CS 565
Business Process Management Systems
Camunda

Tutorial

Teaching: Chrysostomos Zeginis
Teaching Assistant: Nikolaos Fanourakis

2022
Agenda

- What is Camunda?
- BPMN 2.0 Symbols and Notations
- BPMN 2.0 Examples
- Camunda Installation Guide
- Detailed Process Modelling, Implementation and Deployment
- Order example for implementing Service and Send Tasks
What is Camunda?

- Workflow and Decision Automation Platform
- Lightweight
- Java-based framework
- It provides Business Process Model and Notation (BPMN)
BPMN 2.0 Symbols and Notations

- A Task is a unit of work, the job to be performed
- When marked with a symbol (activity markers) it indicates a Sub-Process, an activity that can be refined
- A subprocess is an activity that contains other activities, gateways, events, etc., which itself forms a process that is part of a bigger process. A subprocess is completely defined inside a parent process

Check cheatsheet BPMN 2.0 Cheatsheet

Tasks
BPMN 2.0 Symbols and Notations

Participants

**Pools (Participants) and Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.
A Data Object represents information flowing through the process, such as business documents, e-mails, or letters.
BPMN 2.0 Symbols and Notations

- In BPMN there are start events, intermediate events and end events

<table>
<thead>
<tr>
<th>Events</th>
<th>Start</th>
<th>Intermediate</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td><img src="image" alt="Circle" /></td>
<td><img src="image" alt="Circle" /></td>
<td><img src="image" alt="Circle" /></td>
</tr>
<tr>
<td>Message</td>
<td><img src="image" alt="Envelope" /></td>
<td><img src="image" alt="Envelope" /></td>
<td><img src="image" alt="Envelope" /></td>
</tr>
<tr>
<td>Timer</td>
<td><img src="image" alt="Clock" /></td>
<td><img src="image" alt="Clock" /></td>
<td><img src="image" alt="Clock" /></td>
</tr>
<tr>
<td>Escalation</td>
<td><img src="image" alt="Upward Arrow" /></td>
<td><img src="image" alt="Upward Arrow" /></td>
<td><img src="image" alt="Upward Arrow" /></td>
</tr>
<tr>
<td>Conditional</td>
<td><img src="image" alt="Folder" /></td>
<td><img src="image" alt="Folder" /></td>
<td><img src="image" alt="Folder" /></td>
</tr>
<tr>
<td>Link</td>
<td><img src="image" alt="Link" /></td>
<td><img src="image" alt="Link" /></td>
<td><img src="image" alt="Link" /></td>
</tr>
<tr>
<td>Error</td>
<td><img src="image" alt="Error" /></td>
<td><img src="image" alt="Error" /></td>
<td><img src="image" alt="Error" /></td>
</tr>
<tr>
<td>Cancel</td>
<td><img src="image" alt="Cross" /></td>
<td><img src="image" alt="Cross" /></td>
<td><img src="image" alt="Cross" /></td>
</tr>
<tr>
<td>Compensation</td>
<td><img src="image" alt="Left Arrow" /></td>
<td><img src="image" alt="Left Arrow" /></td>
<td><img src="image" alt="Left Arrow" /></td>
</tr>
<tr>
<td>Signal</td>
<td><img src="image" alt="Triangle" /></td>
<td><img src="image" alt="Triangle" /></td>
<td><img src="image" alt="Triangle" /></td>
</tr>
<tr>
<td>Multiple</td>
<td><img src="image" alt=" Pentagon" /></td>
<td><img src="image" alt=" Pentagon" /></td>
<td><img src="image" alt=" Pentagon" /></td>
</tr>
<tr>
<td>Parallel Multiple</td>
<td><img src="image" alt="Octagon" /></td>
<td><img src="image" alt="Octagon" /></td>
<td><img src="image" alt="Octagon" /></td>
</tr>
<tr>
<td>Terminate</td>
<td><img src="image" alt="Black Circle" /></td>
<td><img src="image" alt="Black Circle" /></td>
<td><img src="image" alt="Black Circle" /></td>
</tr>
</tbody>
</table>

In BPMN there are start events, intermediate events and end events. Each event type is represented by specific symbols and notations. The table above illustrates the different types of events with corresponding icons and descriptions.
BPMN 2.0 Examples

- This diagram shows a simple process triggered by someone being hungry. The result is that someone must shop for groceries and prepare a meal. After that, someone will eat the meal and have his or her hunger satisfied.
Camunda Installation Guide

