camunda
the business process company

Camunda BPM for Developers

Getting started with the Camunda BPM Platform

Agenda

Day 1

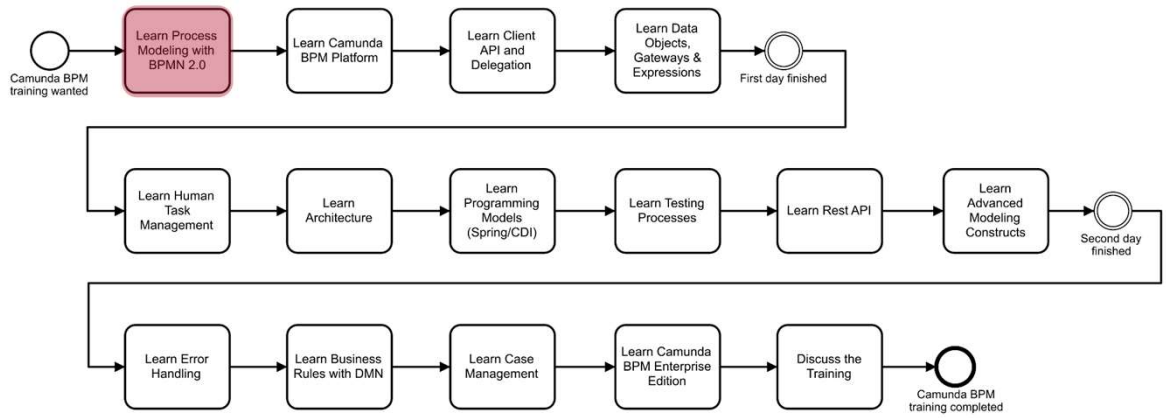
1. Process Modelling with BPMN 2.0
2. Camunda BPM Platform
3. Client-API and Delegation
4. Data Objects, Gateways & Expressions

Day 2

5. Human Task Management
6. Architecture
7. Programming Models (Spring/CDI)
8. Testing Processes
9. REST API
10. Advanced BPMN Constructs

Day 3

11. Error Handling
12. Business Rules with DMN
13. Case Management
14. Camunda BPM Enterprise Edition
15. Wrap up: Outlook, Feedback and Open Discussion



Business Process Model and Notation (BPMN)

- BPMN is a worldwide OMG standard
- Latest version is BPMN 2.0
- Lots of companies joined the standardization committee
- All major BPMS vendors go for BPMN

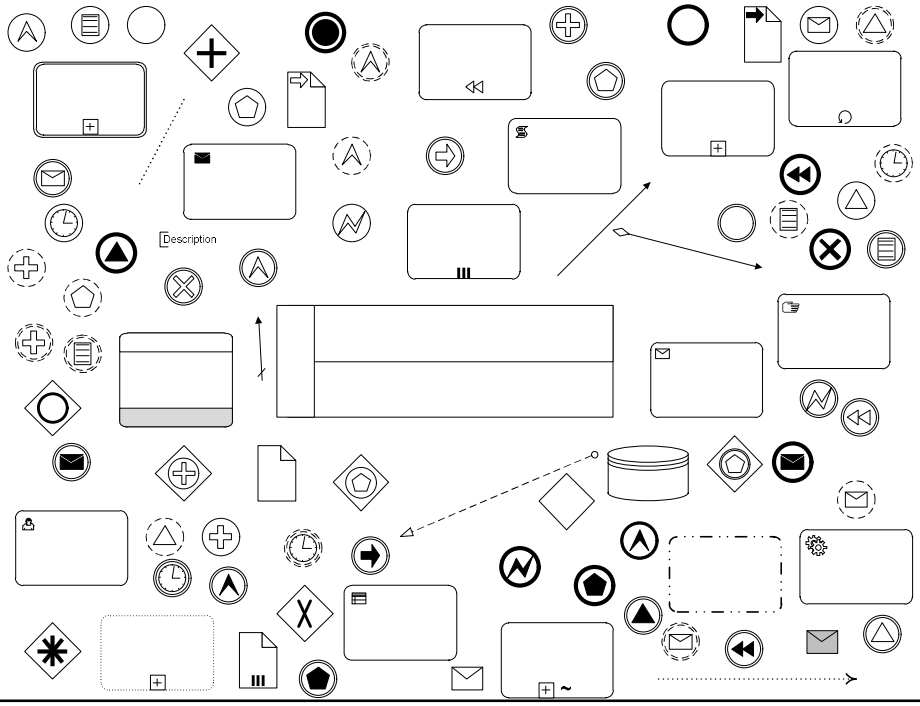
OBJECT MANAGEMENT GROUP

ORACLE SAP IBM

camunda the business process company

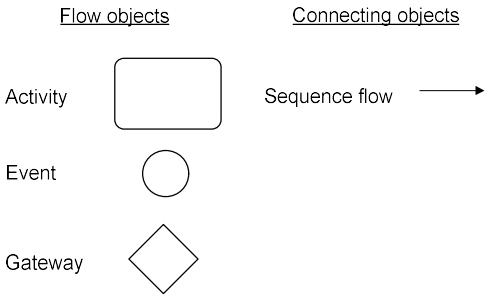
trisotech INTALIO

NIST Global360 TIBCO The Power of Now FUJITSU

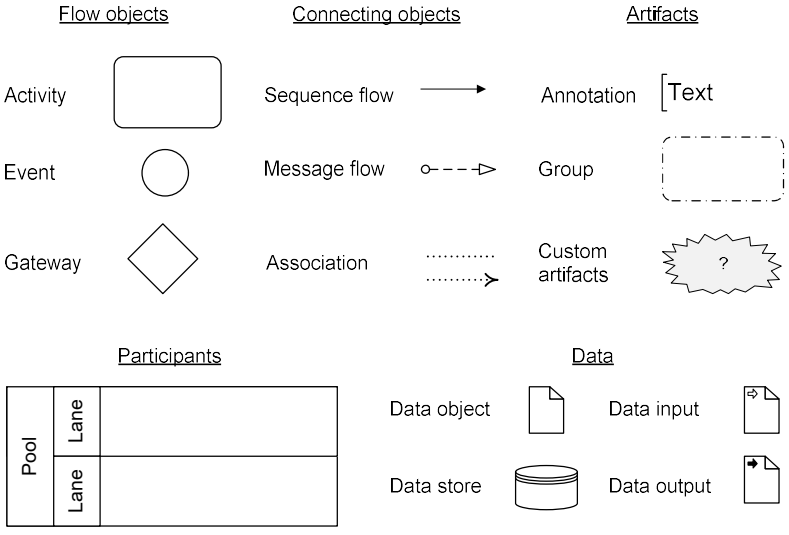


Our First BPMN Process Model

Basic Elements



Basic Elements - Documentation



The Scope of BPMN

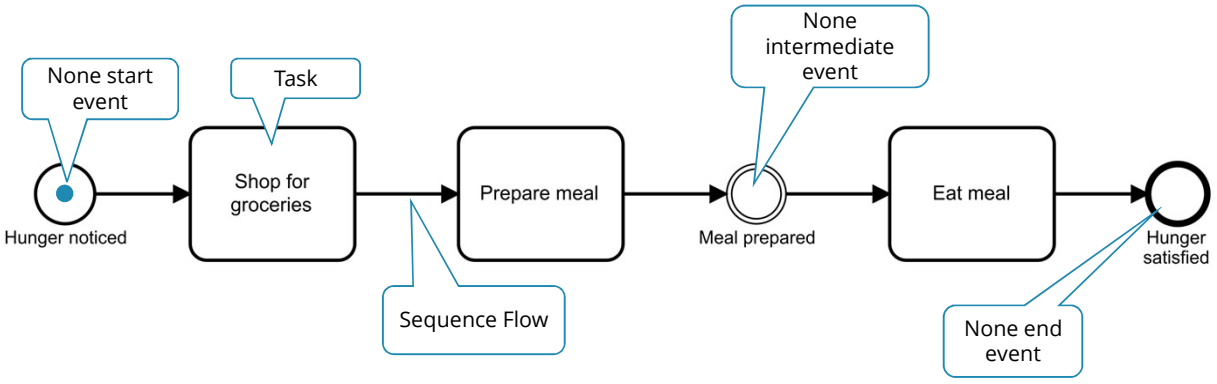
BPMN at it's best:

- Processes

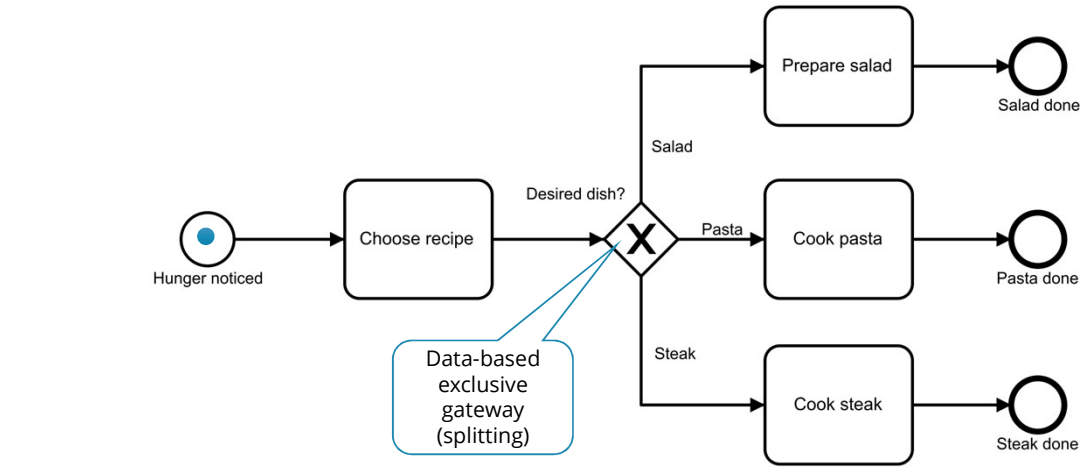
BPMN is not suitable for:

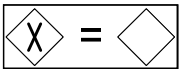
- Process landscapes
- Organizational landscapes
- Data
- Strategies
- Business rules
- IT landscapes

Tasks and events

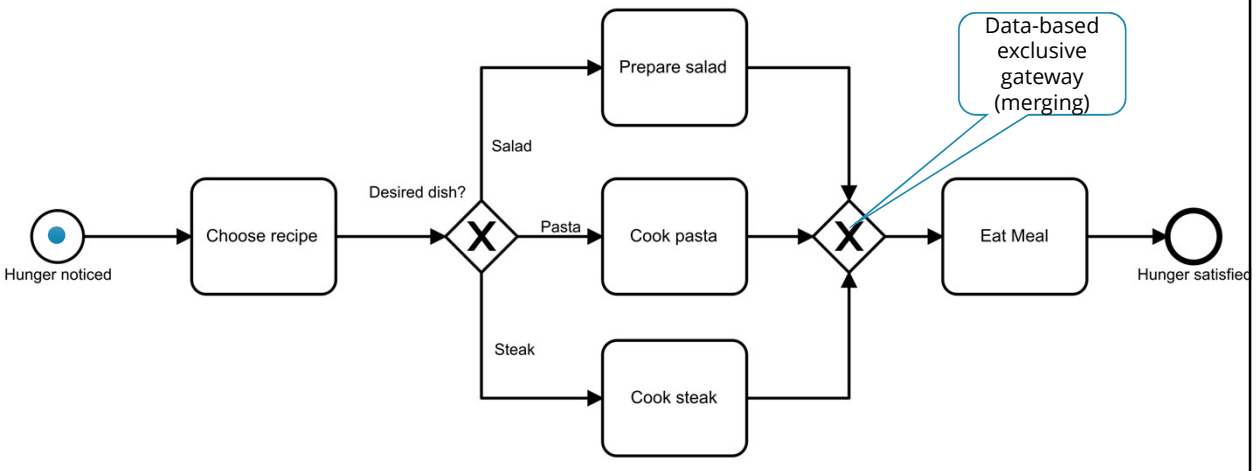


The XOR-gateway

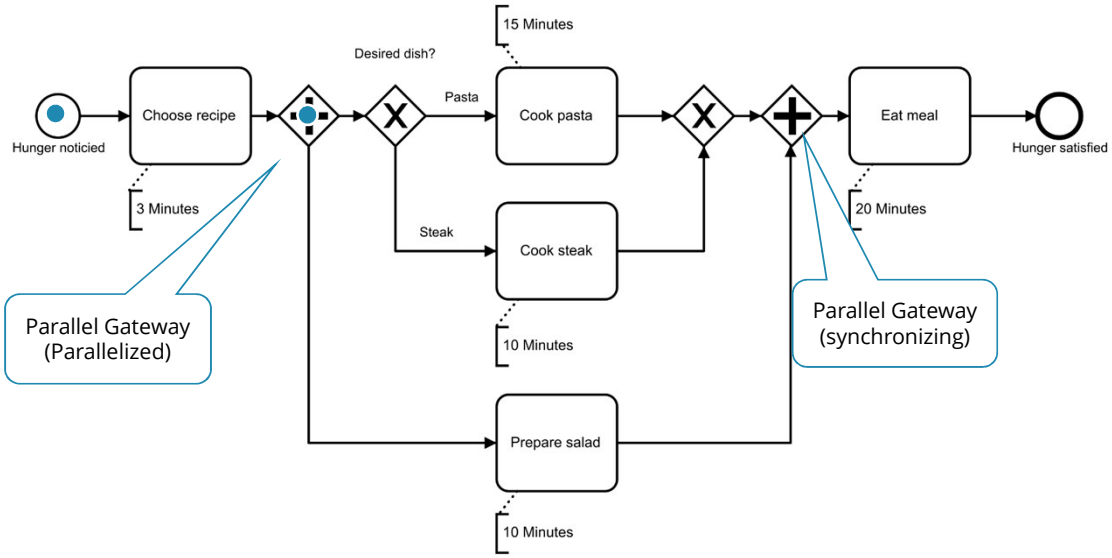




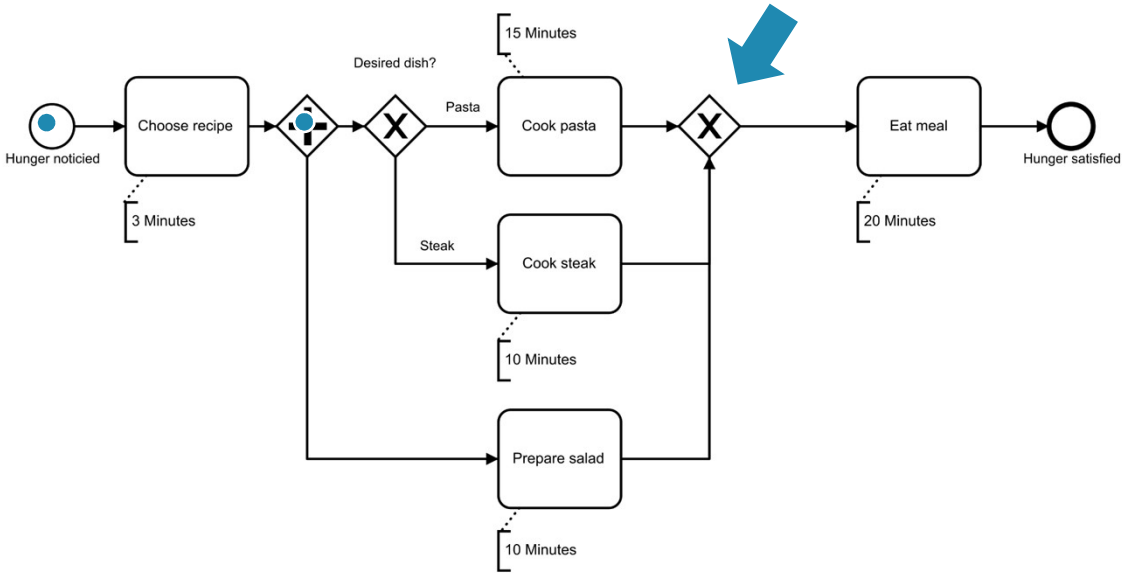
XOR-gateways can also merge



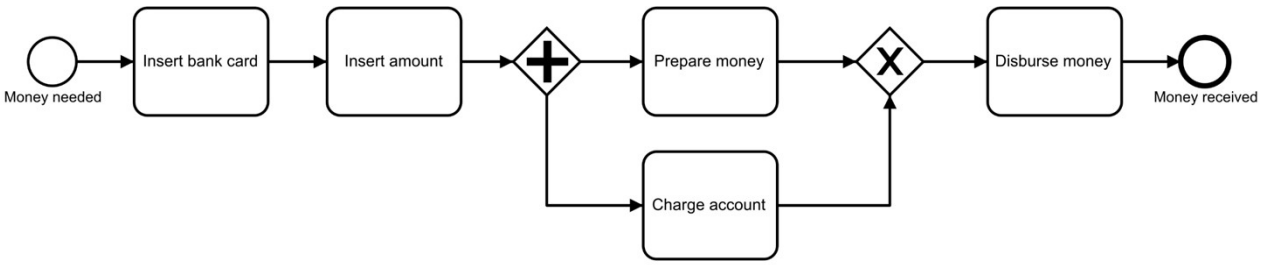
Preparing salad and main course at the same time



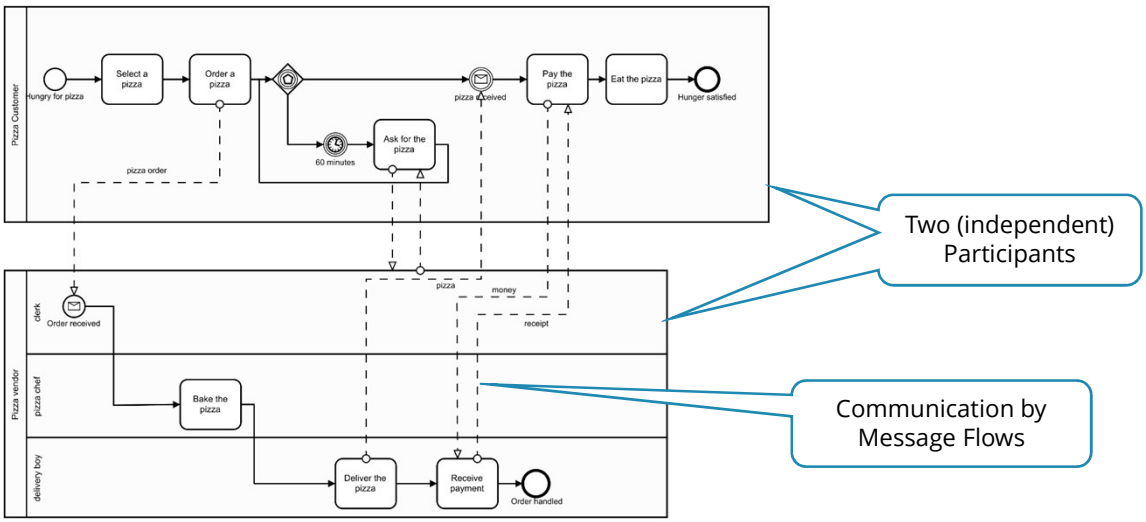
What happens in this process?



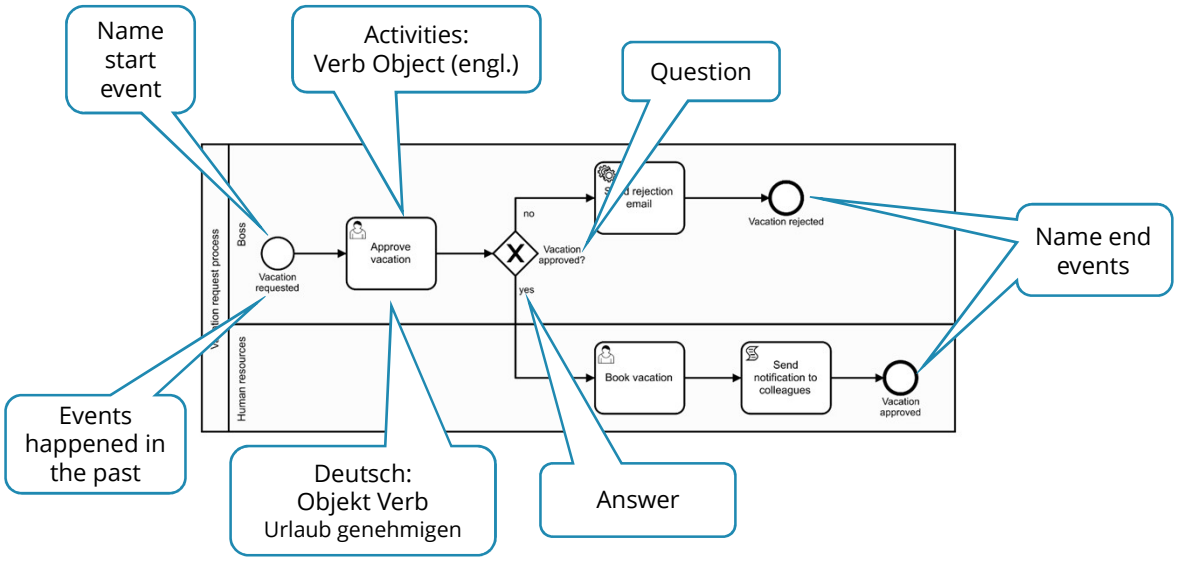
Example: ATM

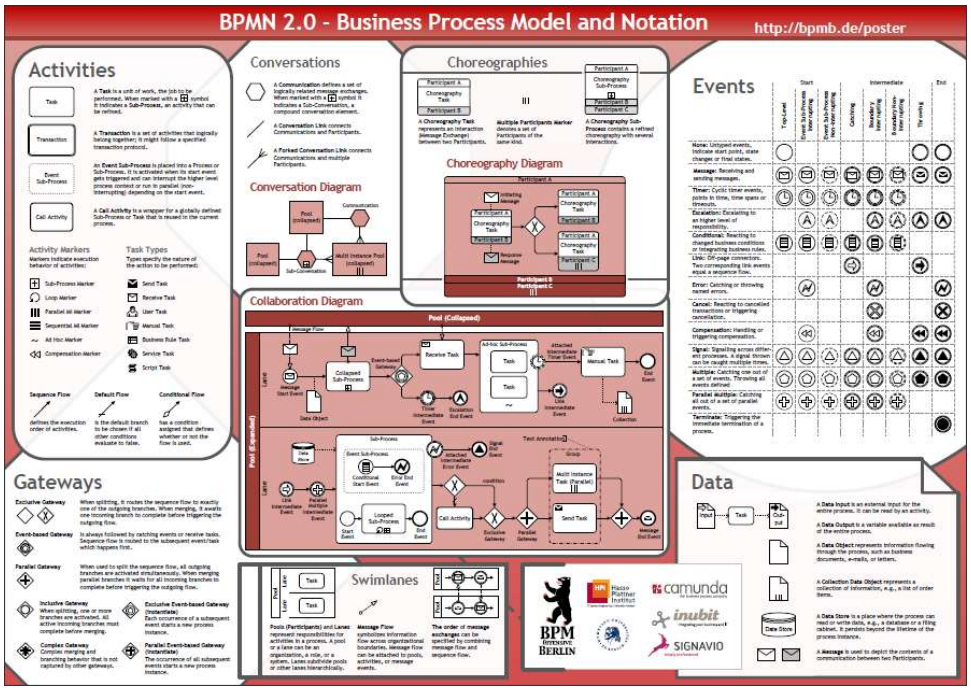


Collaboration Diagram



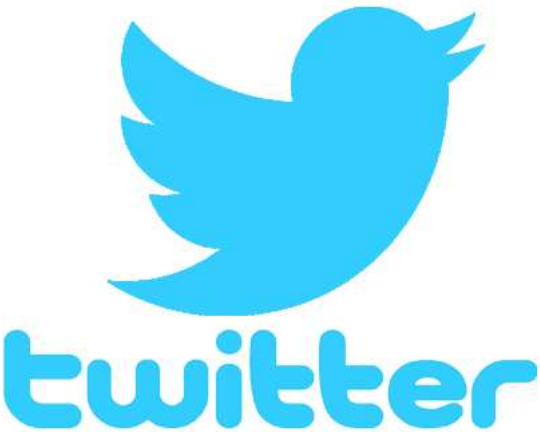
Create Readable Process Models





Scenario for Exercises

You Know Twitter?



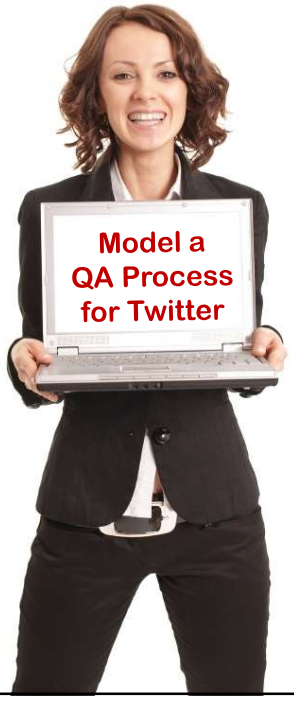
Imagine the Anti Agile Inc.

Why do we need Twitter?

- We do not trust our employees.
- We want to control everything.
- We do not like Twitter,
- But we want to be part of it.

The Anti Agile Inc. Needs Twitter QA!

Group Exercise 0



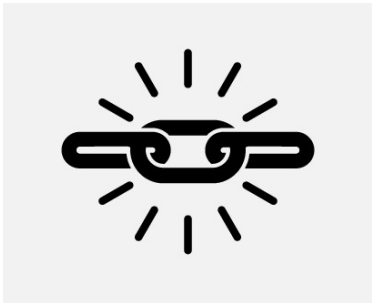
Use a whiteboard or flipchart
to model a Twitter QA process

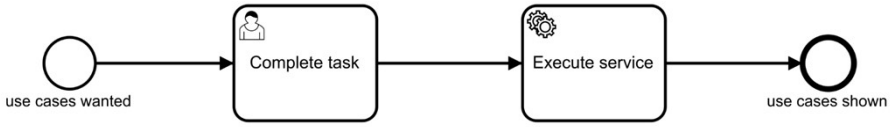
QA Process for Twitter

Process Automation Use Cases

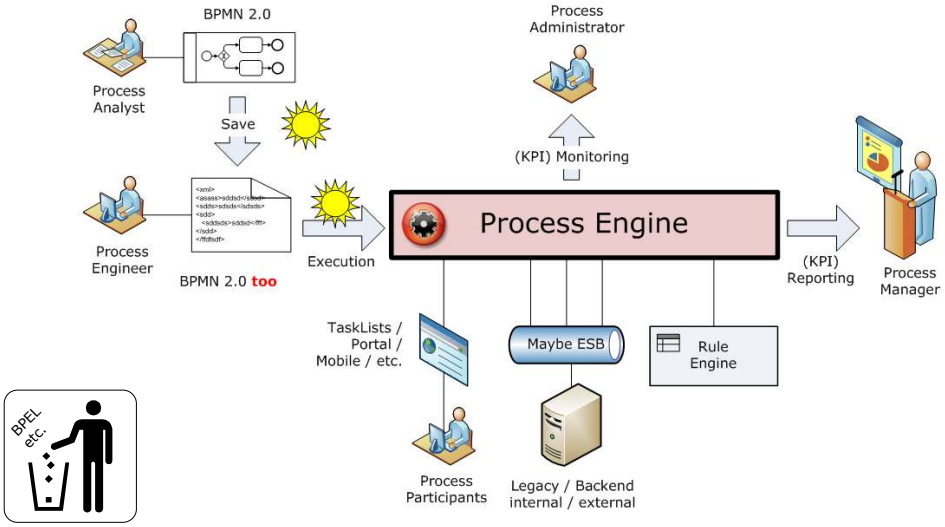
Human Task Management

System Integration





What's So Cool About BPMN 2.0

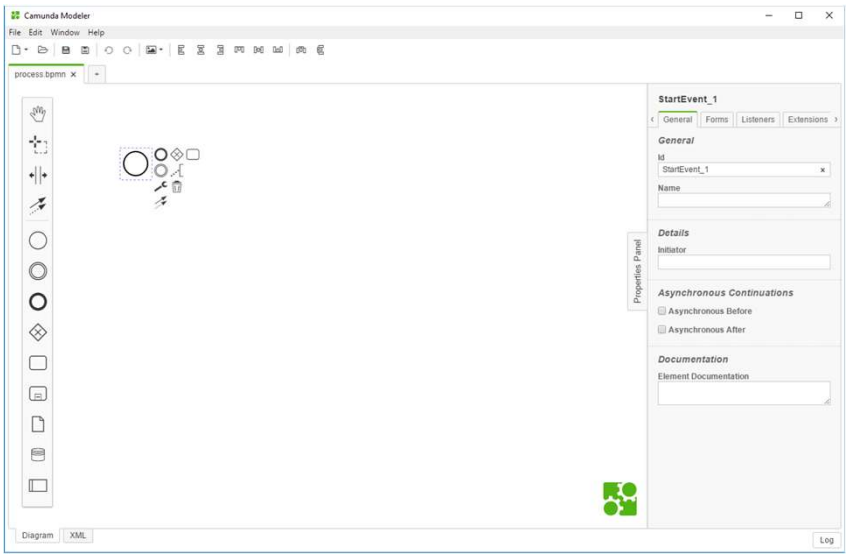


Why Using a Process Engine?

- Features
  - Task management
  - Timers & escalation
  - Error management and retry mechanisms
  - Versioning
  - The power of BPMN 2.0
  - Tools (modeler, cockpit, …)
  - …
- Transparency and Agility
  - Process is not hidden in the code
  - Process model can be used for monitoring and operations
- Quality

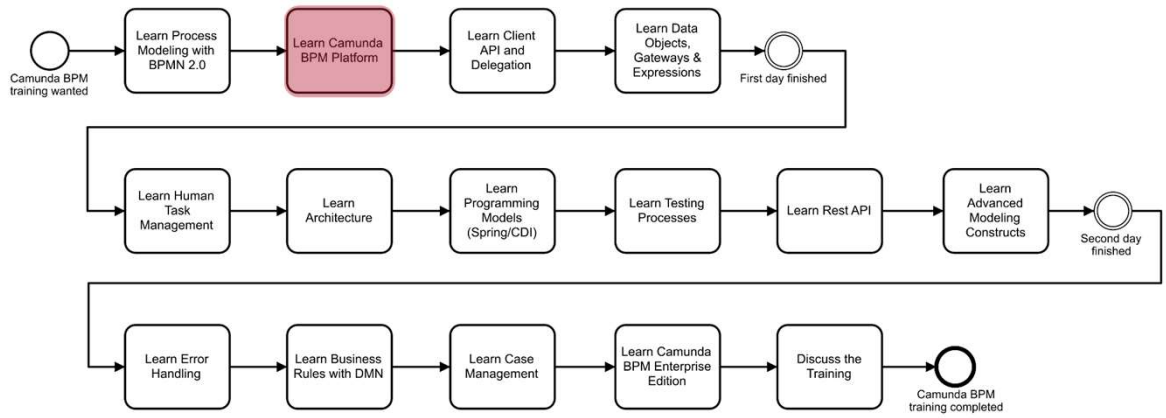
Demo

Start Modeling!

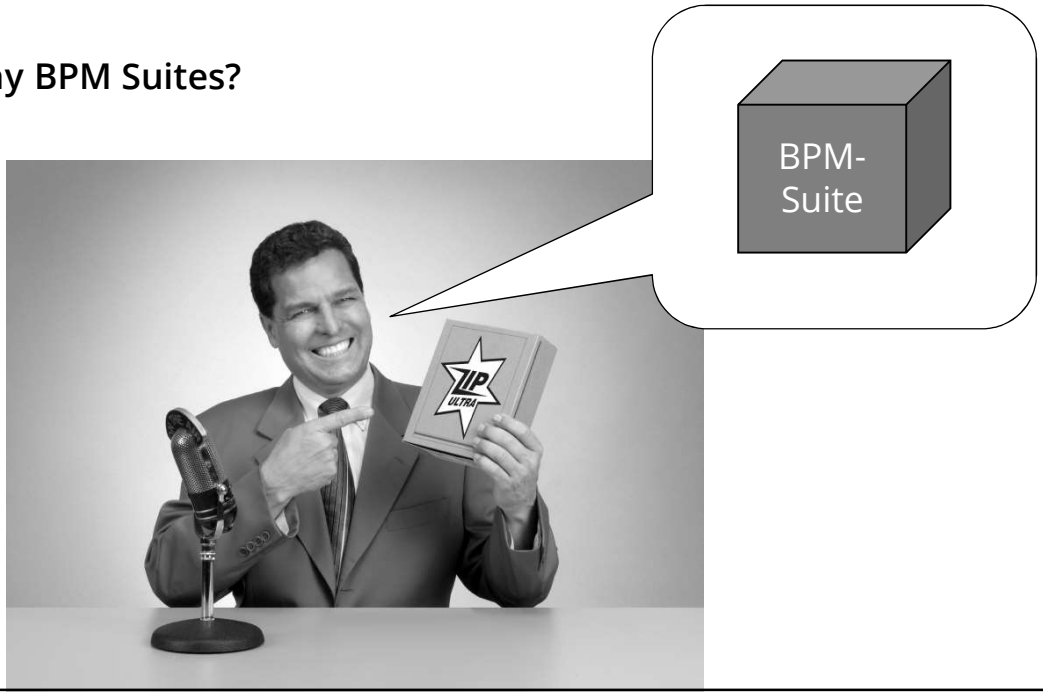


Exercise 1

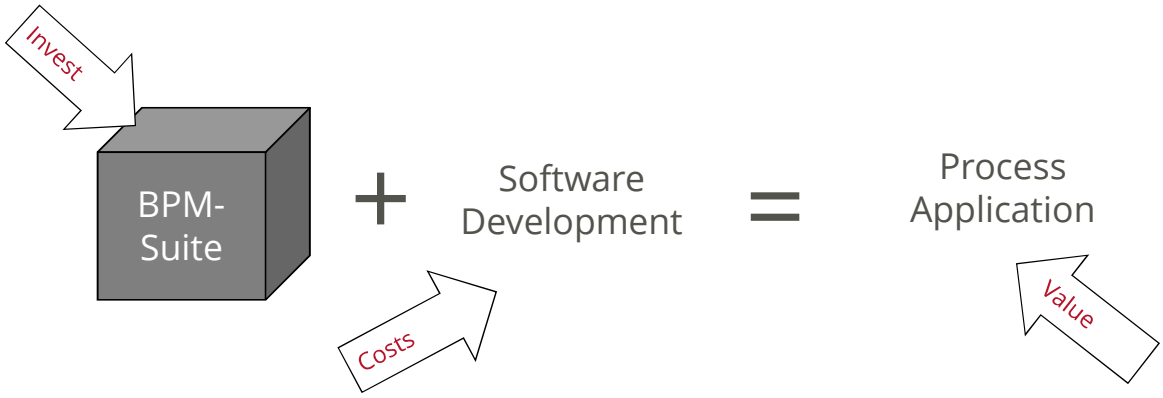
Follow the provided instructions for exercise 1



Shiny BPM Suites?

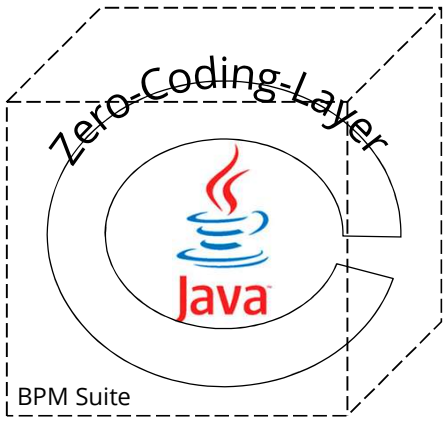


You Always Need Software Development



Tempting, but Deceptive

A Fundamentally Wrong Approach*



complicated
restrictive

Business User

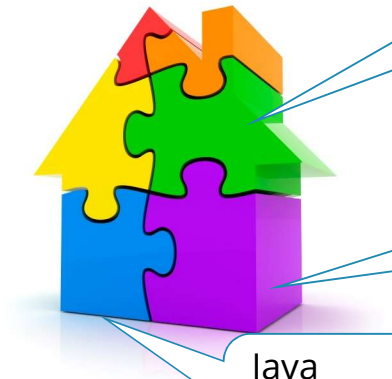
restrictive
proprietary

Software
Developer

***for automating core business processes**

Our Approach: BPM + Java

Best of Breed – Based on Standards



You name it!

Any component from the Java cosmos.

Camunda BPM

Process Automation, Monitoring, Operations, ...

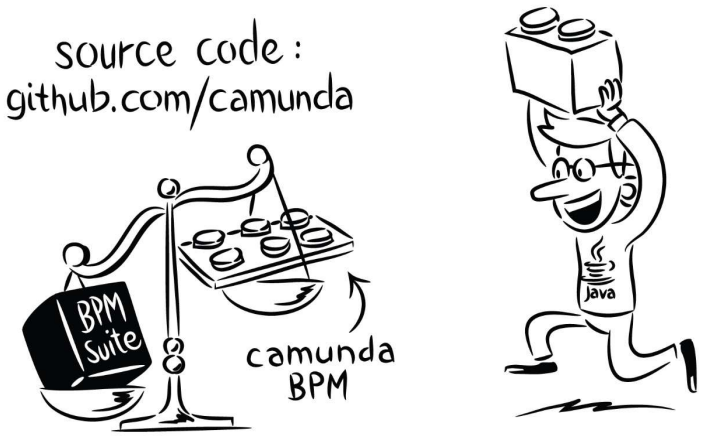
BPMN 2.0

Process Modeling, Requirements Engineering, Roundtrip, ...

Java

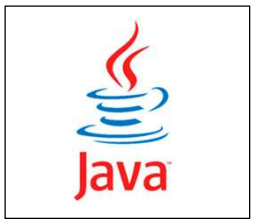
Persistence (JPA), Transactions (JTA, EJB3), Clustering, Connectivity (JAX-WS, JAX-RS, JAX-B), UI (JSF), ...

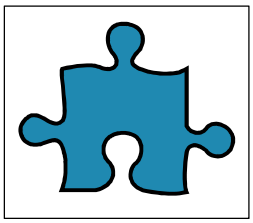
Lightweight and Open Source BPM Framework

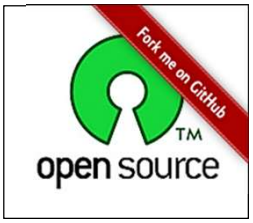


…Plus BPMN Based Business-IT-Alignment

Summary: Core Concepts of Camunda BPM





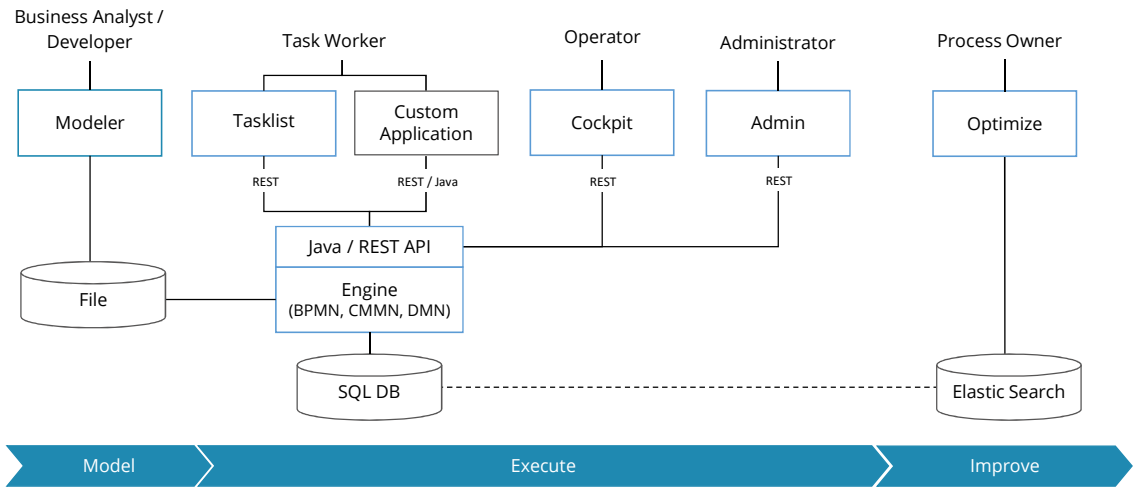


Decide Yourself!

