Sumit Vadaviya

Camunda BPM Expert, Corporate Trainer

Email : sumit.vadaviya@trainosoft.com

I

n this session we will come to know what is process application and implement process application maven project.

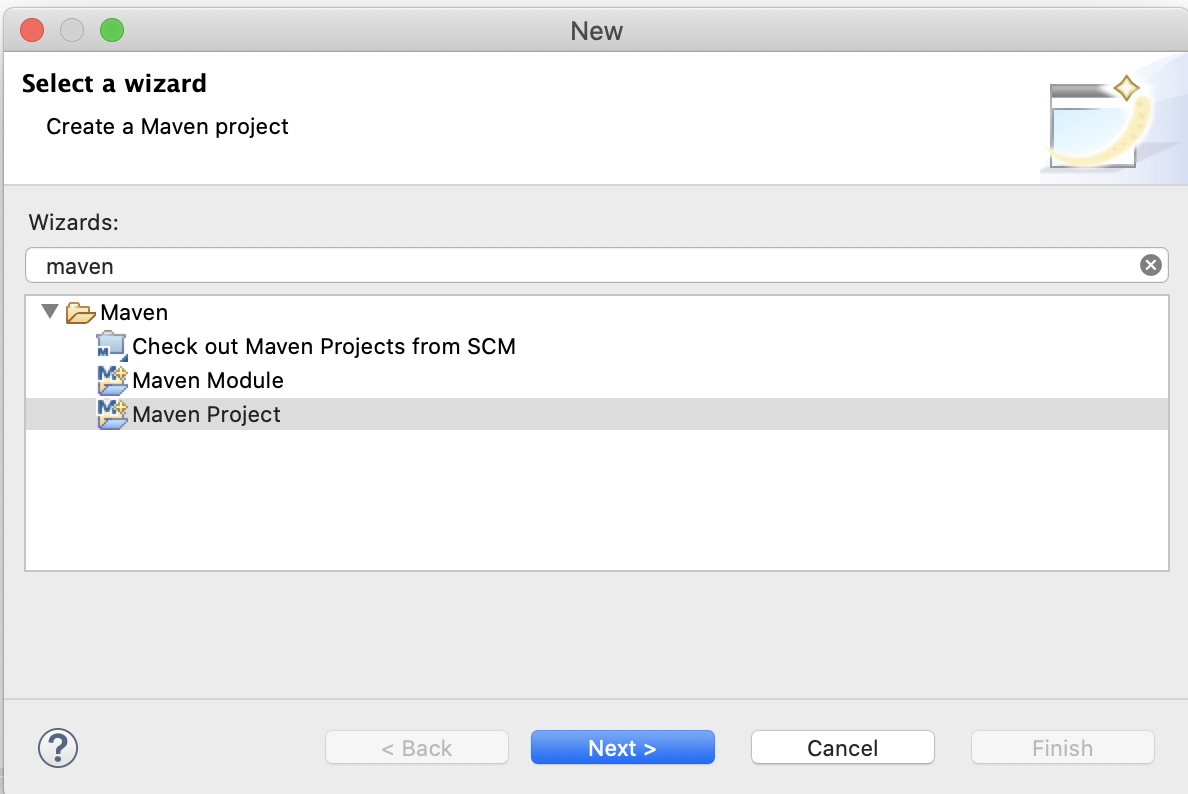
Process application is an ordinary Java application that uses a shared Camunda BPM engine in a Servlet Container, e.g. Apache Tomcat. Contains: Servlet Process Application, BPMN Process, Java Delegate, HTML5-based start and task forms, JUnit Test with in-memory engine, Maven Plugins or Ant build script for one-click deployment in Eclipse

Camunda Process Application

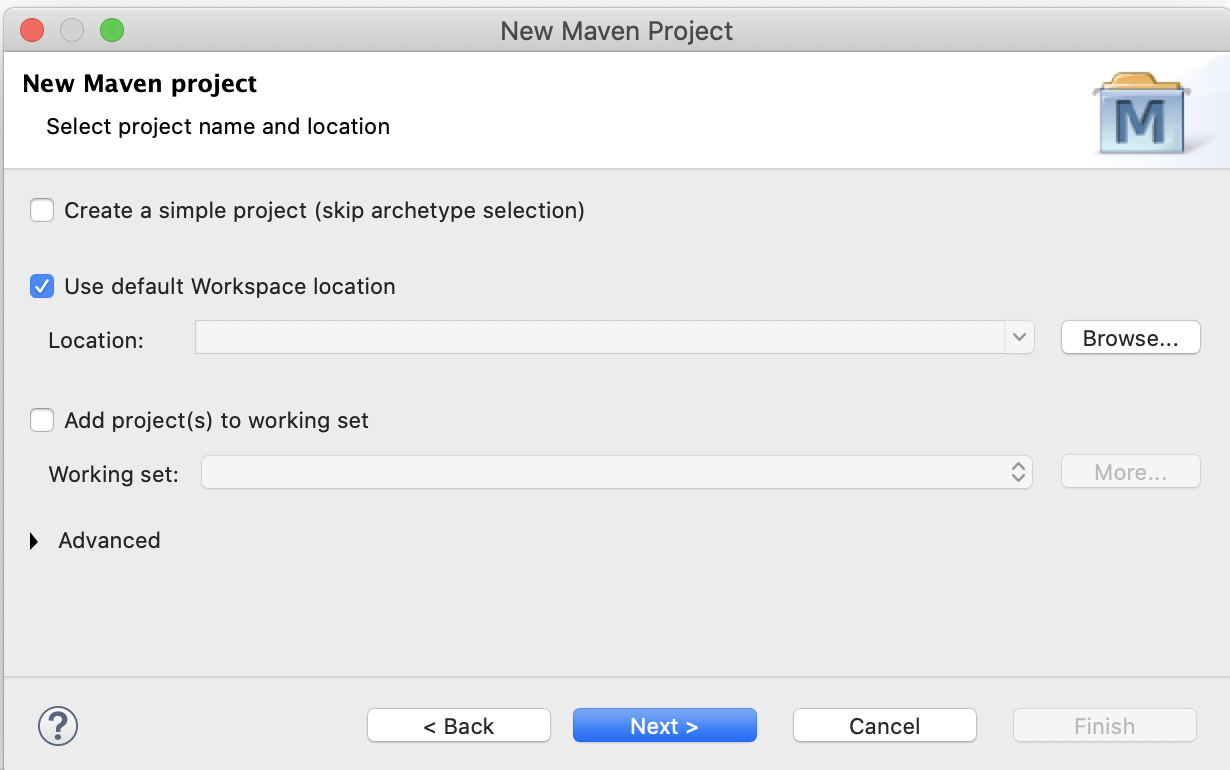
)

