Arbeiten mit der Baumkomponente

AID 037 DE

Autor: FA, MW, KN. Version 10.1. Zuletzt geändert 04.09.2017

Version	Bemerkung
10.1	Anpassung der Formatierungen
10.0	Anpassung an ADITO4.6
5.2	Letzte Version vor Umstellung der Versionshistorie im Dokument

Inhaltsverzeichnis

1. Einbinden der Baumkomponente

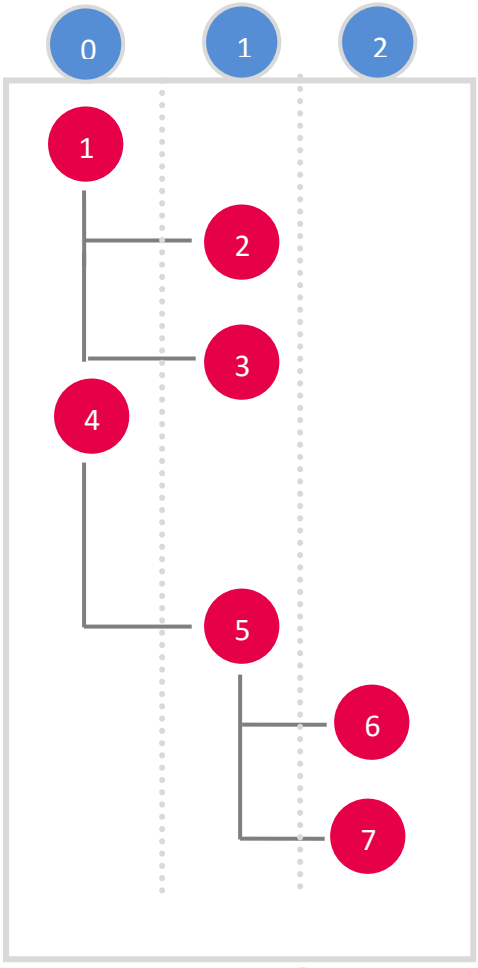
1.1. Ablegen der Baumkomponente auf einem Frame

Starten Sie den Designer und öffnen Sie im Projektordner den System-ALIAS. Erstellen oder öffnen Sie einen Frame, auf dem Sie eine Baumkomponente platzieren möchten.

Klicken Sie in der Symbolleiste mit den verfügbaren Komponenten auf das Symbol zur Erzeugung einer Baumkomponenten Baum.

Ziehen Sie die Komponente durch Anklicken der Ziehpunkte mit der Maus und bei gehaltener linker Maustaste auf die gewünschte Größe.

1.2. Logik der Baumkomponente



Legende: NodeID NodeLayer

Abbildung 1: Aufbau eines Baumes

1.3. Füllen der Baumkomponente mit Daten

Die Baumkomponente wird über den Prozess `contentProcess` auf dem Registerreiter `Baum` des Komponenteninspektors mit Daten befüllt.

1.3.1. Rekursiver Aufruf des Prozesses

Dieser Prozess wird für jede Ebene (**NodeLayer**) der Baumdarstellung rekursiv aufgerufen. Bei jedem Klick auf diese Ebene wird die markierte ID (**NodeID**) automatisch ermittelt.

1.3.2. Lokale Variablen

In dem Prozess `Baum füllen` stehen Ihnen zwei lokale Variablen zur Verfügung, die von außen befüllt werden.

Diese sind:

- `$local.nodeid`: Enthält die `nodeid` des Eltern-Knotens zum eben betrachteten Kind-Knoten.
- `$local.nodelayer`: Enthält die Baumebene, die betrachtet wird.

1.3.3. Datenstruktur zum Befüllen der Baumkomponente

Der Prozess `contentProcess` der Baumkomponente muss mit der JDito-Methode `returnobject` ein zweidimensionales Array zurückliefern.

