Quartz整合spring(注解方式)持久化

> 在实际开发中,我们的定时任务都是随时保存在数据库中的,所以我们需要持久化到数据库,quartz自带持久化功能,我们只需要引入即可

1基础环境

> 进入quartz的官网http://www.quartz-scheduler.org/，点击Downloads，下载后在目录\docs\dbTables下有常用数据库创建quartz表的脚本

1.1 sql

```
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)
);

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          int          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)
);

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)
);

create table QRTZ_LOCKS
(
  SCHED_NAME varchar(120) not null,
  LOCK_NAME  varchar(40)  not null,
  primary key (SCHED_NAME, LOCK_NAME)
);

create table QRTZ_PAUSED_TRIGGER_GRPS
(
  SCHED_NAME    varchar(120) not null,
  TRIGGER_GROUP varchar(200) not null,
  primary key (SCHED_NAME, TRIGGER_GROUP)
);

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)
);

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       int          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),
  constraint QRTZ_TRIGGERS_ibfk_1
  foreign key (SCHED_NAME, JOB_NAME, JOB_GROUP) references QRTZ_JOB_DETAILS
(SCHED_NAME, JOB_NAME, JOB_GROUP)
);

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),
  constraint QRTZ_BLOB_TRIGGERS_ibfk_1
  foreign key (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP) references QRTZ_TRIGGERS
(SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP)
);

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(200) not null,
  TIME_ZONE_ID    varchar(80)  null,
  primary key (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP),
  constraint QRTZ_CRON_TRIGGERS_ibfk_1
  foreign key (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP) references QRTZ_TRIGGERS
(SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP)
);

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),
  constraint QRTZ_SIMPLE_TRIGGERS_ibfk_1
  foreign key (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP) references QRTZ_TRIGGERS
(SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP)
);

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    decimal(13, 4) null,
  DEC_PROP_2    decimal(13, 4) null,
  BOOL_PROP_1   varchar(1)     null,
  BOOL_PROP_2   varchar(1)     null,
```

```
  primary key (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP),
  constraint QRTZ_SIMPROP_TRIGGERS_ibfk_1
  foreign key (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP) references QRTZ_TRIGGERS
(SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP)
);

create index SCHED_NAME
  on QRTZ_TRIGGERS (SCHED_NAME, JOB_NAME, JOB_GROUP);
```

1.2 pom

```
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.10.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>1.3.0</version>
        </dependency>
        <dependency>
            <groupId>org.quartz-scheduler</groupId>
            <artifactId>quartz</artifactId>
            <version>2.3.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
        </dependency>
        <!--<dependency>-->
            <!--<groupId>org.springframework.boot</groupId>-->
            <!--<artifactId>spring-boot-starter-quartz</artifactId>-->
        <!--</dependency>-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
```

```
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.0.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.mchange/c3p0 -->
        <dependency>
            <groupId>com.mchange</groupId>
            <artifactId>c3p0</artifactId>
            <version>0.9.5.2</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>


    </build>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
```

```
            </snapshots>
        </repository>
    </repositories>

    <pluginRepositories>
        <pluginRepository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </pluginRepository>
        <pluginRepository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>
```

1.3application.yml

```
spring:
  datasource:
    url: jdbc:mysql:///quartz?useUnicode=true
    username: root
    password: qishimeiyoumima
    driver-class-name: com.mysql.jdbc.Driver

mybatis:
  # mapper-locations:  classpath:com/example/demo/mapper/*.xml
  type-aliases-package: com.qianfeng.quartz.spring.entity
```

