fhd001

永久域名 http://fhd001.javaeye.com

fhd001

浏览: 32333 次

性别:

来自: 广州

我现在离线

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客

nx0010 Ooocean

执着IT路 ivyloo

博客分类

零基础学习linux | 信号量

2009-09-29

ThreadPoolExecutor使用简介

关键字: threadpoolexecutor使用简介

java.util.concurrent.ThreadPoolExecutor使用简介

在多线程大师Doug Lea的贡献下，在JDK1.5中加入了许多对并发特性的支持，例如：线程池。

一、简介

线程池类为 java.util.concurrent.ThreadPoolExecutor，常用构造方法为：

ThreadPoolExecutor(int corePoolSize, int maximumPoolSize,

long keepAliveTime, TimeUnit unit,

BlockingQueue<Runnable> workQueue,

RejectedExecutionHandler handler)

corePoolSize： 线程池维护线程的最少数量

maximumPoolSize：线程池维护线程的最大数量

keepAliveTime：线程池维护线程所允许的空闲时间

unit： 线程池维护线程所允许的空闲时间的单位

workQueue：线程池所使用的缓冲队列

handler： 线程池对拒绝任务的处理策略

一个任务通过 execute(Runnable)方法被添加到线程池，任务就是一个 Runnable类型的对象，任务的执行方法就是 Runnable类型对象的run()方法。

当一个任务通过execute(Runnable)方法欲添加到线程池时：

如果此时线程池中的数量小于corePoolSize，即使线程池中的线程都处于空闲状态，也要创建新的线程来处理被添加的任务。

如果此时线程池中的数量等于 corePoolSize，但是缓冲队列 workQueue未满，那么任务被放入缓冲队列。

如果此时线程池中的数量大于corePoolSize，缓冲队列workQueue满，并且线程池中的数量小于maximumPoolSize，建新的线程来处理被添加的任务。

如果此时线程池中的数量大于corePoolSize，缓冲队列workQueue满，并且线程池中的数量等于maximumPoolSize，那

- 全部博客 (278)
- java基本功 (56)
- java多线程 (6)
- ant (10)
- uml (10)
- java设计模式 (5)
- 数据结构与算法(java描述) (12)
- hibernate (36)
- oracle与oracle客户端工具 (11)
- struts1.x (2)
- spring (34)
- 杂项 (13)
- jakarta commons组件 (43)
- log4j (5)
- maven2学习 (9)
- linux (5)
- mysql (2)
- corba (0)
- DWR (1)
- NetBeans (5)
- jboss (1)
- web service (8)

么通过 handler所指定的策略来处理此任务。
也就是：处理任务的优先级为：
核心线程corePoolSize、任务队列workQueue、最大线程maximumPoolSize，如果三者都满了，使用handler处理被拒绝的任务。
当线程池中的线程数量大于 corePoolSize时，如果某线程空闲时间超过keepAliveTime，线程将被终止。这样，线程池可以动态的调整池中的线程数。
unit可选的参数为java.util.concurrent.TimeUnit中的几个静态属性：
NANOSECONDS、MICROSECONDS、MILLISECONDS、SECONDS。
workQueue我常用的是：java.util.concurrent.ArrayBlockingQueue
handler有四个选择：
ThreadPoolExecutor.AbortPolicy()
抛出java.util.concurrent.RejectedExecutionException异常
ThreadPoolExecutor.CallerRunsPolicy()
重试添加当前的任务，他会自动重复调用execute()方法
ThreadPoolExecutor.DiscardOldestPolicy()
抛弃旧的任务
ThreadPoolExecutor.DiscardPolicy()
抛弃当前的任务

二、一般用法举例

```
//-----------------------------------------------------------
//TestThreadPool.java
//package cn.simplelife.exercise;
import java.io.Serializable;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
public class TestThreadPool {
private static int produceTaskSleepTime = 2;
private static int consumeTaskSleepTime = 2000;
private static int produceTaskMaxNumber = 10;
public static void main(String[] args) {
//构造一个线程池
ThreadPoolExecutor threadPool = new ThreadPoolExecutor(2, 4, 3,
TimeUnit.SECONDS, new ArrayBlockingQueue<Runnable>(3),
```

谢你的共享。

-- by fenyouze

其他分类

- 我的收藏 (53)
- 我的论坛主题贴 (278)
- 我的所有论坛贴 (0)
- 我的精华良好贴 (0)

最近加入圈子

存档

- 2010-02 (1)
- 2010-01 (1)
- 2009-12 (12)
- 更多存档...

