**什么是Quartz**

[**Quartz**](http://www.quartz-scheduler.org/)是一个完全由Java编写的开源作业调度框架，为在Java应用程序中进行作业调度提供了简单却强大的机制。Quartz允许开发人员根据时间间隔来调度作业。它实现了作业和触发器的多对多的关系，还能把多个作业与不同的触发器关联。简单地创建一个org.quarz.Job接口的Java类，Job接口包含唯一的方法：

**public** **void** execute(JobExecutionContext context) **throws** JobExecutionException;

在Job接口实现类里面，添加需要的逻辑到execute()方法中。配置好Job实现类并设定好调度时间表，Quartz就会自动在设定的时间调度作业执行execute()。

整合了Quartz的应用程序可以重用不同事件的作业，还可以为一个事件组合多个作业。Quartz通过属性文件来配置JDBC事务的数据源、全局作业、触发器侦听器、插件、线程池等等。

Quartz是由James House创建并最初于2001年春天被加入sourceforge工程。接下来的几年里，有很多的新特性和版本出现，但是直到项目迁移到新的站点并成为OpenSymphony项目家族的一员，才开始真正启动并受到也有的关注。

目前的版本已经是2.0以上，v2.x相对于v1.x有很多新特性出现，并有很多的改动，具体参见Quartz官网上说明。这里介绍的仍然是v1.x（v1.8.6）。

"Hello, Quartz"

配置环境：

1. 下载Quartz

2. 阅读Readme.txt，了解每个jar包的作用，将quartz.jar包和lib/下的几个jar包、以及相关依赖的jar包放在工程的classpath中

先来看一个简单的Quartz应用，让它每隔5s打印"Hello, Quartz"，打印10次。

代码清单1：创建任务

**import** java.util.Date;  
**import** org.quartz.Job;  
**import** org.quartz.JobExecutionContext;  
**import** org.quartz.JobExecutionException;  
  
**public** **class** HelloQuartzJob **implements** Job {  
  
    **public** **void** execute(JobExecutionContext context)   
            **throws** JobExecutionException {  
        System.out.println("Hello, Quartz! - executing its JOB at "+   
            **new** Date() + " by " + context.getTrigger().getName());  
    }  
}

为了调度此任务执行，需要先得到一个Schedule实例，然后创建一个包含任务信息的JobDetail，最后创建一个Trigger管理任务的执行。

代码清单2：调度任务

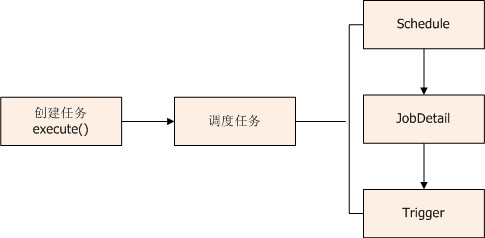
**import** java.sql.Date;   
**import** org.quartz.JobDetail;  
**import** org.quartz.Scheduler;  
**import** org.quartz.SchedulerException;  
**import** org.quartz.SchedulerFactory;  
**import** org.quartz.SimpleTrigger;  
**import** org.quartz.impl.StdSchedulerFactory;  
  
  
**public** **class** HelloQuartzScheduling {  
  
    **public** **static** **void** main(String[] args)**throws** SchedulerException {  
  
        SchedulerFactory schedulerFactory = **new** StdSchedulerFactory();  
        Scheduler scheduler = schedulerFactory.getScheduler();  
  
        JobDetail jobDetail = **new** JobDetail("helloQuartzJob",   
                Scheduler.DEFAULT\_GROUP, HelloQuartzJob.**class**);  
  
