

用心做事

生活因技术而美好

目录视图 摘要视图 RSS 订阅

个人资料



John00000001

访问：672913次

积分：10396

等级：BLOG 7

排名：第1723名

原创：354 篇

转载：0 篇

译文：14 篇

评论：89 条

文章搜索

文章分类

C/C++ (10)

Java Core (56)

security (2)

Spring (2)

Studying And Thinking (6)

Thinking (23)

NO SQL (10)

javascript (6)

CI (2)

Search Engine (6)

Solr (8)

Maven (6)

Concurrent (19)

DesignPattern (24)

English (7)

Optimization (5)

interview (4)

ETL (5)

Python (5)

异步赠书：10月Python畅销书升级 【线路图】人工智能到底学什么？！ 程序员9月书讯 节后荐书：Python、PyQt5、Kotlin（评论送书）

深度理解Thread Pool, Executor, Callable/Future

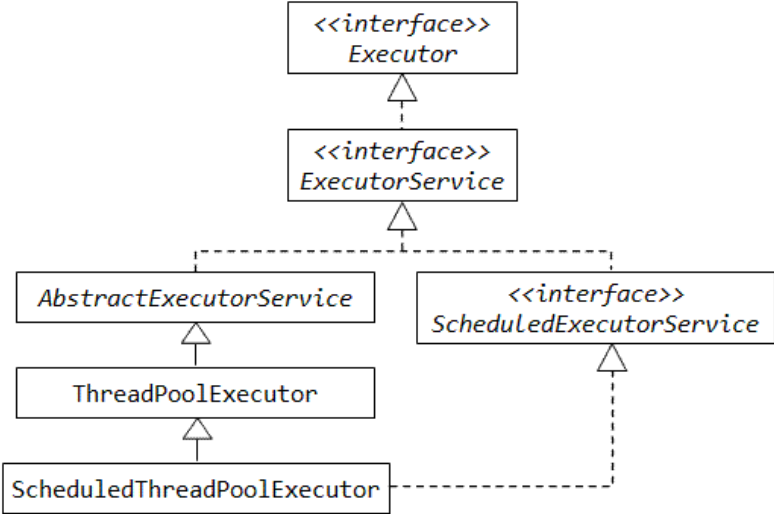
标签：Java 线程池 并发

2013-04-27 23:04 2304人阅读 评论(0) 收藏 举报

分类：Concurrent (18)

版权声明：本文为博主John Lau原创文章，未经博主允许不得转载

在JDK1.5版，Java标准包中就包含了对线程池的支持，提供了包java.lang.concurrent.



1. ThreadPool

线程池是一个线程管理的集合，能够有效执行任务。当大量任务使用线程池执行时，由于线程循环的执行，整个性能得到提高，也减少了每个任务调用上的花费。实现线程池，你能够运用ExecutorService的实现类，比如ThreadPoolExecutor 或是ScheduledThreadPoolExecutor。另外，在类Executors 中提供了如下更方便的工厂方法：

- Executors.newSingleThreadExecutor(): 创建一个单一后台线程实例。
- Executors.newFixedThreadPool(int numThreads): 创建一个
- Executors.newCachedThreadPool(): 运用自动化线程回收利用

使用线程池的步骤如下：

- 编写出实现了Runnable接口的工人线程类，run()方法指定了运行
- 使用由Executor类提供的工厂方法创建一个线程池(ExecutorSer是 Executors.newSingleThreadExecutor(), 或者Executor numThreads), 也可以是Executors.newCachedThreadPool(
- 创建你的工人线程实例，使用execute(Runnable)方法去添加-中有一个有效的线程，这个任务将会被调度并执行。

应该学什么

人工智能工程师

了解更多

| | |
|--------------|------|
| SQL | (5) |
| OS | (10) |
| Tools | (37) |
| MongoDB | (4) |
| Hadoop | (30) |
| HBase | (26) |
| Linux | (54) |
| Hive | (3) |
| sqoop | (2) |
| Flume | (7) |
| Quartz | (1) |
| Zookeeper | (11) |
| Shell | (3) |
| Apache | (8) |
| Ganglia | (2) |
| Architecture | (3) |
| Redis | (2) |
| Web Server | (26) |
| Cloud | (52) |
| Spark | (8) |
| Kafka | (3) |
| Scala | (8) |
| Go | (27) |
| storm | (1) |
| etcd | (2) |

| | |
|----------|-----|
| 文章存档 | |
| 2017年10月 | (8) |
| 2017年09月 | (1) |
| 2017年02月 | (6) |
| 2016年12月 | (2) |
| 2016年11月 | (1) |
| 展开 | |

| | |
|-----------------------|---------|
| 阅读排行 | |
| 流行的Go语言web框: | |
| Solr云(SolrCloud) | (24406) |
| 大数据Lambda架构 | (11870) |
| Kettle调度和监控 | (11289) |
| 异常/usr/bin/install: c | (10718) |
| ssh-copy-id命令详解 | (8454) |
| Nginx的UDP负载均衡 | (8157) |
| Go语言实现将[]string | (8036) |
| 恩尼格密码机原理 | (7204) |
| SolrJ 操作HttpSolrSe | (6927) |
| | (6335) |

| | |
|---------------------|------|
| 评论排行 | |
| 即使没有读者，你应i | (17) |
| Kettle (Pentaho Dat | (10) |

