

Team Learning Session 2 Meeting Notes: **QUARTZ JOBS AND LISTENERS**

TOPICS DISCUSSED:

S.NO	Topic
1	Quartz Jobs
2	Quartz Jobs display and Control from Core Tools UI
3	Purpose of the Jobs
4	Schedulers
5	QUARTZ jobs tables
6	Running Jobs locally, New Job, logs, Common Issues
7	CRON Jobs
8	Important Points about Jobs
9	Listeners

Date	05-13-2015
Topics Covered by:	Venkata Vanukuri
Document Compiled by:	Krishna Bodduluri
Document Version	1.0

QUARTZ JOBS:

Note: Framework team owns and manages all the Quartz Jobs. We don't own/manage the functionality perspective of jobs that are not related to Admin/Framework Team.

There were 7 total different quartz jobs as below. We can schedule jobs from UI too.

Quartz Scheduler			
Total Jobs Executing: 10			
Core Quartz	Audit Quartz	DocGen Quartz	Stipulation Quartz
ICMP Quartz	EventNotification Quartz	EventCapture Quartz	
Scheduler Details			
Scheduler Detail			
Scheduler Name	Scheduler Instance Id	Status	Action
AuditQuartzScheduler	hivra00a0069.wellsfargo.com1431525734612	Started	Pause
AuditQuartzScheduler	hivra00a0070.wellsfargo.com1431525740425	Started	Pause
AuditQuartzScheduler	hivra00a0071.wellsfargo.com1431525958036	Started	Pause

Quartz jobs can be accessed from Core Tools UI.

CORE Tools

Production Support

Edit Email Template

Deal Search

Deal Summary

Load UCA Status and History

Process Inspector

Hogan Purge Indicator Update

Invoke UCA

Invoke POD

Delete Tokens

User Group Attribute Sync

Refresh Dynamic Groups

Quartz Scheduler

Total Jobs Executing: 6

Core Quartz | Audit Quartz | DocGen Quartz | Stipulation Quartz | ICMP Quartz | EventNotification Quartz | EventCapture Quartz

Scheduler Details

Scheduler Name	Scheduler Instance Id	Status	Action
CoreQuartzScheduler	kvra00a0069.wellsfargo.com1431525727793	Started	Pause
CoreQuartzScheduler	kvra00a0070.wellsfargo.com1431525736093	Started	Pause
CoreQuartzScheduler	kvra00a0071.wellsfargo.com1431525964161	Started	Pause

Quartz Scheduler UI gives more info about Current Job Details. There is also a Refresh jobs option in the UI to get most current job info.

Current Job Details

Total Jobs Executing: 0

Refresh Current Jobs

Scheduler Instance ID	Job Name	Job Group Name	Fired Time	State	Action
-----------------------	----------	----------------	------------	-------	--------

We can Control (**PAUSE, RESUME, UNSCHEDULE, RESCHEDULE, CANCEL**) the jobs from UI.

Show Only Triggers with WAITING Status

Get Triggers

Search Trigger

Job Name	Trigger Name	Prev Fire Time	Next Fire Time	Trigger State
AuditEocDataCapture_Job	AuditEocDataCapture_Trigger	12-31-1969 05:59 PM CST	05-13-2015 09:10 AM CDT	WAITING

Pause | Resume | **Unschedule** | Reschedule | Cancel

Purpose of having the jobs:

Here are the few reasons to go with Quartz Scheduler Jobs

- Some services may go down sometimes. During the loan process, if the services are down, services are down, several tasks gets created. Even if one service is down, there can be several tasks gets created.
- To bulk complete several system exceptions.
- When service call is failed, System exception gets created for that. Prod support will looks into it. To handle bulk number of System exception tasks.
- To handle sending bulk quantity Emails etc..
- To Clear the Cache.
- Documents upload etc..

In the UI, we have Several Scheduler Groups. Each group can have 1 or more jobs. If we have to develop a new job, pick the right group as placeholder for the new job.

List Of Scheduler Groups			
List Of Scheduler Groups			
Group Name	Total Jobs	Total Triggers	
Deal Active User Update Scheduler Recurring Every Three Minutes Job	1	1	
DisableUserAccountsSchedulerGroup	1	1	
StipulationParameterUpdate	19	19	
Save Risk Decision Image	8	8	
SyncUserGrpAttrWithBPM_Job_Group	1	1	
SyncUserSecuDynGrp_Job_Group	1	1	
WelcomeCallTask_Job_Group	581	581	
Clear Cache Scheduler Recurring Every Four Hours Job	1	1	
DealActiveUserPurge_Job_Group	1	1	
UpdateUserAttributesTeamTemplateJobGroup	1	1	
RegoAsync_Job_Group	2	2	
Dynamic Refresh Daily Job	1	1	
AgentMaintenanceJobGroup	1	1	
Custom Authentication Failure Group	1	1	
OAS Disclosures Notification	53	53	

The “Scheduler Details” tab gives more info about Scheduler name and Instance where the job is running.

Scheduler Details			
Scheduler Details			
Scheduler Name	Scheduler Instance Id	Status	Action
EventCaptureQuartzScheduler	lvvra00a0069.wellsfargo.com1431525730016	Stopped	Resume
EventCaptureQuartzScheduler	lvvra00a0070.wellsfargo.com1431525738023	Stopped	Resume
EventCaptureQuartzScheduler	lvvra00a0071.wellsfargo.com1431525956383	Stopped	Resume

Quartz job picks up emails for every 15 mins and runs the email job.

QUARTZ JOBS TABLES

- We have separate schema in data base for managing the jobs, schedulers, triggers etc..
- The Schema name itself “QUARTZ”.



- QUARTZ_TRIGGERS is the key table and contains all the triggers, Scheduler Names, Job Name, Start time, End time etc.

