Openl40™ maven packages guide

Sommario

OpenI40
OPEN SOURCE INDUSTRY 4.0

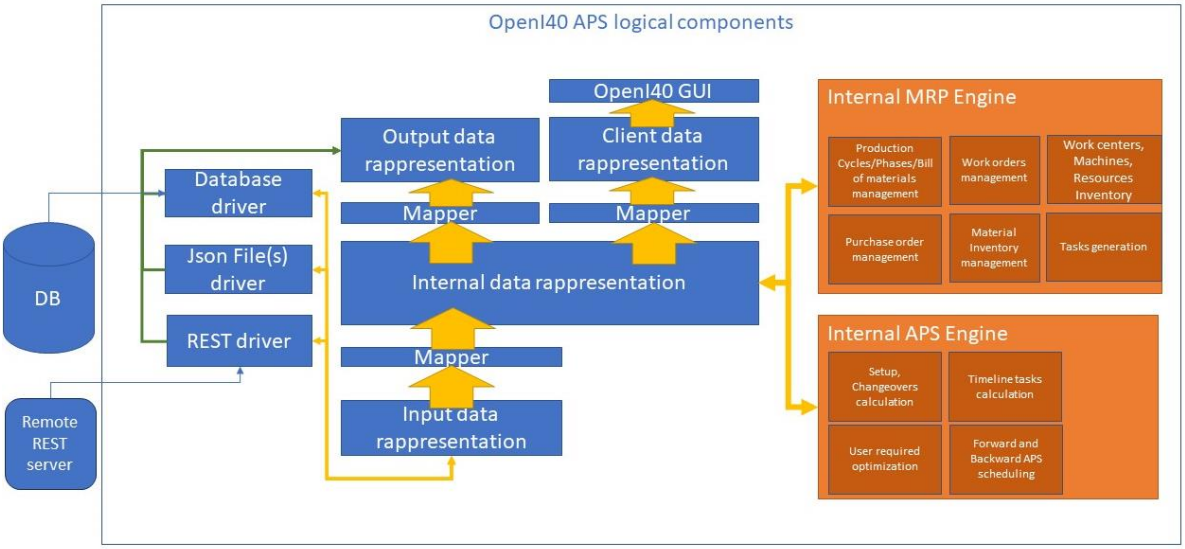
Introduction

This project has been split in various maven packages not only vertically “per application stage” but also “per application stage subsystem” to provide multiple way of working of each subsystem and also a lot of potential design patterns to integrate its subsystems or all the architecture in a 3rd party software.

The overall architecture is split into multiple “parents” maven modules with submodules splitting the platform in various subsystems with multiple “spring boot applications” per parent module reflecting various deployment options:

- “macro services” (not micro because of the size 😊) that can communicate together
- “monolithics applications” that aggregate multiple services working together to offer an easier deployment scheme for a “final full working system”.

Partial architecture overview (APS only)



OpenI40™ maven packages and subsystems

Packages versioning: all maven package has the same version as the main parent, so no issues and complications in having different subsystem versions occurs, we ensure that using unit testing and proper “git flow” management the current version in “development branch” is at least as stable to run 100% of all submodule unit testing.

Parent packages

- Main pom.xml “super parent” ➔ **com.openi40:com.openi40.parent** delineates the overall shared dependencies with spring boot environment and various subsystem versions
- openi40.commons/pom.xml ➔ **com.openi40.commons:com.openi40.commons.parent** common shared modules delineating the basic functionalities and subsystem configurations for all the platform applications behaviours regarding:
    - mapping between data layers
    - REST data mapping
    - Asynchronous services running
    - Web subsystem rest and other services initializations
    - Junit tests shared functionality
- openi40.scheduler/pom.xml ➔ **com.openi40.scheduler: com.openi40.scheduler.parent** scheduling engine parent module, its submodules implements scheduling engine in memory model and business logics, input data model, output data model, aps production messages data model and UI data model with all “mappers between modules” with also apache-ignite HA in memory clustering features.

- openi40.ui/pom.xml ➔ **com.openi40.ui:com.openi40.ui.parent** scheduling angular UI and backoffice UI packaged as maven submodule with building integrated in maven compilation lifecycle to be published by the bootable applications.
- openi40.platform/pom.xml ➔ **com.openi40.platform: com.openi40.platform.parent** various submodules for: mapping to database, database models, remote rest integrations, rest database server for separate data layer deployments, rest integration server for 3rd party applications integration, database to database direct integration module for 3rd party applications, complete standalone application with all stack components, complete standalone HA application with all stack components.
- openi40.mes/pom.xml ➔ **com.openi40.mes:com.openi40.mes.parent** mes submodules and kernel subsystem, architecture studied to be the base for production to aps integration kernel and mes to aps integration kernel with IIOT/MQTT/REST capabilities and base for the future mes subsystem development.