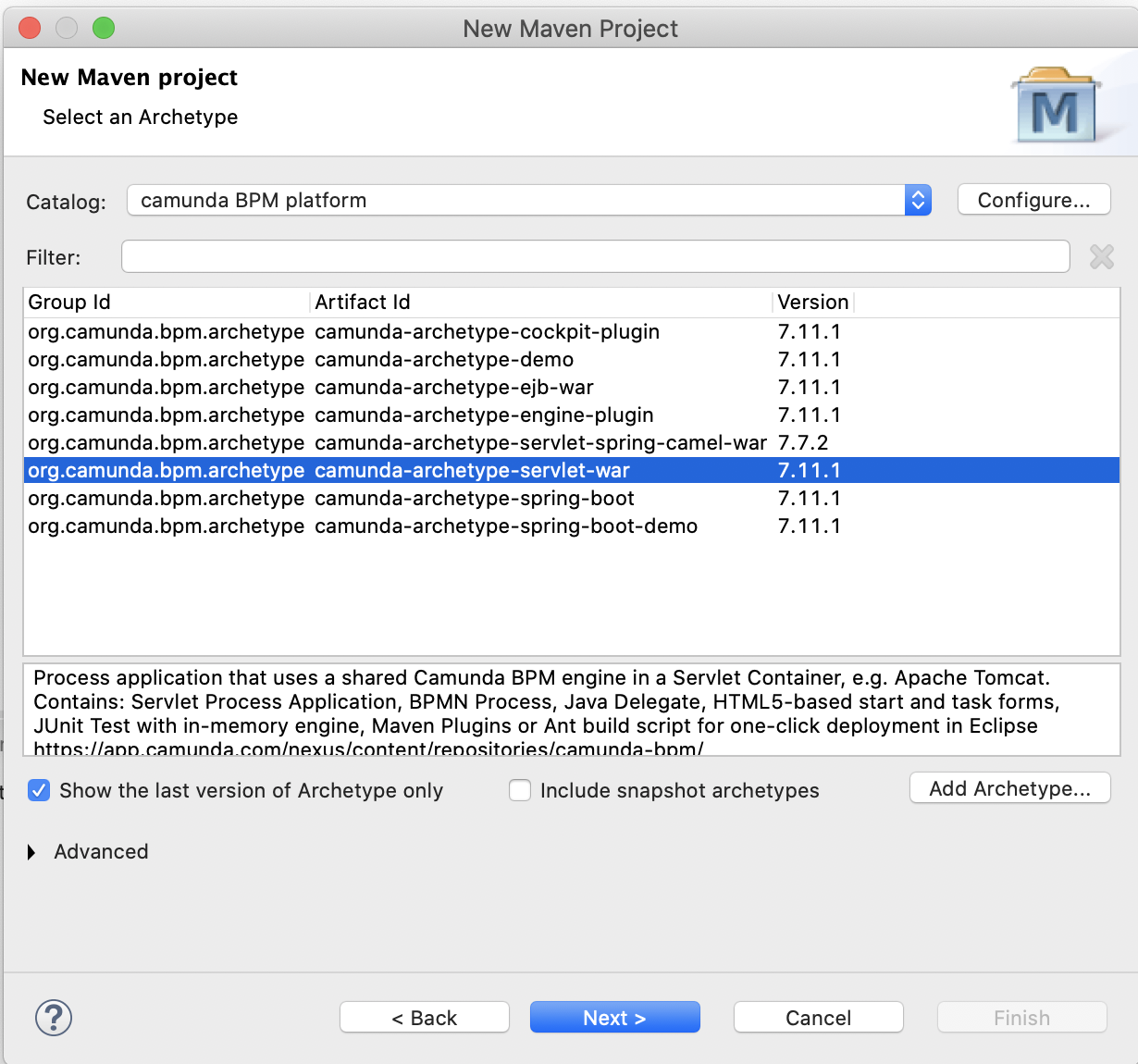
Most such applications will start their own process engine (or use a process engine provided by the runtime container), deploy some BPMN 2.0 process definitions and interact with process instances derived from these process definitions.

