**springboot集成quartz集群案例**

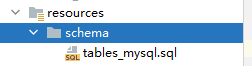
# 方式一

## 引入依赖

|  |
| --- |
| <dependency>  <groupId>org.quartz-scheduler</groupId>  <artifactId>quartz</artifactId>  <version>2.3.1</version> </dependency> <dependency>  <groupId>org.quartz-scheduler</groupId>  <artifactId>quartz-jobs</artifactId>  <version>2.3.1</version> </dependency> |

## 引入schema表结构

resource文件夹下新建



|  |
| --- |
| # # In your Quartz properties file, you'll need to set # org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate # # # By: Ron Cordell - roncordell # I didn't see this anywhere, so I thought I'd post it here. This is the script from Quartz to create the tables in a MySQL database, modified to use INNODB instead of MYISAM.  DROP TABLE IF EXISTS QRTZ\_FIRED\_TRIGGERS; DROP TABLE IF EXISTS QRTZ\_PAUSED\_TRIGGER\_GRPS; DROP TABLE IF EXISTS QRTZ\_SCHEDULER\_STATE; DROP TABLE IF EXISTS QRTZ\_LOCKS; DROP TABLE IF EXISTS QRTZ\_SIMPLE\_TRIGGERS; DROP TABLE IF EXISTS QRTZ\_SIMPROP\_TRIGGERS; DROP TABLE IF EXISTS QRTZ\_CRON\_TRIGGERS; DROP TABLE IF EXISTS QRTZ\_BLOB\_TRIGGERS; DROP TABLE IF EXISTS QRTZ\_TRIGGERS; DROP TABLE IF EXISTS QRTZ\_JOB\_DETAILS; DROP TABLE IF EXISTS QRTZ\_CALENDARS;  CREATE TABLE QRTZ\_JOB\_DETAILS( SCHED\_NAME VARCHAR(120) NOT NULL, JOB\_NAME VARCHAR(190) NOT NULL, JOB\_GROUP VARCHAR(190) NOT NULL, DESCRIPTION VARCHAR(250) NULL, JOB\_CLASS\_NAME VARCHAR(250) NOT NULL, IS\_DURABLE VARCHAR(1) NOT NULL, IS\_NONCONCURRENT VARCHAR(1) NOT NULL, IS\_UPDATE\_DATA VARCHAR(1) NOT NULL, REQUESTS\_RECOVERY VARCHAR(1) NOT NULL, JOB\_DATA BLOB NULL, PRIMARY KEY (SCHED\_NAME,JOB\_NAME,JOB\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_TRIGGERS ( SCHED\_NAME VARCHAR(120) NOT NULL, TRIGGER\_NAME VARCHAR(190) NOT NULL, TRIGGER\_GROUP VARCHAR(190) NOT NULL, JOB\_NAME VARCHAR(190) NOT NULL, JOB\_GROUP VARCHAR(190) NOT NULL, DESCRIPTION VARCHAR(250) NULL, NEXT\_FIRE\_TIME BIGINT(13) NULL, PREV\_FIRE\_TIME BIGINT(13) NULL, PRIORITY INTEGER NULL, TRIGGER\_STATE VARCHAR(16) NOT NULL, TRIGGER\_TYPE VARCHAR(8) NOT NULL, START\_TIME BIGINT(13) NOT NULL, END\_TIME BIGINT(13) NULL, CALENDAR\_NAME VARCHAR(190) NULL, MISFIRE\_INSTR SMALLINT(2) NULL, JOB\_DATA BLOB NULL, PRIMARY KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP), FOREIGN KEY (SCHED\_NAME,JOB\_NAME,JOB\_GROUP) REFERENCES QRTZ\_JOB\_DETAILS(SCHED\_NAME,JOB\_NAME,JOB\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_SIMPLE\_TRIGGERS ( SCHED\_NAME VARCHAR(120) NOT NULL, TRIGGER\_NAME VARCHAR(190) NOT NULL, TRIGGER\_GROUP