Yugandhar Microservice Platform

Code Generation Guide

Yugandhar Microservice Platform Release - V1.0.0
Date – 23/05/2018

Contents

About Yugandhar Project

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

About Yugandhar Microservice Platform

Yugandhar Microservice Platform (also referred as Yugandhar MSP) provides framework for rapid development of your Microservice. This is an architecturally proven Springboot based application having all the basic components needed for a Microservice application to work which just needs to be extended as per your requirements.

Yugandhar Microservice platform codes with code generation templates for rapid development. Just design your data model and generate the code using Yugandhar templates, your base table services will be ready. To create the composite services, you may have to write custom code which would take minimal efforts. Your new Microservice would be ready in just few hours.

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 Microservice platform for EWS however the same also apply to Yugandhar Microservice platform for JEEC.

Import code generator project

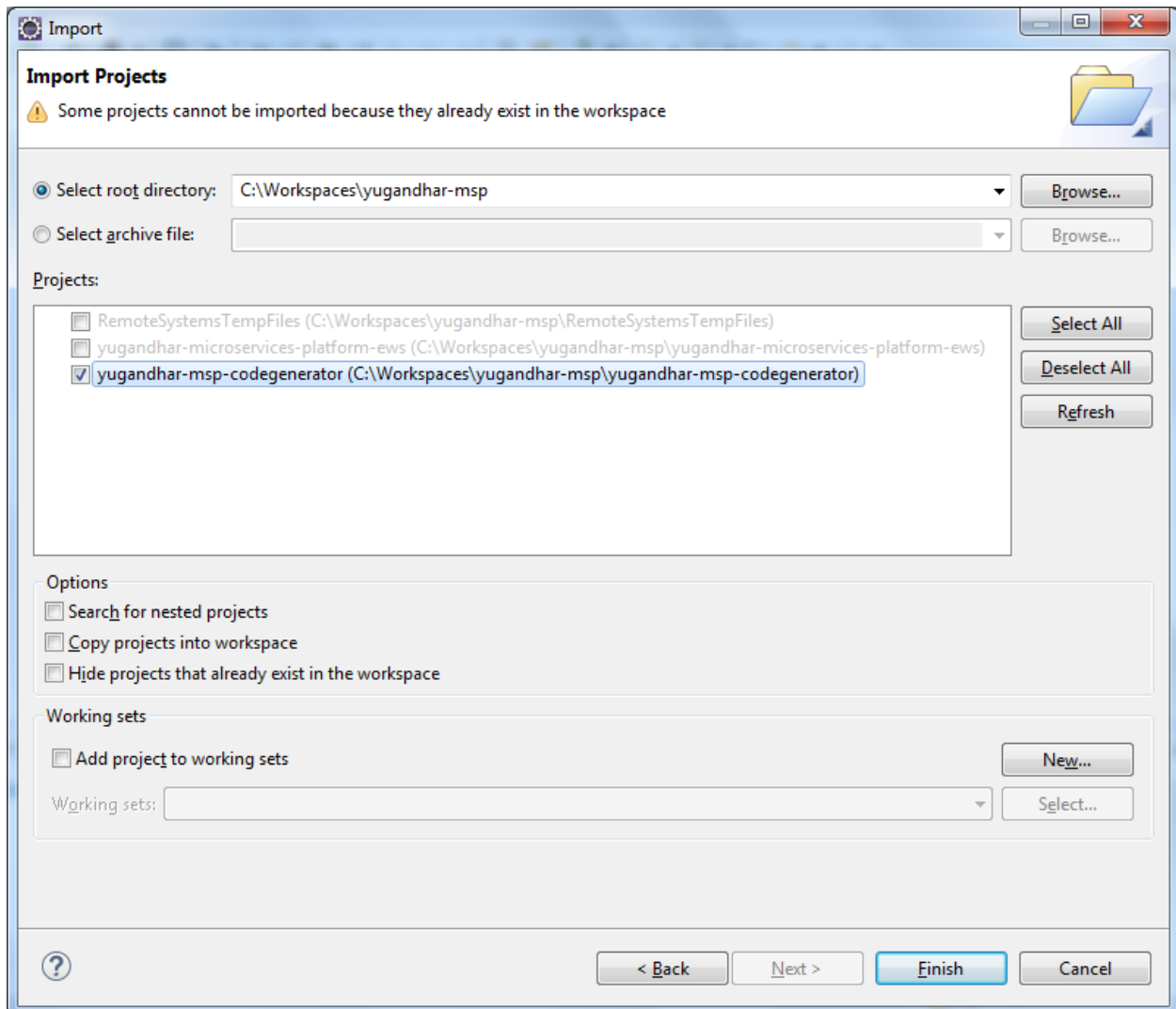Download the code generator project from below github link

