hybris software

An SAP Company

hybris software
An SAP Company

hybris Developer Training Part I - Core Platform

CronJobs

Overview

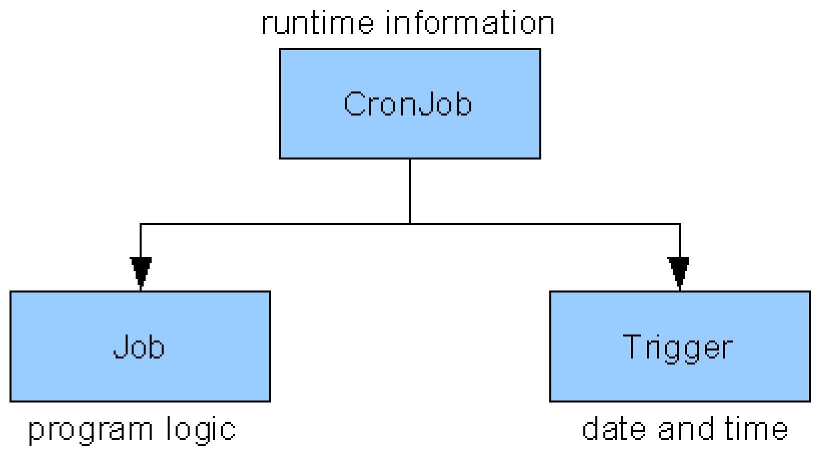
Cronjob scripting

CronJobs – Key Facts

- Automated tasks
- Performed at a certain time (such as 16:05), or at fixed intervals (such as every five minutes)
- Can be used for:
  - Backups
  - Updating / synchronizing catalog contents
  - Imports / Exports
  - Re-calculating prices
  - etc…

CronJobs – Key Facts (2)

- A CronJob consists of a:
  - Job: What to do
  - Trigger: When to run
  - CronJob: Runtime information



- Allows re-using code and items
- CronJobs always run in a SessionContext (i.e. they have a user assigned)

How to create a CronJob

Step 1 – create job logic:

```
   public class MyJob implements JobPerformable<CronJobModel>
   {
      public PerformResult perform(final CronJobModel cronJob)
         {
           //the logic
         }
   }
```

Step 2 – register it as spring bean:

```
   <bean id="myJob" class="my.package.MyJob"/>
```

How to create a CronJob (2)

- Step 3 – create items: `job`, `cronJob` and `trigger` items
  - Run a system update with only essential data checked – for each bean of type JobPerformable a Job item is being created.

```
INSERT_UPDATE CronJob;
code[unique=true];job(code);singleExecutable;sessionLanguage(isocode)
;myCronJob;myJob;false;de


INSERT_UPDATE Trigger;cronjob(code)[unique=true];cronExpression
; myCronJob; 0 0 0 * * ?
```

- Equivalent to running system update is to run such script:

```
INSERT_UPDATE ServicelayerJob;code[unique=true];springId
;myJob;myJob
```

- To start cronJob immediately after its creation, add this line at the end:

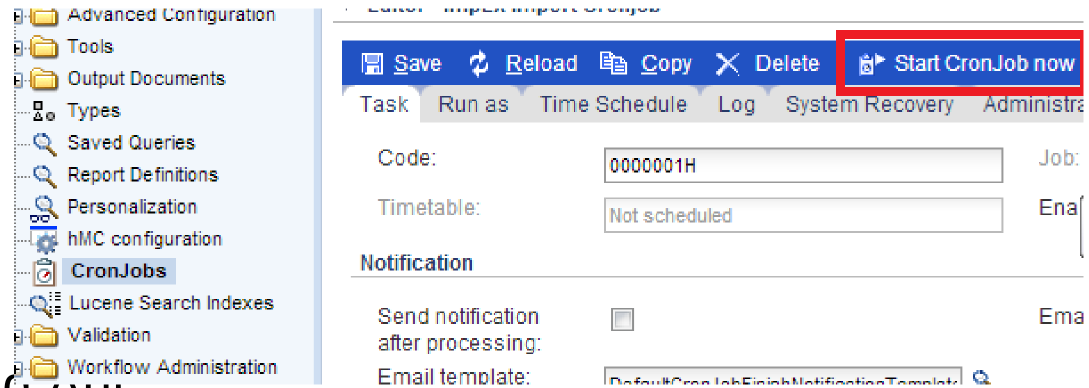
```
#% afterEach: impex.getLastImportedItem().setActivationTime(new Date());
```

How to start a CronJob

- Using Impex

```
#% afterEach: impex.getLastImportedItem().setActivationTime(new Date());
```

- Using the hMC



Code: 0000001H

Timetable: Not scheduled

Send notification after processing: ☐

Email template: DefaultCronJobFinishNotificationTemplat

- Using Ant

```
ant runcronjob -Dcronjob=myCronJob
```

- Using the API

```
cronJobService.performCronJob( myCronJobModel );
```

Useful CronJob properties

- Email template
  - notify certain user using given email template
- Enable code execution
  - enable or disable BeanShell
- User
  - to empower restrictions
- Node id
  - to specify server for job execution

Overview

Cronjob scripting

hybris Developer Training Part I - Core Platform

Benefits of Cronjob scripting

- Traditionally, creating a new cronjob is time-consuming and involves many manual steps:
  - Create a new java class for the Job
  - Define the new job as a Spring bean
  - Rebuild the code and restart the server

Using scripting, creating cronjobs becomes much easier and it can be done dynamically at runtime

Cronjob Scripting API

- **Script** - the item type where the script content is stored

```
INSERT_UPDATE Script; code[unique=true];content
;myGroovyScript;println 'hello groovy! '+ new Date()
```

- **ScriptingJob** - subtype of ServicelayerJob, which contains the scriptURI (the script can be retrieved at runtime from classpath, DB etc.)

```
INSERT_UPDATE ScriptingJob; code[unique=true];scriptURI
;mydynamicJob;model://myGroovyScript
```

- **scriptingJobPerformable** - the implicit spring bean assigned to every **ScriptingJob** instance; it implements the usual **perform()** method.

Executing script-based Cronjobs

- Creating a cronjob instance

```
INSERT_UPDATE CronJob; code[unique=true];job(code)
;mydynamicCronJob;mydynamicJob
```

- Executing a cronjob using a script

```
def dynamicCJ = cronJobService.getCronJob("mydynamicCronJob")
cronJobService.performCronJob(dynamicCJ,true)
```

- All other ways of execution can be used: Trigger, manual execution in hmc/backoffice, and impex beanshell.
- In a script, you can return a cronjob result

```
println 'hello groovy! '+ new Date()
return new PerformResult(CronJobResult.SUCCESS,CronJobStatus.FINISHED)
```

Cronjob Item as Context Parameter

- Context always contains the current **CronJobModel**
- It is passed as a context parameter (key="**cronjob**"**).**

```
println 'hello groovy! '+ new Date()
println cronjob.code
println cronjob.status
```