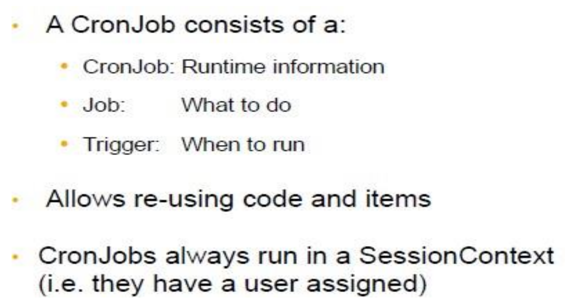
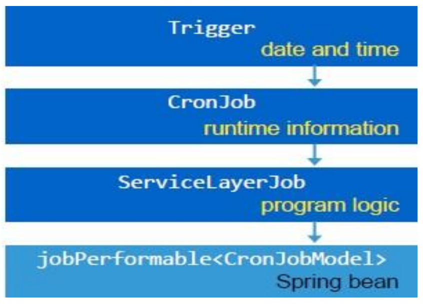
Explain Cron Jobs?

It is a Timer / Scheduler program. It will try to do the activity in schedule way. You want to do some periodic activity then you will write Cron Jobs. Every week Sunday you want to send some **promotional**. Every month Salary Credit. The other way is -- Every day, we need to send some feed to external system.

Q: What CronJob Contains?



- A CronJob consists of a:
  - CronJob: Runtime information
  - Job: What to do
  - Trigger: When to run
- Allows re-using code and items
- CronJobs always run in a SessionContext (i.e. they have a user assigned)



Q: What are the Part of CronJobs? =

- √ **Trigger** = This type is used for scheduling when to run the job. Which uses Cron expression **[* * * * ? *]** which indicates [**minutes, hours, day of month, month, day of week, year**] to run the job.
- √ **CronJob** = This type holds the business logic which is to be performed at particular times and intervals.
- √ **Job** = This type consists of logic to be executed which is defined by Job Performable
  Create a new class that extends **AbstractJobPerformable** class **or** implement **JobPerformable** interface.

Q: What are different steps required to create a cron job?

- √ **Step 1 =** Create new class called "xxxJob.java" by extending **AbstractJobPerformable** class & Override the **perform ()** method & write the business (cron job) logic in this method
- √ **Step 2 =** Goto xxx-spring.xml file & register the XxxxCronJob.java bean
- √ **Step 3 =** Rebuild the hybris platform by calling "ant clean all"
- √ **Step 4 =** Start the hybris server and perform platform update
- √ **Step 5 =** Create the cron job & resigter the cron jobusing trigger

Q: What are the major different types of the preconfigured cronjobs in hybris?

- √ SOLR and Lucene related: indexing, updating, removing data
- √ Clean up unnecessary data from the database or file system
- √ Product Catalog synchronization
- √ Regular data export (Product, Price, Inventory, Order Status, and …. Import / Export).
- √ Workflow
- √ Impex import.

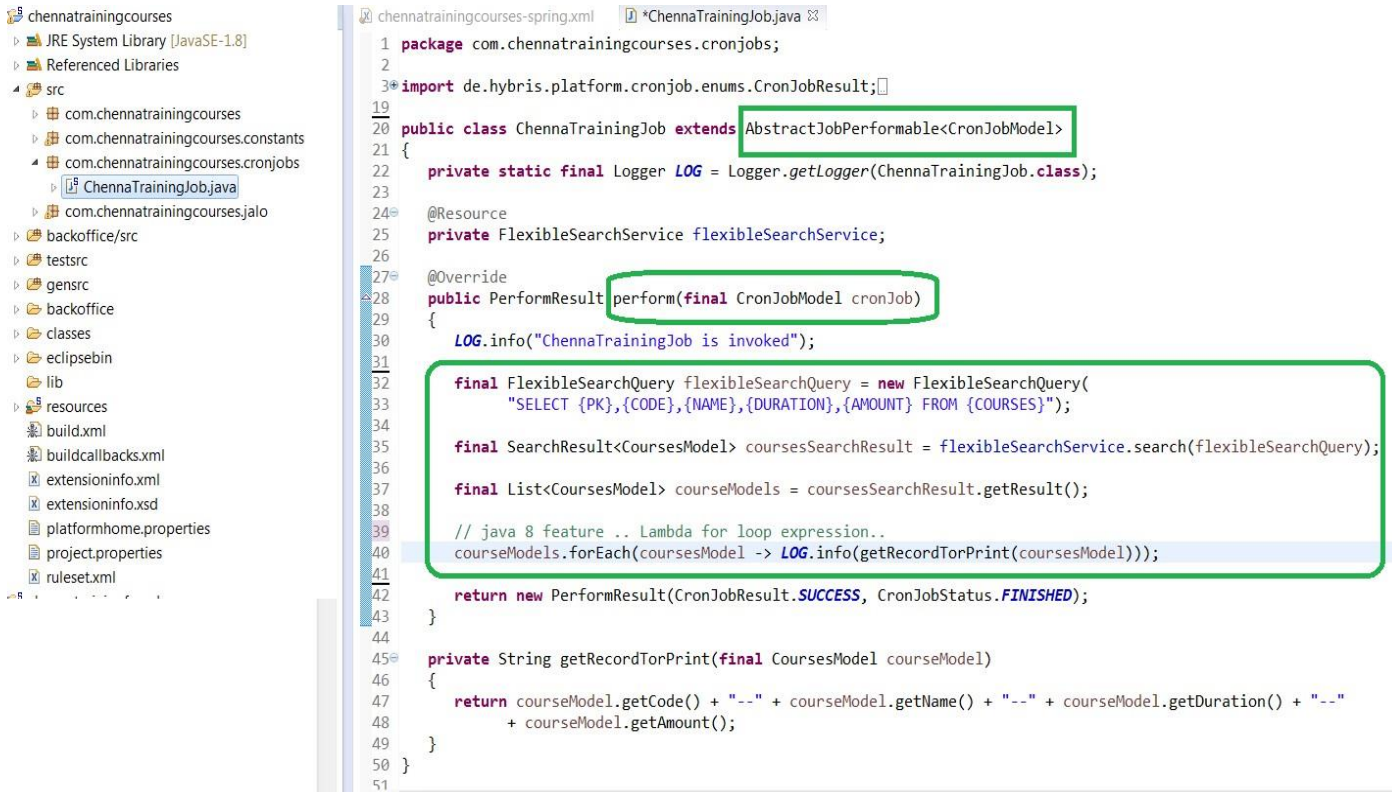
Q: What are the different ways to create cron job?

