**This document will talk about how Jbeam is going to interact with application to execute application specific object.**

Jbeam is responsible for executing batch. But what does a batch means and how Jbeam interacts with an application to run the batch.

A batch is a set of Jobs to be executed. A Job can correspond to any single activity. For example, there can be a job for sending a reminder for a policy renewal on a particular day.

Jbeam interacts with the application for running a batch through JOB\_SCHEDULE table in the application schema. What I mean by application schema here is that the database schema that has the business/ transactional data.

Before we go forward on our discussion, please open ***job\_schedule\_data.xls*** file which is present in the same directory.

The table has columns like JOB\_SEQ, JOB\_NAME, JOB\_STATUS, EXECUTION\_DATE, CREATED\_ON, CREATED\_BY, DATE\_EXECUTED, PRIORITY\_CODE\_1, PRIORITY\_CODE\_2, LAST\_MODIFIED\_ON, LAST\_MODIFIED\_BY, PRE\_POST, ENTITY\_TYPE, LISTENER\_INDICATOR which are generic and most of them being used by Jbeam for execution of jobs. There are other columns which are specific to the business application that you are using Jbeam with. Here they are specific to Insurance BILLING and POLICY applications.

**(Note:** These business specific columns are referred by Jbeam through configurations in **COLUMN\_MAP** table which we will see later. This means, we can have our own table structure as far as business related columns are concerned and modify COLUMN\_MAP table accordingly.**)**

So while running any transaction or doing any activity through an application, whichever task an application wants to run as part of a batch, it can do it by putting an entry into the JOB\_SCHDULE table with the job\_name, execution time and other attributes.

Besides this, jbeam also allows you to define PRE and POST jobs. These are the jobs that are executed with every Day batch. (There are 2 types of batches. A DAY batch executes all the jobs available for execution till the defined date. And a special batch which only executes specified set of jobs.)

PRE jobs are provided for defining any jobs that can be used for initialization purpose before running any other jobs.

POST jobs as the name suggests, allows you to define set of jobs to be executed at the end of batch after all other jobs are executed.

**JOB Execution:**

JOB\_SCHEDULE table has entries for jobs to be executed. JOB\_SCHEDULE.JOB\_NAME tells Jbeam which job to execute and JOB\_SCHEDULE .EXECUTION\_DATE will tell when the job should be executed. So if you are running a batch for, say, 28th Nov 2014, all the jobs which have execution date less than or equal to 28th Nov and which are not already executed will be picked for execution (We will later see how this job is actually executed by Jbeam).

How the jobs defined in JOB\_SCHEDULE table are actually executed? **OBJECT\_MAP** table has the necessary information to run any particular job.

JOB\_NAME in JOB\_SCHEDULE table maps to ID column in OBJECT\_MAP. OBJECT\_MAP.OBJECT\_TYPE tells what execution handler to be used. And depending on OBJECT\_TYPE, it will use either OBJECT\_NAME or CASE data information for executing the job.

For e.g., here is the data for an OBJECT\_MAP row (only required columns included). Also shown are the entries in CONFIGURATION table for OBJECT\_TYPEs.

**OBJECT\_MAP**

|  |  |  |
| --- | --- | --- |
| **ID** | **OBJECT\_NAME** | **OBJECT\_TYPE** |
| ENTITY\_INTERFACE | com.Interface.EntityInterfaceClient.EntityInterfacePort\_Client | JV |

CONFIGURATION

|  |  |  |  |
| --- | --- | --- | --- |
| **CODE1** | **CODE2** | **CODE3** | **VALUE** |
| CORE | EXECUTION\_HANDLER | JV | com.stgmastek.core.logic.JAVAExecutionHandler |

1. ENTITY\_INTERFACE job.

ENTITY\_INTERFACE job will use JAVAExecutionHandler which internally uses object name (com.Interface.EntityInterfaceClient.EntityInterfacePort\_Client). So in this case, JAVAExecutionHandler will instantiate EntityInterfacePort\_Client through reflection and complete its intended work.

**Entities (JOB Categories):**

Each job is categorized in one of the entities like POLICY, ACCOUNT, BROKER, GENERAL, PRE, POST, etc. (You can define your own entities).

COLUMN\_MAP table has the necessary information for executing an Entity. How the information is used is explained below.

**COLUMN\_MAP:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ENTITY** | **LOOKUP\_COLUMN** | **LOOKUP\_VALUE** | **VALUE\_COLUMN** | **PRECEDENCE ORDER** | **ON\_ERROR\_FAIL\_ALL** |
| POLICY | CONTEXT | P | REFERENCE\_ID#SUB\_REFERENCE\_ID | 2 | Y |
| POLICY | POLICY\_NO | NULL | POLICY\_NO#POLICY\_RENEW\_NO | 2 | Y |
| ACCOUNT | ENTITY\_TYPE | ACCOUNT | ACCOUNT\_SYSTEM\_CODE | 4 | Y |
| BROKER | ENTITY\_TYPE | BROKER | BROKER\_SYSTEM\_CODE | 5 | Y |
| PRE | PRE\_POST | PRE | PRIORITY\_CODE\_1 | 1 | NULL |
| POST | PRE\_POST | POST | PRIORITY\_CODE\_1 | 999 | NULL |
| GENERAL | JOB\_SEQ | NULL | JOB\_SEQ | 6 | NULL |

All the jobs besides PRE and POST are created by application and put in JOB\_SCHEDULE table. The order of execution of entities is defined in PRECEDENCE\_ORDER column. Here all PRE jobs will be executed first, then the policy, .. and POST being the last.

