

Documentation zur JEngine

Jaspar, Jan, Sven, Stephan, Juliane, Jannik
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany

{Jaspar,Jan,Sven,Stephan,Juliane,Jannik}@student.hpi.de

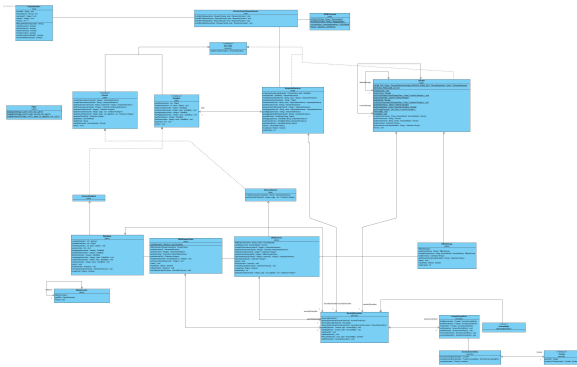


Figure 1: Meta Modell von PCM.

ABSTRACT

Diese Dokumentation ist entstanden im Rahmen des Bachelorprojekts BP2014W1 am Lehrstuhl für "Business Process Technology" betreut durch Prof. Dr. Mathias Weske. Es dient zur Dokumentierung der konzipierten und implementierten JEngine um ein Proof-of-Concept zu ermöglichen und gleichzeitig als Prototype für Anwendungsfälle von Bosch Software Innovations zu fungieren.

1. INTRODUCTION

Productive Case Management (PCM) beschreibt eine... [1] (siehe Abbildung 1).

2. METAMODELL

3. JENGINE

Overall JEngine

3.1 JCore

Der JCore umfasst mehrere Hauptkomponenten unserer Engine. Dazu zählt die Auswertung der Datenbank und das Entscheiden von enableden Aktivitäten etc.

Dazu zählt zum Beispiel auch die REST-API sowie der ExecutionService.

In der JCore liegen die grundsätzlichsten Funktionalitäten, die unsere JEngine anbietet. Zu diesen gehören zum einen die Möglichkeit, dass man Aktivitäten sequenziell ausführen kann. Bei den Aktivitäten, die man ausführen kann, handelt es sich bisher nur um User-Tasks, die später noch mit E-Mail-Tasks erweitert werden sollen.

Bei PCM können Aktivitäten aus Datenfluss- und Datenobjektflusssicht enabled werden. Dies ist auch der Grund, weshalb wir Datenobjekte unterstützen, sowie deren Zustandsübergänge. Außerdem können in PCM Aktivitäten referenziert werden, das bei der Ausführung die Folge hat, dass wenn referenzierte Aktivitäten enabled sind und dieselben Vor- und Nachbedingungen haben, auch beide in den Zustand running wechseln, sobald eine von ihnen gestartet wird.

Darüberhinaus bestehen PCM-Fragmente aus einem Subset von BPMN. Davon unterstützen wir zur Zeit Start-/Endevents, Aktivitäten und AND-Gateways.

3.1.1 REST-API

Um eine Kommunikation zwischen unseren verschiedenen Elementen der Engine und des Front-Ends zu ermöglichen, haben wir uns für ein REST-Interface entschieden. Dabei unterstützen wir bisher 2 Methoden: GET und POST.

1. @GET:

Die GET-Requests existieren, um den Nutzern der Engine über ein User Interface eine Möglichkeit zu bieten, einzusehen welche Aktivitäten noch zu bearbeiten sind (also offen sind) bzw. welche Aktivitäten zu welcher Szenarioinstanz gehören und ähnliches.

1.1.

Um offene bzw. geschlossene Aktivitäten einer Szenarioinstanz ausgegeben zu bekommen, muss ein GET mit folgender URL ausgeführt werden:

`http://172.16.64.113:8080/JEngine/Scenario/{ScenarioID}/{ScenarioInstanceID}/{Status}`

Variablen die benutzt werden:

ScenarioID: hier wird die ID des Szenarios erwartet. Dabei handelt es sich um einen Integer-Wert.

ScenarioInstanceID: hier wird die ID der Szenarioinstanz erwartet. Dabei handelt es sich um einen Integer-Wert.

Status: Der Status ist vom Typ String und muss ein

Element der Menge `{terminated, enabled}` sein.

