Seamless Integrated Consistent Personalised

enactor®

retail systems for a digital world

Enactor Training Course Architecture Introduction

Architecture Introduction

Software Solutions for the Retail Sector

- Point of Sale, CRM & Loyalty, Store Inventory & Order Management

- Product Goals
  - Cover all digital channels
  - Consistency of customer interaction across channels
  - Right balance of decentralised and centralised function
  - Right balance of online/offline functions
  - Ease of integration
  - Ability to support rapid change

Key Architectural and Design Principles

- Service Oriented Architecture (SOA)
- Model View Controller (MVC)
- Object Oriented Design (OOD)
- Process based application development

Process Based Application Development

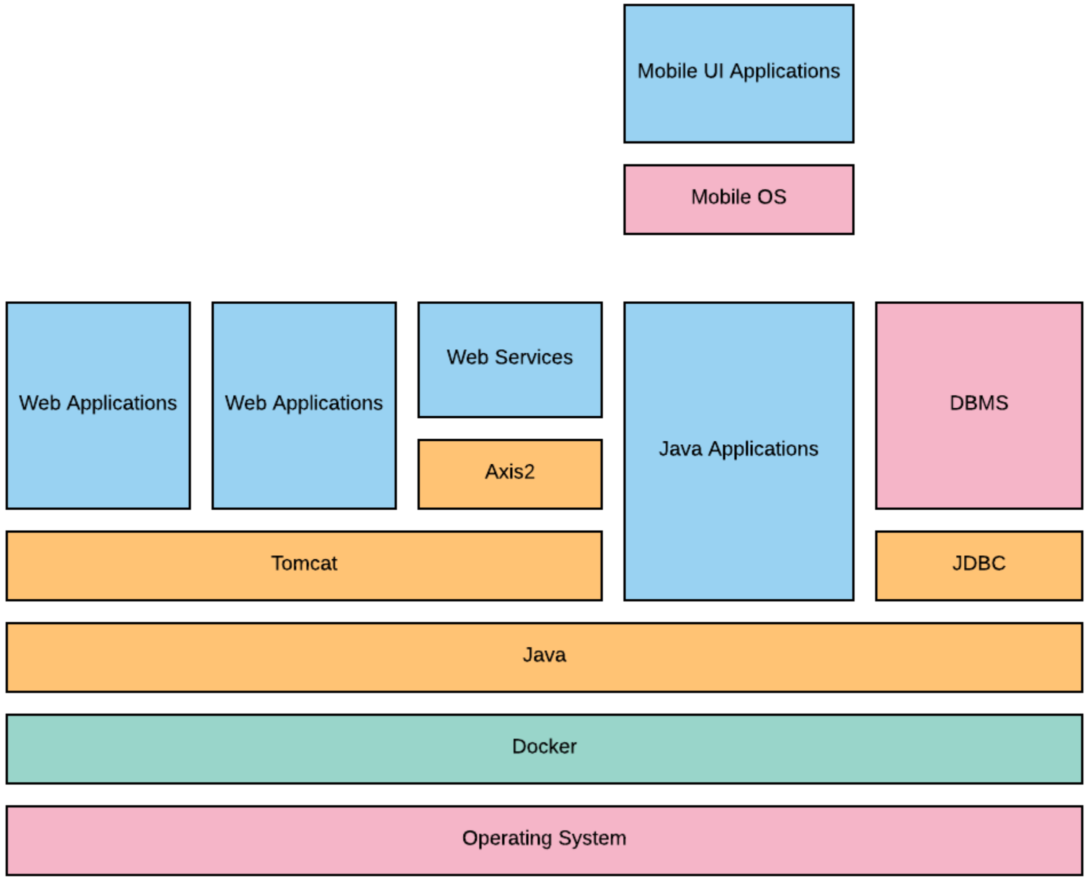
- Application Processes
  - Control flow and behaviour inside applications and services
- Business Processes
  - Workflow based, long-running interactions between applications, services and people
- Integration Processes
  - Web Service & Message Queue based integration configuration

Enactor Solution

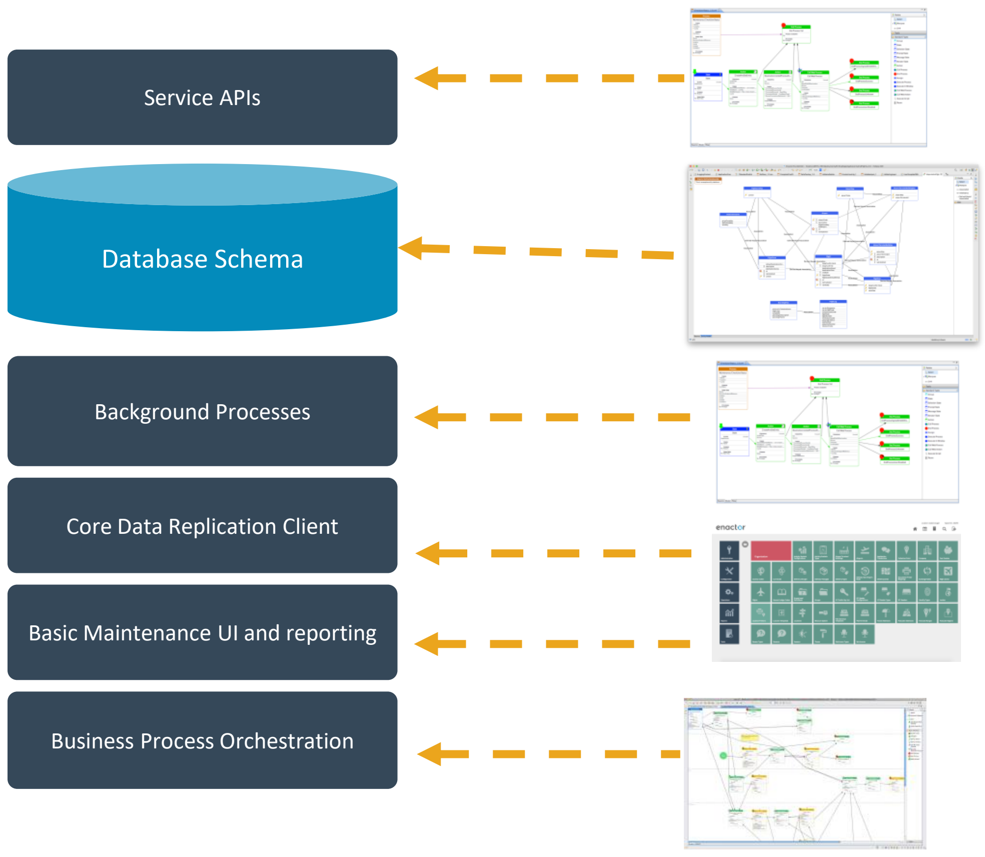
Front-End Applications	Store, POS, Mobile POS, Clientelling, Endless Aisle
Back-End Applications	Estate Manager, Inventory Manager, CRM
Services	SOA - Basket, Promotions, Loyalty, Rewards, Payment, Search
Library	Library of application components for building and enhancing applications – now over 10,000 Micro Services
Toolset	Toolset for allowing rapid application development and change
Platform	Java/J2EE, Digital Technologies, Web Services, ORM, Messaging, Mobile

