[**Spring任务调度实战之Timer**](http://blog.csdn.net/kongxx/article/details/6751151)

标签： [任务调度](http://www.csdn.net/tag/%e4%bb%bb%e5%8a%a1%e8%b0%83%e5%ba%a6)[spring](http://www.csdn.net/tag/spring)[timer](http://www.csdn.net/tag/timer)[class](http://www.csdn.net/tag/class)[string](http://www.csdn.net/tag/string)[delay](http://www.csdn.net/tag/delay)

2011-09-05 21:15 5114人阅读 [评论](http://blog.csdn.net/kongxx/article/details/6751151#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/kongxx/article/details/6751151#report)

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

Java（151） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

本文地址：<http://blog.csdn.net/kongxx/article/details/6751151>

在spring中提供了一些关于任务调度的集成功能，最简单的就是利用JDK自带的Timer和TimerTask类来实现简单任务调度。看下面的小例子：

一个简单的Task类，没有实现任何接口，其中包含一个run方法用来运行这个task

**[java]** [view plain](http://blog.csdn.net/kongxx/article/details/6751151) [copy](http://blog.csdn.net/kongxx/article/details/6751151)

[print](http://blog.csdn.net/kongxx/article/details/6751151)[?](http://blog.csdn.net/kongxx/article/details/6751151)

1. **package** org.garbagecan.springstudy.schedule.timer;
3. **public** **class** MyTask {
4. **private** String name;
6. **public** **void** run() {
7. System.out.println("Run task: " + name + ".");
8. }
10. **public** String getName() {
11. **return** name;
12. }
14. **public** **void** setName(String name) {
15. **this**.name = name;
16. }
17. }

创建一个spring的配置文件，比如spring.xml，内容如下

**[html]** [view plain](http://blog.csdn.net/kongxx/article/details/6751151) [copy](http://blog.csdn.net/kongxx/article/details/6751151)

[print](http://blog.csdn.net/kongxx/article/details/6751151)[?](http://blog.csdn.net/kongxx/article/details/6751151)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd"
5. default-lazy-init="true"**>**
7. **<bean** id="timerFactory" class="org.springframework.scheduling.timer.TimerFactoryBean" lazy-init="false"**>**
8. **<property** name="scheduledTimerTasks"**>**
9. **<list>**
10. **<ref** local="scheduledTask1"**/>**
11. **<ref** local="scheduledTask2"**/>**
12. **</list>**
13. **</property>**
14. **</bean>**
16. **<bean** id="scheduledTask1" class="org.springframework.scheduling.timer.ScheduledTimerTask"**>**
17. **<property** name="delay" value="0" **/>**
18. **<property** name="period" value="10000" **/>**
19. **<property** name="timerTask"**>**
20. **<ref** bean="methodInvokingTask1"**/>**
21. **</property>**
22. **</bean>**
24. **<bean** id="scheduledTask2" class="org.springframework.scheduling.timer.ScheduledTimerTask"**>**
25. **<property** name="delay" value="0" **/>**
26. **<property** name="period" value="10000" **/>**
27. **<property** name="timerTask"**>**
28. **<ref** bean="methodInvokingTask2"**/>**
29. **</property>**
30. **</bean>**
32. **<bean** id="methodInvokingTask1" class="org.springframework.scheduling.timer.MethodInvokingTimerTaskFactoryBean"**>**
33. **<property** name="targetObject" ref="myTask1"**/>**
34. **<property** name="targetMethod" value="run"**/>**
35. **</bean>**
37. **<bean** id="methodInvokingTask2" class="org.springframework.scheduling.timer.MethodInvokingTimerTaskFactoryBean"**>**
38. **<property** name="targetObject" ref="myTask2"**/>**
39. **<property** name="targetMethod" value="run"**/>**
40. **</bean>**
42. **<bean** id="myTask1" class="org.garbagecan.springstudy.schedule.timer.MyTask"**>**
43. **<property** name="name" value="task1"**/>**
44. **</bean>**
46. **<bean** id="myTask2" class="org.garbagecan.springstudy.schedule.timer.MyTask"**>**
47. **<property** name="name" value="task2"**/>**
48. **</bean>**
49. **</beans>**

1. 定义了两个task，task1和task2；  
2. 利用spring提供的MethodInvokingTimerTaskFactoryBean类来实现来实现对对task类和方法的声明，声明目标对象和方法，从而使spring知道要运行那个类的那个方法；  
3. 利用ScheduledTimerTask类来配置每个task的启动时间延时，每次启动之间的间隔，当然还有最重要的是需要运行那个对象，这里使用的上面提到的MethodInvokingTimerTaskFactoryBean类的实例；