1.4 quartz.properties

```
# 固定前缀org.quartz
# 主要分为scheduler、threadPool、jobStore、plugin等部分
#
#
org.quartz.scheduler.instanceName = DefaultQuartzScheduler
org.quartz.scheduler.rmi.export = false
org.quartz.scheduler.rmi.proxy = false
org.quartz.scheduler.wrapJobExecutionInUserTransaction = false

# 实例化ThreadPool时，使用的线程类为SimpleThreadPool
```

```
org.quartz.threadPool.class = org.quartz.simpl.SimpleThreadPool

# threadCount和threadPriority将以setter的形式注入ThreadPool实例
# 并发个数
org.quartz.threadPool.threadCount = 5
# 优先级
org.quartz.threadPool.threadPriority = 5
org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread = true

org.quartz.jobStore.misfireThreshold = 5000

# 默认存储在内存中
#org.quartz.jobStore.class = org.quartz.simpl.RAMJobStore

#持久化
org.quartz.jobStore.class = org.quartz.impl.jdbcjobstore.JobStoreTX
#表的前缀
org.quartz.jobStore.tablePrefix = QRTZ_
#此处的值必须和下面的qzDS保持一致
org.quartz.jobStore.dataSource = qzDS
#qzDS数据源属性
org.quartz.dataSource.qzDS.driver = com.mysql.jdbc.Driver

org.quartz.dataSource.qzDS.URL = jdbc:mysql:///quartz?
useUnicode=true&characterEncoding=UTF-8

org.quartz.dataSource.qzDS.user = root

org.quartz.dataSource.qzDS.password = qishimeiyoumima

org.quartz.dataSource.qzDS.maxConnections = 10
```

2 案例

2.1 定义基类

```
package com.qianfeng.quartz.spring.job;

import org.quartz.Job;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;

public interface BaseJob extends Job{
    public void execute(JobExecutionContext context) throws JobExecutionException;
}
```

2.2 定义实现类

> 实现类用于声明我们要做什么任务

2.2.1 HelloJob

```
package com.qianfeng.quartz.spring.job;

import java.util.Date;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;

public class HelloJob implements BaseJob {

    private static Logger _log = LoggerFactory.getLogger(HelloJob.class);

    public HelloJob() {

    }

    public void execute(JobExecutionContext context)
        throws JobExecutionException {
        _log.error("Hello Job执行时间: " + new Date());

    }
}
```

2.2.2 NewJob

```
package com.qianfeng.quartz.spring.job;

import java.util.Date;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;

public class NewJob implements BaseJob {

    private static Logger _log = LoggerFactory.getLogger(NewJob.class);

    public NewJob() {

    }

    public void execute(JobExecutionContext context)
        throws JobExecutionException {
        _log.error("New Job执行时间: " + new Date());

    }
}
```

2.3 定义任务实体对象

> 用于存放定时任务的相关信息的对象

```
package com.qianfeng.quartz.spring.entity;

import java.math.BigInteger;

public class JobAndTrigger {
    private String JOB_NAME;
    private String JOB_GROUP;
    private String JOB_CLASS_NAME;
    private String TRIGGER_NAME;
    private String TRIGGER_GROUP;
    private BigInteger REPEAT_INTERVAL;
    private BigInteger TIMES_TRIGGERED;
    private String CRON_EXPRESSION;
    private String TIME_ZONE_ID;

    public String getJOB_NAME() {
        return JOB_NAME;
    }
    public void setJOB_NAME(String jOB_NAME) {
        JOB_NAME = jOB_NAME;
    }
    public String getJOB_GROUP() {
        return JOB_GROUP;
    }
    public void setJOB_GROUP(String jOB_GROUP) {
        JOB_GROUP = jOB_GROUP;
    }
    public String getJOB_CLASS_NAME() {
        return JOB_CLASS_NAME;
    }
    public void setJOB_CLASS_NAME(String jOB_CLASS_NAME) {
        JOB_CLASS_NAME = jOB_CLASS_NAME;
    }
    public String getTRIGGER_NAME() {
        return TRIGGER_NAME;
    }
    public void setTRIGGER_NAME(String tRIGGER_NAME) {
        TRIGGER_NAME = tRIGGER_NAME;
    }
    public String getTRIGGER_GROUP() {
        return TRIGGER_GROUP;
    }
    public void setTRIGGER_GROUP(String tRIGGER_GROUP) {
        TRIGGER_GROUP = tRIGGER_GROUP;
    }
    public BigInteger getREPEAT_INTERVAL() {
        return REPEAT_INTERVAL;
    }
    public void setREPEAT_INTERVAL(BigInteger rEPEAT_INTERVAL) {
```

```
        REPEAT_INTERVAL = rEPEAT_INTERVAL;
    }
    public BigInteger getTIMES_TRIGGERED() {
        return TIMES_TRIGGERED;
    }
    public void setTIMES_TRIGGERED(BigInteger tIMES_TRIGGERED) {
        TIMES_TRIGGERED = tIMES_TRIGGERED;
    }
    public String getCRON_EXPRESSION() {
        return CRON_EXPRESSION;
    }
    public void setCRON_EXPRESSION(String cRON_EXPRESSION) {
        CRON_EXPRESSION = cRON_EXPRESSION;
    }
    public String getTIME_ZONE_ID() {
        return TIME_ZONE_ID;
    }
    public void setTIME_ZONE_ID(String tIME_ZONE_ID) {
        TIME_ZONE_ID = tIME_ZONE_ID;
    }

}
```