BPM in a can

BPM cook-it-yourself

Camunda BPM Components

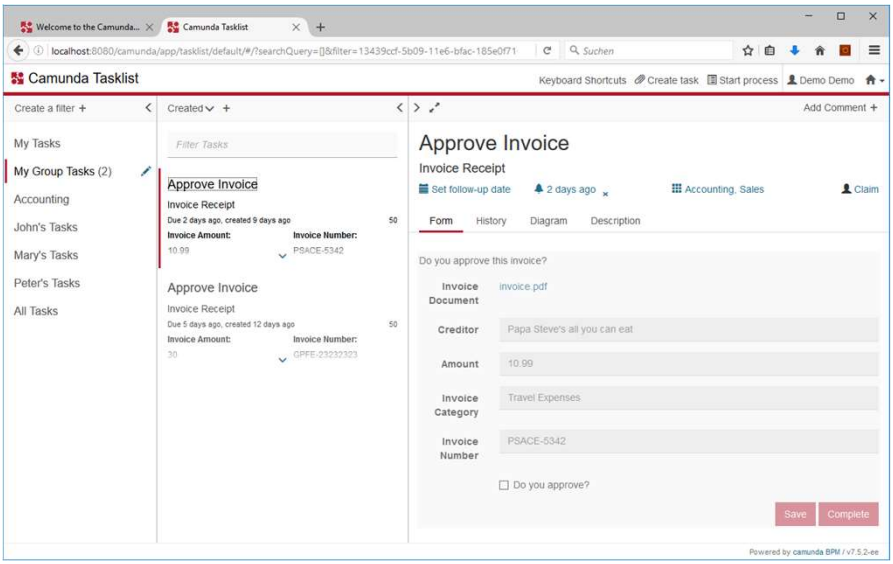


Exercise 2a

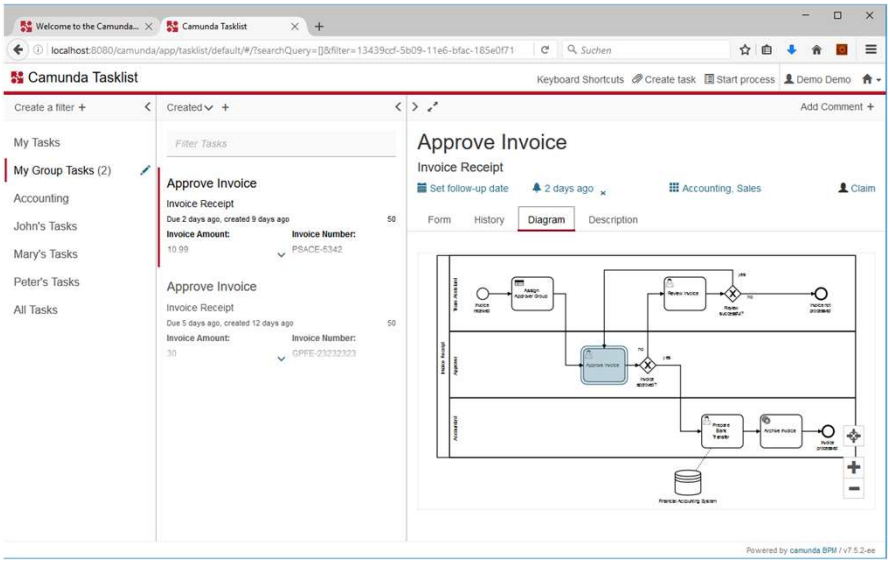
Follow the provided instructions for exercise 2a

Demo

Camunda Tasklist

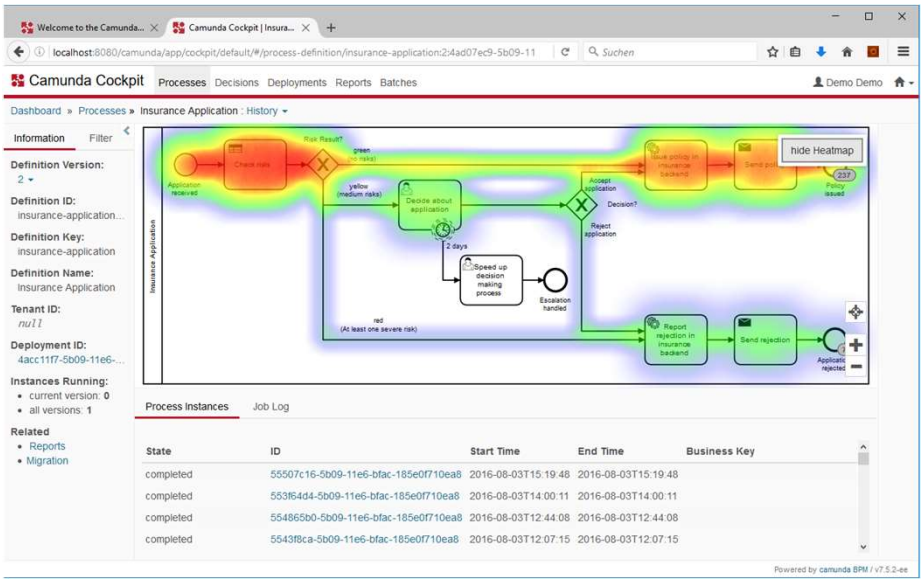


Process Overview



History view and Heatmap
Enterprise Feature

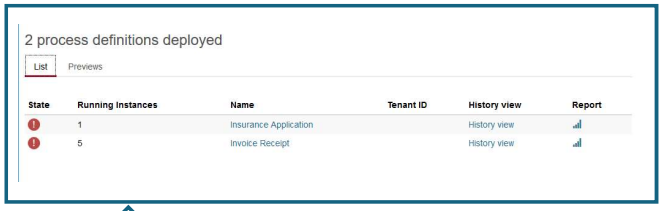
Camunda Cockpit



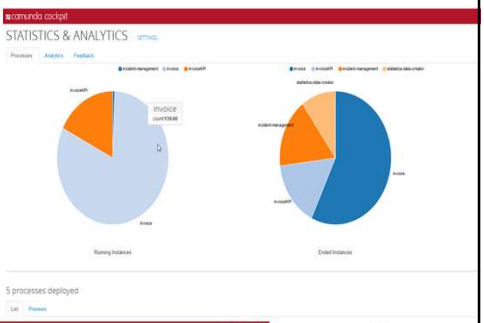
Cockpit Features

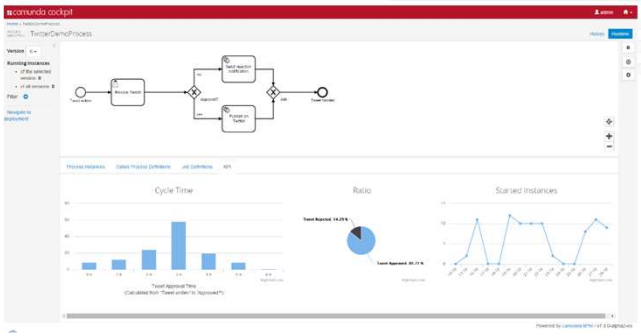
- Dashboard
- Monitor BPMN Processes
- Monitor DMN Decisions
- Browse deployments and resources in the process engine repository
- Graphical Process Instance Modification
- Auditing of Cockpit operations by REST API
- Batch view
- Reports (EE only)

Cockpit Plugins

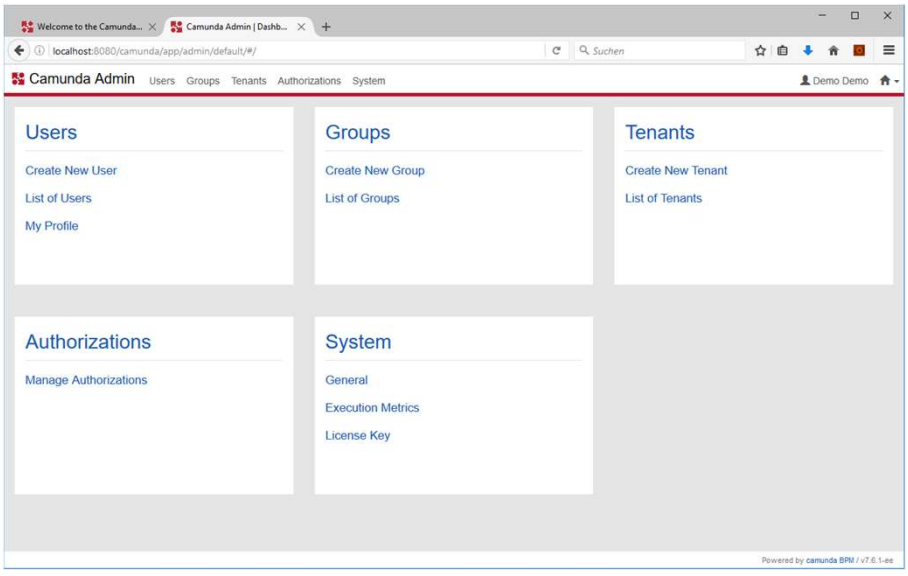


Cockpit Plugin Yeeeeah!

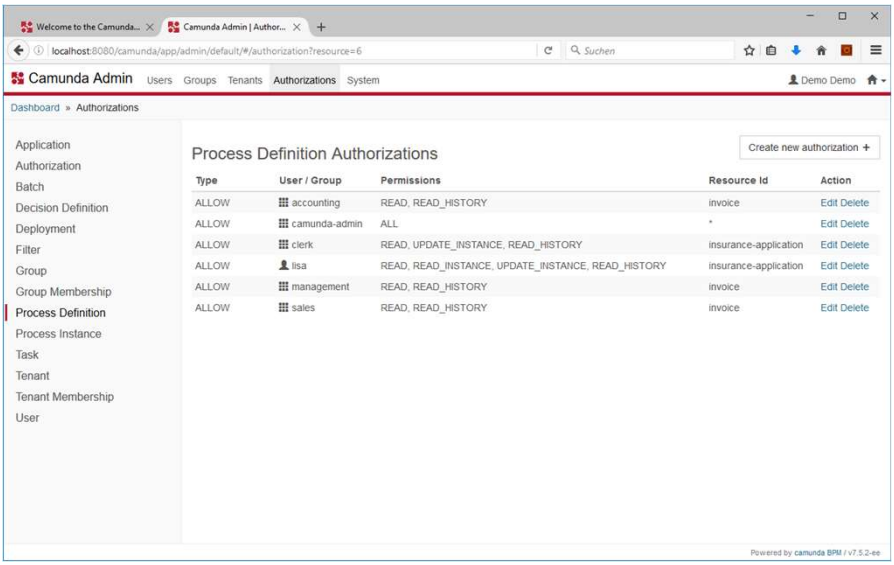




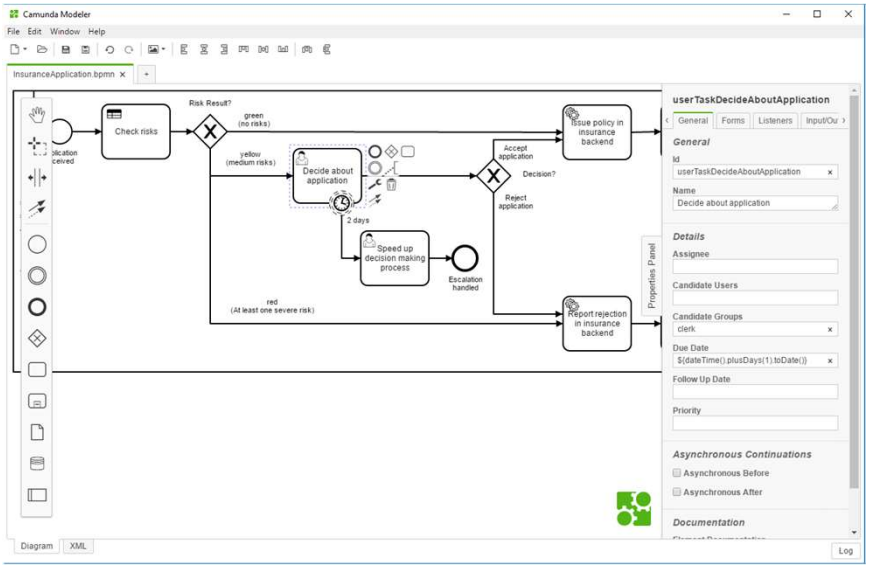
Camunda Admin



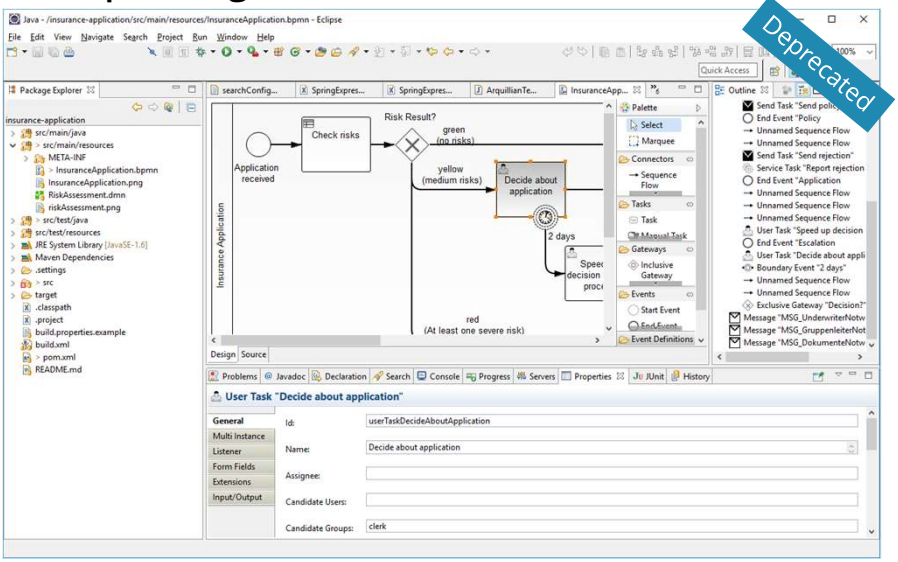
Authorizations



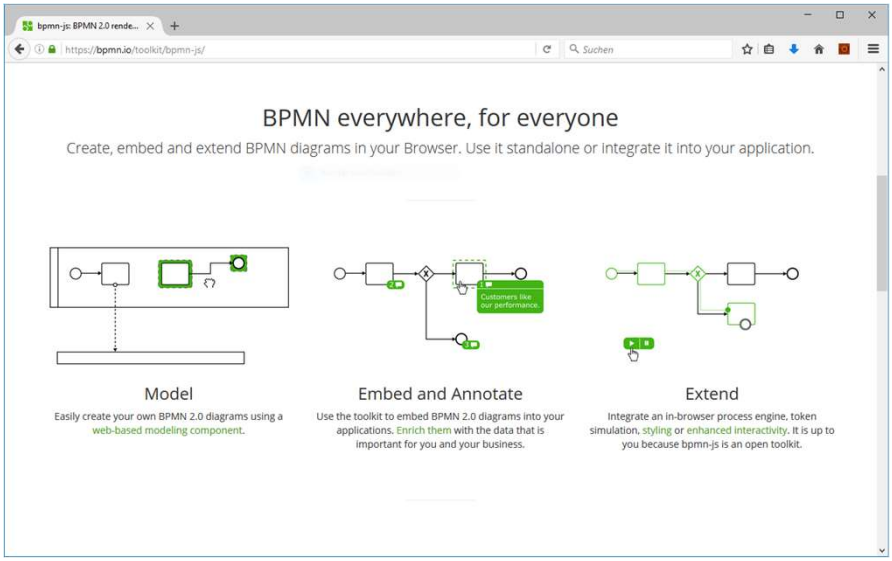
Camunda Modeler



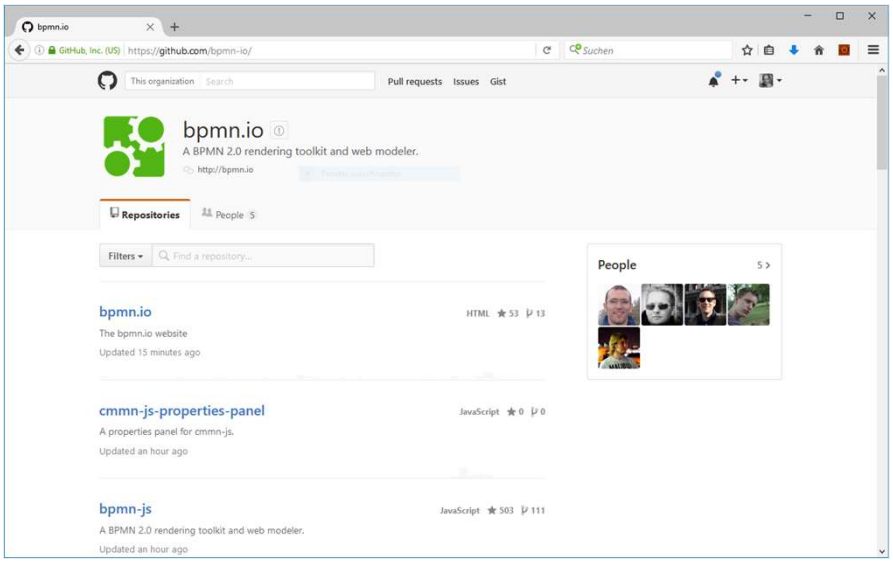
Camunda Eclipse Plugin



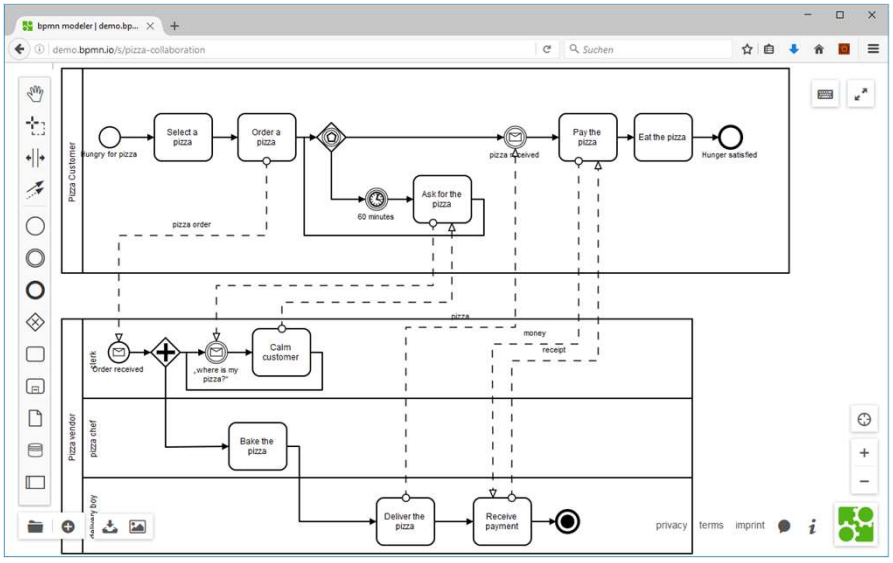
bpmn.io – a Developer Kit and Web Modeler for BPMN 2.0



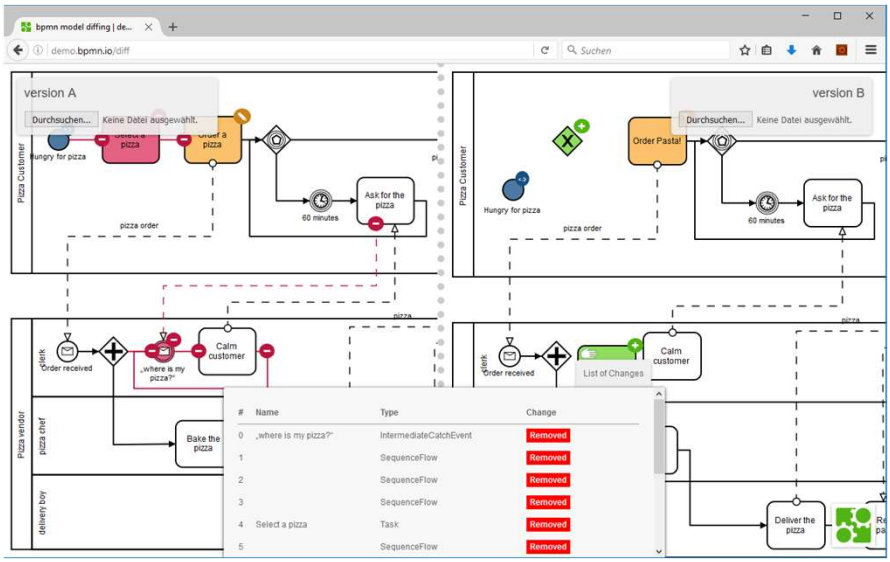
https://github.com/bpmn-io



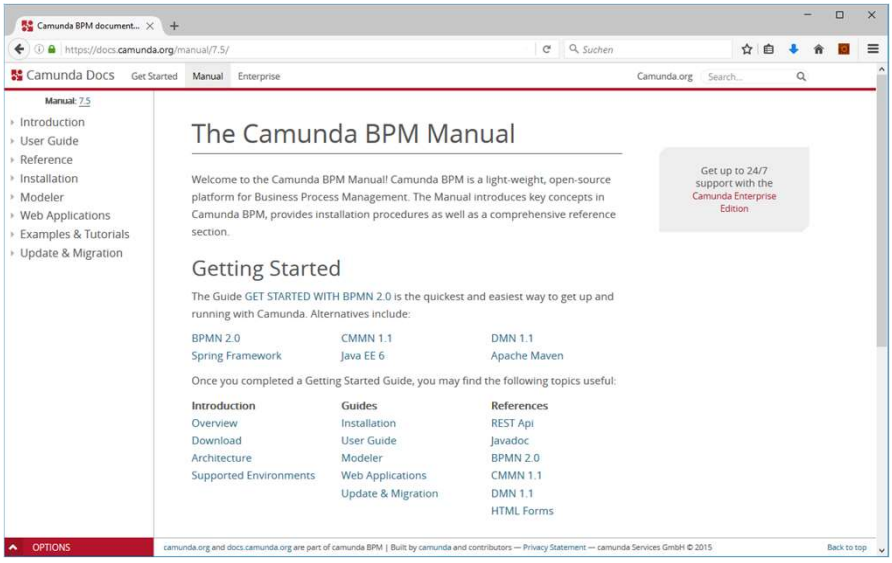
https://demo.bpmn.io



BPMN Diff With bpmn.io

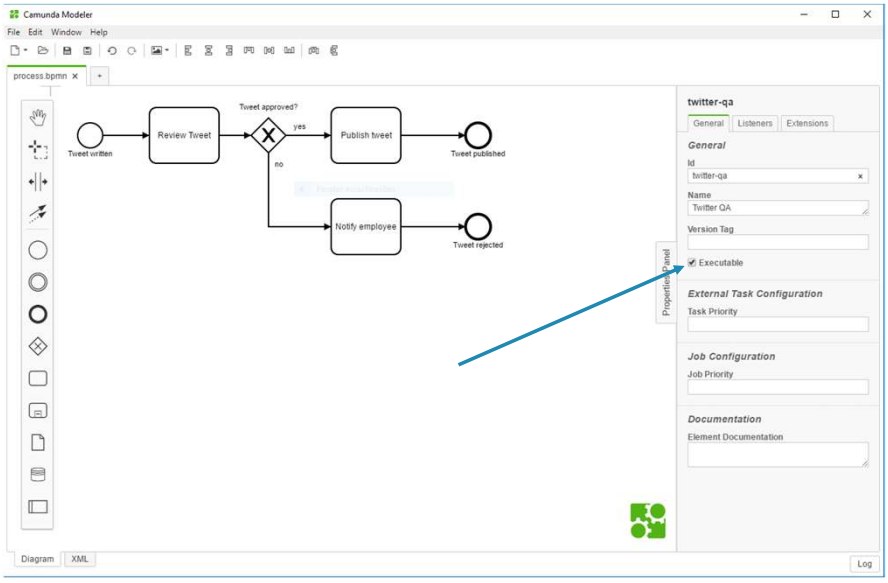


https://docs.camunda.org

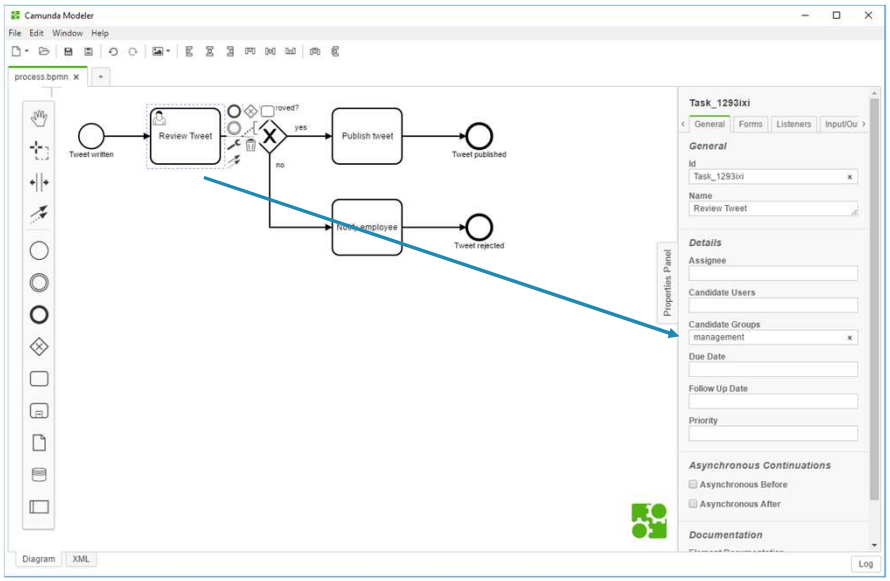


Prepare for First Launch

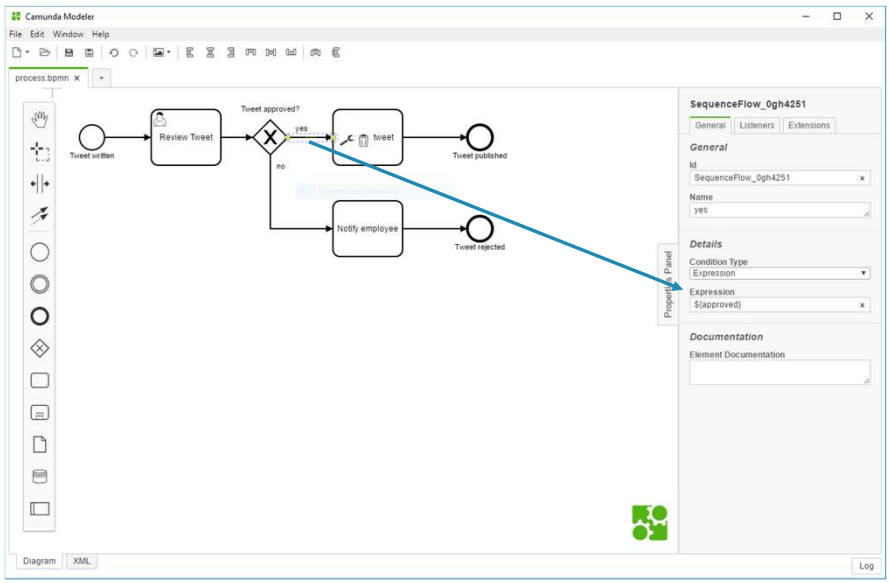
Make Process Executable



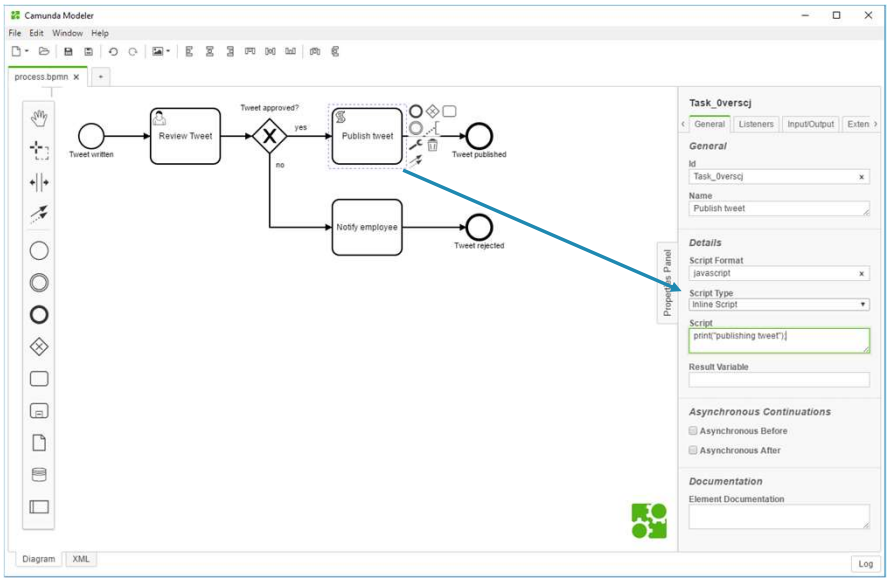
Assign User Task



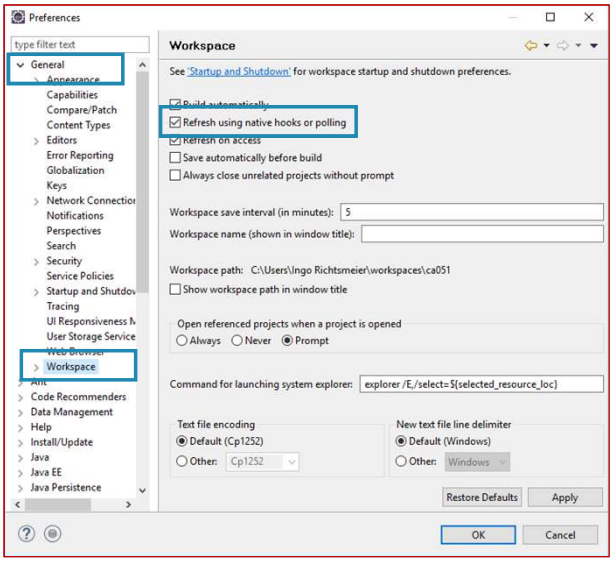
Conditions



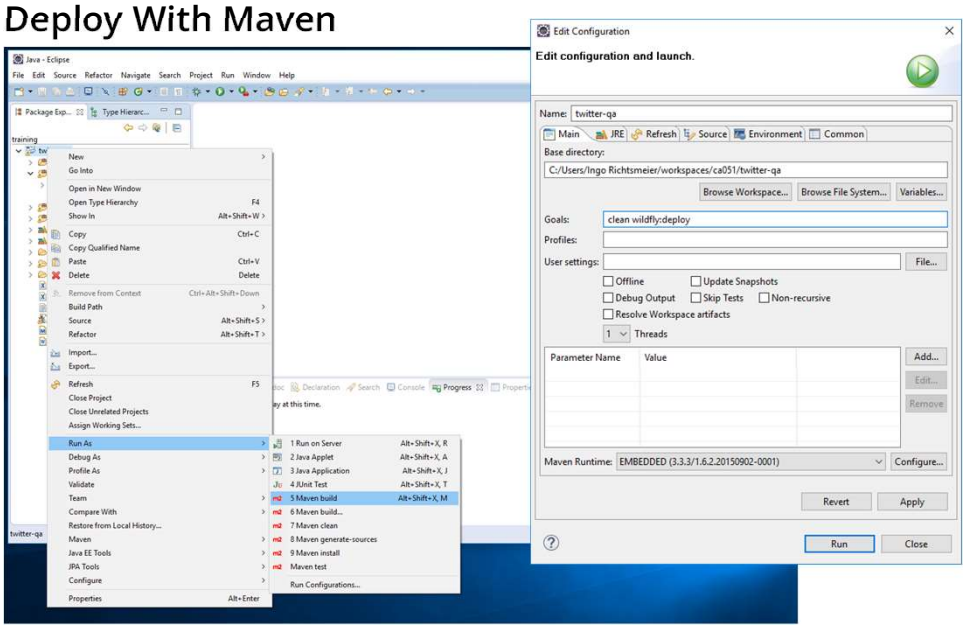
Scripts



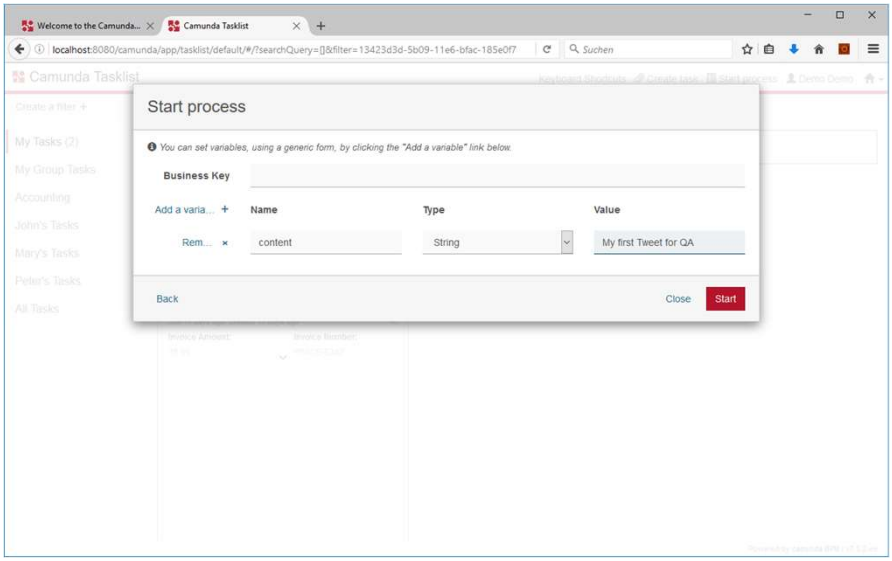
Setup Eclipse for Automatic Synchronization



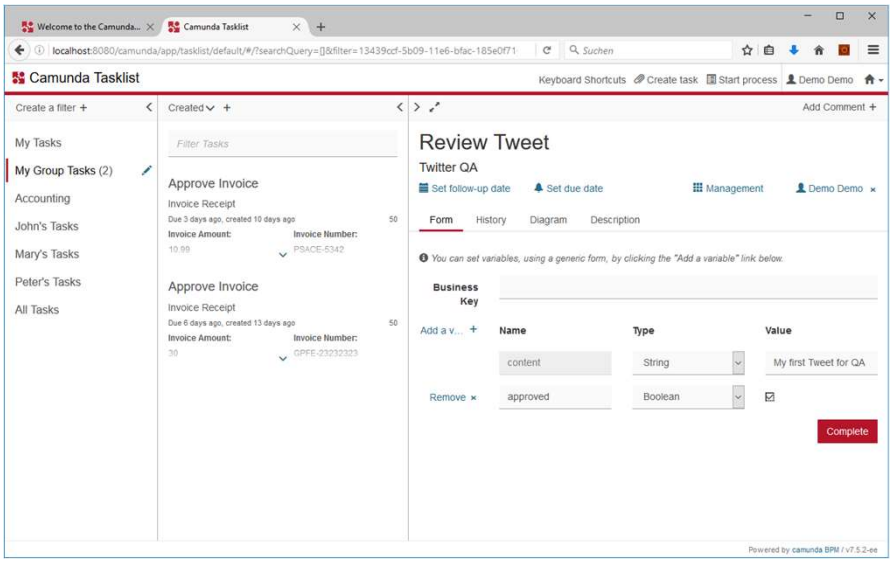
Deploy With Maven



Use Generic Task Forms

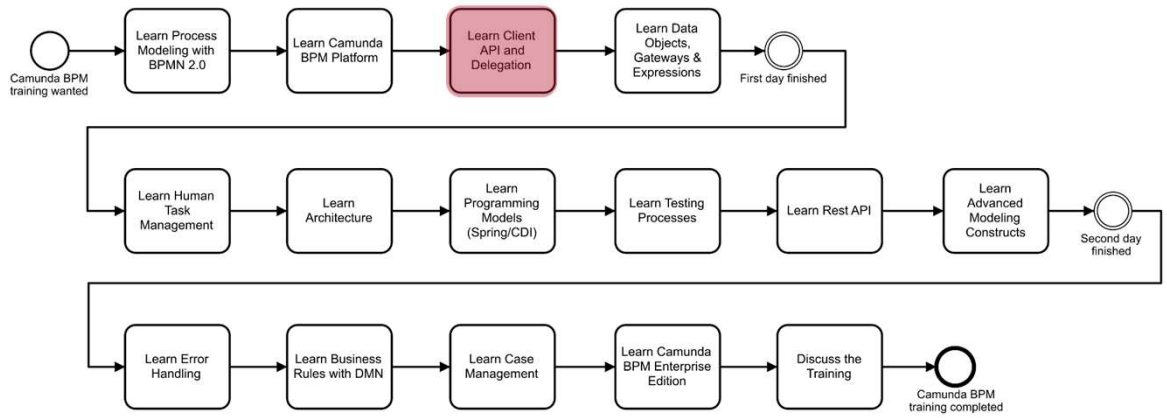


Complete Generic Task Forms



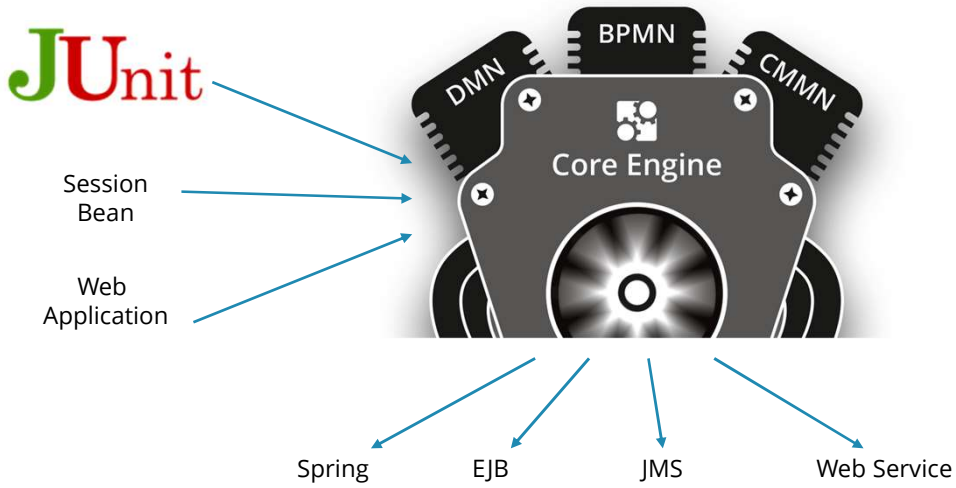
Exercise 2b

Follow the provided instructions for exercise 2b

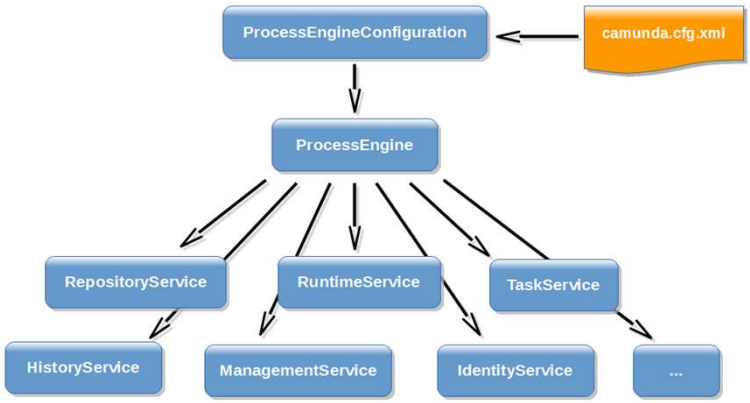

Camunda BPM training wanted
Learn Process Modeling with BPMN 2.0
Learn Camunda BPM Platform
Learn Client API and Delegation
Learn Data Objects, Gateways & Expressions
First day finished
Learn Human Task Management
Learn Architecture
Learn Programming Models (Spring/CDI)
Learn Testing Processes
Learn Rest API
Learn Advanced Modeling Constructs
Second day finished
Learn Error Handling
Learn Business Rules with DMN
Learn Case Management
Learn Camunda BPM Enterprise Edition
Discuss the Training
Camunda BPM training completed

BPM + Java

We Need a Way INTO and OUT of the Engine



Services, the Way IN



Overview Engine Services

Service	Description
Repository	Deployment, Process definitions
Runtime	Start of process instances, Variables, Event correlation
Task	Task lists, Task life cycle
Identity	Users, Groups
Form	Form key, Form property, Form
History	Logs, Statistics, KPIs
Management	Jobs, Statistics, Maintenance
Filter	Tasklist filters
ExternalTask	External task lifecycle
Case	Start of cases, Case execution lifecycle
Decision	Evaluate decisions
Authorization	Permissions

Java Client API

```
ProcessEngine processEngine = ProcessEngineConfiguration
    .createStandaloneProcessEngineConfiguration()
    .setJdbcUrl("jdbc:h2:./camunda-h2-dbs/process-engine;DB_CLOSE_DELAY=1000")
    .setDatabaseSchemaUpdate("true")
    .setJobExecutorActivate(true)
    .buildProcessEngine();

processEngine.getRepositoryService()
    .createDeployment()
    .addClasspathResource("vacation-request.bpmn")
    .deploy();

ProcessInstance processInstance = processEngine.getRuntimeService()
    .startProcessInstanceByKey("VacationRequestProcess");
```

Process Versioning

Process Definitions

ID_	KEY_	VERSION_	...
invoice:1:12ce6c9a-5b09-11e6-bfac-185e0f710ea8	invoice	1	
invoice:2:131a1ba0-5b09-11e6-bfac-185e0f710ea8	invoice	2	

Process Instances

ID_	PROC_DEF_ID_	ACT_ID_	...
1378dcae-5b09-11e6-bfac-185e0f710ea8	invoice:1:12ce6c9a-5b09-11e6-bfac-185e0f710ea8	reviewInvoice	
14b81440-5b09-11e6-bfac-185e0f710ea8	invoice:2:131a1ba0-5b09-11e6-bfac-185e0f710ea8	approveInvoice	

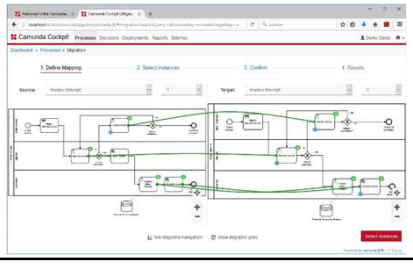
Start New Process Instances

Default Behavior:

1. New version upon each changed BPMN XML deployment
2. New process instances start with the latest version
3. Running process instances continue using the version they started with

```
ProcessInstance processInstance = processEngine.getRuntimeService()
    .startProcessInstanceByKey("VacationRequestProcess");
```

