

• [首页](#) • [开源项目](#) • [问答](#) • [代码](#) • [博客](#) • [翻译](#) • [资讯](#) • [移动开发](#) • [招聘](#) • [城市圈](#)
当前访客身份：游客 [[登录](#) | [加入开源中国](#)] 当前访客身份：游客 [[登录](#) | [加入开源中国](#)]

在 38457 款开源软件中:

软件

软件

搜索



[laigous](#) ☐ [关注此人](#)

[关注\(3\)](#) [粉丝\(20\)](#) [积分\(108\)](#)

强化技术功底

[发送私信](#) [请教问题](#)

博客分类

- [mysql](#)(1)
- [Cassandra](#)(5)
- [java](#)(25)
- [工作日志](#)(0)
- [日常记录](#)(0)
- [linux](#)(2)
- [转贴的文章](#)(2)
- [大数据](#)(4)
- [http](#)(2)
- [android](#)(8)
- [Scala](#)(1)
- [js](#)(0)

阅读排行

1. [1. Android读写文件](#)
2. [2. Android Menu](#)
3. [3. AlertDialog Builder自定义样式](#)
4. [4. Android Service和Binder、AIDL](#)
5. [5. Apache FtpServer与Spring整合](#)
6. [6. Ubuntu编译Hadoop源码异常总结](#)
7. [7. SXSSFWorkbook用于海量数据Excel导出类](#)
8. [8. java 计算文件MD5值 大文件](#)

最新评论

- [@laigous](#) : 引用来自“冉启强”的评论哥们，最近用最新的 ha... [查看»](#)
- [@冉启强](#) : hadoop-2.7.1 [查看»](#)
- [@冉启强](#) : spark 用的 spark-1.4.1-bin-hadoop2.6:... [查看»](#)
- [@冉启强](#) : 哥们，最近用最新的 hadoop jdk1.8 spark1.4 运... [查看»](#)
- [@internetafei](#) : mark [查看»](#)
- [@laigous](#) : 引用来自“s33ker”的评论楼主，mybatis的baseD... [查看»](#)

- [@s33ker](#)：楼主，mybatis的baseDao之类的怎么没贴出来。... [查看»](#)
- [@辉](#)：多些 分享 学习了； [查看»](#)
- [@cedar997](#)：很好 很好 顶起 [查看»](#)
- [@lkfoff](#)：写的不错，从引用分析到clone，思路清晰，例子也... [查看»](#)

访客统计

- 今日访问：22
- 昨日访问：61
- 本周访问：407
- 本月访问：690
- 所有访问：33359

[空间](#) » [博客](#) x [java](#)

转 Spring Quartz的原理

发表于1年前(2014-05-05 16:00) 阅读 (865) | 评论 (0) 15人收藏此文章, [我要收藏](#)