2.4 定义数据库操作对象

```
package com.qianfeng.quartz.spring.dao;

import com.qianfeng.quartz.spring.entity.JobAndTrigger;
import org.apache.ibatis.annotations.Select;

import java.util.List;
//查询所有定义任务相关信息的接口
public interface JobAndTriggerMapper {
    @Select("SELECT " +
            "QRTZ_JOB_DETAILS.JOB_NAME, " +
            "QRTZ_JOB_DETAILS.JOB_GROUP, " +
            "QRTZ_JOB_DETAILS.JOB_CLASS_NAME, " +
            "QRTZ_TRIGGERS.TRIGGER_NAME, " +
            "QRTZ_TRIGGERS.TRIGGER_GROUP, " +
            "QRTZ_CRON_TRIGGERS.CRON_EXPRESSION, " +
            "QRTZ_CRON_TRIGGERS.TIME_ZONE_ID " +
            "FROM " +
            "QRTZ_JOB_DETAILS " +
            "JOIN QRTZ_TRIGGERS " +
            "JOIN QRTZ_CRON_TRIGGERS ON QRTZ_JOB_DETAILS.JOB_NAME =
QRTZ_TRIGGERS.JOB_NAME " +
            "AND QRTZ_TRIGGERS.TRIGGER_NAME = QRTZ_CRON_TRIGGERS.TRIGGER_NAME " +
            "AND QRTZ_TRIGGERS.TRIGGER_GROUP = QRTZ_CRON_TRIGGERS.TRIGGER_GROUP")
    public List<JobAndTrigger> getJobAndTriggerDetails();
}
```

2.5 service

2.5.1 接口

> 操作定时任务信息的接口,目的是获取我们想要的信息

```
package com.qianfeng.quartz.spring.service;


import com.qianfeng.quartz.spring.entity.JobAndTrigger;
import com.github.pagehelper.PageInfo;

public interface IJobAndTriggerService {
    public PageInfo<JobAndTrigger> getJobAndTriggerDetails(int pageNum, int pageSize);
}
```

2.5.2 实现类

```
package com.qianfeng.quartz.spring.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.qianfeng.quartz.spring.dao.JobAndTriggerMapper;
import com.qianfeng.quartz.spring.entity.JobAndTrigger;
import com.qianfeng.quartz.spring.service.IJobAndTriggerService;
import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;


@Service
public class JobAndTriggerImpl implements IJobAndTriggerService{

    @Autowired
    private JobAndTriggerMapper jobAndTriggerMapper;

    public PageInfo<JobAndTrigger> getJobAndTriggerDetails(int pageNum, int pageSize) {
        PageHelper.startPage(pageNum, pageSize);
        List<JobAndTrigger> list = jobAndTriggerMapper.getJobAndTriggerDetails();
        PageInfo<JobAndTrigger> page = new PageInfo<JobAndTrigger>(list);
        return page;
    }

}
```