If needed, process instance migration is possible

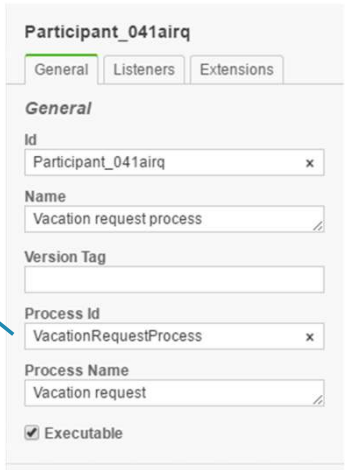


Key vs. ID vs. Name

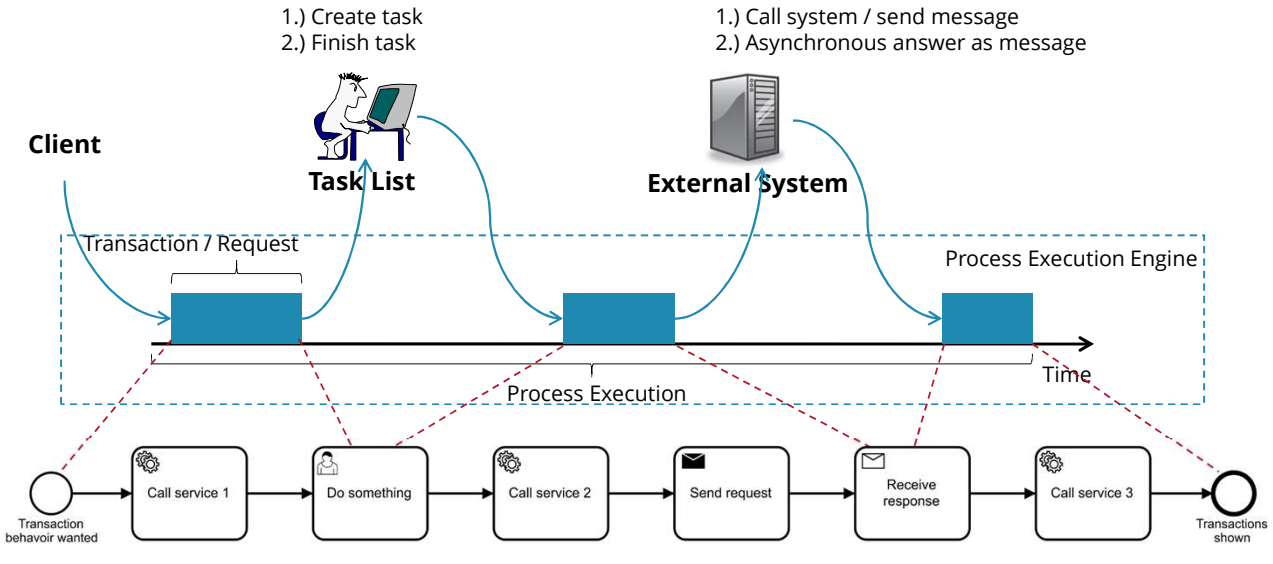
```
ProcessInstance processInstance = processEngine.getRuntimeService()
    .startProcessInstanceByKey("VacationRequestProcess");
```

- ProcessDefinitionKey: BPMN Process ID
- ProcessDefinitionName: BPMN Process Name
- ProcessDefinitionId: Generated Unique ID

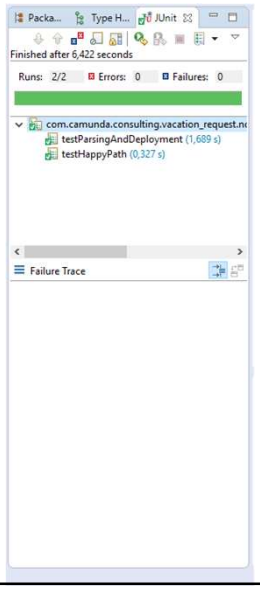
```
<bpmn2:process id="VacationRequestProcess"
    name="Vacation request"
    isExecutable="true">
```



Process Engine and Transactions



Unit Testing



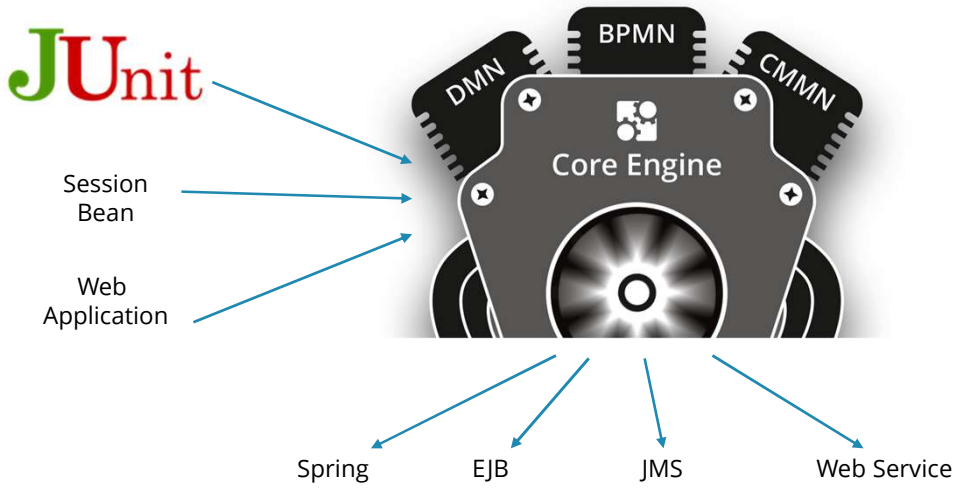

```
import static org.camunda.bpm.engine.test.assertions.ProcessEngineTests.*;

public class VacationRequestProcessTest {

  @Rule
  public ProcessEngineRule rule = new ProcessEngineRule();

  @Test
  @Deployment(resources = "vacation-request.bpmn")
  public void testHappyPath() {
    ProcessInstance processInstance = runtimeService()
        .startProcessInstanceByKey(PROCESS_DEFINITION_KEY);
    assertThat(processInstance).isWaitingAt("approveVacationUserTask");
    complete(task(), withVariables("approved", true));
    assertThat(processInstance).isWaitingAt("bookVacationUserTask");
    complete(task());
    assertThat(processInstance).isEnded();
  }
}
```

Shortcuts from camunda-bpm-assert

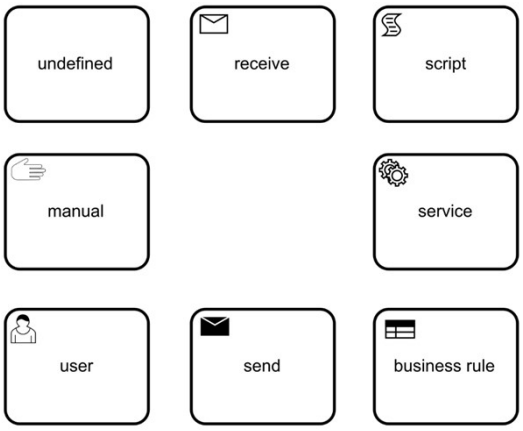
Exercise 3a

Follow the provided instructions for exercise 3a

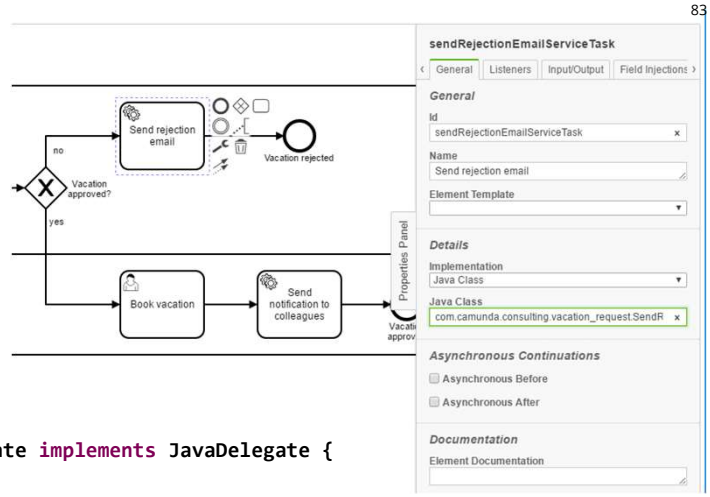
The Way OUT



BPMN 2.0 Task Types



Service Task With Java Class

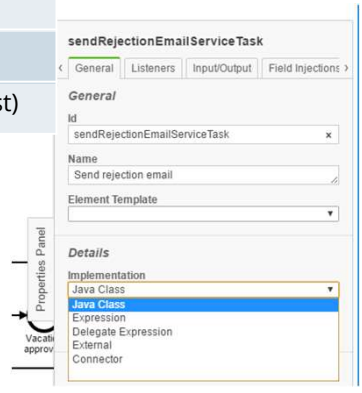



```
public class SendRejectionNotificationDelegate implements JavaDelegate {

  @Override
  public void execute(DelegateExecution execution) throws Exception {
    // Send the rejection to the employee...
    String reason = (String) execution.getVariable("rejectionReason");

  }

}
```


Other Options for Service Tasks

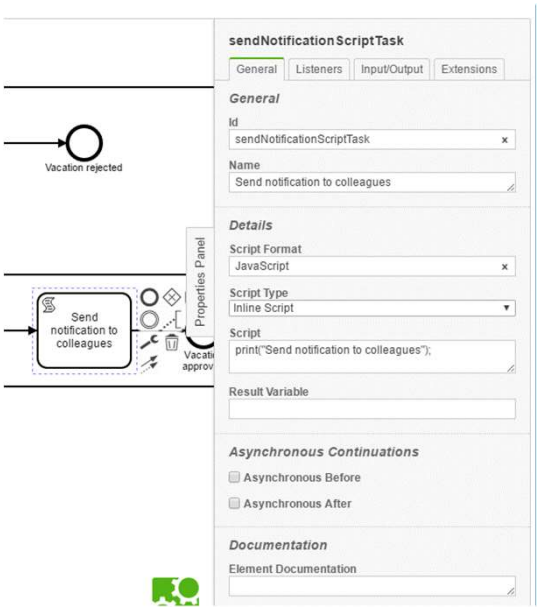
Implementation	Description
Expression	Resolve the implementation from an expression
Delegate Expression	Resolve the implementation from an expression, that implements `JavaDelegate`
External	Implement the service outside of the engine
Connector	Configure the call to a web service (soap or rest)

More details come later



Script Task

- Every JSR-223 compliant scripting engine can be used (we tested JS, Groovy, Jython, JRuby)
- Script sources can be externalized
- Templates can be used as script (for data transformation)
- Scripts get compiled and cached



Script Task Examples

```
<bpmn:scriptTask id="sendNotificationScriptTask" name="Send notification to colleagues"
scriptFormat="JavaScript">
  <bpmn:script><![CDATA[var message = requester + " will be on holiday from ";
message = message + from + " until " + until;
execution.setVariable("message", message);]]></bpmn:script>
</bpmn:scriptTask>
```

Process variables

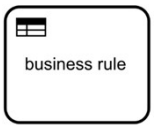
Special key word

```
<bpmn:scriptTask id="checkUniqueRSPIDScriptTask" name="Check unique RSP ID"
scriptFormat="JavaScript" camunda:resultVariable="resultCheckUniqueRspID"
camunda:resource="scripts/check_unique_rsp_id.js">
</bpmn:scriptTask>
```

Load script from class path

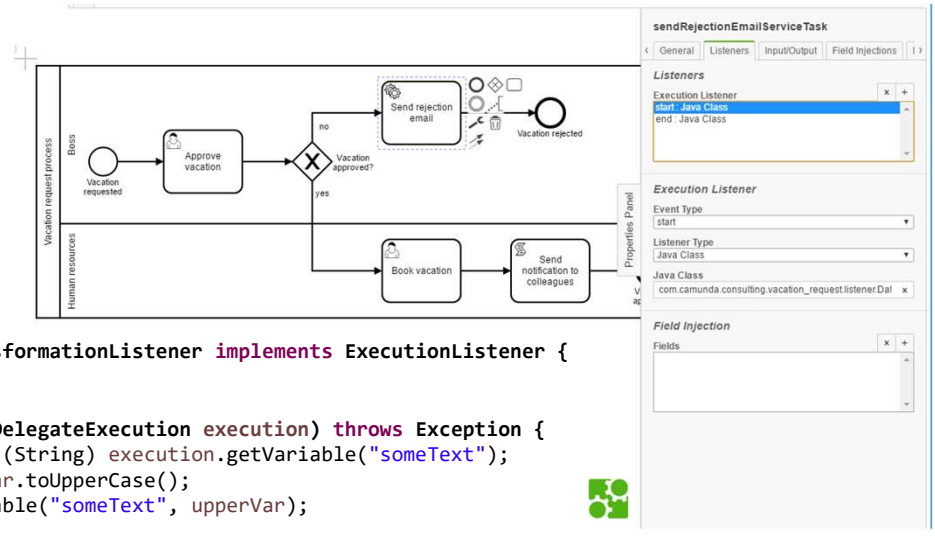
Store script result

Business Rule Task



- Behaves like a Service Task
- Link to a DMN Decision Table
- More Details later

Execution Listeners

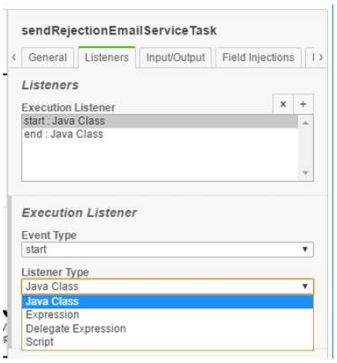



```
public class DataTransformationListener implements ExecutionListener {

  @Override
  public void notify(DelegateExecution execution) throws Exception {
    String upperVar = (String) execution.getVariable("someText");
    upperVar = upperVar.toUpperCase();
    execution.setVariable("someText", upperVar);
  }
}
```

Events for Execution Listeners

Level	Event
On process level	Start, end
On activity level	Start, end
On sequence flow	take

Listener Type
Java Class
Expression
Delegate Expression
Script



Java Delegates for Execution Listeners

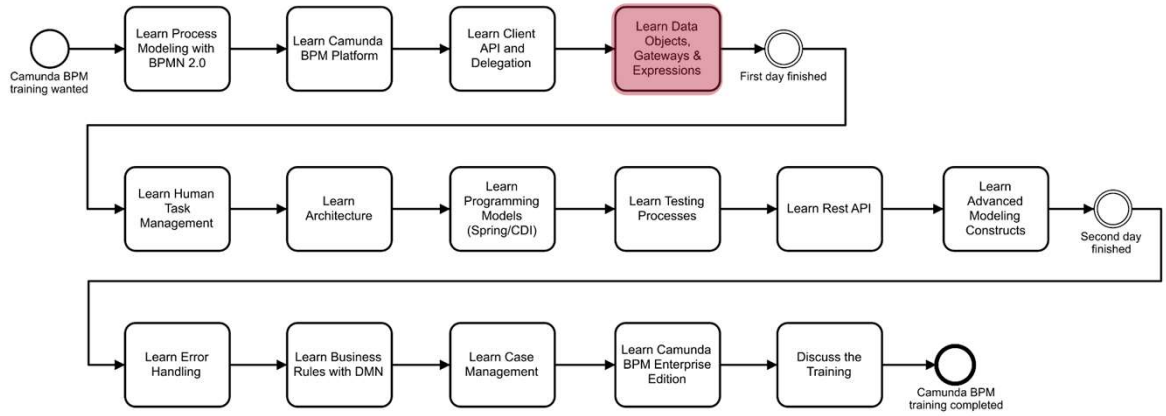
```
public class DataTransformationDelegate implements JavaDelegate {

  @Override
  public void execute(DelegateExecution execution) throws Exception {
    String upperVar = (String) execution.getVariable("someText");
    upperVar = upperVar.toUpperCase();
    execution.setVariable("someText", upperVar);
  }
```

or

```
}
public class DataTransformationListener implements ExecutionListener {

  @Override
  public void notify(DelegateExecution execution) throws Exception {
    String upperVar = (String) execution.getVariable("someText");
    upperVar = upperVar.toUpperCase();
    execution.setVariable("someText", upperVar);
  }

}
```

Exercise 3b

Follow the provided
instructions for exercise 3b



Accessing Process Variables

Client Code:

```
Map<String, Object> variables = new HashMap<String, Object>();
variables.put("employee", "Jim");
runtimeService().startProcessInstanceByKey("VacationRequestProcess", variables);
```

Delegate Code:

```
public class SendRejectionNotificationDelegate implements JavaDelegate {

  @Override
  public void execute(DelegateExecution execution) throws Exception {
    // Send the rejection to the employee...
    String reason = (String) execution.getVariable("rejectionReason");
    String employee = (String) execution.getVariable("employee");

    // create a message to the employee
    execution.setVariable("message", "Some message to " + employee + " about " + reason);
  }
}
```


Process Variables

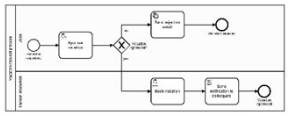
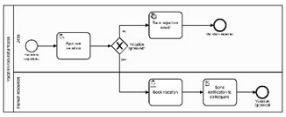
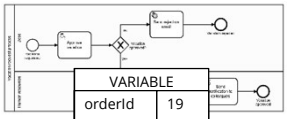
- Are not modeled in BPMN 2.0
- Behave like a `java.util.Map`

- Are persisted with the process instance

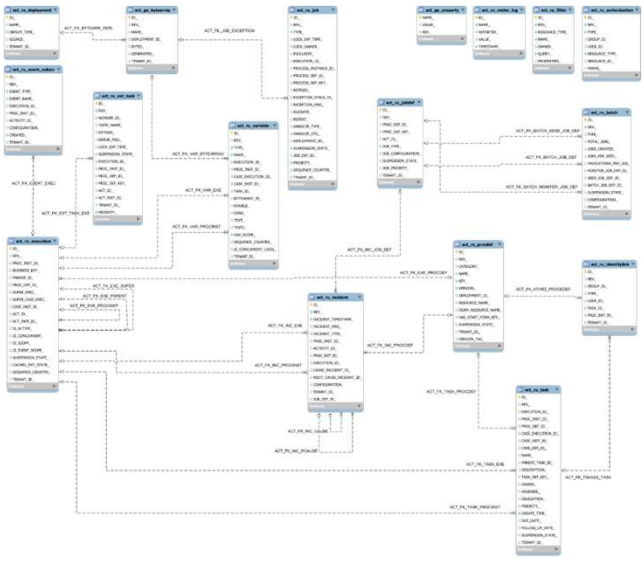
- Recommended: Primitive types, Date & String
- Objects can be serialized as XML or JSON
- Default fallback: Serializable Java Objects (not recommended!)

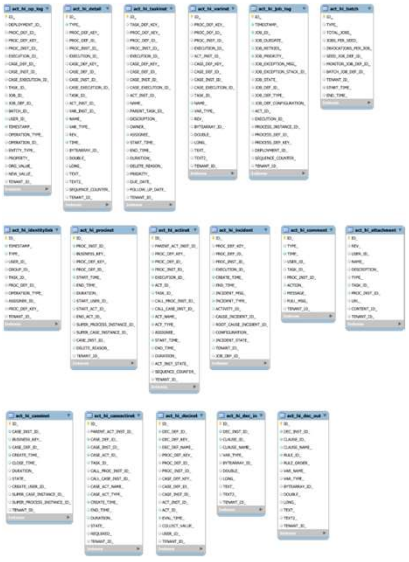
- Serialization can be exchanged: https://docs.camunda.org/manual/latest/user-guide/data-formats/data-formats-in-processes/#extending-serialization

Storage

Process Variable	JPA Entity	External Data (Reference and Service Call)
		
VARIABLE var1 value1 var2 18	VARIABLE orderId 19 ... ... ORDER 19 ... ... 20	VARIABLE orderId 19 ... ... Scope „Process“ ORDER 19 ... ... 20 ... ...
Scope „Process“	Scope „Process“	Scope „Data“
Process Engine Tables	Process Engine Tables Entity Tables	Process Engine Tables Entity Tables

Database Schema

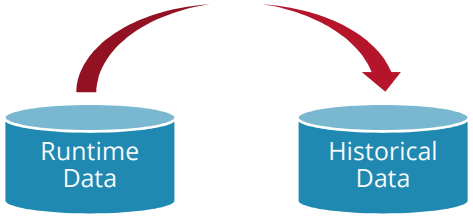




https://docs.camunda.org/manual/latest/user-guide/process-engine/database/#database-schema

History Configuration

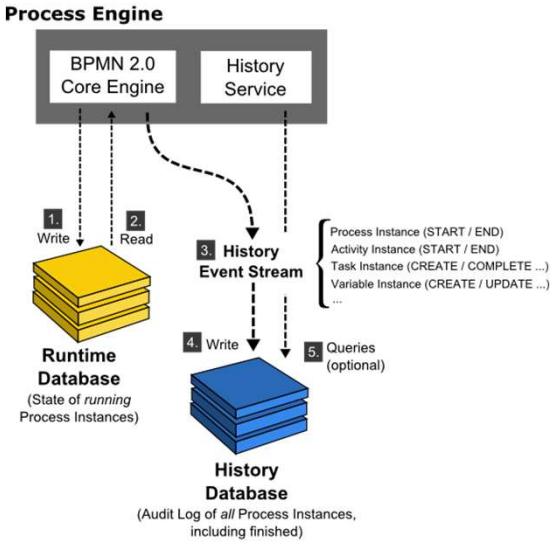
finished process instance are removed from runtime tables



Configuration options:

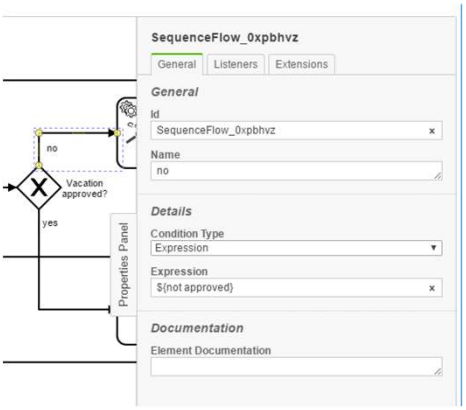
- None: no history at all
- Activity: process instances and trail of executed BPMN activities
- Audit: + last value of every process variable
  + submitted form properties (works only with forms)
- Full: + variable updates
- Custom: e.g. own filter for events of particular elements or variables

Asynchronous History



Expressions

- Work on data objects
- May work on Spring/CDI beans
- Used in:
  - Java service tasks
  - Event listeners
  - Conditional sequence flows
- Can use attributes or methods



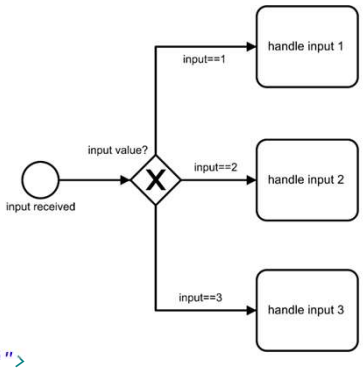

```
${myVar}
${myBean.myProperty}

${printer.print()}
${myBean.addNewOrder('orderName')}
${myBean.doSomething(myVar, execution)}
```


Unified Expression Language (UEL)

from
http://java.sun.com/javaee/5/docs/tutorial/doc/bnahq.html#bnain

EL Expression	Result
${1 > (4/2)}	false
${4.0 >= 3}	true
${100.0 == 100}	true
${(10*10) ne 100}	false
${'a' < 'b'}	true
${'hip' gt 'hit'}	false
${4 > 3}	true
${1.2E4 + 1.4}	12001.4
${3 div 4}	0.75
${10 mod 4}	2
${!empty param.Add}	False if the request parameter named Add is null or an empty string.
${pageContext.request.contextPath}	The context path.
${sessionScope.cart.numberOfItems}	The value of the numberOfItems property of the session-scoped attribute named cart.
${param['mycom.productId']}	The value of the request parameter named mycom.productId.
${header["host"]}	The host.
${departments[deptName]}	The value of the entry named deptName in the departments map.
${requestScope['javax.servlet.forward.servlet_path']}	The value of the request-scoped attribute named javax.servlet.forward.servlet_path.
#{customer.lName}	Gets the value of the property lName from the customer bean during an initial request. Sets the value of lName during a postback

Exclusive Gateway (Decision)




```
<bpmn:exclusiveGateway id="exclusiveGateway" name="input value?">
</bpmn:exclusiveGateway>
<bpmn:sequenceFlow id="flow3" name="input==2" sourceRef="exclusiveGateway" targetRef="handleInput2Task">
  <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${input==2}</bpmn:conditionExpression>
</bpmn:sequenceFlow>

<bpmn:sequenceFlow id="flow2" name="input==1" sourceRef="exclusiveGateway" targetRef="handleInput1Task">
  <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${input==1}</bpmn:conditionExpression>
</bpmn:sequenceFlow>

<bpmn:sequenceFlow id="flow4" name="input==3" sourceRef="exclusiveGateway" targetRef="handleInput3Task">
  <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${input==3}</bpmn:conditionExpression>
</bpmn:sequenceFlow>
```


Service Task With UEL Expression


```
<bpmn:serviceTask id="javaService"
       name="My Java Service Task"
       camunda:expression="${printer.printMessage()}" />

<bpmn:serviceTask id="javaService"
       name="My Java Service Task"
       camunda:expression="${printer.printMessage(execution, myVar)}" />
```

Camunda Spin Project

Motivation

- Handle serialized formats such as JSON or XML while interacting with external systems.
- Process such data like parsing, manipulating or mapping from/to Java objects.

Features

- Provide data format functionality
- Plugged into the engine
- Wrapper for processing data formats like XML and JSON
- Integrated with the engine's data handling functionality
- Designed to be extensible

https://docs.camunda.org/manual/latest/user-guide/data-formats/

https://docs.camunda.org/manual/latest/reference/spin/

Spin Example

```
<?xml version="1.0" encoding="UTF-8"?>
<customer xmlns=http://camunda.org/example name="Jonny">
  <address>
    <street>12 High Street</street>
    <postcode>1234</postcode>
  </address>
</customer>
```

Imagine a response from a web service call

Maybe a condition on a sequence flow

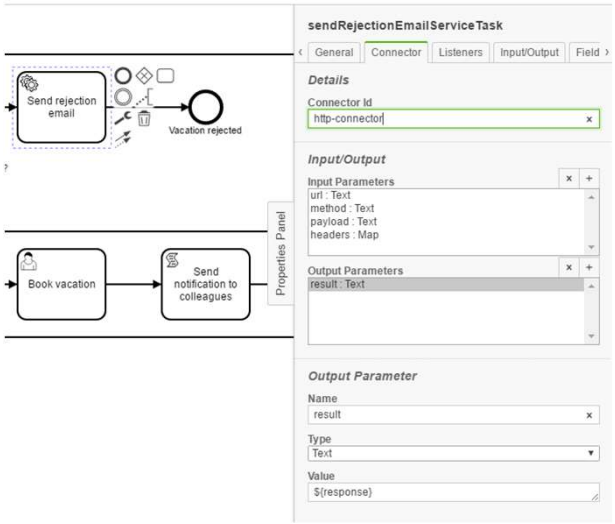
Access in an expression:

```
${XML(customer).xPath("/customer/address/postcode").element().textContent() == "1234"}
```

Connectors: Configure Service Task Without Java Code

From the Java developer view:

- No class needed
- All configuration in the BPMN2.0 File
- Scripting for dynamic service calls
- May lead to a lot of different resources
- May be harder to manage than a `JavaDelegate`
- Difficult to mock during JUnit tests



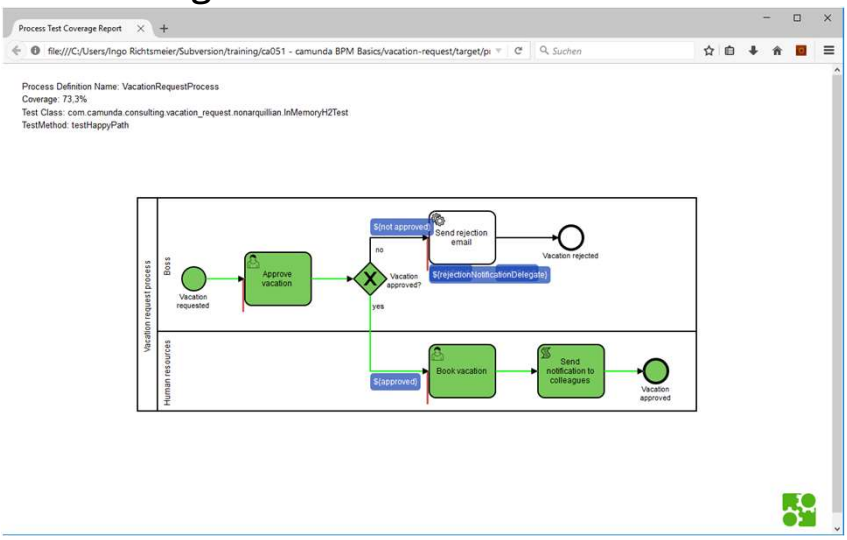
Connectors: Configure Service Task in BPMN2.0 File

From the Script developer view:

- Use expressions or scripts to calculate the parameters of the service call
- Inspect the result with Camunda Spin
- Scripts run in the JVM of the process engine
- Easy access to process variables from the JVM
- Complete control of the service call possible

```
<bpmn:serviceTask id="ServiceTask_1dwr4ge" name="Check mandatory field
values" camunda:asyncBefore="true">
  <bpmn:extensionElements>
    <camunda:connector>
      <camunda:inputOutput>
        <camunda:inputParameter name="url">http://localhost:3000/check
mandatory-fields</camunda:inputParameter>
        <camunda:inputParameter name="method">POST</camunda:inputParam
        <camunda:inputParameter name="headers">
          <camunda:map>
            <camunda:entry key="Content-
Type">application/json</camunda:entry>
          </camunda:map>
        </camunda:inputParameter>
        <camunda:inputParameter name="payload">
          <camunda:script scriptFormat="javascript"
resource="scripts/generate-service-payload.js" />
        </camunda:inputParameter>
        <camunda:outputParameter name="resultCheckMandatoryFields">
          <camunda:script
scriptFormat="javaScript">S(response);</camunda:script>
        </camunda:outputParameter>
      </camunda:inputOutput>
      <camunda:connectorId>http-connector</camunda:connectorId>
    </camunda:connector>
  </bpmn:extensionElements>
</bpmn:serviceTask>
```

Process Test Coverage



https://github.com/camunda/camunda-bpm-process-test-coverage

Exercise 4

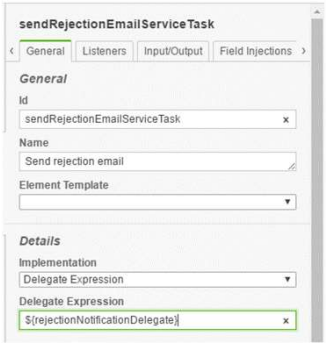
Follow the provided instructions for exercise 4

Improve JUnit with Simple Mocks

1. Make Delegate a Bean with `@Named`

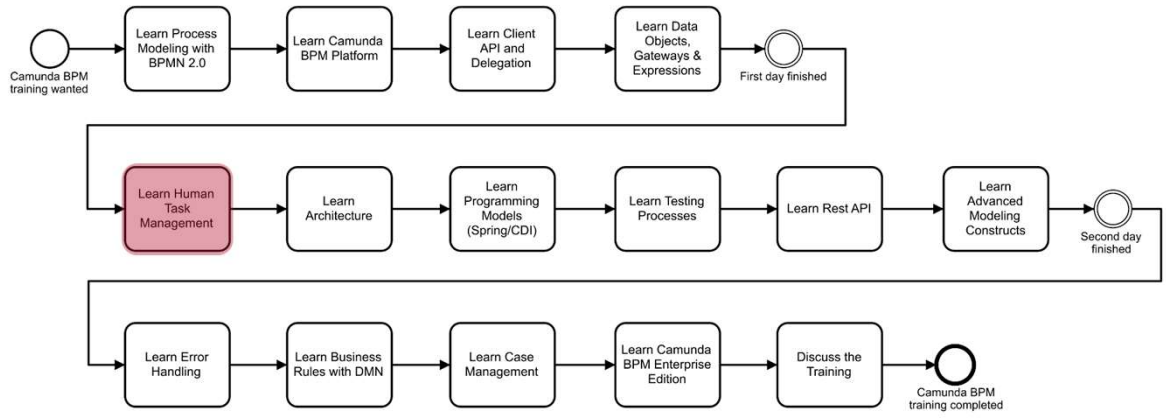
```
@Named("rejectionNotificationDelegate")
public class SendRejectionNotificationDelegate implements JavaDelegate {

  @Override
  public void execute(DelegateExecution execution) throws Exception {
    // Send the rejection to the employee...
```

2. Use Indirection with DelegateExpression in bpmn process

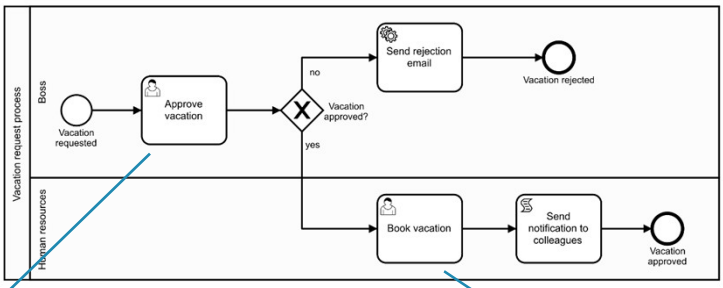


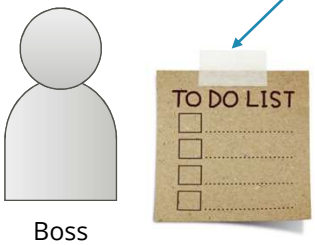
3. Use `Mocks.register("bean", new MockDelegate())` in the test method

```
@Test
@Deployment(resources = "vacation-request.bpmn")
public void testRejectionVacation() {
  Mocks.register("rejectionNotificationDelegate",
      new LoggerDelegate());
  ...
}
```

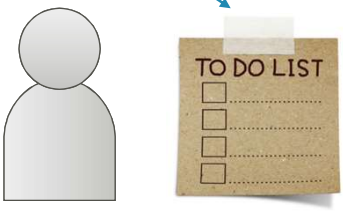


Task Management



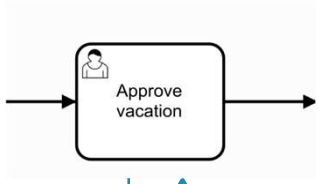


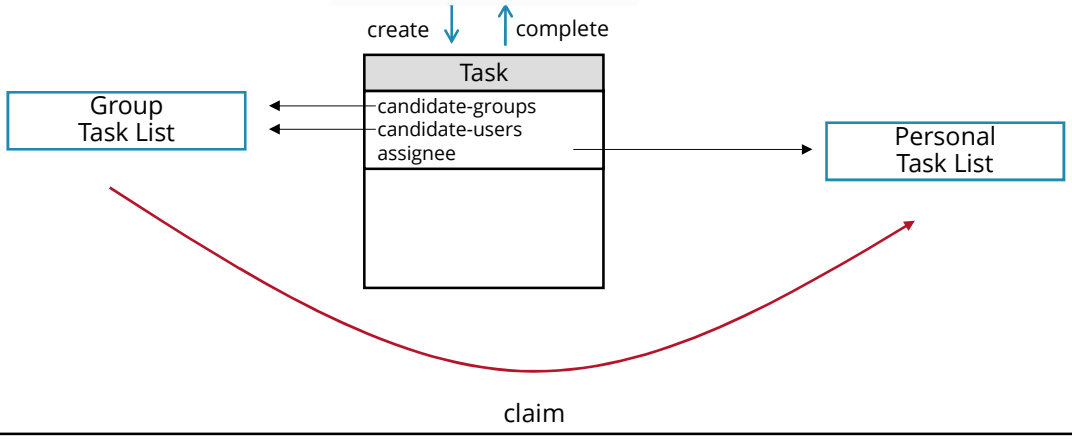
Boss



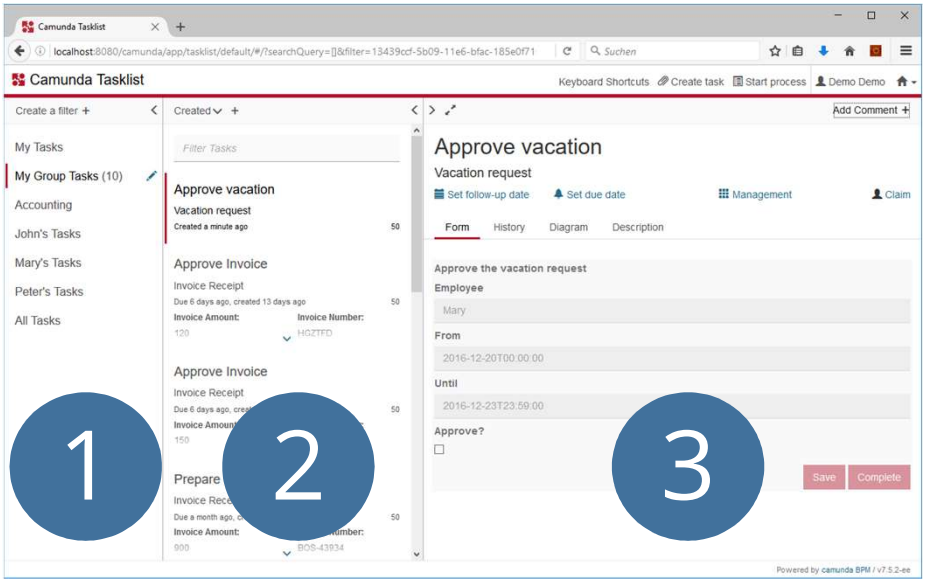
Human Resources

Task Management

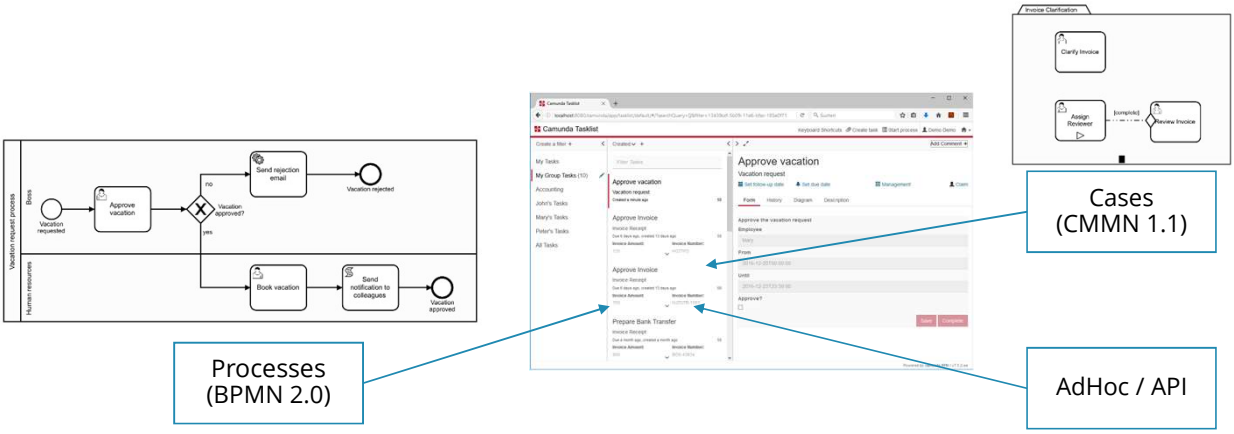




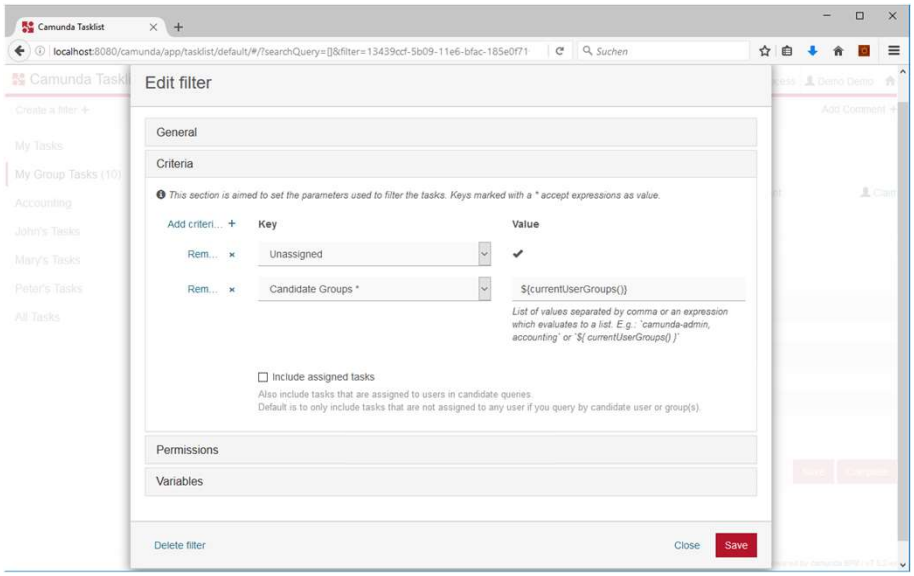
Tasklist in Camunda BPM



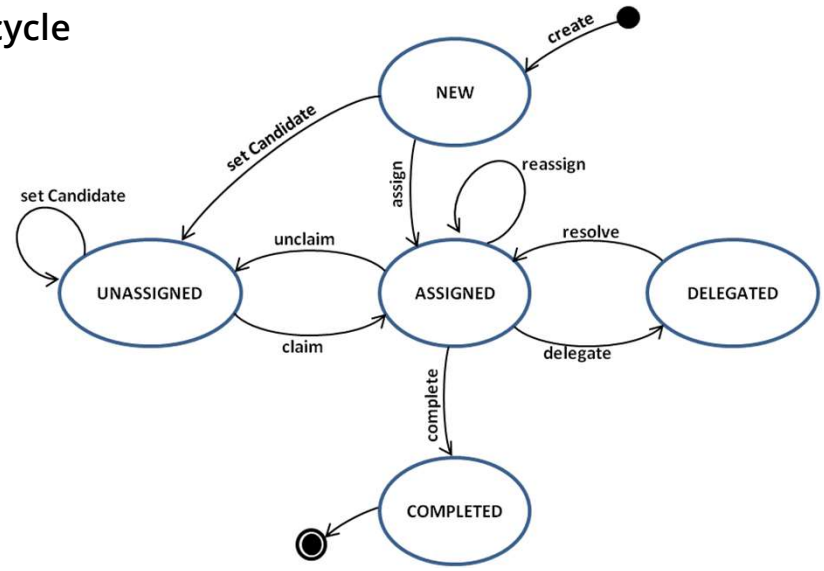
Tasks From Different Sources



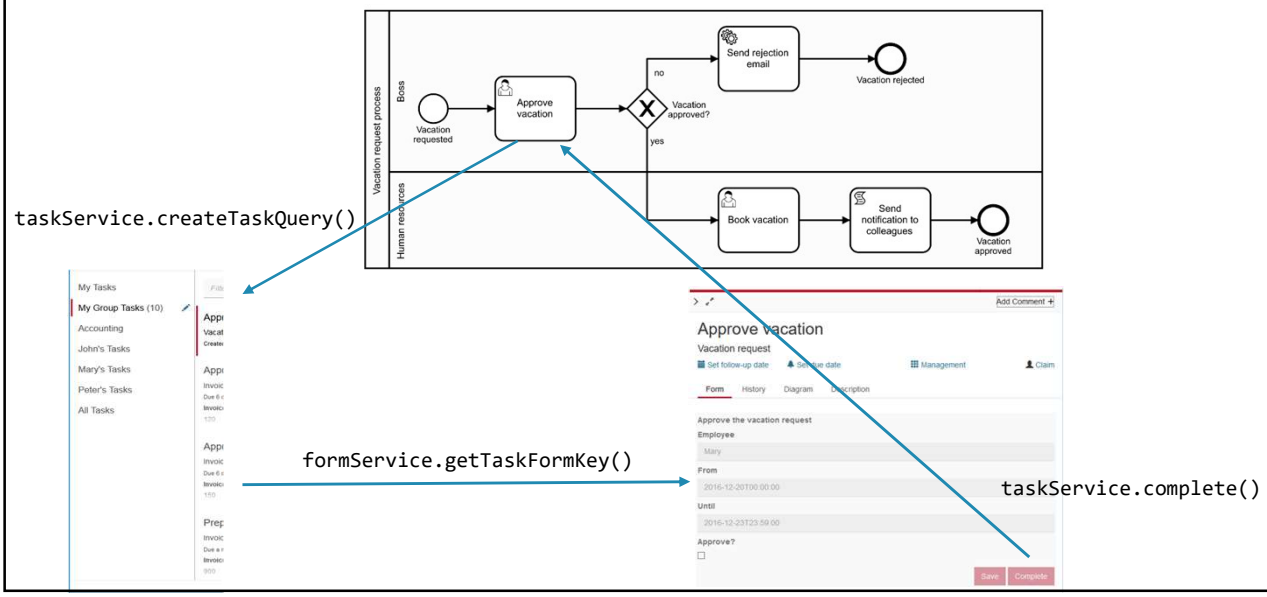
Groups & Personal Task List (Via Filters)



