# Spring定时任务的几种实现

近日项目开发中需要执行一些定时任务，比如需要在每天凌晨时候，分析一次前一天的日志信息，借此机会整理了一下定时任务的几种实现方式，

由于项目采用spring框架，所以我都将结合spring框架来介绍。

## 一．分类

从实现的技术上来分类，目前主要有三种技术（或者说有三种产品）：

1、Java自带的java.util.Timer类，这个类允许你调度一个java.util.TimerTask任务。使用这种方式可以让你的程序按照某一个频度执行，但不能在指定时间运行。

一般用的较少，这篇文章将不做详细介绍。

2、使用Quartz，这是一个功能比较强大的的调度器，可以让你的程序在指定时间执行，也可以按照某一个频度执行，配置起来稍显复杂，稍后会详细介绍。

3、Spring3.0以后自带的task，可以将它看成一个轻量级的Quartz，而且使用起来比Quartz简单许多，稍后会介绍。

从作业类的继承方式来讲，可以分为两类：

1、作业类需要继承自特定的作业类基类，如Quartz中需要继承自org.springframework.scheduling.quartz.QuartzJobBean；java.util.Timer中需要继承自java.util.TimerTask。

2、作业类即普通的java类，不需要继承自任何基类。

注:个人推荐使用第二种方式，因为这样所有的类都是普通类，不需要事先区别对待。

从任务调度的触发时机来分，这里主要是针对作业使用的触发器，主要有以下两种：

1、每隔指定时间则触发一次，在Quartz中对应的触发器为：org.springframework.scheduling.quartz.SimpleTriggerBean

2、每到指定时间则触发一次，在Quartz中对应的调度器为：org.springframework.scheduling.quartz.CronTriggerBean

注：并非每种任务都可以使用这两种触发器，如java.util.TimerTask任务就只能使用第一种。Quartz和spring task都可以支持这两种触发条件。

## 二．用法说明

详细介绍每种任务调度工具的使用方式，包括Quartz和spring task两种。

### Quartz

**第一种，作业类继承自特定的基类：org.springframework.scheduling.quartz.QuartzJobBean。**

第一步：定义作业类

Java代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. import org.quartz.JobExecutionContext;
3. import org.quartz.JobExecutionException;
5. import org.springframework.scheduling.quartz.QuartzJobBean;
7. public class Job1 extends QuartzJobBean {


11. private int timeout;
13. private static int i = 0;
15. //调度工厂实例化后，经过timeout时间开始执行调度
17. public void setTimeout(int timeout) {
19. this.timeout = timeout;
21. }




27. @Override
29. protected void executeInternal(JobExecutionContext context)
31. throws JobExecutionException {
33. System.out.println("定时任务执行中…");
35. }
37. }

 第二步：spring配置文件中配置作业类JobDetailBean

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** name="job1" class="org.springframework.scheduling.quartz.JobDetailBean"**>**
3. **<property** name="jobClass" value="com.gy.Job1" **/>**
5. **<property** name="jobDataAsMap"**>**
7. **<map>**
9. **<entry** key="timeout" value="0" **/>**
11. **</map>**
13. **</property>**
15. **</bean>**

 说明：org.springframework.scheduling.quartz.JobDetailBean有两个属性，jobClass属性即我们在java代码中定义的任务类，jobDataAsMap属性即该任务类中需要注入的属性值。

第三步：配置作业调度的触发方式（触发器）

Quartz的作业触发器有两种，分别是

org.springframework.scheduling.quartz.SimpleTriggerBean

org.springframework.scheduling.quartz.CronTriggerBean

                 第一种SimpleTriggerBean，只支持按照一定频度调用任务，如每隔30分钟运行一次。

       配置方式如下：

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean"**>**
3. **<property** name="jobDetail" ref="job1" **/>**
5. **<property** name="startDelay" value="0" **/>**<!-- 调度工厂实例化后，经过0秒开始执行调度 -->
7. **<property** name="repeatInterval" value="2000" **/>**<!-- 每2秒调度一次 -->
9. **</bean>**

第二种CronTriggerBean，支持到指定时间运行一次，如每天12:00运行一次等。

      配置方式如下：

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean"**>**
3. **<property** name="jobDetail" ref="job1" **/>**
5. <!—每天12:00运行一次 --**>**
7. **<property** name="cronExpression" value="0 0 12 \* \* ?" **/>**
9. **</bean>**

 关于cronExpression表达式的语法参见附录。

第四步：配置调度工厂

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** class="org.springframework.scheduling.quartz.SchedulerFactoryBean"**>**
3. **<property** name="triggers"**>**
5. **<list>**
7. **<ref** bean="cronTrigger" **/>**
9. **</list>**
11. **</property>**
13. **</bean>**