赞1

9月19日成都 OSC 源创会正在报名，送机械键盘和开源无码内裤 

Quartz是一个大名鼎鼎的Java版开源定时调度器，功能强悍，使用方便。

一、核心概念

Quartz的原理不是很复杂，只要搞明白几个概念，然后知道如何去启动和关闭一个调度程序即可。

1、Job

表示一个工作，要执行的具体内容。此接口中只有一个方法

```
void execute(JobExecutionContext context)
```

2、JobDetail

JobDetail表示一个具体的可执行的调度程序，Job是这个可执行程调度程序所要执行的内容，另外JobDetail还包含了这个任务调度的方案和策略。

3、Trigger代表一个调度参数的配置，什么时候去调。

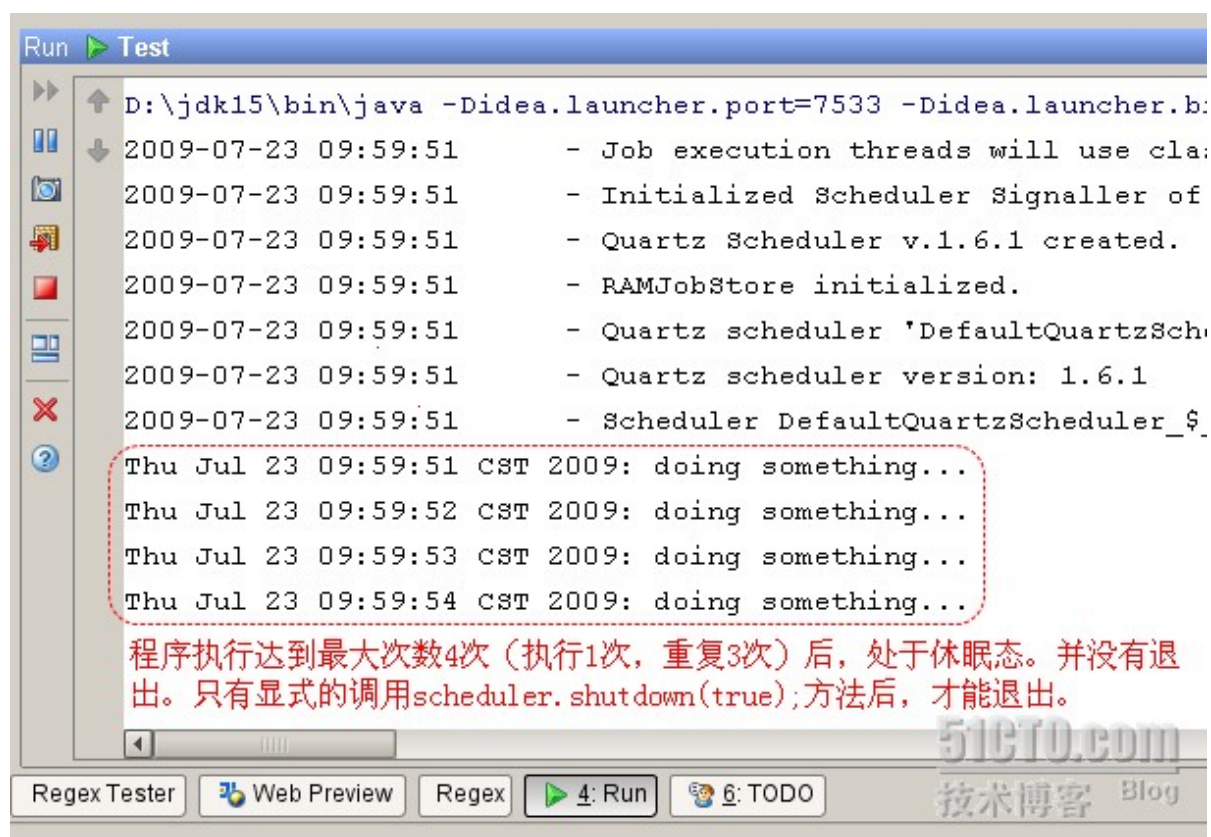
4、Scheduler代表一个调度容器，一个调度容器中可以注册多个JobDetail和Trigger。当Trigger与JobDetail组合，就可以被Scheduler容器调度了。