Task Lifecycle



Task Management API



Personal and Group Task Lists

```
TaskService taskService = processEngine.getTaskService();

List<Task> personalTaskList = taskService.createTaskQuery()
       .taskAssignee("demo").list();


List<Task> groupTaskList = taskService.createTaskQuery()
       .taskCandidateUser("demo").list();

groupTaskList.addAll(taskService.createTaskQuery()
       .taskCandidateGroup("marketing").list());
```

Task Query & Completion

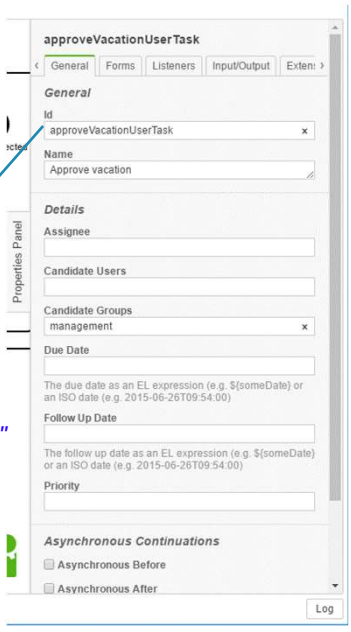
```
TaskService taskService = processEngine.getTaskService();

List<Task> personalTaskList = taskService.createTaskQuery()
       .taskAssignee("demo").list();

Task task = personalTaskList.get(0);

Map<String,Object> variables = new HashMap<String, Object>();
variables.put("approved", false);

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


Key vs. ID vs. Name

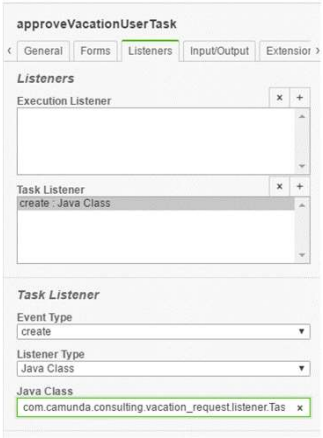
```
taskService.complete(task.getId(), variables);
```

- `task.getId()`: Task Instance ID
- `task.getTaskDefinitionKey()`: BPMN User Task ID
- `task.getName()`: BPMN User Task Name

```
<bpmn:userTask id="approveVacationUserTask" name="Approve vacation"
camunda:candidateGroups="management">
</bpmn:userTask>
```



Task Listener



Typical Use Cases:

- Send notification e-mail
- Task modifications, e.g. priority, due date, display name
- Vacation substitute
- Create task in third-party application

Available events:

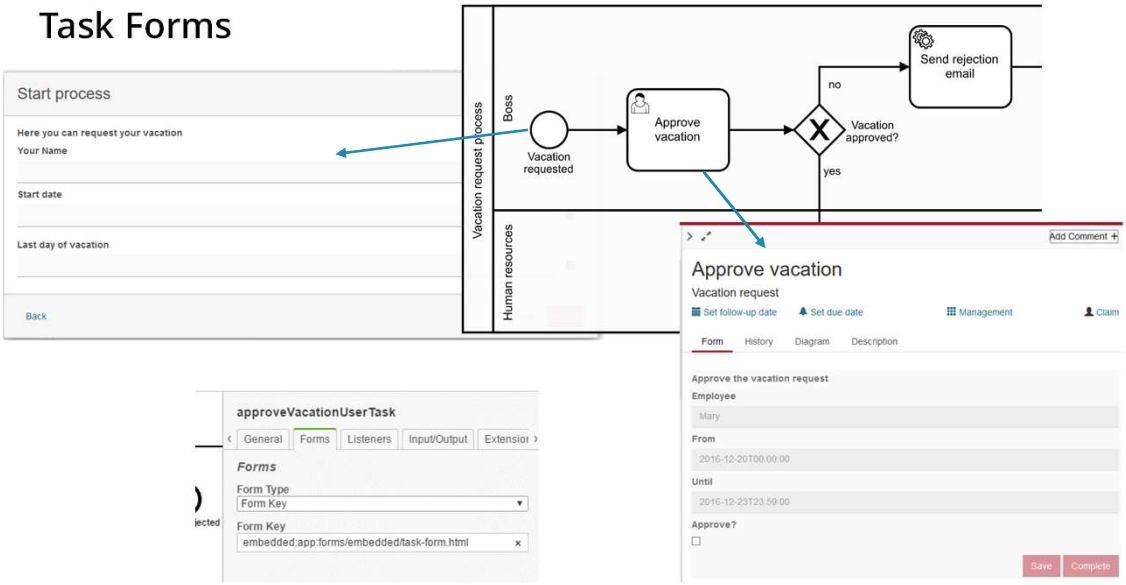
```
create
assignment
complete
delete
```

```java
public class TaskCreationListener implements TaskListener {

  @Override
  public void notify(DelegateTask delegateTask) {
    // send a mail to manager
  }
}
```


Exercise 5

Follow the provided instructions for exercise 5

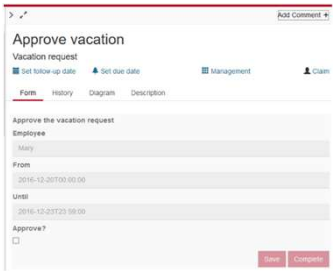
Task Forms



Task Forms in Camunda Tasklist

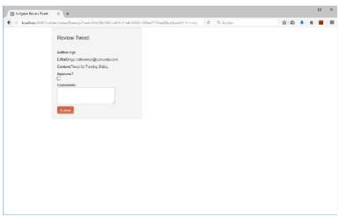
Generic form

Embedded form



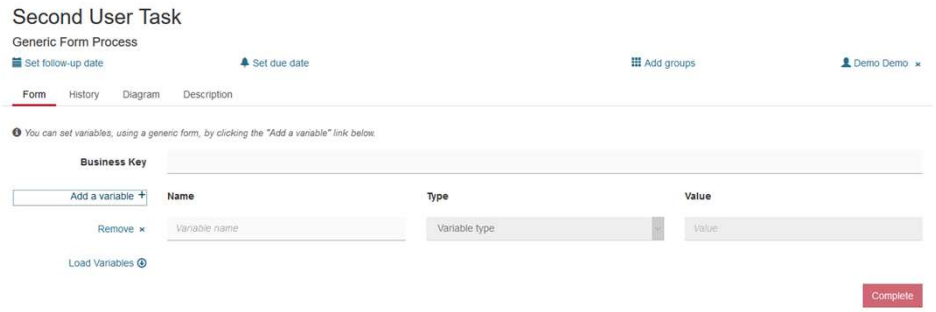
Generated form

External form



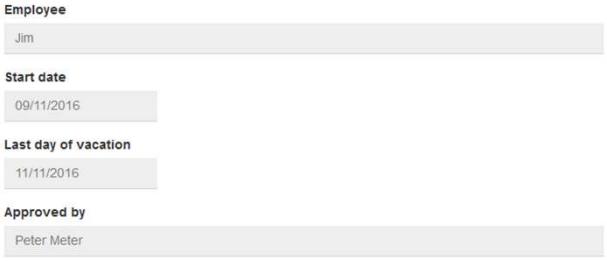
Generic Task Forms

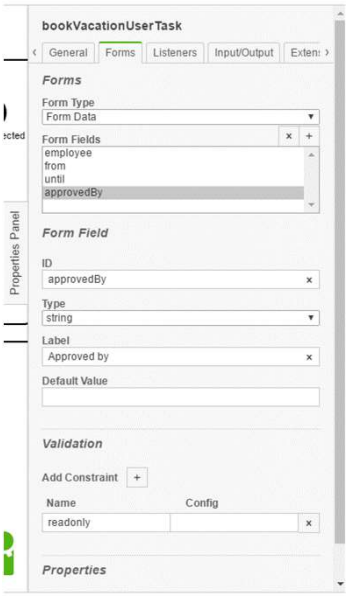
- `formKey=""`
- Completely generic, shows all process variables after load
- Add variables manually on demand
- Edit existing variables



Generated Task Forms (From Form Fields)

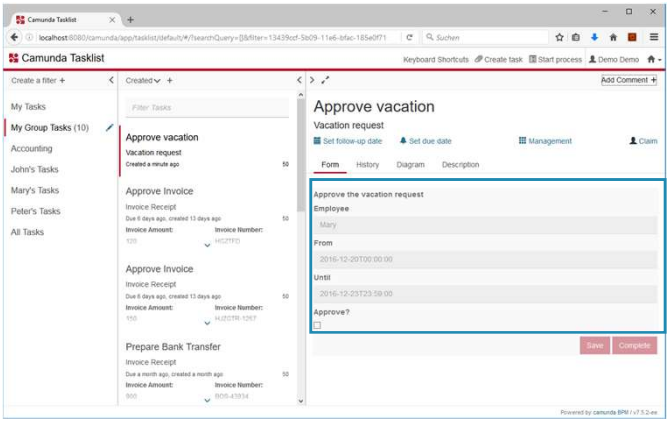
- Form Data Metadata provided by BPMN 2.0 XML
- Rendered in task list





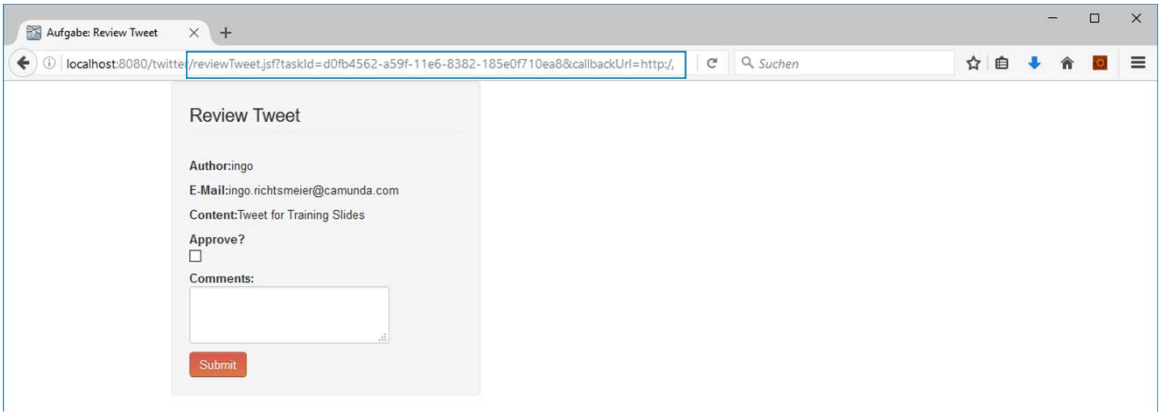
Embedded Task Forms

- `formKey="embedded:app:approve-vacation.html"`
- HTML-Form provided by Process Application (HTML-File)
- Mix HTML and Angular-JS JavaScript
- Rendered in task list



External Task Form

- `formKey="app:reviewTweet.jsf“`
- Params `taskId` and `callbackURL` needed



Embedded Task Forms Form SDK

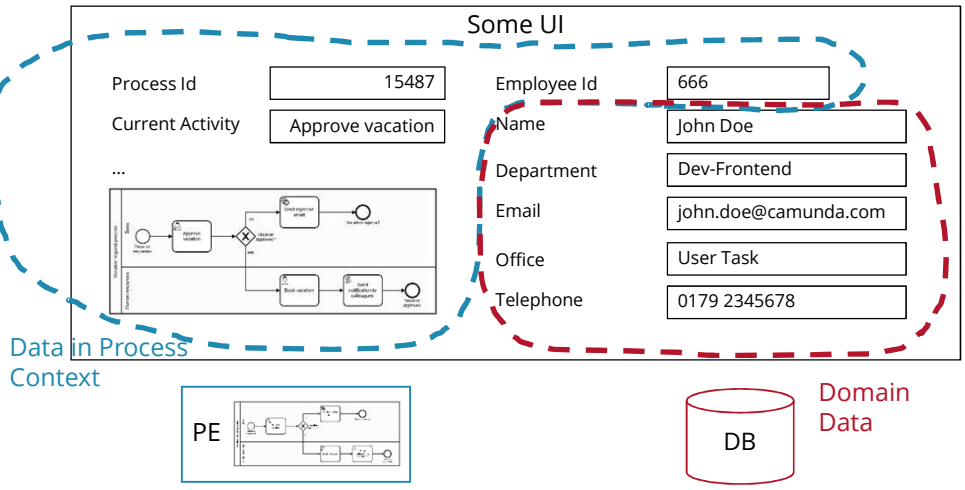
- *Form handling*: Attach to a form existing in the DOM or load a form from a URL.
- *Variable handling*: Load and submit variables used in the form.
- *Script handling*: Execute custom JavaScript in Forms
- *Angular JS Integration*: The Forms SDK optionally integrates with AngularJS to take advantage of AngularJS form validation and other AngularJS goodies.

`cam-variable-name`: Bind input to the model
`cam-variable-type`: Type of process variable

Form templates are generated from the archetypes

https://docs.camunda.org/manual/latest/reference/embedded-forms/

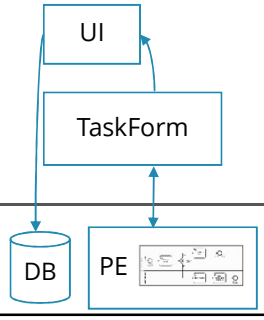
Typical: Process Data and External Data Are Mixed



Alternatives to Call Data Service

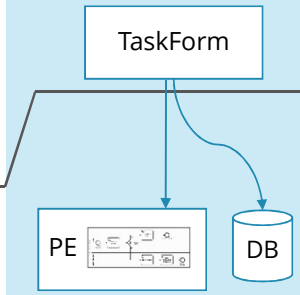
Alternative 1: Separate UI

- Use existing UI
- Started manually or via link in the Task Form
- Completely independent



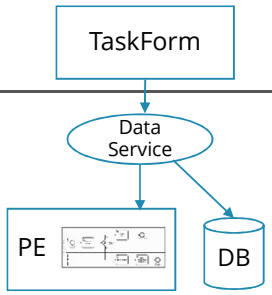
Alternative 2: UI Integration

- The UI calls the Data Service and the Process Service
- No TX



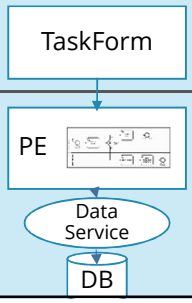
Alternative 3: Composed Service

- The UI calls a combined service
- This service calls the Data Service and the Process Service
- TX possible



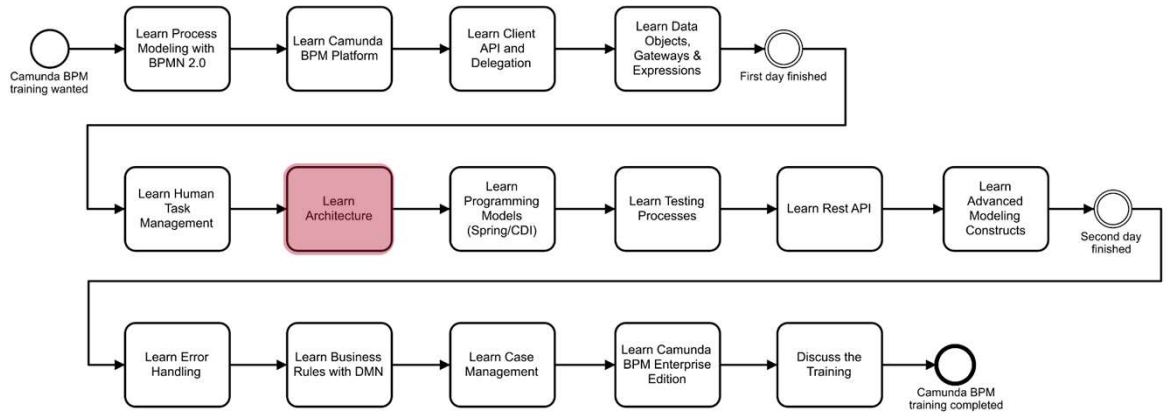
Alternative 4: Process Service

- The UI calls the Process Service
- The Process calls the Data Service
- Full control, but data has to be transported through the process

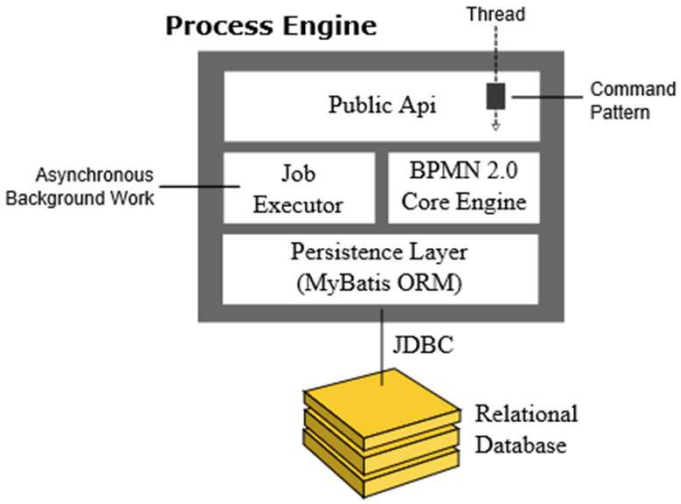


Exercise 6

Follow the provided instructions for exercise 6



Process Engine Architecture



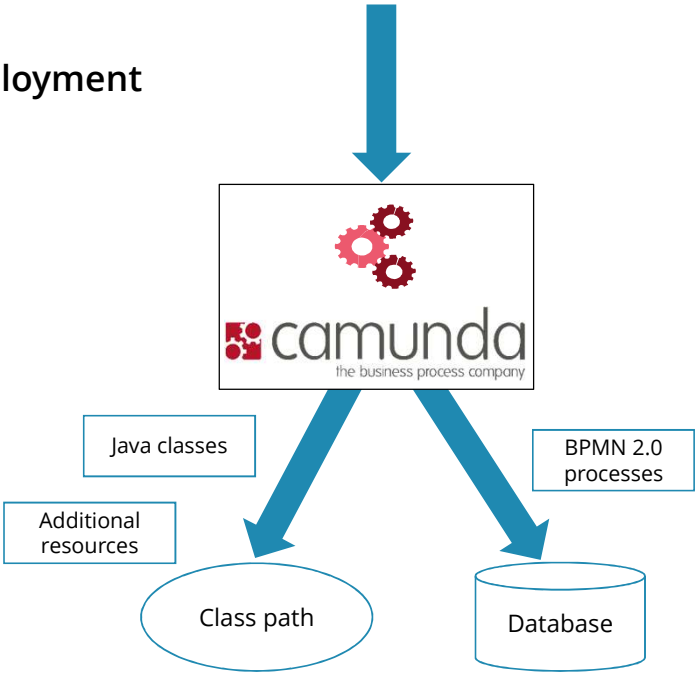
Starting a Process Instance via REST

1. HTTP POST to `/engine-rest/process-definition/key/my-process/start`
2. `ProcessEngine.getRuntimeService()`
3. `RuntimeServiceImpl.startProcessInstanceByKey(…)`
4. `commandExecutor.execute(new StartProcessInstanceCmd(…))`
5. CommandInterceptors (e.g. TransactionInterceptor)
6. `StartProcessInstanceCmd.execute(commandContext)`
7. Internal API composed of so-called Managers:
   1. `DeploymentCache.findDeployedLatestProcessDefinitionByKey(…)`
   2. `processDefinition.createProcessInstance(…)`
   3. `processInstance.start(variables)`
8. AtomicOperations (e.g. start process, execute activity, take transition)
9. ActivityBehaviours implement actual BPMN execution semantics
10. When no token can move further, CommandInterceptors store state changes in DB and commit transaction
11. HTTP response is sent

Start an Engine in Pure Java

```
ProcessEngine processEngine =
ProcessEngineConfiguration.createStandaloneInMemProcessEngineConfiguration()
  .setDatabaseSchemaUpdate(ProcessEngineConfiguration.DB_SCHEMA_UPDATE_FALSE)
  .setJdbcUrl("jdbc:h2:mem:my-own-db;DB_CLOSE_DELAY=1000")
  .setJobExecutorActivate("true")
  .buildProcessEngine();

processEngine.getRepositoryService().createDeployment()
       .addClasspathResource("jax.bpmn").deploy();
```

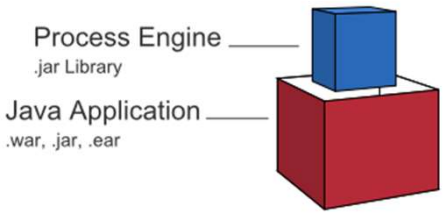
Process Deployment



Deploying Programmatically With Java

```
repositoryService.createDeployment()
      .addClasspathResource("process.bpmn")
      .enableDuplicateFiltering(true)
      .deploy();
```

Embedded Process Engine



Various environments possible:

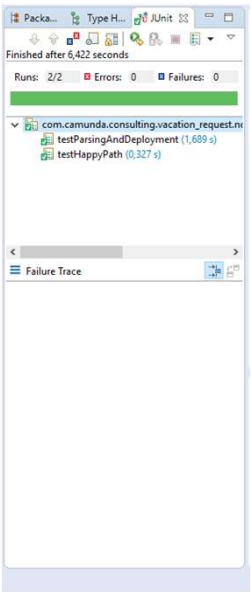
most popular

Pure Java | Spring | CDI | Blueprint

Basic question: Who controls the lifecycle of the engine (startup / shutdown)?

Example: Unit Test

```
import static org.camunda.bpm.engine.test.assertions.ProcessEngineTests.*;

public class VacationRequestProcessTest {

  @Rule
  public ProcessEngineRule rule = new ProcessEngineRule();

  @Test
  @Deployment(resources = "vacation-request.bpmn")
  public void testHappyPath() {
    ProcessInstance processInstance = runtimeService()
        .startProcessInstanceByKey(PROCESS_DEFINITION_KEY);
    assertThat(processInstance).isWaitingAt("approveVacationUserTask");
    complete(task(), withVariables("approved", true));
    assertThat(processInstance).isWaitingAt("bookVacationUserTask");
    complete(task());
    assertThat(processInstance).isEnded();
  }
}
```

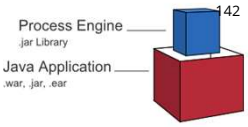


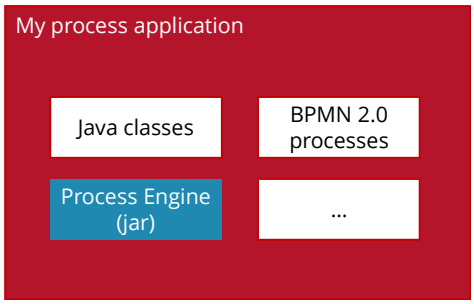
Example: Spring

```
ClassPathXmlApplicationContext applicationContext =
    new ClassPathXmlApplicationContext("/spring-context.xml");
ProcessEngine processEngine = (ProcessEngine) applicationContext.getBean("processEngine");

processEngine.getRepositoryService().createDeployment()
    .addClasspathResource("vacation-request.bpmn").deploy();

processEngine.getRuntimeService().startProcessInstanceByKey("vacation-request-process");

Task task = processEngine.getTaskService().createTaskQuery().taskAssignee("john").singleResult();

//Simulate task completion
processEngine.getTaskService().complete(task.getId());
```

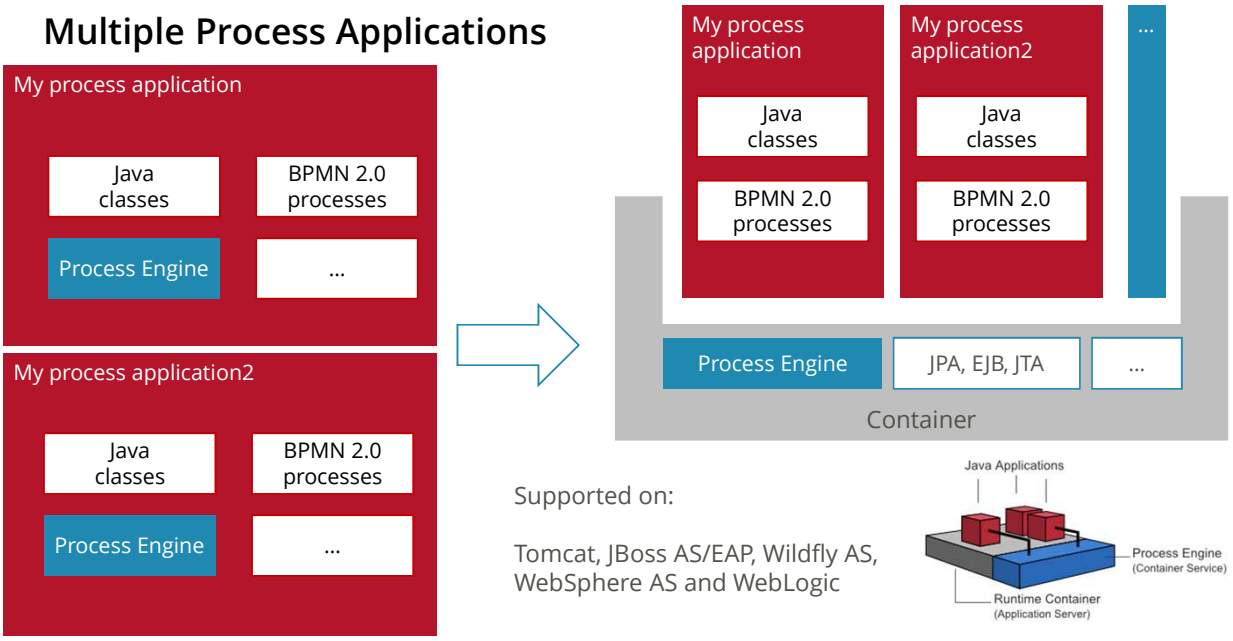
Sample Spring Configuration:
https://docs.camunda.org/get-started/spring/embedded-process-engine/

One Process Application





Multiple Process Applications



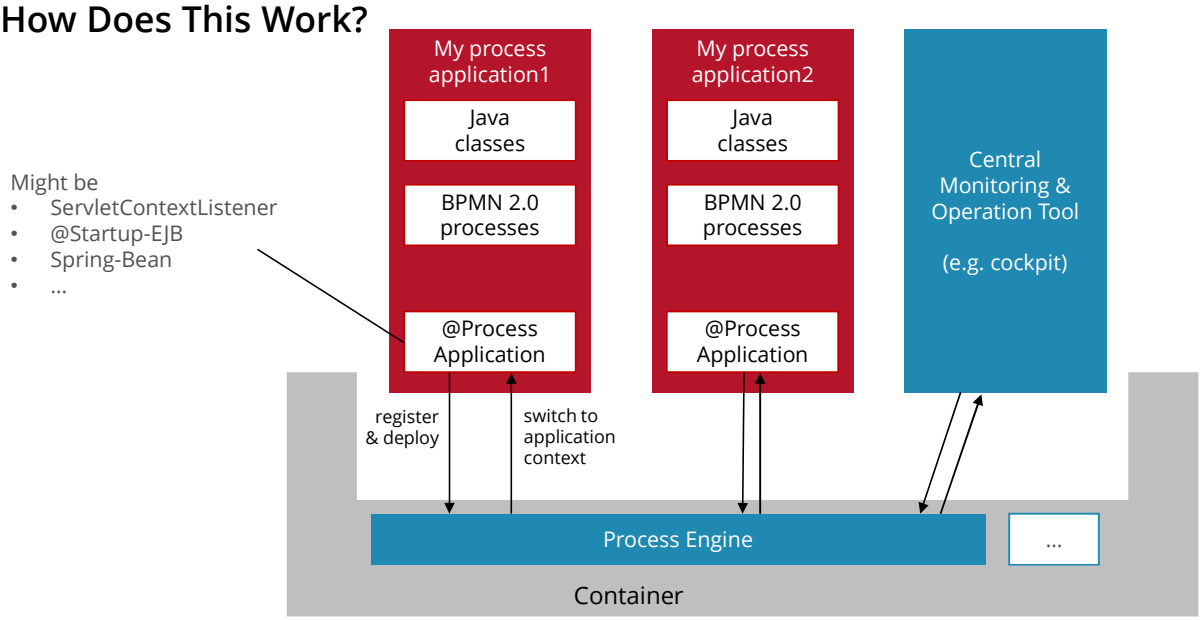
Supported on:

Tomcat, JBoss AS/EAP, Wildfly AS, WebSphere AS and WebLogic

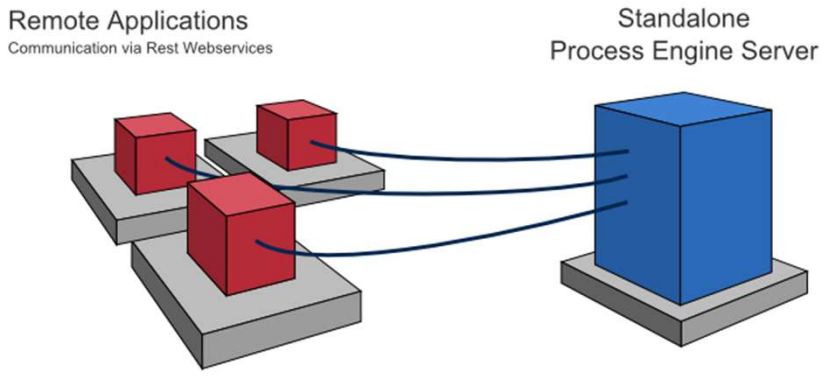
Camunda Process Application: processes.xml

```
<?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>
```

How Does This Work?

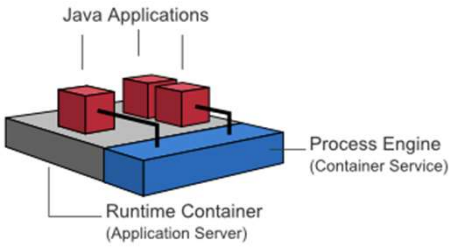


Standalone / Remote Process Engine Server

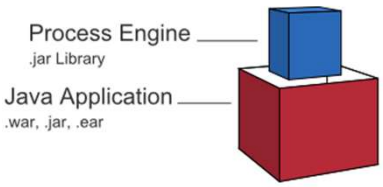


Process Engine Modes

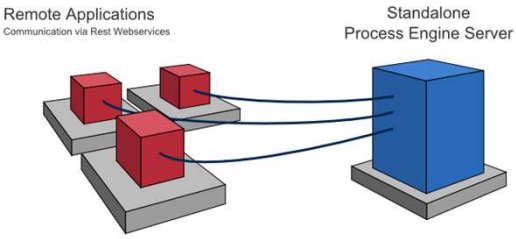
Shared, Container Managed Process Engine



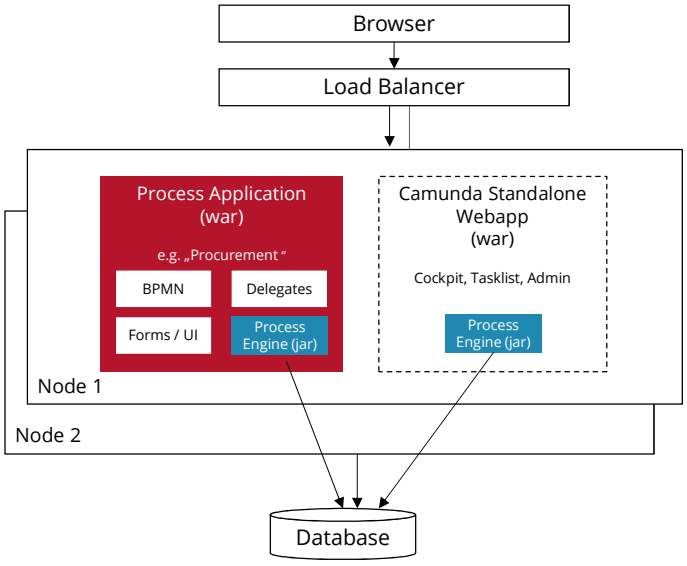
Embedded Process Engine



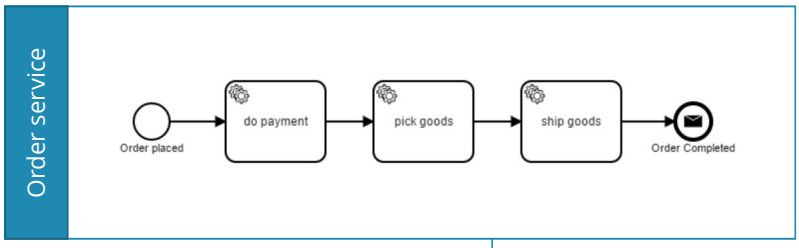
Remote Process Engine



Example Architecture With Embedded Process Engines

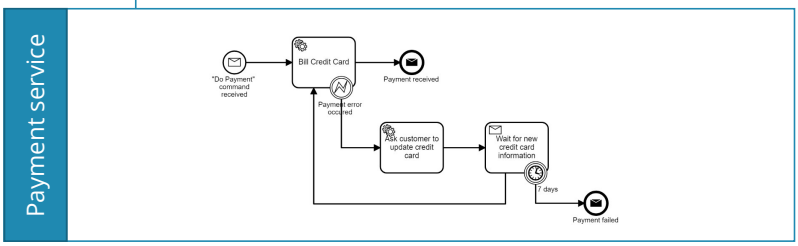


Orchestration With Microservices

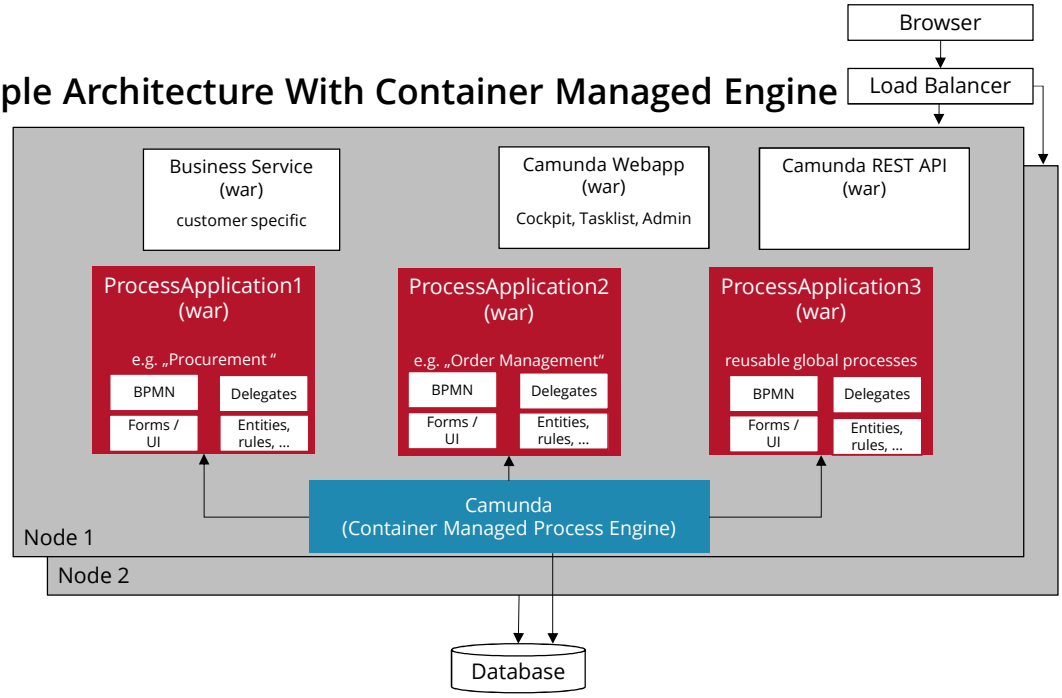


Possible Environments:
- Embedded engines
- Container managed engines with single process application

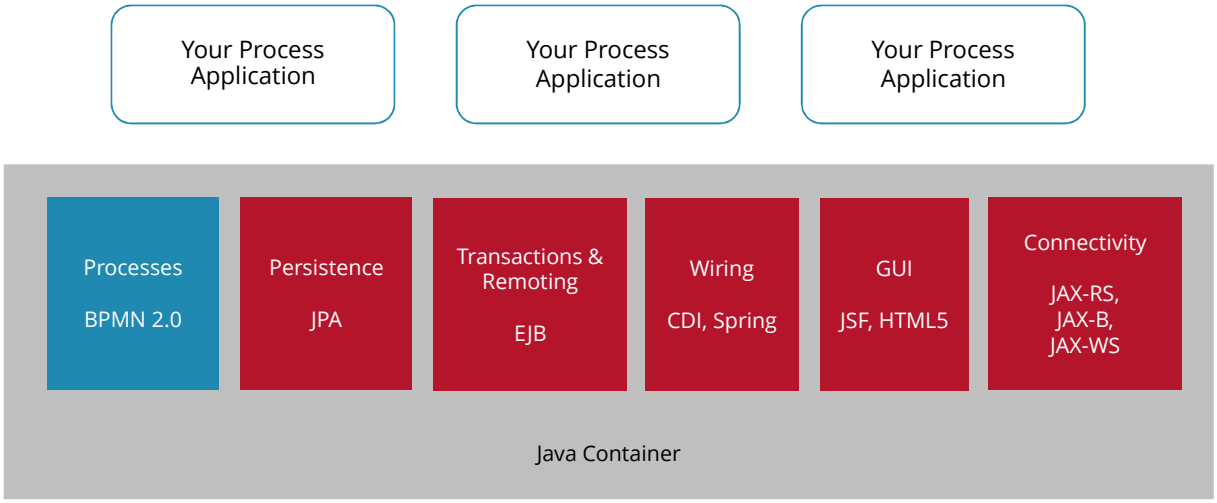
Maybe you add:
- History event stream for central history
- Task events for central task list

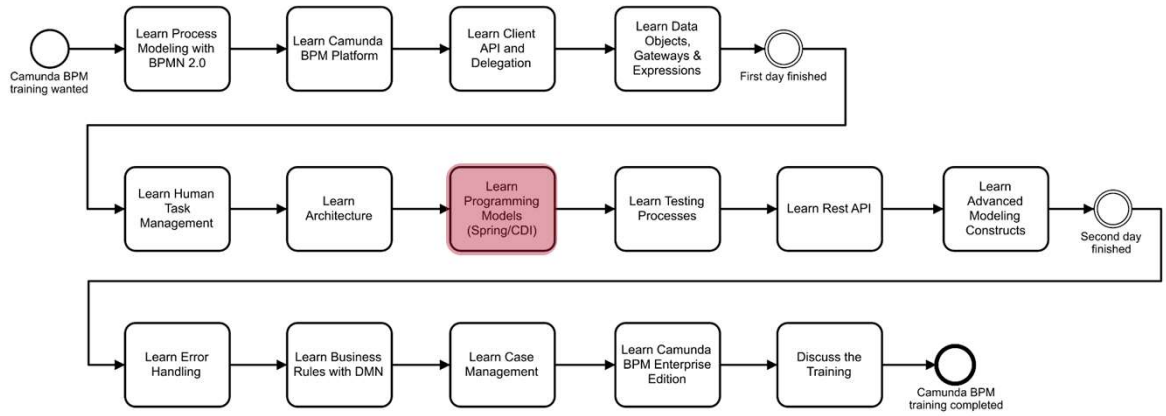


Example Architecture With Container Managed Engine



Process as „First Class Citizen“ in Development Stack





Camunda BPM Provides

Spring Integration

- Control lifecycle of process engine via Spring
- Inject engine and services via Spring
- Call named beans via Expression Language (works with container managed engine too)
- Limited TX integration in container managed engine on Tomcat (without JTA)

CDI Integration

- Inject engine and services via CDI
- Call named beans via Expression Language (works with container managed engine too)
- Contextual Process Implementation and CDI Events (see next slide)

Camunda BPM CDI Integration

basic

advanced

Access Camunda BPM

```
@Inject RuntimeService
@Inject TaskService …
```

Access your beans

```
<serviceTask id="taskId" name="service"
   camunda:expression="${service.method(param)}">
</serviceTask>
```

CDI bean

Contextual Process Implementation

```
@CompleteTask, @StartProcess
@Inject BusinessProcess
@BusinessProcessScoped
@Inject Map<String,Object> processVariables
…
```

Obtaining Engine Services Through Dependency Injection

Inject Process Engine Services
into Spring / CDI / EJB Beans