Technology Stack

- Operating System
- Docker
- Java
- Tomcat
- Axis2
- JDBC/RDBMS
- Enactor Engine



Enactor Microservices Stack


Service APIs
Database Schema
Background Processes
Core Data Replication Client
Basic Maintenance UI and reporting
Business Process Orchestration

Enactor Application Processes implementing service API calls using Nano Services and using

1. Saga transaction management
2. Duplicate message detection, Circuit Breaker, Sharding
3. Materialised Views

Database schema implemented with Entities and Server Classes as part of Enactor's ORM layer using Entity Designer

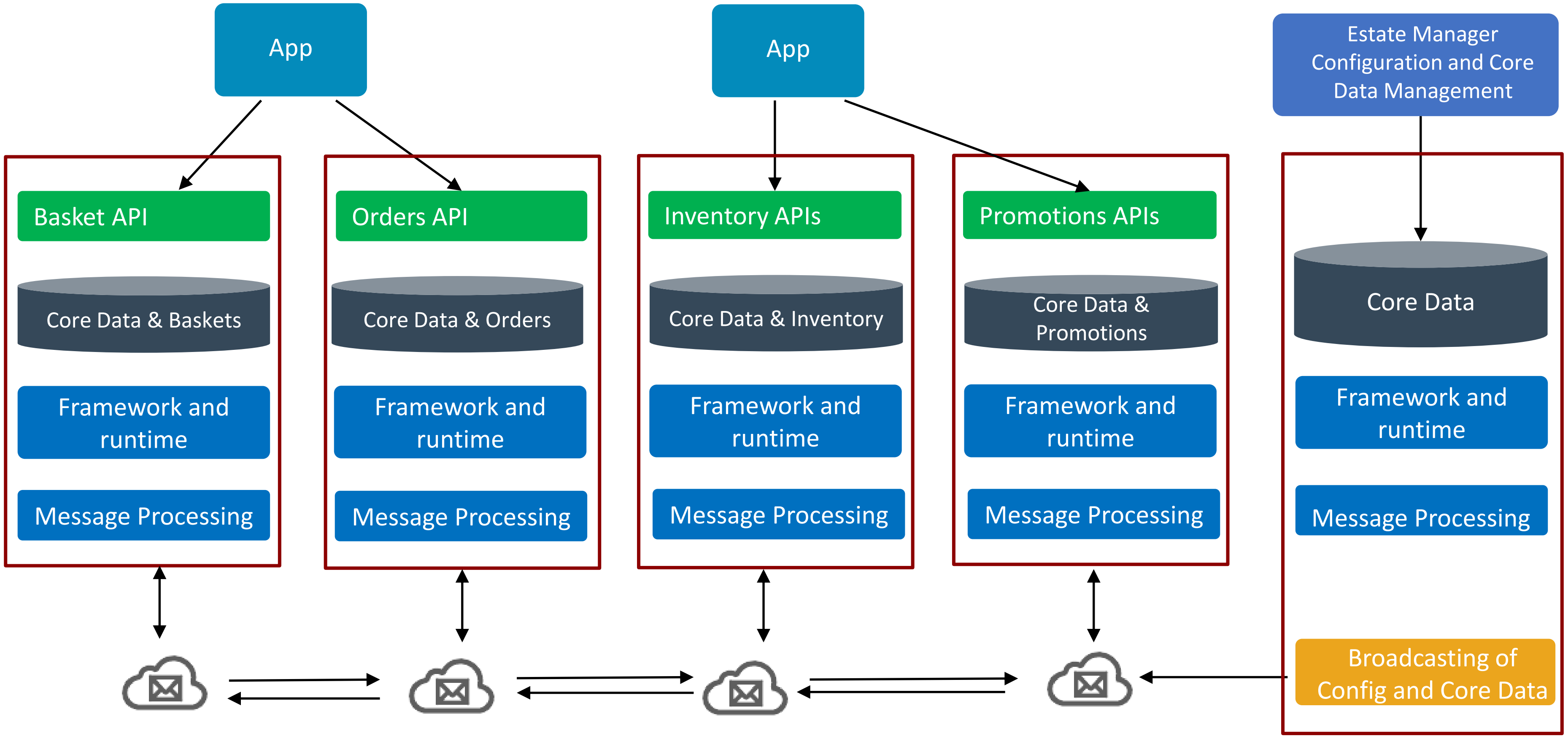
Processing and broadcast of messages using background threads running as Application Processes using Nano services

