JAVA多线程实现的四种方式

Java多线程实现方式主要有四种：继承Thread类、实现Runnable接口、实现Callable接口通过FutureTask包装器来创建Thread线程、使用ExecutorService、Callable、Future实现有返回结果的多线程。

其中前两种方式线程执行完后都没有返回值，后两种是带返回值的。

1、继承Thread类创建线程

Thread类本质上是实现了Runnable接口的一个实例，代表一个线程的实例。启动线程的唯一方法就是通过Thread类的start()实例方法。start()方法是一个native方法，它将启动一个新线程，并执行run()方法。这种方式实现多线程很简单，通过自己的类直接extend Thread，并复写run()方法，就可以启动新线程并执行自己定义的run()方法。例如：

```
public class MyThread extends Thread {
   public void run() {
    System.out.println("MyThread.run()");
   }
}

MyThread myThread1 = new MyThread();
MyThread myThread2 = new MyThread();
myThread1.start();
myThread2.start();
```

2、实现Runnable接口创建线程

如果自己的类已经extends另一个类，就无法直接extends Thread，此时，可以实现一个Runnable接口，如下：

```
public class MyThread extends OtherClass implements Runnable {
   public void run() {
    System.out.println("MyThread.run()");
   }
}
```

为了启动MyThread，需要首先实例化一个Thread，并传入自己的MyThread实例：

```
MyThread myThread = new MyThread();
Thread thread = new Thread(myThread);
thread.start();
```

事实上，当传入一个Runnable target参数给Thread后，Thread的run()方法就会调用target.run()，参考JDK源代码：

```
public void run() {
   if (target != null) {
    target.run();
```

随笔档案 (507)


```
    }
}
```

3、实现Callable接口通过FutureTask包装器来创建Thread线程

Callable接口（也只有一个方法）定义如下：

```
public interface Callable<V>   {
  V call ( ) throws Exception;   }
```

```
public class SomeCallable<V> extends OtherClass implements Callable<V> {

    @Override
    public V call() throws Exception {
        // TODO Auto-generated method stub
        return null;
    }

}
```

```
Callable<V> oneCallable = new SomeCallable<V>();
//由Callable<Integer>创建一个FutureTask<Integer>对象：
FutureTask<V> oneTask = new FutureTask<V>(oneCallable);
//注释：FutureTask<Integer>是一个包装器，它通过接受Callable<Integer>来创建，它同时实现了Future和Runnable接口。
  //由FutureTask<Integer>创建一个Thread对象：
Thread oneThread = new Thread(oneTask);
oneThread.start();
//至此，一个线程就创建完成了。
```

4、使用ExecutorService、Callable、Future实现有返回结果的线程

ExecutorService、Callable、Future三个接口实际上都是属于Executor框架。返回结果的线程是在JDK1.5中引入的新特征，有了这种特征就不需要再为了得到返回值而大费周折了。而且自己实现了也可能漏洞百出。

可返回值的任务必须实现Callable接口。类似的，无返回值的任务必须实现Runnable接口。

执行Callable任务后，可以获取一个Future的对象，在该对象上调用get就可以获取到Callable任务返回的Object了。

注意：get方法是阻塞的，即：线程无返回结果，get方法会一直等待。

再结合线程池接口ExecutorService就可以实现传说中有返回结果的多线程了。

下面提供了一个完整的有返回结果的多线程测试例子，在JDK1.5下验证过没问题可以直接使用。代码如下：

```
import java.util.concurrent.*;
import java.util.Date;
import java.util.List;
import java.util.ArrayList;

/**
* 有返回值的线程
*/
@SuppressWarnings("unchecked")
public class Test {
public static void main(String[] args) throws ExecutionException,
    InterruptedException {
   System.out.println("----程序开始运行----");
   Date date1 = new Date();

   int taskSize = 5;
   // 创建一个线程池
   ExecutorService pool = Executors.newFixedThreadPool(taskSize);
   // 创建多个有返回值的任务
   List<Future> list = new ArrayList<Future>();
   for (int i = 0; i < taskSize; i++) {
    Callable c = new MyCallable(i + " ");
    // 执行任务并获取Future对象
    Future f = pool.submit(c);
```

