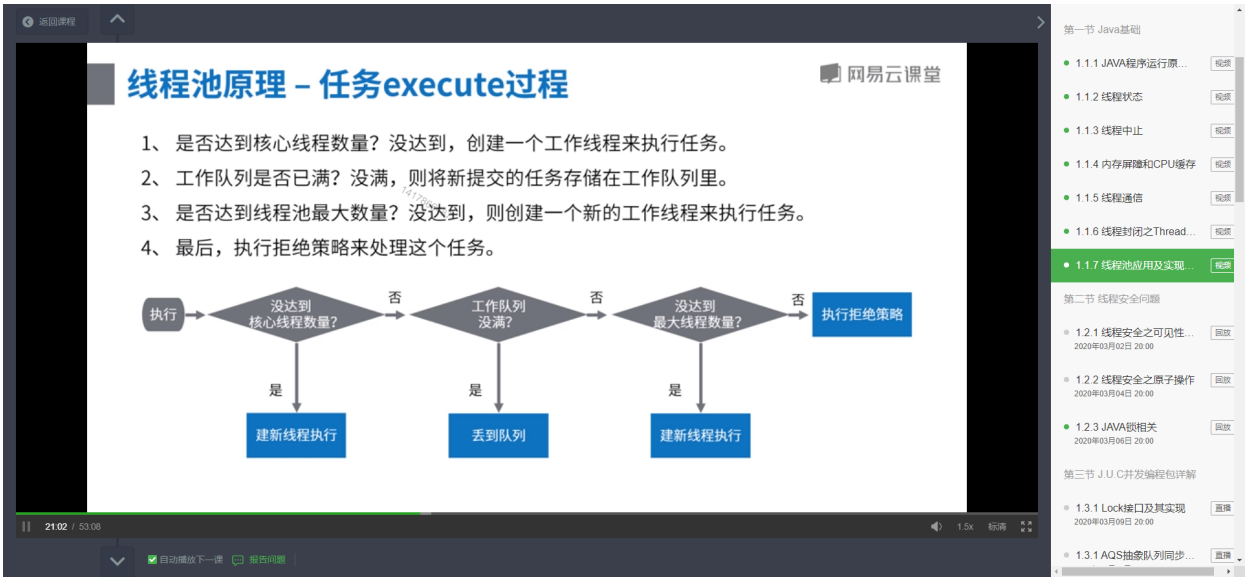
类型	名称	描述
接口	Executor	最上层的接口，定义了执行任务的方法execute
接口	ExecutorService	继承了Executor接口，拓展了Callable 、Future、关闭方法
接口	ScheduledExecutorService	继承了ExecutorService，增加了定时任务相关的方法
实现类	ThreadPoolExecutor	基础、标准的线程池实现
实现类	ScheduledThreadPoolExecutor	继承了ThreadPoolExecutor，实现了ScheduledExecutorService中相关定时任务的方法

04:38 / 53:08


线程池API –方法定义
网易云课堂
ExecutorService
// 监测ExecutorService是否已经关闭，直到所有任务完成执行，或超时发生，或当前线程被中断
awaitTermination(long timeout, TimeUnit unit)
// 执行给定的任务集合，执行完毕后，返回结果
invokeAll(Collection<? extends Callable<T>> tasks)
// 执行给定的任务集合，执行完毕或者超时后，返回结果，其他任务终止
invokeAll(Collection<? extends Callable<T>> tasks, long timeout, TimeUnit unit)
// 执行给定的任务，任意一个执行成功则返回结果，其他任务终止
invokeAny(Collection<? extends Callable<T>> tasks)
// 执行给定的任务，任意一个执行成功或者超时后，则返回结果，其他任务终止
invokeAny(Collection<? extends Callable<T>> tasks, long timeout, TimeUnit unit)
// 如果此线程池已关闭，则返回true。
isShutdown()
// 如果关闭后所有任务都已完成，则返回true。
isTerminated()
// 优雅关闭线程池，之前提交的任务将被执行，但是不会接受新的任务。
shutdown()
// 尝试停止所有正在执行的任务，停止等待任务的处理，并返回等待执行任务的列表。
shutdownNow()
// 提交一个用于执行的Callable返回任务，并返回一个Future，用于获取Callable执行结果
submit(Callable<T> task)
// 提交可运行任务以执行，并发回一个Future对象，执行结果为null
submit(Runnable task)
// 提交可运行任务以执行，并返回Future，执行结果为传入的result
submit(Runnable task, T result)
06:54 / 53:08
第一节 Java基础
1.1.1 JAVA程序运行原...
1.1.2 线程状态
1.1.3 线程中止
1.1.4 内存屏障和CPU缓存
1.1.5 线程通信
1.1.6 线程封闭之Thread...
1.1.7 线程池应用及实现...
第二节 线程安全问题
1.2.1 线程安全之可见性...
1.2.2 线程安全之原子操作
1.2.3 JAVA锁相关
第三节 J.U.C并发编程包详解
1.3.1 Lock接口及其实现
1.3.1 AQS抽象队列同步...


