[Spring --- java定时器,Spring定时器和Quartz定时器](http://blog.csdn.net/etttttss/article/details/7461371)

标签： [spring](http://www.csdn.net/tag/spring)[quartz](http://www.csdn.net/tag/quartz)[java](http://www.csdn.net/tag/java)[任务](http://www.csdn.net/tag/%e4%bb%bb%e5%8a%a1)[class](http://www.csdn.net/tag/class)[delay](http://www.csdn.net/tag/delay)

2012-04-14 20:15 13950人阅读 [评论](http://blog.csdn.net/etttttss/article/details/7461371#comments)(5) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/etttttss/article/details/7461371#report)

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

定时器

  在现实生活中，会出现这样的例子，比如烧水，一般烧了20分钟后，水开了，会需要及时换水，再烧20分钟，水又开了，继续提醒，比如上班，每天早晨8点钟的闹钟会及时提醒, 那么在java程序中如何实现 这种 已经被设定了的定时的任务呢，在下面会依次介绍关于如何实现定时器来操作一些定时任务的业务需求。

     当前java程序中 能够实现定时的 主要有 三种 方式 ，分别是: java定时器 ， spring定时器 ， quartz定时器.下面依次讲讲他们的应用！

**<1>java定时器的应用。**

 其实java很早就有解决定时器任务的方法了，java提供了了类java.util.TimerTask类基于线程的方式来实现定时任务的操作，然后再提供java.util.Timer类来注册调用，先创建一个类 RingTask 继承 java.util.TimerTask,实现run方法，相关代码如下:

package timer;

import java.util.TimerTask;

/\*\*

 \* 这是一个打铃的程序，必须隔一段时间打一次

 \* @author sam

 \*

 \*/

public class RingTask extends TimerTask{

public RingTask() {

// TODO Auto-generated constructor stub

}

public RingTask(int s,int d) {

// TODO Auto-generated constructor stub

this.second = s;

this.delay  = d;

}

int second = 1;

int delay  = 1;

public void setSecond(int second) {

this.second = second;

}

public void setDelay(int delay) {

this.delay = delay;

}

@Override

public void run() {

// TODO Auto-generated method stub

System.out.println("我是打铃程序!"+"我第一次打铃延迟了"+delay+"秒！");

System.out.println("打铃了!每过"+second+"秒一次");

}

}

  定义好后，下面需要注册调用了，注册调用的方法如下:

      public static void main(String[] args) {

//以 java定时器的模式调用

Timer timer = new Timer();

timer.schedule(

new RingTask(3,3),  //需要注册的定时类

3000,             //最开始先延迟3秒的时间

3000);            //每隔3秒的时间调用一次

}

   一个简单的java定时器就写好了，方便而简介，但是有不好的缺点: 如果需要实现每天早晨7点钟的定时执行一次，且周末的时候早晨7点钟不需要提醒，那这个可就不够用了，并且如果需要服务器一开启就触发这个定时器，则这种注册调用的方法也是不行的。

**<2>Spring定时器的应用。**

        spring定时器是在spring框架中应用较成熟的一种方式，spring将定时任务的调用部分提到了配置文件当中，使定时器的触发条件变得更加灵活，spring定时器的实现，仍然需要 继承 java.util.TimerTask,实现run方法 ，示例类上面已给出，调用的配置如下:

    <!-- 定时器的配置 (spring定时器)-->

<!-- 要调度的bean配置 -->

<bean id="ringTask" class="timer.RingTask">

<!-- 给 属性 second 赋值 为 3 -->

<property name="second" >

<value>3</value>

</property>

<!-- 给 属性 delay 赋值 为 3 -->

<property name="delay" >

<value>3</value>

</property>

</bean>

<!--配置一个触发器 配置触发器的参数-->

<bean id="scheduleRingTask" class="org.springframework.scheduling.timer.ScheduledTimerTask">

