# A qui va dirigit

Aquest how-to va dirigit als perfils tècnics (desenvolupadors i arquitectes) que estiguin utilitzant el Quartz com a planificador de tasques.

# Versió de Canigó

Els passos descrits en aquest document apliquen a la versió del framework Canigó 3.1.x. En aquest howto el front-end està realitzat amb Primefaces, partint de l’aplicació inicial amb JSF que genera el plugin de Canigó.

# Introducció

En aquest how-to expliquem com visualitzar les tasques planificades i posar-les en marxa manualment.

# Configurar el Planificador de tasques

Al web de Canigó està la documentació del planificador de tasques per a Canigó:

<http://canigo.ctti.gencat.cat/canigo-documentacio-versions-3x-altres/modul-planificador/>

Podeu seguir les instruccions del web alhora d’importar les llibreries i configurar el Quartz.

Per aquest howto hem creat dues tasques amb els seus triggers corresponents:

**src\main\resources\spring\quartz-config.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.1.xsd"*>

<bean id=*"SchedulerService"* class=*"cat.gencat.howtoquartz.service.SchedulerService"* />

<bean id=*"SchedulerService2"* class=*"cat.gencat.howtoquartz.service.Scheduler2Service"* />

<bean id=*"FirstTaskJobDetail"* class=*"org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"*>

<property name=*"targetObject"* ref=*"SchedulerService"* />

<property name=*"targetMethod"* value=*"testScheduler"* />

<property name=*"group"* value=*"grup1"*/>

</bean>

<bean id=*"SecondTaskJobDetail"* class=*"org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"*>

<property name=*"targetObject"* ref=*"SchedulerService2"* />

<property name=*"targetMethod"* value=*"testScheduler"* />

<property name=*"group"* value=*"grup2"*/>

</bean>

<bean id=*"FirstSimpleTrigger"* class=*"org.springframework.scheduling.quartz.SimpleTriggerFactoryBean"*>

<property name=*"jobDetail"* ref=*"FirstTaskJobDetail"* />

<property name=*"repeatInterval"* value=*"50000"* />

<property name=*"startDelay"* value=*"1000"* />

</bean>

<bean id=*"SecondSimpleTrigger"* class=*"org.springframework.scheduling.quartz.SimpleTriggerFactoryBean"*>

<property name=*"jobDetail"* ref=*"SecondTaskJobDetail"* />

<property name=*"repeatInterval"* value=*"30000"* />

<property name=*"startDelay"* value=*"1000"* />

</bean>

<bean id=*"schedulerFactory"* class=*"org.springframework.scheduling.quartz.SchedulerFactoryBean"*>

<property name=*"jobDetails"*>

<list>

<ref bean=*"FirstTaskJobDetail"* />

<ref bean=*"SecondTaskJobDetail"* />

</list>

</property>

<property name=*"triggers"*>

<list>

<ref bean=*"FirstSimpleTrigger"* />

<ref bean=*"SecondSimpleTrigger"* />

</list>

</property>

</bean>

</beans>

Les tasques que hem creat només escriuen amb System.out per consola que s’han executat quan son cridades:

**src\main\java\cat\gencat\howtoquartz\service\SchedulerService.java**

**package** cat.gencat.howtoquartz.service;

**import** java.util.Date;

**public** **class** SchedulerService {

**public** **void** testScheduler(){

System.*out*.println((**new** Date()) + " Executat metode de test de scheduler");

}

}

**src\main\java\cat\gencat\howtoquartz\service\Scheduler2Service.java**

**package** cat.gencat.howtoquartz.service;

**import** java.util.Date;

**public** **class** Scheduler2Service {

**public** **void** testScheduler(){

System.*out*.println((**new** Date()) + " Executat metode de test de scheduler 2");

}

}

Amb aquesta configuració si arranquem l’aplicació cada 50 segons s’executa la primera tasca i cada 30 segons la segona tasca.