Connections x Reports x

search.sql x invalid_routing.sql x temp_sqli.sql x temp1.sql x temp2.sql x ods_core2.sql x AU_hierarchy.sql x alter_seq

Columns Data Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL

Sort.. Filter

SCHED_NAME	TRIGGER_NAME
1 EventNotificationScheduler	EventNotificationScheduler trigger
2 CoreQuartzScheduler	NotificationEventQueue_fa232254-2719-49ee-99c1-804600a4366ftrigger
3 CoreQuartzScheduler	NotificationEventQueue_13ee4b1-04c9-4119-807d-3907b0b93596trigger
4 CoreQuartzScheduler	Dynamic Refresh Trigger
5 EventNotificationDeleteProcessedRecordsScheduler	EventNotificationDeleteProcessedRecordsScheduler trigger
6 CoreQuartzScheduler	UpdateTitleContactJob_10ab8ec8-c2a0-49e8-baaa-098a65724d87_trigger
7 CoreQuartzScheduler	1000462301_StipulationParameterUpdate_834f5f10-7fbf-47fb-a223-a5655d9569c0_trigger
8 CoreQuartzScheduler	UpdateTitleContactJob_bbf9d7bc-a02b-4c58-97c8-c2629a642810_trigger
9 CoreQuartzScheduler	1000458621_StipulationParameterUpdate_815131af-ca40-4b65-9967-fe24dd7e7f06_trigger
10 CoreQuartzScheduler	17930025_1036058708_77371f24-e1cf-4b47-9763-9b8ad795ceb0trigger
11 CoreQuartzScheduler	1000462353_StipulationParameterUpdate_abd578a7-d6cb-405b-b64b-e9ee9a74ed2_trigger
12 CoreQuartzScheduler	1000462352_StipulationParameterUpdate_0517e806-ef92-47dd-86d0-c447711b8062_trigger
13 CoreQuartzScheduler	1000458621_StipulationParameterUpdate_58a735d0-525e-41f3-85bf-75f53d267220_trigger
14 CoreQuartzScheduler	1000458621_b3864ee0-acd6-11e4-bbfc-0050569122e7trigger
15 CoreQuartzScheduler	17927559_1036058607_a1095f3b-bcaf-436d-b651-86b8310c5ad2trigger
16 CoreQuartzScheduler	1000462301_StipulationParameterUpdate_572afad0-8fe8-43f0-a3f2-69958ef2598c_trigger
17 CoreQuartzScheduler	17929365_1036058750_lbcdfdb-73fa-4c3f-b6eb-0d91cd264571trigger
18 CoreQuartzScheduler	17930041_1036058718_a3bcfb2b-86f9-4a67-bd76-317d2002c41ettrigger
19 CoreQuartzScheduler	UpdateTitleContactJob_6db18413-0209-4b5e-8a4f-3b439518500_trigger
20 CoreQuartzScheduler	1000458621_StipulationParameterUpdate_150e3c5a-1ed5-48f2-bae9-52581d246d84_trigger
21 CoreQuartzScheduler	17927559_1036058722_d535c681-8034-4dd4-95ca-f878a9463a37trigger
22 CoreQuartzScheduler	UpdateTitleContactJob_b5bc0f7b-5d3d-46ce-85d5-5a2e042c277_trigger
23 CoreQuartzScheduler	UpdateTitleContactJob_5551f376-5a53-4b0d-ba3d-5e22ee289f6b_trigger
24 CoreQuartzScheduler	UpdateTitleContactJob_f10054e5-8011-44ed-b383-5b028e1c9123_trigger
25 CoreQuartzScheduler	Deal Active User Update Scheduler Trigger
26 CoreQuartzScheduler	17930082_1036058871_97dbde52-d115-4b2c-39da-8dcdb015f3e8trigger
27 CoreQuartzScheduler	Deal Active User Purge Scheduler Trigger
28 CoreQuartzScheduler	17930074_1036058869_c10e660b-26b0-40a2-8861-16bfe20185ecttrigger
29 CoreQuartzScheduler	1000458617_StipulationParameterUpdate_9e633cbe-d3c9-4d20-9543-f5c12ed6ddc_trigger
30 CoreQuartzScheduler	1000462301_StipulationParameterUpdate_3829f5fa-9ad5-45aa-b079-74d87ed30d8f_trigger
31 CoreQuartzScheduler	1000462301_StipulationParameterUpdate_feca027-c6f4-4958-a5f0-933dfe8e20f_trigger
32 CoreQuartzScheduler	1000458616_StipulationParameterUpdate_affe4ae4-e5a8-42e5-9522-ab055c1326b_trigger
33 CoreQuartzScheduler	UpdateTitleContactJob_fa900cf2-3cd4-4947-bbec-852b714ffa33_trigger
34 CoreQuartzScheduler	1000462351_StipulationParameterUpdate_0cc34d2a-7567-46c2-a47b-2fc1c7019ee2_trigger
35 CoreQuartzScheduler	UpdateTitleContactJob_2ef25d45-c214-452b-a8a1-3850bb0da03f_trigger
36 CoreQuartzScheduler	UpdateTitleContactJob_f061dfe-5739-4793-a97b-930c817eeeb7_trigger
37 CoreQuartzScheduler	UpdateTitleContactJob_4c9ebae2f-eed8-47be-a677-aeb991b7b947_trigger
38 CoreQuartzScheduler	WelcomeCallTask_Trigger_1000453296
39 CoreQuartzScheduler	17646662_CustomerOnlineAccessFailed_null_7e432d7d-861d-4d89-904f-8f438f35827attrigger
40 CoreQuartzScheduler	17789950_CustomerOnlineAccessFailed_null_f103d6a9-8d57-4bb5-a719-eddc9bb36eeettrigger
41 CoreQuartzScheduler	17646662_CustomerOnlineAccessFailed_null_d295a0d9-a9c3-43f3-b651-6157ad4632ettrigger

Tables (Filtered)

- QRTZ_BLOB_TRIGGERS
- QRTZ_CALENDARS
- QRTZ_CRON_TRIGGERS
- QRTZ_CMSR02_TRIGGERS
- QRTZ_JOB_DETAILS
- QRTZ_LOCKS
- QRTZ_PAUSED_TRIGGER_GRP
- QRTZ_SCHEDULER_STATE
- QRTZ_SIMPLE_TRIGGERS
- QRTZ_TRIGGER
- QRTZ_TRIGGERS