- √ **Option 1 =** Using ImpEx
- √ **Option 2 =** Using groovy scripts (scripting engine dynamically at runtime. you can create the crob job with groovy script and this script need to execute in hac console scripting languages)

Cron job example using ImpEx

Scenario – read the records from “TrainingCourses” table for every 1 minutes and display in the logs. Let’s create the cron job under “**ChennaTrainingCourses**” project

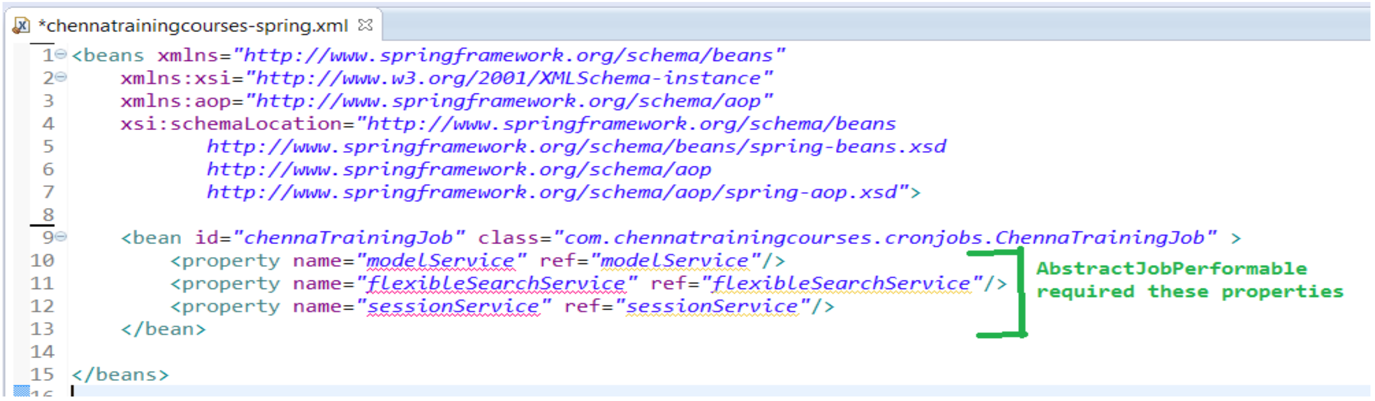
Step 1 = Create new class called “ChennaTrainingJob.java” by extending **AbstractJobPerformable** class & **Override** the perform () method and write the business (cron job) logic in this method



```java
package com.chennatrainingcourses.cronjobs;

import de.hybris.platform.cronjob.enums.CronJobResult;

public class ChennaTrainingJob extends AbstractJobPerformable<CronJobModel>
{
    private static final Logger LOG = Logger.getLogger(ChennaTrainingJob.class);

    @Resource
    private FlexibleSearchService flexibleSearchService;

    @Override
    public PerformResult perform(final CronJobModel cronJob)
    {
        LOG.info("ChennaTrainingJob is invoked");

        final FlexibleSearchQuery flexibleSearchQuery = new FlexibleSearchQuery(
                "SELECT {PK},{CODE},{NAME},{DURATION},{AMOUNT} FROM {COURSES}");

        final SearchResult<CoursesModel> coursesSearchResult = flexibleSearchService.search(flexibleSearchQuery);

        final List<CoursesModel> courseModels = coursesSearchResult.getResult();

        // java 8 feature .. Lambda for loop expression..
        courseModels.forEach(coursesModel -> LOG.info(getRecordTorPrint(coursesModel)));

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

    private String getRecordTorPrint(final CoursesModel courseModel)
    {
        return courseModel.getCode() + "--" + courseModel.getName() + "--" + courseModel.getDuration() + "--"
                + courseModel.getAmount();
    }
}
```

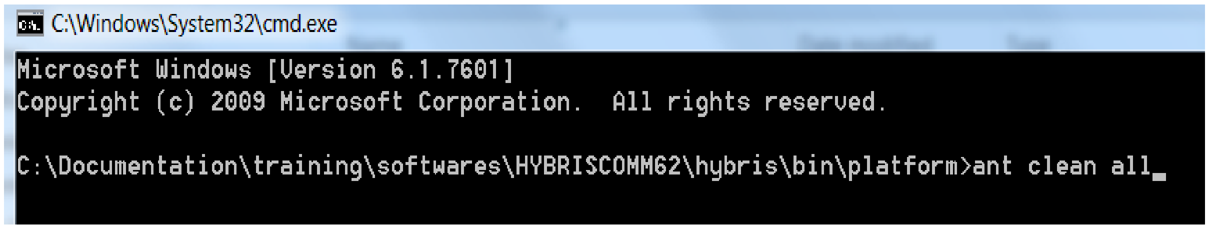
Step 2 = Goto **xxx-spring.xml** file & register the **xxxJob.java** bean

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:aop="http://www.springframework.org/schema/aop"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd
   http://www.springframework.org/schema/aop
   http://www.springframework.org/schema/aop/spring-aop.xsd">
    <bean id="chennaTrainingJob" class="com.chennatrainingcourses.cronjobs.ChennaTrainingJob" >
            <property name="modelService" ref="modelService"/>
            <property name="flexibleSearchService" ref="flexibleSearchService"/>
            <property name="sessionService" ref="sessionService"/>
    </bean>

</beans>
```


*chennatrainingcourses-spring.xml

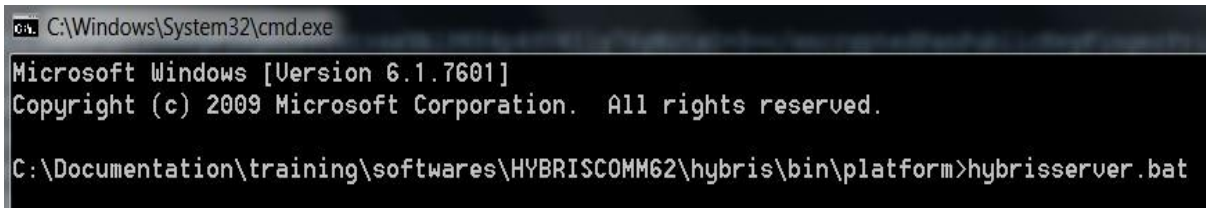
```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/aop
         http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="chennaTrainingJob" class="com.chennatrainingcourses.cronjobs.ChennaTrainingJob" >
        <property name="modelService" ref="modelService"/>
        <property name="flexibleSearchService" ref="flexibleSearchService"/>
        <property name="sessionService" ref="sessionService"/>
    </bean>

