camunda

Camunda BPM for Developers

Getting started with the Camunda BPM Platform

1

Agenda

Day 1

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

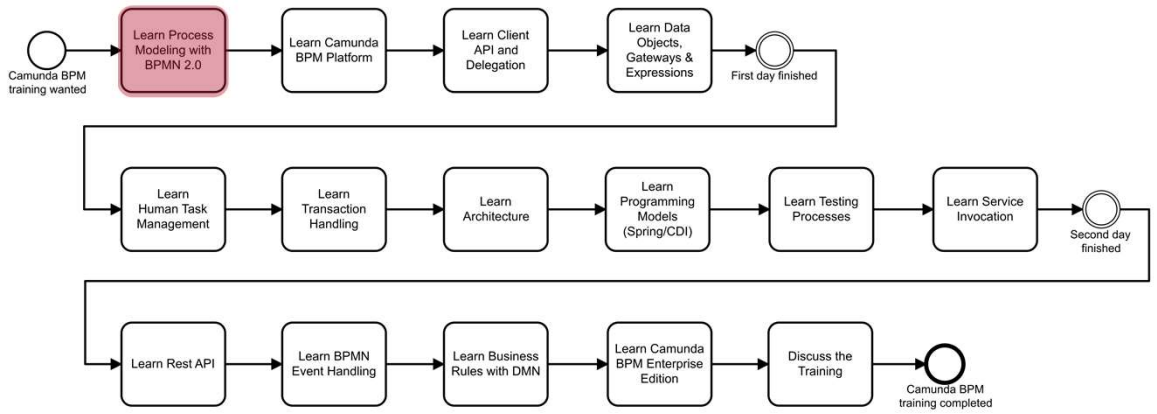
Day 2

5. Human Task Management
6. Transaction Handling
7. Architecture
8. Programming Models (Spring/CDI)
9. Testing Processes
10. Service Invocation

Day 3

11. REST API
12. BPMN Event Handling
13. Business Rules with DMN
14. Camunda BPM Enterprise Edition
15. Wrap up: Outlook, Feedback and Open Discussion

2

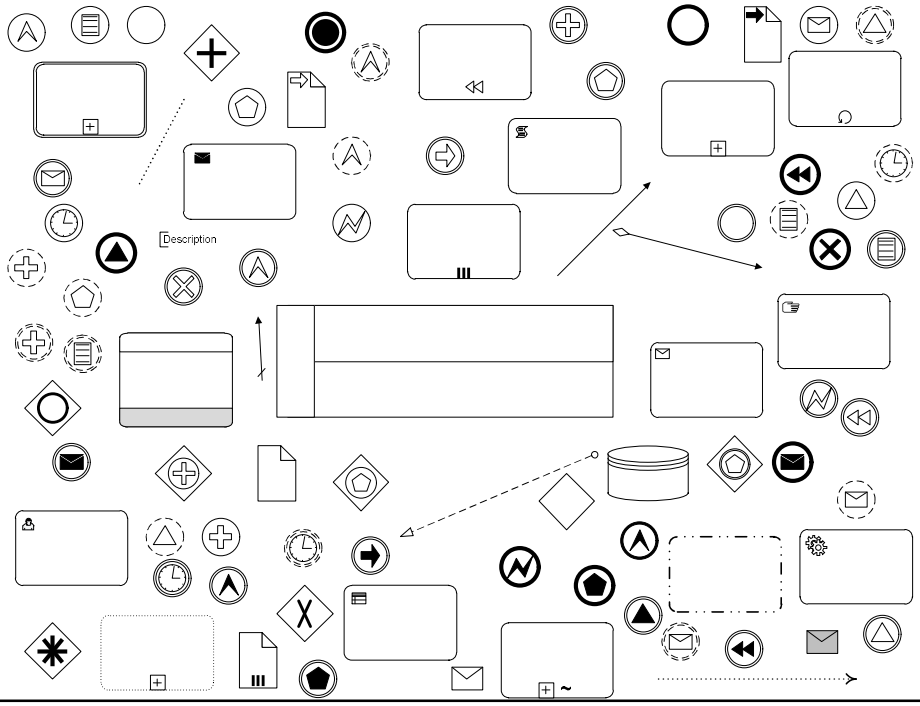


Business Process Model and Notation (BPMN)

- 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

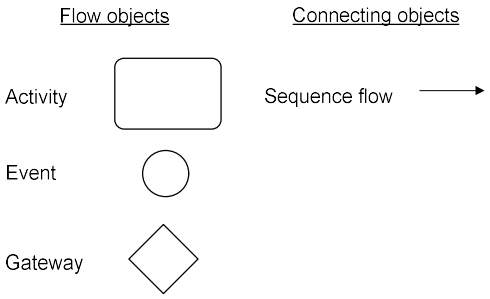
OBJECT MANAGEMENT GROUP

ORACLE SAP IBM camunda the business process company trisotech INTALIO NIST Global360 TIBCO The Power of Now FUJITSU

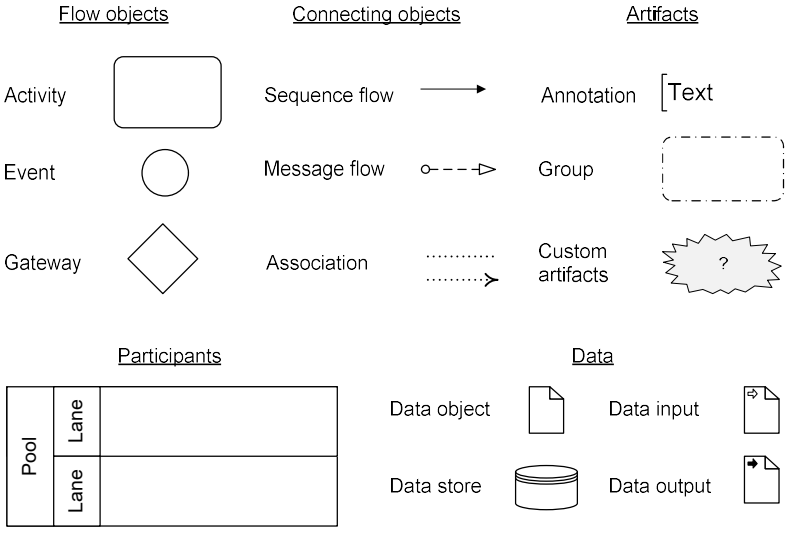


Our First BPMN Process Model

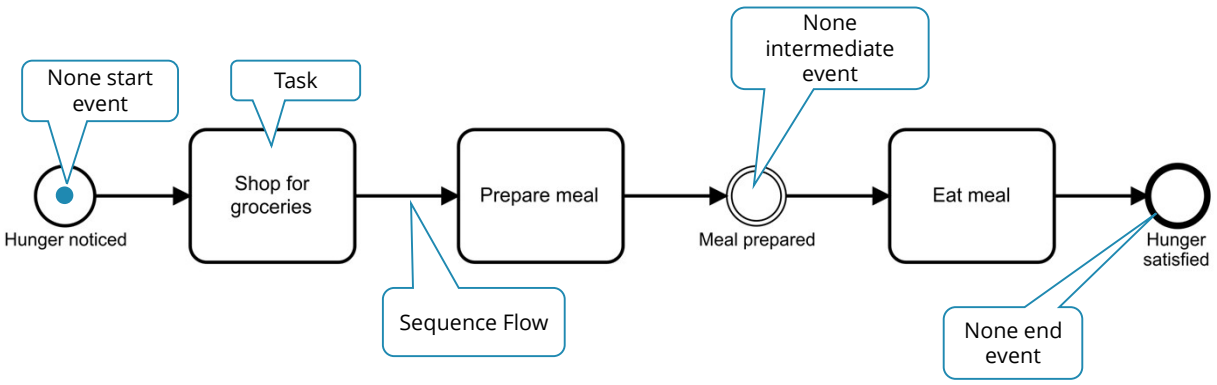
Basic Elements



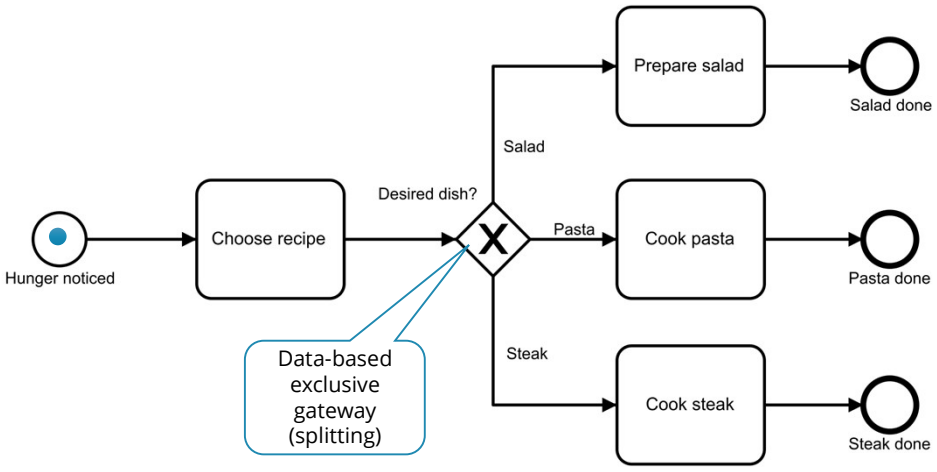
Basic Elements - Documentation

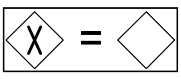


Tasks and events

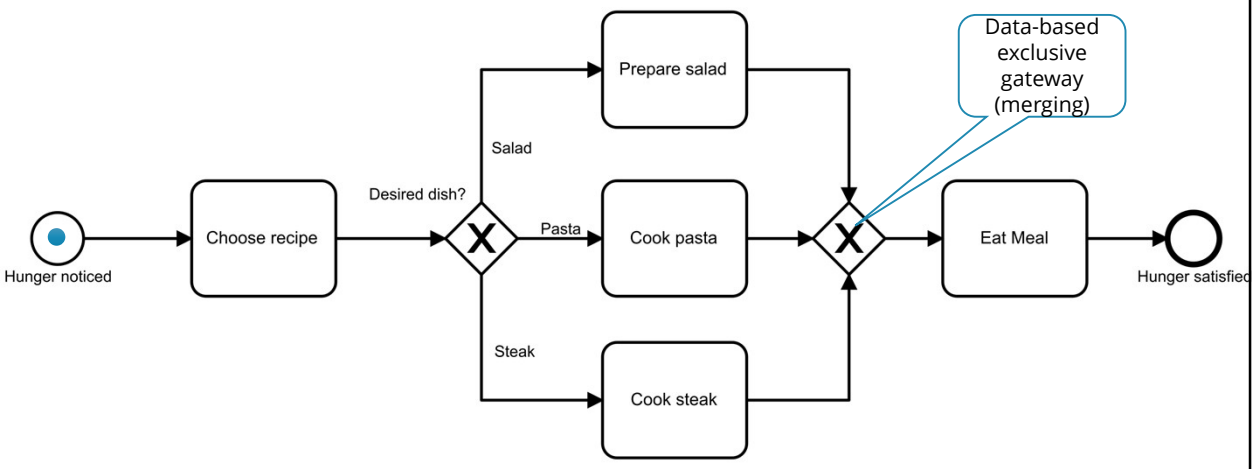


The XOR-gateway

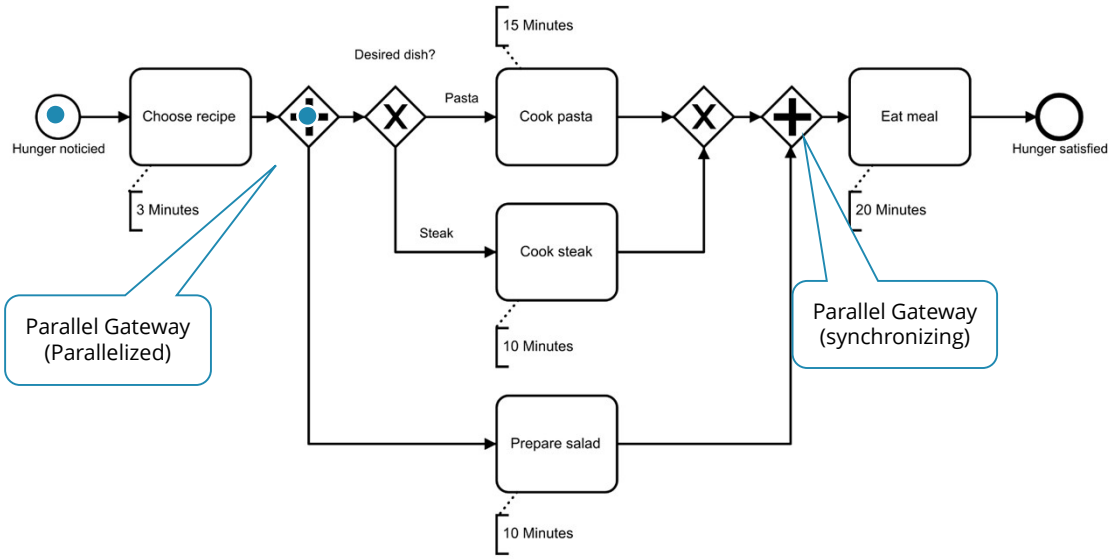




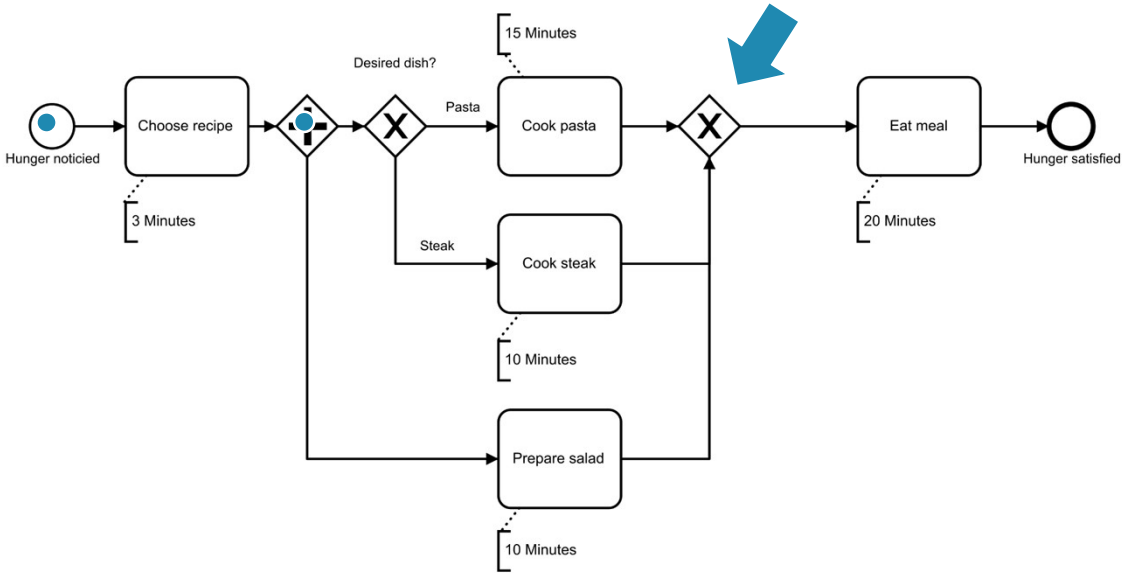
XOR-gateways can also merge



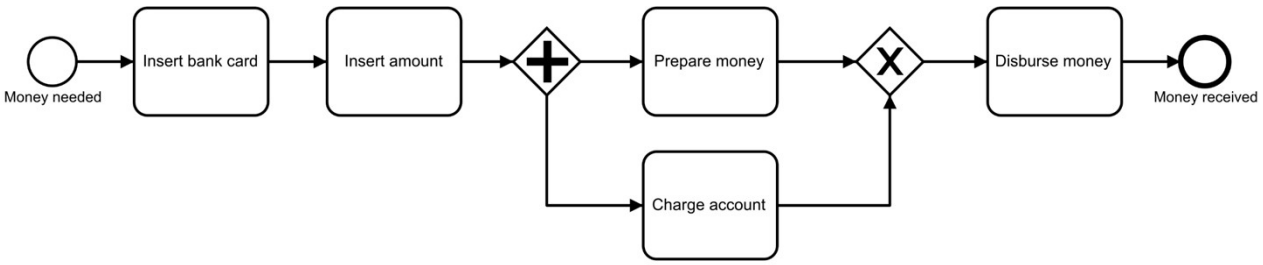
Preparing salad and main course at the same time



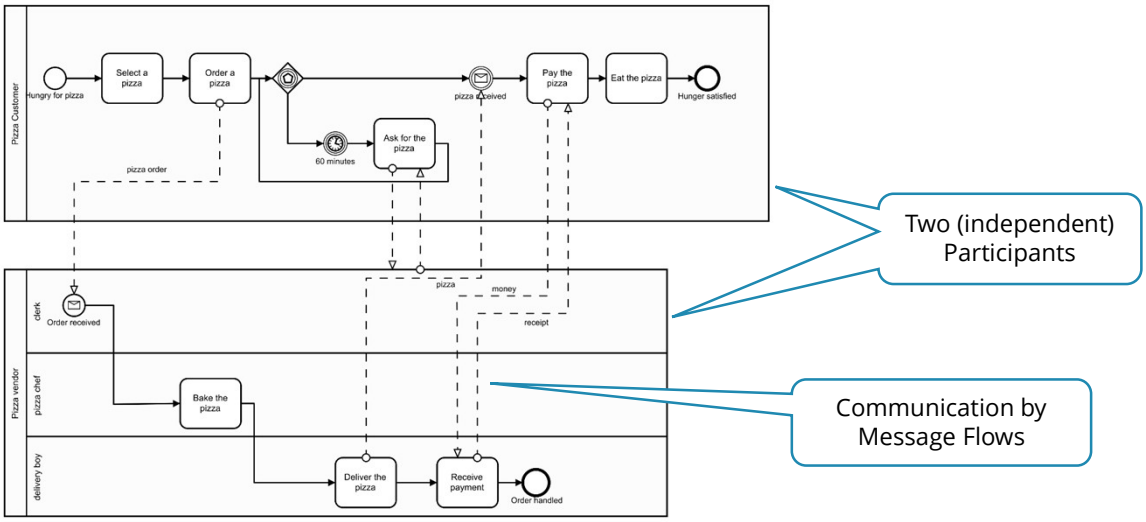
What happens in this process?



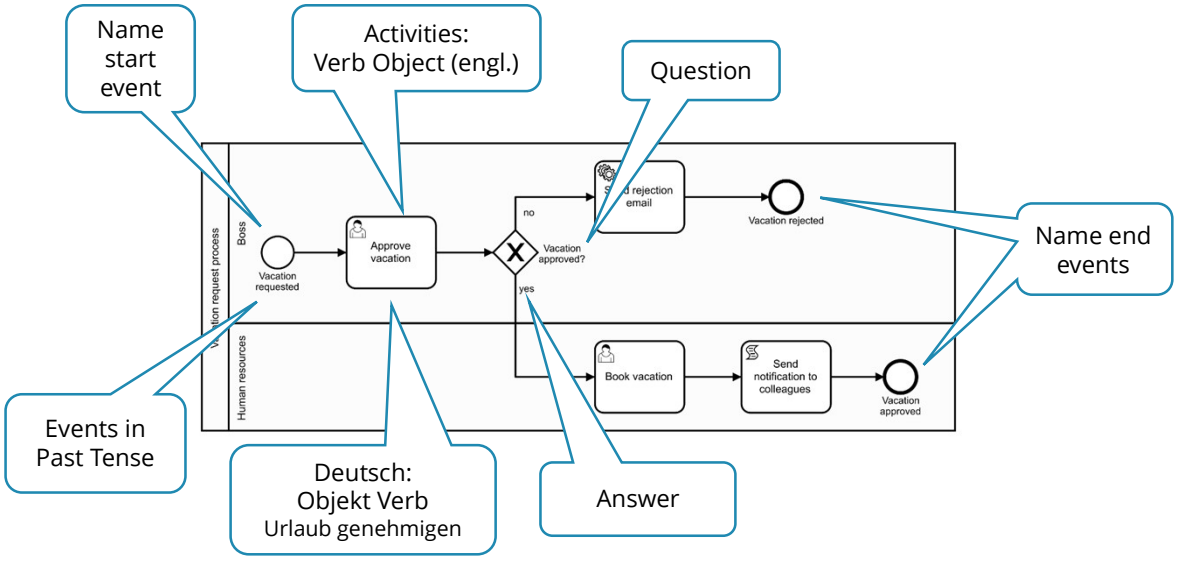
Example: ATM

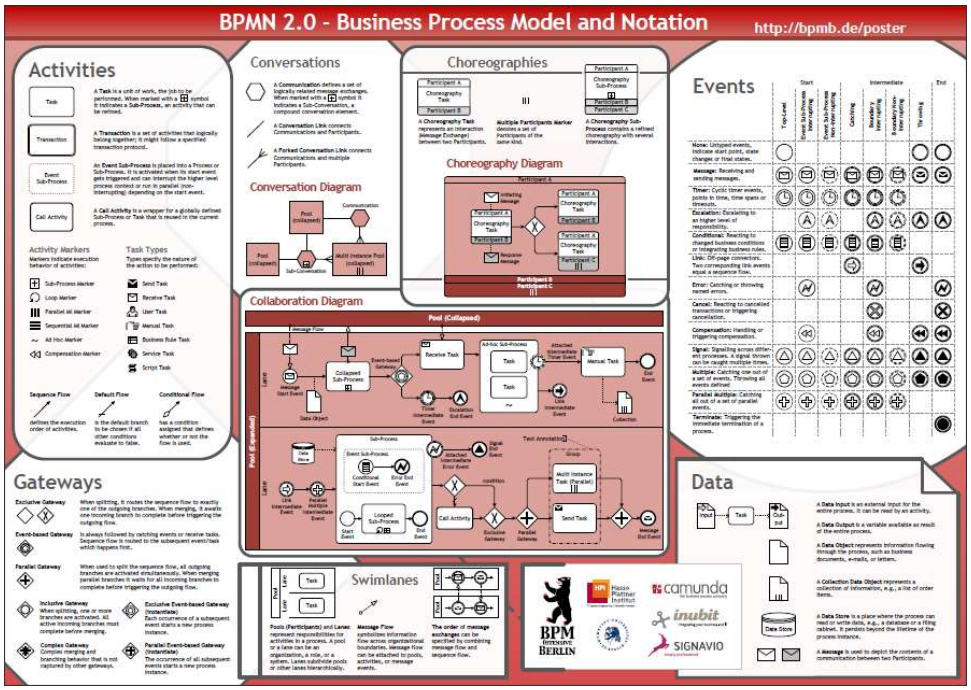


Collaboration Diagram



Create Readable Process Models



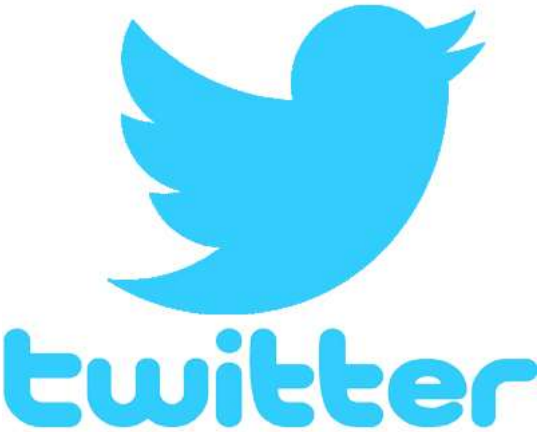


17

Scenario for Exercises

18

You Know Twitter?



19

Imagine the Anti Agile Inc.

Why do we need Twitter?

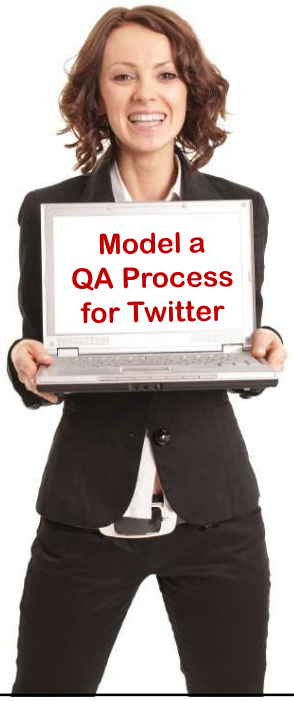
- We do not trust our employees.
- We want to control everything.
- We do not like Twitter,
- But we want to be part of it.

20

The Anti Agile Inc. Needs Twitter QA!

Group Exercise 0

Use a whiteboard or flipchart
to model a Twitter QA process

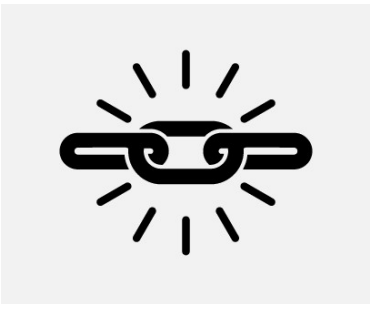


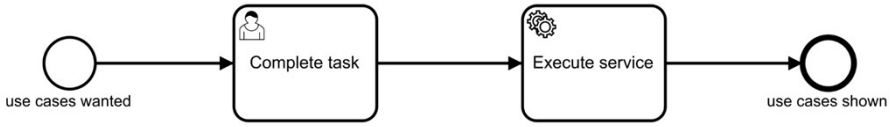
QA Process for Twitter

Process Automation Use Cases

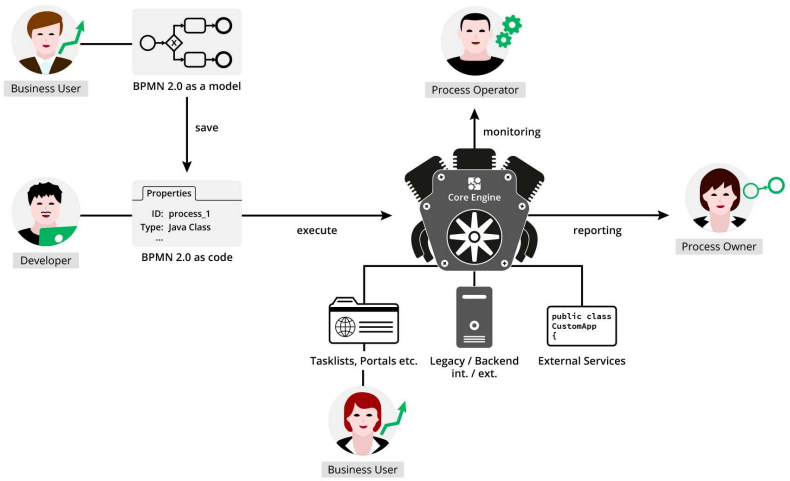
Human Task Management

System Integration





What's So Cool About BPMN 2.0

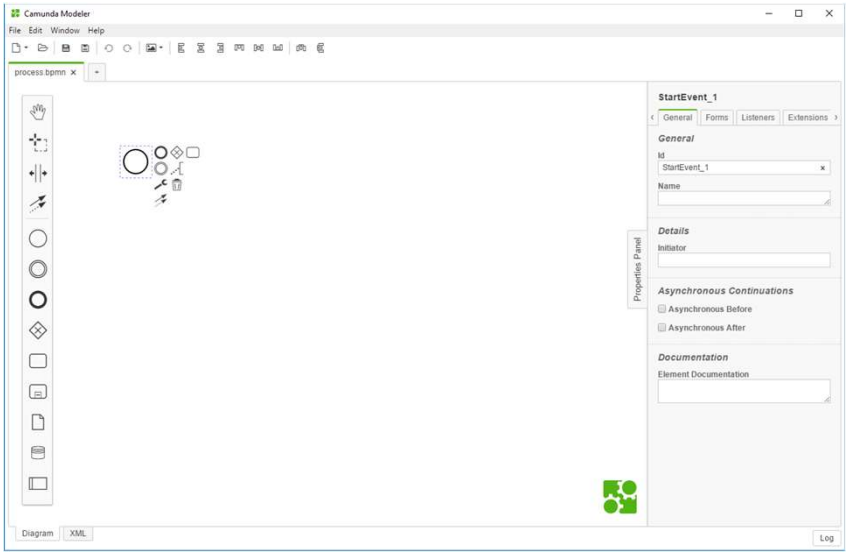


Why Using a Process Engine?

- 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

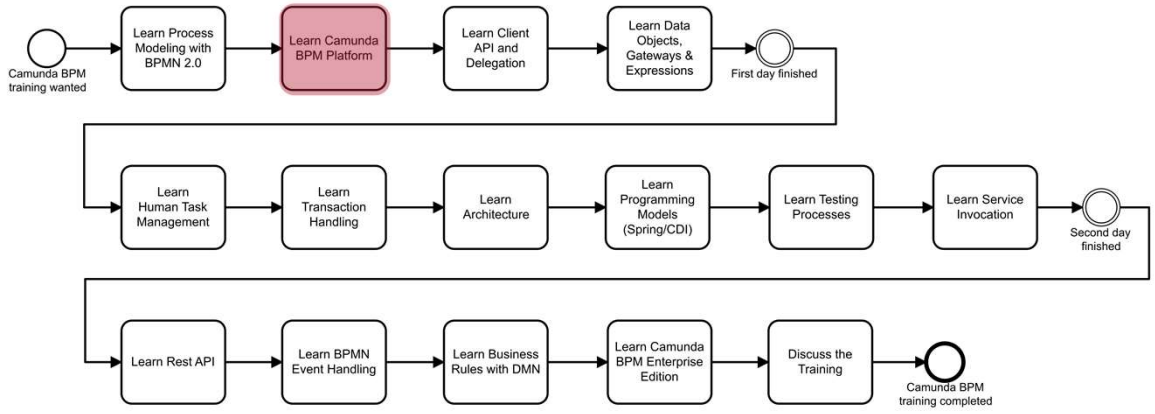
Demo

Start Modeling!

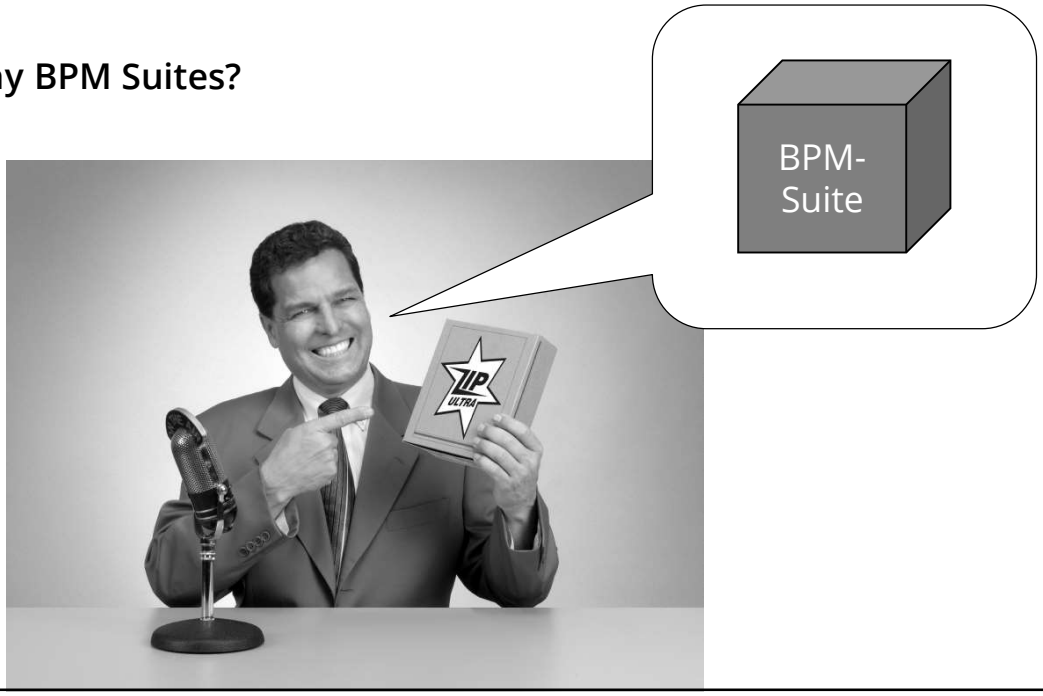


Exercise 1

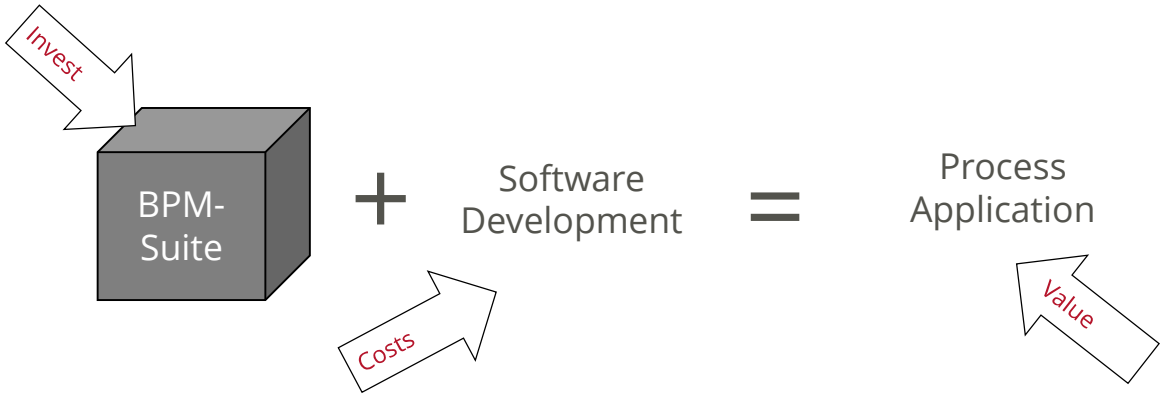
https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



Shiny BPM Suites?



You Always Need Software Development

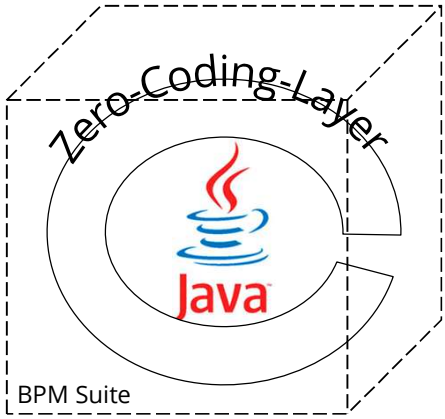


Tempting, but Deceptive

Business User

~~Software Developer~~

A Fundamentally Wrong Approach*



complicated
restrictive

Business User

restrictive
proprietary

Software Developer

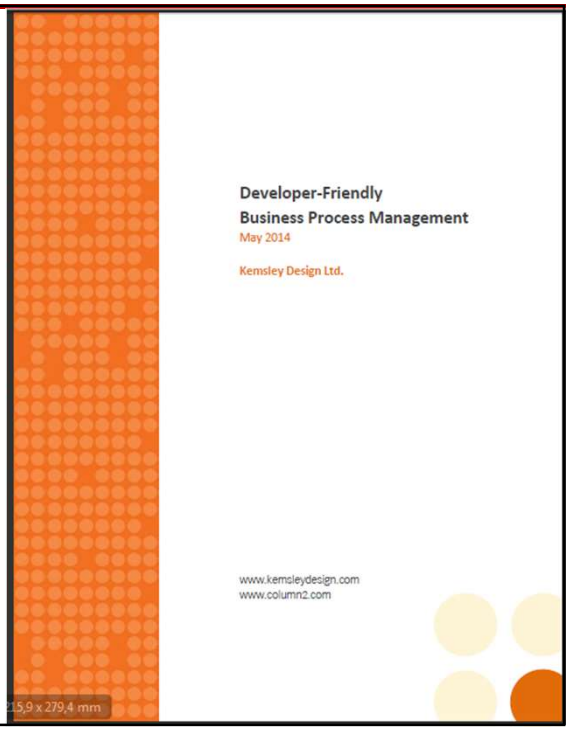
***for automating core business processes**

Our Approach: BPM + Java

Developer-Friendliness

- A fundamentally different approach compared to Zero-Code Suites!
- It's not about „empowering Business Analysts". Even with low code approaches coding is still required
- Camunda helps developers be more productive („Less code")
- Camunda is microservice orchestration ready

https://camunda.com/learn/whitepapers/developer-friendly-bpm/



In short

Camunda

Developer-Friendly System Integration

VS.

Zero-Code BPM Suites

Death by Property Panel

Camunda

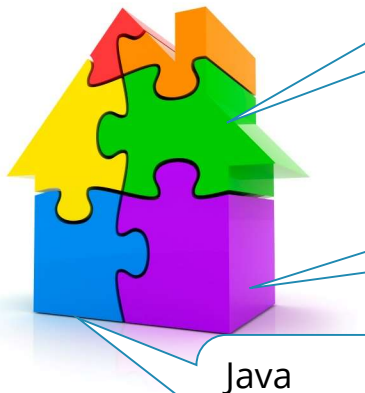
Developer-Friendly System Integration

VS.

Low-Code BPM Suites

You're still coding

Best of Breed – Based on Standards



You name it!

Any component from the Java cosmos.

Camunda BPM

Process Automation, Monitoring, Operations, ...

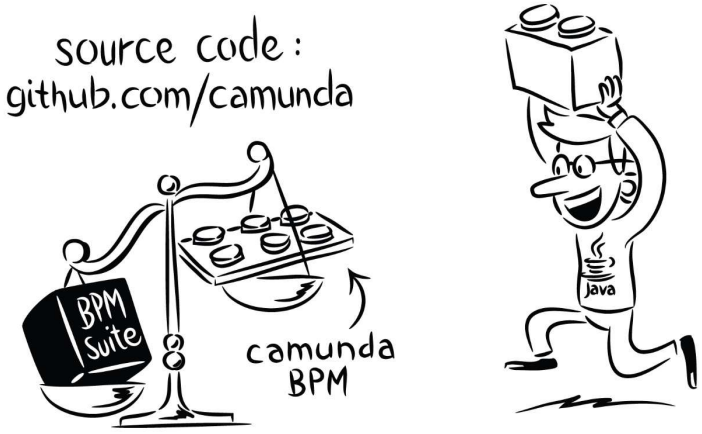
BPMN 2.0

Process Modeling, Requirements Engineering, Roundtrip, ...