 说明：该参数指定的就是之前配置的触发器的名字。

第五步：启动你的应用即可，即将工程部署至tomcat或其他容器。

**第二种，作业类不继承特定基类。**

Spring能够支持这种方式，归功于两个类：

org.springframework.scheduling.timer.MethodInvokingTimerTaskFactoryBean

org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean

这两个类分别对应spring支持的两种实现任务调度的方式，即前文提到到java自带的timer task方式和Quartz方式。这里我只写MethodInvokingJobDetailFactoryBean的用法，使用该类的好处是,我们的任务类不再需要继承自任何类，而是普通的pojo。

第一步：编写任务类

Java代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. public class Job2 {
3. public void doJob2() {
5. System.out.println("不继承QuartzJobBean方式-调度进行中...");
7. }
9. }

 可以看出，这就是一个普通的类，并且有一个方法。

第二步：配置作业类

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** id="job2"
3. class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"**>**
5. **<property** name="targetObject"**>**
7. **<bean** class="com.gy.Job2" **/>**
9. **</property>**
11. **<property** name="targetMethod" value="doJob2" **/>**
13. **<property** name="concurrent" value="false" **/>**<!-- 作业不并发调度 -->
15. **</bean>**

 说明：这一步是关键步骤，声明一个MethodInvokingJobDetailFactoryBean，有两个关键属性：targetObject指定任务类，targetMethod指定运行的方法。往下的步骤就与方法一相同了，为了完整，同样贴出。

第三步：配置作业调度的触发方式（触发器）

Quartz的作业触发器有两种，分别是

org.springframework.scheduling.quartz.SimpleTriggerBean

org.springframework.scheduling.quartz.CronTriggerBean

                第一种SimpleTriggerBean，只支持按照一定频度调用任务，如每隔30分钟运行一次。

    配置方式如下：

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean"**>**
3. **<property** name="jobDetail" ref="job2" **/>**
5. **<property** name="startDelay" value="0" **/>**<!-- 调度工厂实例化后，经过0秒开始执行调度 -->
7. **<property** name="repeatInterval" value="2000" **/>**<!-- 每2秒调度一次 -->
9. **</bean>**

              第二种CronTriggerBean，支持到指定时间运行一次，如每天12:00运行一次等。

  配置方式如下：

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean"**>**
3. **<property** name="jobDetail" ref="job2" **/>**
5. <!—每天12:00运行一次 --**>**
7. **<property** name="cronExpression" value="0 0 12 \* \* ?" **/>**
9. **</bean>**

以上两种调度方式根据实际情况，任选一种即可。

第四步：配置调度工厂

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<bean** class="org.springframework.scheduling.quartz.SchedulerFactoryBean"**>**
3. **<property** name="triggers"**>**
5. **<list>**
7. **<ref** bean="cronTrigger" **/>**
9. **</list>**
11. **</property>**
13. **</bean>**

说明：该参数指定的就是之前配置的触发器的名字。

第五步：启动你的应用即可，即将工程部署至tomcat或其他容器。

 Spring-context-support包

到此，spring中Quartz的基本配置就介绍完了，当然了，使用之前，要导入相应的spring的包与Quartz的包，这些就不消多说了。

其实可以看出Quartz的配置看上去还是挺复杂的，没有办法，因为Quartz其实是个重量级的工具，如果我们只是想简单的执行几个简单的定时任务，有没有更简单的工具，有！

请看我第下文Spring task的介绍。

### Spring-Task

上节介绍了在Spring 中使用Quartz，本文介绍Spring3.0以后自主开发的定时任务工具，spring task，可以将它比作一个轻量级的Quartz，而且使用起来很简单，除spring相关的包外不需要额外的包，而且支持注解和配置文件两种

形式，下面将分别介绍这两种方式。

第一种：配置文件方式

第一步：编写作业类

即普通的pojo，如下：

Java代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. import org.springframework.stereotype.Service;
3. @Service
5. public class TaskJob {


9. public void job1() {
11. System.out.println(“任务进行中。。。”);
13. }
15. }

 第二步：在spring配置文件头中添加命名空间及描述

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:task="http://www.springframework.org/schema/task"
5. 。。。。。。
7. xsi:schemaLocation="http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-3.0.xsd"**>**

 第三步：spring配置文件中设置具体的任务

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. **<task:scheduled-tasks>**
3. **<task:scheduled** ref="taskJob" method="job1" cron="0 \* \* \* \* ?"**/>**
5. **</task:scheduled-tasks>**
7. **<context:component-scan** base-package=" com.gy.mytask " **/>**