</beans>
```

Step 3 = Rebuild the hybris platform using "ant clean all"

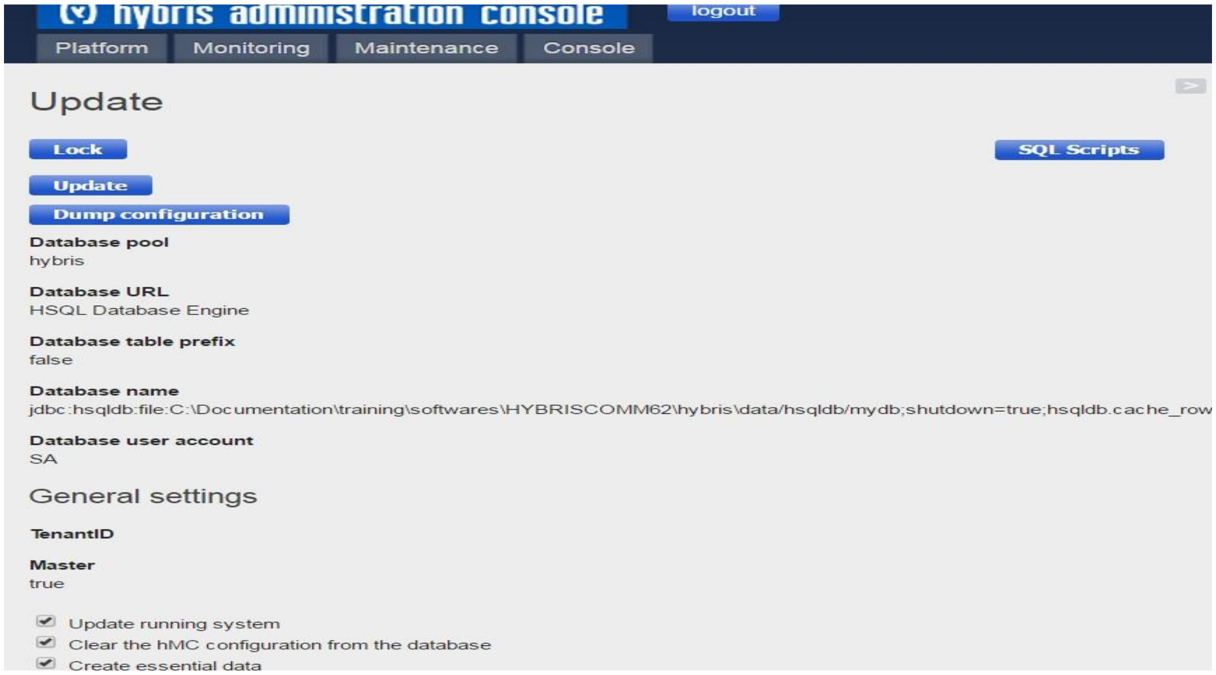

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Documentation\training\softwares\HYBRISCOMM62\hybris\bin\platform>ant clean all
```

Step 4 = Start the hybris server & Perform platform update (click on update)


C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Documentation\training\softwares\HYBRISCOMM62\hybris\bin\platform>hybrisserver.bat
```



Update

Database pool
hybris

Database URL
HSQL Database Engine

Database table prefix
false

Database name
jdbc:hsqldb:file:C:\Documentation\training\softwares\HYBRISCOMM62\hybris\data/hsqldb/mydb;shutdown=true;hsqldb.cache_row

Database user account
SA

General settings

TenantID

Master
true

✓ Update running system
✓ Clear the hMC configuration from the database
✓ Create essential data

Note - 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.

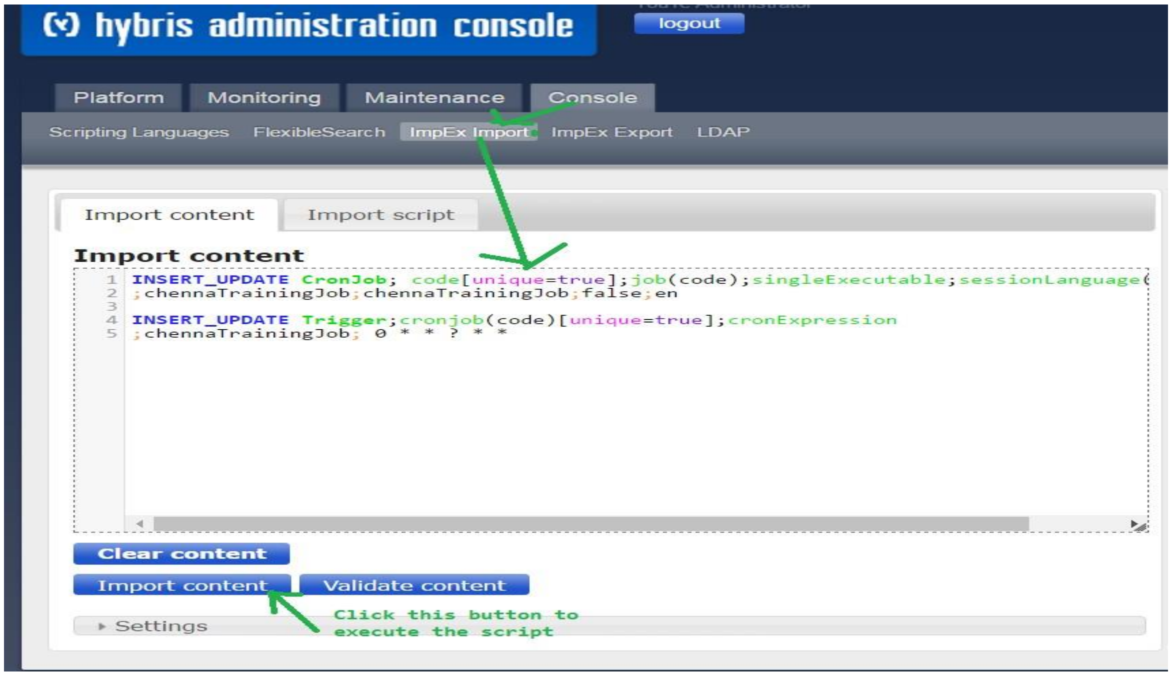
Step 5 = Create the **cron job** & **register** the cron job using trigger

There are 2 ways you can create the cron job and trigger

Option 1 = Goto hac plat form ImpEx Import & execute the following script by clicking on the **Import Content** button

```
INSERT_UPDATE CronJob; code[unique=true];job(code);singleExecutable;sessionLanguage(isocode)
;chennaTrainingJob;chennaTrainingJob;false;en

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

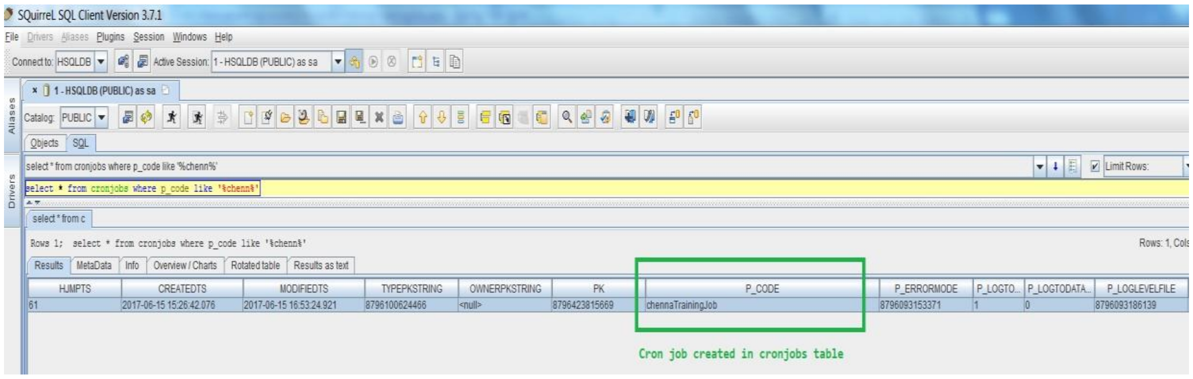
The cronjob runs for very one minute



Import content

```
INSERT_UPDATE CronJob; code[unique=true];job(code);singleExecutable;sessionLanguage(
;chennaTrainingJob;chennaTrainingJob;false;en

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

Verify the cronjob in the database under "**CronJobs**" table


SQuirreL SQL Client Version 3.7.1
File Drivers Aliases Plugins Session Windows Help
Connect to: HSQLDB
Active Session: 1 - HSQLDB (PUBLIC) as sa
1 - HSQLDB (PUBLIC) as sa
Catalog: PUBLIC
Objects SQL
Limit Rows:
Aliases
Drivers
select * from c
Results MetaData Info Overview / Charts Rotated table Results as text
Rows: 1, Cols
Cron job created in cronjobs table