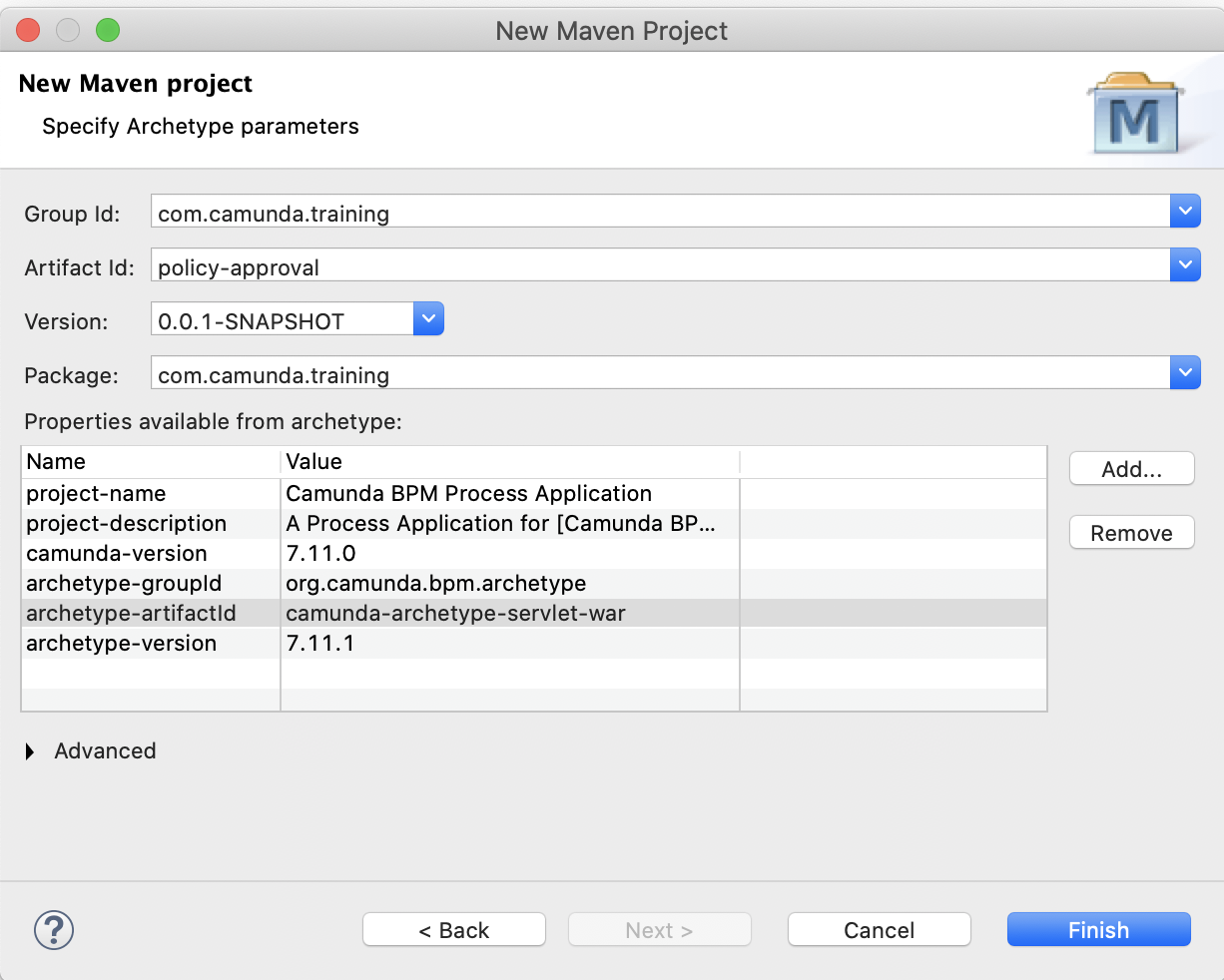
**Create Maven Project.**

****

Hit the Next > button.



Hit the Next > button.

Select camunda-archetype -servlet-war and hit the Next > button.

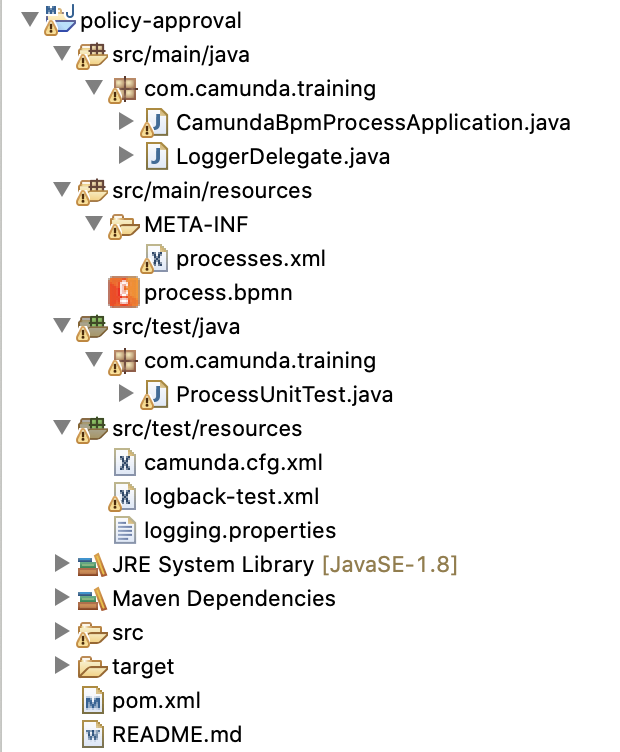
**Group Id:** com.camunda.training

**Artifact-Id:** policy-approval

**Version:** 0.0.1-SNAPSHOT

**Package:** com.camunda.training

Hit the **Finish** button.

Eclipse camunda maven project created as below.

As we are using camunda version 7.13.0. we need to update maven camunda dependency to 7.13.0.

Open pom.xml and update

<camunda.version>7.11.0</camunda.version> to <camunda.version>7.13.0</camunda.version>

Add dependency for

<!-- https://mvnrepository.com/artifact/joda-time/joda-time -->

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

</dependency>

Updated pom.xml looks like below.

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

<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>com.camunda.training</groupId>

<artifactId>policy-approval</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>Camunda BPM Process Application</name>

<description>A Process Application for [Camunda BPM](http://docs.camunda.org). [The project has been generated by the Maven archetype 'camunda-archetype-servlet-war-7.11.1']</description>

<properties>

<camunda.version>7.13.0</camunda.version>

<!--

Adjust if you want to use Camunda Enterprise Edition (EE):

<camunda.version>7.11.0-ee</camunda.version>

Make sure you also switch to EE repository below

-->

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<failOnMissingWebXml>false</failOnMissingWebXml>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.camunda.bpm</groupId>

<artifactId>camunda-bom</artifactId>

<version>${camunda.version}</version>

<scope>import</scope>

<type>pom</type>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

<!-- process engine, needs to be 'provided' because it's already a shared library in the container -->

<groupId>org.camunda.bpm</groupId>

<artifactId>camunda-engine</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<!-- AssertJ Testing Library -->

<groupId>org.camunda.bpm.extension</groupId>

<artifactId>camunda-bpm-assert</artifactId>

<version>1.2</version>

<scope>test</scope>

</dependency>

<!-- Required to use Spin dataformat support in unit tests -->

<dependency>

<groupId>org.camunda.spin</groupId>

<artifactId>camunda-spin-dataformat-all</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.camunda.bpm</groupId>

<artifactId>camunda-engine-plugin-spin</artifactId>

<scope>test</scope>

</dependency>

<!-- Required to use Templates in unit tests -->

<dependency>

<groupId>org.camunda.template-engines</groupId>

<artifactId>camunda-template-engines-freemarker</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.camunda.template-engines</groupId>

<artifactId>camunda-template-engines-velocity</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<dependency>

<!-- Needed for InMemoryH2Test -->

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.4.197</version>

<scope>test</scope>

</dependency>

<dependency>

<!-- Used to generate test coverage reports, see https://github.com/camunda/camunda-consulting/tree/master/snippets/camunda-bpm-process-test-coverage -->

<groupId>org.camunda.bpm.extension</groupId>

<artifactId>camunda-bpm-process-test-coverage</artifactId>

<version>0.3.2</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/joda-time/joda-time -->