Java

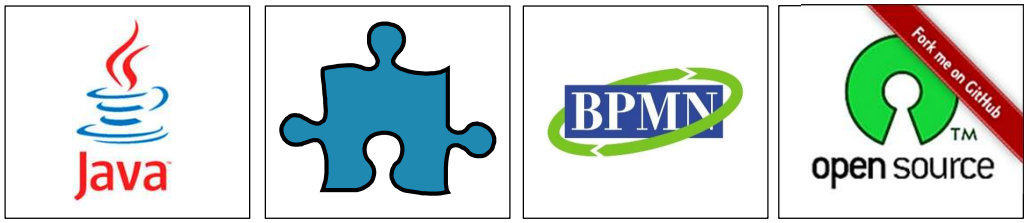
Persistence (JPA), Transactions (JTA, EJB3), Clustering, Connectivity (JAX-WS, JAX-RS, JAX-B), UI (JSF), ...

Lightweight and Open Source BPM Framework



...Plus BPMN Based Business-IT-Alignment

Summary: Core Concepts of Camunda BPM

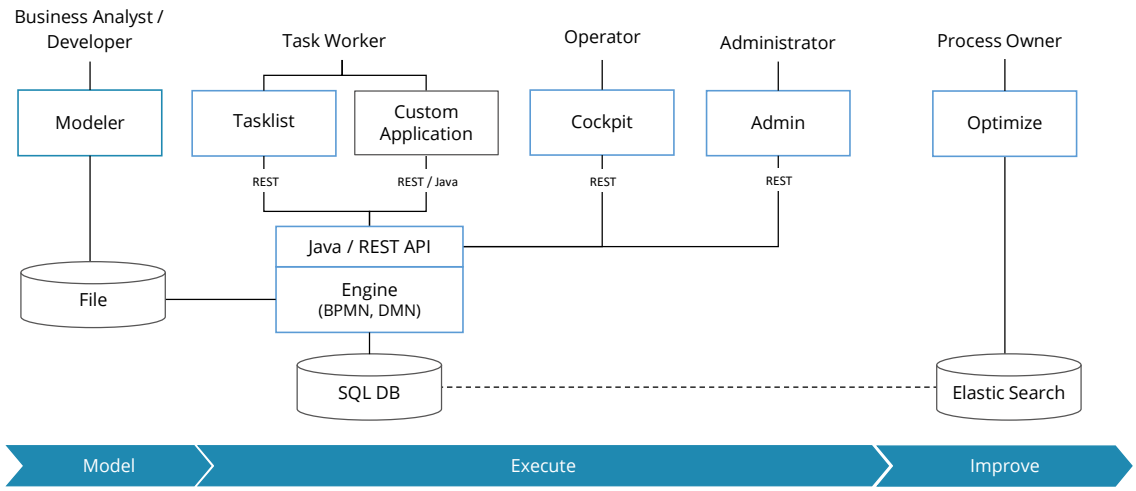


Decide Yourself!

BPM in a can

BPM cook-it-yourself

Camunda BPM Components

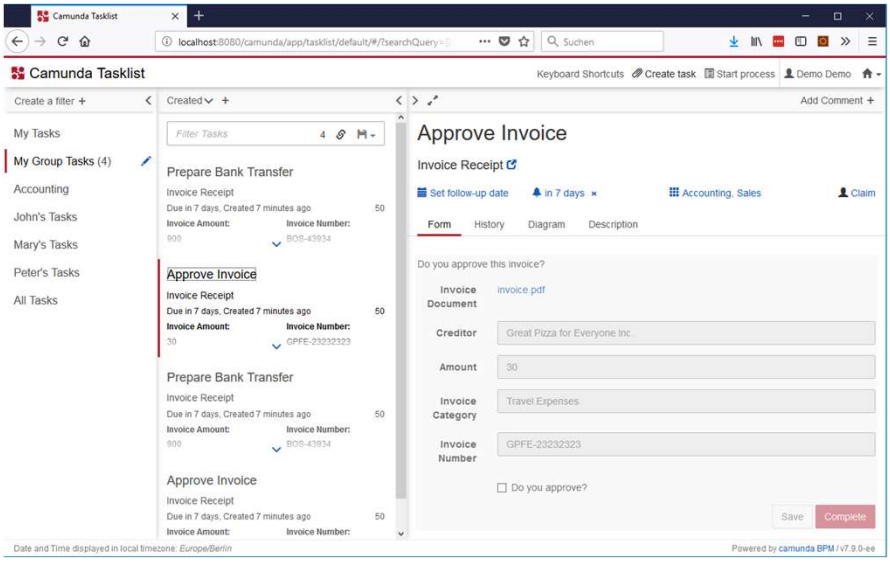


Exercise 2a

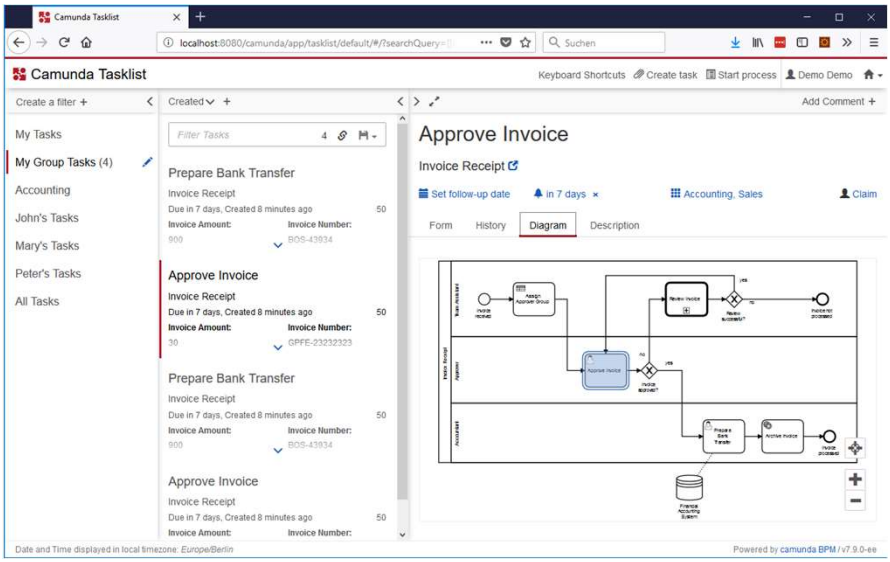
https://training.camunda.com/java-dev/
user: java-dev password: camundarocks

Demo

Camunda Tasklist

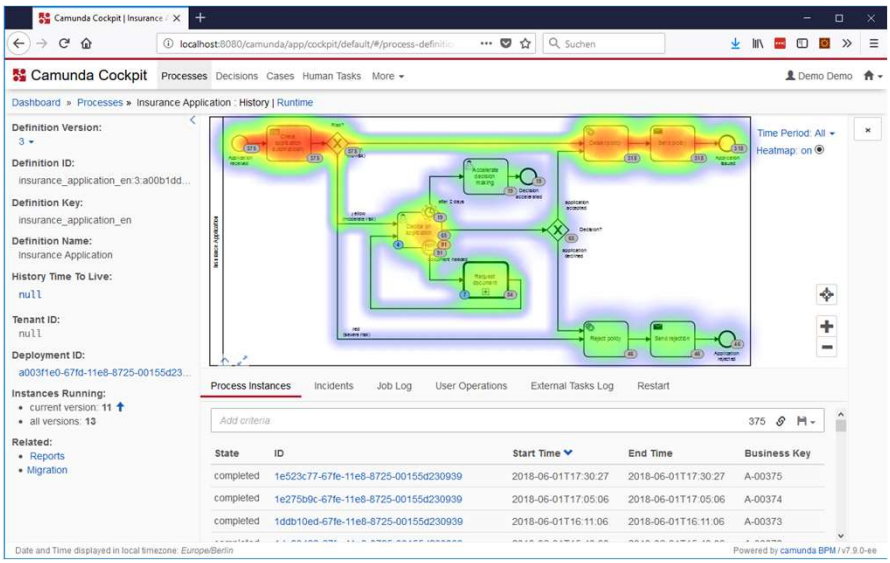


Process Overview



Camunda Cockpit

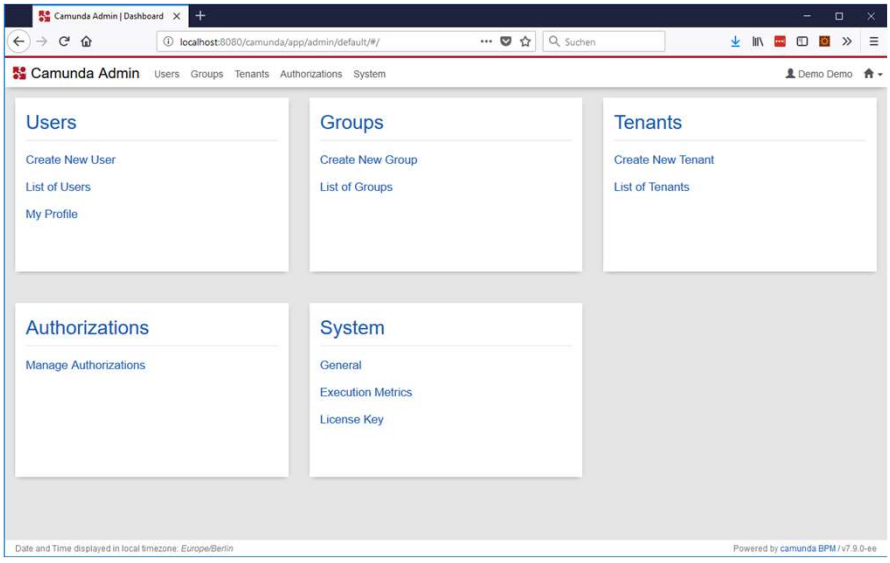
History view and Heatmap
Enterprise Feature



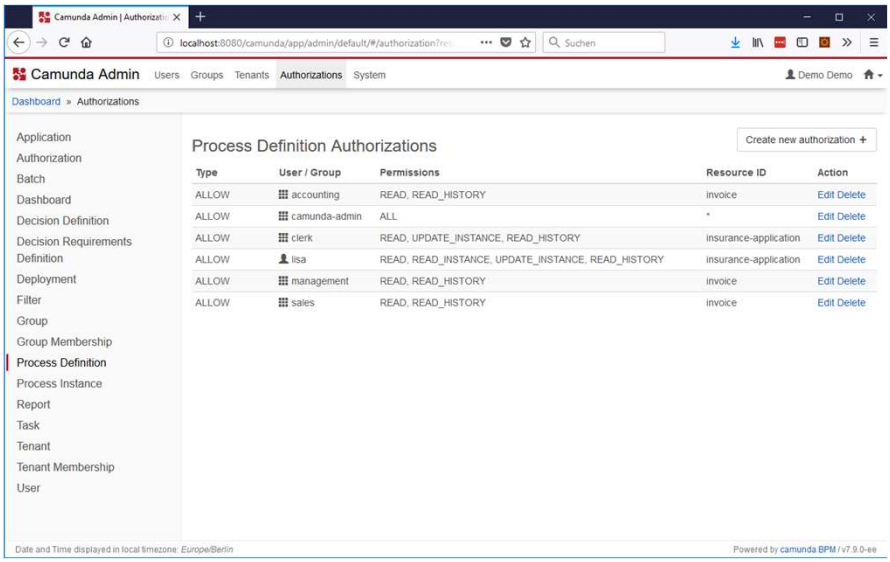
Cockpit Features

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

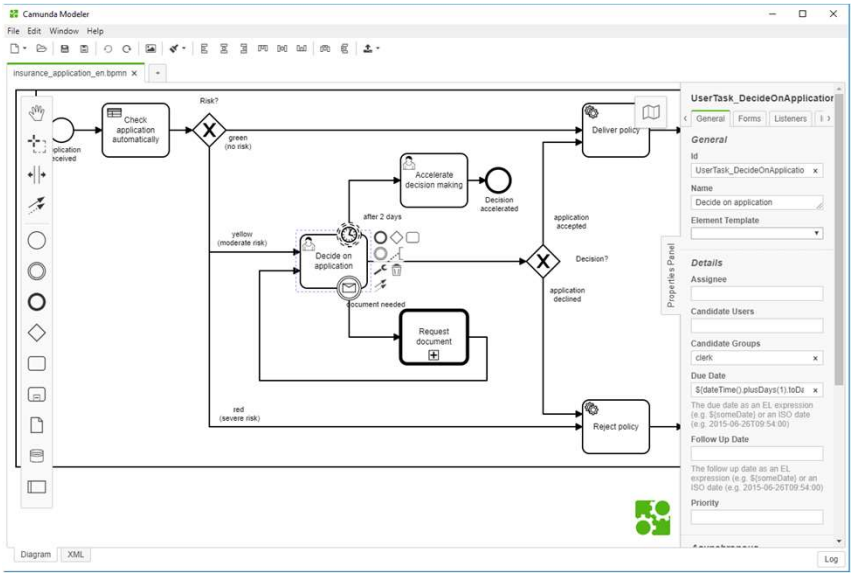
Camunda Admin



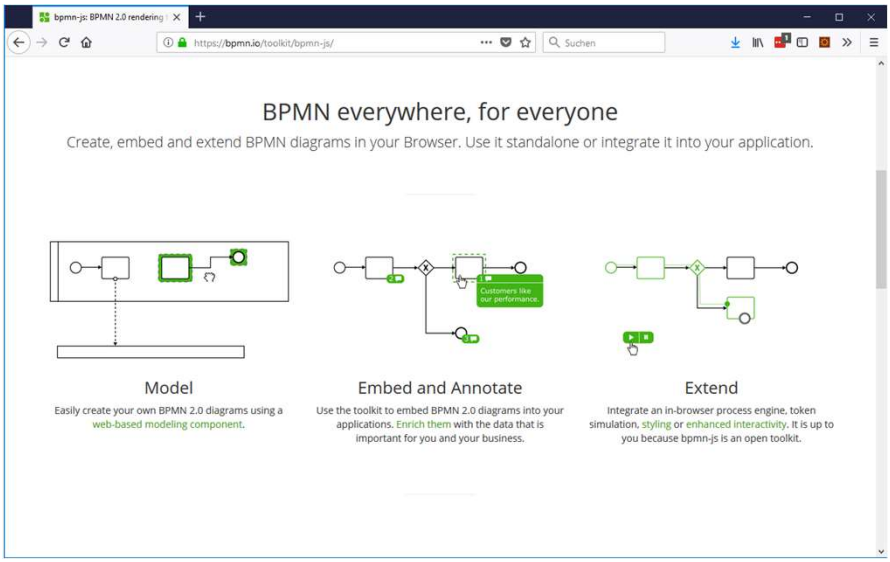
Authorizations



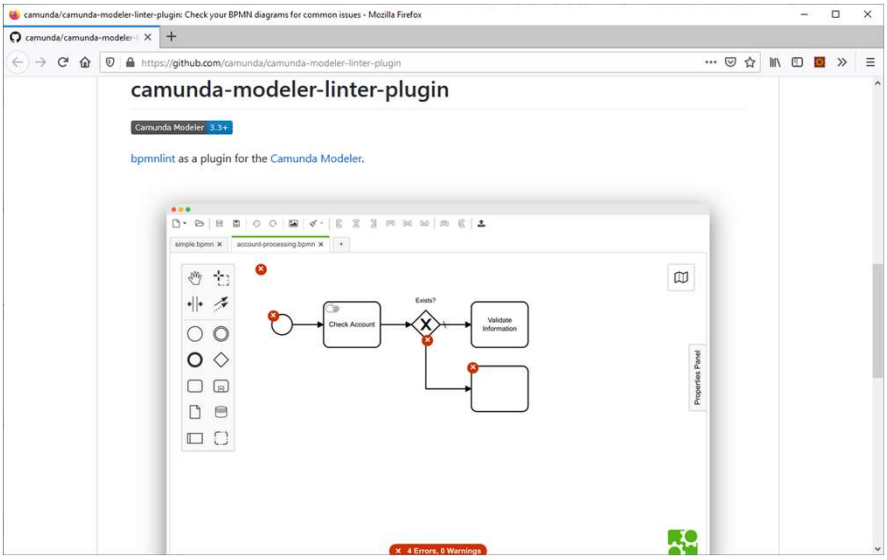
Camunda Modeler



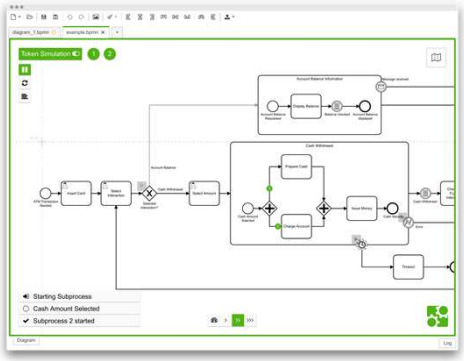
bpmn.io – a Developer Kit and Web Modeler for BPMN 2.0



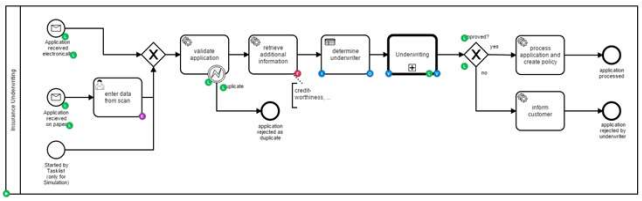
Extend the Camunda Modeler



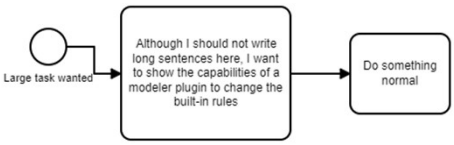
Camunda Modeler Plugin Examples



Token Simulation Plugin



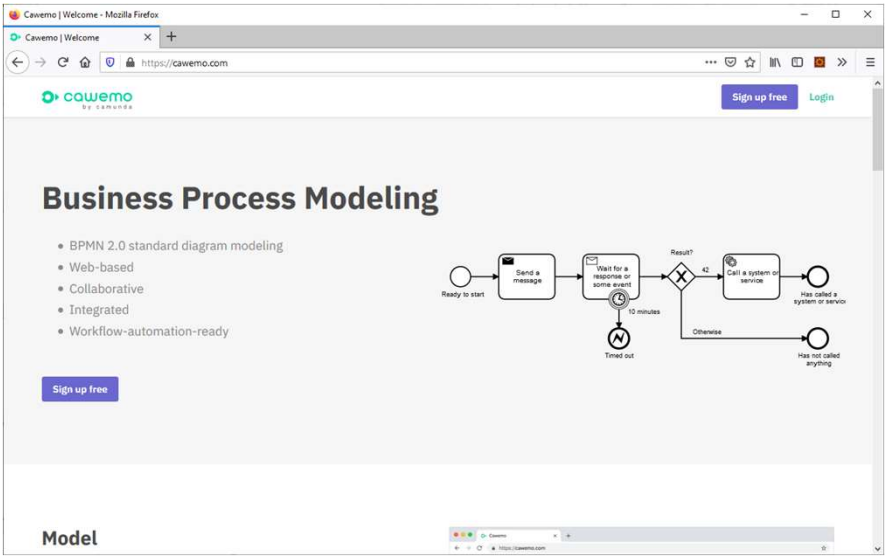
Property Info Plugin



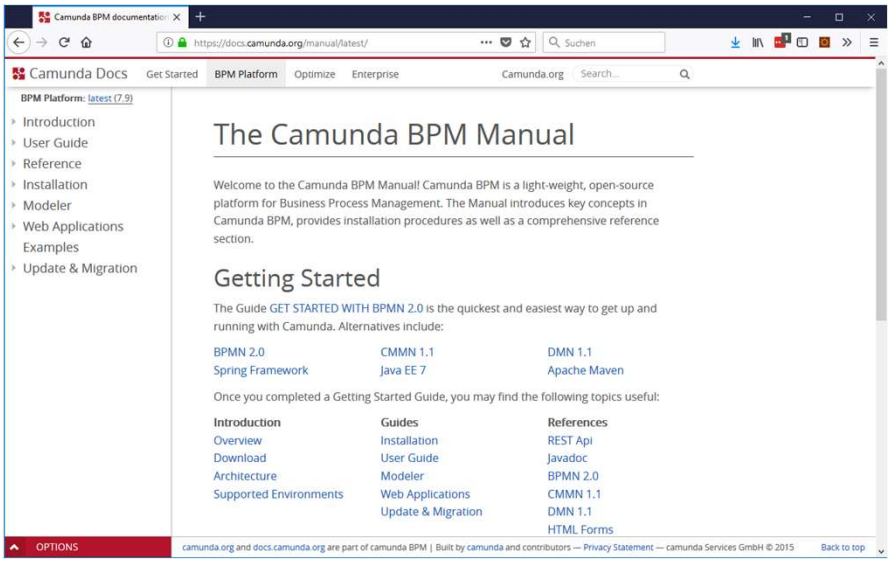
Resize Tasks Plugin

https://github.com/camunda/camunda-modeler-plugins

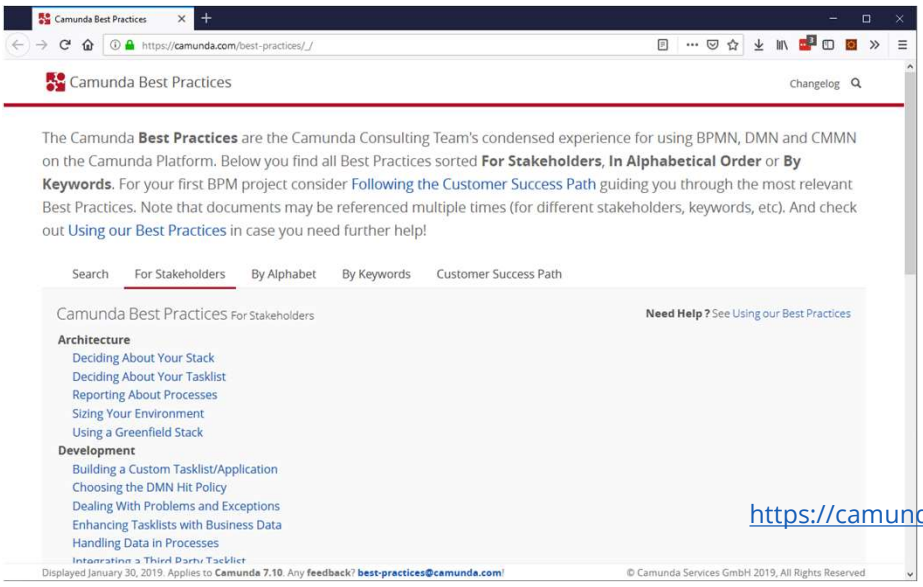
cawemo.com



https://docs.camunda.org



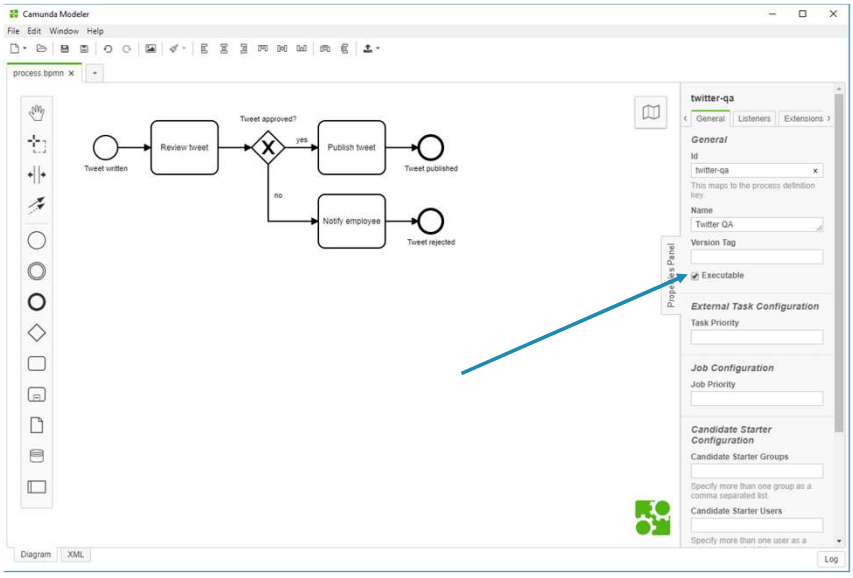
Camunda Best Practices



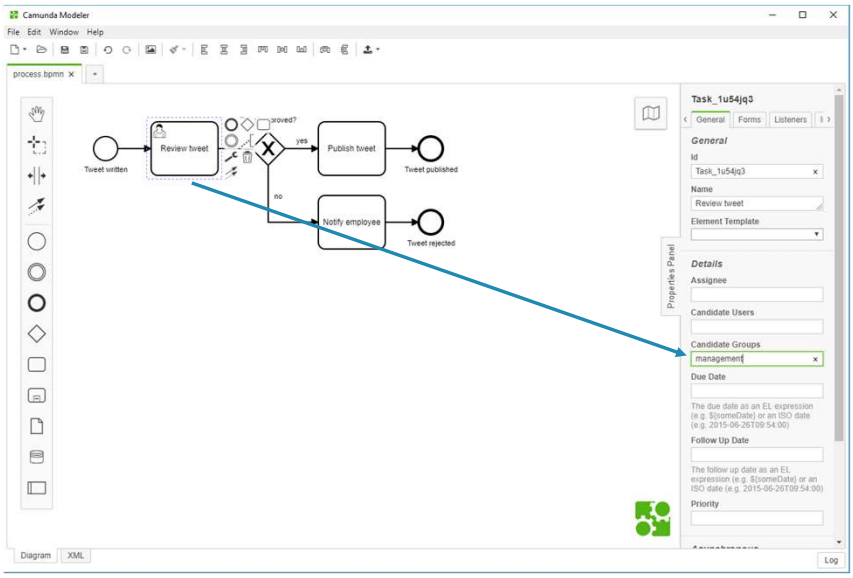
https://camunda.com/best-practices/

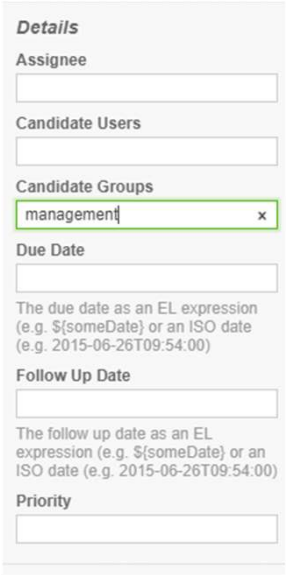
Prepare for First Launch

Make Process Executable

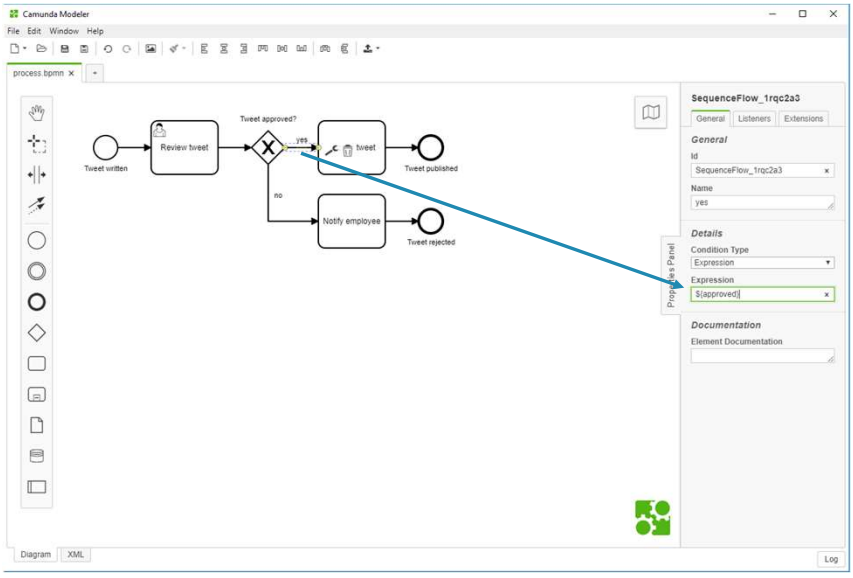


Assign User Task

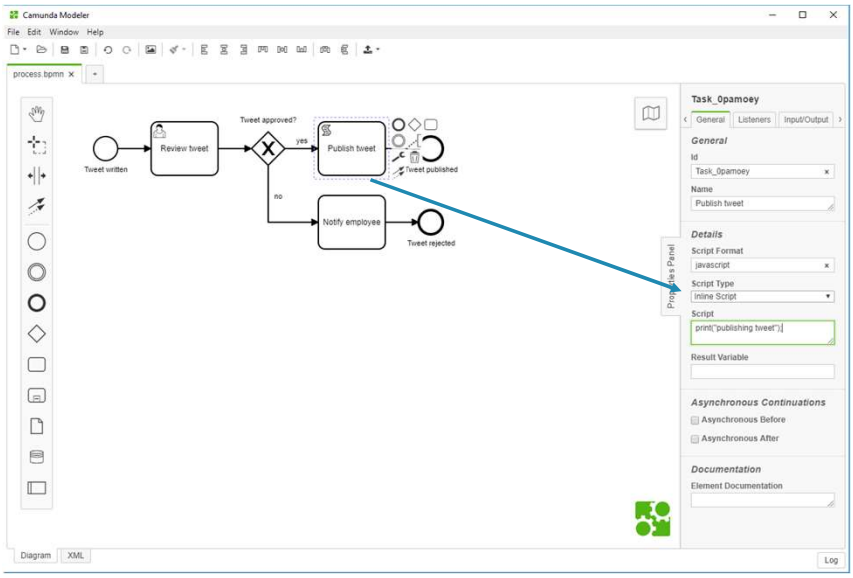




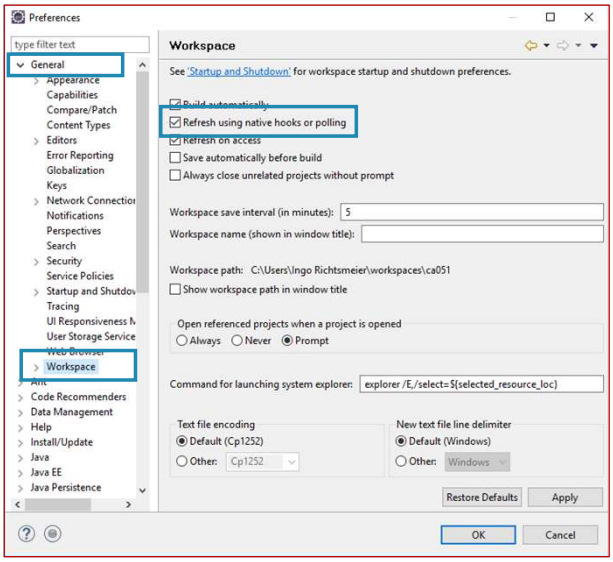
Conditions



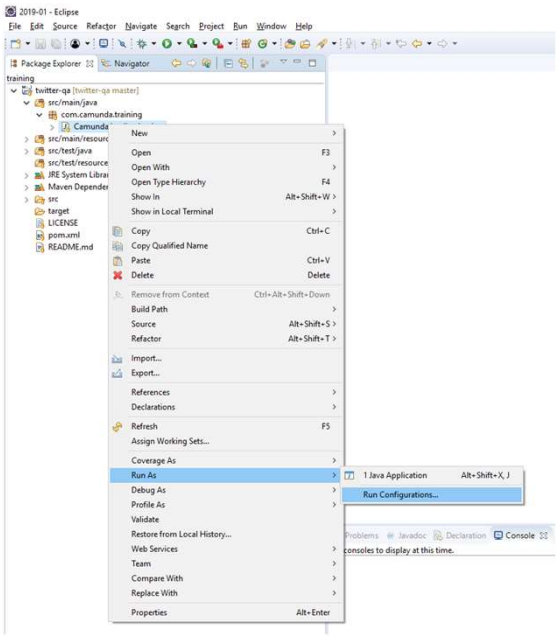
Scripts



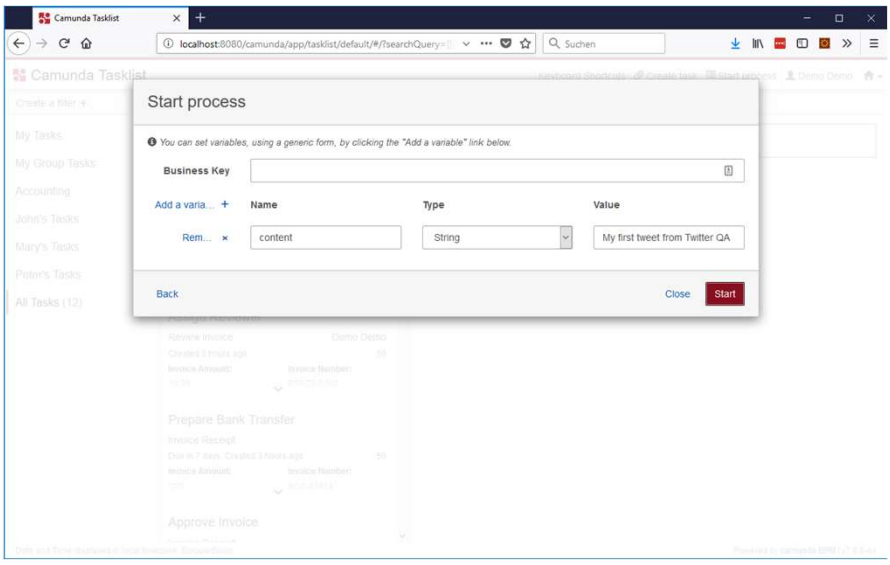
Setup Eclipse for Automatic Synchronization



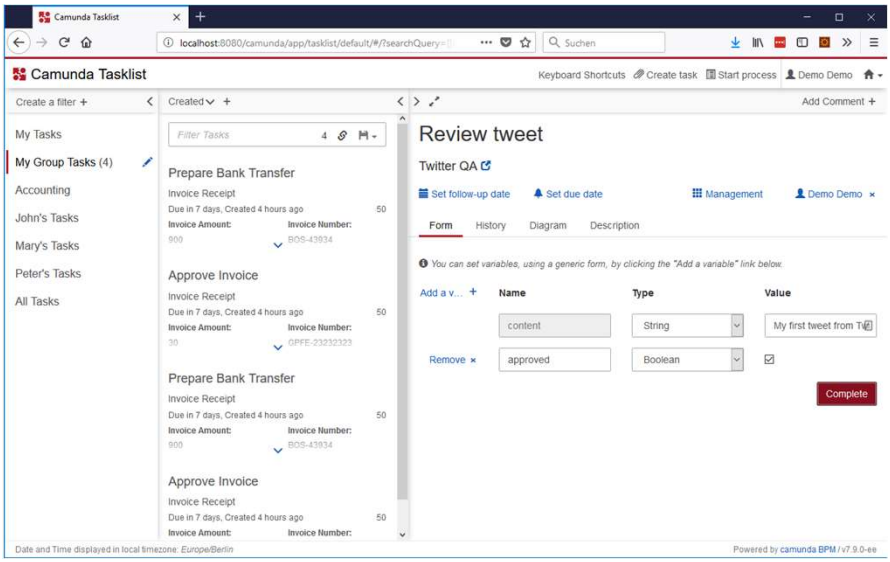
Start the application



Use Generic Task Forms

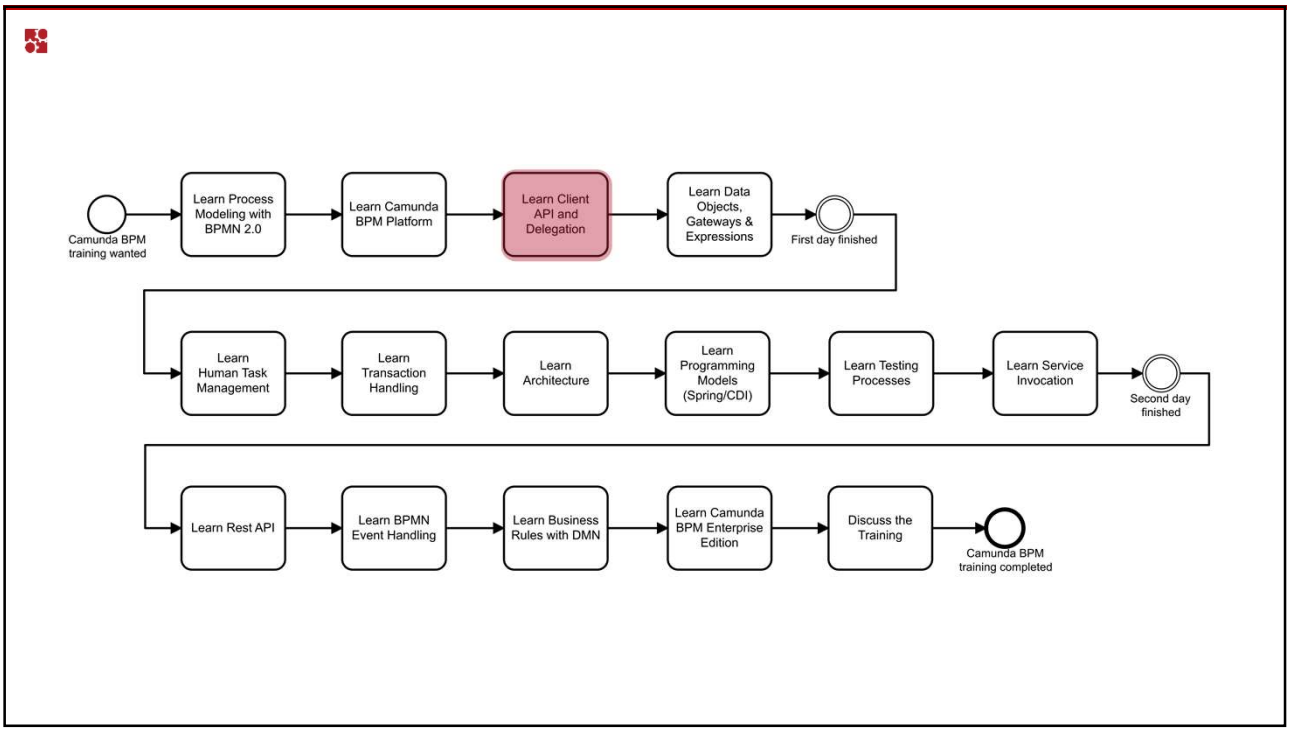


Complete Generic Task Forms



Exercise 2b

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



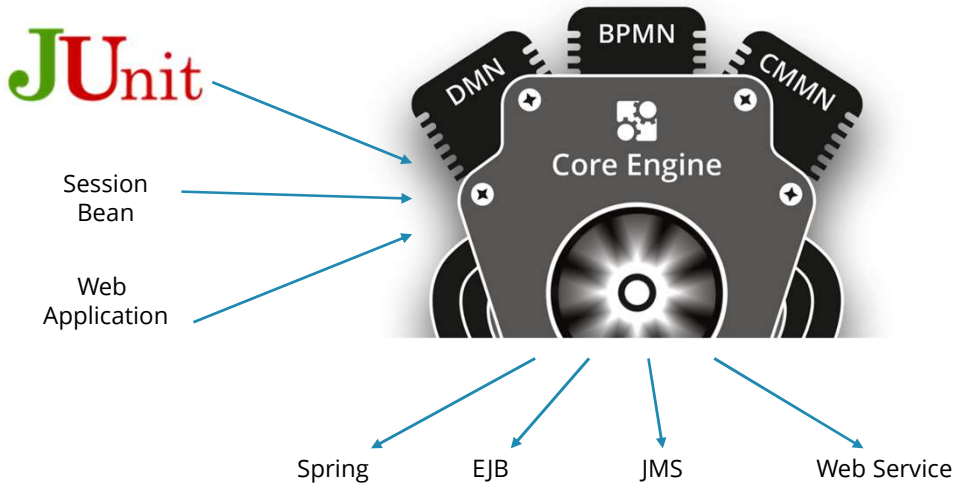
69

BPM + Java

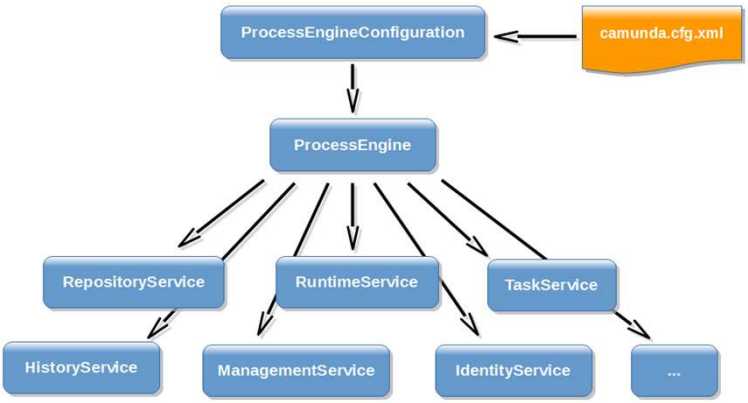
70

70

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
ExternalTask	External task lifecycle
Decision	Evaluate decisions
History	Logs, Statistics, KPIs
Identity	Users, Groups
Form	Form key, Form property, Form
Management	Jobs, Statistics, Maintenance
Filter	Tasklist filters
Authorization	Permissions
Case	Start of cases, Case execution lifecycle

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");
```

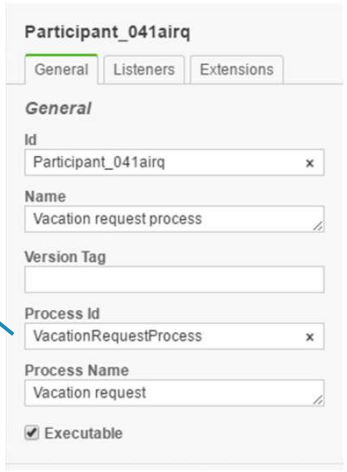
If needed, process instance migration is possible

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 Variables

Are persisted with the process instance

Client Code:

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

More details later

Business Key

- Domain specific identifier of a process instance
- Optional
- Set at the start
- Set or change in a Java Delegate or Execution Listener
- Displayed prominently in Cockpit and Tasklist
- Query for process instances

Candidates for the Business Key in the Twitter QA process?


