**Orchestration**

**Orchestration is the automated arrangement, coordination, and management of computer systems, middleware, and services.**

Orchestration is often discussed as having an inherent intelligence or even implicitly autonomic control, but those are largely aspirations or analogies rather than technical descriptions. In reality, *orchestration* is largely the effect of automation or systems deploying elements of control theory.

This usage of *orchestration* is often discussed in the context of service-oriented architecture, virtualization, provisioning, converged infrastructure and dynamic datacenter topics. Orchestration in this sense is about aligning the business request with the applications, data, and infrastructure.It defines the policies and service levels through automated workflows, provisioning, and change management. This creates an application-aligned infrastructure that can be scaled up or down based on the needs of each application. Orchestration also provides centralized management of the resource pool, including billing, metering, and chargeback for consumption. For example, orchestration reduces the time and effort for deploying multiple instances of a single application. And as the requirement for more resources or a new application is triggered, automated tools now can perform tasks that previously could only be done by multiple administrators operating on their individual pieces of the physical stack.

A somewhat different usage relates to the process of coordinating an exchange of information through web service interactions.Applications that decouple the orchestration layer from the service layer are sometimes called agile applications.

A distinction is often made between orchestration (a local view from the perspective of one participant) and choreography (coordination from a global multi-participant perspective, albeit without a central controller).

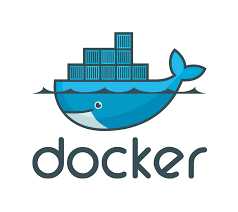
**Orchestartion Tools-**

* **Cloudify**

Cloudify is an open source cloud orchestration framework. Cloudify enables you to model applications and services and automate their entire life cycle, including deployment on any cloud or data center environment, monitoring all aspects of a deployed application, detecting issues and failure, manually or automatically remediating such issues, and performing ongoing maintenance tasks.

* **Heat**

Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. A native Heat template format is evolving, but Heat also endeavours to provide compatibility with the AWS CloudFormation template format, so that many existing CloudFormation templates can be launched on OpenStack. Heat provides both an OpenStack-native ReST API and a CloudFormation-compatible Query API.

* **Docker **

Docker is an open-source project that automates the deployment of applications inside software containers. It is promoted by the company Docker, Inc.

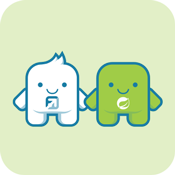
Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux.Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines.

* **StackStorm**

A platform for event-driven automation. StackStorm allows you to integrate and automate across services and tools. It ties together your existing infrastructure and application environment so you can more easily automate that environment -- with a particular focus on taking actions in response to events.StackStorm works on the premise that a human being will instead configure the system to watch for certain events and react autonomously on their behalf.

* **Flowable-Java BPM Engine**

1. a compact and highly efficient workflow and Business Process Management (BPM) platform for developers, system admins and business users
2. a lightning fast, tried and tested BPMN 2 process engine written in Java. It is Apache 2.0 licensed open source, with a committed community.
3. can run embedded in a Java application, or as a service on a server, a cluster, and in the cloud. It integrates perfectly with Spring. With a rich Java and REST API, it is the ideal engine for orchestrating human or system activities.

**Compact and Efficient Lightning Fast Runs Embedded or as**

**and Reliable a Service**

**Process Runtime**



The Flowable process engine has many years of real world use to ensure fast, efficient and reliable process execution. Drive the engine through its rich java and REST APIs, and let it drive your services through its rich integration.

**Process Design**

http://www.flowable.org/img/process_design.png

Flowable provides a web based process designer as well as an Eclipse plug-in. With either of these you can quickly create open standard BPMN process models. The web designer also supports the creation of form and decision table models, which can be combined with your process model to create complete process apps.

**Task Application**



A lightweight and extensible end user UI built with Angular provides an instant way of starting processes or working with tasks and forms. Use this application out of the box, or customize and extend components. Alternatively, use its rich set of APIs to build your own custom application UI.

## **History**

In October 2016, the lead developers of Activiti (software) left Alfresco (software) and started the Flowable Open Source project based on a fork of Activiti code.

The first version of Flowable was 5.22, based on a fork of Activiti 5.21, but added Transient Variables.The first release of Flowable version 6.0was based on a fork of Activiti version 6 beta 4. Version 6 of the Flowable engine includes a rewrite of the core process virtual machine.