```
@Inject
private RuntimeService runtimeService;

@POST
@Consumes(MediaType.APPLICATION_JSON)
public void bookingRequest(Booking booking) {
  runtimeService.startProcessInstanceByMessage("bookingRequest");
}
```


Calling Your Names

The @Named qualifier allows to give a bean an Expression Language name:

```
@Named
public class SimpleGreeter implements Greeter,
Serializable { … }
```

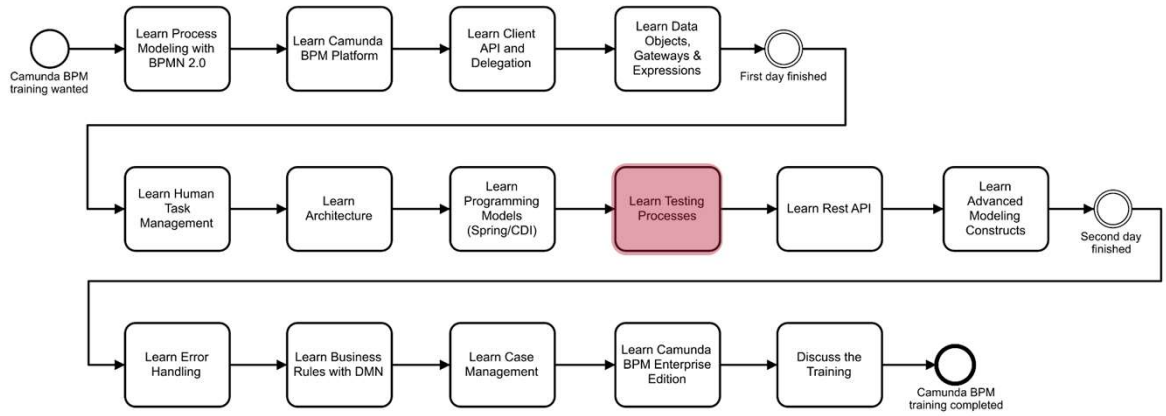
The El-Name of this bean is “simpleGreeter”. This allows us to invoke the bean from an Camunda BPM process:

```
<serviceTask id="taskId“ name="service"
  camunda:expression="${simpleGreeter.greet(‘you’)}">
</serviceTask>
```

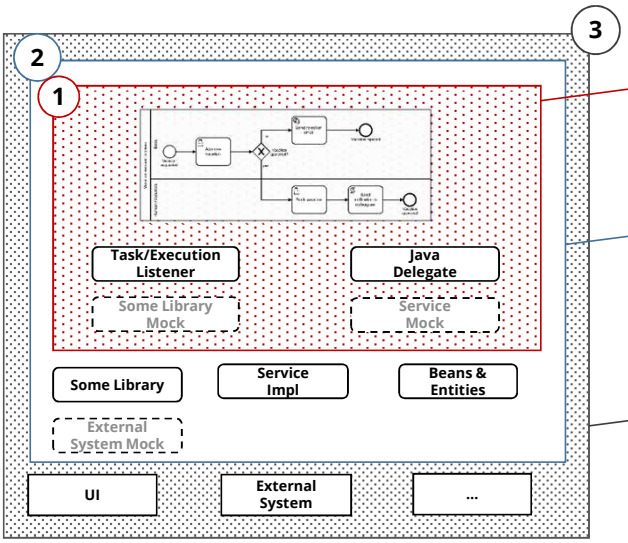
The resolved object is managed by the container (Spring/CDI) and can profit from other services (injection / lifecycle / …).

Delegate Expression


```
@Named("bookingService")
public class BookingService implements JavaDelegate {
  private final static Logger LOG = Logger.getLogger(BookingService.class.getName());

  public void execute(DelegateExecution execution) throws Exception {
    try {
      // perform booking
    } catch (Exception e) {
      LOG.log(Level.WARNING, "Exception while performing booking", e);
      throw new BpmnError("booking error");
    }
  }
}
```



Testing Scopes

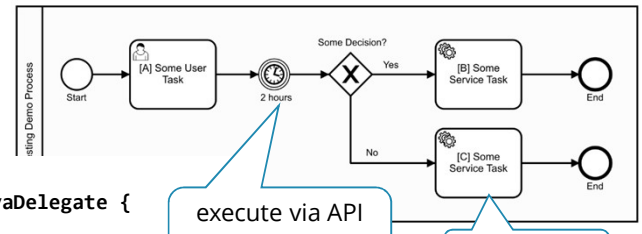


Goal: Process Model with Data, EL & Adapter Logic
- In Memory
- Single Thread
- No Container
- No JobExecutor

Goal: Close to Real-Life environment
- Container
- Arquillian & co

Goal: It REALLY works
- Integration Tests
- Human Driven

Example

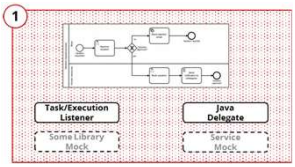



```
@Named
public class CustomerServiceAdapter implements JavaDelegate {

  @Inject
  CustomerService customerService;

  @Override
  public void execute(DelegateExecution execution) throws Exception {
    // Input Mapping
    Customer customer = new Customer();
    customer.setName((String) execution.getVariable("customerName"));

    // Service call
    String customerId = customerService.createCustomer(customer);

    // Output Mapping
    execution.setVariable("customerId", customerId);
  }
}
```

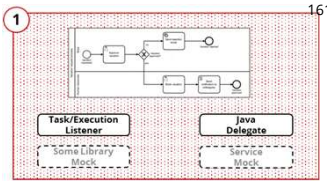
execute via API
execute(job());

use mocks

This is what we want to mock



Testing on Scope 1



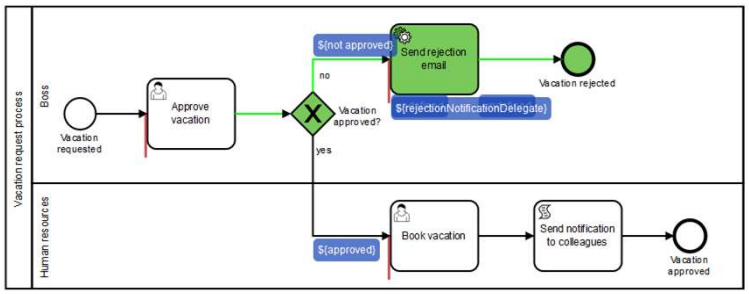
We recommend

- JUnit
- Camunda Rule (included in camunda-engine)
- Camunda-bpm-assert (https://github.com/camunda/camunda-bpm-assert)
- Mock-Provider of your choice, e.g. PowerMock + Mockito (or e.g. EasyMock or Jmockit)
- Process Instance Modification to start before the unit to test

Benefits

- In Memory Process Engine: Fast & always clean
- Single Thread / No Job Executor: Much easier to understand, no timing issues
- No Container: Less environment, faster turnaround times
- Runs quickly on every developer machine! By “right-click -> Run as Junit”!

Real Unit Test

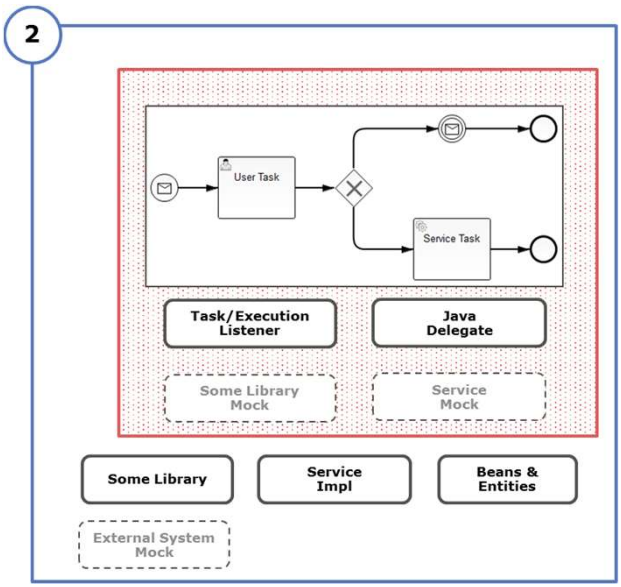



```
@Test
@Deployment(resources = "vacation-request.bpmn")
public void testVacationNotApproved() {
  ProcessInstance processInstance = runtimeService()
      .createProcessInstanceByKey("VacationRequestProcess")
      .setVariables(withVariables("employee", "John",
          "from", new Date(),
          "to", new Date(),
          "approved", false))
      .startAfterActivity("approveVacationUserTask")
      .execute();

  assertThat(processInstance).isEnded().hasPassed("vacationRejectedEndEvent");
}
```

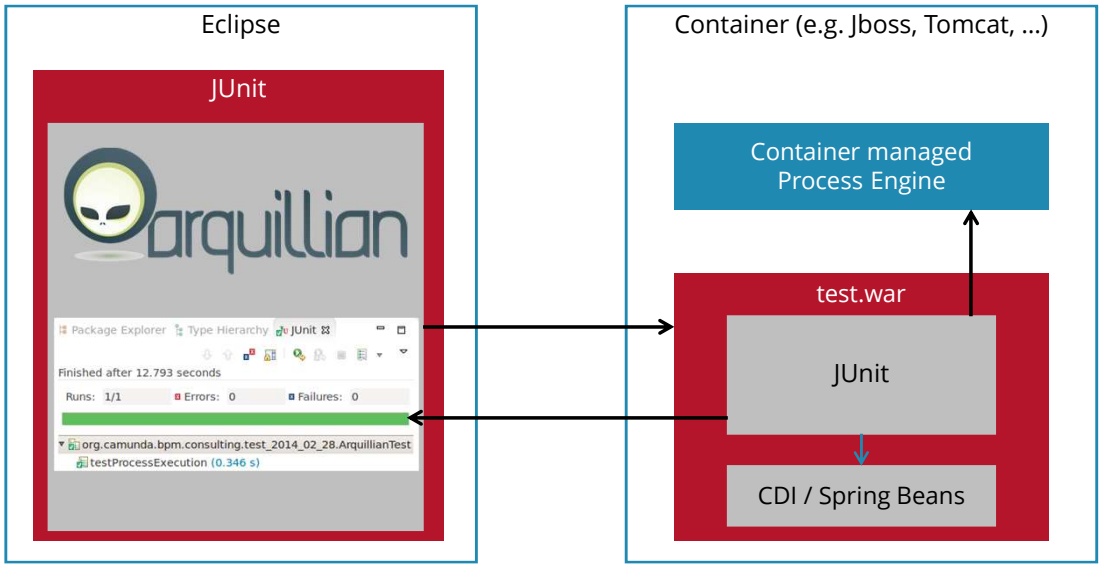
Process Instance Modification

Testing on Scope 2

- Infrastructure needed (e.g. CDI, Transactions, JPA, ...)
- Possible Alternatives
  - Start own test environment
    - Minimalistic (Needle, ...)
    - Real (Weld, Hibernate, OpenEJB, ...)
    - Often Spring-Driven
  - Run tests as integration tests
  - Arquillian
- Forces:
  Effort for setup & maintenance, effort to run, possibility to run on developer machine

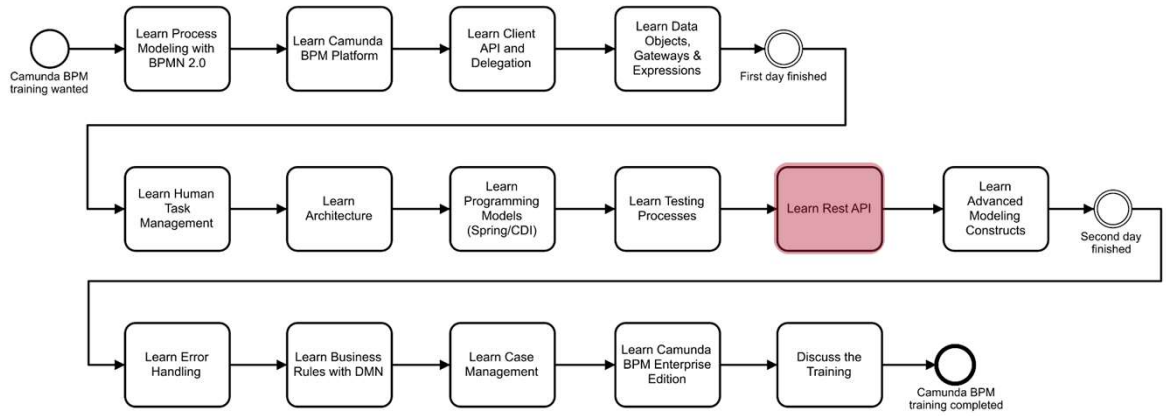


Example on Scope 2: Arquillian

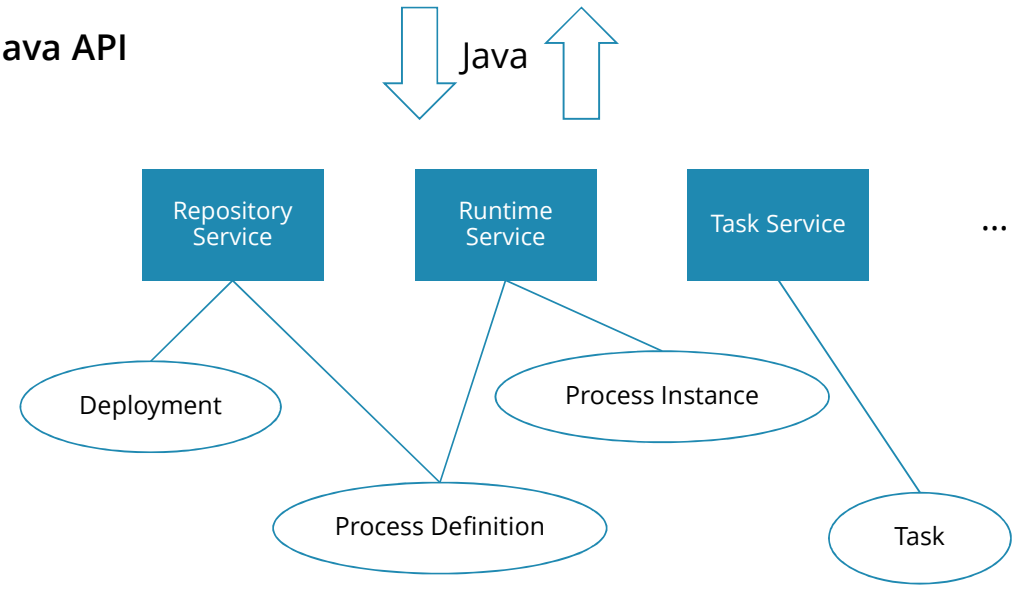


Exercise 7

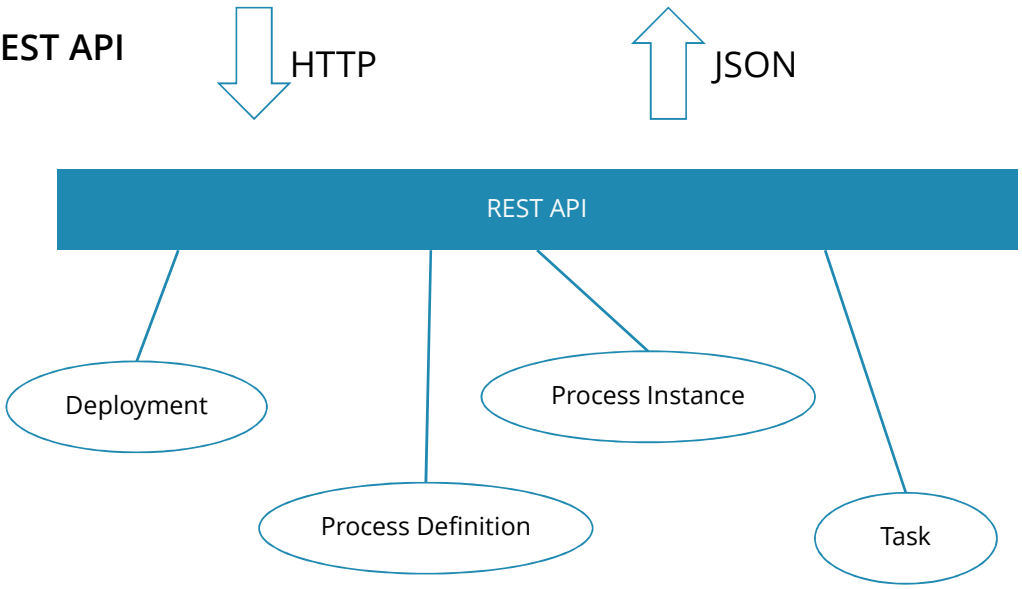
Follow the provided
instructions for exercise 7



Java API

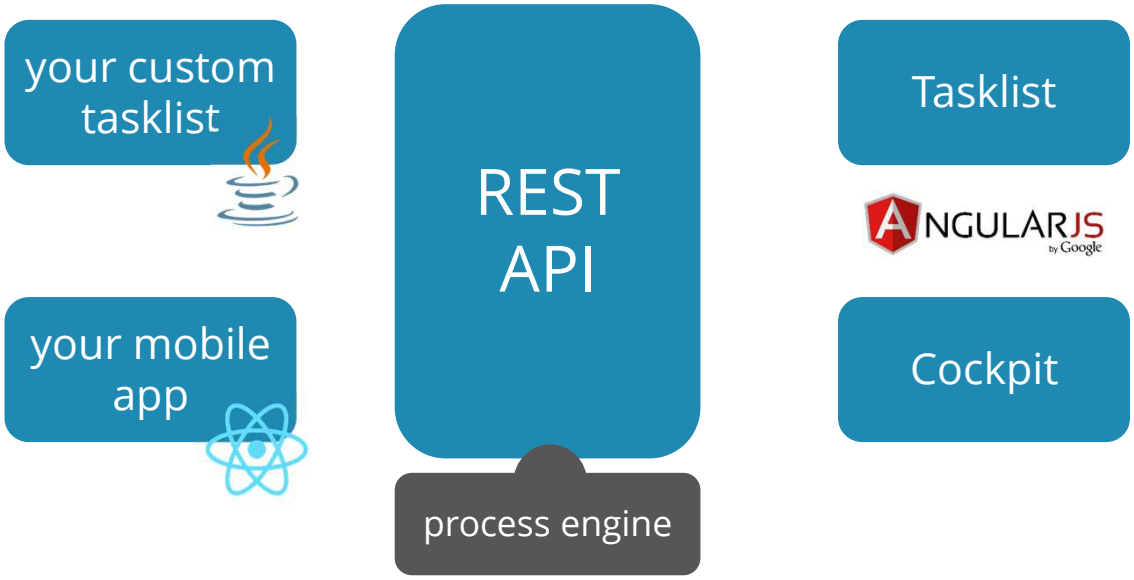


REST API



- Same power as Java API

Example Usage Scenarios



Camunda BPM REST API

Some REST calls:

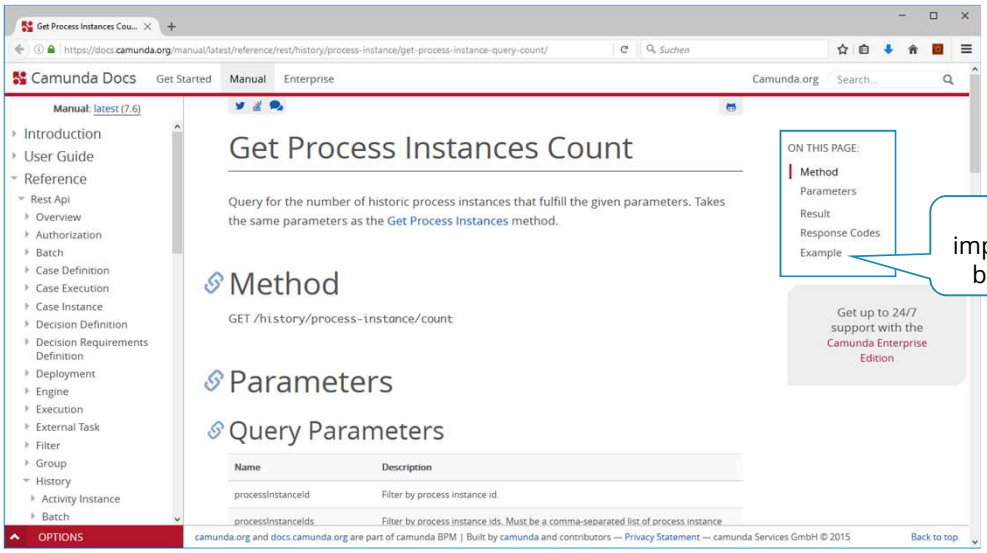
```
GET http://localhost:8080/engine-rest/process-definition
```

```
POST http://localhost:8080/engine-rest/process-definition/key/VacationRequestProcess/start

{"variables":
    {"employee" : {"value" : "Mary", "type": "String"},
     "from" : {"value" : "2016-12-20T00:00:00", "type": "Date"},
     "until" : {"value" : "2016-12-28T23:59:00", "type": "Date"}
    }
}
```

```
GET http://localhost:8080/engine-rest/task?processDefinitionKey=VacationRequestProcess
```

https://docs.camunda.org/manual/latest/reference/rest/

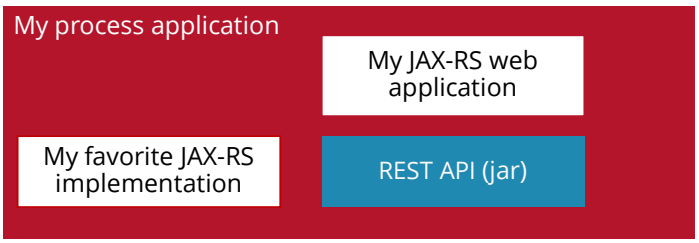
Documentation



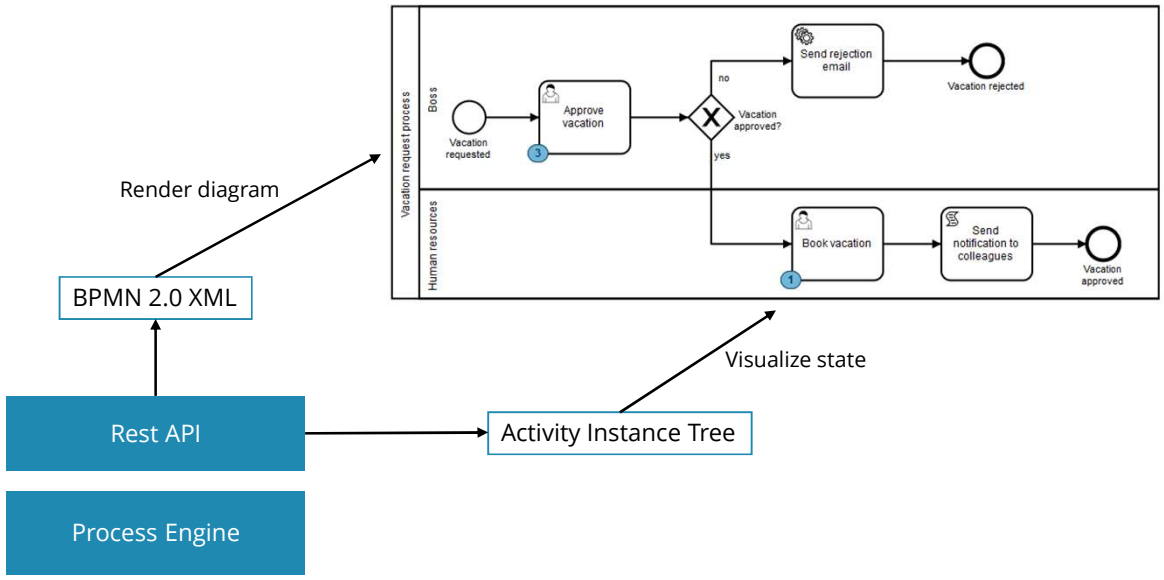
REST API: Embed and Extend!

- REST API as a library
- Own JAX-RS implementation
- Extension/limitation of resources possible

```
@ApplicationPath("/")
public class MyApplication extends Application {
  @Override
  public Set<Class<?>> getClasses() {
    Set<Class<?>> classes = new HashSet<Class<?>>();
    // add your own classes

    // add camunda engine rest classes that you need
    classes.add(ProcessEngineRestServiceImpl.class);
    classes.add(ProcessDefinitionRestServiceImpl.class);
  }
}
```



REST API for Process Visualization



Two REST APIs in the Distribution

`engine-rest.war`

Basic HTTP Authentication

check `web-inf.xml`

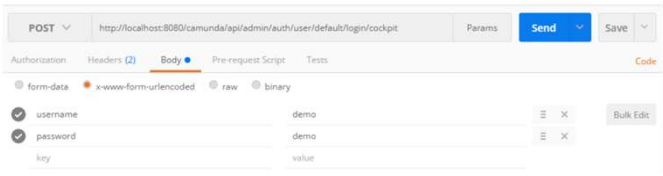
```
<!-- Http Basic Authentication Filter -->
  <!-- <filter>
    <filter-name>camunda-auth</filter-name>
    ...
  </filter-mapping> -->
```

Remove the comment before go to production!

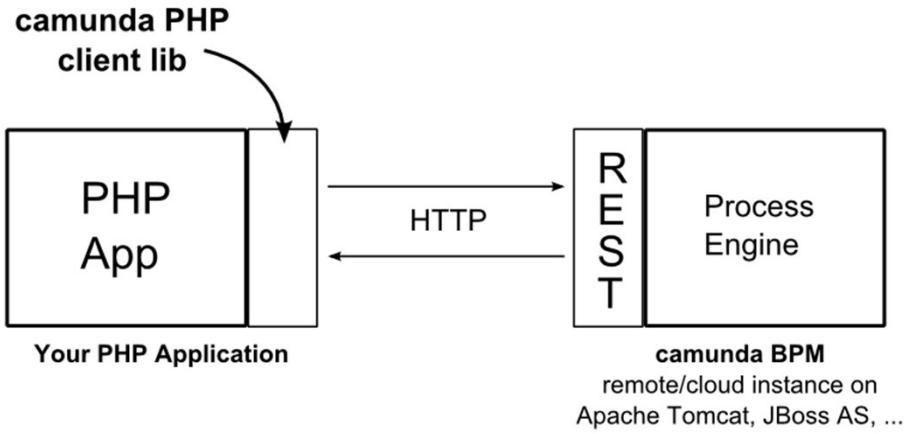
embedded in `camunda-webapp.war`

Cookie based authentication:

```
POST
http://localhost:8080/camunda/api/admin
/auth/user/default/login/cockpit


-H "Content-Type: application/x-www-form-
urlencoded"
-H "Accept: application/json"
-d 'username=demo&password=demo'
```

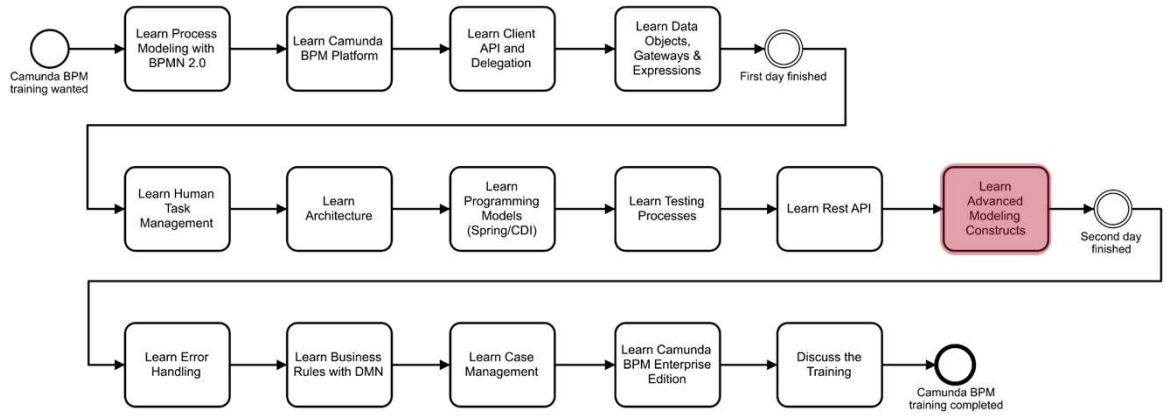


REST API Usage in Camunda PHP SDK

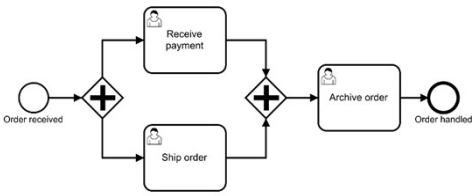


Exercise 8

Follow the provided instructions for exercise 8

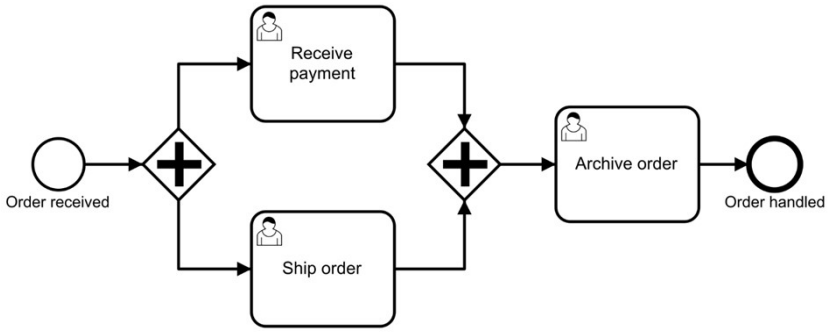


Parallel Gateway




```
<startEvent id="theStart">
<sequenceFlow id="flow1" sourceRef="theStart" targetRef="fork" />

<parallelGateway id="fork" />
<sequenceFlow sourceRef="fork" targetRef="receivePayment" />
<sequenceFlow sourceRef="fork" targetRef="shipOrder" />

<userTask id="receivePayment" name="Receive Payment" />
<sequenceFlow sourceRef="receivePayment" targetRef="join" />

<userTask id="shipOrder" name="Ship Order" />
<sequenceFlow sourceRef="shipOrder" targetRef="join" />

<parallelGateway id="join" />
<sequenceFlow sourceRef="join" targetRef="archiveOrder" />

<userTask id="archiveOrder" name="Archive Order" />
<sequenceFlow sourceRef="archiveOrder" targetRef="theEnd" />

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

How Many Transactions?



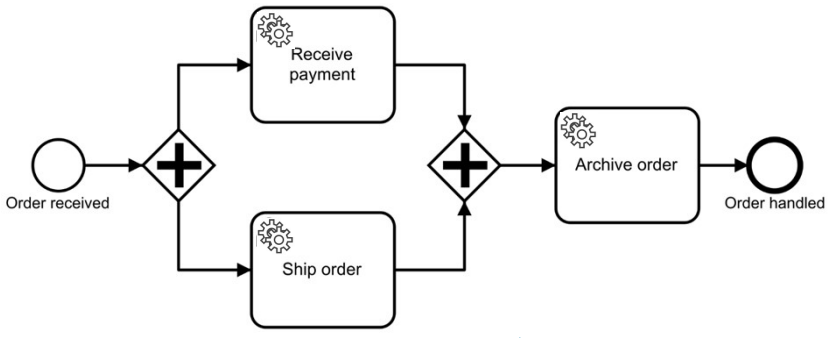
How Many Transactions?

Hidden solution

How Many Transactions? (The Other Way Around)

Hidden solution

How Many Transactions?



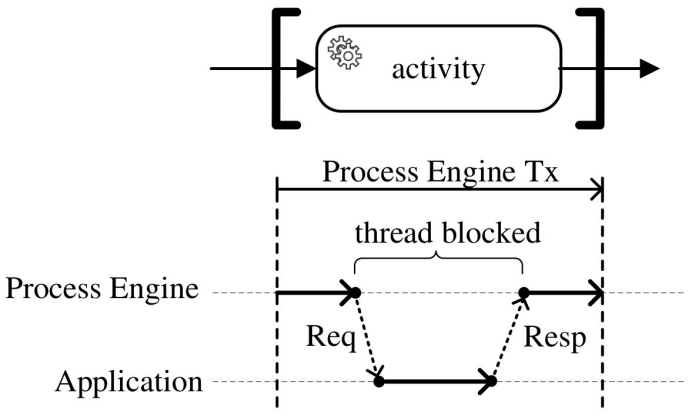
What if all tasks were Service Tasks?

How Many Transactions?

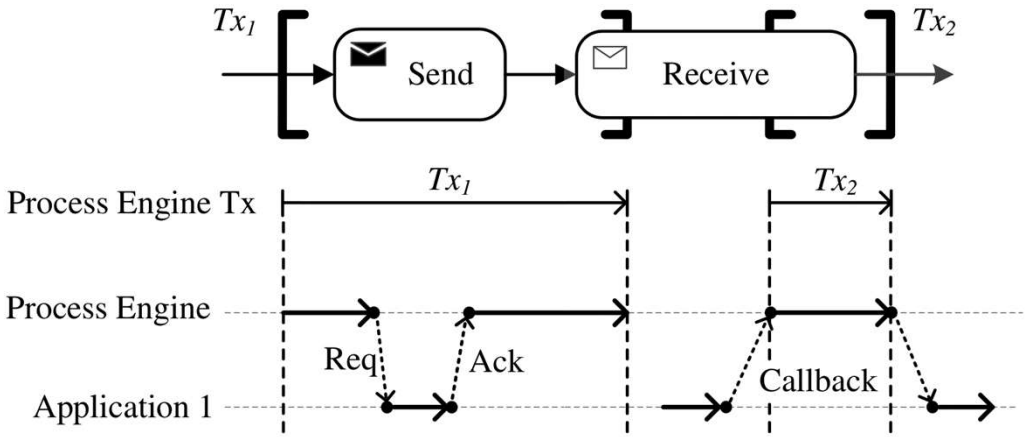
Hidden solution

What if all tasks were Service Tasks?

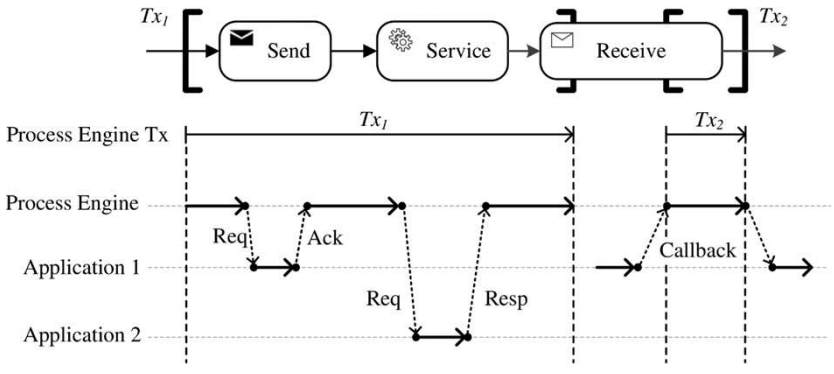
Synchronous Service Invocation: Request / Response



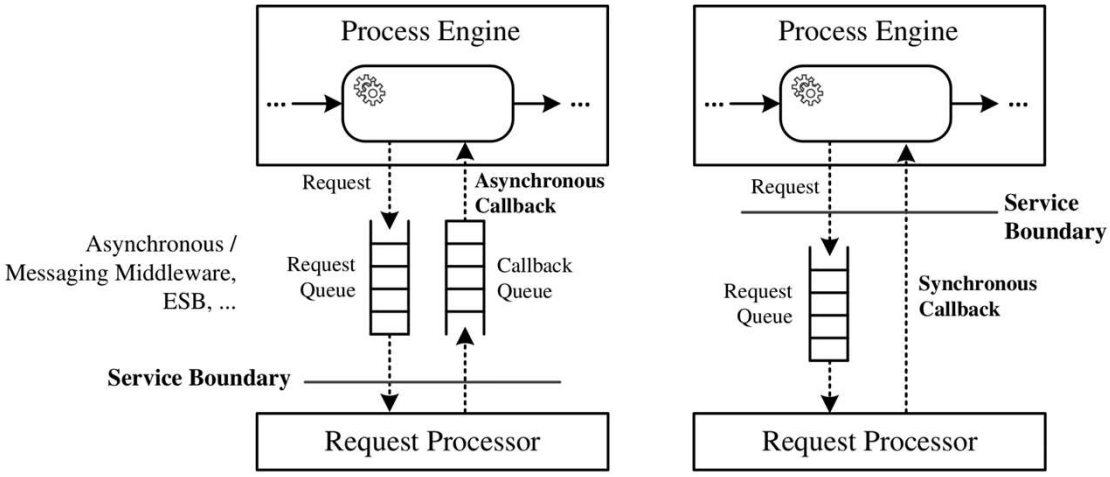
Asynchronous Service Invocation: Req / Ack / Callback



Asynchronous & Synchronous Invocation Combined



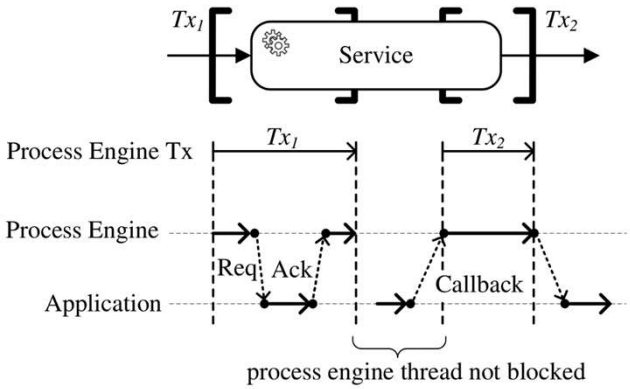
Queuing and Callbacks



e.g. JMS

e.g. External Service Task

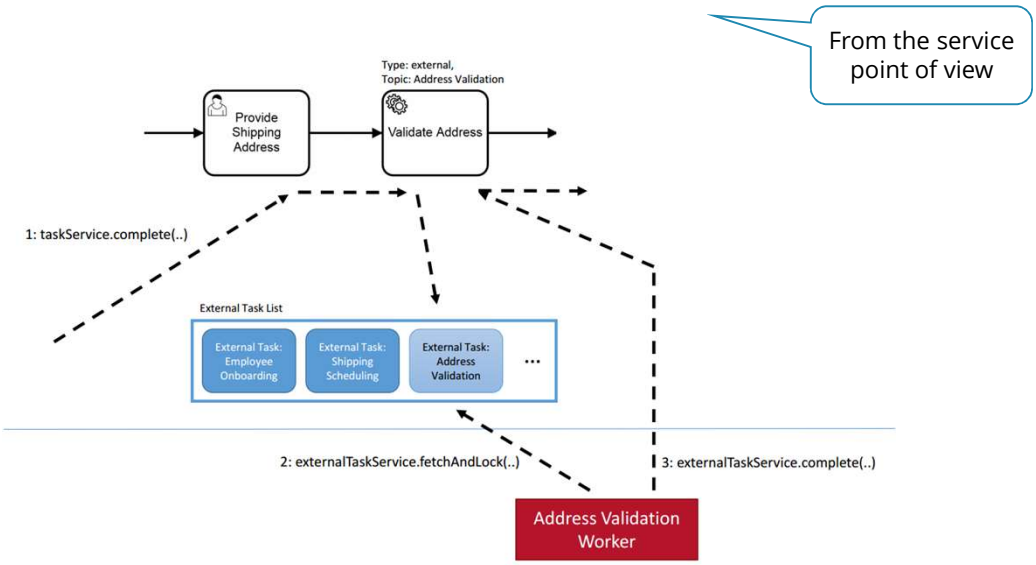
External / Asynchronous Service Task



External Service Task:
https://docs.camunda.org/manual/latest/user-guide/process-engine/external-tasks/

Asynchronous Service Task:
https://github.com/camunda/camunda-bpm-examples/tree/master/servicetask/service-invocation-asynchronous

External Service Task: Don't Call Us! We Call You!



