Anforderungsdefinition

Transitionen im CWF Workflow



Neutorstraße 13  
5020 Salzburg

AUSTRIA

Tel: +43 (662) 276198-11

Fax: +43 (662) 276198-98

Mail: [office@breanos.com](mailto:office@breanos.com)

|  |  |
| --- | --- |
| Version | 0.1 |
| Datum | 02.101.2018 |

Inhaltsverzeichnis

[1 Änderungsverzeichnis 4](#_Toc526237794)

[2.1 Übersicht 5](#_Toc526237795)

[2.1.1 Erste Transition 6](#_Toc526237796)

[2.1.2 Letzte Transition 6](#_Toc526237797)

[2.1.3 Conditions/ Guards 6](#_Toc526237798)

[2.1.3.1 Transitions mit Bedingung 6](#_Toc526237799)

[2.1.3.2 Transitions ohne Bedingung 6](#_Toc526237800)

# Änderungsverzeichnis

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Verfasser | Beschreibung |
| 2018-10-02 | 0.1 | EBE/FKE | Erweiterung des CWF um die Syntax der Transitionen. |
| 2018-10-02 | 0.2 | FKE | ETL gestrichen, Gatter hinzugefügt. |
| 2018-10-09 | 0.3 | EBE | Updated Transition Syntax. |

## Übersicht

Der CWF Workflow wird von seiner ursprünglichen Syntaxdefinition des Workflows auf eine erweiterte Form mit Transitionen umgebaut.

Transitionen übergeben das allgemeine Zustandsobjekt dabei von einer Aktivität an die nächste und können darauf Konditionen auswerten. Transitionen dürfen dabei am Zustandsobjekt selbst keine Veränderung bewirken, da es sonst zu unberechenbarem Verhalten bei parallel laufenden Transitionen kommen kann.

**Ursprünglicher Aufbau:**

<Workflow xmlns="urn:wexflow-schema" id="10" name="HanoiFlowchart" description="A working workflow definition">

<Settings>

<Setting name="launchType" value="trigger" /> <!-- startup|trigger|periodic -->

<Setting name="enabled" value="true" /> <!-- true|false -->

<Setting name="statemachinetype" value="HanoiWorkflowState" />

</Settings>

<Activities>

<Activity id="99" name="SetupHanoiGameActivity" description="evenBetter" enabled="true">

<Setting name="ParameterType" value="HanoiSetupConfiguration" />

<Setting name="NumberDisks" value="5" />

</Activity>

<Activity id="1" name="CheckHanoiNotFinishedActivity" description="DrScott" enabled="true"/>

<Activity id="2" name="RunHanoiRoundActivity" description="Timewarp" enabled="true" />

</Activities>

<ActivitySetup>

<Activity id="99"><Parent id="-1"/></Activity>

<While id="1" while="1" parent="99" >

<Activity id="2"><Parent id="-1"/></Activity>

</While>

</ActivitySetup>

</Workflow>

**Neue Syntaxform:**

<Workflow xmlns="urn:wexflow-schema" id="10" name="HanoiFlowchart" description="A working workflow definition">

<Settings>

<Setting name="launchType" value="trigger" /> <!-- startup|trigger|periodic -->

<Setting name="enabled" value="true" /> <!-- true|false -->

<Setting name="statemachinetype" value="HanoiWorkflowState" />

</Settings>

<Activities>

~~<Transition from=”-1” to=”99”/>~~

<Activity id="99" name="SetupHanoiGameActivity" description="evenBetter" enabled="true">

<Setting name="ParameterType" value="HanoiSetupConfiguration" />

<Setting name="NumberDisks" value="5" />

</Activity>

~~<Transition from=”99” to=”1”/>~~

<Activity id="1" name="CheckHanoiNotFinishedActivity" description="DrScott" enabled="true"/>

~~<Transition from=”1” to “2”/>~~

<Activity id="2" name="RunHanoiRoundActivity" description="Timewarp" enabled="true" />

~~<Transition from=”2” to “1”/>~~

~~<Transition from=”2” to “-2”/>~~

</Activities>

<Transitions>

<Transition id = "1" from="-1" to = "99" condition = "true"/>

<Transition id = "2" from = "99" to= "1" condition="true" />

<Transition id = "3" from = "1" to = "2" condition="true" />

<Transition id = "4" from = "2" to = "-2" condition="true" />

</Transitions>

~~<ActivitySetup>~~

~~<Activity id="99"><Parent id="-1"/></Activity>~~

~~<While id="1" while="1" parent="99" >~~

~~<Activity id="2"><Parent id="-1"/></Activity>~~

~~</While>~~

~~</ActivitySetup>~~

</Workflow>

### Erste Transition

Zum Ausführen der ersten Transition wird die erste Transition geladen. Diese hat die Quell-Id. -1 und diese geht zur ersten Aktivität die ausgeführt wird.

### Letzte Transition

Die letzte Transition ist daran ersichtlich, dass sie auf eine Aktivität mit der Id -2 geht.

### Conditions/ Guards

#### Transitions mit Bedingung

<Transition from=”99” to=”1” Condition=”object.datenelement > 1”/>

Objekte können innerhalb eines State Tokens angesprochen werden und mit Boolschen Operatoren abgeprüft werden.

#### Transitions ohne Bedingung

<Transition from=”99” to=”1”/>

Transition wird immer ausgeführt.

### ~~ETL~~

~~Die Transition wird generisch gehalten, sodass unterschiedliche Aktivitäten mit unterschiedlichen Hintergrundobjekten trotzdem über die Transitionen mittels Extract-Transform-Load-Funktionalität (ETL) miteinander kommunizieren und funktionieren können. Die Transformationslogik muss dabei explizit definiert werden so es sich nicht um eine Identitätstransformation mit dem selben Datentyp auf beiden Seiten handelt.~~

~~<TransformDefinitions>~~

~~<Transform InputType="MyTypes.DataType1" OutputType="MyOtherTypes.DataType2">~~

~~<ValueTransform InputProperty="DegreesCelsius" OutputProperty="DegreesFahrenheit" TransformRule="value / 5 \* 9 + 32" />~~

~~<TypeTransform InputProperty="MyInteger" OutputProperty="MyDouble" />~~

~~</Transform>~~

~~</TransformDefinitions>~~

~~Als Beschränkung des Typentransformationssystems wäre denkbar, dass die der Eingangstyp explizit in den Ausgangstyp konvertiert werden (casting).~~

### Gatter

Gatter können bei Activities definiert werden um festzulegen, wie mit den Eingangstransitionen verfahren werden soll. Denkbar ist z.B, dass

* Auf alle Eingangstransitionen gewartet wird (UND)
* Bei der ersten Transition die fertig wird schon fortgefahren wird (ODER)

Im Fall des ODER-Gatters müssen alle davor geschalteten Aktivitäten abgebrochen werden damit durch diese das Zustandsobjekt nicht weiter verändert werden kann (Wäre dies gewünscht, würde man ein UND-Gatter verwenden).

### Terminplanung

