Spring 4 + Quartz Scheduler Integration Example

Created on: August 25, 2014 | **Last updated on:** September 30, 2017
websystiqueadmin

In this post we will see how to schedule Jobs using `Quartz Scheduler` with Spring. Spring provides couple of classes that simplify the usage of Quartz within Spring-based applications. Let's get going.

Other interesting posts you may like

- Spring Boot+AngularJS+Spring Data+Hibernate+MySQL CRUD App
- Spring Boot REST API Tutorial
- Spring Boot WAR deployment example
- Spring Boot Introduction + Hello World Example
- Secure Spring REST API using OAuth2
- AngularJS+Spring Security using Basic Authentication
- Secure Spring REST API using Basic Authentication
- Spring 4 Caching Annotations Tutorial
- Spring 4 Cache Tutorial with EhCache
- Spring 4 MVC+JPA2+Hibernate Many-to-many Example
- Spring 4 Email With Attachment Tutorial
- Spring 4 Email Template Library Example
- Spring 4 Email Integration Tutorial

Recent Posts

Spring Boot + AngularJS + Spring Data + JPA CRUD App Example
Spring Boot Rest API Example
Spring Boot WAR deployment example
Spring Boot Introduction + hello world example
Secure Spring REST API using OAuth2

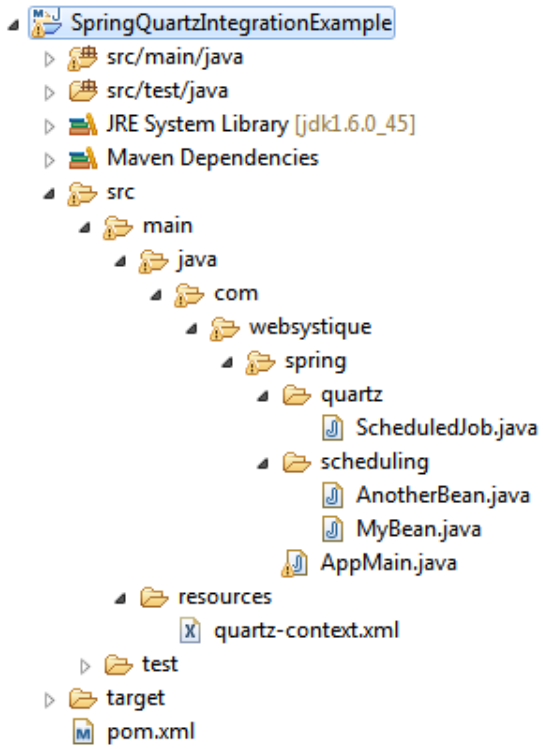
- Spring MVC 4+AngularJS Example
- Spring MVC 4+JMS+ActiveMQ Integration Example
- Spring 4+JMS+ActiveMQ @JmsLister @EnableJms Example
- Spring 4+JMS+ActiveMQ Integration Example
- Spring MVC 4+Hibernate 4 Many-to-many JSP Example
- Spring MVC 4+Hibernate 4+MySQL+Maven integration example using annotations
- Spring MVC4 FileUpload-Download Hibernate+MySQL Example
- TestNG Mockito Integration Example Stubbing Void Methods
- Maven surefire plugin and TestNG Example
- Spring MVC 4 Form Validation and Resource Handling

Following technologies being used:

- Spring 4.0.6.RELEASE
- Quartz 2.2.1
- Maven 3
- JDK 1.6
- Eclipse JUNO Service Release 2

Project directory structure:

Following will be the final project directory structure for this example:



Let's now add the content mentioned in above structure explaining each in detail.

