Sumit Vadaviya

Camunda BPM Expert, Corporate Trainer

Email : sumit.vadaviya@trainosoft.com

I

n this session we will try get overview of Camunda.

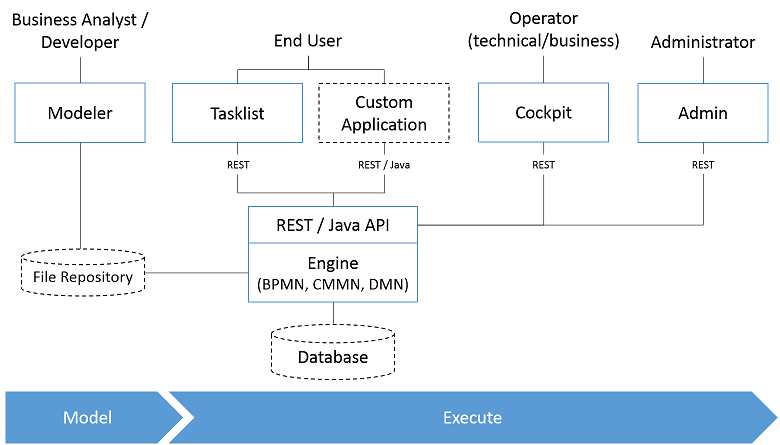
**1. What Camunda is ?**

* Camunda BPM is a lightweight, open source stage for Business Process Management. It is devoted to Java engineers and their run of the mill programming improvement foundation, while giving Business IT-Alignment amid process outline and runtime utilizing the BPMN 2.0 standard.
* Camunda’s core is a native BPMN 2.0 process engine that runs inside the Java Virtual Machine. It is a perfect match for the Spring Framework. On top of the process engine, one can also choose from a stack of tools for human workflow management, operations and monitoring.

Camunda Overview

* Camunda BPM is built around the process engine component. The following illustration shows the most important components of Camunda BPM along with some typical user roles.
* The Camunda has DMN engine as a Java library which can evaluate decision tables based on the DMN 1.1 OMG standard. It can be used as library embedded in an application or in combination with the Camunda BPM platform.
* The Camunda has CMMN engine as a Java library. Case Management Model and Notation (CMMN) is a standard notation and formal specification by the [Object Management Group](http://www.omg.org/spec/CMMN/) for representing case models.

**2. Camunda components and user roles**

****

* Camunda BPM is a Java-based framework. The main components are written in Java and we have a general focus on providing Java developers with the tools they need for designing, implementing and running business processes and workflows on the JVM.
* The process engine technology can also be available to Non-Java developers. This is why camunda BPM also provides a REST API which allows to build applications connecting to a remote process engine.
* Camunda BPM can be used both as a standalone process engine server and embedded inside custom Java applications.

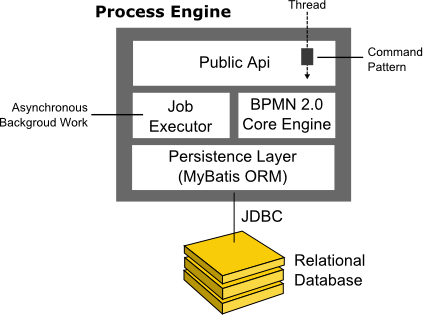
# **2.1 Modeler**

Modeling tool for BPMN 2.0 and CMMN 1.1 diagrams as well as DMN 1.1 decision tables.

# **2.2 Process Engine**

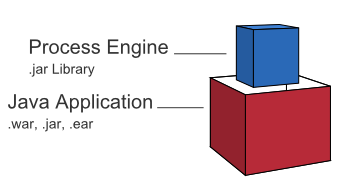
The process engine is a Java library responsible for executing BPMN 2.0 processes, CMMN 1.1 cases and DMN 1.1 decisions. It has a lightweight POJO core and uses a relational database for persistence. ORM mapping is provided by the MyBatis mapping framework.

**2.3 Process Engine Architecture**



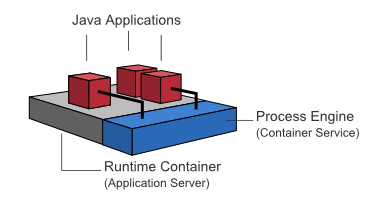
* **Process Engine Public API:** Service-oriented API allowing Java Applications to interact with the process engine. The different responsibilities of the process engine are separated out into individual services.
* **BPMN 2.0 Core Engine:** This is the core of the process engine. It features a lightweight execution engine for graph structures, a BPMN 2.0 parser which transforms BPMN 2.0 Xml files into Java Objects and a set of BPMN behaviour implementations.
* **Job Executor:** the Job Executor is responsible for processing asynchronous background work such as Timers or asynchronous continuations in a process.

**2.3.1** **Embedded Process Engine**



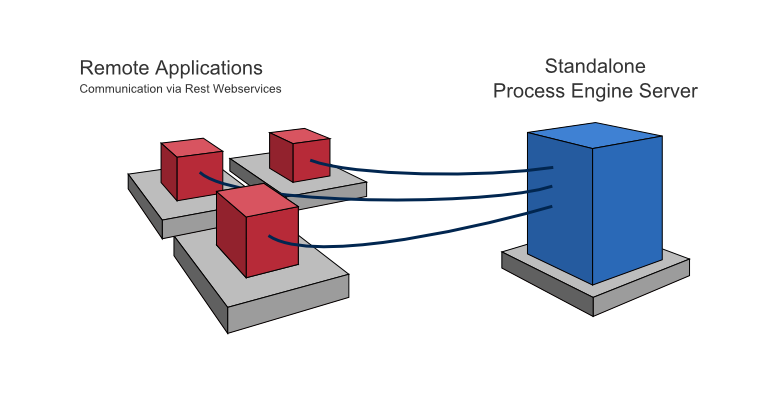
In this case the process engine is added as an application library to a custom application. This way the process engine can easily be started and stopped with the application lifecycle. It is possible to run multiple embedded process engines on top of a shared database.

**2.3.2** **Shared, container-managed Process Engine**



* In this case the process engine is started inside the runtime container .The process engine is provided as a container service and can be shared by all applications deployed inside the container.
* The concept can be compared to a JMS Message Queue which is provided by the runtime and can be used by all applications. There is a one-to-one mapping between process deployments and applications.

**2.3.3** **Standalone Process Engine Server**



* In this case the process engine is provided as a network service. Different applications running on the network can interact with the process engine through a remote communication channel.
* The easiest way for making the process engine accessible remote is to use the built-in REST api. Different communication channels such as SOAP Web Services or JMS are possible but need to be implemented by users. The process engine features a persistence layer responsible for persisting process instance state to a relational database. We use the MyBatis mapping engine for object relational mapping.

**2.4 Web Applications**

**REST API**

The REST API allows you to use the process engine from a remote application or a JavaScript application

**Camunda Tasklist**

A web application for human workflow management and user tasks that allows process participants to inspect their workflow tasks and navigate to task forms in order to work on the tasks, provide data input, save or complete the tasks .

**Camunda Cockpit**

A web application for process monitoring and operations that allows you to search for process instances, inspect their state and repair broken instances.

**Camunda Admin**

A web application that allows you to manage users, groups and authorizations.

**Camunda Optimize**

Camunda Optimize supports continuous process and decision table improvement by providing transparency into your automated workflows and decisions. Business-friendly reports and dashboards as well as alerts help you to identify process bottlenecks and improve your overall end-to-end process