Dabei enthält die erste Dimension einen Eintrag pro Knoten der betrachteten Ebene, die zweite Dimension die folgenden Informationen für jeden Knoten:

nodeid Gibt die ID des Knotens an, der betrachtet wird.

displayValue Gibt die angezeigte Bezeichnung des Knotens an.

tooltip Setzt den Tooltip, der angezeigt wird, wenn man mit der Maus über dem entsprechenden Knoten stehen bleibt.

icon Definiert, welches Icon für die Darstellung des Knotens verwendet werden soll. Hier wird die ID eines Datensatzes aus der Systemtabelle ASYS_ICONS angegeben, der die Bilddatei enthält, die verwendet werden soll. Wird hier nichts angegeben, werden die Standard-Icons (gelbes Ordnersymbol für Knoten mit Kind-Elementen und Dokument-Symbol für Knoten ohne Kind-Elemente), wie man sie aus dem Windows Explorer kennt, verwendet.

isLeaf Mit diesem Element geben Sie an, ob der betrachtete Knoten ein Blatt-Knoten ist, d.h. ob noch weitere Knoten unter diesem Knoten angesiedelt sind. Bei Blatt-Knoten wird vor dem Knoten kein Plus-Zeichen zum Aufklappen des Knotens angezeigt. Können Sie ermitteln, ob der Knoten ein Blatt-Knoten ist, sollten Sie diesen Wert auch setzen! Das erspart dem Benutzer unnötiges "Öffnen" des Knotens.

parentID Dieses Element gibt die ID des Eltern-Knotens an, dem der betrachtete Knoten untergeordnet werden soll.

autoOpen Mit dieser Einstellung legen Sie fest, ob der Knoten automatisch expandiert werden soll.

1.3.4. NodeID oder NodeLayer?

Der Prozess, dem das Objekt zurückgegeben wird, wird immer auf die markierte Ebene ausgeführt. Allerdings kann man diesen Prozess entweder auf Basis der aktuellen `NodeID` bzw. des aktuellen `NodeLayers` ausführen.

- Wird der Prozess auf `NodeLayer`-Ebene ausgeführt, so muss immer der Mutterknoten angegeben werden, damit bekannt ist, zu welcher Mutter der entsprechende Knoten gehört.
- Wird der Prozess auf `NodeID`-Ebene ausgeführt, so ist die Angabe des Mutterknotens nicht notwendig, da ohnehin für jeden Knoten die Kindknoten mit ermittelt werden.

1.3.5. Aufbau des Prozesses zum Baum füllen

Zunächst schaffen Sie die Infrastruktur zum Füllen des Baumes mit Daten, d.h. Sie werten die von außen vorgegebenen Variablen `$local.nodeid` und `$local.nodelayer` aus und erzeugen das Array-Objekt, das an die Baumkomponente zurückgegeben werden soll.

```
var nodeid = vars.getString("$local.nodeid");
var nodelayer = vars.getString("$local.nodelayer");


var res = new Array();
(…)
result.object(res);
```

1.3.5.1. Füllen des Baumes mit festen Werten und nodeID

Legen Sie fest, welche Wurzelknoten (Knoten der ersten Ebene) der Baum haben soll. Da in der Variable `nodeid` die ID des Eltern-Knotens des betrachteten Knotens enthalten ist, ist `nodeid` hier noch `null`.

```
if (nodeid == null)
{
  res[0] = new Array("1", "Sportarten", "Alles über Sport", null,
"false", null, "false");
  res[1] = new Array("2", "Musikinstrumente", "Music is your
friend", null, "false", null, "false");
  res[2] = new Array("3", "Fernsehsendungen", "Serien und
Spielfilme", null, "false", null, "false");
}
```

Definieren Sie, welche Knoten unter den oben definierten Wurzelknoten angesiedelt werden.

```
if (nodeid == "1")
{
```

```
  res[0] = new Array("4", "Fußball", null, null, "true", null,
"false");
  res[1] = new Array("5", "Schwimmen", null, null, "true", null,
"false");
  res[2] = new Array("6", "Wandern", null, null, "true", null,
"false");
}

if (nodeid == "2")
{
  res[0] = new Array("7", "Flöte", null, null, "true", null,
"false");
  res[1] = new Array("8", "Klavier", null, null, "true", null,
"false");
  res[2] = new Array("9", "Geige", null, null, "true", null,
"false");
}

if (nodeid == "3")
{
  res[0] = new Array("10", "Lassie", null, null, "true", null,
"false");
  res[1] = new Array("11", "Tatort", null, null, "true", null,
"false");
  res[2] = new Array("12", "Dirty Dancing", null, null, "true",
null, "false");
}
```

