Tree Walk für ELOas

Version : 7.00.027

Ab der Version 7.00.027 steht für die Abarbeitung der Dokumente in den ELO Automation Services auch eine Tree Walk Funktion zur Verfügung. Es können also nicht nur Suchbereiche abgearbeitet werden sondern auch komplette Baumstrukturen durchlaufen werden.

Inhalt

1 Einleitung

Normalerweise führt der ELOas eine Suche nach einem Indexfeld aus um die Liste der zu bearbeitenden Dokumente zu ermitteln. Alternativ kann nun aber auch ein Tree Walk ausgeführt werden. Über diesen Tree Walk können einzelne Archivzweige oder aber auch das komplette Archiv durchlaufen werden. Dabei wird jeder Eintrag zweimal durchlaufen: einmal beim Betreten, danach werden erst mal alle Untereinträge durchlaufen und dann noch mal beim Verlassen.

Beispiel: es gibt einen Schrank mit den Ordnern 1 und 2. Ordner 1 enthält das Register 1.1. Dann ergibt sich folgender Ablauf:

Schrank (betreten)
Ordner 1 (betreten)
Register 1.1 (betreten)
Register 1.1 (verlassen)
Ordner 1 (verlassen)
Ordner 2 (betreten)
Ordner 2 (verlassen)
Schrank (verlassen)

Ob der Ruleset im aufsteigenden Zweig (betreten) oder im absteigenden Zweig (verlassen) aufgerufen wird, kann ein Skript anhand der Variablen EM_TREE_STATE abprüfen. Diese enthält 0 beim Betreten und 1 beim Verlassen. Gespeichert wird nur beim Verlassen. Änderungen, die beim Betreten ausgeführt werden, bleiben aber bis zum Verlassen erhalten, auch dann, wenn in der Zwischenzeit eine Anzahl anderer Objekte bearbeitet wurde.

Initiiert wird ein Tree Walk indem als Suchindex Gruppenname der Wert „TREEWALK“ eingegeben wird und als Suchbegriff die Nummer des Startknotens. Beachten Sie bitte, dass auf dem Startknoten keine Regeln aufgerufen werden. Nur für die Untereinträge wird das durchgeführt.

2 Anwendungsbeispiel

Das folgende Anwendungsbeispiel durchläuft einen Archivzweig und setzt in allen Objekten vom Maskentyp 6 (Track Item) eine interne Id (TrackId). Der Startordner hat die Id 3352.

In diesem einfachen Beispiel ist keine Fehlerbehandlung vorgesehen, die Fehlerregel ist deshalb leer.

```
<ruleset>
 <base>
  <name>Create TrackId</name>
  <search>
   <name>"TREEWALK"</name>
   <value>3352</value>
   <mask>6</mask>
   <max>200</max>
  </search>
  <interval>10M</interval>
 </base>



<rule>
 <name>CreateId</name>
 <script>
  if ((EM_TREE_STATE == 1)  &amp;amp;&amp;amp; (EM_ACT_SORD.getMask()  == 6)) {
   // nur TrackItems bearbeiten
   //cnt.createCounter("ETSTrackId", 10000);

   if (ETS_TICK == "") {
    log.debug("Create new TrackId: " + NAME);

    ETS_TICK = cnt.getTrackId("ETSTrackId", "V");
    EM_WRITE_CHANGED = true;
   }
  }
 </script>
</rule>

<rule>
 <name>Global Error Rule</name>
 <condition>OnError</condition>
 <script>
 </script>
</rule>
</ruleset>
```

Der interessante Teil des Ruleset liegt in dem Skript Bereich, dieser wird deshalb im Folgenden noch mal einzeln aufgeführt:

```
  if ((EM_TREE_STATE == 1)  &&  (EM_ACT_SORD.getMask()  == 6)) {
```

Das Skript soll nur beim Verlassen ausgeführt warden (EM_TREE_STATE == 1) und nur auf Objekte vom Typ TrackItem (EM_ACT_SORD.getMask() == 6).

```
    // nur TrackItems bearbeiten
    //cnt.createCounter("ETSTrackId", 10000);
```

Das Beispiel verwendet einen Counter, der muss vorher angelegt warden, z.B. durch das oben aufgeführte Kommando. Er darf aber nur einmal erzeugt werden, sonst wird die TrackId immer wieder auf den Startwert zurück gesetzt.

```
    if (ETS_TICK == "") {
```

Nur wenn noch keine TrackId vorhanden ist (Indexzeile ETS_TICK ist leer), dann soll eine erzeugt warden.

```
      log.debug("Create new TrackId: " + NAME);

      ETS_TICK = cnt.getTrackId("ETSTrackId", "V");
```

Zur Erzeugung von Track Ids gibt es eine praktische Methode im Counter Modul cnt: getTrackId(<CounterName>, <Prefix>). Diese Methode holt einen neuen Counter Wert und ergänzt ihn mit dem Präfix und einer Checksumme. Im Beispiel wird also aus dem Counter Wert 10001 die TrackId V10001C2.

```
      EM_WRITE_CHANGED = true;
```

Nur wenn eine neue TrackId erzeugt wurde, soll das Objekt gespeichert werden.

```
    }
  }
```

Der Ruleset wird alle 10 Minuten ausgeführt und durchläuft den kompletten Track Item Ordner. Alle Einträge ohne TrackId werden automatisch ergänzt, egal mit welchem Client sie erzeugt wurden.

