

发表人

内容

2013-11-07 05:13:20

文章主题: Spring定时任务的几种实现 spri.....

lxhwantdl



私塾在线
www.sishuok.com

交流经验: 0

总积分: 90

级别: 普通会员

注册时间: 2012-03-09

文章: 3

离线

Spring定时任务的几种实现 spring框架 quartz spring spring-task 定时任务 注解

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）：

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。

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。

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：

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。

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，因为这样所以的类都是普通类，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：

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

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

查看 复制到剪贴板 打印

```
01. import org.quartz.JobExecutionContext;
02. import org.quartz.JobExecutionException;
03. import org.springframework.scheduling.quartz.QuartzJobBean;
04. public class Job1 extends QuartzJobBean {
05.
06.     private int timeout;
07.     private static int i = 0;
08.     //调度工厂实例化后，经过timeout时间开始执行调度
09.     public void setTimeout(int timeout) {
10.         this.timeout = timeout;
11.     }
12.
13.     /**
14.      * 要调度的具体任务
15.      */
16.     @Override
17.     protected void executeInternal(JobExecutionContext context)
18.         throws JobExecutionException {
19.         System.out.println("定时任务执行中...");
20.     }
21. }
```

第二步：**spring**配置文件中配置作业类**JobDetailBean**

sishuok.com/forum/posts/list/7260.html

1/6

查看 复制到剪贴板 打印

```

01. <bean name="job1" class="org.springframework.scheduling.quartz.JobDetailBean">
02. <property name="jobClass" value="com.gy.Job1" />
03. <property name="jobDataAsMap">
04. <map>
05. <entry key="timeout" value="0" />
06. </map>
07. </property>
08. </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:

查看 复制到剪贴板 打印

```

01. <bean id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean">
02. <property name="jobDetail" ref="job1" />
03. <property name="startDelay" value="0" /><!-- 调度工厂实例化后, 经过0秒开始执行调度 -->
04. <property name="repeatInterval" value="2000" /><!-- 每2秒调度一次 -->
05. </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:

查看 复制到剪贴板 打印

```

01. <bean id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">
02. <property name="jobDetail" ref="job1" />
03. <!--每天12:00运行一次 -->
04. <property name="cronExpression" value="0 0 12 * * ?" />
05. </bean>

```

关于cronExpression表达式的语法参见附录。

第四步: 配置调度工厂

查看 复制到剪贴板 打印

```

01. <bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
02. <property name="triggers">
03. <list>
04. <ref bean="cronTrigger" />
05. </list>
06. </property>
07. </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

查看 复制到剪贴板 打印

```

01. public class Job2 {
02. public void doJob2() {
03. System.out.println("不继承QuartzJobBean方式-调度进行中...");
04. }
05. }

```

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

查看 复制到剪贴板 打印

```

01. <bean id="job2"
02. class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"
03. <property name="targetObject">

```

```
04. <bean class="com.gy.Job2" />
05. </property>
06. <property name="targetMethod" value="doJob2" />
07. <property name="concurrent" value="false" /><!-- 作业不并发调度 -->
08. </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：

```
查看 复制到剪贴板 打印
01. <bean id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean">
02. <property name="jobDetail" ref="job2" />
03. <property name="startDelay" value="0" /><!-- 调度工厂实例化后，经过0秒开始执行调度 -->
04. <property name="repeatInterval" value="2000" /><!-- 每2秒调度一次 -->
05. </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：

```
查看 复制到剪贴板 打印
01. <bean id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">
02. <property name="jobDetail" ref="job2" />
03. <!--每天12:00运行一次 -->
04. <property name="cronExpression" value="12 * * ?" />
05. </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

```
查看 复制到剪贴板 打印
01. <bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
02. <property name="triggers">
03. <list>
04. <ref bean="cronTrigger" />
05. </list>
06. </property>
07. </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。

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：

```
查看 复制到剪贴板 打印
01. import org.springframework.stereotype.Service;
02. @Service
03. public class TaskJob {
04.
05.     public void job1() {
06.         System.out.println("任务进行中。。。");
07.     }
08. }
```

第二步：在spring配置文件头中添加命名空间及描述

```
查看 复制到剪贴板 打印
```

```

01. <beans xmlns="http://www.springframework.org/schema/beans"
02.       xmlns:task="http://www.springframework.org/schema/task"
03.       . . . . .
04.       xsi:schemaLocation="http://www.springframework.org/schema/task http://www.springframework.org/schem
a/task/spring-task-3.0.xsd">

```

第三步：**spring**配置文件中设置具体的任务