2. 理解相关API

一个Executor对象能够执行Runnable提交的任务。Executor接口类（1.6version）中的方法：

```
[java] view plain copy print ?
01. void execute(Runnable command);
```

它在不久的某个时间里执行一个给予任务。

接口ExecutorService定义了一些有用的方法，重要的两个如下：

```
[java] view plain copy print ?
01. public void shutdown();
02. // Initiates an orderly shutdown of the thread pool.
03. // The previously executed/submitted tasks are allowed to complete,
04. // but no new tasks will be scheduled.
05. public <T> Future<T> submit(Callable<T> task);
06. // Submit or schedule the callable task for execution, which returns a Future
```

Executors类中定义一些有用的方法：

```
[java] view plain copy print ?
01. static ExecutorService newSingleThreadExecutor()
02. static ExecutorService newFixedThreadPool(int nThreads)
03. static ExecutorService newCachedThreadPool()
04. static ScheduledExecutorService newSingleThreadScheduledExecutor()
05. static ScheduledExecutorService newScheduledThreadPool(int size)
```

一个简单的例子来说明一下如何调用使用线程池。

```
[java] view plain copy print ?
01. public class WorkerThread implements Runnable {
02.     private int workerNumber;
03.
04.     WorkerThread(int workerNumber) {
05.         this.workerNumber = workerNumber;
06.     }
07.
08.     public void run() {
09.         // The thread simply prints 1 to 5
10.         for (int i = 1; i <= 5; i++) {
11.             System.out.printf("Worker %d: %d\n", workerNumber, i);
12.             try {
13.                 // sleep for 0 to 0.5 second
14.                 Thread.sleep((int)(Math.random() * 500));
15.             } catch (InterruptedException e) {}
16.         }
17.     }
18. }
```

```
[java] view plain copy print ?
01. import java.util.concurrent.ExecutorService;
02. import java.util.concurrent.Executors;
03.
```

关闭



| | |
|--------------------|-----|
| 浅析HBase架构和系 | (6) |
| 高级column family 配 | (3) |
| 机器学习知识体系 | (3) |
| Java HotSpot VM命 | (3) |
| 解决has leaked Servi | (3) |
| Flume 中文件channe | (2) |
| Hadoop REST API -- | (2) |
| 顶级的6个开源的系统 | (2) |

| | |
|------------------------------|--|
| 推荐文章 | |
| * 【观点】第二十三期：程序员应该如何积累财富？ | |
| * Android检查更新下载安装 | |
| * 动手打造史上最简单的Recycleview 侧滑菜单 | |
| * TCP网络通讯如何解决分包粘包问题 | |
| * SDCC 2017之大数据技术实战线上峰会 | |
| * 快速集成一个视频直播功能 | |

| | |
|--|--|
| 最新评论 | |
| Go 多态功能实现 | |
| 胖爸爸: 这.....不是多态吧是不是应该至少嵌套一层struct内层的strcut有附属的方法, 然后外层的stru... | |
| 异常/usr/bin/install: cannot c | |

开发一个app多少钱



app开发报价单



么不return? 2017/08/...

Nginx的 UDP 负载均衡的: client ---> nginx --->server1 server2 ...请问---

Go 多态功能实现

u010566245: 这不是多态啊

解决has leaked ServiceCon

Untitled小明: 也有可能是locationClient 初始化的时候上下文不能写当前activity, 要写getAp...

```
04. public class ThreadPoolTest {
05.     public static void main(String[] args) {
06.         int numWorkers = Integer.parseInt(args[0]);
07.         int threadPoolSize = Integer.parseInt(args[1]);
08.
09.         ExecutorService pool =
10.             Executors.newFixedThreadPool(threadPoolSize);
11.         WorkerThread[] workers = new WorkerThread[numWorkers];
12.         for (int i = 0; i < numWorkers; i++) {
13.             workers[i] = new WorkerThread(i+1);
14.             pool.execute(workers[i]);
15.         }
16.         pool.shutdown();
17.     }
18. }
```

结果如下:

```
[plain] view plain copy print ?
01. Worker 1: 1
02. Worker 2: 1
03. Worker 2: 2
04. Worker 2: 3
05. Worker 2: 4
06. Worker 1: 2
07. Worker 1: 3
08. Worker 2: 5
09. Worker 1: 4
10. Worker 1: 5
11. Worker 3: 1
12. Worker 3: 2
13. Worker 4: 1
14. Worker 3: 3
15. Worker 3: 4
16. Worker 4: 2
17. Worker 3: 5
18. Worker 4: 3
19. Worker 5: 1
20. Worker 4: 4
21. Worker 5: 2
22. Worker 5: 3
23. Worker 4: 5
24. Worker 5: 4
25. Worker 5: 5
```

