Cronjob

The **cronjob** functionality is used for executing tasks, called cron jobs, regularly at a certain point of time.

- You may set up cron jobs to run just once, once an hour, every 35 minutes and 17 seconds, every day, every week and so on.
- Typically cron jobs can be used for creating data for backups, updating catalog contents, or recalculating prices.
- The key idea of applying cron jobs is to start a long or periodic process in a background, having the possibility to protocol each run and to easily check its result.

How to create New CronJob:

- Create a new package in facade layer or storefront layer.In this package create one class for cronjob.
- The Cronjob class can be implement or extends JobPerfomble.
- we can provide our own logic in this class.
- In ExtensionName-spring.xml we can create bean by autowired with byName method.
- Now do **ant build** and **restart server**.
- Create the impex file.
- Goto hac and import the impex file.
- Goto hac and update system.
- Goto hmc -System-Cronjob-search your cronjob name.
- Open cronjob and set your language name english.
- Strart the cronjob method in hmc.

Defining the Job

CronJobTrail.java

```
/**
 *
 */
package com.lycamobile.facades.cronjob;

import de.hybris.platform.cronjob.enums.CronJobResult;
import de.hybris.platform.cronjob.enums.CronJobStatus;
import de.hybris.platform.cronjob.model.CronJobModel;
import de.hybris.platform.servicelayer.cronjob.JobPerformable;
import de.hybris.platform.servicelayer.cronjob.PerformResult;

import org.apache.log4j.Logger;

/**
 * @author jagadish
 *
 */
public class CronJobTrail implements JobPerformable<CronJobModel>
{
	private static final Logger LOG = Logger.getLogger(CronJobTrail.class);

	/*
	 * (non-Javadoc)
	 *
	 * @see
de.hybris.platform.servicelayer.cronjob.JobPerformable#isAbortable()
	 */
	@Override
	public boolean isAbortable()
	{
		// YTODO Auto-generated method stub
		return true;
	}

	/*
```

```
	 * (non-Javadoc)
	 *
	 * @see
de.hybris.platform.servicelayer.cronjob.JobPerformable#isPerformable()
	 */
	@Override
	public boolean isPerformable()
	{
		// YTODO Auto-generated method stub
		return true;
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
de.hybris.platform.servicelayer.cronjob.JobPerformable#perform(de.hybris.platf
orm.cronjob.model.CronJobModel)
	 */
@Override
	public PerformResult perform(final CronJobModel arg0)
	{
		LOG.info("hai");

		return new PerformResult(CronJobResult.SUCCESS,
CronJobStatus.FINISHED);

	}

}
```

ExtensionName-spring.xml

```
<bean id="cronJobTrail"
    class="com.lycamobile.facades.cronjob.CronJobTrail
"  autowire="byName">
</bean>
```

- Rebuild the hybris Platform by calling **`ant`** in the *$/{HYBRIS_BIN_DIR}*`/platform` directory.
- Run a system update with only **essential data** checked - during the phase of essential data creation, for each Spring definition of a class implementing the `JobPerformable` interface, a ServicelayerJob instance gets created and the code attribute of the job is set to the name of the Spring bean.

You can check in the FlexibleSearch console:http://localhost:9001/console/flexsearch that the new item was created by executing the following query:

```
select {code} from {servicelayerjob} where {code} = 'cronJobTrail'
```

Create the CronJob and the Trigger

To create the CronJob and the Trigger, you can either:

- You can go in the hybris Admin Console to the Console tab select ImpEx Import and execute the following impex-script there by clicking on the Import Content button.

Impex File

```
INSERT_UPDATE CronJob       ;code[unique=true]     ;job(code)
        ;baseStore(uid)  ;cmsSite(uid)     ;sessionUser(uid);
sessionLanguage(isocode);sessionCurrency(isocode)
                        ;cronJobTrail            ;cronJobTrail     ;$siteUid   ;$siteUid
        ;admin          ;en                    ;USD

INSERT_UPDATE Trigger;cronjob(code)[unique=true];cronExpression
#% afterEach: impex.getLastImportedItem().setActivationTime(new Date());
                   ;cronJobTrail            ; 30 0 0 * * ?
```