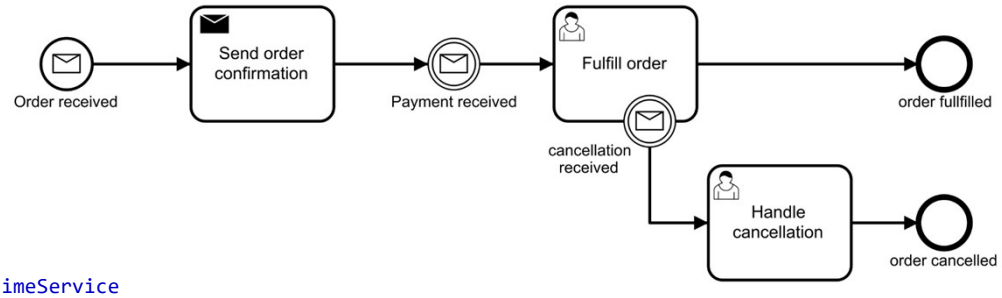
https://docs.camunda.org/manual/latest/user-guide/process-engine/external-tasks/

Event Coverage

Type	Start			Intermediate				End
	Normal	Event Subprocess	Event Subprocess non-interrupt	catch	boundary	boundary non-interrupt	throw	
None	○						○	○
Message	✉	✉	✉	✉	✉	✉	✉	✉
Timer	◷	◷	◷	◷	◷	◷		
Conditional	☰	☰	☰	☰	☰	☰		
Link				⇨			⇨	
Signal	△	△	△	△	△	△	△	△
Error		ϟ			ϟ			ϟ
Escalation		Ⓐ	Ⓐ		Ⓐ	Ⓐ	Ⓐ	Ⓐ
Termination								●
Compensation		⏪			⏪		⏪	⏪
Cancel					⊗			⊗
Multiple	⬠	⬠	⬠	⬠	⬠	⬠	⬠	⬠
Multiple Parallel	✚	✚	✚	✚	✚	✚		

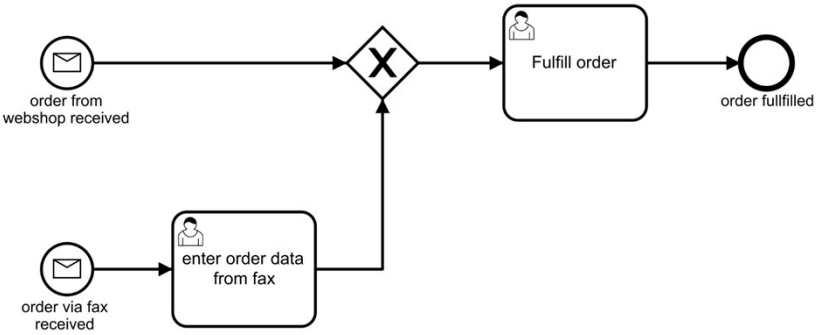
See: docs.camunda.org/manual/latest/reference/bpmn20/#events

Message Events



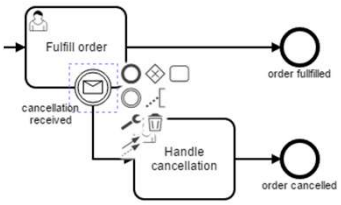

```
runtimeService
  .createMessageCorrelation("messageName")
  .processInstanceVariableEquals("myVariable", "myValue")
  .setVariable("myPayload", "myPayloadValue")
  .correlateWithResult();
```

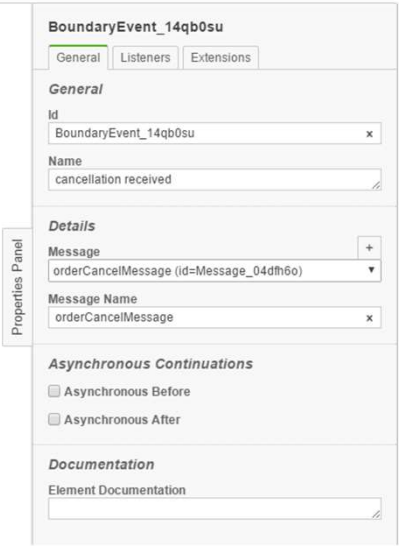

Multiple Message Start Events Possible




```
// correlate the message
runtimeService
  .createMessageCorrelation("messageName")
  .processInstanceBusinessKey("AB-123")
  .setVariable("payment_type", "creditCard")
  .correlateWithResult();
```

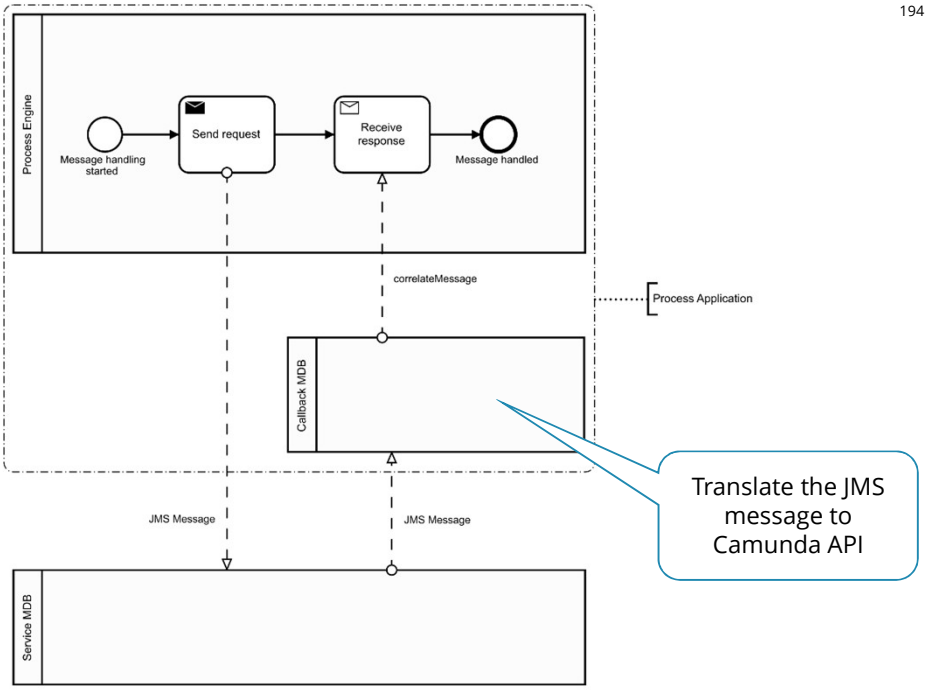
Configuring Message Names



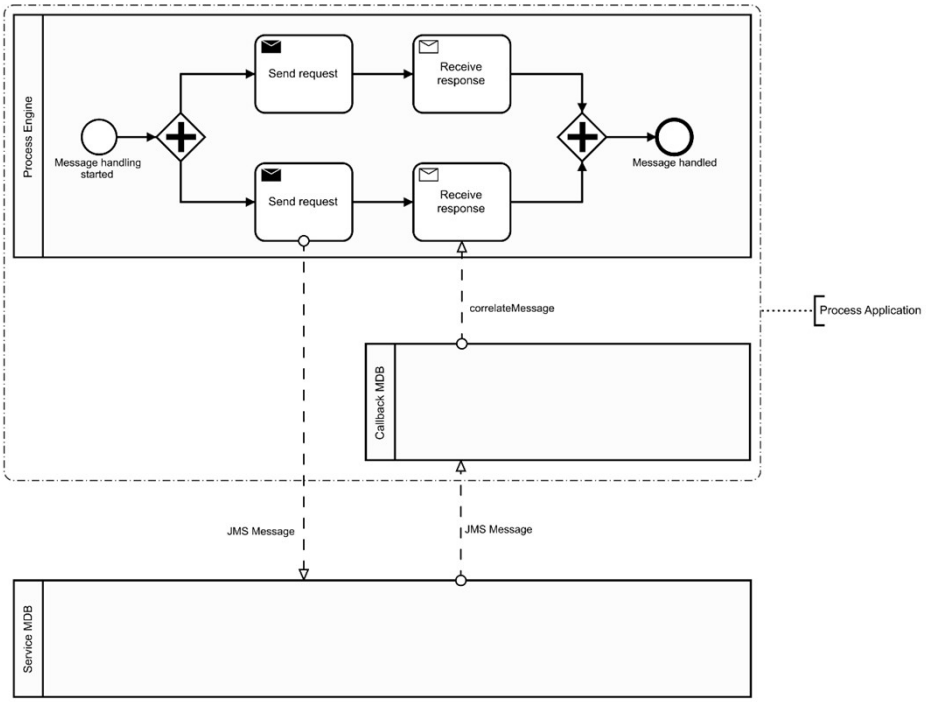



```
<bpmn:process id="OrderHandlingProcess" isExecutable="true">
    ...
    <bpmn:boundaryEvent id="BoundaryEvent_14qb0su"
name="cancellation received" attachedToRef="fulFillOrderUserTask">
      <bpmn:messageEventDefinition messageRef="Message_04dfh6o" />
    </bpmn:boundaryEvent>
    <bpmn:userTask id="fulFillOrderUserTask" name="Fulfill order">
    </bpmn:userTask>
    ...
</bpmn:process>
<bpmn:message id="Message_04dfh6o" name="orderCancelMessage" />
```

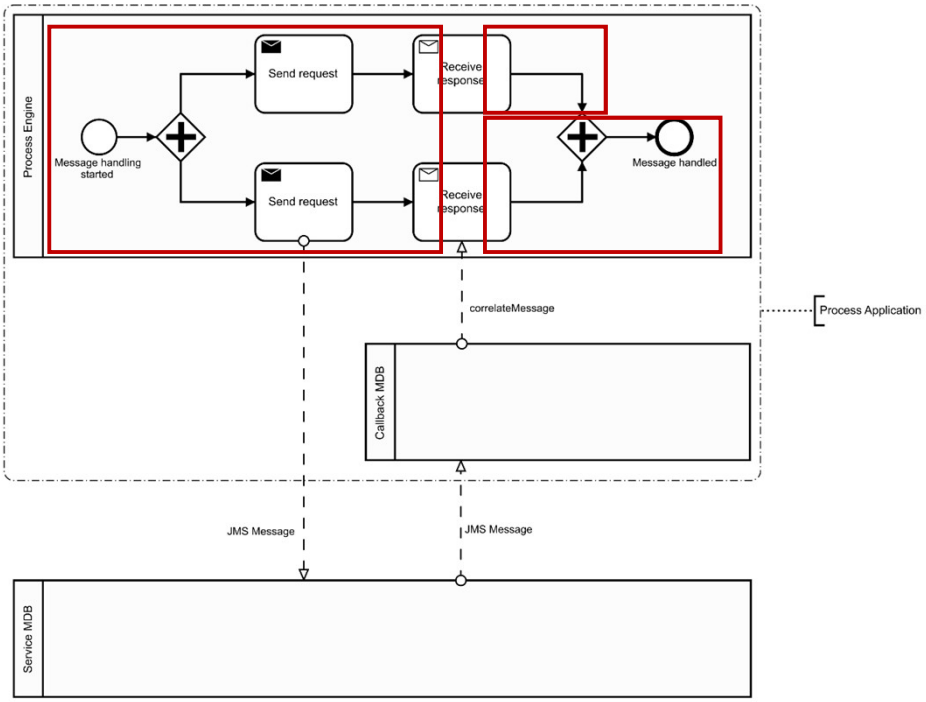

JMS



JMS
(#TX?)


Process Engine
Message handling started
Send request
Receive response
Send request
Receive response
Message handled
correlateMessage
Process Application
Callback MDB
JMS Message
JMS Message
Service MDB

JMS


Process Engine
Message handling started
Send request
Receive response
Send request
Receive response
Message handled
correlateMessage
Process Application
Callback MDB
JMS Message
JMS Message
Service MDB

Correlation Keys

Process Instance Level

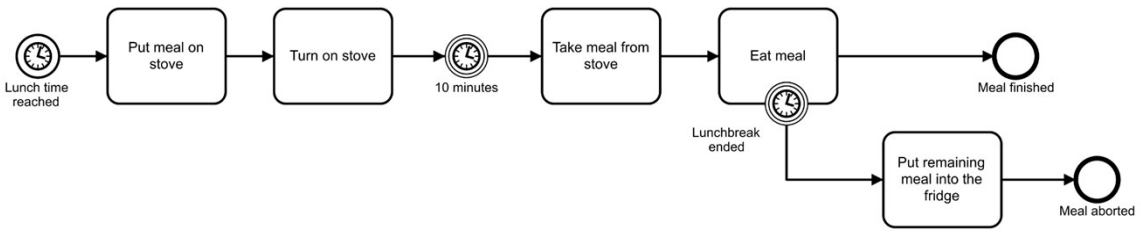
- Business Key
- Process Variable (e.g. UUID)
- Process Instance ID (not recommended)

Token Level (when waiting for multiple messages)

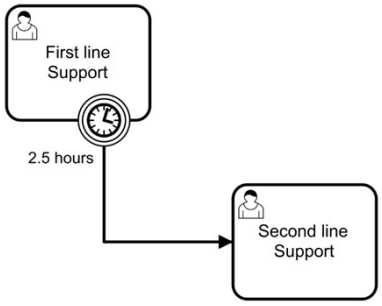
- Local Variable (e.g. UUID)
- Combination of Process Instance (see above) and Activity ID
- Execution ID (not recommended)

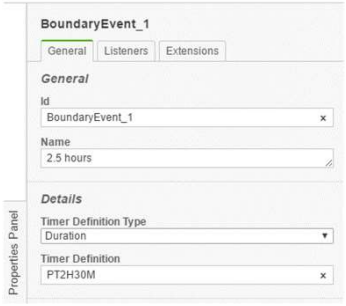
Alternative option: correlation table

Timer Events



Timer Event Configuration

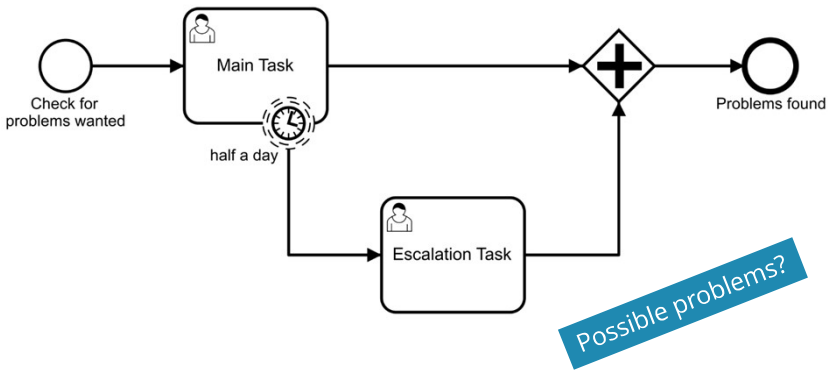




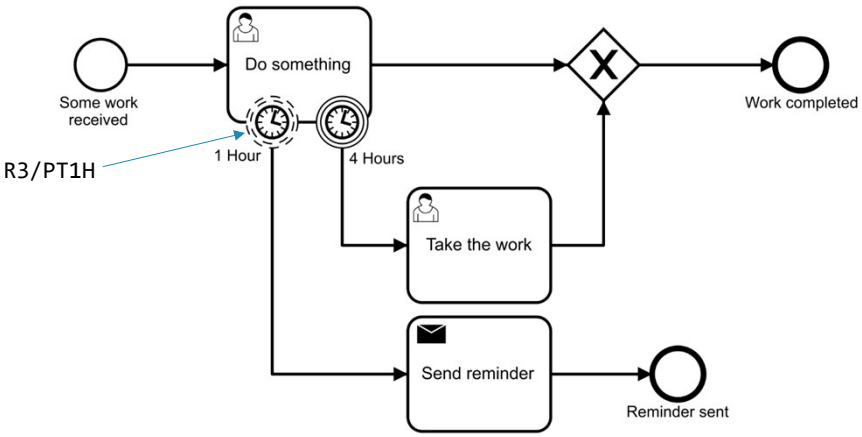

```
<bpmn:process id="Process_1" name="Timer Event Configuration Example" isExecutable="true">
  <bpmn:userTask id="firstLineSupportUserTask" name="First line Support" />
  <bpmn:boundaryEvent id="BoundaryEvent_1" name="2.5 hours" attachedToRef="firstLineSupportUserTask">
    <bpmn:timerEventDefinition>
      <bpmn:timeDuration xsi:type="bpmn:tFormalExpression">PT2H30M</bpmn:timeDuration>
    </bpmn:timerEventDefinition>
  </bpmn:boundaryEvent>
  <!--  -->
</bpmn:process>
```

ISO8601: https://en.wikipedia.org/wiki/ISO_8601

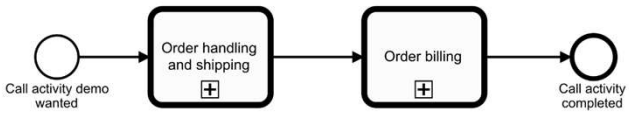
Non Interrupting Timer Event

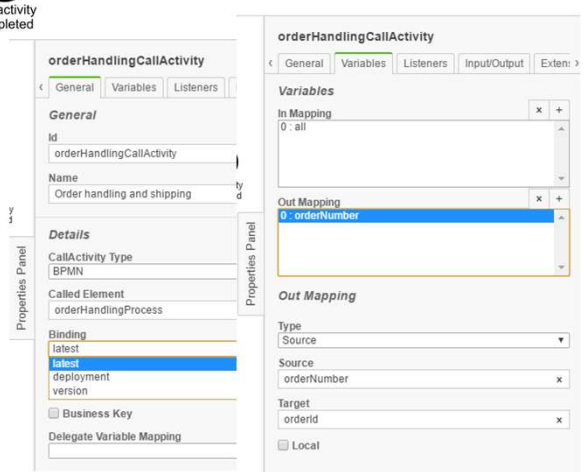


Pattern for Two Stage Escalation



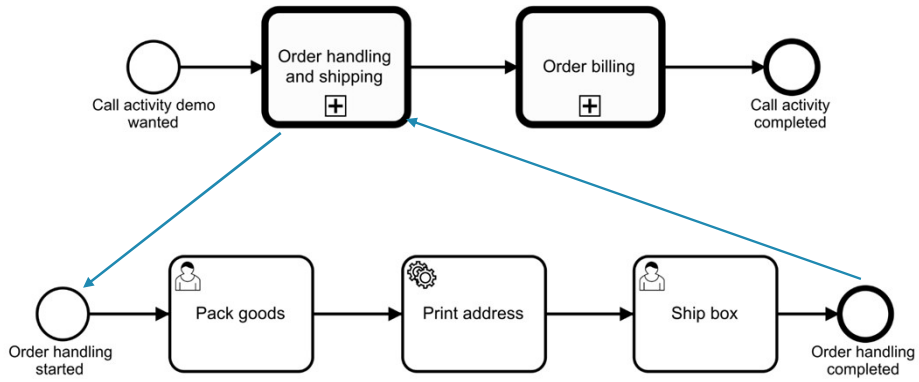
Reusable Sub-Processes: Call Activities



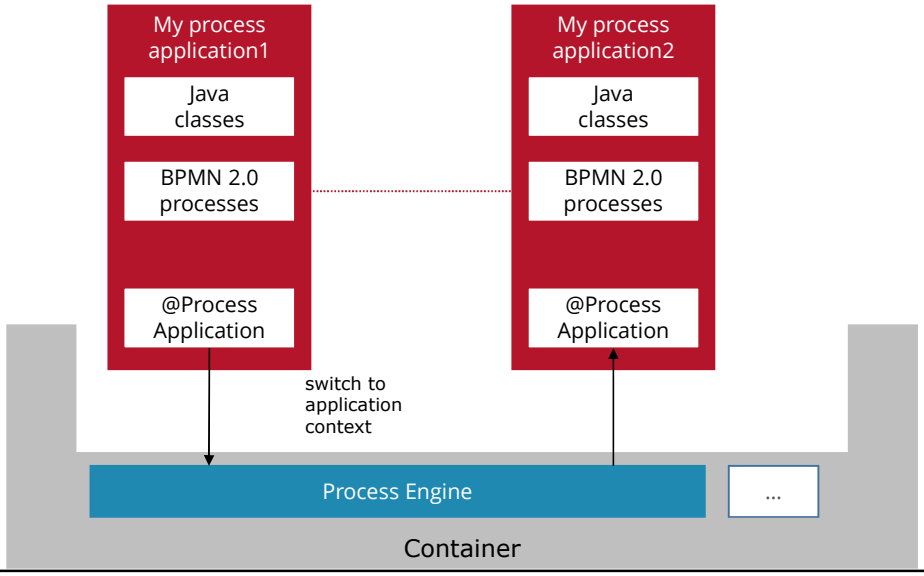


- Key is used to identify called process
- Process is resolved at runtime
- Variables can be mapped

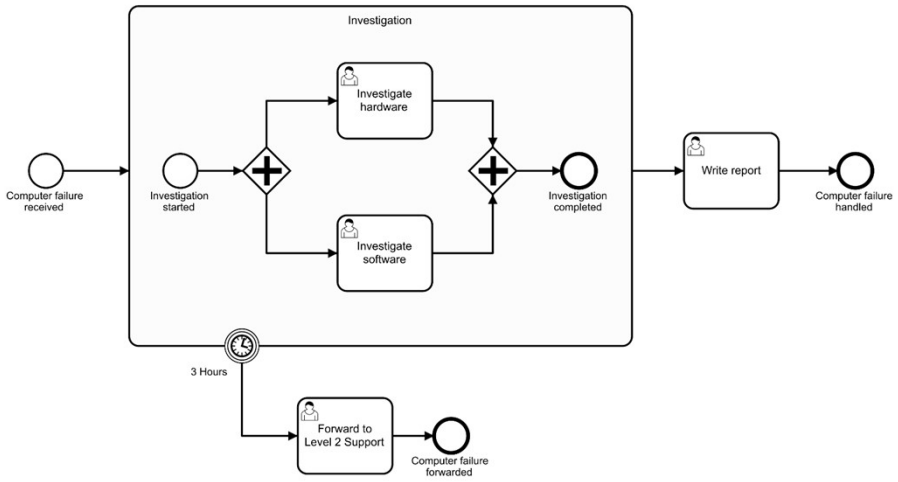
Reusable Sub-Processes: Call Activities



Calling Processes From Other Process Applications



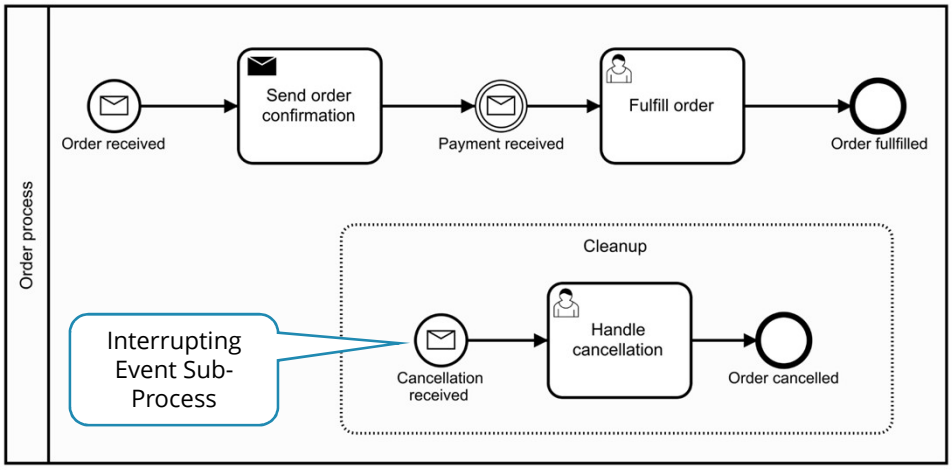
Embedded Sub-Processes



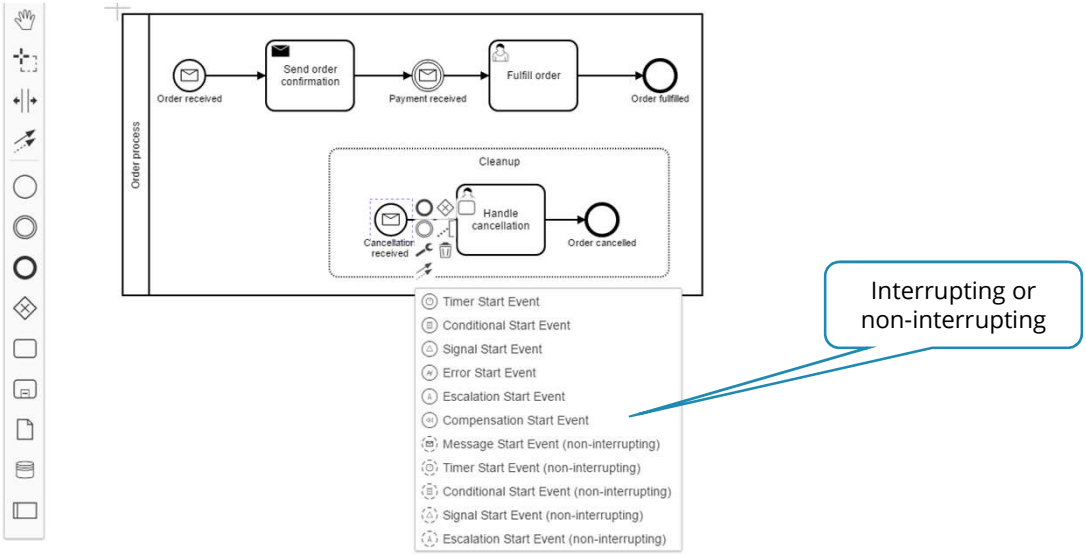
The Sub-Process Is Part of the Main Process

```
<bpmn:process id="Process_1" isExecutable="false">
  <bpmn:startEvent id="StartEvent_1" name="Computer failure received"></bpmn:startEvent>
  <bpmn:sequenceFlow id="SequenceFlow_1ilu0mr" sourceRef="StartEvent_1"
targetRef="investigationSubprocess" />
  <bpmn:subProcess id="investigationSubprocess" name="Investigation">
    <bpmn:startEvent id="investigationStartedStartEvent" name="Investigation
started"></bpmn:startEvent>
    <bpmn:parallelGateway id="ParallelGateway_1"></bpmn:parallelGateway>
    <bpmn:userTask id="investigateHardwareUserTask" name="Investigate hardware"></bpmn:userTask>
    <bpmn:userTask id="investigateSoftwareUserTask" name="Investigate software"></bpmn:userTask>
    <bpmn:parallelGateway id="ParallelGateway_2"></bpmn:parallelGateway>
    <bpmn:endEvent id="investigationCompletedEndEvent" name="Investigation
completed"></bpmn:endEvent>
    <bpmn:sequenceFlow id="SequenceFlow_01s0ans" sourceRef="investigationStartedStartEvent"
targetRef="ParallelGateway_1" />
      ...
  </bpmn:subProcess>
  ...
```

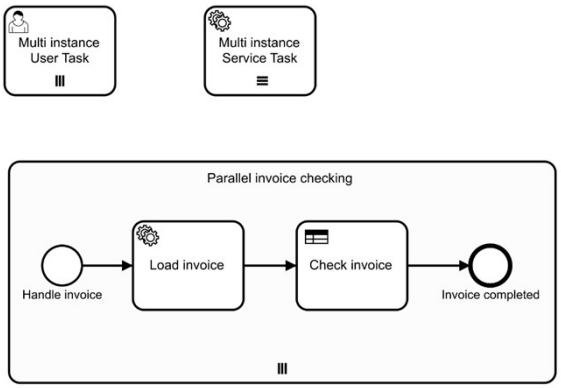
Event Sub-Process

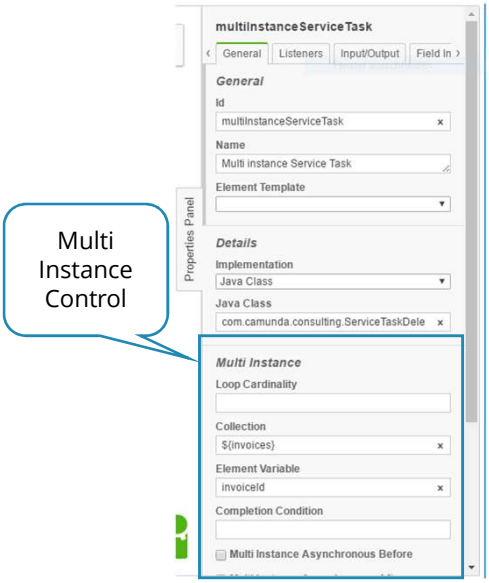


Possible Events to Start an Event Sub-Process

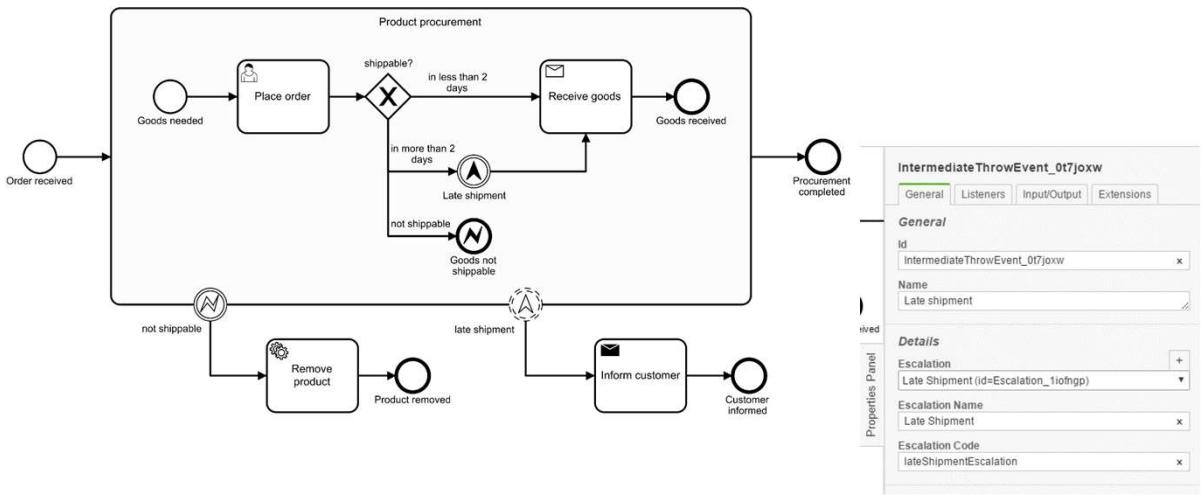


Multiple Instance Marker

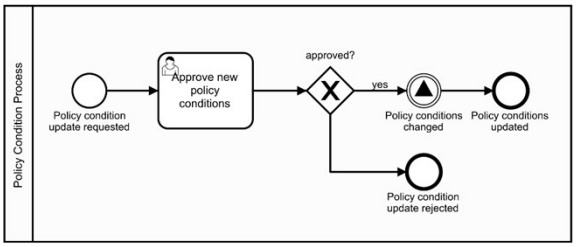


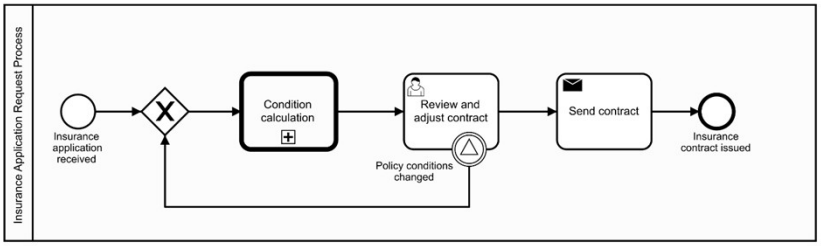


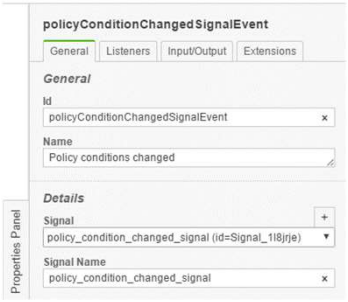
Escalation and Error Events



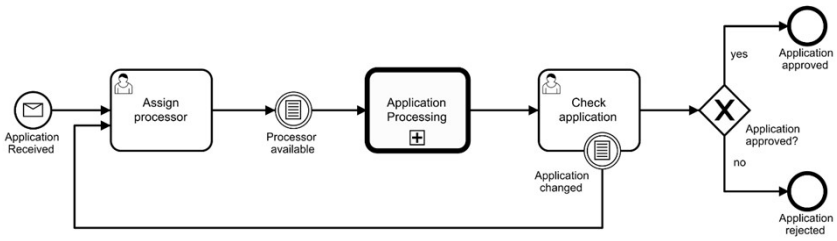
Signal Events – Broadcast to All Process Instances



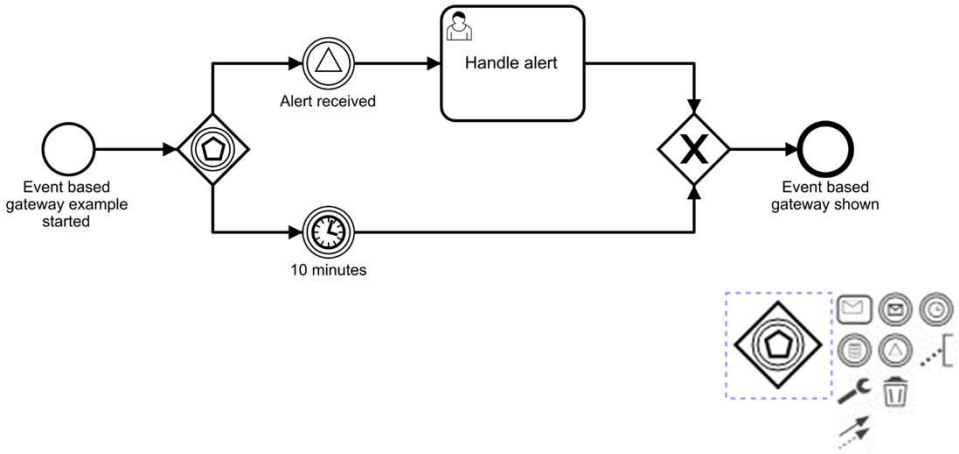




Conditional Events



Event-Based Gateway – Simple Example



Event-Based Gateway – Real World Example

Please model the following process:

If an insurant could be possibly subrogated against, I get information about that. I check that case and if the possibility is really there, I send a request for payment to the insurant and make me a reminder. If recourse is not possible, I close the case.

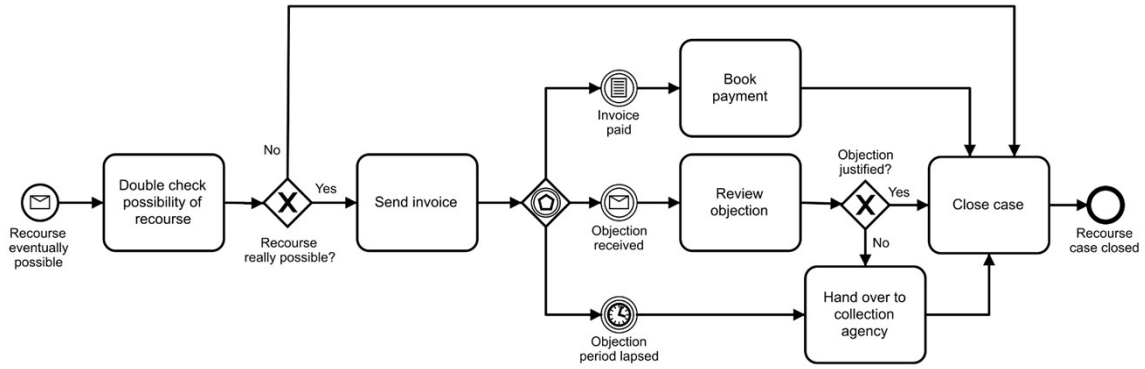
When we receive the money, I make a booking and close the case. If the insurant disagrees with the recourse, I'll have to check the reasoning of that. If he is right, I simply close the case. If he is wrong, I forward the case to a collection agency.

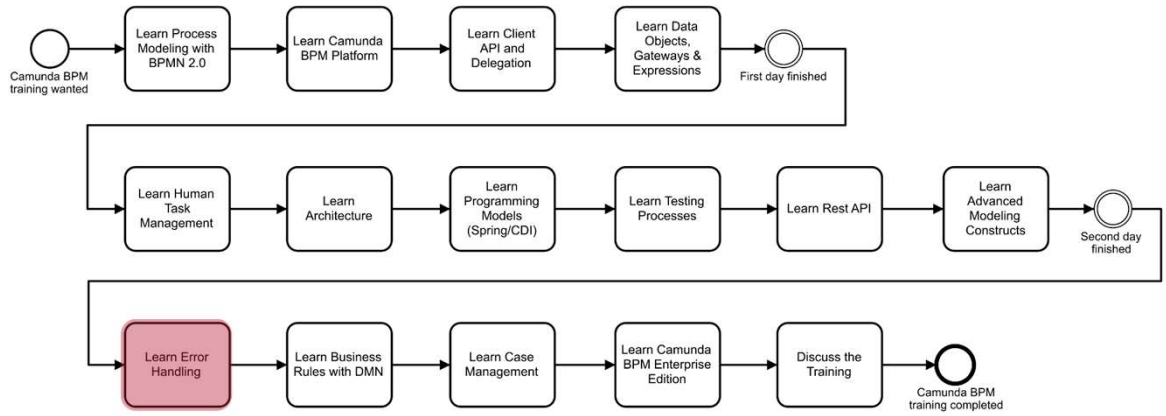
If the deadline for disagreement is reached and we haven't received any money, I forward the case to the collection agency as well.

Background information:

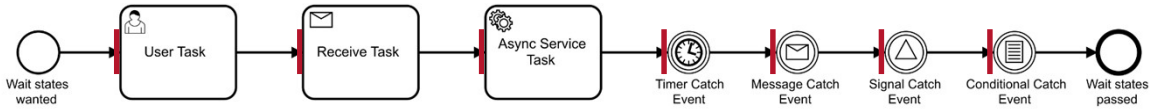
Insurants can be forced to pay back money they received from the insurance company for different reasons. This is called recourse. Here the clerk describes how this process works.

