Integrating Flowable with a Spring Boot application

Integrating Flowable with a Spring Boot application is straightforward. Here's a step-by-step guide to get you started:

1. Setting Up a Spring Boot Project

You can create a Spring Boot project manually or use Spring Initializr.

Use Spring Initializr

1. **Go to Spring Initializr.**
2. **Select:**
    - **Project: Maven**
    - **Language: Java**
    - **Spring Boot Version: Latest stable version (e.g., 3.2.0)**
    - **Dependencies:**
        - **Spring Web**
        - **Spring Data JPA**
        - **H2 Database (for testing)**
        - **Flowable Spring Boot Starter**
3. **Click Generate Project and extract it.**

---

2. Adding Required Dependencies

If you are using Maven, update your pom.xml:

```
<dependencies>
  <!-- Spring Boot Starter Web -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <!-- Flowable Spring Boot Starter -->
  <dependency>
    <groupId>org.flowable</groupId>
```

```
    <artifactId>flowable-spring-boot-starter</artifactId>
    <version>6.8.0</version> <!-- Check for the latest version -->
  </dependency>

  <!-- Flowable REST API -->
  <dependency>
    <groupId>org.flowable</groupId>
    <artifactId>flowable-spring-boot-starter-rest</artifactId>
    <version>6.8.0</version>
  </dependency>

  <!-- H2 Database (for testing) -->
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>

  <!-- JPA (for persistence) -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
</dependencies>
```

---

3. Configuring Flowable

Modify src/main/resources/application.properties to configure Flowable:

```
# H2 Database Configuration
spring.datasource.url=jdbc:h2:mem:flowable;DB_CLOSE_DELAY=-1
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
```

spring.datasource.password=

 # JPA Properties (H2 Dialect)

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

 # Flowable Configuration

flowable.id-generator=dataSource

flowable.database-schema-update=true

flowable.async-executor-activate=true

What this does:

- **Uses H2 in-memory database (no external DB needed for testing).**
- **Auto-updates Flowable's database schema.**
- **Enables the async executor for background process execution.**

---

4. Creating a Simple BPMN Process

Flowable uses BPMN 2.0 (Business Process Model and Notation) to define workflows.

Steps to create a BPMN file:

1. **Create a new file inside src/main/resources/processes/.**
2. **Name it my-process.bpmn20.xml.**
3. **Add the following content:**

```
<?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"
        xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL
BPMN20.xsd">

  <process id="myProcess" name="Simple Process" isExecutable="true">
    <startEvent id="startEvent" />
    <sequenceFlow id="flow1" sourceRef="startEvent" targetRef="serviceTask" />
    <serviceTask id="serviceTask" name="My Service Task"
flowable:class="com.example.flowable.MyServiceTask"/>
    <sequenceFlow id="flow2" sourceRef="serviceTask" targetRef="endEvent" />
```

```
    <endEvent id="endEvent" />
  </process>
</definitions>
```

What this does:

- **Defines a BPMN process with:**
  - **Start Event → Service Task → End Event.**
- **Calls a Java Service Task (MyServiceTask).**

---

5. Implementing a Java Service Task

Create the MyServiceTask class to implement logic inside the BPMN process.

```
package com.example.flowable;

import org.flowable.engine.delegate.DelegateExecution;
import org.flowable.engine.delegate.JavaDelegate;
import org.springframework.stereotype.Component;

@Component
public class MyServiceTask implements JavaDelegate {

  @Override
  public void execute(DelegateExecution execution) {
    System.out.println("Executing Service Task - Process ID: " +
execution.getProcessInstanceId());
  }
}
```

What this does:

- **Implements JavaDelegate to perform custom logic.**
- **Prints a message when the process reaches the service task.**

---

6. Creating a REST Controller

To start a process via an API, create a Spring Boot REST Controller.

```
package com.example.flowable;

import org.flowable.engine.RuntimeService;
import org.flowable.engine.runtime.ProcessInstance;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/processes")
public class ProcessController {

  @Autowired
  private RuntimeService runtimeService;

  @PostMapping("/start")
  public String startProcess() {
    ProcessInstance processInstance =
runtimeService.startProcessInstanceByKey("myProcess");
    return "Process Started: " + processInstance.getId();
  }
}
```

What this does:

- **Exposes an API (/processes/start) to trigger the Flowable process.**
- **Starts a new process instance of myProcess.**

---

7. Running the Application

Run the Spring Boot application with:

```
mvn spring-boot:run
```

or

```
./gradlew bootRun
```

---

8. Testing the Process Execution

Trigger the Process

Use Postman or cURL to trigger the process.

curl -X POST http://localhost:8080/processes/start

Check Logs

If everything works, you should see:

Executing Service Task - Process ID: 12345

---

9. Accessing Flowable REST API

Flowable provides REST APIs to manage processes.

- **Get all process definitions:**

curl -X GET http://localhost:8080/flowable-rest/service/repository/process-definitions

- **Get all active process instances:**

curl -X GET http://localhost:8080/flowable-rest/service/runtime/process-instances

---

10. Optional: Flowable Admin UI

You can use Flowable Admin UI for monitoring:

1. **Add Admin Dependency**

   <dependency>

   <groupId>org.flowable</groupId>

   <artifactId>flowable-ui-admin</artifactId>

   <version>6.8.0</version>

   </dependency>
2. **Run the Admin App**
3. **mvn spring-boot:run**
4. **Access UI:**
   Open http://localhost:8080/flowable-admin.
5. **Login with:**
   - **Username: admin**
   - **Password: test**