```
runtimeService.startProcessInstanceByKey(
    "VacationRequestProcess", employee);
```


Camunda BPM Assert Library

- For comprehensive Junit tests
- Many helper methods
- E.g. withVariables:

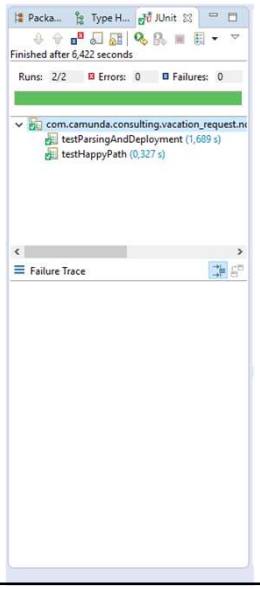
```
public static Map<String, Object> withVariables(final String key, final Object value,
                                                 final Object... furtherKeyValuePairs)
```

- E.g. access process engine services:

```
public static RepositoryService repositoryService()
public static RuntimeService runtimeService()
public static TaskService taskService()
public static ExternalTaskService externalTaskService()
```

https://github.com/camunda/camunda-bpm-assert

Unit Testing



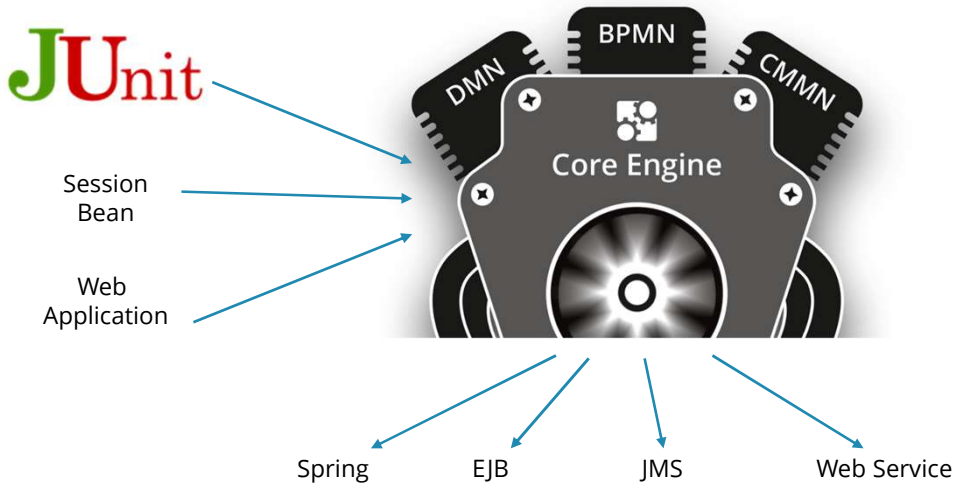

```
import static org.camunda.bpm.engine.test.assertions.ProcessEngineTests.*;

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

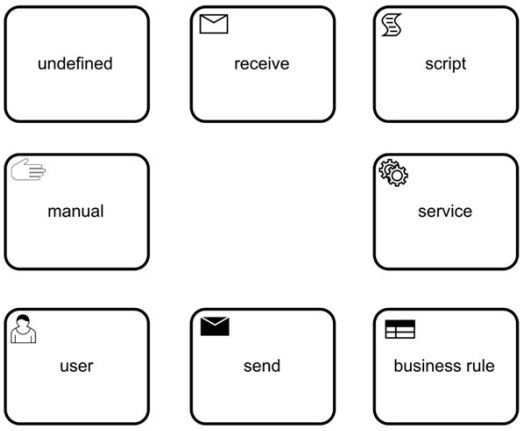
Exercise 3a

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks

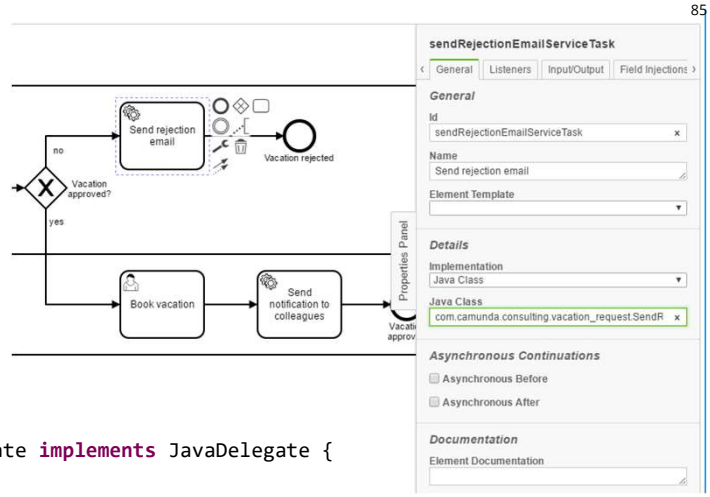
The Way OUT



BPMN 2.0 Task Types



Service Task With Java Class

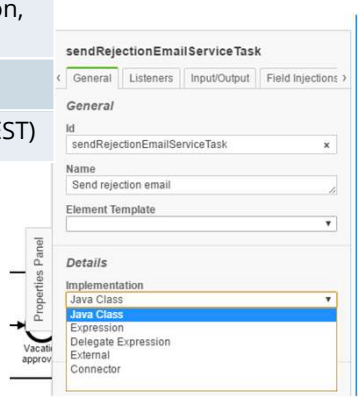



```
public class SendRejectionNotificationDelegate implements JavaDelegate {

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

    execution.setProcessBusinessKey("my new Business Key");
  }
}
```


Other Options for Service Tasks

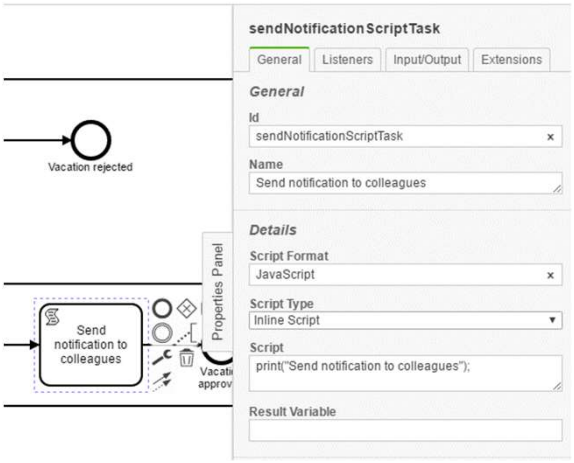
Implementation	Description
Expression	Execute a method expression, e.g. invoke a bean method
Delegate Expression	Resolve the implementation from an expression, that implements `JavaDelegate` , e.g. a bean
External	Implement the service outside of the engine
Connector	Configure the call to a web service (SOAP or REST)

More details come later

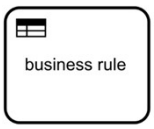


Script Task

- Every JSR-223 compliant scripting engine can be used (we tested JS, Groovy, Jython, JRuby)
- Script sources can be externalized
- Scripts get compiled and cached
- Available variables in all script languages:
  - all process variables
  - `execution`
  - `task`

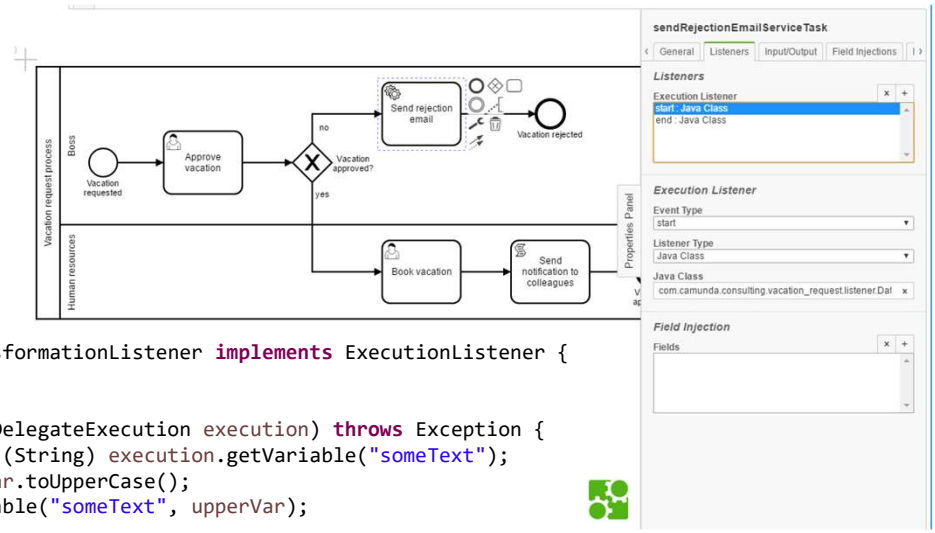


Business Rule Task



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

Execution Listeners

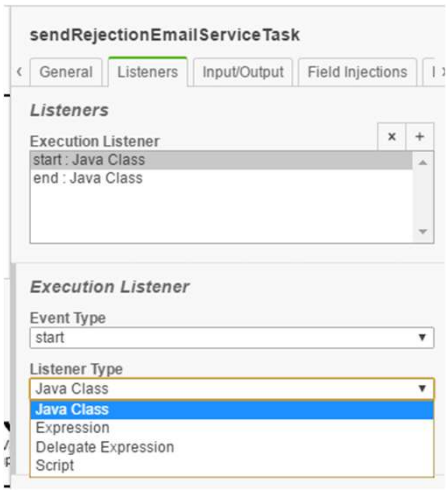



```
public class DataTransformationListener implements ExecutionListener {

  @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

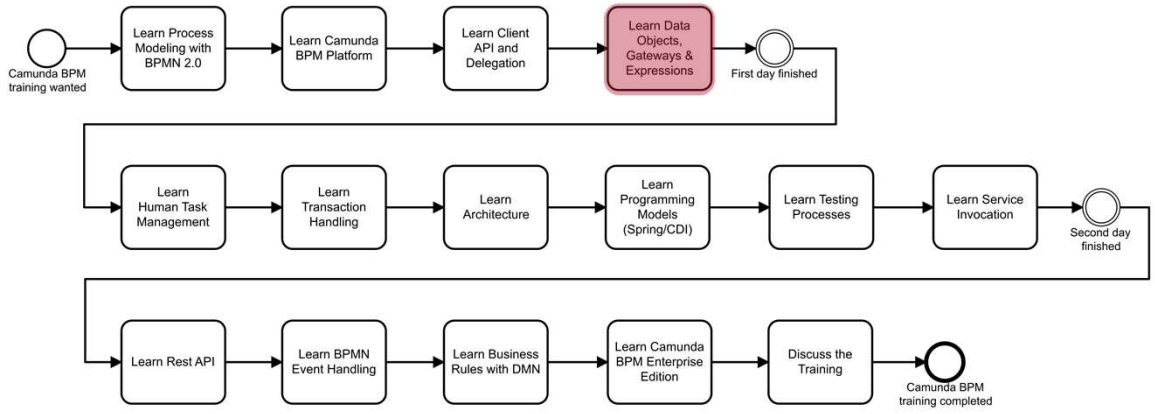
Level	Event
On process level	Start, end
On activity level	Start, end
On sequence flow	take

Listener Type
Java Class
Expression
Delegate Expression
Script



Exercise 3b

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



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

Object Variables

Saving values:

```
Customer customer = new Customer();
...
ObjectValue customerJson = Variables
    .objectValue(customer)
    .serializationDataFormat(SerializationDataFormats.JSON)
    .create();
execution.setVariable("customer", customerJson);
```

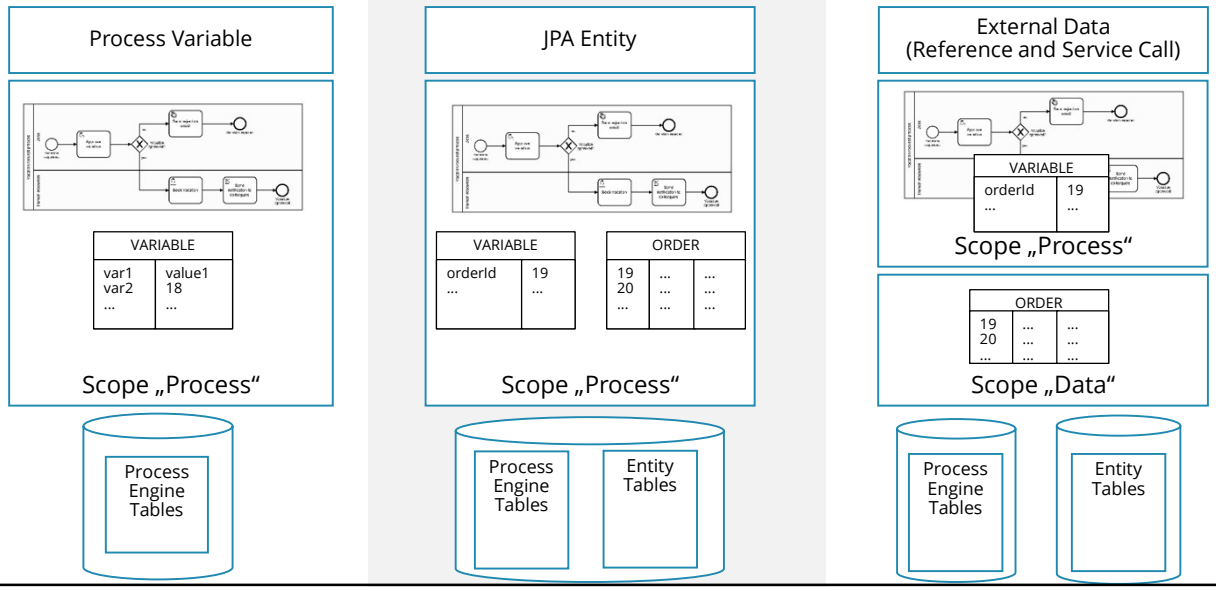
Serialize explicitly

Reading values:

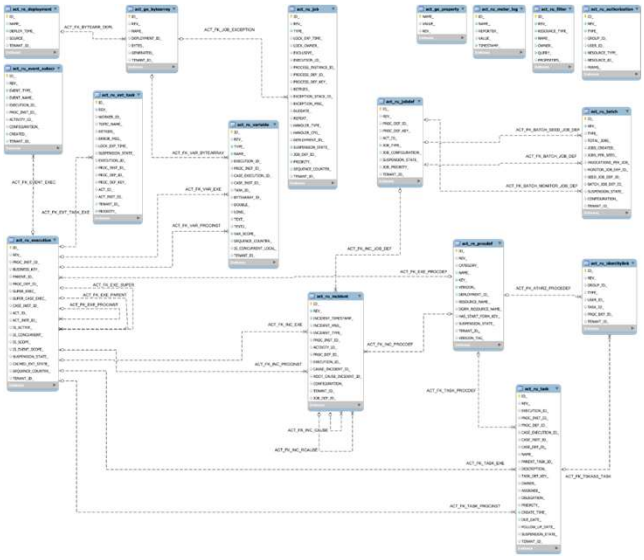
```
ObjectValue customerJson = execution.getVariableTyped("customer");
Customer customer = (Customer) customerJson.getValue();
```

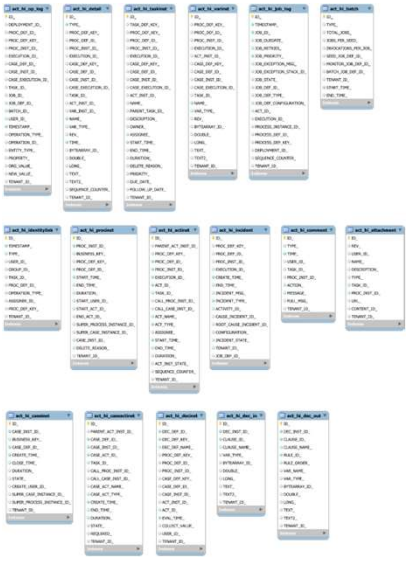
Deserialize transparently

Storage



Database Schema

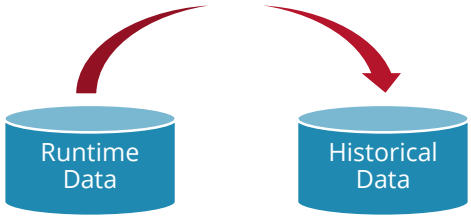




https://docs.camunda.org/manual/latest/user-guide/process-engine/database/#database-schema

History Configuration

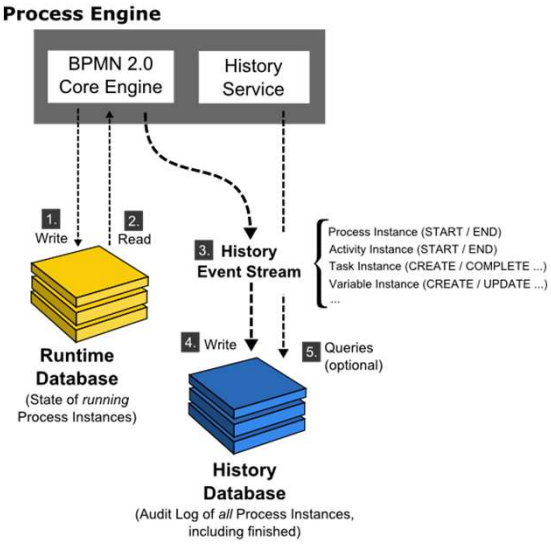
finished process instance are removed from runtime tables



Configuration options:

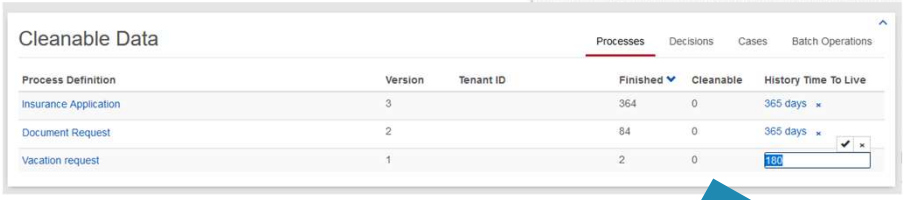
- None: no history at all
- Activity: process instances and trail of executed BPMN activities
- Audit: + last value of every process variable
  + submitted form properties (works only with forms)
- Full: + variable updates
- Custom: e.g. own filter for events of particular elements or variables

Asynchronous History



History Cleanup

- `historyTimeToLive` for each process definition: Number of days
- defined by business needs
- either in the process model
- or by API-Call
- or in the Cockpit

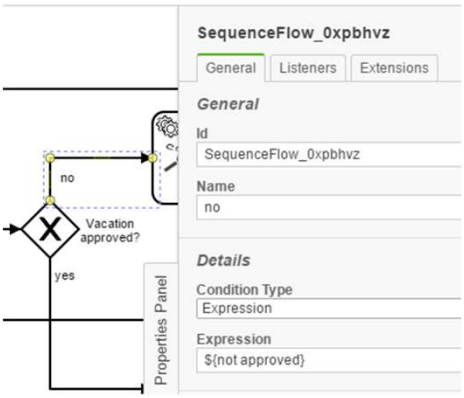


- run periodically, configure by weekdays
- Run manually

Cleanup View
Enterprise Feature

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

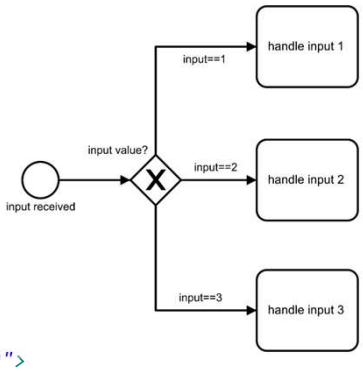
EL Expression	Result
${1 > (4/2)}	false
${4.0 >= 3}	true
${100.0 == 100}	true
${(10*10) ne 100}	false
${'a' < 'b'}	true
${'hip' gt 'hit'}	false
${4 > 3}	true
${1.2E4 + 1.4}	12001.4
${3 div 4}	0.75
${10 mod 4}	2
${!empty param.Add}	False if the request parameter named Add is null or an empty string.
${pageContext.request.contextPath}	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

Predefined Expressions

```
${currentUser()}
${currentUserGroups()}

from org.camunda.bpm.engine.impl.el.CommandContextFunctionMapper


${now()} -> java.util.Date
${dateTime()} -> org.joda.DateTime

from org.camunda.bpm.engine.impl.el.DateTimeFunctionMapper
```

https://docs.camunda.org/manual/latest/user-guide/process-engine/expression-language/#internal-context-functions

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 Example

```
{
  "name" : "jonny",
  "address" : {
    "street" : "12 High Street",
    "post code" : 1234
  }
}
```

Imagine a response from a web service call

Maybe a condition on a sequence flow

Access in an expression:

```
${JSON(customer).prop("address").prop("post code").numberValue() == 1234}
```

Access JSON variable value in an expression:

```
${customer.jsonPath("$.adress.post code").numberValue() == 1234}
```

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.

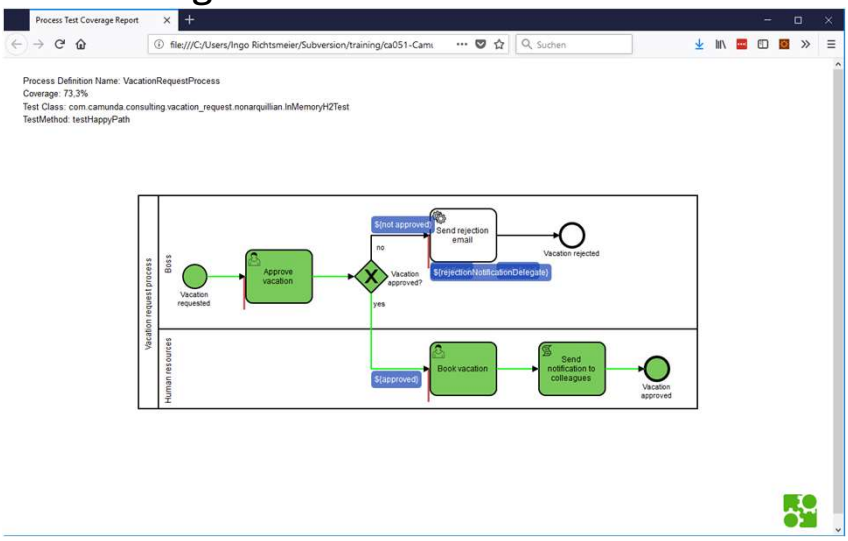
Features

- 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/

Process Test Coverage



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

Process Test Coverage

- Coverage per class and per test method
- Maven dependency:

```
<dependency>
    <groupId>org.camunda.bpm.extension</groupId>
    <artifactId>camunda-bpm-process-test-coverage</artifactId>
    <version>0.3.2</version>
    <scope>test</scope>
</dependency>
```

- process engine configuration class:

org.camunda.bpm.extension.process_test_coverage.junit.rules.
ProcessCoverageInMemProcessEngineConfiguration

- Usage in Junit:

```
@ClassRule
@Rule
public static ProcessEngineRule rule = TestCoverageProcessEngineRuleBuilder.create().build();
```


Exercise 4

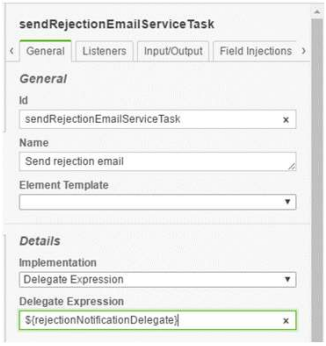
https://training.camunda.com/java-dev/
user: java-dev password: camundarocks

Improve JUnit with Simple Mocks

1. Make Delegate a Bean with `@Component`

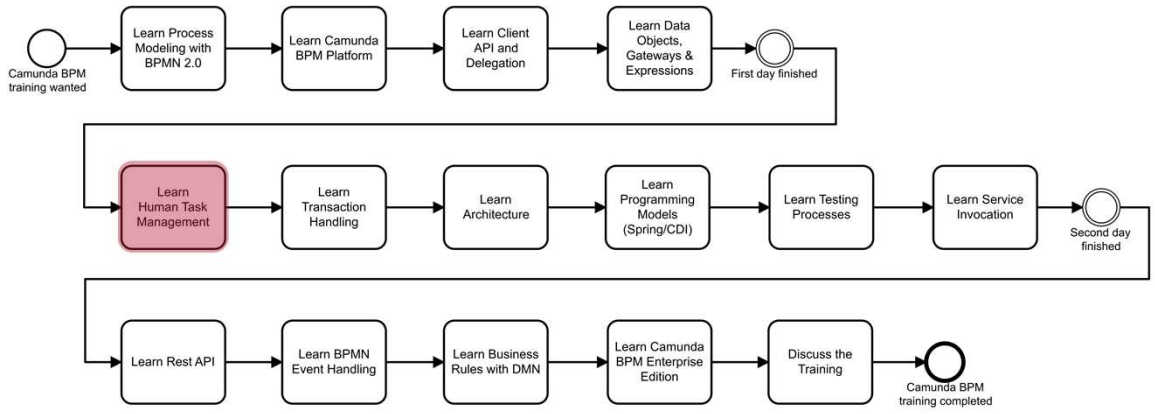
```
@Component("rejectionNotificationDelegate")
public class SendRejectionNotificationDelegate implements JavaDelegate {

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

2. Use Indirection with DelegateExpression in bpmn process

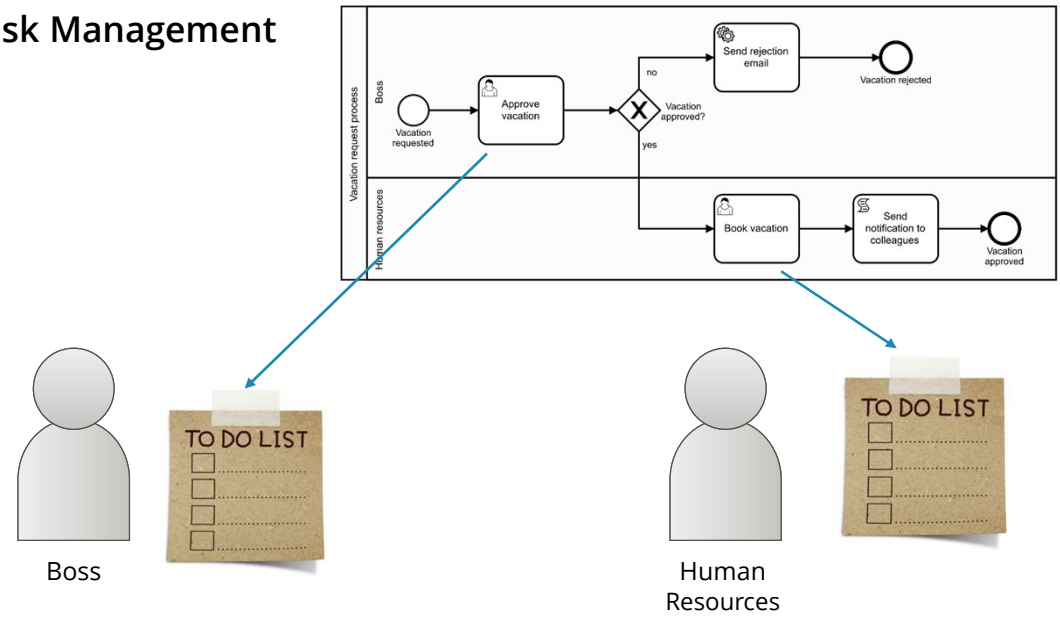


3. Use `Mocks.register("bean", new MockDelegate())` in the test method

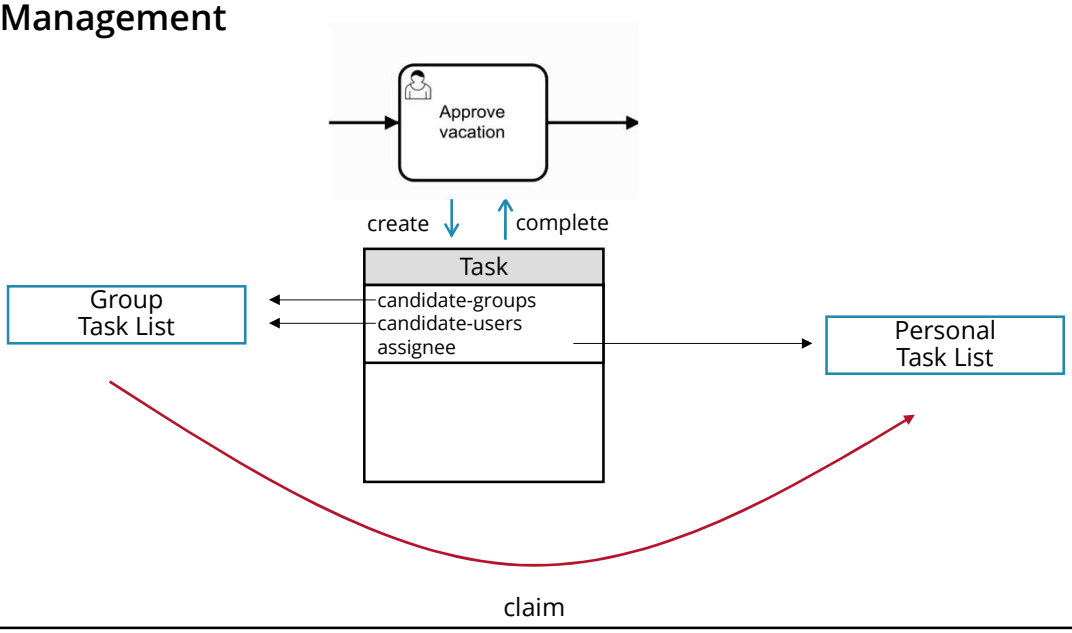
```
@Test
@Deployment(resources = "vacation-request.bpmn")
public void testRejectionVacation() {
  Mocks.register("rejectionNotificationDelegate",
      new LoggerDelegate());
  ...
}
```

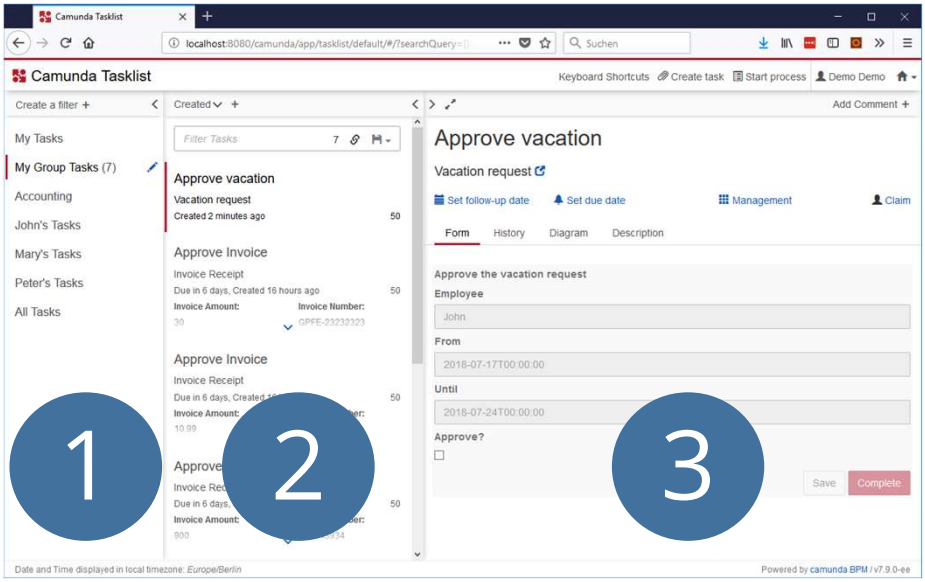
Task Management



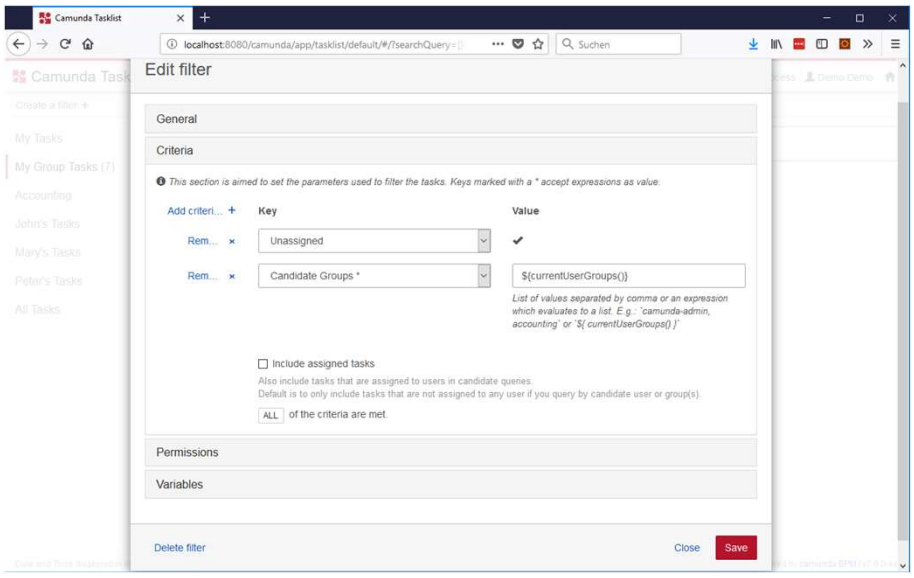
Task Management



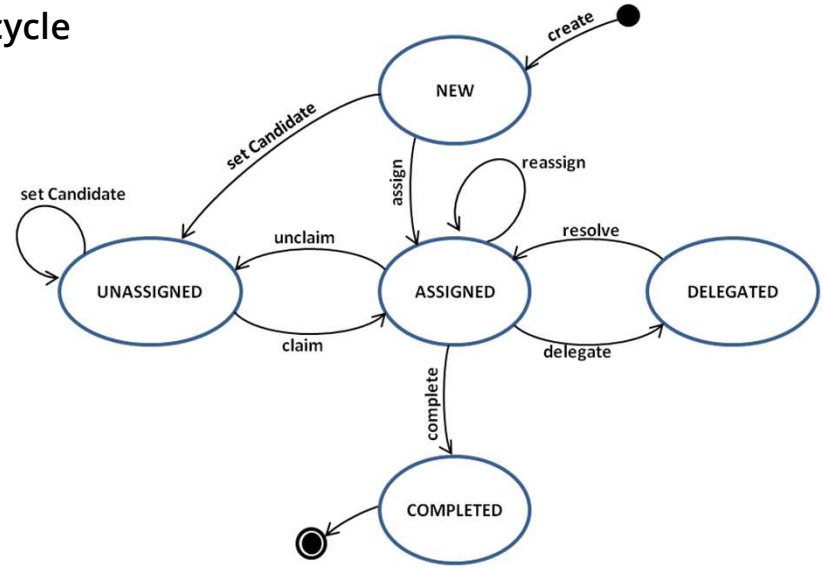
Tasklist in Camunda BPM



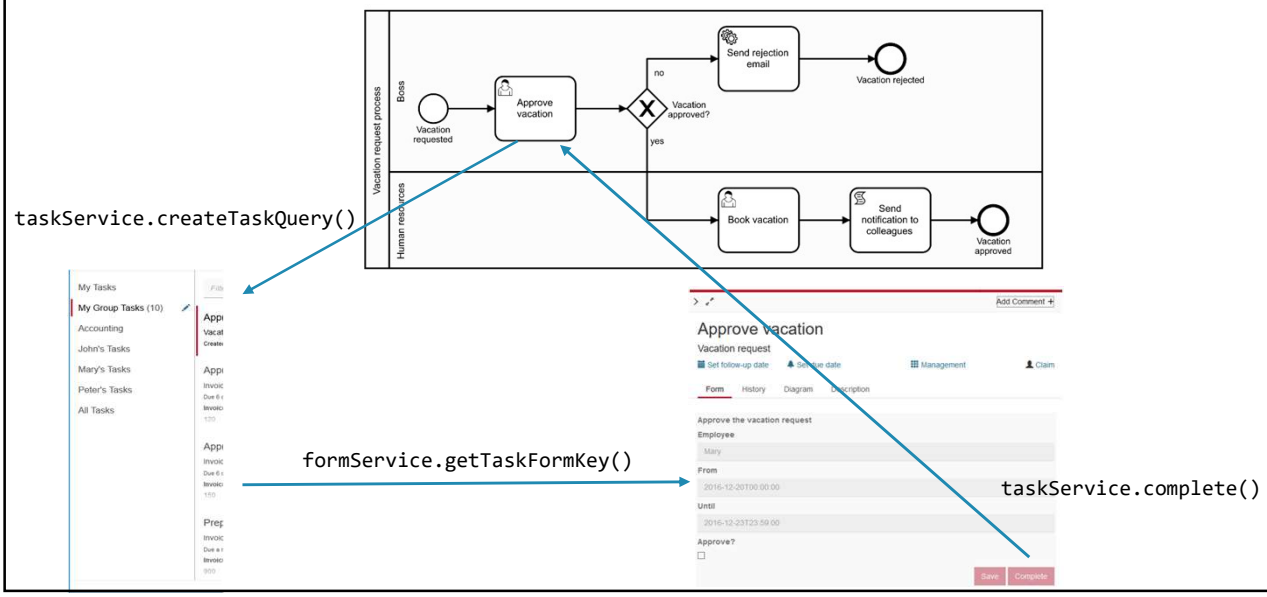
Groups & Personal Task List (Via Filters)



