

Yugandhar Open MDM Hub

Code Generation Guide

Yugandhar Open MDM Hub Release – V2.0.0

Date – 11/06/2018

+++++

Copyright [2017] [Yugandhar Open MDM Hub]

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

Contents

About Yugandhar Project.....	3
About this document	3
Import code generator project	4
Database drivers	5
Understanding the code generator project artifacts.....	5
Hibernate configuration artifacts.....	5
Hibernate Reverse Engineering artifacts.....	5
Hibernate Console Configurations (Launch Configuration).....	5
Code Generation Configurations:.....	6
Code generation configuration does the generation of the code based on templates. It uses hibernate console configuration to connect to database.	6
Hibernate Perspective.....	6
Modify Hibernate configuration artifacts	7
Introduction to Exporters.....	8
Common Generators.....	8
Data Tables Object Generators.....	8
Reference Data tables Object generators.....	9
Generating the Code For demo data entity	12
Plugging generated artifacts	18
Create a New Maven Project for custom code	18
The Artifacts for data entity	24
Step 1: Execute the SQLs in database	24
Step 2. Make an entry in YugandharBootProjectApplication	25
Step 3. Add the generated DO to in TxnPayload object	26
Start Application	27
Test using SOAPUI	27

About Yugandhar Project

The Yugandhar Project is the umbrella project focused on building open source cloud ready solutions. The current offerings include Yugandhar Open MDM Hub (MDM HUB) and Yugandhar Open Master Data Management (MDM) Hub.

About Yugandhar Open MDM Hub

Master Data Management came a long way in last decade or so. There are currently more than 20 MDM solutions catering to various specializations of MDM like Customer Data Integration (CDI), Product Information Management (PIM), vendor and supplier management etc. However most of these solutions come with licensing costs amounting to thousands of dollar. To offer a completely free solution which would be made available through Apache 2.0 license, A Project is started in 2017 under the name 'Yugandhar Open MDM Project' to build Open Source MDM solutions catering to CDI, PIM and Data Governance Capabilities. Yugandhar in Sanskrit means Ever Lasting and the strongest of its time. Our vision is to build the strongest, Open Source, Multi Domain, Cross Industry and completely free MDM Solution.

We are happy to announce that the first release of the Yugandhar MDM Hub catering to CDI solution is built with Open source technologies like Spring and Hibernate etc, inbuilt data Model, 400+ ready to use services and having incredible Out of the Box capabilities is currently being distributed. We aim to make the current CDI offering the strongest and Planning to bring Data Stewardship and PIM solutions in upcoming years.

About this document

This document provides a step by step guide for generating the code using Hibernate reverse engineering tool and Yugandhar Freemarker templates. The generator will generate the artifacts which have services for add, update and retrieve services.

Special Note

All the steps mentioned in the document are executing using Yugandhar Open MDM Hub for EWS however the same also apply to Yugandhar Open MDM Hub for JEEC.

Import code generator project

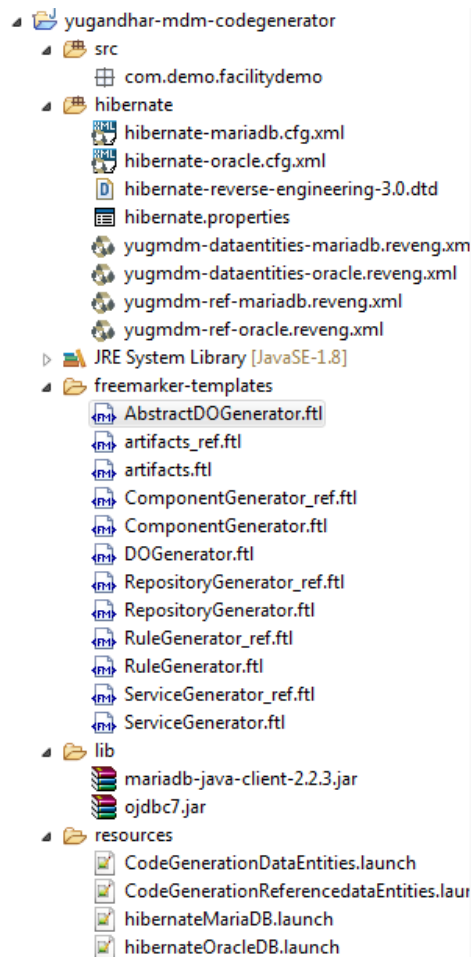
Download the code generator project from below github link

<https://github.com/yugandharproject/yugandhar-open-mdmhub/tree/master/yugandhar-mdm-codegenerator>

Copy the code generator project into the workspace folder

File → Import → existing Projects into Workspace

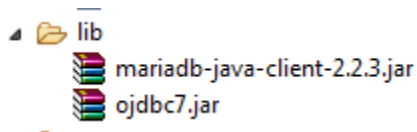
Provide the path of workspace, select the code generator project and click finish.
The project will be imported to the workspace



Database drivers

Download the database drivers from the below location and place it in the code generator.

- Oracle JDBC drivers - ojdbc7.jar: Oracle 11g and 12c driver for JDK7 and JDK8
<http://www.oracle.com/technetwork/database/features/jdbc/jdbc-drivers-12c-download-1958347.html>
- **MariaDB drivers:** Download mariadb-java-client-2.2.3.jar (or any later version) from below location <https://mariadb.com/downloads/connector>
<https://downloads.mariadb.com/Connectors/java/connector-java-2.2.3/mariadb-java-client-2.2.3.jar>



Understanding the code generator project artifacts

Hibernate configuration artifacts

hibernate-mariadb.cfg.xml: Hibernate configuration for MariaDB, contains database credentials

hibernate-oracle.cfg.xml: Hibernate configuration for oracle, contains database credentials

Hibernate Reverse Engineering artifacts

yugmdm--dataentities-mariadb.reveng.xml: Hibernate reverse engineering configuration for generating data entities with mariaDB.

yugmdm-dataentities-oracle.reveng.xml: Hibernate reverse engineering configuration for generating data entities with oracle.

yugmdm-ref-mariadb.reveng.xml: Hibernate reverse engineering configuration for generating reference data entities with mariaDB.

yugmdm-ref-oracle.reveng.xml: Hibernate reverse engineering configuration for generating reference data entities with oracle.

Hibernate Console Configurations (Launch Configuration)

Hibernate console configuration is used to launch the code generation configuration in hibernate perspective.

hibernateMariaDB.launch: Launch Configuration for Hibernate with mariaDB

hibernateOracleDB.launch: Launch Configuration for Hibernate with Oracle

Code Generation Configurations:

Code generation configuration does the generation of the code based on templates. It uses hibernate console configuration to connect to database.

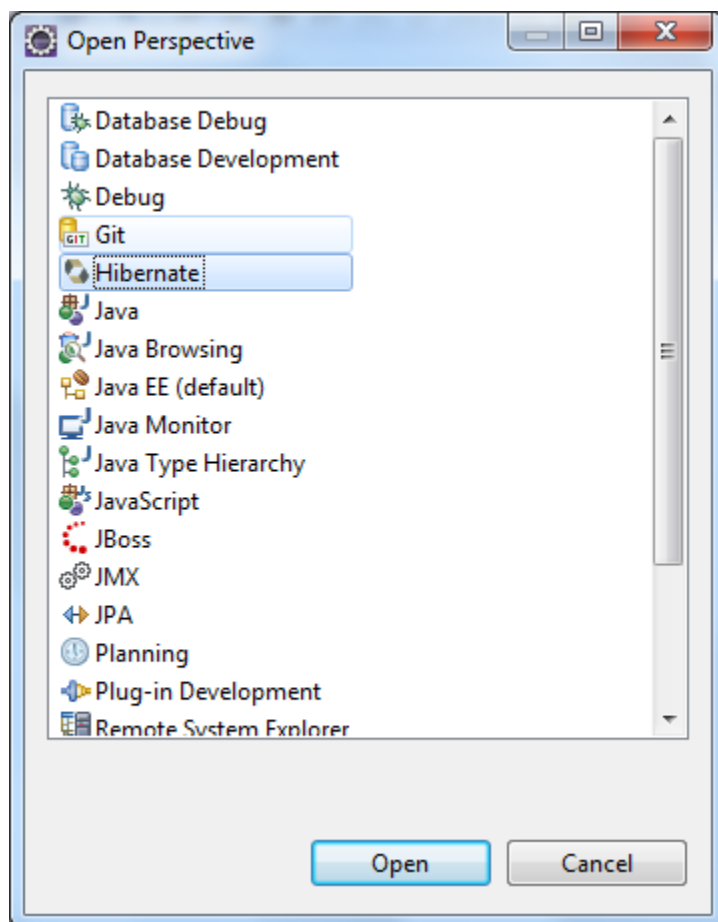
CodeGenerationDataEntities.launch: Code generation configuration for data entities. This can be used to generate the code for mariaDB as well as Oracle.

