Inhaltsverzeichnis

1 Knowledge-Builder ................................................. 6
   1.1 Grundlagen .................................................. 6
      1.1.1 Grundbausteine ........................................... 6
      1.1.2 Typhierarchie - Vererbung .............................. 9
      1.1.3 Objekte anlegen und bearbeiten ......................... 11
      1.1.4 Graph-Editor ............................................. 14
   1.2 Schemadefinition / Modell .................................. 22
      1.2.1 Typen definieren .......................................... 22
      1.2.2 Relations- und Attributtypen ......................... 29
      1.2.3 Modelländerungen ....................................... 34
      1.2.4 Darstellung von Schemata im Graph-Editor .......... 37
      1.2.5 Metamodellierung und fortgeschrittene Konstrukte .... 40
   1.3 Suchen / Abfragen ............................................. 47
      1.3.1 Strukturabfragen ......................................... 47
      1.3.2 Einfache Suche / Volltextsuche ....................... 58
      1.3.3 Such-Pipeline ............................................ 64
      1.3.4 Die Suche im Knowledge-Builder ....................... 75
      1.3.5 Spezialfälle ............................................. 75
   1.4 Ordner und Registrierung .................................... 76
   1.5 Import und Export ............................................ 77
      1.5.1 Abbildungen von Datenquellen ......................... 78
      1.5.2 Attributtypen und -formate ............................ 114
      1.5.3 Konfiguration des Exports ............................. 116
      1.5.4 RDF-Import und -Export ................................ 118
      1.5.5 Gelöschte Individuen aus einem Backup wiederherstellen ............................................. 119
   1.6 Zugriffsrechte und Trigger ................................. 122
      1.6.1 Die Prüfung von Zugriffsrechten ................. 122
      1.6.2 Trigger ............................................... 134
      1.6.3 Filterarten ............................................. 141
      1.6.4 Operationsparameter .................................. 151
      1.6.5 Operationen ............................................ 159
      1.6.6 Testumgebung .......................................... 165
   1.7 View-Konfiguration .......................................... 168
      1.7.1 Konzept ............................................... 170
1.7.2 Menus .................................................................................................................. 173
1.7.3 Aktionen ................................................................................................................. 177
1.7.4 View-Konfigurationselemente .................................................................................. 196
1.7.5 Panels ...................................................................................................................... 232
1.7.6 Grundlagen der View-Konfiguration ....................................................................... 239
1.7.7 Knowledge-Builder-Konfiguration .......................................................................... 241
1.7.8 Style ......................................................................................................................... 247
1.7.9 Detektorsystem zur Ermittlung der View-Konfiguration ........................................ 253
1.8 JavaScript-API ............................................................................................................ 255
   1.8.1 Einführung ............................................................................................................ 255
   1.8.2 Beispiele ............................................................................................................... 259
   1.8.3 Module ................................................................................................................. 273
   1.8.4 Debugger .............................................................................................................. 275
   1.8.5 API-Erweiterungen .............................................................................................. 276
1.9 REST-Services ............................................................................................................ 278
   1.9.1 Konfiguration ....................................................................................................... 279
   1.9.2 Services ............................................................................................................... 279
   1.9.3 Ressourcen .......................................................................................................... 279
   1.9.4 CORS .................................................................................................................... 288
1.10 Berichte und Drucken .............................................................................................. 289
   1.10.1 Druckvorlagen erstellen ...................................................................................... 289
   1.10.2 Druckvorlagen für Listen erstellen ...................................................................... 296
   1.10.3 Dokumentformatkonvertierung mit Open/LibreOffice ........................................ 298
1.11 Entwicklungsunterstützung ....................................................................................... 299
   1.11.1 Dev-Tools .......................................................................................................... 299
   1.11.2 Dev-Service ........................................................................................................ 299

2 Admin-Tool .................................................................................................................. 300
   2.1 Startfenster .............................................................................................................. 300
      2.1.1 Server ................................................................................................................ 300
      2.1.2 Wissensnetz ...................................................................................................... 301
      2.1.3 Info ...................................................................................................................... 301
      2.1.4 Verwalten, Neu und Weiter .............................................................................. 302
      2.1.5 Ende ................................................................................................................... 302
   2.2 Wissensnetzerzeugung ............................................................................................. 302
      2.2.1 Server ................................................................................................................ 303
      2.2.2 Neues Wissensnetz ......................................................................................... 303
| 2.2.3 | Passwort (Mediator) | 304 |
| 2.2.4 | Lizenz | 304 |
| 2.2.5 | Benutzername | 304 |
| 2.2.6 | Passwort (Benutzer) | 304 |
| 2.2.7 | OK und Abbrechen | 304 |
| 2.3 | Serververwaltung | 304 |
| 2.3.1 | Netzübersicht | 305 |
| 2.3.2 | Nachrichtenfeld | 305 |
| 2.3.3 | Menüzeile | 306 |
| 2.4 | Einzelnetzverwaltung | 308 |
| 2.4.1 | Nutzerauthentifizierung | 308 |
| 2.4.2 | Einzelnetzverwaltungsfenster | 309 |
| 3 | ViewConfig-Mapper | 339 |
| 3.1 | Einführung | 339 |
| 3.2 | Konfiguration | 340 |
| 3.2.1 | View-Konfigurationen für den Viewconfig Mapper | 341 |
| 3.2.2 | Login-Konfiguration | 348 |
| 3.3 | Aktionen | 349 |
| 3.4 | View-Konfigurationselemente | 351 |
| 3.4.1 | Allgemeines | 351 |
| 3.4.2 | Alternative | 352 |
| 3.4.3 | Gruppe | 354 |
| 3.4.4 | Hierarchie | 356 |
| 3.4.5 | Eigenschaften | 360 |
| 3.4.6 | Eigenschaft | 362 |
| 3.4.7 | Edit | 365 |
| 3.4.8 | Tabelle | 366 |
| 3.4.9 | Suche | 371 |
| 3.4.10 | Graph-Konfiguration | 383 |
| 3.4.11 | Text | 386 |
| 3.4.12 | Bild | 387 |
| 3.4.13 | Skriptgeneriertes HTML | 387 |
| 3.4.14 | Skriptgenerierte View | 388 |
| 3.5 | Plugins | 389 |
| 3.5.1 | vcm-plugin-calendar | 389 |
| 3.5.2 | vcm-plugin-chart | 390 |
1 Knowledge-Builder

1.1 Grundlagen

Mit i-views funktionieren Datenbanken so, wie Menschen denken: einfach, agil, flexibel. Deswegen ist in i-views einiges anders als bei relationalen Datenbanken: Wir arbeiten nicht mit Tabellen und Keys, sondern mit Objekten und Beziehungen zwischen ihnen. Die Modellierung der Daten ist visuell und beispielorientiert, so dass wir sie auch mit den Nutzern aus den Fachabteilungen teilen können.

Wir bauen mit i-views keinen reinen Datenspeicher, sondern intelligente Datennetze, die bereits viel Business-Logik enthalten und mit denen das Verhalten unserer Anwendung schon weitgehend bestimmt werden kann. Dazu nutzen wir Vererbung, Mechanismen zum Schlussfolgern und zur Definition von Sichten sowie eine Vielzahl von Suchverfahren, die i-views bietet.

Unser zentrales Werkzeug dazu ist der Knowledge-Builder, eine der Kernkomponenten von i-views. Mit dem Knowledge-Builder können wir:

- Schemata definieren, aber auch Beispiele aufbauen und vor allem visualisieren
- Importe und Abbildungen einer Datenquelle definieren
- Abfragen formulieren, vernetzte Daten traversieren, Strings verarbeiten und Nähen berechnen
- Rechte, Trigger und Sichten definieren

Alle diese Funktionen sind Gegenstand dieser Dokumentation. Als durchgehendes Beispiel dient dabei ein semantisches Netz rund um Musik, Bands, Songs etc.

1.1.1 Grundbausteine

Die Grundbausteine der Modellierung in i-views sind:

- konkrete Objekte
- Beziehungen
- Attribute
- Objekttypen
- Beziehungstypen
- Attributtypen

**Konkrete Objekte:** Beispiele für konkrete Objekte sind John Lennon, die Beatles, Liverpool, das Konzert in der Litherland Town Hall, die Fußball-WM 1970 in Mexiko, der schief Turm von Pisa etc.:
Beziehungen werden in i-views als Relationen bezeichnet. Diese konkreten Objekte können wir durch Beziehungen miteinander verbinden: „John Lennon ist Mitglied der Beatles“, „Die Beatles treten auf bei einem Konzert in der Litherland Town Hall“.


Ein konkretes Objekt ist dann einem Objekttyp zugehörig, wenn man eine "ist ein"-Beziehung ziehen kann. Die "ist ein"-Beziehung ist gleichbedeutend mit "hat Typ"-Beziehung; in i-views heißt die Beziehung standardmäßig "ist konkretes Objekt von". Beispiele: John Lennon (konkretes Objekt) "ist eine" Person (Objekttyp), The Beatles (konkretes Objekt) "ist eine" Band (Objekttyp) und Konzert in Litherland Town Hall (konkretes Objekt) "ist ein" Konzert (Objekttyp).
Das Hauptfenster von i-views: Links die Objekttypen, rechts die dazugehörigen konkreten Objekte - Hier sehen wir auch: Die Typen der i-views-Netze stehen in einer Hierarchie. Mehr zur Typenhierarchie erfahren Sie im nächsten Abschnitt Typenhierarchie - Vererbung.


Attributtypen: Neben konkreten Objekten und Beziehungen können auch Attribute Typen haben. Im Fall einer Person können dies der Name oder das Geburtsdatum sein. Konkrete Personen (Objekte des Typs 'Person') können dann Namen, Geburtsdatum, Geburts- und Wohnorte, Augenfarbe etc. als Attributtypen haben. Veranstaltungen können einen Ort und eine Zeitspanne haben. Attribute und Relationen werden immer beim Objekt selbst definiert.
1.1.2 Typhierarchie - Vererbung


So bekommen wir eine Hierarchie von Ober- und Untertypen:


Die hierarchische Struktur muss nicht zwangsläufig die Struktur eines Baumes haben, in der jeder Typ nur genau einen Obertyp haben kann. In einer semantischen Graphdatenbank kann ein Objekttyp mehrere Obertypen haben. Ein konkretes Objekt hingegen kann aber nur genau einen Obertyp haben.

Am Beispiel des konkreten Konzerts von Paul McCartney 1990, das sowohl ein Konzert, als auch eine Großveranstaltung ist, kann man sehen, was das bedeutet. Da das konkrete Konzert nicht zwei Objekttypen haben kann, wird ein neuer Objekttyp benötigt, der die Aspekte Konzert und Großveranstaltung zusammenführt, hier “Stadionkonzert”: 
Typenhierarchie mit Mehrfachvererbung

Die Zugehörigkeit von konkreten Objekten zu einem Objekttyp wird in i-views ebenfalls als Relation ausgedrückt und kann als solche abgefragt werden:

Wann unterscheiden wir überhaupt Typen? Typen unterscheiden sich nicht nur ggf. in Icon und Farbe - bei den Objekttypen werden auch die Eigenschaften definiert und nach Typen kann bei Abfragen ganz einfach gefiltert werden. In allen diesen Fragen spielt die Vererbung eine wichtige Rolle: Eigenschaften vererben sich, auch Eigenschaften, welche die Darstellung im Knowledge-Builder beeinflussen wie Icons und Farben vererben sich. Und wenn wir in Abfragen sagen, dass wir Veranstaltungen sehen wollen, dann werden auch alle Objekte der Untertypen als Ergebnis angezeigt.
Vererbung macht es möglich, Beziehungstypen (und Attributtypen) weiter oben in der Objekttypen-Hierarchie zu definieren und damit für verschiedene Typen von Objekten (z.B. für Bands und andere Organisationen) zu nutzen.

1.1.3 Objekte anlegen und bearbeiten

Anlegen von konkreten Objekten
Konkrete Objekte lassen sich im Knowledge-Builder überall dort anlegen, wo Objekttypen zu sehen sind. Ausgehend von den Objekttypen lassen sich über Kontextmenüs die Objekte neu anlegen.
Über die Schaltfläche "Neu" kann ein neues Objekt angelegt werden. Lediglich der Name des Objektes muss zunächst angegeben werden.

Im Hauptfenster befindet sich unter der Kopfzeile die Liste mit bereits vorhandenen konkreten Objekten. Damit Objekte nicht versehentlich doppelt angelegt werden, lässt sich über das Suchfeld in der Kopfzeile der Name des Objektes suchen. Die Suche unterscheidet per Default nicht zwischen Groß- und Kleinschreibung und der Suchbegriff lässt sich links und rechts abschneiden (durch Platzhalter „*“ und „?“ ergänzen):

Die Suche nach "Paul*" zeigt uns, dass es bereits 2 Personen mit dem Namen "Paul" gibt.

**Bearbeiten von Objekten**

Nach Eingabe und Bestätigung des Objektnamens können im Editor weitere Details für das angelegte Objekt eingaben werden. Dem Objekt lassen sich über Schaltflächen Attribute, Relationen und Erweiterungen zuweisen.
Bei der Bearbeitung eines Objekts können wir neben der Verknüpfung mit einem anderen Objekt gleichzeitig auch das Ziel der Verknüpfung neu erzeugen, sofern es noch nicht existiert.

Beispielsweise sollen Mitglieder einer Musikgruppe vollständig erfasst werden. Über die Relation *hat Mitglied* soll das Zielobjekt Ringo Starr mit dem Objekt „The Beatles“ verknüpft werden. Falls noch nicht bekannt ist, ob das Objekt Ringo Starr schon in i-views erfasst ist, kann im Eingabefeld gesucht werden. Gibt man also "Ringo Starr" ein und bestätigt die Eingabe, und es ist noch kein Objekt mit diesem Namen vorhanden, so öffnet sich ein Dialog, der fragt, ob man ein neues Objekt mit diesem Namen anlegen möchte. Sollte es mehrere Ringo Starrs geben, öffnet sich ein Auswahlfenster.

Man hat zudem die Möglichkeit über die Schaltfläche *Relationsziel wählen* aus einer durchsuchbaren Liste mit allen möglichen Relationszielen ein Objekt auszuwählen.
Das Löschen der Relation hat Mitglied kann auf zwei Arten vorgenommen werden:

1. Im Kontextmenü unter der Schaltfläche Weitere Aktionen mit der Option Löschen.

Das Zielobjekt der Relation selbst ist damit jedoch nicht gelöscht. Soll ein Objekt gelöscht werden, so geht das mit der Schaltfläche im Hauptfenster oder über das Kontextmenü direkt auf diesem Objekt.

Objekte können auch über den Graph-Editor angelegt werden. Das Vorgehen hierzu wird im den nachfolgenden Kapitel beschrieben.

1.1.4 Graph-Editor

1.1.4.1 Einführung Graph-Editor

Mit dem Graph-Editor lassen sich die Inhalte der semantischen Graphdatenbank mit ihren Objekten und Verbindungen graphisch darstellen. Die Objekte werden als Knoten dargestellt, ihre Beziehungen untereinander als Pfeile:
**Zugang zum Graph-Editor:** Mit der Schaltfläche *Graph* kann der Graph-Editor auf einem selektierten Objekt geöffnet werden. Diese Schaltfläche findet sich im Knowledge-Builder an mehreren Stellen wieder:

Der Graph zeigt immer nur einen Ausschnitt des Netzes. Objekte können in der Graphansicht ein- und ausgesperrt und es kann durch den Graph navigiert werden.

Doch nicht nur zur übersichtlichen Darstellung der Objekte und Relationen eignet sich der Graph. Im Graph-Editor können Objekte und Relationen außerdem editiert werden.

Auf der linken Seite eines Knotens befindet sich ein Anfasser in Form eines Plus-Zeichens für die Interaktion mit dem Objekt. Durch einen Doppelklick auf den Anfasser werden alle Benutzerrelationen des Objekts angezeigt bzw. ausgeblendet.

Das Verknüpfen von Objekten über eine Relationen wird im Graph-Editor wie folgt vorgenommen:

1. Den Cursor über dem Anfasser links vom Objekt mit der linken Maustaste positionieren.
Um Objekte im Graph-Editor anzuzeigen, gibt es verschiedene Möglichkeiten:

- Objekte können aus der Trefferliste im Hauptfenster von i-views mit Drag&Drop in das Fenster des Graph-Editors hineingezogen werden.
- Wenn der Name des Objekts bekannt ist, kann durch einen Rechtshlick auf eine leere Stelle im Graph-Editor das Kontextmenü geöffnet werden, wo über die Funktion „Objekt anzeigen“ der Name des gewünschten Objektes eingegeben werden kann.

Soll ein Objekt im Graph-Editor ausgebldendet werden, kann es durch Anklicken mit gleichzeit-
ig gedrückter Strg-Taste aus dem Graph-Editor entfernt werden. Damit wird nichts in den Daten geändert: Das Objekt existiert unverändert im semantischen Netz, es wird nur nicht mehr im aktuellen Graph-Editor-Ausschnitt gezeigt, d.h. es wird ausgebldendet.

Im Graph-Editor können auch neue Objekte angelegt werden. Dazu ziehen wir aus der Legende links im Fenster des Graph-Editors den gewünschten Objekttyp mit Drag&Drop auf die Zeichenfläche:

Sind keine Objekttypen in der Legende zu sehen, können diese mit rechtsklick im Feld der
Legende gesucht werden. Anschließt wird der Name des Objektes vergeben.

Es erscheint wieder der Editor, in dem sich die für das Objekt möglichen Relationen, Attribute und Erweiterungen hinzufügen lassen.

### 1.1.4.2 Operationen auf Objekten im Graph-Editor

Der Name kann im Admin-Tool oder im Knowledge-Builder nachträglich geändert werden. Der hierüber angelegte Nutzer besitzt automatische Graph-Administratorrechte. Über das Kontextmenü können mit Rechtsklick auf das Objekt weitere Operationen ausgeführt werden. Größtenteils bietet dieses Kontextmenü dieselben Funktionen wie der Formular-Editor, enthält aber zusätzlich weitere Graph-Editor-spezifische Komponenten.

In diesem Kontext-Menü stehen folgende Graph-Editor-spezifische Funktionen zur Auswahl:

- **Knoten ausblenden**: Hier kann der Knoten ausgeblendet werden.
- **Navigation - Erweiterungen**: Öffnet die Erweiterungen zu einem Objekt.
- **Navigation - Berechnete Relationen**: Öffnet die berechneten Relationen zu einem Objekt.
- **Navigation - Fixieren**: Fixiert die Position eines Knotens im Graph-Editor, so dass er auch bei einem Neuaufbau des Layouts nicht verschoben wird. Die Fixierung kann mit der Option Lösen wieder aufgehoben werden
- **Navigation - Kürzester Pfad**
1.1.4.3 Ansicht

Das Menü „Ansicht“ stellt viele weitere Funktionen für die graphische Darstellung von Objekten und Objekttypen zur Verfügung:

**Voreinstellungen**: Öffnet das Menü für globale Einstellungen (auch zugänglich im globalen Einstellungsfenster - Registerkarte „Persönlich“ - Graph). Hier kann eingestellt werden, ob Attribute, Relationen und Erweiterungen in einem kleinen Mouse-over-Fenster über dem Objekt erscheinen sollen und wie viele Knoten sich im Graph-Editor maximal in einem Schritt einblenden lassen.

![Example Diagram](image)

**Hintergrund ändern**: Hier kann die Farbe des Hintergrunds geändert oder ein eigenes Bild als Hintergrund eingefügt werden.


**Knoten automatisch positionieren**: Führt für neu eingebundene Knoten automatisch die Layoutfunktion aus.

**Beschreibungen fixieren**: Mit dieser Option sind die Namen aller Relationen immer sichtbar, nicht nur beim Roll-over mit der Maus. Alternativ kann gezielt im Kontextmenü einer Relation deren Beschreibung fixiert werden.

**Interne Namen anzeigen**: Blendet an Typknoten die internen Namen der Typen ein.

**Ausgeblendete Kanten wieder darstellen**: Alle Kanten, die per Kontext-Menü ausgeblandet wurden, werden wieder angezeigt.

**Kanten automatisch wieder abblenden**: Die Kanten werden abgeblendet.

Das Fenster des Graph-Editors und das Hauptfenster des Knowledge-Builders stellen noch weitere Menüpunkte zur Verfügung, die bei der Modellierung eine Hilfestellung bieten können.

Links im Fenster des Graph-Editoren befindet sich die Legende der Objekttypen.
Diese Legende zeigt die Objekttypen zu den konkreten Objekten auf der rechten Seite.

Durch Drag&Drop eines Eintrags aus der Legende in die Zeichenfläche können Sie ein neues konkretes Objekt des entsprechenden Typs erzeugen.

Über das Kontextmenü auf den Legendeneinträgen können alle konkreten Objekte dieses Typs aus der Darstellung ausgelandet werden. Hier lassen sich auch Legendeneinträge „festhalten“, und neue Objekttypen in die Legende aufnehmen (unabhängig davon, ob konkrete Objekte von diesem Typ in der Darstellung vertreten sind).

**Max Knoten:** Wenn ein Knoten/Objekt viele Nachbarobjekte hat, ist es oft nicht sinnvoll, alle beim Klick auf den Anfasser gleich einzublenden. Hierfür kann an zwei Stellen eine maximale Anzahl einzublendender Knoten definiert werden.

1. Über das globale Einstellungsfester Registerkarte „Persönlich“ Graph lässt sich die Anzahl bei max. Knoten festlegen, oder
2. im Fenster des Graph-Editors oben rechts ist diese Aktion ebenfalls aufrufbar.

Wird der Anfasser zur Einblendung der Nachbarobjekte angeklickt, erscheint anstelle der Objekte eine Auswahlliste.
1.1.4.4 Lesezeichen und Historie

**Lesezeichen:** Die erstellten Ansichten im Graph-Editor lassen sich als Lesezeichen speichern. Die Objekte werden in der Position gespeichert, wie sie im Graph-Editor platziert wurden.

Wenn ein Lesezeichen angelegt werden soll, muss hierfür zunächst ein Ordner ausgewählt werden. Anschließend kann das Lesezeichen benannt werden.
Lesezeichen sind jedoch keine Datenbackups: Objekte und Beziehungen, die nach dem Speichern eines Lesezeichens gelöscht wurden, sind auch beim Anzeigen des Lesezeichens nicht mehr in der Darstellung vorhanden.


Layout: Mit der Layout-Funktion lassen sich Knoten automatisch positionieren, wenn viele Knoten nicht manuell positioniert werden sollen. Bei Einblenden weiterer Knoten werden diese ebenfalls über die Layout-Funktion automatisch im Graph positioniert.
**Drucken**: Die Druck-Funktion öffnet das Dialogfenster zum Ausdrucken, oder zur Generierung einer PDF-Datei des angezeigten Graphs.

Im Menü *Graph* stehen weitere Funktionen für den Graph-Editor zur Verfügung:

**In Zwischenablage kopieren**: Diese Funktion erzeugt einen Screenshot des aktuellen Graph-Editor-Inhalts. Dieses Bild kann dann beispielsweise in ein Zeichen- oder Bildbearbeitungsprogramm eingefügt werden.

**Für kooperative Arbeit öffnen**: Diese Funktion ermöglicht anderen Benutzern, gemeinsam und gleichzeitig am Graph zu arbeiten. Alle Änderungen und Selektionen eines Benutzers am Graph (Layout, Ein-/Ausblenden von Knoten usw.) werden dann synchron bei allen anderen Benutzern angezeigt.

![Screenshot des Graph-Editors](image)

**1.2 Schemadefinition / Modell**

**1.2.1 Typen definieren**

Im Kapitel 1.2 ist das Prinzip der Typenhierarchie bereits vorgestellt worden. Sollen neue Typen angelegt werden, erfolgt dies immer als Untertyp eines existierenden Typs. Das Anlegen der Untertypen kann entweder über das Kontextmenü Erstellen -> Untertyp

![Typenmenü](image)

oder im Hauptfenster im Reiter „Untertypen“ über das Suchfeld und die Schaltfläche „Neu“ vorgenommen werden:
**Typenhierarchie ändern**

Zum Ändern der Typenhierarchie stehen uns der Baum der Objekttypen im Hauptfenster und der Graph-Editor zur Verfügung.

Im Hierarchie-Baum des Objekt-Editors finden wir im Kontextmenü die Option „Entferne Obertyp xy“. 
Objekttypen mit Eigenschaften ausstatten

Im einfachsten Fall definieren wir Relationen und Attribute bei einem Objekttyp wie z.B. „Band“ oder „Person“ und stellen sie damit für die konkreten Objekte dieses Typs zur Verfügung. (Bspw. Gründungsjahr und -ort bei Bands, Geburtsdatum und Geschlecht bei Personen, Veranstaltungsort und -datum bei Veranstaltungen.)

Hat der Objekttyp, bei dem die Eigenschaften definiert sind weitere Untertypen, so greift hier das Prinzip der Vererbung: Eigenschaften stehen nun auch für die konkreten Objekte der Untertypen zur Verfügung. Beispiel: eine Band erbt als Untertyp einer Organisation die Möglichkeit, Personen als Mitglieder zu haben. Als Untertyp von „Person oder Gruppe“ erbt die Band auch die Möglichkeit, an Veranstaltungen teilzunehmen:
Der Editor für den Objekttyp „Band“ mit direkt dort definierten und geerbten Relationen.
Beim konkreten Objekt stehen die geerbten Eigenschaften ohne Weiteres zur Verfügung, hier wird der Unterschied gar nicht bemerkt.

**Beziehungen definieren**

Damit ist die Relation definiert und kann jetzt zwischen Objekten per Drag&Drop gezogen werden.

**Attribute definieren**

Bei der Definition neuer Attributtypen benötigt i-views neben dem Namen den technischen Datentyp. Folgende technische Datentypen stehen zur Verfügung:

<table>
<thead>
<tr>
<th>Datentyp</th>
<th>Wie sehen die Werte aus?</th>
<th>Beispiel (Musik-Datenbank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribut</td>
<td>abstraktes Attribut, ohne Attribut-Wert</td>
<td>Bauart eines Musik-Instruments (Hollowbody, Fretless usw.)</td>
</tr>
<tr>
<td>Auswahl</td>
<td>frei definierbare Auswahlliste</td>
<td>Musikgruppe noch aktiv?</td>
</tr>
<tr>
<td>Boolesch</td>
<td>»ja« oder »nein«</td>
<td>wav-Datei eines Musiktitels</td>
</tr>
<tr>
<td>Datei</td>
<td>externe Datei eines beliebigen Formats, die als »Blob« in die semantische Graph-</td>
<td>Veröffentlichungsdatum eines Tonträgers</td>
</tr>
<tr>
<td></td>
<td>Datenbank importiert wird</td>
<td>Beginn einer Veranstaltung, bspw. Konzert</td>
</tr>
<tr>
<td>Datum</td>
<td>Datumsangabe tt.mm.jjjj (in der deutschen Spracheinstellung)</td>
<td></td>
</tr>
<tr>
<td>Datum und Uhrzeit</td>
<td>Datums- und Uhrzeitangabe tt.mm.jjjj hh:mm:ss</td>
<td>Ungefähres Eintrittsdatum eines Mitglieds in eine Musikgruppe</td>
</tr>
<tr>
<td>Farbwert</td>
<td>Farbauswahl aus Farbpalette</td>
<td></td>
</tr>
<tr>
<td>Flexible Zeit</td>
<td>Monat, Monat + Tag, Jahreszahl, Uhrzeit, Timestamp</td>
<td></td>
</tr>
<tr>
<td>Fließkommazahl</td>
<td>Zahlenwert mit beliebiger Anzahl von Nachkommastellen</td>
<td>Preis einer Eintrittskarte zu einer Veranstaltung</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Ganzzahl</td>
<td>Zahlenwert ohne Nachkommastellen</td>
<td>Laufzeit eines Musiktitels in Sekunden</td>
</tr>
<tr>
<td>Geographische Position</td>
<td>Geographische Koordinaten im WGS84-Format</td>
<td>Ort einer Veranstaltung</td>
</tr>
<tr>
<td>Gruppe</td>
<td>ohne Attribut-Wert, dient als Träger zu gruppierender Meta-Attribute</td>
<td></td>
</tr>
<tr>
<td>Internet-Verknüpfung</td>
<td>Link auf eine URL</td>
<td>Webseite einer Musikgruppe</td>
</tr>
<tr>
<td>Intervall</td>
<td>Datumsintervall: Intervalle von Zahlen, Zeichenkette, Zeit- oder Datumszeit</td>
<td>Zeitraum zwischen Produktion eines Albums und seiner Veröffentlichung</td>
</tr>
<tr>
<td>Passwort</td>
<td>je Attribut-Instanz und Passwort ein eindeutig geheister Wert (SHA256), der nur zum Validieren des Passworts verwendet wird</td>
<td></td>
</tr>
<tr>
<td>Zeichenkette</td>
<td>beliebige Folge alphanumerischer Zeichen</td>
<td>Rezensionstext zu einem Tonträger</td>
</tr>
<tr>
<td>Zeit</td>
<td>Zeitangabe hh:mm:ss</td>
<td>Dauer einer Veranstaltung</td>
</tr>
</tbody>
</table>

Intention der Nutzung solcher Datentypen ist es, nicht einfach alles als Zeichenkette zu definieren. Technische Datentypen in festgelegter Formatierung bieten später spezielle Möglichkeiten zum abfragen und vergleichen. Beispielweise können Zahlenangaben innerhalb der Strukturabfragen mit größeren oder kleineren Werten verglichen werden, für Geokoordinaten kann eine Umkreissuche definiert werden, u.v.m.

### 1.2.2 Relations- und Attributtypen

Relations- und Attributtypen (kurz Eigenschaftstypen) sind die Typen der konkreten Eigenschaften, die an den Objekten gespeichert sind.
### 1.2.2.1 Neuen Relationstyp anlegen

Über die Schaltfläche im Objekt-Editor „Relation hinzufügen“ startet der Editor zum Anlegen eines neuen Relationstyps:

![Editor für das Anlegen eines neuen Relationstyps](image)

**Name:** Benennungen für Relationstypen lassen sich in i-views frei vergeben, sollten aber nach der Prämisse eines verständlichen Datenmodells gewählt werden. Die folgende Konvention kann dazu hilfreich sein: Der Namen der Beziehung wird so formuliert, dass die Struktur [Name des Quellobjekts] [Relationsname] [Name des Zielobjekts] einen verständlichen Satz ergibt:

[[John Lennon] [ist Mitglied von] [The Beatles]]

Weiterhin ist es hilfreich, wenn die Gegenrichtung (inverse Beziehung) die Wortwahl der Hauptrichtung aufgreift: „hat Mitglied / ist Mitglied von“.

**Domäne und Zieldomäne:** Hier wird bestimmt bei welchen Typen von Objekten die Relation angelegt werden soll: Ein Objekttyp bildet die Quelle der Relation ("Domäne"), ein weiterer Objekttyp das Ziel ("Zieldomäne"). Der Ziel-Objekttyp bildet wiederum den Definitionsbereich der inversen Relation. Beim Anlegen kann der Einfachheit halber an dieser Stelle nur ein Objekttyp eingetragen werden. Im Relationstyp-Editor (siehe unten) lassen sich im Nachhinein noch weitere Objekttypen definieren.

**Übergeordneter Relationstyp:** Genau wie die Objekttypen können auch die Relations- und Attributtypen in einer Hierarchie stehen. Die Hierarchie der Relationstypen ist ein einfaches, aber mächtiges Instrument, um Komplexität im Griff zu behalten.


Der Untertyp impliziert somit den Obertyp. Dieses Prinzip greift bei Relationen genauso wie bei Attributen und Objekten.
1.2.2.2 Neuen Attributtyp anlegen

Über die Schaltfläche im Objekt-Editor „Neues Attribut definieren“ startet der Editor zum Anlegen eines neuen Attributtyps:

Zweistufiger Dialog zum Anlegen eines neuen Attributtyps

Im Fenster links wird das Format des Attributtyps definiert (Datum, Gleitkommazahl, Zeichenkette usw.) Nach Auswählen und Bestätigen des Attributtyps kann dieser im Folgedialog mitAttributnamen weiter spezifiziert werden.

**Obertyp:** Hier wird festgelegt, an welcher Hierarchiestufe der Attributtyp stehen soll.

**Kann mehrfach vorkommen:** Attribute können je nach Attributtyp einfach oder mehrfach vorkommen: eine Person hat nur ein Geburtsdatum, kann aber bspw. mehrere akademische Titel zur gleichen Zeit haben (bspw. Doktor, Professor und Honorarkonsul).

1.2.2.3 Details bearbeiten


Zu diesen beiden Editoren können Sie über die Auflistung der Relationen und Attribute im Reiter „Schema“ des Objekt-Editors gelangen:

Im Folgenden schauen wir uns die Details der Definition von Eigenschaften am Beispiel des Relationstyp-Editors an (die Attributtypdefinition ist eine Untermenge davon):
**Definiert für:** Hier können wir nachträglich ändern, bei welchen Objekttypen die Relation angelegt werden kann. Relationen können zwischen mehreren Objekten definiert werden und damit mehrere Quellen und Ziele haben.

So können wir es z.B. im Schema erlauben, dass sowohl Personen als auch Bands Autoren eines Songs sein können oder einem Ort zugeordnet werden - auch wenn sie keinen gemeinsamen Obertyp haben.

Mit der Schaltfläche „Hinzufügen“ können wir weitere Objekttypen hinzunehmen. Mit „Entfernen“ können wir dem selektierten Objekttyp und allen seinen Objekte die Möglichkeit entziehen diese Relation einzugehen.


**Ziel:** Hier lässt sich nachträglich ändern, zu welchen Typen von Objekten die Relation gezogen werden kann. Um den Ziel-Objekttyp zu ändern, muss zum inversen Relationstyp gewechselt werden: Die Schaltfläche zum Wechseln trägt die Benennung des inversen Relationstyps. Nach anklicken der Schaltfläche erscheint die inverse Relation im Editor und kann auf dieselbe Weise wie die vorherige Relation bearbeitet werden.

**Abstrakt:** Wenn wir eine Relation definieren wollen, die nur zur Gruppierung dient aber selber keine konkreten Eigenschaften ausprägen soll, dann definieren wir sie als "abstrakt"

Beispiel: Wenn die Relation „schreibt Song“ als abstrakt definiert wird, bedeutet dies: wenn wir Songs und ihre Relation zu Künstlern oder Bands anlegen, können wir nur spezifische Angaben machen (wer hat den Text geschrieben, wer die Musik). Die unspezifische Relation „schreibt Song“ kann nicht in den konkreten Daten angelegt werden, sondern nur für Abfra-

Die Option kann nur ausgeschaltet werden wenn die Relation im tatsächlichen Datenbestand nicht mehrfach vorkommt. Bei mehrfachen Vorkommen kann das System nicht automatisch entscheiden, welche der Relationen entfernt werden soll.

Mix-In: Mix-In werden im Kapitel Erweiterungen erklärt.

Hauptsichtung: Zu jeder Beziehung gehört die Gegenrichtung. Im Kern sind beide Richtungen gleichwertig, aber es gibt zwei Stellen, wo es hilfreich ist eine Hauptsichtung zu bestimmen:

- Im Graph-Editor: Hier stellen sich die Relationen, was Pfeilrichtung und Beschriftung angeht, immer in Hauptsichtung dar - unabhängig davon in welche Richtung sie angelegt wurden.
- Bei einseitigen Relationen (ohne Rückrelation)

1.2.3 Modelländerungen

In i-views lassen sich am Modell zur Laufzeit Änderungen durchführen:

- neue Typen einführen
- die Typenarchitektur beliebig ändern (ohne dafür Tabellen anlegen und uns um Primär- und Fremdschlüssel Gedanken zu machen).


Konsistenz ist auch beim Löschen wichtig: Abhängige Elemente müssen immer mit gelöscht werden, sodass keine Reste von Daten gelöschter Elemente in der semantischen Graphdatenbank zurückbleiben.

- Beim Löschen einer Relation wird automatisch die Gegenrichtung mit gelöscht.
Da i-views immer dafür sorgt, dass die Objekte und Eigenschaften dem Schema entsprechen, ist das Löschen eines Objekttyps ggf. eine Operation mit weitreichenden Konsequenzen: Wenn ein Objekttyp gelöscht wird, werden alle seine konkreten Objekte gelöscht - analog bei Beziehungs- und Attributtypen.

Dabei informiert i-views immer über die Konsequenzen einer Operation. Wenn ein Objekt gelöscht werden soll, listet i-views alle Eigenschaften, die damit wegfallen im Bestätigungsdialog der Löschoperation auf:

---

**Folgende Objekte löschen?**

- Stadionkonzert
  - Name: Stadionkonzert
  - Paul McCartney - 1990 im Estadio Maracana
    - Name: Paul McCartney - 1990 im Estadio Maracana (...)
  - Paul McCartney - 1990 im Estadio Maracana hat Teil neh
    - Paul McCartney nimmt Teil an Paul McCartney - 1990

- Untertypen von Stadionkonzert

---

i-views kontrolliert, wo durch die Änderung Objekte, Beziehungen oder Attribute verloren gehen. Der Benutzer wird auf die Konsequenzen der Lösung hingewiesen.

Nicht nur das Löschen, auch das Umwandeln und Ändern der Typenhierarchie kann Konsequenzen nach sich ziehen. Etwa wenn Objekte Eigenschaften haben die bei Typwechsel oder bei Wechsel in der Vererbung nicht mehr dem Schema entsprechen.

---


Es gibt Möglichkeiten zu verhindern, das durch Modelländerungen zu erhaltende Relationen
wegfallen. Soll bspw. ein Objekttyp innerhalb der Typenhierarchie umziehen, muss zuerst das Schema der betroffenen Relation vorher angepasst werden.


Die Typenhierarchie wird von i-views besonders berücksichtigt. Wenn wir aus der Mitte der Hierarchie einen Typ heraus löschen oder eine Ober-/Untertyp-Relation entfernen, dann schließt i-views die entstehende Lücke und hängt die Typen, denen der Obertypen abhanden gekommen ist, wieder in die Typenhierarchie ein - und zwar so dass die seine Eigenschaften möglichst weitgehend bewahrt bleiben.

**Spezielle Funktionen**


Alternativ dazu finden wir im Kontextmenü den Punkt „Überarbeiten“.

**Typ auswählen:** Mit dieser Operation können wir eine Eigenschaft einem neuen Objekt zuweisen.

**Relationsziel neu wählen:** Bei Relationen gilt das nicht nur für die Quelle, sondern auch für das Relationsziel.

konkrete Objekte in Untertypen. Eventuell bestehende Relationen gehen dabei verloren, falls sie nicht zum neuen Schema passen.

**Beziehung umwandeln:** Quelle und Ziel der Relation bleiben bestehen, nur der Beziehungstyp wird umgewandelt.

**Attribut umwandeln:** Quelle/Objekt bleibt erhalten, aber es wird ein anderer Attributtyp zugeordnet:

![Diagramm](image)


Bei Änderungen am Schema ist Grundsätzlich zu bedenken, dass eine Wiederherstellung eines vorherigen Zustandes nur durch Einspielen eines Backups möglich ist. Analog zu relationalen Datenbanken existiert keine „Rückgängig“-Funktion.

**1.2.4 Darstellung von Schemata im Graph-Editor**

Bisher hatten wir es im Graph-Editor hauptsächlich mit der Vernetzung konkreter Objekte zu tun. Sich solche konkreten Beispiele vor Augen zu führen, mit anderen zu diskutieren und sie ggf. zu bearbeiten ist auch die Hauptfunktion des Graph-Editors. Wir können aber mit dem Graph-Editor auch das Schema des semantischen Netzes direkt darstellen, beispielsweise die Typenhierarchie eines Netzes.

Typen von Objekten werden dabei als farbig hinterlegter Knoten, Typen von Relationen werden als gestrichelte Linie dargestellt:
Relationsbegriffe im Graph-Editor

Wenn bisher im Graph-Editor von Relationen die Rede war, so waren damit Relationsobjekte zwischen bestimmten Objekten der semantischen Graph-Datenbank gemeint. Aber auch die allgemeinen Relationstypen (also die Schemata der Relationen) lassen sich im Graph-Editor darstellen. Eine Relation wird im Graph-Editor durch zwei Halbkreise dargestellt, die die beiden Richtungen (Hauptrichtung und inverse Richtung) repräsentieren. Zwischen diesen beiden Knoten besteht also die Beziehung „Inverser Relationstyp“:

Die Darstellung eines Relationsbegriffs und der Hierarchie im Graph-Editor kann analog zum Objekteditor mit allen Ober- und Untertypen dargestellt werden:
Auch Attributtypen lassen sich im Graph-Editor anzeigen - sie werden durch dreieckige Knoten repräsentiert.