Task Lifecycle



Task Management API

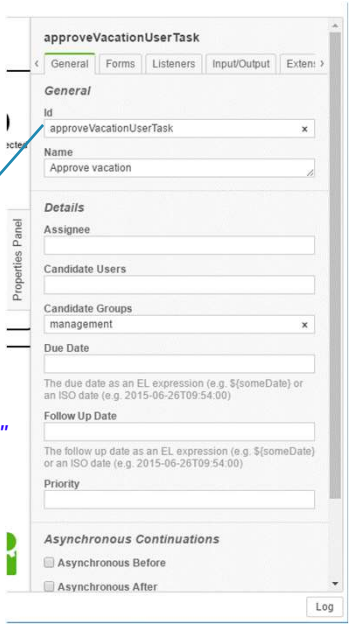


Key vs. ID vs. Name

```
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>
```



Personal and Group Task Lists

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

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


List<Task> groupTaskList = taskService.createTaskQuery()
       .taskCandidateUser("john").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);
```


Tasks in Camunda BPM Assert Library

- Assertions with fluent api:

```
assertThat(processInstance).task().hasCandidateGroup("management");
```

- Helper methods:

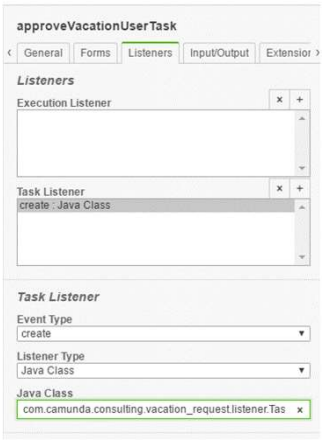
```
complete(task());
complete(task("bookVacationUserTask"));
```

```
public static TaskQuery taskQuery()
```

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:

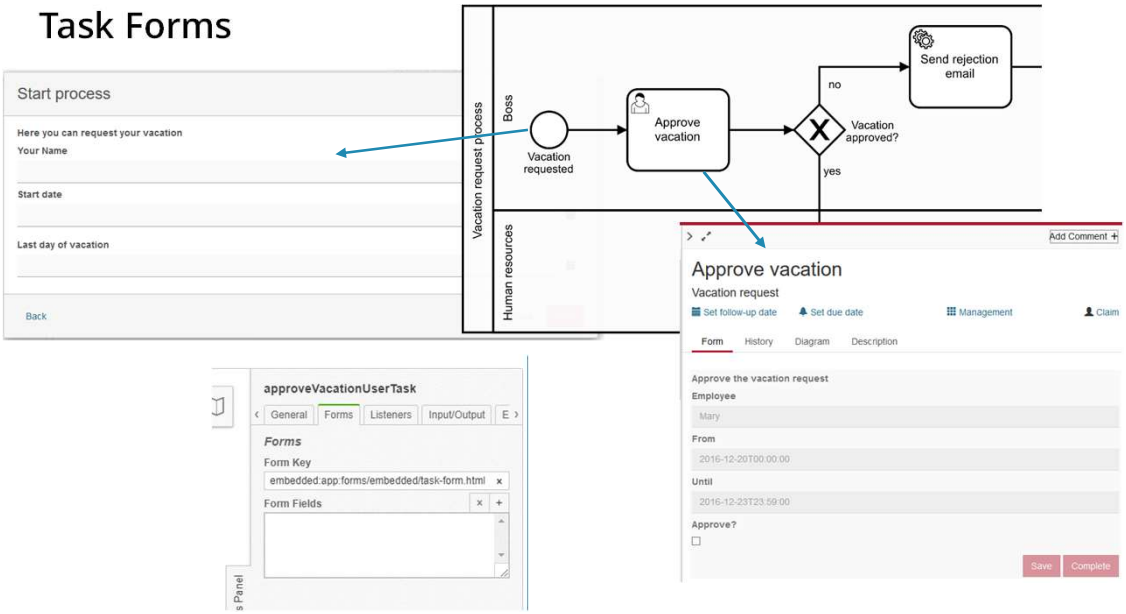
```
create
update
assignment
complete
delete
```

```java
public class TaskCreationListener implements TaskListener {

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


Exercise 5

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks

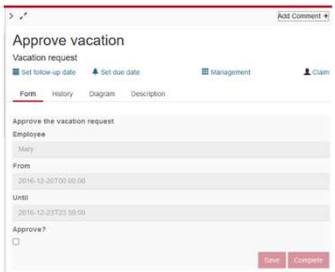
Task Forms



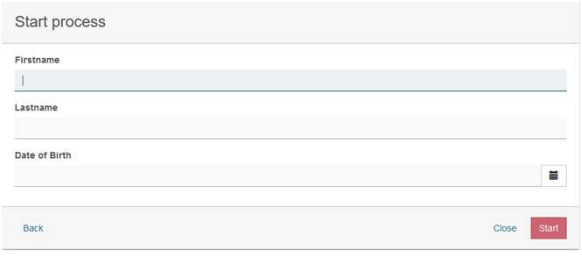
Task Forms in Camunda Tasklist

Generic form

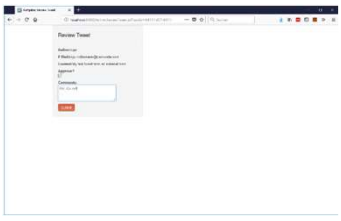
Embedded form



Generated form

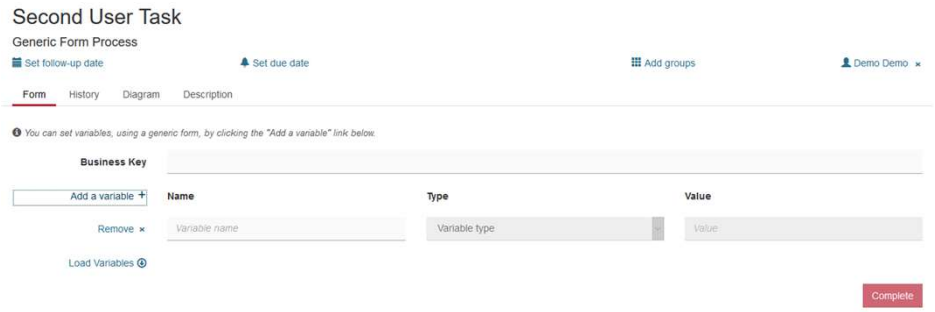


External form



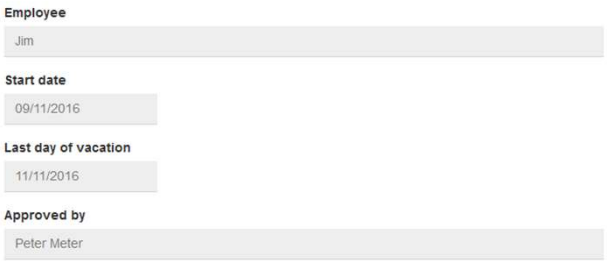
Generic Task Forms

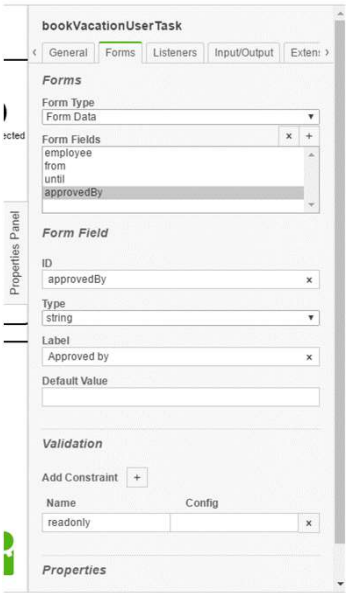
- `formKey=""`
- Completely generic, shows all process variables after load
- Add variables manually on demand
- Edit existing variables



Generated Task Forms (From Form Fields)

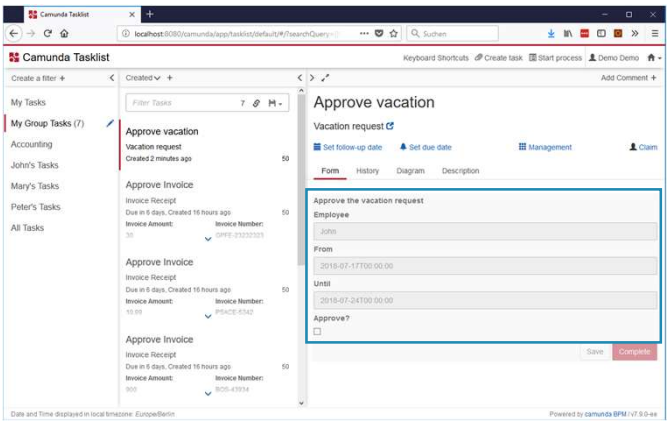
- Form Data Metadata provided by BPMN 2.0 XML
- Rendered in task list





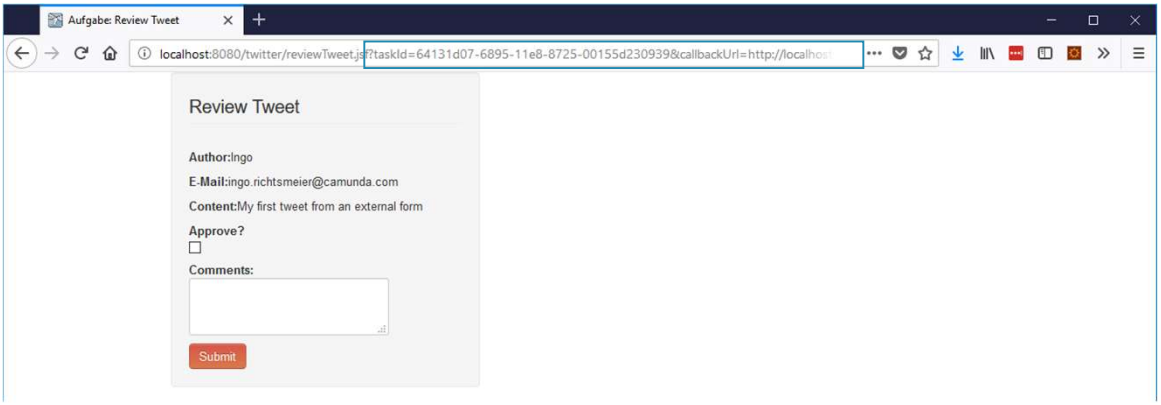
Embedded Task Forms

- `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

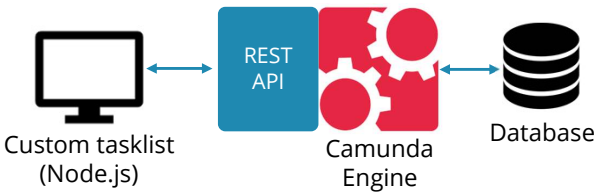
https://docs.camunda.org/manual/latest/reference/embedded-forms/

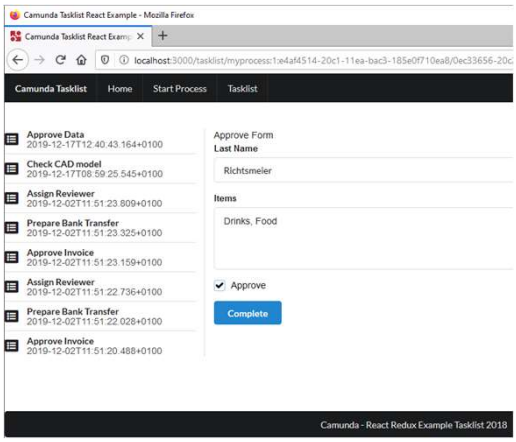
Simple Form Example

```
<strong>Here you can request your vacation</strong>

<form class="form-horizontal">
  <div class="control-group">
    <label class="control-label">Your Name</label>
    <div class="controls">
      <input type="text"
             cam-variable-name="employee"
             cam-variable-type="String"
             required
             class="form-control" />
    </div>
  </div>
  <div class="control-group">
    <label class="control-label">Start date</label>
    <div class="controls">
      <input type="text"
             cam-variable-name="from"
             cam-variable-type="Date"
             required
             class="form-control" />
    </div>
  </div>
  <div class="control-group">
    <label class="control-label">Last day of vacation</label>
    <div class="controls">
      <input type="text"
             cam-variable-name="until"
             cam-variable-type="Date"
             required
             class="form-control" />
    </div>
  </div>

  <script cam-script type="text/form-script">
    // custom JavaScript goes here
  </script>
</form>
```

Custom Tasklists

- Leverage the REST API
- Can use any technology (Vue.js, React, Angular, ...)
- Be creative with the Form Key

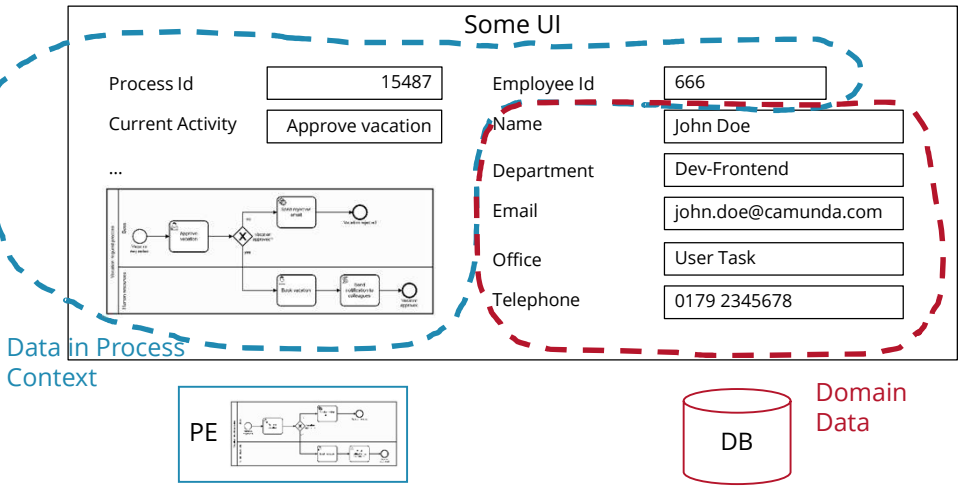




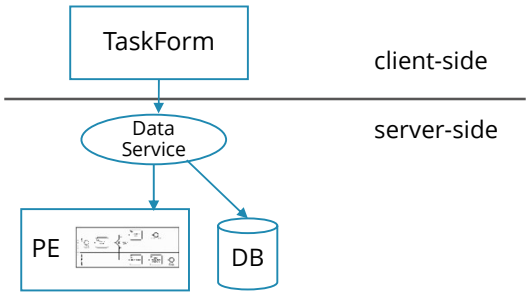
https://blog.camunda.com/post/2018/02/custom-tasklist-examples/

https://github.com/camunda-consulting/code/tree/master/snippets/camunda-tasklist-examples/

Typical: Process Data and External Data Are Mixed



How to Call Data Service

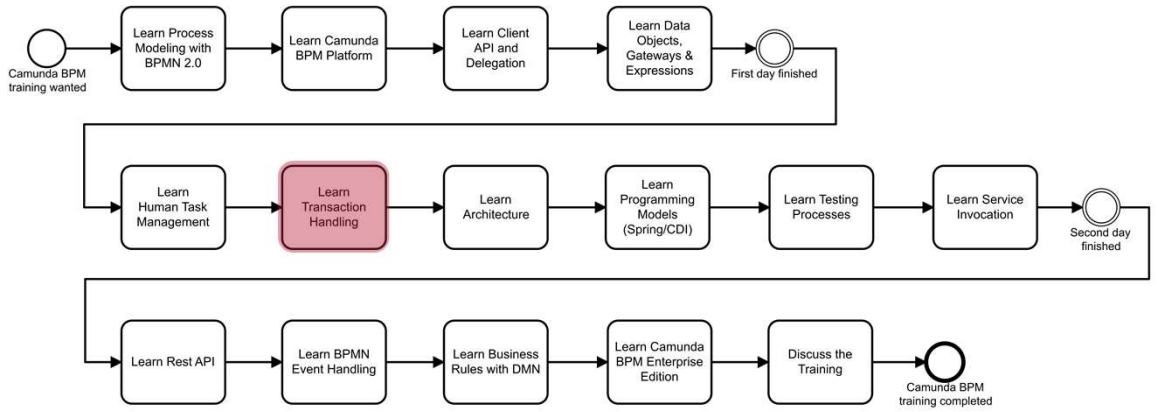


Composed Service

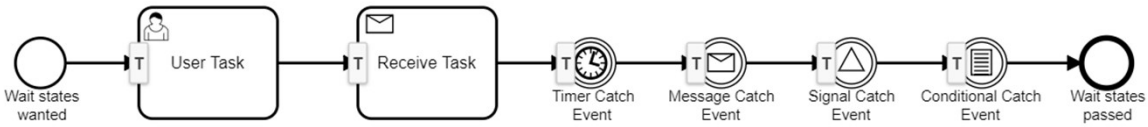
- The UI calls a combined service
- This service calls the Data Service and the Process Service
- TX possible

Exercise 6

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



Wait States

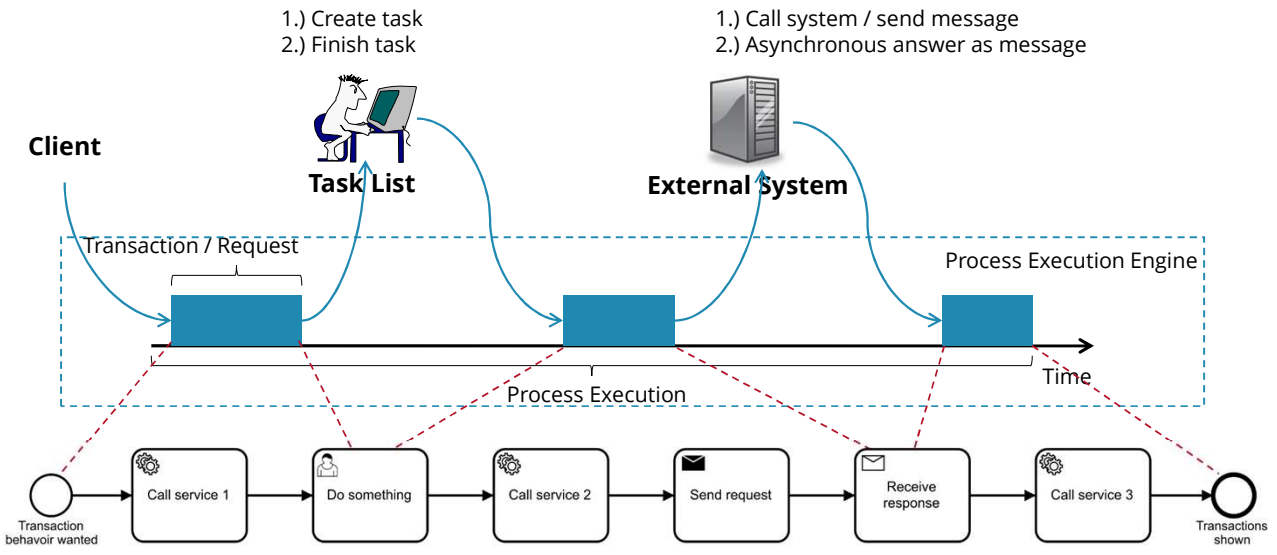


https://github.com/camunda/camunda-modeler-plugins/blob/master/camunda-transaction-boundaries-plugin

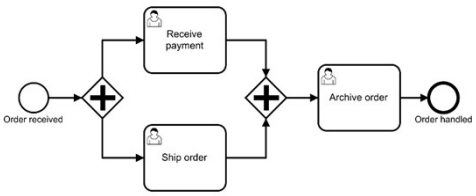
Also called:

- Transaction Boundaries
- Save Points

Process Engine and Transactions



Parallel Gateway – Engine Internals



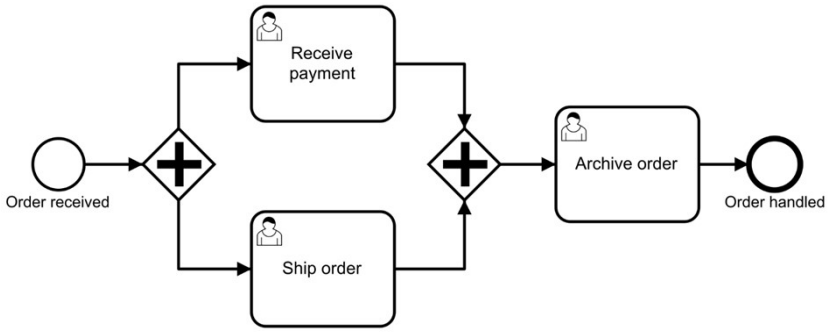
Splitting gateway:

- For every outgoing sequence flow:
  - Move the token until it can't go any further
- Then commit transaction

Joining gateway:

- An incoming sequence flow token arrives:
  - Check number of tokens waiting at joining gateway
    - If one or more tokens is missing
      - Wait and
      - If no other token is active in the transaction, commit
    - If token from last sequence flow arrives, continue

How Many Transactions?



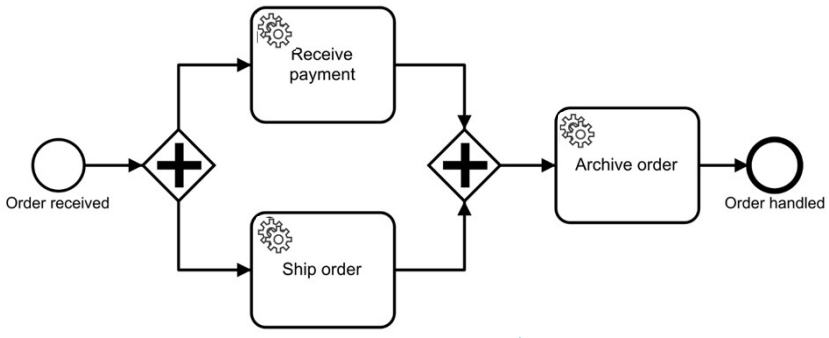
How Many Transactions?

Hidden solution

How Many Transactions? (The Other Way Around)

Hidden solution

How Many Transactions?



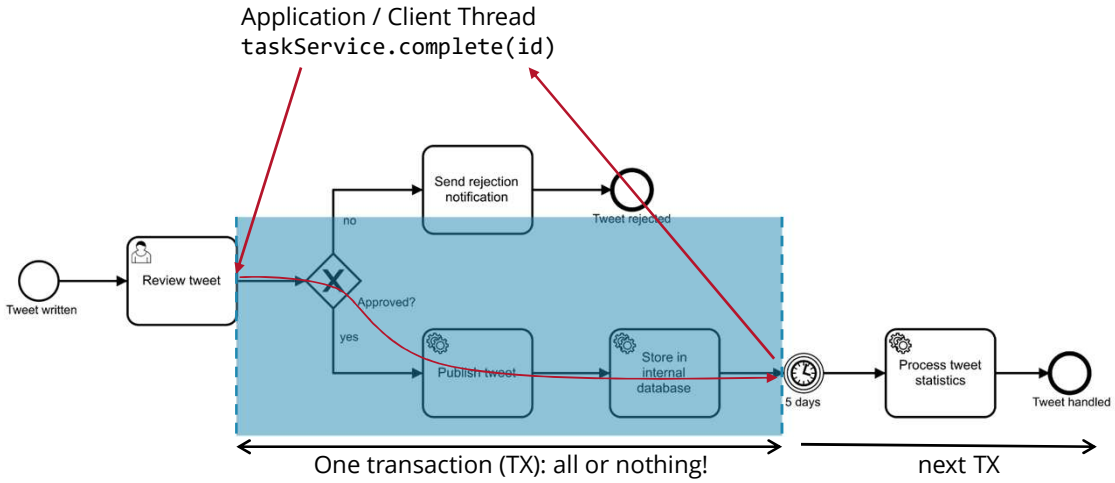
What if all tasks were Service Tasks?

How Many Transactions?

Hidden solution

What if all tasks were Service Tasks?

Transaction Boundaries

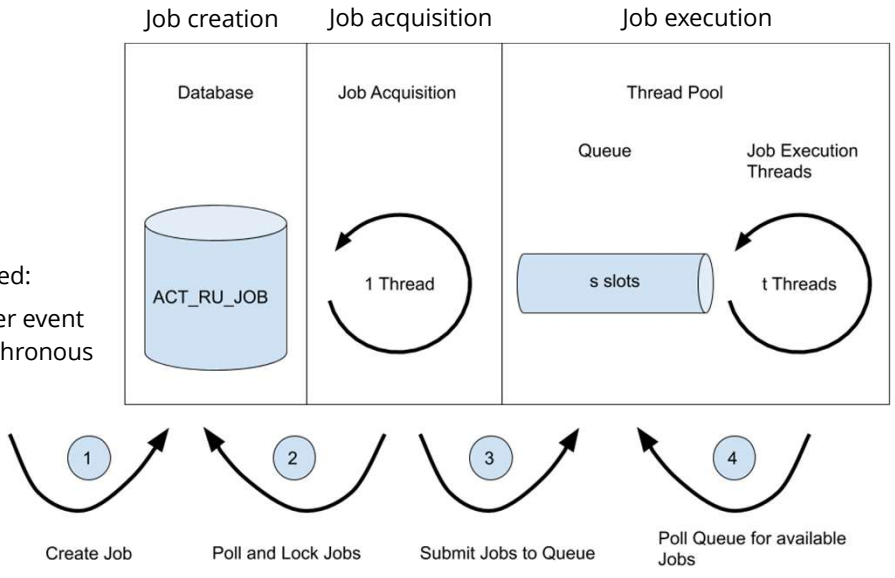


Demo

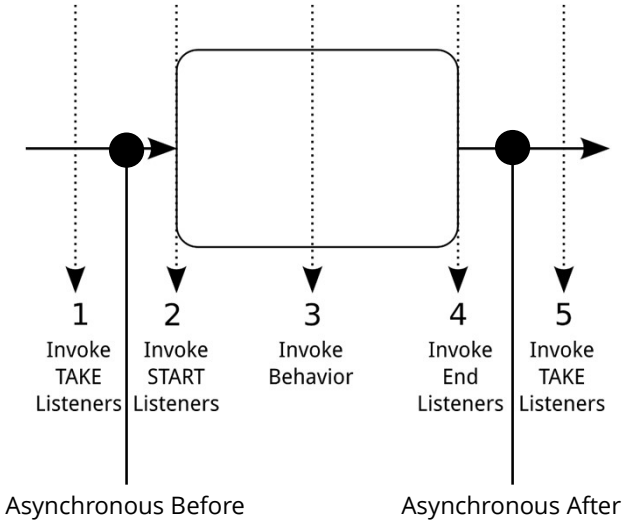
The Job Executor

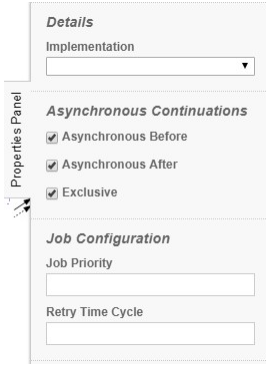
A Job is created:

- For a timer event
- For asynchronous execution

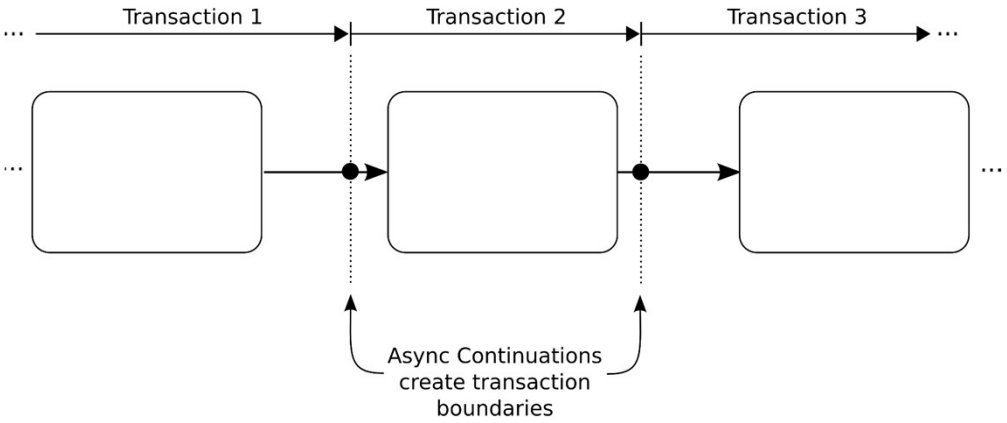


Asynchronous Continuation: Execution of a BPMN Element



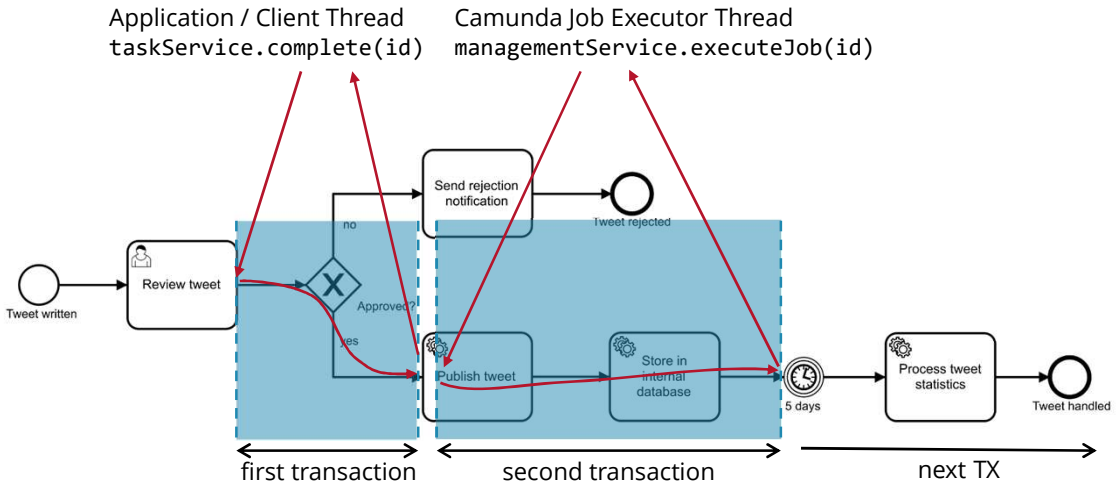


Asynchronous Continuation = Transaction Boundaries/Save Points



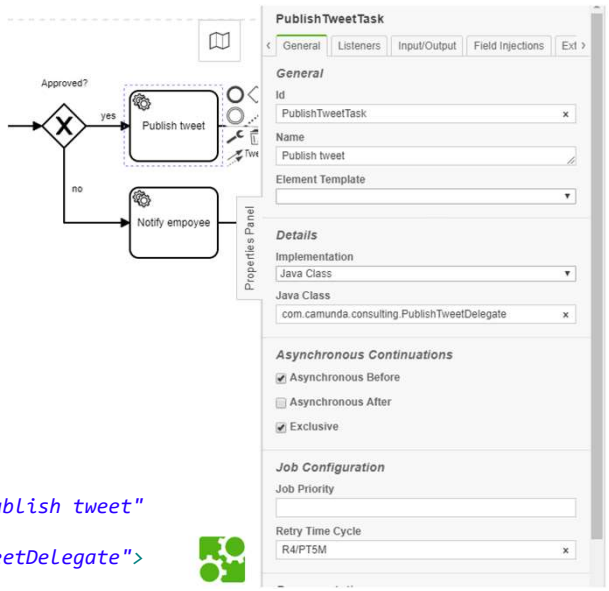

```
<bpmn:serviceTask id="publishTweetServiceTask" name="Publish tweet" camunda:asyncBefore="true"
camunda:asyncAfter="true" camunda:class="com.camunda.consulting.PublishTweetDelegate"/>
```

Transaction Boundaries




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


Retry Strategy




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

Async Continuation in JUnit Test

- In detail:

```
List<Job> jobList = managementService
    .createJobQuery()
    .processInstanceId(processInstanceId)
    .list();
assertThat(jobList).hasSize(1);
Job job = jobList.get(0);
managementService.executeJob(job.getId());
```

- camunda-bpm-assert short cut:

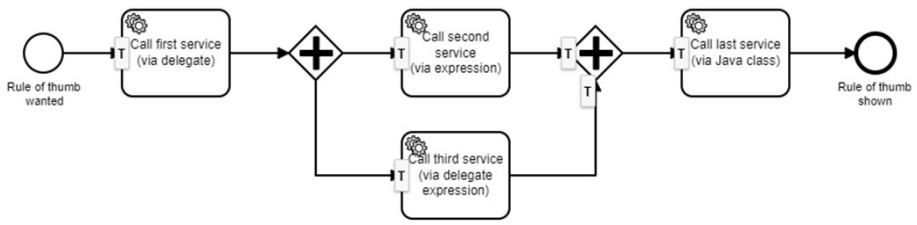
```
assertThat(processInstance).isWaitingAt("PublishTweetTask");
execute(job());
```

- In case of parallel flows:

```
assertThat(processInstance).isWaitingAt("ParallelTask1");
Job job = jobQuery().activityId("ParallelTask1").singleResult();
execute(job);
```

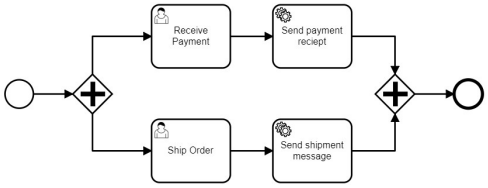

Rule of Thumb for Save Points

- Before each service task that is not an external service task
- Before each joining parallel gateway



Detailed 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



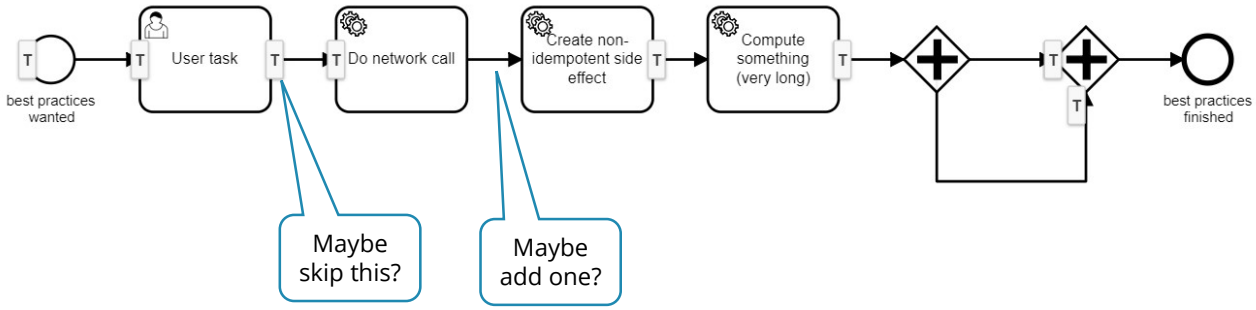
NOT

- 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

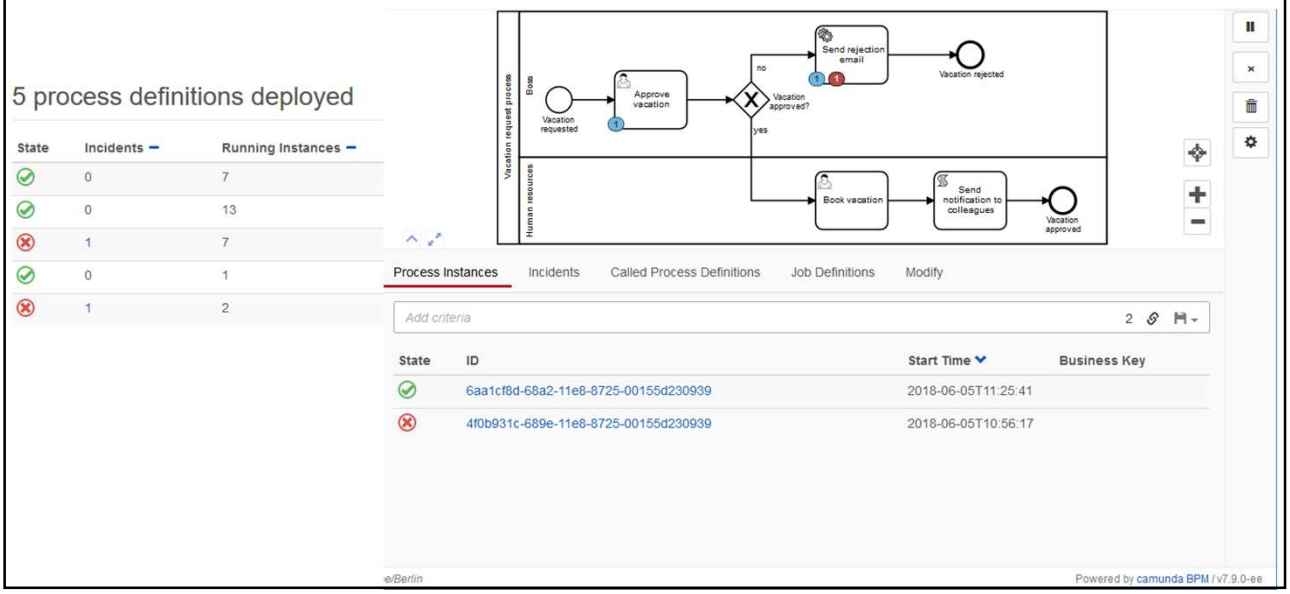
- Listeners, beans or complex expressions, e.g. using SPIN, that could fail are involved