说明：ref参数指定的即任务类，method指定的即需要运行的方法，cron及cronExpression表达式，具体写法这里不介绍了，详情见上篇文章附录。

这个配置不需要多说了，spring扫描注解用的。

到这里配置就完成了，是不是很简单。

第二种：使用注解形式

也许我们不想每写一个任务类还要在xml文件中配置下，我们可以使用注解@Scheduled，我们看看源文件中该注解的定义：

Java代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. @Target({java.lang.annotation.ElementType.METHOD, java.lang.annotation.ElementType.ANNOTATION\_TYPE})
3. @Retention(RetentionPolicy.RUNTIME)
5. @Documented
7. public @interface Scheduled
9. {
11. public abstract String cron();


15. public abstract long fixedDelay();


19. public abstract long fixedRate();
21. }

 可以看出该注解有三个方法或者叫参数，分别表示的意思是：

cron：指定cron表达式

fixedDelay：官方文档解释：An interval-based trigger where the interval is measured from the completion time of the previous task. The time unit value is measured in milliseconds.即表示从上一个任务完成开始到下一个任务开始的间隔，单位是毫秒。

fixedRate：官方文档解释：An interval-based trigger where the interval is measured from the start time of the previous task. The time unit value is measured in milliseconds.即从上一个任务开始到下一个任务开始的间隔，单位是毫秒。

下面我来配置一下。

第一步：编写pojo

Java代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. import org.springframework.scheduling.annotation.Scheduled;
3. import org.springframework.stereotype.Component;


7. @Component(“taskJob”)
9. public class TaskJob {
11. @Scheduled(cron = "0 0 3 \* \* ?")
13. public void job1() {
15. System.out.println(“任务进行中。。。”);
17. }
19. }

 第二步：添加task相关的配置：

Xml代码

**[html]** [view plaincopy](http://blog.csdn.net/kaixuanfeng2012/article/details/14161285)

1. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
3. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:tx="http://www.springframework.org/schema/tx"
7. xmlns:task="http://www.springframework.org/schema/task"
9. xsi:schemaLocation="
11. http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
13. http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
15. http://www.springframework.org/schema/context
17. http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd
19. http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
21. http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-3.0.xsd"
23. default-lazy-init="false"**>**
25. **<context:annotation-config** **/>**
27. <!—spring扫描注解的配置   --**>**
29. **<context:component-scan** base-package="com.gy.mytask" **/>**


33. <!—开启这个配置，spring才能识别@Scheduled注解   --**>**
35. **<task:annotation-driven** scheduler="qbScheduler" mode="proxy"**/>**
37. **<task:scheduler** id="qbScheduler" pool-size="10"**/>**

说明：理论上只需要加上这句配置就可以了，这些参数都不是必须的。

 Ok配置完毕，当然spring task还有很多参数，我就不一一解释了，具体参考xsd文档http://www.springframework.org/schema/task/spring-task-3.0.xsd。

附录：

cronExpression的配置说明，具体使用以及参数请百度google

字段   允许值   允许的特殊字符

秒    0-59    , - \* /

分    0-59    , - \* /

小时    0-23    , - \* /

日期    1-31    , - \* ? / L W C

月份    1-12 或者 JAN-DEC    , - \* /

星期    1-7 或者 SUN-SAT    , - \* ? / L C #

年（可选）    留空, 1970-2099    , - \* /

- 区间

\* 通配符

? 你不想设置那个字段

下面只例出几个式子

CRON表达式    含义

"0 0 12 \* \* ?"    每天中午十二点触发

"0 15 10 ? \* \*"    每天早上10：15触发

"0 15 10 \* \* ?"    每天早上10：15触发

"0 15 10 \* \* ? \*"    每天早上10：15触发

"0 15 10 \* \* ? 2005"    2005年的每天早上10：15触发

"0 \* 14 \* \* ?"    每天从下午2点开始到2点59分每分钟一次触发

"0 0/5 14 \* \* ?"    每天从下午2点开始到2：55分结束每5分钟一次触发

"0 0/5 14,18 \* \* ?"    每天的下午2点至2：55和6点至6点55分两个时间段内每5分钟一次触发

"0 0-5 14 \* \* ?"    每天14:00至14:05每分钟一次触发

"0 10,44 14 ? 3 WED"    三月的每周三的14：10和14：44触发

"0 15 10 ? \* MON-FRI"    每个周一、周二、周三、周四、周五的10：15触发