Step 1: Provide Dependencies in Maven pom.xml

```
<project xmlns="<a class="vglnk" href="http://maven.apache.org/POM
  xsi:schemaLocation="<a class="vglnk" href="http://maven.apache.o
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.websystique.spring</groupId>
  <artifactId>SpringQuartzIntegrationExample</artifactId>
  <version>1.0.0</version>
  <packaging>jar</packaging>
  <name>SpringQuartzIntegrationExample</name>

    <properties>
        <springframework.version>4.0.6.RELEASE</springframework.ve
        <quartz.version>2.2.1</quartz.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${springframework.version}</version>
        </dependency>
        <!-- Transaction dependency is required with Quartz integr
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${springframework.version}</version>
        </dependency>
```

```
        <!-- Quartz framework -->
        <dependency>
            <groupId>org.quartz-scheduler</groupId>
            <artifactId>quartz</artifactId>
            <version>${quartz.version}</version>
        </dependency>
    </dependencies>
    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <source>1.6</source>
                        <target>1.6</target>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>

</project>
```

Step 2: Configure Jobs in Quartz Scheduler

There are 2 ways to configure a Job in Spring using Quartz

A : Using MethodInvokingJobDetailFactoryBean

Really handy when you just need to invoke a method on a specific bean. This is the simplest among two.

```
<!-- For times when you just need to invoke a method on a specific
<bean id="simpleJobDetail" class="org.springframework.scheduling.q
    <property name="targetObject" ref="myBean" />
    <property name="targetMethod" value="printMessage" />
</bean>
```

Above job configuration simply invokes printMessage method of bean myJobBean which is simple POJO

B : Using JobDetailFactoryBean

When you need more advanced setup, need to pass data to job, being more flexible.

```
<!-- For times when you need more complex processing, passing data
<bean name="complexJobDetail"    class="org.springframework.schedu
    <property name="jobClass" value="com.websystique.spring.quartz
    <property name="jobDataMap">
        <map>
            <entry key="anotherBean" value-ref="anotherBean" />
```

```
        </map>
    </property>
    <property name="durability" value="true" />
</bean>
```


`jobClass` refers to a class which extends `QuartzJobBean`, an implementation of Quartz job interface. On invocation of this job, it's `executeInternal` method gets called.
`jobDataMap` provides opportunity to pass some data to underlying job bean. In this case, we are passing a bean 'anotherBean' which will be used by ScheduledJob.

Below is the referred jobclass (`ScheduledJob`) implementation.

`com.websystique.spring.quartz.ScheduledJob`


```
package com.websystique.spring.quartz;

import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;
import org.springframework.scheduling.quartz.QuartzJobBean;

import com.websystique.spring.scheduling.AnotherBean;

public class ScheduledJob extends QuartzJobBean{


    private AnotherBean anotherBean;


    @Override
    protected void executeInternal(JobExecutionContext arg0)
            throws JobExecutionException {
        anotherBean.printAnotherMessage();
```

```
    }

    public void setAnotherBean(AnotherBean anotherBean) {
        this.anotherBean = anotherBean;
    }
}
```

Step 3: Configure Triggers to be used in Quartz Scheduler

Trigger defines the time when scheduler will run your scheduled job. There are two possible trigger type:

A: Simple Trigger , using SimpleTriggerFactoryBean
You can specify start time, delay between triggers and repeatInterval(frequency) to run the job.

```
<!-- Run the job every 2 seconds with initial delay of 1 second --
<bean id="simpleTrigger"  class="org.springframework.scheduling.qu
    <property name="jobDetail" ref="simpleJobDetail" />
    <property name="startDelay" value="1000" />
    <property name="repeatInterval" value="2000" />
</bean>
```

B: Cron Trigger , using CronTriggerFactoryBean

It's more flexible and allows you to choose scheduled job at specific instance (time, day, date,..) and frequency in future.

```
<!-- Run the job every 5 seconds only on Weekends -->
<bean id="cronTrigger"  class="org.springframework.scheduling.quar
    <property name="jobDetail" ref="complexJobDetail" />
    <property name="cronExpression" value="0/5 * * ? * SAT-SUN" />
</bean>
```