Data Broadcasting system from central repository

Enactor's Maintenance UI builders so the Business Domain Data is not a "black box"

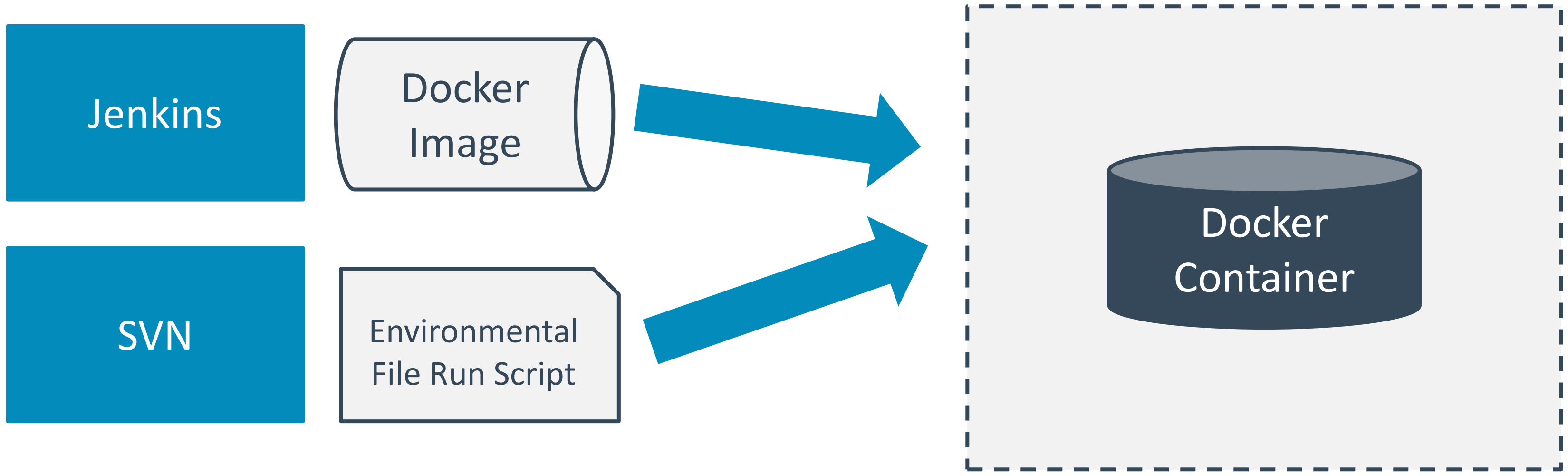
Enactor's Business Process Engine for long running processes that involve Human Tasks

The Core Enactor Microservices



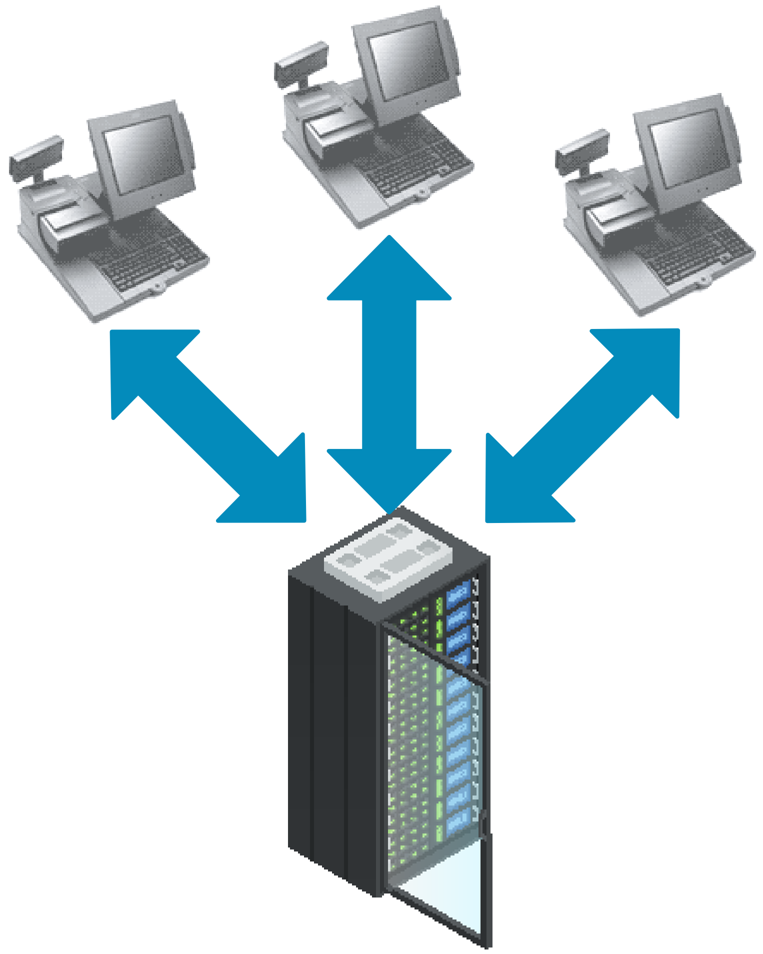
Docker and Enactor

- Why Docker?
- Eliminate “works on my machine” problem
- Makes maintaining CI machines much simpler
- Docker images and bundles provide a straightforward interface with Operations