<property name="delay" value="3000"></property>           <!--第一次延迟3秒的时间-->

<property name="period" value="3000"></property>          <!--每隔3秒的时间执行一次-->

<property name="timerTask" ref="ringTask"></property>   <!--制定触发的类-->

</bean>

<!-- 总调度,用于启动定时器 -->

<bean id="timerFactory" class="org.springframework.scheduling.timer.TimerFactoryBean">

<property name="scheduledTimerTasks">

<list>

<ref bean="scheduleRingTask"/>

</list>

</property>

</bean>

   在调用方面是不是灵活些了，且能够实现服务器已启动，就将定时器的执行纳入的被监控的范围，符合条件马上触发执行。但是还是存在缺点: 对于指定了具体的年月日时分秒而执行的任务还是不能解决。

**<3>Quartz定时器**

 Quartz是基于Spring框架之上的更加强大的定时器，它不仅可以轻松的实现前面两种定时器的功能，还实现了非常繁复的时间触发执行的任务，Quartz有两种方式来调度定时任务，一是使用Spring提供的 MethodInvokingJobDetailFactoryBean 代理类，Quartz通过该代理类直接调度任务类的某个函数；二是任务类继承QuartzJobBean类或者实现org.quartz.Job接口，Quartz通过该父类或者接口进行调度。

   先来看看实现前面个两种定时器的功能，现在先来举个例子，比如烧水，每1小时烧开一次，进行定时提醒，然后重新换水，再烧，。。。。

    具体的烧水定时类代码如下:

package timer;

import org.quartz.Job;

import org.quartz.JobExecutionContext;

import org.quartz.JobExecutionException;

import org.springframework.scheduling.quartz.QuartzJobBean;

/\*\*

 \* 这是一个热水的程序，必要要经过一段时间烧热了，才提醒重新换水

 \* @author sam

 \* 这个类不管是继承 QuartzJobBean还是实现org.quartz.Job都行

 \*/

public class HotWaterTask extends QuartzJobBean /\*implements Job\*/{

// public void execute(JobExecutionContext arg0)

// throws JobExecutionException {

// // TODO Auto-generated method stub

// System.out.println("我是热水程序，我第一烧水需要1小时");

// System.out.println("水现在烧开了，要及时换水哦!");

// }

@Override

protected void executeInternal(JobExecutionContext arg0)

throws JobExecutionException {

// TODO Auto-generated method stub

System.out.println("我是热水程序， 我第一烧水需要1小时 ");

System.out.println("水现在烧开了，要及时换水哦!");

}

}

类定义好了，下面需要配置进去，配置的代码如下:

         <!-- 配置需要调度的任务类 -->

<bean id="hotWaterTask" class="org.springframework.scheduling.quartz.JobDetailBean">

<property name="jobClass" value="timer.HotWaterTask"></property>

</bean>

<!-- 配置一个触发器 -->

<bean id="hotWaterTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean">

<property name="jobDetail" ref="hotWaterTask"></property>

<property name="startDelay" value="3600000"></property>

<property name="repeatInterval" value=" 3600000"></property>

</bean>

        <!-- 总调度,用于启动定时器 -->

<bean id="schedulerFactory" class="org.springframework.scheduling.quartz.SchedulerFactoryBean">

<property name="triggers" >

<list>

<ref bean="hotWaterTrigger"/>

</list>

</property>

</bean>

  看到了吗，在这里我们并没有直接声明一个 HotWaterTask  Bean，而是声明了一个JobDetailBean。这个是Quartz的特点。JobDetailBean是Quartz的org.quartz.JobDetail的子类，它要求通过jobClass属性来设置一个Job对象。

 好了，上面的任务已经实现了，下面看看 如何实现 具体的年月日时分秒执行的代码

**Quartz在指定的时间执行 (很强大的代理定时执行机制)**

(1) 定义上班闹钟定时类代码如下:

 package timer;

/\*\*

 \* 开始上班，这个程序要求每天(非周末)早晨八点需要启动一次

 \* @author sam

 \*

 \*/

public class StartWorkJob {

public void startWork(){

System.out.println("我是上班程序,每天(非周末)早晨八点需要启动一次");