二、一个最简单入门实例

```
1  import org.quartz.*;
2  import org.quartz.impl.StdSchedulerFactory;
3
4  import java.util.Date;
5
6  /**
7   * quartz定时器测试
8   *
9   * @author leizhimin 2009-7-23 8:49:01
10  */
11  public class MyJob implements Job {
12      public void execute(JobExecutionContext jobExecutionContext) throws
13          System.out.println(new Date() + ": doing something...");
14  }
15
16
17  class Test {
18      public static void main(String[] args) {
19          //1、创建JobDetail对象
20          JobDetail jobDetail = new JobDetail();
21          //设置工作项
22          jobDetail.setJobClass(MyJob.class);
23          jobDetail.setName("MyJob_1");
24          jobDetail.setGroup("JobGroup_1");
25
26          //2、创建Trigger对象
27          SimpleTrigger strigger = new SimpleTrigger();
28          strigger.setName("Trigger_1");
29          strigger.setGroup("Trigger_Group_1");
30          strigger.setStartTime(new Date());
31          //设置重复停止时间，并销毁该Trigger对象
32          java.util.Calendar c = java.util.Calendar.getInstance();
33          c.setTimeInMillis(System.currentTimeMillis() + 1000 * 1L);
34          strigger.setEndTime(c.getTime());
35          strigger.setFireInstanceId("Trigger_1_id_001");
36          //设置重复间隔时间
37          strigger.setRepeatInterval(1000 * 1L);
38          //设置重复执行次数
39          strigger.setRepeatCount(3);
40
41          //3、创建Scheduler对象，并配置JobDetail和Trigger对象
42          SchedulerFactory sf = new StdSchedulerFactory();
43          Scheduler scheduler = null;
44          try {
45              scheduler = sf.getScheduler();
46              scheduler.scheduleJob(jobDetail, strigger);
47              //4、并执行启动、关闭等操作
48              scheduler.start();
49          }
```

```
50         } catch (SchedulerException e) {
51             e.printStackTrace();
52         }
53         try {
54             //关闭调度器
55             scheduler.shutdown(true);
56         } catch (SchedulerException e) {
57             e.printStackTrace();
58         }
59     }
60 }
```

执行结果：

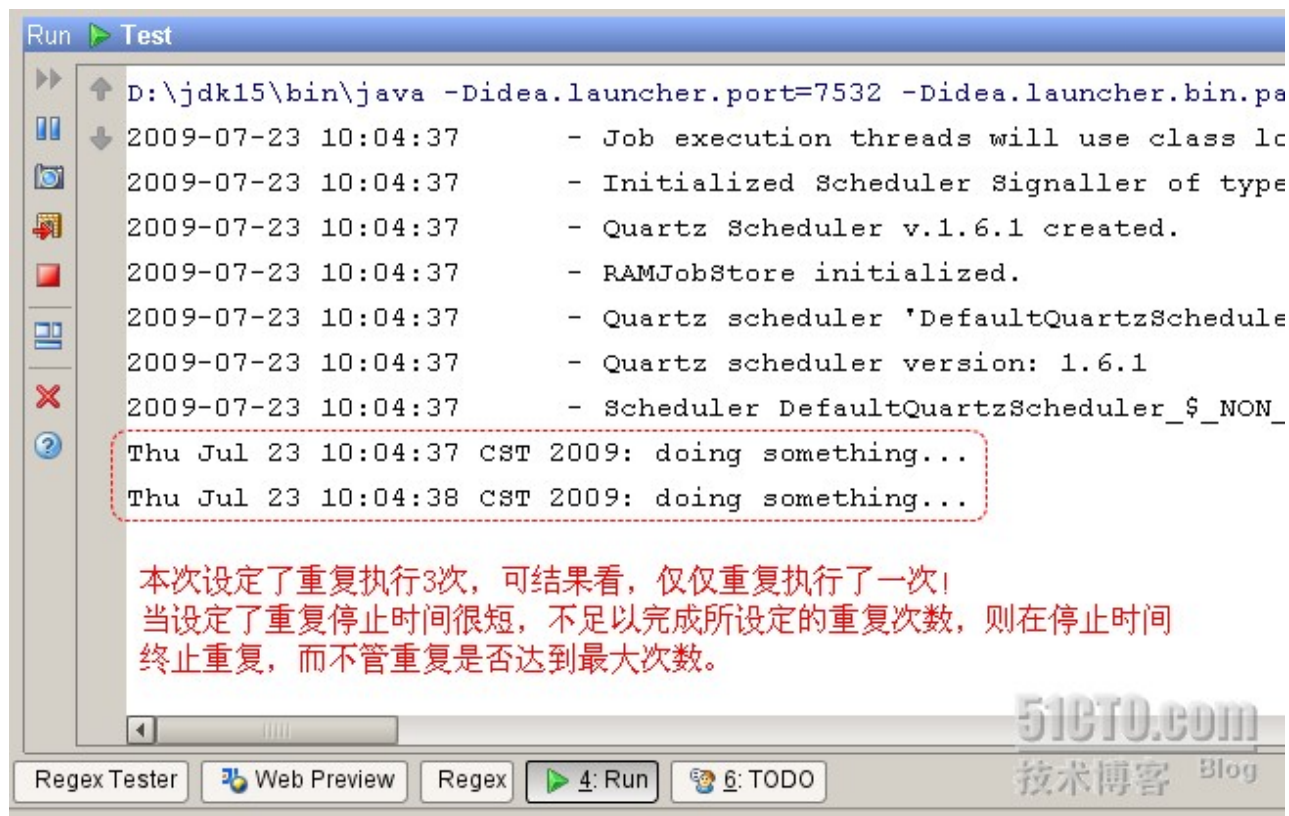


```
Run ▶ Test
D:\jdk15\bin\java -Didea.launcher.port=7533 -Didea.launcher.b:
2009-07-23 09:59:51 - Job execution threads will use cla:
2009-07-23 09:59:51 - Initialized Scheduler Signaller of
2009-07-23 09:59:51 - Quartz Scheduler v.1.6.1 created.
2009-07-23 09:59:51 - RAMJobStore initialized.
2009-07-23 09:59:51 - Quartz scheduler 'DefaultQuartzSch:
2009-07-23 09:59:51 - Quartz scheduler version: 1.6.1
2009-07-23 09:59:51 - Scheduler DefaultQuartzScheduler_$
Thu Jul 23 09:59:51 CST 2009: doing something...
Thu Jul 23 09:59:52 CST 2009: doing something...
Thu Jul 23 09:59:53 CST 2009: doing something...
Thu Jul 23 09:59:54 CST 2009: doing something...
程序执行达到最大次数4次（执行1次，重复3次）后，处于休眠态。并没有退出。只有显式的调用scheduler.shutdown(true);方法后，才能退出。
```