2.6 controller

> 用于对定时任务进行增删改查的controller

```
package com.qianfeng.quartz.spring.controller;

import com.qianfeng.quartz.spring.entity.JobAndTrigger;
import com.qianfeng.quartz.spring.job.BaseJob;
import com.qianfeng.quartz.spring.service.IJobAndTriggerService;
import com.github.pagehelper.PageInfo;
import org.quartz.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;


@RestController
@RequestMapping(value="/job")
public class JobController
{
    @Autowired
    private IJobAndTriggerService iJobAndTriggerService;

    //加入Qulifier注解，通过名称注入bean
    @Autowired @Qualifier("Scheduler")
    private Scheduler scheduler;

    private static Logger log = LoggerFactory.getLogger(JobController.class);


    @PostMapping(value="/addjob")
    public void addjob(@RequestParam(value="jobClassName")String jobClassName,
            @RequestParam(value="jobGroupName")String jobGroupName,
            @RequestParam(value="cronExpression")String cronExpression) throws
Exception
    {
        addJob(jobClassName, jobGroupName, cronExpression);
    }

    public void addJob(String jobClassName, String jobGroupName, String
cronExpression)throws Exception{

        // 启动调度器
        scheduler.start();

        //构建job信息
        JobDetail jobDetail =
JobBuilder.newJob(getClass(jobClassName).getClass()).withIdentity(jobClassName,
jobGroupName).build();

        //表达式调度构建器(即任务执行的时间)
```

```
        CronScheduleBuilder scheduleBuilder =
CronScheduleBuilder.cronSchedule(cronExpression);

        //按新的cronExpression表达式构建一个新的trigger
        CronTrigger trigger = TriggerBuilder.newTrigger().withIdentity(jobClassName,
jobGroupName)
            .withSchedule(scheduleBuilder).build();

        try {
            scheduler.scheduleJob(jobDetail, trigger);

        } catch (SchedulerException e) {
            System.out.println("创建定时任务失败"+e);
            throw new Exception("创建定时任务失败");
        }
    }


    @PostMapping(value="/pausejob")
    public void pausejob(@RequestParam(value="jobClassName")String jobClassName,
@RequestParam(value="jobGroupName")String jobGroupName) throws Exception
    {
        jobPause(jobClassName, jobGroupName);
    }

    public void jobPause(String jobClassName, String jobGroupName) throws Exception
    {
        scheduler.pauseJob(JobKey.jobKey(jobClassName, jobGroupName));
    }


    @PostMapping(value="/resumejob")
    public void resumejob(@RequestParam(value="jobClassName")String jobClassName,
@RequestParam(value="jobGroupName")String jobGroupName) throws Exception
    {
        jobresume(jobClassName, jobGroupName);
    }

    public void jobresume(String jobClassName, String jobGroupName) throws Exception
    {
        scheduler.resumeJob(JobKey.jobKey(jobClassName, jobGroupName));
    }

    /**
     * 更新任务
     * @param jobClassName
     * @param jobGroupName
     * @param cronExpression
     * @throws Exception
     */

    @PostMapping(value="/rescheduleJob")
    public void rescheduleJob(@RequestParam(value="jobClassName")String jobClassName,
```

```
            @RequestParam(value="jobGroupName")String jobGroupName,
            @RequestParam(value="cronExpression")String cronExpression) throws
Exception
    {
        jobreschedule(jobClassName, jobGroupName, cronExpression);
    }


    public void jobreschedule(String jobClassName, String jobGroupName, String
cronExpression) throws Exception
    {
        try {
            TriggerKey triggerKey = TriggerKey.triggerKey(jobClassName, jobGroupName);
            // 表达式调度构建器
            CronScheduleBuilder scheduleBuilder =
CronScheduleBuilder.cronSchedule(cronExpression);

            CronTrigger trigger = (CronTrigger) scheduler.getTrigger(triggerKey);

            // 按新的cronExpression表达式重新构建trigger
            trigger =
trigger.getTriggerBuilder().withIdentity(triggerKey).withSchedule(scheduleBuilder).buil
d();

            // 按新的trigger重新设置job执行
            scheduler.rescheduleJob(triggerKey, trigger);
        } catch (SchedulerException e) {
            System.out.println("更新定时任务失败"+e);
            throw new Exception("更新定时任务失败");
        }
    }

    /**
     * 删除任务
     * @param jobClassName
     * @param jobGroupName
     * @throws Exception
     */
    @PostMapping(value="/deletejob")
    public void deletejob(@RequestParam(value="jobClassName")String jobClassName,
@RequestParam(value="jobGroupName")String jobGroupName) throws Exception
    {
        jobdelete(jobClassName, jobGroupName);
    }

    public void jobdelete(String jobClassName, String jobGroupName) throws Exception
    {
        scheduler.pauseTrigger(TriggerKey.triggerKey(jobClassName, jobGroupName));
        scheduler.unscheduleJob(TriggerKey.triggerKey(jobClassName, jobGroupName));
        scheduler.deleteJob(JobKey.jobKey(jobClassName, jobGroupName));
    }

    /**
```

```
     * 查询所有job
     * @param pageNum
     * @param pageSize
     * @return
     */
    @GetMapping(value="/queryjob")
    public Map<String, Object> queryjob(@RequestParam(value="pageNum")Integer pageNum,
@RequestParam(value="pageSize")Integer pageSize)
    {
        PageInfo<JobAndTrigger> jobAndTrigger =
iJobAndTriggerService.getJobAndTriggerDetails(pageNum, pageSize);
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("JobAndTrigger", jobAndTrigger);
        map.put("number", jobAndTrigger.getTotal());
        return map;
    }

    /**
     * 获取指定job class
     * @param classname
     * @return
     * @throws Exception
     */
    public static BaseJob getClass(String classname) throws Exception
    {
        Class<?> class1 = Class.forName(classname);
        return (BaseJob)class1.newInstance();
    }


}
```

