

王新友的博客

博观而约取,厚积而薄发

目录视图

摘要视图

RSS 订阅

个人资料



wxy_fighting

关注发私信

访问：356771次

积分：5482

等级：BLOG 6

排名：第5075名

原创：143篇

转载：274篇

译文：0篇

评论：18条

文章搜索



郁金香的种子





一点点加盟费



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）

理解Thread Pool, Executor, Callable/Future

标签：Java 线程池 多线程

2013-06-15 13:29

675人阅读

评论(0)

收藏

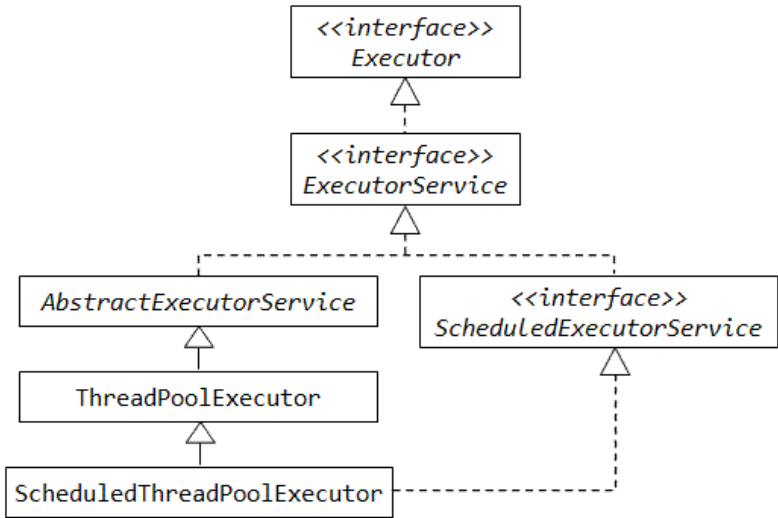
举报

分类：

java (238)

版权声明：本文为博主原创文章，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。要实现类，比如 `ThreadPoolExecutor` 或是 `ScheduledThreadPoolExecutor`，提供了如下更方便的工厂方法：

- `Executors.newSingleThreadExecutor()`: 创建一个单线程池
 - `Executors.newFixedThreadPool(int numThreads)`: 创建一个固定大小的线程池
 - `Executors.newCachedThreadPool()`: 运用自动化线程回收机制的线程池
- 使用线程池的步骤如下：

1. 编写出实现了 `Runnable` 接口的工人线程类，`run()` 方法指明要执行的任务
2. 使用由 `Executor` 类提供的工厂方法创建一个线程池(Executor 是 `Executors.newSingleThreadExecutor()`，或者是 `Executors.newFixedThreadPool(int numThreads)`)



app开发报价单



开发一个app多少钱



app外包公司



喷码机

- mule (3)
- maven (4)
- UML (0)
- excel (0)
- security (5)
- JVM (2)
- 正则表达式 (4)
- quartz (7)
- tomcat (1)
- jboss (1)
- util (24)
- 线程 (1)
- ibatis (0)
- POI (4)
- socket (0)
- 随笔杂谈 (2)
- 技术积累 (3)
- nginx (2)
- 神经网络 (14)
- ESB (3)
- J2EE调优 (3)
- 日常操作技巧 (4)
- Eclipse (3)
- CXF (2)

文章存档

- 2015年02月 (1)
- 2015年01月 (1)
- 2014年12月 (2)



郁金香的种子



一点点加盟费



理解Thread Pool, Executor, Callable/Future - 王新友的博客 - CSDN博客

numThreads), 也可以是Executors.newCachedThreadPool() 创建的。

3. 创建你的工人线程实例, 使用execute(Runnable) 方法去添加一个Runnable任务到线程池中。如何在池中有一个有效的线程, 这个任务将会被调度并执行。

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:

Executors类中定义一些有用的方法:

一个简单的例子来说明一下如何调用使用线程池。

关闭



java实现RSA加密	(3)
征服 Redis	(2)
服务治理过程演讲	(2)
Velocity字符串模板替换	(1)
Java Socket 多线程网络传...	(1)
BP神经网络在期货价格...	(1)
openssl从PEM导出私钥、...	(1)
访问架构师的第一步	(1)
用Java开源项目JOONE实...	(1)

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

openssl从PEM导出私钥、公钥
jm0477 : 博客很有用，看来还是需要去好好了解下原理。今天搞了好久，才看到你的文章。

用Java开源项目JOONE实现人工智...
sinat_36875266 : 好

一键搞定JavaEE应用JRE+Tomcat+...
qq_34213797 : 显示找不到项目是怎么回事？我将项目放到对应的app/web下了

soapUI Pro 4.5.1的新破解方案
wo80da90 : 杠杠的



郁金香的种子



一点点加盟费



结果如下：

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。

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

```
[java] view plain copy print ?
01. V get() // wait if necessary, re
02. V get(long timeout, TimeUnit unit)
03. boolean cancel(boolean mayInterruptIfRunni
04. boolean isCancelled()
05. boolean isDone() // return true if this t
```

关闭

app开发报价单

开发一个app多少钱

app外包公司

喷码机

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 + 1);
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() + " ended");
23.             } catch (InterruptedException ex) {
24.                 ex.printStackTrace();
25.             } catch (ExecutionException ex) {
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
```



郁金香的种子



一点点加盟



app开发报价单



开发一个app多少钱



app外包公司



喷码机


```
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
```

顶 踩
0 0

- 上一篇 解析ThreadPoolExecutor
- 下一篇 比较Java循环的性能

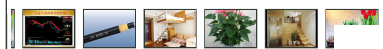
相关文章推荐

- 深度理解Thread Pool, Executor, Callable/Future
- Python全栈工程师入门指南
- 理解Thread Pool, Executor, Callable/Future
- 自然语言处理在“天猫精灵”的实践应用--姜...
- Java Thread&Concurrency(7): 深入理解Calla...
- Vue2.x基本特性解析
- Callable,Runnable,Thread,Future之简单理解
- 程序员都应该掌握的Git和Github实用教程
- Socket Programming & Thread Pool实例
- 深度学习项目实战-人脸检测
- Smart Thread Pool
- Shell脚本编程
- A simple C++11 Thread Pool implementation
- Thread Pool 实例
- Thread Pool(线程池)技术
- Pth

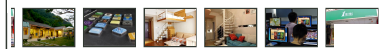
关闭

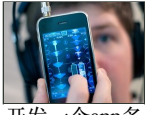


郁金香的种子




一点点加盟费

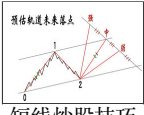





开发一个app多



app开发报价单



短线炒股技巧



公众号开发

查看评论

暂无评论



app开发报价单



开发一个app多少钱



app外包公司



喷码机

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



郁金香的种子





一点点加盟费



关闭


app开发报价单


开发一个app多少钱


app外包公司


喷码机