## **Components**

The project comprises a set of modules that can operate together:

* BPMN Engine, the core Business Process Model and Notation workflow processor
* DMN Engine, an implementation of a subset of Decision Model and Notation based business rules
* Forms Engine, a forms service that can be used in conjunction with the Tasks web app or a custom application
* Modeler, a web-based graphical authoring interface for editing BPMN, DMN and forms models
* Designer, an Eclipse plug-in for designing BPMN models
* IDM, an example web tool to manage user and group identities and privileges
* Tasks, an example web application to start processes, view task queues, tasks and forms

**OBJECTIVE-To extract Pdf headings from all the Pdfs stored in a directory periodically after a specific interval of time using flowable.**

**Process Defination-**

****

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

package SchedulingTest;

import org.flowable.bpmn.model.Task;

import org.flowable.engine.ProcessEngine;

import org.flowable.engine.ProcessEngineConfiguration;

import org.flowable.engine.RepositoryService;

import org.flowable.engine.RuntimeService;

import org.flowable.engine.TaskService;

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

import org.flowable.engine.repository.ProcessDefinition;

import org.flowable.engine.runtime.ProcessInstance;

import org.flowable.engine.test.Deployment;

import java.util.\*;

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);

ProcessEngine processEngine = cfg.buildProcessEngine();

RepositoryService repositoryService = processEngine.getRepositoryService();

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

.addClasspathResource("request.bpmn20.xml")

.deploy();

ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery()

.deploymentId(deployment.getId())

.singleResult();

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

RuntimeService runtimeService = processEngine.getRuntimeService();

}

}

**Work.java in src/main/java**

package SchedulingTest;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.encryption.InvalidPasswordException;

import org.apache.pdfbox.text.PDFTextStripper;

import org.flowable.engine.delegate.DelegateExecution;

import org.flowable.engine.delegate.JavaDelegate;

public class Work implements JavaDelegate {

public void execute(DelegateExecution execution) {

File file = new File("D:/PDFWork/MY\_PDF\_DIRECTORY");

String s[]=file.list();

for(String s1 :s )

{

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

PDDocument document = null;

try {

document = PDDocument.load(f);

} catch (InvalidPasswordException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

PDFTextStripper pdfStripper = null;

try {

pdfStripper = new PDFTextStripper();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

pdfStripper.setParagraphStart("/t");

ArrayList<String> l= new ArrayList<String>();

int n=0;

try {

for (String line: pdfStripper.getText(document).split(pdfStripper.getParagraphStart()))

{

int c=0;

line=line.trim();

for(int i=0;i<line.length();i++)

{

if( ((i>0)&&(line.charAt(i)!=' ')&&(line.charAt(i-1)==' ')) || ((line.charAt(i)!=' ')&&(i==0)) )

{

c++;

}

}

if(c<=5 && c>0)

{

l.add(line);

n++;

System.out.println(line);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

BufferedWriter br = null;

try {

br = new BufferedWriter(new FileWriter("D://PDFWork//PdfBox\_Examples//"+s1+"\_Heading"+".txt"));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

for(int i=0;i<n;i++)

{

try {

br.write(l.get(i));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

br.newLine();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

try {

br.close();

document.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

f.delete();

}

}

}

**request.bpmn20.xml in src/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"*

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

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

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

<startEvent id=*"startEvent"*>

<timerEventDefinition>

<timeCycle>R4/2017-06-09T18:14/PT2M</timeCycle>

</timerEventDefinition>

</startEvent>

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

<serviceTask id=*"workEvent"* name=*"PDF Extract"*

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

</serviceTask>

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

<endEvent id=*"endEvent"*/>

</process>

</definitions>

**log4ju.properties in src/main/resources**

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>

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>pdfbox</artifactId>

<version>2.0.1</version>

</dependency>

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>fontbox</artifactId>

<version>2.0.0</version>

</dependency>

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>jempbox</artifactId>

<version>1.8.11</version>

</dependency>

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>xmpbox</artifactId>

<version>2.0.0</version>

</dependency>

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>preflight</artifactId>

<version>2.0.0</version>

</dependency>

<dependency>

<groupId>org.apache.pdfbox</groupId>

<artifactId>pdfbox-tools</artifactId>

<version>2.0.0</version>

</dependency>

</dependencies>

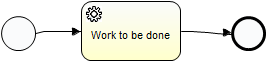
</project>

**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**
* **Update Mysql database**

**Process Definition-**



**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.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

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;

import com.mysql.jdbc.PreparedStatement;

import com.mysql.jdbc.Statement;

public class Work implements JavaDelegate {

public void execute(DelegateExecution execution) {

String url = "jdbc:mysql://localhost:3306/flowable";

String user = "root";

String password = "";

Statement stmt = null;

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);

try {

Connection conn = DriverManager.getConnection(url, user, password);

String query = " insert into Json (file\_name,file\_path)"

+ " values (?,?)";

