链滴

通过 quartz 来实现分布式定时任务

作者：laker

原文链接：https://ld246.com/article/1536320244645

来源网站：链滴

确定spring和quartz的版本

```
<spring.version>4.2.6.RELEASE</spring.version>
<quartz-version>2.2.1</quartz-version>
```

maven引入jar

```
<dependency>
    <groupId>org.quartz-scheduler</groupId>
    <artifactId>quartz</artifactId>
    <version>${quartz-version}</version>
</dependency>
<dependency>
    <groupId>org.quartz-scheduler</groupId>
    <artifactId>quartz-jobs</artifactId>
    <version>${quartz-version}</version>
</dependency>
```

初始化表结构

数据库的表结构和quartz版本需要对应起来

sql

见最后quartz-sql

配置application-quartz.xml

```
<?xml version="1.0" encoding="utf-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.
rg/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/sche
a/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean name="quartzScheduler" class="org.springframework.scheduling.quartz.SchedulerF
ctoryBean">
        <property name="dataSource">
            <ref bean="dataSource"/>
        </property>
        <property name="applicationContextSchedulerContextKey" value="applicationContextK
y"/>
        <property name="configLocation" value="classpath:quartz.properties"/>
        <property name="triggers">
            <list>
                <ref bean="triggerUpsmsTimeout"/>
            </list>
        </property>
        <!-- 修改job时，更新到数据库 -->
        <property name="overwriteExistingJobs" value="true" />
    </bean>
    <bean id="jobDetailUpsmsTimeout" class="org.springframework.scheduling.quartz.JobDet
```

```
ilFactoryBean">
        <property name="jobClass">
            <value>com.channelsoft.quartz.job.UpsmsTimeout</value>
        </property>
        <property name="durability" value="true"/>
        <property name="requestsRecovery" value="true"/>
    </bean>
    <bean id="triggerUpsmsTimeout" class="org.springframework.scheduling.quartz.CronTrig
erFactoryBean">
        <property name="jobDetail" ref="jobDetailUpsmsTimeout"/>
        <property name="cronExpression" value="1 */10 * ? * * *"/>
    </bean>
</beans>
```

java

```
package com.channelsoft.quartz.job;
import com.channelsoft.constant.Constants;
import com.channelsoft.po.IvrOrder;
import com.channelsoft.po.SmsRecord;
import com.channelsoft.service.IvrOrderService;
import com.channelsoft.service.SmsRecordService;
import org.quartz.DisallowConcurrentExecution;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;
import org.quartz.PersistJobDataAfterExecution;
import org.quartz.SchedulerException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.scheduling.quartz.QuartzJobBean;

import java.util.ArrayList;
import java.util.List;


@PersistJobDataAfterExecution
@DisallowConcurrentExecution// 不允许并发执行
public class UpsmsTimeout extends QuartzJobBean {
    private static final Logger logger = LoggerFactory.getLogger(UpsmsTimeout.class);

    @Override
    protected void executeInternal(JobExecutionContext jobexecutioncontext) throws JobExecu
ionException {
        logger.info("UpsmsTimeout start");
        SmsRecordService smsRecordService = getApplicationContext(jobexecutioncontext).get
ean("smsRecordService", SmsRecordService.class);
    }

    private ApplicationContext getApplicationContext(final JobExecutionContext jobexecutionc
ntext) {
        try {
            return (ApplicationContext) jobexecutioncontext.getScheduler().getContext().get("appl
cationContextKey");
```

```
        } catch (SchedulerException e) {
            logger.error("jobexecutioncontext.getScheduler().getContext() error!", e);
            throw new RuntimeException(e);
        }
    }

}
```

quartz-sql

```
#
# 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 crea
e 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(200) NOT NULL,
JOB_GROUP VARCHAR(200) 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(200) NOT NULL,
TRIGGER_GROUP VARCHAR(200) NOT NULL,
JOB_NAME VARCHAR(200) NOT NULL,
JOB_GROUP VARCHAR(200) 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(200) 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(200) NOT NULL,
TRIGGER_GROUP VARCHAR(200) 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(200) NOT NULL,
TRIGGER_GROUP VARCHAR(200) 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(200) NOT NULL,
    TRIGGER_GROUP VARCHAR(200) 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(200) NOT NULL,
TRIGGER_GROUP VARCHAR(200) 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(200) 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(200) 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(200) NOT NULL,
TRIGGER_GROUP VARCHAR(200) NOT NULL,
INSTANCE_NAME VARCHAR(200) 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(200) NULL,
JOB_GROUP VARCHAR(200) 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(200) 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,REQUEST
_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,TR
GGER_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_
IME);
CREATE INDEX IDX_QRTZ_T_NFT_ST ON QRTZ_TRIGGERS(SCHED_NAME,TRIGGER_STATE,NEX
_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_IN
TR,NEXT_FIRE_TIME,TRIGGER_STATE);
CREATE INDEX IDX_QRTZ_T_NFT_ST_MISFIRE_GRP ON QRTZ_TRIGGERS(SCHED_NAME,MISFIR
_INSTR,NEXT_FIRE_TIME,TRIGGER_GROUP,TRIGGER_STATE);

CREATE INDEX IDX_QRTZ_FT_TRIG_INST_NAME ON QRTZ_FIRED_TRIGGERS(SCHED_NAME,IN
TANCE_NAME);
CREATE INDEX IDX_QRTZ_FT_INST_JOB_REQ_RCVRY ON QRTZ_FIRED_TRIGGERS(SCHED_NAM
,INSTANCE_NAME,REQUESTS_RECOVERY);
CREATE INDEX IDX_QRTZ_FT_J_G ON QRTZ_FIRED_TRIGGERS(SCHED_NAME,JOB_NAME,JOB_
ROUP);
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,
RIGGER_GROUP);
CREATE INDEX IDX_QRTZ_FT_TG ON QRTZ_FIRED_TRIGGERS(SCHED_NAME,TRIGGER_GROUP);

commit;
```