最新评论

- webservice的原理及概念

  纠正一处啊，SOAP为simple object access protocol的 ...

  -- by penol

- java中正则表达式运用详解

  ...

  -- by fatbear007

- commons beanutils之Bean ...

  请问这个除了导入commons-beanutils-1.8.0.jar包后还有哪些 ...

  -- by haiyupeter

- java泛型(实例演示)

  不错，学习啦！

```
new ThreadPoolExecutor.DiscardOldestPolicy());
for(int i=1;i<=produceTaskMaxNumber;i++){
try {
//产生一个任务，并将其加入到线程池
String task = "task@ " + i;
System.out.println("put " + task);
threadPool.execute(new ThreadPoolTask(task));
//便于观察，等待一段时间
Thread.sleep(produceTaskSleepTime);
} catch (Exception e) {
e.printStackTrace();
}
}
}

public static class ThreadPoolTask implements Runnable,Serializable{
private static final long serialVersionUID = 0;
//保存任务所需要的数据
private Object threadPoolTaskData;
ThreadPoolTask(Object tasks){
this.threadPoolTaskData = tasks;
}
public void run(){
//处理一个任务，这里的处理方式太简单了，仅仅是一个打印语句
System.out.println("start .."+threadPoolTaskData);
try {
////便于观察，等待一段时间
Thread.sleep(consumeTaskSleepTime);
} catch (Exception e) {
e.printStackTrace();
}
threadPoolTaskData = null;
}
public Object getTask(){
return this.threadPoolTaskData;
}
```

-- by 浪客剑心

- Load与get区别

  不错，楼主说得一针见血，简明扼要

  -- by sefcertyu

评论排行榜

- java泛型(实例演示)
- Load与get区别
- commons beanutils之Bean克隆
- webservice的原理及概念
- java中正则表达式运用详解


```
}
}
//--------------------------------------------------------
```

说明:

1、在这段程序中，一个任务就是一个Runnable类型的对象，也就是一个ThreadPoolTask类型的对象。

2、一般来说任务除了处理方式外，还需要处理的数据，处理的数据通过构造方法传给任务。

3、在这段程序中，main()方法相当于一个残忍的领导，他派发出许多任务，丢给一个叫 threadPool的任劳任怨的小组来做。

这个小组里面队员至少有两个，如果他们两个忙不过来，任务就被放到任务列表里面。

如果积压的任务过多，多到任务列表都装不下(超过3个)的时候，就雇佣新的队员来帮忙。但是基于成本的考虑，不能雇佣太多的队员，至多只能雇佣 4个。

如果四个队员都在忙时，再有新的任务，这个小组就处理不了了，任务就会被通过一种策略来处理，我们的处理方式是不停的派发，直到接受这个任务为止(更残忍！呵呵)。

因为队员工作是需要成本的，如果工作很闲，闲到 3SECONDS都没有新的任务了，那么有的队员就会被解雇了，但是，为了小组的正常运转，即使工作再闲，小组的队员也不能少于两个。

4、通过调整 produceTaskSleepTime和 consumeTaskSleepTime的大小来实现对派发任务和处理任务的速度的控制，改变这两个值就可以观察不同速率下程序的工作情况。

5、通过调整4中所指的数据，再加上调整任务丢弃策略，换上其他三种策略，就可以看出不同策略下的不同处理方式。

6、对于其他的使用方法，参看jdk的帮助，很容易理解和使用。

15:49 | 浏览 (223) | 评论 (0) | 分类: java多线程 | 相关推荐

评论

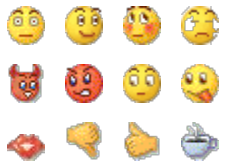
发表评论

表情图标

B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签



您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交