https://github.com/yugandharproject/yugandhar-microservices-platform/tree/master/yugandhar-msp-codegenerator
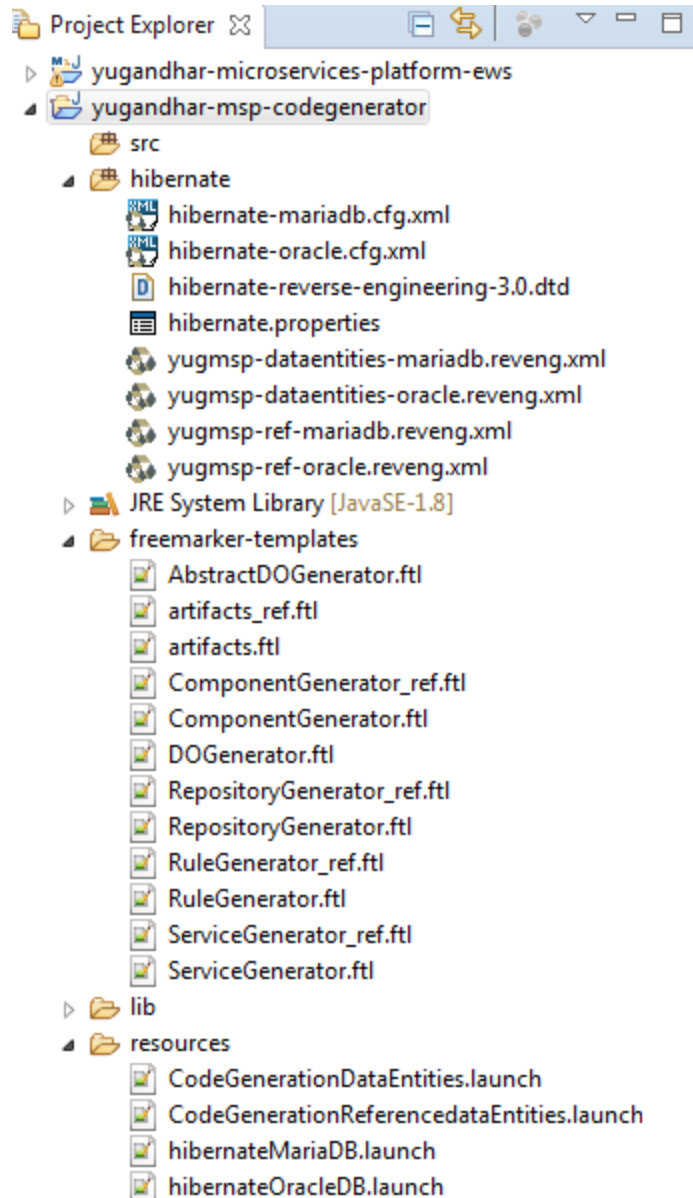
Copy the code generator project into the workspace folder

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

Project Explorer ⊠

▷ yugandhar-microservices-platform-ews
▲ yugandhar-msp-codegenerator
　　src
　▲ hibernate
　　　hibernate-mariadb.cfg.xml
　　　hibernate-oracle.cfg.xml
　　　hibernate-reverse-engineering-3.0.dtd
　　　hibernate.properties
　　　yugmsp-dataentities-mariadb.reveng.xml
　　　yugmsp-dataentities-oracle.reveng.xml
　　　yugmsp-ref-mariadb.reveng.xml
　　　yugmsp-ref-oracle.reveng.xml
　▷ JRE System Library [JavaSE-1.8]
　▲ freemarker-templates
　　　AbstractDOGenerator.ftl
　　　artifacts_ref.ftl
　　　artifacts.ftl
　　　ComponentGenerator_ref.ftl
　　　ComponentGenerator.ftl
　　　DOGenerator.ftl
　　　RepositoryGenerator_ref.ftl
　　　RepositoryGenerator.ftl
　　　RuleGenerator_ref.ftl
　　　RuleGenerator.ftl
　　　ServiceGenerator_ref.ftl
　　　ServiceGenerator.ftl
　▷ lib
　▲ resources
　　　CodeGenerationDataEntities.launch
　　　CodeGenerationReferencedataEntities.launch
　　　hibernateMariaDB.launch
　　　hibernateOracleDB.launch

Database drivers

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

- Oralce 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

- ⊿ 📁 yugandhar-msp-codegenerator
  - ▷ 📦 src
  - ▷ 📦 hibernate
  - ▷ 📚 JRE System Library [JavaSE-1.8]
  - ▷ 📂 freemarker-templates
  - ⊿ 📂 lib
    - 📦 mariadb-java-client-2.2.3.jar
    - 📦 ojdbc7.jar
  - ⊿ 📂 resources

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

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

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

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

yugmsp-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 lauch 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.

