**Objective-**

**To Design an Orchestrator using Flowable which performs the following tasks every hour**

* **which reads files stored in a specified directory**
* **stores them in target directory**
* **convert them into Json format and store it in target directory**

**SchedulingClassTest.java in src/main/java**

package SchedulingTest;

import org.flowable.engine.ProcessEngine;

import org.flowable.engine.ProcessEngineConfiguration;

import org.flowable.engine.RepositoryService;

import org.flowable.engine.impl.cfg.StandaloneProcessEngineConfiguration;

import org.flowable.engine.repository.ProcessDefinition;

public class SchedulingClassTest {

public static void main(String[] args) throws InterruptedException {

ProcessEngineConfiguration cfg = new StandaloneProcessEngineConfiguration()

.setJdbcUrl("jdbc:h2:mem:flowable;DB\_CLOSE\_DELAY=-1")

.setAsyncExecutorActivate(true)

.setJdbcUsername("sa")

.setJdbcPassword("")

.setJdbcDriver("org.h2.Driver")

.setDatabaseSchemaUpdate(ProcessEngineConfiguration.DB\_SCHEMA\_UPDATE\_TRUE);

//Process Engine Configuration

ProcessEngine processEngine = cfg.buildProcessEngine(); //Building Process Engine

RepositoryService repositoryService = processEngine.getRepositoryService();

org.flowable.engine.repository.Deployment deployment = repositoryService.createDeployment()

.addClasspathResource("request.bpmn20.xml")

.deploy();

//Deploying Process Definition

ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery()

.deploymentId(deployment.getId())

.singleResult();

System.out.println("Found process definition : " + processDefinition.getName());

//Printing name of Process Definition

}

}

**Work.java in scr/main/java**

**package** SchedulingTest;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** org.apache.commons.io.FileUtils;

**import** org.flowable.engine.delegate.DelegateExecution;

**import** org.flowable.engine.delegate.JavaDelegate;

**import** org.json.simple.JSONArray;

**import** org.json.simple.JSONObject;

**public** **class** Work **implements** JavaDelegate {

**public** **void** execute(DelegateExecution execution) {

File file1 = **new** File("D:/PDFWork/MY\_DIRECTORY"); //Source Directory

String s[]=file1.list();

File source = **new** File("D:/PDFWork/MY\_DIRECTORY"); //Source Directory

File dest = **new** File("D:/PDFWork/MY\_DIRECTORY1"); //Destination Directory

**try** {

FileUtils.*copyDirectory*(source, dest);

//Copy files from Source Directory to Destination Directory

} **catch** (IOException e) {

e.printStackTrace();

}

JSONArray ja = **new** JSONArray();

//JSON Array

**for**(String s1 :s )

{

File f=**new** File("D:/PDFWork/MY\_DIRECTORY1/"+s1); //File of Destination Directory

JSONObject jo = **new** JSONObject(); //JSON Object

jo.put("FileName", s1);

//Adding file name to JSON Object

jo.put("Path", f.getAbsolutePath());

//Adding file absolute path to JSON Object

ja.add(jo);

}

JSONObject mainObj = **new** JSONObject();

// JSON Main Object

mainObj.put("Files", ja);

// Adding JSON Array to JSON Main Object

**try** {

String timeStamp = **new** SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(**new** Date()); //Time Stamp

FileWriter file2=**new** FileWriter("D://PDFWork//MY\_DIRECTORY1//"+"Batch\_"+timeStamp+".txt"); //New file for JSON

file2.write(mainObj.toJSONString()); //Writing JSON Object to New File

file2.flush();

//flush file

file2.close(); /

/close file

} **catch** (IOException e) {

e.printStackTrace();

}

**for**(String s1 :s)

{

File f=**new** File("D:/PDFWork/MY\_DIRECTORY/"+s1);

f.delete(); //Delete files from Source Directory

}

}

}

**request.bpmn20.xml in scr/main/resources**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<definitions xmlns=*"http://www.omg.org/spec/BPMN/20100524/MODEL"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"*

xmlns:bpmndi=*"http://www.omg.org/spec/BPMN/20100524/DI"*

xmlns:omgdc=*"http://www.omg.org/spec/DD/20100524/DC"*

xmlns:omgdi=*"http://www.omg.org/spec/DD/20100524/DI"*

xmlns:flowable=*"http://flowable.org/bpmn"*

typeLanguage=*"http://www.w3.org/2001/XMLSchema"*

expressionLanguage=*"http://www.w3.org/1999/XPath"*

targetNamespace=*"http://www.flowable.org/processdef"*>

<process id=*"workRequest"* name=*"Work Request"* isExecutable=*"true"*>

<startEvent id=*"startEvent"*>

<timerEventDefinition>

<timeCycle>0 0 \* ? \* \*</timeCycle>

</timerEventDefinition>

</startEvent>

<sequenceFlow sourceRef=*"startEvent"* targetRef=*"workEvent"*/>

<serviceTask id=*"workEvent"* name=*"Work to be done"*

flowable:class=*"SchedulingTest.Work"*>

</serviceTask>

<sequenceFlow sourceRef=*"workEvent"* targetRef=*"endEvent"*/>

<endEvent id=*"endEvent"*/>

</process>

</definitions>

**log4j.properties**

log4j.rootLogger=DEBUG, CA

log4j.appender.CA=org.apache.log4j.ConsoleAppender

log4j.appender.CA.layout=org.apache.log4j.PatternLayout

log4j.appender.CA.layout.ConversionPattern= %d{hh:mm:ss,SSS} [%t] %-5p %c %x - %m%n

**pom.xml**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>org.flowable</groupId>

<artifactId>SchedulingTest</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.flowable</groupId>

<artifactId>flowable-engine</artifactId>

<version>6.0.1</version>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.3.176</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.21</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.21</version>

</dependency>

</dependencies>

</project>

**Refrenced Libraries-**

commons-io-2.4.jar

json-simple-1.1.jar