CodeGenerationReferencedataEntities.launch : Code generation configuration for Reference data entities . This can be used to generate the code for mariaDB as well as Oracle.

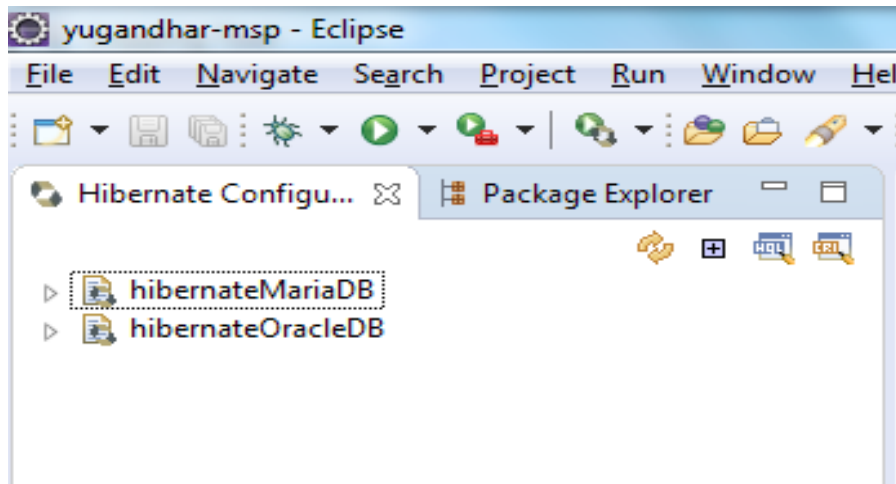
Hibernate Perspective

After installing the Jboss tools it installs the hibernate perspective to the eclipse, the steps for the same are covered in development environment setup guide.

Go to Windows → Perspective →open Perspective → Other → Hibernate

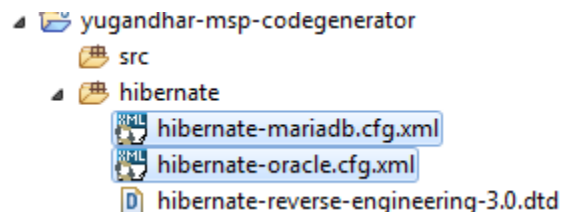


The Hibernate configuration for MariaDB and Oracle will be displayed as below



Modify Hibernate configuration artifacts

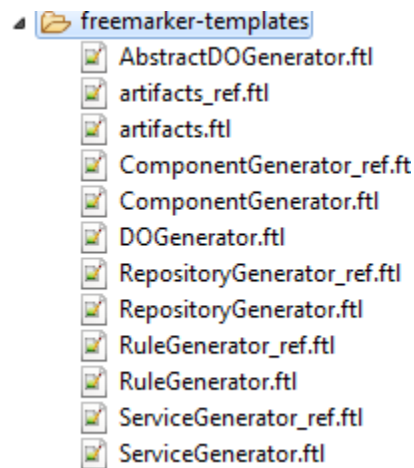
Code generator have couple of hibernate configuration files as per the database you are using i.e. mariaDb or Oracle



Open the files and modify the jdbc connection host, port, username, password, schema and catalog properties.

Introduction to Exporters

Exporters are basically the free-marker templates used to generate the required code.



Common Generators

Abstract DO Generator

File pattern [file_pattern]- {package-name}/Abstract{class-name}DO.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - AbstractDOGenerator.ftl

DO Generator

File pattern [file_pattern]- {package-name}/{class-name}DO.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - DOGenerator.ftl

Data Tables Object Generators

ServiceGenerator

File pattern [file_pattern]- {package-name}/{class-name}Service.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - ServiceGenerator.ftl

ComponentGenerator

File pattern [file_pattern]- {package-name}/{class-name}Component.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - ComponentGenerator.ftl

RepositoryGenerator

File pattern [file_pattern]- {package-name}/{class-name}Repository.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - RepositoryGenerator.ftl

RuleGenerator

File pattern [file_pattern]- {package-name}/{class-name}ComponentRule.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - RuleGenerator.ftl

artifacts

File pattern [file_pattern]- {package-name}/{class-name}_artifacts.txt

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - artifacts.ftl

Reference Data tables Object generators

ServiceGenerator_ref

File pattern [file_pattern]- {package-name}/{class-name}Service.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - ServiceGenerator_ref.ftl

ComponentGenerator_ref

File pattern [file_pattern]- {package-name}/{class-name}Component.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - ComponentGenerator_ref.ftl

RepositoryGenerator_ref

File pattern [file_pattern]- {package-name}/{class-name}Repository.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - RepositoryGenerator_ref.ftl

RuleGenerator_ref

File pattern [file_pattern]- {package-name}/{class-name}ComponentRule.java

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - RuleGenerator_ref.ftl

artifacts_ref

File pattern [file_pattern]- {package-name}/{class-name}_artifacts_ref.txt

Output directory [outputdir]- \yugandhar-mdm-codegenerator\src

Template directory [template_path] - \yugandhar-mdm-codegenerator\freemarker-templates

Template name [template_name] - artifacts_ref.ftl

Note-Yugandhar Open MDM Hub have preconfigured all the exporters so no need to do any additional settings for it to work.

Generating the Code For demo data entity

Let's create a demo entity in MariaDB using below command.