VARCHAR(190) NOT NULL, REPEAT\_COUNT BIGINT(7) NOT NULL, REPEAT\_INTERVAL BIGINT(12) NOT NULL, TIMES\_TRIGGERED BIGINT(10) NOT NULL, PRIMARY KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP), FOREIGN KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP) REFERENCES QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_CRON\_TRIGGERS ( SCHED\_NAME VARCHAR(120) NOT NULL, TRIGGER\_NAME VARCHAR(190) NOT NULL, TRIGGER\_GROUP VARCHAR(190) NOT NULL, CRON\_EXPRESSION VARCHAR(120) NOT NULL, TIME\_ZONE\_ID VARCHAR(80), PRIMARY KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP), FOREIGN KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP) REFERENCES QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_SIMPROP\_TRIGGERS  (  SCHED\_NAME VARCHAR(120) NOT NULL,  TRIGGER\_NAME VARCHAR(190) NOT NULL,  TRIGGER\_GROUP VARCHAR(190) NOT NULL,  STR\_PROP\_1 VARCHAR(512) NULL,  STR\_PROP\_2 VARCHAR(512) NULL,  STR\_PROP\_3 VARCHAR(512) NULL,  INT\_PROP\_1 INT NULL,  INT\_PROP\_2 INT NULL,  LONG\_PROP\_1 BIGINT NULL,  LONG\_PROP\_2 BIGINT NULL,  DEC\_PROP\_1 NUMERIC(13,4) NULL,  DEC\_PROP\_2 NUMERIC(13,4) NULL,  BOOL\_PROP\_1 VARCHAR(1) NULL,  BOOL\_PROP\_2 VARCHAR(1) NULL,  PRIMARY KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP),  FOREIGN KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP)  REFERENCES QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_BLOB\_TRIGGERS ( SCHED\_NAME VARCHAR(120) NOT NULL, TRIGGER\_NAME VARCHAR(190) NOT NULL, TRIGGER\_GROUP VARCHAR(190) NOT NULL, BLOB\_DATA BLOB NULL, PRIMARY KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP), INDEX (SCHED\_NAME,TRIGGER\_NAME, TRIGGER\_GROUP), FOREIGN KEY (SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP) REFERENCES QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_CALENDARS ( SCHED\_NAME VARCHAR(120) NOT NULL, CALENDAR\_NAME VARCHAR(190) NOT NULL, CALENDAR BLOB NOT NULL, PRIMARY KEY (SCHED\_NAME,CALENDAR\_NAME)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_PAUSED\_TRIGGER\_GRPS ( SCHED\_NAME VARCHAR(120) NOT NULL, TRIGGER\_GROUP VARCHAR(190) NOT NULL, PRIMARY KEY (SCHED\_NAME,TRIGGER\_GROUP)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_FIRED\_TRIGGERS ( SCHED\_NAME VARCHAR(120) NOT NULL, ENTRY\_ID VARCHAR(95) NOT NULL, TRIGGER\_NAME VARCHAR(190) NOT NULL, TRIGGER\_GROUP VARCHAR(190) NOT NULL, INSTANCE\_NAME VARCHAR(190) NOT NULL, FIRED\_TIME BIGINT(13) NOT NULL, SCHED\_TIME BIGINT(13) NOT NULL, PRIORITY INTEGER NOT NULL, STATE VARCHAR(16) NOT NULL, JOB\_NAME VARCHAR(190) NULL, JOB\_GROUP