当把结束时间改为：

```
//设置重复停止时间，并销毁该Trigger对象
java.util.Calendar c = java.util.Calendar.getInstance();
c.setTimeInMillis(System.currentTimeMillis() + 1000 * 1L);
trigger.setEndTime(c.getTime());
```

执行结果：



当添加一条关闭调度器的语句：

```
//4、并执行启动、关闭等操作  
scheduler.start();  
scheduler.shutdown(true);
```

程序执行结果：

Thu Jul 23 10:11:50 CST 2009: doing something...

Process finished with exit code 0

仅仅执行了一次，这一次能执行完，原因是设定了scheduler.shutdown(true);true表示等待本次任务执行完成后停止。

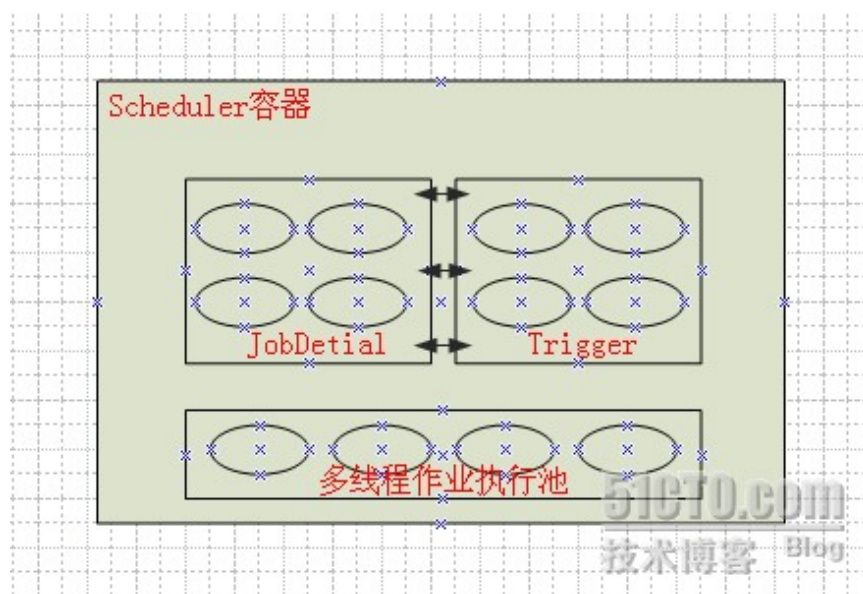
从这里也可以看出，scheduler是个容器，scheduler控制jobDetail的执行，控制的策略是通过trigger。