We have created FACILITYDEMO table (type of data table) for demo purpose

```
/* data table ddl*/
CREATE TABLE YUG_MDM HUB.FACILITYDEMO(
`ID_PK` VARCHAR(50) NOT NULL,
`VERSION` DECIMAL(22,0) NOT NULL,
`CREATED_TS` DATETIME(6) NOT NULL,
`DELETED_TS` DATETIME(6) NULL DEFAULT NULL,
`UPDATED_TS` DATETIME(6) NOT NULL,
`UPDATED_BY_USER` VARCHAR(50) NOT NULL ,
`UPDATED_BY_TXN_ID` VARCHAR(100) NULL DEFAULT NULL ,
`FACILITY_NAME` VARCHAR(100) NOT NULL ,
`LOCATION` VARCHAR(150) NULL DEFAULT NULL ,
PRIMARY KEY (`ID_PK`)
)
ENGINE=InnoDB
;

/* Audit log table ddl for base data table. This table stores the audit log history*/
CREATE TABLE YUG_MDM HUB.AL_FACILITYDEMO(
`AUDITLOG_ID_PK` varchar(50) NOT NULL,
`AUDITLOG_CREATED_TS` datetime(6) NOT NULL,
`AUDITLOG_ACTION_CODE` varchar(1) NOT NULL,
`ID_PK` VARCHAR(50) NOT NULL,
`VERSION` DECIMAL(22,0) NOT NULL,
`CREATED_TS` DATETIME(6) NOT NULL,
`DELETED_TS` DATETIME(6) NULL DEFAULT NULL,
`UPDATED_TS` DATETIME(6) NOT NULL,
`UPDATED_BY_USER` VARCHAR(50) NOT NULL ,
`UPDATED_BY_TXN_ID` VARCHAR(100) NULL DEFAULT NULL ,
`FACILITY_NAME` VARCHAR(100) NOT NULL ,
`LOCATION` VARCHAR(150) NULL DEFAULT NULL ,
PRIMARY KEY (`AUDITLOG_ID_PK`)
)
ENGINE=InnoDB
;

/* Audit log triggers which will make entries in the AL_FACILITYDEMO table whenever
FACILITYDEMO table is updated*/
DELIMITER //
CREATE TRIGGER `D_FACILITYDEMO` AFTER DELETE ON `FACILITYDEMO` FOR EACH
ROW BEGIN INSERT INTO AL_FACILITYDEMO ( AUDITLOG_ID_PK,
AUDITLOG_CREATED_TS, AUDITLOG_ACTION_CODE ,ID_PK, VERSION, CREATED_TS,
DELETED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_BY_TXN_ID, FACILITY_NAME,
LOCATION ) VALUES( uuid(), CURRENT_TIMESTAMP(6) , 'D', OLD.ID_PK,
OLD.VERSION, OLD.CREATED_TS, OLD.DELETED_TS, OLD.UPDATED_TS,
```

```

OLD.UPDATED_BY_USER, OLD.UPDATED_BY_TXN_ID, OLD.FACILITY_NAME,
OLD.LOCATION); END//
CREATE TRIGGER `I_FACILITYDEMO` AFTER INSERT ON `FACILITYDEMO` FOR EACH
ROW BEGIN INSERT INTO AL_FACILITYDEMO ( AUDITLOG_ID_PK,
AUDITLOG_CREATED_TS, AUDITLOG_ACTION_CODE ,ID_PK, VERSION, CREATED_TS,
DELETED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_BY_TXN_ID, FACILITY_NAME,
LOCATION ) VALUES( uuid(), CURRENT_TIMESTAMP(6) , 'I', NEW.ID_PK,
NEW.VERSION, NEW.CREATED_TS, NEW.DELETED_TS, NEW.UPDATED_TS,
NEW.UPDATED_BY_USER, NEW.UPDATED_BY_TXN_ID, NEW.FACILITY_NAME,
NEW.LOCATION); END//
CREATE TRIGGER `U_FACILITYDEMO` AFTER UPDATE ON `FACILITYDEMO` FOR EACH
ROW BEGIN INSERT INTO AL_FACILITYDEMO ( AUDITLOG_ID_PK,
AUDITLOG_CREATED_TS, AUDITLOG_ACTION_CODE ,ID_PK, VERSION, CREATED_TS,
DELETED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_BY_TXN_ID, FACILITY_NAME,
LOCATION ) VALUES( uuid(), CURRENT_TIMESTAMP(6) , 'U', NEW.ID_PK,
NEW.VERSION, NEW.CREATED_TS, NEW.DELETED_TS, NEW.UPDATED_TS,
NEW.UPDATED_BY_USER, NEW.UPDATED_BY_TXN_ID, NEW.FACILITY_NAME,
NEW.LOCATION); END//

/*Rollback scripts
DROP TRIGGER `D_FACILITYDEMO`;
DROP TRIGGER `I_FACILITYDEMO`;
DROP TRIGGER `U_FACILITYDEMO`;
DROP table FACILITYDEMO;
drop table AL_FACILITYDEMO;
*/

```

Note- The first 7 attributes of data table namely ID_PK, VERSION, CREATED_TS, DELETED_TS, UPDATED_TX, UPDATED_BY_USER and UPDATED_BY_TXN_ID are mandatory attributes for Yugandhar code generation framework to work. Effectively it's mandatory for all custom entities to have these attributes.

Now, let's revisit the reverse engineering configuration once again.

For MariaDB: If you are using MariaDB then below two configurations are for you.

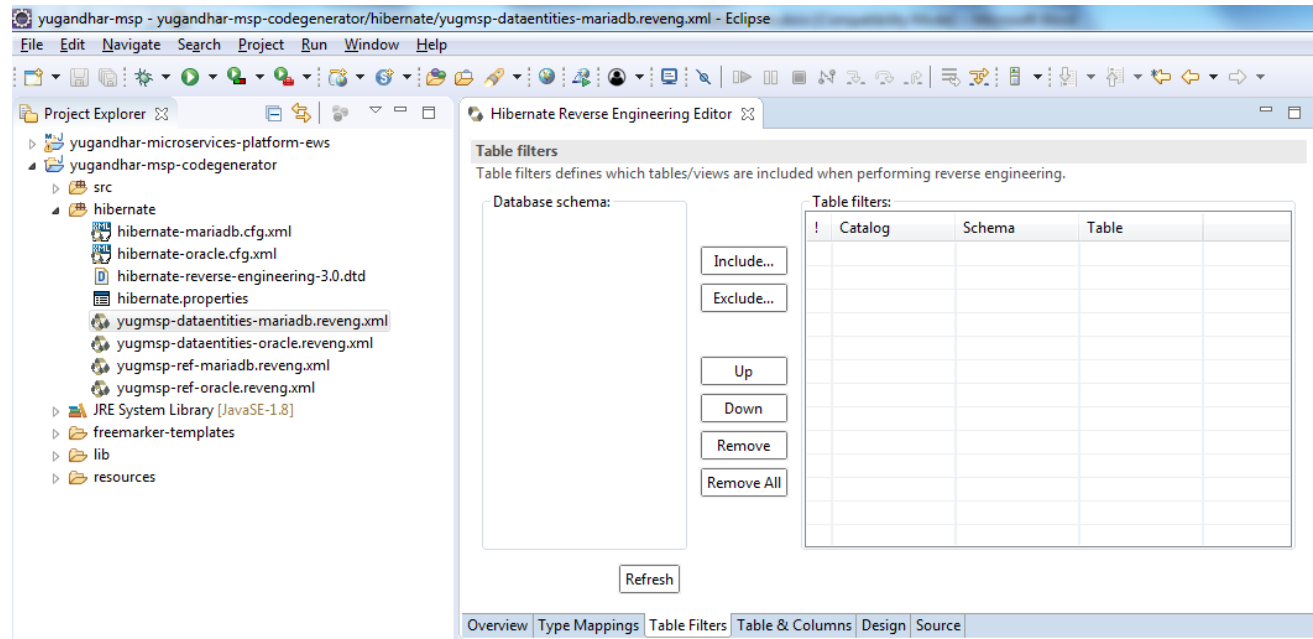
- **Yugmdm-dataentities-mariadb.reveng.xml:** Hibernate reverse engineering configuration for generating data entities with mariaDB.
- **Yugmdm-ref-mariadb.reveng.xml:** Hibernate reverse engineering configuration for generating reference data entities with mariaDB.

For Oracle: If you are using Oracle then below two configurations are for you.