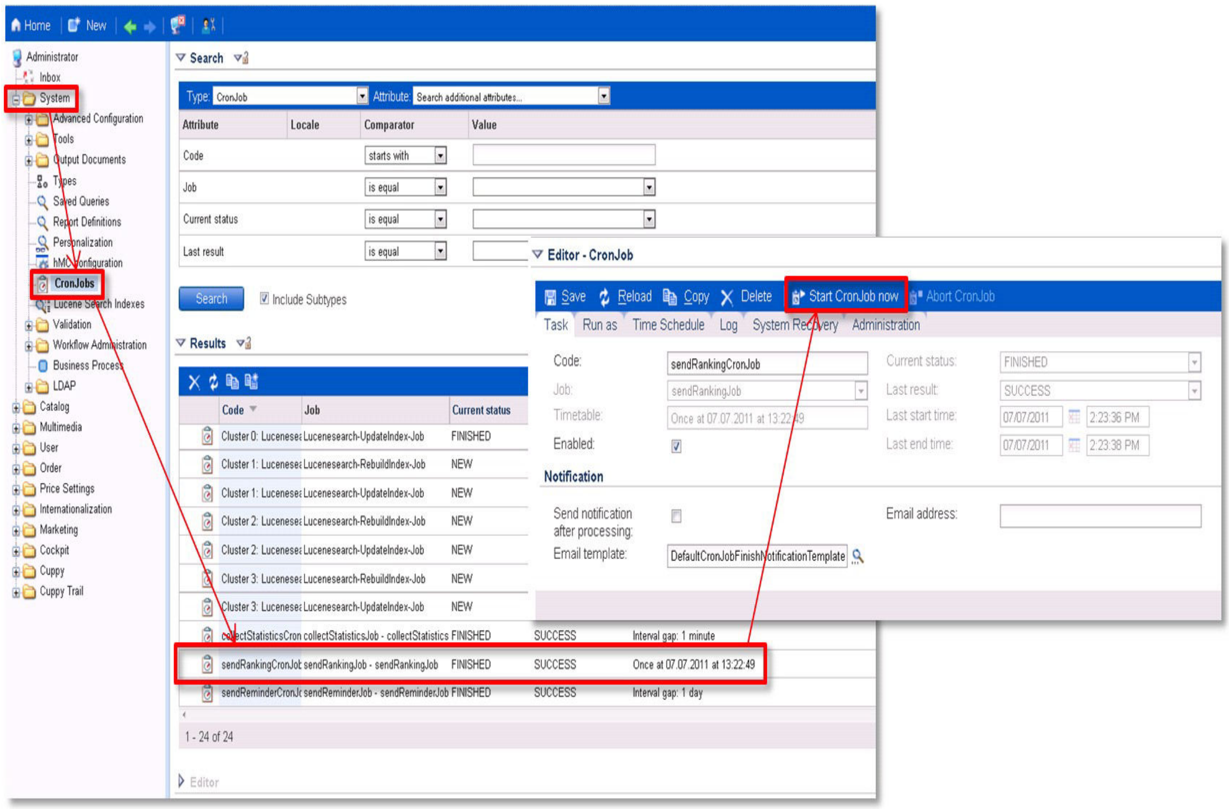
A cron expression is a string comprised of 6 or 7 fields separated by white space. Fields can contain any of the allowed values, along with various combinations of allowed special characters for that field.

FIELD NAME	MANDATORY	ALLOWED VALUES	ALLOWED SPECIAL CHARACTERS
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of week	YES	1-7 or SUN-SAT	, - * ? / L #
Year	NO	empty, 1970-2099	, - * /

- Quartz cron trigger is used, see Quartz Scheduler for more details. Any changes you are making for testing can easily be redeployed or reexecuted.
- OR create the file **resources/impex/essentialdataJobs.impex** with the same content. Further changes are only taken into account after a server-restart and a system update (with only essential data checked). Changes made to resources after being loaded as classloader-resources are not visible.

Test the job

- Simply log in into the hmc
- Open the cuppy-section and edit a player
    - Set the newsletter-attribute to true
    - Change the eMail address of this player to your email address (if you set up email previously)
- Execute the job manually
    - In the left tree, go to System | CronJobs
    - Select sendRankingCronJob
    - Click 'Start CronJob now'



Type: CronJob Attribute: Search additional attributes...

Attribute	Locale	Comparator	Value
Code		starts with	
Job		is equal	
Current status		is equal	
Last result		is equal	

☑ Include Subtypes

Code	Job	Current status		
Cluster 0: Lucenese:	Lucenesearch-UpdateIndex-Job	FINISHED		
Cluster 1: Lucenese:	Lucenesearch-RebuildIndex-Job	NEW		
Cluster 1: Lucenese:	Lucenesearch-UpdateIndex-Job	NEW		
Cluster 2: Lucenese:	Lucenesearch-RebuildIndex-Job	NEW		
Cluster 2: Lucenese:	Lucenesearch-UpdateIndex-Job	NEW		
Cluster 3: Lucenese:	Lucenesearch-RebuildIndex-Job	NEW		
Cluster 3: Lucenese:	Lucenesearch-UpdateIndex-Job	NEW		
collectStatisticsCron	collectStatisticsJob - collectStatistics	FINISHED	SUCCESS	Interval gap: 1 minute
sendRankingCronJot	sendRankingJob - sendRankingJob	FINISHED	SUCCESS	Once at 07.07.2011 at 13:22:49
sendReminderCronJc	sendReminderJob - sendReminderJob	FINISHED	SUCCESS	Interval gap: 1 day

1 - 24 of 24

Editor - CronJob

Code:	sendRankingCronJob	Current status:	FINISHED
Job:	sendRankingJob	Last result:	SUCCESS
Timetable:	Once at 07.07.2011 at 13:22:49	Last start time:	07/07/2011 2:23:36 PM
Enabled:	☑	Last end time:	07/07/2011 2:23:38 PM

Notification

Send notification after processing:	☐	Email address:	
Email template:	DefaultCronJobFinishNotificationTemplate		

- Check your console (and email if appropriate) for output