Yugandhar Microservice Platform

Architectural Overview

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 architectural overview of Yugandhar Microservice Platform.

Architectural Overview of Yugandhar Microservice Platform

Yugandhar Microservice Platform 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.

Capabilities and Salient Features

- Proven architecture for building Microservice.
- Supports embedded Web Server as well as JEE container web servers.
    - MSP for EWS: The Yugandhar MSP for EWS runs with Springboot embedded web Server like tomcat.
    - MSP for JEEC: The Yugandhar MSP for JEEC is built to be run on a Java Enterprise Edition Container (Web Server) like Red Hat Jboss.
- Preconfigured for Maria DB and Oracle, can be extended to support other databases like MySQL, DB2, NoSQL, PostgreSQL etc.
- Fully configured JTA transaction manager, ORM framework and activeMQ integration.
- Rapid Development using Yugandhar Code generation Templates
- Preconfigured and customizable UUID based Primary Key generator
- Embedded and extendable data validations and business rules
- Multilingual Reference data (List of Values) management Capability
- preconfigured application and performance logging using logback
- User and role based Authorization
- Pagination feature for retrieve and search services
- Comprehensive Audit Log framework
- jsr107 compliant Ehcache based caching framework
- DevOps compliance and Maven Integration

Architecture – Deep Dive

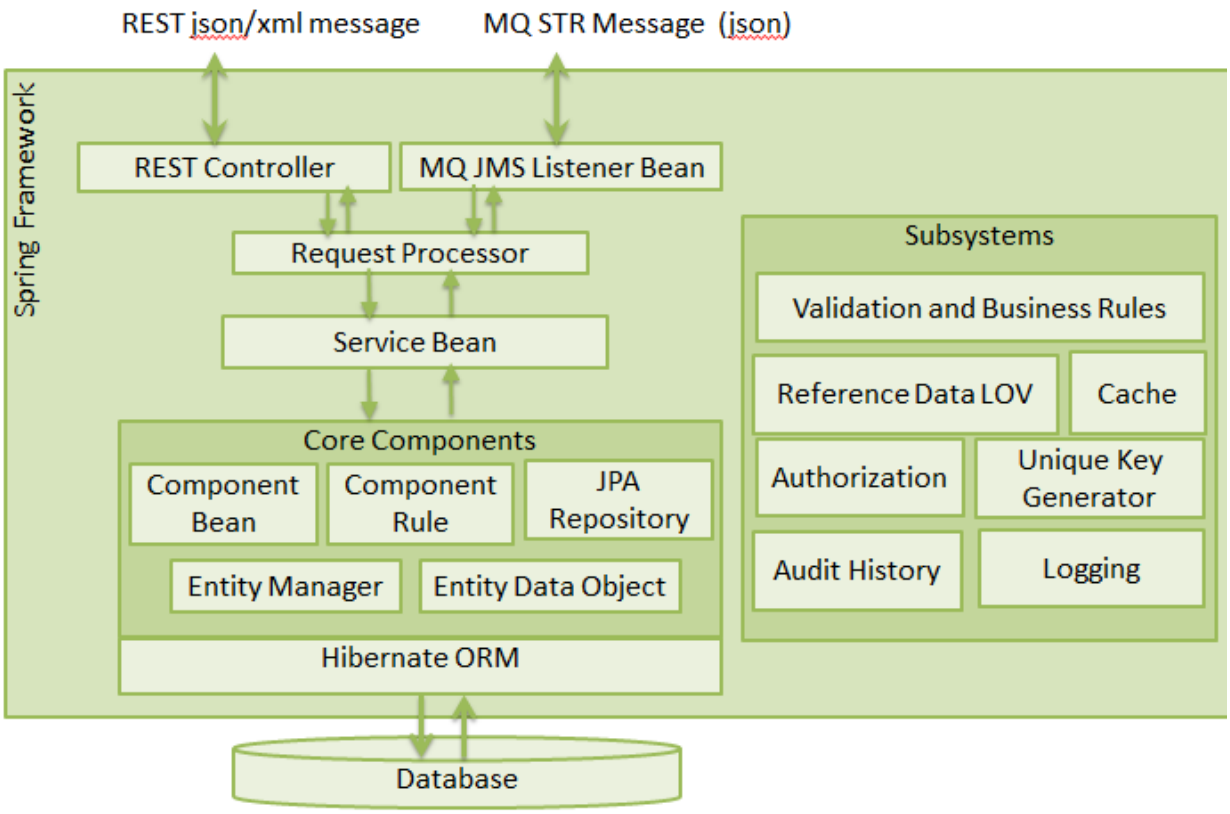

Fig. Yugandhar Microservice platform Architecture

Yugandhar Microservice Platform is having highly simplified framework as covered in the above diagram. Let's understand the components in brief

Interface Components – RESTFul and JMS

REST Controller

Yugandhar Microservice Platform provides default REST controller named DefaultJsonRESTController. It's present in the package com.yugandhar.rest.controller and is configured to accept application/json and application/xml messages using POST method. This can be invoked using REST client using below default endpoints which can be configured as per the requirement.