4. 最后定义了一个TimerFactoryBean类，并且把ScheduledTimerTask类的实例作为需要调度的task；

最后，写一个测试类来测试上面的代码和配置

**[java]** [view plain](http://blog.csdn.net/kongxx/article/details/6751151) [copy](http://blog.csdn.net/kongxx/article/details/6751151)

[print](http://blog.csdn.net/kongxx/article/details/6751151)[?](http://blog.csdn.net/kongxx/article/details/6751151)

1. **package** org.garbagecan.springstudy.schedule.timer;
3. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
5. **public** **class** Test {
6. **public** **static** **void** main(String[] args) **throws** Exception {
7. **new** ClassPathXmlApplicationContext("/org/garbagecan/springstudy/schedule/timer/spring.xml");
8. }
9. }

运行Test类，可以看到两个task都会启动，并且使用同样的10秒作为每次运行之间的间隔。

# [spring任务调度方法之JDK Timer](http://blog.csdn.net/heardy/article/details/6096378)

标签： [spring](http://www.csdn.net/tag/spring)[任务调度](http://www.csdn.net/tag/%e4%bb%bb%e5%8a%a1%e8%b0%83%e5%ba%a6)[timer](http://www.csdn.net/tag/timer)[jdk](http://www.csdn.net/tag/jdk)[import](http://www.csdn.net/tag/import)[任务](http://www.csdn.net/tag/%e4%bb%bb%e5%8a%a1)

2010-12-24 14:57 1390人阅读 [评论](http://blog.csdn.net/heardy/article/details/6096378#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/heardy/article/details/6096378#report)

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

Spring（7） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

  一、基本概念   
任务(job)：需要被调度以指定的时间间隔运行的任务单元。   
触发器(trigger):引发任务运行的条件。   
调度计划(schedule):一个任务和用来管理这个任务的一组触发器的集合。   
通常一个任务在他的调度计划中只有一个触发器，所以触发器和调度计划这两个概念通常可以互换。   
  
二、spring的两种任务调度方法   
1. JDK Timer JVM1.3版本后提供的任务调度能力，不需要依赖spring以外的产品。   
优点：被包含在java中，不需要外部依赖库，当受限于程序大小或公司政策时可以使用。   
不足：在定义一个调度计划时只能够提供有限的灵活性。   
2. Quartz 提供很大的灵活性，允许定义类似于真实世界中的触发器。   
  
三、JDK Timer   
1）Timer触发器类型   
一次性触发器（One-off）   
重复固定延迟触发器（Fixed-Delay）   
重复定时触发器（Fixed-Rated）   
说明：   
一次性触发器只能执行一次，执行完成后，不能被再度重新使用；   
下面举例说明固定延迟触发器和定时触发器区别：   
假如17:00开始任务执行，任务执行时间为30分钟，每小时执行一次，第一次运行将于17:30结束。如果采用固定延迟触发器，第二次运行将在18:30开始，计算方法为前一次结束时间加上间隔时间；如果采用定时触发器，第二次运行将在18:00开始，计算方法为前一次开始时间加上间隔时间。   
  
2）JDK Timer的使用范例

**[java]** [view plain](http://blog.csdn.net/heardy/article/details/6096378) [copy](http://blog.csdn.net/heardy/article/details/6096378)

1. **package** com.learnworld.timer;
3. **import** java.util.TimerTask;
5. **public** **class** HelloTask **extends** TimerTask {
7. @Override
8. **public** **void** run() {
9. System.out.println("Hello!");
10. }
12. }

**[java]** [view plain](http://blog.csdn.net/heardy/article/details/6096378) [copy](http://blog.csdn.net/heardy/article/details/6096378)

1. **package** com.learnworld.timer;
3. **import** java.util.Calendar;
4. **import** java.util.Timer;
6. **public** **class** SchedulingTest {
7. **public** **static** **void** main(String args[]){
8. Timer t = **new** Timer();
10. //一次性触发器，1000毫秒后开始运行
11. t.schedule(**new** HelloTask(), 1000);
13. //固定时间调用
14. Calendar cal = Calendar.getInstance();
15. cal.set(2010,Calendar.OCTOBER, 27);
16. t.schedule(**new** HelloTask(), cal.getTime());
18. //调用重复任务,固定延迟触发器
19. t.schedule(**new** HelloTask(),1000,3000);
21. //定时触发器
22. t.scheduleAtFixedRate(**new** HelloTask(), 1000, 1000);
24. }
25. }

3) Spring对JDK Timer调度的支持   
目的：许多任务需要配置数据，采用spring管理，使得程序易于配置。   
触发器信息是硬编码到程序里的，意味着对任务触发时间的修改都要修改程序并重新编译。   
调度新任务或移除任务都需要修改程序代码，这可以通过spring来配置。   
  