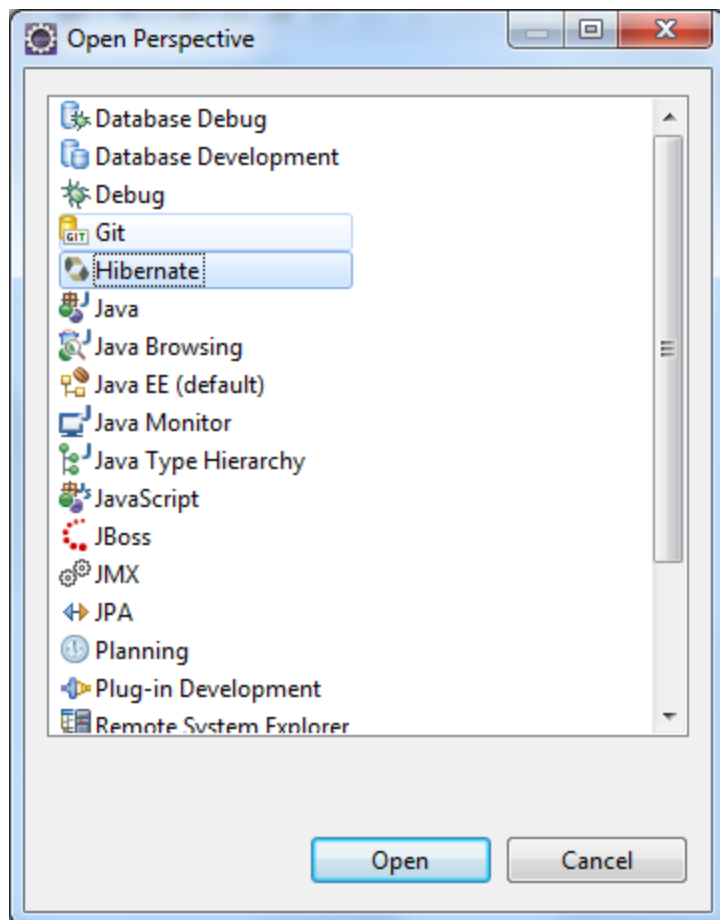*CodeGenerationReferencedataEntities.launch :* Code generation configuration for Reference data entities . This can be used to generate the code for mariaDB as well as Oracle.

Hibernate Perspective

After installing the Jboss tools it installs the hibernate perspective to the eclipse.

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

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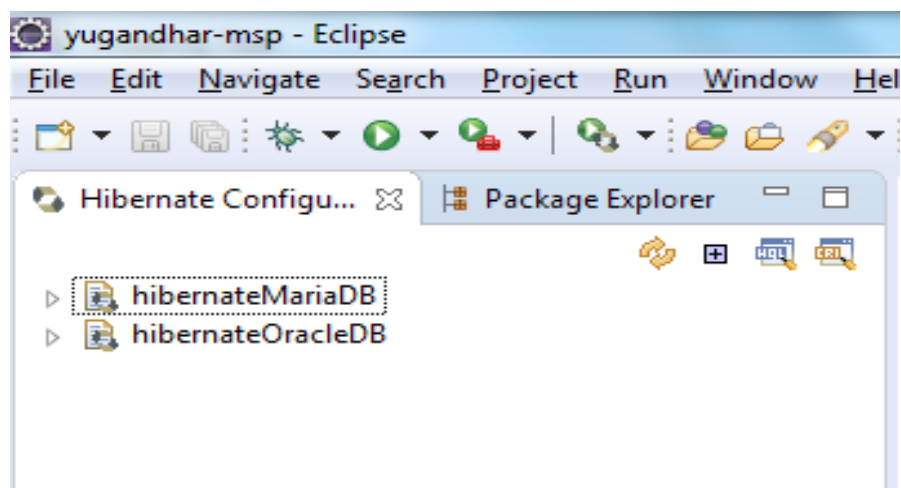
Modify Hibernate configuration artifacts

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

- yugandhar-msp-codegenerator
  - src
  - hibernate
    - hibernate-mariadb.cfg.xml
    - hibernate-oracle.cfg.xml
    - hibernate-reverse-engineering-3.0.dtd
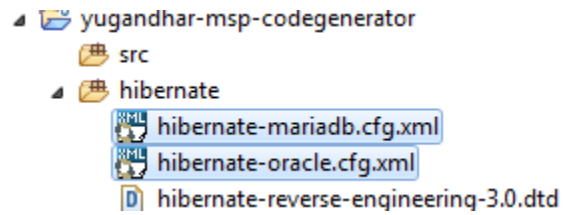
Open the files and modify the jdbc connection host, port, username, password, schema and catalog properties.

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-msp-codegenerator\src

Template directory [template_path] - \yugandhar-msp-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-msp-codegenerator\src
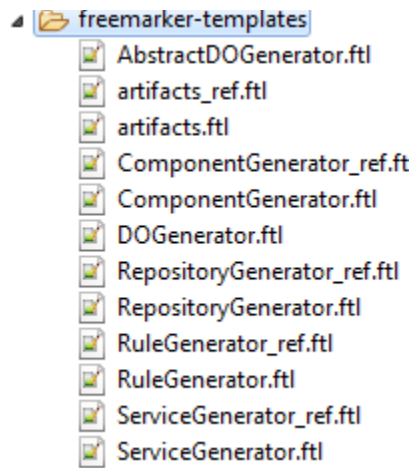
Template directory [template_path] - \yugandhar-msp-codegenerator\freemarker-templates

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-msp-codegenerator\src

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

Template name [template_name] - ServiceGenerator.ftl

ComponentGenerator

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

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

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

Template name [template_name] - ComponentGenerator.ftl

RepositoryGenerator

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

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

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

Template name [template_name] - RepositoryGenerator.ftl

RuleGenerator

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

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

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

Template name [template_name] - RuleGenerator.ftl

artifacts

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

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

Template directory [template_path] - \yugandhar-msp-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-msp-codegenerator\src

Template directory [template_path] - \yugandhar-msp-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-msp-codegenerator\src

Template directory [template_path] - \yugandhar-msp-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-msp-codegenerator\src

Template directory [template_path] - \yugandhar-msp-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-msp-codegenerator\src

Template directory [template_path] - \yugandhar-msp-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-msp-codegenerator\src