Standard Deployment Configurations – POS and Estate Manager

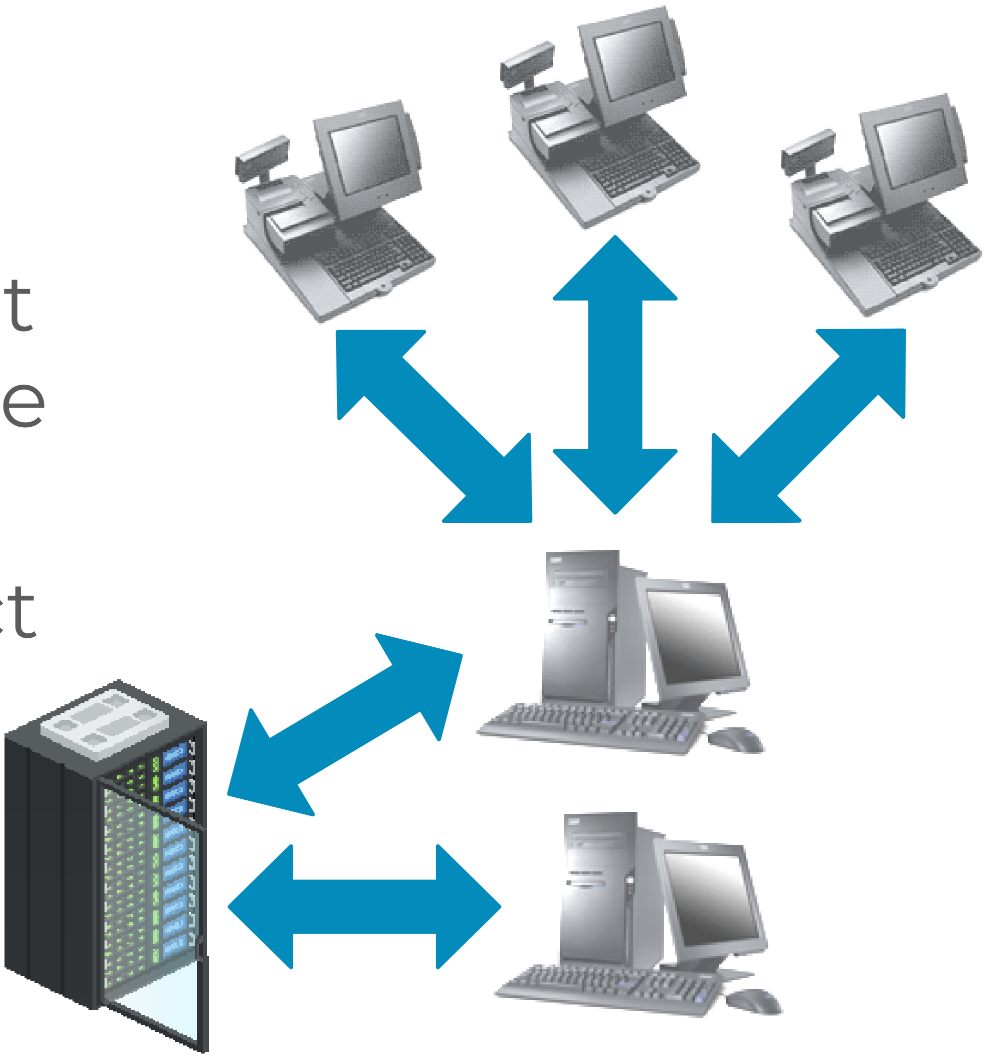
- All data is managed centrally at the Estate Manager
- Cash Management may be done locally
- Suitable where the store only has a few POS's or where the POS's are 'close' to the Estate Manager



Standard Deployment Configurations

POS, Back Office and Estate Manager

- Some data can be managed at the Store
- Cash Management may be done in the Store
- Back Office can act as a ‘proxy’ for the Estate Manager, reducing network traffic



