typora-root-url: images

typora-copy-images-to: images

# **Spring Task 定时任务**

## **主要内容**

## **定时任务概述**

在项目中开发定时任务应该一种比较常见的需求，在 Java 中开发定时任务主要有三种解决方案：一是使用JDK 自带的 Timer，二是使用第三方组件 Quartz，三是使用 Spring Task。

Timer 是 JDK 自带的定时任务工具,其简单易用，但是对于复杂的定时规则无法满足，在实际项目开发中也很少使用到。Quartz 功能强大，但是使用起来相对笨重。而 Spring Task 则具备前两者的优点（功能强大且简单易用），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。

## **使用Spring Task实现定时任务**

Spring Task 开发定时任务有两种任务配置方式：

1. XML 配置

2. 注解配置

### **使用 XML 配置实现定时任务**

#### **创建项目，添加依赖**

创建Maven项目，在pom.xml配置文件中，修改项目环境，添加spring依赖坐标

<!-- 引入spring依赖坐标 -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-context</artifactId>  
    <version>5.2.4.RELEASE</version>  
</dependency>

#### **添加配置文件 spring.xml**

在src/main/resources目录下新建配置文件spring.xml，并设置Spring扫描器

<beans xmlns="http://www.springframework.org/schema/beans"  
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
       xmlns:context="http://www.springframework.org/schema/context"  
       xsi:schemaLocation="http://www.springframework.org/schema/beans  
       https://www.springframework.org/schema/beans/spring-beans.xsd  
       http://www.springframework.org/schema/context  
       http://www.springframework.org/schema/context/spring-context.xsd">  
​  
    <!-- Spring扫描注解的配置 -->  
    <context:component-scan base-package="com.xxxx" />  
​  
</beans>

#### **定义定时任务方法**

新建类，添加自动注入的注解，定义定义任务的方法

package com.xxxx.task;  
​  
import org.springframework.stereotype.Component;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
​  
@Component  
public class TaskJob {  
​  
    public void job1(){  
        System.out.println("任务 1:" +  
                new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date()));  
   }  
​  
    public void job2(){  
        System.out.println("任务 2:" +  
                new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date()));  
   }  
}

#### **定时任务命名空间的添加**

在spring.xml配置文件中，添加定时任务的命名空间

xmlns:task="http://www.springframework.org/schema/task"  
​  
http://www.springframework.org/schema/task  
http://www.springframework.org/schema/task/spring-task.xsd

修改配置文件内容如下：

<beans xmlns="http://www.springframework.org/schema/beans"  
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
       xmlns:context="http://www.springframework.org/schema/context"  
       xmlns:task="http://www.springframework.org/schema/task"  
       xsi:schemaLocation="http://www.springframework.org/schema/beans  
       https://www.springframework.org/schema/beans/spring-beans.xsd  
       http://www.springframework.org/schema/context  
       http://www.springframework.org/schema/context/spring-context.xsd  
       http://www.springframework.org/schema/task  
       http://www.springframework.org/schema/task/spring-task.xsd">

#### **定时任务配置**

<!--   
 配置定时规则  
 ref：指定的类，即任务类  
 method：指定的即需要运行的方法  
 cron：cronExpression表达式  
 -->  
<task:scheduled-tasks>  
 <!-- 每个两秒执行一次任务 -->  
 <task:scheduled ref="taskJob" method="job1" cron="0/2 \* \* \* \* ?"/>  
 <!-- 每隔五秒执行一次任务 -->  
 <task:scheduled ref="taskJob" method="job2" cron="0/5 \* \* \* \* ?"/>  
 <!-- 多个定时任务 在这里配置 -->  
</task:scheduled-tasks>

#### **测试定时任务**

public static void main( String[] args ) {  
          
        System.out.println("定义任务测试...");  
​  
        // 获取Spring上下文环境  
        ApplicationContext ac = new ClassPathXmlApplicationContext("spring.xml");  
        // 获取指定Bean对象  
        TaskJob taskJob = (TaskJob) ac.getBean("taskJob");  
}

### **使用注解配置实现定时任务**

#### **定义定时任务方法**

@Scheduled 注解的使用

package com.xxxx.task;  
​  
import org.springframework.scheduling.annotation.Scheduled;  
import org.springframework.stereotype.Component;  
​  
import java.text.SimpleDateFormat;  
import java.util.Date;  
​  
@Component  
public class TaskJob02 {  
​  
    @Scheduled(cron="0/2 \* \* \* \* ?")  
    public void job1(){  
        System.out.println("任务 1:" +  
                new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date()));  
   }  
​  
    @Scheduled(cron="0/5 \* \* \* \* ?")  
    public void job2(){  
        System.out.println("任务 2:" +  
                new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date()));  
   }  
}  
​

#### **定时任务命名空间的添加**

在spring.xml配置文件中，添加定时任务的命名空间

xmlns:task="http://www.springframework.org/schema/task"  
  
http://www.springframework.org/schema/task  
http://www.springframework.org/schema/task/spring-task.xsd

#### **配置定时任务驱动**

<!-- 配置定时任务驱动。开启这个配置，spring才能识别@Scheduled注解 -->  
<task:annotation-driven />

配置文件内容如下：

<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:task="http://www.springframework.org/schema/task"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 https://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/task  
 http://www.springframework.org/schema/task/spring-task.xsd">  
  