// create the mysql insert preparedstatement

PreparedStatement preparedStmt = (PreparedStatement) conn.prepareStatement(query);

preparedStmt.setString (1, s1);

preparedStmt.setString (2, f.getAbsolutePath());

// execute the preparedstatement

preparedStmt.execute();

conn.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

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: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"*

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

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>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.4</version>

</dependency>

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId>json-simple</artifactId>

<version>1.1.1</version>

</dependency>

</dependencies>

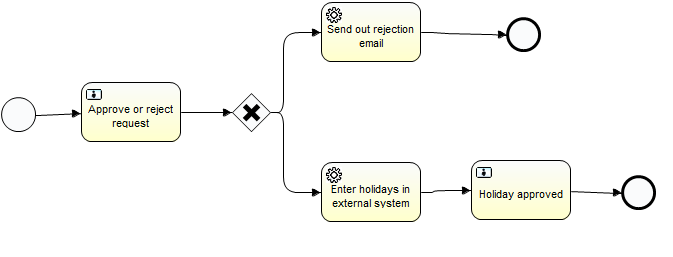
</project>

**Objective-**

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

* **which takes the employee name,no of holidays requested and reason for holidays**
* **lets the manager to either approve or reject the request for holidays**
* **convey the employee about his status of his request**

**Process Defination-**



**HolidayRequest.java** **file in src/main/java folder**

package holidayrequest;

import org.flowable.bpmn.model.Task;

import org.flowable.engine.ProcessEngine;

import org.flowable.engine.ProcessEngineConfiguration;

import org.flowable.engine.RepositoryService;

import org.flowable.engine.RuntimeService;

import org.flowable.engine.TaskService;

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

import org.flowable.engine.repository.ProcessDefinition;

import org.flowable.engine.runtime.ProcessInstance;

import org.flowable.engine.test.Deployment;

import java.util.\*;

public class HolidayRequest {

public static void main(String[] args) {

ProcessEngineConfiguration cfg = new StandaloneProcessEngineConfiguration()

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

.setJdbcUsername("sa")

.setJdbcPassword("")

.setJdbcDriver("org.h2.Driver")

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

ProcessEngine processEngine = cfg.buildProcessEngine();

RepositoryService repositoryService = processEngine.getRepositoryService();

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

.addClasspathResource("holiday-request.bpmn20.xml")

.deploy();

ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery()

.deploymentId(deployment.getId())

.singleResult();

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

Scanner scanner= new Scanner(System.in);

System.out.println("Who are you?");

String employee = scanner.nextLine();

System.out.println("How many holidays do you want to request?");

Integer nrOfHolidays = Integer.valueOf(scanner.nextLine());

System.out.println("Why do you need them?");

String description = scanner.nextLine();

RuntimeService runtimeService = processEngine.getRuntimeService();

Map<String, Object> variables = new HashMap<String, Object>();

variables.put("employee", employee);

variables.put("nrOfHolidays", nrOfHolidays);

variables.put("description", description);

ProcessInstance processInstance =

runtimeService.startProcessInstanceByKey("holidayRequest", variables);

TaskService taskService = processEngine.getTaskService();

List<org.flowable.engine.task.Task> tasks = taskService.createTaskQuery().taskCandidateGroup("managers").list();

System.out.println("You have " + tasks.size() + " tasks:");

for (int i=0; i<tasks.size(); i++) {

System.out.println((i+1) + ") " + tasks.get(i).getName());

}

System.out.println("Which task would you like to complete?");

int taskIndex = Integer.valueOf(scanner.nextLine());

org.flowable.engine.task.Task task = tasks.get(taskIndex - 1);

Map<String, Object> processVariables = taskService.getVariables(task.getId());

System.out.println(processVariables.get("employee") + " wants " +

processVariables.get("nrOfHolidays") + " of holidays. Do you approve this?");

boolean approved = scanner.nextLine().toLowerCase().equals("y");

variables = new HashMap<String, Object>();

variables.put("approved", approved);

taskService.complete(task.getId(), variables);

}

}

**CallExternalSystemDelegate.java** **file in src/main/java folder**

package holidayrequest;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.encryption.InvalidPasswordException;

import org.apache.pdfbox.text.PDFTextStripper;

import org.flowable.engine.delegate.DelegateExecution;

import org.flowable.engine.delegate.JavaDelegate;

public class CallExternalSystemDelegate implements JavaDelegate {

public void execute(DelegateExecution execution) {

System.out.println("Calling the external system for employee "

+ execution.getVariable("employee"));

}

}

**holiday-request.bpmn20.xml in src/main/resources folder**

<?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=*"holidayRequest"* name=*"Holiday Request"* isExecutable=*"true"*>