Step 4: Configure SchedulerFactoryBean that creates and configures Quartz Scheduler

`SchedulerFactoryBean` glues together `jobDetails` and `triggers` to Configure `Quartz Scheduler`

```
<!-- Scheduler factory bean to glue together jobDetails and trigge
<bean  class="org.springframework.scheduling.quartz.SchedulerFacto
    <property name="jobDetails">
        <list>
            <ref bean="simpleJobDetail" />
            <ref bean="complexJobDetail" />
        </list>
    </property>

    <property name="triggers">
        <list>
            <ref bean="simpleTrigger" />
```

```
            <ref bean="cronTrigger" />
        </list>
    </property>
</bean>
```

Below shown is complete context file for our example

`src/main/resources/quartz-context.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="<a class="vglnk" href="http://www.springframework.or
    xmlns:xsi="<a class="vglnk" href="http://www.w3.org/2001/XMLSc
    xsi:schemaLocation="<a class="vglnk" href="http://www.springfr
                            <a class="vglnk" href="http://www.spri

    <context:component-scan base-package="com.websystique.spring"


    <!-- For times when you just need to invoke a method on a spec
    <bean id="simpleJobDetail" class="org.springframework.scheduli
        <property name="targetObject" ref="myBean" />
        <property name="targetMethod" value="printMessage" />
    </bean>


    <!-- For times when you need more complex processing, passing
    <bean name="complexJobDetail"    class="org.springframework.sc
        <property name="jobClass" value="com.websystique.spring.qu
        <property name="jobDataMap">
            <map>
                <entry key="anotherBean" value-ref="anotherBean" /
            </map>
        </property>
        <property name="durability" value="true" />
    </bean>


    <!-- Run the job every 2 seconds with initial delay of 1 secon
    <bean id="simpleTrigger"  class="org.springframework.schedulin
        <property name="jobDetail" ref="simpleJobDetail" />
        <property name="startDelay" value="1000" />
        <property name="repeatInterval" value="2000" />
    </bean>


    <!-- Run the job every 5 seconds only on Weekends -->
    <bean id="cronTrigger"  class="org.springframework.scheduling.
        <property name="jobDetail" ref="complexJobDetail" />
        <property name="cronExpression" value="0/5 * * ? * SAT-SUN
    </bean>


    <!-- Scheduler factory bean to glue together jobDetails and tr
    <bean  class="org.springframework.scheduling.quartz.SchedulerF
        <property name="jobDetails">
            <list>
                <ref bean="simpleJobDetail" />
                <ref bean="complexJobDetail" />
            </list>
        </property>

        <property name="triggers">
```

```
            <list>
                <ref bean="simpleTrigger" />
                <ref bean="cronTrigger" />
            </list>
        </property>
    </bean>

</beans>
```

Step 5: Create simple POJO's Task Beans used in this example

`com.websystique.spring.scheduling.MyBean`

```
package com.websystique.spring.scheduling;

import org.springframework.stereotype.Component;

@Component("myBean")
public class MyBean {

    public void printMessage() {
        System.out.println("I am called by MethodInvokingJobDetail
    }

}
```

`com.websystique.spring.scheduling.AnotherBean`

```
package com.websystique.spring.scheduling;

import org.springframework.stereotype.Component;

@Component("anotherBean")
public class AnotherBean {

    public void printAnotherMessage(){
        System.out.println("I am called by Quartz jobBean using Cr
    }

}
```

Step 6: Create Main, and Run the application

```
package com.websystique.spring;

import org.springframework.context.support.AbstractApplicationCont
import org.springframework.context.support.ClassPathXmlApplication

public class AppMain {
    public static void main(String args[]){
        AbstractApplicationContext context = new ClassPathXmlAppli
    }

}
```

Run it as Java application, you will see following output.