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

Template name [template_name] - artifacts_ref.ftl

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

Generating the Code For demo entity

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

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

```sql
/* data table ddl*/
CREATE TABLE YUG_MSP.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_MSP.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.

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

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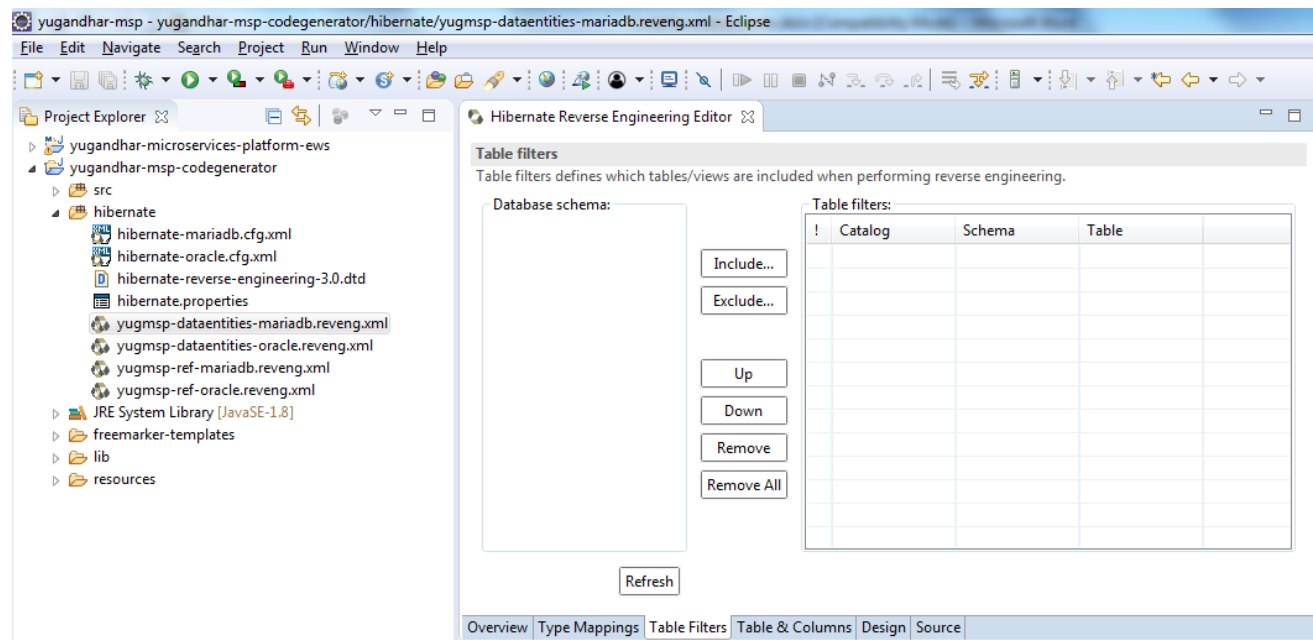

Here select the facility demo in the filter so click on the Refresh button in the Table Filters tab. It will ask for the console configuration to use to connect to database, choose HibernateMariaDB