<startEvent id=*"startEvent"*/>

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

<userTask id=*"approveTask"* name=*"Approve or reject request"* flowable:candidateGroups=*"managers"*/>

<sequenceFlow sourceRef=*"approveTask"* targetRef=*"decision"*/>

<exclusiveGateway id=*"decision"*/>

<sequenceFlow sourceRef=*"decision"* targetRef=*"externalSystemCall"*>

<conditionExpression xsi:type=*"tFormalExpression"*>

<![CDATA[

${approved}

]]>

</conditionExpression>

</sequenceFlow>

<sequenceFlow sourceRef=*"decision"* targetRef=*"sendRejectionMail"*>

<conditionExpression xsi:type=*"tFormalExpression"*>

<![CDATA[

${!approved}

]]>

</conditionExpression>

</sequenceFlow>

<serviceTask id=*"externalSystemCall"* name=*"Enter holidays in external system"*

flowable:class=*"holidayrequest.CallExternalSystemDelegate"*/>

<sequenceFlow sourceRef=*"externalSystemCall"* targetRef=*"holidayApprovedTask"*/>

<userTask id=*"holidayApprovedTask"* name=*"Holiday approved"*/>

<sequenceFlow sourceRef=*"holidayApprovedTask"* targetRef=*"approveEnd"*/>

<serviceTask id=*"sendRejectionMail"* name=*"Send out rejection email"*

flowable:class=*"org.flowable.SendRejectionMail"*/>

<sequenceFlow sourceRef=*"sendRejectionMail"* targetRef=*"rejectEnd"*/>

<endEvent id=*"approveEnd"*/>

<endEvent id=*"rejectEnd"*/>

</process>

</definitions>

**log4j.properties file in src/main/resources folder**

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>holidayrequest</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>

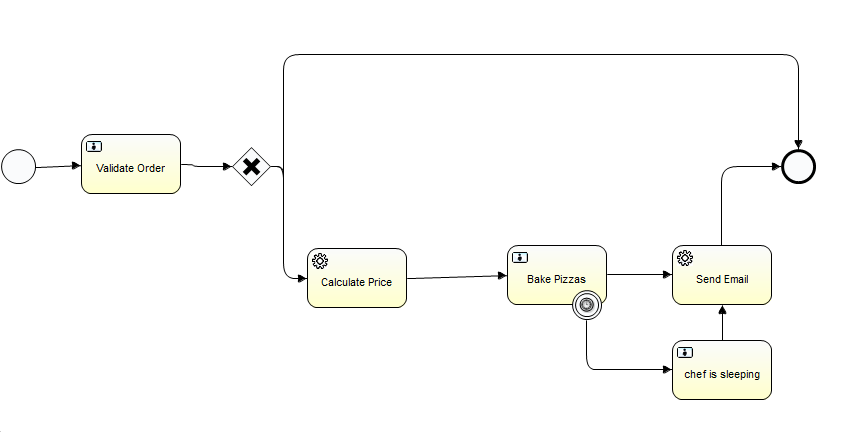
</project>

**Objective-**

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

* **takes the pizza order from customer**
* **asks the counter person to approve or reject the order**
* **print the bill**
* **asks the chef to bake the pizzas ordered by customer**
* **if the chef fails to bake pizzas within a specified time then ask the head chef to bake the pizzas**
* **send an email regarding the order made by customer**

**Process Defination-**



**PizzaOrder.java in src/main/java**

package PizzaOrder;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

import org.flowable.engine.ProcessEngine;

import org.flowable.engine.ProcessEngineConfiguration;

import org.flowable.engine.RepositoryService;

import org.flowable.engine.RuntimeService;

import org.flowable.engine.TaskService;

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

import org.flowable.engine.repository.ProcessDefinition;

import org.flowable.engine.runtime.ProcessInstance;

public class Pizza {

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

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);

ProcessEngine processEngine = cfg.buildProcessEngine();

RepositoryService repositoryService = processEngine.getRepositoryService();

org.flowable.engine.repository.Deployment deployment = repositoryService.createDeployment().addClasspathResource("request.bpmn20.xml").deploy();

ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery().deploymentId(deployment.getId()) .singleResult();

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

Scanner scanner= new Scanner(System.in);

System.out.println("Who are you?");

String customer = scanner.nextLine();

System.out.println("How many pizzas do you want?");

Integer noOfPizzas = Integer.valueOf(scanner.nextLine());

String duration="PT1M";

RuntimeService runtimeService = processEngine.getRuntimeService();

Map<String, Object> variables = new HashMap<String, Object>();

variables.put("customer", customer);

variables.put("noOfPizzas", noOfPizzas);

variables.put("duration", duration);

ProcessInstance processInstance =runtimeService.startProcessInstanceByKey("PizzaRequest", variables);

