QUARTZ SCHEDULER

CONFIGURATION OF QUARTZ SCHEDULER WITH DATABASE IN GRAILS

JULY 15, 2015 | ADMIN | LEAVE A COMMENT

In my steps below, I'm using Grails 2.2.0 and Quartz 2.1.1. I'm also connecting to a local mysql database.

1. Add the "quartz-all-2.1.1.jar" and "c3p0-0.9.1.1.jar" (in the lib folder of your Quartz download) to your lib directory.

2. Add your Quartz.properties file to your "conf" directory (or somewhere else on your classpath). Here's the Quartz.properties file I used (you'll need to change the username and password)

```
#============================================================================

# Configure Main Scheduler Properties

#============================================================================

quartz.scheduler.instanceName = MyClusteredScheduler

quartz.scheduler.instanceId = AUTO

#============================================================================

# Configure ThreadPool
```

```
#============================================================================

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

quartz.threadPool.threadCount = 25

quartz.threadPool.threadPriority = 5

#============================================================================

# Configure JobStore

#============================================================================

quartz.jobStore.misfireThreshold = 60000

quartz.jobStore.class = org.quartz.impl.jdbcjobstore.JobStoreTX

quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate

quartz.jobStore.useProperties = false

quartz.jobStore.dataSource = myDS

quartz.jobStore.tablePrefix = QRTZ_

quartz.jobStore.isClustered = true

quartz.jobStore.clusterCheckinInterval = 20000

#============================================================================

# Configure Datasources

#============================================================================
quartz.dataSource.myDS.driver = com.mysql.jdbc.Driver

quartz.dataSource.myDS.URL = jdbc:mysql://localhost/schedulerproject?
useUnicode=yes&characterEncoding=UTF-8

quartz.dataSource.myDS.user = root

quartz.dataSource.myDS.password = root

quartz.dataSource.myDS.maxConnections = 5

quartz.dataSource.myDS.validationQuery=select 0 from dual.
```

3. In your Config.groovy file, add or modify your “grails.config.locations” property. Here’s what I added

```
grails.config.locations = [

"classpath:conf/Quartz.properties"

]
```

4. I added the JobScheduler.java and HelloJob.java classes to my src/java directory. These could be groovy or whatever, but I just stole the example from Quartz to get it working correctly.

JobScheduler.java

```
package sample.quartz.scheduler;

import static org.quartz.JobBuilder.newJob;

import static org.quartz.TriggerBuilder.newTrigger;

import static org.quartz.CronScheduleBuilder.*;

import org.apache.log4j.Logger;

import org.quartz.JobDetail;
import org.quartz.Scheduler;

import org.quartz.SchedulerException;

import org.quartz.Trigger;

import org.quartz.impl.StdSchedulerFactory;



public class JobScheduler {

private static Logger log = Logger.getLogger(JobScheduler.class);

private static JobScheduler JOB_SCHEDULER = new JobScheduler();

private Scheduler scheduler = null;

public JobScheduler() {

}

public static JobScheduler getInstance() {

return JOB_SCHEDULER;

}

public void startup() {

try {

// and start it off

scheduler = StdSchedulerFactory.getDefaultScheduler();

System.out.println("NAME: " + cheduler.getSchedulerName());

scheduler.start();
```

```
// define the job and tie it to our HelloJob class

JobDetail job = newJob(HelloJob.class)

.withIdentity("job1″, "group1″)

.build();

// Trigger a job that repeats every 20 seconds

Trigger trigger = newTrigger()

.withIdentity("trigger1″, "group1″)

.withSchedule(cronSchedule("0/20 * * * * ?"))

.build();

System.out.println("Starting Jobs");

// Tell quartz to schedule the job using our trigger

scheduler.scheduleJob(job, trigger);

scheduler.start();

} catch (SchedulerException se) {

se.printStackTrace();

}

}

public void shutdown() {

try {

scheduler.shutdown();
```

```
} catch (SchedulerException se) {

se.printStackTrace();

}

}

}
```

5. I added the JobScheduler and HelloJob java classes to my src/java directory.

HelloJob.java

```
package sample.quartz.scheduler;

import java.util.Date;

import org.apache.log4j.Logger;

import org.quartz.Job;

import org.quartz.JobExecutionContext;

import org.quartz.JobExecutionException;



public class HelloJob implements Job {

private static Logger log = Logger.getLogger(HelloJob.class);

public HelloJob() {

}

public void execute(JobExecutionContext context)

throws JobExecutionException {
```

```
System.out.println("Hello!  HelloJob is executing. " + new Date());

}

}
```

6. In your BootStrap.groovy file, add…

```
import sample.quartz.scheduler.JobScheduler

class BootStrap {

def init = { servletContext ->

JobScheduler.getInstance().startup()

}

def destroy = {

JobScheduler.getInstance().shutdown()

}

}
```

7. Create database. Here, my database name is schedulerproject.

and create table like as given below .

```
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)

);
```

```
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)

);



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)

);

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),

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)

);

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)

);

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),

FOREIGN KEY (SCHED_NAME,TRIGGER_NAME,TRIGGER_GROUP)

REFERENCES QRTZ_TRIGGERS(SCHED_NAME,TRIGGER_NAME,TRIGGER_GROUP)

);

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_PAUSED_TRIGGER_GRPS

(

SCHED_NAME VARCHAR(120) NOT NULL,

TRIGGER_GROUP  VARCHAR(200) NOT NULL,

PRIMARY KEY (SCHED_NAME,TRIGGER_GROUP)

);

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,

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)

);

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_LOCKS

(

SCHED_NAME VARCHAR(120) NOT NULL,

LOCK_NAME  VARCHAR(40) NOT NULL,

PRIMARY KEY (SCHED_NAME,LOCK_NAME)

);
```

commit;

8. Finally run on different port or server.

grails –Dserver.port=8080 run-app

and

grails –Dserver.port=8090 run-app

clean when we get proxy related classCastException.

ProsperaSoft offers Grails development solutions. You can email at info@prosperasoft.com *to get in touch with ProsperaSoft Grails experts and consultants.*

GRAILS MYSQL QUARTZ SCHEDULER