线程池API –方法定义
网易云课堂
ScheduledExecutorService
schedule(Callable<V> callable, long delay, TimeUnit unit)
schedule(Runnable command, long delay, TimeUnit unit)
创建并执行一个一次性任务
过了延迟时间就会被执行
scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit)
scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit)
创建并执行一个周期性任务
过了初始延迟时间，第一次被执行，后续以给定的周期时间执行
执行过程中发生了异常，那么任务就停止
一次任务执行时长超过了周期时间，下一次任务会在该次任务执行结束的时间基础上，计算执行延时。
对于超过周期的长时间处理任务的不同处理方式，这是它和scheduleAtFixedRate的重要区别。
创建并执行一个周期性任务
过了给定的初始延迟时间，会第一 次被执行
执行过程中发生了异常，那么任务就停止
一次任务执行时长超过了周期时间，下一次任务会等到该次任务执行结束后，立刻执行，这也是它和scheduleWithFixedDelay的重要区别。
此处结合代码示例进行理解即可！
08:38 / 53:08
第一节 Java基础
1.1.1 JAVA程序运行原...
1.1.2 线程状态
1.1.3 线程中止
1.1.4 内存屏障和CPU缓存
1.1.5 线程通信
1.1.6 线程封闭之Thread...
1.1.7 线程池应用及实现...
第二节 线程安全问题
1.2.1 线程安全之可见性...
1.2.2 线程安全之原子操作
1.2.3 JAVA锁相关
第三节 J.U.C并发编程包详解
1.3.1 Lock接口及其实现
1.3.1 AQS抽象队列同步...


线程池API – Executors工具类
网易云课堂
你可以自己实例化线程池，也可以用Executors 创建线程池的工厂类，常用方法如下：
newFixedThreadPool(int nThreads) 创建一个固定大小、任务队列容量无界的线程池。核心线程数=最大线程数。
newCachedThreadPool() 创建的是一个大小无界的缓冲线程池。它的任务队列是一个同步队列。任务加入到池中，如果池中有空闲线程，则用空闲线程执行，如无则创建新线程执行。池中的线程空闲超过60秒，将被销毁释放。线程数随任务的多少变化。适用于执行耗时较小的异步任务。池的核心线程数=0 ，最大线程数= Integer.MAX_VALUE
newSingleThreadExecutor() 只有一个线程来执行无界任务队列的单一线程池。该线程池确保任务按加入的顺序一个一个依次执行。当唯一的线程因任务异常中止时，将创建一个新的线程来继续执行后续的任务。与newFixedThreadPool(1)的区别在于，单一线程池的池大小在newSingleThreadExecutor方法中硬编码，不能再改变的。
newScheduledThreadPool(int corePoolSize) 能定时执行任务的线程池。该池的核心线程数由参数指定，最大线程数= Integer.MAX_VALUE
18:40 / 53:08
第一节 Java基础
1.1.1 JAVA程序运行原...
1.1.2 线程状态
1.1.3 线程中止
1.1.4 内存屏障和CPU缓存
1.1.5 线程通信
1.1.6 线程封闭之Thread...
1.1.7 线程池应用及实现...
第二节 线程安全问题
1.2.1 线程安全之可见性...
1.2.2 线程安全之原子操作
1.2.3 JAVA锁相关
第三节 J.U.C并发编程包详解
1.3.1 Lock接口及其实现
1.3.1 AQS抽象队列同步...


返回课程
线程池原理 – 任务execute过程
网易云课堂
1、 是否达到核心线程数量？没达到，创建一个工作线程来执行任务。
2、 工作队列是否已满？没满，则将新提交的任务存储在工作队列里。
3、 是否达到线程池最大数量？没达到，则创建一个新的工作线程来执行任务。
4、 最后，执行拒绝策略来处理这个任务。
执行
没达到核心线程数量?
否
工作队列没满?
否
没达到最大线程数量?
否
执行拒绝策略
是
建新线程执行
是
丢到队列
是
建新线程执行
21:02 / 53:08
1.5x
标清
自动播放下一课
报告问题
第一节 Java基础
1.1.1 JAVA程序运行原...
1.1.2 线程状态
1.1.3 线程中止
1.1.4 内存屏障和CPU缓存
1.1.5 线程通信
1.1.6 线程封闭之Thread...
1.1.7 线程池应用及实现...
第二节 线程安全问题
1.2.1 线程安全之可见性...
2020年03月02日 20:00
1.2.2 线程安全之原子操作
2020年03月04日 20:00
1.2.3 JAVA锁相关
2020年03月06日 20:00
第三节 J.U.C并发编程包详解
1.3.1 Lock接口及其实现
2020年03月09日 20:00
1.3.1 AQS抽象队列同步...


返回课程
线程数量
网易云课堂
如何确定合适数量的线程?
计算型任务：cpu数量的1-2倍
IO型任务：相对比计算型任务，需多一些线程，要根据具体的IO阻塞时长进行考量决定。
如tomcat中默认的最大线程数为：200。
也可考虑根据需要在一个最小数量和最大数量间自动增减线程数。
51:58 / 53:08
1.5x
标清
自动播放下一课
报告问题
1.1.4 内存屏障和CPU缓存
1.1.5 线程通信
1.1.6 线程封闭之Thread...
1.1.7 线程池应用及实现...
第二节 线程安全问题
1.2.1 线程安全之可见性...
2020年03月02日 20:00
1.2.2 线程安全之原子操作
2020年03月04日 20:00
1.2.3 JAVA锁相关
2020年03月06日 20:00
第三节 J.U.C并发编程包详解
1.3.1 Lock接口及其实现
2020年03月09日 20:00
1.3.1 AQS抽象队列同步...
2020年03月11日 20:00
1.3.2 并发容器类-1
2020年03月13日 20:00
1.3.3 并发容器类-2
2020年03月16日 20:00
1.3.4 Fork/Join框架详解