

#### 咕泡学院（每天20：30腾讯课堂公开课）

今日课程：基于Spring+Quartz实现定时任务的动态调度

咕泡学院-Tom老师 QQ : 441221062

咕泡学院-VIP咨询 QQ : 2082233439

咕泡学院-乐乐老师 QQ ：2279002544 (视频下载联系)

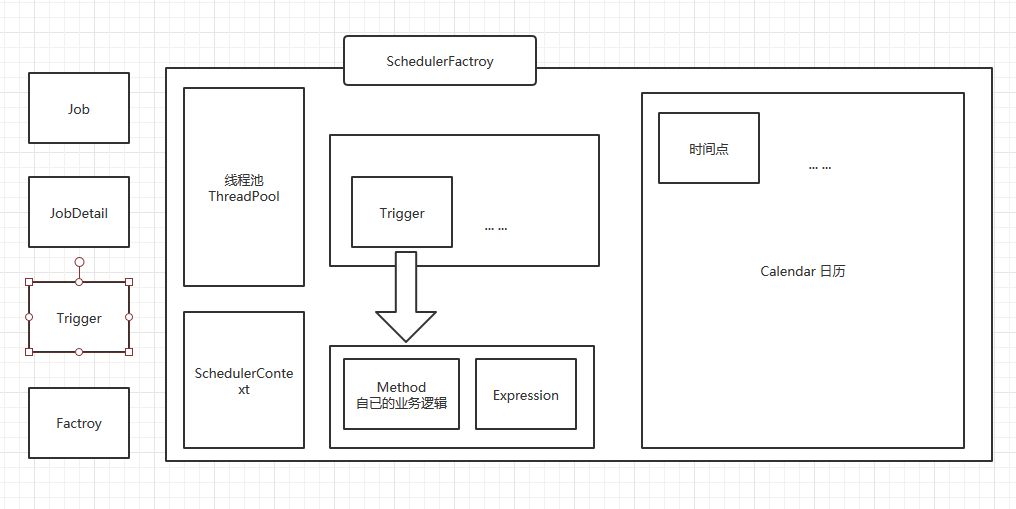
咕泡学院-技术交流 QQ ：608583947

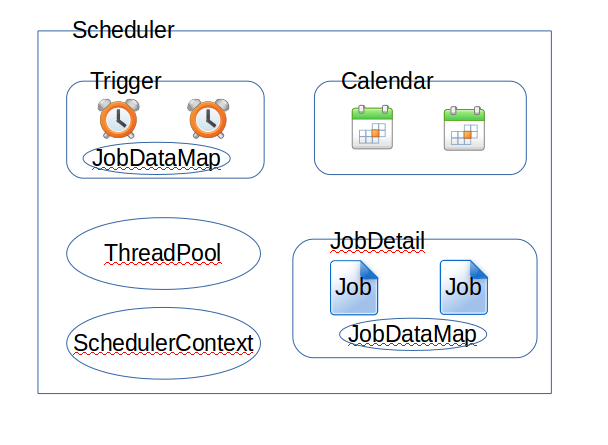
打开腾讯课堂 搜索 咕泡学院，立即报名即可天天听老师讲课啦。

格言：不只做一个技术者，更要做一个思考者。

课程安排

# Quartz内部原理介绍





# 二、Quartz常用API

1、常用API

1.Job

Job是一个接口，只有一个方法

void execute(JobExecutionContext ctx) throws JobExecutionException;

开发者实现该接口定义需要执行的任务。这个方法的参数JobExecutionContext var1提供了调度上下文中的信息。

下面的demo片段如下：

2.JobDetail

Quartz在每次执行job时，都重新创建一个job实例，所以，quartz不是接受一个job实例，而是一个Job实现类，在运行时，通过newInstance()的反射调用机制实例化Job。因此，需要通过一个对象来描述Job的实现类及一些静态信息，如Job名称，组名，关联监听器等信息。

3.Trigger

Trigger描述触发job执行的时间规则。有SimpleTrigger和CronTrigger这两个子类。SimpleTrigger主要用于触发一次或者固定频率触发；CronTrigger则可以通过Cron表达式定义各种复杂的触发时间，如每周一上午10点执行、每月的最后一个星期日执行。这个跟Linux中的cron定时任务是一样的，具体的定义规则，自己google吧。

4.Calendar

有时每月的最后一个星期日执行任务，但是如果遇到中秋节、春节就不执行，这种需求就用到了Calendar。也就是说，从tigger中排除若干个特殊日期，使用的就是Calendar。一个trigger可以和多Calendar配合使用。

5.Scheduler

Scheduler代表一个Quartz独立运行的容器,Trigger和JobDetail可以注册到Scheduler中,两者在Scheduler中拥有各自的组和名称, 组和名称是在Scheduler中定位一个对象的依据。

Spring中的

SchedulerFactory

2、CronExpression语法介绍

CronTrigger配置完整格式为： [秒] [分] [小时] [日] [月] [周] [年]