VARCHAR(190) NULL, IS\_NONCONCURRENT VARCHAR(1) NULL, REQUESTS\_RECOVERY VARCHAR(1) NULL, PRIMARY KEY (SCHED\_NAME,ENTRY\_ID)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_SCHEDULER\_STATE ( SCHED\_NAME VARCHAR(120) NOT NULL, INSTANCE\_NAME VARCHAR(190) NOT NULL, LAST\_CHECKIN\_TIME BIGINT(13) NOT NULL, CHECKIN\_INTERVAL BIGINT(13) NOT NULL, PRIMARY KEY (SCHED\_NAME,INSTANCE\_NAME)) ENGINE=InnoDB;  CREATE TABLE QRTZ\_LOCKS ( SCHED\_NAME VARCHAR(120) NOT NULL, LOCK\_NAME VARCHAR(40) NOT NULL, PRIMARY KEY (SCHED\_NAME,LOCK\_NAME)) ENGINE=InnoDB;  CREATE INDEX IDX\_QRTZ\_J\_REQ\_RECOVERY ON QRTZ\_JOB\_DETAILS(SCHED\_NAME,REQUESTS\_RECOVERY); CREATE INDEX IDX\_QRTZ\_J\_GRP ON QRTZ\_JOB\_DETAILS(SCHED\_NAME,JOB\_GROUP);  CREATE INDEX IDX\_QRTZ\_T\_J ON QRTZ\_TRIGGERS(SCHED\_NAME,JOB\_NAME,JOB\_GROUP); CREATE INDEX IDX\_QRTZ\_T\_JG ON QRTZ\_TRIGGERS(SCHED\_NAME,JOB\_GROUP); CREATE INDEX IDX\_QRTZ\_T\_C ON QRTZ\_TRIGGERS(SCHED\_NAME,CALENDAR\_NAME); CREATE INDEX IDX\_QRTZ\_T\_G ON QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_GROUP); CREATE INDEX IDX\_QRTZ\_T\_STATE ON QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_STATE); CREATE INDEX IDX\_QRTZ\_T\_N\_STATE ON QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP,TRIGGER\_STATE); CREATE INDEX IDX\_QRTZ\_T\_N\_G\_STATE ON QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_GROUP,TRIGGER\_STATE); CREATE INDEX IDX\_QRTZ\_T\_NEXT\_FIRE\_TIME ON QRTZ\_TRIGGERS(SCHED\_NAME,NEXT\_FIRE\_TIME); CREATE INDEX IDX\_QRTZ\_T\_NFT\_ST ON QRTZ\_TRIGGERS(SCHED\_NAME,TRIGGER\_STATE,NEXT\_FIRE\_TIME); CREATE INDEX IDX\_QRTZ\_T\_NFT\_MISFIRE ON QRTZ\_TRIGGERS(SCHED\_NAME,MISFIRE\_INSTR,NEXT\_FIRE\_TIME); CREATE INDEX IDX\_QRTZ\_T\_NFT\_ST\_MISFIRE ON QRTZ\_TRIGGERS(SCHED\_NAME,MISFIRE\_INSTR,NEXT\_FIRE\_TIME,TRIGGER\_STATE); CREATE INDEX IDX\_QRTZ\_T\_NFT\_ST\_MISFIRE\_GRP ON QRTZ\_TRIGGERS(SCHED\_NAME,MISFIRE\_INSTR,NEXT\_FIRE\_TIME,TRIGGER\_GROUP,TRIGGER\_STATE);  CREATE INDEX IDX\_QRTZ\_FT\_TRIG\_INST\_NAME ON QRTZ\_FIRED\_TRIGGERS(SCHED\_NAME,INSTANCE\_NAME); CREATE INDEX IDX\_QRTZ\_FT\_INST\_JOB\_REQ\_RCVRY ON QRTZ\_FIRED\_TRIGGERS(SCHED\_NAME,INSTANCE\_NAME,REQUESTS\_RECOVERY); CREATE INDEX IDX\_QRTZ\_FT\_J\_G ON QRTZ\_FIRED\_TRIGGERS(SCHED\_NAME,JOB\_NAME,JOB\_GROUP); CREATE INDEX IDX\_QRTZ\_FT\_JG ON QRTZ\_FIRED\_TRIGGERS(SCHED\_NAME,JOB\_GROUP); CREATE INDEX IDX\_QRTZ\_FT\_T\_G ON QRTZ\_FIRED\_TRIGGERS(SCHED\_NAME,TRIGGER\_NAME,TRIGGER\_GROUP); CREATE INDEX IDX\_QRTZ\_FT\_TG ON QRTZ\_FIRED\_TRIGGERS(SCHED\_NAME,TRIGGER\_GROUP);  commit; |