```
select * from cronjobs where p_code like '%chenn%'
```

```
Rows 1;  select * from cronjobs where p_code like '%chenn%'
```

HJMPTS	CREATEDTS	MODIFIEDTS	TYPEPKSTRING	OWNERPKSTRING	PK	P_CODE	P_ERRORMODE	P_LOGTO...	P_LOGTODATA...	P_LOGLEVELFILE
61	2017-06-15 15:26:42.076	2017-06-15 16:53:24.921	8796100624466	<null>	8796423815669	chennaTrainingJob	8796093153371	1	0	8796093186139

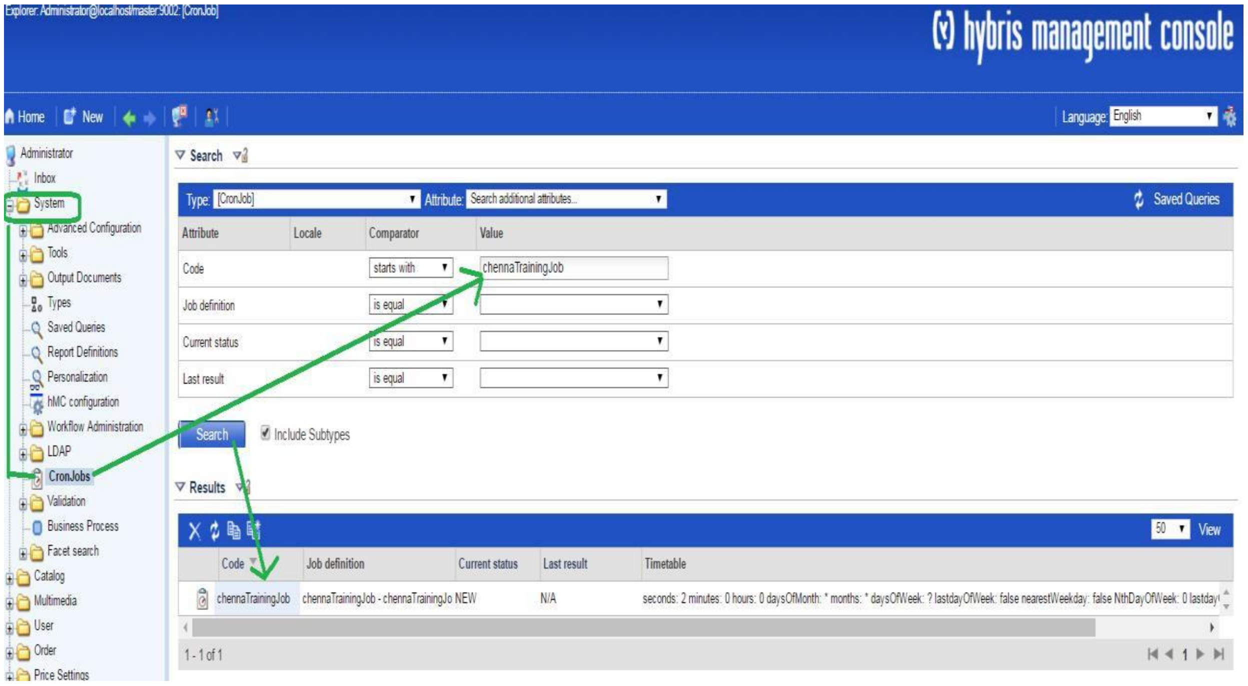
Optioin 2 = Create the file resources/impex/**essentialdataJobs.impex** with the same content as above & restart the server and perform update (with only essential data checked).

Step 6 = Test the cron job

You can test the cron job using different options.

Option 1 = using hmc

Step 1 = Goto hmc System CronJobs search for the job (chennaTrainingJob) run the job


Explorer: Administrator@localhost/master:9002: [CronJob]

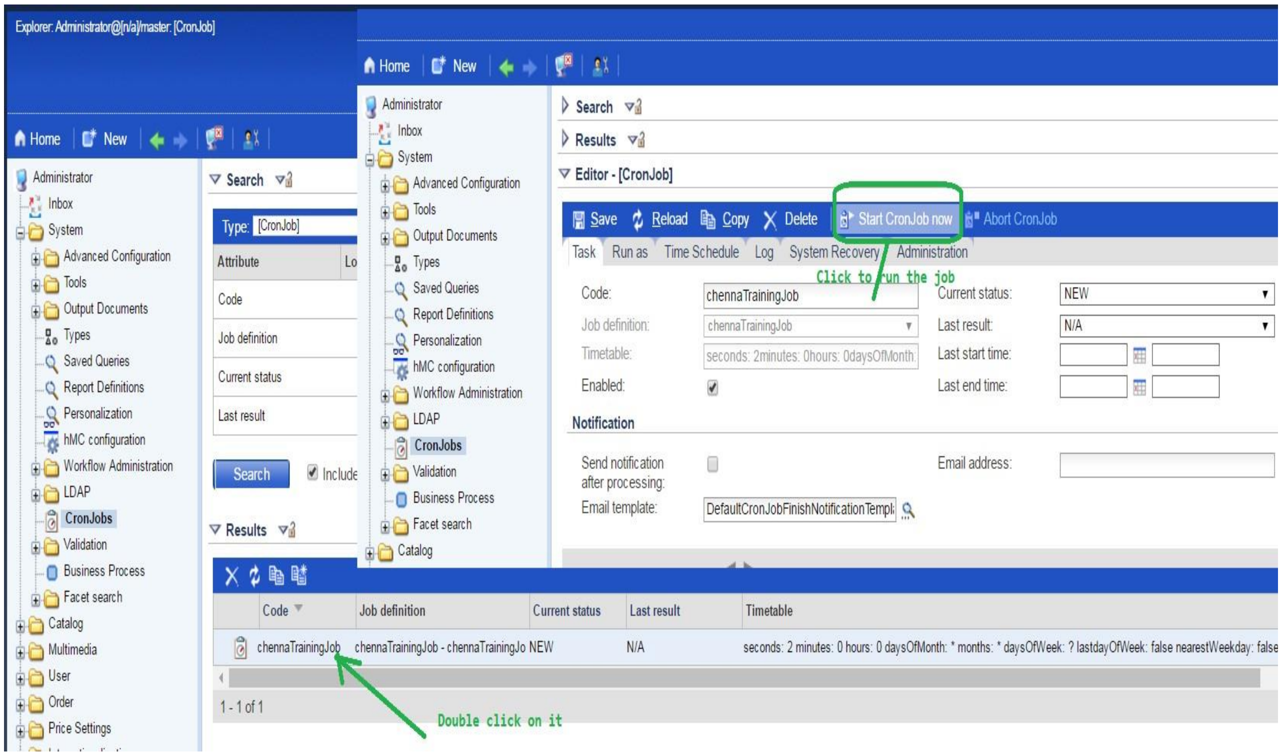
Type: [CronJob] Attribute: Search additional attributes...

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

Include Subtypes ✓

Code	Job definition	Current status	Last result	Timetable
chennaTrainingJob	chennaTrainingJob - chennaTrainingJo	NEW	N/A	seconds: 2 minutes: 0 hours: 0 daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastday

1 - 1 of 1


Explorer: Administrator@[n/a]/master: [CronJob]