Best Practices for Save Points

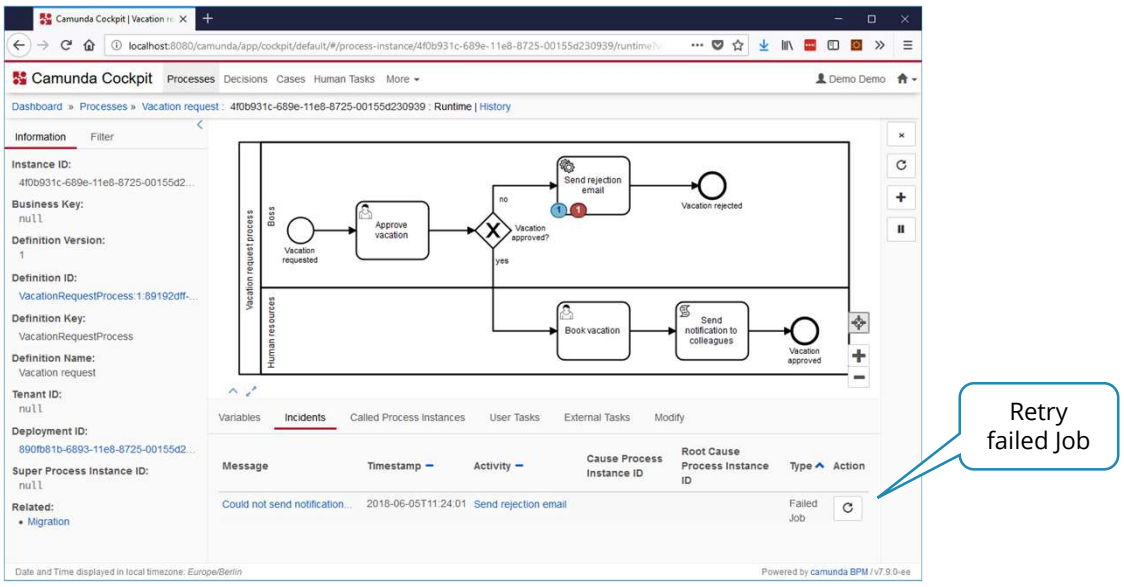


Add save points automatically:
https://github.com/camunda-consulting/code/tree/master/snippets/engine-plugin-add-save-points
https://github.com/camunda-consulting/code/tree/master/snippets/async-joins

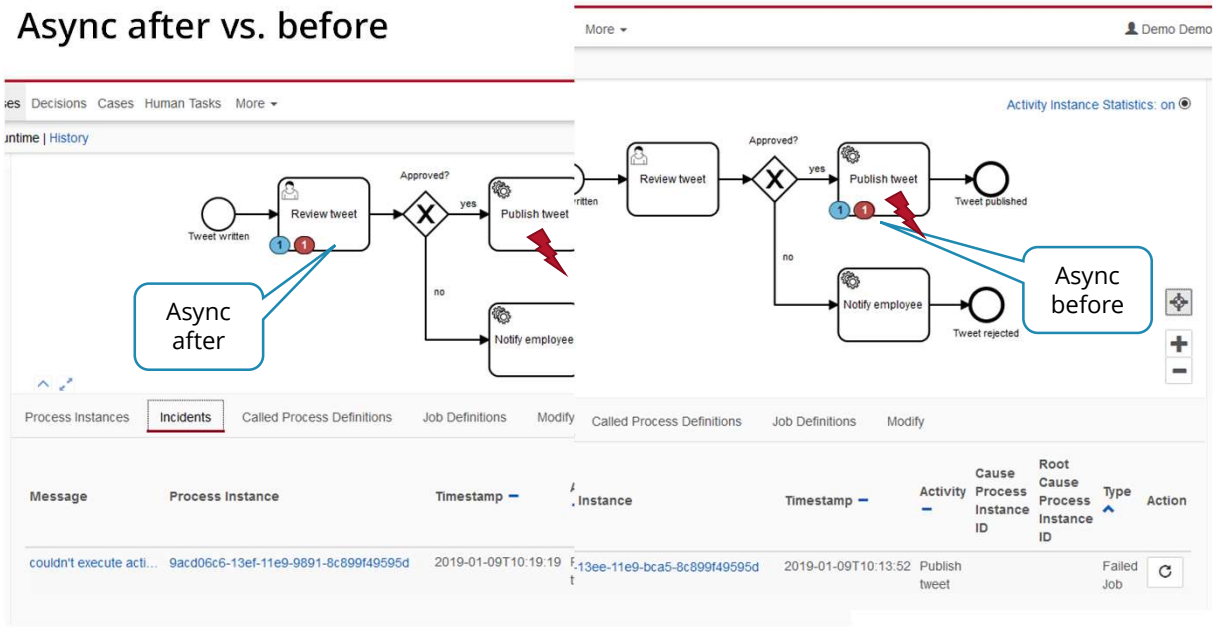
Incidents Overview



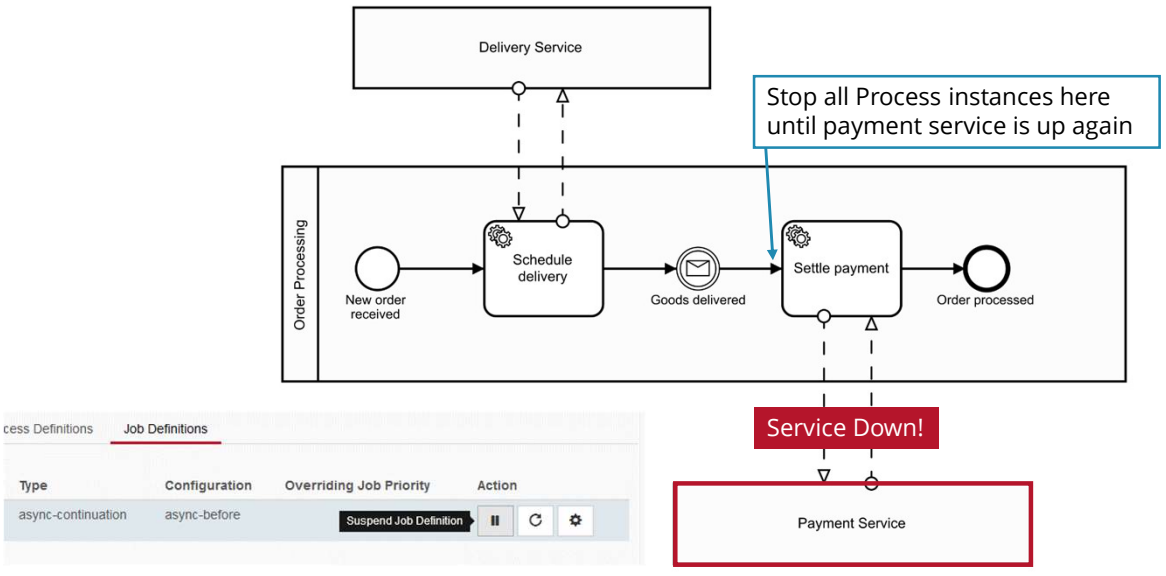
Incidents Details



Async after vs. before

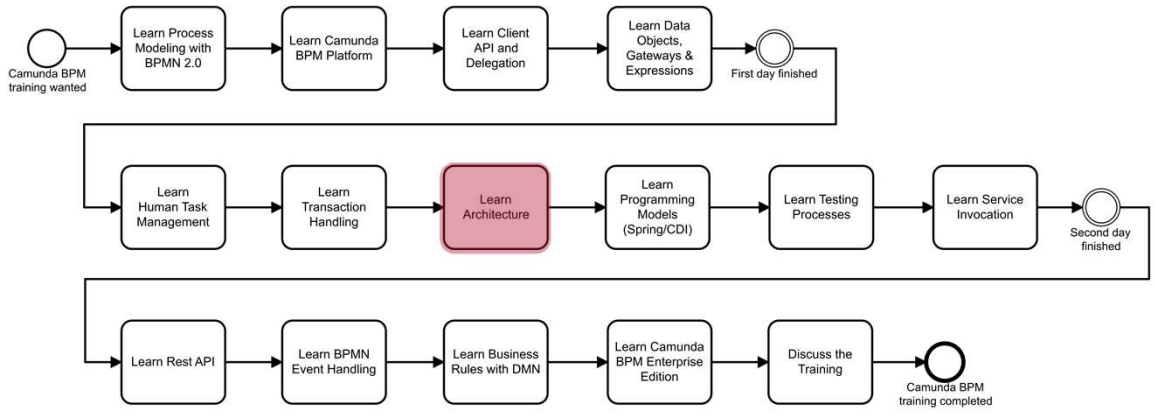


Suspension

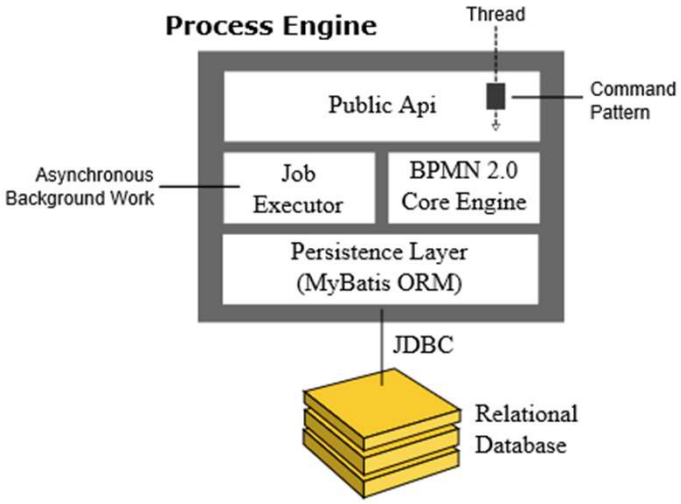


Exercise 7

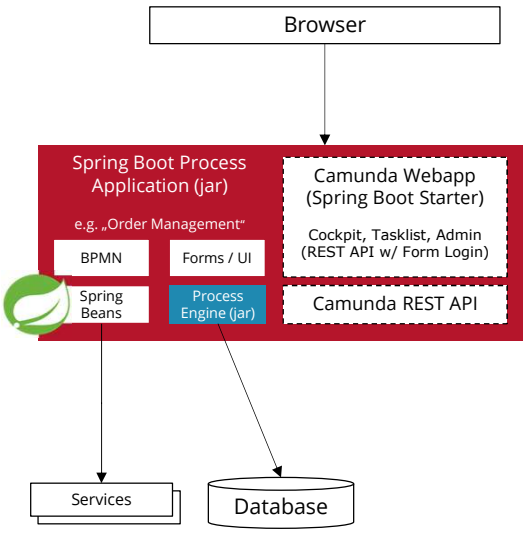
https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



Process Engine Architecture



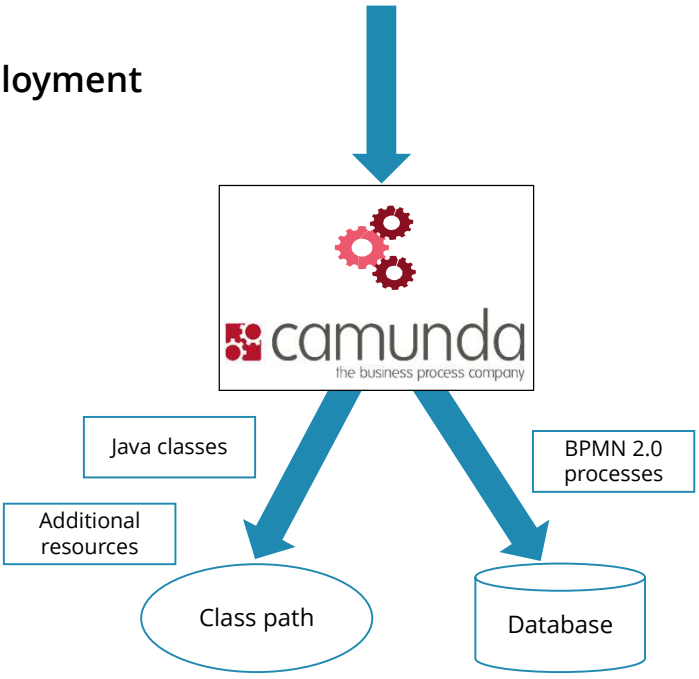
Spring Boot: Embedded Process Engine, Webapp & REST API



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();
```

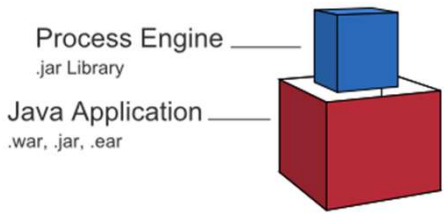

Process Deployment



Deploying Programmatically With Java

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


Embedded Process Engine



Various environments possible:

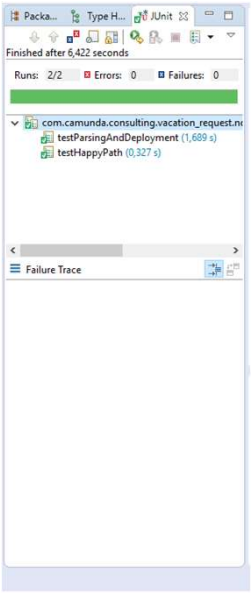
most popular

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();
  }
}
```

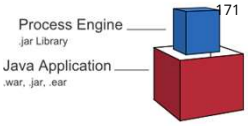


Process Engine as a Spring Component

```
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



My process application

- Java classes
- BPMN 2.0 processes
- Process Engine (jar)
- ...

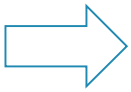
Multiple Process Applications

My process application

- Java classes
- BPMN 2.0 processes
- Process Engine
- ...

My process application2

- Java classes
- BPMN 2.0 processes
- Process Engine
- ...



My process application

- Java classes
- BPMN 2.0 processes

My process application2

- Java classes
- BPMN 2.0 processes

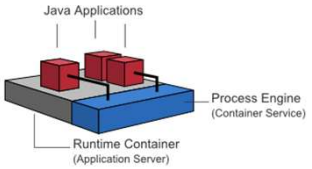
...

Process Engine | JPA, EJB, JTA | ...

Container

Supported on:

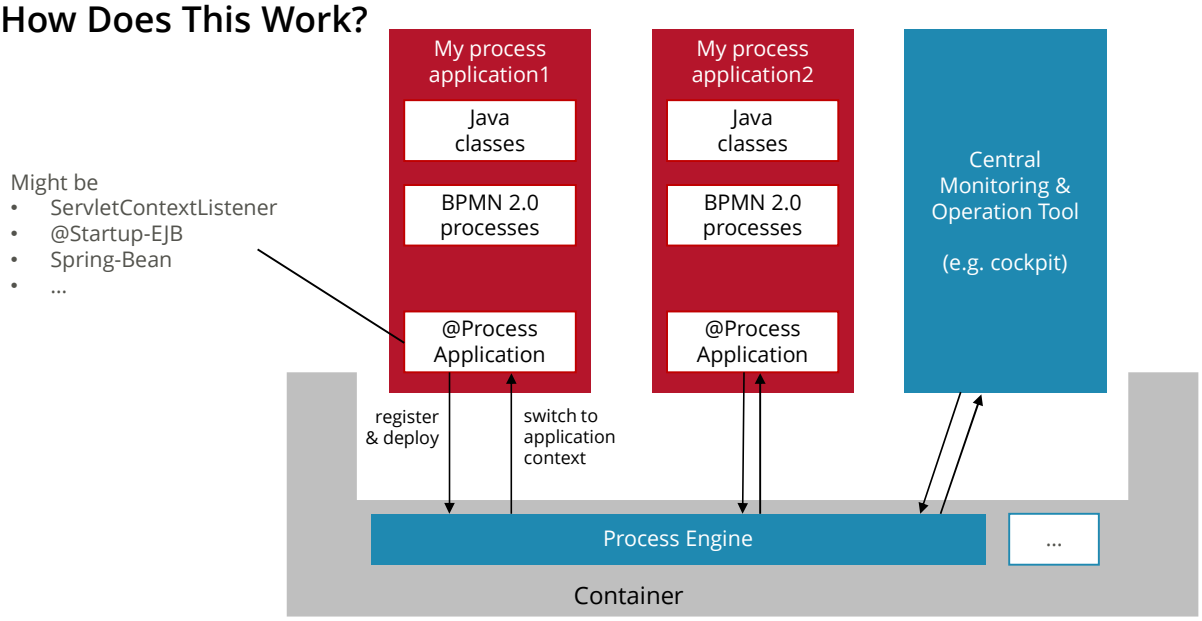
Tomcat, JBoss AS/EAP, Wildfly AS, WebSphere AS and WebLogic



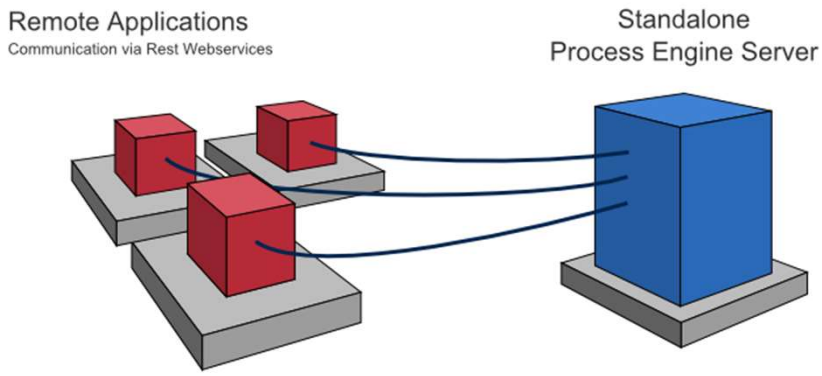
Camunda Process Application: processes.xml

```
<?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?

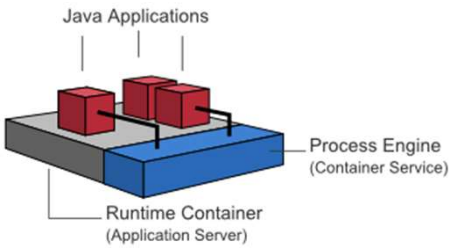


Standalone / Remote Process Engine Server

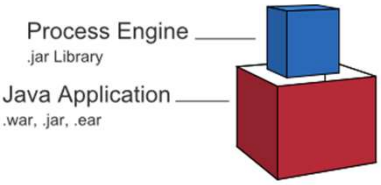


Process Engine Modes

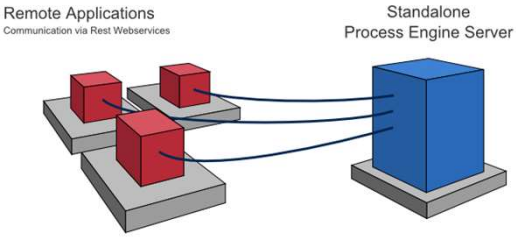
Shared, Container Managed Process Engine



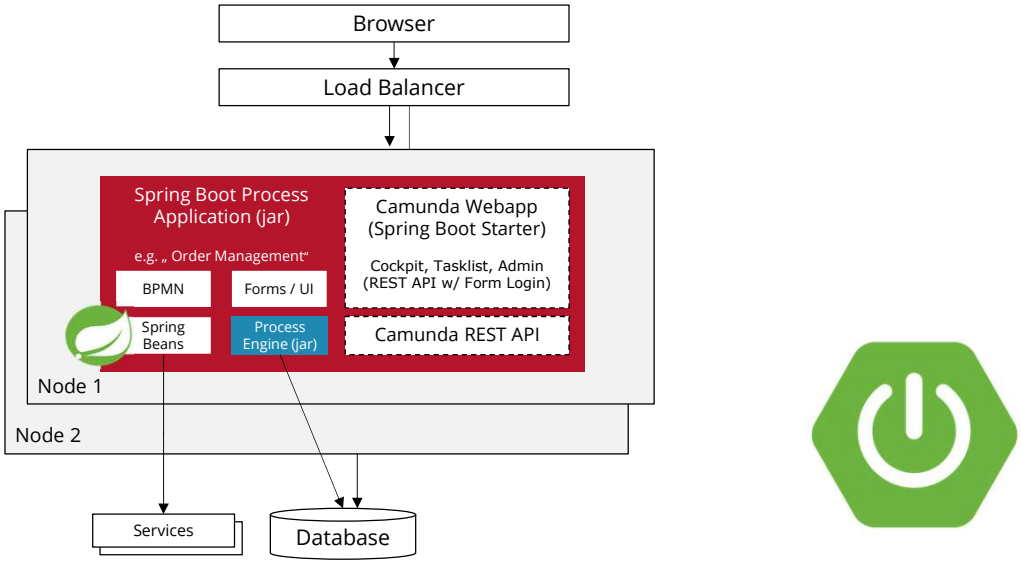
Embedded Process Engine



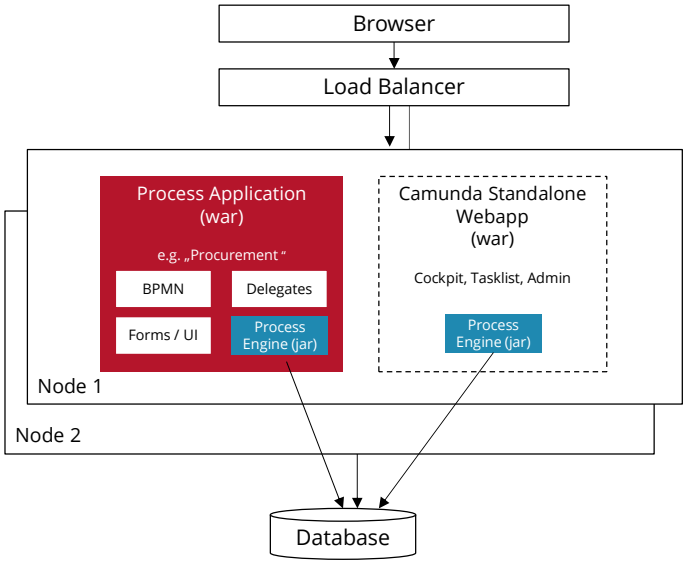
Remote Process Engine



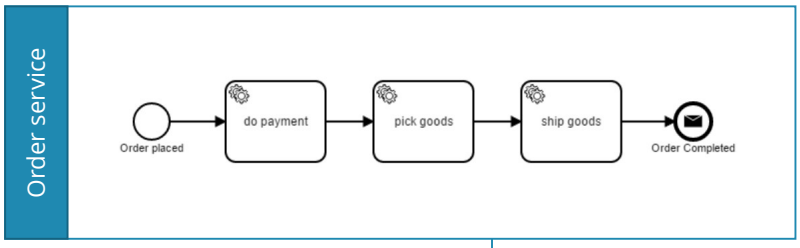
Spring Boot: Embedded Process Engine, Webapp & REST API



Example Architecture With Embedded Process Engines

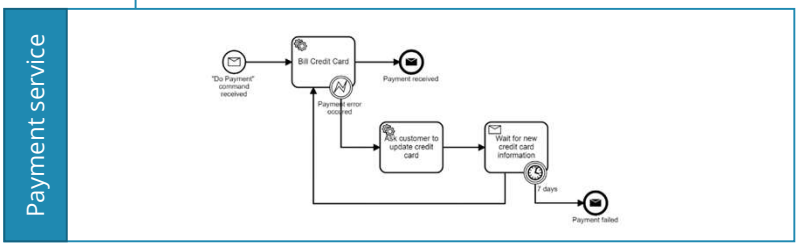


Orchestration With Microservices

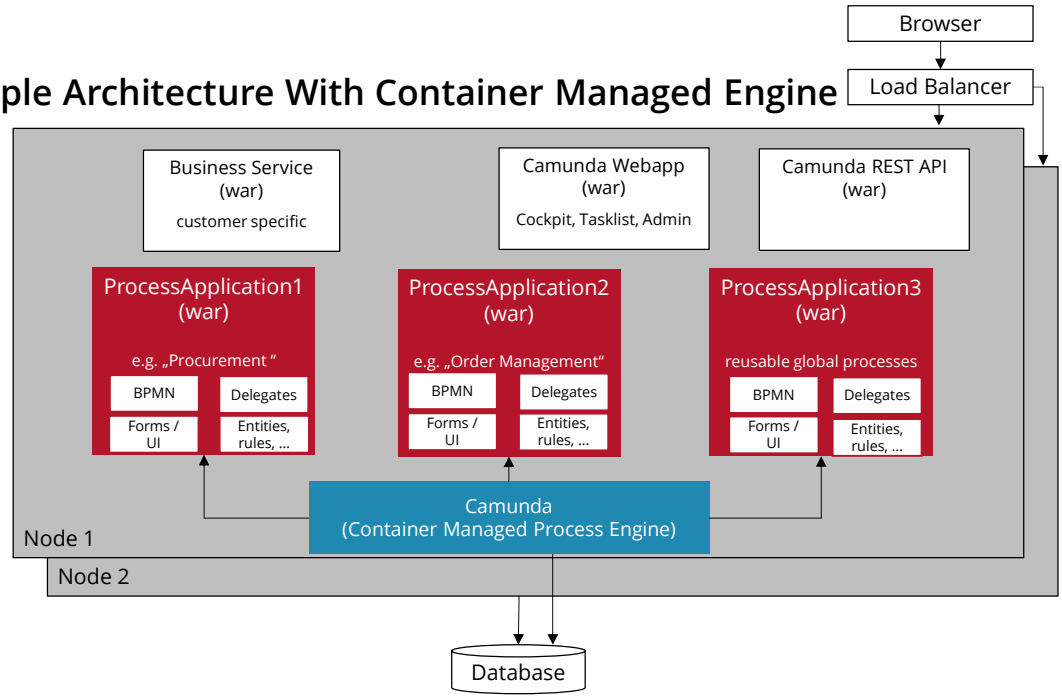


Possible Environments:
- Embedded engines
- Container managed engines with single process application

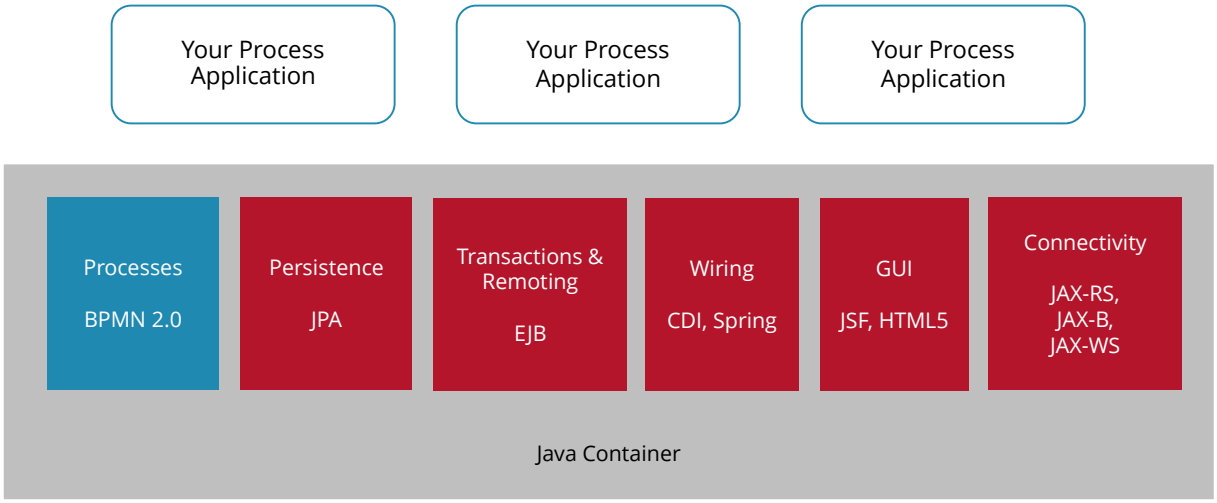
Maybe you add:
- History event stream for central history
- Task events for central task list

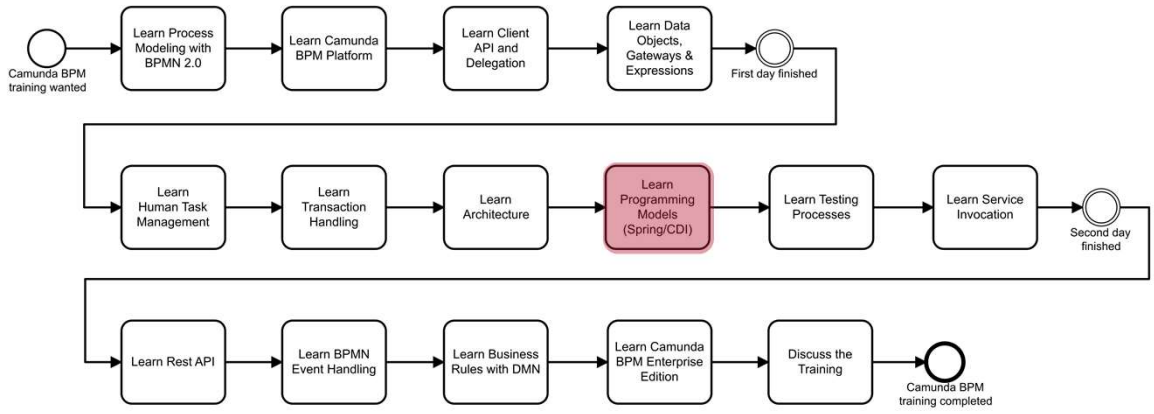


Example Architecture With Container Managed Engine



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)

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 `@Component` qualifier allows to give a bean an Expression Language name:

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

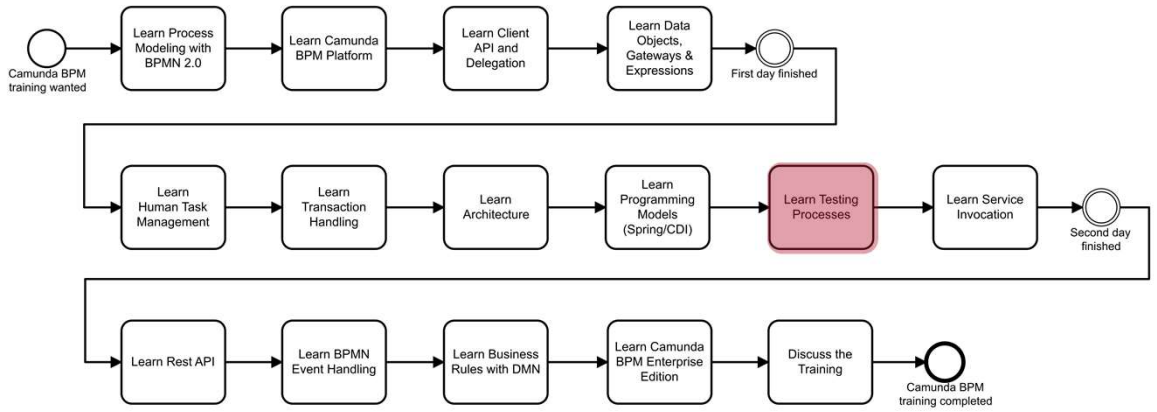
The El-Name of this bean is “simpleGreeter”. This allows us to invoke the bean from an Camunda BPM process:

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

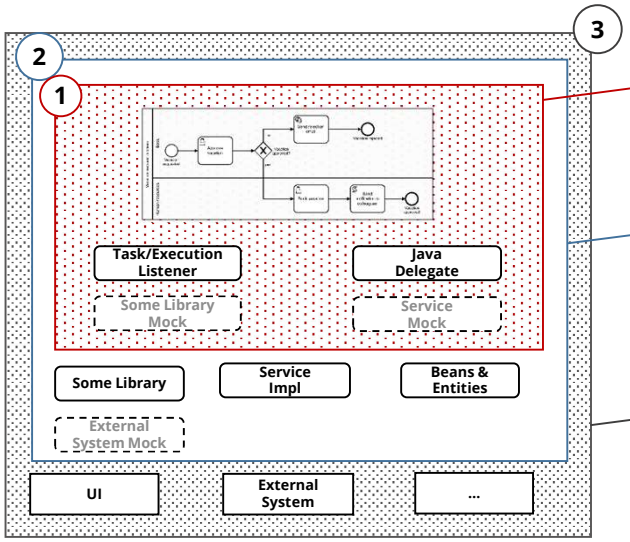
The resolved object is managed by the container (Spring/CDI) and can profit from other services (injection / lifecycle / …).