2.7 操作页面 JobManager.html

> 此页面需要放入resource下的static目录下

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
    <title>QuartzDemo</title>
    <link rel="stylesheet" href="https://unpkg.com/element-ui@2.0.5/lib/theme-
chalk/index.css">
    <script src="https://unpkg.com/vue/dist/vue.js"></script>
    <script src="http://cdn.bootcss.com/vue-resource/1.3.4/vue-resource.js"></script>
    <script src="https://unpkg.com/element-ui@2.0.5/lib/index.js"></script>

    <style>
      #top {
          background:#20A0FF;
          padding:5px;
          overflow:hidden
```

```
      }
    </style>

</head>
<body>
    <div id="test">

        <div id="top">
                <el-button type="text" @click="search" style="color:white">查询</el-
button>
                <el-button type="text" @click="handleadd" style="color:white">添加</el-
button>
            </span>
        </div>

        <br/>

        <div style="margin-top:15px">

          <el-table
            ref="testTable"
            :data="tableData"
            style="width:100%"
            border
            >
            <el-table-column
              prop="job_NAME"
              label="任务名称"
              sortable
              show-overflow-tooltip>
            </el-table-column>

            <el-table-column
              prop="job_GROUP"
              label="任务所在组"
              sortable>
            </el-table-column>

            <el-table-column
              prop="job_CLASS_NAME"
              label="任务类名"
              sortable>
            </el-table-column>

            <el-table-column
              prop="trigger_NAME"
              label="触发器名称"
              sortable>
            </el-table-column>

            <el-table-column
              prop="trigger_GROUP"
              label="触发器所在组"
```

```
            sortable>
          </el-table-column>

          <el-table-column
            prop="cron_EXPRESSION"
            label="表达式"
            sortable>
          </el-table-column>

          <el-table-column
            prop="time_ZONE_ID"
            label="时区"
            sortable>
          </el-table-column>

          <el-table-column label="操作" width="300">
            <template scope="scope">
              <el-button
                size="small"
                type="warning"
                @click="handlePause(scope.$index, scope.row)">暂停</el-button>

              <el-button
                size="small"
                type="info"
                @click="handleResume(scope.$index, scope.row)">恢复</el-button>