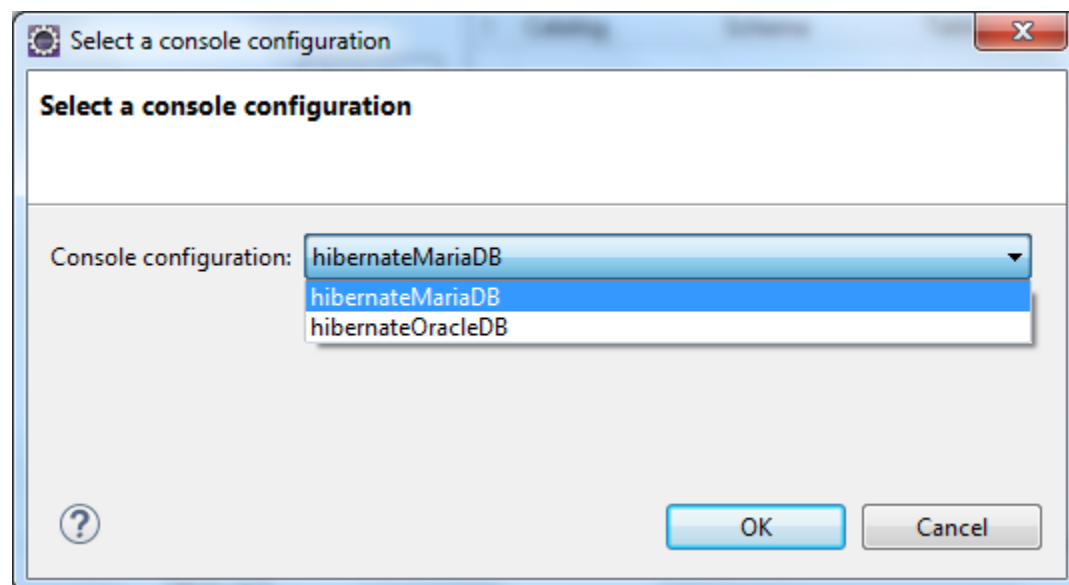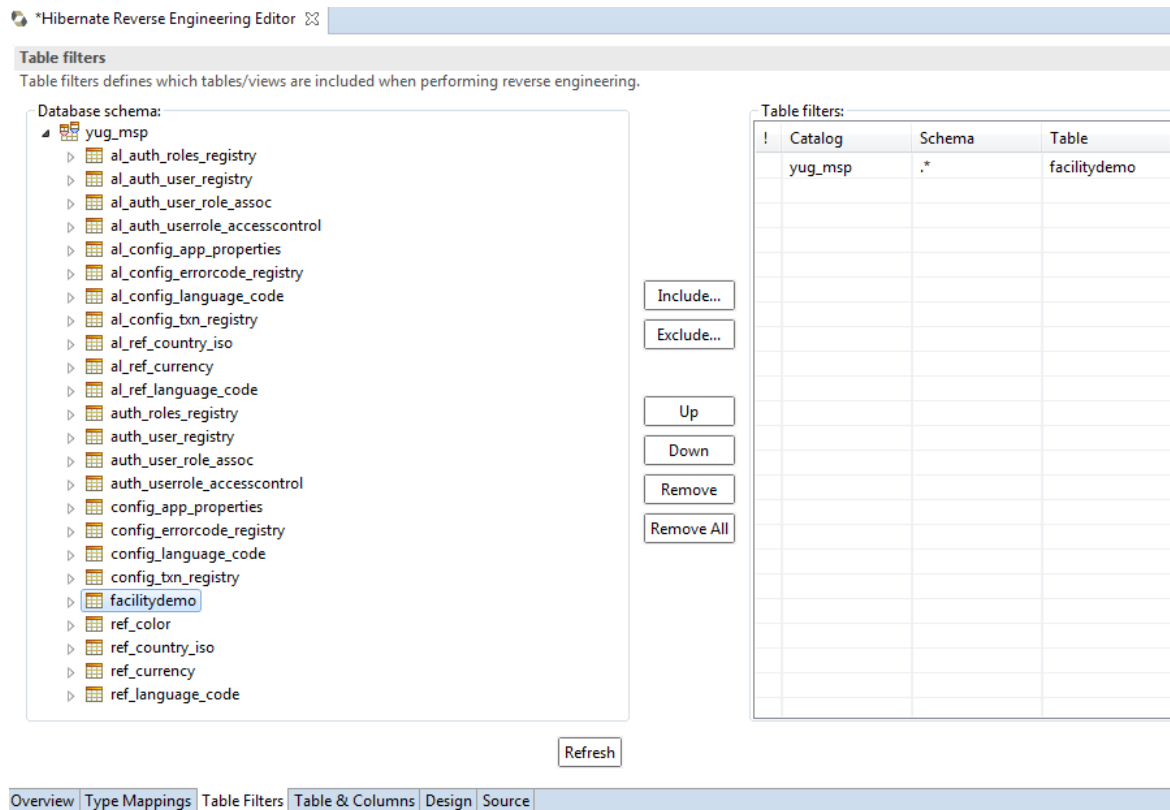
Select the facilitydemo entity, click include.. button and save the reverse engineering file. You may choose multiple tables in one go, for the demo purpose we are using only one.

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.
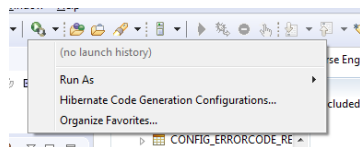

It will display the below shown dialog box with a set of two code generation configurations already configured. As we are generating the code for data entity, choose the CodeGenerationDataEntities configuration. Provide output directory as the src folder of 'yugandhar-msp-codegenerator 'project and also provide the package name. For the demo purpose we will be providing 'com.demo.facility' as package name. Choose the reveng.xml as the 'yugmsp-dataentities-mariadb.reveng.xml'.