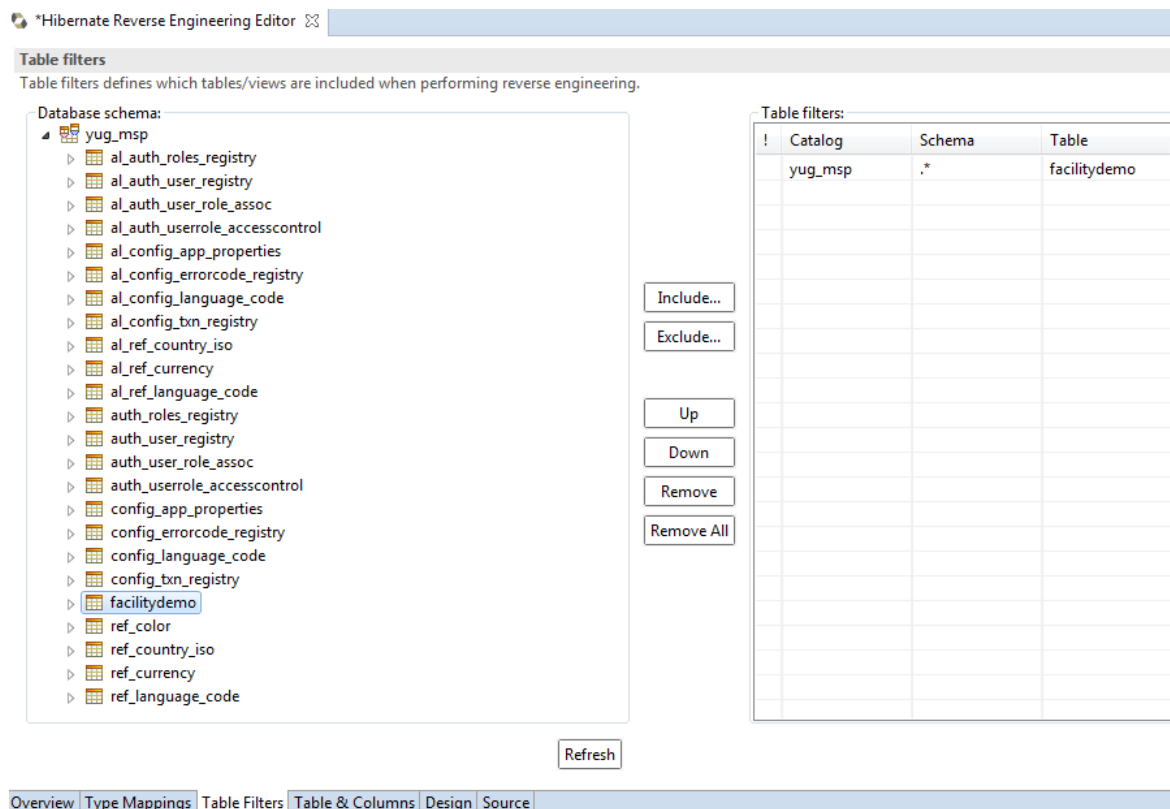
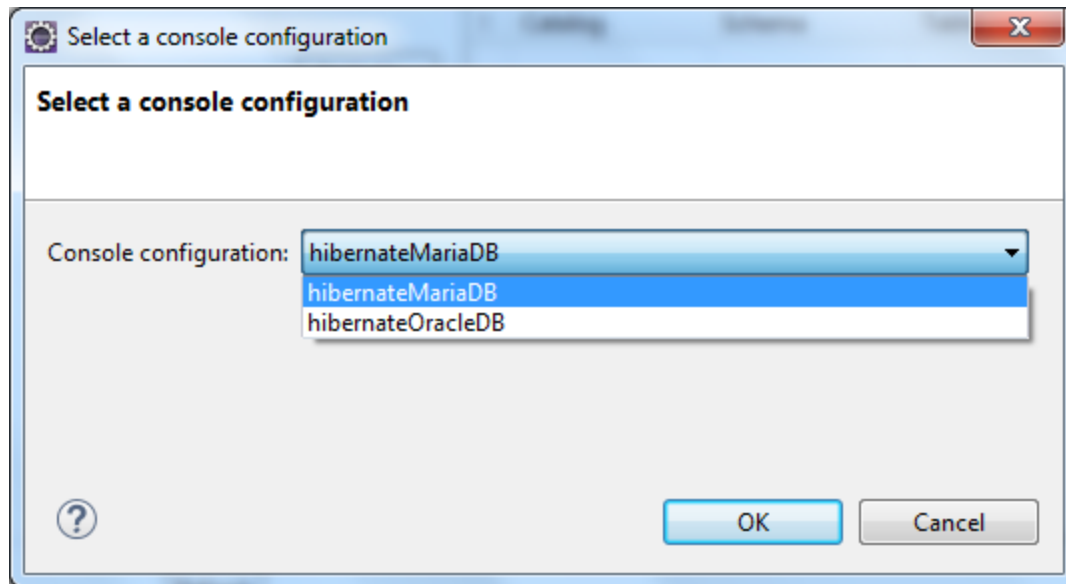
- **Yugmdm-dataentities-oracle.reveng.xml:** Hibernate reverse engineering configuration for generating data entities with oracle.
- **Yugmdm-ref-oracle.reveng.xml:** Hibernate reverse engineering configuration for generating reference data entities with oracle.

To understand the difference between data entities and Reference data entities, go through architectural overview and the data model guide.

Open the reverse engineering configuration for the type of database and entity you want to generate code. The FACILITYDEMO entity is of type data entity and demo database is mariaDB so open **yugmdm-dataentities-mariadb.reveng.xml**.

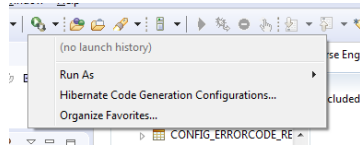


Here select the facility demo in the filter so click on the Refresh button in the Table Filters tab. It will ask for the console configuration to use to connect to database, choose HibernateMariaDB

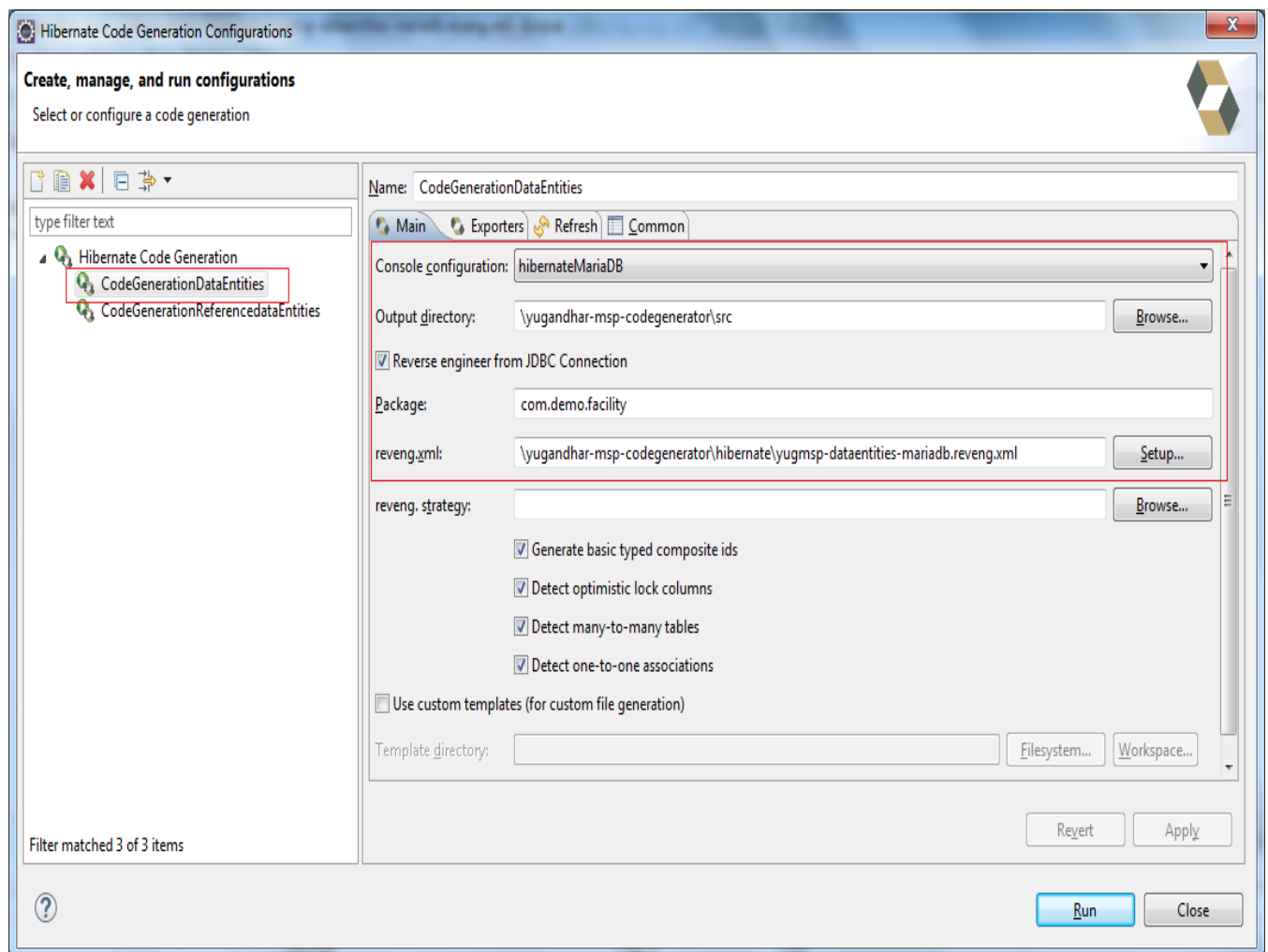


Select the facilitydemo entity, click include.. button and save the reverse engineering file. You may choose multiple tables in one go, for the demo purpose we are using only one.