● Prerequisites
  ○ Java JDK 1.8+
  ○ Eclipse IDE (recommended) or another

● Install Camunda Platform
  ○ Download a distribution (https://camunda.com/download)
  ○ Unpack it inside a directory of your choice
  ○ Execute the script named start.bat (for Windows users) or start.sh (for Unix users)
  ○ The application server is at http://localhost:8080/camunda/app/welcome/default/#!/login
  ○ Credentials: demo/demo
  ○ Do not stop it until the end of the tutorial
  ○ If port issue, then add the following to Camunda_Platform/configuration/default.yml

    server:
      port: 8081

● Install Camunda Modeler
  ○ Download from the download page
  ○ Unpack it inside a directory of your choice
  ○ Run Camunda Modeler.exe (Windows) or camunda-modeler (Linux)
Create your first BPMN 2.0 process with the Camunda Modeler

Start of a simple example

- After running **Camunda Platform** and **Camunda Modeler** ...
- Create a new BPMN 2.0 Diagram
Create your first BPMN 2.0 process with the Camunda Modeler

- Start by modelling a simple process
  - Double-click on the Start Event. A text box will open. Name the Start Event “Payment Retrieval Requested”
  - Click on the start event. From its context menu, select the activity shape (rounded rectangle)
Create your first BPMN 2.0 process with the Camunda Modeler

- The rectangle will be placed automatically on the canvas
- Name it *Charge Credit Card*
- Change the activity type to *Service Task* by clicking on the activity shape and using the wrench button
Create your first BPMN 2.0 process with the Camunda Modeler

- Add an End Event named *Payment Received*
Create your first BPMN 2.0 process with the Camunda Modeler

- Configure the service task
  - Click on the service task you just created
  - Change the implementation to *External* and use *charge-card* as the topic

---

If it’s not already visible, click on the **Properties Panel** tab.
Create your first BPMN 2.0 process with the Camunda Modeler

- Configure the process payment-retrival
  - Give it a **Process ID** (payment-retrival), a **Process Name** (Payment Retrival) and check **Executable**
  - Save the BPMN Diagram *File > Save File As...*
Implement an external task worker

- Since modelling has completed, we should implement the service task we created
  - Using the maven project we provide you based in Java and implement the `ChargeCardWorker`
  - Run the java application (Run as Java). The worker should remain running throughout the tutorial
  - Documentation: [https://docs.camunda.org/manual/7.16/user-guide/process-engine/external-tasks/](https://docs.camunda.org/manual/7.16/user-guide/process-engine/external-tasks/)

```java
public class ChargeCardWorker {
    private final static Logger LOGGER = Logger.getLogger(ChargeCardWorker.class.getName());

    public static void main(String[] args) {
        ExternalTaskClient client = ExternalTaskClient.create()
            .baseUrl("http://localhost:8080/engine-rest")
            .asyncResponseTimeout(10000) // long polling timeout
            .build();

        // subscribe to an external task topic as specified in the process
        client.subscribe("charge-card")
            .lockDuration(1000) // the default lock duration is 20 seconds, but you can override this
            .handler((externalTask, externalTaskService) -> {
                // Put your business logic here

                // Get a process variable
                String item = (String) externalTask.getVariable("item");
                Long amount = (Long) externalTask.getVariable("amount");

                LOGGER.info("Charging credit card with an amount of "+ amount + "( for the item "+ item + "... ");

                try {
                    Desktop.getDesktop().browse(new URI("https://docs.camunda.org/get-started/quick-start/complete");
                } catch (Exception e) {
                    e.printStackTrace();
                }

                // Complete the task
                externalTaskService.complete(externalTask);
            });

        .open();
    }
}
```
Deploy the process and Start a new instance

- **Deploy**
  - Make sure that the **Camunda Platform** and the **implemented service worker** are still running !!!
  - Click on the deploy button in the Camunda Modeler, then give it the Deployment Name “Payment Retrieval” and click the Deploy button
Deploy the process and Start a new instance

- You should see a success message in the Camunda Modeler
Deploy the process and Start a new instance

- Verify the Deployment with Cockpit (monitoring tool)
  - Visit http://localhost:8080/camunda/app/cockpit/
  - Login with the credentials demo / demo
  - Your process Payment Retrieval should be visible on the dashboard
Deploy the process and Start a new instance

- Start the first process instance
  - Send the following **POST** request to Camunda REST API using Postman
  - Request to URL: http://localhost:8080/engine-rest/process-definition/key/payment-retrieval/start
  - **JSON Body:**
    - `{"variables": {"amount": {"value":555,"type":"long"},"item": {"value": "item-xyz"}}}"
Deploy the process and Start a new instance

- As long as you **send** the request ...
  - The instance will be started and executed/completed immediately
    - `externalTaskService.complete(externalTask)` in our code
  - The following message will appear in worker's console

```
633 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01026 Discovered data for
635 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data for
635 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data for
636 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01026 Discovered data for
660 [main] INFO org.camunda.bpm.client - TASK/CLIENT-01025 Discovered data for
INFO: Charging credit card with an amount of '355' for the item 'item-xyz'...
```