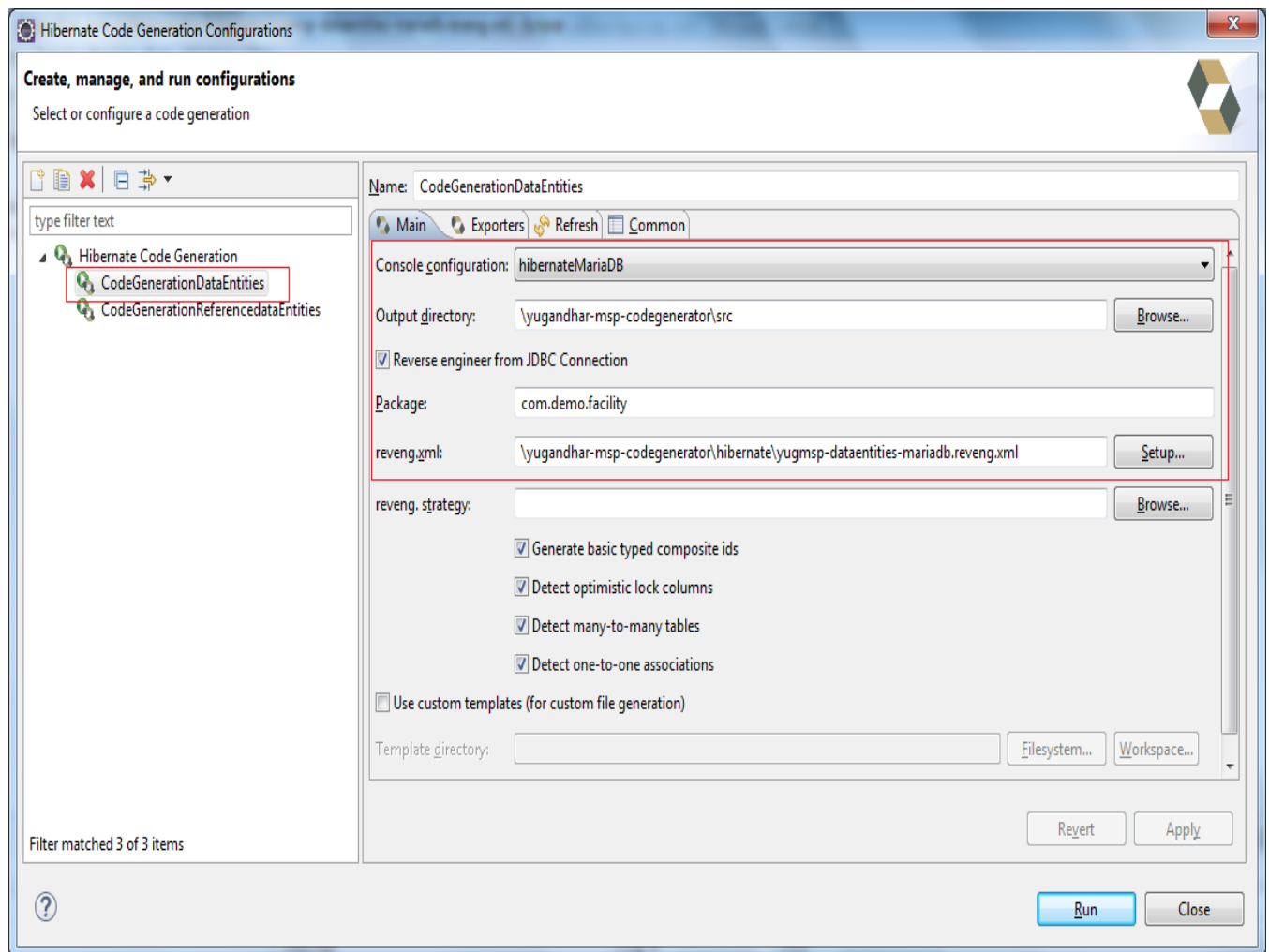
Confirm that the checkboxes for all the exporters are checked in the exporters tab
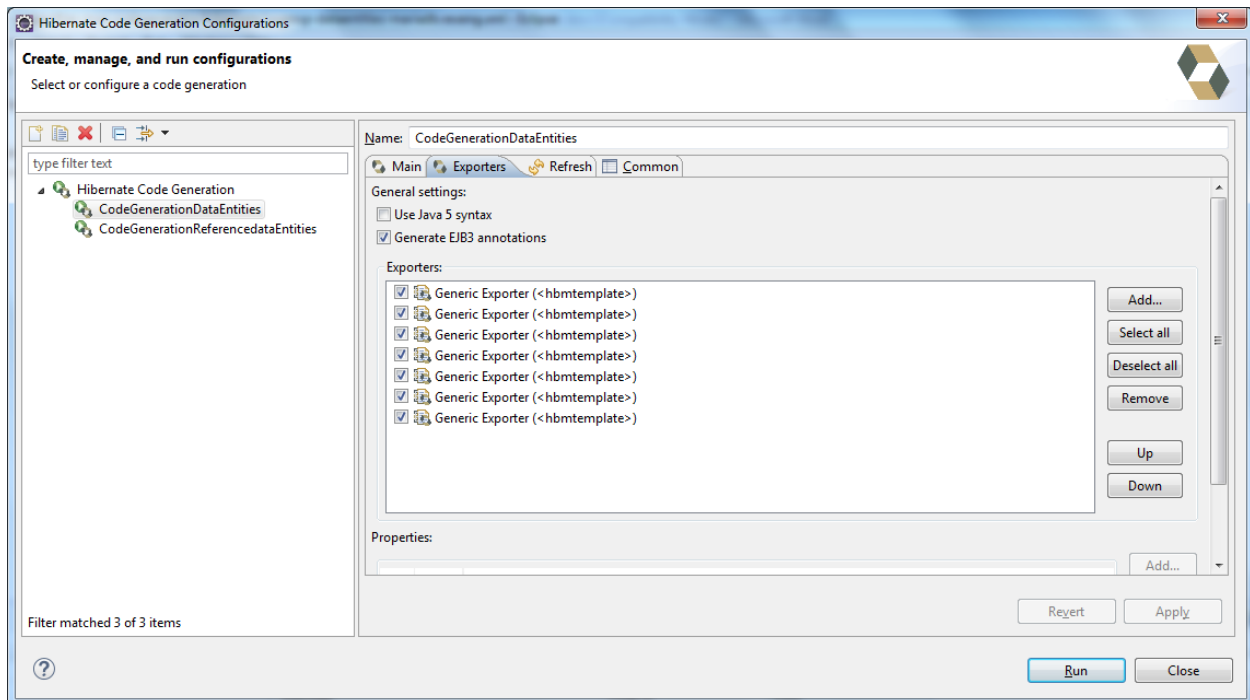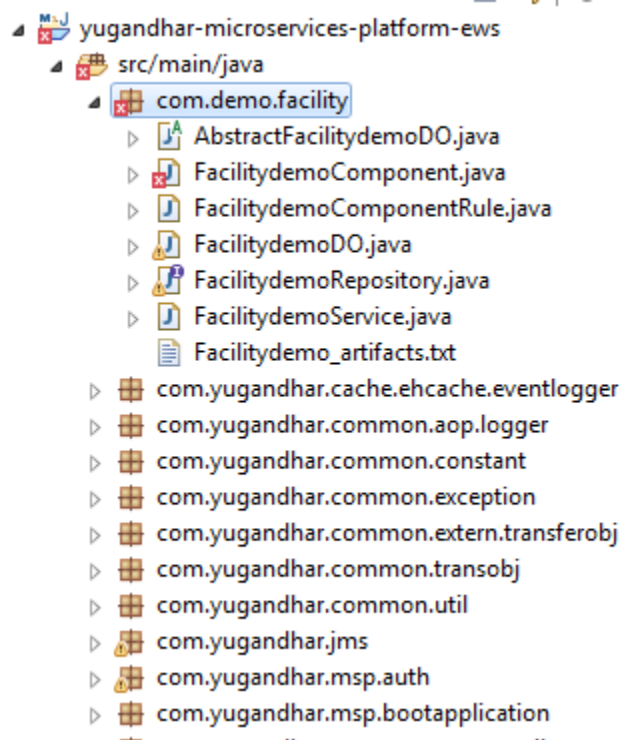
Click on Apply and Run. It will generate the artifacts in com.demo.facility package
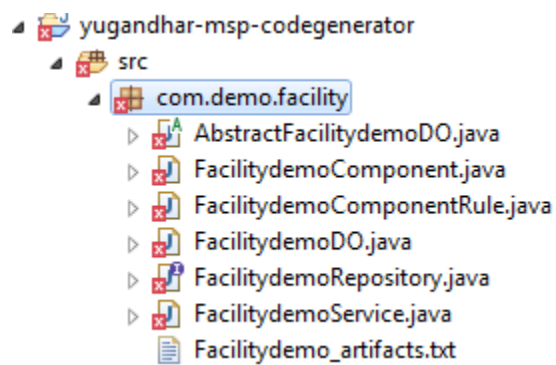
Plugging generated artifacts

Now add the generated classes to 'yugandhar-Microservice-platform-ews' project. You may choose to create a new Maven project and add the classes to the same and then refer that project to 'yugandhar-Microservice-platform-ews' project.

- yugandhar-microservices-platform-ews
  - src/main/java
    - com.demo.facility
      - AbstractFacilitydemoDO.java
      - FacilitydemoComponent.java
      - FacilitydemoComponentRule.java
      - FacilitydemoDO.java
      - FacilitydemoRepository.java