- For each job, there should be each scheduler instance. Each job is supposed to run in each scheduler.
- In the cases where Server goes down or Job was down, Scheduler is smart enough to run the job again when it is available. Scheduler will restart all the jobs when the server came back up.
- If I want to re-run the job at specific time, I can go to QRTZ_TRIGGERS table and update the column "NEXT_FIRE_TIME"

Connections x Reports x

search.sql x invalid_routing.sql x temp_sqli.sql x temp1.sql x temp2.sql x ods_core2.sql x AU_hie

Columns Data Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL

Sort.. Filter

NEXT_FIRE_TIME	PREV_FIRE_TIME	PRIORITY
1 sup	1431541260000	5 NA
2 19ee-99c1-804600a4366ftrigger	-1	5 NA
3 1119-807d-3907b0b93596trigger	-1	5 NA
4	1422310320000	5 NA
5 1e5Scheduler trigger group	1422310320000	5 NA
6 1e5-baaa-098a65724d87_trigger	1422310320000	5 NA
7 834f5f10-7fbf-47fb-a223-a5655d9569c0_trigger	1422573291891	-1
8 58-97c8-c2629a642810_trigger	1422573305548	-1
9 815131af-ca40-4b65-9967-fe24dd7e7f06_trigger	1422573322670	-1
10 7-9763-9b8ad795ceb0trigger	1422573693044	-1
11 abd578a7-d6cb-405b-b64b-e9ee9a74ed2_trigger	1422574507551	-1
12 0517e806-ef92-47dd-86d0-c447711b8062_trigger	1422574507551	-1
13 58a735d0-525e-41f3-85bf-75f53d267220_trigger	1422574507551	-1
14 10569122e7trigger	1422574507551	-1
15 1-b651-86b8310c5ad2trigger	1422574507551	-1
16 572afad0-8fe8-43f0-a3f2-69958ef2598c_trigger	1422574507551	-1
17 1-b6eb-0d91cd264571trigger	1422574507551	-1
18 7-bd76-317d2002c41ettrigger	1422574507551	-1
19 58e-8a4f-436439518500_trigger	1422574507551	-1
20 150e3c5a-1ed5-48f2-bae9-52581d246d84_trigger	1422574507551	-1
21 1-95ca-f878a9463a37trigger	1422574507551	-1
22 1ce-85d5-5a2e042c277_trigger	1422574507551	-1
23 30d-ba3d-5e22ee289f6b_trigger	1422574507551	-1
24 1ed-b383-5b028e1c9123_trigger	1422574507551	-1
25 1rring Every Three Minutes	1422574507551	-1
26 1rring Every Two Hours	1422574507551	-1
27 1rring Every Two Hours	1422574507551	-1
28 1-8861-16bfe20185ecttrigger	1422574507551	-1
29 9e633cbe-d3c9-4d20-9543-f5c12ed6ddc_trigger	1422574507551	-1
30 3829f5fa-9ad5-45aa-b079-74d87ed30d8f_trigger	1422574507551	-1
31 feca027-c6f4-4958-a5f0-933dfe8e20f_trigger	1422574507551	-1
32 affe4ae4-e5a8-42e5-9522-ab055c1326b_trigger	1422574507551	-1
33 347-bbec-852b714ffa33_trigger	1422574507551	-1
34 0cc34d2a-7567-46c2-a47b-2fc1c7019ee2_trigger	1422574507551	-1
35 52b-a8a1-3850bb0da03f_trigger	1422574507551	-1
36 793-a97b-930c817eeeb7_trigger	1422574507551	-1
37 7be-a677-aeb991b7b947_trigger	1422574507551	-1
38	1422574507551	-1
39 111_7e432d7d-861d-4d89-904f-8f438f35827attrigger	1422574507551	-1

Tables (Filtered)

- QRTZ_BLOB_TRIGGERS
- QRTZ_CALENDARS
- QRTZ_CRON_TRIGGERS
- QRTZ_CMSR02_TRIGGERS
- QRTZ_JOB_DETAILS
- QRTZ_LOCKS
- QRTZ_PAUSED_TRIGGER_GRP
- QRTZ_SCHEDULER_STATE
- QRTZ_SIMPLE_TRIGGERS
- QRTZ_TRIGGER
- QRTZ_TRIGGERS

Views

- Editing Views
- Indexes
- Packages
- Procedures
- Functions
- Queues
- Queues Tables
- Queues

Running Jobs locally:

- core-quartz.properties file is important. This file has the entire configuration to make the job to run locally. Since we don't have local database to manage/run the jobs. For local jobs, we configured to run in-memory. If we want to run local jobs, we should go for in memory. RAM job store uses in memory

Name	Date modified	Type	Size
concurrency.properties	5/12/2015 11:32 PM	PROPERTIES File	
core-quartz.properties	5/12/2015 11:32 PM	PROPERTIES File	
coretools.properties	5/12/2015 11:32 PM	PROPERTIES File	
DataChangeDisplay.properties	5/12/2015 11:32 PM	PROPERTIES File	
deal-quartz.properties	5/12/2015 11:32 PM	PROPERTIES File	
DeliveryQueries.properties	5/12/2015 11:32 PM	PROPERTIES File	

```
core-quartz.properties
10 org.quartz.scheduler.instanceName = CoreQuartzScheduler
11 org.quartz.scheduler.instanceId = AUTO
12 #org.quartz.scheduler.batchTriggerAcquisitionMaxCount = 40
13 #org.quartz.scheduler.batchTriggerAcquisitionFireAheadTimeWindow = 300000
14 #org.quartz.jobStore.acquireTriggersWithinLock = true
15 #=====
16 # Configure ThreadPool
17 #=====
18 org.quartz.threadPool.class = org.quartz.simpl.SimpleThreadPool
19 org.quartz.threadPool.threadCount = 10
20 org.quartz.threadPool.threadPriority = 5
21 #=====
22 # Configure JobStore
23 #=====
24 org.quartz.jobStore.misfireThreshold = 3540000
25 org.quartz.jobStore.class = org.quartz.simpl.RAMJobStore
26 #org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate
```