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

</dependency>

<dependency>

<!-- use logback as logger -->

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.1.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<!-- apache commons logging => slf4j -->

<artifactId>jcl-over-slf4j</artifactId>

<version>1.7.25</version>

<scope>test</scope>

</dependency>

<dependency>

<!-- java util logging => slf4j -->

<groupId>org.slf4j</groupId>

<artifactId>jul-to-slf4j</artifactId>

<version>1.7.25</version>

<scope>test</scope>

</dependency>

<!-- Add your own dependencies here, if in compile scope, they are added to the war -->

</dependencies>

<repositories>

<repository>

<id>camunda-bpm-nexus</id>

<name>Camunda Maven Repository</name>

<url>https://app.camunda.com/nexus/content/groups/public</url>

</repository>

<!-- enable this for EE dependencies (requires credentials in ~/.m2/settings.xml)

<repository>

<id>camunda-bpm-nexus-ee</id>

<name>Camunda Enterprise Maven Repository</name>

<url>https://app.camunda.com/nexus/content/repositories/camunda-bpm-ee</url>

</repository>

-->

</repositories>

<build>

<finalName>${project.artifactId}</finalName>

<plugins>

<plugin>

<!-- Deploy to Tomcat using: mvn clean package antrun:run

Follow the instructions in build.properties.example to make it work!-->

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-antrun-plugin</artifactId>

<configuration>

<tasks>

<ant antfile=*"${basedir}/build.xml"*>

<target name=*"copy.war.into.tomcat"* />

</ant>

</tasks>

</configuration>

</plugin>

<!-- Tomcat Maven Plugin

Deploy to Tomcat using:

mvn clean tomcat7:deploy

Redeploy:

mvn clean tomcat7:redeploy

Undeploy:

mvn tomcat7:undeploy

To use this plugin, add these lines to your tomcat-users.xml: (inside the <tomcat-users>-tag)

<role rolename="manager-script"/>

<user username="admin" password="admin" roles="manager-script"/>

-->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<url>http://localhost:8080/manager/text</url>

<username>admin</username>

<password>admin</password>

</configuration>

</plugin>

<plugin>

<!-- Deploy to JBoss AS7: mvn clean jboss-as:deploy See also: https://docs.jboss.org/jbossas/7/plugins/maven/latest/examples/deployment-example.html -->

<groupId>org.jboss.as.plugins</groupId>

<artifactId>jboss-as-maven-plugin</artifactId>

<version>7.9.Final</version>

</plugin>

<plugin>

<!-- Deploy to Wildfly: mvn clean wildfly:deploy See also: https://docs.jboss.org/wildfly/plugins/maven/latest/examples/deployment-example.html -->