JEEC endpoints: http://<hostname>:<port>/yugandhar-Microservice-platform-jeec-<version>/rest/YugandharRequestProcessor

e.g http://localhost:8091/yugandhar-Microservice-platform-jeec-1.0.0.RELEASE/rest/YugandharRequestProcessor

EWS: http://<hostname>:<port>/rest/YugandharRequestProcessor

e.g http://localhost:8091/rest/YugandharRequestProcessor

Please note – The above link is for your reference only. The actual link may vary if you change the port, application name, version, request mapping of the REST controller etc.

After receiving the message, this controller sends the message to request processor for further processing.

JMS Listener

The JMS listener bean is located under com.yugandhar.jms package named

YugDefaultRequestQueueListener. After receiving the message, this Listener sends the message to request processor for further processing.

EWS: This listener bean is mapped to destination YUG.DEFAULT.REQUEST on embedded MQ.

JEEC: This listener is mapped to destination="java:jboss/com/yugandhar/default/requestQueue" which is mapped to YUG.DEFAULT.REQUEST on JBoss ActiveMQ.

Request Processor

The request processor is Service type of Spring Bean which is single entry point of invoke transactions on the Hub server. The processMessage() method of request processor reads the incoming message and invokes the appropriate service bean as mentioned in the txnTransferObj.txnHeader.transactionServiceName attribute. Please refer the API reference guide for the structure of transfer object and learn how to invoke transactions on Microservice platform.

Core Components

The core components of the MSP include Service and Component Beans, Business Rule Class, JPA Repository class, Entity Data Objects (referred as DO) and JPA injected Entity Manager.

Service Bean

This is a Service bean of spring framework and the request from request Processor would always come to Service bean for further processing. The Service bean is a logical layer between RequestProcessor and Component bean introduced to create base as well as composite transaction and infuse any additional business logic so to insulate Component beans from having any business logic.

Component Bean

The Component beans handle the entity level data CRUD (Create, Update, Retrieve, Delete) operations. This bean refers and makes use of Entity Data Objects (DO), entity manager and JPA repository classes.

Component Bean primarily includes below methods

Persist() –This method is used to create a new record in a database entity.

merge() –This method is used to update existing record in a database entity based on primary key.

findById() –This method is used to retrieve existing record in a database based on primary key.

findByBusinessKey() – This method is used to search a database table based on business key e.g. firstname and lastName from Person table or key and config language code attributes of the reference table

Entity Data Object Classes

The entity data objects are used by hibernate to map the DO to database entities. We have bifurcated the DOs in AbstractDO and Child DO. The Yugandhar Template will generate the AbstractDO and ChildDO and the thumb rule is that the custom attributes and methods must to be added to Child DOs and Abstract DO be kept intact as generated by the template. This would keep the code clean and any further enhancement by Yugandhar team in the templates in future releases can be seamlessly incorporated in your code.

Optimistic Lock

To avoid concurrent update of the same record, MSP use optimistic lock based on VERSION attribute of the database. Yugandhar MSP relies heavily on Hibernate Optimistic lock feature.

JPA Repository Class

The JPA repository is implementation of org.springframework.data.jpa.repository.JpaRepository interface used primarily to search the data from the single table based business keys.

ComponentRule class

The component rule class used to write the business rules for a specific entity transaction based on the method from component bean is invoked. The entity transaction can have rules. The business rules are defined at below levels based on the transaction

Rule cross points for Persist() method:

prevalidatePersit

preExecutePersist

postExecutePersit

Rule cross points for Merge() method

PrevalidateMerge

preExecuteMerge

postExecuteMerge

Rule cross points for FindById()

postExecuteFindById

prevalidateFindById

Rule cross points for FindByBusinessKey() method

postExecuteFindByBusinessKey

prevalidateFindByBusinessKey

The developer can incorporate their business rules at any of the above methods as per requirements.

JPA injected Entity Manager

This is the default PersistenceContext used by Spring framework to make database interactions.

Yugandhar MSP Data Model

Yugandhar Microservice Platform comes with basic data entities for application configuration like application properties, error codes, service registry, authorization framework and few Reference Lists like Country codes, Currency and language codes.