TaskService taskService = processEngine.getTaskService();

List<org.flowable.engine.task.Task> tasks = taskService.createTaskQuery().taskCandidateGroup("counterboy").list();

System.out.println("You have " + tasks.size() + " tasks:");

for (int i=0; i<tasks.size(); i++) {

System.out.println((i+1) + ") " + tasks.get(i).getName());

}

System.out.println("Which task would you like to complete?");

int taskIndex = Integer.valueOf(scanner.nextLine());

org.flowable.engine.task.Task task = tasks.get(taskIndex - 1);

Map<String, Object> processVariables = taskService.getVariables(task.getId());

System.out.println(processVariables.get("customer") + " wants to order " + processVariables.get("noOfPizzas") + " of Pizzas. Do you approve this?");

boolean approved = scanner.nextLine().toLowerCase().equals("y");

variables = new HashMap<String, Object>();

variables.put("approved", approved);

taskService.complete(task.getId(), variables);

TaskService taskService1 = processEngine.getTaskService();

List<org.flowable.engine.task.Task> tasks1 = taskService1.createTaskQuery().taskCandidateGroup("chef").list();

System.out.println("You have " + tasks1.size() + "chef tasks:");

for (int i=0; i<tasks1.size(); i++) {

System.out.println((i+1) + ") " + tasks1.get(i).getName());

}

System.out.println("Which task would you like to complete chef?");

int taskIndex1 = Integer.valueOf(scanner.nextLine());

org.flowable.engine.task.Task task1 = tasks1.get(taskIndex1 - 1);

Map<String, Object> processVariables1 = taskService1.getVariables(task1.getId());

System.out.println(processVariables1.get("customer") + " wants to order " + processVariables1.get("noOfPizzas") + " of Pizzas chef");

boolean made= false;

System.out.println("Made?");

long startTime = System.currentTimeMillis();

while(made==false && (System.currentTimeMillis()-startTime)<70000)

{

BufferedReader br=new BufferedReader(new FileReader("D:/chef.txt"));

made=br .readLine().toLowerCase().equals("y");

}

if(made==false){ System.out.println("Time out..");}

if(made==true && (System.currentTimeMillis()-startTime)<=70000) taskService1.complete(task1.getId());

Scanner scanner1= new Scanner(System.in);

TaskService taskService2 = processEngine.getTaskService();

List<org.flowable.engine.task.Task> tasks2 = taskService2.createTaskQuery().taskCandidateGroup("headchef").list();

System.out.println("You have " + tasks2.size() + " tasks: head chef");

for (int i=0; i<tasks2.size(); i++) {

System.out.println((i+1) + ") " + tasks2.get(i).getName());

}

System.out.println("Which task would you like to complete head chef?");

int taskIndex2 = Integer.valueOf(scanner1.nextLine());

org.flowable.engine.task.Task task2 = tasks2.get(taskIndex2 - 1);

Map<String, Object> processVariables2 = taskService2.getVariables(task2.getId());

System.out.println(processVariables2.get("customer") + " wants to order " + processVariables2.get("noOfPizzas") + " of Pizzas head chef");

taskService2.complete(task2.getId());

}

}

**Work1.java in src/main/java**

package PizzaOrder;

import org.flowable.engine.delegate.DelegateExecution;

import org.flowable.engine.delegate.JavaDelegate;

public class Work1 implements JavaDelegate {

public void execute(DelegateExecution execution) {

System.out.println("Printing Bill..."

+ execution.getVariable("noOfPizzas"));

}

}

**Work2.java in src/main/java**

package PizzaOrder;

import org.flowable.engine.delegate.DelegateExecution;

import org.flowable.engine.delegate.JavaDelegate;

public class Work2 implements JavaDelegate {

public void execute(DelegateExecution execution) {

System.out.println("Sending Email..."

);

}

}

**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:activiti=*"http://activiti.org/bpmn"* 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=*"PizzaRequest"* name=*"Pizza Request"* isExecutable=*"true"*>

<startEvent id=*"startEvent"*></startEvent>

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

<userTask id=*"validateOrder"* name=*"Validate Order"* flowable:candidateGroups=*"counterboy"*></userTask>

<sequenceFlow sourceRef=*"validateOrder"* targetRef=*"validateGateway"*></sequenceFlow>

<exclusiveGateway id=*"validateGateway"*></exclusiveGateway>

<sequenceFlow sourceRef=*"validateGateway"* targetRef=*"calculatePrice"*>

<conditionExpression xsi:type=*"tFormalExpression"*><![CDATA[${approved}]]></conditionExpression>