Delegate Expression


```
@Component
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

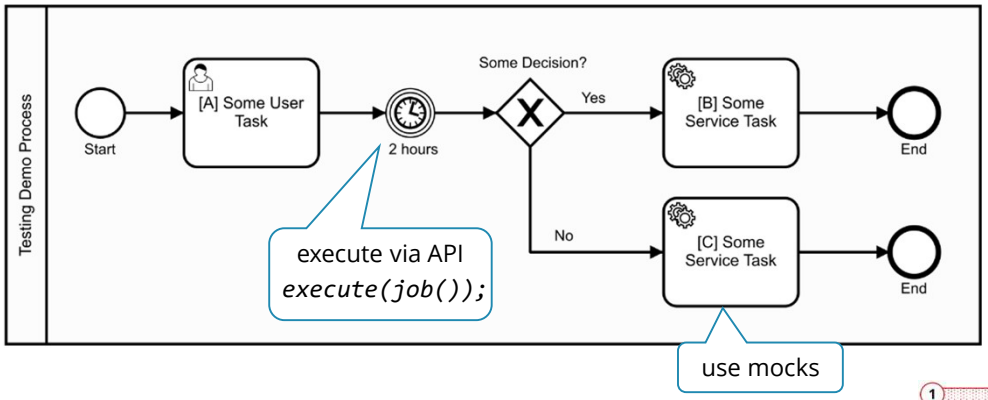


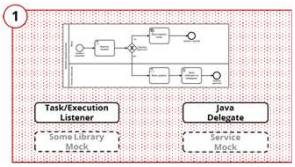
Goal: Process Model with Data, EL & Adapter Logic
- In Memory
- Single Thread
- No Container
- No JobExecutor

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

Goal: It REALLY works
- Integration Tests
- Human Driven

Example





Separate Process- and Business Logic

Separate business- from process logic

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

  private EmailService emailService;

  @Inject
  public SendRejectionNotificationDelegate(EmailService emailService) {
    this.emailService = emailService;
  }

  @Override
  public void execute(DelegateExecution execution) throws Exception {
    // Input mapping
    String reason = (String) execution.getVariable("rejectionReason");
    String employee = (String) execution.getVariable("employee");
    String message = "Some message to " + employee + " about " + reason;
    // Service call
    String emailId = emailService.sendEmail(employee, "manager", reason, message);
    // Output mapping
    execution.setVariable("message", message);
    execution.setVariable("emailId", emailId);
  }
}
```

Field injection is evil

```java
@Service
public class EmailService {

  public String sendEmail(...) {
      // Connect to mail server
      // send the email
      // maybe receive an mail id
      return messageId;
    }
}
```

This is what we want to mock

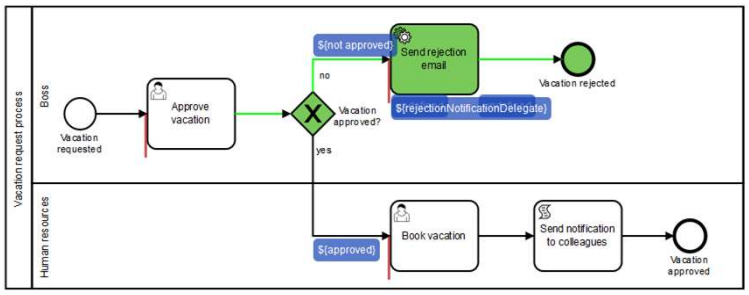
Mock the Business Logic

```
@Mock
private EmailService emailService;

@Before
public void setup() {
  init(rule.getProcessEngine());

  MockitoAnnotations.initMocks(this);
  Mocks.register("rejectionNotificationDelegate", new SendRejectionNotificationDelegate(emailService));
}

@Test
@Deployment(resources = "vacation-request.bpmn")
public void testVacationNotApproved() {
  Mockito.when(emailService.sendEmail(anyString(), anyString(), anyString(), anyString()))
    .thenReturn("mockEmailId");
  ProcessInstance processInstance = runtimeService()
      .createProcessInstanceByKey("VacationRequestProcess")
      .setVariables(withVariables("employee", "John", "from", new Date(), "to", new Date(),
          "approved", false, "rejectionReason", "some reason"))
      .startAfterActivity("approveVacationUserTask")
      .execute();
  assertThat(processInstance).isEnded().hasPassed("vacationRejectedEndEvent")
    .variables().containsEntry("emailId", "mockEmailId");
  Mockito.verify(emailService).sendEmail(eq("John"), anyString(), eq("some reason"), anyString());
}
```

Register Delegate with mocked service

Assert variable passing between service and process

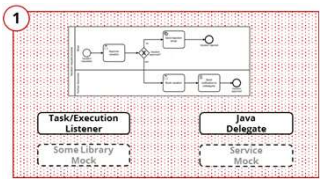
Real Unit Test




```
@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 1



We recommend

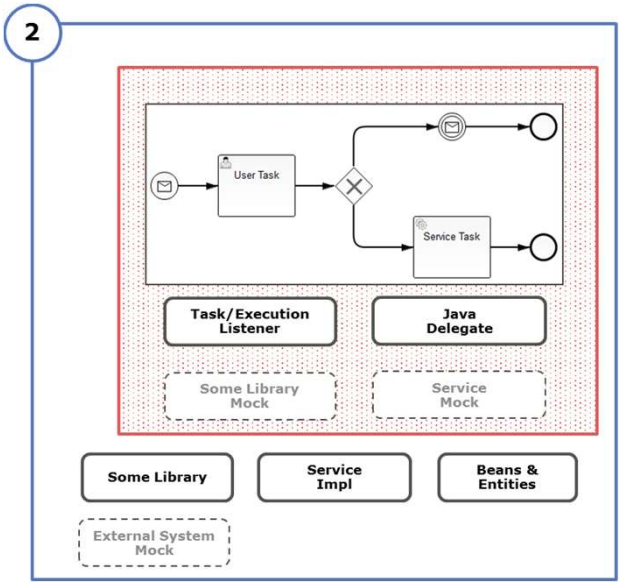
- JUnit
- Camunda Rule (included in camunda-engine)
- Camunda-bpm-assert (https://github.com/camunda/camunda-bpm-assert)
- Mock-Provider of your choice, e.g. Mockito or EasyMock or Jmockit (maybe + PowerMock)
- Process Instance Modification to start before the unit to test

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”!

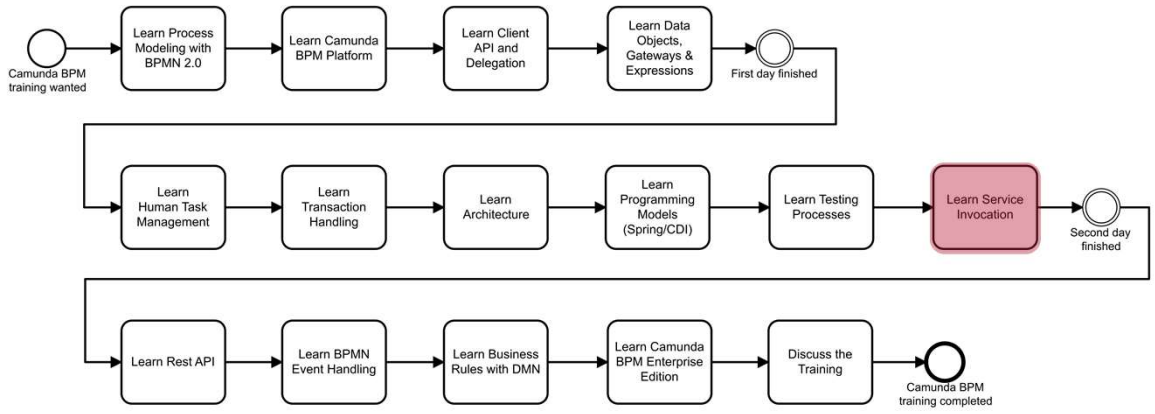
Testing on Scope 2

- 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

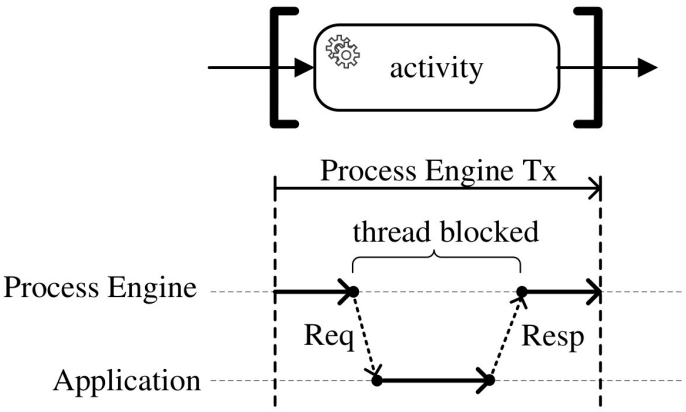


Exercise 8

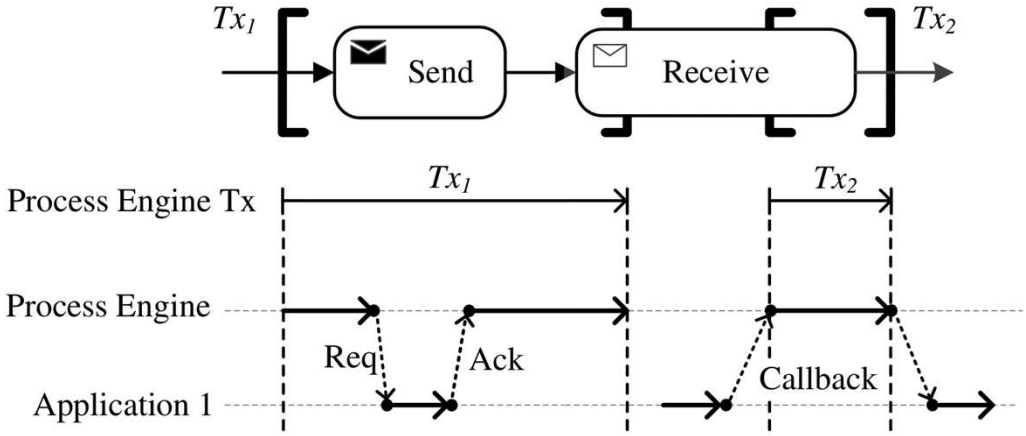
https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



Synchronous Service Invocation: Request / Response

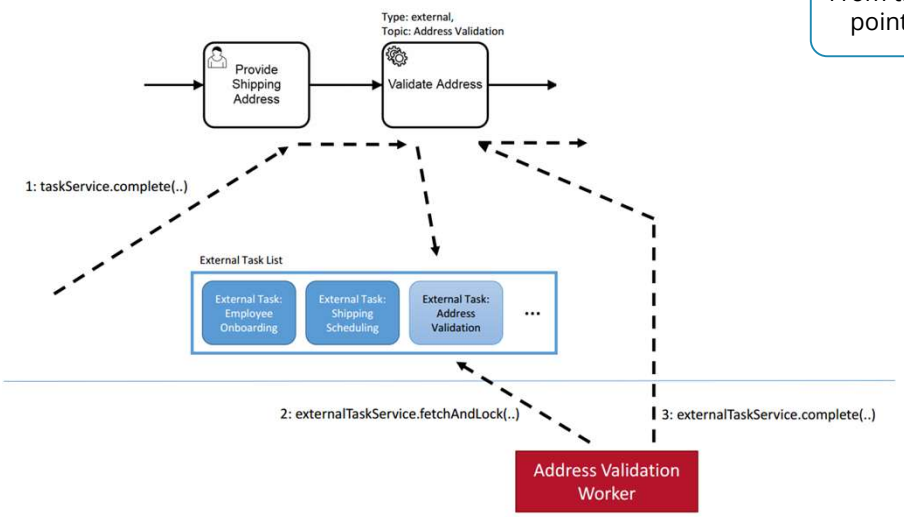


Asynchronous Service Invocation: Req / Ack / Callback



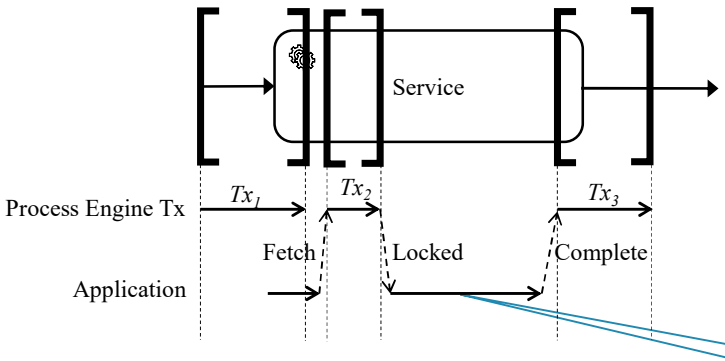
External Service Task: Don't Call Us! We Call You!

From the service point of view



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

External Service Task Transactions

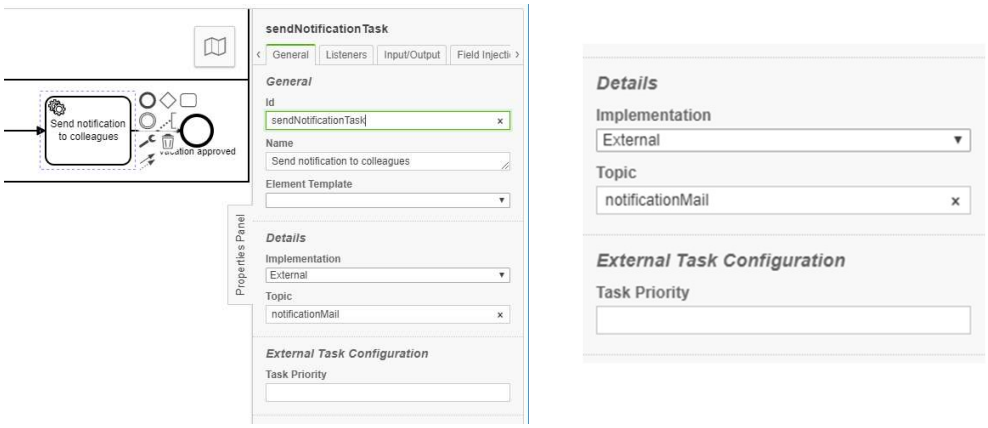


External tasks are wait states, transaction boundaries, save points

(very) long running services

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

External Task Configuration



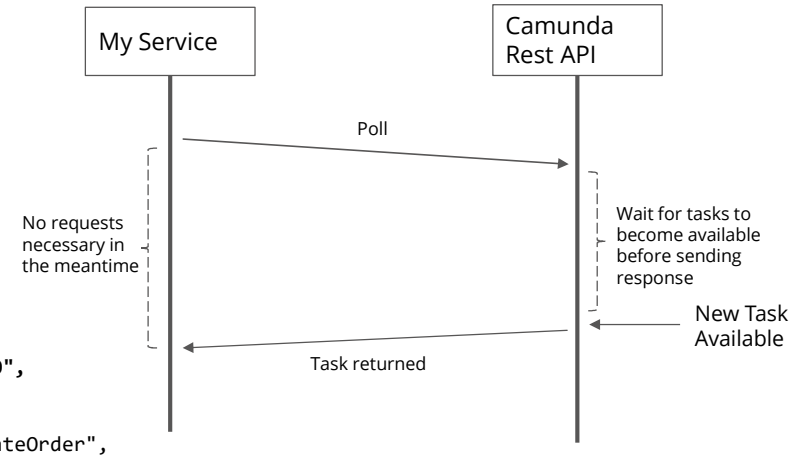
External Task Client for Java Features

- Complete External Tasks
- Extend the lock duration of External Tasks
- Unlock External Tasks
- Report BPMN errors and failures
- Share variables with the Workflow Engine

https://docs.camunda.org/manual/latest/user-guide/ext-client/

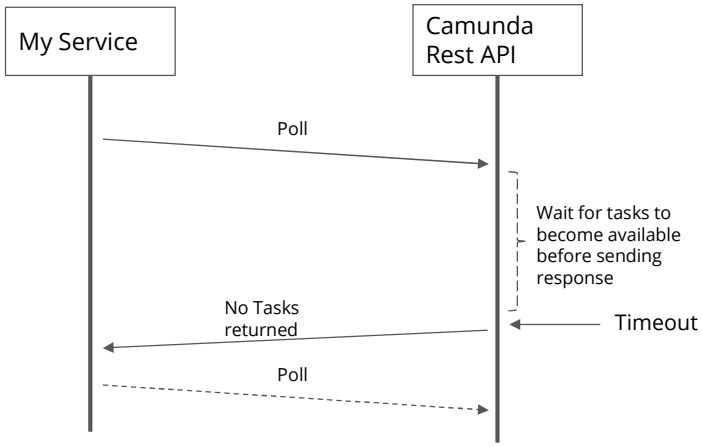
Long Polling

Reduce latency between create and work

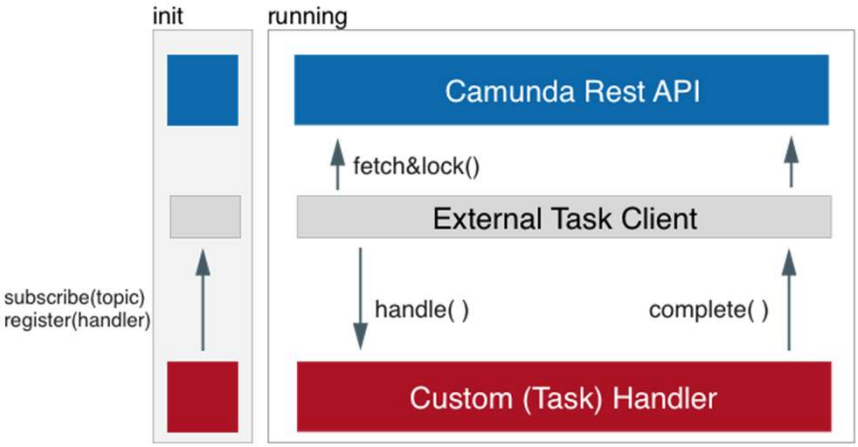



```
POST /external-task/fetchAndLock

Request Body:

{
   "workerId": "myService",
   "asyncResponseTimeout": "60000",
   "maxTasks": 1,
   "usePriority": true,
   "topics": [{"topicName": "createOrder",
   "lockDuration": 10000, "variables": ["orderId"]}]
}
```


What if no task is created?



External Task Client



External Task Client Setup

```
<properties>
  <camunda-ext-task-client.version>1.1.0</camunda-ext-task-client.version>

  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <version.java>1.8</version.java>
</properties>


<dependencies>
  <dependency>
    <groupId>org.camunda.bpm</groupId>
    <artifactId>camunda-external-task-client</artifactId>
    <version>${camunda-ext-task-client.version}</version>
  </dependency>

  <!-- use logback as logger -->
  <dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.1.2</version>
  </dependency>
</dependencies>
```

Implementation

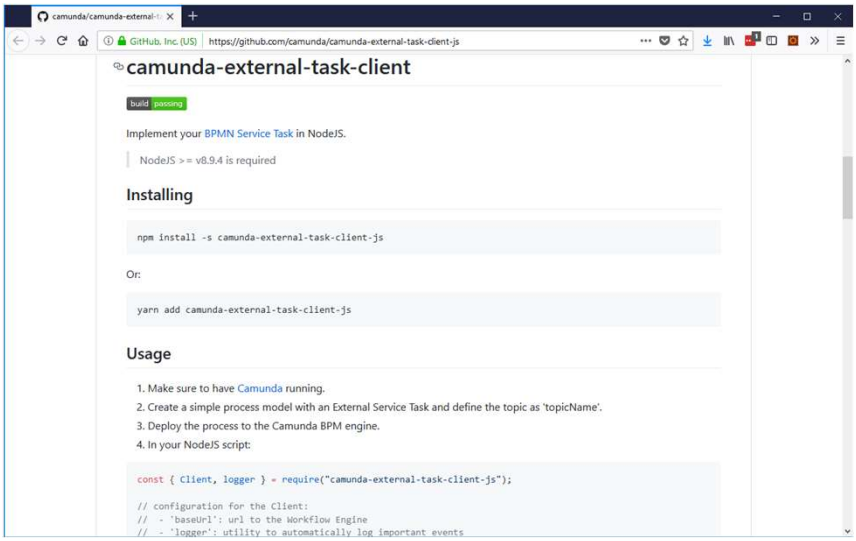
```
public static void main(String[] args) {
   // bootstrap the client
   ExternalTaskClient client = ExternalTaskClient.create()
     .baseUrl("http://localhost:8080/rest")
     .lockDuration(60000)
     .maxTasks(2)
     .build();

   // subscribe to the topic
   TopicSubscriptionBuilder subscribtionBuilder = client.subscribe("notificationMail");
   subscribtionBuilder
     .handler((externalTask, externalTaskService) -> {
       String receiver = externalTask.getVariable("receiver");
       String subject = externalTask.getVariable("subject");
       String message = externalTask.getVariable("message");
       if (mailServerAvailable()) {
         LOGGER.info("send {} with subject {} to {}", message, subject, receiver);
         Map<String, Object> variables = new HashMap<>();
         variables.put("notficationTimestamp", new Date());
         externalTaskService.complete(externalTask, variables );
       } else {
         LOGGER.info("unlock the task as mailserver is unavailable");
         externalTaskService.unlock(externalTask);
       }
     }).open();
 }
```

```
// Long polling
.asyncResponseTimeout(90000)
// reduce backoff to max 8 sec
.backoffStrategy(
    new ExponentialBackoffStrategy(500, 2, 8000))
```

Business Logic

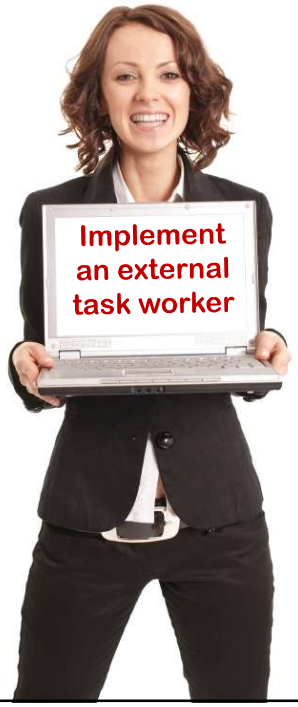
Called sequentially for each task

External Task Client in JavaScript

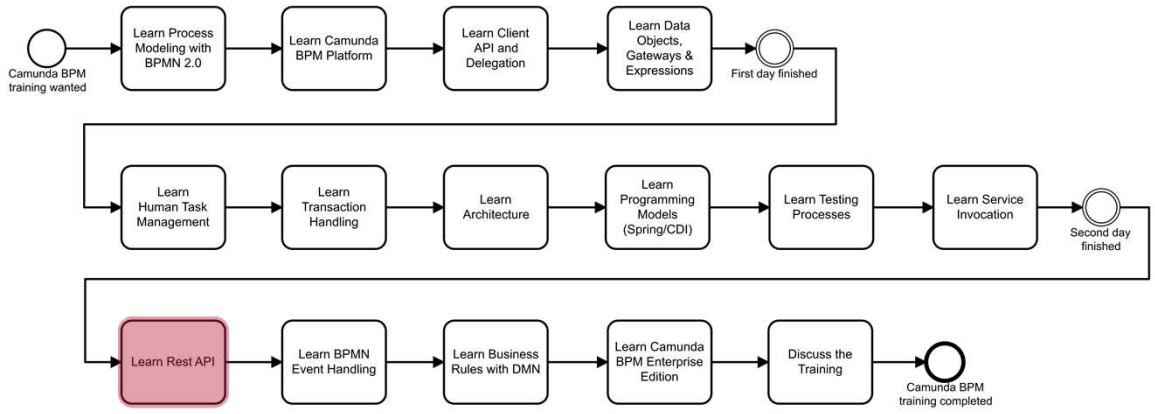


https://github.com/camunda/camunda-external-task-client-js

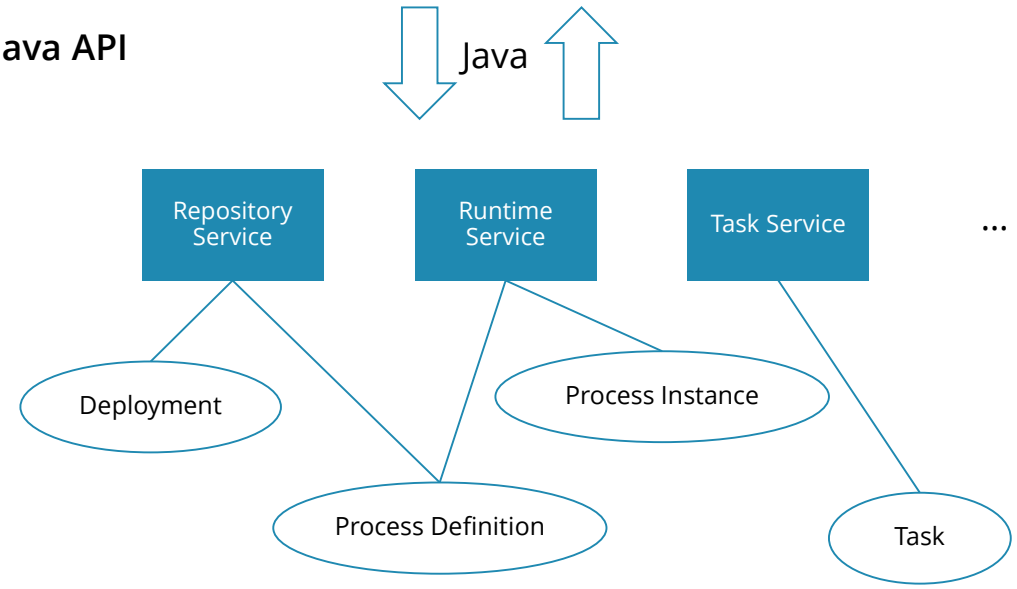
Exercise 9



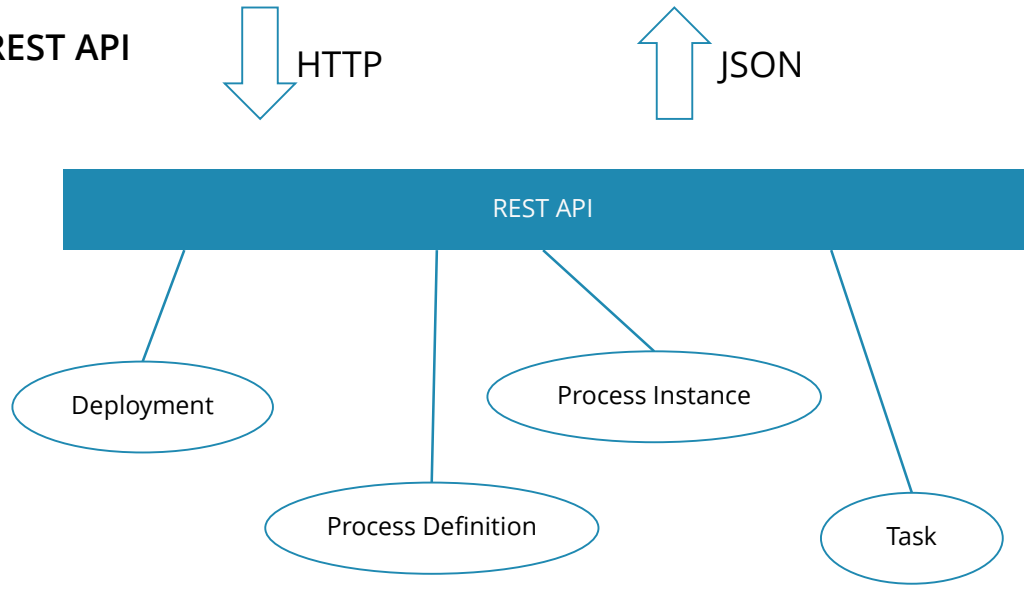
https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



Java API

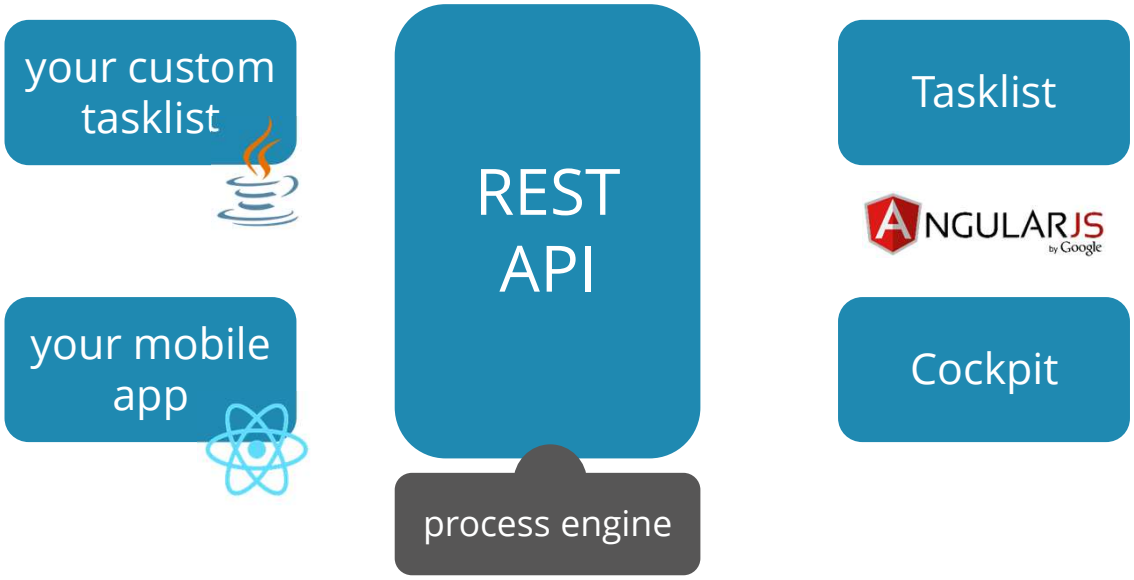


REST API



- Same power as Java API

Example Usage Scenarios



Camunda BPM REST API

Some REST calls:

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

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

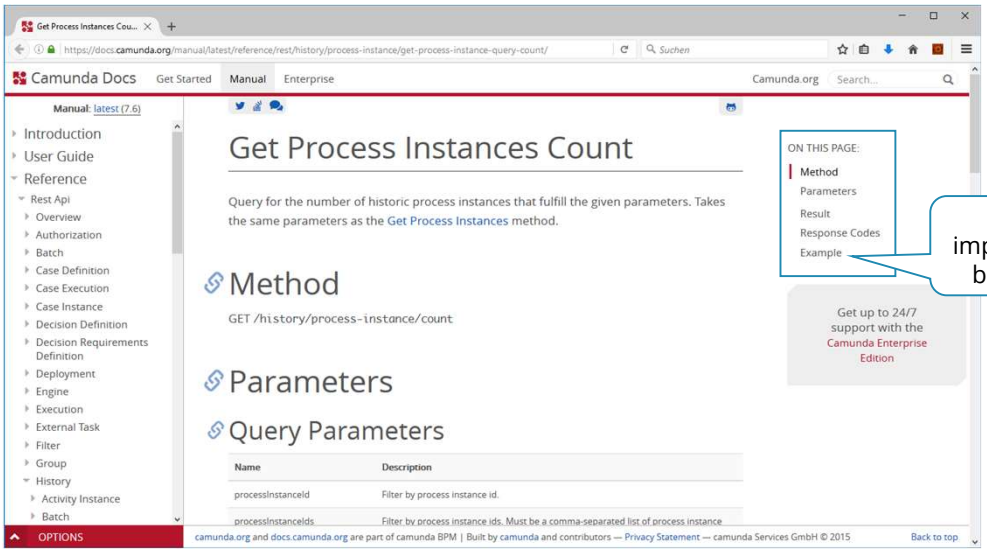
```
{"variables":
    {"employee" : {"value" : "Mary", "type": "String"},
     "from" : {"value" : "2016-12-20T00:00:00", "type": "Date"},
     "until" : {"value" : "2016-12-28T23:59:00", "type": "Date"}
    }
}
```

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

Endpoint for prepackaged distribution: `http://localhost:8080/engine-rest/`

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

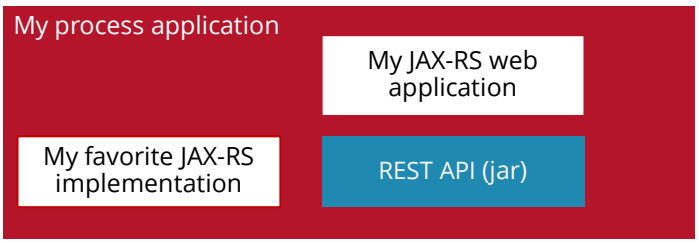
Documentation



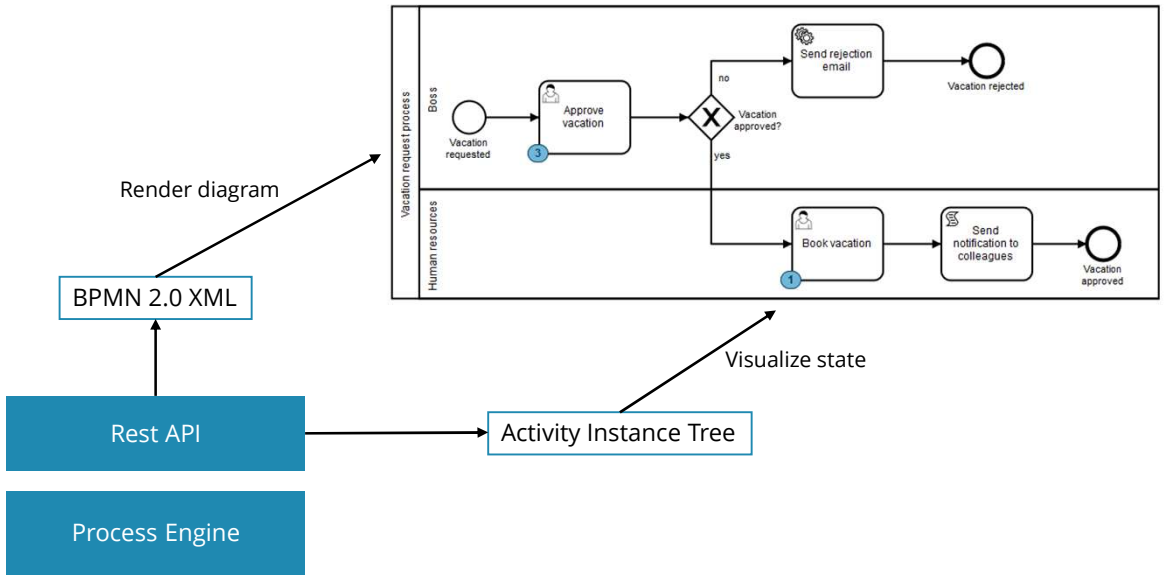
REST API: Embed and Extend!

- 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);
  }
}
```



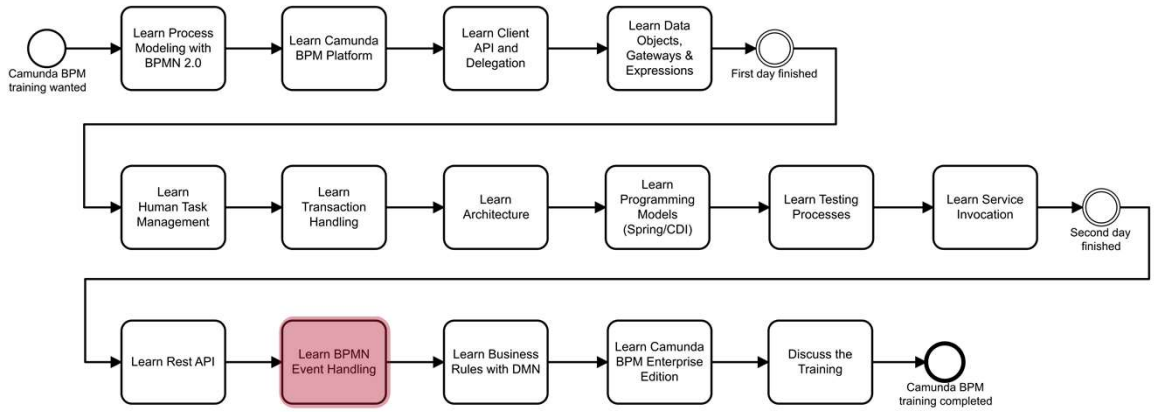
REST API for Process Visualization



Exercise 10

Optional for advanced rest users, a better exercise will follow tomorrow

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks

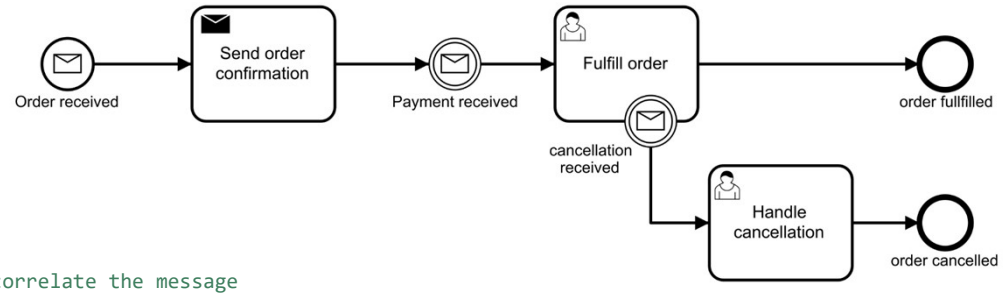


Event Coverage

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

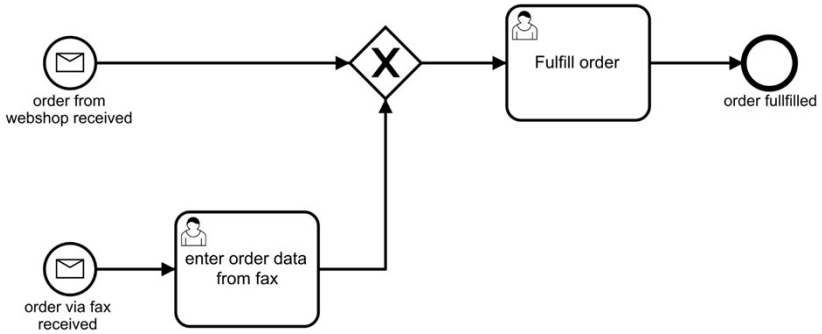
See: docs.camunda.org/manual/latest/reference/bpmn20/#events

Message Events



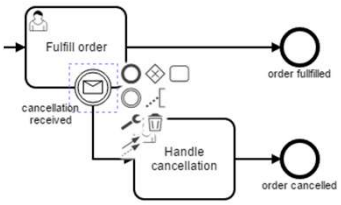

```
// correlate the message
runtimeService
  .createMessageCorrelation("Msg_PaymentReceived") // message name
  .processInstanceBusinessKey("AB-123")
  .setVariable("payment_type", "creditCard")
  .correlateWithResult();
```

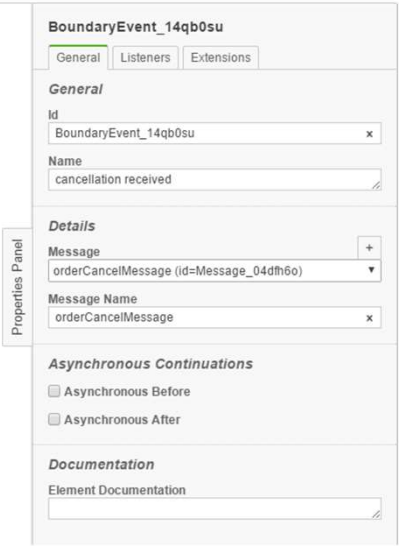

Multiple Message Start Events Possible




```
runtimeService
    .createMessageCorrelation("orderFromWebShop")
    .processInstanceBusinessKey("Order 123")
    .setVariable("myPayload", "myPayloadValue")
    .correlateWithResult();
```