System.out.println("上班了！~");

}

}

  看到了吗，这个类StartWorkJob 并没有继承任何类也没有实现任何接口，且方法 startWork也是自己定义的,原有的业务代码不需要做任何更改。下面就要提到Quartz实现的一种机制，通过Spring提供的代理类（MethodInvokingJobDetailFactoryBean）来实现定时任务，这个类只需要提供它要代理的类以及要代理的方法，就能够很好的就行定时监控了，强大吧，相关的代码如下:

<!-- 配置需要定时的bean类 -->

<bean id="startWorkJob" class="timer.StartWorkJob"></bean>

    <!-- 配置任务的具体类和方法 -->

<bean id="startWorkTask" class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">

<!-- 要调用的bean -->

<property name="targetObject" ref="startWorkJob"></property>

<!-- 要调用的Method -->

<property name="targetMethod" value="startWork"></property>

<!-- 是否并发,false表示 如果发生错误也不影响下一次的调用 -->

<property name="concurrent" value="false"></property>

</bean>

<!-- 配置一个触发器 -->

<bean id="startWorkTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">

<property name="jobDetail" ref="startWorkTask"></property>

<property name="cronExpression" value="0 \* 13 \* \* ?"></property> <!--每天的下午1点的每分钟的0秒都执行一次-->

</bean>

<!-- 总调度,用于启动定时器 -->

<bean id="schedulerFactory" class="org.springframework.scheduling.quartz.SchedulerFactoryBean">

<property name="triggers" >

<list>

<ref bean="startWorkTrigger"/>

</list>

</property>

</bean>

    好了一个指定了具体时间的定时触发任务也已经实现了，下面来看看cronExpression 有哪些需要知道的配置信息，信息如下:

一个cron表达式有至少6个（也可能7个）有空格分隔的时间元素。从左到右：  
  
1.秒2.分3.小时4.月份中的日期（1-31）5.月份（1-12或JAN-DEC)6.星期中的日期（1-7或SUN-SAT）7.年份（1970-2099）   
每个元素都显示的规定一个值（如6），一个区间（9-12），一个列表（9，11，13）或一个通配符（\*）。因为4和6这两个元素是互斥的，因此应该通过设置一个问号（？）来表明不想设置的那个字段，“/”如果值组合就表示重复次数（10/6表示每10秒重复6次）。

示例如下:

0 0 12 \* \* ?---------------在每天中午12：00触发   
0 15 10 ? \* \*---------------每天上午10:15 触发   
0 15 10 \* \* ?---------------每天上午10:15 触发   
0 15 10 \* \* ? \*---------------每天上午10:15 触发   
0 15 10 \* \* ? 2005---------------在2005年中的每天上午10:15 触发   
0 \* 14 \* \* ?---------------每天在下午2：00至2：59之间每分钟触发一次   
0 0/5 14 \* \* ?---------------每天在下午2：00至2：59之间每5分钟触发一次   
0 0/5 14,18 \* \* ?---------------每天在下午2：00至2：59和6：00至6：59之间的每5分钟触发一次   
0 0-5 14 \* \* ?---------------每天在下午2：00至2：05之间每分钟触发一次   
0 10,44 14 ? 3 WED---------------每三月份的星期三在下午2：00和2：44时触发   
0 15 10 ? \* MON-FRI---------------从星期一至星期五的每天上午10：15触发   
0 15 10 15 \* ?---------------在每个月的每15天的上午10：15触发   
0 15 10 L \* ?---------------在每个月的最后一天的上午10：15触发   
0 15 10 ? \* 6L---------------在每个月的最后一个星期五的上午10：15触发   
0 15 10 ? \* 6L 2002-2005---------------在2002, 2003, 2004 and 2005年的每个月的最后一个星期五的上午10：15触发   
0 15 10 ? \* 6#3---------------在每个月的第三个星期五的上午10：15触发   
0 0 12 1/5 \* ?---------------从每月的第一天起每过5天的中午12：00时触发