Refer the Database documentation to understand more..

Subsystems

Validation and Business Rules

Validation and Business rules are defined in Component rule classes. Yugandhar Microservice Platform provides several Out of the Box (OOTB) business validations for Create, update, retrieve, find and search transactions of every entity. These rules are extendable and developer can write their own code directly in the rule class or can use Aspect Oriented programming (AOP) feature of Spring framework to add completely new logic or change the logic already written in OOTB code.

Cache

Yugandhar Microservice Platform cache the Reference Data values (List of values stored as key-value pairs) so to improve transaction response time. It uses ehcahce an open source, jsr 107 compliant cache framework that boosts performance, offloads your database, and simplifies scalability. It's the most widely-used Java-based cache because it's robust, proven, full-featured, and integrates with other popular libraries and frameworks. The ehcache configuration file is kept in /src/main/resources/ehcache.xml of the yugandhar-Microservice-platform project. The default ehcache expiry time is configured as 1296000 seconds (15 days) in the ehcache.xml configuration file. This may be revisited as per the requirements. Visit www.ehcache.org for understanding ehcache in further detail.

Reference Data (LOV) Management

Yugandhar Microservice Platform supports storing the reference data Management and have standard pre-defined format for storing the key-value pairs. This format is extendable as per requirements. The typical Reference data table starts with the name REF_ e.g. REF_COUNTRY_ISO which is a LOV which stores the ISO Country Codes which can be referred while storing address or country of domicile etc. The reference data table has CONFIG_LANGUAGE_CODE_KEY, KEY, VALUE and DESCRIPTION attributes as predefined format.

The CONFIG_LANGUAGE_CODE_KEY attribute provides multilingual support. e.g. The country 'France' in English is written as फ्रांस in hindi and as 'francés' in Spanish. So the LOV can be populated as

CONFIG_LANGUAGE_CODE_KEY	KEY	VALUE	Description
1 (for English)	250	FRA	France
2 (For Hindi)	250	FRA	फ्रांस
3 (For Spanish)	250	FRA	'francés'

While executing the transaction, the request header is provided with a mandatory attribute named request language which signifies the language in which the response needs to be sent. This way, based on the requestor Language of the transaction the relevant record can be fetched from the database and returned in the response.

Authorization Framework

Yugandhar Microservice Platform provides user and role based Authorization framework. The tables AUTH_ROLES_REGISTRY, AUTH_USER_REGISTRY, AUTH_USER_ROLE_ASSOC and AUTH_USERROLE_ACCESSCONTROL comprise of the Authorization framework.

AUTH_ROLES_REGISTRY – stores the list of roles

AUTH_USER_REGISTRY – stores the list of users

AUTH_USER_ROLE_ASSOC – stores the user to role mapping

AUTH_USERROLE_ACCESSCONTROL – Stores if the user or role is authorized to perform given transaction. List of transaction is referred from CONFIG_TXN_REGISTRY table and the ID_PK of CONFIG_TXN_REGISTRY table is referred as FK to provide the access.

Refer Database documentation for further understanding of the data model. Also refer the Authorization framework section of the API reference guide to understand the OOTB transaction used to make entries these tables.

Unique key (Primary key) Generator

Yugandhar Microservice Platform currently uses UUID based Primary key generator. This implementation class of the yugandhar key generator is AbstractYugandharKeygenerator. However if you want to customize the Unique key generation logic then you may write your custom logic in the below class

com.yugandhar.mdm.keygen.YugandharKeygenerator

The Primary key generation logic is externalized using YugandharKeygenerator class mentioned above. Please make sure that you should not modify the AbstractYugandharKeygenerator as it is supposed to be modified only by Yugandhar team.

Logging

Yugandhar Microservice Platform have comprehensive Logging framework for application logging. The logback configuration file is kept in /YugandharBootProject/src/main/resources/yugandhar_logback.xml location in the code base. By default the SizeAndTimeBasedFNATP rolling file appeder is used for the logging but this can be modified as per requirements. Please refer below url to learn more about logback https://logback.qos.ch/

Audit History

Yugandhar Microservice Platform advises a data model to store the Audit log wherein each entity table should have Audit Log table appended with 'AL_' which stores all the Insert, Update and Delete related changes to the base table. The rows should be populated based on insert, update and delete triggers in the main table. This way one additional row gets created in 'AL_' tables after every Insert, Update and Delete on the record of the base table so that the complete history of changes on the data record gets tracked.

Rapid Development using Code generation Templates

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. Refer the code generation guide for more details.