Analog zur Objekttyp-Hierarchie kann auch die Hierarchie der Relationen und Attribute im Graph-Editor durch Löschen und Ziehen der Obertyp-Relation verändert werden.
1.2.5 Metamodellierung und fortgeschrittene Konstrukte

1.2.5.1 Erweiterungen

i-views bietet als weiteres Modellierungsmittel die Möglichkeit, Objekte zu erweitern.

Beispielweise tritt eine Person in der Rolle als Gitarrist in einer Band auf, spielt aber in einer anderen Band ein wiederum anderes Instrument. Zusätzlich übt die Person noch die Rolle des Komponisten aus.

Die verschiedenen Rollen einer Person können über eine spezielle Form eines Objekttyps abgebildet werden. Dieser kann keine Objekte enthalten, jedoch Objekte eines anderen Objekttyps (hier zum Beispiel „Person“) erweitern. Hierzu wird bspw. der Objekttyp „Rolle“ in die semantische Graph-Datenbank eingeführt und die verschiedenen Rollen für Personen als Untertyp angelegt: Gitarrist, Komponist, Sänger, Bassist usw. Damit diese „Rollen-Objekttypen“ Objekte erweitern können, wird diese Funktion im Objekttyp-Editor definiert, indem „Typ kann Objekte erweitern“ ausgewählt wird:
Im Graph-Editor werden Erweiterungen durch eine blaue gestrichelte Linie dargestellt:

Mit dieser Erweiterung haben wir mehrere Dinge gleichzeitig erreicht:

- Wir haben Subobjekte für die Personen gebildet (können wir uns auch als Abschnitte vorstellen oder - bei Personen - als Rollen). Diese Subobjekte können einzeln betrachtet
und abgefragt werden. Subobjekte sind unselbstständig: Wenn die Person gelöscht wird, ist auch die Erweiterung „Gitarrist“ mitsamt den Relationen zu den Bands oder Titeln weg.

Wir haben einen mehrstelligen Sachverhalt ausgedrückt. Über separate Relationen zwischen Person, Instrument, Titel/Band können wir das nicht ausdrücken - hier würde die Zuordnung nicht mehr gelingen.


Bei der Definition einer neuen Erweiterung spielen zwei Objekttypen eine Rolle: in unserem Beispiel wollen wir Personen eine Erweiterung geben, das müssen wir ihrem Typ „Person“ mitteilen. Die Erweiterung selber hat auch wieder einen Objekttyp (meist sogar eine ganze Menge von Objekttypen); in unserem Fall „Gitarrist“. Beim Typ „Gitarrist“ (und bei allen anderen mit denen wir Personen erweitern wollen) werden seine konkreten Objekte unselbstständig sein.

**Mix-In**


Wenn die Option Mix-In angewählt ist, wird die Relation dagegen beim Kernobjekt (Person) selbst angelegt. Grund dafür ist, dass Erweiterungen gelegentlich nicht benutzt werden um komplexere Sachverhalte auszudrücken, sondern um ein Objekt polyhierarchisch verschiedenen Typen zuzuordnen. Dieses Objekt erbt auf diese Weise Relationen und Attribute mehrerer Typen.


Wird eine Mix-In Erweiterung gelöscht, verhält sie sich wie eine „normale“ Erweiterung: Es muss mindestens eine Erweiterung vorhanden sein, die die Mix-In-Eigenschaft mit sich bringt. Wenn die letzte dieser Erweiterungen gelöscht wird, wird auch die Relation oder das Attribut beim Kernobjekt gelöscht.

**1.2.5.2 Berechnete Relationen**

Eine spezielle Form der Relation ist die berechnete Relation. Dahinter verbirgt sich die Möglichkeit, mehrere bereits vorhandene und definierte Relationen, die in einer semantis-
Ihre Graph-Datenbank in hintereinandergeschalteter Form vorliegen, durch eine geeignete Relation abzukürzen. Auf diese Weise kann das System in gewissem Rahmen von einem Objekt A der semantischen Graph-Datenbank, das über mehrere Knoten mit einem anderen Objekt B verbunden ist, einen direkten Schluss von A auf B ziehen.

Beispielsweise veröffentlicht eine Musikgruppe einen Tonträger in einem bestimmten Musikstil, ergo kann dieser Musikstil ebenfalls der Musikgruppe zugewiesen werden:

Im Formular-Editor wird der Berechnungspfad über die Relationen „ist Autor von“ und „hat Stil“ definiert.

Einstellungsmöglichkeiten bei der Definition des Berechnungspfades:

- "Transitiv": Die Relation darf beliebig oft auftreten (ein bis unendlich).
- "mit allen Erweiterungen": Die Erweiterungen werden berücksichtigt. (Die Einstellung wird an der Relation ausgewählt, die an der Erweiterung definiert ist. Möchte man einen Berechnungspfad definieren, in dem man von der Erweiterung zurück zum Kern der Erweiterung laufen möchte, so muss die Relation "erweitert Objekt" explizit im Berechnungspfad angegeben werden.)

In den Abfragen kann die berechnete Relation benutzt werden wie jede andere Relation auch.
Anmerkung: In der aktuellen Version von i-views wird wegen der besseren Übersicht bei Strukturabfragen empfohlen, mehrere Knoten und Kanten über Suchbausteine abzufragen.

1.2.5.3  Meta-Eigenschaften


Attribute auf Attributen: Beispielsweise sind für Musik-Alben in der Musik-Datenbank Besprechungen als Textattribute hinterlegt. Soll festhalten wollen, wann die Besprechung hinzugefügt wurde oder wann das letzte Mal geändert, können wir ein Datumsattribut definieren, das den Besprechungs-Attributen angegliedert wird:

![Diagramm mit Attributen](attachment://diagramme.png)

Attribute auf Relationen: Dieses Datumsattribut kann sich auch auf einer Relation zwischen Alben und persönlichen Stimmungen, etwa „Moods“ befinden, wenn den Nutzern die Möglichkeit zum Tagging gegeben wurde:

![Diagramm mit Relationen](attachment://diagramme.png)

Relationen lassen sich auf Attribute und auf Relationen anwenden. Bspw. sollen Nutzer protokolliert werden, die zu bestimmten Zeitpunkten Attribute (etwa eine Besprechung eines Albums), oder eine Relation zwischen einem Album und einem Mood erzeugt oder geändert haben:

![Diagramm mit Schreiboperationen](attachment://diagramme.png)

Diese Beispiele werden mit den Bearbeitungsinformationen bilden eine klar abgegrenzte Meta-Ebene. Eigenschaften auf Eigenschaften sind aber auch für komplexe „Primärinformationen“ verwendbar:
Soll bspw. die Zuordnung von Bands oder Titeln zu den Genres gewichtet werden, kann ein Wert als „Gewicht“ in Form eines Attributs an die Relation vergeben werden.

Ein Attribut einer Relation kann aber auch der Betrag einer Überweisung, oder die Dauer einer Teilnahme oder Mitgliedschaft sein.

Mit Relationen auf Relationen lassen sich ebenso „mehrstellige Sachverhalte“ ausdrücken. Bspw. die Tatsache, dass eine Band bei einem Festival auftritt (das ist eine Relation) und sich dabei einen Gastmusiker dazu holt. Der spielt nicht immer bei der Band und hat somit keine direkte Relation zu ihr. Ebenfalls kann er auch nicht pauschal dem Festival zugewiesen werden, sondern wird der Auftrittsrelation zugewiesen.


Übrigens: Eigenschaften können nicht nur für konkrete Objekte sondern auch für die Typen selbst definiert werden. Ein typisches Beispiel dafür ist eine ausführliche schriftliche Definition bei einem Objekttyp, bspw. „was verstehen wir unter einer Firma?“ Deswegen werden wir beim Anlegen neuer Eigenschaften immer wieder gefragt ob wir sie für konkrete Objekte oder Untertypen anlegen wollen.

1.2.5.4 Mehrsprachigkeit

Die Attribute „Zeichenkette“, „Datei-Attribut“ und „Auswahl“ können mehrsprachig angelegt werden. Beim Zeichenketten-Attribut und bei Dateien können dann mehrere Zeichenketten für ein Attribut eingegeben werden:

Bei Datei-Attributen können analog mehrere Bilder (z.B. mit anderssprachigen Beschriftungen) hochgeladen werden. Bei Auswahl-Attributen werden alle Auswahlmöglichkeiten in der Attributdefinition hinterlegt; hier ist es egal, in welcher Sprache dann die Auswahl für das konkrete Objekt getroffen wird.

Alle anderen Attribute stellen sich in allen Sprachen gleich dar, wie z.B. Boolesche Attribute, Ganzzahlen oder URLs.


In i-views werden für anderssprachige Werte nicht einfach separate Attribute angelegt, sondern es bleibt bei einem Attribut mit Sprachvarianten als separate Layer. Es muss bei Entwicklung einer Anwendung nicht auf das Management der verschiedenen Sprachen geachtet wer-
den, sondern nur auf die gewünschte Sprache bei der jeweiligen Anfrage:

In i-views lassen sich bevorzugte Ersatzsprachen definieren: sollte ein Attributwert, z.B. ein Beschreibungstext in der angefragten Sprache nicht vorliegen, kann der fehlende Text, wenn er in anderen Sprachen vorliegt angezeigt werden. Die Reihenfolge der Ersatzsprachen lässt sich ebenfalls festlegen.

Verwendet werden die Mehrsprachigkeitseinstellungen beispielsweise in der Suche.

1.3 Suchen / Abfragen

Bei Abfragen des semantischen Netzes unterscheiden wir zwischen verschiedenen Teilauflagen: Gelegentlich möchten wir eine Eingabe des Nutzers über ein Suchfeld (Zeichenkette) verarbeiten. Meistens möchten wir bestimmte Pfade im semantischen Netz herausfiltern, manchmal wollen wir dabei Gewichte vergeben. In i-views stehen dazu verschiedene Arten von Suchen zur Verfügung:

- Strukturabfragen
- Direkte Abfragen
- Such-Pipeline-Abfragen

1.3.1 Strukturabfragen

Mit Strukturabfragen können Objekte zusammengesucht werden, die bestimmte Bedingungen erfüllen. Ein einfaches Beispiel für eine Strukturabfrage ist folgende: Es sollen alle Personen gefiltert werden, die ein bestimmtes Instrument beherrschen.

In der Strukturabfrage bilden die Relation „beherrscht Instrument“ und ihr Ziel „Violine“ zwei verschiedene Zeilen und somit auch zwei Suchbedingungen. Die zweite Bedingung, dass das Instrument eine Violine sein muss, kann optional auch weggelassen werden. In der Treffermenge werden dann alle Personen zu finden sein, welche ein beliebiges Instrument spielen:

Oft sollen Bedingungen (hier das Instrument) nicht schon vorher festlegt, oder aber vollständig zugelassen werden. Je nach Situation lässt sich in der Anwendung ein Instrument als Parameter mitgeben:

Die Bedingungen können dabei beliebig komplex werden und das Netz beliebig weit traversieren:

Geringfügig komplexeres Beispiel: Personen oder Bands, die in ihren Songs (genauer gesagt in mindestens einem) ein bestimmtes Thema behandeln. Hier wird der Suche nicht der Name, sondern die ID des Themas als Parameter mitgegeben - typisch für Suchen, die z.B. über einen REST-Service von der Anwendung aus aufgerufen werden [Bild - statt „Name“ „ID“]

Eine Differenzierung gibt (z.B. „schreibt Text“ oder „schreibt Musik“) werden beide Unterrelationen mit in die Suche einbezogen. Das gleiche gilt für die Attributtyp-Hierarchie.

**Interaktion**

Wird eine neue Strukturabfrage angelegt, ist per Default erst einmal der oberste aller Typen eingetragen. Um die Abfrage weiter einzuschränken kann der Namen einfach überschrieben , oder über Klick auf das Icon über „Typ ändern“ ausgewählt werden.

Die Schaltfläche lässt weitere Bedingungen zur Strukturabfrage hinzufügen. Das Löschen von Bedingungen geschieht jeweils am Anfang jeder Zeile, wo die Art der Bedingung aufgeführt ist (Relation, Attribut, Ziel etc.). Beim Klick auf die Schaltfläche erscheint folgendes Menü, das je nach Kontext leicht abweichen kann.

Von allen möglichen Bedingungen wurde bisher auf die allerersten Punkte im Menü fokussiert. Eine vollständige Erläuterung aller Bedingungen und Optionen der Strukturabfragen befinden sich in den nächsten Kapiteln.

**1.3.1.1 Verwendung Strukturabfragen**


Dazu wird die Strukturabfrage in der Regel über ihren Registrierungsschlüssel in einen REST-Service eingebaut. Das Thema, für das sich der Nutzer gerade interessiert, geben wir der Abfrage mit seiner ID als Parameter mit.

Strukturabfragen sind unter anderem deshalb ein zentrales Werkzeug innerhalb von i-views, weil auch die Bedingungen für **Rechte und Trigger** mit Strukturabfragen formuliert werden: Angenommen es wird in einem Musikportal nur Künstlern und Bands erlaubt, Kommentare zu hinterlassen. Im Rechtesystem lässt sich entsprechend formulieren, dass nur Künstler und Bands, die zu einem bestimmten Thema mindestens einen Song geschrieben haben, zu diesem Thema Kommentare hinterlassen dürfen. Strukturabfragen können auch in Exporten benutzt werden, um zu bestimmen, welche Objekte exportiert werden soll.

Alle diese Verwendungen haben eines gemeinsam: wir sind nur an qualitativen, keinen gewichteten Aussagen interessiert. Das ist die Domäne der Strukturabfragen gegenüber den Such-Pipelines.

Nicht zuletzt sind Strukturabfragen auch Hilfsmittel für uns Knowledge-Engineers. Mit ihnen können wir uns einen Überblick über das Netz verschaffen und **Reports** sowie **To-do-Listen** zusammenstellen. Beispielfragen, die mithilfe von Strukturabfragen beantwortet werden können, sind:

- Zu welchen Themen gibt es wie viele Künstler/Bands?
- Müssen bestimmte Themen ausgebaut werden weil sich zu viele Relationen angesammelt haben, oder sollten umgekehrt spärlich besetzte Themen zusammengelegt oder geschlossen werden?

Für diese Verwendung ist es sinnvoll, die Strukturabfragen in **Ordern** organisieren zu können.

**Ausführen**

Durch einen Klick auf die Schaltfläche *Suchen* über dem Fenster für die Trefferliste wird die Strukturabfrage ausgeführt.

Das Suchergebnis kann dann weiter bearbeitet (z.B. mithilfe des Speichersymbols in einen neuen Ordner kopiert) werden.

Den Weg, den die Strukturabfrage genommen hat, kann zur Rückverfolgung der Suchergebnisse im Graph-Editor betrachtet werden. Damit die entsprechende Schaltfläche angeklickt werden kann, müssen die zu betrachtenden Suchergebnisse zunächst selektiert werden.


### 1.3.1.2 Aufbau von Strukturabfragen

Innerhalb von Strukturabfragen lassen sich sehr indirekte Bedingungen formulieren: durch die Struktur der semantischen Graph-Datenbank kann beliebig zwischen den Elementen traversiert werden. Es lassen sich Künstler oder Bands heraussuchen, die Songs zu bestimmten Themen geschrieben haben, welche wir jedoch nicht konkret mit Titel benennen können.
1.3.1.2.1 Mehrere Bedingungen

Entweder können Bedingungsketten beliebig tief sein, oder es lassen sich mehrere parallele Bedingungen formulieren: an einem beliebigen Bedingungselement werden zusätzliche Bedingungen als weiterer Ast angefügt:

**Mehrere Bedingungen: Englische Bands mit Songs über ein bestimmtes Thema**

1.3.1.2.2 Alternative Bedingungen

Im oben genannten Beispiel werden nur Künstler oder Bands gefunden, die sowohl Songs zu einem festgelegten Thema geschaffen haben, als auch aus England stammen. Wenn wir stattdessen alle Künstler und Bands finden wollen, auf die eine der beiden Bedingungen zutrifft, werden sie als Alternative formuliert. Durch anklicken des Symbols der Bedingung in Form der Relation „ist Autor von“ kann dort im Menü eine Alternative gewählt werden:

**Alternative Bedingungen - Die Band muss entweder englisch sein oder Songs zu einem bestimmten Thema haben**
Sind weitere Bedingungen außerhalb der Alternativ-Klammer vorhanden, befinden sich Objekte in der Treffermenge die **eine** der Alternativen, sowie **alle weiteren** Bedingungen erfüllen.

1.3.1.2.3 Transitivität / Wiederholungen

Angenommen, im Netz sind die Bands entweder Städten oder Ländern zugeordnet. Von diesen wiederum ist bekannt, welche Städte in welchen Ländern liegen. Um diesen Sachverhalt in der Suche zu erfassen, ließe sich die Bedingungskette einfach erweitern: Wir können beispielsweise nach Bands suchen, die einer Stadt zugeordnet sind, die wiederum in England liegt. Auf diese Weise werden jedoch die Bands nicht gefunden, welche direkt England zugeordnet sind. Um das zu vermeiden können wir bei der Relation „liegt in“ angeben, dass sie optional ist, also nicht vorliegen muss.

Gleichzeitig können wir mit der Funktion "Wiederholungen" auch Hierarchien berücksichtigen, die mehrere Ebenen tief sind. Beispielsweise ist von der Band ZZ Top bekannt, dass sie aus der Stadt Houston stammt. Diese Stadt liegt wiederum in Texas. Um die Band auch als Ergebnis zu erhalten, wenn nach Bands aus den USA gefragt wird, können wir bei der Relation "liegt in" angeben, dass bis zu n Wiederholungen der Relation nachgegangen werden soll:

1.3.1.2.4 Negativ-Bedingungen

Bedingungen lassen sich ebenso gezielt negieren. Beispielsweise sollen Punk-Bands gesucht werden, die jedoch nicht aus England kommen. Dazu wird die Negativ-Bedingung als eine sogenannte Hilfs suche aufgebaut
Die Hilfssuche liefert Bands auf England - von der Hauptsuche aus kann eine Referenz hergestellt und dabei angegeben werden, dass die Suchergebnisse den Kriterien der Hilfssuche gerade nicht entsprechen dürfen - damit ziehen wir die Ergebnisse der Hilfssuche von denen der Hauptsuche ab und erhalten nur Bands, die nicht aus England kommen.


1.3.1.2.5 Entspricht / Referenz

Die Referenz ermöglicht es, sich innerhalb einer Strukturabfrage auf andere Bedingungen der selben Abfrage zu beziehen:

Hier referenziert die letzte Bedingung (Nr. 7) die allererste, d.h. die Band, die die Coverversion schreibt, muss auch Autor des Originals sein. Ohne Referenz würde sich die Suche folgendermaßen lesen: Bands, die Songs geschrieben haben, die andere Songs covern, die von (beliebigen) Bands geschrieben wurden. Übrigens, die Auflösung ist u.a. die Band „Radiohead“ (haben ihren eigenen Song „Like Spinning Plates“ gecoverd).

1.3.1.2.6 Weitere Optionen im Aufbau der Strukturabfragen

Beispiel aus unserem Musiknetz: Man möchte von beliebigen Bands zu allen ihren Werken gelangen; egal ob es Alben sind oder Songs, die der Band direkt zugeordnet sind oder Songs auf den Alben der Band. Diese Teilabfrage brauchen wir häufiger, u.a. in einer Strukturabfrage, die Bands zu einem bestimmten Mood zurückliefern. Wir beginnen diese Abfrage mit einer Typbedingungen - wir suchen Bands - und binden als Bedingung für diese Bands den vorher definierten Baustein ein:


**Einfache Suche:** Mit der Suchbedingung „einfache Suche“ kann das Ergebnis einer einfachen Suche oder einer Such-Pipeline als Input für eine Strukturabfrage dienen. Mit dem Auswahl-Symbol kann die jeweilige einfache Suche ausgewählt werden. Das Eingabefeld beinhaltet die Suche-Eingabe an die einfache Suche. Durch weitere Bedingungen können z.B. einfache Suchen weiter gefiltert werden.


Damit haben wir alles abgedeckt, was wir im Menü "Anfragestruktur" finden:
1.3.1.3 Die Bedingungen im Detail

Die Typbedingung


Wir können statt konkreter Objekte nur Objekttypen suchen oder beides gleichzeitig, indem im Menü „Schema“ der Haken zusätzlich bei „Untertypen“ gesetzt wird.

So sieht die Bedingung aus, wenn sowohl konkrete Werke als auch Untertypen von Werk (Alben, Songs) gesucht werden.

Ohne Vererbung: Normalerweise wirkt bei allen Typbedingungen der Strukturabfragen automatisch die Vererbung. Wird nach Veranstaltungen gesucht, bei denen Bands einer bes-
timmten Musikrichtung spielen, dann werden automatisch alle Untertypen von Veranstaltung mit in die Suche einbezogen und entsprechend Hallenkonzerte, Clubkonzerte, Festivals etc. mit zurückgeliefert. In der überwiegenden Mehrzahl der Fälle ist das genauso gewünscht. Für die Ausnahmen gibt es die Möglichkeit, die Vererbung auszuschalten und die Suche auf direkte Objekte vom Typ Veranstaltung einzuschränken, d.h. die Objekte der Untertypen auszuschließen.

**Operatoren zum Vergleich von Attributwerten**

Attribute können ebenfalls als Bedingungen für eine Strukturabfrage verwendet werden. Bspw. wenn es nicht reicht Objekte ermitteln zu können, die genau einen vorgegebenen oder als Parameter eingegebenen Wert aufweisen. Etwa sollen Bands, die in der Zeit nach dem Jahre 2005 gegründet wurden oder Songs, die mehr oder weniger 3 Minuten lang sind oder Songs, die im Titel das Wort „Planet“ enthalten. Hierzu werden Vergleichsoperatoren benötigt. Welche Vergleichsoperationen uns i-views anbietet, ist abhängig vom technischen Datentyp des Attributs:

![Vergleichsoperationen](image)

**Vergleichsoperationen für Datumsangaben und Quantitäten**

Der Vergleichsoperator *Exakt Gleich* bildet einen Sonderfall: Der Indexfilter wird abgeschaltet und es lässt sich nach dem Sonderzeichen * suchen, welches normalerweise als Wildcard benutzt wird.

Der Vergleichsoperator *Zwischen* erfordert eine Schreibung des Parameterwerts mit Bindestrich, also z.B. "10.1.2005 - 20.1.2005".

Der Vergleichsoperator *Abstand* erfordert eine Schreibung des Parameterwerts mit Tilde, also z.B. "15.1.2005 ~ 5" - d.h. am 15.1.2005 plus/minus 5 Tage.
Vergleichsoperationen für Zeichenketten

Vergleichswert ergibt sich aus Skript: Attributwertbedingungen lassen sich in Teil suchen entfernen und durch eine Skript-Attributwertbedingung ersetzen. Die Ergebnisse des Skripts werden dann als Vergleichswert für die Attributwertbedingung benutzt, bspw. falls die Vergleichsoperationen für eine spezifische Abfrage nicht ausreichen.

Objekte identifizieren


Hier haben wir bereits die Kombination mit dem Namensattribut kennen gelernt, das kann natürlich ein beliebiges Attribut sein. Im Menüpunkt "Identifizieren" finden wir noch einige andere Möglichkeiten, Startpunkte für die Strukturabfrage zu bestimmen:

Zugriffsrechtparameter: Das Ergebnis der Abfrage vom kann vom Anwendungskontext ab-

**Skript:** Die an dieser Stelle einzugebenden Objekte werden mit dem Ergebnis eines Skriptes bestimmt.

**Wissensnetzelement mit ID:** Man kann ein Objekt auch über seine interne ID festlegen. Diese Bedingung wird in der Regel nur in Verbindung mit Parametern und der Benutzung über die REST-Schnittstelle benutzt.

**Ordner:** Mit der Suchbedingung „liegt in Ordner“ kann der Inhalt einer Sammlung semantischer Objekte als Input in eine Strukturabfrage übergeben werden. Mit dem Auswahlsymbol kann ein Ordner innerhalb der Arbeitsordnerhierarchie ausgewählt werden. Die Objekte der Sammlung werden bezüglich aller weiteren Bedingungen (inkl. Begriffsbedingungen) gefiltert.

### 1.3.2 Einfache Suche / Volltextsuche

Die Verarbeitung der Suchanfragen von Nutzern kann ohne oder mit Interaktion (z.B. mit Type-Ahead-Vorschlagen) vorgenommen werden. Ausgangspunkt ist in jedem Fall die eingegebene Zeichenkette. In der Konfiguration der einfachen Suchen können wir nun festlegen, bei welchen Objekten wir in welchen Attributen nach der Nutzereingabe suchen und wie weit wir dabei von der eingegebenen Zeichenkette abweichen. Dazu ein Beispiel:

Wie müssen wir die Suche gestalten, um auf die Eingabe „white“ die Objekte unten zurückzuleiten? In allen Fällen müssen wir bei der Abfrage konfiguriert haben, dass wir Personen und Bands als Ergebnisse haben wollen. Wie steht es aber mit den Abweichungen von der Nutzereingabe?
• Wann ist die (völlig unbekannte) chinesische Experimentalband namens "WHITE" ein Treffer? Wenn wir angeben, dass Klein-/Großschreibung egal ist
• Wann bekommen wir „Whitesnake“ als Treffer geliefert? Wenn wir die Eingabe als Teilstring verstehen und eine Wildcard anhängen
• Wann „Barry Eugene Carter“? Wenn wir nicht nur den Objektnamen durchsuchen, sondern andere Attribute mit einbeziehen - sein Künstlername ist nämlich „Barry White“

Diese Optionen finden sich folgendermaßen in der Konfiguration der Suche wieder:
Konfiguration der einfachen Suchen mit (1) Angabe, welche Objekttypen durchsucht werden, (2) in welchen Attributen gesucht wird, (3) Groß- und Kleinschreibung und (4) Platzhaltern

1.3.2.1  Einfache Suche - Einstellungen im Detail

**Platzhalter/Wildcard**

Oft ist die Eingabe unvollständig oder wir wollen die Eingabe in längeren Attributfeldern
wiederfinden. Dazu können wir in der einfachen Suche Platzhalter benutzt. Folgende Einstellungen für Platzhalter finden sich in den einfachen Suchen:

- **Platzhalter hinten (Präfix)** findet bei der Eingabe "White" die [White Lies]
- **Platzhalter vorne (Suffix)** findet [Jack White]
- **Platzhalter hinten und vorne (Teilzeichenfolge)** findet [The White Stripes]
- Achtung! Platzhalter vorne ist langsam.

Die Option **Immer Platzhalter** funktioniert so als hätten wir tatsächlich einen Stern vorne oder/und hinten angehängt. Hinter **Automatische Platzhalter** steckt eine Eskalationsstrategie: Bei automatischen Platzhaltern wird erst mit der exakten Benutzereingabe gesucht. Falls diese keine Ergebnisse liefert, so wird mit Platzhalter gesucht, je nachdem welche Platzhalter eingestellt sind. Bei der Option **Präfix oder Teilzeichenfolge** gibt es noch einmal eine Reihefolge: hier wird zuerst nach einem Präfix gesucht (durch Anhängen einer Wildcard) und, falls immer noch nichts gefunden wurde, nach einer Teilzeichenfolge (durch Voranstellen und Anhängen einer Wildcard).

Falls der Suche das Hinzufügen von Platzhaltern erlaubt ist, kann über das Feld **Minimale Anzahl Buchstaben** angegeben werden, wie viele Buchstaben die Sucheingabe mindestens aufweisen muss, damit die Platzhalter tatsächlich angefügt werden. Bei Eingabe von 0 ist diese Bedingung deaktiviert. Das ist vor allem wichtig, wenn wir eine Type-Ahead-Suche bauen.

Mit dem Gewichtungsfaktor für Wildcards läßt sich die Trefferqualität dahingehend anpassen, dass die Anwendung von Platzhaltern niedrigere Qualitäten ergibt. So können wir, wenn wir den Treffern ein Ranking mitgeben wollen, die Unsicherheit, die in den Platzhaltern steckt mit einem niedrigeren Ranking ausdrücken.

Falls die Option „keine Platzhalter“ gewählt ist, so wird die Sucheingabe nicht verändert. Die einzelnen Platzhalter-Einstellungen stehen dann nicht zur Verfügung.

Der Benutzer kann natürlich selbst Platzhalter in der Sucheingabe verwenden, die dann bei der Suche berücksichtigt werden.

**Suchsyntax anwenden:** Wenn kein Haken bei der Option „Suchsyntax anwenden“ gesetzt ist, so wird eine einfachere Form der Analyse der Sucheingabe verwendet, in der die steuernden Zeichenketten „|“ (Oder-Bedingung) , „&“ (Und-Bedingung) und „!” (Negierung) keine Wirkung mehr haben. Um dennoch festlegen zu können, wie die Treffer für die einzelnen Tokens zusammengefasst werden sollen, läßt sich noch der Default-Operator auf „#and“ oder „#or“ schalten. Für alle Verknüpfungsoperatoren gilt, dass sie sich nicht auf Werte eines einzelnen Attributes beziehen, sondern auf die Ergebnisobjekte (je nach dem, ob „Treffer nur auf Attributen“ gesetzt ist). Ein Treffer für Online System liefert somit semantische Objekte, die sowohl für Online als auch für System ein passendes Attribut besitzen (welches nicht notwendigerweise dasselbe ist).

**Filterung:** Einfache Suchen, Volltextsuchen und auch einige der Spezialsuchen können nach Objekttypen gefiltert werden. Im Beispiel des letzten Abschnitts haben wir so dafür gesorgt, dass nur Personen und Bands als Suchergebnis zurückgeliefert werden. Attribute, die nicht zu einer eventuellen Filterung passen, werden im Suchkonfigurationsdialog fett und rot dargestellt. Das könnte in unserem Fall beispielsweise ein Attribut „Rezension“ sein, das nur
für Alben definiert ist.

**Übersetzte Attribute:** Bei übersetzten Attributen können wir entweder eine Übersetzung wählen, oder die Sprache dynamisch bestimmen lassen. Mehrsprachige Attribute suchen dann in der aktiven Sprache oder in allen Sprachen, je nachdem, ob die Option „in allen Sprachen suchen“ gesetzt ist.

**Ergebnismenge:** Eine maximale Ergebnismenge kann durch die Eingabe der maximalen Anzahl im Feld „Ergebnisse“ bestimmt werden. Durch die Checkbox Ergebnismenge begrenzen kann dieser Mechanismus aktiviert oder deaktiviert werden. Durch Eingabe einer Zahl in Ergebnisse wird die Checkbox automatisch aktiviert. Achtung: Wird die Anzahl überschritten, werden keine Ergebnisse angezeigt!

**Serverbasierte Suche:** Es kann generell jede Suche auch als serverbasierte Suche ausgeführt werden. Voraussetzung ist, dass ein dazugehöriger Jobclient läuft. Diese Option kann eingesetzt werden, wenn abzusehen ist, dass sehr viele User Suchanfragen stellen werden. Durch die Auslagerung bestimmter Suchen auf externe Server wird der i-views-Server entlastet.

**1.3.2.2 Mehrwort-Eingaben**

Bei unseren Beispielen zu Abfragen haben die Nutzer bisher immer nur einen Suchbegriff eingegeben. Was aber, wenn der Nutzer z.B. "Abba Reunion News" eingibt und damit vielleicht einen News-Artikel finden möchte, der mit den Themen "Abba" und "Reunion" verschlagwortet ist? Diese Eingabe müssen wir zerlegen, auf den gesamten String wird keines unserer Objekte matchen oder zumindest nicht der gesuchte Artikel:

![Diagramm zur Erläuterung von Mehrwort-Eingaben](image)

Unsere bisherigen Beispiele greifen aber nicht nur wegen Mehrwort-Sucheingaben zu kurz. Wir haben auch oft Suchsituationen, in denen es nicht sinnvoll ist, die Namen oder anderen Zeichenketten aus dem Netz, gegen die wir die Eingabe vergleichen, als Blöcke zu betrachten, z.B. weil wir eine Eingabe in einem längeren Text wiederfinden möchten. Hier sind die Wildcards irgendwann kein adäquates Mittel mehr: Wenn wir auch auf der Seite der Objekte und der durchsuchten Textattribute zerlegen möchten, dann wird besser die Volltextsuche verwendet.
1.3.2.3 Volltextsuche und Indexierung

Wenn wir längere Texte, z.B. Beschreibungsattribute, wortweise betrachten bzw. durchsuchen wollen, empfiehlt sich ein Volltextindex. Wie sieht so etwas aus?

<table>
<thead>
<tr>
<th>Name</th>
<th>Volltext</th>
</tr>
</thead>
<tbody>
<tr>
<td>artikel</td>
<td>Dolph505, Pos. 539644 / Dolph535, Pos. 946715 / Dolph535, Pos. 26526</td>
</tr>
<tr>
<td>abla</td>
<td>Dolph512, Pos. 542065 / Dolph555, Pos. 167166 / Dolph505, Pos. 534915 / Dolph535, Pos. 367773 / Dolph711, Pos. 92634</td>
</tr>
<tr>
<td>akk</td>
<td>Dolph404, Pos. 946774 / Dolph4356, Pos. 219456 / Dolph437, Pos. 90745</td>
</tr>
<tr>
<td>alktz</td>
<td>Dolph305, Pos. 541246</td>
</tr>
<tr>
<td>allnorm</td>
<td>Dolph402, Pos. 230756</td>
</tr>
<tr>
<td>alksten</td>
<td>Dolph505, Pos. 208456 / Dolph631, Pos. 187725</td>
</tr>
</tbody>
</table>

Der Volltextindex verzeichnet alle in einem Bestand an Texten vorkommenden Terme/Worte, so dass i-views schnell und einfach nachschlagen kann in welchen Texten (und an welcher Stelle im Text) ein bestimmtes Wort vorkommt.

„Texte“ sind dabei in i-views i.d.R. nicht separate Dokumente, sondern die Zeichenkettenattribute, die durchsucht werden sollen. Ihre Volltextindexierung ist Voraussetzung dafür, dass diese Attribute in der Suchkonfiguration angeboten werden.

Auch bei der Volltextindexierung geht es um die Abweichungen zwischen der genauen Zeichenfolge im Text und dem, was in den Index eingetragen und dementsprechend wiedergefunden wird. Beispiel: eine Nachricht aus der deutschen Musikszene:

André Rieu gemeinsam mit Bela B. auf unglaublicher Tour! Fun-Punk trifft Kuschel-Klassik?

In diesem Beispiel finden wir einen kleinen Teil der Filter- und Wortabgrenzungs-Operationen wieder, die typischerweise beim Aufbau eines Volltextindex zum Einsatz kommen:

**Wortabgrenzung / Tokenizing:** Im Deutschen werden Satzzeichen wie z.B. das Ausrufezeichen direkt ohne Leerzeichen an das letzte Wort des Satzes gesetzt. In den Volltextindex wollen wir aber den Eintrag (Tour), nicht den Eintrag (Tour!) aufnehmen - nach letzterem wird wohl kaum jemand suchen. Dazu müssen wir beim Aufbau des Volltextindex angeben, dass bestimmte Zeichen nicht zum Wort gehören. Nicht immer ist die Entscheidung so einfach: Bei einer Zeichenfolge wie „Kuschel-Klassik“, die in einem Text vorkommt, müssen wir uns entscheiden ob wir diese als einen Eintrag in den Volltextindex aufnehmen wollen oder als (Kuschel) und (Klassik). Im ersten Fall wird unsere Nachricht nur dann gefunden, wenn genau nach „Kuschel-Klassik“ oder z.B. „*uschel-K*K“ gesucht wird, im zweiten Fall auch bei allen „Klassik“-Suchen.

Was wir trotz des Vorkommens von Satzzeichen wahrscheinlich zusammenhalten, d.h. vom Tokenizing ausnehmen wollen, sind Abkürzungen: wenn AC/DC nur a.d.D. nach Deutschland kommen (auf der Durchreise), dann ist es wahrscheinlich besser, die Abkürzung mit im Index zu haben statt der einzelnen Buchstaben.

**Filtern:** Durch Filteroperationen können wir sowohl Worte bei ihrer Aufnahme in den Volltextindex abwandeln als auch ihre Aufnahme komplett unterdrücken. Bekannt: **Stoppworte**, hier können wir Liste pflegen. Auch **einzeln Buchstaben** (Bela B.) wollen wir wahrscheinlich nicht so im Index stehen haben - die Verwechslungsgefahr ist zu groß. Mit anderen Filtern können wir **Worte auf ihre Grundformen** zurückführen oder **Ersetzungslisten für einzelne Zeichen** definieren (um z.B. Akzente zu eliminieren). Andere Filter wiederum bere-
...inigen den Text um XML-Tags.

All das können wir im Admin-Tool unter „Indexkonfiguration“ einstellen. Diese Konfigurationen können wir dann (im KB oder im Admin-Tool) den Zeichenkettenattributen zuweisen. Die Indexkonfiguration ist so organisiert, dass eine Filterung vor der Wortabgrenzung und eine Filterung nach der Wortabgrenzung stattfinden kann.

Bei der Volltextsuche greift die Wildcard-Automatik der anderen Abfragen nicht, aber der Nutzer kann natürlich seine Eingabe mit Wildcards versehen.

1.3.3 Such-Pipeline

Such-Pipelines ermöglichen es, einzelne Komponenten zu komplexen Abfragen zu kombinieren. Die einzelnen Bausteine führen Operationen aus dabei z.B.:

- Das Netz traversieren und dabei Gewichtung bestimmen
- Strukturabfragen und einfache Abfragen ausführen
- Treffermengen zusammenfassen

Jeder Abfragenschnitt erzeugt eine Ergebnismenge (i.d.R. eine Menge von Objekten). Diese Ergebnismenge kann wiederum als Eingabe für nachfolgende Komponenten in der Pipeline verwendet werden.

Beispiel

Nehmen wir an, Songs und Künstler unseres Musiknetzes sind mit Tags für ihre typische Stimmung charakterisiert. Ausgehend von einer bestimmten Stimmung wollen wir nun die Bands herausfinden, welche diese Stimmung am besten vertreten.

Schritt 0 unserer Such-Pipeline nimmt eine Ausgangsstimmung als Parameter entgegen und belegt damit die Variable namens "mood", in unserem Beispiel nehmen wir die Stimmung "Aggressiv" als Input und schauen uns jetzt an, wie wir mit Hilfe der Suchpipelines für diese Stimmung typische Bands ermitteln: Schritt 1 geht von der Ausgangsstimmung über die Relation ist Stimmung von zu den Songs, denen diese Stimmung zugeordnet ist:
Im zweiten Schritt gehen wir von der Menge an ermittelten Songs zu den entsprechenden Bands über die Relation *hat Autor*:
Jetzt möchten wir noch einen zweiten Weg verfolgen: Vom Ausgangspunkt 'mood' „Aggressiv“ zu Musikrichtungen, die durch Aggressivität charakterisiert werden. Von dieser Menge an relevanten Musikrichtungen ausgehend soll es wieder zu Bands gehen, die zu dieser Musikrichtung zählen. Wir könnten jetzt wieder wie oben Relationen aneinanderhängen, wir können dazu aber auch eine Strukturabfrage in die Suchpipeline einbauen:
Diesem zweiten Weg über die Musikrichtungen geben wir ein etwas niedrigeres Gewicht und fassen zum Schluss die Ergebnisse zusammen:

![Diagramm von Musikgenres wie Aggressiv, Punk, Heavy Metal, Dog faced boy, Eels, Heart in a cage, The Strokes, By Myself, Faint, Crawing, Linkin Park, A Place for my Head, Sex Pistols, Anarchy In The U.K., Elephant, Seven Nation Army.]


**Das Prinzip der Gewichtungen**


1.3.3.1 Zusammenstellung von Such-Pipelines

Die einzelnen Komponenten einer Such-Pipeline, werden im Hauptfenster im Feld *Komponente* in der Reihenfolge ihrer Ausführung dargestellt.
Mit der Schaltfläche **Hinzufügen** können wir eine neue Komponente am Ende der vorhandenen Komponenten anfügen.

Die Gruppierung mit Blöcken dient allein der Übersichtlichkeit, etwa zur Zusammenfassung mehrerer Komponenten zu einem Funktionsbereich der Such-Pipeline.

Mit den Schaltflächen **Nach oben** und **Nach unten** oder mit Drag&Drop kann die Reihenfolge der Schritte geändert werden.

Mit **Entfernen** wird die ausgewählte Komponente entfernt, inklusive aller evtl. enthaltenen Unterkomponenten. Auf der rechten Seite im Hauptfenster wird die Konfiguration für die ausgewählte Komponente angezeigt.

**Konfiguration einer Komponente**


Über die Eingabe- und Ausgabe-Variable können wir auch in späteren Schritten wieder neu bei den ersten Ergebnissen aufsetzen, wie wir schon im letzten Abschnitt gesehen haben.

