JAVA线程池的简单实现及优先级设置Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_JAVA_E7_BA_BF_E7_A8_8B_c104_644787.htm 我们大家都知道，在处理多线程服务并发时，由于创建线程需要占用很多的系统资源，所以为了避免这些不必要的损耗，通常我们采用线程池来解决这些问题。 线程池的基本原理是，首先创建并保持一定数量的线程，当需要使用线程时，我们从池中取得线程，再将需要运行的任务交给线程进行处理，当任务完成后再将其释放回池中。 下面，我给出一个很简单的实现模型，仅供参考。 ThreadPool.java package org.loon.framework.util.test. import java.util.LinkedList. import java.util.List. /** */ /** * * Title: LoonFramework * * * Description: * * * Copyright: Copyright (c) 2007 * * * Company: LoonFramework * * * @author chenpeng * @email：ceponline@yahoo.com.cn * @version 0.1 */ public class ThreadPool ... { private static ThreadPool instance = null . // 优先级低 public static final int PRIORITY_LOW = 0 . // 普通 public static final int PRIORITY_NORMAL = 1 . // 高 public static final int PRIORITY_HIGH = 2 . // 用以保存空闲连接 private List[] _idxThreads. // 关闭 private boolean _shutdown = false . // 线程数量 private int _threadCount = 0 . // debug信息是否输出 private boolean _debug = false . /** */ /** * 返回ThreadPool实例 * * @return */ public static ThreadPool getInstance() ... { if (instance == null) ... { instance = new ThreadPool(). } return instance. } // 初始化线程list private ThreadPool() ... { this ._idxThreads = new List[] ... { new LinkedList(), new LinkedList(), new LinkedList() } . this

._threadCount = 0 . } /** */ /** * 同步方法，完成任务后将资源放回线程池中 * @param repool */ 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