核心类：   
ScheduleTimerTask: 扮演TimerTask的包装器实现，允许你为这个任务定义触发器信息。   
TimerFactoryBean:让spring使用配置创建触发器，为一组指定的scheduledTimerTask bean自动创建Timer实例。   
  
配置文件:

**[xhtml]** [view plain](http://blog.csdn.net/heardy/article/details/6096378) [copy](http://blog.csdn.net/heardy/article/details/6096378)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd"**>**
3. **<beans>**
4. **<bean** id="job" class="com.learnworld.timer.HelloTask" **/>**
5. **<bean** id="timerTask" class="org.springframework.scheduling.timer.ScheduledTimerTask"**>**
6. **<property** name="delay"**>**
7. **<value>**1000**</value>**
8. **</property>**
9. **<property** name="period"**>**
10. **<value>**3000**</value>**
11. **</property>**
12. **<property** name="timerTask"**>**
13. **<ref** local="job" **/>**
14. **</property>**
15. **</bean>**
17. **<bean** id="timerFactory" class="org.springframework.scheduling.timer.TimerFactoryBean"**>**
18. **<property** name="scheduledTimerTasks"**>**
19. **<list>**
20. **<ref** local="timerTask" **/>**
21. **</list>**
22. **</property>**
23. **</bean>**
24. **</beans>**

**[java]** [view plain](http://blog.csdn.net/heardy/article/details/6096378) [copy](http://blog.csdn.net/heardy/article/details/6096378)

1. **package** com.learnworld.timer;
3. **import** java.io.IOException;
5. **import** org.springframework.context.ApplicationContext;
6. **import** org.springframework.context.support.FileSystemXmlApplicationContext;
8. **public** **class** TimerFactoryBeanTest {
9. **public** **static** **void** main(String[] args) **throws** IOException{
10. ApplicationContext ac = **new** FileSystemXmlApplicationContext("src/conf/timer.xml");
11. System.in.read();
12. }
13. }

4)当进行任务调度时，通常需要执行一段现有逻辑，我们不需要创建一个包装你逻辑的TimerTask类，使用MethodInvokingTimerTaskFactoryBean,可以调度bean里任意方法或者指定类的静态方法。   
  
配置文件methodInvoker.xml:

**[xhtml]** [view plain](http://blog.csdn.net/heardy/article/details/6096378) [copy](http://blog.csdn.net/heardy/article/details/6096378)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd"**>**
3. **<beans>**
4. **<bean** id="target" class="com.learnworld.timer.FooBean" **/>**
5. **<bean** id="job"
6. class="org.springframework.scheduling.timer.MethodInvokingTimerTaskFactoryBean"**>**
7. **<property** name="targetObject"**>**
8. **<ref** local="target" **/>**
9. **</property>**
10. **<property** name="targetMethod"**>**
11. **<value>**someJob**</value>**
12. **</property>**
13. **<property** name="arguments"**>**
14. **<value>**Hello!**</value>**
15. **</property>**
16. **</bean>**
17. **<bean** id="timerTask" class="org.springframework.scheduling.timer.ScheduledTimerTask"**>**
18. **<property** name="delay"**>**
19. **<value>**1000**</value>**
20. **</property>**
21. **<property** name="period"**>**
22. **<value>**3000**</value>**
23. **</property>**
24. **<property** name="timerTask"**>**
25. **<ref** local="job" **/>**
26. **</property>**
27. **</bean>**
29. **<bean** id="timerFactory" class="org.springframework.scheduling.timer.TimerFactoryBean"**>**
30. **<property** name="scheduledTimerTasks"**>**
31. **<list>**
32. **<ref** local="timerTask" **/>**
33. **</list>**
34. **</property>**
35. **</bean>**
36. **</beans>**

测试程序TimerMethodInvokerTest.java:

**[java]** [view plain](http://blog.csdn.net/heardy/article/details/6096378) [copy](http://blog.csdn.net/heardy/article/details/6096378)

1. **package** com.learnworld.timer;
3. **import** java.io.IOException;
5. **import** org.springframework.context.ApplicationContext;
6. **import** org.springframework.context.support.FileSystemXmlApplicationContext;
8. **public** **class** TimerMethodInvokerTest {
9. **public** **static** **void** main(String[] args) **throws** IOException{
10. ApplicationContext ac = **new** FileSystemXmlApplicationContext("src/conf/methodInvoker.xml");
11. System.in.read();
12. }
13. }

转自:http://www.javaeye.com/topic/794865