Als globale Einstellungen für die Suche legen wir die Eingabeparameter fest. Unter dem Namen, den wir hier vergeben, können wir in den einzelnen Schritten unserer Such-Pipeline dann auf diese Eingaben zugreifen. In unserem Beispiel ist der Mood, zu dem wir typische Bands ermitteln, der Eingabeparameter.

Einige Komponenten erlauben eine Abweichung von der Standard-Verarbeitungsreihenfolge:


**Bedingung Parameter gesetzt**: Diese Komponente führt weitere Unterkomponenten nur dann aus, wenn vorgegebene Parameter gesetzt sind. Der Wert ist dabei unerheblich. Mit
Hinzufügen kann man eine neue Unterkomponente hinzufügen.

**KPath-Bedingung:** Mit einer KPath-Bedingung können wir bestimmen, dass die Unterkomponenten nur dann ausgeführt werden, wenn eine in KPath formulierte Bedingung erfüllt ist. Falls die Bedingung nicht erfüllt ist, wird die Eingabe übernommen. KPath ist im Handbuch zu KScript beschrieben.

**Ergebnis:** An beliebigen Stellen der Suche können wir die Suche abbrechen und ein Ergebnis zurückgeben. Diese Komponente ist unter anderem für das Testen der Such-Pipeline nützlich.


**Testumgebung**

Auf die Testumgebung kann an folgenden Stellen zugegriffen werden:

Mit der Testumgebung können wir die Arbeitsweise der Suche analysieren. Der obere Teil enthält die Sucheingabe, der untere Teil das Ergebnis. Die Sucheingabe kann ein Suchtext oder ein Element der semantischen Graph-Datenbank sein, je nachdem welche geforderten und optionalen Eingabeparameter wir global in der Such-Pipeline definiert haben. Wenn wir ein Element der semantischen Graph-Datenbank als Startpunkt eingeben wollen, selektieren wir die entsprechende Parameter-Zeile und fügen, je nach Typ einen Attributwert oder ein Element der semantischen Graph-Datenbank hinzu.
Auf dem Reiter Suchlauf wird ein Protokoll der Suche angezeigt. Dieses besteht im wesentlich aus der Belegung der AusgabevARIABLEN sowie der Dauer zur Ausführung der einzelnen Komponenten.

Verrechnungsmöglichkeiten

Bei einigen Komponenten ist es möglich, mehrere Qualitätswerte zu einem einzigen Qualitätswert zusammenzufassen - z.B. bei „Treffer zusammenfassen“, aber auch beim Traversieren von Relationen (siehe Beispiel oben). Dabei stehen folgende Berechnungsmethoden zur Verfügung:

- Addieren / Multiplizieren
- Arithmetischer Mittelwert / Median
- Minimum / Maximum
- Ranking

die Ähnlichkeit immer weiter erhöhen, ohne dass die 100% überschritten werden. In der Such-Pipeline werden Qualitätswerte immer als Fließkomma-Felder angegeben. Der Wert 1 entspricht dabei einer Qualität von 100%.

1.3.3.2 Die einzelnen Komponenten

Gewichtete Relationen und Attribute

Ausgehend von semantischen Objekten können wir mit diesem Schritt den Graph traver-
sieren und Relationsziele oder Attribute aufsammeln. Dazu müssen wir den Typ der Relation oder des Attributs angeben.

Achtung: Ausgabe sind nur noch die aufgesammelten Ziele, nicht mehr die Ausgangsmenge. Wenn diese angezeigt werden sollen, müssen wir anschließend die Eingangs- und die Aus-
gangstreffer zusammenfassen.

Engangsgang mit einem pauschalen Wert - z.B. 0,5 - belegen, und mit dem Eingangsgewicht verrechnen, beispielsweise multiplizieren. Dann zählen die in diesem Schritt hinzugefügten Untermodos nur halb so viel wie die direkten.

Statt eines pauschalen Gewichts für das Entlanggehen der Relation zu vergeben, können wir den Wert auch aus einer Metaeigenschaft des Basistyps Fließkomma-Felder der ausgewählten Relation entnehmen. Falls dieses Attribut nicht vorhanden ist oder keines konfiguriert wurde, wird der Standardwert verwendet. Der Wert sollte zwischen 0 und 1 liegen. Die Treffererzeu-
gung kann detailierter konfiguriert werden: Bei Relationen kann optional auch für die Rela-
tionsquelle (statt für das Relationsziel) ein neuer Treffer erzeugt werden.

Wenn eine Relation als Eigenschaft ausgewählt wurde und Treffer für Relationsziele erzeugt werden, können wir optional auch die Relation transitiv verfolgen. Bei jedem Schritt ver-
ringert sich der Qualitätszr, bis der angegebene Schwelleindruck unterschritten wird. Falls ein Objekt mehr Relationen hat als bei maximaler Fanout angegeben, werden diese Relationen nicht verfolgt. Je höher der Dämpfungsfaktor ist, desto stärker wird der Qualitätszr ver-
ringert.

Strukturabfrage

Mit Strukturabfrage-Komponenten können wir entweder semantische Objekte suchen / von einer bestehende Menge zu anderen Objekten gehen (wie mit der gewichteten Relation) oder einer Menge filtern.

Zur Filterung geben wir dagegen als Eingabe eine Menge von Objekten an. In der Ausgabe sind alle Objekte enthalten, auf die die Suchbedingung zutrifft. Objekte, die nicht zur Suchbedingung passen, können optional in einer weiteren Variable (Rest) gespeichert werden.

Wir können die Strukturabfrage entweder direkt in der Komponente ad hoc definieren oder eine bestehende Strukturabfrage verwenden.

Achtung: Wenn eine bestehende Suche ausgewählt wurde, wird keine Kopie angelegt, Änderungen, die wir an für Zwecke der Such-Pipeline an der Strukturabfrage vornehmen, ändern sie auch für alle anderen Verwendungen.

**Abfrage**


Über das Einbinden von Such-Pipelines in anderen Such-Pipelines können wir häufiger auftretende Teilschritte ausfaktorisieren. Anderen Such-Pipelines können mehrere Parameter übergeben werden und ganze Treffermengen übergeben. Mit eingebundenen Such-Pipelines können wir auch mehrere Parameter austauschen, d.h. wir können in der eingebundenen Suche auf jede Teilschrittausgabe der umgebenden Suche zugreifen und umgekehrt. Wenn wir auf "ausgewählte Parameter gehen, können wir diese auch umbenennen, falls wir z.B. eine Treffermenge aus der eingebundenen Suche nutzen wollen, aber den Namen schon verwendet haben. Oder wir können, um solche Konflikte zu vermeiden, nur einen Teil der Parameter aus der eingebundenen Suche übernehmen.
**Treffer zusammenfassen**

Mit dieser Komponente können wir mehrere Treffermengen aus unterschiedlichen vorangegangenen Schritten zusammenfassen. Folgende Methoden zur Zusammenfassung stehen zur Verfügung:

**Vereinigungsmenge**: Alle Treffer, die in mindestens einer der Mengen vorkommen, werden als Ergebnis ausgegeben.

**Schnittmenge**: Nur Treffer, die in allen Mengen vorkommen, werden als Ergebnis ausgegeben.

Bei Vereinigungsmengen und Schnittmengen werden kommt es vor, dass ein semantisches Objekt in mehreren Treffermengen vorkommt und zu einem Gesamttreffer mit neuer Trefferqualität verrechnet werden muss. Hier stehen wieder die erwähnten Verrechnungsmethoden zur Verfügung.

**Differenz**: Eine der Treffermengen muss als Ausgangsmenge definiert werden. Von dieser werden die anderen Mengen abgezogen.

**Symmetrische Differenz**: Die Ergebnismenge besteht aus den Objekten, die in genau einer Teilmenge enthalten sind (= alles außer dem Schnitt, bei zwei Mengen).

Es können drei unterschiedliche Arten von Gesamttreffern erzeugt werden. Die Auswahl ist insbesondere dann relevant, wenn die Teiltreffer zusätzliche Informationen tragen.

- Einheitliche Treffer erzeugen, ursprüngliche Treffer als Ursache merken: Es werden neue Treffer erzeugt, die den ursprünglichen Treffer als Ursache enthalten.
- Ursprüngliche Treffer erweitern: Der ursprüngliche Treffer wird kopiert und erhält einen neuen Qualitätswert. Falls mehrere Treffer für dasselbe semantische Objekt vorliegen, wird ein beliebiger Treffer gewählt.
- Einheitliche Treffer erzeugen: Es wird ein neuer Treffer erzeugt. Die Eigenschaften des ursprünglichen Treffers gehen verloren.

**Teiltreffer zusammenfassen**


**Skript**

Eine Such-Pipeline kann ein Skript (JavaScript oder KScript) enthalten. Dieses kann auf die Variablen der Such-Pipeline zugreifen. Außerdem kann ein Skript mehrere Parameter an die Such-Pipeline übergeben. Das Ergebnis des Skripts wird als Ergebnis der Komponente verwendet.

Die JavaScript-API sowie KScript sind in separaten Handbüchern beschrieben.

**Qualität aus Attributwert übernehmen**

Für Treffer können wir den Qualitätswert aus einem Attribut des semantischen Objekts
übernehmen. Falls das Objekt nicht genau ein solches Attribut besitzt, wird der Standardwert verwendet. Der Wert sollte zwischen 0 und 1 liegen.

**Gesamtqualität aus gewichteten Qualitäten berechnen**
Um die Qualität eines Suchtreffers anzupassen, kann es hilfreich sein, aus einzelnen Teilqualitäten einen Gesamtwert zu berechnen. Die Qualitäten müssen dabei als Zahlenwerte vorliegen. Aus diesen Werten wird eine neue Gesamtqualität berechnet.

**Gesamtqualität einer Treffermenge berechnen**
Aus den einzelnen Qualitätswerten einer Treffermenge kann man eine Gesamtqualität berechnen.

**Qualität beschränken**
Treffermengen können wir auf Treffer beschränken, deren Qualitätswert innerhalb vorgegebener Schranken (Minimum und Maximum) liegen. Im Normalfall möchten wir Treffer, die unterhalb einer bestimmten Qualitätsschwelle liegen, ausfiltern.

**Trefferanzahl beschränken**
Falls die Gesamtzahl einer Treffermenge begrenzt werden soll, können wir die Komponente „Trefferanzahl beschränken“ hinzufügen. Mit der Option „Treffer gleicher Qualität nicht zerteilen“ verhindern wir, dass bei mehreren Treffern mit gleicher Qualität eine willkürliche Auswahl erfolgt, um die Gesamtzahl einzuhalten. Wir erhalten dann mehr Treffer als vorgegeben.

Für einige sehr spezielle Fälle können wir die Treffer auch zufällig auswählen lassen, z.B. wenn wir eine große Menge an Treffern gleicher Qualität haben und eine Vorschau generieren wollen.

**Qualität skalieren**
Die Qualitätswerte einer Treffermenge kann skaliert werden. Es wird eine neue Treffermenge mit skalierten Qualitätswerten berechnet. Die Berechnung erfolgt ein zwei Schritten:


**Trefferqualität berechnen**
Mit Hilfe eines KPath-Ausdrucks wird für einen Treffer ein neuer Treffer mit berechneter Qualität erzeugt. Der KPath-Ausdruck wird ausgehend von der Eingabe berechnet.

1.3.4 Die Suche im Knowledge-Builders

Bis auf die Strukturabfragen, die in den Ordnern angelegt und auch dort ausgeführt werden, können alle Suchen in der Kopfzeile des Knowledge-Builders für die interne Nutzung verfügbar gemacht werden.


Entfernen können wir die Suche über die globalen Einstellungen. Hier können wir auch die Reihenfolge der verschiedenen Suchen im Menu ändern.

1.3.5 Spezialfälle

1.3.5.1 Volltextsuche Lucene

Die Volltextsuche lässt sich auch alternativ über den externen Indexer Lucene betreiben. Die Konfiguration der Suche ist dann analog zur normalen Volltextsuche, d.h. es können wiederum Attribute in die Suche konfiguriert werden, die an den Lucene-Index angeschlossen sind; der Suchvorgang erfolgt ebenfalls analog.

1.3.5.2 Suche mit regulären Ausdrücken

Reguläre Ausdrücke sind ein mächtiges Mittel, um je nach Aufgabe Datenbestände nach komplexen Suchausdrücken zu durchforsten.

<table>
<thead>
<tr>
<th>Suche mit regulärem Ausdruck</th>
<th>Treffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>The [CF]all</td>
<td>The Call, The Fall</td>
</tr>
<tr>
<td>Car.</td>
<td>Cars</td>
</tr>
<tr>
<td>Car.*</td>
<td>Cars, Caravans, Carmen etc.</td>
</tr>
<tr>
<td>[^R]oom</td>
<td>Doom, Loom etc. (aber nicht Room)</td>
</tr>
</tbody>
</table>

1.3.5.3 Suche in Ordnern

Die Suche in Ordnern sucht in Ordnernamen und -inhalten:

- Ordnernamen, die auf die Sucheigabe passen
- Ordnernamen, die Objekte enthalten, die auf die Sucheigabe passen
- Expertensuchen, die Elemente enthalten, auf die die Sucheigabe passt
- Skripte, in denen die Sucheigabe auftaucht
- Rechte- und Triggerdefinitionen, die Elemente enthalten, auf die die Sucheigabe passt


1.4 Ordner und Registrierung

Neben den Objekten und ihren Eigenschaften bauen wir in einem typischen Projekt auch diverse andere Elemente: wir definieren z.B. Abfragen und Importe/Exporte oder schreiben Skripte für spezielle Funktionen. Alles, was wir bauen und konfigurieren, können wir in Ordnern organisieren.

Die Ordner werden geteilt mit allen anderen, die am Projekt arbeiten. Wenn wir das nicht möchten, können wir Dinge im Privatordner ablegen, etwa für Testzwecke. Dieser ist nur für den jeweiligen Nutzer sichtbar.


Registrierung

Abfragen, Skripte etc. können sich gegenseitig aufrufen (eine Abfrage kann in eine andere Abfrage oder in ein Skript eingebaut werden, umgekehrt kann ein Skript von einer Suchpipeline aus aufgerufen werden). Zu diesem Zweck gibt es Registrierungsschlüssel, mit denen wir Abfragen, Import-/Export-Abbildungen, Skripte und sogar Sammlungen semantischer Objekte und Strukturordner ausstatten können, damit sie anderen Konfigurationen Funktionalität zur Verfügung stellen. Der Registrierungsschlüssel muss eindeutig sein. Alles, was einen Registrierungsschlüssel hat, wird automatisch in den Ordner „Registrierte Objekte“ aufgenommen bzw. in den seinem Typ entsprechenden Unterordner

Verschieben, Kopieren, Löschen


Wollen wir in einem Schritt die Abfrage löschen, unabhängig davon, in wie vielen Ordnern sie sich findet, so können wir das nur von der Registratur aus.

**Ordnereinstellungen**


### 1.5 Import und Export

Mit den Abbildungen von Datenquellen können wir Daten aus strukturierten Quellen in i-views importieren und Objekte und ihre Eigenschaften in strukturierter Form exportieren. Die Quellen können Excel/CSV-Tabellen, Datenbanken oder XML-Strukturen sein.

Die Funktionen für den Import und Export decken sich größtenteils und sind daher alle in einem Editor verfügbar. Um auf die Funktionen für den Import und Export zugreifen zu können, muss zunächst ein Ordner (z.B. der Arbeitsordner) ausgewählt werden. Dort kann über die Schaltfläche "Neue Abbildung einer Datenquelle" eine Datenquelle ausgewählt werden, aus der importiert, bzw. in die exportiert werden soll.

Alternativ findet man die Schaltfläche auch im Reiter "TECHNIK" unter "Registrierte Objekte“ -> "Abbildungen von Datenquellen“. 

Folgende Schnittstellen und Dateiformate stehen für den Import und Export zur Verfügung:

- CSV/Excel-Datei
• XML-Datei
• MySQL-Schnittstelle
• ODBC-Schnittstelle
• Oracle-Schnittstelle
• PostgreSQL-Schnittstelle
• Für den Austausch von Benutzer-IDs ist eine Standard-LDAP-Schnittstelle implementiert.

Im Folgenden wird anhand einer CSV-Datei beschrieben, wie man einen tabellenorientierten Import/Export anlegt. Da bis auf den XML-Import/Export alle Importe/Exporte tabellenorientiert sind und sich die einzelnen Datenquellen ansonsten nur in ihrer Konfiguration unterscheiden, kann das Beispiel der Abbildung der CSV-Datei auch auf die Abbildungen der anderen Datenbanken und Dateiformate übertragen werden.

1.5.1 Abbildungen von Datenquellen


1.5.1.1 Funktionsprinzip

Nehmen wir als erstes Beispiel eine Tabelle mit Songs: Beim Import dieser Tabelle möchten wir für jede Zeile ein neues konkretes Objekt vom Typ Song anlegen. Aus den Inhalten der Spalten B bis G werden Attribute des Songs bzw. Relationen zu anderen Objekten:

Ausgehend vom Song, bauen wir die Struktur von Attributen, Relationen und Zielobjekten auf, die durch den Import angelegt werden soll (linke Seite). So wird für die Zeile 18 beispielweise ein Objekt vom Typ Song mit folgenden Attributen und Beziehungen angelegt:
Wir können uns aber auch dafür entscheiden, die Angaben aus der Tabelle anders zu verteilen also z.B. Erscheinungsjahr und Interpret dem Album zuordnen und das Genre wiederum dem Interpreten. Eine Zeile bildet immer noch einen Kontext, muss deswegen aber nicht zu genau einem Objekt gehören:

Überall, wo wir in unserem Beispiel neue konkrete Objekte als Relationsziele aufbauen, müssen wir immer mindestens ein Attribut zu diesem Objekt angeben, hier jeweils das Namensattribut, mit dem wir das entsprechende Objekt identifizieren können.

**1.5.1.2 Datenquelle - Auswahl und Optionen**

Nachdem wir, die Schaltfläche "Neue Abbildung einer Datenquelle" ausgewählt haben, öffnet sich ein Dialog, mit dem wir die Art der Datenquelle und den Namen der Abbildung angeben müssen. Haben wir die Datenquelle bereits in der semantischen Graph-Datenbank registriert, können wir diese im unteren Auswahlmenü finden.
Die Struktur unserer Beispiel-Tabelle entspricht komplett den Standard-Einstellungen, sodass wir unter dem Menüpunkt Optionen nichts weiter berücksichtigen müssen. CSV-Dateien können jedoch sehr unterschiedliche Strukturen aufweisen, die mit folgenden Einstellungsmöglichkeiten berücksichtigt werden müssen:


**Zeilentrekker**: In den meisten Fällen reicht die Einstellung "automatisch erkennen", die auch standardmäßig ausgewählt ist. Sollte man jedoch feststellen, dass Zeilenumbrüche nicht richtig erkannt werden, sollte man die entsprechende korrekte Einstellung manuell auswählen. Zur Auswahl stehen CR (carriage return, engl. für Wagenrücklauf), LF (line feed, engl. für Zeilenvorschub), CR-LF und Keine. Der Standard, der den Zeilenumbruch in einer Textdatei kodiert, ist bei Unix, Linux, Android, Mac OS X, AmigaOS, BSD und weiteren LF, bei Windows, DOS, OS/2, CP/M und TOS (Atari) CR-LF und bei Mac OS bis Version 9, Apple II und C64 CR.

1. **Zeile ist Überschrift**: Es kann vorkommen, dass die erste Zeile keine Überschrift enthält, was mit dem Entfernen des standardmäßig gesetzten Häckchens bei "1. Zeile ist Überschrift" dem System mitgeteilt werden muss.

2. **Werte in Zellen sind in Anführungszeichen eingeschlossen**: Wählt man aus, damit die Anführungszeichen nicht mit importiert werden, wenn man das nicht möchte.

**Spalten identifizieren**: Ob die Spalten über ihre Überschrift, die Position oder die Zeichenpa-
sition identifiziert werden, muss angegeben werden, da ansonsten die Tabelle nicht korrekt erfasst werden kann.

*Trennzeichen*: Falls ein anderes Trennzeichen als das standardmäßige Semikolon verwendet wird, muss dies ebenfalls angegeben werden, sofern die Spalte nicht über die Zeichenposition identifiziert wird.

Darüber hinaus gelten folgende Regeln: Falls ein Wert der Tabelle das Trennzeichen oder einen Zeilennumbruch enthält, muss der Wert in doppelte Anführungszeichen gestellt werden. Falls der Wert ein Anführungszeichen enthält, muss dieses verdoppelt ("""") werden.

### 1.5.1.3 Definition von Zielstruktur und Abbildungen

#### 1.5.1.3.1 Die Objektabbildung

Nun fangen wir an die Zielstruktur, die in der semantischen Graph-Datenbank entstehen soll, aufzubauen. In unserem Beispiel beginnen wir mit einer Objektabbildung der Songs. Um ein neues Objekt abzubilden müssen wir den Button "Neue Objektabbildung" bestätigen.

Als nächstes muss der Typ des zu importierenden Objektes angegeben werden. Ist der Haken bei "Mit Objekten aller Untertypen" gesetzt, werden beim Import auch Objekte aus allen Untertypen von "Song" berücksichtigt.

![Objektabbildung der Songs](image1)

#### 1.5.1.3.2 Die Attributabbildung / Identifizieren von Objekten

Nun wollen wir die in der Tabelle enthaltenen Informationen mit der Objektabbildung der Songs verknüpfen. Es sind sowohl Attribute zu den einzelnen Songs, als auch Relationen vertreten. Um zunächst den Titelnamen eines Songs in der Abbildung anzulegen, fügen wir der Objektabbildung von Song ein Attribut hinzu. Ein Klick auf die Schaltfläche "Neue Attributabbildung" öffnet einen Dialog, mit dem die entsprechende Spalte aus der zu importierenden Tabelle ausgewählt werden muss.

![Attributabbildung](image2)
Da dieses Attribut, das erste ist, das wir zu der Objektabbildung von Songs angelegt haben, wird es daraufhin automatisch auf den Namen des Objekts abgebildet, da der Name in der Regel das meistgenutzte Attribut ist.

Das erste zu einem Objekt angelegte Attribut wird zudem automatisch zur Identifizierung des Objektes verwendet.
Ein Objekt muss über mindestens ein Attribut identifiziert werden - sei es über seinen Namen oder seine ID oder eine Kombination aus mehreren Attributen (wie bei Personen aus Vor- und Nachname und dem Geburtsdatum), damit es in der semantischen Graph-Datenbank eindeutig wiedergefunden werden kann, sollte es bereits vorhanden sein. So wird das unge- wollte Anlegen von Dubletten beim Import vermieden.

Im Reiter "Identifizieren" kann das Attribut, das das Objekt identifiziert noch nachträglich geändert oder mehrere Attribute hinzugefügt werden. Zudem kann hier eingestellt werden, ob die Groß- und Kleinschreibung beim Abgleich der Werte beachtet werden soll und ob nach exakt gleichen Werten gesucht werden soll (ohne Indexfilter/Wildcards). Letzteres ist dann relevant, wenn im Index Filter oder Wildcards definiert sind, die z.B. festlegen, dass ein Bindestrich im Index entfallen soll. Der Begriff würde mit Bindestrich nicht gefunden werden, wenn nur über den Index gesucht wird, also müsste in diesem Fall hier der Haken gesetzt werden, sodass nach dem exakt gleichen Wert gesucht wird.

Jetzt können wir dem Objektmapping weitere Attribute hinzufügen, die nicht zur Identifi-
zierung beitragen müssen, z.B. die Dauer eines Songs - dies geschieht wieder über die Schaltfläche "Neue Attributabbildung". (Achtung: Die Objektabbildung "Objekte von Song" muss zunächst wieder ausgewählt werden.) Jetzt wählen wir die Spalte "Dauer" aus der zu importierenden Tabelle aus. Dieses Mal müssen wir das Attribut, auf das die Spalte "Dauer" abgebildet werden soll, manuell auswählen. In dem Feld rechts unten befindet sich die Lister aller im Schema festgelegten möglichen Attribute, die uns für Objekte des Typs "Song" zur Auswahl stehen, darunter auch das Attribut "Dauer".
1.5.1.3.3 Die Relationsabbildung

Als nächstes wollen wir das Album abbilden, auf dem der Song sich befindet. Da Alben eigene konkrete Objekte in der semantischen Graph-Datenbank sind, benötigen wir hierfür die Relation, die den Song und das Album verbindet. Um eine Relation abzubilden wählen wir zunächst das Objekt aus, für das die Relation definiert wird und klicken dann auf die Schaltfläche "Neue Relationsabbildung".

Daraufhin erhalten wir - wie bei den Attributen - eine Liste aller möglichen Relationen, auch die benötigte Relation "Ist enthalten in" ist selbstverständlich dabei.
Nun müssen wir im nächsten Schritt festlegen, wo aus der Tabelle die Zielobjekte herkommen. Für das Ziel wird eine neue Objektabbildung gebraucht, die über die Schaltfläche "Neu" angelegt wird. Ist der Typ des Zielobjektes eindeutig im Schema definiert, wird dieser automatisch übernommen, ansonsten erscheint eine Liste der möglichen Objekttypen.
Bei der neuen Objektabbildung müssen wir anschließend wieder das Attribut auswählen, das das Zielobjekt identifiziert usw. So wird die Zielstruktur des Imports aufgebaut.

1.5.1.3.4 Die Typabbildung

Auch Typen können importiert und exportiert werden. Nehmen wir beispielhaft an, wir wollten die Genres der Songs als Typen importieren.
Um einen neuen Typen abzubilden, wählen wir den Button "Neue Typabbildung".

Daraufhin müssen wir den Obertyp der neu anzulegenden Typen angeben, in unserem Beispiel wäre der Obertyp "Song":

Anschließend müssen wir angeben, aus welcher Spalte der importierten Tabelle der Name unserer neuen Typen entnommen werden soll:

Schließlich müssen wir unter dem Reiter "Import" noch angeben, dass unsere neuen Typen nicht abstrakt sein sollen:
Wenn wir nun die entsprechenden Songs ihren neuen Typen zuordnen wollen, müssen wir die Systemrelation "hat Objekt" verwenden. In älteren Versionen von i-views heißt diese Relation "hat Individuum". Als Ziel wählen wir alle Objekte von Song (inkl. der Untertypen) aus, die sich über das Attribut Name entsprechend der Spalte Songtitel definieren.


Es gibt noch einen Spezialfall. Angenommen, wir haben eine Tabelle, in der in einer Spalte verschiedene Typen vorkommen, dann können wir auch dies in unseren Importeinstellungen abbilden.

<table>
<thead>
<tr>
<th>Person/Band</th>
<th>Herkunft</th>
<th>Typ des Ortes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul McCarrtney</td>
<td>Liverpool</td>
<td>Stadt</td>
</tr>
<tr>
<td>The Beatles</td>
<td>Großbritannien</td>
<td>Land</td>
</tr>
</tbody>
</table>

Dazu wählen wir die Abbildung der Objekte aus, denen wir die Untertypen zuordnen wollen (in diesem Fall "Objekte von Ort") und wählen dann unter dem Reiter "Optionen" den entsprechenden Obertyp aus.
Wichtig ist auch hier wieder nicht zu vergessen, unter dem Reiter "Import" festzulegen, dass der Typ nicht abstrakt sein soll, damit konkrete Objekte angelegt werden können.


### 1.5.1.3.5 Abbildung von Erweiterungen

Auch Erweiterungen können importiert und exportiert werden. Angenommen, wir haben eine Tabelle, die die Rolle eines Bandmitglieds in einer Band zeigt:

<table>
<thead>
<tr>
<th>Person</th>
<th>Band</th>
<th>Rolle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron Wood</td>
<td>Faces</td>
<td>Gitarrist</td>
</tr>
<tr>
<td>Ron Wood</td>
<td>Jeff Beck Group</td>
<td>Bassist</td>
</tr>
<tr>
<td>Ron Wood</td>
<td>Rolling Stones</td>
<td>Gitarrist</td>
</tr>
</tbody>
</table>


Die Abbildung einer Erweiterung fragt - wie eine Objektabbildung - einen zugehörigen Typ ab. Im Schema des Musik-Netzes ist der Typ "Rolle" ein abstrakter Typ. Deswegen muss in der Abbildung definiert werden, dass die Rolle auf Untertypen des Typs " Rolle" abgebildet werden sollen (siehe Kapitel Die Typabbildung).

Die Relation kann - wie auch bei Objekten und Typen - an der Erweiterung (bzw. an den
Untertypen einer Erweiterung) abgebildet werden.

1.5.1.3.6 Die Skriptabbildung

Die Skriptabbildung kann ausschließlich beim Export verwendet werden. Das Skript kann entweder in JavaScript oder KScript geschrieben sein.

Die Skriptabbildung findet beispielsweise dann Verwendung, wenn wir drei Attribute aus der semantischen Graph-Datenbank zu einer ID zusammensetzen wollen. Allerdings kann es sein, dass der Export dann langsamer ist. (Bei einem Import könnte man dies einfacher über eine virtuelle Eigenschaft abbilden. Die Verwendung von virtuellen Eigenschaften wird im Kapitel Tabellenspalten erklärt.)

Der folgende Fall ist ein weiteres Beispiel für die Verwendung eines Skripts bei einem Export. Es zeigt wie mehrere Eigenschaften mit einem Trennzeichen in eine Zelle geschrieben werden können. In diesem Fall wollen wir eine Tabelle erzeugen, die in der ersten Spalte die Songnamen und in der zweiten Spalte alle Stimmungen der Songs mit Komma getrennt aufführt:

Um die zweite Spalte zu erzeugen benötigen wir folgendes Skript:

```javascript
function exportValueOf(element) {
    var stimmung = "";
    var relTargets = $k.Registry.query("stimmungenZuSongs").findElements({songName: element.attributeValue("objektName")});
    if(relTargets && relTargets.length > 0){
        for(var i = 0; i < (relTargets.length-1); i++){
            stimmung += relTargets[i].attributeValue("objektName") + ", ";
        }
```
 Das Skript beinhaltet folgende Strukturabfrage (Registrierungsschlüssel: "stimmungenZuSongs"): 

Über den Ausdruck "findElements" können wir auf einen Parameter (hier "songName") innerhalb der Abfrage zugreifen. "objetName" ist der interne Name des Namensattributs in diesem semantischen Modell.


Das Ergebnis ist eine Liste der Songs mit allen ihren Stimmungen, die durch Komma getrennt in der zweiten Spalte der Tabelle stehen:

<table>
<thead>
<tr>
<th>Songtitel</th>
<th>Stimmung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black Country Rock</td>
<td></td>
</tr>
<tr>
<td>19th Nervous Breakdown</td>
<td></td>
</tr>
<tr>
<td>A Manic Depressive Named Laughing Boy</td>
<td></td>
</tr>
<tr>
<td>A Place for my Head</td>
<td>Aggressiv</td>
</tr>
<tr>
<td>All the Madmen</td>
<td></td>
</tr>
<tr>
<td>Bipolar</td>
<td></td>
</tr>
<tr>
<td>Bleed It Out</td>
<td></td>
</tr>
<tr>
<td>Bleed Like Me</td>
<td></td>
</tr>
<tr>
<td>Breaking the Habit</td>
<td></td>
</tr>
<tr>
<td>By Myself</td>
<td>Aggressiv</td>
</tr>
<tr>
<td>Back To Black</td>
<td>Dramatisch, Bittersüß, Schwungvoll</td>
</tr>
<tr>
<td>China Girl (Bowie)</td>
<td>Melancholisch/Düster, Kalt</td>
</tr>
<tr>
<td>Climbing up the Walls</td>
<td></td>
</tr>
<tr>
<td>Crawling</td>
<td></td>
</tr>
<tr>
<td>Creep</td>
<td>Hymnisch, Elegisch, Dramatisch, Lethargisch, Melancholisch/Düster</td>
</tr>
<tr>
<td>Digging In The Dirt</td>
<td></td>
</tr>
</tbody>
</table>

1.5.1.4 Abbildung von mehreren Werten für einen Objekttyp bei einem Objekt

Wenn für einen Objekttyp bei einem Objekt mehrere Werte angegeben sind (in unserem Beispiel etwa mehrere „Moods“ für jeden Song), dann gibt es drei Möglichkeiten, wie die Tabelle aussehen kann. Für zwei der drei Möglichkeiten muss der Import angepasst werden, was im Folgenden beschrieben ist.

Möglichkeit 1 - Trennzeichengetrennte Werte: Die einzelnen Werte befinden sich in einer Zelle und sind durch ein Trennzeichen (z.B. ein Komma) getrennt.
In diesem Fall gehen wir auf die Abbildung der Datenquelle, wo sich die allgemeinen Einstellungen befinden und dort auf den Reiter "Optionen". Hier finden wir im unteren Bereich die Einstellungsmöglichkeit, Trennzeichnen innerhalb einer Zelle anzugeben. Nun müssen wir nur noch die entsprechende Spalte der zu importierenden Tabelle heraussuchen ("Mood") und in die Spalte "Trennzeichen" das verwendete Trennzeichen (",")) eingeben.

Möglichkeit 2 - Mehrere Spalten: Die einzelnen Werte befinden sich jeweils in einer eigenen Spalte, wobei nicht jedes Feld ausgefüllt sein muss. Es werden so viele Spalten benötigt, wie maximal Moods pro Song vorhanden sind.

In diesem Fall muss die entsprechende Relation so oft angelegt werden, wie Spalten vorhanden sind. In diesem Beispiel müssen demnach die erste Relation auf "Mood1", die zweite Relation auf "Mood2" und die dritte Relation auf "Mood3" abgebildet werden.
Möglichkeit 3 - Mehrere Zeilen: Die einzelnen Werte befinden sich jeweils in einer eigene Zeile. Achtung: Hierbei ist es zwingend nötig, dass die Attribute, die für die Identifizierung des Objektes benötigt werden (in diesem Fall der Titelname), in jeder Zeile auftreten, ansonsten würden die Zeilen als jeweils eigenes Objekt ohne Name gedeutet werden und ein korrekter Import wäre nicht möglich.

<table>
<thead>
<tr>
<th>Titelname</th>
<th>Genre</th>
<th>Mood</th>
<th>Dauer</th>
<th>Jahr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eleanor Rigby</td>
<td>Oldies</td>
<td>Reflective</td>
<td>127</td>
<td>1966</td>
</tr>
<tr>
<td>Eleanor Rigby</td>
<td>Oldies</td>
<td>Dreamy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For No One</td>
<td>Oldies</td>
<td>Acerbic</td>
<td>121</td>
<td>1966</td>
</tr>
<tr>
<td>I'm Only Sleeping</td>
<td>Oldies</td>
<td>Quirky</td>
<td>181</td>
<td>1966</td>
</tr>
<tr>
<td>I'm Only Sleeping</td>
<td></td>
<td>Mellow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yellow Submarine</td>
<td>Oldies</td>
<td>Spacey</td>
<td>160</td>
<td>1966</td>
</tr>
<tr>
<td>Yellow Submarine</td>
<td></td>
<td>Trippy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yellow Submarine</td>
<td></td>
<td>Playful</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In diesem Fall sind keine besonderen Import-Einstellungen nötig, da das System über das identifizierende Attribut das Objekt erkennt und die Relationen korrekt zieht.

1.5.1.5 Einstellungen des Import-Verhaltens


Beispiel: Wenn in unserem Netz bereits ein Song mit dem Namen „Eleanor Rigby“ existiert, dann ist das Namensattribut (abgebildet auf die Spalte „Titelname“ unserer Importtabelle) ein existierendes Attribut und folglich der Song ein existierender Song, solange der Song nur
über das Namensattribut identifiziert wird.

Mit den Einstellungen für das Importverhalten können wir steuern, wie der Import auf bereits vorhandene und neue Wissensnetzelemente reagieren soll. Folgende Tabelle zeigt eine Kurzbeschreibung der einzelnen Einstellungen, während die Unterkapitel dieses Kapitels ausführliche und anschauliche Beschreibungen beinhalten.

<table>
<thead>
<tr>
<th>Einstellung</th>
<th>Kurzbeschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aktualisieren</td>
<td>Vorhandene Elemente werden überschrieben (aktualisiert), keine neuen Elemente werden angelegt.</td>
</tr>
<tr>
<td>Aktualisieren oder neu anlegen wenn nicht vorhanden</td>
<td>Vorhandene Elemente werden überschrieben, sollten keine vorhanden sein, werden sie neu angelegt.</td>
</tr>
<tr>
<td>Alle mit selbem Wert löschen (nur bei Eigenschaften verfügbar)</td>
<td>Alle Attributwerte, die mit dem importierten Wert übereinstimmen, werden für die jeweils entsprechenden Objekte gelöscht.</td>
</tr>
<tr>
<td>Alle vom selben Typ löschen</td>
<td>Alle Attributwerte des ausgewählten Typs werden für die entsprechenden Objekte gelöscht, unabhängig davon, ob die Werte übereinstimmen oder nicht.</td>
</tr>
<tr>
<td>Löschen</td>
<td>Wird verwendet, um genau das eine Element zu löschen.</td>
</tr>
<tr>
<td>Neu anlegen</td>
<td>Legt eine neue Eigenschaft/Objekt an, ohne zu beachten, ob der Attributwert oder das Objekt bereits vorhanden ist.</td>
</tr>
<tr>
<td>Neu anlegen wenn noch nicht vorhanden (nur bei Attributen verfügbar)</td>
<td>Nur, wenn noch kein Attribut des gewünschten Typs vorhanden ist, wird eines angelegt.</td>
</tr>
<tr>
<td>Nicht importieren</td>
<td>Kein Import.</td>
</tr>
<tr>
<td>Synchronisieren</td>
<td>Um die zu importierenden Inhalte mit den Inhalten der Datenbank zu synchronisieren, werden bei dieser Aktion alle Elemente, die noch nicht vorhanden sind, neu angelegt, alle die sich geändert haben, aktualisiert und alle, die nicht mehr vorhanden sind, gelöscht.</td>
</tr>
</tbody>
</table>


1.5.1.5.1 Aktualisieren

Wird diese Einstellung bei einem Attribut angewendet, sorgt sie dafür, dass der Wert aus der Tabelle den Attributwert genau eines bereits existierenden Attributs überschreibt. Mit dieser Einstellung werden keine neuen Attribute angelegt. Falls das Objekt mehr als einen Attributwert des ausgewählten Typs hat, wird kein Wert importiert.
Verwendet man die Einstellung "Aktualisieren" bei einem identifizierenden Attribut, während man bei dem dazugehörigen Objekt die Einstellung "Aktualisieren oder neu anlegen, wenn nicht vorhanden" verwendet, erscheint die Fehlermeldung "Attribut nicht gefunden", wenn das identifizierende Attribut nicht in i-views vorhanden ist.

Wird "Aktualisieren" bei einem Objekt angewendet, sorgt die Einstellung dafür, dass alle Eigenschaften des Objekts durch den Import hinzugefügt, bzw. geändert werden können. Neue Objekte werden nicht angelegt.

Beispiel: Angenommen wir führen eine Datenbank mit unseren Lieblings-Songs. Nun haben wir eine Liste mit Songs, die neue Informationen, enthalten. Wir wollen diese Informationen in unsere Datenbank bringen, gleichzeitig aber nicht, dass Songs importiert werden, die nicht zu unseren Lieblings-Songs gehören. Hierfür verwenden wir die Einstellung "Aktualisieren".

Der Song "About A Girl" ist bereits im Knowledge-Builder vorhanden.


Wir legen für Objekte von Song fest, dass sie aktualisiert werden sollen. Alle Attribute, Relationen und Relationsziele erhalten die Import-Einstellung "Aktualisieren oder neu anlegen wenn noch nicht vorhanden".
Das Ergebnis: Der Song wurde aktualisiert und hat nun neue Attribute und Relationen erhalten. Bereits vorhandene Eigenschaften wurden aktualisiert (Wertung).

1.5.1.5.2 Aktualisieren oder neu anlegen wenn nicht vorhanden


1.5.1.5.3 Alle mit selben Wert löschen

In solchen Fällen greift die Import-Einstellung "Löschen" nicht, da sie bei Mehrfachvorkommen nicht weiß, welche der Relationen sie löschen soll. Hier muss also "Alle mit selben Wert löschen" verwendet werden.

1.5.1.5.4 Alle vom selben Typ löschen

Diese Import-Einstellung wird verwendet, wenn alle Attribute, Objekte oder Relationen eines Typs gelöscht werden sollen, unabhängig von den vorhandenen Werten. Im Gegensatz dazu berücksichtigen die Einstellungen "Löschen" und "Alle mit selben Wert löschen" die vorhandenen Werte. Es werden nur die Elemente der Objekte gelöscht, die in der Import-Tabelle vorkommen.

Beispiel: Wir haben eine Import-Tabelle mit Songs und der Dauer der Songs. Wir sehen, dass sich die Dauer in vielen Fällen unterscheidet und beschließen, die Dauer für diese Songs zu Löschen, damit wir keinesfalls falsche Angaben haben.

