Java学习:线程池的简单构建 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_Java_E5_AD_A6_E4_B9_A0_c104_145284.htm 现在服务器端的应用程序几乎都采用了“ 线程池 ”技术，这主要是为了提高系统效率。因为如果服务器对应每一个请求就创建一个线程的话，在很短的一段时间内就会产生很多创建和销毁线程动作，导致服务器在创建和销毁线程上花费的时间和消耗的系统资源要比花在处理实际的用户请求的时间和资源更多。线程池就是为了尽量减少这种情况的发生。 下面我们来看看怎么用Java实现一个线程池。一个比较简单的线程池至少应包含线程池管理器、工作线程、任务队列、任务接口等部分。其中线程池管理器(ThreadPool Manager)的作用是创建、销毁并管理线程池，将工作线程放入线程池中；工作线程是一个可以循环执行任务的线程，在没有任务时进行等待；任务队列的作用是提供一种缓冲机制，将没有处理的任务放在任务队列中；任务接口是每个任务必须实现的接口，主要用来规定任务的入口、任务执行完后的收尾工作、任务的执行状态等，工作线程通过该接口调度任务的执行。 一种就是以固定线程数目作为基准，让每一个线程的工作线程都处于无限循环中，利用Java中基类的nodify（）与wait（）进行协同工作。 基本思想如下：在构建线程池的时候创建所有工作线程，并且让所有工作线程开始运行。 public ThreadPool(int nPoolSize) { if(nPoolSize { nPoolSize=DEFAULT_POOL_SIZE. } m_ThreadList=new ArrayList(). m_RunList=new LinkedList(). for(int i=0.i { WorkerThread temp=new WorkerThread(i 1).

m_ThreadList.add(temp). temp.start(). } } 在工作线程的run（）方法中用wait（）进行等待，当线程处于wait（）状态基本不占用CPU，这样所有工作线程都处于挂起状态，等待任务来唤醒。 实现如下： while (true) { synchronized (m_RunList) { while (m_RunList.isEmpty()) { //任务列表为空则进行等待，否则运行任务，并在本任务列表里面剔除任务 try { m_RunList.wait(). } catch (InterruptedException e) {} } r = (Runnable) m_RunList.removeFirst(). file://System.out.println(m_nThreadID ":Start"). if (r == null)return. } try { r.run(). } catch (Exception e) {} } 以上代码就是一个工作线程主要代码。这个线程永远不会停止，只可能被挂起，或者运行任务。 