2017年1月 (16)
2016年12月 (15)
2016年11月 (58)
2016年10月 (11)
2016年9月 (25)
2016年8月 (4)
2016年5月 (1)
2016年4月 (4)
2016年3月 (1)
2016年2月 (2)
2016年1月 (22)
2015年12月 (4)
2015年11月 (25)
2015年10月 (2)
2015年9月 (3)
2015年8月 (19)
2015年7月 (19)
2015年6月 (5)
2015年5月 (15)
2015年4月 (10)
2015年3月 (4)
2015年2月 (3)
2015年1月 (6)
2014年12月 (6)
2014年11月 (9)
2014年10月 (1)

积分与排名

积分 - 200823
排名 - 1307

最新评论

1. Re:java判断字符串是否为数字或中文或字母
jquery里判断是否为数字的各种方法及每种方法的优缺点

这里非常详细，还有具体的实例演示！
--aijquery.cn
2. Re:ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component 解决
老哥稳啊，果然是url-pattern过滤器中缺少/
必须赞！
--大橙子22c
3. Re:Java中继承thread类与实现Runnable接口的区别
多线程共享对象会有并发问题
--啪布里克
4. Re:JAVA多线程实现的四种方式
多线程就一种实现方式实现runnable接口
--Rock_Tao
5. Re:Zookeeper的功能以及工作原理
通俗易懂，简洁全面，不错。
--cuqing
6. Re:bin/sh^M: bad interpreter: No such file or directory解决
:set ff应该是vim的选项，aix下的vi没这个选项其实也可以直接替换那个换行符达到同样的目的：:%s/^v^m//g （aix) ^v^m代表ctrl+v , ctrl+m:%s/.....
--snailwong
7. Re:ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component 解决

```
    // System.out.println(">>>" + f.get().toString());
    list.add(f);
   }
   // 关闭线程池
   pool.shutdown();

   // 获取所有并发任务的运行结果
   for (Future f : list) {
    // 从Future对象上获取任务的返回值，并输出到控制台
    System.out.println(">>>" + f.get().toString());
   }

   Date date2 = new Date();
   System.out.println("----程序结束运行----，程序运行时间【"
     + (date2.getTime() - date1.getTime()) + "毫秒】");
}
}

class MyCallable implements Callable<Object> {
private String taskNum;

MyCallable(String taskNum) {
   this.taskNum = taskNum;
}

public Object call() throws Exception {
   System.out.println(">>>" + taskNum + "任务启动");
   Date dateTmp1 = new Date();
   Thread.sleep(1000);
   Date dateTmp2 = new Date();
   long time = dateTmp2.getTime() - dateTmp1.getTime();
   System.out.println(">>>" + taskNum + "任务终止");
   return taskNum + "任务返回运行结果,当前任务时间【" + time + "毫秒】";
}
}
```

代码说明：
上述代码中Executors类，提供了一系列工厂方法用于创建线程池，返回的线程池都实现了ExecutorService接口。
public static ExecutorService newFixedThreadPool(int nThreads)
创建固定数目线程的线程池。
public static ExecutorService newCachedThreadPool()
创建一个可缓存的线程池，调用execute 将重用以前构造的线程（如果线程可用）。如果现有线程没有可用的，则创建一个新线程并添加到池中。终止并从缓存中移除那些已有 60 秒钟未被使用的线程。
public static ExecutorService newSingleThreadExecutor()
创建一个单线程化的Executor。
public static ScheduledExecutorService newScheduledThreadPool(int corePoolSize)
创建一个支持定时及周期性的任务执行的线程池，多数情况下可用来替代Timer类。