      - FacilitydemoService.java
      - Facilitydemo_artifacts.txt
    - com.yugandhar.cache.ehcache.eventlogger
    - com.yugandhar.common.aop.logger
    - com.yugandhar.common.constant
    - com.yugandhar.common.exception
    - com.yugandhar.common.extern.transferobj
    - com.yugandhar.common.transobj
    - com.yugandhar.common.util
    - com.yugandhar.jms
    - com.yugandhar.msp.auth
    - com.yugandhar.msp.bootapplication

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 YugandharMsplatformApplication
- **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',
```
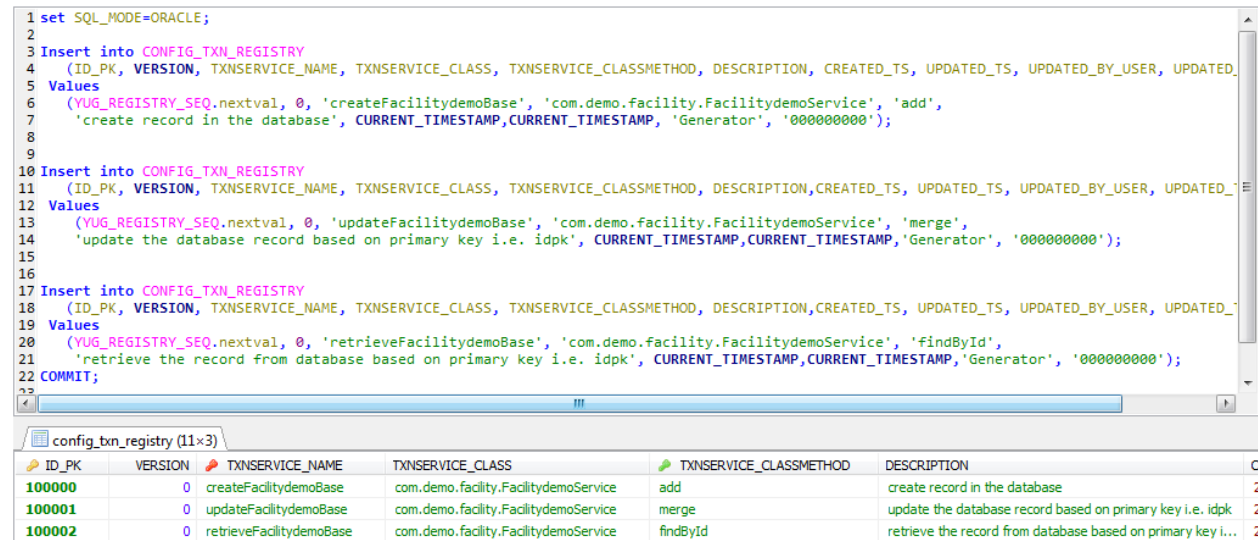
```
'com.demo.facility.FacilitydemoService', 'findById',
    'retrieve the record from database based on primary key i.e. idpk',
CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,'Generator', '000000000');
COMMIT;
```