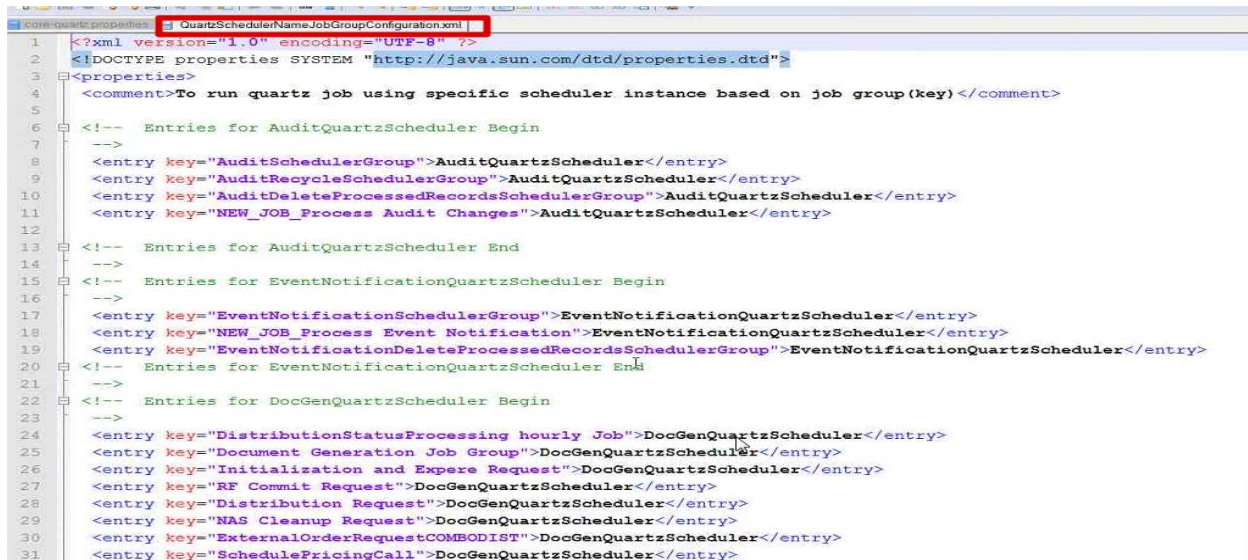
- Setting needs to be done to run the job local: **CoreActivatorImpl-> localQuartzEnableStatus = "true"** to run the job local.

```
*CoreActiva... NewQuartzSc... QuartzSched... AsyncProces... QuartzDelega...
120 @Autowired
121 protected RegisterEventListenerService registerEventListenerService;
122
123
124 private static Logger LOG = Logger.getLogger(CoreActivatorImpl.class);
125
126
127 private long startTime = 0;
128 private String appName = "Quartz";
129 private String localquartzEnableStatus = "true";
130 private boolean isStarted = false;
131
```


New Jobs:

- How do I know what scheduler to use for my new job?

We do that through configuration. QuartzSchedulerNameJobGroupConfiguration.xml. This configuration file has all the info about schedulers etc..



```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3 <properties>
4 <comment>To run quartz job using specific scheduler instance based on job group(key)</comment>
5
6 <!-- Entries for AuditQuartzScheduler Begin -->
7 -->
8 <entry key="AuditSchedulerGroup">AuditQuartzScheduler</entry>
9 <entry key="AuditRecycleSchedulerGroup">AuditQuartzScheduler</entry>
10 <entry key="AuditDeleteProcessedRecordsSchedulerGroup">AuditQuartzScheduler</entry>
11 <entry key="NEW_JOB_Process Audit Changes">AuditQuartzScheduler</entry>
12
13 <!-- Entries for AuditQuartzScheduler End -->
14 -->
15 <!-- Entries for EventNotificationQuartzScheduler Begin -->
16 -->
17 <entry key="EventNotificationSchedulerGroup">EventNotificationQuartzScheduler</entry>
18 <entry key="NEW_JOB_Process Event Notification">EventNotificationQuartzScheduler</entry>
19 <entry key="EventNotificationDeleteProcessedRecordsSchedulerGroup">EventNotificationQuartzScheduler</entry>
20 <!-- Entries for EventNotificationQuartzScheduler End -->
21 -->
22 <!-- Entries for DocGenQuartzScheduler Begin -->
23 -->
24 <entry key="DistributionStatusProcessing hourly Job">DocGenQuartzScheduler</entry>
25 <entry key="Document Generation Job Group">DocGenQuartzScheduler</entry>
26 <entry key="Initialization and Expere Request">DocGenQuartzScheduler</entry>
27 <entry key="RF Commit Request">DocGenQuartzScheduler</entry>
28 <entry key="Distribution Request">DocGenQuartzScheduler</entry>
29 <entry key="NAS Cleanup Request">DocGenQuartzScheduler</entry>
30 <entry key="ExternalOrderRequestCOMBODIST">DocGenQuartzScheduler</entry>
31 <entry key="SchedulePricingCall">DocGenQuartzScheduler</entry>
```

Logs:

- Quartz logs will be on quartz instances.DEV8- 803 is a QUARTZ_SERVER.
- All the BPM's runs on WAS servers.

Possible Issues:

- Jobs configured to run at 2.00, ran at 8.00 AM.
- Most of the time, the above issue might be because of server was down.