        SimpleTrigger simpleTrigger = **new** SimpleTrigger("simpleTrigger",   
                Scheduler.DEFAULT\_GROUP);  
  
        simpleTrigger.setStartTime(**new** Date(System.currentTimeMillis()));  
        simpleTrigger.setRepeatInterval(5000);  
        simpleTrigger.setRepeatCount(10);  
  
        scheduler.scheduleJob(jobDetail, simpleTrigger);  
  
        scheduler.start();  
    }  
  
}

运行结果：

http://img8.ph.126.net/LSom4rSegQ74bfTQjROH5Q==/2393944676941408638.jpg

可以看到，其实它执行了11次。此处没有配置log4j.properties属性文件。

整个任务创建及调度的简单示意图如下。



Job接口包含唯一方法execute()，将任务逻辑添加到该方法中。StdSchedulerFactory.getScheduler()返回一个可运行的实例，然后创建调度任务的JobDetail实例，并传递3个参数给构造方法。第一个参数是任务名，用于引用该任务。第二个参数是任务组名，这里使用默认名，任务组名用于引用集合起来的一组任务，如可以使用Scheduler.pauseJobGroup()来暂停一组任务，每个组中的任务名是唯一的。第三个参数是实现特定任务的类。创建JobDetail实例后，需要创建一个Trigger，这里使用的是SimpleTrigger类，它提供了JDK Timer风格的触发器行为。传递给SimpleTrigger构造方法的两个参数分别是触发器名和任务组名，触发器名在它所在的任务组中必须是唯一的。接下来是设置触发器的一些属性，setStartTime()是设置启动时间，setRepeatInterval()是设置重复间隔，setRepeatCount()是设置重复次数。最后，scheduler.start()启动调度，终止调度可以用stop()方法。

**CronTrigger类**

Quartz有两大触发器，除了上面使用的SimpleTrigger外，就是CronTrigger。CronTrigger能够提供复杂的触发器表达式的支持。CronTrigger是基于Unix Cron守护进程，它是一个调度程序，支持简单而强大的触发器语法。

使用CronTrigger主要的是要掌握Cron表达式。Cron表达式包含6个必要组件和一个可选组件，如下表所示。

|  |  |  |
| --- | --- | --- |
| 位置 | 含义 | 允许的特殊字符 |
| 1 | 秒（0~59） | , -  \*  / |
| 2 | 分（0~59） | , -  \*  / |
| 3 | 小时（0~24） | , -  \*  / |
| 4 | 日期（1~31） | , -  \*  /  ?  L  W  C |
| 5 | 月（JAN~DEC或1~12） | , -  \*  / |
| 6 | 星期（SUN~SAT或1~7） | , -  \*  /  ?  L  C  # |
| 7 | 年（可选，1970~2099），若为空，表示全部时间范围 | , -  \*  / |

特殊字符的含义，见下表。

|  |  |
| --- | --- |
| 特殊字符 | 说明 |
| \* | 通配符，任意值 |
| ? | 无特定值。通常和其他指定的值一起使用，表示必须显示该值但不能检查 |
| - | 范围。e.g.小时部分10-12表示10:00，11:00， 12:00 |
| , | 列分隔符。可以让你指定一系列的值。e.g.在星期域中指定MON、TUE和WED |
| / | 增量。表示一个值的增量，e.g.分钟域中0/1表示从0开始，每次增加1min |
| L | 表示Last。它在日期和星期域中表示有所不同。在日期域中，表示这个月的最后一天，而在星期域中，它永远是7（星期六）。当你希望使用星期中某一天时，L字符非常有用。e.g.星期域中6L表示每一个月的最后一个星期五 |
| W | 在本月内离当天最近的工作日触发，所谓的最近工作日，即当天到工作日的前后最短距离，如果当天即为工作日，则距离是0；所谓本月内指的是不能跨月取到最近工作日，即使前/后月份的最后一天/第一天确实满足最近工作日。e.g. LW表示本月的最后一个工作日触发，W强烈依赖月份。 |
| # | 表示该月的第几个星期，e.g. 1#2表示每一个月的第一个星期一 |
| C | 日历值。日期值是根据一个给定的日历计算出来的。在日期域中给定一个20C将在20日（日历包括20日）或20日后日历中包含的第一天（不包括20日）激活触发器。例如在一个星期域中使用6C表示日历中星期五（日历包括星期五）或者第一天（日历不包括星期五） |