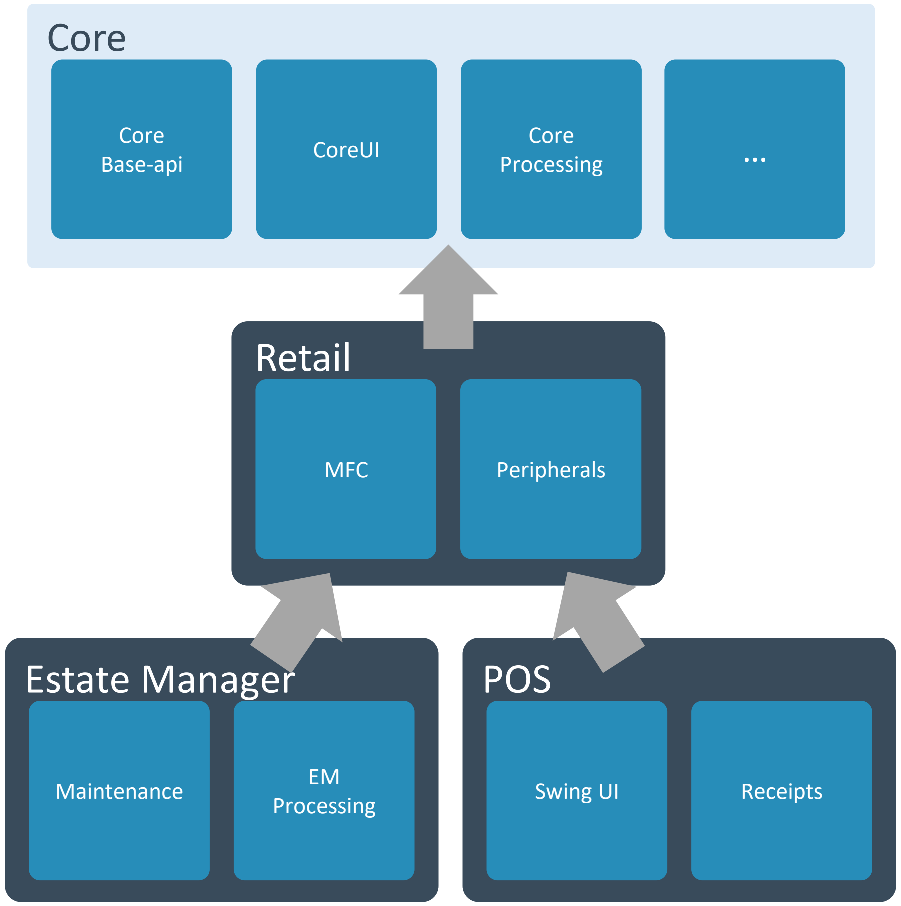
Software Architecture

Platform Software Structure

Enactor Software is organised into Components and Projects

Components describe a Functional or Technical boundary

Projects contain the source code and resources

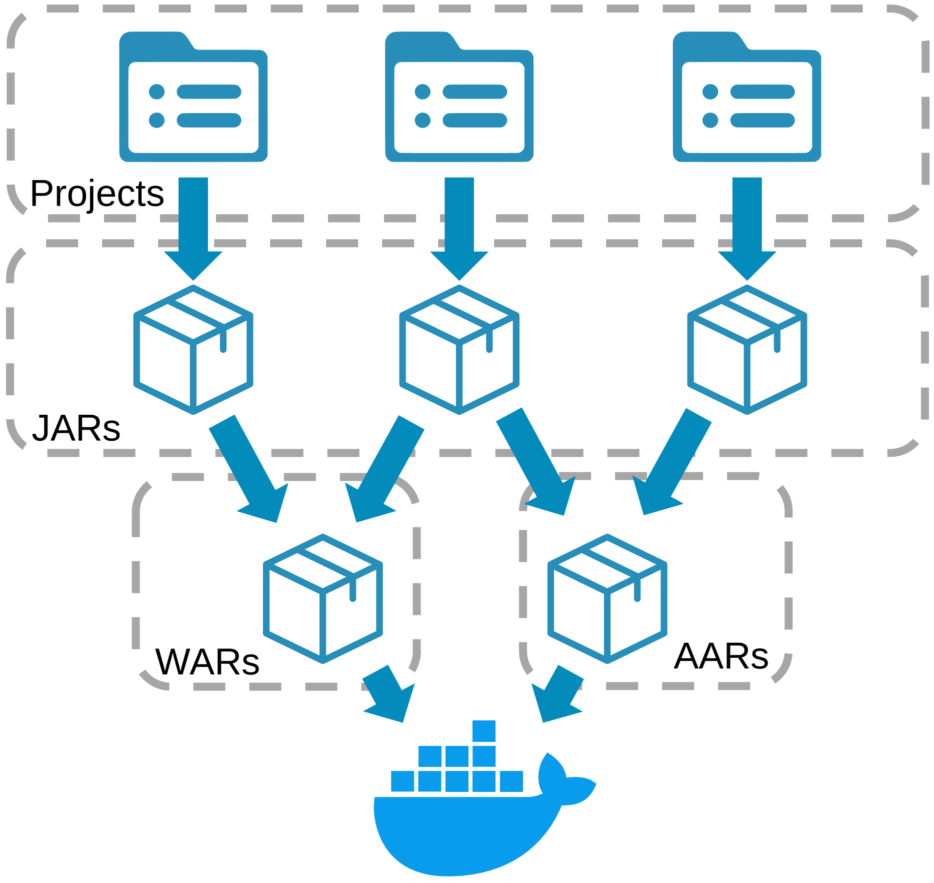


Platform Software Structure

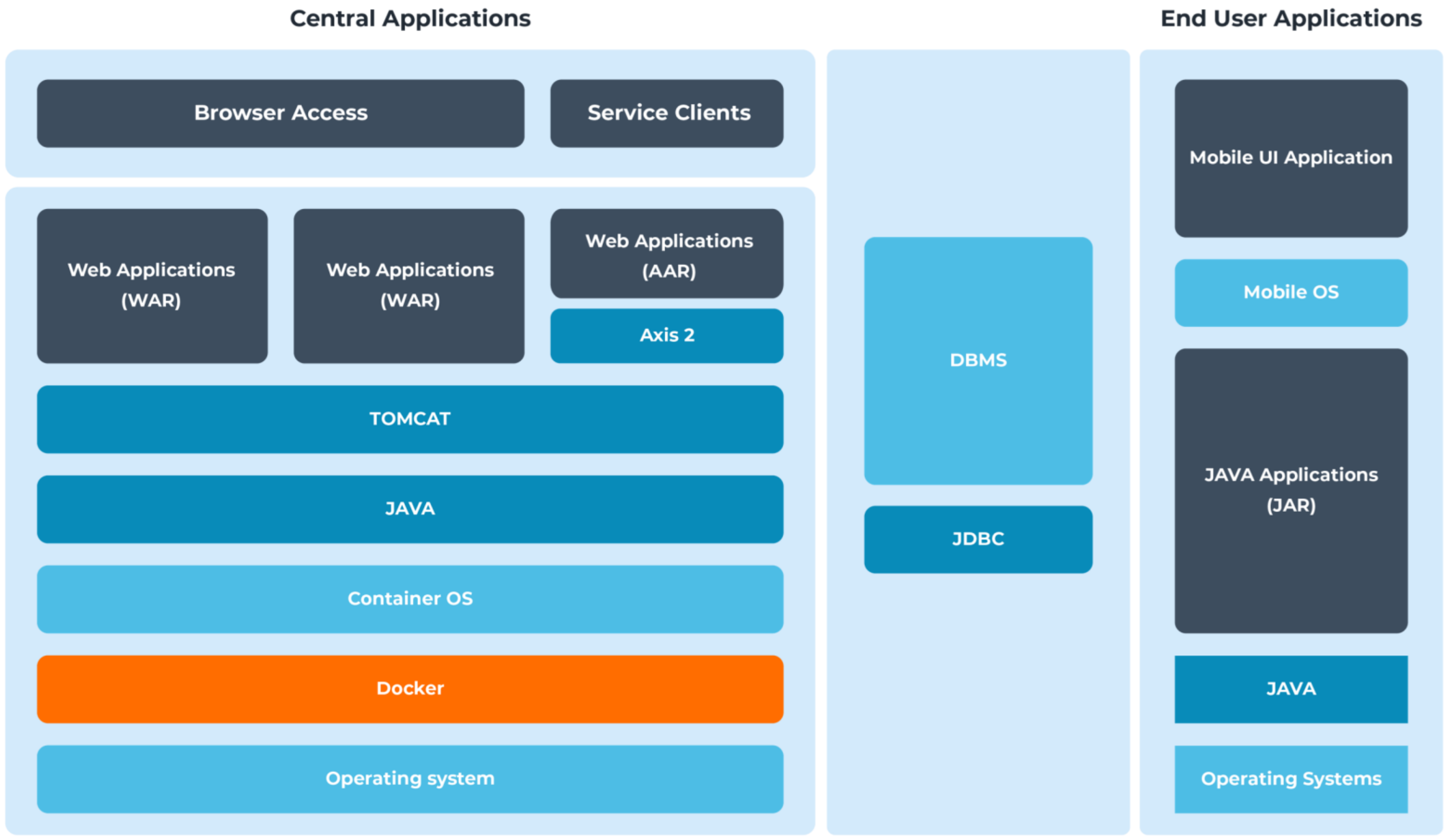
Projects are packaged into Java Archive (JAR) files