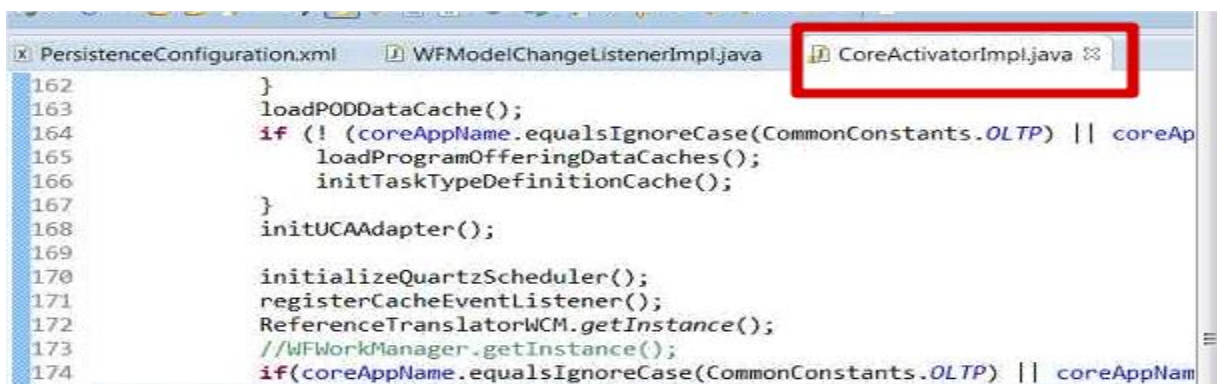
CRON JOBS:

- CRON Jobs are for processing repeating tasks at regular intervals.
- These can be scheduled in DB with in QRTZ_CRON_TRIGGERS table.
- CRON jobs get started after server gets started.
- Between Simple vs CRON jobs- it's just a flag that makes the difference.
- The cron jobs are used when the jobs needs to be repeated. These jobs are not removed from DB as long as there is a future execution scheduled. We typically create cron jobs during server startup.

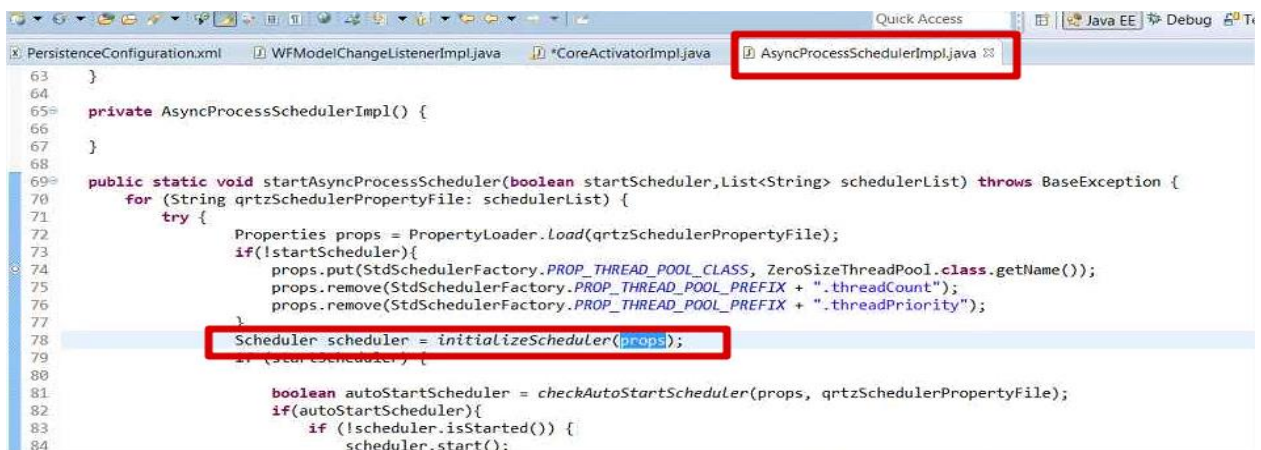


Important Points:

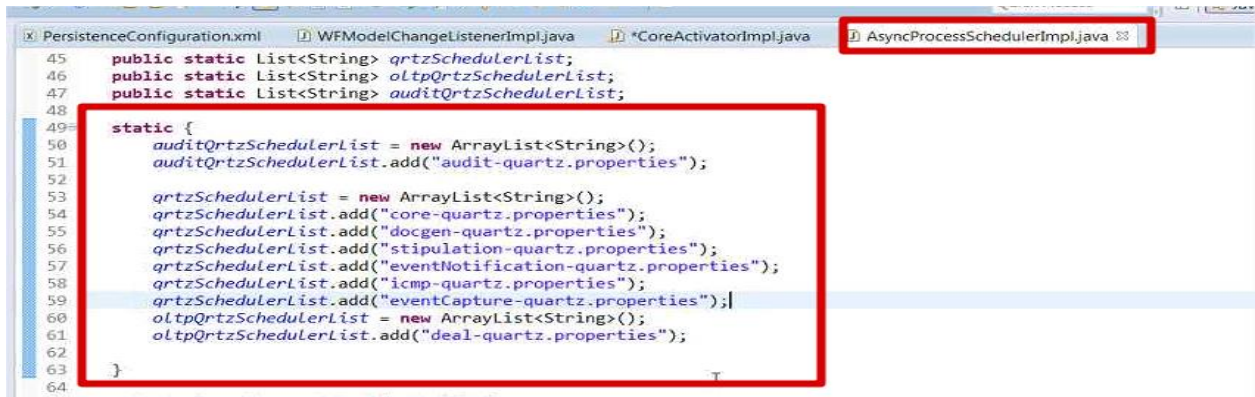
- During server start up itself, we start quartz scheduler.
- CoreActivatorImpl.java is the key program with respect to Quartz jobs.



- We are using plain Quartz for the jobs, but not the Spring-Quartz.
- For all Jboss schedulers, we initialize all the 7 properties files. Properties are nothing but, quartz properties for initialization start up.



- Below is the code where, we specify the properties for each high level Quartz job.