</sequenceFlow>

<sequenceFlow sourceRef=*"validateGateway"* targetRef=*"theEnd"*>

<conditionExpression xsi:type=*"tFormalExpression"*><![CDATA[${!approved}]]></conditionExpression>

</sequenceFlow>

<serviceTask id=*"calculatePrice"* name=*"Calculate Price"* flowable:class=*"PizzaOrder.Work1"*></serviceTask>

<sequenceFlow sourceRef=*"calculatePrice"* targetRef=*"bakePizzas"*></sequenceFlow>

<userTask id=*"bakePizzas"* name=*"Bake Pizzas"* flowable:candidateGroups=*"chef"*></userTask>

<sequenceFlow sourceRef=*"bakePizzas"* targetRef=*"sendEmail"*></sequenceFlow>

<boundaryEvent id=*"escalationTimer"* attachedToRef=*"bakePizzas"* cancelActivity=*"true"*>

<timerEventDefinition>

<timeDuration>PT1M</timeDuration>

</timerEventDefinition>

</boundaryEvent>

<sequenceFlow sourceRef=*"escalationTimer"* targetRef=*"bakePizzasByHeadChef"*></sequenceFlow>

<userTask id=*"bakePizzasByHeadChef"* name=*"chef is sleeping"* flowable:candidateGroups=*"headchef"*></userTask>

<sequenceFlow sourceRef=*"bakePizzasByHeadChef"* targetRef=*"sendEmail"*></sequenceFlow>

<serviceTask id=*"sendEmail"* name=*"Send Email"* flowable:class=*"PizzaOrder.Work2"*></serviceTask>

<sequenceFlow sourceRef=*"sendEmail"* targetRef=*"theEnd"*></sequenceFlow>

<endEvent id=*"theEnd"*></endEvent>

</process></definitions>

**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>PizzaOrder</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>

**REST**

REST is an architectural style which is based on web-standards and the HTTP protocol. REST was first described by Roy Fielding in 2000. In a REST based architecture everything is a resource. A resource is accessed via a common interface based on the HTTP standard methods. In a REST based architecture you typically have a REST server which provides access to the resources and a REST client which accesses and modifies the REST resources.

Every resource should support the HTTP common operations. Resources are identified by global IDs (which are typically URIs). REST allows that resources have different representations, e.g., text, XML, JSON etc. The REST client can ask for a specific representation via the HTTP protocol (content negotiation).

**HTTP Methods**

The *PUT*, *GET*, *POST* and *DELETE* methods are typical used in REST based architectures. The following table gives an explanation of these operations.

* GET defines a reading access of the resource without side-effects. The resource is never changed via a GET request, e.g., the request has no side effects (idempotent).
* PUT creates a new resource. It must also be idempotent.
* DELETE removes the resources. The operations are idempotent. They can get repeated without leading to different results.
* POST updates an existing resource or creates a new resource

**RESTFul Web Services**

A RESTFul web services are based on HTTP methods and the concept of REST. A RESTFul web service typically defines the base URI for the services, the supported MIME-types (XML, text, JSON, user-defined, …​) and the set of operations (POST, GET, PUT, DELETE) which are supported.

**Customer.java**

**package** org.vibhu.messenger;

**import** javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement

**public** **class** Customer {

**private** String name;

**private** **long** age;

**public** Customer()

{

}

**public** Customer(String x ,**int** y)

{

name=x;

age=y;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **long** getAge() {

**return** age;

}

**public** **void** setAge(**long** age) {

**this**.age = age;

}

}

**CustomerService.java**

package org.vibhu.messenger;

import java.util.ArrayList;

import java.util.List;

public class CustomerService {

static List<Customer> list=new ArrayList<>();

public static List<Customer> getAllCustomers()

{

return list;

}

public static void addCustomer(String x ,int y)

{

Customer c1=new Customer(x,y);

list.add(c1);

}

}

**CustomerResource.java**

package org.vibhu.messenger;

import java.util.List;

import javax.ws.rs.Consumes;

import javax.ws.rs.FormParam;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

@Path("/customer")

public class CustomerResource {

@POST

@Path("/add")

@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

public Response addCustomer(

@FormParam("name") String name,

@FormParam("age") int age) {

CustomerService.addCustomer(name, age);

return Response.status(200)

.entity("addUser is called, name : " + name + ", age : " + age)

.build();

}

@GET

@Produces({MediaType.TEXT\_HTML, MediaType.APPLICATION\_XML})

public List<Customer> getCustomer() {

return CustomerService.getAllCustomers();}}

**PDFBox API**

The Portable Document Format (PDF) is a file format that helps to present data in a manner that is independent of Application software, hardware, and operating systems.

Each PDF file holds description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.