So sieht dann der gesamte Prozess zum Füllen der Baumkomponente mit Daten aus:

```
import("system.result");
import("system.vars");

var nodeid = vars.getString("$local.nodeid");
var nodelayer = vars.getString("$local.nodelayer");

var res = new Array();

if (nodeid == null)
{
  res[0] = new Array("1", "Sportarten", "Alles über Sport", null,
"false", null, "false");
```

```
  res[1] = new Array("2", "Musikinstrumente", "Music is your
friend", null, "false", null, "false");
  res[2] = new Array("3", "Fernsehsendungen", "Serien und
Spielfilme", null, "false", null, "false");
}

if (nodeid == "1")
{
  res[0] = new Array("4", "Fußball", null, null, "true", null,
"false");
  res[1] = new Array("5", "Schwimmen", null, null, "true", null,
"false");
  res[2] = new Array("6", "Wandern", null, null, "true", null,
"false");
}

if (nodeid == "2")
{
  res[0] = new Array("7", "Flöte", null, null, "true", null,
"false");
  res[1] = new Array("8", "Klavier", null, null, "true", null,
"false");
  res[2] = new Array("9", "Geige", null, null, "true", null,
"false");
}

if (nodeid == "3")
{
  res[0] = new Array("10", "Lassie", null, null, "true", null,
"false");
  res[1] = new Array("11", "Tatort", null, null, "true", null,
"false");
  res[2] = new Array("12", "Dirty Dancing", null, null, "true",
null, "false");
}

result.object(res);
```

Dieser Prozess liefert folgendes Ergebnis:

Das Füllen einer Baumstruktur mit festen Werten ist also mit relativ viel Aufwand verbunden.

Natürlich kann eine Baumstruktur auch dynamisch aus den Werten einer Tabelle gefüllt werden.

1.3.5.2. Füllen eines Baumes mit festen Werten und NodeLayer

```
import("system.result");
import("system.vars");


var nodeid = vars.getString("$local.nodeid");
var nodelayer = vars.getString("$local.nodelayer");


//res[0] = new Array("1", "Knoten 1. ID " + nodeid + ", Layer " +
nodelayer, ".", null, "false", null, false);
//                      id , Anzeigewert
,Tool, icon, isLeaf ,parnt, auto
var res = new Array();


if(nodelayer == 0)
{
 res[0] = new Array("1", "Essen", "Tooltip", null, "false", null,
false);
}


if(nodelayer == 1)
{
```

```
 res[0] = new Array("2", "Fastfood", "Tooltip", null, "false", "1",
false);
 res[1] = new Array("3", "Quality Food", "Tooltip", null, "false",
"1", false);
}


if(nodelayer == 2)
{
 res[0] = new Array("4", "Burger", "Tooltip", null, "true", "2",
false);
 res[1] = new Array("5", "Pommes", "Tooltip", null, "true", "2",
false);
 res[2] = new Array("6", "Risotto", "Tooltip", null, "true", "3",
false);
 res[3] = new Array("7", "Tom Yum", "Tooltip", null, "true", "3",
false);
}


result.object(res);
```

Ein bekannter Fallstrick ist hier die `nodeID`. Ist diese nicht im ganzen Prozess eindeutig, so kann der Baum nicht befüllt werden und bricht mit einer Fehlermeldung ab.

1.3.5.3. Dynamisches Füllen des Baumes mit Werten aus einer Tabelle

Beispiel 1

```
import("system.result");
import("system.vars");
import("system.db");


var nodeid = vars.getString("$local.nodeid");
var nodelayer = vars.getString("$local.nodelayer");


var res = new Array();
if (nodeid == null)
{
  //nodeID displayValue tooltip icon isLeaf parentID, autoOpen
  var res = db.table("select baumtabelleid, bezeichnung,
bezeichnung, '', 'false', '', '' from baumtabelle where superid =
,0`");


  for (var i=0; i<res.length; i++)
```

```
  {
    res[i][3] = null;
    res[i][5] = null;
    res[i][6] = false;
  }
}

if (nodeid != null)
{
  var res = db.table("select baumtabelleid, bezeichnung,
bezeichnung, '', 'true', '', '' from baumtabelle where superid = ,"
+ nodeid + "`");

  for (var i=0; i<res.length; i++)
  {
    res[i][3] = null;
    res[i][5] = null;
    res[i][6] = false;
  }
}

result.object(result);
```

Achten Sie hier darauf, dass Sie die Werte für `icon` und `parentID`, sofern Sie sie nicht angeben haben, im Ergebnis-Array tatsächlich auf `null` setzen! Die Leerstrings führen zu einem Fehler.