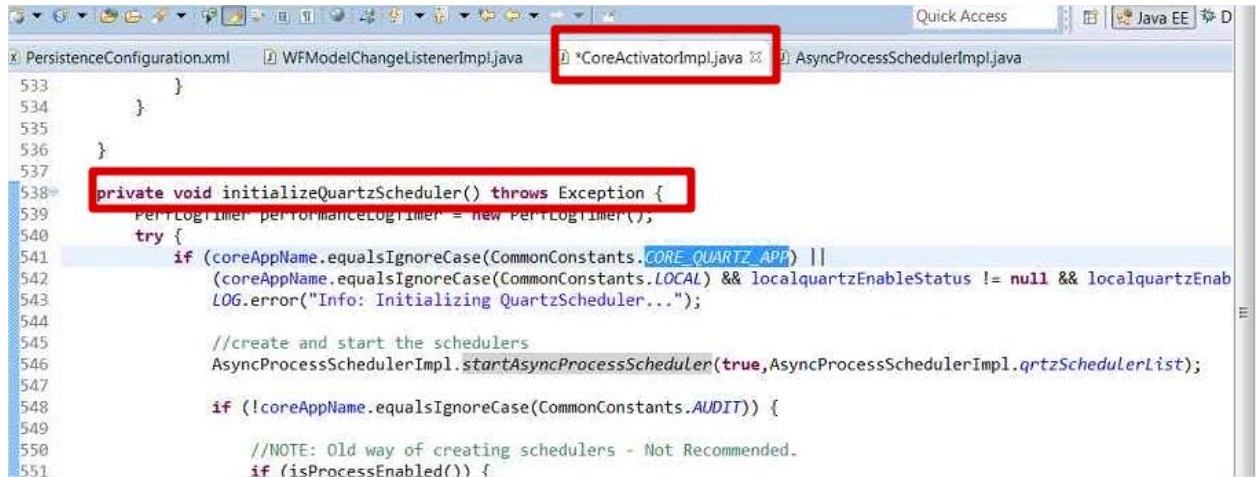


```

45 public static List<String> qrtzSchedulerList;
46 public static List<String> oltpQrtzSchedulerList;
47 public static List<String> auditQrtzSchedulerList;
48
49 static {
50     auditQrtzSchedulerList = new ArrayList<String>();
51     auditQrtzSchedulerList.add("audit-quartz.properties");
52
53     qrtzSchedulerList = new ArrayList<String>();
54     qrtzSchedulerList.add("core-quartz.properties");
55     qrtzSchedulerList.add("docgen-quartz.properties");
56     qrtzSchedulerList.add("stipulation-quartz.properties");
57     qrtzSchedulerList.add("eventNotification-quartz.properties");
58     qrtzSchedulerList.add("icmp-quartz.properties");
59     qrtzSchedulerList.add("eventCapture-quartz.properties");
60     oltpQrtzSchedulerList = new ArrayList<String>();
61     oltpQrtzSchedulerList.add("deal-quartz.properties");
62
63 }
64

```

- Below is the Initialization part of Quartz jobs:



```

533     }
534 }
535
536 }
537
538 private void initializeQuartzScheduler() throws Exception {
539     PerfLogger perfLogger = new PerfLogger();
540     try {
541         if (coreAppName.equalsIgnoreCase(CommonConstants.CORE_QUARTZ_APP) ||
542             (coreAppName.equalsIgnoreCase(CommonConstants.LOCAL) && localquartzEnableStatus != null && localquartzEnableStatus)) {
543             LOG.error("Info: Initializing QuartzScheduler...");
544
545             //create and start the schedulers
546             AsyncProcessSchedulerImpl.startAsyncProcessScheduler(true, AsyncProcessSchedulerImpl.qrtzSchedulerList);
547
548             if (!coreAppName.equalsIgnoreCase(CommonConstants.AUDIT)) {
549                 //NOTE: Old way of creating schedulers - Not Recommended.
550                 if (isProcessEnabled()) {
551

```

- We can stop specific job with single property change, with in the code.
- Quartz is available in all the nodes. So scheduling can be done on any nodes(JBoss).
- Quartz is initialized in all the servers(including OLTP) but only started on QUARTZ/TIER2 cluster.
- For some reason, OLTP is up but QUARTZ/TIER2 is down then the job is scheduled but executed only when Quartz is started.
- Same is true when QUARTZ/TIER2 is up but Quartz scheduler is stopped/paused.
- In a normal scenario for a simple job, The job is executed almost immediately and removed from DB once executed if the job is scheduled to run with current timestamp. The jobs scheduled for future execution are retained until the execution timestamp.
- When we display schedules in UI, we are trying to get node info from JVM. We go to JVM to get the status of Instance that running the job, not from DB. For this, we are using EJB's.
- We initially used EJB remote calls to get the status of node and node status. But recently they replaced EJB calls with HTTP requests

```
PersistenceConfiguration.xml  WFModelChangeListenerImpl.java  CoreActivatorImpl.java  NewQuartzSchedulerBusinessServiceImpl.java
775     appliedUrl = quartzUrls;
776 }
777 String parameters = buildParameterString(request);
778 for(String url : appliedUrl){
779     try {
780         Request httpReq = HttpFactory.INSTANCE.createRequest();
781         httpReq.setHttpMethod(HttpMethod.POST);
782
783         String urlWithParams = url + "/servlet/quartzSchedulerServlet" + "?" + parameters;
784         httpReq.setUrl(urlWithParams);
785         httpReq.setRequestHeader("Content-type", "application/json");
786         //String json = JSONUtil.convertToJSON(request);
787         //HttpURLConnection httpConn = new HttpURLConnection(urlWithParams, httpReq); does not support to read request content at this time
788         Response response = httpSrv.execute(httpReq);
789         if(response != null && String.valueOf(response.getResponseBody()) != null){
790             QuartzSchedulerResponse result = JSONUtil.convertJSONToObject(response.getResponseBody(), QuartzSchedulerResponse.class);
791             if(request.getUserName() != null){
792                 result.setEnvironmentAdministrator(isEnvironmentAdmin(request.getUserName()));
793             }
794             result.setTargetUrl(url);
795             retResponseList.add(result);
796         }
797     } catch (Exception e) {
798         LOG.error("Error during making request to [" + url + "]", e);
799     }
800 }
801 return retResponseList;
802 }
```