LOOKUP\_COLUMN, LOOKUP\_VALUE together define how an object should be picked up from JOB\_SCHEDULE table. For example, when it is the turn of ACCOUNT entity to run, JBEAM will pick all jobs with ENTITY\_TYPE as ACCOUNT. You can see the *com.stgmastek.core.util.QueryGenerator class* in either (oracle/mssql/postgre) of the plugin jars of JBEAM-CORE.

When a **SPECIAL batch** is run you need to mention the values of columns defined in VALUE\_COLUMN. For e.g., if you need to run a PRE job as a SPECIAL batch, you need to mention the PRIORITY\_CODE\_1 of the PRE job. So to run a PRE job with priority\_code\_1 as 6, you will mention entity as PRE and value as 6.

Similarly to run a SPECIAL policy batch, you will specify POLICY as entity and POLICY\_NO#POLICY\_RENEW\_NO (12345#1, where 12345 is policy no and 1 is policy renew no) as the value.

**PRE and POST jobs:**

PRE and POST jobs are handled in different way than other entities. These jobs are not created by application in the JOB\_SCHEDULE table. Whenever a DAY batch runs, these jobs are pro-created by JBEAM and put into JOB\_SCHEDULE table for execution (The definition for these jobs exists in META\_DATA table of CORE schema) Which means, for every DAY batch, all the PRE and POST jobs will be executed once. PRE jobs will be the first to execute and POST jobs the last.

**Priorities within entities:**

If COLUMN\_MAP table defines the order for entities what defines the order for job execution within an entity. ORDERBY\_MAP defines the order in which jobs associated with an entity should be executed. For e.g., currently we have priorities defined for

**ORDERBY\_MAP:**

|  |  |
| --- | --- |
| **ENTITY** | **ORDER\_BY\_COLUMN** |
| ACCOUNT | account\_system\_code, priority\_code\_1, priority\_code\_2 |
| POLICY | policy\_no, priority\_code\_1, priority\_code\_2 |

**Sample Data for META\_DATA and OBJECT\_MAP:**

**META\_DATA table: (Only important columns included. See meta\_data\_data.xls for complete details.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SEQ\_NO** | **OBJECT\_NAME** | **PRE\_POST** | **JOB\_TYPE** | **PRIORITY\_CODE\_1** | **PRIORITY\_CODE\_2** |
| 7 | EFT\_SUBMISSION | POST | FE | 3 | 0 |
| 2 | OUTPUT\_FORMS\_SCHEDULE\_JOB | POST | JV | 1 | 0 |
| 21 | REFUNDSUBMISSION\_EFT | POST | FE | 17 | 0 |
| 20 | AP\_INTERFACE | POST | FE | 4 | 0 |
| 14 | POLICY\_STATUS\_INTERFACE | POST | FE | 8 | 0 |
| 16 | GL\_INTERFACE | POST | FE | 12 | 0 |
| 17 | OFS\_MERGE\_XML\_JOB | POST | JV | 13 | 0 |
| 22 | REFUNDSUBMISSION\_CREDITCARD | POST | FE | 18 | 0 |
| 12 | RCAN\_INTERFACE | POST | FE | 10 | 0 |
| 1 | REFRESH\_BUSINESS\_DAY | PRE | FE | 1 | 0 |
| 9 | CLOSE\_BUSINESS\_DAY | PRE | FE | 6 | 0 |
| 10 | ACCOUNTING\_YEAR\_MONTH | PRE | FE | 7 | 0 |

Here**3** **PRE** jobs and **9** **POST** jobs are defined. While executing any DAY batch, first entries for 3 PRE jobs will be created/ put in JOB\_SCHEDULE table and they will be executed in the order of PRIORITY\_CODE\_1.

Similarly, when PRE and all other jobs for the day complete execution, entries for 9 POST jobs will be created JOB\_SCHEDULE table, and executed in the order of PRIORITY\_CODE\_1.

**OBJECT\_MAP table: (Only important columns included. See object\_map\_data.xls for complete details)**

|  |  |  |
| --- | --- | --- |
| **ID** | **OBJECT\_NAME** | **OBJECT\_TYPE** |
| GENERATE\_FILE\_NAME | com.stgmastek.core.print.GenerateFileName | JV |
| OUTPUT\_FORMS\_SCHEDULE\_JOB\_BILLING | com.stgmastek.core.print.OutputFormsScheduleBatchJobBilling | JV |
| PREMIUM\_INTERFACE | com.Interface.PremiumInterfaceClient.PremiumInterfacePort\_Client | JV |
| ENTITY\_INTERFACE | com.Interface.EntityInterfaceClient.EntityInterfacePort\_Client | JV |
| OUTPUT\_FORMS\_SCHEDULE\_JOB\_POLICY | com.stgmastek.core.print.OutputFormsScheduleBatchJobPolicy | JV |
| OFS\_MERGE\_XML\_JOB | com.stgmastek.core.merge.OFSMergeXmlBatchJob | JV |
| OUTPUT\_FORMS\_SCHEDULE\_JOB | com.stgmastek.core.print.OutputFormsScheduleBatchJob | JV |

Here for job\_name GENERATE\_FILE\_NAME, JavaExecutionHandler will instantiate com.stgmastek.core.print.GenerateFileName and run the required job.