## Quartz定时任务管理集成

### 前言

本次做的quartz定时任务管理功能，是基于spring框架相关技术的探讨。另外quartz所用版本为2.2.2，数据表所支持数据库为mysql。若是低于2.2.0版本的quartz或其他数据库，此文档不适用，有待新的探索。

### 步骤

1. 加依赖jar包
   1. Maven导入包

在pom.xml文件中加上依赖：

<**dependency**>  
 <**groupId**>org.quartz-scheduler</**groupId**>  
 <**artifactId**>quartz</**artifactId**>  
 <**version**>2.2.2</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.quartz-scheduler</**groupId**>  
 <**artifactId**>quartz-jobs</**artifactId**>  
 <**version**>2.2.2</**version**>  
</**dependency**>

1. 配置properties文件

参见quartz.properties

注:如若不配置properties文件，则会用quartz.jar包中的quartz.properties文件。

*#调度标识名 集群中每一个实例都必须使用相同的名称***org.quartz.scheduler.instanceName** = **DefaultQuartzScheduler***#ID设置为自动获取 每一个必须不同***org.quartz.scheduler.instanceId** = **AUTO***#是否打开Quartz的RMI支持***org.quartz.scheduler.rmi.export** = **false***#是否打开Quartz的RMI支持***org.quartz.scheduler.rmi.proxy** = **false***#是否在用户指定的事务中运行任务***org.quartz.scheduler.wrapJobExecutionInUserTransaction** = **false***#ThreadPool 实现的类名***org.quartz.threadPool.class** = **org.quartz.simpl.SimpleThreadPool***#线程数量(每次scheduler.start()时，系统都会启动10个线程)***org.quartz.threadPool.threadCount** = 1**0***#线程优先级  
#org.quartz.threadPool.threadPriority = 5  
#自创建父线程***org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread** = **true***#容许的最大作业延长时间***org.quartz.jobStore.misfireThreshold** = **60000***#数据保存方式为持久化***org.quartz.jobStore.class**=**org.quartz.impl.jdbcjobstore.JobStoreTX***#数据库平台***org.quartz.jobStore.driverDelegateClass**=**org.quartz.impl.jdbcjobstore.StdJDBCDelegate***#设置为TRUE不会出现序列化非字符串类到 BLOB 时产生的类版本问题***org.quartz.jobStore.useProperties**=**false***#表的前缀***org.quartz.jobStore.tablePrefix**=**QRTZ\_***#加入集群***org.quartz.jobStore.isClustered**=**true**

1. 写Manager类

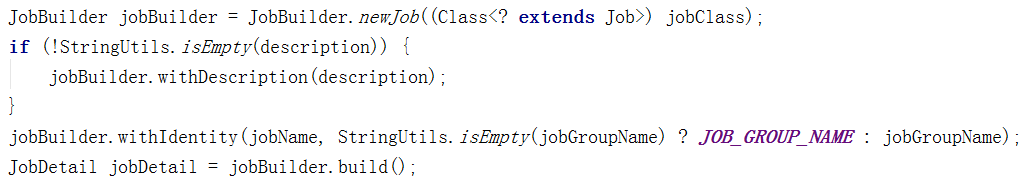
参见com.yinhai.ec.quartz.manager.QuartzManager类

创建一个类，包含所有需要的方法。并且调用通过spring自动注入的

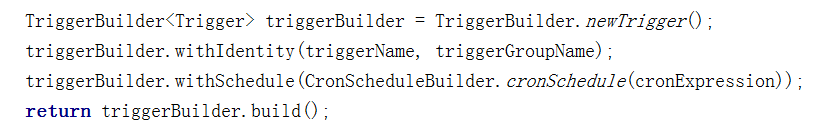
factorybean类。利用factorybean类生成scheduler对象,scheduler包含对任务调度的各种操作方法。

Scheduler相关操作需要JobDetail和Trigger，所以在调用方法之前需要获取JobDetail和Trigger。

通过JobBuilder获取jobDetail 。



通过TriggerBuilder获取Trigger 。



1. 配置xml文件

参见spring-quartz.xml

<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"**>  
  
 <**bean name="schedulerFactory" class="org.springframework.scheduling.quartz.SchedulerFactoryBean"**>  
  
 *<!-- 应用上下文 -->* <**property name="applicationContextSchedulerContextKey" value="applicationContext"**/>  
  
 *<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\* 两种配置数据源的方式根据实际情况进行选择 \*\*\*\*\*\*\*\*\*\*\*\* -->  
 <!-- 配置与应用同数据源 -->* <**property name="dataSource" ref="dataSource"** />  
 <**property name="transactionManager" ref="transactionManager"**/>  
 *<!-- 配置根据配置文件配置数据源 -->* <**property name="configLocation" value="classpath:config/quartz/quartz.properties"**/>  
  
 *<!-- 这个是必须的，QuartzScheduler 延时启动，应用启动完后 QuartzScheduler 再启动 -->* <**property name="startupDelay" value="30"**/>*<!-- 这个是可选，QuartzScheduler  
 启动时更新己存在的Job，这样就不用每次修改targetObject后删除qrtz\_job\_details表对应记录了 -->* <**property name="overwriteExistingJobs" value="true"**/>  
 <**property name="autoStartup" value="true"**/>  
 </**bean**>  
  
 <**bean id="quartzManager" class="com.yinhai.ec.quartz.manager.QuartzManager"**/>  
</**beans**>

说明：

<**property name="dataSource" ref="dataSource"** />：配置quartz相关表存放的数据源。

<**property name="configLocation" value="classpath:config/quartz/quartz.properties"**/>：指定qurtz的properties配置文件位置。

<**bean id="quartzManager" class="com.yinhai.ec.quartz.manager.QuartzManager"**/>：配置manager类的bean。

1. 写一个用于测试的执行类，必须实现org.quartz.Job接口

参见com.yinhai.ec.quartz.QuartzJobDemo类

创建一个类，实现Job类的execute方法，在execute方法中写任务调度需要运行的程序即可。若为了更好的拓展，也可以再封装两层，如com.yinhai.ec.quartz.job.QuartzDemo类，这里暂不详说。

1. 写测试类

参见com.yinhai.ec.quartz.test.QuartzTest类