Natürlich müssen Sie die im Beispiel angegebenen Anfragen an die Datenbank nach Ihren Bedürfnissen abändern!

Beispiel 2

Der folgende Codeblock erstellt einen Baum, der in der ersten Ebene alle Organisationen anzeigt, in der zweiten Ebene alle dazugehörigen Kontaktpersonen und in der dritten Ebene alle Historien der jeweiligen Personen.

```
import("system.result");
import("system.vars");
import("system.db");

var nodeid = vars.getString("$local.nodeid");
var nodelayer = vars.getString("$local.nodelayer");
```

```
var res = new Array();
if (nodeid == null)
{
 //nodeID displayValue tooltip icon isLeaf parentID, autoOpen
 var res = db.table("select 'o_' + orgid, orgname, 'test', '',
'false', '', '' from Org ");

  for (var i=0; i<res.length; i++)
    {
     res[i][3] = null;
     res[i][5] = null;
     res[i][6] = false;
    }
}

if (nodeid != null && nodelayer == 1)
{
 //nodeID displayValue tooltip icon isLeaf parentID, autoOpen
 var res = db.table("select 'r_' + relationid, Firstname + ' ' +
Lastname, 'test', '', 'false', '', '' "
 + " from Relation, Pers where pers.persid = relation.pers_id and
'o_' + relation.org_id = '" + nodeid + "' and relation.status = 1");
//orgid  264e6319-e2fc-4ee1-9f8a-96247de48471
  for (var i=0; i<res.length; i++)
    {
     res[i][3] = null;
     res[i][5] = null;
     res[i][6] = false;
    }
}

if (nodeid != null && nodelayer == 2)
{
 //nodeID displayValue tooltip icon isLeaf parentID, autoOpen
 var res = db.table("select 'h_' + historyid, subject, 'test', '',
'true', '', '' from History, Historylink "
 + "where 'r_' + Historylink.row_id = '" + nodeid + "' and
historylink.history_id = history.historyid ");
```

```
 for (var i=0; i<res.length; i++)
  {
   res[i][3] = null;
   res[i][5] = null;
   res[i][6] = false;
  }
}

result.object(res);
```

1.4. Markieren von Knoten

Damit Knoten des Baums markiert werden können, muss die Eigenschaft `Immer editierbar` auf dem Registerreiter `Baum` des Komponenteninspektors auf `Ja` stehen.

1.5. Ausführen bei Doppelklick

Durch den Prozess `Ausführen bei Doppelklick` können keine Aktionen ausgelöst werden, die sich auf einen spezifischen Knoten beziehen. Hier können nur Aktionen hinterlegt werden, die unabhängig vom markierten oder angeklickten Knoten funktionieren.

Um Aktionen auf dem angeklickten Knoten auszuführen, verwenden Sie das Kontextmenü.

1.6. Kontextmenü

Über ein Kontextmenü können Aktionen auf dem angeklickten Knoten ausgeführt werden.

Erstellen Sie dazu einen Kontextmenüeintrag. Innerhalb des Prozesses für den Kontextmenüeintrag können Sie mit `vars.getString("$comp.<Name der Baumkomponente>.context")` auf die `nodeid` des angeklickten Knotens zugreifen und diese weiter verwenden. Um einen Wert der Baumkomponente über `.context` ermitteln zu können, muss dieser zunächst in eine eigene Komponente (z.B. unsichtbares Label oder Editfeld) eingetragen werden.

1.6.1. Nodeid des angeklickten Knotens anzeigen lassen

Ersetzen Sie in folgendem Code den Namen "Baum" durch den Namen Ihrer Baumkomponente.

```
question.showMessage(vars.getString("$comp.Baum.context"));
```

1.6.2. Verknüpften Frame öffnen mit Übergabe der Nodeid des angeklickten Knotens

Die JDito-Methode `openLinkedFrame` benötigt für die Verknüpfungsbedingung eine Komponente, über die verknüpft werden kann. Da der Wert der Baumkomponente nur über `.context` ermittelt werden kann, muss dieser zunächst in eine eigene Komponente (z.B. unsichtbares Label oder Editfeld) eingetragen werden, auf der wiederum die Verknüpfung arbeitet.

```
swing.setValue("$comp.node", vars.getString("$comp.Baum.context"));
swing.openLinkedFrame("ORG", null, false, swing.FRAMEMODE_SHOW,
"$comp.node|ORG.ORGID");
```

Ersetzen Sie auch in diesem Code die Komponentennamen durch die von Ihnen verwendeten Komponentennamen.