Editor - [CronJob]

Save | Reload | Copy | Delete | Start CronJob now | Abort CronJob

Task | Run as | Time Schedule | Log | System Recovery | Administration

Click to run the job

Code:	chennaTrainingJob	Current status:	NEW
Job definition:	chennaTrainingJob	Last result:	N/A
Timetable:	seconds: 2minutes: 0hours: 0daysOfMonth:	Last start time:	
Enabled:	✓	Last end time:	

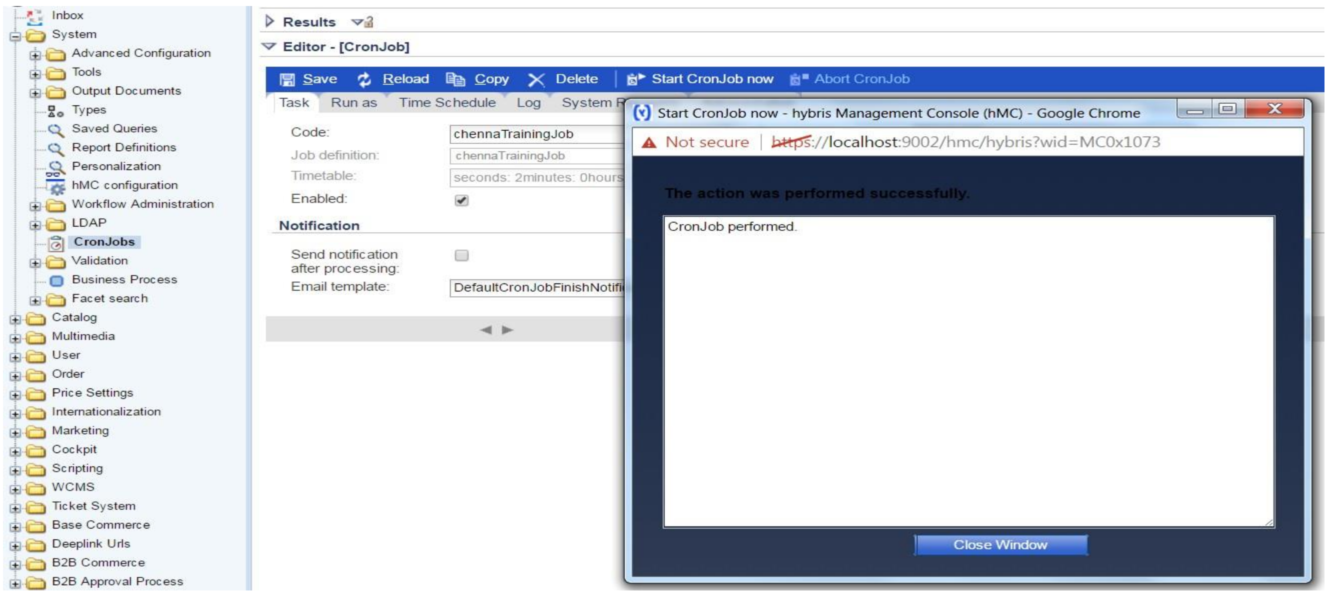
Notification

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

Code	Job definition	Current status	Last result	Timetable
chennaTrainingJob	chennaTrainingJob - chennaTrainingJo	NEW	N/A	seconds: 2 minutes: 0 hours: 0 daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false

1 - 1 of 1

Double click on it



Code: chennaTrainingJob
Job definition: chennaTrainingJob
Timetable: seconds: 2minutes: 0hours
Enabled: ✓
Send notification after processing: ☐
Email template: DefaultCronJobFinishNotifi

Start CronJob now - hybris Management Console (hMC) - Google Chrome
Not secure | ~~https~~://localhost:9002/hmc/hybris?wid=MC0x1073

The action was performed successfully.

CronJob performed.

You can close the window. The trigger will run cron job for the specified time.

Step 2 = Validate the crob job results by looking the logs (whatever the outcome you are expecting)

