Java5.0多线程编程实践 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022_Java50_E5_A4_9A_c104_144403.htm Java5增加了新的类库并发集java.util.concurrent，该类库为并发程序提供了丰富的API多线程编程在Java 5中更加容易，灵活。本文通过一个网络服务器模型，来实践Java5的多线程编程，该模型中使用了Java5中的线程池，阻塞队列，可重入锁等，还实践了Callable，Future等接口，并使用了Java 5的另外一个新特性泛型。 简介 本文将实现一个网络服务器模型，一旦有客户端连接到该服务器，则启动一个新线程为该连接服务，服务内容为往客户端输送一些字符信息。一个典型的网络服务器模型如下： 1. 建立监听端口。 2. 发现有新连接，接受连接，启动线程，执行服务线程。 3. 服务完毕，关闭线程。 这个模型在大部分情况下运行良好，但是需要频繁的处理用户请求而每次请求需要的服务又是简短的时候，系统会将大量的时间花费在线程的创建销毁。Java 5的线程池克服了这些缺点。通过对重用线程来执行多个任务，避免了频繁线程的创建与销毁开销，使得服务器的性能方面得到很大提高。因此，本文的网络服务器模型将如下： 1. 建立监听端口，创建线程池。 2. 发现有新连接，使用线程池来执行服务任务。 3. 服务完毕，释放线程到线程池。 下面详细介绍如何使用Java 5的concurrent包提供的API来实现该服务器。 初始化 初始化包括创建线程池以及初始化监听端口。创建线程池可以通过调用java.util.concurrent.Executors类里的静态方法newChahedThreadPool或是newFixedThreadPool来创建，也

可以通过新建一个java.util.concurrent.ThreadPoolExecutor实例来执行任务。这里我们采用newFixedThreadPool方法来建立线程池。 ExecutorService pool = Executors.newFixedThreadPool(10). 表示新建了一个线程池，线程池里面有10个线程为任务队列服务。 使用ServerSocket对象来初始化监听端口。 private static final int PORT = 19527. serverListenSocket = new ServerSocket(PORT). serverListenSocket.setReuseAddress(true). serverListenSocket.setReuseAddress(true). 服务新连接 当有新连接建立时，accept返回时，将服务任务提交给线程池执行。 while(true){ Socket socket = serverListenSocket.accept(). pool.execute(new ServiceThread(socket)). } 这里使用线程池对象来执行线程，减少了每次线程创建和销毁的开销。任务执行完毕，线程释放到线程池。 服务任务 服务线程ServiceThread维护一个count来记录服务线程被调用的次数。每当服务任务被调用一次时，count的值自增1，因此ServiceThread提供一个increaseCount和getCount的方法，分别将count值自增1和取得该count值。由于可能多个线程存在竞争，同时访问count，因此需要加锁机制，在Java 5之前，我们只能使用synchronized来锁定。Java 5中引入了性能更加粒度更细的重入锁ReentrantLock。我们使用ReentrantLock保证代码线程安全。下面是具体代码： private static ReentrantLock lock = new ReentrantLock (). private static int count = 0. private int getCount(){ int ret = 0. try{ lock.lock(). ret = count. }finally{ lock.unlock(). } return ret. } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com