<table>
<thead>
<tr>
<th>Song</th>
<th>Dauer</th>
</tr>
</thead>
<tbody>
<tr>
<td>19th Nervous Breakdown</td>
<td>120</td>
</tr>
<tr>
<td>A Manic Depressive Nan</td>
<td>306</td>
</tr>
<tr>
<td>A Place for my Head</td>
<td>239</td>
</tr>
<tr>
<td>About A Girl</td>
<td>168</td>
</tr>
</tbody>
</table>

Die Dauer in der Import-Tabelle unterscheidet sich bei den meisten Songs...

<table>
<thead>
<tr>
<th>Name</th>
<th>Dauer</th>
</tr>
</thead>
<tbody>
<tr>
<td>19th Nervous Breakdown</td>
<td>113</td>
</tr>
<tr>
<td>A Manic Depressive Named Laughing Boy</td>
<td>300</td>
</tr>
<tr>
<td>A Place for my Head</td>
<td>249</td>
</tr>
<tr>
<td>About A Girl</td>
<td>168</td>
</tr>
</tbody>
</table>

... von der Dauer der Songs in der Datenbank.
Beim Attribut "Dauer" verwenden wir die Import-Einstellung "Alle vom selben Typ löschen".

Nach dem Import, sind alle Attributwerte des Attributtyps Dauer für diese 4 Songs gelöscht.

1.5.1.5.5 Löschens

Die Import-Einstellung "Löschen" wird verwendet, um genau das eine Objekt / genau die eine Relation / genau den einen Attributwert zu löschen. Falls keine oder mehrere Objekte / Relationen / Attributwerte mit den zu importierenden Elementen übereinstimmen, erscheint diesbezüglich eine Fehlermeldung und die betroffenen Elemente werden nicht gelöscht.

1.5.1.5.6 Neu anlegen

Diese Import-Einstellung legt eine neue Eigenschaft / ein neues Objekt an, ohne zu beachten, ob der Attributwert oder das Objekt bereits vorhanden ist. Einzige Ausnahme: Sollte eine Eigenschaft nur einmal vorkommen (man beachte die Einstellung "kann mehrfach vorkommen" bei der Attributdefinition), so wird das neue Attribut nicht angelegt und eine Fehlermeldung erscheint, die dies mitteilt.

1.5.1.5.7 Neu anlegen wenn noch nicht vorhanden


Beispiel: Angenommen, wir haben eine Import-Tabelle, die Musiker mit ihren alias-Namen beinhaltet. Einige Musiker haben auch mehrere alias-Namen. Hier können wir die Einstellung "Neu anlegen wenn noch nicht vorhanden" nicht verwenden, da dann alle Musiker mit
mehreren alias-Namen keinen erhalten würden.

1.5.1.5.8 Nicht importieren
Mit der Import-Einstellung "Nicht importieren" können wir sagen, dass ein Objekt oder eine Eigenschaft nicht importiert werden soll. Dies ist dann nützlich, wenn wir bereits eine Abbildung definiert haben und diese wiederverwenden wollen, jedoch bestimmte Objekte und Eigenschaften nicht noch einmal importieren wollen.

1.5.1.5.9 Synchronisieren
Die Import-Einstellung "Synchronisieren" ist mit Vorsicht zu genießen, denn sie betrifft als einzige Import-Einstellung nicht nur die Objekte und Eigenschaften in i-views, die in ihren Werten mit denen der Import-Tabelle übereinstimmen, sondern darüber hinaus alle Elemente des gleichen Typs in i-views. Wenn man eine Import-Tabelle mit i-views synchronisiert, bedeutet das prinzipiell, dass das Ergebnis in i-views nach dem Import exakt so aussehen soll wie in der Tabelle.

Wenn Objekte eines Typs synchronisiert werden, werden alle Objekte dieses Typs gelöscht, die nicht in der Import-Tabelle vorkommen. Die Objekte, die vorkommen, werden aktualisiert und die Objekte, die nicht in i-views vorkommen, werden neu angelegt.

Beispiel: Wir wollen die Musikmessen in i-views (links) mit einer Tabelle mit den Messen und ihrem Datum (rechts) synchronisieren:

<table>
<thead>
<tr>
<th>Name</th>
<th>Messedatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamJam Europe</td>
<td>26. - 27.09.2015</td>
</tr>
<tr>
<td>chor.com Messe</td>
<td>01. - 04.10.2015</td>
</tr>
<tr>
<td>Musikmesse 2015</td>
<td>15. - 18.04.2015</td>
</tr>
</tbody>
</table>

Für die Objekte des Typs Messe wählen wir die Import-Einstellung "Synchronisieren", für die einzelnen Attribute Name und Messedatum wird die Import-Einstellung "Aktualisieren oder neu anlegen, wenn nicht vorhanden" verwendet:

Das Attribut Name ist das identifizierende Attribut von Messe. Für das Objekt Musikmesse 2015 fehlt der Name in der Import-Tabelle. Importieren wir die Tabelle so, erhalten wir dazu eine Fehlermeldung:

Nach dem Import sehen wir nun, dass durch den Import zwei Objekte entfallen sind, die keine Entsprechung in der Import-Tabelle hatten. Das Datum bei Musikmesse 2016 wurde
aktualisiert:

<table>
<thead>
<tr>
<th>Name</th>
<th>Messedatum</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamJam Europe</td>
<td>26. - 27.09.2015</td>
</tr>
</tbody>
</table>

Wenn Attribute synchronisiert werden gilt folgendes: Wenn ein bestehendes Attribut durch einen Import keinen Wert erhält, wird es für das entsprechende Objekt der Import-Tabelle gelöscht. Wenn das bestehende Attribut einen anderen Wert hat als in der Import-Tabelle, wird es aktualisiert, auch dann, wenn es mehrmals vorkommen darf. Wenn das Attribut noch nicht vorhanden ist, wird es neu angelegt.


1.5.1.6 Tabellenspalten


Will man aus der semantischen Graph-Datenbank exportieren, muss man die Spalten von Hand eingeben. Es können nur Standard-Spalten, nicht jedoch virtuelle Spalten exportiert werden.

Virtuelle Tabellenspalte / Virtuelle Eigenschaft


Auch beim Export können wir virtuelle Eigenschaften definieren.

Den reguläre Ausdruck schreiben wir dabei einfach in die Spaltenüberschrift (in den Namen der Spalte). Dabei werden Teilzeichenketten, die in spitze Klammern <...> eingeschlossen sind, nach den folgenden Regeln ersetzt, wobei n, n1, n2, ... für die Inhalte anderer Tabellenspalten mit der Spaltennummer n stehen.

<table>
<thead>
<tr>
<th>Ausdruck</th>
<th>Beschreibung</th>
<th>Beispiel</th>
<th>Eingabe</th>
<th>Ausgabe</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;np&gt;</td>
<td>Druckausgabe des Inhalts von Spalte n</td>
<td>Treffer: &lt;1p&gt;</td>
<td>1 (integer)</td>
<td>Treffer: 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'keine' (String)</td>
<td>'keine'</td>
</tr>
<tr>
<td><strong>&lt;ns&gt;</strong></td>
<td>Ausgabe der Zeichenkette in Spalte n</td>
<td>Hallo &lt;1s&gt;!</td>
<td>'Peter'</td>
<td>Hallo Peter!</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------</td>
<td>-------------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>&lt;nu&gt;</strong></td>
<td>Ausgabe der Zeichenkette in Spalte n in Großbuchstaben</td>
<td>Hallo &lt;1u&gt;!</td>
<td>'Peter'</td>
<td>Hallo PETER!</td>
</tr>
<tr>
<td><strong>&lt;nl&gt;</strong></td>
<td>Ausgabe der Zeichenkette in Spalte n in Kleinbuchstaben</td>
<td>Hallo &lt;1l&gt;!</td>
<td>'Peter'</td>
<td>Hallo peter!</td>
</tr>
<tr>
<td><strong>&lt;ncstart-stop&gt;</strong></td>
<td>Teilzeichenkette von Position start bis stop aus Spalte n</td>
<td>&lt;1c3-6&gt;</td>
<td>'Spalten'</td>
<td>alte \n ten \n alten</td>
</tr>
<tr>
<td><strong>&lt;nmregex&gt;</strong></td>
<td>Test, ob der Inhalt von Spalte n den regulären Ausdruck regex matcht. Die folgenden Ausdrücke werden nur ausgewertet, wenn der reguläre Ausdruck zutrifft.</td>
<td>&lt;1m0[0-9]&gt;hi</td>
<td>01</td>
<td>hi \n (leer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1m$&gt;test</td>
<td>123</td>
<td>(leer) \n test</td>
</tr>
<tr>
<td><strong>&lt;nxregex&gt;</strong></td>
<td>Test, ob der Inhalt von Spalte n den regulären Ausdruck regex matcht. Die folgenden Ausdrücke werden nur ausgewertet, wenn der reguläre Ausdruck nicht zutrifft.</td>
<td>&lt;1x0[0-9]&gt;hallo</td>
<td>01</td>
<td>(leer) \n hallo</td>
</tr>
<tr>
<td><strong>&lt;nerregex&gt;</strong></td>
<td>Selektiert alle Treffer von regex aus dem Inhalt von Spalte n. Einzeltreffer sind im Ergebnis durch Komma voneinander getrennt.</td>
<td>&lt;1eL+&gt;</td>
<td>HELLO WORLD</td>
<td>LL,L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1e\d\d\d&gt;</td>
<td>02.10.2001</td>
<td>2001</td>
</tr>
<tr>
<td><strong>&lt;nrregex&gt;</strong></td>
<td>Entfernt alle Treffer von regex aus dem Inhalt von Spalte n</td>
<td>&lt;1rL&gt;</td>
<td>HELLO WORLD</td>
<td>HEO WORD</td>
</tr>
<tr>
<td><strong>&lt;ngregex&gt;</strong></td>
<td>Überträgt den Inhalt aller Gruppen des regulären Ausdrucks</td>
<td>&lt;1g+(\d+)&gt;-</td>
<td>+42-13</td>
<td>42</td>
</tr>
</tbody>
</table>
Tabelle: Spaltendarstellung

<table>
<thead>
<tr>
<th>Ausdruck</th>
<th>Beschreibung</th>
<th>Beispiele</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;nFormat&gt;</code></td>
<td>Formatiert Zahlen, Datums- und Zeitangaben aus Spalten gemäß der Formatangabe 'format'</td>
<td><code>&lt;1f#,0.00&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;1fd/m/y&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;1fdd/mmm&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabelle: Spaltenbezeichner

<table>
<thead>
<tr>
<th>Bezeichner</th>
<th>Ein-/Ausgabe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Company</td>
</tr>
</tbody>
</table>

Example 1: Usage of regular expressions (reference by column number)


Wir können die virtuelle Eigenschaft auch nutzen, um einen Usernamen herzustellen, der aus den ersten 4 Buchstaben des Vornamens und des Nachnamens zusammengesetzt ist. Heißt die Person Maximilian Mustermann und wir definieren die virtuelle Spalte mit dem entsprechenden Ausdruck `<1c1-4><2c1-4>`, erhalten wir das Ergebnis "MaxiMust".

Die virtuelle Eigenschaft kann auch dazu genutzt werden, einem User beim Import ein initiales Passwort anzulegen. Der Ausdruck könnte Pass4<2s> lauten. Das daraus resultierende Passwort für Maximilian Mustermann wäre "Pass4Mustermann".

Ein etwas umfangreicheres Beispiel zeigt, wie die virtuelle Eigenschaft dazu genutzt werden kann, Objekten die korrekten direkte Obergruppe zuzuordnen:
Die drei rechten Spalten sind virtuelle Spalten.

$1mUG$: In die erste der virtuellen Spalten wird die Nummer der Obergruppe des Objekts nur geschrieben, wenn der Begriff "UG" (für Untergruppe) in der ersten Spalte für das Objekt vorkommt.

$2c1-3>000$: Die Nummer die in die Spalte geschrieben werden soll, setzt sich aus den ersten drei Zeichen der zweiten Spalte zusammen und drei Nullen.

$1m>$: Nur wenn die erste Spalte für das Objekt leer ist, also keinen Wert enthält, wird die Nummer der Obergruppe des Objekts in die Spalte geschrieben.

$2c1-4>00$: Die Nummer, die in die Spalte geschrieben werden soll, setzt sich aus den ersten vier Zeichen der zweiten Spalte zusammen.

*Heimtextil 2016*: Dieser Ausdruck wird für alle Objekte in die Spalte geschrieben.

**Beispiel 2: Verwendung individueller Bezeichner (in Kombination mit regulären Ausdrücken)**

Im folgenden Beispiel werden die Inhalte der Spalte Company mittels virtueller Spalten in Großbuchstaben transformiert: Spalte 5 verwendet eine Referenz per Spalten-Nummer, Spalte 6 verwendet eine Referenz per Spalten-Bezeichner.
Ein Klick auf die Vorschau zeigt die transformierten Spalteneinträge:

Folgende Abbildung zeigt die Auswirkung von vertauschten Spalten einer Importtabelle: Während bei der alleinigen Verwendung von regulären Ausdrücken (<1u>) die falsche Spalte transformiert wird, bleibt bei Verwendung eines Bezeichners mit nachgelagertem regulären Ausdruck (<$\text{Comp}$u>) der Inhalt gleich.

1.5.1.7 Konfiguration weiterer tabellenorientierter Datenquellen

Datenbanken
In einem Mapping für eine PostgreSQL-, Oracle- oder ODBC-Schnittstelle müssen die Datenbank, der Benutzer und das Passwort angegeben werden.

Angabe der Datenbank
Die Angabe für die Datenbank setzt sich aus Name des Host, dem Port und dem Namen der Datenbank zusammen. Die Syntax lautet.

<table>
<thead>
<tr>
<th>Datenbanksystem</th>
<th>Angabe der Datenbank</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL</td>
<td>hostname:port_datenbank</td>
</tr>
<tr>
<td>Oracle</td>
<td>//hostname:[port]/[datenbankService]</td>
</tr>
<tr>
<td>ODBC</td>
<td>Name der konfigurierten Datenquelle</td>
</tr>
<tr>
<td>MySQL</td>
<td>Getrennte Konfiguration von Datenbank und Hostname</td>
</tr>
</tbody>
</table>

### Benutzername und Passwort konfigurieren

Benutzername und Passwort werden so angegeben, wie sie in der Datenbank abgelegt sind. Unter dem Punkt Tabelle kann die Tabelle angegeben werden, die importiert werden soll. Für den Import besteht aber auch die Möglichkeit, dass unter dem Punkt "Query" eine Query formuliert wird, die angibt, welche Daten importiert werden sollen.

### Encoding

Handelt es sich um ein PostgreSQL-Mapping, dann kann auf dem Reiter "Encoding" das Encoding angegeben werden.

### Spezielle Anforderungen der Oracle-Schnittstelle

Die Funktion zum direkten Import aus einer Oracle Datenbank setzt voraus, dass auf dem importierenden Rechner bestimmte Laufzeit-Bibliotheken installiert sind.


Nach der Installation ist sicher zu stellen, dass die Bibliothek für den importierenden Client auffindbar ist, entweder indem sie im gleichen Verzeichnis liegt oder für das entsprechende Betriebssystem passende Umgebungsvariablen definiert werden (ist beim OCI dokumentiert).

Je nach Betriebssystem auf dem der Import stattfinden soll, sind weitere Bibliotheken notwendig, die nicht immer installiert sind.

- MS Windows: neben der benötigten "oci.dll" sind noch zwei weitere Bibliotheken notwendig: advapi32.dll (Erweitertes Windows 32 Base-API) und msvcr71.dll (Microsoft C Runtime Library)

Bis auf den XML-Import/Export sind alle Importe/Exporte tabellenorientiert und unterscheiden sich nur in der Konfiguration der Quelle. Für die Beschreibung einer tabellenorientierten Abbildung kann das Beispiel der CSV-Datei herangezogen werden.
1.5.1.8 Abbildung einer XML-Datei

Das Prinzip von XML-Dateien ist, die unterschiedlichen Angaben zu einem Datensatz über Tags (<>- Tags) explizit zu machen (nicht über Tabellenspalten). Dementsprechend sind Tags auch die Grundlage der Abbildung beim Import von XML-Strukturen in i-views.

Ein Beispiel: Nehmen wir an, unsere Liste von Songs liegt als XML-Datei vor:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<Inhalt>
   <Album type="Oldie">
      <Title>Revolver</Title>
      <Song nr="1">
         <Title>Eleanor Rigby</Title>
         <lengthSec>127</lengthSec>
         <Interpret>The Beatles</Interpret>
         <Thema>Mental illness</Thema>
         <Mood>Dreamy</Mood>
         <Mood>Reflective</Mood>
      </Song>
      [...]
   </Album>
   [...]
</Inhalt>
```


**JSON preprocessing** ermöglicht das Umwandeln einer JSON-Datei in XML vor dem eigentlichen Import.

**Mit XSLT transformieren** kann man auswählen, wenn man die XML-Daten aus der ausgewählten XML-Datei noch vor dem Import in andere XML-Daten umwandeln möchte, um beispielsweise die Struktur zu ändern oder einzelne Werte weiter aufzutrennen. Über die Schaltfläche "Bearbeiten" öffnet sich die XML-Datei, in der man die Änderungen mittels XSLT definieren kann.

Ist die Datei ausgewählt, können wir mit dem Button "Aus Datenquelle lesen" die XML-Struktur auslesen lassen, die uns daraufhin in rechten Fenster angezeigt wird.

Wie auch beim CSV-Import muss über den Reiter "Identifizieren" bei der Objektabbildung festgelegt werden, durch welche Attributwerte das Objekt in der semantischen Graph-Datenbank
identifiziert werden soll. Das erste angelegte Attribut für ein Objekt wird auch hier wieder automatisch als identifizierendes Attribut verwendet.

**Möglichkeiten mit XPath-Ausdrücken**

Angenommen wir würden nur Songs aus Alben des Musik-Stils "Oldie" importieren wollen. In unserem XML-Dokument ist die Information über den Musik-Stil direkt im Album-Tag angegeben unter `type="..."`. Im Editor müssen wir also einen XPath-Ausdruck definieren, der den Pfad im XML-Dokument beschreibt, der nur diejenigen Songs enthält, die aus Oldie-Alben stammen. Im rechten unteren Bereich des Editors finden wir ein Feld zum hinzufügen von XPath-Ausdrücken.

Der passende XPath-Ausdruck lautet:

```
//Album[@type="Oldie"]/Song
```

Erklärung im Einzelnen:

<table>
<thead>
<tr>
<th>XPath</th>
<th>Erklärung</th>
</tr>
</thead>
<tbody>
<tr>
<td>//Album</td>
<td>Selektiert alle Alben, wobei es keine Rolle spielt, wo sie sich im Dokument befinden.</td>
</tr>
<tr>
<td>Album[@type=&quot;Oldie&quot;]</td>
<td>Selektiert alle Alben vom Typ &quot;Oldie&quot;</td>
</tr>
<tr>
<td>Album/Song</td>
<td>Selektiert alle Songs, die Subelemente von Alben sind.</td>
</tr>
</tbody>
</table>

Diesen Ausdruck können wir nun verwenden, um eine Entsprechung für die Objektabbildung der Songs zu definieren.

Mit XPath stehen uns außerdem viele weitere nützliche Selektions-Funktionen zur Verfügung. So können wir beispielsweise Elemente über ihre Position im Dokument selektieren, Vergleichsoperatoren einsetzen, sowie alternative Pfade angeben.

**1.5.1.9 Weitere Optionen, Log und Registratur**
1.5.1.9.1 Weitere Optionen beim Import

Im Reiter "Optionen" stehen uns folgende Funktionen zur Auswahl, die unabhängig von der Datenquelle sind:


- **Mehrere Transaktionen verwenden:** Standardeinstellung für einen schnellen Import.

- **Journaling:** Das Journaling sollte verwendet werden, wenn extrem viele Daten mit einem Import gelöscht oder geändert werden. Erst nach jeweils 4.096 Einträgen (die Zahl ist variabel), sollen die Änderungen, bzw. Löschungen für diese Einträge auch am Index vorgenommen werden. Dadurch wird der Import beschleunigt, da nicht für jede einzelne Änderung/Lösung der Index herangezogen werden muss, sondern spätestens nach 4.096 Veränderungen diese gemeinsam in den Index übernommen werden.

- **Metriken aktualisieren:** Die Metriken sollten aktualisiert werden, wenn der Import eine große Auswirkung auf die Menge von Objekttypen oder Eigenschaftstypen hat, wenn also sehr viele Objekte oder Eigenschaften eines Typs der semantischen Graph-Datenbank hinzugefügt werden. Würden die Metriken nicht aktualisiert, könnte dies negative Auswirkungen auf die Performance von Suchen haben, in denen die entsprechenden Typen eine Rolle spielen.

- **Trigger aktiviert:** Ob Trigger beim Import aktiviert sein sollen oder nicht kann hier über das Häkchen bestimmt werden. Falls es gewünscht ist, dass ein Trigger greift und ein anderer nicht, müssen zwei verschiedene Abbildungen mit den entsprechenden semantischen Elementen definiert werden. Informationen zu Triggern stehen im Kapitel Trigger zur Verfügung.

- **Automatische Namensgenerierung für namenlose Objekte:** Ermöglicht die automatische Namensgenerierung für namenlose Objekte.

Liegt eine tabellenorientierte Quelle vor, können wir folgende Einstellungen vornehmen:
Komplette Tabelle einlesen: Obwohl es länger dauern kann, die komplette Tabelle auf einmal einzulesen, macht es Sinn diese Option auszuwählen, wenn Vorwärtsreferenzen vorhanden sind, d.h., wenn Relationen zwischen den zu importierenden Objekten gezogen werden sollen. In diesem Fall müssen namentlich beide Objekte bereits vorhanden sein, was beim zeilenweisen Einlesen der Tabelle nicht der Fall ist. Zudem ist die Fortschrittsanzeige genauer als beim zeilenweisen Einlesen.

Tabelle zeilenweise einlesen: Das zeilenweise Einlesen der Tabelle sollte immer dann verwendet werden, wenn keine Querreferenzen in der Tabelle vorhanden sind, da der Import so schneller geht.


Liegt eine XML-basierte Datenquelle vor, stehen uns folgende Funktionen zur Verfügung:


DTD einlesen: Liest die Dokumenttypdefinition (DTD) ein.

1.5.1.9.2 Log

Die Funktionen im Reiter "Log" ermöglichen Änderungen, die beim Import vorgenommen werden, verfolgen zu können.
Erzeugte Wissensnetzelemente in einen Ordner stellen: Werden neue Objekte, Typen oder Eigenschaften durch den Import erzeugt, können diese in einen Ordner in der semantischen Graph-Datenbank gestellt werden.

Veränderte Wissensnetzelemente in einen Ordner stellen: Alle Eigenschaften oder Objekte, deren Eigenschaften sich durch den Import geändert haben, können in einen Ordner gestellt werden.


Letzter Import / Letzter Export: Hier werden das Datum und die Uhrzeit des zuletzt vorgenommen Imports und des zuletzt vorgenommen Export angezeigt.

1.5.1.9.3 Registratur

Unter dem Reiter "Registratur" findet man die Funktion "Registrierungsschlüssel setzen" mit der man die Datenquelle für andere Importe und Exporte registrieren kann.
Die Funktion "Bestehende Quelle verknüpfen" ermöglicht die Wiederverwendung einer registrierten Quelle. Unter "Verwendungen" kann man einsehen, wo eine Datenquelle noch verwendet wird:

1.5.2 Attributtypen und -formate

Eine häufig auftretende Aufgabe einer Attributabbildung ist der Import bestimmter Daten von konkreten Objekten, beispielsweise von Personen: Telefonnummer, Geburtsdatum etc. Beim Import von Attributen, für die i-views ein bestimmtes Format verwendet (z.B. Datum), müssen die Einträge der zu importierenden Spalte in einer Form vorliegen, die von i-views unterstützt wird. Beispielsweise kann in ein Attributfeld vom Typ Datum keine Zeichenkette der Form abcede... importiert werden; in einem solchen Fall wird für das entsprechende Objekt kein Wert importiert.

Die folgende Tabelle listet die von i-views unterstützten Formate beim Import von Attributen auf. Ein Tabellenwert ja oder 1 wird also beispielsweise korrekt als boolescher Attributwert (bei einem entsprechend definierten Attribut) importiert, ein Wert wie ein oder ähnliches hingegen nicht.
<table>
<thead>
<tr>
<th>Attribut</th>
<th>Unterstützte Werte-Formate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auswahl</td>
<td>Die Abbildung der Import- auf die Attributwerte kann über den Reiter &quot;Wertzuweisung&quot; konfiguriert werden.</td>
</tr>
<tr>
<td>Boolesch</td>
<td>Die Abbildung der Import- auf die Attributwerte kann über den Reiter &quot;Wertzuweisung&quot; konfiguriert werden.</td>
</tr>
<tr>
<td>Datei</td>
<td>Das Importieren von Dateien (z.B. Bildern) ist möglich. Dazu muss entweder der absolute Pfad zur Datei angegeben werden oder die zu importierenden Dateien müssen im gleichen (oder einem anzugebenden Unterverzeichnis) liegen wie die Import-Datei.</td>
</tr>
</tbody>
</table>
| Datum            | - `<day> <monthName> <year>`, z. B. 5 April 1982, 5-APR-1982  
- `<monthName> <day> <year>`, z. B. April 5, 1982  
- `<monthNumber> <day> <year>`, z. B. 4/5/1982  
 Das Trennzeichen zwischen `<day>`, `<monthName>` und `<year>` kann z.B. ein Leerzeichen, ein Komma oder ein Bindestrich sein (es sind aber noch weitere Zeichen möglich). Gültige Monatsnamen sind:  
- 'Januar', 'Februar', 'März', 'April', 'Mai', 'Juni', 'Juli', 'August', 'September', 'Oktober', 'November', 'Dezember'  
- 'Jan', 'Feb', 'Mrz', 'Apr', 'Mai', 'Jun', 'Jul', 'Aug', 'Sep', 'Okt', 'Nov', 'Dez'.  
Achtung: Zweisstellige Jahreszahlen xy werden zu 20xy expandiert (aus 4/5/82 wird also 4/5/2082). Wenn das Mapping auf "Frei definierbares Format" eingestellt ist, können folgende Tokens verwendet werden: YYYY und YY (Jahr), MM und M (Monatsnummer), MMMM (Monatsname), MMM (abgekürzter Monatsname), DD und D (Tag) |
| Datum und Uhrzeit | Für Datum und Uhrzeit siehe die jeweiligen Attribute. Das Datum muss vor der Uhrzeit stehen. Wenn die Uhrzeit weggelassen wird, wird 0:00 verwendet.                                                                                      |
| Farbe            | Import nicht möglich.                                                                                                                                                                                                    |
| Festkommazahl    | Import möglich.                                                                                                                                                                                                         |
| Ganzzahl         | - Ganzzahlen beliebiger Größe  
- Fließkommazahlen (mit Punkt getrennt), z.B. 1.82. Die Zahlen werden beim Import gerundet.                                                                  |
| Internet-Verknüpfung | Jede beliebige URL möglich.                                                                                                                                                                                               |
Boolesche Attribute und Auswahlattribute

Auswahl- oder boolesche Attribute können nur Werte aus einer vorgegebenen Menge annehmen; bei Auswahlattributen ist dies eine vorgegebene Liste, bei booleschen Attributen das Wertepaar ja/nein in Form eines Klickfelds. Beim Import dieser Attribute kann angegeben werden, wie die Werte aus der Import-Tabelle in Attributwerte der semantischen Graph-Datenbank übersetzt werden. Zum einen können die Werte so, wie sie in der Tabelle stehen, übernommen werden; entsprechen sie keiner der in der semantischen Graph-Datenbank definierten möglichen Werte des Attributs, werden sie nicht importiert. Zum anderen können Wertzuweisungen zwischen Tabellenwerten und Attributwerten, die dann importiert werden, festgelegt werden.

1.5.3 Konfiguration des Exports

Der Export von Daten aus einer semantischen Graph-Datenbank in eine Tabelle wird in demselben Editor wie der Import und ganz analog vorbereitet:

1. Im einem Tabellen-Mapping-Ordner im Hauptfenster wird ein neues Mapping angelegt.
2. Im Tabellen-Mapping-Editor wird die zu erzeugende Datei angegeben.

Der Unterschied zum Import liegt darin, dass die Spalten jetzt nicht aus der Tabelle eingeleget werden, sondern im Tabellen-Mapping-Editor angelegt werden müssen. Da der Import- und der Export-Editor derselbe sind, muss man beim Anlegen einer neuen Spalte zunächst auswählen, ob es sich um eine Standard-Spalte oder eine Virtuelle Eigenschaft handelt. Virtuelle Eigenschaften sind bei einem Export jedoch nicht verwendbar.

Export von Strukturabfragen

Es besteht die Möglichkeit das Ergebnis einer Strukturabfrage zu exportieren. Diese Vorgehensweise bietet sich an, wenn nur bestimmte Objekte, die durch eine Suche eingeschränkt werden, exportiert werden sollen. Nehmen wir als Beispiel an, wir wollten alle Bands, die Songs geschrieben haben, die länger als 10 min. dauern, exportieren. Dafür müssen wir zunächst eine Strukturabfrage definieren, die die gewünschten Objekte zusammenstellt.


Export von Sammlungen semantischer Objekte
Auch Sammlungen semantischer Objekte können exportiert werden. Diese brauchen ebenfalls einen Registrierungsschlüssel, den man unter TECHNIK -> Strukturordner setzen kann.

Export der Frame-ID
Die Abbildung der Frame-ID ermöglicht es uns, die in der semantischen Graph-Datenbank für ein Wissensnetzelement vergebene ID, zu exportieren. Hierzu wählen wir einfach das Objekt,
den Typ oder die Eigenschaft aus, für die wir die ID brauchen und wählen dann den Button "Neue Abbildung der Frame-ID":

Wir können außerdem entscheiden, ob wir die ID im Format eines Strings wollen (ID123_456) oder, ob wir sie als 64 Bit Integer ausgegeben haben wollen.

**Export mithilfe von Skripten**

Schließlich steht uns beim Export noch ein weiteres mächtiges Werkzeug zur Verfügung: die Skriptabbildung. Informationen hierzu sind im Kapitel Die Skriptabbildung verfügbar.

**Export-Aktionen bei Datenbankexporten**

Die Abbildung der Eigenschaften eines Objekts wird für einen Export in eine Datenbank genauso vorgenommen wie für einen Import und wie für alle anderen Mappings. Einzig ist für den Export die Export-Aktion zu bestimmen. Diese gibt an, welche Art von Query in der Datenbank ausgeführt werden soll. Es stehen drei Export-Aktionen zur Verfügung:

Folgende Aktionen stehen in dem sich öffnenden Auswahldialog zur Verfügung:

- **Datensätze in Tabelle neu anlegen**: Es werden neue Datensätze in der Datenbanktabelle hinzugefügt. Diese Aktion entspricht einem INSERT.
- **Existierende Datensätze aktualisieren**: Die Datensätze werden über eine ID in der Tabelle identifiziert. Sie werden nur überschrieben, wenn der Wert sich geändert hat. Gibt es keinen passenden Datensatz, dann wird ein neuer hinzugefügt. Diese Aktion entspricht einem UPDATE.