These are in-turn combined to form Web Application Archives (WAR), Axis2 Archives (AAR) and ZIP files

WAR and AAR files are further combined to form the standard Docker containers



Software Stack

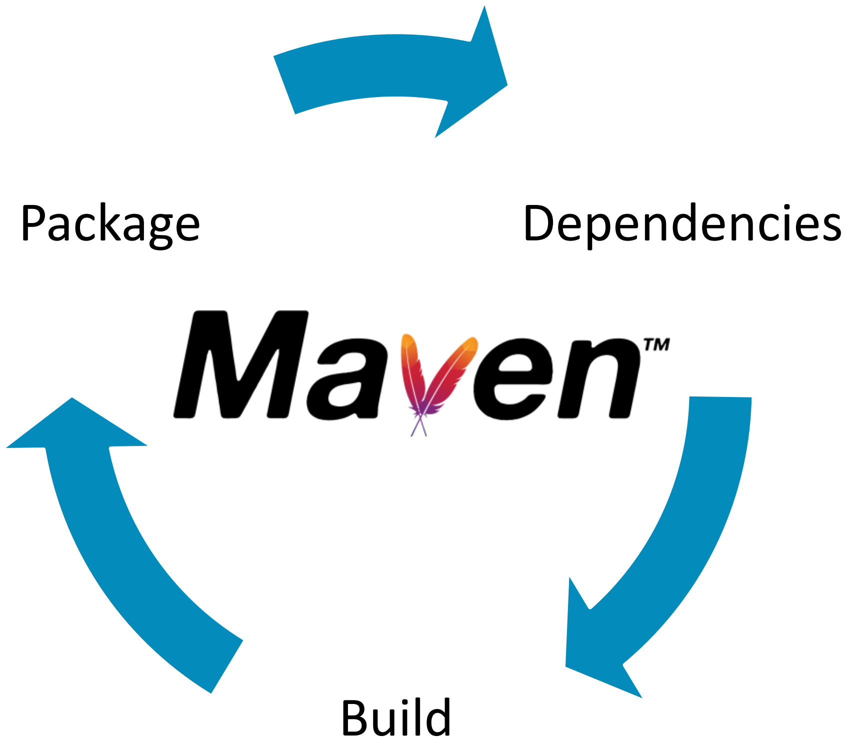


Dependency Management