<groupId>org.wildfly.plugins</groupId>

<artifactId>wildfly-maven-plugin</artifactId>

<version>1.2.1.Final</version>

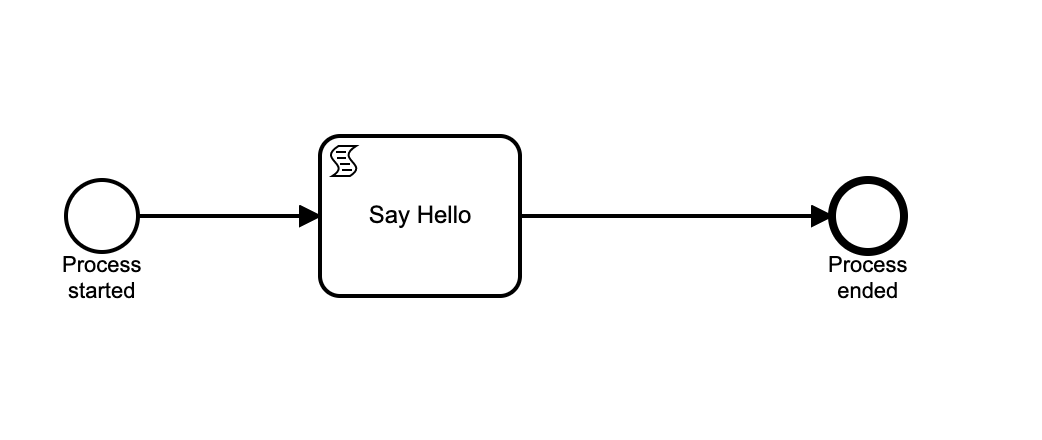
</plugin>

</plugins>

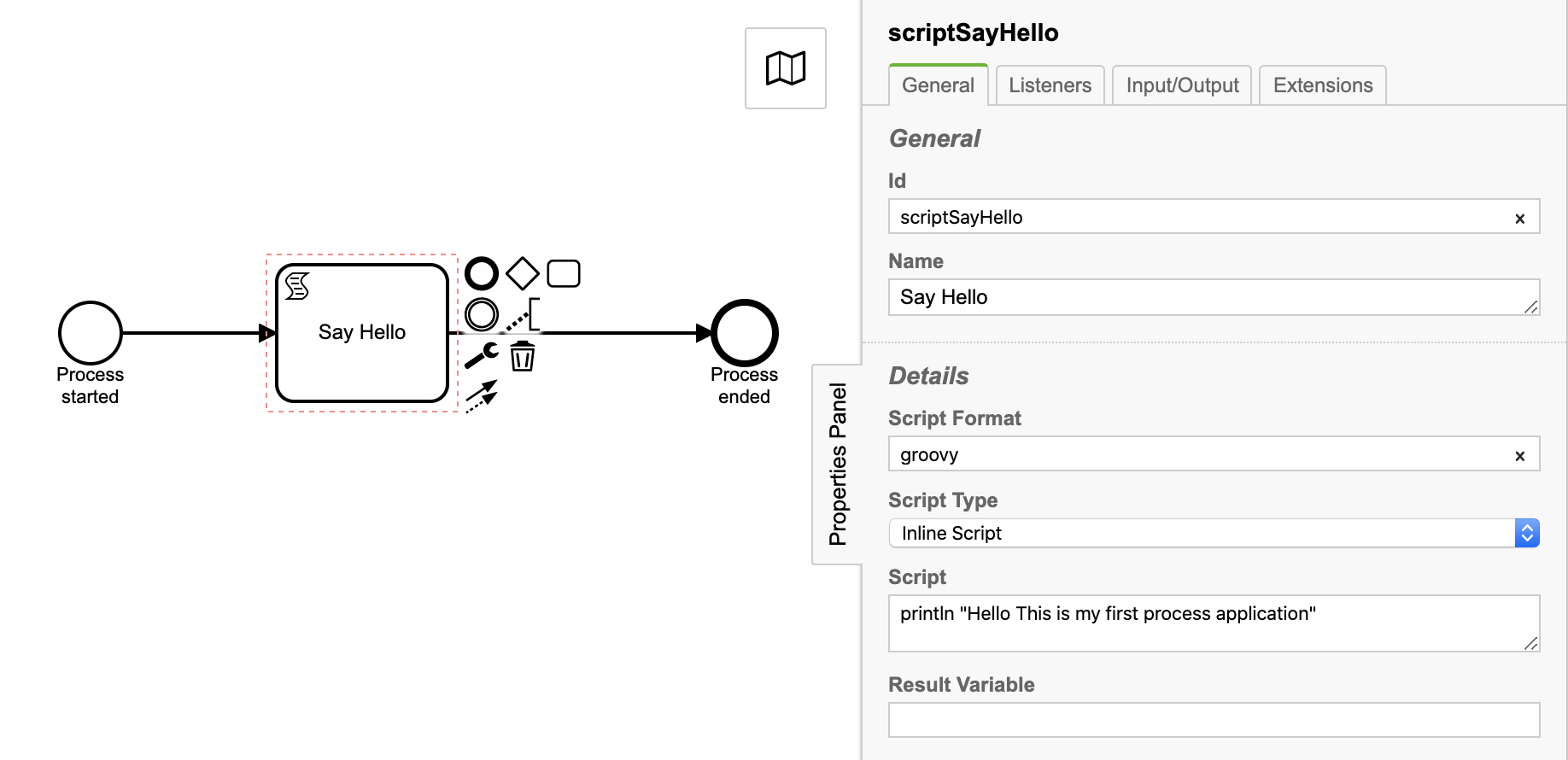
</build>

</project>

**Now let’s update process.bpmn as below.**

****

**Script task : Say Hello**

****

**Properties:**