There are several libraries available to create and manipulate PDF documents through programs, such as −

* **Adobe PDF Library** − This library provides API in languages such as C++, .NET and Java and using this we can edit, view print and extract text from PDF documents.
* **Formatting Objects Processor** − Open-source print formatter driven by XSL Formatting Objects and an output independent formatter. The primary output target is PDF.
* **iText** − This library provides API in languages such as Java, C#, and other .NET languages and using this library we can create and manipulate PDF, RTF and HTML documents.
* **JasperReports** − This is a Java reporting tool which generates reports in PDF document including Microsoft Excel, RTF, ODT, comma-separated values and XML files.

## **What is a PDFBox**

Apache PDFBox is an open-source Java library that supports the development and conversion of PDF documents. Using this library, you can develop Java programs that create, convert and manipulate PDF documents.

In addition to this, PDFBox also includes a command line utility for performing various operations over PDF using the available Jar file.

## **Features of PDFBox**

Following are the notable features of PDFBox −

* **Extract Text** − Using PDFBox, you can extract Unicode text from PDF files.
* **Split & Merge** − Using PDFBox, you can divide a single PDF file into multiple files, and merge them back as a single file.
* **Fill Forms** − Using PDFBox, you can fill the form data in a document.
* **Print** − Using PDFBox, you can print a PDF file using the standard Java printing API.
* **Save as Image** − Using PDFBox, you can save PDFs as image files, such as PNG or JPEG.
* **Create PDFs** − Using PDFBox, you can create a new PDF file by creating Java programs and, you can also include images and fonts.
* **Signing**− Using PDFBox, you can add digital signatures to the PDF files.

## **Applications of PDFBox**

The following are the applications of PDFBox −

* **Apache Nutch** − Apache Nutch is an open-source web-search software. It builds on Apache Lucene, adding web-specifics, such as a crawler, a link-graph database, parsers for HTML and other document formats, etc.
* **Apache Tika** − Apache Tika is a toolkit for detecting and extracting metadata and structured text content from various documents using existing parser libraries.

## **Components of PDFBox**

The following are the four main components of PDFBox −

* **PDFBox** − This is the main part of the PDFBox. This contains the classes and interfaces related to content extraction and manipulation.
* **FontBox** − This contains the classes and interfaces related to font, and using these classes we can modify the font of the text of the PDF document.
* **XmpBox** − This contains the classes and interfaces that handle XMP metadata.
* **Preflight** − This component is used to verify the PDF files against the PDF/A-1b standard.

**Creating a PDF Document**

**package** myproject;

**import** java.io.IOException;

**import** org.apache.pdfbox.pdmodel.PDDocument;

**public** **class** DocumentCreation {

**public** **static** **void** main (String args[]) **throws** IOException

{

PDDocument document = **new** PDDocument();

document.save("D:/PDFWork/PdfBox\_Examples/my\_doc.pdf");

System.***out***.println("PDF created");

document.close();

}

}

**Adding Pages**

package myproject;

import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

public class AddingPages {

public static void main(String args[]) throws IOException

{

PDDocument document = new PDDocument();

for (int i=0; i<10; i++) {

PDPage blankPage = new PDPage();

document.addPage( blankPage );

}

document.save("D:/PDFWork/PdfBox\_Examples/my\_doc.pdf");

System.out.println("PDF created");

document.close();

}

}

**Loading a Document**

package myproject;

import java.io.File;

import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

public class LoadingExistingDocument {

public static void main(String args[]) throws IOException

{

File file = new File("D:/PDFWork/PdfBox\_Examples/my\_doc.pdf");

PDDocument document = PDDocument.load(file);

System.out.println("PDF loaded");

document.addPage(new PDPage());

document.save("D:/PDFWork/PdfBox\_Examples/sample.pdf");

document.close();

}

}

**Removing Pages**

package myproject;

import java.io.File;

import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;

public class RemovingPages {

public static void main(String args[]) throws IOException

{

File file = new File("D:/PDFWork/PdfBox\_Examples/sample.pdf");

PDDocument document = PDDocument.load(file);

int n= document.getNumberOfPages();

System.out.print(n);

document.removePage(2);

System.out.println("page removed");

document.save("D:/PDFWork/PdfBox\_Examples/sample.pdf");

document.close();

}

}

**Document Properties**

package myproject;

import java.io.IOException;

import java.util.Calendar;

import java.util.GregorianCalendar;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDDocumentInformation;

import org.apache.pdfbox.pdmodel.PDPage;

public class AddingAttributes {

public static void main(String args[]) throws IOException

{

PDDocument document = new PDDocument();

PDPage blankPage = new PDPage();

document.addPage( blankPage );