当scheduler容器启动后，jobDetail才能根据关联的trigger策略去执行。当scheduler容器关闭后，所有的jobDetail都停止执行。

三、透过实例看原理

通过研读Quartz的源代码，和本实例，终于悟出了Quartz的工作原理。

- 1、scheduler是一个计划调度器容器（总部），容器里面可以盛放众多的JobDetail和trigger，当容器启动后，里面的每个JobDetail都会根据trigger按部就班自动去执行。
- 2、JobDetail是一个可执行的工作，它本身可能是有状态的。
- 3、Trigger代表一个调度参数的配置，什么时候去调。
- 4、当JobDetail和Trigger在scheduler容器上注册后，形成了装配好的作业（JobDetail和Trigger所组成的一对儿），就可以伴随容器启动而调度执行了。
- 5、scheduler是个容器，容器中有一个线程池，用来并行调度执行每个作业，这样可以提高容器效率。
- 6、将上述的结构用一个图来表示，如下：



四、总结

- 1、搞清楚了上Quartz容器执行作业的的原理和过程，以及作业形成的方式，作业注册到容器的方法。就认识明白了Quartz的核心原理。
- 2、Quartz虽然很庞大，但是一切都围绕这个核心转，为了配置强大时间调度策略，可以研究专门的CronTrigger。要想灵活配置作业和容器属性，可以通过Quartz的properties文件或者XML来实现。
- 3、要想调度更多的持久化、结构化作业，可以通过数据库读取作业，然后放到容器中执行。
- 4、所有的一切都围绕这个核心原理转，搞明白这个了，再去研究更高级用法就容易多了。
- 5、Quartz与Spring的整合也非常简单，Spring提供一组Bean来支持：MethodInvokingJobDetailFactoryBean、SimpleTriggerBean、SchedulerFactoryBean，看看里面需要注入什么属性即可明白了。Spring会在Spring容器启动时候，启动Quartz容器。
- 6、Quartz容器的关闭方式也很简单，如果是Spring整合，则有两种方法，一种是关闭Spring容器，一种是获取到SchedulerFactoryBean实例，然后调用一个shutdown就搞定了。如果是Quartz独立使用，则直接调用scheduler.shutdown(true);
- 7、Quartz的JobDetail、Trigger都可以在运行时重新设置，并且在下次调用时候起作用。这就为动态作业的实现提供了依据。你可以将调度时间策略存放到数据库，然后通过数据库数据来设定Trigger，这样就能产生动态的调度。

分享到： 新浪微博  腾讯微博 1赞

原文地址：<http://lavasoft.blog.51cto.com/62575/181907/>

- [« 上一篇](#)
- [下一篇 »](#)

最新热门职位

更多开发者职位上 开源中国·人才

java工程师 耐固车品

中级JAVA工程师 乐钱

耐 乐	月薪：10-15K	乐	月薪：8-16K
	初级JAVA工程师 乐钱		
	月薪：6-10K		

评论0



插入：[表情](#) [开源软件](#)

发表评论

[关闭](#)插入表情

[关闭](#)相关文章阅读

- 2012/07/25 [spring quartz](#)
- 2014/09/20 [Spring Quartz任务调度](#)
- 2012/07/24 [spring quartz 时间配置格式...](#)
- 2012/12/04 [Spring配置quartz调度任务](#)
- 2014/10/23 [spring+quartz简单整合](#)

© 开源中国(OSChina.NET) | [关于我们](#) | [广告联系](#) | [@新浪微博](#) | 开源中国手机客户端：
[开源中国手机版](#) | 粤ICP备12009483号-3

开源中国社区(OSChina.net)是工信部 [开源软件推进联盟](#) 指定的官方社区