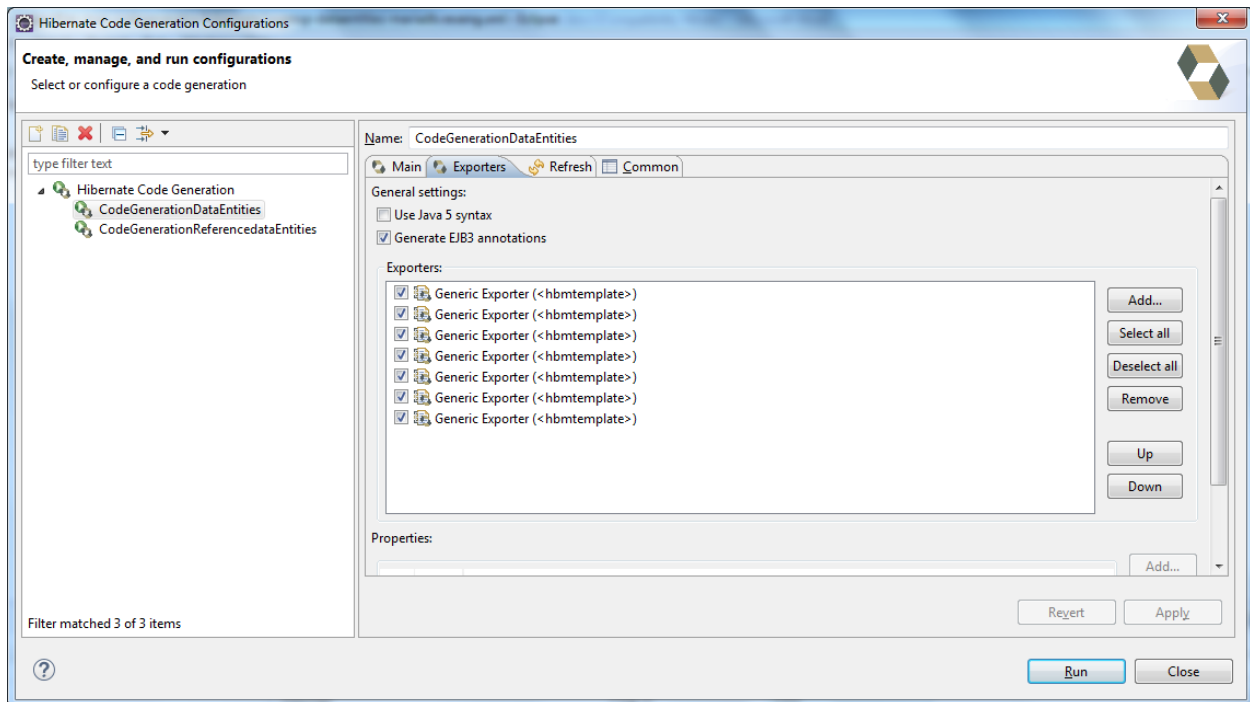
Now to generate the code, go to Hibernate perspective and click on the “Hibernate Code generation Configurations...” which will open the exporters we had configured.



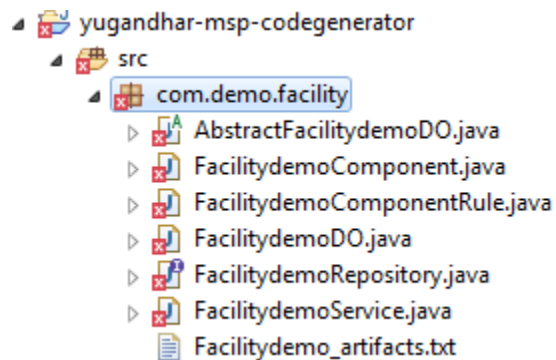
It will display the below shown dialog box with a set of two code generation configurations already configured. As we are generating the code for data entity, choose the CodeGenerationDataEntities configuration. Provide output directory as the src folder of 'yugandhar-mdm-codegenerator' project and also provide the package name. For the demo purpose we will be providing 'com.demo.facility' as package name. Choose the reveng.xml as the 'yugmdm-dataentities-mariadb.reveng.xml'.



Confirm that the checkboxes for all the exporters are checked in the exporters tab



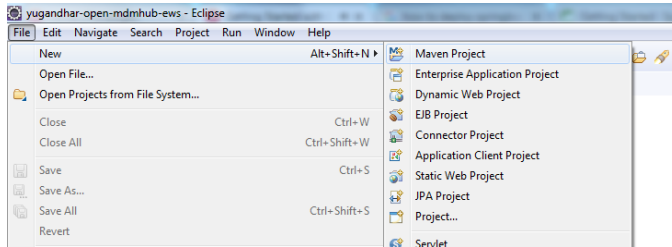
Click on Apply and Run. It will generate the artifacts in com.demo.facility package



Plugging generated artifacts

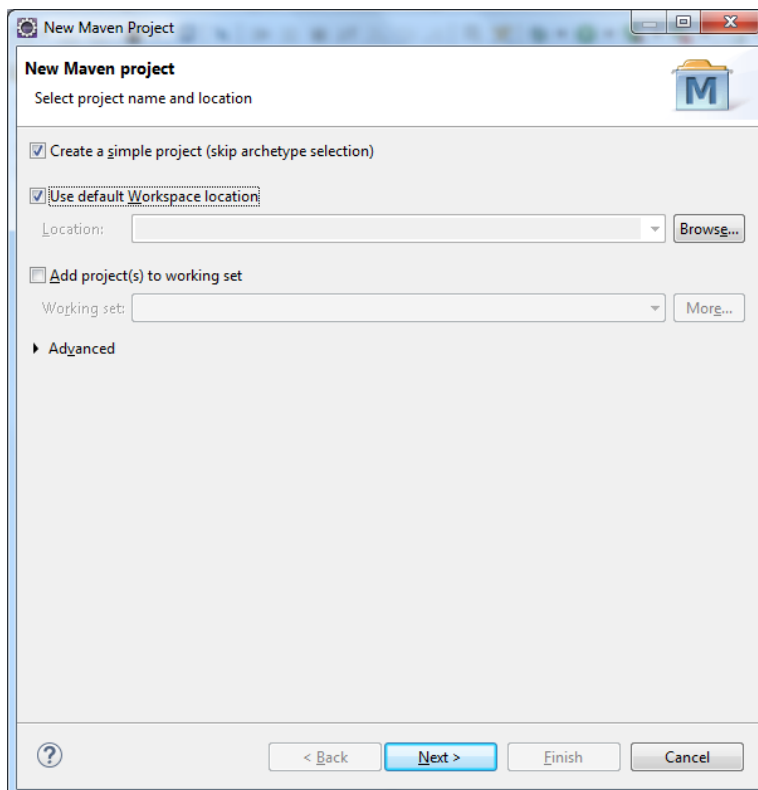
Create a New Maven Project for custom code

Select File → New → Maven Project



Check 'Create Simple Project (skip archetype selection)' and use default Workspace location. If project needs to be created in some other location then choose the respective location.

click Next.



Provide the details as shown in below screenshot

New Maven Project
Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

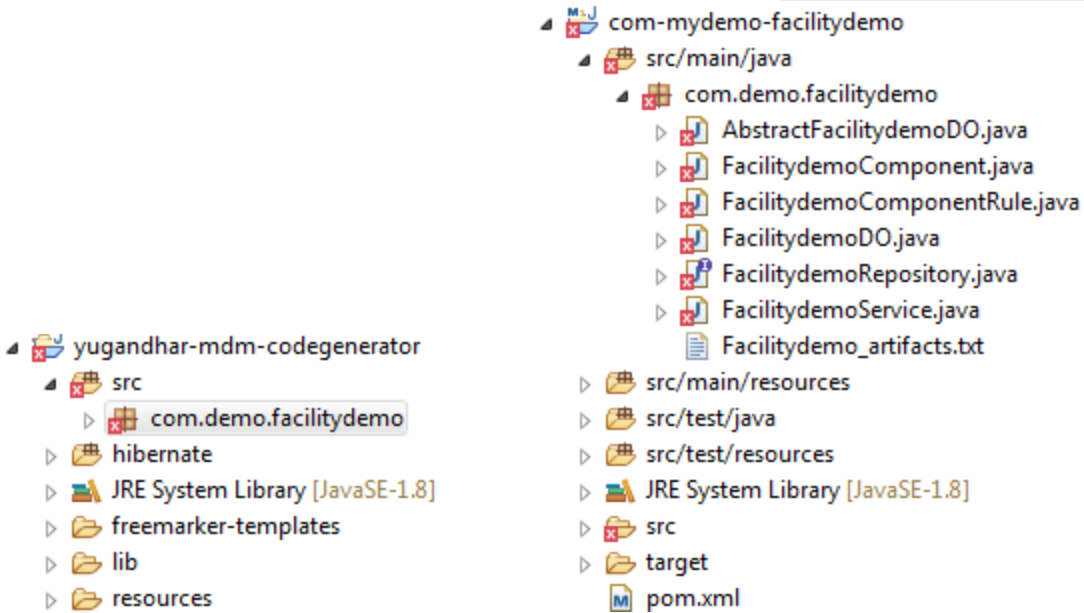
Group Id:

Artifact Id:

Version:

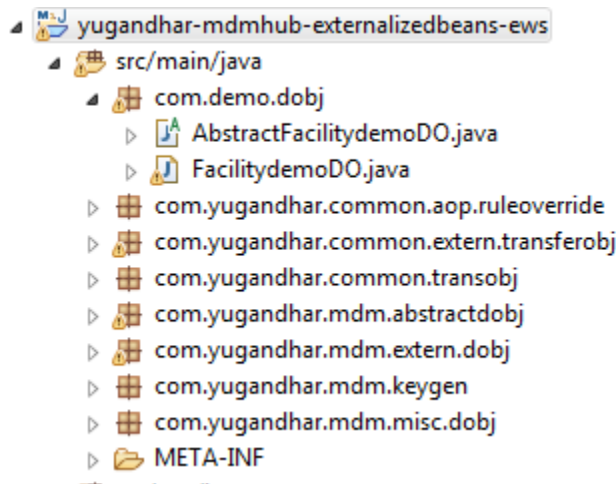
▶ **Advanced**

Copy the generated artifacts from code generator to facility demo project

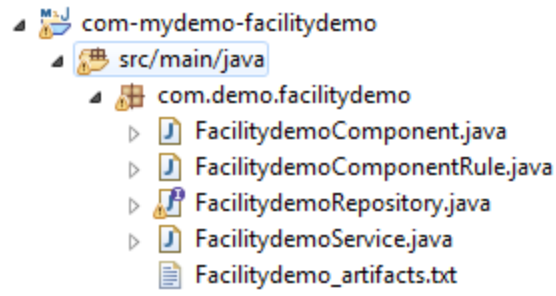


Due to Maven Limitation in which the project does not get compiled with maven if circular dependency is present between projects, we need to move the DO classes to externalizedbeans project.

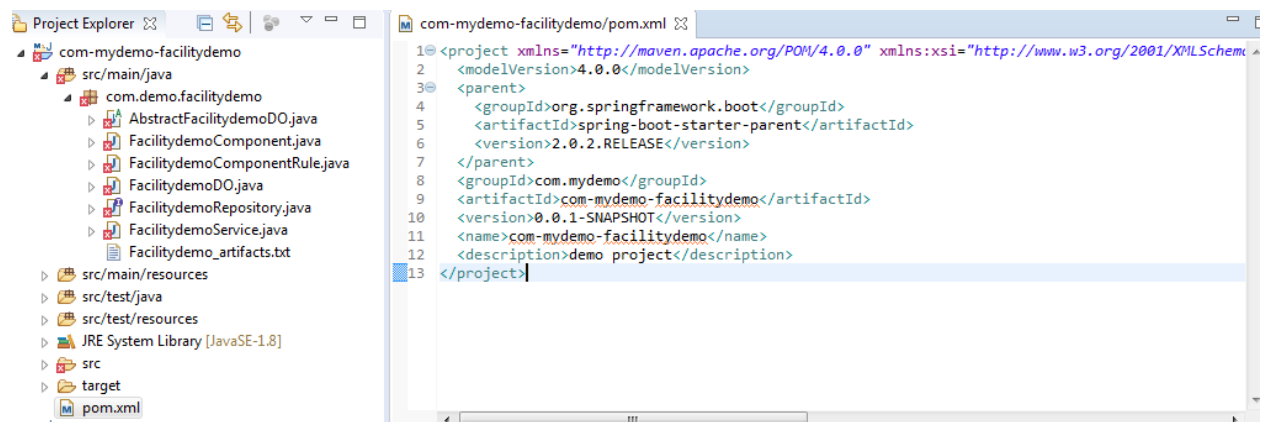
Create a new package in named 'com.demo.dobj' (or any of your choice) and move the AbstractFacilitydemoDO and FacilitydemoDO to yugandhar-mdmhub-externalizedbeans project.



So effectively your custom project will have only below classes



Open the pom.xml of the facilitydemo project



Add all the dependencies present in the Yugandhar boot project to the pom.xml of facility demo project

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <!-- <exclusions> <exclusion>
<groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
</exclusion> </exclusions> -->
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependencies>
```

```

</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
    <!-- <exclusions> <exclusion>
<groupId>org.apache.tomcat</groupId> <artifactId>tomcat-jdbc</artifactId>
        </exclusion> </exclusions> -->
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <!-- <artifactId>spring-boot-starter-jta-narayana</artifactId> --
>
        <artifactId>spring-boot-starter-jta-atomikos</artifactId>
</dependency>

<!-- <dependency> <groupId>com.zaxxer</groupId>
<artifactId>HikariCP</artifactId>
    <version>2.6.0</version> </dependency> -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-cache</artifactId> <!--Starter
for using Spring Framework's caching support -->
</dependency>
<dependency>
    <groupId>javax.cache</groupId> <!-- JSR-107 API -->
    <artifactId>cache-api</artifactId>
</dependency>
<dependency>
    <groupId>org.ehcache</groupId>
    <artifactId>ehcache</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-broker</artifactId>
</dependency>

<dependency>

```

```

        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
    </dependency>

    <!-- xml supports -->
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-xml</artifactId>
    </dependency>

    <!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
    <!-- for soundex algorithms -->
    <dependency>
        <groupId>commons-codec</groupId>
        <artifactId>commons-codec</artifactId>
    </dependency>

    <!-- For text comparision and matching.
https://commons.apache.org/proper/commons-text/ -->
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-text</artifactId>
        <version>1.1</version>
    </dependency>

    <!-- MariaDB as data repository -->
    <dependency>
        <groupId>org.mariadb.jdbc</groupId>
        <artifactId>mariadb-java-client</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
        <version>2.0.0.BUILD-SNAPSHOT</version>
    </dependency>

    <dependency>
        <groupId>com.yugandhar</groupId>
        <artifactId>
            yugandhar-mdmhub-externalizedbeans-ews
        </artifactId>
        <version>2.0.0.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>com.yugandhar</groupId>
        <artifactId>yugandhar-mdmhub-component-ews</artifactId>
        <version>2.0.0.RELEASE</version>
    </dependency>
</dependencies>

```

The Artifacts for data entity

For the data entity the generated file <entityname>_artifacts.txt (e.g. Facilitydemo_artifacts.txt) will have the generated artifacts.

The artifacts for the data entity are mentioned below

- **Step 1:** Execute the SQLs in database. For Maria DB enable the SQL_MODE=ORACLE option and then execute the insert scripts.
- **Step 2.** Make an entry in YugandharBootProjectApplication
- **Step 3.** Add the generated DO to in TxnPayload object
- **Step 4.** execute the transaction using sample message generated

Step 1: Execute the SQLs in database

Following sqls will get generated in the Facilitydemo_artifacts.txt file which registers the auto generated create, update and retrieve transactions in the configuration. All of the transactions are related to base table so suffix 'Base' is added at the end of the transaction name.

```
/*#enable below SQL_MODE option for mariaDB/MySQL else the insert SQL will fail. No  
need to enable this option for Oracle Database */  
/*set SQL_MODE=ORACLE; */
```

```
Insert into CONFIG_TXN_REGISTRY  
(ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS,  
TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS,  
UPDATED_BY_USER, UPDATED_TXN_REF_ID)  
Values  
(YUG_REGISTRY_SEQ.nextval, 0, 'createFacilitydemoBase',  
'com.demo.facility.FacilitydemoService', 'add',  
'create record in the database', CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,  
'Generator', '000000000');
```

```
Insert into CONFIG_TXN_REGISTRY  
(ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS,  
TXNSERVICE_CLASSMETHOD, DESCRIPTION,CREATED_TS, UPDATED_TS,  
UPDATED_BY_USER, UPDATED_TXN_REF_ID)  
Values  
(YUG_REGISTRY_SEQ.nextval, 0, 'updateFacilitydemoBase',  
'com.demo.facility.FacilitydemoService', 'merge',  
'update the database record based on primary key i.e. idpk',  
CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,'Generator', '000000000');
```

```
Insert into CONFIG_TXN_REGISTRY  
(ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS,
```




```
TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS,  
UPDATED_BY_USER, UPDATED_TXN_REF_ID)
```

Values

```
(YUG_REGISTRY_SEQ.nextval, 0, 'retrieveFacilitydemoBase',  
'com.demo.facility.FacilitydemoService', 'findById',  
'retrieve the record from database based on primary key i.e. idpk',  
CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 'Generator', '000000000');  
COMMIT;
```

Note- The rollback script to delete the entries from database is also provided in the artifacts. Do not execute the same unless the rollback is needed.

You must see the added entries in the database.