PDDocumentInformation pdd = document.getDocumentInformation();

pdd.setAuthor("Tutorialspoint");

pdd.setTitle("Sample document");

pdd.setCreator("PDF Examples");

pdd.setSubject("Example document");

Calendar date = new GregorianCalendar();

date.set(2015, 11, 5);

pdd.setCreationDate(date);

date.set(2016, 6, 5);

pdd.setModificationDate(date);

pdd.setKeywords("sample, first example, my pdf");

document.save("D:/PDFWork/PdfBox\_Examples/doc\_attributes.pdf");

System.out.println("Properties added successfully ");

document.close();

}}

**Adding Text**

package myproject;

import java.io.File;

import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

import org.apache.pdfbox.pdmodel.PDPageContentStream;

import org.apache.pdfbox.pdmodel.font.PDType1Font;

public class AddingText {

public static void main (String args[]) throws IOException

{

File file = new File("D:/PDFWork/PdfBox\_Examples/my\_doc.pdf");

PDDocument document = PDDocument.load(file);

PDPage page = document.getPage(1);

PDPageContentStream contentStream = new PDPageContentStream(document, page);

contentStream.beginText();

contentStream.setFont(PDType1Font.TIMES\_ROMAN, 12);

contentStream.newLineAtOffset(25, 500);

String text = "This is the sample document and we are adding content to it.";

contentStream.showText(text);

contentStream.endText();

System.out.println("Content added");

contentStream.close();

document.save(new File("D:/PDFWork/PdfBox\_Examples/new.pdf"));

document.close();

}

}

**Adding Multiple lines**

package myproject;

import java.io.File;

import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

import org.apache.pdfbox.pdmodel.PDPageContentStream;

import org.apache.pdfbox.pdmodel.font.PDType1Font;

public class AddingMultipleLines {

public static void main(String args[]) throws IOException

{

File file = new File("D:PDFWork//PdfBox\_Examples/new.pdf");

PDDocument doc = PDDocument.load(file);

PDPage page = doc.getPage(0);

PDPageContentStream contentStream = new PDPageContentStream(doc, page);

contentStream.beginText();

contentStream.setLeading(14.5f);

contentStream.newLineAtOffset(25, 725);

contentStream.setFont( PDType1Font.TIMES\_BOLD, 16 );

String text="HOLY" ;

contentStream. showText(text);

contentStream.newLine();

contentStream.setFont( PDType1Font.TIMES\_ROMAN, 16 );

String text1="This is a paragraph.A paragraph has multiples lines" ;

contentStream. showText(text1);

contentStream.newLine();

contentStream.endText();

System.out.println("Content added");

contentStream.close();

doc.save(new File("D:PDFWork//PdfBox\_Examples/new1.pdf"));

doc.close();

}

}

**Reading Text**

package myproject;

import java.io.File;

import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.text.PDFTextStripper;

public class ReadingText {

public static void main(String args[]) throws IOException

{

File file = new File("D:/PDFWork/PdfBox\_Examples/new1.pdf");

PDDocument document = PDDocument.load(file);

PDFTextStripper pdfStripper = new PDFTextStripper();

String text = pdfStripper.getText(document);

System.out.println(text);

document.close();

}

}

**Inserting Image**

package myproject;import java.io.File;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.pdmodel.PDPage;

import org.apache.pdfbox.pdmodel.PDPageContentStream;

import org.apache.pdfbox.pdmodel.graphics.image.PDImageXObject;

public class InsertingImage {

public static void main(String args[]) throws Exception

{

File file = new File("D:/PDFWork/PdfBox\_Examples/sample.pdf");

PDDocument doc = PDDocument.load(file);

PDPage page = doc.getPage(0);

PDImageXObject pdImage = PDImageXObject.createFromFile("D:/PDFWork/PdfBox\_Examples/images.png",doc);

PDPageContentStream contents = new PDPageContentStream(doc, page);

contents.drawImage(pdImage, 70, 250);

System.out.println("Image inserted");

contents.close();

doc.save("D:/PdfBox\_Examples/sample.pdf");

doc.close();

}

}

**Extarcting Image**

package myproject;

import java.awt.image.BufferedImage;

import java.io.File;

import javax.imageio.ImageIO;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.rendering.PDFRenderer;

public class PDFToImage {

public static void main(String args[]) throws Exception

{

File file = new File("D:/PDFWork/PdfBox\_Examples/sample.pdf");

PDDocument document = PDDocument.load(file);

PDFRenderer renderer = new PDFRenderer(document);

BufferedImage image = renderer.renderImage(0);

ImageIO.write(image, "JPEG", new File("D:/PDFWork/PdfBox\_Examples/myimage.jpg"));

System.out.println("Image created");

document.close();

}

}