## application.properties 配置quartz相关参数

|  |
| --- |
| #quartz 相关配置  spring.quartz.scheduler-name=YizhuScheduler  spring.quartz.job-store-type=jdbc  spring.quartz.jdbc.initialize-schema=always  spring.quartz.jdbc.schema=classpath:schema/tables\_mysql.sql |

## 具体job案例（PrintLogJob）

|  |
| --- |
| package com.jr.job;  import com.jr.entity.User;  import com.jr.service.IUserService;  import lombok.extern.slf4j.Slf4j;  import org.quartz.JobExecutionContext;  import org.quartz.JobExecutionException;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.scheduling.quartz.QuartzJobBean;  import org.springframework.stereotype.Service;  import java.time.LocalDateTime;  @Component  @Slf4j  public class PrintLogJob extends QuartzJobBean {  @Autowired  IUserService iUserService;  @Override  protected void executeInternal(JobExecutionContext jobExecutionContext) throws JobExecutionException {  User user = new User();  user.setName("user1");  user.setCreated(LocalDateTime.now());  iUserService.save(user);  log.info("COMPUTER NAME:【{}】,logTime:【{}】",System.getenv().get("COMPUTERNAME"), LocalDateTime.now());  }  } |

## quartz配置类(PrintLogQuartzConfig)

|  |
| --- |
| /\*\*  \* Quartz相关配置  \*/  @Configuration  public class PrintLogQuartzConfig {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*printLogJob begin\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  @Bean  public JobDetail printLogJobConfig(){  return JobBuilder.newJob(PrintLogJob.class)  .withIdentity("PrintLogJob")  .storeDurably()  .build();  }  @Bean  public Trigger printLogJobTrigger(){  return TriggerBuilder.newTrigger().forJob(printLogJobConfig())  .withIdentity("PrintLogJob")  .withSchedule(CronScheduleBuilder.cronSchedule("1/1 \* \* \* \* ?"))  .build();  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*printLogJob end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  } |

# 方式二

## 引入依赖

|  |
| --- |
| <dependency>  <groupId>org.quartz-scheduler</groupId>  <artifactId>quartz</artifactId>  <version>2.3.1</version> </dependency> <dependency>  <groupId>org.quartz-scheduler</groupId>  <artifactId>quartz-jobs</artifactId>  <version>2.3.1</version> </dependency> |

## resource下新建quartz.properties 配置文件

|  |
| --- |
| #调度配置  #调度器实例名称  org.quartz.scheduler.instanceName=SsmScheduler  #调度器实例编号自动生成  org.quartz.scheduler.instanceId=AUTO  #是否在Quartz执行一个job前使用UserTransaction  org.quartz.scheduler.wrapJobExecutionInUserTransaction=false  #线程池配置  #线程池的实现类  org.quartz.threadPool.class=org.quartz.simpl.SimpleThreadPool  #线程池中的线程数量  org.quartz.threadPool.threadCount=10  #线程优先级  org.quartz.threadPool.threadPriority=5  #配置是否启动自动加载数据库内的定时任务，默认true  org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread=true  #是否设置为守护线程，设置后任务将不会执行  #org.quartz.threadPool.makeThreadsDaemons=true  #持久化方式配置  #JobDataMaps是否都为String类型  org.quartz.jobStore.useProperties=true  #数据表的前缀，默认QRTZ\_  org.quartz.jobStore.tablePrefix=QRTZ\_  #最大能忍受的触发超时时间  org.quartz.jobStore.misfireThreshold=60000  #是否以集群方式运行  org.quartz.jobStore.isClustered=true  #调度实例失效的检查时间间隔，单位毫秒  org.quartz.jobStore.clusterCheckinInterval=2000  #数据保存方式为数据库持久化  org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStoreTX  #数据库代理类，一般org.quartz.impl.jdbcjobstore.StdJDBCDelegate可以满足大部分数据库  org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.StdJDBCDelegate  #数据库别名 随便取  org.quartz.jobStore.dataSource=qzDS  #数据库连接池，将其设置为druid  org.quartz.dataSource.qzDS.connectionProvider.class=com.example.cluster.quartz.config.DruidConnectionProvider  #数据库引擎  org.quartz.dataSource.qzDS.driver=com.mysql.cj.jdbc.Driver  #数据库连接  org.quartz.dataSource.qzDS.URL=jdbc:mysql://127.0.0.1:3306/test-quartz?serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8&useSSL=true  #数据库用户  org.quartz.dataSource.qzDS.user=root  #数据库密码  org.quartz.dataSource.qzDS.password=123456  #允许最大连接  org.quartz.dataSource.qzDS.maxConnection=5  #验证查询sql,可以不设置  org.quartz.dataSource.qzDS.validationQuery=select 0 from dual |

## 具体job案例（PrintLogJob）

|  |
| --- |
| package com.hu.job;  import lombok.extern.slf4j.Slf4j;  import org.quartz.JobExecutionContext;  import org.quartz.JobExecutionException;  import org.springframework.scheduling.quartz.QuartzJobBean;  import org.springframework.stereotype.Component;  import java.text.SimpleDateFormat;  import java.time.LocalDateTime;  import java.util.Date;  @Component  @Slf4j  public class HelloWorldJob extends QuartzJobBean {  @Override  protected void executeInternal(JobExecutionContext jobExecutionContext) throws JobExecutionException {  log.info("case1 HelloWorldJob at {}", LocalDateTime.now());  }  } |