 <!-- Spring扫描注解的配置 -->  
 <context:component-scan base-package="com.xxxx" />  
  
 <!-- 配置定时任务驱动。开启这个配置，spring才能识别@Scheduled注解 -->  
 <task:annotation-driven />  
  
</beans>

#### **测试定时任务**

public static void main( String[] args ) {  
   
 System.out.println("定义任务测试...");  
  
 // 获取Spring上下文环境  
 ApplicationContext ac = new ClassPathXmlApplicationContext("spring.xml");  
 // 获取指定Bean对象  
 TaskJob02 taskJob02 = (TaskJob02) ac.getBean("taskJob02");  
}

## **Cron 表达式简介**

关于 cronExpression 表达式有至少 6 个（也可能是 7 个）由空格分隔的时间元素。从左至右，这些元素的定义如下：

1．秒（0–59）

2．分钟（0–59）

3．小时（0–23）

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）

0 0 10,14,16 \* \* ?   
   
每天上午 10 点,下午 2 点和下午 4 点

0 0,15,30,45 \* 1-10 \* ?   
   
每月前 10 天每隔 15 分钟

30 0 0 1 1 ? 2012   
   
在 2012 年 1 月 1 日午夜过 30 秒时

各个时间可用值如下：

秒 0-59 , - \* /

分 0-59 , - \* /

小时 0-23 , - \* /

日 1-31 , - \* ? / L W C

月 1-12 or JAN-DEC , - \* /

周几 1-7 or SUN-SAT , - \* ? / L C #

年(可选字段) empty, 1970-2099 , - \* /

可用值详细分析如下：

"\*" —— 字符可以用于所有字段,在"分"字段中设为"\*"，表示"每一分钟"的含义。  
"?" —— 字符可以用在"日"和"周几"字段,它用来指定"不明确的值"。  
 这在你需要指定这两个字段中的某一个值而不是另外一个的时候会被用到。在后面的例子中可以看到其含义。  
"-" —— 字符被用来指定一个值的范。  
 比如在"小时"字段中设为"10-12"，表示"10 点到 12 点"。   
"," —— 字符指定数个值。  
 比如在"周几"字段中设为"MON,WED,FRI",表示"the days Monday, Wednesday, and Friday"。   
"/" —— 字符用来指定一个值的的增加幅度。  
 比如在"秒"字段中设置为"0/15"表示"第 0, 15, 30,和 45 秒"。  
 而"5/15"则表示"第 5, 20, 35,和 50"。  
 在'/'前加"\*"字符相当于指定从 0 秒开始。每个字段都有一系列可以开始或结束的数值。  
 对于"秒"和"分"字段来说，其数值范围为 0 到 59。  
 对于"小时"字段来说其为 0 到 23,对于“日”字段来说为 0 到 31。  
 而对于"月"字段来说为 1 到 12。  
 "/"字段仅仅只是帮助你在允许的数值范围内从开始"第 n"的值。  
"L" —— 字符可用在"日"和"周几"这两个字段。它是"last"的缩写,但是在这两个字段中有不同的含义。  
 "日"字段中的"L"表示"一个月中的最后一天",对于一月就是 31 号,对于二月来说就是 28 号（非闰年）。  
 "周几"字段中,它简单的表示"7" or "SAT"。  
 但是如果在"周几"字段中使用时跟在某个数字之后,它表示"该月最后一个星期×"。  
 比如"6L"表示"该月最后一个周五"。  
 当使用"L"选项时,指定确定的列表或者范围非常重要，否则你会被结果搞糊涂的。  
"W" —— 可用于"日"字段。用来指定历给定日期最近的工作日(周一到周五)。  
 比如将"日"字段设为"15W"，意为: "离该月 15 号最近的工作日"。  
 因此如果 15 号为周六，触发器会在 14 号即周五调用。  
 如果 15 号为周日,触发器会在 16 号也就是周一触发。如果 15 号为周二,那么当天就会触发。  
 如果"日"字段设为"1W",而一号是周六,会于下周一即当月的 3 号触发,它不会越过当月的值的范围边界。  
 "W"字符只能用于"日"字段的值为单独的一天而不是一系列值的时候。  
 "L"和"W"可以组合用于“日”字段表示为'LW'，意为"该月最后一个工作日"。   
"#" —— 字符可用于"周几"字段。该字符表示"该月第几个周×"。  
 比如"6#3"表示该月第三个周五( 6 表示周五，而"#3"该月第三个)。  
 再比如: "2#1" 表示该月第一个周一，而"4#5" 该月第五个周三。  
 注意如果你指定"#5"该月没有第五个"周×"，该月是不会触发的。  
"C" —— 字符可用于"日"和"周几"字段，它是"calendar"的缩写。  
 它表示为基于相关的日历所计算出的值（如果有）。如果没有关联的日历,那它等同于包含全部日历。  
 "日"字段值为"5C"，表示"日历中的第一天或者 5 号以后"。  
 "周几"字段值为"1C"，则表示"日历中的第一天或者周日以后"。  
 对于"月份"字段和"周几"字段来说合法的字符都不是大小写敏感的。

一些例子：

"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   
  
"0 15 10 15 \* ?" 每月 15 号的 10：15 触发   
  
"0 15 10 L \* ?" 每月的最后一天的 10：15 触发   
  
"0 15 10 ? \* 6L" 每月最后一个周五的 10：15