```
1 set SQL_MODE=ORACLE;  
2  
3 Insert into CONFIG_TXN_REGISTRY  
4 (ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS, TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_TXN_REF_ID)  
5 Values  
6 (YUG_REGISTRY_SEQ.nextval, 0, 'createFacilitydemoBase', 'com.demo.facilitydemo.FacilitydemoService', 'add',  
7 'create record in the database', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 'Generator', '000000000');  
8  
9  
10 Insert into CONFIG_TXN_REGISTRY  
11 (ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS, TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_TXN_REF_ID)  
12 Values  
13 (YUG_REGISTRY_SEQ.nextval, 0, 'updateFacilitydemoBase', 'com.demo.facilitydemo.FacilitydemoService', 'merge',  
14 'update the database record based on primary key i.e. idpk', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 'Generator', '000000000');  
15  
16  
17 Insert into CONFIG_TXN_REGISTRY  
18 (ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS, TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_TXN_REF_ID)  
19 Values  
20 (YUG_REGISTRY_SEQ.nextval, 0, 'retrieveFacilitydemoBase', 'com.demo.facilitydemo.FacilitydemoService', 'findById',  
21 'retrieve the record from database based on primary key i.e. idpk', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 'Generator', '000000000');  
22 COMMIT;  
23  
24
```

set SQL_MODE=ORACLE;
Insert into CONFIG_TXN_REGISTRY (ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS, TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_TXN_REF_ID) Values
Insert into CONFIG_TXN_REGISTRY (ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS, TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_TXN_REF_ID) Values
Insert into CONFIG_TXN_REGISTRY (ID_PK, VERSION, TXNSERVICE_NAME, TXNSERVICE_CLASS, TXNSERVICE_CLASSMETHOD, DESCRIPTION, CREATED_TS, UPDATED_TS, UPDATED_BY_USER, UPDATED_TXN_REF_ID) Values
COMMIT;
/* Affected rows: 3 Found rows: 0 Warnings: 0 Duration for 5 queries: 0.172 sec. */

Step 2. Make an entry in YugandharBootProjectApplication

Modify the entry in YugandharBootProjectApplication for your generated packages for spring component scan, entity scan and repository scan. Make the changes as below

```
@ComponentScan({"com.yugandhar.*", "<your package name here>"})  
@EntityScan({"com.yugandhar.*", "<your package name here>"})  
@EnableJpaRepositories({"com.yugandhar.*", "<your package name here>"})
```

e.g.

```
@ComponentScan({"com.yugandhar.*", "com.demo.*"})  
@EntityScan({"com.yugandhar.*", "com.demo.*"})  
@EnableJpaRepositories({"com.yugandhar.*", "com.demo.*"})
```

```

Facilitydemo_artifacts.txt  YugandharBootProjectApplication.java
1 package com.yugandhar.YugandharBootProject;
2
3+ import org.springframework.boot.SpringApplication;
14
15 @SpringBootApplication
16 @ComponentScan({"com.yugandhar.*","com.demo.*"})
17 @EntityScan({"com.yugandhar.*","com.demo.*"})
18 @EnableJpaRepositories({"com.yugandhar.*","com.demo.*"})
19 @EnableCaching
20 @EnableAspectJAutoProxy
21 @EnableJms
22 @EnableAsync
23 public class YugandharBootProjectApplication {
24 //public class YugandharBootProjectApplication extends SpringBootServle
25- public static void main(String[] args) {
26     SpringApplication.run(YugandharBootProjectApplication.class, ar
27 }
28
29 }
30

```

Step 3. Add the generated DO to in TxnPayload object

Add the below code snippet in

com.yugandhar.common.extern.transferobj.TxnPayload object. This will make the enable the transfer of Facilitydemo object through REST services.

```

protected FacilitydemoDO facilitydemoDO;
protected List<FacilitydemoDO> facilitydemoDOList;
/**
 * @return the demoDO
 */
public FacilitydemoDO getFacilitydemoDO() {
    return facilitydemoDO;
}
/**
 * @param demoDO the demoDO to set
 */
public void setFacilitydemoDO(FacilitydemoDO facilitydemoDO) {
    this.facilitydemoDO = facilitydemoDO;
}
/**
 * @return the demoDOList
 */
public List<FacilitydemoDO> getFacilitydemoDOList() {
    return facilitydemoDOList;
}
/**
 * @param demoDOList the demoDOList to set
 */

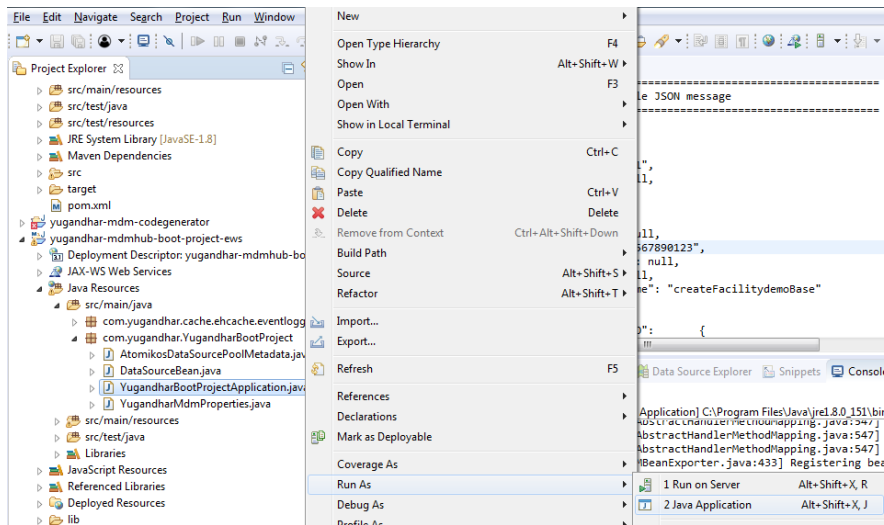
```

```
public void setFacilitydemoDOList(List<FacilitydemoDO> facilitydemoDOList) {
    this.facilitydemoDOList = facilitydemoDOList;
}
```

Start Application

Start the

Start the application, right click on YugandharMDM HubplatformApplication.java → Run As → Java Application.



Confirm the logs that application is started

```
2018-06-08 17:35:52,451 [main] [MBeanExporter.java:895] Bean with name 'refreshScope' has been autodetected for JMX exposure
2018-06-08 17:35:52,451 [main] [MBeanExporter.java:895] Bean with name 'configurationPropertiesRebinder' has been autodetected for JMX exposure
2018-06-08 17:35:52,451 [main] [MBeanExporter.java:675] Located managed bean 'environmentManager': registering with JMX server as MBean [org.sp
2018-06-08 17:35:52,529 [main] [MBeanExporter.java:675] Located managed bean 'refreshScope': registering with JMX server as MBean [org.springfr
2018-06-08 17:35:52,638 [main] [MBeanExporter.java:675] Located managed bean 'configurationPropertiesRebinder': registering with JMX server as
2018-06-08 17:35:52,670 [main] [DefaultLifecycleProcessor.java:351] Starting beans in phase 2147483647
2018-06-08 17:35:52,716 [main] [DirectJDKLog.java:180] Starting ProtocolHandler ["http-nio-8090"]
2018-06-08 17:35:53,262 [main] [DirectJDKLog.java:180] Using a shared selector for servlet write/read
2018-06-08 17:35:53,795 [main] [TomcatWebServer.java:206] Tomcat started on port(s): 8090 (http) with context path ''
2018-06-08 17:35:53,841 [main] [StartupInfoLogger.java:59] Started YugandharBootProjectApplication in 51.81 seconds (JVM running for 52.931)
```

Test using SOAPUI

The REST url would be something like below

<http://localhost:8091/rest/YugandharRequestProcessor>

Right click on SOAP project and click 'new REST Service from URI' and click **ok**

Copyright [2017] [Yugandhar Open MDM Hub] Licensed under the Apache License, Version 2.0

Method	Endpoint	Resource
POST	http://localhost:8091	/rest/YugandharRequestProcessor

In the application/json, give the sample message from generated in artifacts.txt file. Change the attributes e.g. facilityName and location and then execute the step.