## QuartzSchedulerConfig

|  |
| --- |
| package com.hu.config;   import com.hu.job.HelloWorldJob; import lombok.extern.slf4j.Slf4j; import org.quartz.CronTrigger; import org.quartz.JobDetail; import org.quartz.spi.JobFactory; import org.quartz.spi.TriggerFiredBundle; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.beans.factory.config.AutowireCapableBeanFactory; import org.springframework.beans.factory.config.PropertiesFactoryBean; import org.springframework.context.ApplicationContext; import org.springframework.context.ApplicationContextAware; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.ClassPathResource; import org.springframework.scheduling.quartz.CronTriggerFactoryBean; import org.springframework.scheduling.quartz.JobDetailFactoryBean; import org.springframework.scheduling.quartz.SchedulerFactoryBean; import org.springframework.scheduling.quartz.SpringBeanJobFactory;  import javax.sql.DataSource; import java.io.IOException; import java.util.Properties;  */\*\*  \* Simple to Introduction  \* className: QuartzSchedulerConfig  \*  \* @author EricYang  \* @version 2018/11/17 17:53  \*/* @Configuration @Slf4j public class QuartzSchedulerConfig {   @Autowired  private DataSource dataSource;   private static final String *QUARTZ\_PROPERTIES\_NAME* = "/quartz.properties";    @Bean  public Properties quartzProperties() throws IOException {  PropertiesFactoryBean propertiesFactoryBean = new PropertiesFactoryBean();  propertiesFactoryBean.setLocation(new ClassPathResource(*QUARTZ\_PROPERTIES\_NAME*));  propertiesFactoryBean.afterPropertiesSet();  return propertiesFactoryBean.getObject();  }   */\*\*  \* 继承org.springframework.scheduling.quartz.SpringBeanJobFactory  \* 实现任务实例化方式  \*/* class AutowiringSpringBeanJobFactory extends SpringBeanJobFactory implements ApplicationContextAware {   private transient AutowireCapableBeanFactory beanFactory;   @Override  public void setApplicationContext(final ApplicationContext context) {  beanFactory = context.getAutowireCapableBeanFactory();  }   @Override  protected Object createJobInstance(final TriggerFiredBundle bundle) throws Exception {  final Object job = super.createJobInstance(bundle);  beanFactory.autowireBean(job);  return job;  }  }   @Bean  public JobFactory jobFactory(ApplicationContext applicationContext) {  AutowiringSpringBeanJobFactory jobFactory = new AutowiringSpringBeanJobFactory();  jobFactory.setApplicationContext(applicationContext);  return jobFactory;  }   @Bean  public SchedulerFactoryBean schedulerFactoryBean(JobFactory jobFactory, CronTrigger[] cronTrigger,  JobDetail[] jobDetails) {  SchedulerFactoryBean factoryBean = new SchedulerFactoryBean();  try {  factoryBean.setQuartzProperties(quartzProperties());  factoryBean.setDataSource(dataSource);  factoryBean.setJobFactory(jobFactory);  factoryBean.setTriggers(cronTrigger);  factoryBean.setJobDetails(jobDetails);  factoryBean.setOverwriteExistingJobs(true);  } catch (Exception e) {  *log*.error("Failed to load config file {}.", *QUARTZ\_PROPERTIES\_NAME*, e);  throw new RuntimeException("LoadingConfigFileError", e);  }   return factoryBean;  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分布式任务调度helloWorld begin\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* @Bean  public JobDetailFactoryBean helloWorldJobDetail() {  JobDetailFactoryBean jobDetailFactoryBean = new JobDetailFactoryBean();  jobDetailFactoryBean.setJobClass(HelloWorldJob.class);  jobDetailFactoryBean.setDurability(true);  return jobDetailFactoryBean;  }  @Bean  public CronTriggerFactoryBean helloWorldjobTrigger(JobDetail helloWorldJobDetail) {  CronTriggerFactoryBean cronTriggerFactoryBean = new CronTriggerFactoryBean();  cronTriggerFactoryBean.setJobDetail(helloWorldJobDetail);  cronTriggerFactoryBean.setCronExpression("2/2 \* \* \* \* ?");  return cronTriggerFactoryBean;  }   */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分布式任务调度helloWorld end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* } |

# 两者对比

方式一更加灵活，修改cron表达式，可以重建表结构并及时更新。