- **Tabelleninhalt beim Export überschreiben**: Alle Datensätze werden erst gelöscht und dann neu geschrieben. Diese Aktion entspricht einem DELETE auf der ganzen Tabelle mit folgendem INSERT.

**1.5.4 RDF-Import und -Export**

RDF ist ein Standardformat für semantische Datenmodelle. Mit dem RDF-Import und -Export können wir die Daten der semantischen Graph-Datenbank mit anderen Anwendungen austauschen, aber auch Daten von einem i-views-Wissensnetz in ein anderes transportieren.

Beim RDF-Export wird das gesamte Wissensnetz in eine RDF-Datei ausgeleitet. Der RDF-Import dagegen ist interaktiv und selektiv. D.h. wir können sowohl auf Schema-Ebene als auch bei den einzelnen Objekten und Eigenschaften angeben, was importiert werden soll und was nicht.
Abgleich der Objekte aus RDF mit den existierenden Objekten im semantischen Netz

Wenn die RDF-Daten aus demselben Schema stammen, wie das Netz in das sie importiert werden - z.B. aus einer Sicherungskopie - dann ordnet der RDF-Import Objekte und Objekttypen automatisch anhand ihrer ID zu. In den Import-Einstellungen können wir wie bei den Tabellen- und XML-Importen nun bestimmen z.B. existierende Objekte durch den Import aktualisiert werden sollen, ob neue angeschlossen werden etc.

Wenn die Daten aus einer anderen Quelle kommen, ist die Default-Einstellung der Import in ein eigenes Subnetz. Wir können aber auch diese externen Informationen in unseren Bestand integrieren - durch manuelle Zuordnungen mit der Abbilden-auf-Funktion im Mapping-Interface.


1.5.5 Gelöschte Individuen aus einem Backup wiederherstellen

Der RDF-Export und -Import eignet sich dazu, gelöschte Individuen aus einem Backup-Netz wieder herzustellen. Dazu wie folgt vorgehen:

1. Das Backup-Netz im Knowledge-Builder öffnen
2. Einen neuen Ordner anlegen und die wiederherzustellenden Individuen dort ablegen. Hierzu in der Listenansicht der zu übernehmenden Individuen mit Rechtsklick das Kontextmenü öffnen und „Inhalt in neuen Ordner kopieren“ wählen, dabei neuen Ordner als Ablageziel wählen
3. Per Kontextmenü auf dem neu erstellten Ordner den RDF-Export öffnen
4. Im Exportdialog einen Dateinamen angeben, die Optionen "URLs verwenden (rdf:about)"
und "Frame-URLs verwenden (krdfframe:)" auswählen und den Export ausführen:

![Exportfenster](image)

Hinweis: Die Option "KRDF verwenden" bewirkt, dass i-views spezifische Inhalte zusätzlich mit übernommen werden, welche mittels RDF-Syntax nicht vollständig abgebildet werden können.

5. Knowledge Builder schließen und das Zielnetz im Knowledge Builder öffnen
6. Im Hauptmenü unter Werkzeuge > RDF > RDF Import den RDF-Import-Dialog öffnen:

![RDF-Import-Fenster](image)

7. Datei auswählen und „Weiter“ drücken:
8. Im Auswahl-Dialog die Option "Änderungen am Schema erlauben" deaktivieren, "Ordner mit importierten Objekten anlegen" aktivieren:

9. Import ausführen

10. Die wiederhergestellten Individuen überprüfen
1.6 Zugriffsrechte und Trigger

In diesem Abschnitt wird die Prüfung von Zugriffsrechten und Trigger behandelt:

- **Zugriffsrechte** regeln, welche Operationen am semantischen Modell bestimmte Nutzergruppen durchführen dürfen. Sie werden in i-views im Rechtessystem definiert. Das Rechtessystem befindet sich im Bereich Technik > Rechte.

- **Trigger** sind automatische Operationen, die bei einem bestimmten Ereignis ausgelöst werden und die zugehörigen Aktionen ausführen. Der Bereich Trigger befindet sich unter Technik > Trigger.

Das Rechtessystem und Trigger sind in einer neu angelegten semantischen Graph-Datenbank initial noch nicht aktiviert. Diese Bereiche müssen erst aktiviert werden, bevor sie eingesetzt werden können.

Bei der Erstellung von Rechten und Triggern ist die grundsätzliche Vorgehensweise identisch: Es werden Filter benötigt, die prüfen ob bestimmte Bedingungen erfüllt sind oder nicht. Sind diese Bedingungen erfüllt, wird beim Rechtessystem ein Zugriffsrecht oder -verbot erteilt sowie bei Triggern ein Log eingetragen oder ein Script ausgeführt. Im Rechtessystem wird die Anordnung der Filter als Rechtebaum und bei Triggern als Triggerbaum bezeichnet.

1.6.1 Die Prüfung von Zugriffsrechten

Mit Rechten regeln wir den Zugriff von Nutzern auf die Daten im semantischen Netz. Die zwei grundsätzlichen Ziele, deren Erreichung mit dem sogenannten Rechtessystem ermöglicht werden, sind:

- **Schutz von sensiblen Daten**: Es werden Nutzern oder Nutzergruppen, nur die Daten angezeigt, die sie auch lesen dürfen. Damit werden Geheimhaltungs- und Vertraulichkeitsbeschränkungen gewährleistet.

- **Arbeitsspezifische Übersicht**: Bestimmte Nutzer benötigen für ihre Arbeit mit dem System häufig nur einen Ausschnitt der Daten des Modells. Mit Hilfe des Rechtessystems ist es möglich ihnen nur die Elemente anzuzeigen, die sie für das Erledigen ihrer Aufgaben brauchen.


**Wie funktioniert das Rechtessystem?**

Zugriffsrechte im System werden immer dann geprüft, wenn durch einen Nutzer eine Operation auf die Daten vorgenommen wird. Die grundsätzlichen Operationen sind:

- **Lesen**: Ein Element soll angezeigt werden.
- **Modifizieren**: Ein Element soll geändert werden.
- **Erzeugen**: Ein neues Element soll erstellt werden.
- **Löschen**: Ein Element soll gelöscht werden.

In Bezug auf die Zugriffssituation werden Aspekte ausgewählt, die als Bedingung für die Erlaubnis oder das Verbot des Zugriffes eingesetzt werden. In Zugriffssituationen werden häufig folgende Aspekte für die Entscheidung herangezogen:

- die Operation (Erzeugen, Lesen, Löschen oder Modifizieren)
- das Element, auf das zugegriffen werden soll
- der aktuelle Nutzer


1.6.1.1 Die Aktivierung des Rechtssystems

In einem neu angelegten Wissensnetz ist das Rechtssystem standardmäßig deaktiviert. Damit es genutzt werden kann, muss es in den Einstellungen des Knowledge-Builders aktiviert werden.

Anleitung für die Aktivierung des Rechtssystems

2. Setzen Sie im Feld Rechtssystem aktiviert einen Haken.
3. Geben Sie im Feld Benutzertyp den Objekttyp an, dessen Objekte die Benutzer des Rechtssystems sind. Das ist i.d.R. der Objekttyp "Person". (Typ darf nicht abstrakt sein.)


Die Verknüpfung ist für die Verwendung des Operationsparameter Benutzer bei Suchfiltern bzw. bei für die Verwendung des Zugriffparameters Benutzer bei Strukturabfragen generell
notwendig, wenn das Rechtesystem bzw. die Suche nicht in einer Anwendung sondern im Knowledge-Builder selbst ausgeführt.

**Anleitung für die Verknüpfung von Knowledge-Builder Nutzern mit Objekten des Personen Typs**

1. Im Knowledge-Builder das Menü *Einstellungen* aufrufen und den Reiter *System* wählen. Dort das Feld *Benutzer* auswählen.
   Die Funktion *Verknüpfung aufheben* führt dazu, dass die Verknüpfung des Knowledge-Builder-Account mit dem Personenobjekt aufgehoben wird.

Beachte: Der aktuell angemeldete Nutzer kann nicht verknüpft werden.

Benutzer mit Administratorenrechten dürfen generell alle Operationen durchführen, unabhängig davon welche Rechte im Rechtesystem definiert wurden. Die Definition als Administrator wird ebenfalls im Menü *Einstellungen auf dem Reiter System im Feld Benutzer* durchgeführt.

**1.6.1.2 Der Rechtebaum**

**Traversierung des Rechtebaumes**


Filter und Entscheider werden in Form von Ordnern ineinander geschachtelt, so dass ein Baumkonstrukt entsteht, das aus verschiedenen Teilbäumen besteht. Ein Ordner kann mehrere Unterordner haben (mehrere Nachfolgefilter auf einer Ebene), wodurch Verzweigungen im Rechtebaum entstehen. Ordner, die auf einer Ebene definiert sind, werden nacheinander abgearbeitet (von oben nach unten).

**Gestaltung des Rechtebaumes**

Bei der Erstellung des Rechtebaumes ist es wichtig die Regeln sinnvoll zu gruppieren, denn wenn eine Entscheidung für eine Zugriffserlaubnis oder ein Zugriffsverbot getroffen wurde, werden keine weiteren Regeln mehr geprüft. Deswegen sollten Ausnahmen vor den globalen Regeln definiert werden.

Die zwei grundlegenden Fälle, die man unterscheiden muss, sind:

- **Negativ-Konfiguration**: Im untersten Teilbaum wird pauschal alles erlaubt, darüber werden Verbote formuliert.
- **Positiv-Konfiguration**: Unten ist pauschal alles verboten, außer dem, was weiter oben erlaubt ist.
Die Reihenfolge der Teilbäume ist also ausschlaggebend bei der Erstellung des Rechtebaumes. Die Reihenfolge der Bedingungen in einem Teilbaum dagegen (ob wir zuerst die Operation und dann die Eigenschaft prüfen oder umgekehrt) ist beliebig.


Beispiel: Rechtebaum

In dieses einfache Beispiel zeigt einen Rechtebaum bestehend aus einem Rechteteilbaum und einen Default-Entscheider, der alles erlaubt:

\begin{itemize}
  \item [Löschen oder Modifizieren]
  \item [Name, Dauer, Erscheinungsdatum]
  \item [Verboten]
  \item [Erlaubt]
\end{itemize}


Dieser einfache Rechtebaum würde in i-views folgender Maßen aussehen:
Prüfung einer Operation anhand des Rechtebaums Beispiels:


1.6.1.3 Entscheider im Rechtebaum


<table>
<thead>
<tr>
<th>Symbol</th>
<th>Zugriffsrecht</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>♂</td>
<td>Zugriff gewähren</td>
<td>Der Zugriff wird in der zu prüfenden Zugriffssituation erlaubt.</td>
</tr>
<tr>
<td>♂</td>
<td>Zugriff verweigern</td>
<td>Der Zugriff wird in der zu prüfenden Zugriffssituation nicht erlaubt.</td>
</tr>
</tbody>
</table>

Es gibt grundsätzlich zwei verschiedene Entscheider einen positiven - Zugriff erlaubt und einen negativen - Zugriff verboten.

Anleitung zum Anlegen eines Entscheiders

1. Wählen Sie im Rechtebaum die Stelle aus, an der sie einen Entscheider anlegen wollen.
2. Über die Buttons ♂ und ♂ werden neue Entscheider als Unterordner des aktuell ausgewählten Ordners angelegt.
1.6.1.4 Zusammensetzen von Rechten


Um Regeln im Rechtesystem zu definieren stehen die folgenden Funktionen zur Verfügung:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>🍃️</td>
<td>Neuer Operationsfilter</td>
<td>Ein neuer Operationsfilter wird erstellt.</td>
</tr>
<tr>
<td>🍃️</td>
<td>Neuer Suchfilter</td>
<td>Ein neuer Suchfilter wird erstellt.</td>
</tr>
<tr>
<td>🌧️</td>
<td>Neuer Eigenschaftsfilter</td>
<td>Ein neuer Eigenschaftsfilter wird erstellt.</td>
</tr>
<tr>
<td>🌧️</td>
<td>Neuer Strukturordener</td>
<td>Ein neuer Strukturordener wird erstellt.</td>
</tr>
<tr>
<td>🐳️</td>
<td>Zugriff gewähren</td>
<td>Ein positiver Entscheider, der den Zugriff erlaubt, wird erstellt.</td>
</tr>
<tr>
<td>🐳️</td>
<td>Zugriff verweigern</td>
<td>Ein negativer Entscheider, der den Zugriff verhindert, wird erstellt.</td>
</tr>
</tbody>
</table>

Um Rechte sinnvoll zu strukturieren, können Strukturordner verwendet werden. Sie haben keinen Einfluss auf die Traversierung des Rechtebaumes. Sie dienen lediglich dazu bei einer Vielzahl von Rechten, inhaltlich zusammengehörige Teilbäume des Rechtebaumes zu gruppieren.

Anordnung von Ordner im Rechtebaum ändern

Um die Filter und Entscheider im Rechtebaum in die richtige Reihenfolge zu bringen, kann über ein Klick mit der rechten Maustaste ein Kontextmenü aufgerufen werden:

In diesem Kontextmenü kann der Filter oder Entscheider umbenannt, gelöscht und exportiert sowie die Position im Rechtebaum verändert werden. Liegen zwei Ordner (Filter oder
Entscheider) auf der gleichen Ebene, kann mithilfe der Funktion *Nach oben, Nach unten* der Ordner im Rechtebaum weiter nach vorne oder hinten verschoben werden. *Ganz nach oben* und *Ganz nach unten* verschiebt den Ordner entsprechend an die erste bzw. letzte Stelle der Ebene im Rechtebaum.

Sollen Ordner ineinander geschachtelt werden, also die Ebene im Entscheidungsbaum verändert werden, kann dies mit Drag & Drop durchgeführt werden.

**Zusammensetzen von Rechten**

Durch das Zusammensetzen von Filtern und Entscheidern im Rechtebaum gibt es eine Vielzahl von Kombinationsmöglichkeiten um Rechte zu definieren. Es gibt grundsätzlich 3 verschiedene Vorgehensweisen um Rechte zu definieren:

- Definition von Rechten für jede mögliche Zugriffssituation
- Positiv-Konfiguration
- Negativ-Konfiguration

Da die Definition von Zugriffssichten für jede mögliche Zugriffssituation eine sehr aufwendige Vorgehensweise ist, wird i.d.R. eine der beiden anderen Konfigurationswiesen angewendet. Diese werden in den beiden folgenden Abschnitten erläutert.

**1.6.1.4.1 Positiv-Konfiguration von Rechten**


*Beispiel: Positiv-Konfiguration*

Dieses Beispiel zeigt, wie ein positiv formulierter Rechtebaum im Knowledge-Builder aussehen kann:
Rechte
- Lesen
  - Name, Dauer, Erscheinungsdatum
    - Zugriff erlaubt
- Anlegen
  - Objekte von Song
    - Zugriff erlaubt
  - Zugriff verweigert


1.6.1.4.2 Negativ-Konfiguration von Rechten


Beispiel: Negativ-Konfiguration

Dieses Beispiel zeigt, wie ein negativ formulierter Rechtebaum im Knowledge-Builder aussehen kann:
Rechte

- Löschen oder Modifizieren
  - Name, Dauer, Erscheinungsdatum
  - Zugriff verweigert
- Neu Anlegen
  - Song gehört zu Album
  - Zugriff verweigert
- Zugriff erlaubt

Der erste Teilrechtebaum verweigert im Gegensatz zum Beispiel Positiv-Konfiguration die Zugriffsrechte für das Löschen und Modifizieren der Attribute Name, Dauer und Erscheinungsdatum. Der zweite Teilrechtebaum verbietet das Löschen der Relation die Songs mit dem Album verbindet, in dem sie enthalten sind. Alle anderen Operationen dürfen durchgeführt werden.

1.6.1.4.3 Beispiel: Jeder Benutzer darf selbst erstellte Elemente ändern und löschen


Operationsfilter

Im Operationsfilter wurden die Operationen Löschen und Modifizieren ausgewählt.

Suchfilter
Im Suchfilter wird die Relation wurde erstellt von mit dem Relationsziel Person ausgewählt. An dem Relationsziel Person wurde der Zugriffsparameter Benutzer angegeben. Die Einstellung Alle Parameter müssen zutreffen und Suchbedingung muss erfüllt sein sind ausgewählt. In diesem Fall wurde der Operationsparameter Primärelement ausgewählt.

Ein Frage, die das Schema betrifft, ist: An welchen Elementen ist die Relation wurde erstellt von definiert? Es gibt verschiedene Möglichkeiten diese Relation in einem semantischen Netz umzusetzen:

1. Fall Definition an Objekten und Typen: Nur an Objekten und Typen wird die Relation verwendet.
2. Fall Definition an allen Elementen: An allen Objekten, Typen, Erweiterungen, Attributen und Relationen wird die Relation verwendet.

Im ersten Fall macht es Sinn den Operationsparameter Primärelement oder übergeordnetes Element zu verwenden. Definiert man das Recht mit dem übergeordneten Element, so gilt es für nicht nur für das Objekt an sich sondern auch für alle Eigenschaften, die an Objekten gespeichert sind, welche vom Nutzer erstellt wurden. Verwendet man stattdessen den Operationsparameter Primärelement so gilt das Recht ebenfalls für alle Metaeigenschaften des Objektes.

Im zweiten Fall wird der Operationsparameter Zugriffselement verwendet, da nur die Elemente geändert werden dürfen, an denen die Relation wurde erstellt von mit dem entsprechenden Relationsziel, dem Benutzer, vorkommt.

**Das Recht im Rechtebaum zusammensetzen**

Es gibt zwei verschiedene Varianten die Filter zu kombinieren. Gibt es in dem Rechteteilbaum keine Verzweigungen so ist die Reihenfolge der Teilbäume nicht relevant.


Empfehlung: Es ist sinnvoll den Operationsfilter an erster Stelle zu haben, so ist es möglich unter ihm alle anderen Rechte, welche auf die selbe Operation filtern, anzulegen. Dies schafft eine einfacher nachvollziehbare Struktur in den Rechtebaum.

**Erweitertes Recht: Elemente die nicht vom Nutzer erstellt wurden, dürfen nicht geändert oder gelöscht werden**


**Auswirkungen der verschiedenen Versionen auf das Rechtssystem**

**Version 1 (links)**
- Erlaubt wird das Modifizieren und Löschen selbst erstellter Elemente.
- Verboten wird das Modifizieren und Löschen aller anderen Elemente.
- Es wird keine Aussage über alle anderen Operationen gemacht.

**Version 2 (rechts)**
- Erlaubt wird das Modifizieren und Löschen selbst erstellter Elemente.
- Verboten werden alle anderen Operationen auf selbst erstellte Elemente (wie z.B. das Lesen)
- Es wird keine Aussage über alle anderen Elemente gemacht.


1.6.1.5 **Konfiguration von eigenen Operationen**

Wird im Bereich **System** der Ordner **Rechte** ausgewählt, werden im Hauptfenster die Reiter **Gespeicherte Testfälle** und **Konfigurieren** angeboten. Auf dem Reiter **Konfigurieren** können eigene Operationen konfiguriert werden.

**Anleitung zur Konfiguration von eigenen Operationen**

1. Wählen Sie im Knowledge-Builder den Bereich *System* den Ordner *Rechte* aus.
2. Wählen Sie im Hauptfenster den Reiter *Konfigurieren* aus.
4. Geben Sie in nachfolgenden Fenstern für die neue Operation einen internen Namen und eine Beschreibung an.
5. Die neue Operation wird als *Benutzerdefinierte Operation* hinzugefügt.
1.6.2 Trigger

Trigger sind automatische Operationen, die in i-views ausgeführt werden, wenn ein bestimmtes Ereignis eintritt. Sie helfen dabei Arbeitsabläufe zu unterstützen, in dem immer gleich bleibende Arbeitsschritte automatisiert werden.

Beispiele für den Einsatz von Trigger sind:

- Versenden von E-Mails aufgrund einer bestimmten Änderung
- die Bearbeitung von Dokumenten in einer bestimmten Reihenfolge durch bestimmte Personen
- die Kennzeichnung von Aufgaben als offen oder erledigt aufgrund einer bestimmten Bedingung
- die Erstellung von Objekten und Relationen, wenn eine bestimmte Änderung durchgeführt wird
- die Berechnung von Werten in einer vorher definierten Art und Weise
- automatische Generierung des Namensattribut von Objekten (z.B. Zusammensetzung aus Eigenschaften des Objektes)

Wie funktionieren Trigger?


1.6.2.1 Trigger aktivieren

Um mit Triggern arbeiten zu können, muss die Trigger-Funktionalität zunächst im Knowledge-Builder aktiviert werden.

Anleitung zur Aktivierung von Triggern

1. Rufen Sie die Einstellungen des Knowledge-Builders auf.
2. Wählen Sie dort den Reiter System und das Feld Trigger aus.
3. Setzen Sie im Feld Trigger aktiviert einen Haken.


Vor der Aktivierung des Trigger-Funktionalität heißt der Trigger Ordner im Technikbereich
von i-views Trigger (deaktiviert). Durch die Aktivierung wird der Ordner in Trigger umbenannt.


1.6.2.2 Der Triggerbaum

Der Triggerbaum ist wie der Rechtebaum aufgebaut. Er besteht aus Ästen (Teilbäumen), die aus Filtern und Triggern bestehen. Die Filter sind die Bedingungen, die geprüft werden müssen, damit der Trigger am Ende des Teilbaumes ausgeführt werden kann, wenn alle vorher zu prüfenden Bedingungen erfüllt sind.

Der Triggerbaum wird bei jeder Operation auf die Daten abgefragt - der Baum wird "traversiert". Passt ein Teilbaum auf die Zugriffs situation, so wird der Trigger ausgeführt. Passt die Bedingung eines Filters nicht auf die Zugriffs situation, so wird zum nächsten Teilbaum gewechselt. Nach der Ausführung einer Trigger-Aktion wird der Triggerbaum weiter durchlaufen, im Gegensatz zum Rechtesystem, dessen Abarbeitung mit dem Erreichen eines Entscheiders beendet ist. Um im Triggerbaum zu definieren, dass nach der Ausführung einer Aktion keine weiteren Filter geprüft werden sollen, dient die Schaltfläche Keine weiteren Trigger auslösen:

<table>
<thead>
<tr>
<th>Sym- bol</th>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>⏋️</td>
<td>Keine weiteren Trigger auslösen</td>
<td>Die Traversierung des Triggerbaumes wird beendet.</td>
</tr>
</tbody>
</table>

Am Ende eines Teilbaumes steht im Gegensatz zum Rechtesystem kein Entscheider sondern Aktionen zur Verfügung.

<table>
<thead>
<tr>
<th>Sym- bol</th>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚡️</td>
<td>Trigger definieren</td>
<td>Es wird eine neue Trigger-Aktion erstellt.</td>
</tr>
</tbody>
</table>

Die verfügbaren Trigger-Aktionen sind:

- *Log eintragen:* Ein Logeintrag wird geschrieben.
- *Script ausführen > JavaScript:* Eine Script-Datei in JavaScript wird ausgeführt.
- *Script ausführen > KScript:* Eine Script-Datei in KScript wird ausgeführt.

**Gestaltung des Triggerbaumes**

Für die übersichtlichere Gestaltung der Trigger können diese in Strukturordnern gesammelt werden. Die Strukturordner selbst haben keinen Einfluss auf die Traversierung des Triggerbaumes.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>🏡</td>
<td>Strukturordner</td>
<td>Strukturordner für die Gruppierung von Teilbäumen</td>
</tr>
</tbody>
</table>

Beispiel: Triggerbaum

Dieses Beispiel zeigt einen Triggerbaum, der die Namen von Personen und Konzerten automatisch aus Eigenschaften der Objekte zusammensetzt:

Dieser einfach Triggerbaum beginnt mit einem Operationsfilter und teilt sich nach den Operationsfilter in zwei getrennte Teilbäume. Wird einer der beiden Operationen Modifizieren oder Erzeugen ausgeführt, wird diese vom Operationsfilter durchgelassen. Der Teilbaum Person filtert Operationen, die an Attributen, Relationen von Objekten des Typs Person durchgeführt werden. Ist von der Operation entweder das Attribut Vorname oder das Attribut Nachname betroffen, wird diese vom Eigenschaftsfilter durchgelassen. Das dazugehörige Skript, welches das Namensattribut einer Person aus Vor- und Nachname zusammensetzt wird ausgeführt. Der zweite Teilbaum bezieht sich ebenfalls auf den Operationsfilter Modifizieren oder Erstellen aber filtert dann Attribute und Relationen, die an Objekte des Typs Konzert gespeichert sind. Der Eigenschaftsfilter lässt nur Operationen durch, welche an den Attributen Datum, die Relationen zum Veranstaltungsort oder dem Künstler durchgeführt werden. Treffen diese Bedingungen zu, wird das dazugehörige Skript ausgeführt, welches den Name des Konzertes zusammensetzt.

So würde dieser Trigger Baum in i-views aussehen:
1.6.2.3 Trigger erstellen

Wie im Abschnitt Triggerbaum beschrieben, bestehen Trigger aus Filtern und Trigger-Aktionen. Diese werden miteinander kombiniert, so dass eine bestimmte Trigger-Aktion nur dann ausgeführt wird, wenn sie benötigt wird.

Die folgenden Funktionen stehen im Bereich Trigger zur Verfügung:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Funktion</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟</td>
<td>Neuer Operationsfilter</td>
<td>Ein neuer Operationsfilter wird erstellt.</td>
</tr>
<tr>
<td>🌟</td>
<td>Neuer Suchfilter</td>
<td>Ein neuer Suchfilter wird erstellt.</td>
</tr>
<tr>
<td>🌟</td>
<td>Neuer Eigenschaftsfilter</td>
<td>Ein neuer Eigenschaftsfilter wird erstellt.</td>
</tr>
<tr>
<td>🌟</td>
<td>Neuer Löschfilter</td>
<td>Ein neuer Löschfilter wird erstellt.</td>
</tr>
<tr>
<td>🌟</td>
<td>Neuer Strukturordner</td>
<td>Ein neuer Strukturordner wird erstellt.</td>
</tr>
<tr>
<td>🌟</td>
<td>Neuer Trigger</td>
<td>Eine neue Trigger-Aktion wird erstellt.</td>
</tr>
<tr>
<td>🌟</td>
<td>Keine weiteren Trigger auslösen</td>
<td>Ein neuer &quot;Stopp&quot;-Ordner wird erstellt. Dieser beendet die Traversierung des Triggerbaumes.</td>
</tr>
</tbody>
</table>

Bei der Erstellung von Triggern sollten zwei grundsätzliche Eigenschaften des Trigger-Mechanismus beachtet werden:

- Die Ausführung eines Trigger Skriptes, kann dazu führen, dass weitere Trigger ausgelöst werden. Dies passiert dann wenn im Trigger-Skript selbst Operationen in der semantischen Graph-Datenbank ausgeführt werden.
- Nach der Ausführung einer Trigger-Aktion wird der Triggerbaum weiter durchlaufen. Alle Trigger-Aktionen der Teilbaume, die auf die Zugriffssituation zutreffen, werden ausgeführt.
1.6.2.4 Trigger-Aktionen

Trigger-Aktionen dienen dazu intelligente Operationen in der semantischen Graph-Datenbank durchzuführen, die beispielsweise Arbeitsabläufe automatisieren oder unterstützen. Sie werden nur in bestimmten Situationen ausgeführt, nämlich dann wenn die Zugriffssituation und die Verknüpfungen im semantischen Netz einen bestimmten Zustand annehmen, der durch Filter definiert wird.

Anleitung zum Anlegen von Trigger-Aktionen

1. Wählen Sie im Triggerbaum die Stelle, an der die Trigger-Aktion angelegt werden soll.
2. Fügen Sie über den Button \( \rightarrow \) einen neuen Trigger ein.
3. Wählen Sie aus der Liste den Aktionstyp aus: Log eintragen oder Skript ausführen (Wenn Sie ein Skript ausführen wollen, wählen Sie die Skriptsprache aus.)
4. Der Trigger wird als Unterordner des aktuell ausgewählten Ordners erstellt.

1.6.2.4.1 Skript Trigger

Für die Ausführung des Skriptes muss ein Operationsparameter angegeben werden. Im Gegensatz zu Suchfiltern, kann nur ein Operationsparameter angegeben werden. Auf dem im Operationsparameter enthaltenem Element startet die Ausführung des Skriptes.

Zeitpunkt/Art der Ausführung

- Vor der Änderung: Der Trigger wird ausgeführt bevor die Operation durchgeführt wird.
- Nach der Änderung: Der Trigger wird direkt nach der Durchführung der Operation ausgeführt.
- Ende der Transaktion: Der Trigger wird erst am Ende der gesamten Transaktion ausgeführt.
- Job-Client: Der Jobclient bestimmt den Zeitpunkt der Ausführung.

Beachte: Trigger, die bei Löschoperationen ausgelöst werden, sollten vorzugsweise als Zeitpunkt Vor der Änderung verwendet, da ansonsten das zu löschen Element nicht mehr zur Verfügung steht. Für andere Operationen bietet sich als Zeitpunkt eher Nach der Änderung oder Ende der Transaktion an, da dann beispielsweise eine Eigenschaft zu dem neu erstellten Element hinzugefügt werden kann oder automatisch der Name aus verschiedenen Eigenschaften eines Objektes generiert werden kann, wenn eine oder mehrere Eigenschaften geändert wurden.

Werden z.B. mehrere Datensätze in einem Import in i-views importiert, die eine Trigger-Aktion auslösen, die auf Basis von importierten Relationen, Aktionen im sem. Netz durchführen, kann es sinnvoll sein den Import in einer Transaktion durchzuführen und entsprechend Ende der Transaktion als Zeitpunkt der Ausführung auszuwählen, da sonst noch nicht alle Relationen die das Script benötigt importiert wurden.

Je Operationsparameter nur ein mal ausführen

Ist diese Einstellung ausgewählt, dann wird das in Operationsparameter ausgewählte Element maximal ein mal pro Transaktion ausgeführt. Wenn diese Einstellung gesetzt ist, sollte der Ausführungszeitpunkt auf Ende der Transaktion gesetzt werden, damit im Skript der endgültige Zustand des Elements verwendet wird.

Beispiel: Bei Personen soll der Name des Objekts aus Vorname und Nachname zusammengesetzt werden. Mit dieser Einstellung wird bei gleichzeitiger Änderung von Vor-
Nachname der Trigger nur ein mal ausgeführt.

**Ausführung löst keine Trigger aus**

Mit dieser Einstellung wird festgelegt, dass durch die Operationen, die innerhalb eines Triggers ausgeführt werden, keine weiteren Trigger ausgelöst werden können. Mit dieser Einstellung lassen sich Endlosschleifen vermeiden.

**Bei Skriptfehlern Skript weiter ausführen**

Ist diese Einstellung aktiv, so wird versucht nach Ausführungsfehlern wieder aufzusetzen und die Ausführung des Skriptes fortzuführen. Diese Einstellung eignet sich vorwiegend für Skripte, die voneinander unabhängige Anweisungen ausführen sollen, nicht für solche, die auf vorherige Schritte des Skriptes aufbauen.

**Transaktion abbrechen, wenn Trigger fehlschlägt**


**Ausführen während eines Daten-Refactorings**


Der Funktionsrumpf für Skript-Trigger wird automatisch angelegt. Das Skript hat drei Parameter:

<table>
<thead>
<tr>
<th>parameter</th>
<th>$k.SemanticElement / $k.Folder</th>
<th>Der ausgewählte Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>object</td>
<td>Objekt mit Daten der Änderung (neuer Attribuwert usw.)</td>
</tr>
<tr>
<td>user</td>
<td>$k.User</td>
<td>Der Benutzer der die Änderung ausgelöst hat</td>
</tr>
</tbody>
</table>

Folgendes Beispiel setzt die Attribute mit den internen Namen "geaendertAm" / "geaendertVon". Als Parameter sollte hier "Primäres Kernobjekt" ausgewählt werden.