**Id:** scriptSayHello

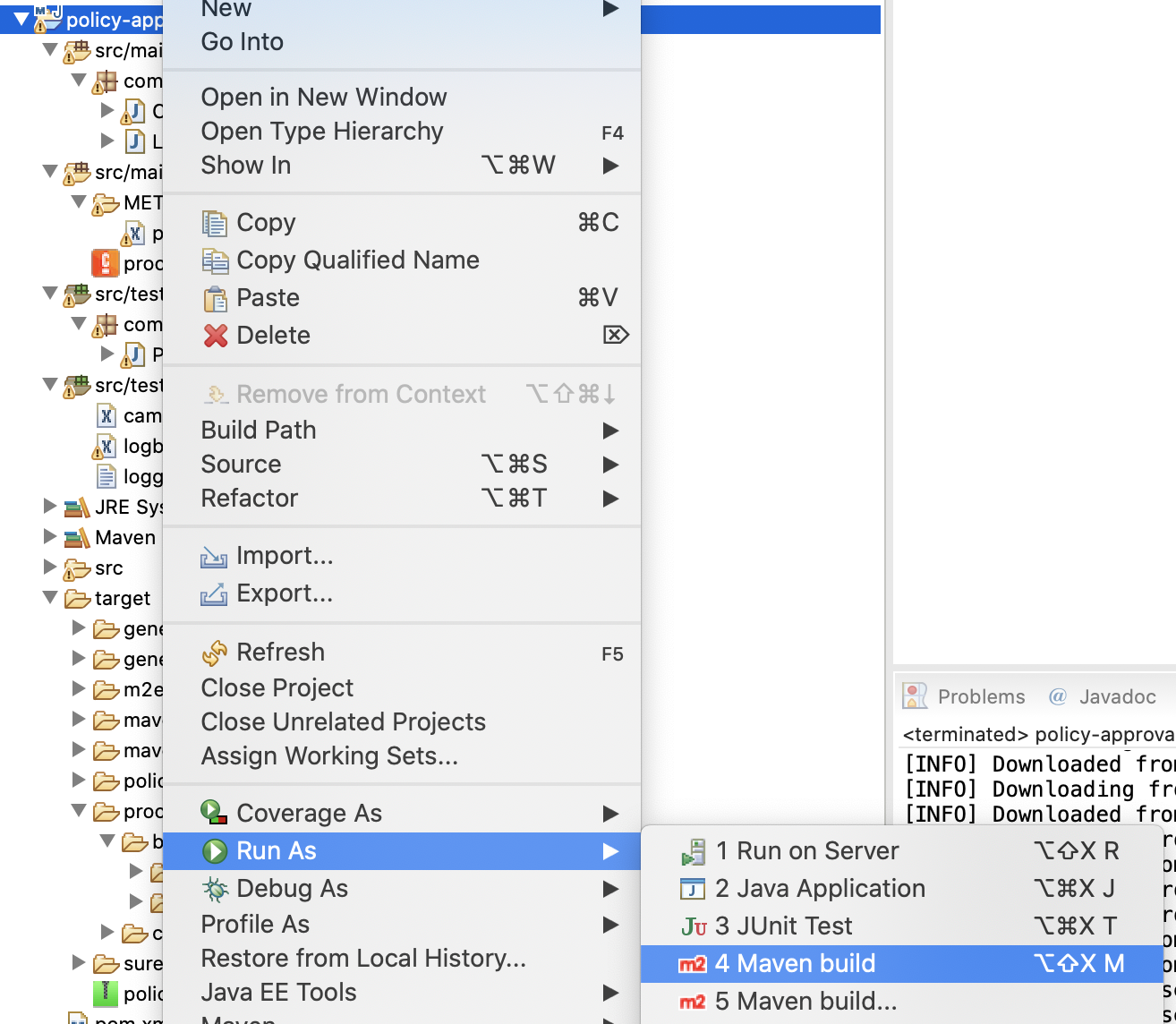
**Name:** Say Hello

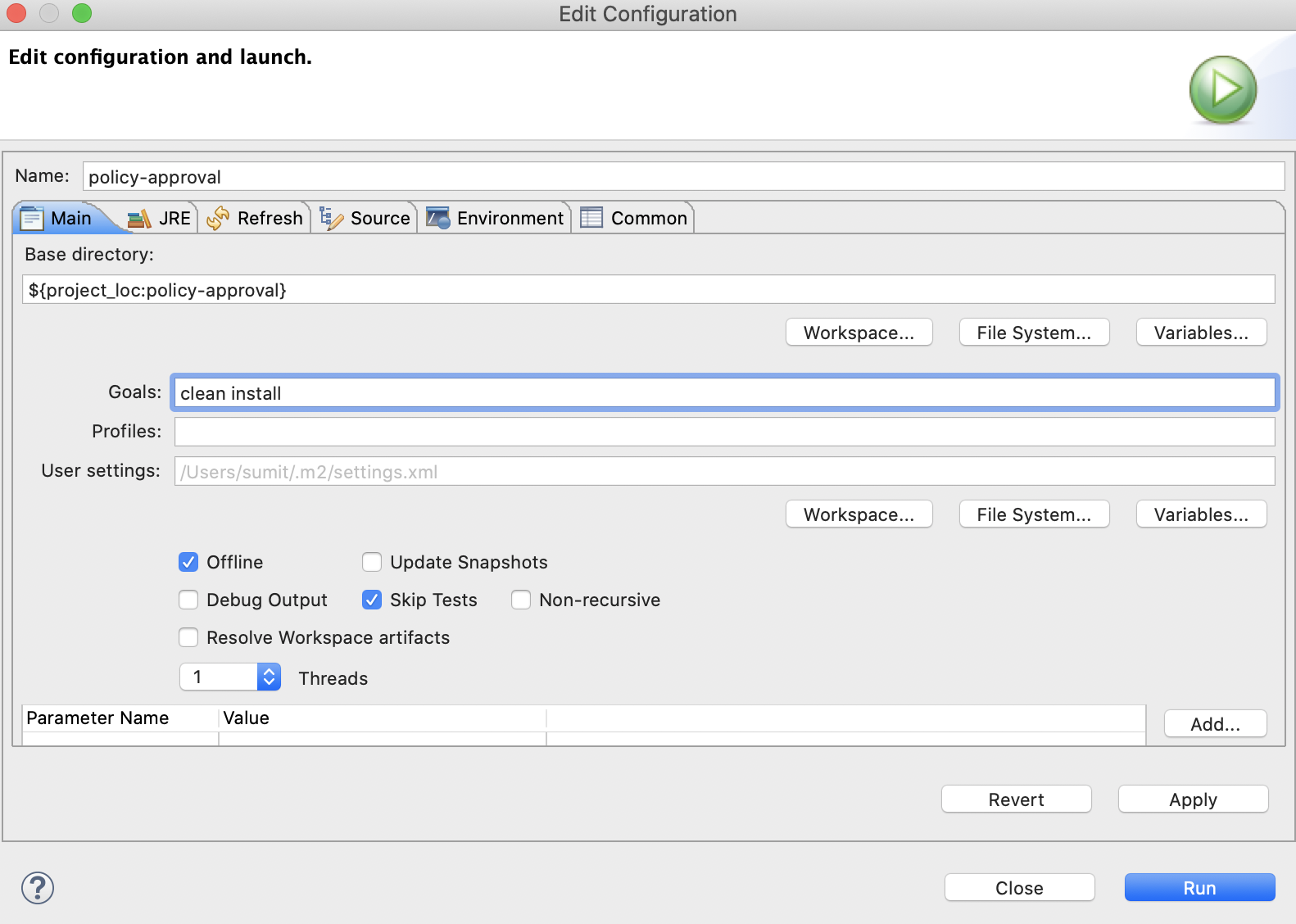
**Script Format:** groovy

**Script Type:** Inline Script

**Script:** println "Hello This is my first process application”

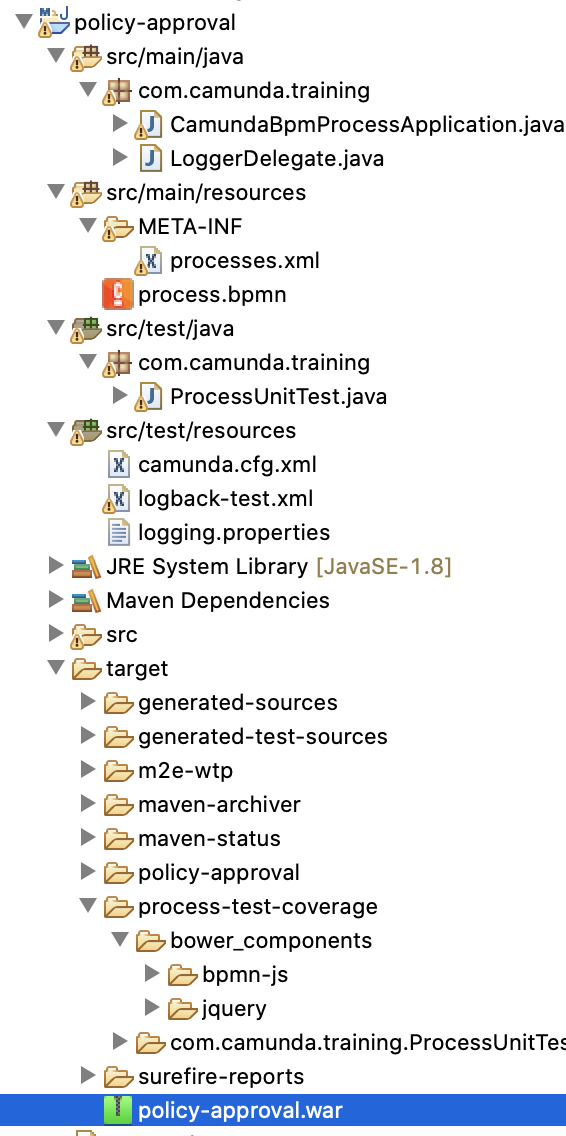
**Now generate build and deploy to camunda tomcat server.**