              <el-button
                size="small"
                type="danger"
                @click="handleDelete(scope.$index, scope.row)">删除</el-button>

              <el-button
                size="small"
                type="success"
                @click="handleUpdate(scope.$index, scope.row)">修改</el-button>
            </template>
          </el-table-column>
        </el-table>

        <div align="center">
            <el-pagination
                @size-change="handleSizeChange"
                @current-change="handleCurrentChange"
                :current-page="currentPage"
                :page-sizes="[10, 20, 30, 40]"
                :page-size="pagesize"
                layout="total, sizes, prev, pager, next, jumper"
                :total="totalCount">
            </el-pagination>
        </div>
      </div>
```

```
        <el-dialog title="添加任务" :visible.sync="dialogFormVisible">
          <el-form :model="form">
            <el-form-item label="任务名称" label-width="120px" style="width:35%">
              <el-input v-model="form.jobName" auto-complete="off"></el-input>
            </el-form-item>
            <el-form-item label="任务分组" label-width="120px" style="width:35%">
              <el-input v-model="form.jobGroup" auto-complete="off"></el-input>
            </el-form-item>
            <el-form-item label="表达式" label-width="120px" style="width:35%">
              <el-input v-model="form.cronExpression" auto-complete="off"></el-input>
            </el-form-item>
          </el-form>
          <div slot="footer" class="dialog-footer">
            <el-button @click="dialogFormVisible = false">取 消</el-button>
            <el-button type="primary" @click="add">确 定</el-button>
          </div>
        </el-dialog>

        <el-dialog title="修改任务" :visible.sync="updateFormVisible">
          <el-form :model="updateform">
            <el-form-item label="表达式" label-width="120px" style="width:35%">
              <el-input v-model="updateform.cronExpression" auto-complete="off"></el-
input>
            </el-form-item>
          </el-form>
          <div slot="footer" class="dialog-footer">
            <el-button @click="updateFormVisible = false">取 消</el-button>
            <el-button type="primary" @click="update">确 定</el-button>
          </div>
        </el-dialog>

    </div>

    <footer align="center">
        <p>&copy; Quartz 任务管理</p>
    </footer>

    <script>
    var vue = new Vue({
            el:"#test",
            data: {
                //表格当前页数据
                tableData: [],

                //请求的URL
                url:'job/queryjob',

                //默认每页数据量
                pagesize: 10,

                //当前页码
                currentPage: 1,
```

```
                //查询的页码
                start: 1,

                //默认数据总数
                totalCount: 1000,

                //添加对话框默认可见性
                dialogFormVisible: false,

                //修改对话框默认可见性
                updateFormVisible: false,

                //提交的表单
                form: {
                    jobName: '',
                    jobGroup: '',
                    cronExpression: '',
                  },

                updateform: {
                    jobName: '',
                    jobGroup: '',
                    cronExpression: '',
                },
            },

            methods: {

                //从服务器读取数据
                loadData: function(pageNum, pageSize){
                    this.$http.get('job/queryjob?' + 'pageNum=' +  pageNum +
'&pageSize=' + pageSize).then(function(res){
                        console.log(res)
                        this.tableData = res.body.JobAndTrigger.list;
                        this.totalCount = res.body.number;
                    },function(){
                        console.log('failed');
                    });
                },

                //单行删除
                handleDelete: function(index, row) {
                    this.$http.post('job/deletejob',
{"jobClassName":row.job_NAME,"jobGroupName":row.job_GROUP},{emulateJSON:
true}).then(function(res){
                        this.loadData( this.currentPage, this.pagesize);
                    },function(){
                        console.log('failed');
                    });
                },

                //暂停任务
                handlePause: function(index, row){
```

```
                    this.$http.post('job/pausejob',
{"jobClassName":row.job_NAME,"jobGroupName":row.job_GROUP},{emulateJSON:
true}).then(function(res){
                        this.loadData( this.currentPage, this.pagesize);
                    },function(){
                        console.log('failed');
                    });
                },