3 Variablen der Laufzeitumgebung

Wenn das Ruleset ausgeführt wird, gibt es neben dem EM_TREE_STATUS Wert eine Reihe weiterer Variablen, die zur Bearbeitung heran gezogen werden können.

Name	Inhalt
EM_TREE_STATUS	Gibt an, ob der Ruleset im aufsteigenden Ast (0) oder absteigenden Ast (1) ausgeführt wird.
EM_ACT_SORD	Enthält das Sord Objekt mit den aktuellen Objektdaten.
EM_PARENT_SORD	Enthält das Sord Objekt mit den Daten des Parent Knotens. Diese Daten dürfen prinzipiell auch verändert werden. Es muss aber darauf geachtet werden, dass diese Änderungen dann auch gespeichert werden. Dazu muss im absteigenden Ast die Veränderung erkannt werden und das EM_WRITE_CHANGED Flag auf true gesetzt werden.
EM_ROOT_SORD	Enthält das Sord Objekt mit dem Startknoten. Da auf diesen Eintrag der Ruleset nicht angewendet wird, muss bei Veränderungen ein manuelles Speichern erfolgen. Das kann durch Setzen der Variablen EM_SAVE_TREE_ROOT passieren.
EM_INDEX_LOADED	Im Gegensatz zu einer Abarbeitung nach einer Suche kann man beim Tree Walk nicht sicher davon ausgehen, dass ein geladenes Sord Objekt eine bestimmte Maske besitzt. Prinzipiell kann jede beliebige Maske auftreten. Die vorbelegten Indexvariablen können aus den Indexzeilen aber nur für Masken generiert und gefüllt werden, die in der Definition unter <mask> und unter <masks> angemeldet werden. In diesem Fall wird die Variable EM_INDEX_LOADED auf true gesetzt. Falls eine unbekannte Maske vorliegt, ist ein Zugriff auf die Indexzeilen nur direkt über das EM_ACT_SORD Objekt möglich, EM_INDEX_LOADED steht dann auf false. Hinweis: Wenn die Indexvariablen gefüllt wurden, sollte man die Indexzeilen in EM_ACT_SORD nicht direkt bearbeiten. Diese Änderungen gehen vor dem Speichern wieder verloren wenn die Indexvariablen zurück geschrieben werden.
EM_TREE_LEVEL	Über diese Variable kann man feststellten, auf welcher Ebene innerhalb des Tree Walks man sich befindet. Die Untereinträge der Startknotens befinden sich im Level 0 (für den Startknoten werden ja keine Regeln aufgerufen).

EM_TREE_MAX_LEVEL	Über diese Regel kann man die maximale Tiefe festlegen. Noch tiefer geschachtelte Untereinträge werden ignoriert. In Normalfall liegt dieser Wert bei 32. Falls er geändert werden soll, kann er vor der Abarbeitung in der onstart Routine auf den gewünschten Wert gesetzt werden.
EM_SAVE_TREE_ROOT	Für den Startknoten des Tree Walk werden keine Regeln aufgerufen. Falls dieser durch den Zugriff über EM_TREE_ROOT oder EM_PARENT_SORD geändert wurde, muss durch das Setzen der Variablen EM_SAVE_TREE_ROOT eine Speicherung dieser Änderungen angemeldet werden. <onend> var result = ... var oldstate = ... EM_SAVE_TREE_ROOT = result != oldstate; log.debug("now save root: " + EM_SAVE_TREE_ROOT); </onend>
EM_TREE_EVAL_CHILDREN	Wenn beim Durchlauf festgestellt wird, dass ein Unterbereich von der Verarbeitung ausgeschlossen werden soll, kann die Variable EM_TREE_CHILDREN auf false gesetzt werden. Dieser Wert wird nur beim aufsteigenden Zweig ausgewertet (beim absteigenden Ast wäre es ohnehin zu spät, dann ist der Unterbereich ja bereits durchlaufen worden) und er wird bei jedem Objekt mit true initialisiert (Standardverhalten: durchlaufe den vollständigen Unterbereich).
EM_TREE_ABORT_WALK	Wenn ein Durchlauf komplett abgebrochen werden soll, dann kann man zu einem beliebigen Zeitpunkt das Flag EM_TREE_ABORT_WALK gesetzt werden. In diesem Fall werden keine weiteren Untereinträge durchlaufen. Auch weitere Einträge der gleichen Ebene, die noch nicht abgearbeitet wurden, bleiben unbearbeitet. Dieses Flag kann man setzen um die Arbeit nach einem schweren Fehler abzubrechen. Hinweis: In der onstart Routine können notwendige Laufzeitkontrollen durchgeführt werden um zu prüfen, ob der Tree Walk durchgeführt werden darf. Falls nicht, kann durch dieses Flag der Lauf abgebrochen werden.