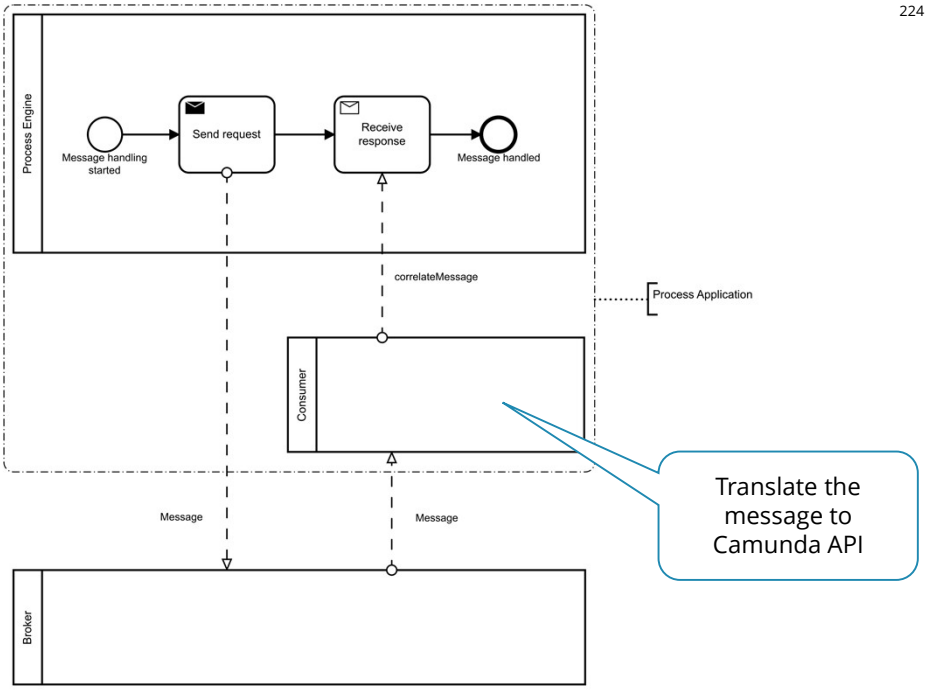
Configuring Message Names



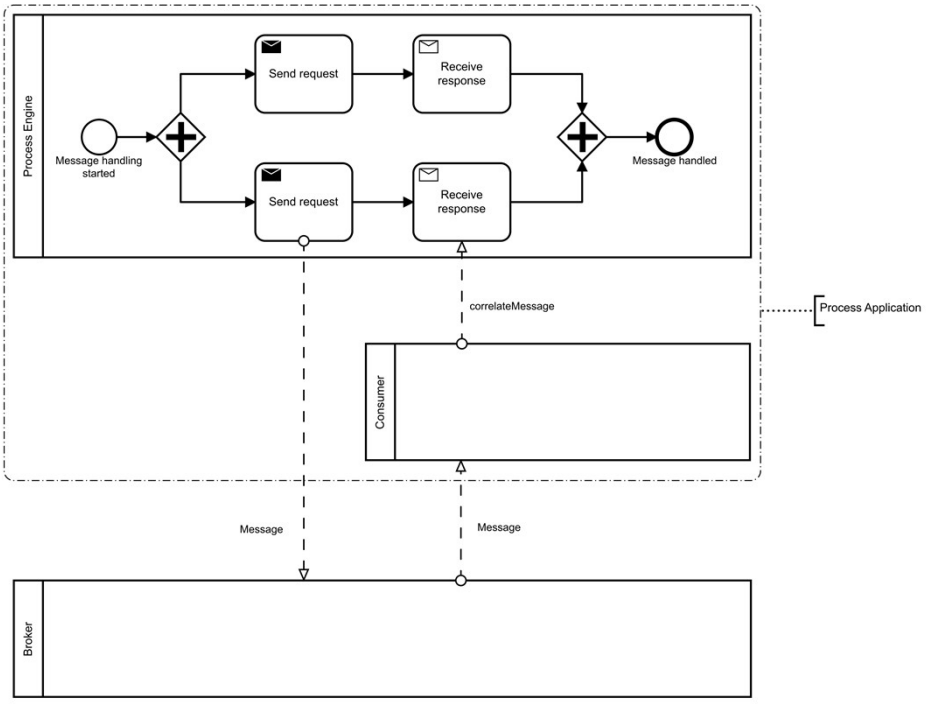



```
<bpmn:process id="OrderHandlingProcess" isExecutable="true">
    ...
    <bpmn:boundaryEvent id="BoundaryEvent_14qb0su"
name="cancellation received" attachedToRef="fulFillOrderUserTask">
      <bpmn:messageEventDefinition messageRef="Message_04dfh6o" />
    </bpmn:boundaryEvent>
    <bpmn:userTask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


Messaging

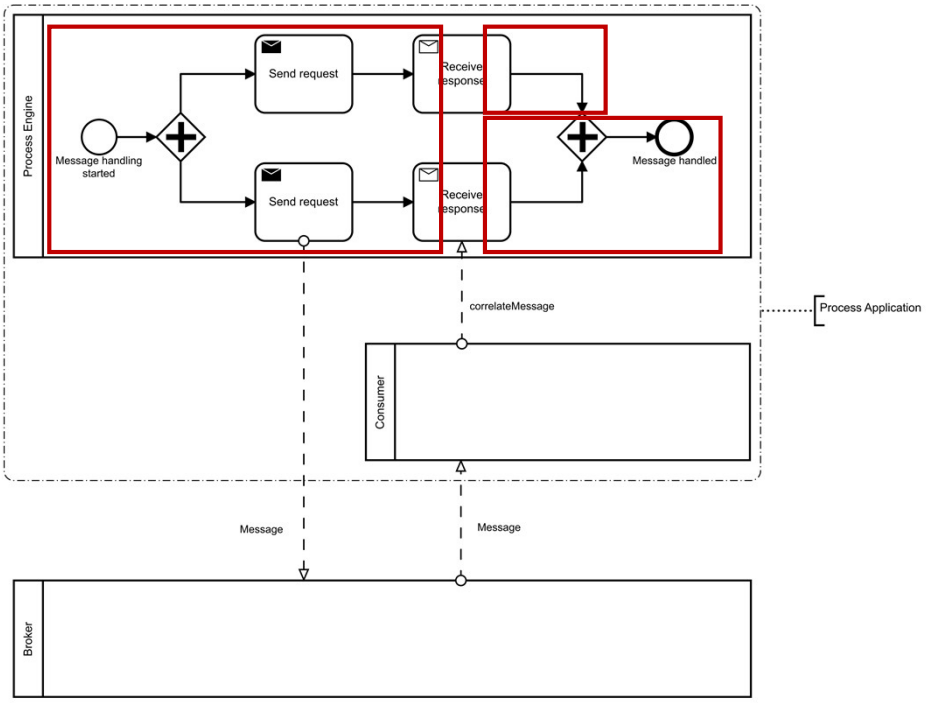


Messaging (#TX?)



225

Messaging

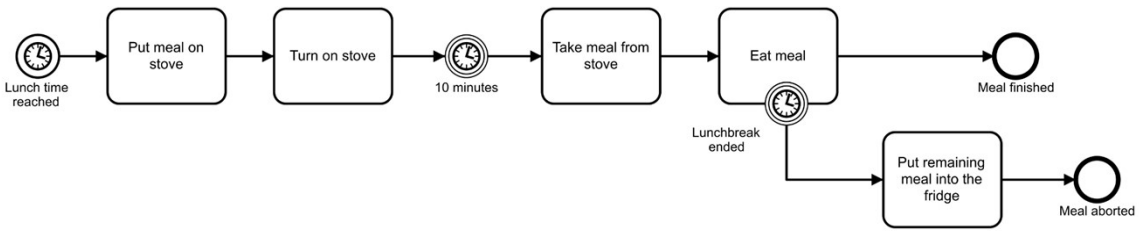


226

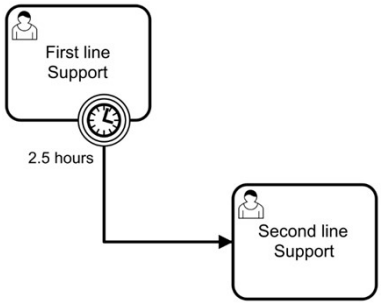
Exercise 11

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks

Timer Events



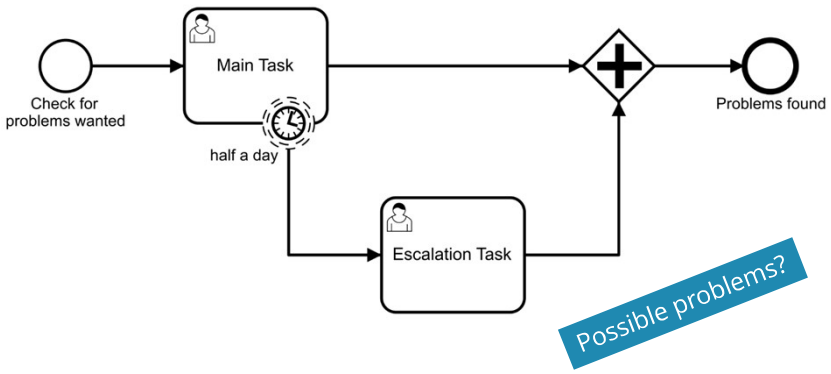
Timer Event Configuration



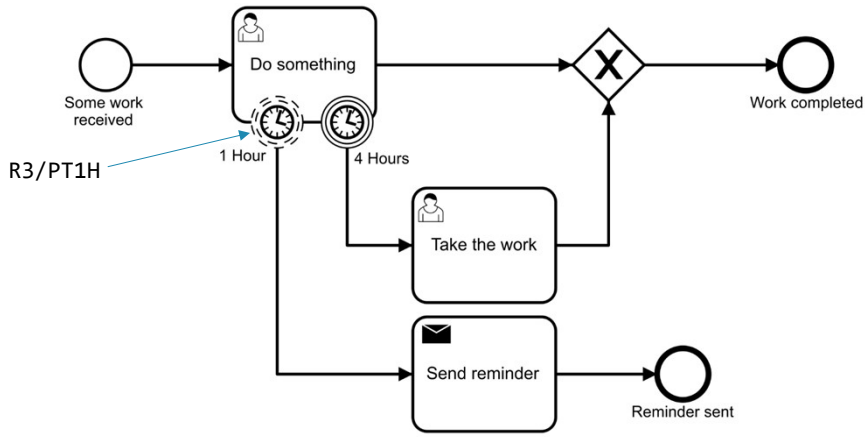

```
<bpmn:process id="Process_1" name="Timer Event Configuration Example" isExecutable="true">
  <bpmn:userTask id="firstLineSupportUserTask" name="First line Support" />
  <bpmn:boundaryEvent id="BoundaryEvent_1" name="2.5 hours" attachedToRef="firstLineSupportUserTask">
    <bpmn:timerEventDefinition>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

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

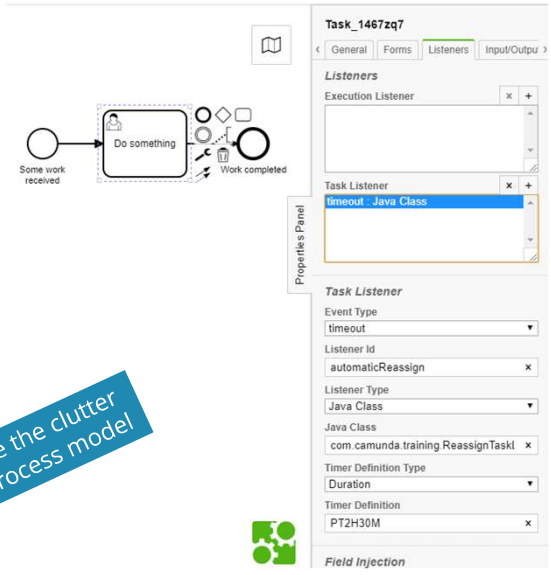
Non Interrupting Timer Event

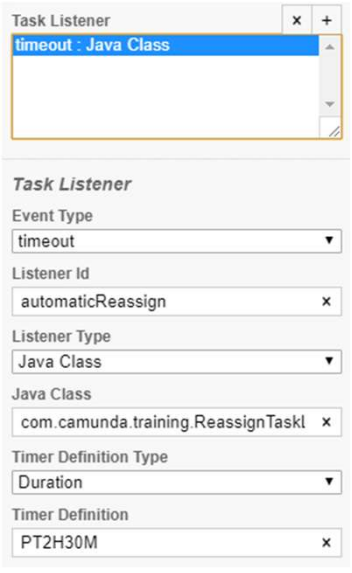


Pattern for Two Stage Escalation



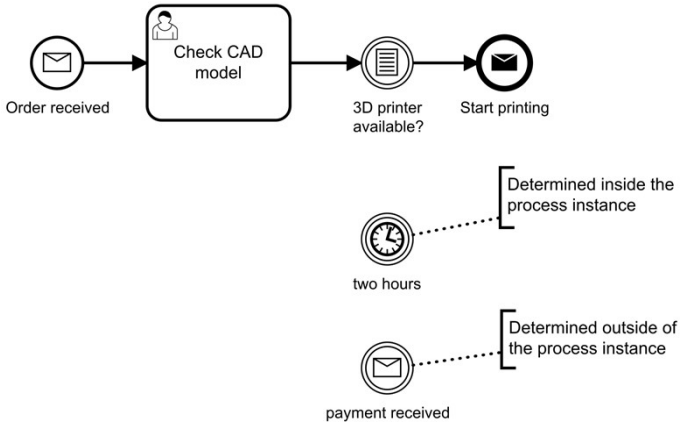
Time-triggered Events

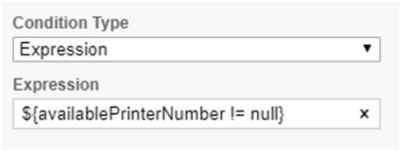




To reduce the clutter in the process model

Conditional Events

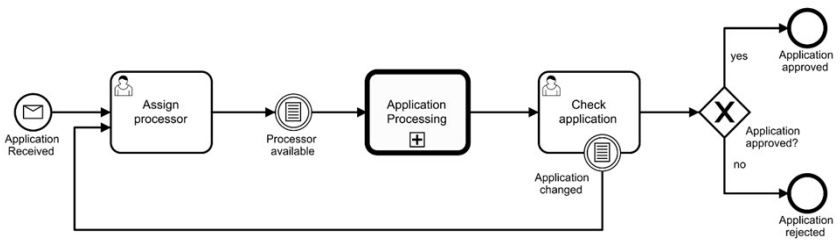




Trigger evaluation:

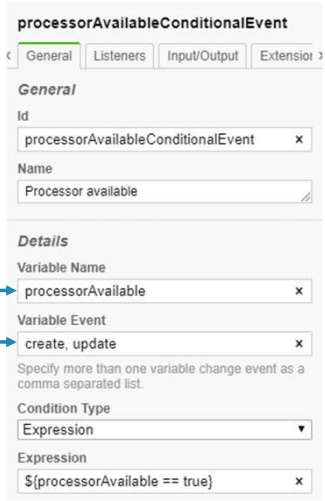
- via API
- in a Java Delegate
  (e.g. on a parallel branch)

Conditional Events



Limit evaluation to process variable

Limit evaluation to events



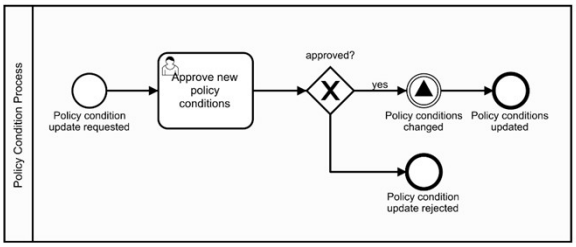
Transient Variables

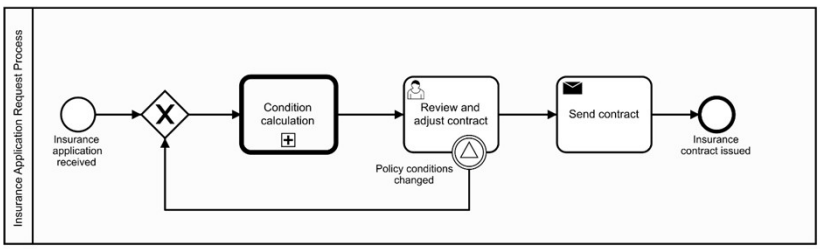
- Not stored in the database
- No update and delete
- Available with `TypedValue`

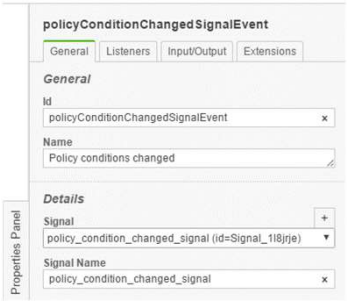
```
//primitive values
TypedValue typedTransientStringValue = Variables.stringValue("foobar", true);

//object value
com.camunda.consulting.Customer customer = new com.camunda.consulting.Customer();
TypedValue typedTransientObjectValue = Variables.objectValue(customer, true).create();

//file value
TypedValue typedTransientFileValue = Variables.fileValue("file.txt", true)
 .file(new File("path/to/the/file.txt"))
 .mimeType("text/plain")
 .encoding("UTF-8")
 .create();
```

`boolean isTransient`

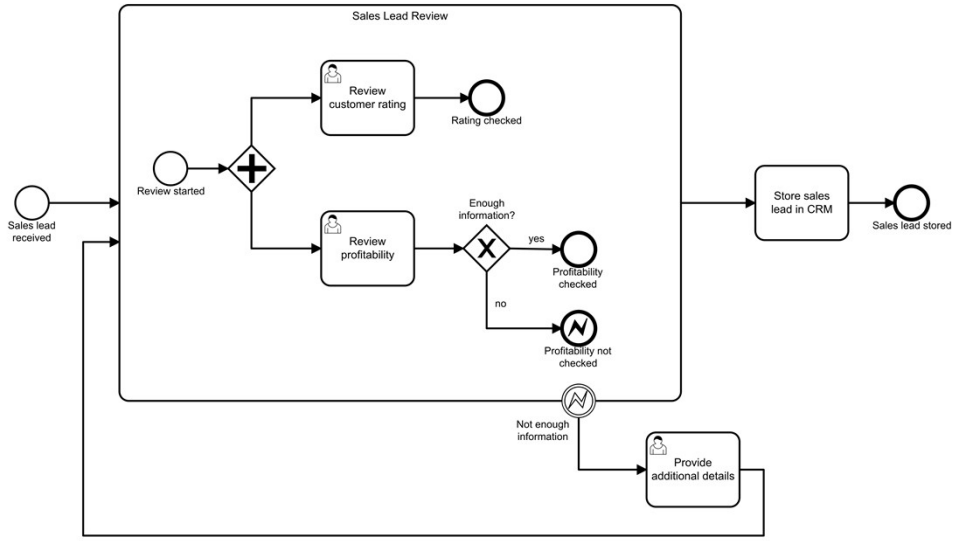
Signal Events – Broadcast to All Process Instances



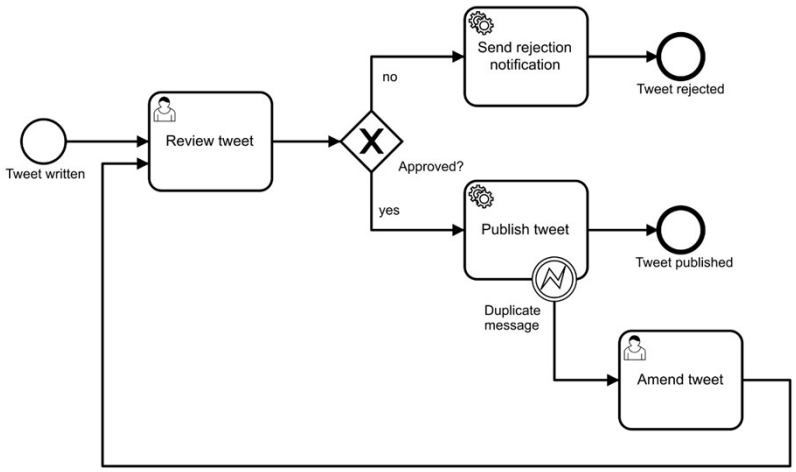




Error Event

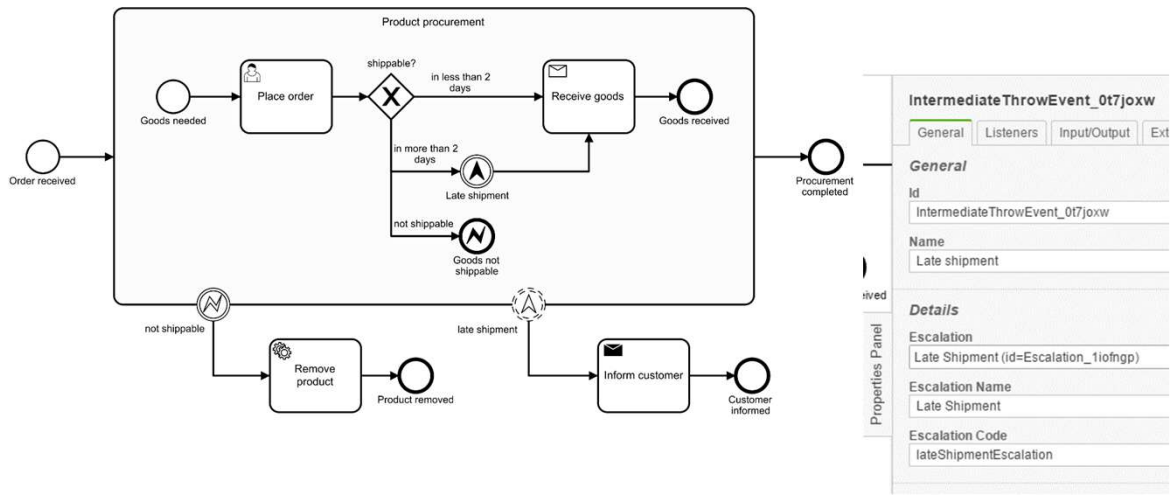


Error Handling

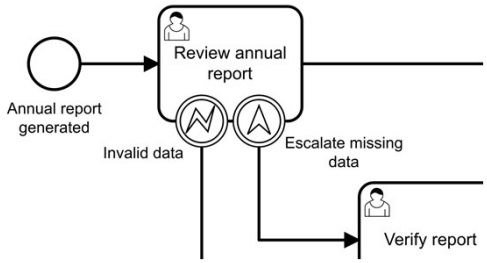


Demo

Escalation and Error Events



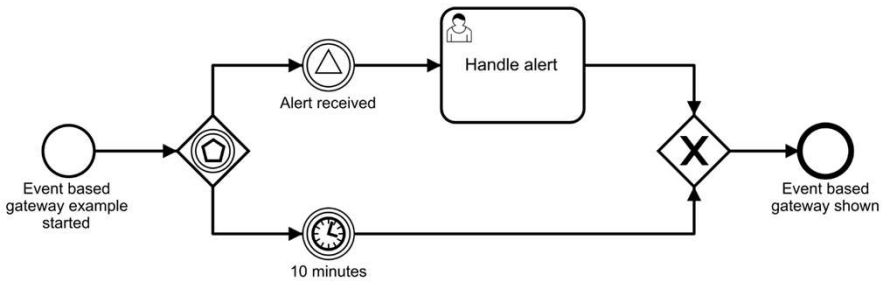
Error and Escalation Event on User Tasks




```
<button cam-error-code="bpmn-error-543"
    cam-error-message="anErrorMessage"
    class="form-control">Abort</button>

<button cam-escalation-code="escalation-123"
    class="form-control">Escalate</button>
```

```
taskService.handleBpmnError(task.getId(),
    "errorCode", ex.getMessage(), variables);

taskService.handleEscalation(task.getId(),
    "myEscalationcode", variables);
```


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.

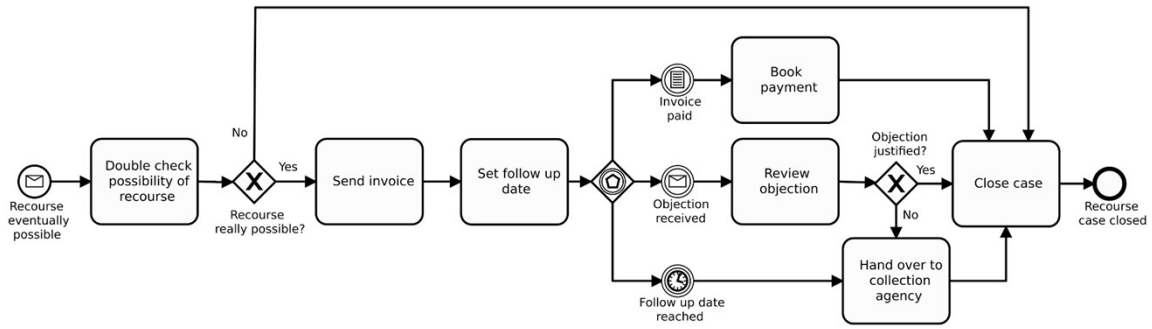
When we receive the money, I make a booking and close the case. If the insurant disagrees with the recourse, I'll have to check the reasoning of that. If he is right, I simply close the case. If he is wrong, I forward the case to a collection agency.

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

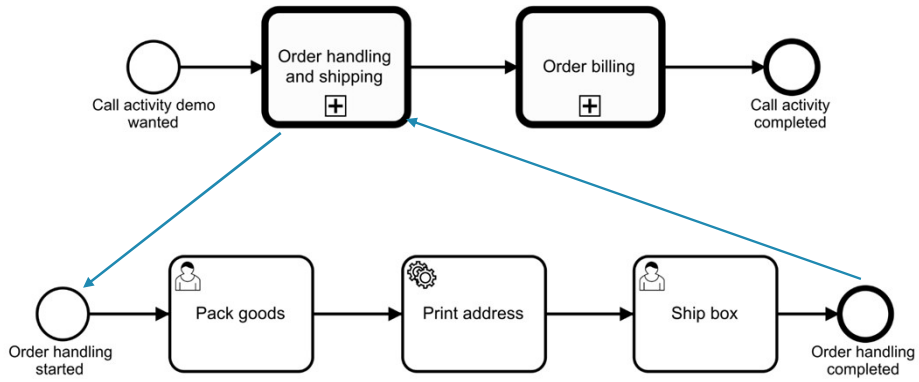
Background information:

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

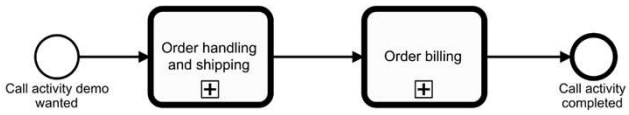
Event-Based Gateway – Real World Example

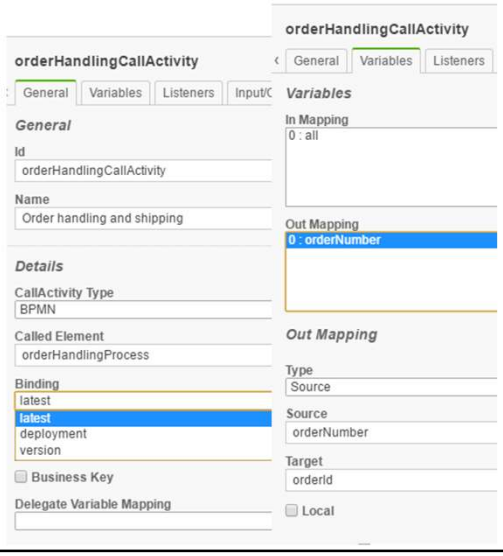


Reusable Sub-Processes: Call Activities



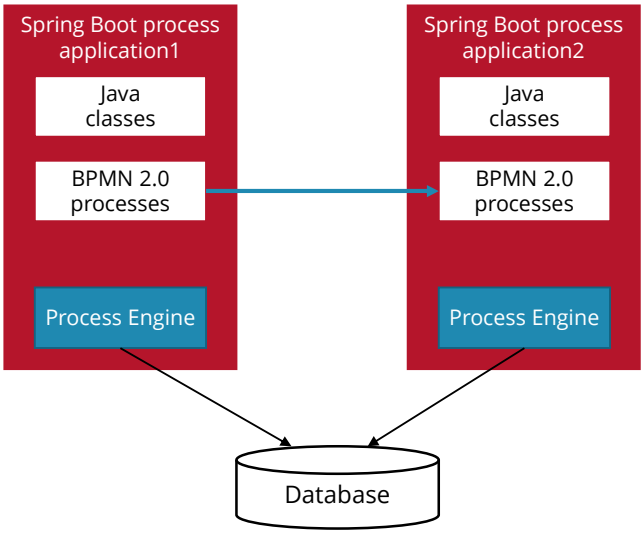
Reusable Sub-Processes: Call Activities



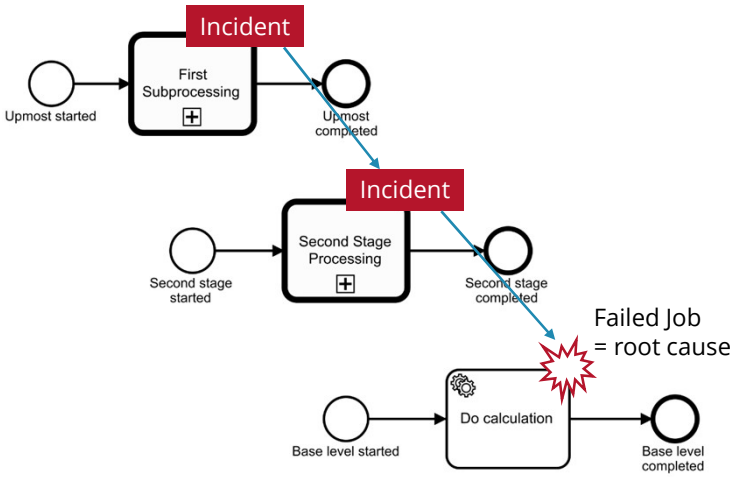


- Key is used to identify called process
- Process is resolved at runtime
- Variables can be mapped

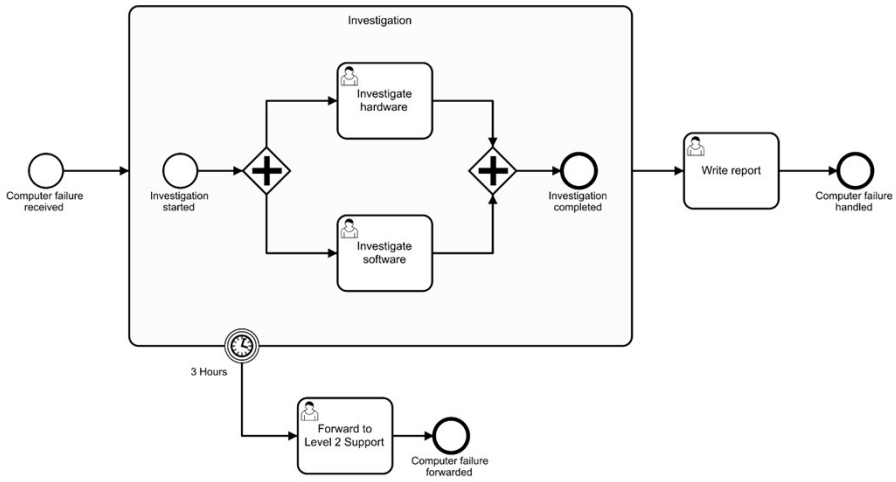
Calling Processes From Other Process Applications



Drill Down



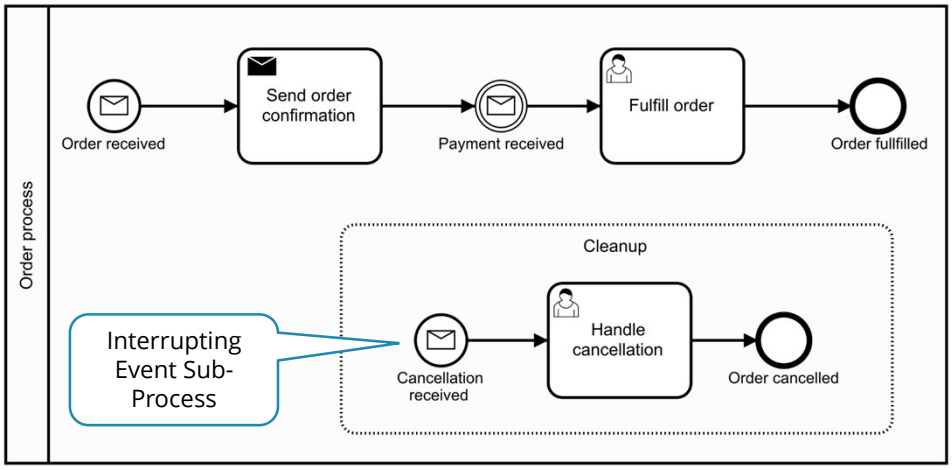
Embedded Sub-Processes



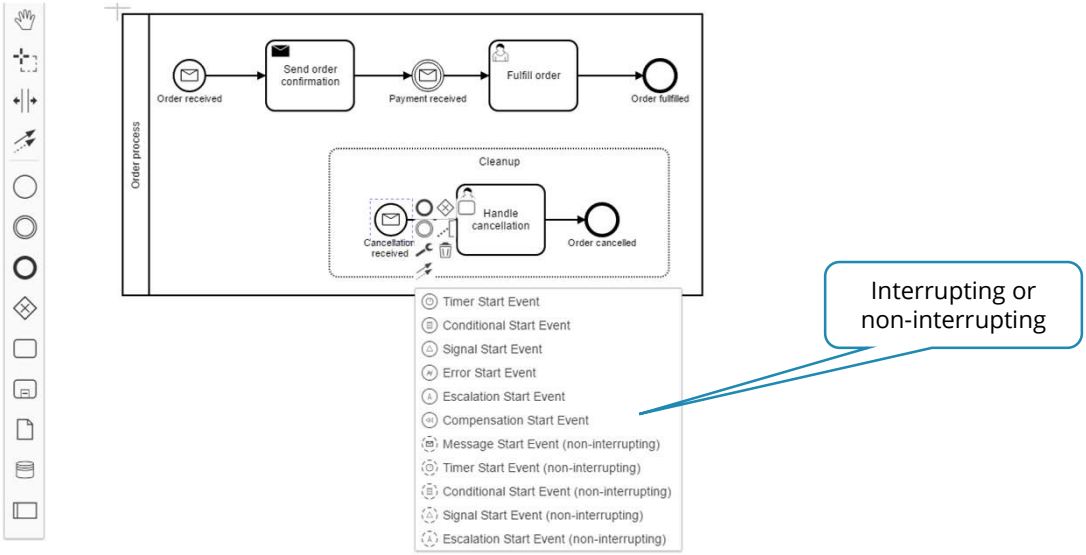
The Sub-Process Is Part of the Main Process

```
<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>
  ...
```

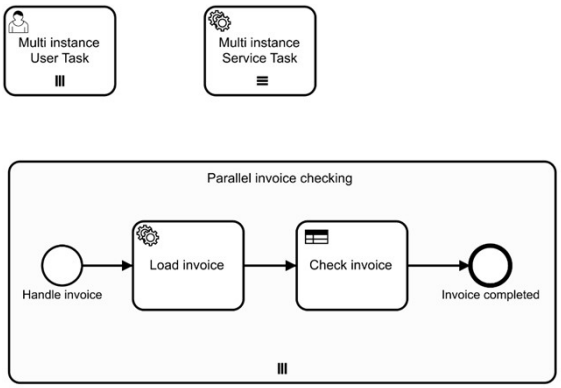
Event Sub-Process

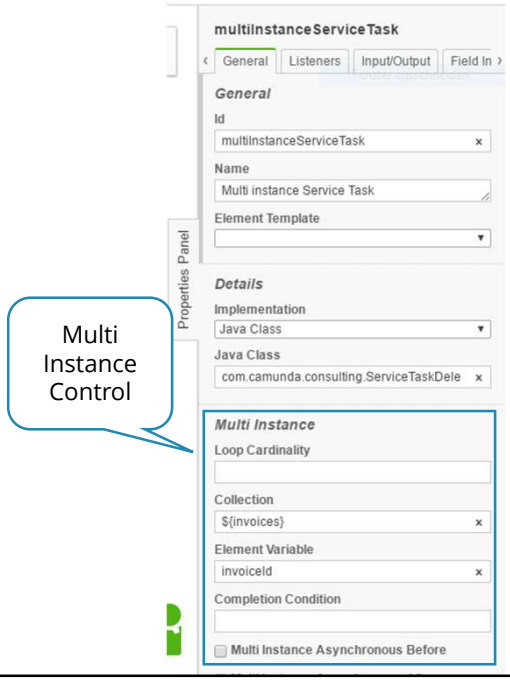


Possible Events to Start an Event Sub-Process



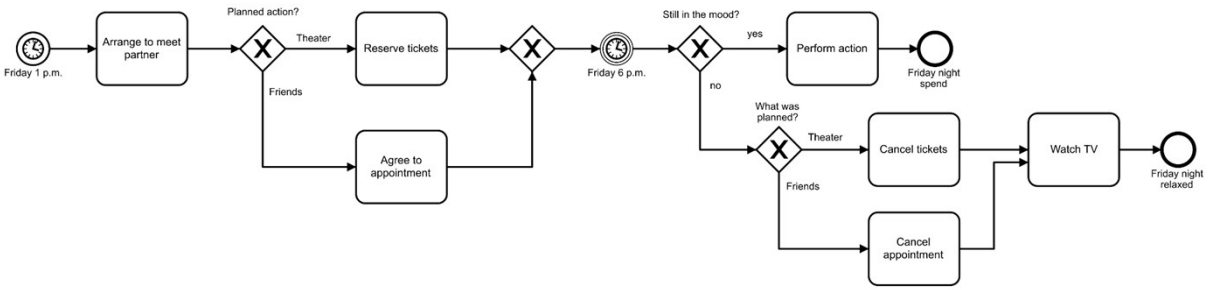
Multiple Instance Marker



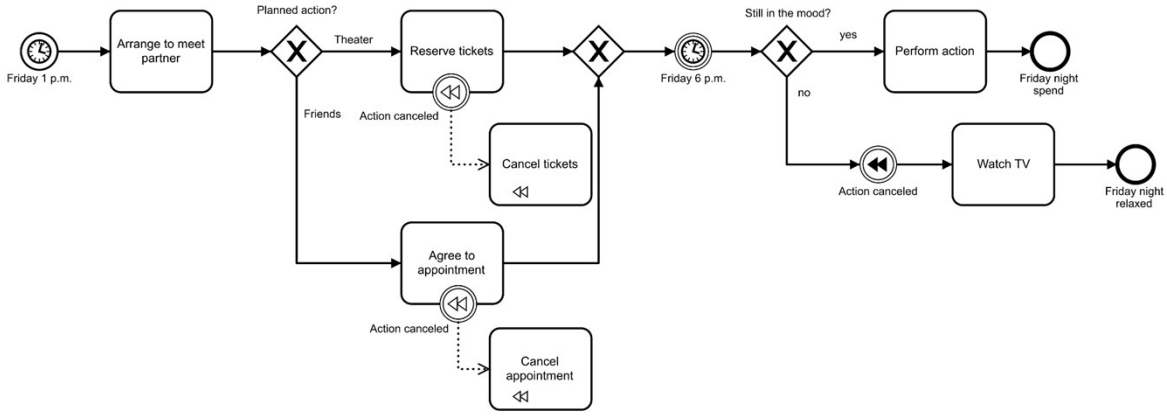


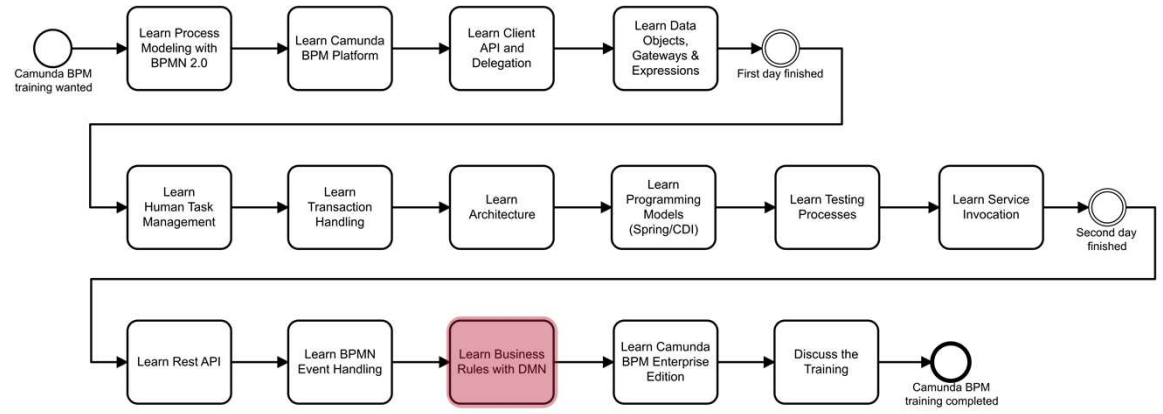
Multi Instance Control

Compensation

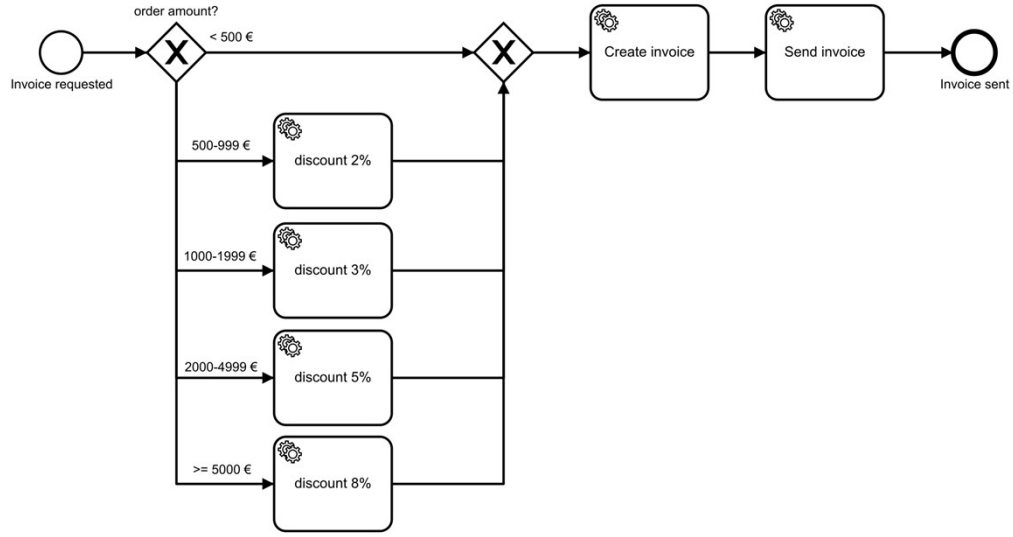


Compensation Events

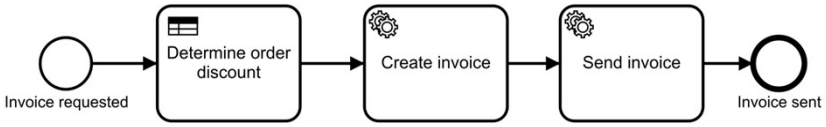




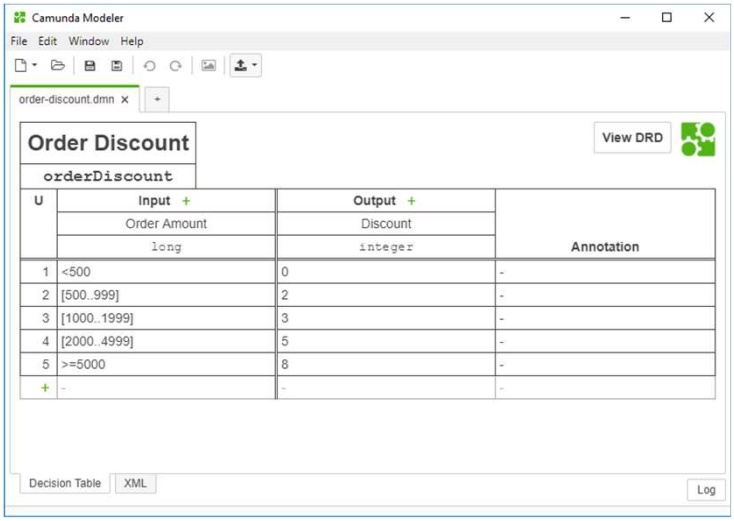
Is This a Good Process Model?



