Scheduling Service

Summary

The Scheduling Service supports the tasks that occur regularly or repeatedly in the application server and provides the function similar to Cron command of UNIX.
The execution environment scheduling service uses the Quartz scheduler as open source software. This chapter will examine the basic concept of Quartz Scheduler and then how to integrate and use the Quartz scheduler and Spring that provides the IoCservice.

Description

Quartz Scheduler

Main elements related to the Quartz scheduler execution include Scheduler, Job, JobDetail and Trigger.

- **Scheduler**is the core entity that manages Quartz run environment.
- **Job**is the interface that defines the tasks to be performed by user. It can be scheduled using Trigger entity.
- **JobDetail**is the entity that defines detailed information for the job such as task name and task group.
- **Trigger**is the entity that defines the run schedule of defined Job entity and informs the Scheduler entity of Job execution time.

The Quartz scheduler provides flexibility by separating triggers that defines the execution schedule and Job defining the execution tasks. If Job and run schedule are defined, only execution schedule can be changed while leaving the job as it is. In addition, several execution schedule can be defined in one job.

Example of Using Quartz Scheduler

Let's see a simple example illuminating the Quartz scheduler. The following refers to the Quartz manual and shows how to use Quartz and how to set a user's job.

Personalized Job

User can implement org.quartz.Job interface to create Job entity and if serious error occurs, JobExecutionException exception may be thrown. The Job interface defines execute() in the single method.

```
public class DumbJob implements Job {
public void execute(JobExecutionContext context)
throwsJobExecutionException
    {
System.out.println("DumbJob is executing.");
    }
  }
```

- DumbJob implements execute() methods of job interface.
- execute() method simply outputs the message that Job is executed.

Codes using Quartz

```
JobDetailjobDetail =
newJobDetail("myJob",// Job name
sched.DEFAULT_GROUP,   // Job group name(if it is 'null' value, it is defined as DEFAULT_GROUP)
DumbJob.class);          // Job class to execute

  Trigger trigger = TriggerUtils.makeDailyTrigger(8, 30);   // Execute at 08:30 everyday
trigger.setStartTime(new Date()); // Start immediately
trigger.setName("myTrigger");
```

sched.scheduleJob(jobDetail, trigger);

- First, define JobDetailclass for job setting.
- Create the Trigger that is executed at 8:30 everyday using TriggerUtils.
- Lastly, register JobDetail and Trigger in Scheduler.

Spring and Quartz Integration

Spring provides an integrated class for scheduling support. Spring 2.5 supports the Quartz scheduler that is an open source software and a timer included since JDK 1.3 version. Here, let's examine how to integrate and use the Quartz scheduler and Spring.
For integration with Quartz scheduler, Spring supports Quart Scheduler,JobDetail and Trigger to bean within Spring context. Then, let's examine how to create Quartz job and how to schedule and start the job focusing on examples.

Create Job

Spring provides the following 2 methods to create jobs.

- The method of creating Job class inheriting QuartzJobBean, using JobDetailBean
- The method of calling the method of Bean object directly using MethodInvokingJobDetailFactoryBean

Crate Job using JobDetailBean

JobDetail is the object that contains information required for execution of tasks. Spring provides JobDetailBean to crate JobDetail bean. For example,

JobDetailBean Source Code

```
packageegovframework.rte.fdl.scheduling.sample;

public class SayHelloJob extends QuartzJobBean {

	privateString name;

	public void setName (String name) {
		this.name = name;
	}

	@Override
	protected void executeInternal (JobExecutionContextctx) throws JobExecutionException {
		System.out.println("Hello, " + name);
	}
}
```

- SayHelloJobclass overrides the executeInternal(..) function of QuartzJobBean for job creation.

JobDetailBean Setting

```
<bean id="jobDetailBean"
	class="org.springframework.scheduling.quartz.JobDetailBean">
	<property name="jobClass" value="egovframework.rte.fdl.scheduling.sample.SayHelloJob"
/>
	<property name="jobDataAsMap">
		<map>
			<entry key="name" value="JobDetail"/>
		</map>
	</property>
</bean>
```

- This method delivers the property information required for a job setting to JobDetail entity using jobDataAsMapentity.

Creation of Jobs using MethodInvokingJobDetailFactoryBean

Source Code

```
packageegovframework.rte.fdl.scheduling.sample;

public class SayHelloService {

        privateString name;

        public void setName (String name) {
                this.name = name;
        }

        public void sayHello () {
                System.out.println("Hello, " + this.name);
        }
}
```

- Define Bean class to perform jobs.

Setting

```
<bean id="sayHelloService" class="egovframework.rte.fdl.scheduling.sample.SayHelloService">
        <property name="name" value="FactoryBean"/>
</bean>

<bean id="jobDetailFactoryBean"
        class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
        <property name="targetObject" ref="sayHelloService" />
        <property name="targetMethod" value="sayHello" />
        <property name="concurrent" value="false" />
</bean>
```

- This method defines MethodInvokingJobDetailFactoryBean to create the job for directly calling the method of defined Bean object.

Job Scheduling

Trigger type mainly used at Spring includes SimpleTriggerBean and CronTriggerBean. SimpleTriggeris used for simple scheduling like specific time, number of repetition and waiting time. CronTrigger is similar to Cron command of Unix and is used for complex scheduling. CronTriggercan be set to perform job on specific time, day of the month or month like using calendar. Next, let's examine how to schedule jobs created earlier usingSimpleTriggerBean andCronTriggerBean.

Setting using SimpleTriggerBean

```
<bean id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean">
        <property name="jobDetail" ref="jobDetailBean" />
<!—Start immediately -->
        <property name="startDelay" value="0" />
        <!—Execute in every 10 seconds -->
        <property name="repeatInterval" value="10000" />
</bean>
```

- Register the job created earlier using JobDetailBeanin the Trigger for scheduling. SimpleTriggerBean is set to start immediately and to execute every 10 seconds.

Setting using CronTriggerBean

```
<bean id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">
        <property name="jobDetail" ref="jobDetailFactoryBean" />
        <!—Execute in every 10 seconds-->
        <property name="cronExpression" value="*/10 * * * * ?" />
</bean>
```

- Register the job created earlier using MethodInvokingJobDetailFactoryBean in the Trigger for scheduling. CronTriggerBean was set to execute every 10 seconds. See Quartz Cron Expressionfor details on Crone expression

Start Job

For starting of scheduled job, Spring provides SchedulerFactoryBean.

Setting

```
<bean id="scheduler" class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
        <property name="triggers">
                <list>
                        <ref bean="simpleTrigger" />
                        <ref bean="cronTrigger" />
                </list>
        </property>
</bean>
```

- Start each Trigger's tasks of SimpleTriggerBeanandCronTriggerBeanbasis using SchedulerFactoryBean.

Reference

- Quartz Manual
- Spring Scheduling Manual
- Quartz API
- Spring API
- Quartz Cron Expression