```javascript
/**
 * Perform the trigger
 * @param parameter The chosen parameter, usually a semantic element
 * @param {object} access Object that contains all parameters of the access
 * @param {$k.User} user User that triggered the access
 **/

function trigger(parameter, access, user)
{
    parameter.setAttributeValue("geaendertAm", new Date());

```
var userName = $k.user().name();
if (userName)
  parameter.setAttributeValue("geaendertVon", userName);
else
  parameter.attributes("geaendertVon").forEach(function(old) { old.remove });
}

Das Parameter "access" kann (je Operation varierend) folgende Eigenschaften enthalten:

<table>
<thead>
<tr>
<th>Eigenschaft</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessedObject</td>
<td>Zugriffselement</td>
</tr>
<tr>
<td>core</td>
<td>Kernobjekt</td>
</tr>
<tr>
<td>detail</td>
<td>Detail</td>
</tr>
<tr>
<td>inversePrimaryCoreTopic</td>
<td>Primäres Relationsziel</td>
</tr>
<tr>
<td>inverseRelation</td>
<td>Inverse Relation</td>
</tr>
<tr>
<td>inverseTopic</td>
<td>Relationsziel</td>
</tr>
<tr>
<td>operationSymbol</td>
<td>&quot;read&quot;, &quot;deleteRelation&quot;, etc.</td>
</tr>
<tr>
<td>primaryCoreTopic</td>
<td>Primäres Kernobjekt</td>
</tr>
<tr>
<td>primaryProperty</td>
<td>Primäreigenschaft</td>
</tr>
<tr>
<td>primaryTopic</td>
<td>Primärelement</td>
</tr>
<tr>
<td>property</td>
<td>Eigenschaft</td>
</tr>
<tr>
<td>topic</td>
<td>Übergeordnetes Element</td>
</tr>
<tr>
<td>user</td>
<td>Benutzer (identisch zu &quot;user&quot;-Parameter der Funktion)</td>
</tr>
</tbody>
</table>

1.6.2.4.2 Log Trigger

Möchte man die Trigger-Funktionalität überwachen bzw. dokumentieren, wann welcher Trigger ausgelöst wurde und welche Operationen im sem. Netz ausgeführt wurden, eignen sich Log Trigger. Der Log wird in das jeweilige Log File (bridge.log, batchtool.log etc.) geschrieben in dessen Anwendungsumgebung die Operation, welche den Trigger ausgelöst hat, durchgeführt wird.

<table>
<thead>
<tr>
<th>Zeilen des Logeintrages</th>
<th>Zustand des sem. Netzes zum Zeitpunkt</th>
</tr>
</thead>
<tbody>
<tr>
<td>#pre</td>
<td>vor Auslösung</td>
</tr>
<tr>
<td>#post</td>
<td>nach Auslösung</td>
</tr>
<tr>
<td>#end</td>
<td>am Ende der Transaktion</td>
</tr>
</tbody>
</table>
Logeinträge dienen dazu nachzuvollziehen, ob in einer bestimmten Zugriffs situation, die tatsächlich geschehen ist, ein Trigger ausgeführt wurde und was er gemacht hat. Im Gegensatz dazu kann in der Testumgebung getestet werden, ob in einer bestimmten Zugriffs situation ein Trigger ausgelöst werden würde oder nicht, ohne dass die konkrete Zugriffs situation durchgeführt wird.

**Anleitung zum Anlegen von Log Triggern**

1. Wählen Sie im Triggerbaum das Trigger-Skript aus, welches geloggt werden soll.
2. Erstellen Sie über den Button ein Trigger vom Typ Log eintragen im Triggerbaum direkt vor dem Skript-Trigger.

Beispiel:

![Logeintrag](image)

der das Ändern des Attributs e-mail durch einen Trigger dokumentiert.

### 1.6.3 Filterarten


Im Rechtesystem stehen die drei Filterarten Operationsfilter, Suchfilter und Eigenschaftsfilter zur Verfügung. Zusätzlich zu den drei grundsätzlichen Filterarten bietet der Bereich Trigger einen spezifischen Filter - den Löschfilter.

**Es gibt verschiedene Arten von Filtern - Wann benutzen wir welchen Filter?**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Filter</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟</td>
<td>Operationsfilter</td>
<td>Filtert die Operationen; Auswahl aus Liste</td>
</tr>
<tr>
<td>🎶</td>
<td>Suchfilter</td>
<td>Filtert Elemente durch Strukturabfrage</td>
</tr>
<tr>
<td>🏆</td>
<td>Eigenschaftsfilter</td>
<td>Filtert Relationen und Attribute; Auswahl aus Liste</td>
</tr>
<tr>
<td>🗑️</td>
<td>Löschfilter</td>
<td>Filtert das Löschen von Elementen</td>
</tr>
</tbody>
</table>


**Anleitung zum Anlegen eines Filters**

1. Wählen Sie im Rechte- bzw. Triggerbaum die Stelle aus, an der Sie einen neuen Filter anlegen wollen.
2. Erstellen Sie über die Buttons ♻️, 🔫 oder ✂️ einen neuen Filter.
3. Der Filter wird als Unterordner des aktuell ausgewählten Ordners im Baum angelegt.

**1.6.3.1 Operationsfilter**

Für welche Operationen ein Zugriffsrecht gelten soll oder ein Trigger ausgeführt werden soll, kann nur mithilfe von Operationsfiltern angegeben werden. Durch die Auswahl der gewünschten Operation kann diese dem Filter hinzugefügt oder wieder entfernt werden.

Im Kapitel Operationen werden alle verfügbaren Operationen aufgelistet und zusätzlich wird angegeben, welche Operationsparameter in Kombination verwendet werden können. Die verschiedenen Operationsparameter werden entsprechend im Kapitel Operationsparameter erklärt.

### 1.6.3.2 Eigenschaftsfilter

Mit Eigenschaftsfilters können Attribute und Relationen gefiltert werden. Es gibt zwei verschiedene Vorgehensweisen einen Eigenschaftsfilter zu verwenden:

- **Einschränkung auf Eigenschaften**: Angabe der Eigenschaften für die die Bedingung gelten soll. Nachfolgende Filter oder Entscheider des Teilbaumes werden nur ausgeführt, wenn
die Zugriffseigenschaft mit den ausgewählten Eigenschaft übereinstimmt.

- **Ausgenommen folgende Eigenschaften**: Angabe der Eigenschaften für die die Bedingung nicht gelten soll. Stimmt die Zugriffseigenschaft mit einer der ausgewählten Eigenschaften überein, werden nachfolgende Filter, Entscheider oder Trigger nicht ausgeführt.


1.6.3.3 **Suchfilter**

Suchfilter ermöglichen es Elemente im Umfeld des Elementes, auf das zugegriffen werden soll, einzubeziehen. So können nicht nur einzelne Eigenschaften sondern auch Zusammen-

Es gibt zwei verschiedene Vorgehensweisen Suchfilter zu definieren:

- **Suchbedingung muss erfüllt sein**: Diese Einstellung ist initial ausgewählt. Stimmt das Suchergebnis der Strukturabfrage mit dem Operationsparameter überein, ist die Bedingung des Filters erfüllt und nachfolgende Filter, Entscheider oder Trigger werden ausgeführt.

- **Suchbedingung darf nicht erfüllt sein**: Liefer die Strukturabfrage als Ergebnis das selbe Element wie der Zugriffsparameter, ist die Bedingung nicht erfüllt und die Prüfung des Rechte- bzw. Triggerbaumes wechselt zum nächsten Teilbaum. Ist das Ergebnis der Strukturabfrage ein anderes als der Zugriffsparameter liefert, ist die Bedingung erfüllt und der nachfolgende Filter, Entscheider oder Trigger wird ausgeführt.

Die Objekte des Typs links oben, die auf die Suchbedingung passen, sind das Ergebnis der Strukturabfrage. Diese werden mit dem Element, das vom Operationsparameter übergeben wird, verglichen. In der Strukturabfrage können Zugriffsparameter verwendet werden, mit diesen können beispielsweise der Benutzer, das Zugriffsobjekt usw. in die Suche einbezogen werden.

Bei der Auswahl der Operationsparameter kann konfiguriert werden, ob

- alle ausgewählten Parameter zutreffen müssen (Alle Parameter müssen zutreffen)
- oder nur ein Parameter zutreffen muss (Ein Parameter muss zutreffen).

Beachte: Initial ist die Einstellung Alle Parameter müssen zutreffen ausgewählt. Werden beispielweise die Operationsparameter Zugriffselement und Primärelement ausgewählt, ist die Bedingung nur dann erfüllt, wenn das Ergebnis der Strukturabfrage sowohl Zugriffselement als auch Primärelement der zu prüfenden Operation ist.

**Beispiel 1: Suchfilter im Rechtesystem**
Es soll ein Recht definiert werden, das besagt, dass bereits veröffentlichte Songs von allen gesehen werden dürfen unveröffentlichte Songs hingegen nicht.

In diesem Beispiel möchte die Benutzer Paul den Song X lesen. Diese Operation wird nun vom Rechtssystem geprüft. Dort ist ein Suchfilter definiert, der prüft, ob der Song bereits veröffentlicht ist. In der Strukturabfrage des Suchfilters werden Objekte vom Typ Song gesucht, mit der Einschränkung, dass das Attribut Erscheinungsdatum in der Vergangenheit liegt. Die Strukturabfrage liefert alle Songs, die diese Bedingung erfüllen. Ist der Song X einer davon, fällt die Prüfung des Filters positiv aus und der auf den Suchfilter nachfolgende Ordner (mit einem Filter oder Entscheider) wird ausgeführt.

Bei dem Suchfilter wurden die Einstellungen Suchbedingung muss erfüllt sein und Alle Parameter müssen zutreffen ausgewählt.

Beispiel 2: Suchfilter im Rechtssystem

In den meisten Fällen gibt es eine Verbindung zwischen dem Benutzer, der zugreifen will und den Objekten oder Eigenschaften, auf die er zugreifen will. Ein Beispiel dafür wäre: "Mitarbeiter einer Abteilung die eine Branche betreuen, dürfen alle Kunden aus dieser Branche bearbeiten." Eine andere Version dieses Beispiels, das unten dargestellt wird, wäre: "Nutzer, die einen Künstler pflegen, dürfen diesen bearbeiten und löschen."

1.6.3.4 Löschfilter

Löschfilter stehen nur bei der Definition von Triggern zur Verfügung. Sie werden dazu eingesetzt, in einer Löschsituation zu testen, ob das übergeordnete Element auch von dem Löschvorgang betroffen ist. Will man beispielsweise, dass ein Trigger nicht ausgeführt wird, wenn ein Objekt samt all dessen Eigenschaften gelöscht wird, aber dann wenn eine bestimmte Eigenschaft des Objektes gelöscht wird, muss ein Löschfilter verwendet werden.

Bei der Definition eines Löschfilters, muss mindestens ein Operationsparameter angegeben werden, der bestimmt, die Löschung welches Objektes getestet werden soll.

- **Alle Parameter müssen zutreffen**: Alle angegebenen Operationsparameter müssen zutreffen. Werden beispielsweise zwei Operationsparameter angegeben (Zugriffsobjekt und Primärobject), dann wird geprüft, ob der Löschvorgang sowohl für Zugriffsobjekt als auch für Primärobject gilt, das kann nur der Fall sein, wenn das Primärobject auch das Zugriffsobjekt ist.
- **Ein Parameter muss zutreffen**: Nur einer der angegebenen Operationsparameter muss zutreffen.

Anmerkung: In den meisten Fällen bietet sich der Operationsparameter übergeordnetes Element oder Primärobject an, da überprüft werden soll, ob entweder nur die Eigenschaft gelöscht wird, oder ob die Eigenschaft gelöscht wird, weil das gesamte Objekt gelöscht wurde.

- **Nicht vom Löschvorgang erfasst**: Die Bedingung des Filters ist positiv, wenn das in Operationsparameter übergebene Element in dieser Transaktion nicht gelöscht wird.
- **Vom Löschvorgang erfasst**: Die Bedingung des Filters ist entsprechend positiv, wenn das in Operationsparameter übergebene Element in dieser Transaktion gelöscht wird.

**Beispiel: Löschfilter bei Triggern**

In diesem Beispiel soll ein Trigger nur dann ausgeführt werden, wenn der Künstler, der Ort oder das Datum einer Veranstaltung geändert oder gelöscht wird, aber nicht wenn das Objekt gelöscht wird, an denen die Eigenschaften gespeichert sind. Dafür wird die Einstellung *Nicht vom Löschvorgang erfasst* verwendet. Ist das übergeordnete Zugriffselement vom Löschvorgang erfasst, das in diesem Fall das Konzert-Objekt selbst ist, dann wird die Prüfung des Teilbaumes, aufgrund des negativen Ergebnisses des Filters, abgebrochen.
Verwendet wird der Operationsparameter Übergeordnetes Element und die Einstellung Nicht vom Löschvorgang erfasst.


In dieser Zugriffsituation wird das Objekt "Konzert X" vom Nutzer Paul gelöscht. Durch das Löschen des Objektes werden automatisch alle Eigenschaften des Objektes mit gelöscht - also auch alle Attribute des Objektes. Die Prüfung des Triggerbaums wird sowohl für die Lösung des Objektes


1.6.4 Operationsparameter

Operationsparameter steuern bei Suchfiltern, mit welchem Element das Ergebnis der Strukturabfrage für die Prüfung der Bedingung verglichen werden soll. Im einfachsten Fall wird das Ergebnis mit dem Element verglichen, mit dem die zu prüfende Operation durchgeführt werden soll. Mit Hilfe von Operationsparametern kann das übergebene Element verändert werden. Es kann der aktuelle Benutzer oder Elemente aus dem Umfeld des Elements ausgewählt werden, die als Vergleichselement für den Suchfilter verwendet werden sollen.

Sie werden unter anderem auch bei Löschfiltern und Skript-Triggern verwendet. Dort geben sie an, ausgehend vom Element auf dem der Zugriff durchgeführt wird, auf welchem Element das Skript ausgeführt werden soll bzw. das Löschen welchen Elements gefiltert werden soll.


<table>
<thead>
<tr>
<th>Operationsparameter</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benutzer</td>
<td>Der Benutzer ist das Objekt des Benutzers, der die Operation ausführt.</td>
</tr>
<tr>
<td>Inverse Relation</td>
<td>Falls die von der Operation betroffene Eigenschaft eine Relation ist, enthält der Parameter die inverse Relationshälfte.</td>
</tr>
<tr>
<td>Inverser Relationstyp</td>
<td>Der <em>Inverse Relationstyp</em> ist der Typ der inversen Relation. Dieser kann bei Erzeugung von Relationen verwendet werden.</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Kernobjekt</td>
<td>Wenn das übergeordnete Element eine Erweiterung ist, dann ist das Kernobjekt das Objekt an dem die Erweiterung gespeichert ist. Ansonsten ist das Kernobjekt identisch mit Zugriffselement.</td>
</tr>
<tr>
<td>Ordner</td>
<td>Der Operationsparameter <em>Ordner</em> ist der von der Operation betroffene Ordner.</td>
</tr>
<tr>
<td>Primäreigenschaft</td>
<td>Bei Metaeigenschaften ist die <em>Primäreigenschaft</em> die dem Objekt, Typ oder Erweiterung nächste Eigenschaft. Ansonsten ist <em>Primäreigenschaft</em> identisch mit Eigenschaft.</td>
</tr>
<tr>
<td>Primäres Kernobjekt</td>
<td>Wenn das Primärelement eine Erweiterung ist, dann ist das <em>Primäre Kernobjekt</em> das Kernobjekt der Erweiterung. Ansonsten ist das <em>Primäre Kernobjekt</em> identisch mit Kernobjekt.</td>
</tr>
<tr>
<td>Primäres Relationsziel</td>
<td>Das <em>Primäre Relationsziel</em> ist das Primärelement des Relationssziels.</td>
</tr>
<tr>
<td>Primärelement</td>
<td>Falls das übergeordnete Zugriffselement eine Eigenschaft ist, ist das <em>Primärelement</em> das Objekt, der Typ oder die Erweiterung an dem die Eigenschaft gespeichert ist (transitiv). Ansonsten ist das <em>Primärelement</em> identisch mit dem übergeordneten Element.</td>
</tr>
<tr>
<td>Relationsziel</td>
<td>Falls die von der Operation betroffene Eigenschaft eine Relation ist, enthält der Parameter <em>Relationsziel</em> das Relationssziel der Relationshälfte. (Die Relationsquelle wäre in diesem Fall das übergeordnete Element.)</td>
</tr>
<tr>
<td>Übergeordnetes Element</td>
<td>Das <em>Übergeordnete Element</em> ist das von der Operation betroffene Objekt, der Typ oder die Erweiterung. Bei Eigenschaften ist das <em>Übergeordnete Element</em> das Objekt, der Typ oder die Erweiterung an der die Eigenschaft gespeichert ist.</td>
</tr>
<tr>
<td>Zugriffselement</td>
<td>Das <em>Zugriffselement</em> ist das von der Operation betroffene Element.</td>
</tr>
</tbody>
</table>

**1.6.4.1 Operationsparameter Zugriffselement**

1.6.4.2 Operationsparameter Benutzer

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Benutzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objekt, Typ, Erweiterung oder Eigenschaft</td>
<td>Objekt des aktuell angemeldeten Nutzers</td>
</tr>
</tbody>
</table>

1.6.4.3 Operationsparameter (Ober)typ

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>(Ober)typ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objekt oder Erweiterung</td>
<td>Der Typ des Objektes oder der Erweiterung</td>
</tr>
<tr>
<td>Typ</td>
<td>Der Obertyp</td>
</tr>
<tr>
<td>Eigenschaft</td>
<td>Der Typ der Eigenschaft</td>
</tr>
</tbody>
</table>

1.6.4.4 Operationsparameter Übergeordnetes Element
Das übergeordnete Element wird dann verwendet, wenn direkte Eigenschaften eines Elementes abgefragt werden sollen.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Übergeordnetes Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objekt, Typ oder Erweiterung</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Eigenschaft</td>
<td>Objekt, Typ oder Erweiterung an dem oder der die Eigenschaft gespeichert ist</td>
</tr>
<tr>
<td>Metaeigenschaft</td>
<td>Eigenschaft, an der die Metaeigenschaft gespeichert ist</td>
</tr>
</tbody>
</table>
1.6.4.5 Operationsparameter Eigenschaft
Als Eigenschaften werden Attribute und Relationen verstanden. Der Operationsparameter enthält das Attribute oder die Relation auf der die Operation durchgeführt wird. Wird die Operation auf einem Objekt oder Typ durchgeführt, ist der Operationsparameter Eigenschaft leer.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Eigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute oder Relation</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Objekt, Typ oder Erweiterung</td>
<td>Leer</td>
</tr>
</tbody>
</table>

1.6.4.6 Operationsparameter Inverse Relation

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Inverse Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationshälftie</td>
<td>Die inverse Relationshälftie</td>
</tr>
<tr>
<td>Objekt, Typ, Erweiterung oder Attribut</td>
<td>Leer</td>
</tr>
</tbody>
</table>

1.6.4.7 Operationsparameter Inverser Relationstyp
Der inverse Relationstyp ist der Typ der inversen Relation.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Inverser Relationstyp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationshälftie</td>
<td>Typ der inversen Relationshälftie</td>
</tr>
<tr>
<td>Objekt, Typ, Erweiterung oder Attribut</td>
<td>Leer</td>
</tr>
</tbody>
</table>

1.6.4.8 Operationsparameter Relationsziel
Das Relationsziel ist nicht die Quelle sondern das "Ziel" einer Relationshälftie. Es kann auch als Relationsquelle der inversen Relationshälftie betrachtet werden.
<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Relationsziel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationshälfte</td>
<td>Das Relationsziel ist die Relationsquelle der inversen Relation</td>
</tr>
<tr>
<td>Objekt, Typ, Erweiterung oder Attribut</td>
<td>Leer</td>
</tr>
</tbody>
</table>

1.6.4.9 Operationsparameter Primärelement

Das Primärelement liefert immer ein Objekt, Typ oder Erweiterung. Wird das Primärelement auf Metaeigenschaften ausgeführt, werden die Eigenschaften transitiv abgearbeitet, bis das Objekt, der Typ oder die Erweiterung gefunden wurde, an dem die Eigenschaften aufgehängt sind.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Primärelement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objekt, Typ oder Erweiterung</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Eigenschaft</td>
<td>Objekt, Typ oder Erweiterung an dem oder der die Eigenschaft gespeichert ist</td>
</tr>
<tr>
<td>Metaeigenschaft</td>
<td>Objekt, Typ oder Erweiterung an dem oder der die Eigenschaft gespeichert ist (transitiv)</td>
</tr>
</tbody>
</table>

1.6.4.10 Operationsparameter Primäres Relationsziel

Das primäre Relationsziel ist im Gegensatz zum Primärelement einer Relationshälfte nicht das Objekt, der Typ oder die Erweiterung an der die Relationshälfte angebracht ist sondern das Objekt, der Typ oder die Erweiterung an der die inverse Relationshälfte aufgehängt ist.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Primäres Relationsziel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationshälfte</td>
<td>Das Primärelement des Relationsziels (Objekt, Typ oder Erweiterung an dem oder der die inverse Relationshälfte gespeichert ist)</td>
</tr>
<tr>
<td>Relationshälfte deren Relationsziel eine Eigenschaft oder Metaeigenschaft ist</td>
<td>Das Primärelement des Relationsziels (Objekt, Typ oder Erweiterung der Metaeigenschaft oder Eigenschaft an der die inverse Relationshälfte gespeichert ist)</td>
</tr>
<tr>
<td>Objekt, Typ, Erweiterung oder Attribut</td>
<td>Leer</td>
</tr>
</tbody>
</table>
1.6.4.11 Operationsparameter Kernobjekt
Das Kernobjekt wird verwendet, wenn mit Erweiterungen gearbeitet wird. Das Kernobjekt liefert anstatt der Erweiterung das Objekt, an dem die Erweiterung gespeichert ist.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Kernobjekt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objekt, Typ oder Eigenschaft</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Erweiterung</td>
<td>Das Objekt an dem die Erweiterung gespeichert ist</td>
</tr>
</tbody>
</table>

1.6.4.12 Operationsparameter Primäres Kernobjekt
Wenn bei einem Element das zugehörige Objekt oder der zugehörige Typ verarbeitet werden soll, muss das primäre Kernobjekt verwendet werden. Im Gegensatz zum Primärelement werden keine Erweiterungen zugelassen. Bei diesen wird das Kernobjekt ausgegeben.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Primäres Kernobjekt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erweiterung</td>
<td>Das Objekt an dem die Erweiterung gespeichert ist</td>
</tr>
<tr>
<td>Objekt oder Typ</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Eigenschaft oder einer Erweiterung</td>
<td>Das Objekt an dem die Erweiterung gespeichert ist</td>
</tr>
<tr>
<td>Eigenschaft oder eines Objektes oder Typs</td>
<td>Primärelement - Objekt oder der Typ an dem die Eigenschaft gespeichert ist (transitiv)</td>
</tr>
</tbody>
</table>

1.6.4.13 Operationsparameter Primäreigenschaft
Die Primäreigenschaft ist immer eine Eigenschaft. Sie ähnelt dem Primärelement in der Hinsicht, dass sie transitiv Metaeigenschaften abarbeitet. Sie liefert aber im Gegensatz die letzte Eigenschaft die vor dem Primärelement kommt - also die Eigenschaft, die direkt am Primärelement gespeichert ist.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Primäreigenschaft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenschaft</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Metaeigenschaft (oder Metaeigenschaft einer Metaeigenschaft)</td>
<td>Die Eigenschaft, die dem Objekt, Typ oder der Erweiterung am nächsten ist</td>
</tr>
</tbody>
</table>
1.6.4.14 Operationsparameter Ordner
Soll ein Ordner aus dem Bereich Ordner des Wissensnetzes als Parameter an die Suche übergeben werden, dann muss der Operationsparameter Ordner verwendet werden.

<table>
<thead>
<tr>
<th>Zugriffselement</th>
<th>Ordner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordner</td>
<td>Das Zugriffselement selbst</td>
</tr>
<tr>
<td>Objekt, Typ, Erweiterung oder Eigenschaft</td>
<td>Leer</td>
</tr>
</tbody>
</table>

1.6.4.15 Beispiele: Die Verwendung von Operationsparametern
Beispiel 1: Zugriffselement und Eigenschaft im Rechtesystem

Das unten aufgeführte Beispiel zeigt auf der linken Seite die Zugriffs situation und auf der rechten Seite den dazugehörigen Suchfilter.

Zugriffs situation: Der Nutzer Paul möchte das Attribut Dauer von Song X ändern.
Suchfilter: Es werden alle Attribute gefiltert die, die von einem bestimmten Benutzer angelegt wurden. In der Strukturabfrage wird der Zugriffsparameter Benutzer verwendet, der die Objekte von Nutzer auf die Person einschränkt, welche die Operation ausführen möchte. Entsprechend sind das alle Attribute, die von Paul angelegt wurden.
Prüfung der Zugriffsrechte: Für die Prüfung der Zugriffsrechte wird das Attribut (das Zugriffselement/die Eigenschaft), an dem die Operation durchgeführt werden soll, an den Suchfilter übergeben. Ist dieses Attribut in der Menge der Suchergebnisse enthalten, dann ist die Prüfung des Suchfilters positiv.
Operationsparameter: Das Attribut Dauer selbst wird an den Suchfilter übergeben. In
diesem Fall könnte sowohl der Operationsparameter Zugriffselement als auch Eigenschaft verwendet werden, da das Attribut Dauer selbst eine Eigenschaft ist und das Zugriffselement der Operation darstellt.

**Beispiel 2: Übergeordnetes Element und Primärelement im Rechtesystem**

Dieses Beispiel zeigt auf der linken Seite die Zugriffsituation und auf der rechten Seite den dazugehörigen Suchfilter.

**Zugriffssituation:** Der Nutzer Paul nimmt eine Änderung des Attributes Dauer, das aktuell den Wert 02:30 annimmt und zum Objekt Song X gehört, vor.

**Suchfilter:** Der Suchfilter ist so definiert, dass er alle Objekte sucht, die von einem bestimmten Benutzer angelegt wurden, das ist als Zugriffselement der aktuell angemeldete Nutzer. Entsprechend findet der Suchfilter alle Objekte, die von Paul angelegt wurden.

**Prüfung der Zugriffsrechte:** Ist in der Ergebnismenge des Suchfilters der Song X enthalten, wird der nachfolgende Ordner (Filter oder Entscheider) ausgeführt.

**Operationsparameter:** Die Verwendung des Operationsparameter übergeordnetes Element führt dazu, dass nicht das Attribut Dauer an dem die Änderung stattfinden soll, an den Suchfilter übergeben wird, sondern das Objekt an dem es definiert wurde. Das ist in diesem Fall der Song X. Neben dem übergeordneten Element könnte in diesem Fall auch der Operationsparameter Primärelement verwendet werden. Der Operationsparameter übergeordnetes Element führt dazu, dass alle Eigenschaften und das Objekt selbst positiv von Filter bewertet würde. Zusätzlich würde der Operationsparameter Primärelement auch Metaeigenschaften des Objektes zulassen, egal wie viele andere Eigenschaften zwischen Objekt und Metaeigenschaft hängen.

**Beispiel 3: (Ober)typ im Rechtesystem**

Das Beispiel stellt auf der linken Seite die Zugriffsituation dar und auf der rechten Seite wird der Suchfilter abgebildet, der in dieser Situation zum Einsatz kommt.
**Zugriffs situation**: Der Nutzer Paul möchte das Attribut Dauer am Objekt Song X erstellen. Es soll den Wert 02:30 haben.

**Suchfilter**: Der Suchfilter liefert den Attributtyp Dauer.

**Prüfung der Zugriffsrechte**: Ist das zu erstellende Attribut vom Typ Dauer, dann fällt die Prüfung des Suchfilters positiv aus.


### 1.6.5 Operationen

In Operationsfiltern können Operationen angegeben werden, die dann im Filterprozess von Operationsfilter zugelassen werden. Wird in der Zugriffs situation eine andere Operation ausgeführt, als im Operationsfilter angegeben, wird bei der Traversierung des Rechte bzw. Triggerbaumes zum nächsten Teilbaum gewechselt.

Die allgemeinen Operationen Erzeugen, Lesen, Modifizieren und Löschen bestehen aus mehreren einzelnen Operationen. Wird eine der Operationen verboten, werden somit auch alle darin enthaltenen Operationen nicht erlaubt und umgekehrt wird eine Operation verboten, so werden alle enthaltenen Operationen automatisch mit verboten.


**Besonderheiten bei Triggern**

Bei Triggern können keine lesenden Operationen verwendet werden. Außerdem stehen bei Triggern die Operationsgruppen Abfrage (Operation: In Strukturabfrage verwenden), Anzeige von Objekten (Operation: Im Grapheditor anzeigen) und Bearbeiten (Operation: Attributwert validieren) nicht zur Verfügung.

Außerdem steht bei den Erzeugen Operationen bei Triggern der Operationsparameter Zugriffsselement zur Verfügung, wenn Zeitpunkt/Art der Ausführung auf Nach der Änderung oder Ende der Transaktion gesetzt ist.
<table>
<thead>
<tr>
<th>Operationsgruppe</th>
<th>Operation</th>
<th>Operationsparameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abfrage</td>
<td>In Strukturabfrage verwenden</td>
<td>Zugriffselement</td>
</tr>
<tr>
<td>Anzeigen von Objekten</td>
<td>im Grapheditor anzeigen</td>
<td>Zugriffselement</td>
</tr>
<tr>
<td>Bearbeiten</td>
<td>Attributwert validieren</td>
<td>Zugriffselement, Eigenschaft, übergeordnetes Element, (zu prüfender Parameter: Attributwert)</td>
</tr>
<tr>
<td>Benutzerdefinierte</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erzeugen</td>
<td>Attribut erzeugen</td>
<td>(Ober)typ, übergeordnetes Element</td>
</tr>
<tr>
<td></td>
<td>Erweiterung erzeugen</td>
<td>(Ober)typ, übergeordnetes Element, Kernobjekt</td>
</tr>
<tr>
<td></td>
<td>Objekt erzeugen</td>
<td>(Ober)typ</td>
</tr>
<tr>
<td></td>
<td>Ordnern erzeugen</td>
<td>Ordner</td>
</tr>
<tr>
<td></td>
<td>Relation erzeugen</td>
<td>(Ober)typ, übergeordnetes Element, Relationssziel, inverser Relationstyp</td>
</tr>
<tr>
<td></td>
<td>Relationshälfte erzeugen</td>
<td>(Ober)typ, übergeordnetes Element, Relationssziel</td>
</tr>
<tr>
<td></td>
<td>Typ erzeugen</td>
<td>(Ober)typ</td>
</tr>
<tr>
<td></td>
<td>Übersetzung hinzufügen</td>
<td>Zugriffselement, Eigenschaft, übergeordnetes Element</td>
</tr>
<tr>
<td>Lesen</td>
<td>Alle Objekte/Eigenschaften des Typs lesen</td>
<td>(Ober)typ</td>
</tr>
<tr>
<td></td>
<td>Attribut lesen</td>
<td>Zugriffselement, Eigenschaft, übergeordnetes Element</td>
</tr>
<tr>
<td></td>
<td>Objekt lesen</td>
<td>Zugriffselement, übergeordnetes Element</td>
</tr>
<tr>
<td></td>
<td>Relation lesen</td>
<td>Zugriffselement, übergeordnetes Element, Eigenschaft, inverse Relation, Relationsziel, inverses Relationsziel</td>
</tr>
<tr>
<td></td>
<td>Typ lesen</td>
<td>Zugriffselement, übergeordnetes Element</td>
</tr>
<tr>
<td>Löschen</td>
<td>Attribut löschen</td>
<td>Zugriffselement, übergeordnetes Element</td>
</tr>
<tr>
<td>Erweiterung löschen</td>
<td>Zugriffselement, Eigenschaft, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Objekt löschen</td>
<td>Zugriffselement, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td>Ordner löschen</td>
<td>Ordner</td>
<td></td>
</tr>
<tr>
<td>Relationshälfte löschen</td>
<td>Zugriffselement, inverse Relation, Eigenschaft, übergeordnetes Element, Relationsziel, inverses Relationsziel</td>
<td></td>
</tr>
<tr>
<td>Typ löschen</td>
<td>Zugriffselement, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td>Übersetzung entfernen</td>
<td>Zugriffselement, Eigenschaft, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td>Modifizieren</td>
<td>Atrributwert modifizieren</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zugriffselement, Eigenschaft, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ordner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ordner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Schema modifizieren</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zugriffselement, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Typ wechseln</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zugriffselement, übergeordnetes Element</td>
<td></td>
</tr>
<tr>
<td>Werkzeuge verwenden</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Import</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Script bearbeiten/ausführen</td>
<td></td>
</tr>
</tbody>
</table>

**Objekt lesen**
Die Operation *Objekt lesen* deckt das Anzeigen von Objekten auf dem Reiter Objekte bei dem entstehenden Objekttyp ab. Die Operation verbietet aber nicht das Anzeigen des Objektes, wenn es über ein verknüpftes Objekt aufgerufen wird. In diesem Fall gelten dann die Operationen für Eigenschaften *Attribut lesen* und *Relation lesen*.

**Alle Objekte/Eigenschaften des Typs lesen**

In Strukturbfrage verwenden (veraltet)
Ist ein negatives Zugriffsrecht für ein Element definiert, das auf die Operation In Strukturbfrage verwenden gefiltert wird, dann darf das Element nicht in einer Strukturbfrage verwendet werden. Es wird auch dann nicht in Strukturbfragen berücksichtigt, wenn der (abstrakte) Obertyp angegeben wird.

Attributwert validieren
Die Operation Attributwert validieren wird dann verwendet, wenn der zu setzende Attributwert bestimmte Bedingungen erfüllt muss. Die Definition der Bedingung an den Attributwert wird in einer Strukturbfrage gemacht. Dort stehen für die Validierung des Attributwertes zwei Definitionsmöglichkeiten zur Verfügung:

- Bedingung für den zu setzenden Attributwert:
  Der neue Wert des Attributes kann durch Vergleich mit einem angegebenen Wert in der Strukturbfrage validiert werden.

  Beispiel: Der Attributwert darf nur kleiner gleich 4,0 sein.

- Vergleiche mit dem zu setzenden Attributwert:
  Hierbei wird der aktuelle Wert mit dem neuen Wert verglichen.

  Beispiel: Der neue Wert des Attributes Alter darf in diesem Fall nur größer werden. Kleinere Werte werden nicht zugelassen.

- Vergleiche den zu setzenden Wert mit dem Ergebnis eines Skripts:
  Hierbei wird zunächst ein Vergleichswert mittels eines Skriptes ermittelt.

  Das Skript wird mit einem Parameter-Objekt aufgerufen, welches folgende Eigenschaften enthält:

Für die Validierung stehen verschiedene Vergleichsoperatoren zur Verfügung, mit denen der
zu setzende Attributwert gegen einen anderen Wert geprüft werden kann. Entspricht der neue Wert nicht der definierten Bedingung, so ergibt die Prüfung des Filters ein negatives Ergebnis, sofern die initiale Einstellung *Suchbedingung muss erfüllt sein* ausgewählt ist.

**Schema modifizieren**  

**1.6.5.1 Beispiel: Die Verwendung von Operationsgruppen im Rechtssystem**

In diesem Beispiel wird gezeigt wie Operationsgruppen (Lesen, Erzeugen, Modifizieren, Löschen) bei der Rechtedefinition sinnvoll eingesetzt werden können. Es sollen alle Operationen für den Typ Song und dessen Objekte verboten werden. Dies umfasst die folgenden Aktionen:

- Das Löschen des Objekttyps Song
- Das Löschen von bestimmten Songs (Objekte von Songs)
- Das Löschen von Attributen, welches an einem Song vorkommt
- Das Löschen von Relationen, die an einem Song vorkommt (Relationsziel und -quelle)
- Das Löschen von Erweiterungen, die Objekte von Song erweitern
- Das Löschen von Attribut- und Relationstypen die Objekte oder Untertypen von Song als Definitions bereich haben

Sollen beispielsweise alle Löschen Operationen bei einem Objekt und dem dazugehörigen Typen verboten werden, muss man bei der Auswahl der Operationsparameter im Suchfilter des Rechtes darauf achten alle Löschen Operationen durch die entsprechenden Parameter abzudecken:

![Diagramm](image)

**Der verwendete Suchfilter hat als einzige Bedingung den Objekttyp Song, bei dem die Einstellung Objekte und Untertypen ausgewählt ist. Der Operationsparameter Zugriffselement deckt den Objekttyp Song und alle Objekte, die zu diesem Typ gehören, ab. Der Parameter Kernobjekt deckt die Erweiterungsobjekte ab, die zu Songs gehören. Attribute und Relationen werden durch den Operationsparameter übergeordnetes Element abgedeckt.**

Im Rechtebaum kommt der Operationsfilter der Operation Löschen an erster Stelle. Darauf
folgt der unten abgebildete Suchfilter und als letztes der Entscheider Zugriff verweigert.

Im Beispiel verwendeter Suchfilter: Kernobjekt, übergeordnetes Element und Zugriffselement wurden als Operationsparameter ausgewählt. Die Einstellungen Ein Parameter muss zutreffen und Suchbedingung muss erfüllt sein werden verwendet.

**Erweiterung des Rechtes um Attribut- und Relationstypen**

Ein so definiertes Recht deckt die alle bis auf einen der oben formulierten Anforderungspunkte des Rechtes ab. Lediglich das Löschen von Attribut- und Relationstypen, die für Objekte und Untertypen von Songs definiert sind, wird in dieser Rechtedefinition nicht berücksichtigt.

Eine Erweiterung der Rechtedefinition wird durch den folgenden Filter realisiert:

1.6.6 Testumgebung

Wird im Bereich System der Ordner Rechte ausgewählt, werden im Hauptfenster die Reiter Gespeicherte Testfälle und Konfigurieren angeboten. Der Bereich des Testsystems befindet sich im Reiter Gespeicherte Testfälle. Das Testsystem für Trigger wird über den Bereich System im Ordner Trigger aufgerufen.

Hier können die gespeicherten Testfälle erneut getestet werden. Die Testoberfläche in der die Testfälle definiert werden können, kann über die Schaltfläche Testumgebung öffnen aufgerufen werden.

Zusätzlich zu den Funktionalitäten, die in den folgenden Kapiteln Eine Zugriffssituation testen und Testfälle definieren beschrieben werden, gibt es die Möglichkeit direkt an einem Objekt oder Typ Zugriffsrechte zu testen. Über das Kontextmenü (rechte Maustaste) die Funktion Zugriffsrechte auswählen. Dort stehen die folgenden Menüpunkte zur Auswahl:

- **Objekt**: Es werden alle Operationen (Modifizieren, Löschen, Lesen und im Graph-Editor anzeigen) am Objekt geprüft und deren Ergebnis ausgegeben.

- **Alles**: Es werden alle Operationen (Modifizieren, Löschen, Lesen und im Graph-editor anzeigen) am Objekt und all dessen Eigenschaften (Attribute und Relationen) geprüft.
• **Testumgebung Berechtigungssystem:** Die Testumgebung für die Rechteprüfung wird geöffnet.

1.6.6.1 Eine Zugriffssituation testen

Zum Testen des Rechtssystems und der Trigger-Funktionalität sind zwei Bereiche relevant:

- Die Testumgebung selbst: Die Testumgebung bietet die Möglichkeit für einen bestimmten Testfall die Zugriffsrechte bzw. wann ein Trigger ausgeführt wird zu testen.
- Der Reiter *Gespeicherte Testfälle:* Hier werden die Testfälle aufgelistet und für spätere Überprüfungen zur Verfügung gestellt.

**Anleitung zum Öffnen der Testumgebung**

2. Wenn Sie im Rechtssystem arbeiten, wählen Sie im Hauptfenster den Reiter *Gespeicherte Testfälle* aus.
3. Klicken Sie *Testumgebung öffnen* (rechts unten) an, damit sich die Testumgebung in einem neuen Fenster öffnet.

Die Testumgebung besteht aus mehreren Bereichen: Im oberen Bereich wird der Benutzer und das Element definiert, an dem die Eigenschaft angebracht ist, die geprüft werden soll. Das Element kann ein Objektes, ein Typ oder eine Eigenschaft (wenn diese als Element übergeben wird) sein.


Im Fenster *Operation* kann die Operation ausgewählt werden, die getestet werden soll. Je nach ausgewählten Parametern, ist eine Rechteprüfung möglich oder nicht.

Das Ergebnis der Prüfung wird im unteren Fenster angezeigt. Hierfür muss die Schaltfläche **Überprüfen** ausgewählt werden. Das Ergebnisfenster zeigt alle getesteten Fälle an.

- **Element**: das Objekt, der Typ oder die Eigenschaft an dem oder der die Eigenschaft definiert ist
- **Eigenschaft**: die konkrete Eigenschaft die getestet werden soll (ist leer wenn kursive Eigenschaften getestet werden)
- **Operation**: die Operation, die überprüft werden soll
- **Zugriff erlaubt**: das Ergebnis der Prüfung des Testfalls
- **Entscheidungspfad**: die entsprechenden Ordner, die zu dem Testergebnis führen
- **Zeit**: die Zeit, die für die Rechteprüfung benötigt wurde

Beachte: Bei der Prüfung von Relationen werden i.d.R. die Relation, die inverse Relation und beide Relationshälften einzeln getestet.

### 1.6.6.2 Testfälle definieren

Um die Funktionalität des Rechtesystems zu überwachen, können Testfälle gespeichert werden. Dies ist gerade dann wichtig, wenn Änderungen am Rechtesystem vorgenommen werden und hinterher geprüft werden sollen, ob das neue Ergebnis noch dem erwarteten Ergebnis entspricht. Alle gespeicherten Testfälle werden auf dem Reiter **Gespeicherte Testfälle** angezeigt. Dort können alle Testfälle gleichzeitig geprüft werden.

**Anleitung zur Definition eines Testfalls**

1. Wählen Sie in der Testumgebung das Element und die zu prüfende Eigenschaft aus.
2. Wählen Sie die Operation aus, die getestet werden soll.

4. Wählen Sie in der Ergebnisausgabe den Testfall aus, der gespeichert werden soll. (Es kann immer nur eine Operation als Testfall gespeichert werden.)

5. Betätigen Sie die Schaltfläche Testfall. Der ausgewählte Testfall wird gespeichert und steht für spätere Prüfungen zur Verfügung.

Mehrere Testfälle gleichzeitig testen

![Screenshot mit gespeicherten Testfällen, der zweite Testfall wird in Rot angezeigt.](image)

In Grün werden alle Testfälle angezeigt, deren Testergebnis mit dem erwarteten Testergebnis übereinstimmen. Wird ein Testfall Rot angezeigt, dann ist das Ergebnis der Prüfung ein anderes als das erwartete Testergebnis. Das erwartete Testergebnis wird dadurch bestimmt, dass bei der Definition des Testfalls die Prüfung des Testfalls erstmalig durchgeführt wurde. Das Ergebnis dieser ersten Prüfung wird bei späteren Prüfungen des Testfalls als erwartetes Ergebnis angezeigt. Im Testsystem ist das erwartete Ergebnis entweder Zugriff erlaubt oder Zugriff verweigert; bei Triggern ist das erwartete Ergebnis entweder Skript ausführen oder "nichts passiert" in Form eines Bindestreiches.


1.7 View-Konfiguration

Die View-Konfiguration ermöglicht es, verschiedene Sichten auf die Daten von i-views zu konfigurieren. Die konfigurierten Sichten kommen in Anwendungen zum Einsatz. Es können beispielsweise Teilausschnitte des semantischen Modells gezeigt oder bestimmte Zusam-
menstellungen der Daten (z.B. in Formularen, Tabellen, Ergebnislisten u. v.m.) erstellt werden. 
So können wir u. a. folgende Fragen entscheiden und die entsprechend gewünschten An- 
sichten mit View-Konfigurationen erstellen:

- Wie sollen die Eigenschaften von bestimmten Objekten dargestellt werden?
- In welcher Reihenfolge sollen die Eigenschaften dargestellt werden?
- Wenn wir ein neues Objekt anlegen, welche Attribute und Relationen sollen dann so 
dargestellt werden, damit sie auf keinen Fall übersehen und nicht ausgefüllt werden?
- Wie soll die Liste von Objekten zu einem Typ aussehen?
- Soll es überhaupt eine einfache Liste sein oder sollen die Objekte in Tabellen dargestellt 
dargestellt werden?
- Welche Elemente sollen dann in den einzelnen Spalten zu sehen sein?
- Sollen Relationsziele direkt dargestellt werden? Oder nur bestimmte Attribute?
- Sollen wir verschiedene Reiter definieren, die zusammengehörige Eigenschaften und 
Attribute zusammenfassen?...

Ein Beispiel: Konkrete Personen haben die Eigenschaften Name, Alter, Geschlecht, Adresse, 
Festnetznummer, E-Mail, Mobilnummer, Fax, kennt, ist befreundet mit und ist Kollege von. Nun 
könten wir mithilfe der View-Konfiguration mehr Struktur in die Ansicht der Daten bring-
gen, indem wir einen Reiter mit der Überschrift „Allgemeines“ definieren, der Name, Alter 
und Geschlecht zusammenfasst, einen mit der Überschrift „Kontaktdaten“, der Adresse, Fes-
tnetznummer, E-Mail, Mobilnummer und Fax beinhaltet und einen Reiter mit der Überschrift 
„Kontakte“, der die Eigenschaften kennt, ist befreundet mit und ist Kollege von enthält.

1.7.1 Konzept

Das Konzept von i-views besteht darin, dass intern Wissensnetzelemente zur Konfiguration verwendet werden. Die Ansichten im Knowledge-Builder werden mithilfe einer voreingestellten View-Konfiguration generiert.

View-Konfiguration

Die View-Konfiguration ist dazu vorgesehen, die Daten des Wissensnetzes für die Anwendungen so aufzubereiten, dass sie entweder im Knowledge-Builder oder mithilfe der Bridge in Form einer Anwendung im Web-Frontend dargestellt werden können.
Im Wissensnetz lassen sich daher spezielle "View-Konfigurationen" sowohl für die Verwendung im Knowledge-Builder als auch für Anwendungen wie dem Viewconfiguration-Mapper erstellen.

Damit Änderungen an in der View-Konfiguration für die Anwendung übernommen werden, muss im Knowledge-Builder die View-Konfiguration durch Klick auf den Button "View-Konfiguration Aktualisieren" aktualisiert werden. Dieser Button befindet sich jeweils in der Menüleiste einer View-Konfiguration.

Die View-Konfiguration im Knowledge-Builder enthält folgende Kategorien:

- Anwendungen
- Graph-Konfiguration
- Konfiguration der KB-Ordner-Struktur
- Panel
- Relationszielsuche
- Startansicht (KB)
- Suchfeld (KB)

Näheres ist hierzu im Kapitel "Die Verwendung von View-Konfigurationen" beschrieben.

**Viewkonfiguration-Mapper**

Der Viewkonfiguration-Mapper dient dazu, die vorkonfigurierten Ansichten der View-Konfiguration auf das Web-Frontend des Browsers abzubilden, also zu "mappen".

Die Struktur des Viewkonfiguration-Mappers ist grundsätzlich hierarchisch aufgebaut und enthält die Panels zum Aufbau des Layouts (= Inhaltsanordnung) des Web-Frontends. Zum Anzeigen der Inhalte benötigt ein Panel eine Sub-Konfiguration, die sogenannte "View" (= aufbereiteter Inhalt).

Konkret enthält der Viewkonfiguration-Mapper ein Hauptfensterpanel und beliebig viele
Dialog-Panel. Das Hauptfensterpanel spiegelt den gesamten Darstellungsbereich der Webseite im Web-Frontend wider und enthält beispielsweise folgende Panels:

- Fenstertitelpanel
- Panel mit festgelegter Ansicht
- Panel mit flexibler Ansicht
- Panel mit linearem Layout
- Panel mit wechselndem Layout

Zu beachten ist, dass der Viewkonfiguration-Mapper eine Single-Page-Applikation ist, d. h. es wird nicht die Sichtbarkeit von Panels über mehrere Seiten hinweg gesteuert, sondern die Sichtbarkeit der in fest vorhandenen Panels enthaltenen Elemente.

Menüreiter " Kontext"


**Konzept**

"Kontext" bezieht sich auf die Anwendung einer View-Konfiguration auf bestimmte Wissensnetz-Elemente oder auf die Verwendung innerhalb einer anderen Anwendung der View-Konfiguration:

- "anwenden auf": Wird bei Anwendungen verwendet, um mit einem Anwendungselement die Objekte eines bestimmten Objekttyps des Wissensnetzes darzustellen; Beispiel: Knoten-Kategorie des Net-Navigators stellt Objekte eines bestimmten Objekttyps aus dem Wissensnetz dar
- "anwenden auf Untertypen": Wird gewählt, um auch die Objekte der Untertypen des obigen Objekttyps mit der Anwendung darzustellen
- "anwenden in" spezifiziert, dass eine Konfiguration innerhalb des Kontextes einer anderen Anwendung (meistens: Viewkonfiguration-Mapper) oder einer Konfiguration angewen-
det wird

**Verwendungen**

"Verwendungen" bezieht sich auf die Wieder- und Weiterverwendung einer View-Konfiguration innerhalb einer anderen View-Konfiguration:

- "ist enthalten in Panel": Zeigt an, welche übergeordneten Panels in der Viewkonfigurations-Hierarchie vorhanden sind
- "beinhaltet Panel": Zeigt an, welche Panels in untergeordneten Hierarchiestufen vorhanden sind
- "Reihenfolge": Bestimmt die Reihenfolge des Panels, wenn das übergeordnete Panel ein lineares Layout (horizontal oder vertikal) hat
- "Sub-Konfiguration": Bezieht sich auf eine untergeordnete Konfiguration, welche die View (= konkrete Darstellung des Inhalts) enthält
- "Aktionen aktivieren aus Panel": Zeigt an, dass eine Aktion in diesem Panel durch die Aktion in einem anderen Panel beeinflusst wird (Bsp.: Anzeige des Suchergebnisses in einem Panel wird durch die Sucheingabe in einem anderen Panel beeinflusst)
- "Ergebnis anzeigen aus Aktion": Bestimmt, dass durch die Aktion eines anderen Panels in diesem Panel ein Ergebnis in bestimmter Form angezeigt wird (Bsp.: Net-Navigator zeigt die Elemente zu dem Objekt an, das im Suchergebnis-Feld eines anderen Panels angeklickt wurde)

1.7.2 **Menüs**


Im Folgenden werden die direkten Einstellungsmöglichkeiten an einem Menü und die bereits existierenden Menüarten und deren Verwendung beschrieben.

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Ob die Beschriftung angezeigt wird, richtet sich nach der Menüart und dem Interface, das sich um die Anzeige kümmert.</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Menüarten:**

**Menüleiste**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
</table>

**Anmerkungen**

- Ist der Parameter *Ersetzt Standardmenü* nicht gesetzt, so werden die Aktionen, die in den Menüs enthalten sind, der Reihe nach hinten angefügt.
- Soll die Reihenfolge der Standardaktionen geändert werden, so muss der Parameter *Ersetzt Standardmenü* gesetzt sein. Anschließend können die Standardaktionen mit der Aktion *Standardaktionen hinzufügen* ergänzt werden. Die Standardaktionen können nun beliebig sortiert und mit eigenen Aktionen gemischt werden.

**Kontextmenü**

<table>
<thead>
<tr>
<th>Icon</th>
<th></th>
</tr>
</thead>
</table>
Knowledge-Builder

Derzeit lassen sich Kontextmenüs für eine Tabellenzeile und einen Objekteditor erweitern oder neu definieren.

**Objektkonfiguration:**

In einer beliebigen Top-Konfiguration eines Elementes können unter dem Reiter Menü Menüs angelegt werden. Auch hier kann das Standardmenü durch das Setzen des Parameters Ersetzt Standardmenü ausgeschaltet werden.

**Tabellen-Konfiguration:**

Im Kontextmenü für eine Tabellenzeile gibt es zwei Abschnitte. Der erste bezieht sich auf das ausgewählte Element, der zweite bezieht sich auf die Tabelle. Für die beiden Abschnitte gibt es zwei unterschiedliche Konfigurationsorte. Für den ersten Fall muss das Menü für ein Element mit einer beliebigen, am besten neuen Konfiguration verknüpft werden, die wiederum über anwenden in an die Tabelle, die das Kontextmenü anzeigen soll, gehängt wird. Im zweiten Fall kann das Menü direkt an der Tabelle angebracht werden.

**ViewConfigMapper**

Findet derzeit keine Verwendung im ViewConfigMapper.

**JSON**

```
"label" : "Menü (Kontext)",
"actions" : [{...}],
"type" : "contextMenu"
```

**Liste**

**Icon**
### Knowledge-Builder

Findet nur Anwendung in der Startansicht-Konfiguration. Es werden die konfigurierten Aktionen in einer Liste dargestellt. Werden für die Menüs Beschriftungen vergeben, werden diese mit angezeigt und bieten somit eine Strukturierungsmöglichkeit.

![Diagramm Oberes Menü]

**Menü1**
Beschriftung: Oberes Menü
Mit zwei Aktionen

**Menü2**
Beschriftung: E-Mail
Mit einer Aktion

### ViewConfigMapper

Findet derzeit keine Verwendung im ViewConfigMapper.

### JSON

```json
"label" : "Menü (Liste)",
"actions" : [{...}],
"type" : "listMenu"
```

### Werkzeugliste

<table>
<thead>
<tr>
<th>Icon</th>
<th>Knowledge-Builder</th>
<th>ViewConfigMapper</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon Knowledge-Builder" /></td>
<td>Die Aktionen, die in den Menüs enthalten sind, werden der Reihe nach angefügt. Eine Unterteilung nach Menüs und eine Beschriftung der Menüs werden derzeit nicht berücksichtigt.</td>
<td>Die Aktionen, die in den Menüs enthalten sind, werden der Reihe nach angefügt. Eine Unterteilung nach Menüs und eine Beschriftung der Menüs werden derzeit nicht berücksichtigt.</td>
</tr>
</tbody>
</table>

### JSON

