Camunda

Automation of recurring workflows

JUNE 2021

Introduction

Camunda Platform is a light-weight, open-source platform for Business Process Management. This Platform ships with tools for creating workflow and decision models, operating deployed models in production, and allowing users to execute workflow tasks assigned to them.

It provides a Business Process Model and Notation (BPMN), and a Decision Model and Notation (DMN) standard compliant decision engine, which can be embedded in Java applications and with other languages via REST.

The major components of the Camunda Platform are:

- Admin - Web application for user management
- Cockpit - Web application for monitoring and operations
- Tasklist - Web application for human task management
- Welcome - Entry point web application with user profile
- Shared Configuration Options - Shared options for Web Application configuration

*~ **Camunda Modeler** is used for modelling BPMN and DMN.*

Features

Basic Use: BPM tool to automate recurring (daily-based) tasks.

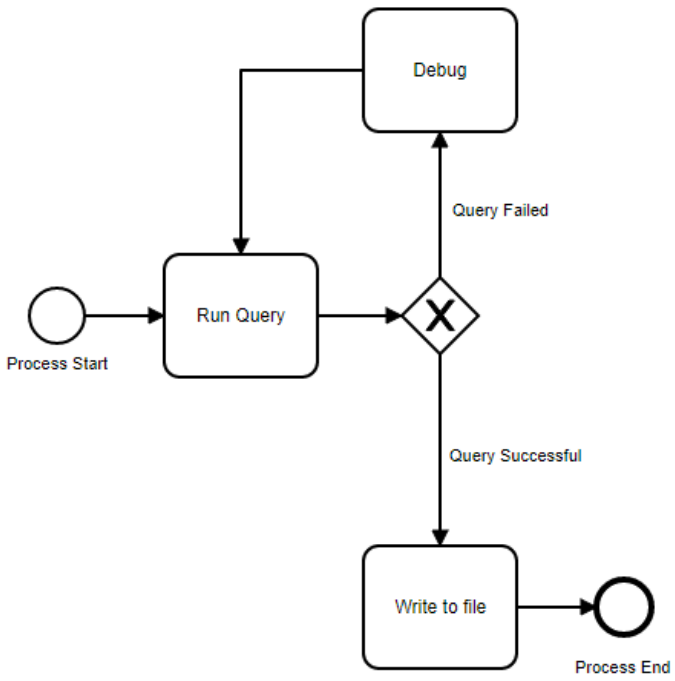
Make it's instance -> Use - > Discard

Workflows in Camunda are defined in BPMN which is basically an XML file. It can be graphically modeled using the *Camunda Modeler*.

Camunda can not only call services right away (Push-Principle) using some built-in connectors, but also put work items into a kind of built-in queue. Then a worker can fetch work items via REST, do the work and let Camunda know of the completion. So first you fetch tasks and lock them for you (as other workers might fetch at the same time to scale your system) and once done, tell Camunda the worker has completed its work.

Use Case

- For the use case, (filtering database tables based on predefined conditions and exporting/writing the resulting data), the following is the probable workflow in Camunda:
    - Create a task (BPM diagram workflow), which has the stages that first check the table for integrity, then a simple conditional checking stage that filters wanted data, and finally an exporting stage.
    - All these stages/sections can work independently without worrying about the other steps being done. Using Camunda, the work items can be placed into a kind of built-in queue. Then a worker can fetch work items via REST, do the work and let Camunda know of the completion.
    - Now for the tables that need to be filtered according to a particular condition, we can create an instance of the task, and then pass all the tables that need to be filtered.



In this simple BPM diagram that I made, we can automate the process of filtering a table based on a Query and exporting the resulting data.

Comparison with PySpark

- Pyspark SQL Framework works as a distributed SQL query engine, and provides very efficient query performance.
- Pyspark is a diverse framework that provides efficient components like Streaming, MlLib that can be integrated with Pyspark SQL for future uses.
- Camunda can serve well for automation of the workflow, but we can't comment on the throughput and efficiency without deploying and testing for varying sized tables of a database.
- To take advantage of Pyspark, we can use the Pyspark SQL engine in the Query Processing task of Camunda while designing the workflow. This would make the query processing faster, and also support automation.