Event-Based Gateway – Real World Example





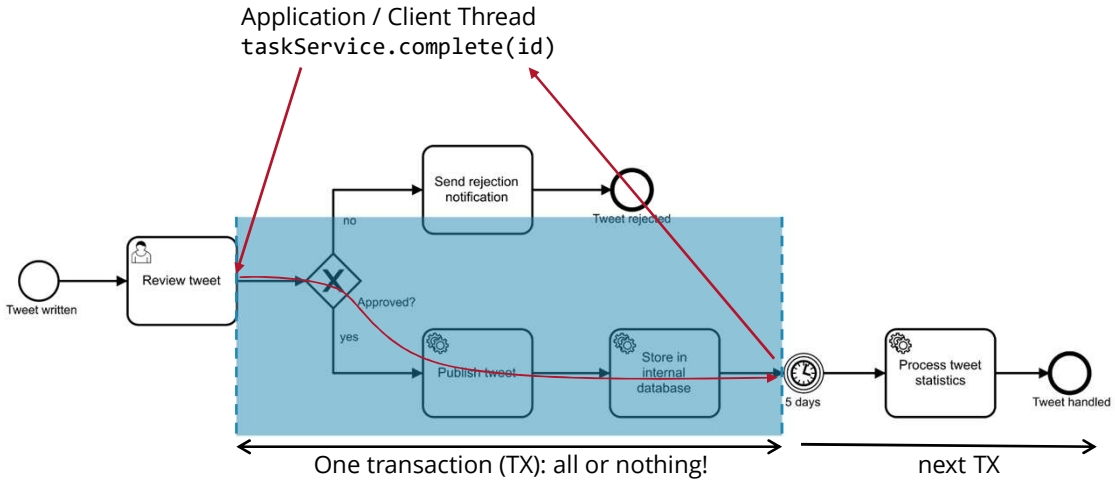
Wait States



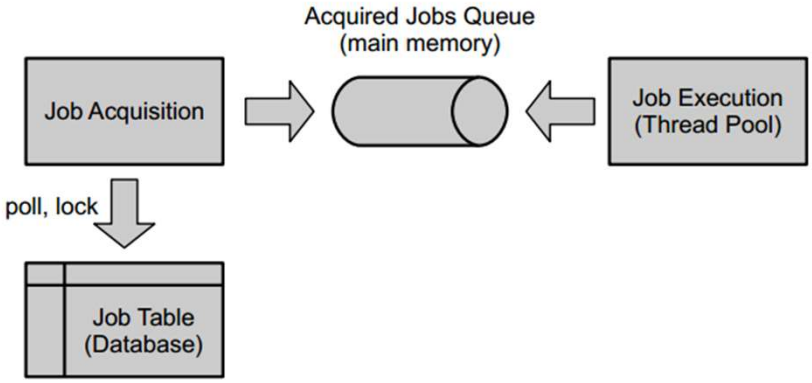
Also called:

- Transaction Boundaries
- Save Points

Transaction Boundaries

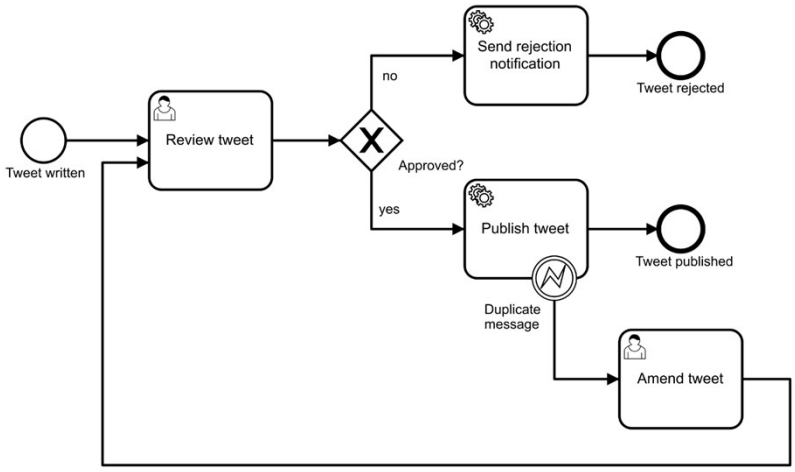


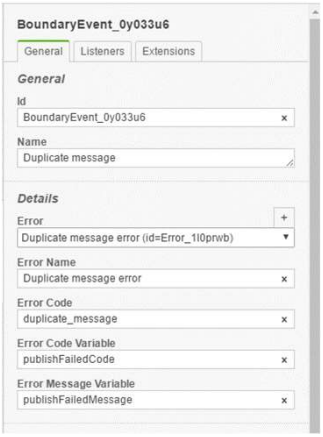
The Job Executor



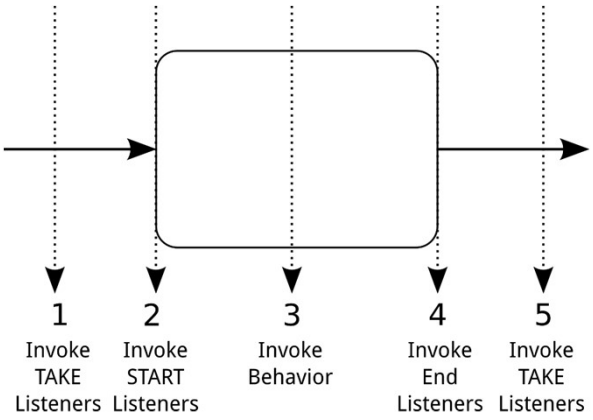
Demo

Error Handling

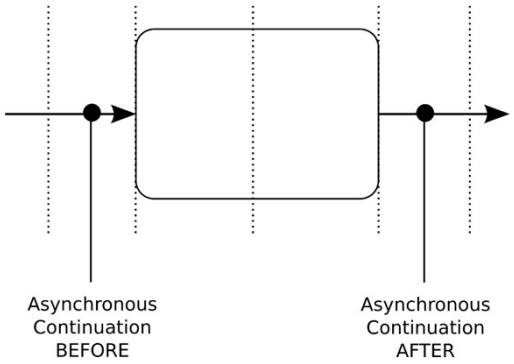




Asynchronous Continuation: Execution of a BPMN Element



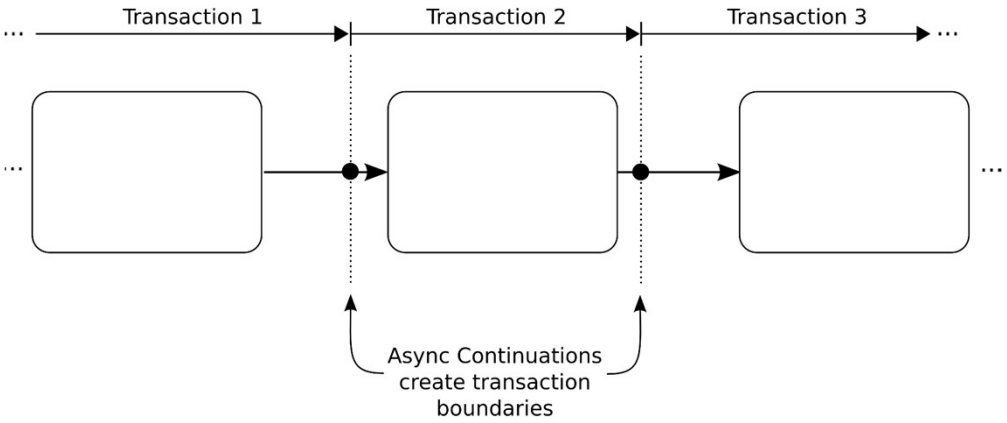
Asynchronous Continuation




```
<bpmn:serviceTask id="publishTweetServiceTask" name="Publish tweet" camunda:asyncBefore="true"
camunda:class="com.camunda.consulting.PublishTweetDelegate"/>
```

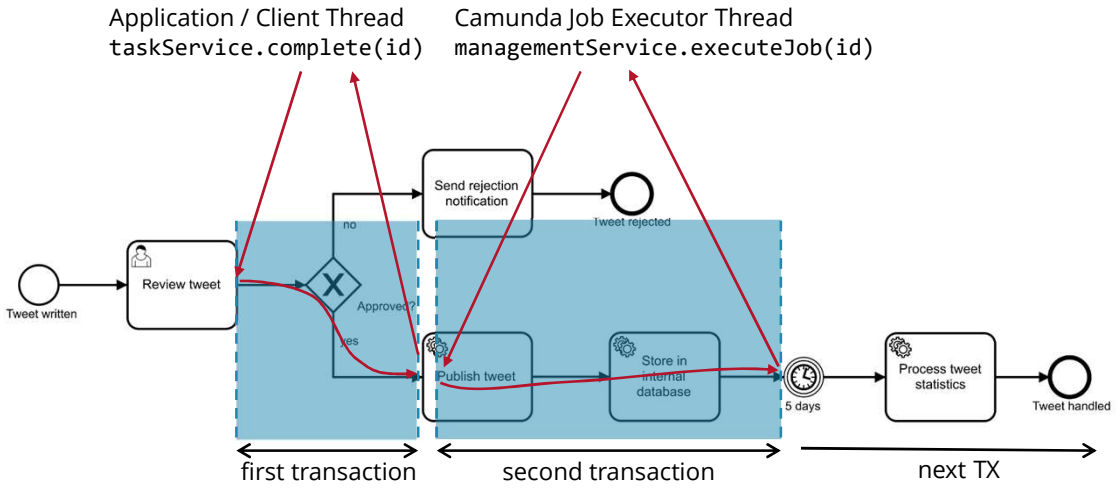
```
<bpmn:serviceTask id="publishTweetServiceTask" name="Publish tweet" camunda:asyncAfter="true"
camunda:class="com.camunda.consulting.PublishTweetDelegate"/>
```


Asynchronous Continuation = Transaction Boundaries/Save Points



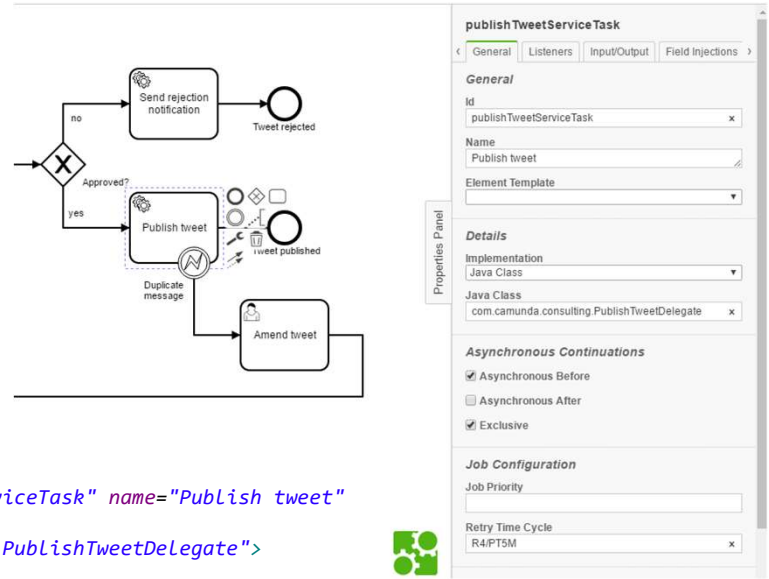

```
<bpmn:serviceTask id="publishTweetServiceTask" name="Publish tweet" camunda:asyncBefore="true"
camunda:asyncAfter="true" camunda:class="com.camunda.consulting.PublishTweetDelegate"/>
```

Transaction Boundaries




```
<bpmn:serviceTask id="publishTweetServiceTask" name="Publish tweet" camunda:asyncBefore="true"
camunda:class="com.camunda.consulting.PublishTweetDelegate"/>
```


Retry Strategy




```
<bpmn:serviceTask id="publishTweetServiceTask" name="Publish tweet"
camunda:asyncBefore="true"
camunda:class="com.camunda.consulting.PublishTweetDelegate">
  <bpmn:extensionElements>
    <camunda:failedJobRetryTimeCycle>R4/PT5M</camunda:failedJobRetryTimeCycle>
  </bpmn:extensionElements>
</bpmn:serviceTask>
```

Best Practices for Save Points

- Before start event of process instance
- After user task
- Before invocations of external systems especially over network
- After non-transactional, non-idempotent side-effects
- After expensive external computation
- Before parallel joins like parallel and inclusive gateways or at the end of each instance of parallel multi-instance activities

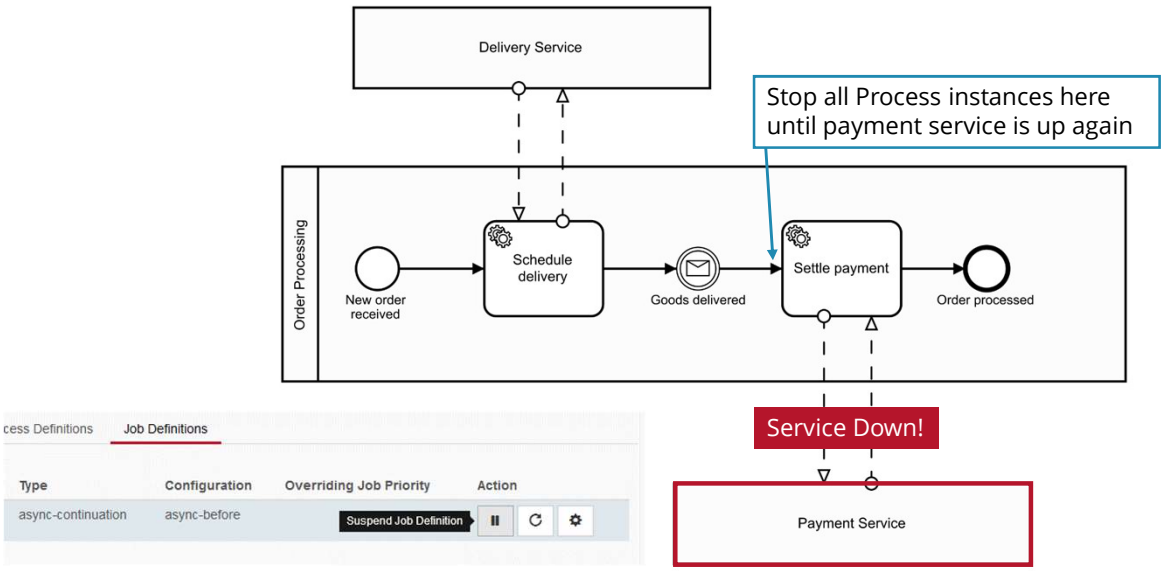
NOT

- Before tasks that wait anyway (user tasks, receive tasks, external service tasks, catching events like timer, message, signal)
- Before XOR & Event-based gateways
- Before splitting AND & OR gateways

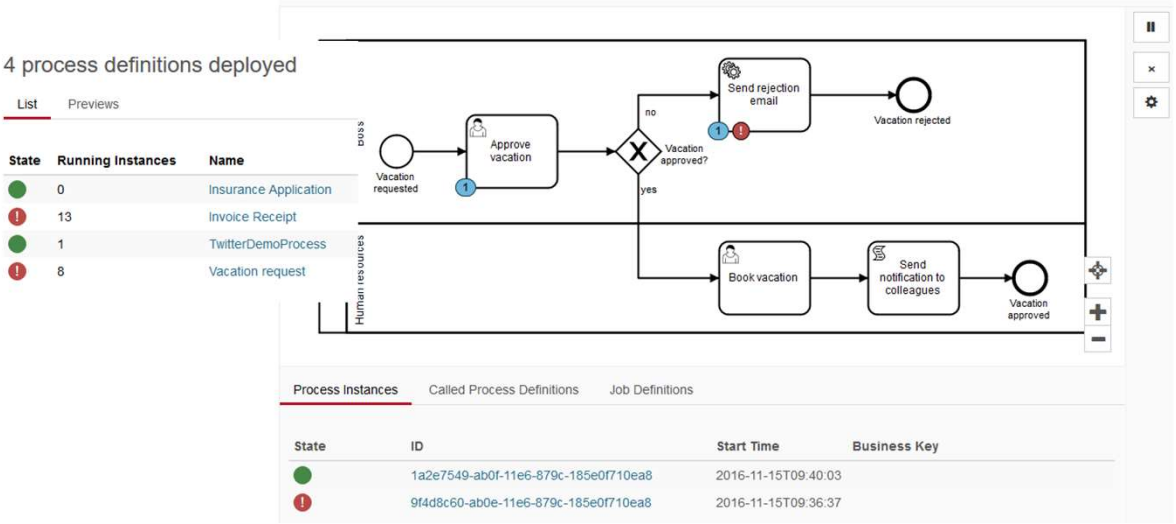
UNLESS

- Listeners, beans or complex expressions, e.g. using SPIN, that could fail are involved

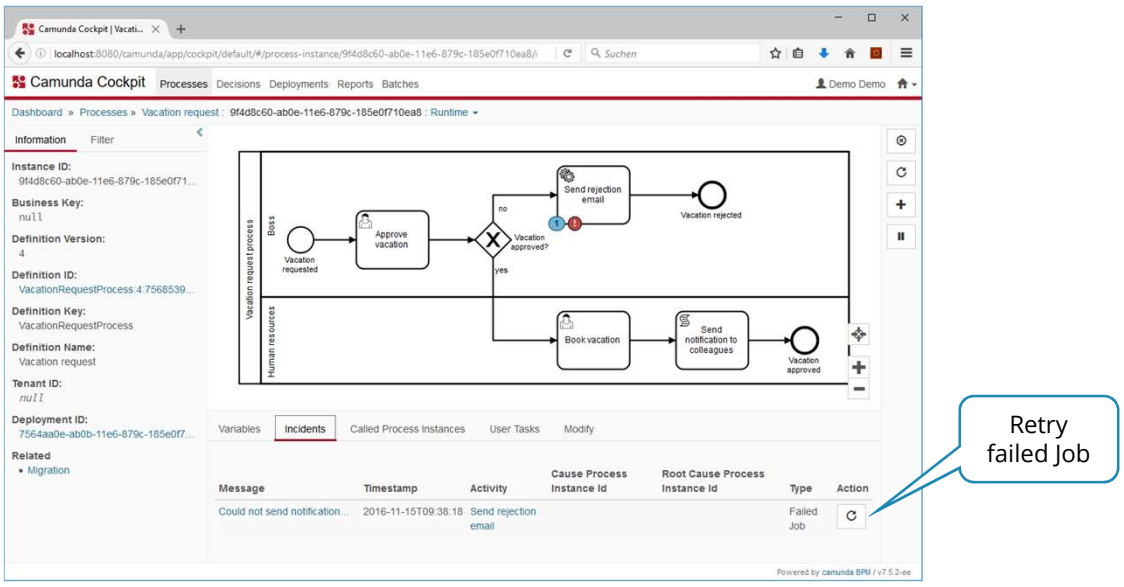
Suspension



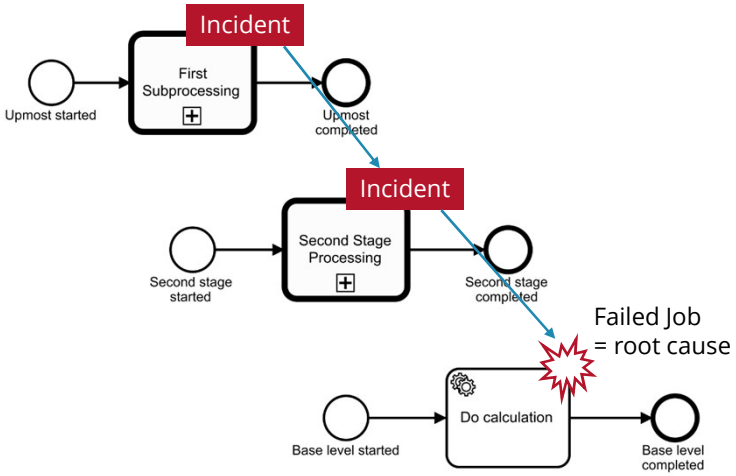
Incidents Overview



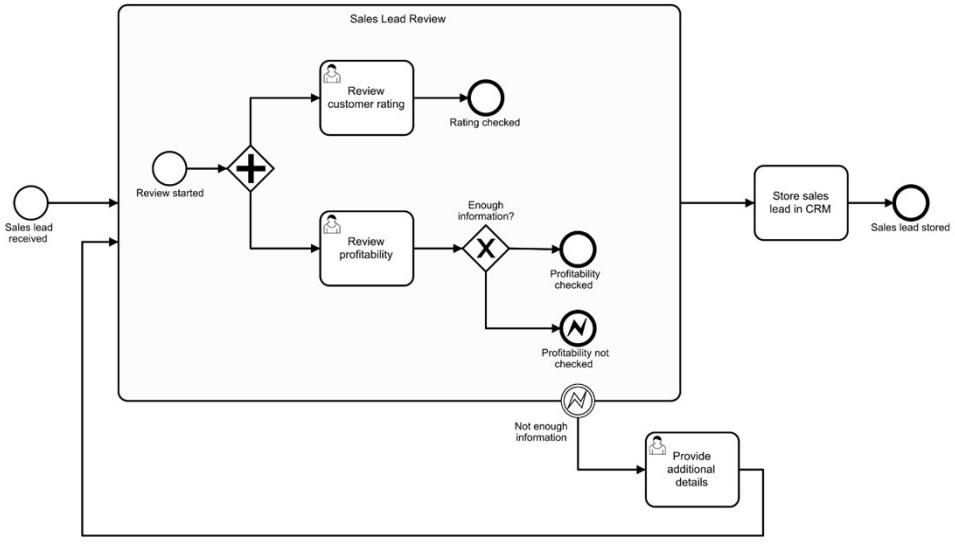
Incidents Details



Drill Down



Error Event



What if No Error Handler Is Defined?

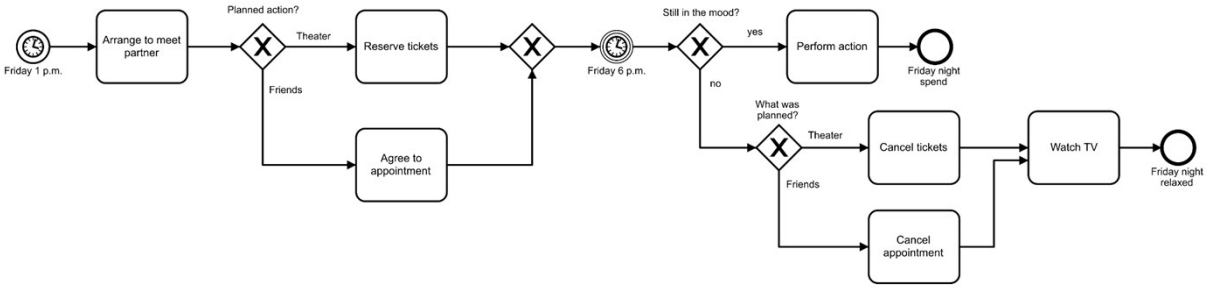
Quotes from BPMN 2.0.1:

"Error triggers are critical and **suspend execution** at the **location of throwing**."

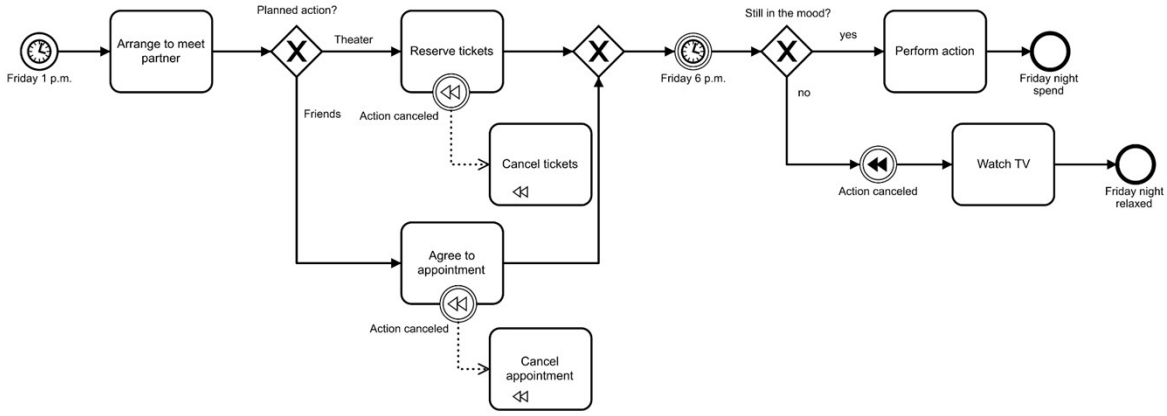
"If no catching Event is found for an error or escalation trigger, this trigger is **unresolved**."

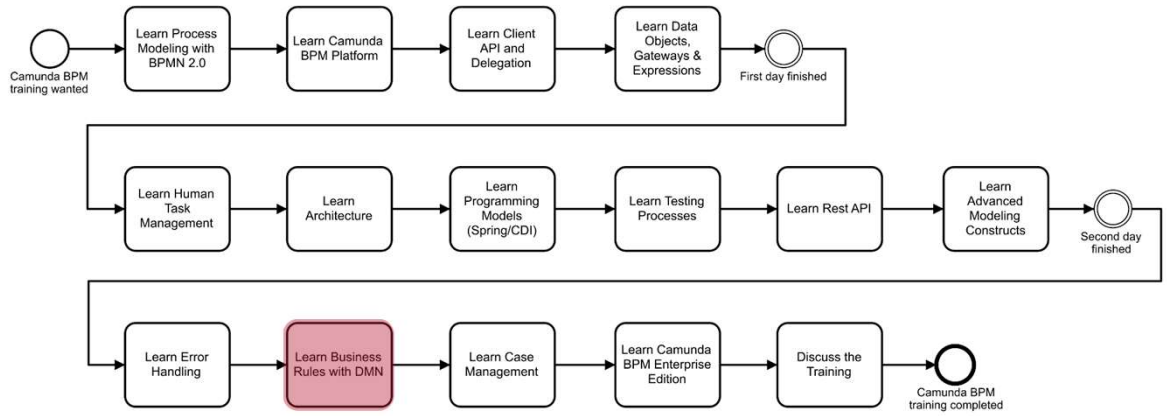
"The behavior of the Process is **unspecified** if no Activity in the hierarchy has such an Error Intermediate Event. The system executing the process can define additional Error handling in this case, a common one being **termination of the Process instance**."

Compensation

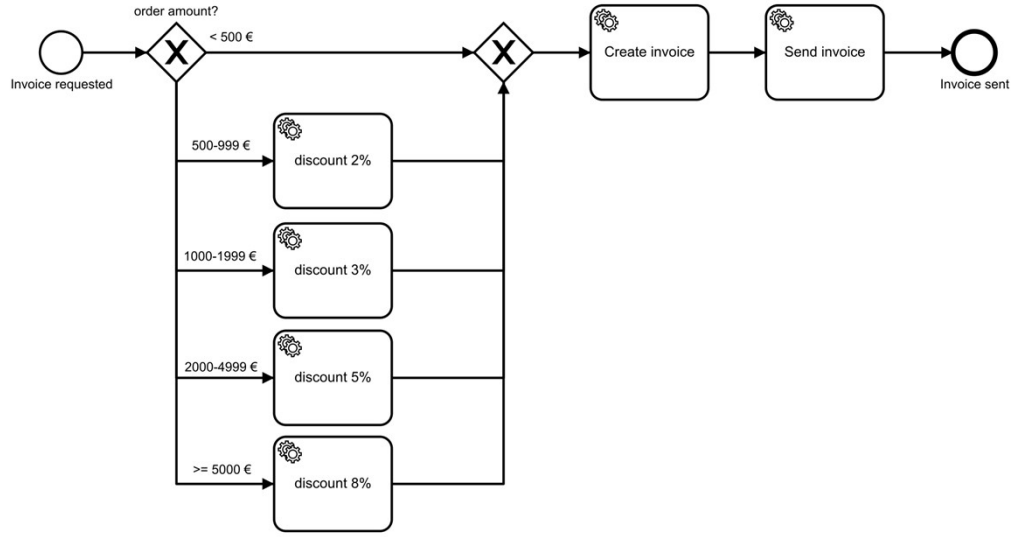


Compensation Events

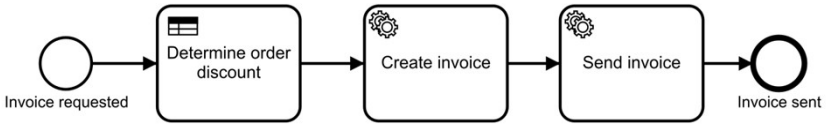




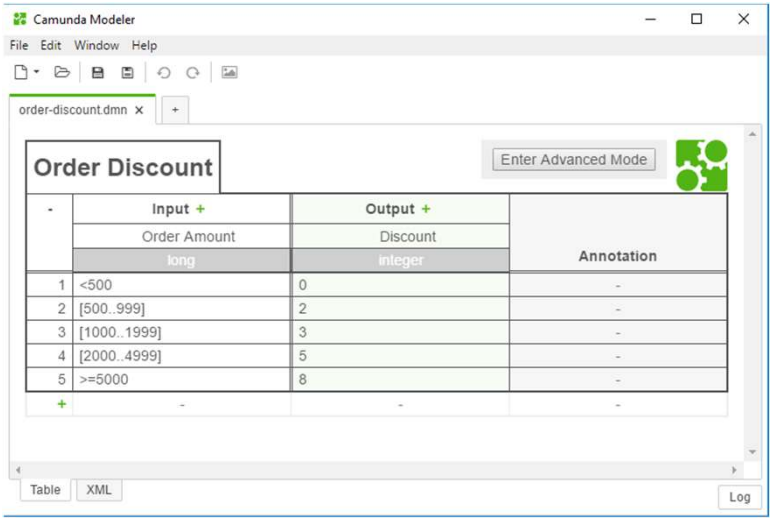
Is This a Good Process Model?



An Improved Version



Discount Rules as DMN Decision Table



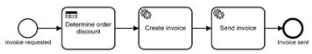
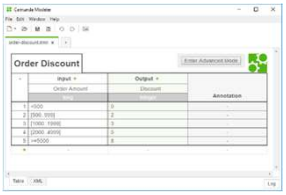
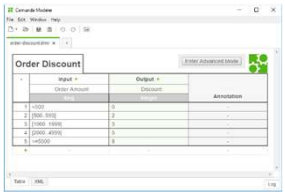
The Standard DMN

- Decision Model and Notation
- Is currently in publications by OMG for Decisions/Business Rules

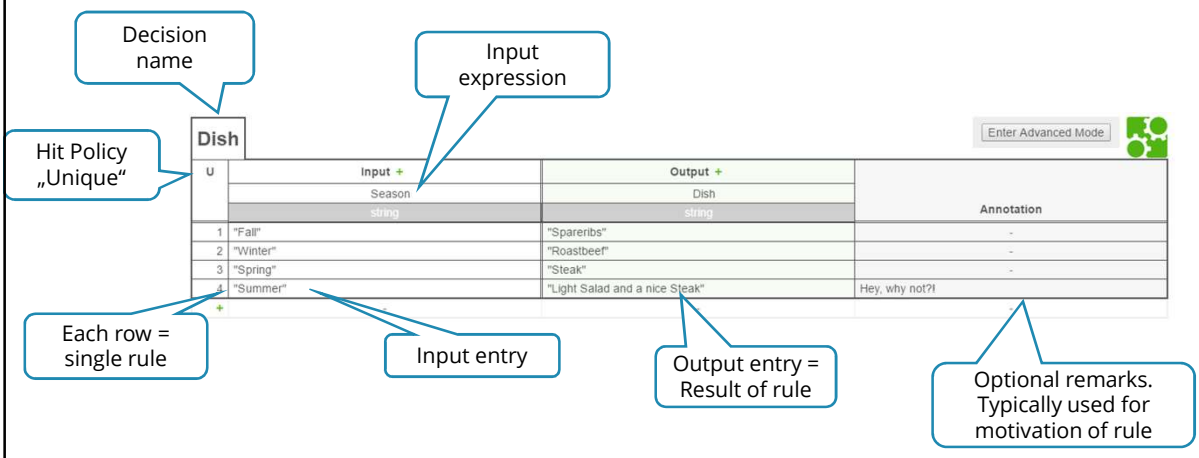
- Specification: „The primary goal of DMN is to provide a **common notation** that is readily understandable by all **business users**, from the business analysts needing to create initial decision requirements and then more detailed decision models, to the **technical developers** responsible for automating the decisions in processes, and finally, to the business people who will manage and monitor those decisions. DMN creates a standardized **bridge for the gap between the business decision design and decision implementation**. DMN notation is designed to be **useable alongside the standard BPMN** business process notation."

- Camunda shipped DMN 1.1 support with 7.4

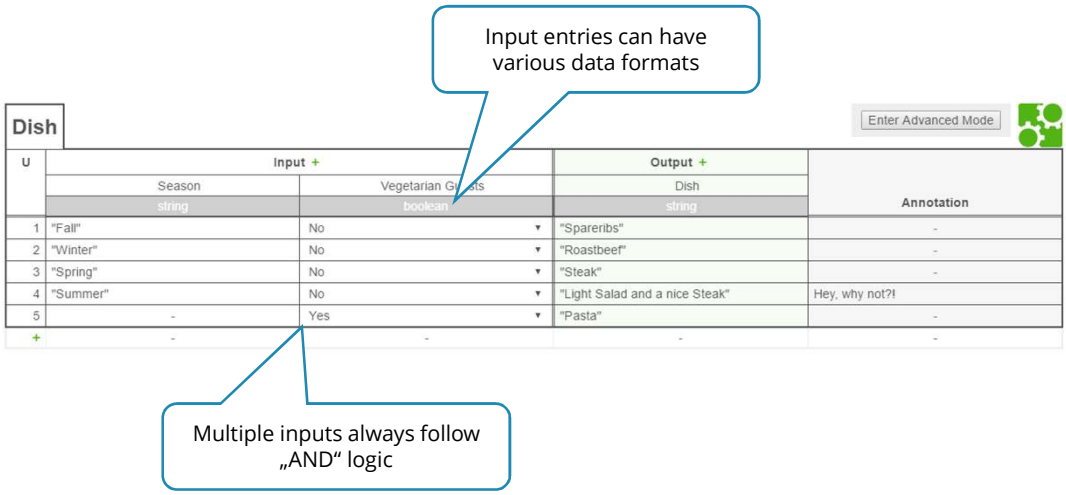
Typical Frequency of Changes

Business Processes	Rule Structure (Columns)	Rules (Rows)
		
Changes of Process Model causes effort for system integration or even organizational change	Change of the structure causes effort, as you may need additional input parameters	Changes of rules (rows in table) are very easy to do, but think about validation & testing
Less frequent (months or years)	Less frequent (months)	Very frequent (weeks)

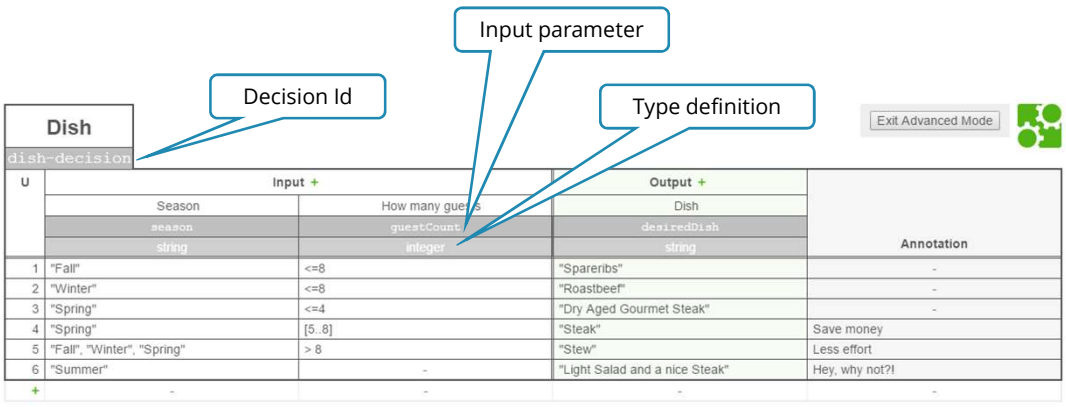
Rules Expressed as Decision Table



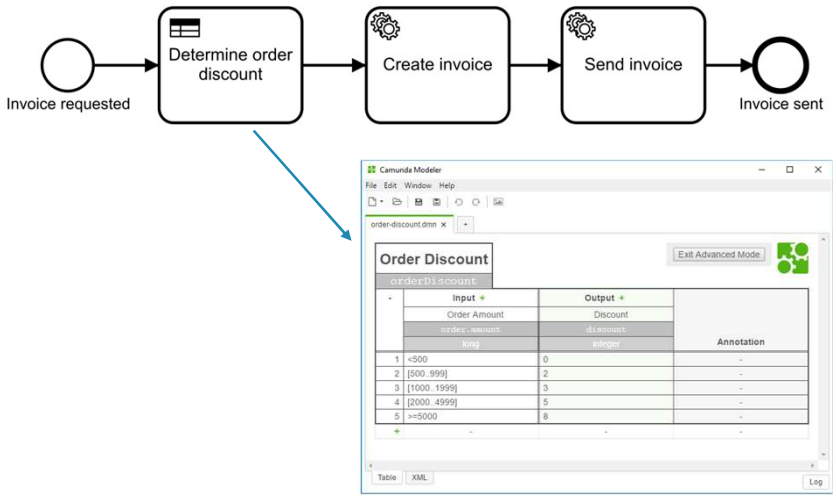
Multiple Inputs



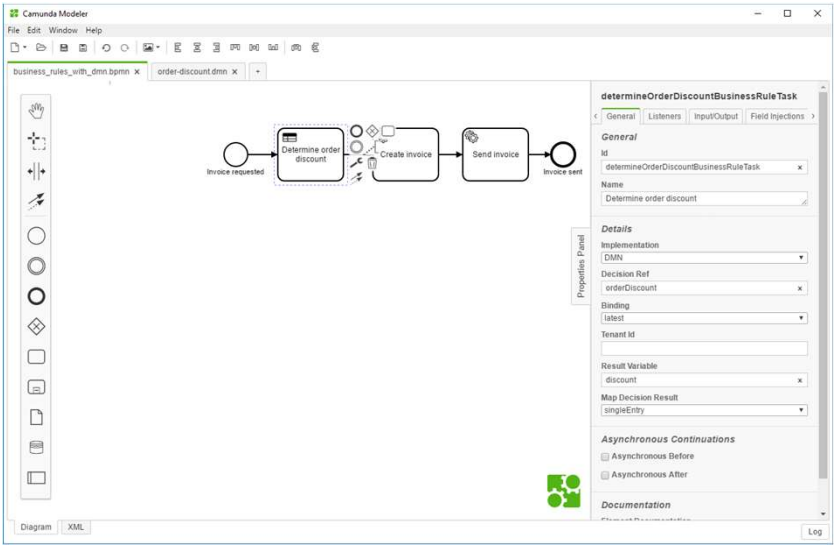
Technical Details



Business Rule Task Reference a DMN Decision Table

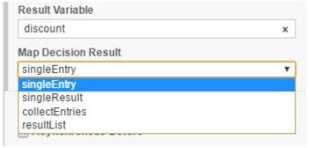


Reference in Camunda Modeler

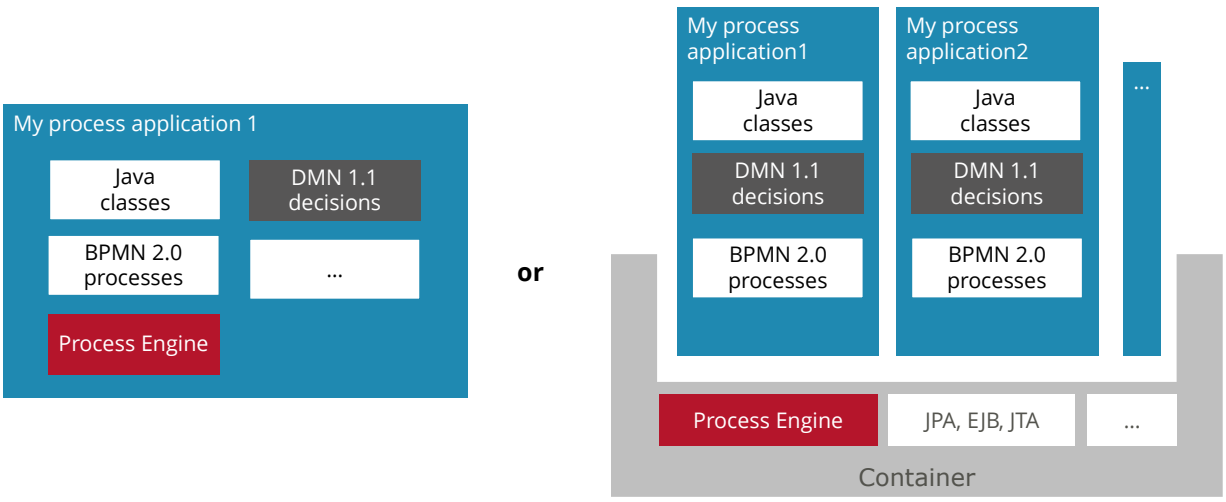


Map the Result to the Process

Mapper	Result	Is suitable for decision tables with
singleEntry	`TypedValue`	No more than one matching rule and only one output
singleResult	`Map<String, Object>`	No more than one matching rule and multiple outputs
collectEntries	`List<Object>`	Multiple matching rules and only one output
resultList	`List<Map<String, Object>>`	Multiple matching rules and multiple outputs



DMN Is Part of the BPM Platform



Standalone Stateless Decision Engine

- You can use the core DMN engine as stateless Rule Engine separately from the BPM Platform if you like
- Single Maven Dependency:

```
<dependency>
  <groupId>org.camunda.bpm.dmn</groupId>
  <artifactId>camunda-engine-dmn</artifactId>
</dependency>
```


Standalone (Stateless) Decision Engine

```
// create configuration
DmnEngineConfiguration configuration = DmnEngineConfiguration
    .createDefaultDmnEngineConfiguration();

// configure if needed...

// build engine
DmnEngine dmnEngine = configuration.buildEngine();

// load DMN file
InputStream inputStream = ...;

//create and add variables
VariableMap variables = Variables.createVariables();

// create decision
DmnDecision decision = dmnEngine.parseDecision("decision", inputStream);

// evaluate decision
DmnDecisionTableResult tableResult = dmnEngine
    .evaluateDecisionTable(decision, variables);
DmnDecisionRuleResult ruleResult = tableResult.getSingleResult();
String firstEntry = ruleResult.getEntry("desiredDish");
```

Decision Service in the Platform

```
DmnDecisionTableResult result  = processEngine
    .getDecisionService()
    .evaluateDecisionTableByKey(decisionDefinitionKey, variables);
```

Also available as REST-Service

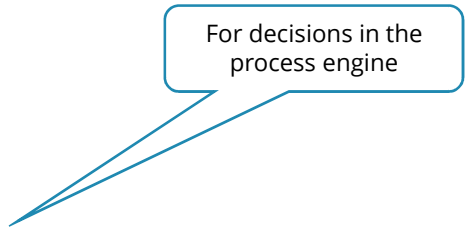
```
POST /decision-definition/key/aDecisionDefinitionKey/evaluate

Request body:
{
  "variables" : {
    "amount" : { "value" : 600, "type" : "Double" },
    "invoiceCategory" : { "value" : "Misc", "type" : "String" }
  }
}

Response:
[
  {
    "result": { "value" : "management", "type" : "String", "valueInfo" : null }
  }
]
```


JUnit for Decisions

For embedded decision engine

```
@Test
public void test() {
    DmnEngine dmnEngine = dmnEngineRule.getDmnEngine();

    // load DMN file
    InputStream inputStream = ...;

    // create decision
    DmnDecision decision = dmnEngine.parseDecision("decision", inputStream);

    // evaluate decision
    DmnDecisionTableResult tableResult = dmnEngine
        .evaluateDecisionTable(
            decision,
            withVariables(key, value, furtherKeyValuePairs));
    assertThat(tableResult.getSingleResult()).containsEntry(key, expectedValue);
}
```

JUnit for Decisions




```
@Test
@Deployment(resources = {"myRule.dmn"})
public void testWithProcessEngine() {
  Map<String, Object> variables = withVariables("orderAmount", 1999.90);
  DmnDecisionTableResult result = processEngine()
      .getDecisionService()
      .evaluateDecisionTableByKey("my_decision", variables);
  assertThat(result.getFirstResult()).containsEntry("discount", 5);
}
```


Hit Policies

- Single
  - **U**(nique): Only one rule can match
  - **A**(ny): Multiple rules may match – must have same output
  - ***P***(riority): Rule with highest priority (output field) is selected
  - **F**(irst): First matching rule (order in table!) is selected
- Multiple
  - ***O***(utput Order): List of rules in order of priority (output field)
  - **R**(ule Order): List of rules in order of table
  - **C**(ollect): List of all hits without order, might be combined with operator **+** (sum), **<** (min), **>** (max), **#** (count)

Italic hit policies are not yet supported by the Camunda DMN engine

Examples From the Spec

Applicant Risk Rating

U	Applicant Age	Medical History	Applicant Risk Rating
1	> 60	*good*	*Medium*
2		*bad*	*High*
3	[25..60]	-	*Medium*
4	< 25	*good*	*Low*
5		*bad*	*Medium*

Student Financial Package Eligibility

R	Student GPA	Student Extra-Curricular Activities Count	Student National Honor Society Membership	Student Financial Package Eligibility List
1	> 3.5	>= 4	*Yes*	20% Scholarship
2	> 3.0	-	*Yes*	30% Loan
3	> 3.0	>= 2	*No*	20% Work-On-Campus
4	<= 3.0	-	-	5% Work-On-Campus

Friendly Enough Expression Language (FEEL)

[date and time(„2016-12-24T00:00:00“)
..
date and time(„2016-12-26T23:59:99“)]

Support for different „endpoint“ data types: number, string, boolean, time, date, date-time, time-duration.

examplesSimpleExpressions

F	Input +				Output +	Annotation
	Season	Date	Number of guests	Children	Dish	
	season	date	guestCount	children	dish	
	string	date	integer	boolean	string	
1	-	[date and time("2016-12-24T00:00:00")..date and time("2016-12-26T23:59:99")]	-	-	"Christmas Dinner"	-
2	"Winter"	-	-	true	"Vegetables"	-
3	"Winter"	-	<=6	-	"Steak"	-
4	"Winter"	-	]6..10[	-	"Pizza"	Same as (6..10)
5	"Winter"	-	[10..12]	-	"Huge Pizza"	-
6	"Winter"	-	12,[13..14],>14	-	"Family Deluxe Pizza"	-
7	"Spring", "Summer", "Fall"	-		true	"Salad"	-
8	not("Winter")	-				-
9	not("Spring","Summer","Fall")					It is Winter again ;-)
+	-					-

Support for Comparison (<, <=, >, >=) and Ranges ([x..y]):
- Start included: [
- Start excluded:] or (
- End included:]
- End excluded: [or)

Negation

JUEL as Expression Language

Tweet approval

approveTweet

F	Input +	Output +
	Tweet Conditions	-
	-	-
	boolean	boolean
1	lastTweet.content == currentTweet.content	false
2	currentTweet.content.contains('#ibm')	false
3	currentTweet.content.matches('someRegEx')	
4	-	
+	-	-