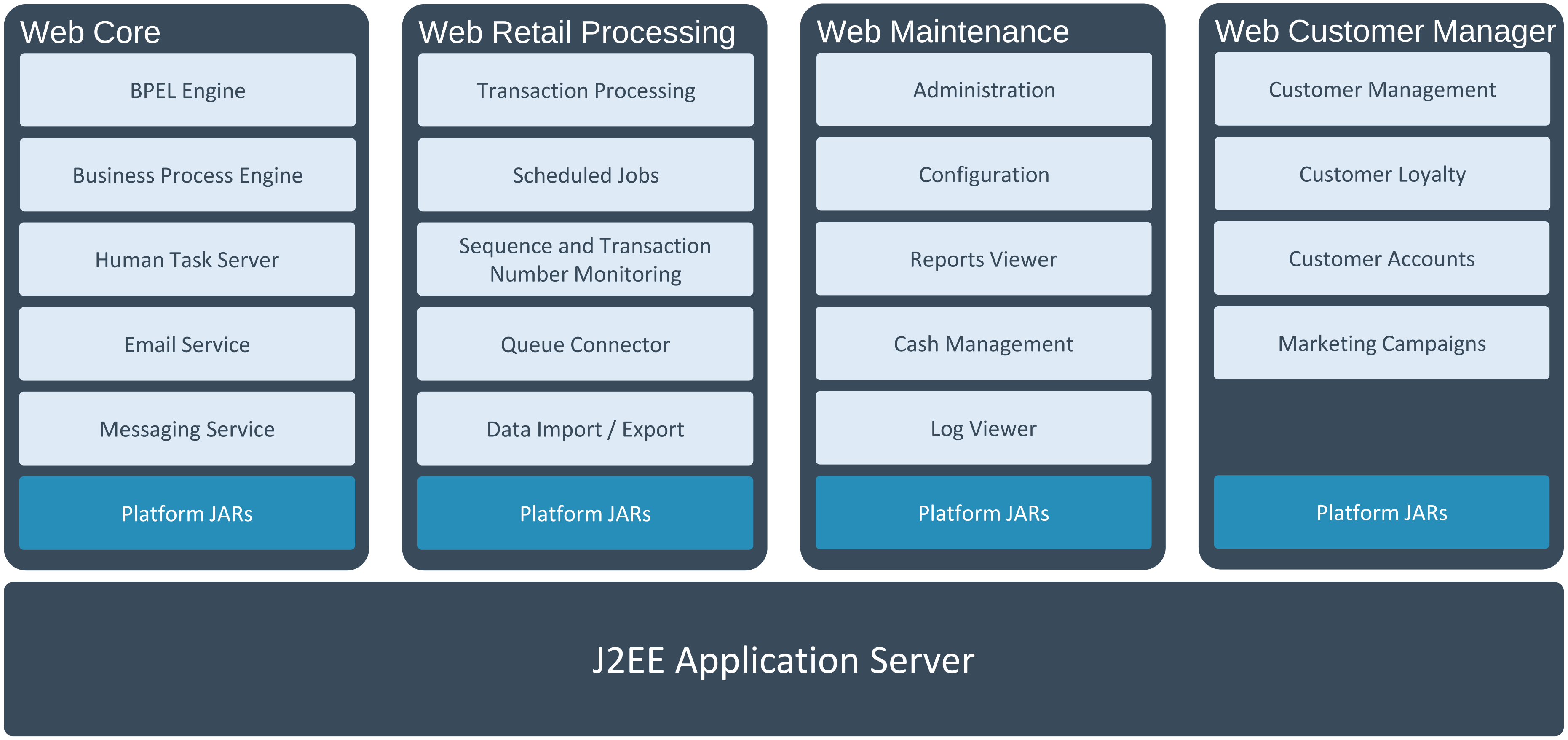
Project dependencies are managed using Maven

We also use Maven to compile and package the Projects

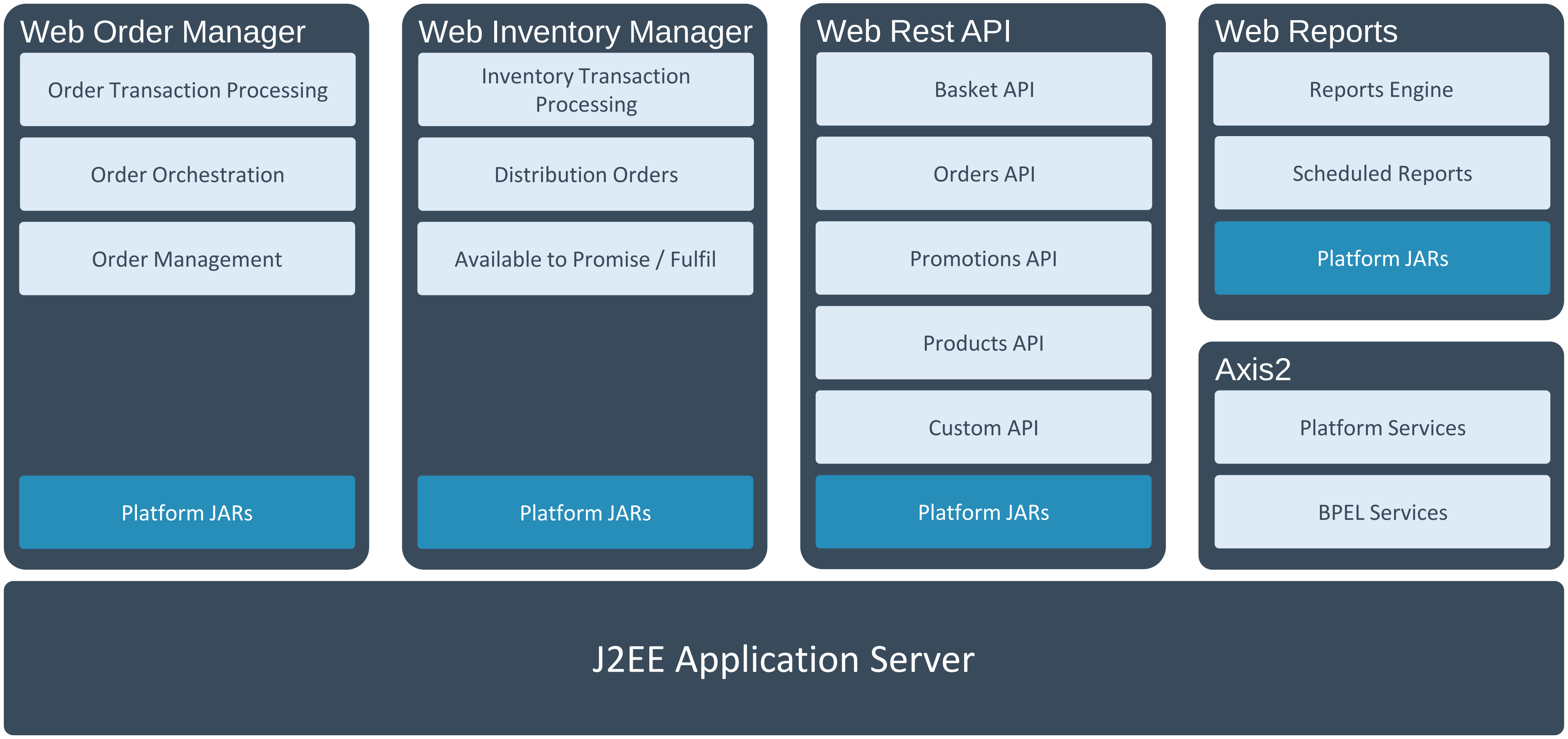
Dependency versions are centrally managed using a “Parent POM” so we can ensure that all projects share the same version



