COScheduler

Philippe Rabier - twitter.com/prabier
Sophiacom / bebook

Very often, if not always, we need to create jobs running periodically. It can be database cleanup, mail sending, statistics calculations, …

One implementation among other is to create DirectActions invoked by Cron but there is a better option: Quartz. Quartz is a famous java open source library that allows developers to create jobs easily. In this session, you will learn how it's easy to integrate Quartz in your WebObjects application and how you can use job persistance based on EOF.

COScheduler = Corason + Quartz Scheduler

But maybe, 1 day, we'll call it

ERXScheduler

Agenda

- Quartz Overview
- COScheduler Overview
- Implementation Process
- First Demo
- Diving into COScheduler
- Second Demo
- Q&A

Quartz Overview

Quartz Job Scheduler

- Very popular open source java library
- Exists for a long time (earlier than 2004)
- Can be used to create simple or complex schedules for executing tens, hundreds, or even tens-of-thousands of jobs
- Quartz 2.0 with new APIs live since 03/29/11

Main Components

- The JobDetails (JobDetail.java)
- The Triggers (Trigger.java)
- The Jobs (Job.java)
- The Scheduler (Scheduler.java)
- The Job Store (JobStore.java)

➡ All of this objects are interface!

COScheduler Overview

The Goals

- Hide the (relative) complexity of Quartz
- Make simple job scheduling very simple to implement
- EOF integration
- Based on Project Wonder but with few dependancies (ERJars, ERExtensions and ERJavaMail)
- Based on the “Framework Principal” pattern (mandatory)
  http://wiki.objectstyle.org/confluence/display/WONDER/Creating+a+ERXFrameworkPrincipal+subclass
- You can also use pure java objets rather than EOs!

But…

- No EOModel embedded
- Reduce the possibilities provided by Quartz
  - Only one Trigger linked to a Job Detail
  - Only CronTrigger is supported
  - Persistence must be handled by you (use of the RAMJobStore)
- No clustering
- Doesn't fit your needs if you have to handle a big number of jobs

Your Best Friends

- COJobDescription
  - Interface that your EOs must implement
- COSchedulerServiceFrameworkPrincipal
  - Abstract class that you have to subclass.
- COJob
  - Abstract class that you can use to implement your jobs

Implementation Process

COJobDescription

- COJobDescription = JobDetail + 1 Trigger
- There are 13 methods exposed by the interface
- Only 4 are mandatory:
  - name() (the name must be unique)
  - cronExpression()
  - classPath() (ex: org.wocommunity.demo.job.JobDemo)
  - isEnterpriseObject() (must return true if EO)

COSchedulerFrameworkPrincipal

```
public class MyFrameworkPrincipal extends COSchedulerServiceFrameworkPrincipal
{

     static
     {
          setUpFrameworkPrincipalClass(MyFrameworkPrincipal.class);
     }

     @Override
     public NSArray<? extends COJobDescription> getListOfJobDescription(final EOEditingContext editingContext)
     {
          return myMethodWhichFetchesObjects(editingContext);
     }

     @Override
     public EOEditingContext newEditingContext()
     {
          return MyFactory.newEditingContext(new EOObjectStoreCoordinator());
     }

     @Override
    // can be removed
     public void finishInitialization()
     {
          // Your code if you need;
          super.finishInitialization();
     }
}
```

COJob, the worker

- It's an abstract class that exposes 5 methods to fill out
- Only 1 is mandatory:
  - _execute()
- The other are optional
  - willDelete(final COJobDescription jobDescription)
  - willSave(final COJobDescription jobDescription)
  - validateForDelete(final COJobDescription jobDescription)
  - validateForSave(final COJobDescription jobDescription)

La Demo I

It’s time to dive…

COJobDescription

- The complete description:
  - name() (the name must be unique)
  - cronExpression()
  - classPath() (ex: org.wocommunity.demo.job.JobDemo)
  - isEnterpriseObject() (must return true if EO)
  - group() (can return null, a default value is provided)
  - jobDescription() (useful when displaying the list of jobs)
  - jobInfos() (return a map passed to the job when running)

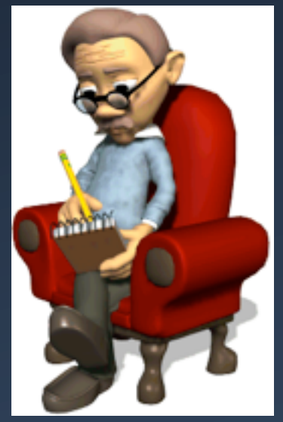
COJobDescription

- The complete description (follow):
  - recipients(boolean executionSucceeded)
    a mail can be sent automatically when the job is done
  - lastExecutionDate()
    setLastExecutionDate(NSTimestamp lastExecutionDate)
  - firstExecutionDate()
    setFirstExecutionDate(NSTimestamp firstExecutionDate)
  - setNextExecutionDate(NSTimestamp nextExecutionDate)

Your Good Friends

The Job

The JobSupervisor



The JobListener

COJobSupervisor

- It's a job triggered by the scheduler periodically
- Fetches the job descriptions
  - It calls getListOfJobDescription()
- Update the list of jobs handled by Quartz
  - add and remove jobs
  - update existing one (JobDetail is rescheduled if only cronExpression changed, otherwise recreated)