- **QuartzSchedulerBusinessServiceImpl.java** is also main class- **lookupSchedulerList** method is one of the key methods. If we look at **getAsyncProcessScheduler()**- instance name gets populated from this method.
- Get the URL list from Properties files and loop over the list of URL's. But the idea is to read from JVM.

```
WFModelChang...  CoreActiva...  NewQuartzSc...  QuartzSched...  AsyncProces...  QuartzDelega...  AsyncProces...  AsyncProces...
828     if(request.getSchedulerStatus() != null){
829         params.append("&schedulerStatus=" + request.getSchedulerStatus());
830     }
831     if(request.getSchedulerInstanceId() != null){
832         params.append("&schedulerInstanceId=" + request.getSchedulerInstanceId());
833     }
834     if(request.getGroupStatus() != null){
835         params.append("&groupStatus=" + request.getGroupStatus());
836     }
837     if(request.getListOfGrps() != null){
838         params.append("&listOfGrps=" + listToString(request.getListOfGrps()));
839     }
840     return params.toString().substring(1).replace(" ", "%20");
841 }
842 private HttpService getHttpService() {
843     if (httpService == null) {
844         HttpServiceUtil httpUtil = new HttpServiceUtil();
845
846         HttpServiceConfig httpServiceConfig = HttpFactory.INSTANCE.createHttpService();
847         httpServiceConfig.setConnectionTimeout(httpUtil.getTimeout());
848         httpServiceConfig.setMaxTotalConnections(httpUtil.getMaxTotalConnections());
849         httpServiceConfig.setDefaultMaxConnectionsPerHost(httpUtil.getMaxConnectionsPerHost());
850         httpServiceConfig.setStaleCheckingEnabled(httpUtil.isConnectionStaleCheckingEnabled());
851
852         HttpService service = new HttpServiceImpl();
853         service.setServiceId("QuartzMgtService");
854         httpService = service;
855         httpService.setHttpServiceConfig(httpServiceConfig);
856     }
857 }
```

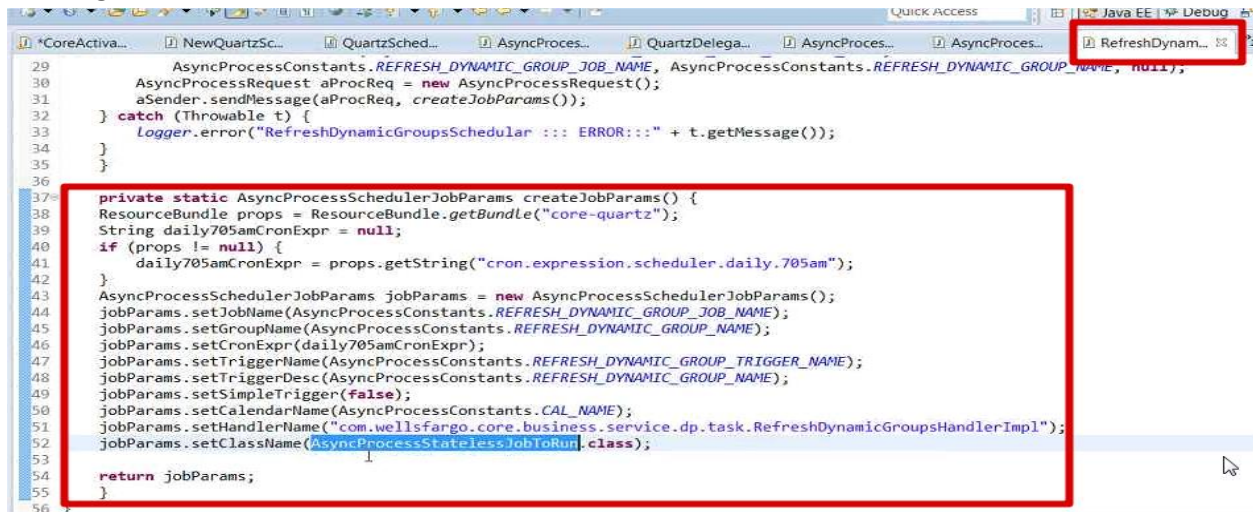
```
PersistenceConfiguration.xml  WFModelChangeListenerImpl.java  *CoreActivatorImpl.java  NewQuartzSchedulerBusinessServiceImpl.java  33
760     }
761     }
762     return schedulerTypeProperty;
763 }
764
765 @Override
766 public List<QuartzSchedulerResponse> processRequest(QuartzSchedulerRequest request) {
767     List<QuartzSchedulerResponse> retResponseList = new ArrayList<QuartzSchedulerResponse>();
768     getQuartzUrls();
769     HttpService httpSrv = getHttpService();
770     List<String> appliedUrl = null;
771     if(request.getTargetUrl() != null) {
772         appliedUrl = new ArrayList<String>();
773         appliedUrl.add(request.getTargetUrl());
774     } else {
775         appliedUrl = quartzUrls;
776     }
777     String parameters = buildParameterString(request);
778     for(String url : appliedUrl){
779         try {
780             Request httpReq = HttpFactory.INSTANCE.createRequest();
781             httpReq.setHttpMethod(HttpMethod.POST);
782
783             String urlWithParams = url + "/servlet/quartzSchedulerServlet" + "?" + parameters;
784             httpReq.setUrl(urlWithParams);
785             httpReq.setRequestHeader("Content-type", "application/json");
786             //Calling json = JSONUtil.toJson(HTTPRequest);
787             //HttpReq.setDestinationData(json); // send the data back to the client
788         } catch (Exception e) {
789             log.error("Error in processRequest: " + e.getMessage());
790         }
791     }
792     return retResponseList;
793 }
```

- All the jobs we start during server start up is **CRON** jobs.
- Only emailing is STATEFUL job. All the other jobs are stateless.