Commons modules: com.openi40:com.openi40.parent

Submodule	Description
Mapper	Base mappers infrastructure to map between different data stages/rappresentations
Multithreading-config	Basic configuration reused in spring boot applications to configure nr. Of thread for multithreading, workers, async handling threads
openi40-tests-support	Some basic JUNIT reusable utilities
Persistence-datamodel-utils	Some fast persistence classes and utilities

Scheduling core and standalone scheduler: com.openi40.scheduler: com.openi40.scheduler.parent

Submodule	Description
ApsDataCache	Manages the handling “in memory” of an aps data set
Client.Model	Data model to interact with UI and/or remote client that pilot scheduling
Client.RestApi	Scheduling REST api
Common	Common framework classes for scheduling layers
CommonDatamodel	Annotations and basic classes for data model rules and validation
Engine	Scheduling logics and engine
Engine.HA.Standalone	Spring bootable standalone scheduling engine with ignite in-memory cache to handle high availability configurable to load datasets from a JSON datasource or remote REST services. Serves also the angular UI on /openi40/ web URL and swagger UI on /openi40/swagger-ui.html
Engine.Standalone	Spring bootable standalone scheduling engine configurable to load datasets from a JSON datasource or remote REST services. Serves also the angular UI on /openi40/ web URL and swagger UI on /openi40/swagger-ui.html

Input.IOMessages	Models for event messages contents coming from production.
Input.Model	Data model for the loading stage of the APS engine
InputChannels	Layer abstraction for multiple loadable datasources via JSON files, remote REST services or REST services that a 3rd party application can load the data on the engine(extended by PersistenceInput module)
InputOutputConfig	Centralized configuration for multiple input data sources and multiple output data sources and "in memory" coherent aps data sets rappresentation.
MapperModules	Various submodules mapping the internal rappresentation of the "in memory" aps engine model to the UI model and the input and output models.
Model	Scheduling engine "in memory" rappresentation model
Model.Daos	Data access object services to insert/delete/update in memory engine data model rappresentation.
openi40-ignite-config	Ignite engine verticalization to support both ingesting the input model and persisting the output model from/to database but with distribuited "in memory" cache working in realtime.
Output.Channels	Layer abstraction for multiple datasources to save
Output.Model	Output model for scheduling results
SchedulerKernelArchitecture	Annotations and base architecture for the scheduling engine, with annotations handlers that create part of the infrasctucture at maven compile time. Designs a pattern for make business logic implementation classes customizable with alternative runtime versions switched accordingly to data model contents.
TestUtils	Test utilities used in scheduling engines junit's

Angular UI's for scheduling and aps backoffice: com.openi40.ui:com.openi40.ui.parent

Submodule	Description
openi40-scheduling-ui	Angular UI for scheduling and backoffice of platform data. The angular artifact is compiled and packaged in a maven jar that is served as statical content from depending bootable apps

Deployable platforms: com.openi40.platform: com.openi40.platform.parent

Submodule	Description
IOMessages.Processing	Module that implements the processing of spooled messages coming from production
openi40.app	Standalone openi40 spring bootable application with read/save from db capabilities, UI for scheduling and backoffice and backoffice + integration services
openi40.backoffice	Standalone backoffice services
openi40.dbmodel	JPA persistence database model for integration purposes
openi40.dbmodel.integration.app	Standalone database integration REST services bootable application
openi40.dbmodel.repositories	JPA repositories and REST services for integration purposes
openi40.generical.dbintegration	Configurable and extensible application for database to database integration
openi40.ha.app	Standalone high availability clusterizable application with database load/save, scheduling engine, scheduling UI and backoffice UI and services.
openi40.integration.clients	Swagger codegen generated integration clients for 3rd party applications
Persistence.Config	Platform persistence layer configuration
Persistence.Input.Channel	Data loading stage from persistent layer (database) via JPA streaming
Persistence.IOMessages.Controllers	REST Controller for ingestion of messages coming from production
Persistence.IOMessages.Spooling	Spooling functionalities for messages coming from production
Persistence.Output.Channel	Data saving stage for persistent layer (database)

Mes subsystem architecture and kernel: com.openi40.mes:com.openi40.mes.parent

Submodule	Description
`openi40.mes.iiot`	IIOT MES Kernel, various submodules implementing a “message passing” kernel application able to receive messages via MQTT/RABBITMQ/REST from devices and application and routing them also having the APS platform as endpoint.
`openi40.mes.workstation`	Various submodules implementing an application to integrate worcenters workstations and machines in the overall MES process.
`openi40-mes-shared-model`	Common MES persistent classes

`openi40-mes-shared-assets`	Abstraction layer to rappresents “assets” as workstations/sensors/machines/every IIOT device integrable directly or via 3rd party IIOT application.