Cron表达式举例：

"30 \* \* \* \* ?" 每半分钟触发任务

"30 10 \* \* \* ?" 每小时的10分30秒触发任务

"30 10 1 \* \* ?" 每天1点10分30秒触发任务

"30 10 1 20 \* ?" 每月20号1点10分30秒触发任务

"30 10 1 20 10 ? \*" 每年10月20号1点10分30秒触发任务

"30 10 1 20 10 ? 2011" 2011年10月20号1点10分30秒触发任务

"30 10 1 ? 10 \* 2011" 2011年10月每天1点10分30秒触发任务

"30 10 1 ? 10 SUN 2011" 2011年10月每周日1点10分30秒触发任务

"15,30,45 \* \* \* \* ?" 每15秒，30秒，45秒时触发任务

"15-45 \* \* \* \* ?" 15到45秒内，每秒都触发任务

"15/5 \* \* \* \* ?" 每分钟的每15秒开始触发，每隔5秒触发一次

"15-30/5 \* \* \* \* ?" 每分钟的15秒到30秒之间开始触发，每隔5秒触发一次

"0 0/3 \* \* \* ?" 每小时的第0分0秒开始，每三分钟触发一次

"0 15 10 ? \* MON-FRI" 星期一到星期五的10点15分0秒触发任务

"0 15 10 L \* ?" 每个月最后一天的10点15分0秒触发任务

"0 15 10 LW \* ?" 每个月最后一个工作日的10点15分0秒触发任务

"0 15 10 ? \* 5L" 每个月最后一个星期四的10点15分0秒触发任务

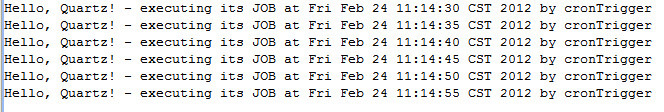
"0 15 10 ? \* 5#3" 每个月第三周的星期四的10点15分0秒触发任务

将上面HelloQuartz例子中SimpleTrigger换成CronTrigger，代码如下。

代码清单3：CronTrigger调度器

**import** java.text.ParseException;  
**import** org.quartz.CronTrigger;  
**import** org.quartz.JobDetail;  
**import** org.quartz.Scheduler;  
**import** org.quartz.SchedulerException;  
**import** org.quartz.SchedulerFactory;  
**import** org.quartz.impl.StdSchedulerFactory;  
  
**public** **class** HelloQuartzScheduling {  
  
    **public** **static** **void** main(String[] args)   
        **throws** SchedulerException, ParseException {  
  
        SchedulerFactory schedulerFactory = **new** StdSchedulerFactory();  
        Scheduler scheduler = schedulerFactory.getScheduler();  
  
        JobDetail jobDetail = **new** JobDetail("helloQuartzJob",   
                Scheduler.DEFAULT\_GROUP, HelloQuartzJob.**class**);  
  
        **String cronExpression = "30/5 \* \* \* \* ?";** // 每分钟的30s起，每5s触发任务          
        CronTrigger cronTrigger = **new** CronTrigger("cronTrigger",   
                Scheduler.DEFAULT\_GROUP, cronExpression);  
  
        scheduler.scheduleJob(jobDetail, cronTrigger);  
  
        scheduler.start();  
    }  
  
}

运行结果：



CronTrigger使用HolidayCalendar类可以排除某一段时间，比如说国庆节不执行调度任务，代码示例如下：

代码清单4：HolidayCalendar的使用

**import** java.text.ParseException;  
**import** java.util.Calendar;  
**import** org.quartz.CronTrigger;  
**import** org.quartz.JobDetail;  
**import** org.quartz.Scheduler;  
**import** org.quartz.SchedulerException;  
**import** org.quartz.SchedulerFactory;  
**import** org.quartz.impl.StdSchedulerFactory;  
**import** org.quartz.impl.calendar.HolidayCalendar;  
  
  
**public** **class** HelloQuartzScheduling {  
  
    **public** **static** **void** main(String[] args)   
        **throws** SchedulerException, ParseException {  
  
        SchedulerFactory schedulerFactory = **new** StdSchedulerFactory();  
        Scheduler scheduler = schedulerFactory.getScheduler();  
  
        JobDetail jobDetail = **new** JobDetail("helloQuartzJob",   
                Scheduler.DEFAULT\_GROUP, HelloQuartzJob.**class**);  
  