An Improved Version



Discount Rules as DMN Decision Table



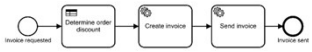
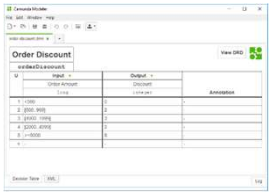
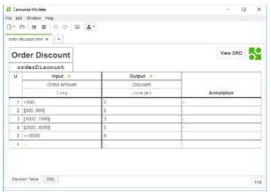
The Standard DMN

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

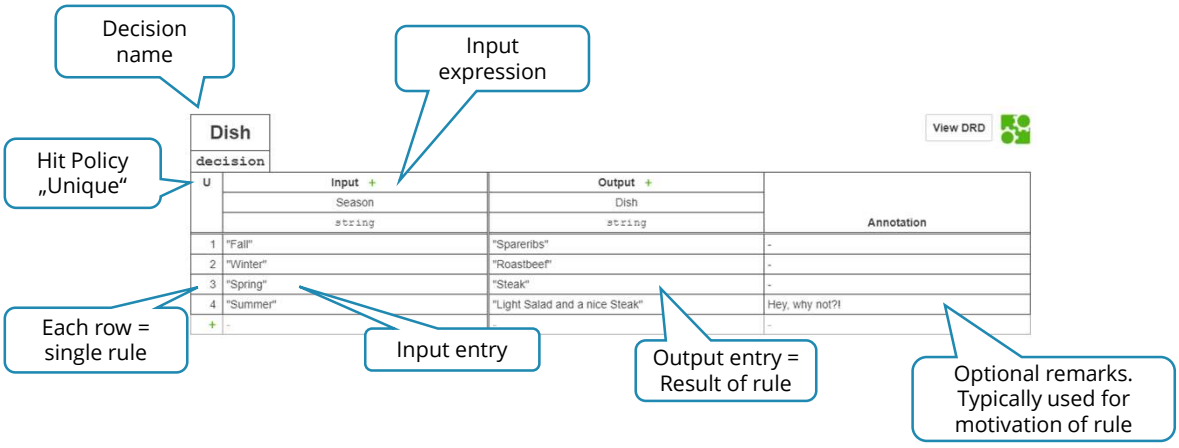
- Specification: „The primary goal of DMN is to provide a **common notation** that is readily understandable by all **business users**, from the business analysts needing to create initial decision requirements and then more detailed decision models, to the **technical developers** responsible for automating 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

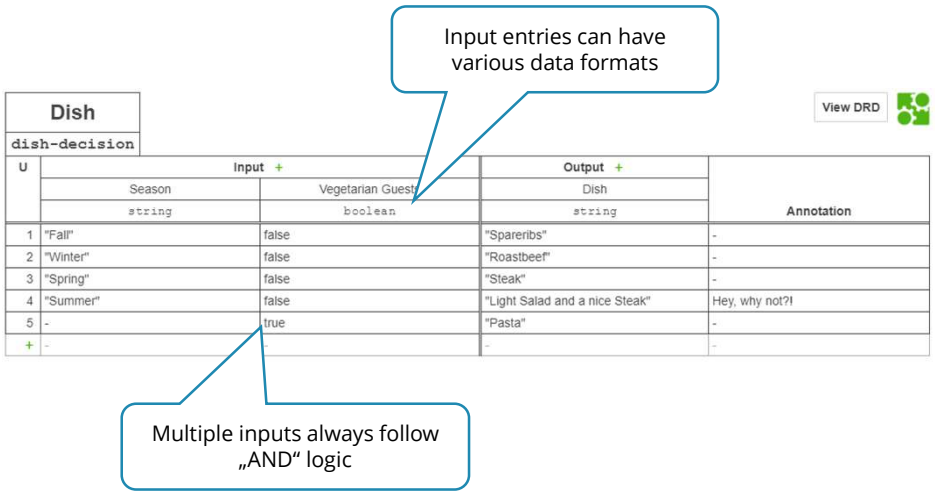
Typical Frequency of Changes

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)

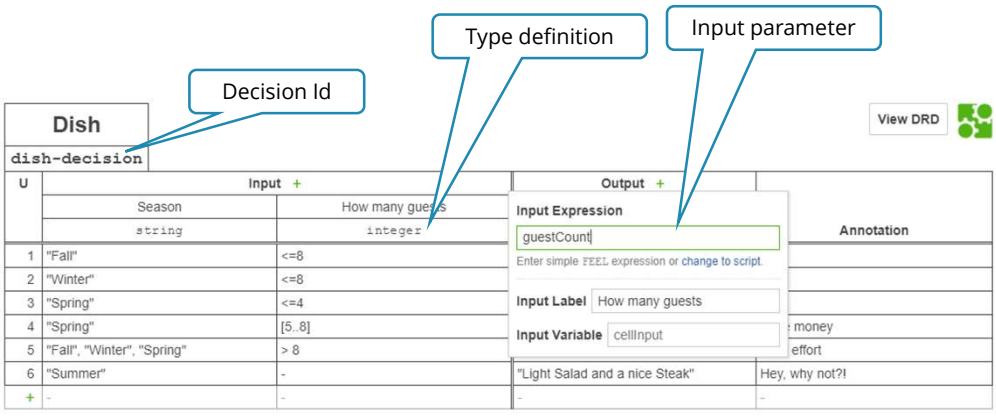
Rules Expressed as Decision Table



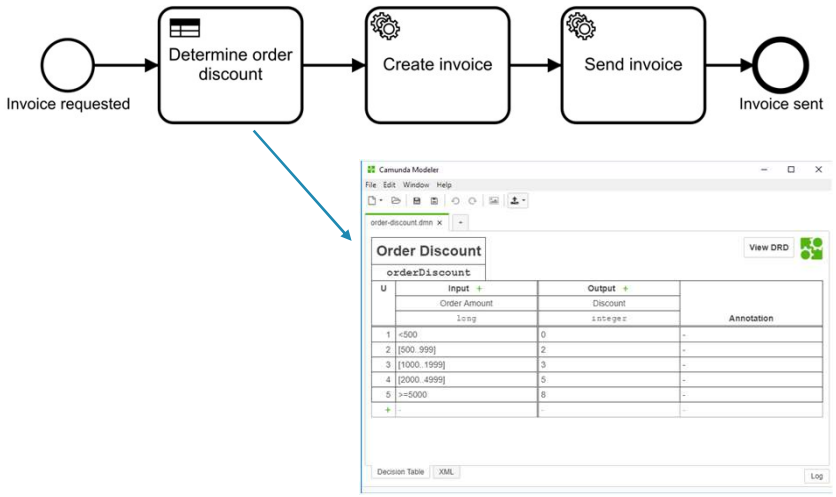
Multiple Inputs



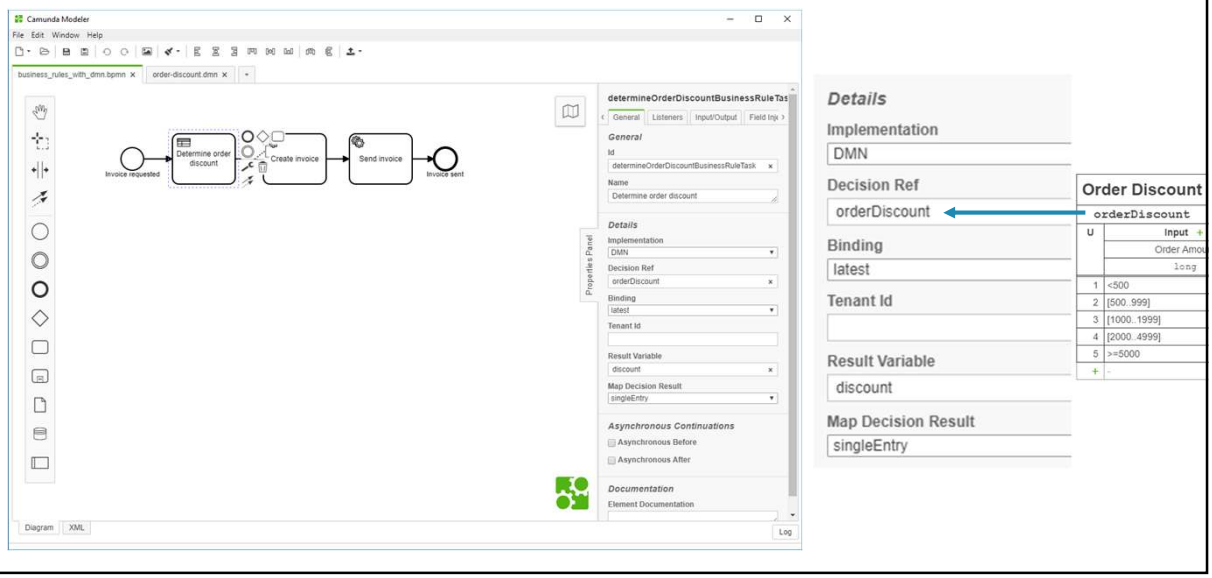
Technical Details



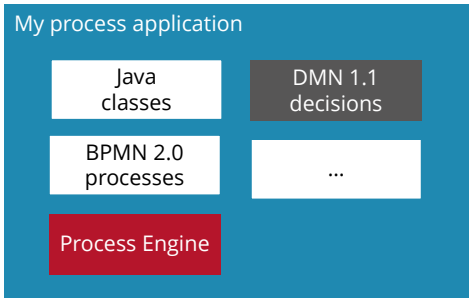
Business Rule Task Reference a DMN Decision Table



Reference in Camunda Modeler



DMN Is Part of the BPM Platform



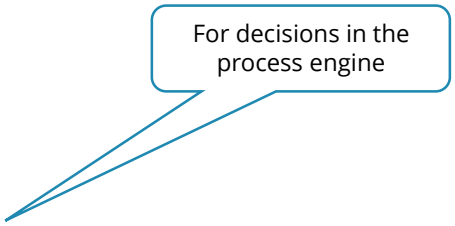
Decision Service in the Platform

```
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




```
@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

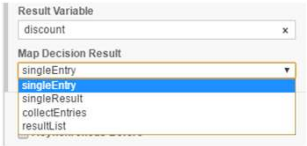
U	Applicant Age	Medical History	Applicant Risk Rating
1	> 60	*good*	*Medium*
2		*bad*	*High*
3	[25..60]	-	*Medium*
4	< 25	*good*	*Low*
5		*bad*	*Medium*

Student Financial Package Eligibility

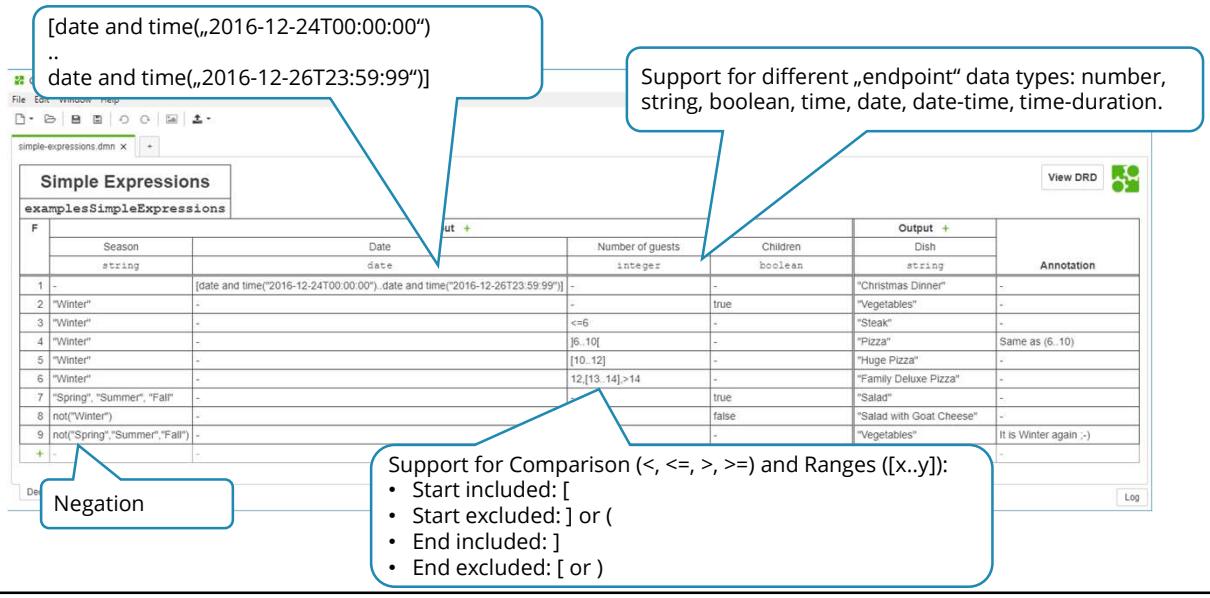
R	Student GPA	Student Extra-Curricular Activities Count	Student National Honor Society Membership	Student Financial Package Eligibility List
1	> 3.5	>= 4	*Yes*	20% Scholarship
2	> 3.0	-	*Yes*	30% Loan
3	> 3.0	>= 2	*No*	20% Work-On-Campus
4	<= 3.0	-	-	5% Work-On-Campus

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



Friendly Enough Expression Language (FEEL)



Community Extension FEEL-Scala

- Full support for FEEL
- Built-in functions (see https://github.com/camunda/feel-scala/wiki):
- Example expressions and functions:

```
applicant.monthly.income * 12

if applicant.maritalStatus in ("M","S") then "valid" else "not valid"

sum( [applicant.monthly.repayments, applicant.monthly.expenses] )

sum( credit_history[record_date > date("2011-01-01")].weight )

some ch in credit_history satisfies ch.event = "bankruptcy"

contains("foobar", "of")
```

https://github.com/camunda/feel-scala

FEEL-Scala Usage

Maven dependency:

```
<dependency>
  <groupId>org.camunda.bpm.extension.feel.scala</groupId>
  <artifactId>feel-engine-plugin</artifactId>
  <version>1.6.2</version>
</dependency>
```

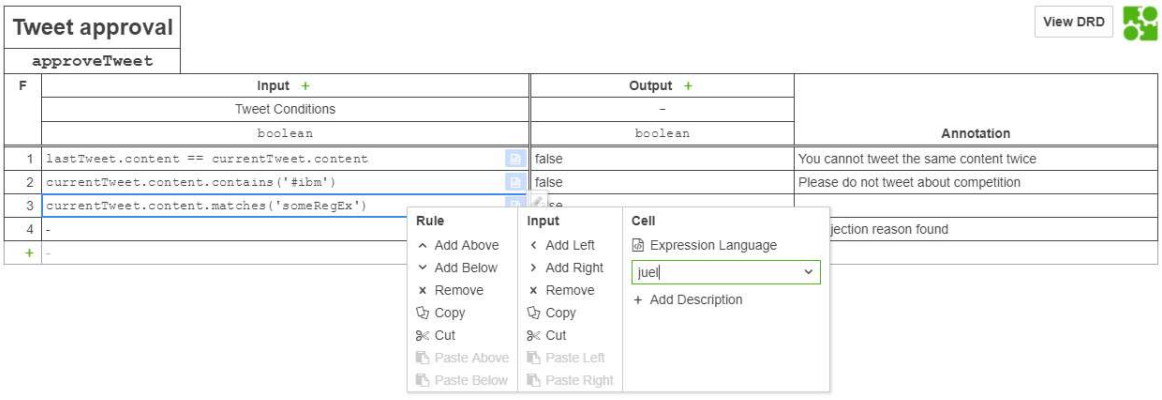
Spring Boot application:

```
@Configuration
public class BpmPlatformConfiguration {

  @Bean
  public static ProcessEnginePlugin feelScalaPlugin() {
    return new CamundaFeelEnginePlugin();
  }
}
```

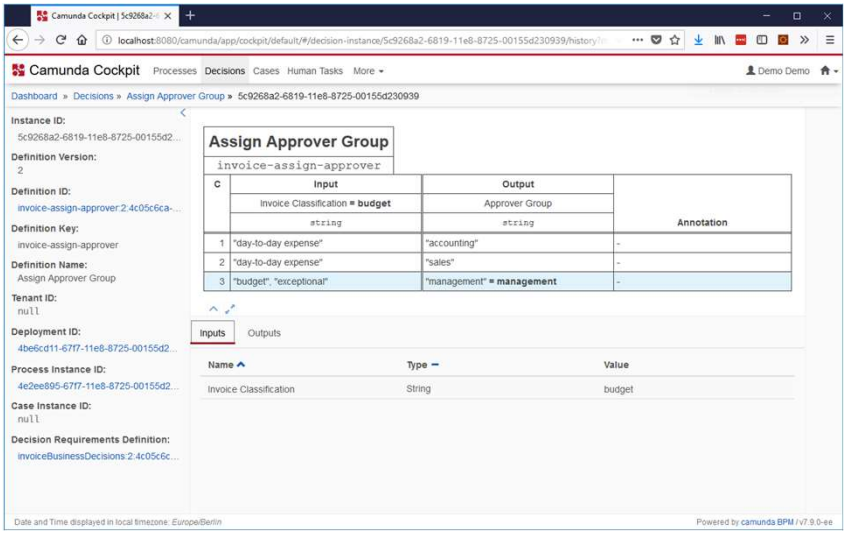
JUnit engine configuration:

```
<property name="processEnginePlugins">
  <list>
    <bean class="org.camunda.feel.CamundaFeelEnginePlugin"/>
  </list>
</property>
```


JUEL as Expression Language

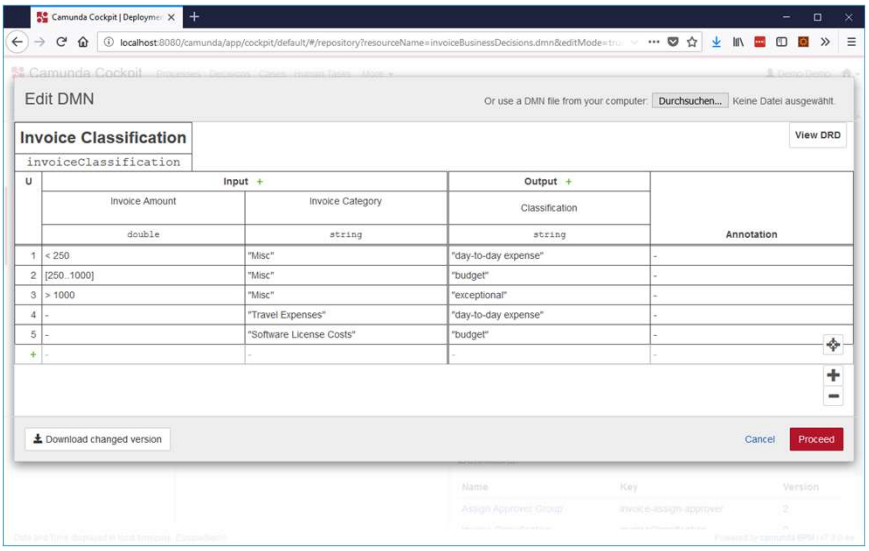


Cockpit for Decisions



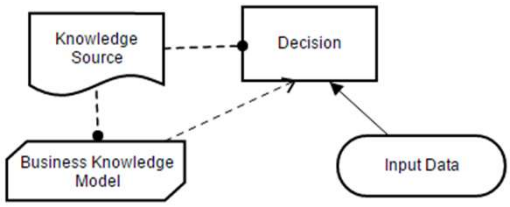
Enterprise Feature

Live Editing of Decision Tables

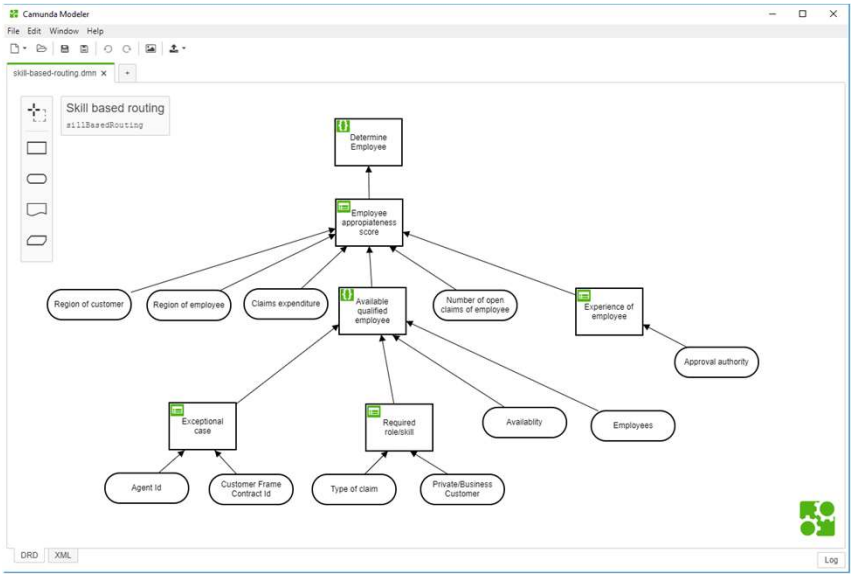


Beyond (Or Before) Decision Tables

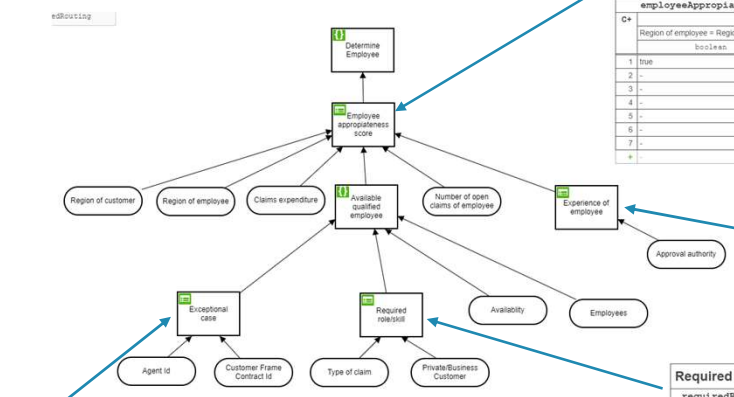
- DMN defines „Decision Requirements Diagram“
- Decision Requirements Diagrams will define the decisions [...], their interrelationships, and their requirements for decision logic
- Starting point for modeling complex decisions
- Elements of Decision Requirements Diagram



DRD Example: Skill Based Routing



Decisions Reference Decision Tables



Employee appropiateness score

`employeeAppropiatenessScore`

C+	Input +			Output +
	Region of employee = Region of customer	Experience of employee	Claims expenditure	Number of open claims of employee	-	
	boolean	string	integer	integer	integer	
1	true	-	-	-	100	-
2	-	"low"	[1000..10000]	-	-100	-
3	-	"low"	> 10000	-	-1000	-
4	-	"medium"	> 10000	-	-100	-
5	-	-	-	[10..19]	-100	-
6	-	-	-	[20..29]	-500	-
7	-	-	-	>= 30	-1000	-
+	-					

Experience of employee

`experienceOfEmployee`

U	Input +	Output +
	Approval Authority	Experience
	integer	string
1	< 1000	"low"	-
2	[1000..10000]	"medium"	-
3	> 10000	"high"	-
+	-		

Exceptional case

`exceptionalCase`

U	Input +		Output +	
	Agent Id	Customer Frame Contract Id	Required Role	Special Employee
	string	string	string	string
1	"4711"	-	"Special Task Force Berlin"	-	-
2	-	"0815"	"Special Task Force Berlin"	-	-
3	-	"camunda"	-	"Mr. important"	-
+					

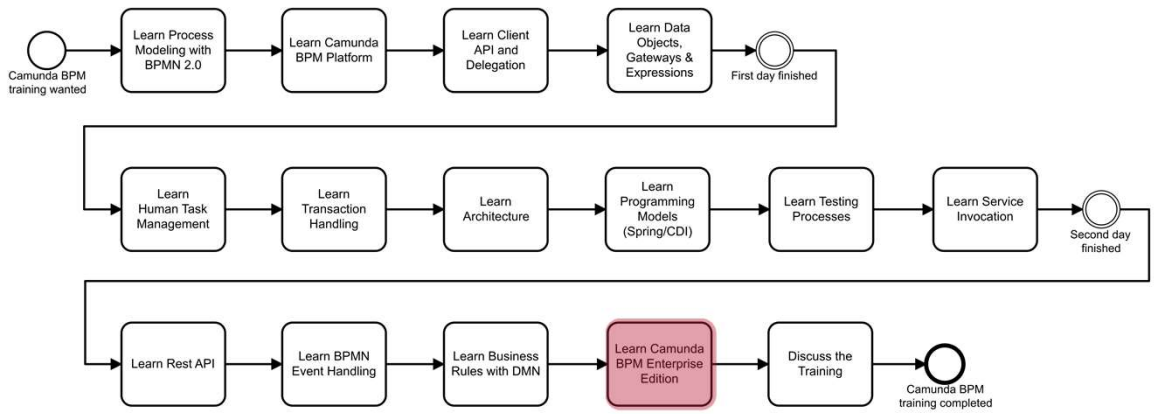
Required role/skill

`requiredRoleSkill`

U	Input +		Output +	
	Type of claim	Private/Business Customer	Required Role	Required Skill
	string	string	string	string
1	"Third Party Liability"	"Private"	"Service Center"	-	-
2	"Third Party Liability"	"Business"	"Service Center"	"Business Law Qualification"	-
3	"Accident"	"Private"	"Service Center"	-	-
4	"Accident"	"Business"	"Business Accident Team"	-	-
+					

Exercise 12

https://training.camunda.com/java-dev/
user: java-dev password: camundarocks



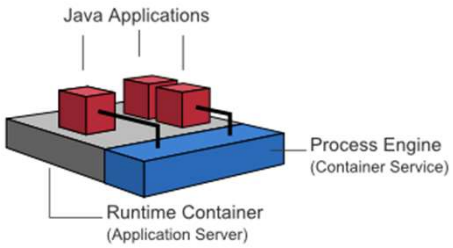
Enterprise Exclusive Cockpit Features

- 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

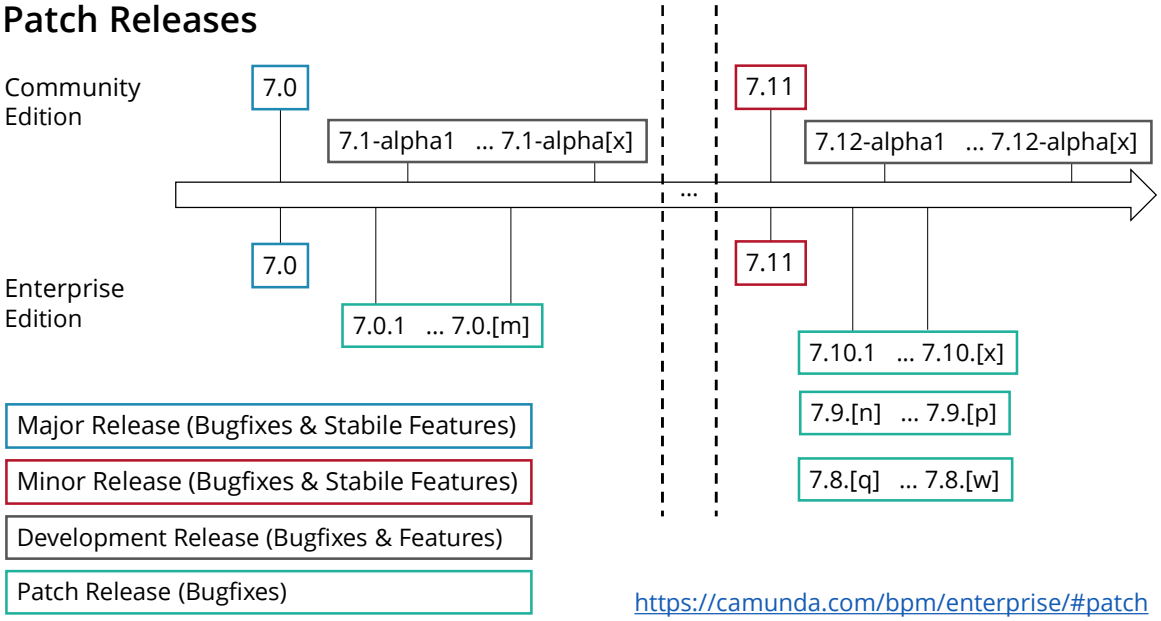
SLA Based Support

- 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

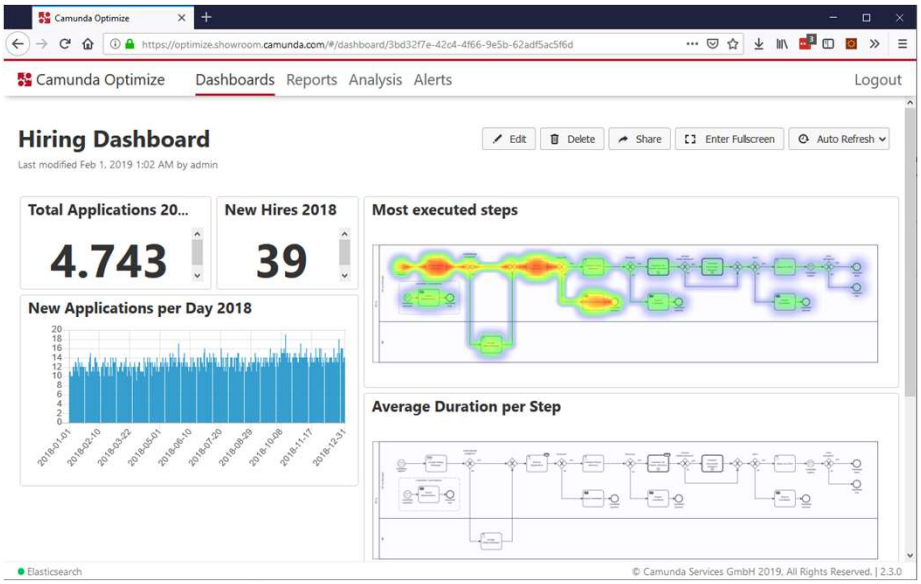


https://camunda.com/bpm/enterprise/#patch

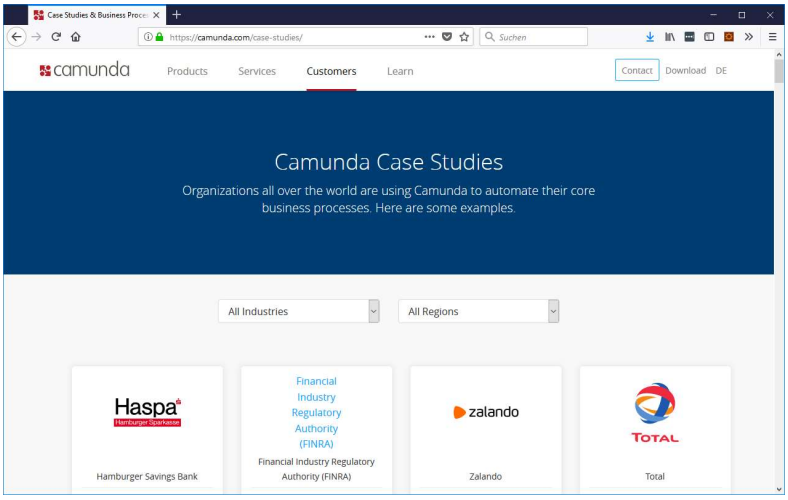
Please Plan your Updates

- Check patch releases: Bugs that bother you?
- Minor release:
  - 6 Months before End of Support you will receive a mail
  - Check if you run the version mentioned here
  - If yes, start planning your update

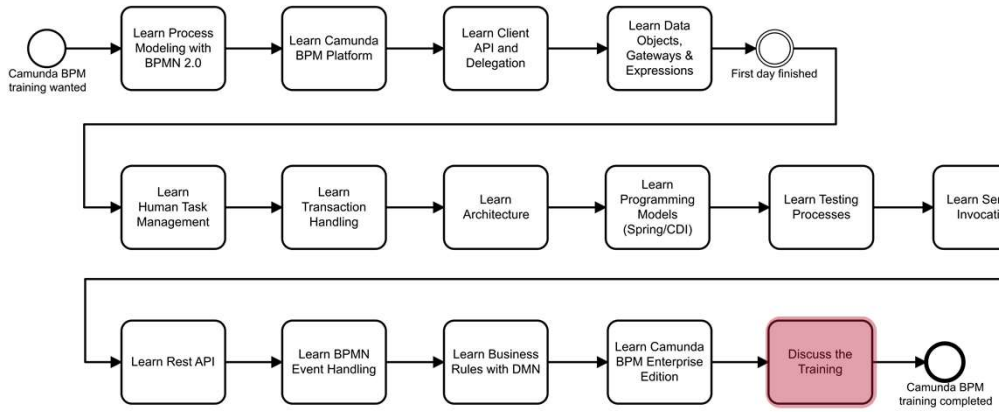
Optimize



References



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



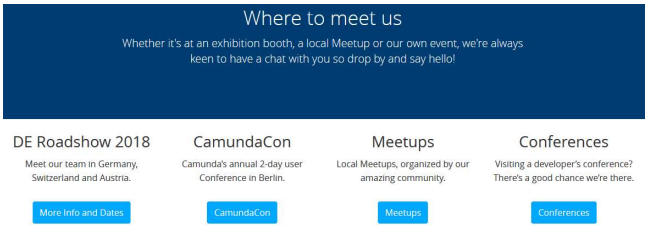
How to Get Updates From the Team?

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

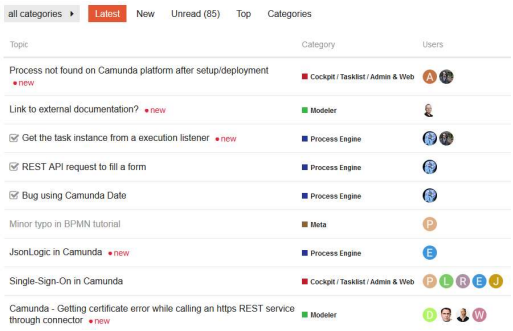
Camunda Team Blog

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

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



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



Follow us on Twitter: @camundabpm

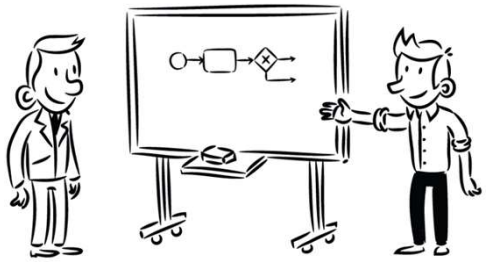
Camunda training offering

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

You'll find further information under
camunda.com/de/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/