Right click on project—> Run As—> Maven build

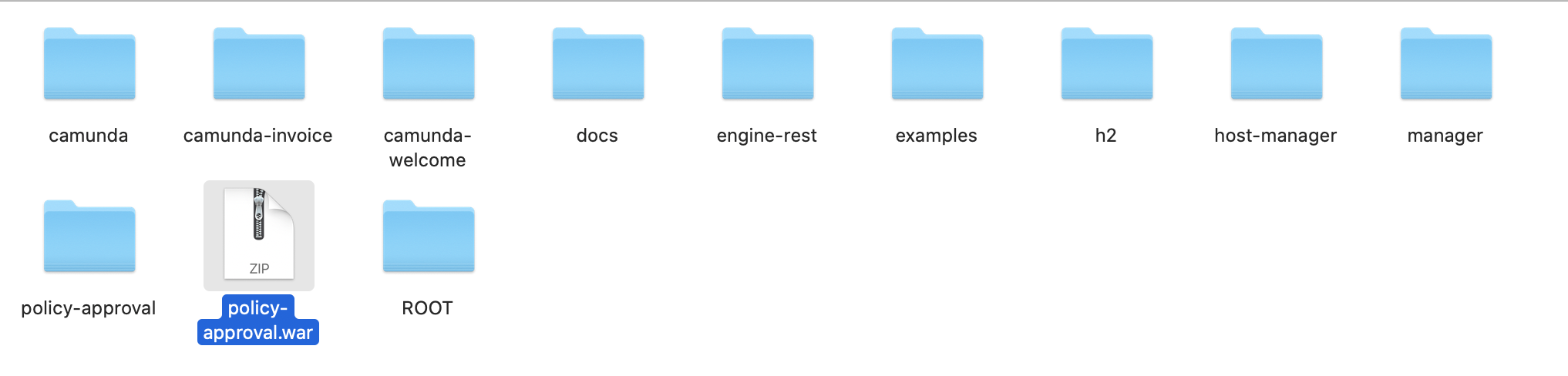
****

**Goals:** clean install

**Click check boxes:** Offline and Skip Tests

Then hit the Run button. This will generate policy-approval.war into target folder as per below screen.

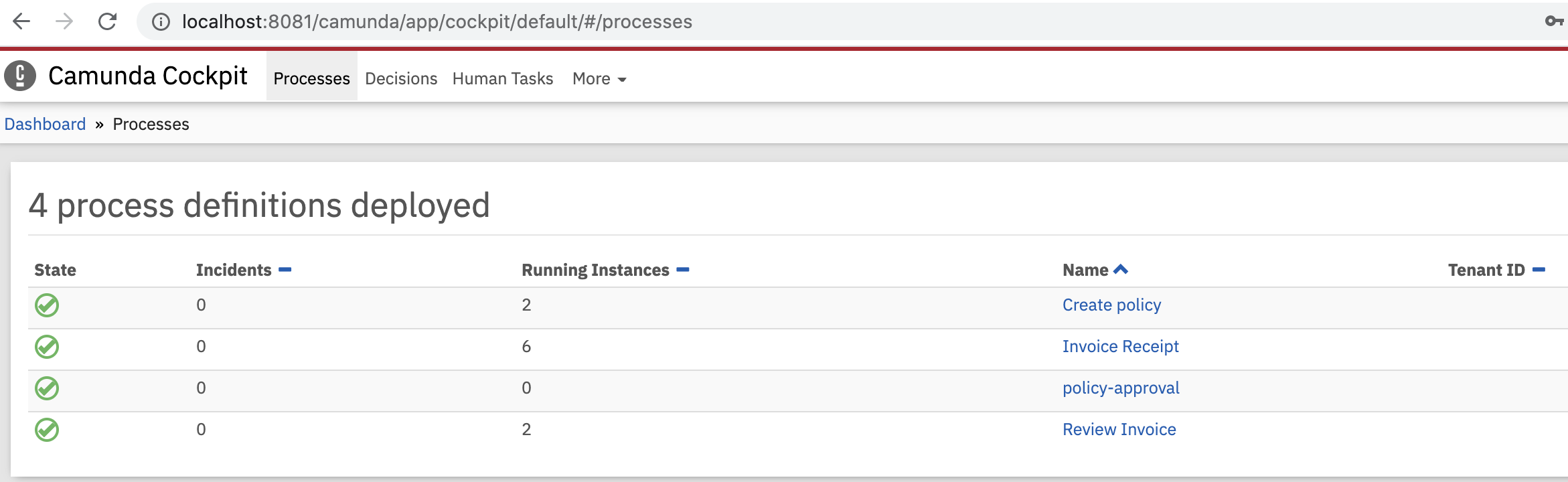
Deploy generated policy-approval.war to camunda-bpm-tomcat-7.13.0/server/apache-tomcat-9.0.33/webapps directory.

****

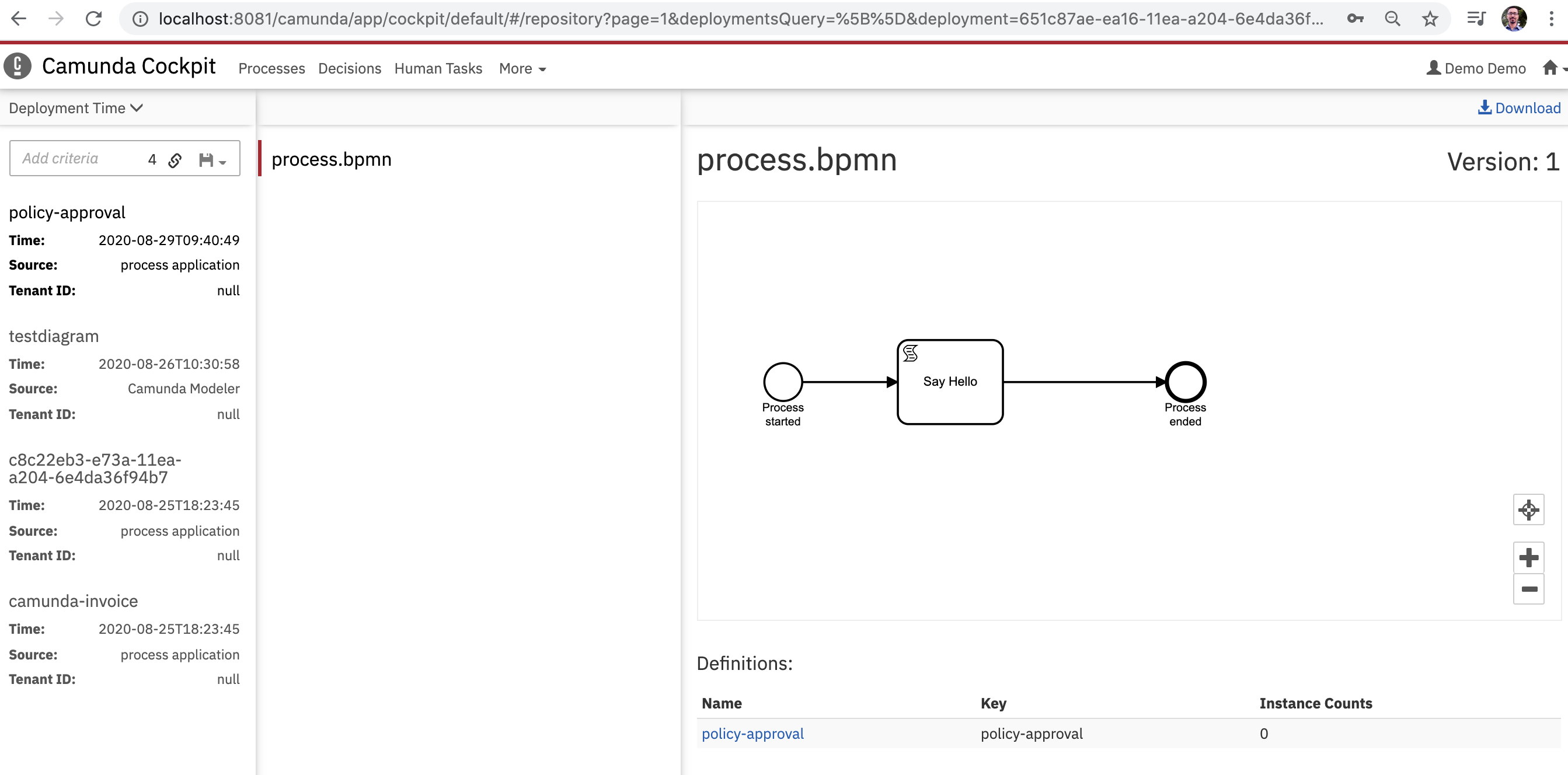
**policy-approval.war will be deployed and can be verified from tomcat console log**

