<http://blog.csdn.net/u013613428/article/details/71190828>

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/u013613428/article/details/71190828)

## 1. Overview

这篇文章中我们将讨论Spring的@Scheduled annotation，并且，将展示如何用它来配置一个定时任务。

要使用@Scheduled annotation有两条基本的原则：

* a method should have void return type
* a method should not accept any parameters

## 2. 在spring中开启Scheduling

要想打开@Scheduled annotation的scheduling功能，我们可以通过annotation的方式：

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Configuration  @EnableScheduling  public class SpringConfig {      ...  } |

也可用通过XML配置:

|  |  |
| --- | --- |
| 1 | <task:annotation-driven> |

## 3. 启动一个固定时延（fixed delay）的任务

以下代码配置了一个拥有固定时延的任务:

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Scheduled(fixedDelay = 1000)  public void scheduleFixedDelayTask() {      System.out.println(        "Fixed delay task - " + System.currentTimeMillis() / 1000);  } |

在这种模式下，上一个任务的结束时间和下一个任务的开始时间之间的时间间隔是固定的。下一个任务总是等上一个任务结束之后，并等待固定时延之后才开始。

这个选项可以用于任务必须等待上一个任务结束的场景。

## 4. 启动一个固定频率（fixed rate）的定时任务

以下的定时任务，就不会要求任务的开始和结束之间有固定的时间差:

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Scheduled(fixedRate = 1000)  public void scheduleFixedRateTask() {      System.out.println(        "Fixed rate task - " + System.currentTimeMillis() / 1000);  } |

在这种模式下，下一个任务不会等到上个任务结束之后才开始，无论如何，在1000ms之后，一个新的任务就会开始执行。

这种模式适合那种任务之间是独立的，没有上下文关系的场景。

## 5. 配置有初始时延的定时任务

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @Scheduled(fixedDelay = 1000, initialDelay = 1000)  public void scheduleFixedRateWithInitialDelayTask() {        long now = System.currentTimeMillis() / 1000;      System.out.println(        "Fixed rate task with one second initial delay - " + now);  } |

顾名思义，配置了initialDelay，任务的第一次运行会是在进程启动的1000ms之后。然后，就会按1000ms的间隔启动独立的任务。

这个选项，通常用于定时任务需要一个set-up时间的场景。

## 6. 使用cron 表达式配置定时任务

很多时候，仅仅有时间间隔和频率是不够的，我们需要一个比较灵活的，复杂的配置去控制定时任务，这时，我们就可以使用cron 表达式：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @Scheduled(cron = "0 15 10 15 \* ?")  public void scheduleTaskUsingCronExpression() {        long now = System.currentTimeMillis() / 1000;      System.out.println(        "schedule tasks using cron jobs - " + now);  } |

在这个例子中，我们启动了一个任务，在每月的15号的早上10点15分启动任务。如果你搞不清楚该怎么写cron表达式，请用工具：

http://www.cronmaker.com/

## 7. 动态配置定时任务的参数

把定时任务的参数写死很多时候是很不灵活的，有时我们需要动态的改变定时任务的配置而不需要重新编译和部署。

以下方法可以从spring的properties文件中读入参数值，动态的配置定时任务:

A fixedDelay task:

|  |  |
| --- | --- |
| 1 | @Scheduled(fixedDelayString = "${fixedDelay.in.milliseconds}") |

A fixedRate task:

|  |  |
| --- | --- |
| 1 | @Scheduled(fixedRateString = "${fixedRate.in.milliseconds}") |

A cron expression based task:

|  |  |
| --- | --- |
| 1 | @Scheduled(cron = "${cron.expression}") |

## 8. 使用XML配置定时任务的参数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <!-- Configure the scheduler -->  <task:scheduler id="myScheduler" pool-size="10" />    <!-- Configure parameters -->  <task:scheduled-tasks scheduler="myScheduler">      <task:scheduled ref="beanA" method="methodA"        fixed-delay="5000" initial-delay="1000" />      <task:scheduled ref="beanB" method="methodB"        fixed-rate="5000" />      <task:scheduled ref="beanC" method="methodC"        cron="\*/5 \* \* \* \* MON-FRI" />  </task:scheduled-tasks> |