Dabei können folgende Fehler geworfen werden:

Error: not a correct scenario instance

Dies bedeutet das eine ScenarioInstanzID angegeben worden ist, die nicht existiert.

Error: status not clear

Dieser Fehler besagt, dass ein Status angegeben wurde, der nicht der Menge `{terminated, enabled}` entspricht.

1.2.

Wenn man wissen möchte, welche Szenarien man in der JEngine starten kann, erhält man diese über folgende GET URL:

http://172.16.64.113:8080/JEngine/Scenario/Show

1.3.

Um alle ScenarioInstanzen eines Szenarios zu bekommen, muss ein GET-Request mit folgender URL ausgeführt werden:

http://172.16.64.113:8080/JEngine/Scenario/Instances/{ScenarioID}

Variable die benutzt wird:

ScenarioID: Dabei handelt es sich um einen Integer, der die ID des zu betrachtenden Szenarios angibt.

Dabei kann folgender Fehler geworfen werden:

Error: not a correct scenario

Dieser tritt auf, wenn eine ID übergeben wurde, die nicht existiert.

1.4.

Wenn man alle Datenobjekte in ihren entsprechenden Zuständen anzeigen möchte, bezogen auf eine ScenarioInstanz, muss ein GET-Request mit folgender URL ausgeführt werden:

http://172.16.64.113:8080/JEngine/Scenario/DataObjects/{ScenarioID}/{ScenarioInstanceID}

Variablen die benutzt werden:

ScenarioID: hier wird die ID des Szenarios erwartet. Dabei handelt es sich um einen Integer-Wert.

ScenarioInstanceID: hier wird die ID der ScenarioInstanz erwartet. Dabei handelt es sich um einen Integer-Wert.

Dabei kann folgender Fehler geworfen werden:

Error: not a correct scenario instance

Wenn eine falsche ScenarioInstanzID angegeben wird, produziert es diesen Fehler.

1.5.

Um von einer SzenarioInstanzID die dazugehörige ScenarioID zu erhalten, kann ein GET-Request mit folgender URL ausgeführt werden:

http://172.16.64.113:8080/JEngine/Scenario/

Get/ScenarioID/{ScenarioInstanceID}

Variable die benutzt wird:

ScenarioInstanceID: hier wird die ID der ScenarioInstanz erwartet. Dabei handelt es sich um einen Integer-Wert.

Dabei kann folgender Fehler geworfen werden:

Error: not a correct scenario instance

Wurde eine ScenarioInstanzID angegeben, die nicht existiert, wird dieser Fehler geworfen.

1.6.

Wenn man eine AktivitätsInstanzId besitzt, und das dazugehörige Label wissen möchte, kann man einen GET-Request mit folgender URL ausführen:

http://172.16.64.113:8080/JEngine/Scenario/ActivityID/{ActivityID}

Variable die benutzt wird:

ActivityID: hier wird die ID der AktivitätsInstanz erwartet. Dabei handelt es sich um einen Integer-Wert.

Dabei kann folgender Fehler geworfen werden:

Error: not correct Activity ID

Wurde eine AktivitätsInstanzID angegeben, die nicht existiert, wird dieser Fehler geworfen.

2. @POST:

Über POST-Request ist es möglich, Aktivitäten zu starten sowie gestartete Aktivitäten zu terminieren. Außerdem besteht auch die Möglichkeit ganze Szenarien zu starten, falls dies benötigt wird. All dies soll mit der Funktion ausgestattet sein, dass man dazu noch einen Kommentar abgeben kann.

2.1.

Um über einen POST-Request Aktivitäten zu starten bzw. zu beenden, wird die folgende URL genutzt:

http://172.16.64.113:8080/JEngine/Scenario/{ScenarioID}/{ScenarioInstanceID}/{ActivityID}/{Status}/comment

Variablen die benutzt werden:

ScenarioID: hier wird die ID des Szenarios erwartet. Dabei handelt es sich um einen Integer-Wert.

ScenarioInstanceID: hier wird die ID der ScenarioInstanz erwartet. Dabei handelt es sich um einen Integer-Wert.

ActivityID: hier wird die ID der AktivitätsInstanz erwartet. Dabei handelt es sich um einen Integer-Wert.