        Calendar cal = Calendar.getInstance();  
        cal.set(2012, Calendar.OCTOBER, 1); // 国庆节  
  
        HolidayCalendar holidayCal = **new** HolidayCalendar();  
        holidayCal.addExcludedDate(cal.getTime()); // 排除该日期  
  
        // addCalendar(String calName, Calendar calendar,   
        //             boolean replace, boolean updateTriggers)  
        scheduler.addCalendar("calendar", holidayCal, true, false);  
  
        String cronExpression = "30/5 \* \* \* \* ?"; // 每5s触发任务          
        CronTrigger cronTrigger = **new** CronTrigger("cronTrigger",   
                Scheduler.DEFAULT\_GROUP, cronExpression);  
  
        cronTrigger.setCalendarName("calendar");  
  
        scheduler.scheduleJob(jobDetail, cronTrigger);  
  
        scheduler.start();  
    }  
  
}

**JobStore: 任务持久化**

Quartz支持任务持久化，这可以让你在运行时增加任务或者对现存的任务进行修改，并为后续任务的执行持久化这些变更和增加的部分。中心概念是JobStore接口。默认的是RAMJobStore。

[Quartz使用-入门使用(java定时任务实现)](http://blog.csdn.net/lotusyangjun/article/details/6450421)

标签： [任务](http://www.csdn.net/tag/%e4%bb%bb%e5%8a%a1)[quartz](http://www.csdn.net/tag/quartz)[java](http://www.csdn.net/tag/java)[string](http://www.csdn.net/tag/string)[javamail](http://www.csdn.net/tag/javamail)[date](http://www.csdn.net/tag/date)

2011-05-27 16:20 45618人阅读 [评论](http://blog.csdn.net/lotusyangjun/article/details/6450421/#comments)(6) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lotusyangjun/article/details/6450421/#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

java学习（14） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

注：这里使用的是Quartz1.6.5版本(包：quartz-1.6.5.jar)

**[java]** [view plain](http://blog.csdn.net/lotusyangjun/article/details/6450421/) [copy](http://blog.csdn.net/lotusyangjun/article/details/6450421/)

1. //测试main函数
2. //QuartzTest.java
3. **package** quartzPackage;

6. **import** java.text.SimpleDateFormat;
7. **import** java.util.Date;
9. **public** **class** QuartzTest {
11. /\*\* \*//\*\*
12. \* @param args
13. \*/
14. **public** **static** **void** main(String[] args) {
15. // TODO Auto-generated method stub
16. SimpleDateFormat DateFormat = **new** SimpleDateFormat("yyyyMMddHHmmss");
17. Date d = **new** Date();
18. String returnstr = DateFormat.format(d);
20. TestJob job = **new** TestJob();
21. String job\_name ="11";
22. **try** {
23. System.out.println(returnstr+ "【系统启动】");
24. QuartzManager.addJob(job\_name,job,"0/2 \* \* \* \* ?"); //每2秒钟执行一次
26. //            Thread.sleep(10000);
27. //            System.out.println("【修改时间】");
28. //            QuartzManager.modifyJobTime(job\_name,"0/10 \* \* \* \* ?");
29. //            Thread.sleep(20000);
30. //            System.out.println("【移除定时】");
31. //            QuartzManager.removeJob(job\_name);
32. //            Thread.sleep(10000);
33. //
34. //            System.out.println("/n【添加定时任务】");
35. //            QuartzManager.addJob(job\_name,job,"0/5 \* \* \* \* ?");
37. }  **catch** (Exception e) {
38. e.printStackTrace();
39. }
40. }
41. }

简单的任务管理类

**[java]** [view plain](http://blog.csdn.net/lotusyangjun/article/details/6450421/) [copy](http://blog.csdn.net/lotusyangjun/article/details/6450421/)

