YunaML zur Definition von Dashboard Inhalten

Dashboard Inhalte werden im Kontext von YUNA in YunaML definiert und anschließend in JSON umgewandelt, um so in der Datenbank abgelegt zu werden. Die Definition des Inhalts in YunaML statt JSON mag zunächst umständlich erscheinen, hat aber gute Gründe:

Durch die Nutzung des YunaML

- Können einzelne Elemente wiederverwendet werden. Sourcing ermöglicht es, YunaML-Elemente nur an einer Stelle editieren zu müssen, die Änderung aber an vielen Stellen gleichzeitig wirken zu lassen.
- ist eine Versionsverwaltung des Dashboards möglich
- ist es möglich, definierten Dashboard Inhalt parallel über mehrere Instanzen zu deployen
- können einfach Kommentare in den Dashboard-Code geschrieben werden.

UTF-8 Kodierung

YUNAML Dateien müssen in UTF-8 Kodierung abgespeichert werden.

Funktion	Definition	betroffene Widgets

Definition des Widget-Typs	`<widget>`
	`  <widgettype>tabledirective</widgettype>`
	`</widget>`
		• alle

Definition der Position des Widgets im Grid	`<position>`
	`  <x>0</x>`
	`  <y>0</y>`
	`</position>`
		• alle

Definition von Arrays	`<triggerParams>`
	`  <mandatory>`
	`  <list>sensor</list>`
	`  <list>equi</list>`
	`  </mandatory>`
	`  <optional>`
	`  <list>startdate</list>`
	`  <list>enddate<`
	`/list>`
	`  </optional>`
	`</triggerParams>`
		• alle

Definition der Spaltenbreite von Tabellen	`<cols>`
	`  <list>`
	`  ...`
	`  <width>100</width>`
	`  <style>`
	`  ...`
	`  </style>`
	`  ...`
	`  </list>`
	`</cols>`
		Tabellen-Widget

Triggerparameter zur Steuerung von Widgets bzw. deren Inhalten	<triggerParams> <mandatory> <list>sensor</list> <list>equi</list> </mandatory> <optional> <list>startdate</list> <list>enddate</list> <list>selectedRelativePeriod</list> </optional> </triggerParams>	Alle Widgets

Dashboard Inhalte definieren

Objekt mit Array zur Definition der Objekteigenschaften

```
<Objekt>
        <list>Name des ersten Listenelements</list>
        <list>Name des zweiten Listenelements</list>
        <list>Name des dritten Listenelements</list>
</Objekt>
```

JSON

```
"position": { "x": 0, "y": 0 }
```

Objekt mit festen Eigenschaften

XML

```
<size>
    <x>0</x>
    <y>0</y>
<size>
```

JSON

```
"size": { "x": 0, "y": 0 }
```

Source

Bei Code, der exakt gleich an mehreren Stellen des Portals eingesetzt werden soll (z.B. einheitliche Kontextmenüs der Widgets im Portal) können Sie den Befehl **source** nutzen, um sich die Arbeit zu erleichtern. source erzeugt einen link im XML-Code zu einer Quelle, die Sie im Zusammenhang mit source definieren. Die Funktion source lässt sich verschachteln.

Beispiel:

```
<label source="labeldef" />
```

```
<labeldef>Mein Label</labeldef>
```

analog:

```
<wasweisich name="labeldef">Mein Label</wasweisich>
```

Die gesourcten Elemente müssen sich dabei auf der höchsten Ebene im XML befinden.
Also:

```
<xml>
  <meinedefinition>daten</meinedefinition>
</xml>
```

- "widget snippets/definitionen" sollen in der Datei widget.xml angelegt werden
- links werden in dem file Global/glb_Link_Prefixes.xml definiert

Übersetzung von Dashboard Inhalten

Damit ein im Rahmen der Dashboard-Entwicklung definierter Text übersetzt werden kann, muss ein entsprechender Übersetzungsschlüssel definiert werden. Wie der folgenden Tabelle entnommen werden kann, ist es möglich entweder nur den Übersetzungsschlüssel zu definieren oder zusätzlich einen Standardwert zu setzen, der verwendet wird, wenn in der ausgewählten Sprache keine Übersetzung für diesen Schlüssel verfügbar ist.

Werden Dashboard Inhalte mit Übersetzungen via dsedep deployt, werden die Standardwerte in der Portal-Datenbank gespeichert und es ist möglich eine Übersetzungsdatei zu generieren.

Verhalten je nach XML-Definition

	XML-Definition	Verhalten

1.	`<label>`
	
	`Tabellenüberschrift`
	`</label>`
		Da kein Übersetzungsschlüssel definiert ist, wird unabhängig von der gewählten Sprache als Label "Tabellenüberschrift" angezeigt.

2.	`<label>`
	
	`<tkey>content.table.header</tkey>`
	`</label>`
		Da ein Übersetzungsschlüssel, jedoch kein Standartext definiert ist, wird als Label immer der zu dem Übersetzungsschlüssel gehörende Wert der aktuell ausgewählten Sprache angezeigt. Ist keine Übersetzung vorhanden, wird der Übersetzungsschlüssel angezeigt.

3.	`<label>`
	
	`<default>Tabellenüberschrift</default>`
	
	`<tkey>content.table.header</tkey>`
	`</label>`
		Da sowohl Übersetzungsschlüssel als auch Standardtext definiert sind, wird immer der Wert der aktuell ausgewählten Übersetzung angezeigt. Ist in der aktuell ausgewählten Übersetzung kein Wert mit dem entsprechenden Übersetzungsschlüssel hinterlegt, wird der Standard-String, also "Tabellenüberschrift" angezeigt.

Beispiel: Widget mit übersetztem Titel

```
<xml>
  <widget>
    <position>
      <y>8.4</y>
      <x>0</x>
    </position>
    <size>
      <x>3</x>
      <y>5.7</y>
    </size>
    <widgettype>singlechoicedirective</widgettype>
    <caption>
      <show>true</show>
      <label>
                <default>Standard-Widgetüberschrift</default>
                <tkey>content.view1.singlechoicedirective.label</tkey>
      </label>
    </caption>
    <dataID>qy_EquipmentList</dataID>
    <urlParam>equi</urlParam>
    <triggerParams>
      <mandatory>
        <list>filter2</list>
      </mandatory>
    </triggerParams>
  </widget>
</xml>
```