2. Testen des dynamischen Füllen des Baumes mit Werten aus einer Tabelle

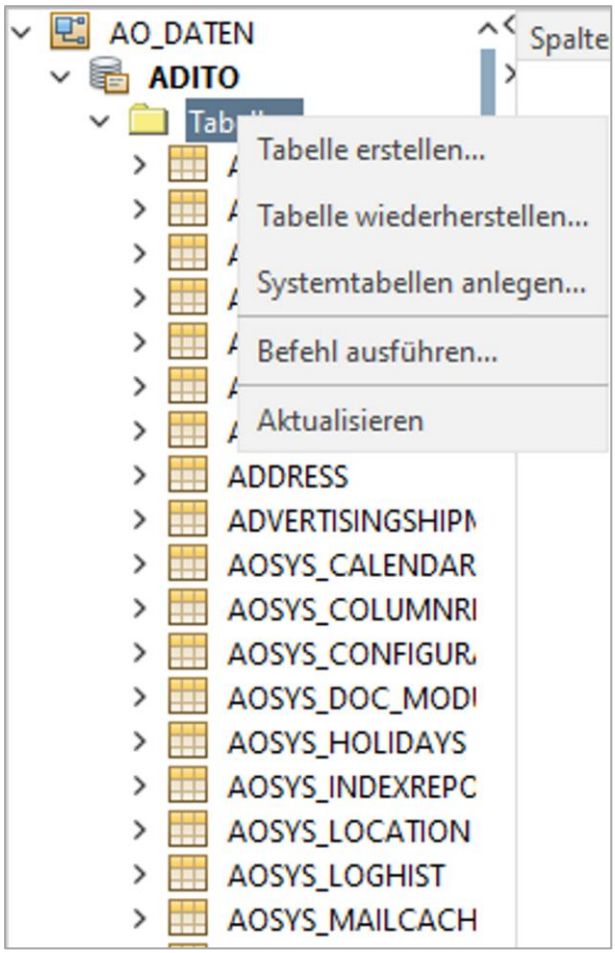
Um das dynamische Füllen des Baumes in Kapitel 1.3.5.3 testen zu können, benötigen Sie folgende Testtabelle (hier Baumtabelle genannt) mit einigen Datensätzen.

2.1. Beispieltabelle Baumtabelle

Step 1: Öffnen Sie im Projektordner den System-ALIAS im System-Editor [SE].

Step 2: Öffnen Sie die Tabellen durch Doppelklick auf den Alias.

Step 3: Öffnen Sie die Mappe ADITO und klicken Sie mit der rechten Maustaste auf „Tabellen“ und erstellen Sie die neue Tabelle „Baumtabelle“.



Und verwenden Sie dabei folgende Ausprägungen:

Spaltenname	Ausprägung
BAUMTABELLEID	CHAR(36)
SUPERID	CHAR(36)
BEZEICHNUNG	VARCHAR(30)

Spaltenname	Ausprägung
DATE_EDIT	TIMESTAMP
DATE_NEW	TIMESTAMP
USER_EDIT	VARCHAR(30)
USER_NEW	VARCHAR(30)

2.2. Einfügen einiger Beispielsätze

Legen Sie einige Datensätze in der Tabelle gemäß folgendem Beispiel an:

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('1', NULL, NULL,
'RP1', '0', 'Sportarten')
```

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('2', NULL, NULL,
'RP1', '0', 'Musikinstrumente')
```

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('3', NULL, NULL,
'RP1', '0', 'Fernsehsendungen')
```

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('4', NULL, NULL,
'RP1', '1', 'Fussball')
```

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('5', NULL, NULL,
'RP1', '1', 'Schwimmen')
```

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('7', NULL, NULL,
'RP1', '2', 'Flöte')
```

```
INSERT INTO BAUMTABELLE (BAUMTABELLEID, DATE_EDIT, DATE_NEW,
USER_EDIT, USER_NEW, SUPERID, BEZEICHNUNG) VALUES ('10', NULL, NULL,
'RP1', '3', 'Lassie')
```