# Visualitzar i llençar les tasques planificades

Creem un bean, SchedulerBean on injectarem el nostre SchedulerFactoryBean i d’aquí obtindrem les dades de les tasques i la possibilitat de llençar-les manualment

**src\main\java\cat\gencat\howtoquartz\bean\SchedulerBean.java**

@Component("SchedulerBean")

**public** **class** SchedulerBean {

@Autowired

**private** SchedulerFactoryBean schedulerFactory;

**private** Scheduler scheduler;

**public** List<QuartzJob> getQuartzJobList() **throws** SchedulerException{

List<QuartzJob> quartzJobList = **new** ArrayList<QuartzJob>();

scheduler = schedulerFactory.getScheduler();

**for** (String groupName : scheduler.getJobGroupNames()) {

**for** (JobKey jobKey : scheduler.getJobKeys(GroupMatcher.*jobGroupEquals*(groupName))) {

String jobName = jobKey.getName();

String jobGroup = jobKey.getGroup();

List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey);

Date nextFireTime = triggers.get(0).getNextFireTime();

quartzJobList.add(**new** QuartzJob(jobName, jobGroup, nextFireTime));

}

}

**return** quartzJobList;

}

**public** **void** fireNow(String jobName, String jobGroup) **throws** SchedulerException {

JobKey jobKey = **new** JobKey(jobName, jobGroup);

scheduler.triggerJob(jobKey);

}

}

En aquest bean obtenim les dades de les tasques iterant el scheduler i retornem un objecte QuartzJob per a mostrar-los en una llista.

El mètode fireNow llença manualment el job indicat.

**src\main\java\cat\gencat\howtoquartz\utils\QuartzJob.java**

**public** **class** QuartzJob {

String jobName;

String jobGroup;

Date nextFireTime;

**public** QuartzJob(String jobName, String jobGroup, Date nextFireTime) {

**this**.jobName = jobName;

**this**.jobGroup = jobGroup;

**this**.nextFireTime = nextFireTime;

}

**public** String getJobName() {

**return** jobName;

}

**public** **void** setJobName(String jobName) {

**this**.jobName = jobName;

}

**public** String getJobGroup() {

**return** jobGroup;

}

**public** **void** setJobGroup(String jobGroup) {

**this**.jobGroup = jobGroup;

}

**public** Date getNextFireTime() {

**return** nextFireTime;

}

**public** **void** setNextFireTime(Date nextFireTime) {

**this**.nextFireTime = nextFireTime;

}

}

Per mostrar les tasques planificades i permetre a l’usuari interactuar amb elles creem la vista scheduler.xhtml

...

<ui:define name=*"body"*>

<h1>Tasques Planificades</h1>

<h:form>

<h:dataTable value=*"#{SchedulerBean.quartzJobList}"* var=*"quartz"*>

<h:column>

<f:facet name=*"header"*>Nom</f:facet>

#{quartz.jobName}

</h:column>

<h:column>

<f:facet name=*"header"*>Grup</f:facet>

#{quartz.jobGroup}

</h:column>

<h:column>

<f:facet name=*"header"*>Proxima execucio</f:facet>

<h:outputText value=*"#{quartz.nextFireTime}"*>

<f:convertDateTime pattern=*"dd.MM.yyyy HH:mm:ss"* />

</h:outputText>

</h:column>

<h:column>

<f:facet name=*"header"*>Accio</f:facet>

<h:commandLink value=*"Executa"* action=*"#{SchedulerBean.fireNow(quartz.jobName, quartz.jobGroup)}"* />

</h:column>

</h:dataTable>

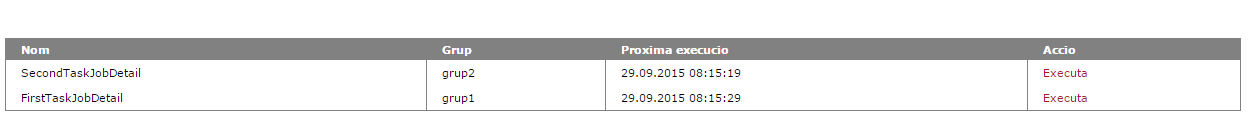
</h:form>

</ui:define>

…

**src\main\webapp\views\scheduler.xhtml**

**Resultat**



Tue Sep 29 10:17:49 CEST 2015 Executat metode de test de scheduler2

Tue Sep 29 10:17:59 CEST 2015 Executat metode de test de scheduler

Tue Sep 29 10:18:19 CEST 2015 Executat metode de test de scheduler2

*Tue Sep 29 10:18:25 CEST 2015 Executat metode de test de scheduler*

*Tue Sep 29 10:18:31 CEST 2015 Executat metode de test de scheduler2*

*Tue Sep 29 10:18:32 CEST 2015 Executat metode de test de scheduler*

Tue Sep 29 10:18:49 CEST 2015 Executat metode de test de scheduler

Tue Sep 29 10:18:49 CEST 2015 Executat metode de test de scheduler2

Les línies en vermell i cursiva són execucions manuals