```

查看 复制到剪贴板 打印
01. <task:scheduled-tasks>
02.     <task:scheduled ref="taskJob" method="job1" cron="0 * * * * ?"/>
03. </task:scheduled-tasks>
04.
05. <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。

<context:component-scan base-package="com.gy.mytask" />这个配置不消多说了，**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：

```

查看 复制到剪贴板 打印
01. @Target({java.lang.annotation.ElementType.METHOD, java.lang.annotation.ElementType.ANNOTATION_TYPE})
02. @Retention(RetentionPolicy.RUNTIME)
03. @Documented
04. public @interface Scheduled
05. {
06.     public abstract String cron();
07.
08.     public abstract long fixedDelay();
09.
10.     public abstract long fixedRate();
11. }

```

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**

```

查看 复制到剪贴板 打印
01. import org.springframework.scheduling.annotation.Scheduled;
02. import org.springframework.stereotype.Component;
03.
04. @Component("taskJob")
05. public class TaskJob {
06.     @Scheduled(cron = "0 0 3 * * ?")
07.     public void job1() {
08.         System.out.println("任务进行中。。。");
09.     }
10. }

```

第二步：添加**task**相关的配置：

```

查看 复制到剪贴板 打印
01. <?xml version="1.0" encoding="UTF-8"?>
02. <beans xmlns="http://www.springframework.org/schema/beans"
03.       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
04.       xmlns:context="http://www.springframework.org/schema/context"
05.       xmlns:tx="http://www.springframework.org/schema/tx"
06.       xmlns:task="http://www.springframework.org/schema/task"
07.       xsi:schemaLocation="
08.         http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd
09.         http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-
3.0.xsd
10.         http://www.springframework.org/schema/context
11.         http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd
12.         http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0
.xsd
13.         http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-ta
sk-3.0.xsd"
14.       default-lazy-init="false">

```

```
13.
16.
17.     <context:annotation-config />
18.     <!--spring扫描注解的配置 -->
19.     <context:component-scan base-package="com.gy.mytask" />
20.
21.     <!--开启这个配置，spring才能识别@Scheduled注解 -->
22.     <task:annotation-driven scheduler="qbScheduler" mode="proxy"/>
23.     <task:scheduler id="qbScheduler" pool-size="10"/>
```

说明：理论上只需要加上<task:annotation-driven />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/sc-hema/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

推广链接

- 研磨设计模式——跟着cc学设计系列
- 云计算 之 Hadoop实战-中高级部分
- 云计算 之 Hadoop实战-初级部分
- 云计算开发技术——企业级高端培训课程（业余班）
- 独家Android4就业经典视频课程，2012年首发！！
- 2012 最给力的Java就业解决方案——Java私塾首推：远程学习+地面冲刺=高薪就业

精品视频课程推荐

Javascript基础视频教程

JavaScript的内置对象--Array、String、Date、Math等，可以通过DOM对象进行对象控制，创建控制菜单及复选框的控制，创建二级联动列表框及列表框选项的移动，JavaScript项目，创建基于JS的商品管理系统。

深入浅出学Spring Web MVC视频教程

系统、完整的学习Spring Web MVC开发的知识。包括：Spring Web MVC入门；理解DispatcherServlet；注解式控制器开发详解；数据类型转换；数据格式化；数据验证； 拦截器；对Ajax的支持；文件上传下载；表单标签等内容；最后以一个综合的CRUD带翻页的应用示例来综合所学的知识

Ajax+JSON基础实战视频教程

数据校验、Javascript模拟多线程、下拉列表联动、操作XML、AJAX结合JSON的操作、Json-lib的使用

高级软件架构师实战培训阶段一

内容概述：本课程专注于构建：高可扩展性、高性能、大数据量、高并发、分布式的系统架构。 从零开始、全面系统、成体系的软件架构课程，循序渐进的讲述构建上述系统架构所需要的各种技术知识和技能。

技术要点： 1：构建基本的业务功能块，基于Maven+Git+Spring mvc+spring+mybatis+ehcache+mysql+X-gen代码生成

2：高扩展性的分布式体系架构（基于Nginx+Varnish+Memcache+ActiveMQ）

3：NoSQL的合理使用和架构优化（基于MongoDB）

4：分布式文件存储和架构优化（基于MogileFS）

透彻理解JavaBean视频教程

深入浅出的讲解JavaBen的写法、JavaBean的用法、JavaBean的实现机制、JavaBean对应翻译的代码理解。



交流首页 » **Java**

前往:

选择一个版面

Go

[关于我们](#) | [联系我们](#) | [用户协议](#) | [私塾在线服务协议](#) | [版权声明](#) | [隐私保护](#)

版权所有 Copyright (C) 2009-2012 私塾在线学习网