```json
"label" : "Menü (Werkzeugleiste)",
"actions" : [{...}],
"type" : "toolbar"
```
1.7.3 Aktionen

Die Aktionen von i-views sind in vorkonfigurierte Aktionsarten unterteilt. Diese Aktionsarten sind wie folgt kategorisiert:

- Universelle Aktionen (anwendbar in Knowledge und Viewconfiguration-Mapper)
- Knowledge-Builder spezifische Aktionen
- Viewconfiguration-Mapper spezifische Aktionen
- Interne Aktionen (nur für administrativen Gebrauch)

Je nach Aktionsart und Anwendungsfall sind zusätzliche Konfigurationen erforderlich, wie beispielsweise das Anlegen zusätzlicher Panels zur Anzeige der Ergebnisse einer Aktion.

1.7.3.1 Allgemein

Mit Hilfe von Aktionen lassen sich Zusatzfunktionalitäten an View-Konfigurationen anbringen.

Im Knowledge-Builder werden die vollständig konfigurierten Aktionen als zusätzliche Schaltflächen angezeigt. Bei einer Selektion wird das enthaltene Skript ausgeführt.


**Aktion an einer Objektliste**

Die Beschriftung wird als Tooltip im Knowledge-Builder angezeigt. Das ausgewählte Symbol (eine beliebige Bilddatei) wird auf Buttongröße skaliert.


In einer anderen Applikation sind Schaltflächen mit einer Beschriftung und/oder einer Symbolgrafik möglich. Zusätzlich kann ein Tooltip konfiguriert werden.


**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ausführen in View</td>
<td>View in der die Aktion ausgeführt werden soll.</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Benachrichtigung</td>
<td>Text der in einer Benachrichtigung angezeigt wird, die nach der Aktion eingeblendet wird.</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Hier lässt sich eine Beschriftung für die Schaltfläche der Aktion festlegen.</td>
</tr>
<tr>
<td>Ergebnis anzeigen in Panel</td>
<td>Ein Panel in dem das Ergebnis der Aktion angezeigt werden soll.</td>
</tr>
<tr>
<td>Skript für Zielobjekt</td>
<td>.</td>
</tr>
<tr>
<td>Frage vor der Ausführung</td>
<td>Hier lässt sich ein Text angeben, der dem Nutzer vor dem Ausführen der Aktion in einem Dialogfenster angezeigt werden soll. Der Dialog bietet die Möglichkeit die Aktion abzubrechen oder fortzusetzen.</td>
</tr>
<tr>
<td>Kontext von</td>
<td>.</td>
</tr>
<tr>
<td>Nachricht</td>
<td>.</td>
</tr>
<tr>
<td>Panel schließen</td>
<td>Legt fest ob das Panel nach der Aktion geschlossen werden soll.</td>
</tr>
<tr>
<td>Skript</td>
<td>Das Skript das bei dieser Aktion ausgeführt werden soll. Nicht bei allen Aktionsarten verfügbar.</td>
</tr>
<tr>
<td>Skript (enabled)</td>
<td>Hier kann über ein Skript ermittelt werden, ob die Schaltfläche der Aktion aktiviert und damit ausführbar sein soll.</td>
</tr>
<tr>
<td>Skript (visible)</td>
<td>Hier kann über ein Skript ermittelt werden, ob die Schaltfläche der Aktion angezeigt werden soll.</td>
</tr>
<tr>
<td>Skript für Benachrichtigung</td>
<td>Der Inhalt der Benachrichtigung kann hier über ein Skript ermittelt werden.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Die Beschriftung kann hier über ein Skript festgelegt werden.</td>
</tr>
<tr>
<td>Skript für Frage vor der Ausführung</td>
<td>Der Text des Bestätigungs-Dialogs für die Aktion kann hier über ein Skript ermittelt werden. Wird eine leere Zeichenkette zurückgegeben, erscheint der Dialog nicht.</td>
</tr>
<tr>
<td>Skript für Nachricht</td>
<td>.</td>
</tr>
<tr>
<td>Skript für Tooltip</td>
<td>Hier kann über ein Skript der Inhalt des Tooltips der Aktion bestimmt werden, anstatt den Text der Beschriftung zu verwenden.</td>
</tr>
<tr>
<td>Start-Wissensnetzelement von</td>
<td>.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Hier lässt sich ein Symbol auswählen, daß auf der Schaltfläche der Aktion angezeigt werden soll.</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Hier kann der Inhalt des Tooltips der Aktion festgelegt werden, anstatt den Text der Beschriftung zu verwenden.</td>
</tr>
<tr>
<td>Ursprüngliche Position verwenden</td>
<td>.</td>
</tr>
</tbody>
</table>

1.7.3.2 Universell anwendbare Aktionen

1.7.3.2.1 Graphisch-Darstellen-Aktion

Die Aktion "Graphisch darstellen" wird in einer Frontend-Konfiguration dazu verwendet, um Objekttypen, Relationen und Objekte graphisch im Net-Navigator darzustellen. Die Konfiguration sieht dabei folgendermaßen aus:

Hierfür muss unter "Ergebnis anzeigen in Panel" ein Panel angegeben werden, das als Sub-Konfiguration ein Graph-Objekt enthält. Das Graph-Objekt wiederum muss für die Definition der darzustellenden Elemente eine Graph-Konfiguration enthalten:
1.7.3.2.2 Löschen-Aktion
Diese Aktion löscht das jewelige Element. Im Web-Frontend ist diese Aktion unpraktikabel, weil danach das gelöschte Element angezeigt wird - also nichts mehr zu sehen ist. Löschen wird daher im FrontEnd fast immer durch eine Aktion mit Skript realisiert.

1.7.3.2.3 Suchen-Aktion
Diese Aktion löst die Suche aus. Im KB ist diese Funktion als Button in der Menüleiste von Objektlisten integriert (Shortcut Strg + S):

Bei Verwendung für die Konfiguration des Web-Frontends wird die Aktion mittels Dropdown-Auswahl unter dem Eintrag "Aktionsart" einer Aktion zugewiesen:

Tipp: Wird eine Such-Funktion mit Zeichenketten-Eingabe (Stichwortsuche) benötigt, so kann alternativ hierzu die Suchfeld-Ansicht in der Viewkonfiguration verwendet werden. Hier sind Eingabezeile und Suche-Button bereits vorkonfiguriert; das Suchergebnis kann in Kombination mit der Suchergebnis-Ansicht angezeigt werden.

1.7.3.3 Aktionen für den Knowledge-Builder

1.7.3.3.1 Aktualisieren-Aktion
Im KB wird mit dieser Aktion der sichtbare Inhalt von Tabellenzellen neu berechnet. Verfügbare ist diese Option in der Menüleiste von Objektlisten unter dem Button "Aktualisieren" (Shortcut F5).

1.7.3.3.2 Drucken-Aktion
Diese Aktion findet in der Menüleiste von Listenansichten Verwendung. Mit der voreingestellten Konfiguration können Objektlisten ausgedruckt oder in Form einer Excel-Tabelle aus-
gegeben werden, ohne dass dafür ein Export-Mapping angelegt werden muss.

Die Aktion "Drucken" öffnet den Drucken-Dialog im Knowledge-Builder:

Die Drucken-Aktion ist des Weiteren in den Ergebnislisten von Strukturabfragen verfügbar. Für die Konfiguration individueller Ansichten im Knowledge-Builder ist die Aktion für die jeweilige View oder das Konfigurationselement hinzuzufügen:

Voraussetzung für die Anwendbarkeit der Aktionsart "Drucken" ist das Vorhandensein der Drucken-Komponente, welche bei Bedarf mithilfe des Admin-Tools nachinstalliert werden kann.

1.7.3.3 Handbuch-Aktion
Die Aktionsart "Handbuch" ist für die Startansicht des KB verwendbar.
Bei dieser Aktion wird ein Web-Handbuch im Browser geöffnet.

Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Weblink zum Handbuch.</td>
</tr>
</tbody>
</table>

1.7.3.3.4   Homepage-Aktion

Diese Aktionsart ist für die Startansicht des KB verwendbar. Die Homepage wird im Browser geöffnet.
Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Link zu einer Webseite.</td>
</tr>
</tbody>
</table>

1.7.3.5  Im-Baum-Anzeigen-Aktion

Mithilfe der Im-Baum-Anzeigen-Aktion kann die Verortung eines Elementes aus dem Semantischen Netz angezeigt werden. Das Ausführen der Aktion führt dazu, dass zum gewählten Element (bspw. Eintrag einer Listenansicht) die entsprechende Stelle im Strukturaufbau des Organizers (linke Spalte des KB) markiert wird und sich die Detailansicht des Elements öffnet.

1.7.3.6  Suchergebnis-Speichern-Aktion

Werden im Knowledge-Builder Suchen mittels einer Strukturabfrage ausgeführt, so können die Ergebnisse per Klick auf den Button in der Menüleiste abgespeichert werden:

Diese Aktion speichert das Suchergebnis in einem wählbaren Ordner:

1.7.3.3.7 Support-Email-Aktion

Diese Aktionsart ist für die Startansicht des KB verwendbar. Die daraus enthaltene Aktion öffnet einen Dialog, in dem man eine e-mail an die konfigurierte Adresse senden kann.
Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
</table>

Support-E-Mail (spezialisierter Web-Link)

URL: support@i-views.com

Ursprüngliche Position verwenden: ☑️

Attribut oder Relation hinzufügen
1.7.3.3.8 Web-Link-Aktion
Diese Aktionsart ist für die Startansicht des KB verwendbar.
Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Adresse des Weblinks.</td>
</tr>
</tbody>
</table>

1.7.3.3.9 Zuletzt-Verwendete-Objekte-Aktion
Zeigt die zuletzt verwendeten Objekt (Wissensnetzelemente) in der jeweiligen Tabelle an. Je nach Definition der Tabelle werden die Objekte ggf. filtriert.

Diese Aktion ist im KB für Listenansichten vorkonfiguriert und kann mittels Tastenkürzel Strg-R aufgerufen werden.

1.7.3.3.10 Neu-Aktion

<table>
<thead>
<tr>
<th>Knowledge-Builder</th>
<th>ja</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrontEnd</td>
<td>ja</td>
</tr>
</tbody>
</table>

1.7.3.4 Aktionen für den Viewconfiguration-Mapper
1.7.3.4.1 Anzeigen-Aktion
1.7.3.4.2 Auswahl-Aktion

Diese Aktion entspricht der "Anzeigen"-Aktion mit dem einzigen Unterschied, dass die Aktion auf dem Parameter "selectionElement" - also auf einem ausgewählten Element ausgeführt wird. **Achtung:** Dieser Effekt gilt auch für ein ggf. vorhandenes Skript.

Die Aktion "Auswahl" wird ausschließlich (aber nicht zwingend) verwendet, um bei Klick auf einen Tabelleneintrag oder auf einen Listeneintrag aus einem Suchergebnis in einem anderen Panel eine Anzeige hervorzurufen. Eine häufiger Anwendungsfall ist das Anzeigen von Detailinformationen zu einem bestimmten semantischen Element.

Zu beachten ist, dass in der jeweiligen "Auswahl"-Aktion selbst angegeben ist, auf welches Panel sich die Aktion auswirken soll. Dies wird unter "Ergebnis anzeigen in Panel" angegeben.

1.7.3.4.3 Update-Aktion

Die Update-Aktion speichert die Formulardaten aus dem Web-Frontend im Wissensnetz. Das Web-Frontend erkennt die Aktionsart automatisch und schickt sie an die konfigurierte View. Ist keine View als Empfänger der Aktion konfiguriert, versucht das Web-Frontend eine passende View in einem benachbarten Panel zu finden.

Hierzu wird der Aktion in einem Menü die Aktionsart "Speichern" zugewiesen:
Die Update-Aktion kann beispielsweise dazu verwendet werden, um die einzelnen Speichern-Buttons mehrerer Edit-Felder in einem Dialog durch einen individualisierten Speichern-Button zu ersetzen.

1.7.3.4 NN-Expand-Aktion

Bei NN-Expand handelt es sich um eine Aktionsart, die das Aufklappen eines Graph-Knoten im Net-Navigator ermöglicht. D.h. es werden alle Knoten eingeblendet, die über eine Relation mit diesem Knoten verbunden sind und durch die Graph-Konfiguration zugelassen werden. Die betroffenen Relationen zwischen den Knoten werden ebenfalls angezeigt. Knoten, die bereits im Net-Navigator angezeigt wurden, zeigen nur zusätzlich die relevanten Relationen an.


Die Aktion wird in der Graph-Konfiguration an allen Knotenkategorien angebracht, die sie besitzen sollen. Im Reiter "Knoten" wird ein Menü erstellt, das alle NN-Aktionen enthalten kann. In der Aktion selbst muss nur die Aktionsart "NN-Expand" ausgewählt werden, andere

1.7.3.4.5 NN-Hide-Aktion

Mit der Konfiguration dieser Aktionsart wird an den Graph-Knoten ein Menü-Button bereitgestellt, der den ausgewählten Graph-Knoten und dessen angezeigte Relationen einmalig ausblendet (s. durchgestrichenes Auge im Bild). Der Knoten kann beispielsweise mit dem Ausklappen eines anderen verbundenen Knotens wieder angezeigt werden.

1.7.3.4.6 NN-Pin-Aktion


Die Konfiguration der Aktionsart erfolgt wie in der "NN-Expand-Aktion" beschrieben.
1.7.3.5 Interne Aktionen

Der Gebrauch interner Aktionen setzt fachspezifisches Wissen voraus. Bei Unklarheiten hierzu wenden Sie sich an den Support von i-views: support@i-views.com.

Die in diesem Kapitel aufgeführten Aktionen sind lediglich aus Gründen der Vollständigkeit aufgeführt. Hierzu zählen Aktionen wie:

- Einblenden-Aktion
- Sortierung-Aktion
- Springen-Aktion
- Ziel anlegen-Aktion

Skript-Aktion

Dieser Aktionstyp hat keine Auswirkung. Das Vorhandensein eines Skriptes an einer Aktion bewirkt automatisch dessen Ausführung, überschreibt also die eingebaute Funktion der jeweiligen Aktion.

1.7.3.6 Skripte von Aktionen

Skript

Dieses Skript wird ausgeführt, wenn die Aktion ausgeführt wird. Der Rückgabewert wird an das optionale ActionResponse-Skript weitergereicht.

```javascript
function onAction(element, context) {
```
return element;
}

**Argumente**

<table>
<thead>
<tr>
<th>element</th>
<th>Das semantische Element, in dessen Kontext die Aktion ausgeführt wird</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>Weitere vordefinierte Variablen, die den Kontext der Aktion näher beschreiben</td>
</tr>
</tbody>
</table>

Das Skript einer Aktion kann auf folgende vordefinierte Variablen, in `context` enthalten, zugreifen:

**Detaileditor**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectedElement</td>
<td>Ausgewähltes Objekt oder ausgewählter Typ</td>
</tr>
<tr>
<td>type</td>
<td>Objekttyp. Falls das Element ein Typ ist, wird der Typ selbst verwendet</td>
</tr>
</tbody>
</table>

**Objekttipe**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectedElement</td>
<td>Ausgewähltes Objekt oder ausgewählter Typ. Undefined, falls kein Element oder mehrere Elemente ausgewählt wurden.</td>
</tr>
<tr>
<td>selectedElements</td>
<td>Ausgewählte Elemente</td>
</tr>
<tr>
<td>elements</td>
<td>Alle Elemente der Objekttipe</td>
</tr>
<tr>
<td>type</td>
<td>Typ der Objekttipe</td>
</tr>
</tbody>
</table>

**Transaktionen**


Im Knowledge-Builder ist grundsätzlich keine Transaktion aktiv. Das Skript muss Transaktionen selber steuern.

**Knowledge-Builder**
Im Knowledge-Builder steht ein weitere Variable zur Interaktion mit dem Benutzer zur Verfü-
gung:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>ui</td>
<td>Objekt $k.UIObject</td>
</tr>
</tbody>
</table>

Beispielsweise kann eine Meldung anzeigen:

```javascript
ui.alert("Aktuelles Element: " + element.name());
```

**Skript (ActionResponse)**

Dieses Skript wird nach der Ausführung der Aktion ausgeführt. Hauptaufgabe ist es, das
Ergebnis der Aktion für den ViewConfigMapper (oder andere Frontends) aufzubereiten. Das
Skript muss ein Objekt vom Typ $k.ActionResponse liefern.

```javascript
function actionResponse(element, context, actionResult) {
  var actionResponse = new $k.ActionResponse();
  actionResponse.setData(actionResult);
  actionResponse.setFollowup("new");
  actionResponse.setNotification("Erledigt", "warn");
  return actionResponse;
}
```

**Argumente**

<table>
<thead>
<tr>
<th>element</th>
<th>Das semantische Element, in dessen Kontext die Aktion ausgeführt wird</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>Weitere vordefinierte Variablen, die dem Kontext der Aktion näher beschreiben (siehe vorherigen Abschnitt)</td>
</tr>
<tr>
<td>action-Result</td>
<td>Der Rückgabewert des onAction-Skripts bzw. falls nicht definiert der Rückgabewert der konfigurierten Aktionsart.</td>
</tr>
</tbody>
</table>

**ActionResponse**


Im Knowledge-Builder sind folgende Werte von Followup in Tabellen möglich:
Rendert die aktuelle Tabelle neu, ohne die Liste neu zu berechnen

Berechnet die Tabelle neu

Selektiert das Element in data in der Tabelle an. Alternativ kann in data ein Objekt mit "element": actionResult, "viewMode": "edit" das Ergebnis in einem neuen Detaileditor geöffnet werden.

In Detail-Editoren wird Followup nicht ausgewertet.

**Skript (visible)**

```javascript
function actionVisible(element, context)
{
    return true;
}
```

Anhand des Rückgabewertes wird entschieden, ob der Knopf angezeigt werden soll oder nicht.

In Tabellen wird bei Aktionen auf den Elementen folgende Funktion aufgerufen, die einen Array von Elementen übergibt und einen Array von booleschen Werten erwartet. Dies kann dazu verwendet werden, die Sichtbarkeit für die Elemente effizienter am Stück zu berechnen.

```javascript
function actionsEnabled(elements, contexts)
{
    return elements.map(function (element, index) {
        return actionEnabled(element, contexts[index]);
    });
}
```

**Skript (enabled)**

```javascript
function actionEnabled(element, context)
{
    return true;
}
```

Anhand des Rückgabewertes wird entschieden, ob der Knopf aktiv ist.

In Tabellen wird bei Aktionen auf den Elementen folgende Funktion aufgerufen, die einen Array von Elementen übergibt und einen Array von booleschen Werten erwartet:

```javascript
function actionsVisible(elements, contexts)
{
    return elements.map(function (element, index) {
        return actionVisible(element, contexts[index]);
    });
}
**UI-spezifische Funktionen**

Das die Aktion realisierende Skript kann im Knowledge-Builder über `context.ui` auf UI-spezifische Funktionen zurückgreifen.

UI-Funktionen sollten nach Möglichkeit nicht innerhalb von Transaktionen ausgeführt werden, da sich die Anzeige innerhalb der Transaktion nicht aktualisiert.

```javascript
context.ui.alert(message, windowTitle)

Zeigt eine Meldung an.

context.ui.requestString(message, windowTitle)

Benutzer kann eine Zeichenkette eingeben.

context.ui.confirm(message, windowTitle)

Öffnet einen Abbrechen-Dialog.

context.ui.choose(objects, message, windowTitle, stringFunction)

Objekt aus einer Menge auswählen lassen.

context.ui.openEditor(element)

Standardeditor für das Objekt öffnen.

context.ui.notificationDialog(notificationFunction, parameters, windowTitle)

Es wird ein Warte- bzw. Benachrichtigungsdialog geöffnet. Dieser kann, je nachdem wie er konfiguriert ist, abgebrochen werden.

Mögliche Parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Beschreibung</th>
<th>Standardwert</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoExpand</td>
<td>Ist der Anzeigebereich des Dialogs initial geöffnet.</td>
<td>true</td>
</tr>
<tr>
<td>canCancel</td>
<td>Kann der Dialog abgebrochen werden.</td>
<td>true</td>
</tr>
<tr>
<td>stayOpen</td>
<td>Bleibt der Dialog nach Beendigung der Funktion geöffnet.</td>
<td>true</td>
</tr>
</tbody>
</table>

Beispiel:

```javascript
ui.notificationDialog(
```
function() {
    ui.raiseNotification("start");
    for (var i = 0; i < 10; i++)
        ui.raiseNotification("" + i + "*" + i + "+" + (i*i));
    ui.raiseNotification("end");
    return undefined;
}

{ "canCancel" : false },
"Ein Wartedialog"

Mit der folgenden Function raiseNotification können Meldungen auf dem Anzeigebereich ausgegeben werden.

$k.UI.raiseNotification(message)

Diese Benachrichtigung wird nur von der Function notificationDialog gefangen und die Nachricht wird nur dort im Anzeigebereich ausgegeben.

### 1.7.4 View-Konfigurationselemente

Eine Viewkonfiguration beschreibt, wie Objekte oder Typen dargestellt werden sollen. Im Folgenden werden die verschiedenen Elementarten, die der View-Konfiguration zur Verfügung stehen, beschrieben.

Die einzelnen Viewkonfigurationselemente lassen sich teilweise beliebig zusammenstecken. Ebenfalls können die Konfigurationen mehrfach als Unterkonfiguration verwendet werden.

**Liste der verschiedenen Detailkonfigurationstypen**

<table>
<thead>
<tr>
<th>Konfigurationstyp</th>
<th>Top-Level-Konfiguration</th>
<th>Kann folgende Unterkonfiguration enthalten</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative</td>
<td>x</td>
<td>beliebig</td>
</tr>
<tr>
<td>Eigenschaft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eigenschaften</td>
<td>x</td>
<td>Eigenschaft</td>
</tr>
<tr>
<td>Gruppe</td>
<td>x</td>
<td>beliebig</td>
</tr>
<tr>
<td>Hierarchie</td>
<td>x</td>
<td>beliebig</td>
</tr>
<tr>
<td>Skriptgenerierter Inhalt</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Statischer Text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suche</td>
<td></td>
<td>Tabelle</td>
</tr>
</tbody>
</table>

Einstellungsmöglichkeiten, die alle Detailkonfigurationstypen gemeinsam haben
<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Konfigurationsname</td>
<td>Findet keine Verwendung im Userinterface. Der Ersteller einer Konfiguration hat hier die Möglichkeit einen für ihn verständlichen Namen zu vergeben, um diese Konfiguration später besser wiederfinden und in anderen Konfigurationen wieder verwenden zu können.</td>
</tr>
<tr>
<td>Skript für Fenstertitel</td>
<td>Nur zur Verwendung im Knowledge-Builder. Öffnet man ein Objekt beispielsweise per Doppelklick in der Objektliste, öffnet sich ein Fenster mit den Eigenschaften dieses Objektes. Der Titel dieses Fensters kann durch ein Skript bestimmt werden.</td>
</tr>
</tbody>
</table>

**Anmerkung:** In den folgenden Abschnitten werden die Einstellungsmöglichkeiten für die einzelnen Konfigurationstypen beschrieben. Die obligatorischen Parameter sind fett gedruckt.

1.7.4.1 Alternative


**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. einer weiteren Alternative eingebettet ist.</td>
</tr>
<tr>
<td>hat Default-Alternative</td>
<td>Die Unteransicht, die initial selektiert sein soll, kann hier festgelegt werden.</td>
</tr>
</tbody>
</table>

**Darstellung in einer Anwendung**

Werden die Ansichten in JSON herausgeschrieben, werden die einzelnen Unteransichten in
einem ARRAY an den KEY alternatives gehängt.

Darstellung im Knowledge-Builder

Im Knowledge-Builder werden dem Anwender die verschieden konfigurierten und mit der Alternative verknüpften Ansichten eines Objekts durch Reiter zur Verfügung gestellt.

Anmerkung
Diese Konfiguration ist ähnlich der Konfiguration *Gruppe*. Allerdings werden bei einer *Gruppe* alle Unteransichten gleichzeitig angezeigt.

### 1.7.4.2 Gruppen


#### Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. <em>Alternative</em> eingebettet ist.</td>
</tr>
</tbody>
</table>

#### Darstellung in einer Anwendung

Wird die Ansicht in JSON herausgeschrieben, werden die einzelnen Unteransichten in einem ARRAY an den KEY *group* angehängt.

**Darstellung im Knowledge-Builder**

Im Knowledge-Builder wird um eine Gruppe ein Rahmen gezeichnet. Die Ansichten der Unterkonfigurationen werden dann in diesem Rahmen angezeigt.
Eine Gruppe mit folgenden Unterkonfigurationen: der Eigenschaftsliste "Bild und Text", der Eigenschaftsliste "Eigenschaften" und der Suche "Ähnliche Sehenswürdigkeiten"

1.7.4.3 Hierarchie

Der Konfigurationstyp "Hierarchie" stellt Elemente eines semantischen Modells hierarchisch in einer Baumstruktur dar, in der einzelne Äste auf- und zugeklappt werden können.


**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschreibung</td>
<td>Eine Beschreibung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
<tr>
<td>Skript für Beschreibung</td>
<td>Es ist auch möglich durch ein Skript eine Beschreibung festzulegen.</td>
</tr>
<tr>
<td>Banner der Hierarchiewurzel anzeigen</td>
<td>Betrifft nur den Knowledge-Builder: Banner wird angezeigt.</td>
</tr>
<tr>
<td>Aktion (Auswahl)</td>
<td>Verweis auf eine Aktion, die beim Anklicken eines Hierarchie-Elements aufgerufen wird.</td>
</tr>
<tr>
<td>Detailansicht ausblenden</td>
<td>Standardmäßig wird die Detailansicht eines ausgewählten Objektes angezeigt (Knowledge-Builder) oder ausgegeben (json, als subview). Durch Aktivieren dieser Option wird keine Detailansicht angezeigt bzw. ausgegeben.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td></td>
</tr>
<tr>
<td>Unterelemente erzeugen ohne Frage nach Namen</td>
<td>Wenn neue Unterelemente in der Hierarchie erzeugt werden, wird standardmäßig gefragt, wie ihr Name lauten soll. Ein Häkchen hier, erzeugt ohne Frage nach Namen namenlose Objekte.</td>
</tr>
</tbody>
</table>

Einstellungsmöglichkeiten für die Sortierung

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absteigend sortieren</td>
<td>Steuert, ob auf- oder absteigend sortiert wird. Ist dieser Parameter nicht gesetzt, wird aufsteigend sortiert.</td>
</tr>
</tbody>
</table>
| Primäres Sorterkriterium      | Auswahlmöglichkeit für das Kriterium, nach dem die Eigenschaften sortiert werden:  
  - **Position**: Die in der Konfiguration festgelegte Reihenfolge wird verwendet (Default).  
  - **Wert**: Inhalt des Attributes bzw. Anzeigename des Relationszieles wird verwendet.  
  - **Skript für Sortierung**: Das in dem Attribut Skript zur Sortierung hinterlegte Skript wird zur Ermittlung des Sorterkriteriums verwendet. |
| Sekundäres Sorterkriterium um | Sorterkriterium für Eigenschaften, die für das primäre Sorterkriterium den gleichen Wert haben. Einstellmöglichkeiten analog zum Primären Sorterkriterium. |
| Skript für Sortierung          | Verweis auf ein registriertes Skript, das den Sortierschlüssel für das primäre bzw. sekundäre Sorterkriterium zurückgibt. |

Ermittlungsmöglichkeiten der hierarchiebildenden Elemente

<table>
<thead>
<tr>
<th>Name</th>
<th>Ermittlung von ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation (absteigend)</td>
<td>Unterelementen</td>
</tr>
<tr>
<td>Relation (aufsteigend)</td>
<td>Oberelementen</td>
</tr>
<tr>
<td>Strukturabfrage (absteigend)</td>
<td>Unterelementen</td>
</tr>
<tr>
<td>Strukturabfrage (aufsteigend)</td>
<td>Oberelementen</td>
</tr>
<tr>
<td>Skript (absteigend)</td>
<td>Unterelementen</td>
</tr>
<tr>
<td>Skript (aufsteigend)</td>
<td>Oberelementen</td>
</tr>
</tbody>
</table>
Aktionen und Styles
Es lassen sich sowohl für die gesamte Hierarchie als auch für die einzelnen Knoten Aktionen und Styles anbringen. Ab Version 5.2 kann man auch automatisch Style-Klassen über ein Skript zuweisen lassen.

Darstellung in einer Anwendung

Erst ab Version 4.1 gibt es die JSON-Repräsentation einer Konfiguration vom Typ *Hierarchie*.

Darstellung im Knowledge-Builder
In der Detail-Anzeige eines Elements wird im linken Bereich eine Hierarchie eingeblendet. Im rechten Bereich wird das Element mit einer View-Konfiguration ohne Hierarchie angezeigt. Diese View-Konfiguration muss eigens definiert werden und unter *Verwendung > anwenden in* muss der Konfigurationsname der Hierarchie angegeben werden. Die Subkonfiguration lässt sich alternativ auch direkt an der Hierarchie unter *Subkonfiguration* angeben.

Anmerkung

- Die Werte aller Eigenschaften, die für die Hierarchiebildung ausgefüllt werden können, sind Relationen.
- Die einzelnen Attribute wie z.B. Relation *absteigend* können mehrfach vergeben werden.
- Für jeden Attributtyp werden die Relation oder Relationen ermittelt und aufgesammelt.
Sind verschiedene Attributtypen angegeben, wird mit den Teilmengen eine Schnittmenge gebildet.

**Beispiel - Anwendungsfall**

Typischerweise werden Hierarchien verwendet, um Ober-/Unterthema-Relationen oder Teilv-Rrelationen darzustellen.

1. Hierarchiebildende Relation

Die direkteste Variante. Die Relationen, die die Hierarchie bilden, werden eingetragen.

<table>
<thead>
<tr>
<th>Relation (aufsteigend)</th>
<th>[hat Oberthema]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation (absteigend)</td>
<td>[hat Unterthema]</td>
</tr>
</tbody>
</table>

2. Hierarchiebildende Strukturabfrage

Die Relationen lassen sich ebenfalls über eine Strukturabfrage ermitteln.

3. Hierarchiebildendes Skript

Auch durch ein Skript lassen sich die möglichen hierarchiebildenden Relationen aufsammeln. Es bekommt das aktuelle Element als Parameter übergeben und muss eine Menge an Relationen zurückgeben. Stattd auf Relationen kann man aber auch auf Elementen arbeiten.

**Skript hat Oberthema**

```javascript
function relationsOf(element) {
    return element.relations('hatOberthema');
}
function targetsOf(element) {
    return element.relationTargets('hatOberthema');
}
```

1.7.4.4 Eigenschaften


**Einstellungsmöglichkeiten**
<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Anzeigename der Sammlung von Eigenschaften. Ist keine Beschriftung angegeben wird im Knowledge-Builder die Zeichenkette 'Eigenschaften' verwendet.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ kann der Anzeigename auch über ein Skript ermittelt werden.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>.</td>
</tr>
</tbody>
</table>

### Einstellungsmöglichkeiten für die Sortierung

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absteigend sortieren</td>
<td>Steuert, ob auf- oder absteigend sortiert wird. Ist dieser Parameter nicht gesetzt, wird aufsteigend sortiert.</td>
</tr>
</tbody>
</table>
| Primäres Sortierkriterium  | Auswahlmöglichkeit für das Kriterium, nach dem die Eigenschaften sortiert werden:  
  • *Position*: Die in der Konfiguration festgelegte Reihenfolge wird verwendet (Default).  
  • *Wert*: Inhalt des Attributes bzw. Anzeigename des Relationszieles wird verwendet.  
  • *Skript zur Sortierung*: Das in dem Attribut Skript zur Sortierung hinterlegte Skript wird zur Ermittlung des Sortierkriteriums verwendet.                                                      |
| Sekundäres Sortierkriteri-um| Sortierkriterium für Eigenschaften, die für das primäre Sortierkriterium den gleichen Wert haben. Einstellmöglichkeiten analog zum *Primären Sortierkriterium*                                                                 |
| Skript zur Sortierung       | Verweis auf ein registriertes Skript, das den Sortierschlüssel für das primäre bzw. sekundäre Sortierkriterium zurückgibt.                                                                                     |

### Darstellung in Anwendungen

Die Ansichten der Unterkonfiguration einzelner Eigenschaften werden in einem ARRAY abgelegt und mit dem KEY `properties` angehängt.
Darstellung im Knowledge-Builder

Die in der Konfiguration eingestellte Beschriftung wird prominent angezeigt. Ihm folgen die Ansichten der konfigurierten Eigenschaften.

Anmerkung

Meta-Eigenschaften werden mit dem gleichen Vorgehen angehängt.

1.7.4.5 Eigenschaft

Mit der View-Konfiguration Eigenschaft können einzelne Attribute oder Relationen deﬁniert werden, die in einer Eigenschafts-Liste angezeigt werden sollen. Es kann auch eine abstrakte Eigenschaft benutzt werden, die eine Menge von Eigenschaften zusammenfasst.

Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absteigend sortieren</td>
<td>Steuert, ob die Eigenschaften auf- oder absteigend nach ihrem Namen sortiert werden. Ist dieser Parameter nicht gesetzt, wird aufsteigend sortiert.</td>
</tr>
</tbody>
</table>
### Anzeigeart

<table>
<thead>
<tr>
<th>Steht in zwei Fällen zur Verfügung:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> Die Eigenschaft ist eine Relation:</td>
</tr>
<tr>
<td>Auswahlmöglichkeit für die Darstellung derBeschriftung eines Relationsziels. Diese Einstellungsmöglichkeit steht nur dann zur Verfügung, wenn die Einstellung Relationszielansicht den Wert Auswahl oder Relationsstruktur hat.</td>
</tr>
<tr>
<td><strong>2.</strong> Die Eigenschaft ist ein Datei-Attribut:</td>
</tr>
<tr>
<td>Auswahlmöglichkeit für die Darstellung des Wertes eines Datei-Attributs.</td>
</tr>
</tbody>
</table>

### Auswahlmöglichkeiten:

- **Symbol (topicIcon):** Icon des Relationsziels/die Datei als Icon
- **Symbol und Zeichenkette**
- **Zeichenkette (Namensattribut):** Name des Relationsziels/Dateiname der Datei

### Beschriftung

| Anzeigenname der Eigenschaft. Ist keine Beschriftung angegeben wird der Name des Eigenschaftstyps ausgegeben. |

### Eigenschaft

| Verknüpfung zu einem Eigenschaftstyp, der angezeigt werden soll. |

### Skript für virtuelle Eigenschaften

| Alternativ zu 'Eigenschaft': Skript zur Berechnung anzuzeigender Werte |

### Einblendung des Relationsziels

| Steht nur bei Relationen zur Verfügung. Standardmäßig wird lediglich der Name des Relationsziels angezeigt. Wird der Name angeklickt, öffnet sich das Relationssziel in einem weiteren Editor. Wird hingegen die Option **Einblendung des Relationsziels** gewählt, werden die Relationsziele direkt angezeigt, d.h. nicht nur ihr Name, sondern alle ihre Eigenschaften. |

### Einblendung zusätzlicher Eigenschaften


<table>
<thead>
<tr>
<th>hat Oberthema</th>
<th>Kunst</th>
</tr>
</thead>
<tbody>
<tr>
<td>hat Oberthema</td>
<td>Persische Architektur</td>
</tr>
<tr>
<td>hat Oberthema</td>
<td></td>
</tr>
</tbody>
</table>
| Einblendung zusätzlicher neuer Eigenschaften | Nur bei der Ansicht zum Bearbeiten von Objekten relevant: Ist diese Option gesetzt, wird die Eigenschaft nur dann eingeblendet, wenn die Eigenschaft noch nicht angelegt wurde. So kann sie bequem und schnell ausgefüllt werden und wird nicht so leicht vergessen.  
- Name  
- Alternativname/Synonym |
| --- | --- |
- Name  
- Bevölkerung |
| Konfiguration für eingebettete Metaeigenschaften | Angabe der Konfiguration, die verwendet werden soll, um Metaeigenschaften anzuzeigen. Die Metaeigenschaften werden eingebettet, d.h. hinter dem Wert der Eigenschaft angezeigt. Der Name des Eigenschaftstyps wird nicht angezeigt.  
- Name  
- Bevölkerung  
- Jahr zu Bevölkerung |
| Konfiguration für Metaeigenschaften | Angabe der Konfiguration, die verwendet werden soll, um Metaeigenschaften anzuzeigen. Die Metaeigenschaften werden unter dem Wert der Eigenschaft angezeigt. Für die Anzeige am Frontend müssen die Eigenschaften mit den Metas auf "initial ausgeklappt" eingestellt sein.  
- Name  
- Bevölkerung  
- Jahr zu Bevölkerung |
| Relationszielansicht | Wird als Eigenschaft eine Relation gewählt, kann mit diesem Parameter die Ansicht der Relationsziele definiert werden:  
- **Auswahl**: Alle Relationsziele werden aufgelistet und mit einer vorangestellten Checkbox angezeigt. Bei bestehenden Relationen ist die CheckBox mit einem Häkchen versehen.  
- **Drop Down**: Diese Einstellung ist nur sinnvoll, wenn die Relation nur einmal vorkommen darf. Es wird eine Drop Down-Liste mit allen möglichen Relationszielen zur Auswahl angezeigt.  
- **Relationsstruktur**: Alle Relationsziele werden im linken Bereich aufgelistet, ähnlich einer Hierarchie. Im rechten Bereich ist dann die Detailansicht des selektierten Relationsziels zu sehen. Diese Ansicht kommt nur zur Geltung, wenn die Konfiguration direkt einer Top-Level-Konfiguration untergeordnet ist.  
- **Tabelle**: Tabellenansicht der Relationen. Die Tabellenansicht kann nicht im Knowledge-Builder angewendet werden. Für die Tabellenansicht muss die Einstellung Tabelle ausgefüllt werden.  
- **Tabelle (Relationsziele)**: Tabellenansicht der Relationsziele. Diese Tabelle kann im Knowledge-Builder angewendet werden. |

| Skript für Beschriftung | Die Beschriftung kann durch ein hier angegebenes Skript ermittelt werden. |

| Skript für Relationsziel-bezeichner | Über dieses Skript kann für Relationen gesteuert werden, wie das Relationszeil dargestellt wird. Wenn kein Skript eingetragen ist, wird der Objektname des Relationsziels zur Darstellung verwendet.  
Anwendungsbeispiel: Eine Person gehört zu einer Abteilung mit Namen 'Abt. IV'. Durch ein passendes Skript kann erreicht werden, dass in der Anzeige bei der Person statt 'Abt. IV' die Angabe 'Stadtverwaltung Darmstadt, Abt. IV' erscheint. |

| Skript für Sortierung | Anhand des Skriptes wird ein Wert, nach dem sortiert wird, ermittelt. Siehe Beispiel unten. |


Beispiel für Skript für Sortierung
Vorbedingung: An allen Eigenschaften kann das Attribut sortKey angebracht werden.