1. //简单的任务管理类
2. //QuartzManager.java
4. **package** quartzPackage;
6. **import** java.text.ParseException;
7. **import** org.quartz.CronTrigger;
8. **import** org.quartz.Job;
9. **import** org.quartz.JobDetail;
10. **import** org.quartz.Scheduler;
11. **import** org.quartz.SchedulerException;
12. **import** org.quartz.SchedulerFactory;
13. **import** org.quartz.Trigger;
14. **import** org.quartz.impl.StdSchedulerFactory;
16. /\*\* \*//\*\*
17. \* @Title:Quartz管理类
18. \*
19. \* @Description:
20. \*
21. \* @Copyright:
22. \* @author zz  2008-10-8 14:19:01
23. \* @version 1.00.000
24. \*
25. \*/
26. **public** **class** QuartzManager {
27. **private** **static** SchedulerFactory sf = **new** StdSchedulerFactory();
28. **private** **static** String JOB\_GROUP\_NAME = "group1";
29. **private** **static** String TRIGGER\_GROUP\_NAME = "trigger1";

32. /\*\* \*//\*\*
33. \*  添加一个定时任务，使用默认的任务组名，触发器名，触发器组名
34. \* @param jobName 任务名
35. \* @param job     任务
36. \* @param time    时间设置，参考quartz说明文档
37. \* @throws SchedulerException
38. \* @throws ParseException
39. \*/
40. **public** **static** **void** addJob(String jobName,Job job,String time)
41. **throws** SchedulerException, ParseException{
42. Scheduler sched = sf.getScheduler();
43. JobDetail jobDetail = **new** JobDetail(jobName, JOB\_GROUP\_NAME, job.getClass());//任务名，任务组，任务执行类
44. //触发器
45. CronTrigger  trigger =
46. **new** CronTrigger(jobName, TRIGGER\_GROUP\_NAME);//触发器名,触发器组
47. trigger.setCronExpression(time);//触发器时间设定
48. sched.scheduleJob(jobDetail,trigger);
49. //启动
50. **if**(!sched.isShutdown())
51. sched.start();
52. }
54. /\*\* \*//\*\*
55. \* 添加一个定时任务
56. \* @param jobName 任务名
57. \* @param jobGroupName 任务组名
58. \* @param triggerName  触发器名
59. \* @param triggerGroupName 触发器组名
60. \* @param job     任务
61. \* @param time    时间设置，参考quartz说明文档
62. \* @throws SchedulerException
63. \* @throws ParseException
64. \*/
65. **public** **static** **void** addJob(String jobName,String jobGroupName,
66. String triggerName,String triggerGroupName,
67. Job job,String time)
68. **throws** SchedulerException, ParseException{
69. Scheduler sched = sf.getScheduler();
70. JobDetail jobDetail = **new** JobDetail(jobName, jobGroupName, job.getClass());//任务名，任务组，任务执行类
71. //触发器
72. CronTrigger  trigger =
73. **new** CronTrigger(triggerName, triggerGroupName);//触发器名,触发器组
74. trigger.setCronExpression(time);//触发器时间设定
75. sched.scheduleJob(jobDetail,trigger);
76. **if**(!sched.isShutdown())
77. sched.start();
78. }
80. /\*\* \*//\*\*
81. \* 修改一个任务的触发时间(使用默认的任务组名，触发器名，触发器组名)
82. \* @param jobName
83. \* @param time
84. \* @throws SchedulerException
85. \* @throws ParseException
86. \*/
87. **public** **static** **void** modifyJobTime(String jobName,String time)
88. **throws** SchedulerException, ParseException{
89. Scheduler sched = sf.getScheduler();
90. Trigger trigger =  sched.getTrigger(jobName,TRIGGER\_GROUP\_NAME);
91. **if**(trigger != **null**){
92. CronTrigger  ct = (CronTrigger)trigger;
93. ct.setCronExpression(time);
94. sched.resumeTrigger(jobName,TRIGGER\_GROUP\_NAME);
95. }
96. }
98. /\*\* \*//\*\*
99. \* 修改一个任务的触发时间
100. \* @param triggerName
101. \* @param triggerGroupName
102. \* @param time
103. \* @throws SchedulerException
104. \* @throws ParseException
105. \*/
106. **public** **static** **void** modifyJobTime(String triggerName,String triggerGroupName,
107. String time)
108. **throws** SchedulerException, ParseException{
109. Scheduler sched = sf.getScheduler();
110. Trigger trigger =  sched.getTrigger(triggerName,triggerGroupName);
111. **if**(trigger != **null**){
112. CronTrigger  ct = (CronTrigger)trigger;
113. //修改时间
114. ct.setCronExpression(time);
115. //重启触发器
116. sched.resumeTrigger(triggerName,triggerGroupName);
117. }
118. }
120. /\*\* \*//\*\*
121. \* 移除一个任务(使用默认的任务组名，触发器名，触发器组名)
122. \* @param jobName
123. \* @throws SchedulerException
124. \*/
125. **public** **static** **void** removeJob(String jobName)
126. **throws** SchedulerException{
127. Scheduler sched = sf.getScheduler();
128. sched.pauseTrigger(jobName,TRIGGER\_GROUP\_NAME);//停止触发器
129. sched.unscheduleJob(jobName,TRIGGER\_GROUP\_NAME);//移除触发器
130. sched.deleteJob(jobName,JOB\_GROUP\_NAME);//删除任务
131. }
133. /\*\* \*//\*\*
134. \* 移除一个任务
135. \* @param jobName
136. \* @param jobGroupName
137. \* @param triggerName
138. \* @param triggerGroupName
139. \* @throws SchedulerException
140. \*/
141. **public** **static** **void** removeJob(String jobName,String jobGroupName,
142. String triggerName,String triggerGroupName)
143. **throws** SchedulerException{
144. Scheduler sched = sf.getScheduler();
145. sched.pauseTrigger(triggerName,triggerGroupName);//停止触发器
146. sched.unscheduleJob(triggerName,triggerGroupName);//移除触发器
147. sched.deleteJob(jobName,jobGroupName);//删除任务
148. }
149. }