COJobSupervisor

- You can define the periodicity by setting a property which is by default 5mn.
  - Example: fr.sophiacom.corason.scheduler.COJobSupervisor.sleepduration=10
    triggers the supervisor every 10 mn
- You can replace the default supervisor by your own

➡ Use an annotation @COMySupervisor defined in your framework principal class

COJobSupervisor

```
    @COMySupervisor("org.wocommunity.wowodc11.COJobSupervisor")
public class COSchedulerFP4Test extends COSchedulerServiceFrameworkPrincipal
{

    @Override
    public NSArray<COJobDescription> getListOfJobDescription(final EOEditingContext editingContext)
    {
        return NSArray.emptyArray();
    }

    @Override
    public EOEditingContext newEditingContext()
    {
        return new MockEditingContext();
    }
}
```

COJobSupervisor

- Manage only the job descriptions returned by getListOfJobDescription().
- How does it do that?
- There is a convention: it adds a prefix to the group
  - “CO.” by default
  - prefix can be changed by the property fr.sophiacom.corason.scheduler.foundation.COJobSupervisor.prefix

COJobListener

- Just before the job runs, it posts a notification
  - Register an observer with COJobListener.JOB_WILL_RUN
  - Get the Job Description in the notification userInfo
  - 2 potential keys:
    COJob.ENTERPRISE_OBJECT_KEY
    COJob.NOT_PERSISTENT_OBJECT_KEY

COJobListener

- When the job is done, it does the following:
  - Post a notification
  - Update the Job Description (firstExecutionDate, last and next)
  - Send an email
  - Log the result

********** Job 'CO.DEFAULT.My first job' is starting. FireTime:Wed Jun 29 15:47:20 CEST 2011 /previousFireTime:Wed Jun 29 15:47:00 CEST 2011 /nextFireTime:Wed Jun 29 15:47:40 CEST 2011 **********

********** Job 'CO.DEFAULT.My first job' is done and it took: 10s **********

COJobListener

- The notification when the job is done
  - Register an observer with COJobListener.JOB_RAN
  - Get the Job Description in the notification userInfo
  - Get the exception with the key COJob.EXCEPTION_KEY

COJobListener

- Email sending when the job is done
  - A default text mail template is provided
  - You can give more informations by calling setResultMessage() just before the job ends up
  - Call recipients(boolean executionSucceeded)
  - You must set the following properties

    fr.sophiacom.ynp.schedulerService.COJobListener.sendingmail=true

    fr.sophiacom.ynp.schedulerService.COJobListener.from=noreply@wocommunity.org

    fr.sophiacom.ynp.schedulerService.COJobListener.to=admin@wocommunity.org

COJobListener

```
    protected void _execute() throws JobExecutionException
{
    COJobDescription jd = getJobDescription();
    for (int i = 0; i < 10; i++)
    {
        System.out.println("job name: " + jd.name() + " /i value: " + i);
        setResultMessage(jd.name() + " : " + i);
        try {
            Thread.sleep(1000L);
        } catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }
    setResultMessage("The job " + jd.name() + " is happy!");
}
```

Example of sent message:

From: noreply@sophiacom.fr
Subject: Job info: CO.Figaroclassifieds.CadrEmploiNotificationSender is done.
Date: 29 juin 2011 16:24:28 HAEC
To: admin

More informations: # new notifications: 16561 /# dks: 36587 /# cache: 33987 /max parsing duration: 223ms /# HTTP requests: 33992 /# HTTP errors: 5. It took 19mn 28s

And if you are not happy…

COJobListener

```
    @COMyJobListener("org.wocommunity.MyJobListener")
public class COSchedulerFP4Test extends COSchedulerServiceFrameworkPrincipal
{

    @Override
    public NSArray<COJobDescription> getListOfJobDescription(final EOEditingContext editingContext)
    {
        return NSArray.emptyArray();
    }

    @Override
    public EOEditingContext newEditingContext()
    {
        return new MockEditingContext();
    }
}
```

Das Demo 2!

Sum Up

- Easy to integrate
- Start with basic functionalities
- Available…soon

The One Most Important Thing!

Co-operation with JavaMonitor

- The goals for co-operation with JavaMonitor are these:
  - Allow the application running the scheduled jobs to be scheduled for restarts in JavaMonitor
  - Don't kill scheduled Jobs, allow them to shut down cleanly
  - Able to defer a scheduled restart or a termination until the last thread ended cleanly.

Co-operation with JavaMonitor

Add the following code in your Application class

```
public void refuseNewSessions(final boolean shouldRefuse)
{
   if (shouldRefuse)
   {
      COSchedulerServiceFrameworkPrincipal.getSharedInstance().deleteAllJobs();
   }
   super.refuseNewSessions(shouldRefuse);
}

public void _terminateFromMonitor()
{
   COSchedulerServiceFrameworkPrincipal.getSharedInstance().deleteAllJobs();
   super._terminateFromMonitor();
}

public boolean isTerminating()
{
if (super.isTerminating())
{
     try
     {
          if (COSchedulerServiceFrameworkPrincipal.getSharedInstance().hasRunningJobs())
               return false;
          if (COSchedulerServiceFrameworkPrincipal.getSharedInstance().getScheduler().isStarted())
               COSchedulerServiceFrameworkPrincipal.getSharedInstance().stopScheduler();
     } catch (SchedulerException e)
     {
          log.error("method: isTerminating", e);
     }
}
return super.isTerminating();
}
```

Q&A