Platform Applications



Platform Applications

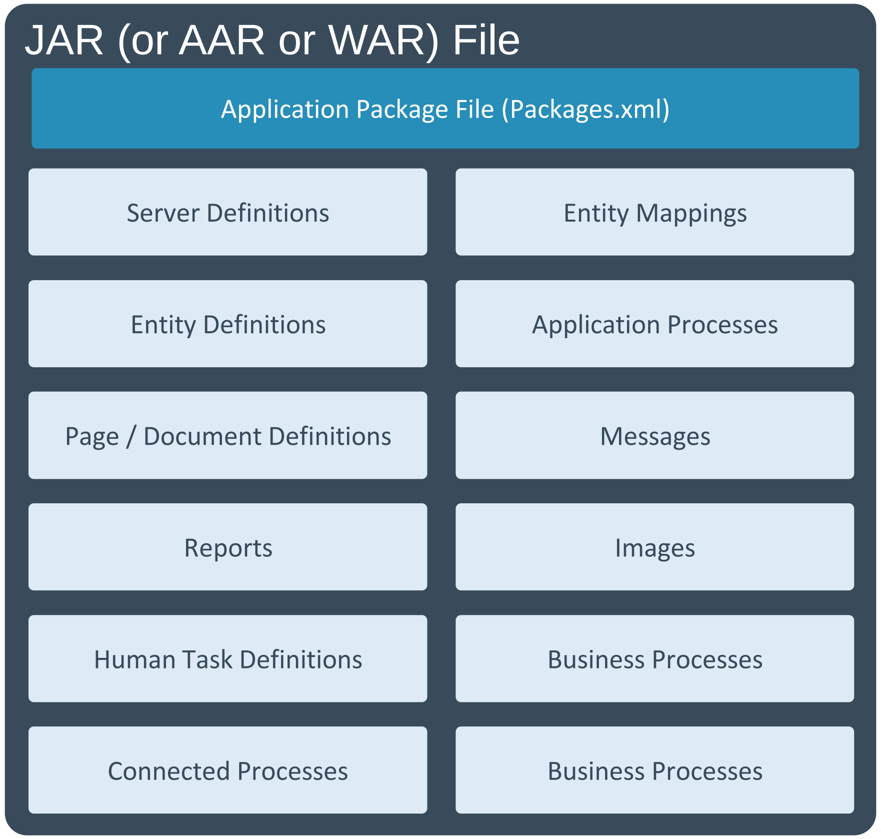


Artefact Management

Enactor is comprised of many different types of Artefact

Artefacts are registered in a Packages.xml present in each JAR file

Deployment Handlers are configured against these artefacts to process them at runtime

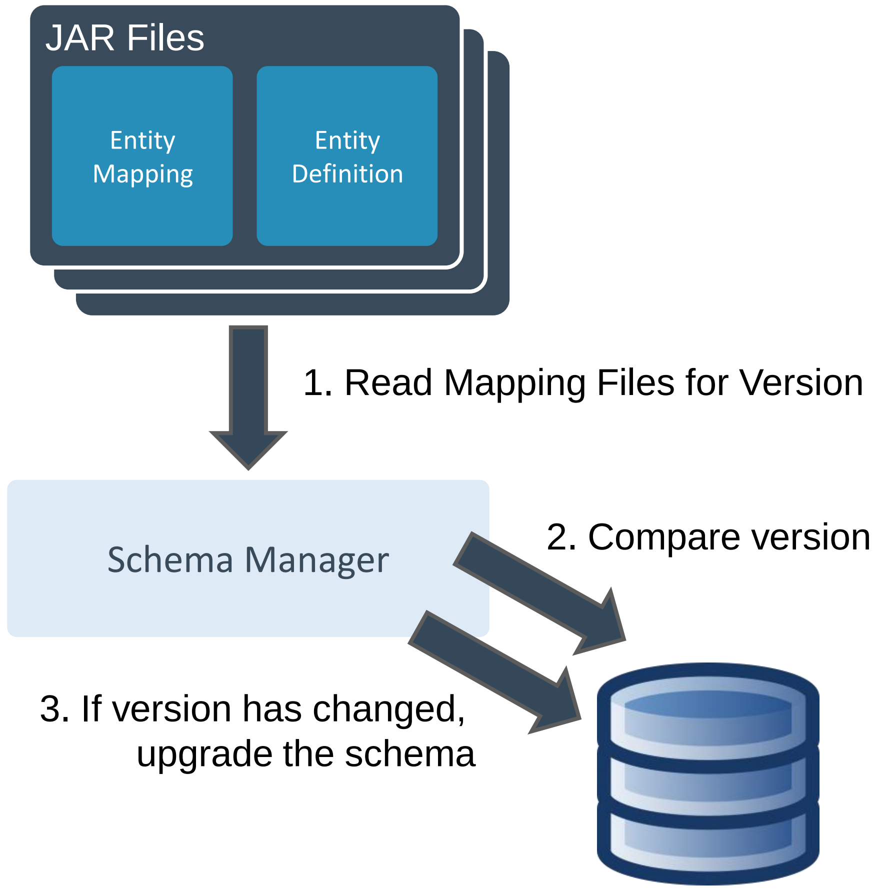


Schema Management

A Deployment Handler called the “Schema Manager” is responsible for managing the Database Schema

This uses information in the Entity Mapping and Entity Definition files to create and upgrade the database as necessary

This process can be fully automatic, or can be overridden to perform a manual upgrade if necessary


JAR Files
Entity Mapping
Entity Definition
1. Read Mapping Files for Version
Schema Manager
2. Compare version
3. If version has changed, upgrade the schema

enactor®

retail systems for a digital world

Q & A