End of the simple example
Add a user Task

- Extending the simple example by involving humans
- The objective is an human to approve the payment and not to charge immediately
  - Select the activity shape (rounded rectangle)
  - Drag it into position between the Start Event and the “Charge Credit Card” Service Task
  - Name it Approve Payment
Add a user Task

- Change the activity type to *User Task* by clicking on it and using the wrench button menu.
Configure a User Task

- Click on User Task
- In Property Panel complete Assignee to “demo”
- In Forms tab add “camunda-forms:deployment:payment.form” in the key
Configure a User Task

- Create an empty form `File > New File > Form`
- Add the specified input fields and the checkbox
Configure a User Task

- Select Camunda as execution platform
- Apply and save as **payment.form**
- Deploy the process and select the payment.form
Start a new instance

- Send again the same request with Postman, in order to start a new instance with amount = 355 and item = item-xyz
- In the Cockpit you will see the instance pending for approval
Approve Task

- In the **Tasklist** ([http://localhost:8080/camunda/app/tasklist/](http://localhost:8080/camunda/app/tasklist/)) you will see the form completed and the human needs to approve the request.
- As long as you click, **complete** you approve the request and the instance is completed.
Dynamic

- Making the process dynamic
  - Select the gateway shape (diamond) and drag it into position between the Start Event and the Service Task
  - Move the User Task down and add another Gateway after it
Open the properties panel and select the <1000 € Sequence Flow after the Gateway on the canvas.

Scroll to the property named Condition Type and change it to Expression.

Set input ${amount<1000}$ as the Expression.
Dynamic

- For the $\geq 1000$ € Sequence Flow, use the Expression ${\text{amount} \geq 1000}$
- For the Yes Sequence Flow, use the Expression ${\text{approved}}$
- For the No Sequence Flow, use the Expression ${!\text{approved}}$
Dynamic

- Deploy again the process
- Send again the request with Postman
  - In case that amount < 1000, we do not need to approve
  - In case that amount >= 1000, we need to approve
Order Example

we must implement as external tasks, otherwise we cannot deploy
Order Example - Implementation using Java API

see the "order" project for the whole example

```java
package org.camunda.bpm.getstarted.order;

import java.util.ArrayList;

public class Order {
    public String id;
    public String food;

    public Order(String id, String food) {
        this.id = id;
        this.food = food;
    }

    public Order() {
    }

    @Override
    public String toString() {
        return "Order [id = " + id + " food = " + food + "]";
    }
}
```

```java
public class App {
    public static void main(String... args) {
        // bootstrap the client
        ExternalTaskClient client = ExternalTaskClient.create() .baseUrl("http://localhost:8080/engine-rest") .asyncResponseTimeout(18000) .build();

        // subscribe to the topic
        client.subscribe("sendOrder") .handler((externalTask, externalTaskService) -> {
            // get the id and the food variables that we sent using Postmen
            String id = externalTask.getVariable("id");
            String food = externalTask.getVariable("food");

            // instantiate an order object
            Order order = new Order(id, food);

            // create an object typed variable with the serialization format JSON
            ObjectValue orderValue = ClientValues .objectValue(order) .serializationDataFormat("application/json") .create();

            // add the order object and its id to a map
            Map<String, Object> variables = new HashMap<>();
            variables.put("orderId", order.id);
            variables.put("order", orderValue);

            // complete the task and move to next one
            externalTaskService.complete(externalTask, variables);

            System.out.println("The External Task sendOrder" + externalTask.getId() + " has been completed!");
        }).open();

        client.subscribe("prepareOrder") .handler((externalTask, externalTaskService) -> {
            TypedValue typedOrder = externalTask.getVariableTyped("order");

            if(typedOrder != null) {
                Order order = (Order) typedOrder.getValue();
                System.out.println("Order + order + " + order + " prepared");
                externalTaskService.complete(externalTask);
            }
        }).open();
    }
```
Order Example - Implementation using Java API

- Run the project as Java application and the tasks are waiting for POST request

**Postman**

**POST**

http://localhost:8080/engine-rest/process-definition/key/order/start

### Body

```json
{"variables": {
  "id": {
    "value": "order1",
    "type": "String"
  },
  "food": {
    "value": "pizza",
    "type": "String"
  }
}}
```

**Console Java Application**

```
Order Order [id= order1 food= pizza] prepared
```
Useful Links

Getting Started Guide
- https://docs.camunda.org/get-started/quick-start/
- import maven project to eclipse: link

Documentation and Examples of BPMN 2.0 Symbols and Notations
- https://camunda.com/bpmn/examples/
- https://camunda.com/bpmn/
- https://docs.camunda.io/docs/components/best-practices/development/routing-events-to-processes/

Implementation
- https://docs.camunda.org/manual/7.16/reference/bpmn20/
- https://docs.camunda.org/manual/7.16/user-guide/process-engine/external-tasks/
Questions ?