                //恢复任务
                handleResume: function(index, row){
                    this.$http.post('job/resumejob',
{"jobClassName":row.job_NAME,"jobGroupName":row.job_GROUP},{emulateJSON:
true}).then(function(res){
                        this.loadData( this.currentPage, this.pagesize);
                    },function(){
                        console.log('failed');
                    });
                },

                //搜索
                search: function(){
                    this.loadData(this.currentPage, this.pagesize);
                },

                //弹出对话框
                handleadd: function(){
                    this.dialogFormVisible = true;
                },

                //添加
                add: function(){
                    this.$http.post('job/addjob',
{"jobClassName":this.form.jobName,"jobGroupName":this.form.jobGroup,"cronExpression":th
is.form.cronExpression},{emulateJSON: true}).then(function(res){
                        this.loadData(this.currentPage, this.pagesize);
                        this.dialogFormVisible = false;
                    },function(){
                        console.log('failed');
                    });
                },

                //更新
                handleUpdate: function(index, row){
                    console.log(row)
                    this.updateFormVisible = true;
                    this.updateform.jobName = row.job_CLASS_NAME;
                    this.updateform.jobGroup = row.job_GROUP;
                },

                //更新任务
                update: function(){
                    this.$http.post
```

```
                    ('job/reschedulejob',
                            {"jobClassName":this.updateform.jobName,
                             "jobGroupName":this.updateform.jobGroup,
                             "cronExpression":this.updateform.cronExpression
                             },{emulateJSON: true}
                    ).then(function(res){
                        this.loadData(this.currentPage, this.pagesize);
                        this.updateFormVisible = false;
                    },function(){
                        console.log('failed');
                    });

                },

                //每页显示数据量变更
                handleSizeChange: function(val) {
                    this.pagesize = val;
                    this.loadData(this.currentPage, this.pagesize);
                },

                //页码变更
                handleCurrentChange: function(val) {
                    this.currentPage = val;
                    this.loadData(this.currentPage, this.pagesize);
                },

            },


        });

        //载入数据
        vue.loadData(vue.currentPage, vue.pagesize);
    </script>

</body>
</html>
```

2.8 配置类

```
package com.qianfeng.quartz.spring;

import org.quartz.Scheduler;
import org.quartz.ee.servlet.QuartzInitializerListener;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.config.PropertiesFactoryBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.scheduling.quartz.SchedulerFactoryBean;
```

```
import java.io.IOException;
import java.util.Properties;

@Configuration
public class SchedulerConfig {
    /**
     * 调度器工厂对象,并设置quartz属性
     * @return
     * @throws IOException
     */
    @Bean(name="SchedulerFactory")
    public SchedulerFactoryBean schedulerFactoryBean(@Qualifier("quartzProperties")
Properties quartzProperties) throws IOException {
        SchedulerFactoryBean factory = new SchedulerFactoryBean();
        factory.setQuartzProperties(quartzProperties);
        return factory;
    }

    /**
     * 加载quartz的配置
     * @return
     * @throws IOException
     */
    @Bean(name = "quartzProperties")
    public Properties quartzProperties() throws IOException {
        PropertiesFactoryBean propertiesFactoryBean = new PropertiesFactoryBean();
        propertiesFactoryBean.setLocation(new ClassPathResource("/quartz.properties"));
        //在quartz.properties中的属性被读取并注入后再初始化对象
        propertiesFactoryBean.afterPropertiesSet();
        return propertiesFactoryBean.getObject();
    }

    /*
     * quartz初始化监听器
     */
    @Bean
    public QuartzInitializerListener executorListener() {
       return new QuartzInitializerListener();
    }

    /*
     * 通过SchedulerFactoryBean获取Scheduler 调度器的实例
     */
    @Bean(name="Scheduler")
    public Scheduler scheduler(SchedulerFactoryBean schedulerFactoryBean) throws
IOException {
        return schedulerFactoryBean.getScheduler();
    }

}
```

2.9启动类

```
@MapperScan("com.qianfeng.quartz.spring.dao")
@SpringBootApplication(scanBasePackages = {"com.qianfeng.quartz.spring"})
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```