```
{
  "txnHeader": {
    "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
    "requestOriginSource": null,
    "requesterSystem": null,
    "transactionServiceName": "createFacilitydemoBase"
  },
  "txnPayload": {
    "facilitydemoDO": {
      "idPk": null,
      "version": null,
      "createdTs": null,
      "deletedTs": null,
      "updatedTs": null,
      "updatedByUser": null,
      "updatedByTxnId": null,
      "facilityName": "DEMO FACILITY",
      "location": "Demo Locaiton"
    }
  }
}
```

Request 1

Method: POST, Endpoint: http://localhost:8090, Resource: /rest/YugandharRequestProcessor

Media Type: application/json

Request Body (JSON):

```
{
  "requesterSystem": null,
  "transactionServiceName": "createFacilitydemoBase",
  "txnPayload": {
    "facilitydemoDO": {
      "idPk": null,
      "version": null,
      "createdTs": null,
      "deletedTs": null,
      "updatedTs": null,
      "updatedByUser": null,
      "updatedByTxnId": null,
      "facilityName": "DEMO FACILITY",
      "location": "Demo Locaiton"
    }
  }
}
```

Response Body (JSON):

```
{
  "responseCode": "SUCCESS",
  "txnHeader": {
    "requesterLanguage": "1",
    "userName": "admin",
    "userRole": "admin",
    "txnMessageId": "1234567890123",
    "transactionServiceName": "createFacilitydemoBase",
    "totalExecutionTimeMillies": "4597"
  },
  "txnPayload": {
    "facilitydemoDO": {
      "idPk": "94ef7985-5cad-44be-9688-04fb8119f097",
      "version": 0,
      "createdTs": "2018-06-08T12:15:59.334+0000",
      "updatedTs": "2018-06-08T12:15:59.334+0000",
      "updatedByUser": "admin",
      "updatedByTxnId": "1234567890123",
      "facilityName": "DEMO FACILITY",
      "location": "Demo Locaiton"
    }
  }
}
```

Response time: 10827ms (511 bytes)

Logs: SoapUI log, http log, jetty log, error log, wsrm log, memory log

Likewise execute the update

```
{
  "txnHeader": {
    "requesterLanguage": "1",
    "requesterLocale": null,
    "userName": "admin",
    "userRole": "admin",
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
    "requestOriginSource": null,
    "requesterSystem": null,
    "transactionServiceName": "updateFacilitydemoBase"
  },
  "txnPayload": {
    "facilitydemoDO": {
      "idPk": "94ef7985-5cad-44be-9688-04fb8119f097",
      "version": 0,
      "createdTs": null,
      "deletedTs": null,

```

```
}}
```

```
"updatedTs": null,  
"updatedByUser": null,  
"updatedByTxnId": null,  
"facilityName": "DEMO FACILITY updated",  
"location": "Demo Locaiton updated"
```

The screenshot displays a REST client interface with a POST request to `http://localhost:8090/rest/YugandharRequestProcessor`. The request body is a JSON object with the following structure:

```
{  
  "requestOriginSource": null,  
  "requesterSystem": null,  
  "transactionServiceName": "updateFacilitydemoBase",  
  "txnPayload": {  
    "facilitydemoDO": {  
      "idPk": "94ef7985-5cad-44be-9688-04fb8119f097",  
      "version": 0,  
      "createdTs": null,  
      "deletedTs": null,  
      "updatedTs": null,  
      "updatedByUser": null,  
      "updatedByTxnId": null,  
      "facilityName": "DEMO FACILITY updated",  
      "location": "Demo Locaiton updated"  
    }  
  }  
}
```

The response is a JSON object with the following structure:

```
{  
  "responseCode": "SUCCESS",  
  "txnHeader": {  
    "requesterLanguage": "1",  
    "userName": "admin",  
    "userRole": "admin",  
    "txnMessageId": "1234567890123",  
    "transactionServiceName": "updateFacilitydemoBase",  
    "totalExecutionTimeMillies": "1580"  
  },  
  "txnPayload": {  
    "facilitydemoDO": {  
      "idPk": "94ef7985-5cad-44be-9688-04fb8119f097",  
      "version": 1,  
      "createdTs": "2018-06-08T12.15.59.334+0000",  
      "updatedTs": "2018-06-08T12.17.34.034+0000",  
      "updatedByUser": "admin",  
      "updatedByTxnId": "1234567890123",  
      "facilityName": "DEMO FACILITY updated",  
      "location": "Demo Locaiton updated"  
    }  
  }  
}
```

And execute retrieve as well

```
{  
  "txnHeader": {  
    "requesterLanguage": "1",  
    "requesterLocale": null,  
    "userName": "admin",  
    "userRole": "admin",  
    "accessToken": null,  
    "txnCorrelationId": null,  
    "txnMessageId": "1234567890123",  
    "requestOriginSource": null,  
    "requesterSystem": null,  
    "transactionServiceName": "updateFacilitydemoBase",  
    "txnPayload": {  
      "facilitydemoDO": {  
        "idPk": "94ef7985-5cad-44be-9688-04fb8119f097",  
        "version": 1,  
        "createdTs": "2018-06-08T12.15.59.334+0000",  
        "updatedTs": "2018-06-08T12.17.34.034+0000",  
        "updatedByUser": "admin",  
        "updatedByTxnId": "1234567890123",  
        "facilityName": "DEMO FACILITY updated",  
        "location": "Demo Locaiton updated"  
      }  
    }  
  }  
}
```

```

"requesterSystem": null,
  "transactionServiceName": "retrieveFacilitydemoBase"
},
"txnPayload": {
  "facilitydemoDO": {
    "idPk": "94ef7985-5cad-44be-9688-04fb8119f097"
  }
}
}

```

The screenshot shows a REST client interface with a POST request to `http://localhost:8090/rest/YugandharRequestProcessor`. The request body is a JSON object with fields like `requesterSystem`, `transactionServiceName`, and `txnPayload`. The response is also a JSON object with fields like `responseCode`, `txnHeader`, and `txnPayload`. The `txnPayload` in the response contains the same `facilitydemoDO` object as the request.

Check the database table and audit log table to confirm everything is working fine.

select * from FACILITYDEMO;

yug_owner.facilitydemo: 1 rows total (approximately) Next Show all Sorting Columns (9/9) Filter

ID_PK	VERSION	CREATED_TS	DELETED_TS	UPDATED_TS	UPDATED_BY_USER	UPDATE
94ef7985-5cad-44be-9688-04fb8119f097	1	2018-06-08 17:45:59.334000	(NULL)	2018-06-08 17:47:34.034000	admin	123456

select * from AL_FACILITYDEMO;

This shows the audit history the original row having AUDITLOG_ACTION_CODE as 'I' and and the updated one having 'U'

yug_owner.al_facilitydemo: 2 rows total (approximately)						Next	Show all	Sorting	Columns (12/12)	Filter
AUDITLOG_ID_PK	AUDITLOG_CREATED_TS	AUDITLOG_ACTION_CODE	ID_PK	VERSION	CREATE					
b47bfb19-6b15-11e8-83d4-c85b764d5d1a	2018-06-08 17:45:59.982403	I	94ef7985-5cad-44be-9688-04fb8119f097	0	2018-06					
ed0a1f98-6b15-11e8-83d4-c85b764d5d1a	2018-06-08 17:47:34.840403	U	94ef7985-5cad-44be-9688-04fb8119f097	1	2018-06					

This certifies the code generation for base entity. You may generate the code for multiple entities and merge it to create entire application. The Yugandhar Open MDM Hub is built on the top of Yugandhar MDM HUB; you may take a look of the MDM Hub to understand how a great Microservice application can be built using Yugandhar Open MDM Hub.