Status: Der Status ist vom Typ String und muss ein Element der Menge `{terminate, begin}` sein.

Im Fehlerfall:

Wird versucht eine Aktivität zu starten, die nicht existiert bzw. eine Aktivität zu beenden, die gar nicht gestartet war, wird einem ein Boolean mit dem Wert *false* zurückgegeben.

2.2.

Wenn eine neue Instanz eines Szenarios gestartet werden soll, nutzt man den POST-Request mit folgender URL:

http://172.16.64.113:8080/JEngine/Scenario/Start/{ScenarioID}

Variable die benutzt wird:

ScenarioID: hier wird die ID des Szenarios erwartet. Dabei handelt es sich um einen Integer-Wert.

Im Fehlerfall:

Wird versucht ein Szenario zu starten, das nicht existiert, wird einem ein Integer mit dem Wert -1 zurückgegeben.

3.1.2 ExecutionService

Der ExecutionService ist die Komponente, auf die die REST-API aufbaut. Diese Komponente der JEngine ist essentiell um Anfragen, die über das REST-Interface entgegengenommen werden, zu bearbeiten. Dabei stellt es eine Verbindung zur Datenbank her, um auf dieser Queries auszuführen. Die meisten Abfragen benötigen eine ID des Objekts, um mehr Informationen über dieses zu gewinnen. Es gibt zudem auch Methoden um Aktivitäten bzw. ganze Szenarien zu instanzieren.

3.2 JComparser

3.3 JFrontEnd

3.4 JDatabase

4. PROCESSEDITOR

5. PCM MODELLING USING THE PROCESSEDITOR

This document explains how to use the Processeditor to create PCM models. A PCM-Process can be described by many PCM fragments and one PCM scenario.

5.1 Preparations

Currently you need both, the Processeditor Workbench and the Processeditor Server to model and Save PCM. You will use the Workbench for modelling and the Server as a global repository.

5.2 PCM Fragments

PCM Fragments are small Business Process models. They can be modelled using a subset of the BPMN-Notation:

- Tasks
- Events ** Blanko Start-Event ** Blanko End-Event
- Gateways ** Parallel Gateway ** Exclusive Gateway
- Data Objects
- Sequence Flow
- Data Flow

All this elements are offered by the model type PCM Fragment.

5.2.1 Marking a Task as Global

PCM allows to use the same task in more than one fragment. To do so

1. model the Task (in one scenario)
2. Save the model to the repository
3. Right click on the Task and choose *Properties*
4. Set the *global flag*

5.2.2 Copy and Refer an existing Task

1. In another Fragment right click on any node
2. Choose "Copy and Refer Task"
3. Connect to the server if necessary
4. Choose the Model and the Task you want to refer
5. Click on Ok

5.3 PCM Scenario

A Scenario defines which PCM Fragments are part of one Process. All PCM Fragments have to be saved on the Server. You can alter the Scenario only by moving the nodes and adding/removing PCM Fragments.

5.3.1 Defining a PCM Scenario

1. Create a new PCM Scenario Model.
2. Right Click on one of the two nodes
3. Choose Add Fragments
4. Mark all Models you want to add in the left List (CTRL for multi select)
5. click on add than on ok

Now there should be entries for all the fragments (inside green node) and for all their data objects (inside white node).

5.3.2 Removing a Fragment From an Scenario

1. Right Click on one of the two nodes
2. Choose *Add Fragments*
3. Select all the models you want to remove from the right list
4. Click on *Remove* than click *Ok*

5.3.3 Set a Termination Condition

If a termination condition is full filled the process is terminated. Currently only one termination condition consisting of one Data Object in one specific state is possible.

1. Open your Scenario
2. Right Click on the canvas (not the Nodes)
3. Choose *Properties*
4. Fill out the *Termination Data Object* and *Termination State* fields

5.3.4 Copy and Alter a Complete Fragment

You can create a variation of an existing PCM Fragment using the Plug-in *Create Variant*.

1. First click on *Plug-Ins*

2. Choose *Create Variant*
3. Choose your Fragment and click on *Ok*

5.4 Processeditor Server

5.5 Processeditor Client

6. REFERENCES

- [1] A. Meyer, N. Herzberg, M. Weske, and F. Puhlmann. Implementation framework for product case management: Modeling and execution. 2013.