```
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] ChennaTrainingJob is invoked
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 10--Hybris--60 hours--500
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 11--Fiori--70 hours--600
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 12--Java--60 hours--500
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 20--C++--100 hours--500
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 22--Cloud Computing--80 hours--600
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 21--Hadoop--80 hours--600
INFO  [chennaTrainingJob::de.hybris.platform.servicelayer.internal.jalo.ServicelayerJob] (chennaTrainingJob) [ChennaTrainingJob] 23--Angular2--80 hours--600
```

Note – You can run the cron job using **hmc** / **back office** / **manually** / **javacode** etc.

Q: How to Start a CronJob? = There are different ways to start a cronjob which are given below :

- √ **Manually** start using HMC = hmc system cronjobs select CronJob "StartCronJobNow".
- √ **Automatically** running the CronJob Through Impex file
- √ Using the **ant** command ant runcronjob -d cronjob=“**CronJobName**“
- √ Using the **javacode** using the CronJob services we can run the cronjob.

Q: How to stop a cronjob? = We can stop the cronjob by following ways:

- √ Using the abort method in the java code. It is done automatically after performing the CronJob.
- √ Manually from hmc we can stop the CronJob.

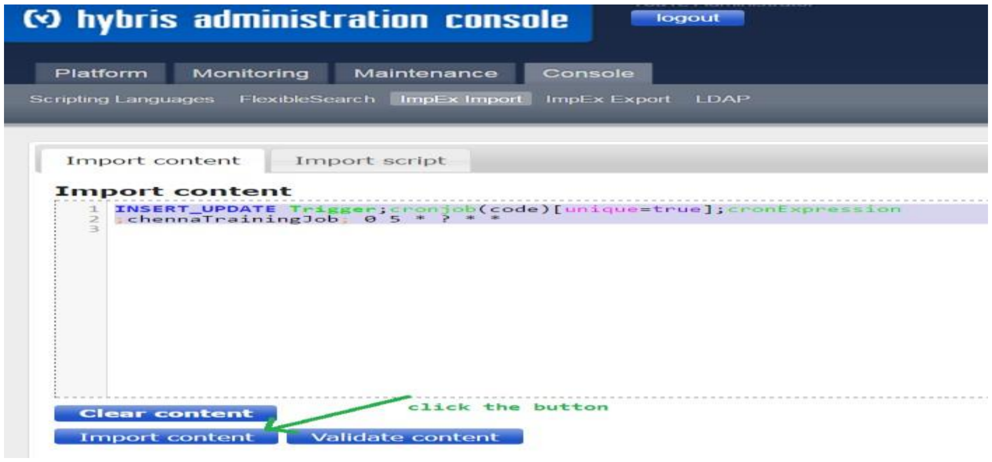
Q: Where to see the created CronJob? = hmc System CronJobs

Q: How to change the CronJob schedule time?

Option 1 = using ImpEx

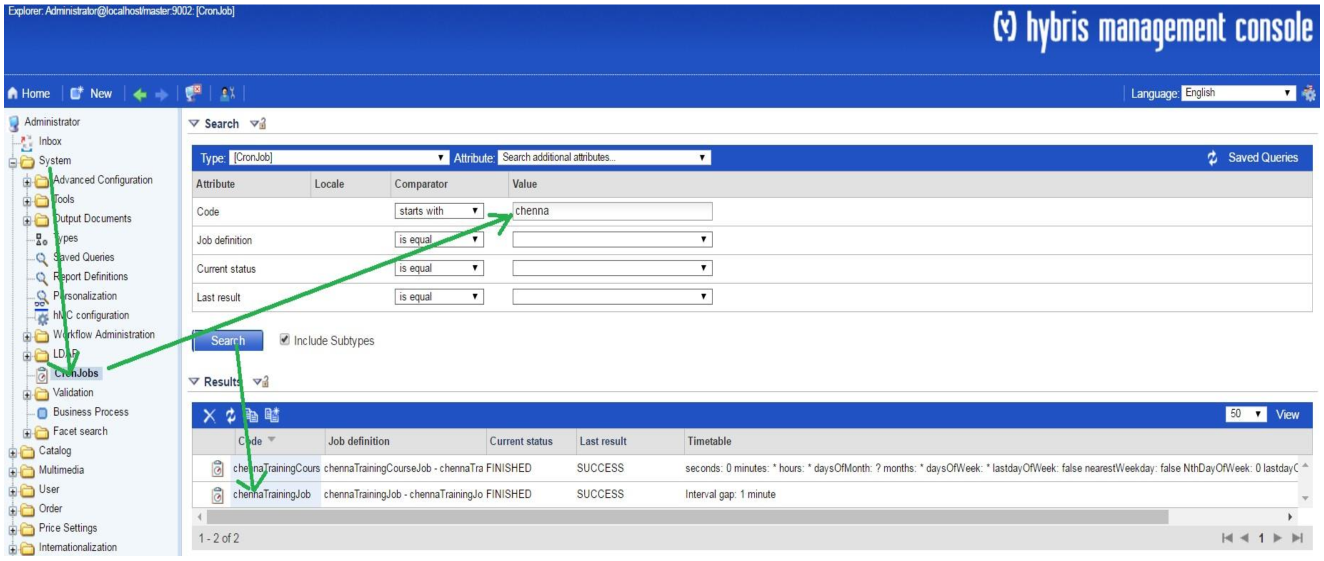
change cronExpression parameters (specify whatever time you want to execute the cronjob) & execute the script from hac console ImpEx Import