Key Service:

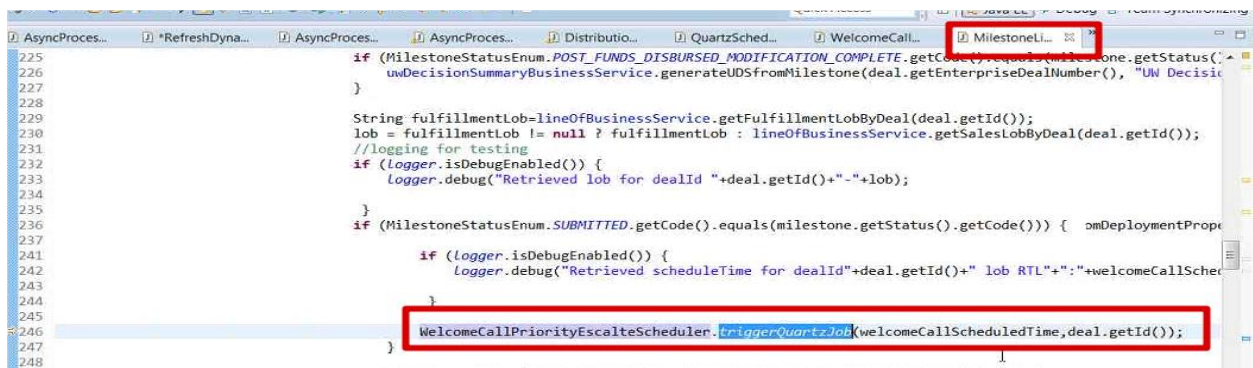
```
WFModelChange...  *CoreActiva...  NewQuartzSc...  QuartzSched...  AsyncProces...  QuartzDelega...  AsyncProces...  AsyncProces...  33
1 package com.wellsfargo.core.sharedlib.delegator.appadmin;
2
3
4 public interface AsyncProcessSchedulerLocal {
5
6     public String getQuartzUrls();
7
8     public String getSchedulerList(String schedulerType) ;
9
10    public void changeSchedulerStatus(String jsonString);
11
12    public void pauseAllSchedulers(String jsonString);
13
14    public void interruptSchedulerJob(String jsonString);
15
16
17    //Quartz Scheduler production support tool
18    public String loadSchedulerGroups(String jsonString);
19    public String loadJobsInAGroups(String jsonString);
20    public String loadGroupsInAScheuler(String jsonString);
21    public void pauseListOfSeletedGrps(String jsonString);
22    public void resumeListOfSeletedGrps(String jsonString);
23    public void pauseListOfSeletedJobs(String jsonString);
24    public void resumeListOfSeletedJobs(String jsonString);
25    public void unScheduleListOfSeletedJobs(String jsonString);
26    public void reScheduleListOfSeletedJobs(String jsonString);
27    public String processRequest(String jsonString);
28 }
```


Creating Job Params:



```
29 AsyncProcessConstants.REFRESH_DYNAMIC_GROUP_JOB_NAME, AsyncProcessConstants.REFRESH_DYNAMIC_GROUP_NAME, null);
30 AsyncProcessRequest aProcReq = new AsyncProcessRequest();
31 aSender.sendMessage(aProcReq, createJobParams());
32 } catch (Throwable t) {
33     Logger.error("RefreshDynamicGroupsScheduler ::: ERROR:::" + t.getMessage());
34 }
35 }
36
37 private static AsyncProcessSchedulerJobParams createJobParams() {
38     ResourceBundle props = ResourceBundle.getBundle("core-quartz");
39     String daily705amCronExpr = null;
40     if (props != null) {
41         daily705amCronExpr = props.getString("cron.expression.scheduler.daily.705am");
42     }
43     AsyncProcessSchedulerJobParams jobParams = new AsyncProcessSchedulerJobParams();
44     jobParams.setJobName(AsyncProcessConstants.REFRESH_DYNAMIC_GROUP_JOB_NAME);
45     jobParams.setGroupName(AsyncProcessConstants.REFRESH_DYNAMIC_GROUP_NAME);
46     jobParams.setCronExpr(daily705amCronExpr);
47     jobParams.setTriggerName(AsyncProcessConstants.REFRESH_DYNAMIC_GROUP_TRIGGER_NAME);
48     jobParams.setTriggerDesc(AsyncProcessConstants.REFRESH_DYNAMIC_GROUP_NAME);
49     jobParams.setSimpleTrigger(false);
50     jobParams.setCalendarName(AsyncProcessConstants.CAL_NAME);
51     jobParams.setHandlerName("com.wellsfargo.core.business.service.dp.task.RefreshDynamicGroupsHandlerImpl");
52     jobParams.setClassName(AsyncProcessStatelessJobToRun.class);
53
54     return jobParams;
55 }
```

Sample Quartz call from Listener:



```
225
226
227
228
229 String fulfillmentLob=lineOfBusinessService.getFulfillmentLobByDeal(deal.getId());
230 lob = fulfillmentLob != null ? fulfillmentLob : lineOfBusinessService.getSalesLobByDeal(deal.getId());
231 //logging for testing
232 if (logger.isDebugEnabled()) {
233     logger.debug("Retrieved lob for dealId "+deal.getId()+"-"+lob);
234 }
235
236 if (MilestoneStatusEnum.SUBMITTED.getCode().equals(milestone.getStatus().getCode())) {
237     omDeploymentPrope
238
239     if (logger.isDebugEnabled()) {
240         logger.debug("Retrieved scheduleTime for dealId"+deal.getId()+" lob RTL"+":"+welcomeCallSched
241     }
242
243
244
245
246 WelcomeCallPriorityEscalateScheduler.triggerQuartzJob(welcomeCallScheduledTime,deal.getId());
247
248
```

LISTENERS:

- Whenever any events occurred on specific columns on a specific table, the listeners gets fired and will process the logic.
- All the configuration for listeners is under : **PersistenceConfiguration.xml**
- Listeners with Priority = 0 -> highest priority. Listeners with 0 will be executed first.
-
- Listeners with Priority = 100 -> lowest priority

```
serviceInterface="com.wellsfargo.core.extended.entity.Listener.audit.WFModelChangeListener" priority="100"
```

```
serviceInterface="com.wellsfargo.core.extended.entity.Listener.stipulation.StipulationDataChangeListener" priority="95"
```


Key Listener: WFModelChangeListener

A screenshot of a configuration field, likely from a web application, showing the value of the 'serviceInterface' property. The text is enclosed in a red rectangular border. The background is blurred, showing other UI elements.

```
serviceInterface="com.wellsfargo.core.extended.entity.listener.audit.WFModelChangeListener"
```