测试工作类

**[java]** [view plain](http://blog.csdn.net/lotusyangjun/article/details/6450421/) [copy](http://blog.csdn.net/lotusyangjun/article/details/6450421/)

1. //测试工作类
2. //TestJob.java
4. **package** quartzPackage;
6. **import** java.text.SimpleDateFormat;
7. **import** java.util.Date;
9. **import** org.quartz.Job;
10. **import** org.quartz.JobExecutionContext;
11. **import** org.quartz.JobExecutionException;
13. **public** **class** TestJob **implements** Job {
14. SimpleDateFormat DateFormat = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
15. Date d = **new** Date();
16. String returnstr = DateFormat.format(d);
18. **public** **void** execute(JobExecutionContext arg0) **throws** JobExecutionException {
19. // TODO Auto-generated method stub
20. System.out.println(returnstr+"★★★★★★★★★★★");
21. }
23. }

第三方包

通过测试。一个简单任务只需要以下几个包：commons-beanutils.jar、commons-collections.jar、commons-logging.jar、commons-digester.jar、quartz.jar即可

|  |  |  |
| --- | --- | --- |
| 名称 | 必须/备注 | 网址 |
| activation.jar | 主要是 JavaMail 要用到 | <http://java.sun.com/products/javabeans/glasgow/jaf.html> |
| commons-beanutils.jar | 是 | <http://jakarta.apache.org/commons/beanutils> |
| commons-collections.jar | 是 | <http://jakarta.apache.org/commons/collections> |
| commons-dbcp-1.1.jar | 是，假如用到数据库作为作业存储 | <http://jakarta.apache.org/commons/dbcp> |
| commons-digester.jar | 是 | 假如你使用了某些插件，就需要它 |
| commons-logging.jar | 是 | <http://jakarta.apache.org/commons/logging/> |
| commons-pool-1.1.jar |  | <http://jakarta.apache.org/commons/pool/> |
| javamail.jar | 发送 e-mail 用 | <http://java.sun.com/products/javamail/> |
| jdbc2\_0-stdext.jar | 是，假如用到数据库作为作业存储 | <http://java.sun.com/products/jdbc/> |
| jta.jar | 是，假如用到数据库作为作业存储 | <http://java.sun.com/products/jta/database> |
| quartz.jar | 是 | Quart 框架核心包 |
| servlet.jar | 假如使用了Servlet 容器，但容器中应该存在 | <http://java.sun.com/products/servlet/> |
| log4j.jar | 是，日志 | <http://logging.apache.org/> |