ExecutoreService提供了submit()方法，传递一个Callable，或Runnable，返回Future。如果Executor后台线程池还没有完成Callable的计算，这调用返回Future对象的get()方法，会阻塞直到计算完成。

软件-注重思想、逻辑

« 上一篇：Java四种线程池的使用
» 下一篇：Spring官网改版后下载

posted @ 2016-11-06 19:33 FelixZh 阅读(77849) 评论(4) 编辑 收藏

评论

#1楼 2018-01-17 17:17 | 流星一箭

实现线程的方式
1 继承thread类
2 实现runnable接口
3 实现Callable接口通过FutureTask包装器来创建Thread线程

支持(0) 反对(4)

#2楼 2018-01-23 20:38 | icuke

@xuxuxu123不得不给你赞...
--FelixZh
8. Re:ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component解决
呜呜呜拼错了，找了一天的jar包。看到这忽然想起来。。。。。。。
--xuxuxu123
9. Re:JAVA多线程实现的四种方式
@icuke可以...
--FelixZh
10. Re:JAVA多线程实现的四种方式
总结的好，可以转发么？
--icuke

阅读排行榜

1. Zookeeper的功能以及工作原理(112475)
2. JAVA多线程实现的四种方式(77864)
3. ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component解决(45060)
4. java判断字符串是否为数字或中文或字母(22256)
5. SecureCRT上传和下载文件(下载默认目录)(14162)
6. Kafka、RabbitMQ、RocketMQ消息中间件的对比 —— 消息发送性能-转自阿里中间件(12939)
7. Nginx使用教程(四)：提高Nginx网络吞吐量之buffers优化(10509)
8. Nginx使用教程(二)：Nginx配置性能优化之worker配置(7732)
9. Java配置环境变量、方法和原因(7292)
10. java Clone使用方法详解(6848)

评论排行榜

1. Zookeeper的功能以及工作原理(7)
2. JAVA多线程实现的四种方式(4)
3. ContainerBase.addChild: start: org.apache.catalina.LifecycleException: Failed to start component解决(3)
4. 个人经常查阅的网站（顺序无先后，持续更新）(3)
5. 深入理解Java的接口和抽象类(2)
6. java判断字符串是否为数字或中文或字母(1)
7. java Clone使用方法详解(1)
8. Java中继承thread类与实现Runnable接口的区别(1)
9. yum使用问题：Repodata is over 2 weeks old. Install yum-cron? Or run: yum makecache fast(1)
10. Nginx使用教程(六)：使用Nginx缓存之FastCGI缓存(1)

推荐排行榜

1. Zookeeper的功能以及工作原理(13)
2. JAVA多线程实现的四种方式(12)
3. 深入理解Java的接口和抽象类(2)
4. ContainerBase.addChild: start: org.apache.catalina.LifecycleExce

总结的好，可以转发么？

支持(1) 反对(0)

#3楼[楼主] 2018-01-26 19:09 | FelixZh

@ icuke
可以

支持(1) 反对(0)

#4楼 2018-04-24 22:31 | Rock_Tao

多线程就一种实现方式实现runnable接口

支持(0) 反对(3)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:
· 腕上蓝牙耳机！华为手环B5正式发布：三个版本/999元起
· 苏宁任性付将进行品牌升级：对标蚂蚁花呗
· 京东全面赋能 曲美实体店日均销售额比6月提升53%
· 谷歌与欧盟之间8年的反垄断"爱恨情仇"
· 外媒：爱奇艺股价或在未来20个月下跌50%
» 更多新闻...

最新知识库文章:
· 危害程序员职业生涯的三大观念
· 断点单步跟踪是一种低效的调试方法
· 测试 | 让每一粒尘埃有的放矢
· 从Excel到微服务
· 如何提升你的能力？给年轻程序员的几条建议
» 更多知识库文章...

历史上的今天:
2015-11-06 高可用,完全分布式Hadoop集群HDFS和MapReduce安装配置指南
2014-11-06 Cygwin: connection closed by ::1