```
INSERT_UPDATE Trigger;cronjob(code)[unique=true];cronExpression
;chennaTrainingJob; 0 5 * ? * *
```



```
INSERT_UPDATE Trigger;cronjob(code)[unique=true];cronExpression
;chennaTrainingJob; 0 5 * ? * *
```

Option 2 = change using time schedule the hmc System CronJobs

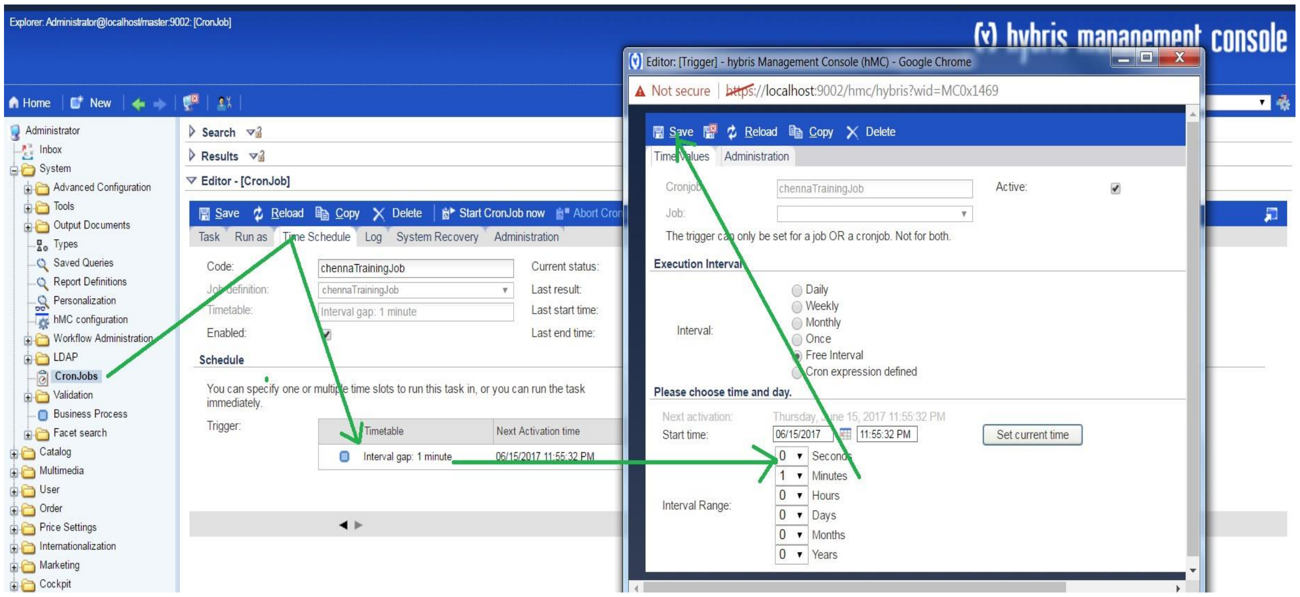


Type: [CronJob] Attribute: Search additional attributes...

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

Code	Job definition	Current status	Last result	Timetable
chennaTrainingCours	chennaTrainingCourseJob - chennaTra	FINISHED	SUCCESS	seconds: 0 minutes: * hours: * daysOfMonth: ? months: * daysOfWeek: * lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayC
chennaTrainingJob	chennaTrainingJob - chennaTrainingJo	FINISHED	SUCCESS	Interval gap: 1 minute

1 - 2 of 2



Code: chennaTrainingJob
Job definition: chennaTrainingJob
Timetable: Interval gap: 1 minute
Enabled: ✓
Current status:
Last result:
Last start time:
Last end time:

Schedule

You can specify one or multiple time slots to run this task in, or you can run the task immediately.

Trigger:

Timetable	Next Activation time
Interval gap: 1 minute	06/15/2017 11:55:32 PM

~~https~~://localhost:9002/hmc/hybris?wid=MC0x1469

Cronjob: chennaTrainingJob
Active: ✓
Job:
The trigger can only be set for a job OR a cronjob. Not for both.

Execution Interval

Interval: Daily / Weekly / Monthly / Once / [✓ Free Interval] / Cron expression defined

Please choose time and day.

Next activation: Thursday, June 15, 2017 11:55:32 PM
Start time: 06/15/2017 11:55:32 PM

Interval Range:

0	Seconds
1	Minutes
0	Hours
0	Days
0	Months
0	Years

Q: Explain Cron job Format: -

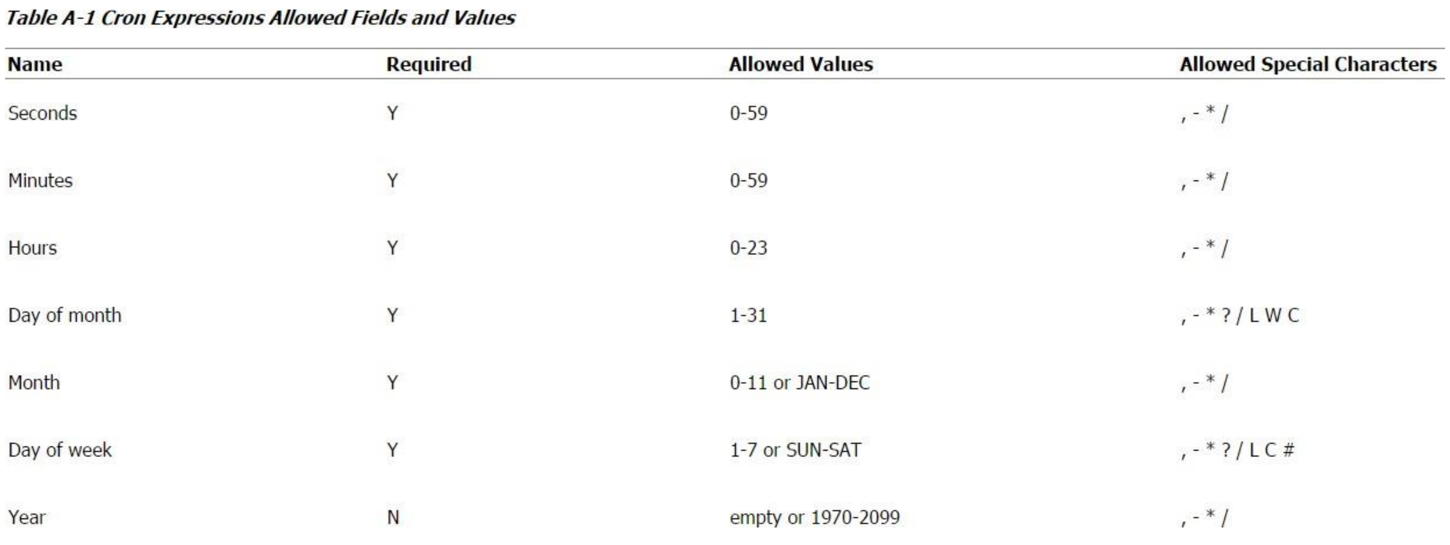
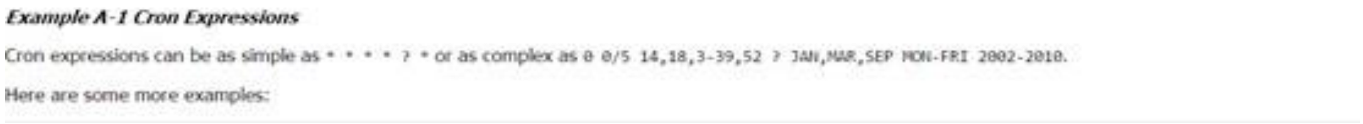


Table A-1 Cron Expressions Allowed Fields and Values

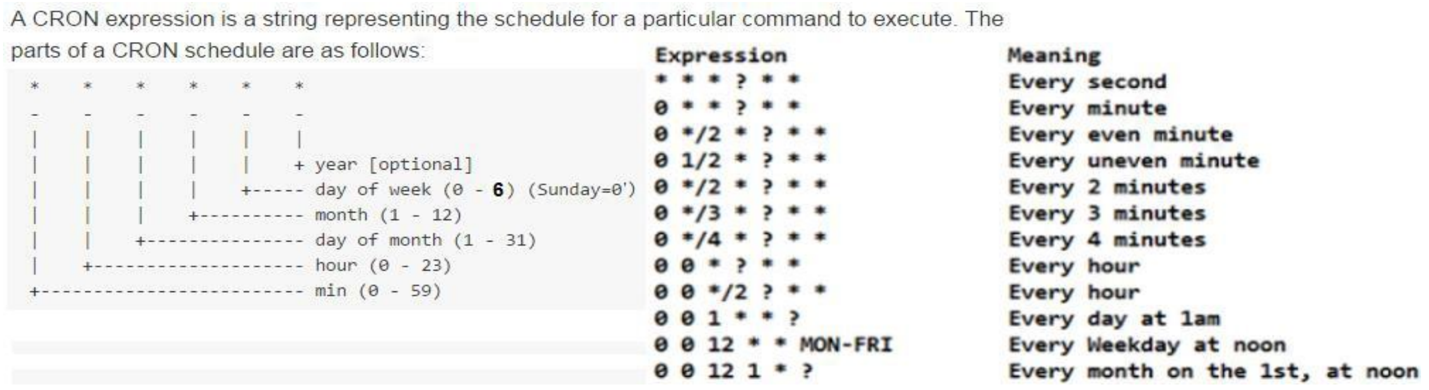
Name	Required	Allowed Values	Allowed Special Characters
Seconds	Y	0-59	, - * /
Minutes	Y	0-59	, - * /
Hours	Y	0-23	, - * /
Day of month	Y	1-31	, - * ? / L W C
Month	Y	0-11 or JAN-DEC	, - * /
Day of week	Y	1-7 or SUN-SAT	, - * ? / L C #
Year	N	empty or 1970-2099	, - * /



Example A-1 Cron Expressions

Cron expressions can be as simple as `* * * * ? *` or as complex as `0 0/5 14,18,3-39,52 ? JAN,MAR,SEP MON-FRI 2002-2010`.

Here are some more examples:



A CRON expression is a string representing the schedule for a particular command to execute. The parts of a CRON schedule are as follows:

```
 *    *    *    *    *    *
 -    -    -    -    -    -
 |    |    |    |    |    |
 |    |    |    |    |    + year [optional]
 |    |    |    |    +----- day of week (0 - 6) (Sunday=0')
 |    |    |    +---------- month (1 - 12)
 |    |    +--------------- day of month (1 - 31)
 |    +-------------------- hour (0 - 23)
 +------------------------- min (0 - 59)
```

Expression	Meaning
`* * * ? * *`	Every second
`0 * * ? * *`	Every minute
`0 */2 * ? * *`	Every even minute
`0 1/2 * ? * *`	Every uneven minute
`0 */2 * ? * *`	Every 2 minutes
`0 */3 * ? * *`	Every 3 minutes
`0 */4 * ? * *`	Every 4 minutes
`0 0 * ? * *`	Every hour
`0 0 */2 ? * *`	Every hour
`0 0 1 * * ?`	Every day at 1am
`0 0 12 * * MON-FRI`	Every Weekday at noon
`0 0 12 1 * ?`	Every month on the 1st, at noon

Q: What are the **Params** & **Configurations** required for Job to Execute?

- √ Define Second, Min, Hour and etc. (or) **Cron Expression** = Trigger defined through **Impex**
  hybris\bin\custom\chennatrainingcore\resources\impex\essentialdataJobs.impex
- √ **CronJob** Item Type Definition = ***.items.xml**
  hybris\bin\custom\chennatrainingcore\resources\chennatrainingcore-items.xml
- √ Job Definition / **Business Logic** (CronJob1.java extends AbstractJobPerformable) = **Java** class for Job
  hybris\hybris\bin\custom\chennatrainingcore\src\org\chennatraining\core\jobs\MyCronJob.java
- √ **Bean** Definition of Job (Example: - <bean id =”CronJob1” …) = ***-spring.xml**
  hybris\bin\custom\chennatrainingcore\resources\chennatrainingcore-spring.xml
- √ CronJob Model (Example: - SampleCronJobModel) = Platform

Q: How to see the Job Details? = select * from {servicelayerjob} where {code} = 'ChennaTraininglJob'

Q: How to Run Cron Job through Ant? = ant runcronjob -Dcronjob=chennaCronJob -Dtenant=master

Q: Explain Cron Job execution during the Server startup?

- √ When we start Hybris server, each trigger will be **evaluated** first, if trigger evaluated time is **overdue** or **matches** the current time then Trigger will be fired which means Cron jobs will be executed immediately.
  Q: How much **overdue time** we can allow for the triggers to get fired during the server startup?
  This can be done by setting **maxAcceptableDelay** (Value is in Seconds) attribute to the Trigger. Let's say – Value set = 600 (10 mins) & Trigger is defined to run **@ 8PM,** If you start server @ **8.10PM** then it will be fired. If you start server **@ 8.11PM** then it will not be fired.

Q: How to run CronJobs in specified cluster? (or) CronJob runs in all nodes / only 1 node? = **Only Single Node.**
Q: How to specify **Which Node CronJob** need to run?

```
CronJobModel myCronJob=modelService.create(CronJobModel.class);

// Set JobModel to CronJobModel & Set session attributes if required

    myCronJob.setNodeID(3);      modelService.save(myCronjob);
        cronJobService.performCronJob(myCronJob);
```

Note: - If we **don't set any Node Id** for CronJob then it can be executed by any Node within the cluster.

Q: How to set session related attributes to the CronJob? = Sometime CronJob logic requires session attributes like user, sessionLanguage & sessionCurrency etc. **2 ways to set: -**

- √ **Session attributes Using Impex**:-
  INSERT_UPDATE
  CronJob;code[unique=true];job(code);sessionUser(uid);sessionLanguage(**isocode**);sessionCurrency(**isocode**)
  ;ChennaCronJob;chennaJob;**user1**;**en**;**EUR**
  Note: - These session values can be used while writing the logic inside Perform() method of Job
- √ **Session attributes Using Code** = Create instance of CronJobModel & set all session attributes then save Cron job model to DB. Then call **performCronJob**() method by passing the Cron job code.

```
CronJobModel chennaCronJob=modelService.create(CronJobModel.class);
// set JobModel to CronJobModel
chennaCronJob.setSessionUser(chennaSessionUserModel);
chennaCronJob.setSessionLanguage(chennaSessionLanguage);
chennaCronJob.setSessionCurrency(chennaSessionCurrency);
modelService.save(chennaCronjob);
cronJobService.performCronJob(chennaCronJob);
```

Q: How to Abort the CronJob? =

Abort / Terminate a running CronJob ? = Let our CronJob is running from last 2 hour, now you want to abort it, then we can do that in Hybris. Job is Abortable only if 1 of below satisfied: -

√ **(1) isAbortable**() method in CronJob performable class should be **overridden** to **return true**.

√ **(2)** Define a new property in ***-spring.xml** file, where we defined our Job's bean definition

<property name="abortable" value="true"/>

Now Job is **abortable** and can be done using below code =

cronJobService.requestAbortCronJob(running_cronjob_code);

Note: - After aborting CronJob, it results should be results as **ERROR** & status as **ABORTED**

Note: - Execution of CronJob is taken care by **Trigger** as per schedule defined in **Trigger** using **Cron Expression**.

√ **Q1:** Who is responsible for Invoking this Trigger? **Q2**: Who evaluates the Cron Expression?

Task Engine = For every trigger there is always 1 task item gets created in Hybris.

It will keep on polling **Tasks** for every **X seconds** which configured in **local.properties** as:

cronjob.trigger.interval=30

By **default,** timer is set to **30 Seconds**. If you want to change it then do in "**local.properties**" as above.

Timer task fires a **DB query** to check for any triggers to be fired. If any matches / overdue then fires.

Note: - Don't give **too less** value until its essential (bcoz **DB calls** will be increased).

Task polled by **Task Engine** will check **Cron Expression** whether it matches / exceeded the current time.

Q: What is the Role of Jalo Session in Hybris?

√ The **Jalo layer** in hybris is **deprecated**, **not** the **jalosession**.

√ **Whenever a request** is made to hybris server, it may **need** current user details, currency, language, time zone etc to serve that request efficiently. Since **HttpSession does not** hold all these details, hybris came up with the concept of **JaloSession**.

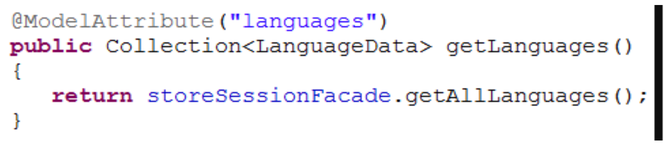
√ Whenever a request comes to Hybris, the filter **HybrisInitFilter** creates an object of **JaloSession**. **Every JaloSession object** is associated with a **SessionContext object**, which has current user, language, currency etc and the **current httpSession** object.

- o Cron Jobs also run in a **JaloSession**.
- o **JaloSession** is bound to a **tenant**.
  This cannot be changed, after the instance of JaloSession is created.
- o **JaloSession** is **never** made **persistent** in database.

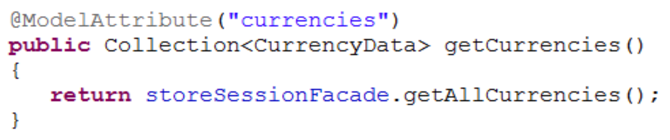
Q: Explain Model Attributes?

√ Some time it is necessary to get few data in many JSPs, and we don't want to pass them as a part of DTO (data objects). For example, the titles (Mr and Mrs etc). They can be used in many JSPs and tags file, like registration, delivery address etc.

√ In such cases, what we do is, create a convenient method in Abstract controllers, and use model attribute annotation for them. In this way they are available from all JSPs directly using model attribute.



```
@ModelAttribute("languages")
public Collection<LanguageData> getLanguages()
{
    return storeSessionFacade.getAllLanguages();
}
```



```
@ModelAttribute("currencies")
public Collection<CurrencyData> getCurrencies()
{
    return storeSessionFacade.getAllCurrencies();
}
```

√ These can be accessed directly in JSPs or even tag files. like Current Language : **${currencies}**