接口Callable和Runnable。Callable与Runnable很相似。然而， Callable提供了一种能够固定在线程上的返回结果或是出现的异常。

关闭

```
[java] view plain copy print ?
01. public V call()
02. // Call() returns a result of type <V>, or th
```

在线程池中，使用submit(Callable r)可以返回一个Future<V>象。当需要有返回结果时，可以检索 get()方法来获得。如果不出现异常，Future<V>的方法如下：



[java] view plain copy print ?

```
01. V get() // wait if necessary, retrieve result
02. V get(long timeout, TimeUnit unit)
03. boolean cancel(boolean mayInterruptIfRunning)
04. boolean isCancelled()
05. boolean isDone() // return true if this task completed
```

同样的例子，换一种写法：

[java] view plain copy print ?

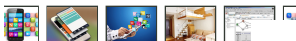
```
01. public class CallableWorkerThread implements Callable<String> {
02.
03.     private int workerNumber;
04.
05.     CallableWorkerThread(int workerNumber) {
06.         this.workerNumber = workerNumber;
07.     }
08.
09.     public String call() { // use call() instead of run()
10.         for (int i = 1; i <= 5; i++) { // just print 1 to 5
11.             System.out.printf("Worker %d: %d\n", workerNumber, i);
12.             try {
13.                 Thread.sleep((int)(Math.random() * 1000));
14.             } catch (InterruptedException e) {}
15.         }
16.         return "worker " + workerNumber;
17.     }
18. }
```

[java] view plain copy print ?

```
01. package org.concurrency.simple;
02.
03. /**
04.  * @author: John Liu
05.  */
06. import java.util.concurrent.*;
07.
08. public class CallableThreadPoolTest {
09.     public static void main(String[] args) {
10.         int numWorkers = 5;
11.
12.         ExecutorService pool = Executors.newCachedThreadPool();
13.         CallableWorkerThread workers[] = new CallableWorkerThread[numWorkers];
14.         Future[] futures = new Future[numWorkers];
15.
16.         for (int i = 0; i < numWorkers; i++) {
17.             workers[i] = new CallableWorkerThread(i);
18.             futures[i] = pool.submit(workers[i]);
19.         }
20.         for (int i = 0; i < numWorkers; i++) {
21.             try {
22.                 System.out.println(futures[i].get());
23.             } catch (InterruptedException ex) {
24.                 ex.printStackTrace();
25.             } catch (ExecutionException ex) {
```

关闭

开发一个app多少钱



app开发报价单



应该学什么
人工智能工程师

了解更多

```
26.         ex.printStackTrace();
27.     }
28. }
29. }
30. }
```

结果:

```
[java] view plain copy print ?
01. Worker 2: 1
02. Worker 4: 1
03. Worker 5: 1
04. Worker 1: 1
05. Worker 3: 1
06. Worker 4: 2
07. Worker 4: 3
08. Worker 2: 2
09. Worker 4: 4
10. Worker 5: 2
11. Worker 1: 2
12. Worker 3: 2
13. Worker 4: 5
14. Worker 5: 3
15. Worker 2: 3
16. Worker 5: 4
17. Worker 3: 3
18. Worker 1: 3
19. Worker 2: 4
20. Worker 1: 4
21. Worker 3: 4
22. Worker 5: 5
23. Worker 2: 5
24. Worker 1: 5
25. Worker 3: 5
26. worker 1 ended
27. worker 2 ended
28. worker 3 ended
29. worker 4 ended
30. worker 5 ended
```

开发一个app多少钱



app开发报价单



顶

2

踩

0

关闭

上一篇

Extjs实现年月日时分秒格式的时间选择器

下一篇

解析ThreadPoolExecutor

相关文章推荐

- java中Executor、ExecutorService、Threa...
- Java Thr

应|该|学|什|么

人工智能工程师

了解更多

- Python全栈工程师入门指南
 - callable和runnable以及线程池对他们的执行
 - 自然语言处理在“天猫精灵”的实践应用--姜...
 - 深度理解Thread Pool, Executor, Callable/...
 - Vue2.x基本特性解析
 - 理解Thread Pool, Executor, Callable/Future
 - 程序员都应该掌握的Git和Github实用教程
- 深度学习项目实战-人脸检测
 - Callable,Runnable,Thread,Future之简单...
 - Shell脚本编程
 - Socket Programming & Thread Pool实例
 - Smart Thread Pool
 - A simple C++11 Thread Pool implementati...
 - Thread Pool 实例



郁金香的种子



OA办公系统



开发一个app多少



喷码机



app外包公司



查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之网络科技有限公司

| 江苏乐知网络科技有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved



开发一个app多少钱



app开发报价单



关闭