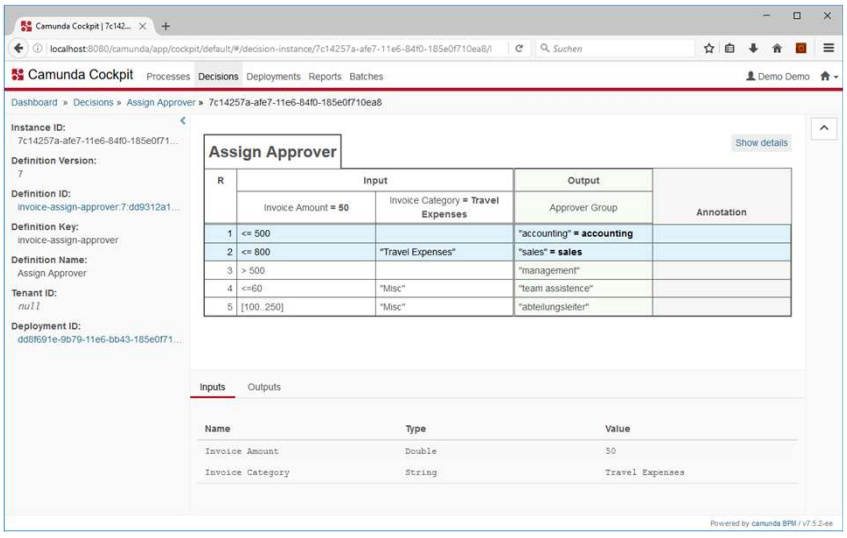
Rule
+ add
+ copy
− remove
clear

Input
+ add
− remove

Add description

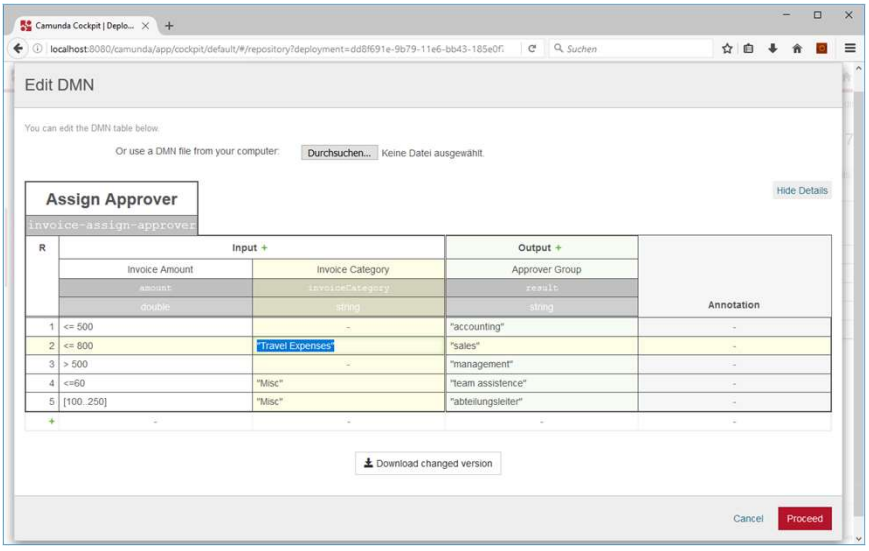
Expression Language
juel

Cockpit for Decisions



Enterprise Feature

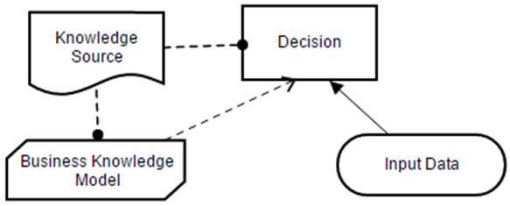
Live Editing of Decision Tables



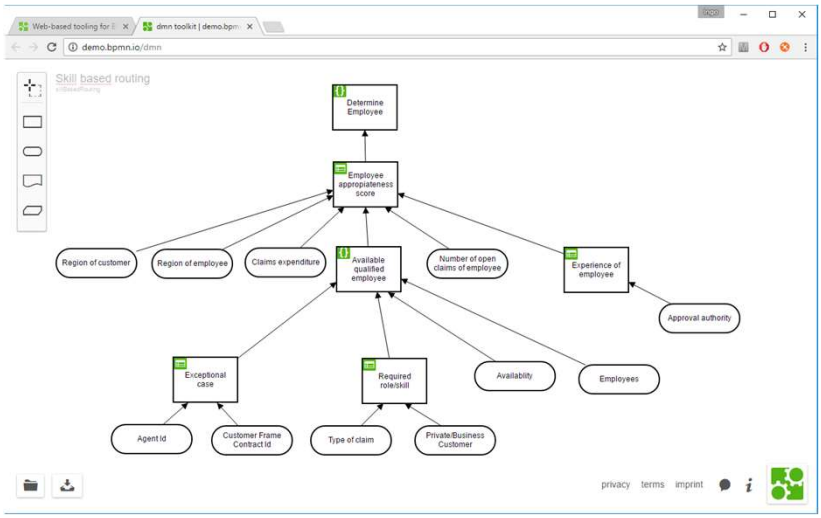
R	Input +		Output +	
	Invoice Amount	Invoice Category	Approver Group	
	amount	invoiceCategory	result	
	double	string	string	Annotation
1	<= 500	-	"accounting"	-
2	<= 800	"Travel Expenses"	"sales"	-
3	> 500	-	"management"	-
4	<=60	"Misc"	"team assistence"	-
5	[100..250]	"Misc"	"abteilungsleiter"	-
+	-	-	-	-

Beyond (Or Before) Decision Tables

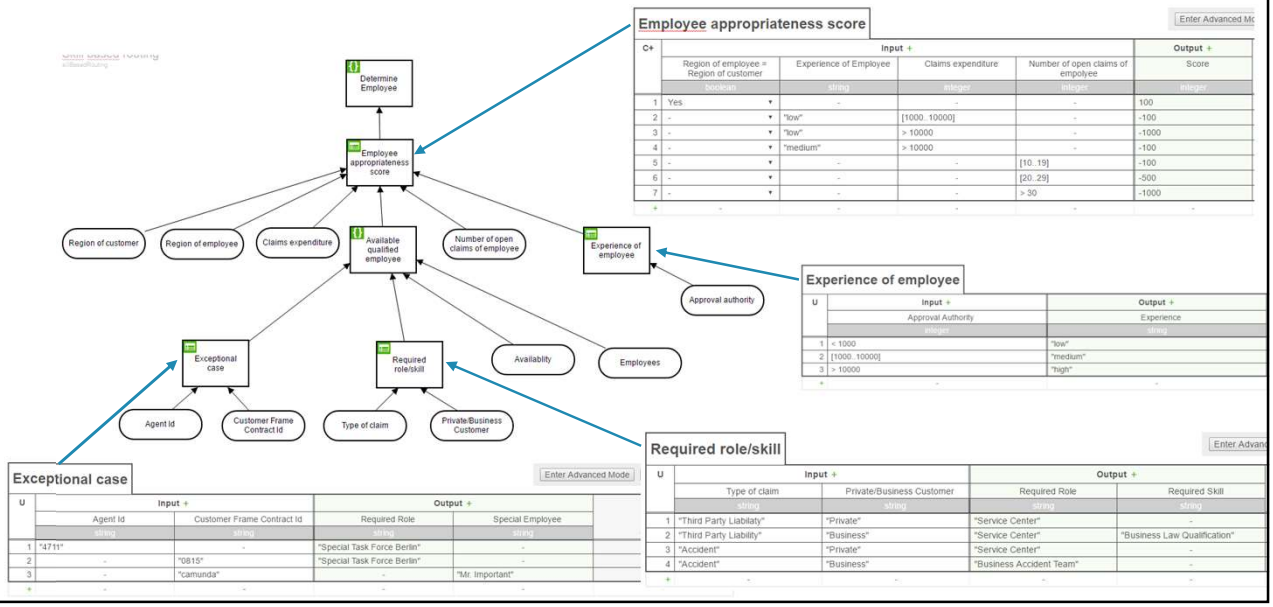
- DMN defines „Decision Requirements Diagram“
- Decision Requirements Diagrams will define the decisions [...], their interrelationships, and their requirements for decision logic
- Starting point for modeling complex decisions
- Elements of Decision Requirements Diagram



DRD Example: Skill Based Routing

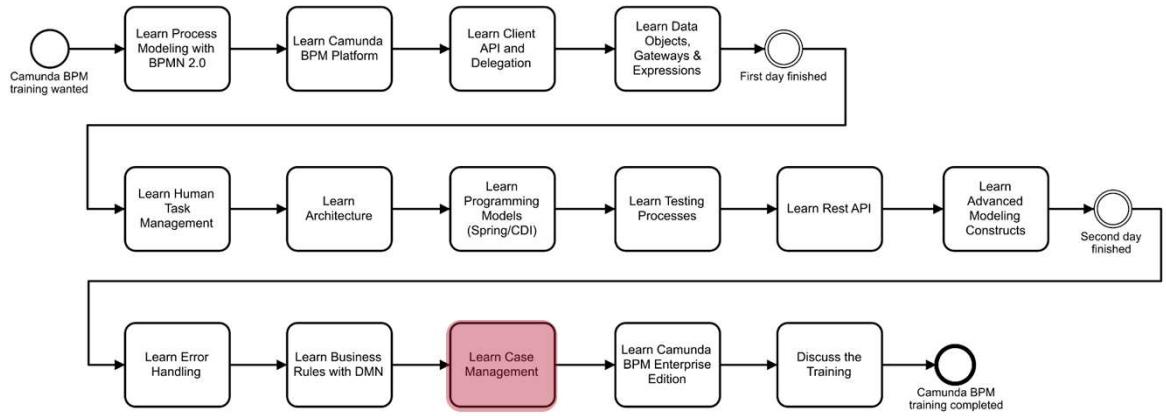


Decisions Reference Decision Tables

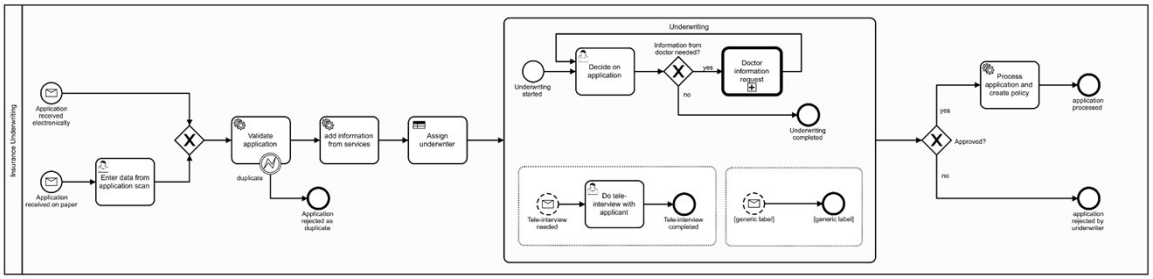


Exercise 9

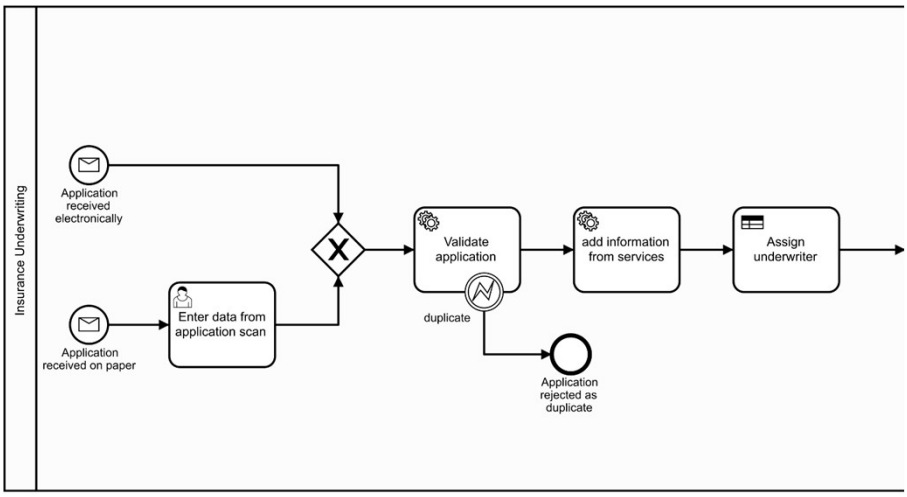
Follow the provided instructions for exercise 9



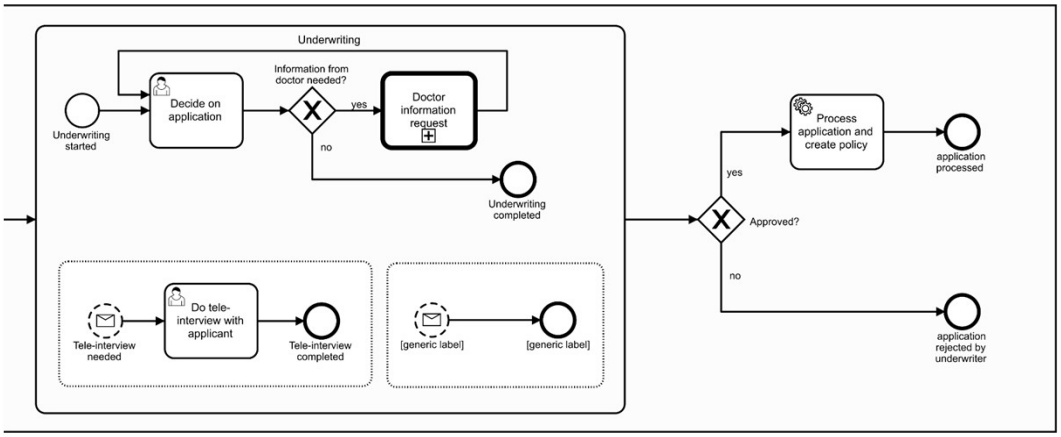
Example „Underwriting“



Example „Underwriting“

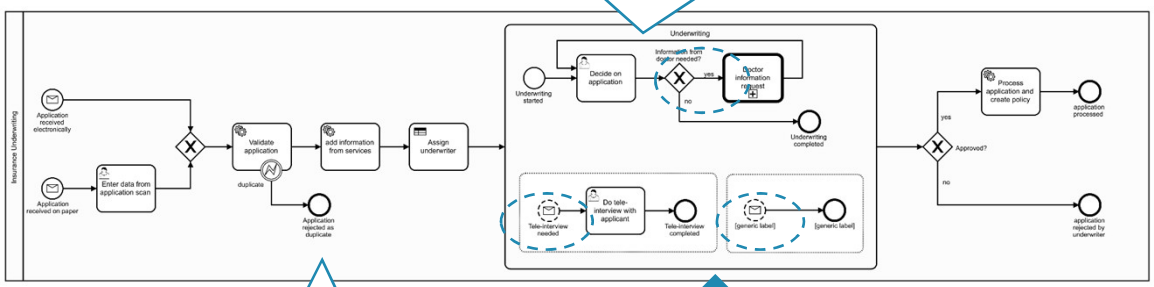


Example „Underwriting“



Example „Underwriting“

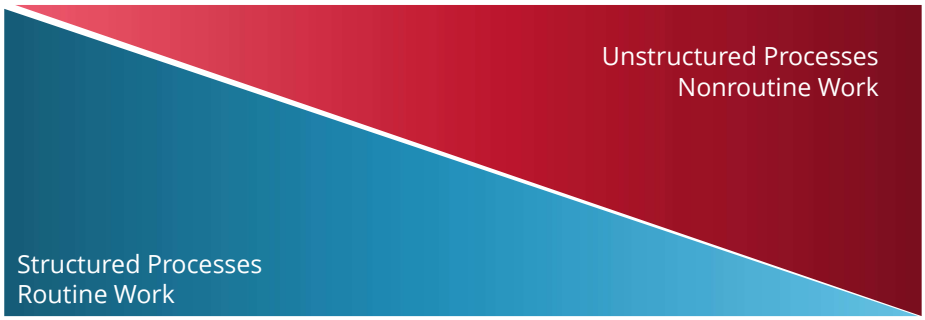
It is hard to give a stable ordering of the tasks! Flexibility needed.

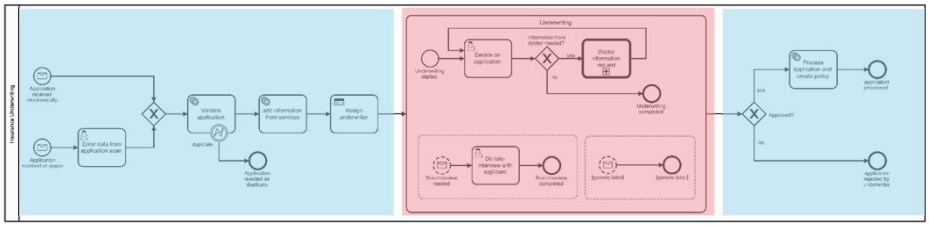


100 % standardized – no flexibility needed

There are workarounds – but this is hard to model in BPMN. You need more flexibility for the „Knowledge Worker“

Structured vs. Unstructured Work



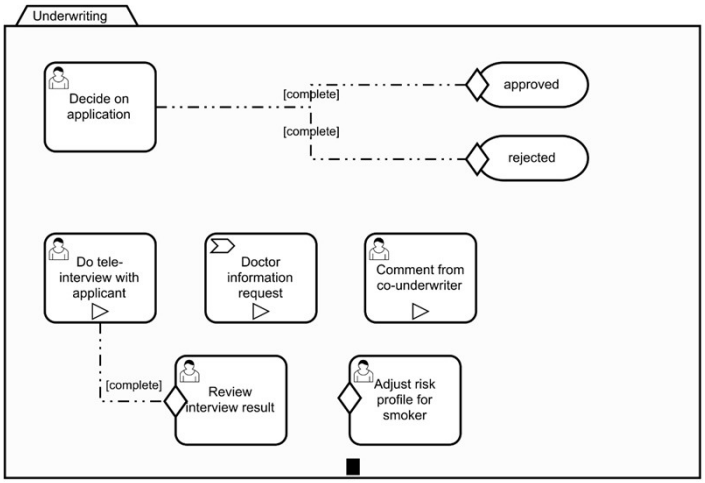


CMMN – Case Management Model and Notation

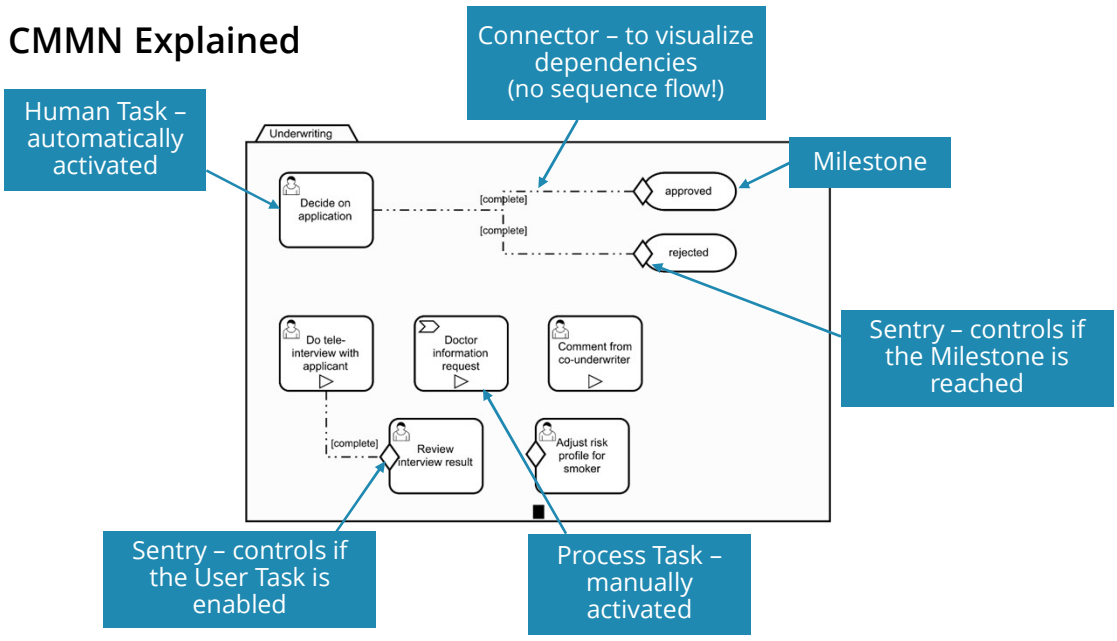
- First published by OMG in May 2014 as the BPMN-equivalent for „Cases“
- „Activities that are not so predefined and repeatable, but instead depend on evolving circumstances and ad hoc decisions by knowledge workers regarding a particular situation”

- Typical Use Cases:
  - application and claim processing in insurance
  - patient care and medical diagnosis in healthcare
  - exception handling, e.g. problem resolution in call centers, invoice discrepancy handling, data validation errors, ...
- See http://www.omg.org/spec/CMMN/1.1/

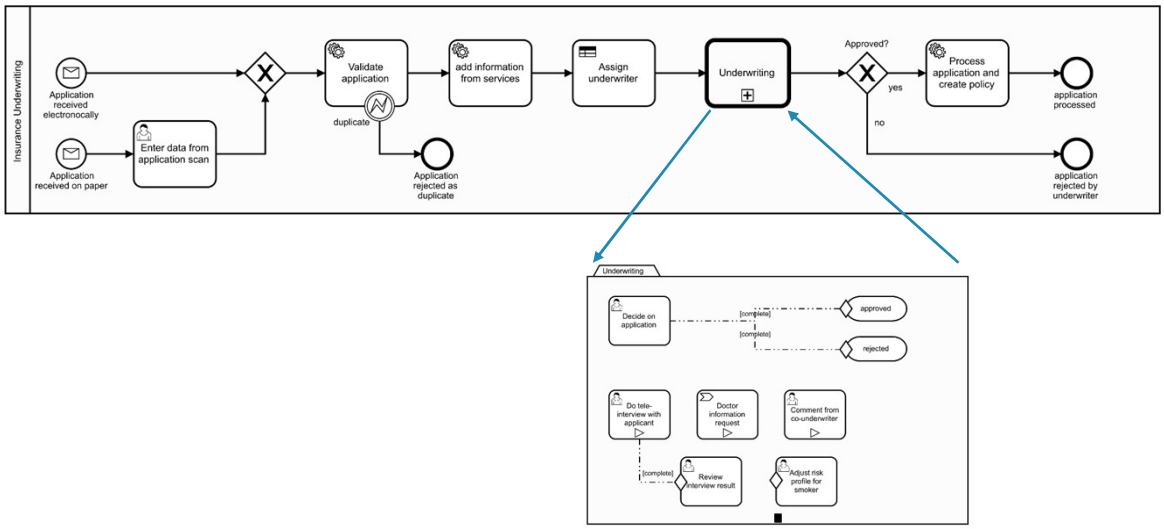
CMMN as Alternative



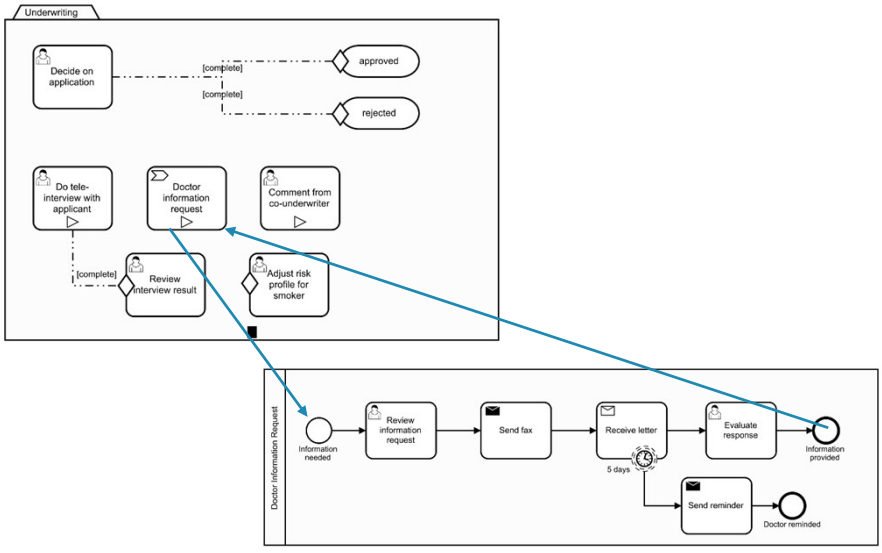
CMMN Explained



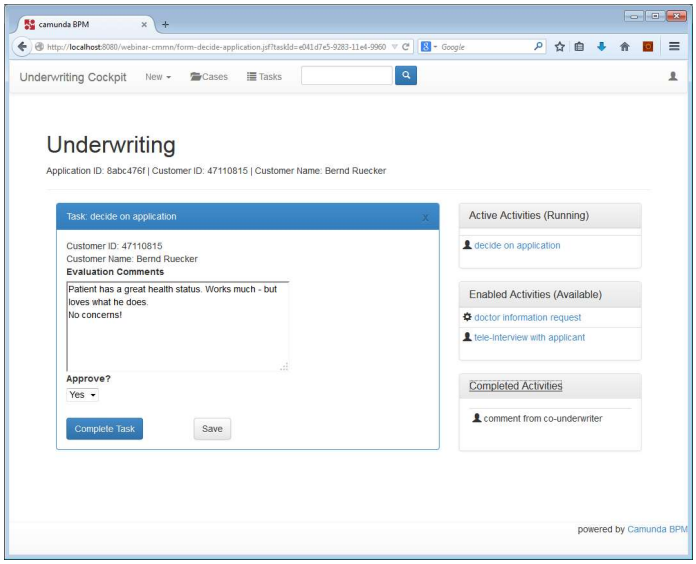
Stable and Easier Process



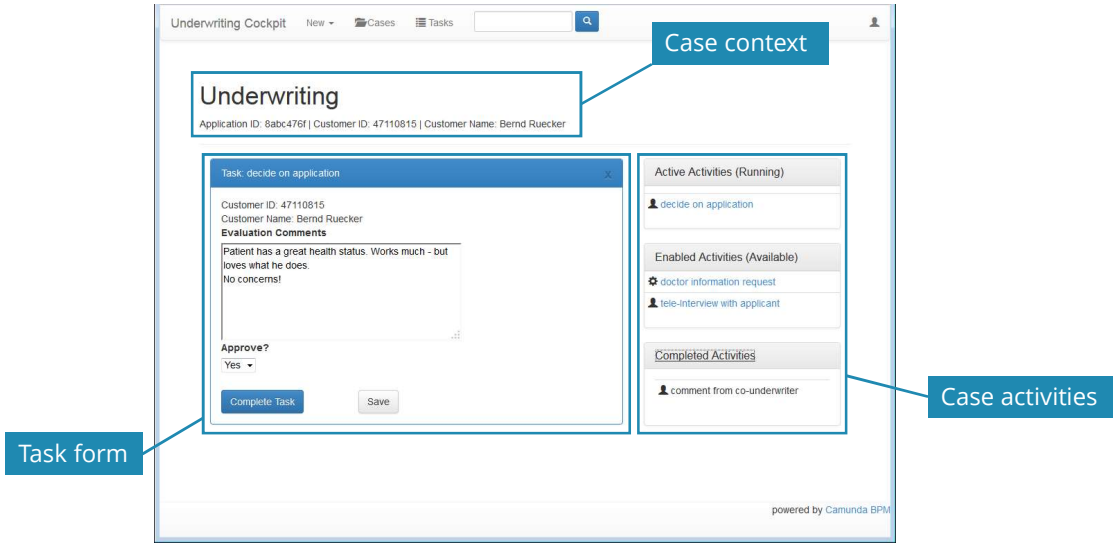
CMMN Calls BPMN



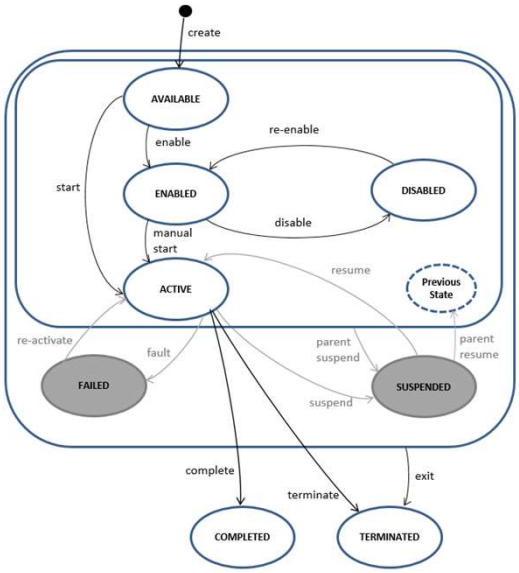
Case UI



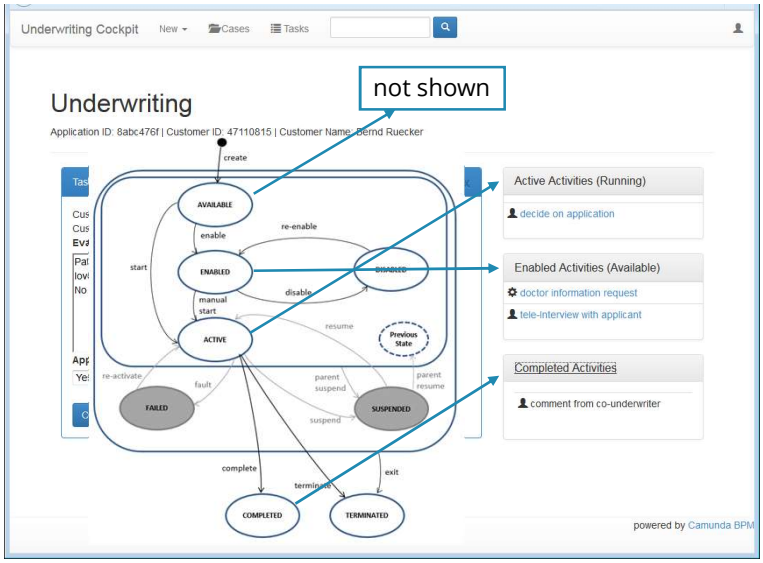
Example for Case Instance



Task Lifecycle

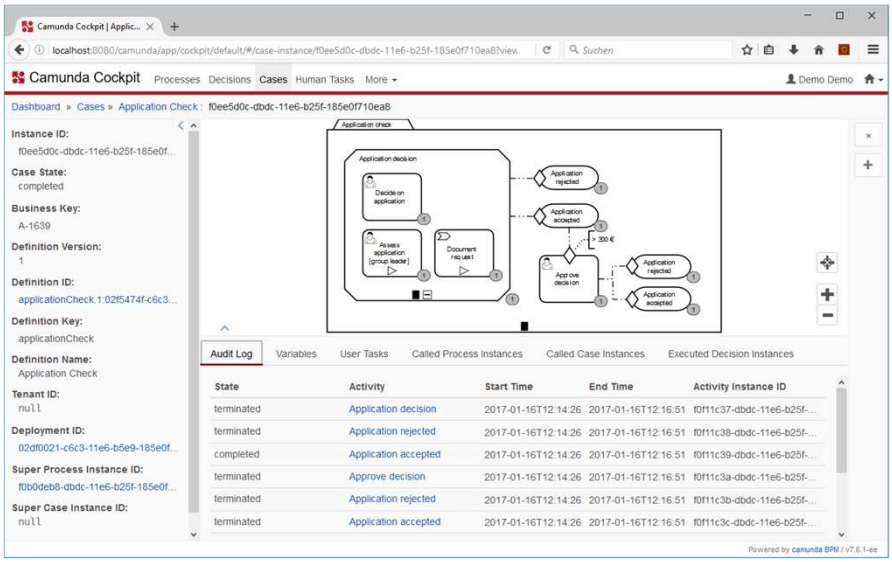


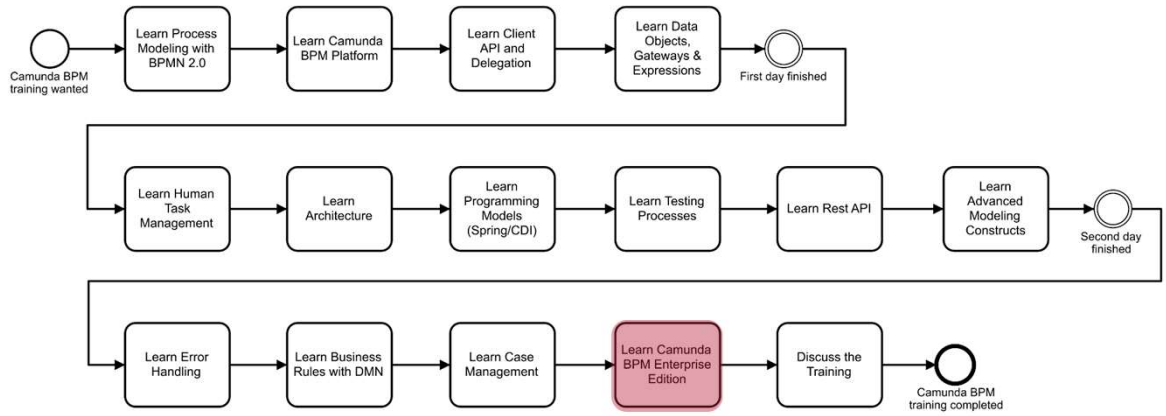
Lifecycle Mapped on UI



Enterprise Feature

Cockpit for Cases





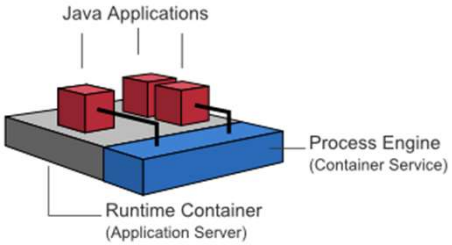
Enterprise Exclusive Cockpit Features

- Process Definition History
- Process Definition Heatmap
- Process Instance History
- Process Instance Migration
- Process Instance Modification
- Bulk Retry and Bulk Cancel
- Decision Table Live Editing
- Process Duration Report
- Redeploy Definitions

https://camunda.com/bpm/enterprise/#cockpit

Runtime Support

- Integration with IBM WAS
- Integration with Oracle WLS



https://camunda.com/bpm/enterprise/#as

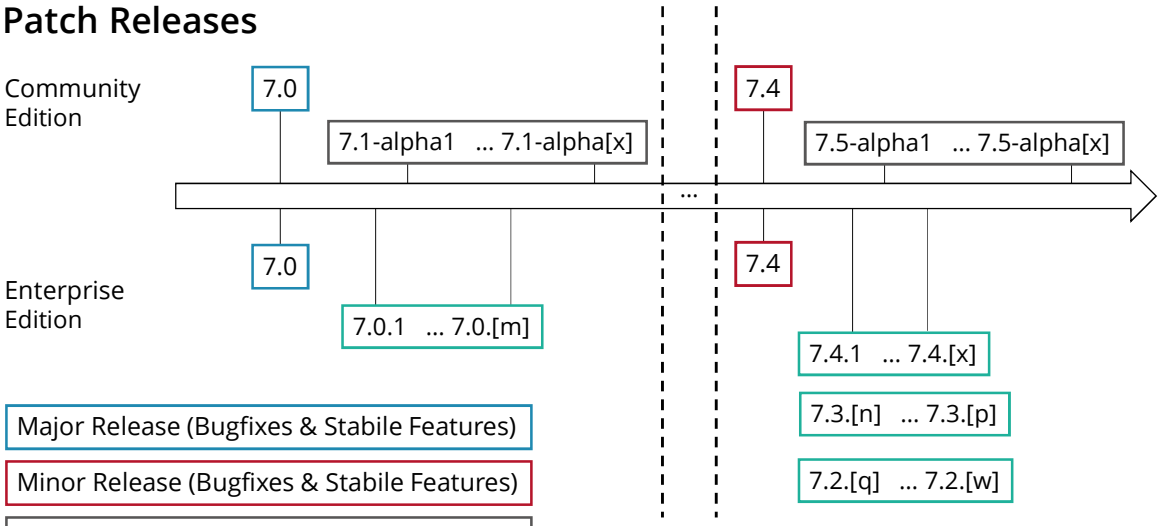
SLA Based Support

- Reliable support infrastructure with direct contact to core developers
- Service Level Agreements (SLA)

Level	Name	Description	Availability		Max. response time	
			Standard SLA	Advanced SLA	Standard SLA	Advanced SLA
L1	Blocker	Core components (i.e., process engine) of Camunda BPM do not work at all/produce critical errors that prevent usage in production mode.	8x5	24x7	8 business hours	2 hours
L2	Critical	Usage of Camunda BPM seriously affected, a workaround is urgently needed.	8x5		8 business hours	
L3	Help Request	Non-critical errors, Help Requests, Feature Requests.	8x5		16 business hours	

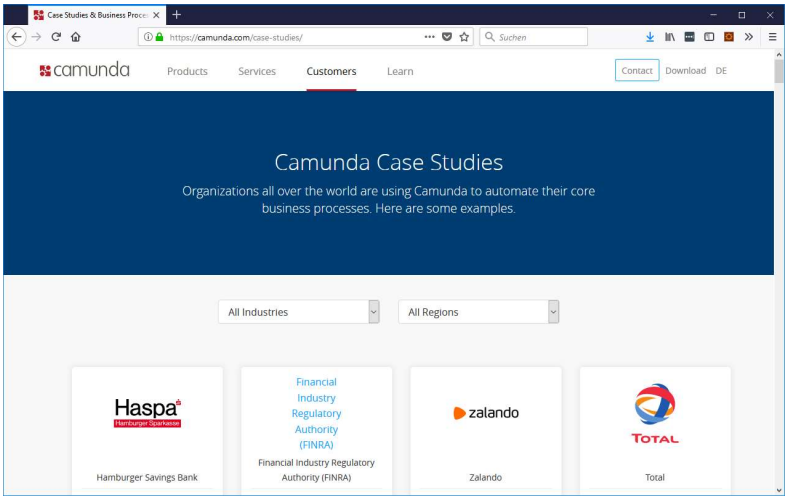
https://camunda.com/bpm/enterprise/#support

Patch Releases

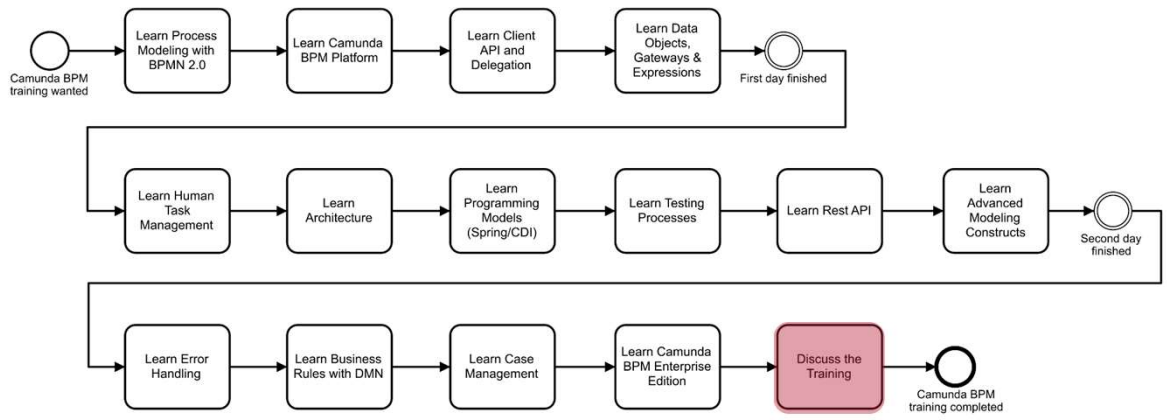


https://camunda.com/bpm/enterprise/#patch

References



https://camunda.com/case-studies/



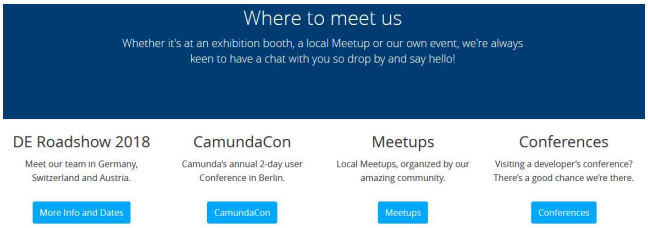
How to Get Updates From the Team?

Blog: https://blog.camunda.org/

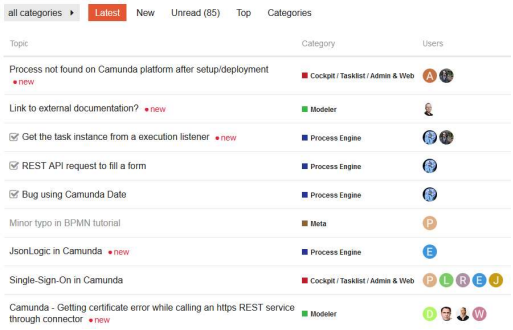
Camunda Team Blog

Whitepapers: https://camunda.com/learn/whitepapers/

Events: https://camunda.com/events/



Forum: https://forum.camunda.org/



Follow us on Twitter: @camundabpm

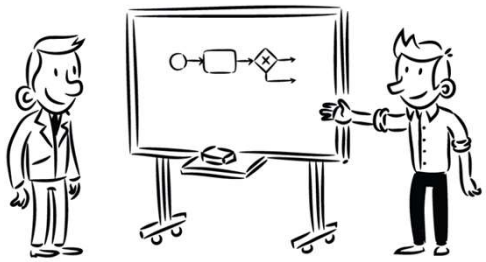
Camunda Training Offering

Overview trainings	Modeling trainings	Development trainings
BPM-trio overview (2 days) → Values of BPM → Introduction into BPMN, CMMN, DMN → Benefits of the BPM trio → Outlook: Process automation	**BPMN (3 days)** → BPMN in detail → Implementing BPM-projects with BPMN → BPMN real world examples → BPMN in context of CMMN & DMN	**Camunda BPM for developers (3 days)** → Introduction BPMN → Camunda BPM Architecture → Automating processes with Camunda → Testing, deployment, versioning
Camunda BPM Overview (1 day) → Introduction into BPMN → Hands on Camunda BPM → Monitoring & Reporting → How to start with Camunda	**DMN (1 day)** → DMN in detail → Decision design → Complex decisions with DRDs → DMN in context of BPMN & CMMN	**Camunda BPM DevOps (2 days)** → Camunda BPM installation → Monitoring & alarming → Process version migration → Advanced DevOps topics
OCEB training (2 days) → Prepare for the OCEB BPM certification	**CMMN (1 day)** → Why CMMN? → CMMN in detail → CMMN real world examples → CMMN in context of BPMN & DMN	

Details see
https://camunda.com/services/training/

Want to Learn More BPMN?

- 3-day training
- Learn in-depth BPMN 2.0
  - Understand BPMN
  - Apply BPMN
  - Introduce BPMN



Feedback

Please fill out this form:

https://camunda.com/services/training/feedback/