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.

Step 2. Make an entry in make entry in YugandharMsplatformApplication

Modify the entry in YugandharMsplatformApplication 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.*"})
```

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;
     }
```
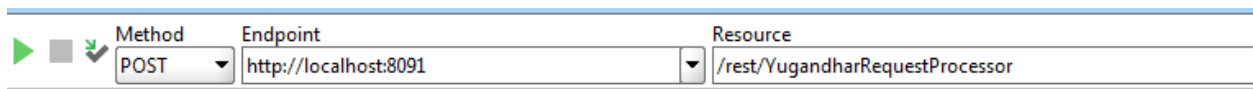
Test using SOAPUI

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

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

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.

Likewise execute the update

Request 1 — POST http://localhost:8091 /rest/YugandharRequestProcessor

Request body:
```
    "requestOriginSource": null,
    "requesterSystem": null,
    "transactionServiceName": "updateFacilitydemoBase"
    },
    "txnPayload": {
            "facilitydemoDO":            {
            "idPk": "bfd2aee2-7255-4780-8ec9-99d87b3eb3ad",
    "version": 0,
    "updatedByUser": "admin",
    "updatedByTxnId": "1234567890123",
    "facilityName": "MY HOUSE",
    "location": "LANE2"
    }

    }
```

Response:
```
1  {
2      "responseCode": "SUCCESS",
3      "txnHeader":      {
4          "requesterLanguage": "1",
5          "userName": "admin",
6          "userRole": "admin",
7          "txnMessageId": "1234567890123",
8          "transactionServiceName": "updateFacilitydemoBase",
9          "totalExecutionTimeMillies": "172"
10     },
11     "txnPayload": {"facilitydemoDO":      {
12         "idPk": "bfd2aee2-7255-4780-8ec9-99d87b3eb3ad",
13         "version": 1,
14         "createdTs": "2018-05-29T08.04.42.963+0000",
15         "updatedTs": "2018-05-29T08.05.54.043+0000",
16         "updatedByUser": "admin",
17         "updatedByTxnId": "1234567890123",
18         "facilityName": "MY HOUSE",
19         "location": "LANE2"
20     }}
21  }
```

Headers:
Header	Value
Accept	application/json
Content-Type	application/json

And retrieve as well

Request 1 — POST http://localhost:8091 /rest/YugandharRequestProcessor

Request body:
```
    "accessToken": null,
    "txnCorrelationId": null,
    "txnMessageId": "1234567890123",
    "requestOriginSource": null,
    "requesterSystem": null,
    "transactionServiceName": "retrieveFacilitydemoBase"
    },
    "txnPayload": {
            "facilitydemoDO":            {
            "idPk": "bfd2aee2-7255-4780-8ec9-99d87b3eb3ad"

    }

    }
```

Response:
```
1  {
2      "responseCode": "SUCCESS",
3      "txnHeader":      {
4          "requesterLanguage": "1",
5          "userName": "admin",
6          "userRole": "admin",
7          "txnMessageId": "1234567890123",
8          "transactionServiceName": "retrieveFacilitydemoBase",
9          "totalExecutionTimeMillies": "31"
10     },
11     "txnPayload": {"facilitydemoDO":      {
12         "idPk": "bfd2aee2-7255-4780-8ec9-99d87b3eb3ad",
13         "version": 1,
14         "createdTs": "2018-05-29T08.04.42.963+0000",
15         "updatedTs": "2018-05-29T08.05.54.043+0000",
16         "updatedByUser": "admin",
17         "updatedByTxnId": "1234567890123",
18         "facilityName": "MY HOUSE",
19         "location": "LANE2"
20     }}
21  }
```

Headers:
Header	Value
Accept	application/json
Content-Type	application/json

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

```sql
select * from FACILITYDEMO;
```

ID_PK	VERSION	CREATED_TS	DELETED_TS	UPDATED_TS	UPDATED_BY_USER	UPDATED_BY_TXN
bfd2aee2-7255-4780-8ec9-99d87b3eb3ad	1	2018-05-29 13:34:42.963000	(NULL)	2018-05-29 13:35:54.043000	admin	1234567890123

facilitydemo (9×1)

```sql
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'

al_facilitydemo (12×2)

AUDITLOG_ID_PK	AUDITLOG_CREATED_TS	AUDITLOG_ACTION_CODE	ID_PK	VERSION	CREATED_TS
1adf99a1-6317-11e8-8cda-c85b764d5d1a	2018-05-29 13:35:54.123658	U	bfd2aee2-7255-4780-8ec9-99d87b3eb3ad	1	2018-05-29 1
f07ad472-6316-11e8-8cda-c85b764d5d1a	2018-05-29 13:34:42.970658	I	bfd2aee2-7255-4780-8ec9-99d87b3eb3ad	0	2018-05-29 1

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 MSP; you may take a look of the MDM Hub to understand how a great Microservice application can be built using Yugandhar Microservice platform.