序号 说明 是否必填 允许范围 允许的通配符

1 秒 是 0-59 , - \* /

2 分 是 0-59 , - \* /

3 时 是 0-23 , - \* /

4 日 是 1-31 , - \* ? / L W

5 月 是 1-12 or JAN-DEC , - \* /

6 周 是 1-7 or SUN-SAT , - \* ? / L #

7 年 否 empty 或 1970-2099 , - \* /

3、通配符说明：

1） \* 表示所有值. 例如:在分的字段上设置 "\*",表示每一分钟都会触发。

2） ? 表示不指定值。使用的场景为不需要关心当前设置这个字段的值。例如:要在每月的10号触发一个操作，但不关心是周几，所以需要周位置的那个字段设置为"?" 具体设置为 0 0 0 10 \* ?- 表示区间。例如 在小时上设置 "10-12",表示 10,11,12点都会触发。

3） , 表示指定多个值，例如在周字段上设置 "MON,WED,FRI" 表示周一，周三和周五触发

4） /用于递增触发。如在秒上面设置"5/15" 表示从5秒开始，每增15秒触发(5,20,35,50)。 在月字段上设置'1/3'所示每月1号开始，每隔三天触发一次。

5） L 表示最后的意思。在日字段设置上，表示当月的最后一天(依据当前月份，如果是二月还会依据是否是润年[leap]), 在周字段上表示星期六，相当于"7"或"SAT"。如果在"L"前加上数字，则表示该数据的最后一个。例如在周字段上设置"6L"这样的格式,则表示“本月最后一个星期五"

6） W 表示离指定日期的最近那个工作日(周一至周五). 例如在日字段上设置"15W"，表示离每月15号最近的那个工作日触发。如果15号正好是周六，则找最近的周五(14号)触发, 如果15号是周未，则找最近的下周一(16号)触发.如果15号正好在工作日(周一至周五)，则就在该天触发。如果指定格式为 "1W",它则表示每月1号往后最近的工作日触发。如果1号正是周六，则将在3号下周一触发。(注，"W"前只能设置具体的数字,不允许区间"-").

7） # 序号(表示每月的第几个周几)，例如在周字段上设置"6#3"表示在每月的第三个周六.注意如果指定"#5",正好第五周没有周六，则不会触发该配置(用在母亲节和父亲节再合适不过了) ；

8） 小提示：'L'和 'W'可以一组合使用。如果在日字段上设置"LW",则表示在本月的最后一个工作日触发；周字段的设置，若使用英文字母是不区分大小写的，即MON与mon相同；

4、举例

0 0 12 \* \* ? 每天12点触发

0 15 10 ? \* \* 每天10点15分触发

0 15 10 \* \* ? 每天10点15分触发

0 15 10 \* \* ? \* 每天10点15分触发

0 15 10 \* \* ? 2005 2005年每天10点15分触发

0 \* 14 \* \* ? 每天下午的 2点到2点59分每分触发

0 0/5 14 \* \* ? 每天下午的 2点到2点59分(整点开始，每隔5分触发)

0 0/5 14,18 \* \* ? 每天下午的 2点到2点59分、18点到18点59分(整点开始，每隔5分触发)

0 0-5 14 \* \* ? 每天下午的 2点到2点05分每分触发

0 10,44 14 ? 3 WED 3月分每周三下午的 2点10分和2点44分触发

0 15 10 ? \* MON-FRI 从周一到周五每天上午的10点15分触发

0 15 10 15 \* ? 每月15号上午10点15分触发

0 15 10 L \* ? 每月最后一天的10点15分触发

0 15 10 ? \* 6L 每月最后一周的星期五的10点15分触发

0 15 10 ? \* 6L 2002-2005 从2002年到2005年每月最后一周的星期五的10点15分触发

0 15 10 ? \* 6#3 每月的第三周的星期五开始触发

0 0 12 1/5 \* ? 每月的第一个中午开始每隔5天触发一次

0 11 11 11 11 ? 每年的11月11号 11点11分触发(光棍节)

# 三、xml配置方式添加Quartz回顾

<bean id="firstTimer" class="com.gupaoedu.timer.XmlTimer"></bean>

<bean id="firstTask"

class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">

<property name="targetObject">

<ref bean="firstTimer" />

</property>

<property name="targetMethod">

<value>execute</value>

</property>

</bean>

<bean id="firstTrigger" class="org.springframework.scheduling.quartz.CronTriggerFactoryBean">

<property name="jobDetail">

<ref bean="firstTask" />

</property>

<property name="cronExpression">

<value>0/5 \* \* \* \* ?</value>

</property>

</bean>

<bean id="startQuertz" lazy-init="false" autowire="no"

class="org.springframework.scheduling.quartz.SchedulerFactoryBean">

<property name="triggers">

<list>

<ref bean="firstTrigger" />

</list>

</property>

</bean>

# Annotation方式添加Quartz回顾

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

http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-3.0.xsd

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

# 利用反射机制实现定时任务的动态调度

实现 ApplicationContextAware

扫描类

# 定时任务重复启动的问题

1、定时任务重复启动的问题，避免注解和xml方式重复使用

2、同一时间触发多个任务，有些任务不执行的问题，使用多线程异步解决 @Async

# 总结