29-Aug-2020 09:40:49.571 INFO [Catalina-utility-1] org.camunda.commons.logging.BaseLogger.logInfo ENGINE-08050 Process application policy-approval successfully deployed

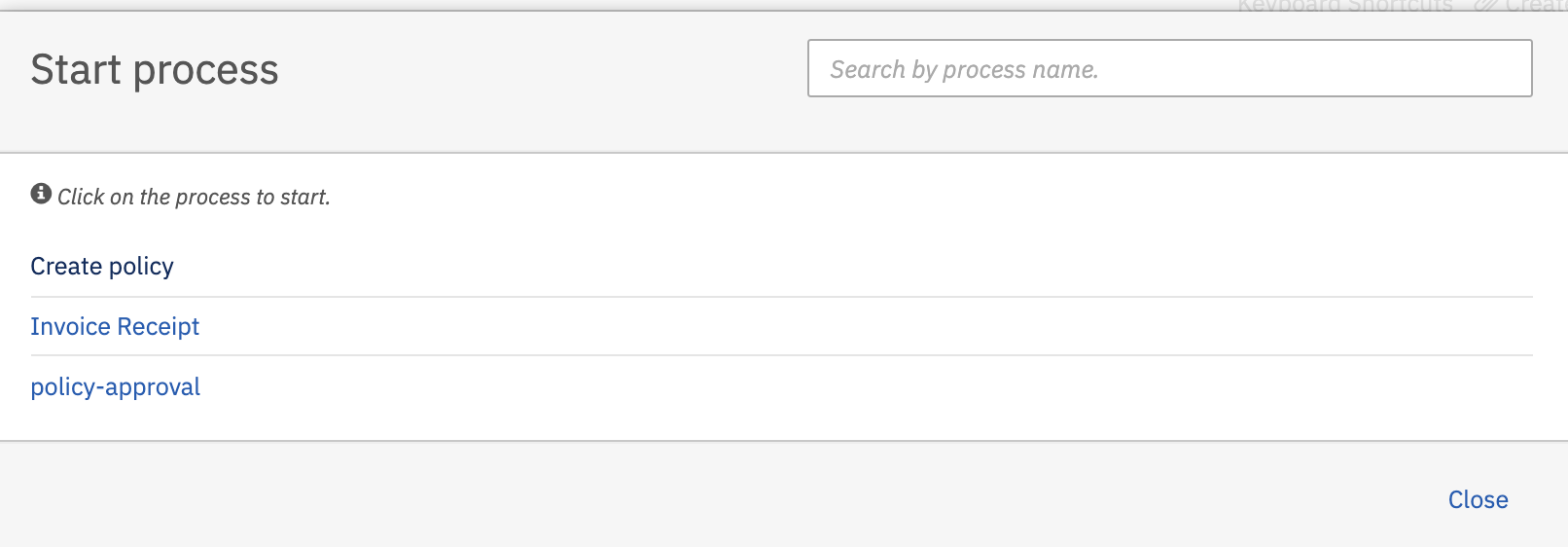
**Now let’s see deployed application through camunda Cockpit web application.**

**Processes:**

**Deployments:**

****

**Let run the deployed process through camunda tasklist web application.**

****

Hit the poilcy-approval and then hit the start button to start process.

Observe the tomcat console log to verify process is started and script task is executed.

29-Aug-2020 09:40:49.579 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/Volumes/Trainings/Camunda/Softwares/camunda-bpm-tomcat-7.13.0/server/apache-tomcat-9.0.33/webapps/policy-approval.war] has finished in [3,289] ms

Hello This is my first process applcation

Now let’s understand the process application components.

**CamundaBpmProcessApplication.java**

The Process Application class constitutes the interface between your application and the process engine.

@ProcessApplication

public class CamundaBpmProcessApplication extends ServletProcessApplication {

private static final String PROCESS\_DEFINITION\_KEY = "policy-approval";

/\*\*

\* In a @PostDeploy Hook you can interact with the process engine and access

\* the processes the application has deployed.

\*/

@PostDeploy

public void onDeploymentFinished(ProcessEngine processEngine) {}

}

**processes.xml**

The last step to set up the process application is to add the META-INF/processes.xml deployment descriptor file. This file allows us to provide a declarative configuration of the deployment(s) this process application makes to the process engine.

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

<process-application

xmlns=*"http://www.camunda.org/schema/1.0/ProcessApplication"*

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

<process-archive>

<process-engine>default</process-engine>

<properties>

<property name=*"isDeleteUponUndeploy"*>false</property>

<property name=*"isScanForProcessDefinitions"*>true</property>

</properties>

</process-archive>

</process-application>