```
INFO: Starting Quartz Scheduler now
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
I am called by Quartz jobBean using CronTriggerFactoryBean
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
I am called by Quartz jobBean using CronTriggerFactoryBean
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
I am called by Quartz jobBean using CronTriggerFactoryBean
I am called by MethodInvokingJobDetailFactoryBean using SimpleTrig
```

You can see that SimpleTrigger invoked job is called every two seconds while the one from CronTrigger is called every five seconds.

That's it.

Download Source Code

Download Now!

References

- Spring framework
- Spring Task Scheduling
- Quartz Scheduler

websystiqueadmin

If you like tutorials on this site, why not take a step further and connect me on Facebook , Google Plus & Twitter as well? I would love to hear your thoughts on these articles, it

will help improve further our learning process.

Related Posts:

1. **Spring Job Scheduling using TaskScheduler (XML Config)**
2. **Spring Job Scheduling with @Scheduled & @EnableScheduling Annotations**
3. **Spring Batch & Quartz Scheduler Example (Tasklet usage)**
4. **Spring 4 Email Integration Tutorial**

spring. permalink.

← Spring Job Scheduling with @Scheduled & @EnableScheduling Annotations

Maven Installation and Setup (Windows + Unix) →

Join the discussion…

Rahul Joshi • 4 years ago

I tried to run this code, but the cron scheduler print statement is never being displayed to me somehow. i checked everything seems correct not sure whats going wrong.

1 • Reply • Share ›

reagul ➚ Rahul Joshi • 4 years ago

run this on a SAT or SUN and it will print them.

• Reply • Share ›

chad_baker • 2 years ago

For the SchedulerFactoryBean config, why are you including the "jobDetails" along with the "triggers" elements? Seems that the "triggers" elements are enough since those configs reference the jobs. Thanks!

• Reply • Share ›

websystique Mod ➚ chad_baker • 2 years ago

Hi Baker, i see what you mean, agree that it should work even without jobDetails. Good remark, thanks.

1 • Reply • Share ›

chad_baker ➚ websystique • 2 years ago

Sounds like having both does not cause an issue for you, though. We're trying to debug an issue where the job runs continuously and not according the the schedule/trigger. Our configuration (that I've inherited) also includes both "jobDetails" and "triggers" sections, which appears to be the norm for some implementations even though the documentation on the Spring website only includes "triggers". Wasn't sure if you had experience such that both were necessary. Thanks again ...

• Reply • Share ›

Zhi Du • 3 years ago

I encountered a weird issue. My QuartzJobBean implementation class cannot get [anotherBean] auto injected with the data configured in jobDataMap. It's always throw a NullPointerException when invoking method in [anotherBean]. So I use JobExecutionContext parameter passed into executeInternal method to get JobDetail, then get [anotherBean] from the jobDataMap.

```
protected void executeInternal(JobExecutionContext
paramJobExecutionContext) throws JobExecutionException {
JobDataMap jobDataMap =
paramJobExecutionContext.getJobDetail().getJobDataMap();
anotherBean = (OrderAssignmentJobDataBean)
jobDataMap.get("anotherBean");
```

...
}
I've already tried @Component annotation for AnotherBean class and xml based configure for beanBean injection. I ma using spring 4.2.5, quartz 2.2.1.

Reply • Share ›

Vignesh S Kannan • 3 years ago

Thanks for this post. Was very helpful. Thank You!!

Reply • Share ›

PA • 3 years ago

Could you please developed a Spring MVC code which will query to database in every 5 mins and fetches a records ??

Reply • Share ›

websystique Mod ➚ PA • 3 years ago

Hi Prateek,
Even if it can be done, this kind of [non-interactive] functionality is better suited for spring batch. Have a look at this post.

Reply • Share ›

Diwakar Choudhary • 4 years ago

Thank you so much for wonderful and concise code.

Reply • Share ›

Chandan Gopalkrishna • 4 years ago

Thank you very much for wonderful tutorial. Can you please provide a post using only annotations so that we can get rid of clumsy xml code.