```javascript
function sortKey(element) {
    if (element instanceof $k.Property) {
        var attribute = element.attribute('sortKey')
        if (attribute) {
            return attribute.value();
        }
    }
    if (element instanceof $k.Domain) {
        var attribute = element.type().attribute('sortKey');
        if (attribute) {
            return attribute.value();
        }
    }
    return undefined;
}
```

Eigenschaftsanzeige einer Person

<table>
<thead>
<tr>
<th>Eigenschaften</th>
<th>ординierter Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vorname</td>
<td>Herlinde</td>
</tr>
<tr>
<td>sortKey</td>
<td>1</td>
</tr>
<tr>
<td>Vorname</td>
<td>Trude</td>
</tr>
<tr>
<td>sortKey</td>
<td>2</td>
</tr>
<tr>
<td>Name</td>
<td>Neumayer</td>
</tr>
<tr>
<td>sortKey</td>
<td>3</td>
</tr>
<tr>
<td>Vorname</td>
<td></td>
</tr>
</tbody>
</table>

Anmerkung: Beim Attributtyp Name ist für sortKey der Wert 3 eingetragen, daher steht dieser vorläufige Wert am Ende der Liste.

1.7.4.6 Edit

Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
</table>
1.7.4.7 Tabelle

Tabellen können als Unterkonfiguration für die Ergebnisanzeige von Abfragen des Konfigurationstyps "Suche" oder als eigenständige Konfiguration zur Darstellung der Objektlisten im Knowledge-Builder verwendet werden.


<table>
<thead>
<tr>
<th>Beschreibung</th>
<th>Eine Beschreibung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editiermodus umschaltbar</td>
<td>Wird diese Option ausgewählt, werden die Eigenschaften zunächst nur als normale Liste angezeigt. Zusätzlich wird jedoch ein Schalter angeboten, mit dem man zwischen der normalen und der Editieransicht umschalten kann.</td>
</tr>
<tr>
<td>Nur benutzerdefinierte Schaltflächen</td>
<td>Ist diese Option gesetzt, wird der Speichern-Knopf nicht angezeigt.</td>
</tr>
</tbody>
</table>

1.7.4.7 Tabelle

Tabellen können als Unterkonfiguration für die Ergebnisanzeige von Abfragen des Konfigurationstyps "Suche" oder als eigenständige Konfiguration zur Darstellung der Objektlisten im Knowledge-Builder verwendet werden.


Die hierarchische Darstellung aller Unterconfigurationselemente der Tabellenconfiguration weist eine Menü-Zeile auf, die wie folgt mit Aktionen belegt ist:

- Neues Unterelement anlegen und verknüpfen.
- Bereits vorhandene mögliche Unterelemente durchsuchen und verknüpfen
- Verknüpfung wieder löschen. Das Unterelement bleibt dabei als Objekt erhalten und kann in anderen Konfigurationen wieder verwendet werden.
- Gewähltes Unterelement komplett löschen. Falls es in anderen Konfigurationen verwendet wird, öffnet sich vor der Löschung eine Warnung, die alle vorhandenen Verknüpfungen aufzeigt.
- Gewähltes Unterelement in der Liste nach oben schieben.
- Gewähltes Unterelement in der Liste nach unten schieben.
- Aktualisierung. Erst nach einer Aktualisierung werden alle Änderungen für die Anwendungen übernommen.

Beispiel einer einfachen Tabellenconfiguration
Zu einer Liste von Sehenswürdigkeiten soll der Ort, an dem sie sich befinden, Themen, mit denen sie zu tun haben und Reisen, bei denen sie besucht werden in einer Tabelle zu sehen sein. Nicht zu vergessen ist das Namensattribut, durch das die Sehenswürdigkeiten in der ersten Spalte repräsentiert werden.
Einstellungs möglichkeiten (Tabelle)

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anzahl Zeilen (Page size)</td>
<td>Gibt an, wie viele Zeilen auf einer Seite angezeigt werden sollen. Standardwert: 20</td>
</tr>
<tr>
<td>Keine automatische Suche</td>
<td>Es wird keine automatische Suche durchgeführt.</td>
</tr>
<tr>
<td>Namenlose Unterelemente</td>
<td>Namenlose Unterelemente werden ebenfalls angezeigt.</td>
</tr>
</tbody>
</table>
| Reihenfolge                   | Durch Angabe einer Ganzzahl lässt sich steuern an welcher Stelle, falls mehreren Konfigurationen vom Typ *Tabelle* angezeigt werden sollen, die aktuelle Konfiguration angezeigt wird. Die Sortierung wird nach zwei Kriterien durchgeführt, die in der folgenden Reihenfolge überprüft werden:  
1. Attribut *Reihenfolge* vorhanden, wenn ja, dann wird dieses als Sortierkriterium verwendet, wenn nein, werden erst die Konfigurationen für Typen und dann die für Objekte angezeigt.  
2. Sortierung nach Anzeigename |
| Aktionen und Styles           | Aktionen und Styles lassen sich für die gesamte Tabelle, aber auch für Zeilen festlegen. |
| Verwendung                    | Wo die Tabelle zur Verwendung kommt, wird auf dem Reiter Verwendung angegeben.  
Unter *anwenden auf* wird der Objekttyp angegeben, auf den die Tabelle angewendet werden soll. Tabellen können in anderen View-Konfigurationen wieder verwendet werden. Falls die Tabelle Baustein einer anderen View-Konfiguration ist, wird dies unter *inverse* anwenden in angezeigt.  
Die Eigenschaft *anwenden in* verweist auf eine Anwendung. Mehrere Verknüpfungen sind möglich. |
Beispiele:

- Soll die Tabelle im Knowledge-Builder rechts im Hauptfenster bei der Navigation durch die Ordnerstruktur verwendet werden, dann muss die Tabellenkonfiguration mit dem entsprechenden Ordnerstrukturerelement verknüpft sein.

- Sollen mögliche Relationsziele im Knowledge-Builder tabellarisch dargestellt werden, dann muss die Tabelle mit der Anwendung Knowledge-Builder verknüpft sein.

Tabellen / Objektslisten im Knowledge-Builder


Objekte vom Typ Theme bieten selbst als zusätzliche Eigenschaften Synonym und die Relation ist Theme von an. Diese Eigenschaften, zusammen mit dem Namen des Themes sollen die Bestandteile der zu bauenden Tabelle ergeben.

Der Typ Theme wird bearbeitet, in der Ansicht wird der Reiter Details gewählt und hier der Punkt View-Konfiguration -> Objekt -> Objekliste angeklickt.

Noch wurde keine Tabellenkonfiguration mit diesem Typ verknüpft. Durch Klicken auf den Knopf Neu wird eine neue, leere Konfiguration erzeugt. Diese kann dann selektiert und wie oben beschrieben bearbeitet werden.

Einstellungsmöglichkeiten für die Sortierung

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absteigend sortieren</td>
<td>Steuert, ob auf- oder absteigend sortiert wird. Ist dieser Parameter nicht gesetzt, wird aufsteigend sortiert.</td>
</tr>
</tbody>
</table>
Primäres Sorterkriterium | Auswahlmöglichkeit für das Kriterium, nach dem die Eigenschaften sortiert werden:
- **Position**: Die in der Konfiguration festgelegte Reihenfolge wird verwendet (Default).
- **Wert**: Inhalt des Attributes bzw. Anzeigenname des Relationszieles wird verwendet.
- **Skript zur Sortierung**: Das in dem Attribut Skript zur Sortierung hinterlegte Skript wird zur Ermittlung des Sortierkriteriums verwendet.

Sekundäres Sorterkriterium | Sortierkriterium für Eigenschaften, die für das primäre Sorterkriterium den gleichen Wert haben. Einstellmöglichkeiten analog zum Primären Sorterkriterium

Skript zur Sortierung | Verweis auf ein registriertes Skript, das den Sortierschlüssel für das primäre bzw. sekundäre Sorterkriterium zurückgibt.

### 1.7.4.7.1 Spaltenkonfigurationen

Wie bereits erwähnt, tragen Spaltenkonfigurationen Eigenschaften, die der Festlegung der Darstellung und des Verhaltens der Spalte in der Tabelle dienen. Erst wenn Eigenschaften an den in der Spaltenkonfiguration enthaltenen Spaltenelementen konfiguriert werden, wird die Spalte angezeigt.

**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Beschriftung</strong></td>
<td>Wird in der Titelzeile der Spalte angezeigt. Hierbei ist zu beachten, dass <strong>Beschriftung</strong> der Anzeige in der Tabelle diert, die Spaltenkonfiguration aber zusätzlich noch das Attribut <strong>Konfigurationsname</strong> enthält. Dieser Name dient allein der Verwaltung und dem Auffinden der Konfiguration in der semantischen Graph-Datenbank und wird nicht angezeigt oder ausgegeben.</td>
</tr>
<tr>
<td>Breite der Spalte (%)</td>
<td>Für die Breite der Spalte wird hier ein prozentualer Wert erwartet (für 60% muss also “60” eingegeben werden).</td>
</tr>
<tr>
<td>Nicht anzeigen</td>
<td>Ist dieser Wert gesetzt, wird die komplette Spalte nicht angezeigt. Dies dient dazu z.B. eine Suche mit den Werten dieser Spalte durchzuführen. Der Anwender soll aber nicht die Möglichkeit haben Veränderungen an dieser Spalte vorzunehmen.</td>
</tr>
<tr>
<td>Obligatorisch für Abfrage</td>
<td>Ist dieser Wert gesetzt, muss die Spalte ausgefüllt sein, um suchen zu dürfen.</td>
</tr>
</tbody>
</table>
Sortierpriorität | Nach der Spalte mit Sortierpriorität 1 wird primär sortiert. Bei gleichen Werten wird nach der Spalte mit der nächst höheren Sortierpriorität sortiert usw. Die Standardsortierreihenfolge ist aufsteigend. Um absteigend zu sortieren negiert man die Sortierpriorität oder setzt das Meta-Attribut "Absteigend sortieren".

Standard-Operator | Operator, der initial bei der Suche für einen Suchtext angewendet wird.

Suchtext | Initial kann eine Spalte mit einem Suchtext versehen werden.

### Beispiel

![Spaltenkonfiguration für die Spalte Name](image)

#### 1.7.4.7.2 Spaltenelemente

Ein Spaltenelement dient der Zuweisung, welche Inhalte eine Tabellenspalte darstellen soll und wie dies zu geschehen hat. Es können entweder an den semantischen Objekten definierte Eigenschaften wie Attribute und Relationen spezifiziert oder Strukturabfrage-Bausteine oder Skript-Bausteine verwendet werden.

#### Einstellungsmöglichkeiten
<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenschaft (obligatorisch oder Skript)</td>
<td>Verknüpfung zu einem Eigenschaftstyp, der angezeigt werden soll.</td>
</tr>
<tr>
<td>Skript (obligatorisch oder Eigenschaft)</td>
<td>Ausführen des Skripts <code>cellValues</code> zur Ermittlung der Werte, die angezeigt werden sollen.</td>
</tr>
<tr>
<td>Nicht anzeigen</td>
<td>Über dieses boolesche Attribut kann gesteuert werden, ob Werte der ausgewählten Eigenschaft angezeigt werden sollen. Standardmäßig werden alle Eigenschaften angezeigt.</td>
</tr>
<tr>
<td>Nicht suchen</td>
<td>Hier kann eingestellt werden, dass die konfigurierte Eigenschaft nicht in die Suche übernommen wird. D.h. eingegebene Suchwerte werden nicht über diese Eigenschaft gesucht. Achtung: Wenn alle Spaltenelemente einer Spalte auf &quot;Nicht suchen&quot; geschaltet werden, hat dies denselben Effekt wie &quot;Nicht anzeigen&quot;!</td>
</tr>
<tr>
<td>Hervorhebung</td>
<td>Hier können für das Anzeigen von Werten Formatierungsvorgaben gemacht werden, zur Wahl steht derzeit nur Unterstreichen.</td>
</tr>
<tr>
<td>Hits verwenden</td>
<td>Standardmäßig werden Objekte erzeugt. Möchte man allerdings in dem Skript <code>cellValues</code> die Hits weiterverarbeiten, muss <code>Hits verwenden</code> eingeschaltet werden.</td>
</tr>
<tr>
<td>Relationszielansicht</td>
<td>Derzeit steht nur die Alternative <code>Drop down</code> zur Verfügung. Wird diese ausgewählt, so werden die möglichen Werte, die sich für die Filterung in der Tabelle für diese Spalte eintragen lassen, aus den möglichen Relationszielen gemäß Schema als Dropdown-Liste zusammengestellt, so dass ein möglicher Wert schnell spezifiziert werden kann. Dies empfiehlt sich für überschaubare Mengen an möglichen Relationszielen. Anmerkung: Dieser Parameter steht nur zur Verfügung, wenn eine Eigenschaft vom Typ Relation gewählt wurde.</td>
</tr>
</tbody>
</table>

Es ist möglich für eine Spaltenkonfiguration mehrere Spaltenelemente zu definieren. Das ist z.B. dann sinnvoll, wenn mehrere Attribute in der Suche berücksichtigt werden sollen wie beispielsweise das Attribut Name und Synonym, aber nur eines davon angezeigt werden soll.

**Beispiel**

Im ersten Spaltenelement der Spaltenkonfiguration `Name` wurde das Attribut `Name` hinter-
legt.

Auf der zweiten Spalte wurde im Spaltenelement die Beziehung *ist Thema von* hinterlegt.

Auf der dritten Spalte wurde im Spaltenelement der Strukturabfrage-Baustein *transitive Beziehungskette vom Thema nach oben* hinterlegt.
Hinterlegte Struktur-Abfrage

Um Werte aus dem Eingabefeld der Spalte übernehmen zu können, muss die hinterlegte Strukturabfrage konfigurierte Parameter haben. Es können mehrere Parameter angebracht werden, diese sind beim Auswerten der Strukturabfrage alle mit demselben Wert belegt.

Vorsicht: Hier liegt ein Unterschied zu sonstigen Fällen, in denen die Strukturabfrage verwendet wird. Normalerweise bestimmt das Ausgangsobjekt (in diesem Fall wäre dies "Thema") die Ergebnisse, hier sind es jedoch die Objekte oder Eigenschaften, an denen der Parameter angebracht ist (in diesem Fall das Namensattribut).

Der in der Spalte gezeigte Wert ist, wenn keine weiteren Anpassungen vorgenommen werden, der Wert des zum Filtern verwendeten Attributes. Wenn sich der angezeigte Wert nicht aus dem zur Filterung verwendeten Attribut ergibt, so gibt es zwei Möglichkeiten:

- der Bezeichner "renderTarget" kann an Relationszielen angebracht werden. Die hiermit markierten Objekte werden als Spaltenwert in der Tabelle angezeigt. "renderTarget" bewirkt außerdem, dass bei einer Ausgabe über die JavaScript API die Eigenschaften zur Darstellung als Link mit ausgegeben werden.
- der Bezeichner "renderProperty" kann an Attributen angebracht werden. Die hiermit markierte Eigenschaften werden als Spaltenwerte in der Tabellenspalte angezeigt.

Wird der Suchbaustein nicht zur Filterung verwendet, so muss das anzuzeigende Element mittels renderTarget oder renderProperty bestimmt werden!

Die Strukturabfragen, die in den Baustein des Spaltenelements eingefügt werden, können aus einer Liste bereits registrierter Strukturabfragen ausgewählt werden, sie können aber auch für genau diesen Baustein neu angelegt werden, was auch die Vergabe eines Registrierungsschlüssels mit sich bringt. Die Eigenschaft Nicht anlegen hat auf Spalten, die mit einem Strukturabfrage-Baustein belegt sind, keine Wirkung.

Auf die vierte Spalte wurde ein Script-Baustein abgebildet
Es sollen die Verantwortlichen für die Objekte, mit denen das in der Tabelle gelistete Thema mit *ist Thema von* verknüpft ist, angezeigt werden. Wie bei der Strukturabfrage kann das zugeordnete Skript aus einer Liste von bereits registrierten Skripten ausgewählt oder aber im Dialog neu angelegt (und registriert) werden. Der Skript-Editor öffnet sich nach Klicken auf den Skript-Baustein-Namen.

```javascript
// Returns matching elements for column search value "objectListArgument"
function filter(elements, queryParameters, objectListArgument) {
    return elements;
}

// Returns cell values for the given element
function cellValues(element, queryParameters) {
    var result = new Array();
    var firstTargets = element.relationTargets("istThemaVon") ;
    if ( firstTargets.length == 0 ) { return result ;} 
    else {
        for (var i = 0; i < firstTargets.length; i++) {
            var secondTargets = firstTargets[i].relationTargets("hatVerantwortlichen");
            for (var j = 0; j < secondTargets.length; j++) {
                result.push(secondTargets[j].name());
            }
        }
    return result.join(', ');
}
```

In diesem Fall ist die Sprache des Skriptbausteins JavaScript. Hier müssen zwei Teile gepflegt werden, der obere Teil dient der Filterung aller Elemente der Tabelle anhand des in der Spalte eingetragenen Wertes *objectListArgument*, der zweite Teil gibt an, wie für ein Element ein auszugebender Wert berechnet wird. Der erste Teil ist im Moment nicht ausgeführt. Zu bei-
den Teilen wird ein Code-Muster beim Erzeugen eingefügt, auf das beim Erstellen aufgebaut werden kann.

Wenn KScript als Sprache im Skript-Baustein gewählt wurde, um die Ausgabe einer Spalte zu steuern, dann muss das ausgewählte (registrierte) Skript zu jedem Objekt, das eine Zeile bildet, einen Rückgabewert für die Spalte liefern.

Da in KScript im Prinzip nur eine Ausgabe vorgesehen ist, wurde für die Filterung folgende Konvention getroffen:

Wenn es in dem ausgewählten Skript eine Funktion mit Namen objectListScriptResults und einem deklarierten Parameter gibt, so wird diese Funktion mit dem Argument der zugehörigen Sucheingabe aufgerufen, um die Menge der passenden Objekte zurückzuliefern. Die Funktion wird auf dem Wurzelbegriff oder der bisherigen Treffermenge als Ausgangsobjekt aufgerufen - je nach dem, wie die Suche am besten gelöst werden kann. Damit diese Variante wirklich effizient wird, ist es empfehlenswert, die Sucheingabe entsprechend auszuwerten und mit dem Ergebnis eine registrierte Strukturabfrage aufzurufen, um deren Ergebnis an die Objekliste weiterzuleiten.

1.7.4.8 Suche


Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abfrage</td>
<td>Hier kann die Suche ausgewählt werden, die sofort bei der Anzeige des Konfigurationselements ausgeführt wird. Das semantische Objekt für welches die View-Konfiguration angezeigt wird, kann als Zugriffselement in der Abfrage verwendet werden.</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
<tr>
<td>Tabelle</td>
<td>Hier wird eine Tabellenkonfiguration angegeben, die zur Darstellung des Suchergebnisses dient.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Die Beschriftung lässt sich alternativ auch über ein Skript ermitteln.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>Über ein Skript lässt sich steuern, ob das Konfigurationselement angezeigt werden soll.</td>
</tr>
</tbody>
</table>
Skript für Tabellenkonfiguration | Die Tabelle kann auch über ein Skript ermittelt werden.

### Einstellungsmöglichkeiten für eine Abfrage

Die folgenden Parameter werden als Meta-Eigenschaften für eine Abfrage gepflegt:

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parametername</td>
<td>Angabe eines Parameternamens, wie er in der Abfrage verwendet wird.</td>
</tr>
</tbody>
</table>

### Einstellungsmöglichkeiten für einen Parameternamen

Die folgenden Parameter werden als Meta-Eigenschaften für einen `Parametername` gepflegt:

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skript</td>
<td>Das Skript <code>parameterValue</code> wird zur Ermittlung des Suchwertes für den angegebenen Parameternamen verwendet.</td>
</tr>
</tbody>
</table>
| Wertermittlung  | Angabe über den Weg der Wertermittlung  
|                 | `Skript`: Der Wert wird aus dem Skript ermittelt und darf nicht von einem Benutzer überschrieben werden.  
|                 | `Skript, überschreibbar`: Das Skript ermittelt Wert. Der Benutzer darf überschreiben.  
| Typ             | xsd-Typ                                                               |
| Beschriftung    | Beim Herausschreiben nach JSON landet dieser Wert in `label`.        |

### Darstellung in einer Anwendung

Eine Ausgabe für die Suchergebnisse gibt es erst ab Version 4.1 von i-views.
Ähnliche Sehenswürdigkeiten

Senckenberg Naturmuseum Frankfurt
Frankfurt a.M.

Uluğ
Red Centre

Uluğ-Kata-Tjuṭa-Nationalpark
Red Centre

Karlo Karlu
Red Centre

Watarrka-Nationalpark
Red Centre

Darstellung im Knowledge-Builder

Die Ergebnisse einer beliebigen Abfrage werden im Knowledge-Builder stets in einer Objektliste angezeigt.

Beispiel:

Definition der Suche

Das Ergebnis der Abfrage wird im Reiter "Bekannte" bei Objekten des Typs "Person" im Knowledge-Builder angezeigt.

Statt eines einzelnen Konfigurationselements, lassen sich Suchen auch in mehrere seperate Konfigurationen aufteilen:

Facetten
Um die Ergebnismenge einer Suche einschränken zu können, lassen sich Facetten auf Beziehungsziele definieren. Facetten-Konfigurationselemente können einer Suche-Konfiguration untergeordnet werden.

Anmerkungen:

- Dieses Feature steht ab dem Patch-Release 4.3.1 zur Verfügung.
- Es ist ein experimentelles Feature, d.h. in späteren Versionen kann es zu Modelländerungen kommen, die unter Umständen nicht in einem Upgrade-Code beachtet werden und somit von Hand durchgeführt werden müssen.
- Das Einschränken auf einen dynamisch ermittelten Term ist noch nicht implementiert.

Mit der Aktion Facette neu anlegen oder Facette hinzufügen wird eine Facette zur der Suche-Konfiguration hinzugefügt.

Es können natürlich mehrere Facetten an eine Suche-Konfiguration angebracht werden. Da sich keine Synergieeffekte bei der Facettenberechnung untereinander ergeben, verlängert sich dadurch jedoch der Berechnungsvorgang.

An einer Facetten-Konfiguration stehen folgende Einstellungsmöglichkeiten zur Verfügung.

Einstellungsmöglichkeiten
<table>
<thead>
<tr>
<th>Beschriftung</th>
<th>Idealerweise ist die Beschriftung angegeben. Ist dieser Wert nicht gesetzt, wird der Name des Eingangsobjektes der Abfrage verwendet.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abfrage zur Termermittlung</strong></td>
<td>Diese Abfrage ist obligatorisch. Sie wird benötigt um die Terme für eine Facette zu ermitteln. Das Eingangsobjekt ist vom Typ gleich den Suchergebnissen der Suche, die in der Suche-Konfiguration definiert ist. Die zu findenden Terme sind durch den Bezeichner term gekennzeichnet.</td>
</tr>
<tr>
<td>Abfrage zur Elterntermermittlung</td>
<td>Soll eine Term-Hierarchie gebildet werden, muss durch diese Abfrage definiert werden, wie sich Eltern-Kind-Elemente finden. Das Eingangsobjekt ist dabei das Kind-Element. Der Bezeichner parentTerm kennzeichnet das Eltern-Element. Anmerkung: Alle Terme, die eine Hierarchie bilden sollen, müssen durch die Abfrage an der Facetten-Konfiguration gefunden werden.</td>
</tr>
<tr>
<td>Ausblenden ab Anzahl von Termen</td>
<td>.</td>
</tr>
<tr>
<td>Terme Absteigend sortieren</td>
<td>Standardmäßig werden die Namen oder die Anzahlen aufsteigend sortiert. Mit diesem Flag lässt sich die Sortierung umdrehen.</td>
</tr>
<tr>
<td>Term-Anzahl nicht anzeigen</td>
<td>Dieses Flag muss gesetzt werden, wenn die Anzahl nicht mit angezeigt werden soll.</td>
</tr>
<tr>
<td>Kindterm initial anzeigen</td>
<td>Standardmäßig werden bei einer Hierarchie die Kindelemente erst angezeigt, wenn das dazugehörige Elternlement ausgewählt wurde. Wird dieses Flag gesetzt, werden initial alle Kindelemente mit ausgeliefert.</td>
</tr>
<tr>
<td>Leere Term anzeigen</td>
<td>Standardmäßig werden Termen, die keine Anzahl haben, nicht eingeblendet. Durch setzen dieses Flags werden auch diese angezeigt.</td>
</tr>
<tr>
<td>Maximale Anzahl an Termen</td>
<td>Hier kann die Anzahl der Term festgelegt werden, die angezeigt werden sollen. Standardmäßig werden immer alle Term dargestellt.</td>
</tr>
<tr>
<td>Terme nach Anzahl sortieren</td>
<td>Standardmäßig werden die Terme nach dem Namen sortiert. Durch das setzen dieses Flags werden die Terme nach der Anzahl der Vorkommnisse in der Treffermenge sortiert.</td>
</tr>
</tbody>
</table>
Hier stehen folgende Einstellungsmöglichkeiten zur Verfügung:


- **Statisch**: Es müssen alle Terme, die angezeigt werden sollen, für sich konfiguriert werden. Es muss für jeden Term eine Suche angelegt werden, die die möglichen Treffer in der Hauptsuche beschreibt.

Wird keine Termart ausgewählt (Standardverhalten), werden die Terme anhand der Abfrage an der Facetten-Konfiguration ermittelt. In der Abfrage können nur Relationsziele als mögliche Terme ausgebildet werden.

---

**Termoperator**

Sollen die Suchergebnisse auf alle Terme zutreffend sein *(und)* oder nur auf (mindestens) einen Term zutreffen *(oder)*.

---

### 1.7.4.8.1 Suchfeld-Ansicht

Dieses Konfigurationselement bietet eine separate Eingabemaske um Suchparameter für eine Abfrage eingeben zu können.

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abfrage</td>
<td>Hier kann eine Abfrage festgelegt werden.</td>
</tr>
<tr>
<td>Parametername</td>
<td>Angabe eines Parameternamens, wie er in der Abfrage verwendet wird.</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Hier kann eine Beschriftung für die Elementkonfiguration angegeben werden.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ lässt sich die Beschriftung auch über ein Skript ermitteln.</td>
</tr>
</tbody>
</table>

---

**Einstellungsmöglichkeiten für Parameternamen**

Die folgenden Parameter werden als Meta-Eigenschaften für einen `Parametername` gepflegt:

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skript</td>
<td>Das Skript <code>parameterValue</code> wird zur Ermittlung des Suchwertes für den angegebenen Parameternamen verwendet.</td>
</tr>
</tbody>
</table>
Wertermittlung | Angabe über den Weg der Wertermittlung
---|---
Skript: Der Wert wird aus dem Skript ermittelt und darf nicht von einem Benutzer überschrieben werden.
Skript, überschreibbar: Das Skript ermittelt Wert. Der Benutzer darf überschreiben.

<table>
<thead>
<tr>
<th>Typ</th>
<th>Datentyp des Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Beim Herausschreiben nach JSON landet dieser Wert in <em>label</em></td>
</tr>
<tr>
<td>Reihenfolge</td>
<td>Mehrere Parameter können hier eine Reihenfolge zugewiesen bekommen</td>
</tr>
</tbody>
</table>

1.7.4.8.2 Suchergebnis-Ansicht
Dieses Konfigurationselement beinhaltet eine separate Ergebnisansicht einer Suche.

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abfrage</td>
<td>Hier kann eine Suchabfrage ausgewählt werden, die beim Anzeigen der Konfiguration ausgeführt wird.</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. <em>Alternative</em> eingebettet ist.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ kann die Beschriftung auch über ein Skript ermittelt werden.</td>
</tr>
<tr>
<td>Tabelle</td>
<td>Hier wird eine Tabellenkonfiguration ausgewählt, die zur Darstellung der Suchergebnisse dient.</td>
</tr>
<tr>
<td>Skript für Tabellenkonfiguration</td>
<td>Alternativ kann die Tabellenkonfiguration über ein Skript generiert werden.</td>
</tr>
</tbody>
</table>

1.7.4.8.3 Facetten-Ansicht
Für facettierte Suchen bietet dieses Konfigurationselement eine separate Ansicht für die Facettenauswahl.

**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abfrage</td>
<td>Hier kann eine Abfrage festgelegt werden.</td>
</tr>
</tbody>
</table>
Einzelne Facetten werden als untergeordnete Konfigurationen zu einer Facetten-Ansicht hinzugefügt.
Für die Facetten-Konfiguration siehe Abschnitt Suche.

### 1.7.4.9 Graph

In einem Graph werden die Inhalte der semantischen Datenbank mit ihren Objekten und Verbindungen graphisch dargestellt (siehe Knowledge-Builder > Grundlagen > Graph-Editor).

#### Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung wird nur ausgegeben, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Ein Skript, welches die Beschriftung zurückliefert.</td>
</tr>
<tr>
<td><strong>Graph-Konfiguration</strong></td>
<td>Hier wird ein Graph-Konfigurations-Objekt festgelegt.</td>
</tr>
<tr>
<td>Legende ausblenden</td>
<td>Legt fest ob die Legende zu den Knotentypen angezeigt werden soll.</td>
</tr>
<tr>
<td>Breite / Höhe</td>
<td>Legt die Breite und Höhe des Konfigurationselements, entweder prozentual oder pixelgenau, fest.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>In einem hier referenzierten Skript lässt sich die Sichtbarkeit des Konfigurationselements festlegen.</td>
</tr>
<tr>
<td>Strukturabfrage für Starthelemente</td>
<td>Eine Abfrage, welche die anzuzeigenden Objekte ermittelt.</td>
</tr>
</tbody>
</table>

#### 1.7.4.9.1 Graph-Konfiguration


Einer Graph-Konfiguration werden Knotenkategorie-Elemente untergeordnet.

#### Einstellungsmöglichkeiten
**Name** | **Wert**
--- | ---
Beschriftung | Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. einer weiteren *Alternative* eingebettet ist.
maximale Pfadlänge | .
Schritte bis Knotenausblendung | .

### 1.7.4.9.2 Knotenkategorie

Knotenkategorien werden Graph-Konfigurationen untergeordnet. Ihnen werden Verknüpfung-Elemente untergeordnet.

**Einstellungsmöglichkeiten**

**Name** | **Wert**
--- | ---
Beschriftung | Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. einer weiteren *Alternative* eingebettet ist.
Erweiterungen initial anzeigen | .
Farbe | Farbgebung der Knoten dieser Kategorie
In Legende anzeigen | *Immer, Bei Bedarf oder Nie*
Knotengröße | .
Nur das Icon malen | .

### 1.7.4.9.3 Verknüpfung

Verknüpfungen sind *was genau*?
Sie werden einer Knotenkategorie untergeordnet.

**Einstellungsmöglichkeiten**

**Name** | **Wert**
--- | ---
Abfrage für Verknüpfung | .
Beschriftung | Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. einer weiteren *Alternative* eingebettet ist.
1.7.4.10 Text
Dieses Konfigurationselement gibt einen einfachen Text aus. Dieser wird entweder fest konfiguriert oder über ein Skript ermittelt.

**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung wird nur ausgegeben, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
</tbody>
</table>
| Skript für Beschriftung | Ein Skript, welches die Beschriftung zurückliefer.
| Text                  | Text, der ausgegeben werden soll.                                   |
| Skript für Text       | Ein Skript, welches den anzuzeigenden Text zurückliefer.

1.7.4.11 Bild
Mit Hilfe dieses Konfigurationselements kann eine statische Grafik eingebunden werden.

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung wird nur ausgegeben, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ kann hiermit die Beschriftung durch ein Skript ermittelt werden.</td>
</tr>
<tr>
<td>Bild</td>
<td>Die Bilddatei, welche ausgegeben werden soll.</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Skript für Bild</td>
<td>Alternativ kann die Grafik durch ein Skript zurückgegeben werden.</td>
</tr>
<tr>
<td>Breite / Höhe</td>
<td>Skaliert die Bilddatei auf die angegebenen Maße.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>Durch ein Skript lässt sich bestimmen ob die Grafik angezeigt werden soll.</td>
</tr>
</tbody>
</table>

1.7.4.12 Skriptgenerierter View/HTML

Skriptgenerierte View


Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
<tr>
<td>Skript</td>
<td>Skript zur Generierung der View.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>Skript zum Ermitteln der Sichtbarkeit.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Skript zum ermitteln der Beschriftung.</td>
</tr>
<tr>
<td>viewType</td>
<td>Name des Partial.</td>
</tr>
</tbody>
</table>

Skriptgeneriertes HTML


Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Eine Beschriftung findet nur ihre Anwendung, wenn diese Konfiguration in einer anderen Konfiguration wie z.B. Alternative eingebettet ist.</td>
</tr>
<tr>
<td>Skript</td>
<td>Skript zur Generierung einer HTML-Ausgabe.</td>
</tr>
</tbody>
</table>
Beispiel für ein Skript, das eine einfache HTML-Ausgabe erzeugt:

```javascript
function render(element, document) {
    var writer = document.xmlWriter();
    writer.startElement("div");
    writer.startElement("h2");
    writer.cdata(element.name());
    writer.endElement();
    writer.endElement();
}
```

Ausgabe:

```html
<div>
    <h2>Hermann</h2>
</div>
```

1.7.4.13 Label

Mit der Label-Konfiguration kann beispielsweise die Beschriftung einer Webseite oder die Beschriftung einer Tabelle konfiguriert werden. Im Knowledge-Builder werden die Label-Konfigurationen unter der Kategorie „Nachgeordnete Konfiguration“ verwaltet. Label finden im Fenstertitel-Panel Verwendung; hierzu muss ein neues Objekt unter „Label-Konfiguration“ angelegt werden:

Unter „Beschriftung“ und „Bild“ können dann die Einträge vorgenommen werden, die sich später auf dem Browser-Tab der Webseite wiederfinden:

Bezüglich des Web-Frontends, welches durch den Viewkonfiguration-Mapper erzeugt wird, entspricht dies dem Element <title> in der Sektion <head>:

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>YourApplication</title>
</head>
```

Ein Vergleich zeigt die unterschiedlichen Zustände der Webseite ohne Label (Titel = Pfad der Webseite) oder mit Label (Titel = Beschriftung):

1.7.5 Panels

Panels sind Konfigurationselemente, welche die Anwendungsoberfläche in Bereiche aufteilen. Mit ihnen wird das grundsätzliche Layout einer Anwendung aufgebaut.
Panels beinhalten weitere Panels oder View-Konfigurationen und können ineinander verschachtelt werden. Sie können sich gegenseitig beeinflussen.

Panels erhalten immer genau ein Startelement (ein Objekt oder einen Typ) bei ihrer Aktivierung, welches sie an ihre Unterkonfigurationen weiter geben.


Es gibt verschiedene Arten von Panels:

- Hauptfensterpanel
- Dialogpanel
- Fenstertitelpanel
- Fußzeilenpanel
- Normale Panel

Für jede Anwendung muss genau ein sogenanntes Hauptfensterpanel existieren, welches durch untergeordnete Panels aufgeteilt werden kann. Zusätzlich kann ihm ein Fenstertitelpanel zugeordnet werden, welches den Titel und das Logo (Favicon) der Anwendung festlegt.

Weiterhin können einer Anwendung weitere Dialogpanel zugewiesen werden, die als Pop-Up über dem Hauptfenster angezeigt werden können. Diese können neben weiteren Panels auch Fenstertitel- und Fußzeilenpanel enthalten.

Für jedes Panel muss ein bestimmter Paneltyp ausgewählt werden.

- Layout-Panels (enthalten weitere Panels):
  - Lineares Layout (alle untergeordnete Panels werden horizontal oder vertikal angeordnet dargestellt)
  - Wechselndes Layout (nur eines der untergeordneten Panels wird zur gleichen Zeit angezeigt)
- Ansicht-Panels (enthalten View-Konfiguration(en)):
  - Festgelegte Ansicht (enthält ein einziges festgelegtes Konfigurationselement)
  - Flexible Ansicht (mehrere Ansichten je nach Typ des Startelementes möglich)

**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aktionen aktivieren in Panel</td>
<td>Alle Aktionen, die im Quell-Panel aktiviert werden, führen dazu, dass das Ziel-Panel mit dem jeweilig übergebenen Objekt angezeigt werden (Beispiel: Jeder Klick im Panel Objektliste führt dazu, dass im Panel Detailansicht das Ergebnis angezeigt wird).</td>
</tr>
<tr>
<td>beeinflusst</td>
<td>Hier kann ein Panel festgelegt werden, dass vom aktuellen Panel beeinflusst wird (Beispiel: Je nachdem welche Objekte bei Suchergebnis angezeigt werden, beeinflusst das welche Facetten dazu angezeigt werden).</td>
</tr>
</tbody>
</table>
### Skript für Zielobjekt

Mithilfe von Skripten können hier nicht einfach nur Panels, sondern auch Bedingungen angegeben werden unter denen bestimmte Panels durch das aktuelle Panel beeinflusst werden.

### Einstellungsmöglichkeiten Layout

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>CSS-Klassen für das Panel (wird nur für Web-Anwendungen bzw. im ViewConfig-Mapper berücksichtigt)</td>
</tr>
<tr>
<td>Breite / Höhe</td>
<td>Die exakten Maße des Panels können hier jeweils entweder prozentual oder pixelgenau gesetzt werden.</td>
</tr>
<tr>
<td>Maximale Breite / Höhe</td>
<td>Alternativ lassen sich hier die Höchstmaße des Panels angeben. Das Panel nimmt sich so viel Platz wie es benötigt, ohne diese Werte zu überschreiten.</td>
</tr>
<tr>
<td>overflow-x / -y (Scrollbar)</td>
<td>Hierüber lässt sich die Darstellung von Scrollbars in der Applikation festlegen, wenn der Inhalt des Panels nicht in seine horizontale (x) und vertikale (y) Abmessungen passt. Zur Auswahl stehen auto, scroll und hidden.</td>
</tr>
<tr>
<td>Style</td>
<td>CSS-Styling-Regeln für das Panel (wird nur in Web-Anwendungen bzw. im ViewConfig-Mapper berücksichtigt)</td>
</tr>
</tbody>
</table>

### 1.7.5.1 Aktivierung von Panels

Panels kennen zwei grundsätzliche Zustände: "aktiv" und "inaktiv". Ein Panel ist sichtbar, wenn es aktiv ist.

Die Aktivierung von Panels funktioniert über folgende Mechanismen:

A. Zum Start einer Anwendung ist immer das Haupfenster-Panel der Anwendung aktiv
B. Beim Ausführen einer Aktion bestimmt der Ausführungsort, welches Panel aktiv wird

Ausgehend von A/B gibt es Folge-Aktivierungen nach diesen Regeln:

1. Beeinflusste Panels werden aktiviert
2. Panels mit einer spezialisierten Funktion (z.B. Fenstertitel) werden aktiviert - und zwar von allen Panels in der entsprechenden Hierarchie aus
3. Unterpanels werden aktiviert
4. Im Falle eines Panels mit wechselndem Layout: Geschwister-panels des aktiven Unterpansels werden deaktiviert
5. Fortfahren bei 1. bis keine weiteren Panels mehr aktiviert werden können (eine einge-
   baute Zykenprüfung verhindert Endlosschleifen)

Folge-Aktivierungen transportieren jeweils das angezeigte Modell. Wenn also beispielsweise
Panel A das Objekt "Herr Meier" anzeigt, dann zeigt das aktivierte Unterpansel B ebenso "Herr
Meier" an.

Zuletzt wird sichergestellt, dass alle Panels oberhalb der aktivierten Panels ebenso aktiv sind.
Dabei wird deren Inhalt aber nicht neu berechnet.

**Fortgeschrittene Aktivierungsmechanismen (ab Version 5.2):**

In Schritt A (Aktionsaktivierung) sowie in Schritt 1 (Beeinflussung) kann über den sogenannten
"Aktivierungsmodus" die Berechnung der Panel-Inhalte optimiert werden.

Auf diese Weise kann vermieden werden, dass Panel-Inhalte neu berechnet werden, die ak-
tuell nicht angezeigt werden, weil sie trotz Aktivierung nicht im Sichtbarkeitsbereich liegen
(z.B. ein Warenkorb). Für diesen Fall gibt es die Option "Lazy".

Analog dazu kann mit der Option "Aktualisierung" eine Auslösung der Aktivierungskette ver-
mieden werden. Hier wird nur der Inhalt des Panels neu berechnet.

Die Option "Anzeige" ist der Standard, wenn keine der beiden obigen Optionen gewählt
wurde.

**1.7.5.2 Layout-Panel**

Mit Layout-Panel wird die Anwendung in verschiedene Bereiche unterteilt. Lineare Layouts
ordnen untergeordnete Panels entweder nebeneinander oder untereinander an. Wechsel-
nde Layouts erlauben alternative Darstellungen auf der gleichen Visualisierungsmöglichkeit, bei
denen nur eines der untergeordneten Panels gleichzeitig angezeigt wird.

**Einstellungsmöglichkeiten Konfiguration**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
</table>
| Standardmäßig erstes aktivieren (nur bei wechsel-
  ndes Layout)                                      | Wird hier der Haken gesetzt, heißt das, dass das erste unter-
                                                      | geordnete Panel standardmäßig aktiviert ist (im Beispiel
                                                      | unten ist das die Startseite)                                    |

**1.7.5.3 Ansicht-Panel**

Ansicht-Panel dienen als Container für einzelne Ansichten. Sie können dafür keine weiteren Panels enthalten.

**Einstellungsmöglichkeiten**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-Wissensnetzelement</td>
<td>Hier kann ein konkreres Objekt oder ein konkreter Typ angegeben werden, der als Ausgangselement dient, von dem aus weitere Wege durch das Netz gegangen werden können.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Sub-Konfiguration (nur bei festgelegte Ansicht)</td>
<td>Hier kann die eine View-Konfiguration angegeben werden, die für die festgelegte Ansicht genutzt wird.</td>
</tr>
</tbody>
</table>

### 1.7.5.4 Dialog-Panels


Zum Ausblenden ("Schließen") von Dialogfenstern müssen ebenfalls Aktionen verwendet werden. Ist in einer Aktionskonfiguration das Attribut "Panel schließen" angehakt, so führt die Ausführung dieser Aktion in einem Dialogfenster dazu, dass das Fenster ausgebldendet wird. Die Aktion muss dau also mit einem Menü verknüpft sein, welches im Dialog-Panel selbst oder einem seiner untergeordneten Panels angezeigt wird.

Dialogfenster werden inhaltlich in die folgenden drei Bereiche unterteilt:

- Fenstertitel
- Inhaltsbereich
- Fußzeile

Über das Attribut "Paneltyp" am Dialog-Panel selbst sowie an dessen Fenstertitel- und Fußzeilen-panels kann bestimmt werden, ob das jeweilige Panel Layout- oder Ansichtsfunktionalitäten bereitstellt. Details zu den verschiedenen Paneltypen sind in den vorangehenden Kapiteln beschrieben.

Dialog-panels können im Knowledge-Builder folgendermaßen angelegt werden:

1. Melden Sie sich mit einem Benutzerkonto im Knowledge-Builder an, das über Administrator-Berechtigungen verfügt

2. Öffnen Sie im Navigationsbereich auf der linken Seite die Rubrik "Technik" und wählen Sie den Unterpunkt "View-Konfiguration" aus.

3. Wählen Sie den Reiter "Anwendung" auf der rechten Seite aus.
4. Wählen Sie in der Liste darunter die Anwendung, zu der Sie das Dialog-Panel hinzufügen möchten (Normalerweise "Viewkonfiguration-Mapper").

5. Wählen Sie das oberste Element im Panel-Baum unten aus und Klicken Sie auf das Anlegen-Symbol


1.7.6 Grundlagen der View-Konfiguration

1.7.6.1 Die Verwendung von View-Konfigurationen

Auf dem Reiter Verwendung einer View-Konfiguration muss angegeben werden, in welcher Anwendung und für welchen Objekttyp die View-Konfiguration angezeigt werden soll.
**Im Feld anwenden in auf dem Reiter Verwendungen wird die Anwendung eingetragen.**


**Anwendungen der View-Konfiguration festlegen**

Die folgenden Anwendungen stehen von Anfang an zur Verfügung:

- **Graph-Editor:** Die Konfigurationen haben Einfluss auf die Darstellung im Graph-Editor. Der Graph-Editor dient zur Visualisierung der semantischen Elemente und deren Zusammenhängen.
- **Knowledge-Builder:** Die View-Konfigurationen werden im Knowledge-Builder selbst angewendet. Hier stehen neben den Detailkonfigurationen auch die Objektlisten-Konfigurationen zur Verfügung.
- **Knowledge-Portal:** Das Knowledge-Portal ist eine Komponente von i-views, die als Frontend eingesetzt werden kann. Es stellt die Objekte des semantischen Netzes auf Detailseiten und in Kontextboxen basierend auf deren semantischen Kontext dar.
- **Net-Navigator:** Er dient zur Visualisierung von semantischen Elementen. Er kann im Gegensatz zum Graph-Editor der Teil des Knowledge-Builders ist, in den Anwendungen Knowledge-Portal und Viewkonfigurations-Mapper eingesetzt werden.
- **Topic-Chooser:** Er ermöglicht die Auswahl von Relationszielen in einem Fenster.

**Viewkonfigurations-Mapper:** Der Viewkonfigurations-Mapper ist ein intelligentes Frontend, das im Gegensatz zum Knowledge-Portal die View-Konfigurationen verwendet. Mit ihm können einfach und schnell Sichten auf die Daten erstellt werden.

Darüber hinaus können auch eigene beliebige Anwendungen definiert werden, die an dieser Stelle mit der View-Konfiguration verknüpft werden können.

**Objekttyp der View-Konfiguration festlegen**

Im Feld *anwenden auf* wird der Objekttyp automatisch eingetragen, wenn die View-Konfiguration am Objekttyp definiert wird. Hier können nach Bedarf weitere Objekttypen angegeben werden. Je nach dem wo man die View-Konfiguration angelegt hat, entweder unter *Objekt* oder unter *Typ*, wird die View für Objekte des ausgewählten Typs oder den Typ einschließlich sein-
er Untertypen angezeigt. Durch das Setzen oder Wegnehmen des Häckchens des Attributs *anwenden auf Untertypen* kann diese Auswahl (gültig für Objekte oder Typen) auch nach der Definition der View-Konfiguration geändert werden.

**Wiederverwendung von View-Konfigurationen**

View-Konfigurationen können in anderen View-Konfigurationen wieder verwendet werden. So kann man beispielsweise eine Tabelle konfigurieren, die für die Anzeige von mehreren View-Konfigurationen vom Typ Suche eingesetzt wird. Falls die View-Konfiguration Baustein einer anderen View-Konfiguration ist, wird dies unter *inverse anwenden* in angezeigt.

### 1.7.6.2 Die Gültigkeit von View-Konfigurationen

Im Kapitel *Die Verwendung von View-Konfigurationen* wurde bereits beschrieben, dass es für View-Konfigurationen ausschlaggebend ist, ob, in welcher Anwendung und für welche Objekte bzw. Typen die View angezeigt wird. Trotzdem ist es möglich, dass die View-Konfiguration nicht in der ausgewählten Anwendung angezeigt wird. Hier stellt sich die Frage: Wann ist eine View-Konfiguration gültig? Und für welche Objekt bzw. Typen ist die View-Konfiguration gültig?

**Vererbung von View-Konfigurationen**

View-Konfigurationen verhalten sich in Bezug auf die Vererbung wie Eigenschaften. View-Konfigurationen werden auf die Untertypen bzw. die Objekte der Untertypen vererbt.

**Anwendung der konkretesten View-Konfiguration**

Die Untertypen verwenden nach dem Prinzip der Vererbung die View-Konfiguration der Obertypen solange sie keine eigenen View-Konfigurationen besitzen. Es wird immer die konkreteste View-Konfiguration angewendet: Das ist die Konfiguration, die direkt am Typ definiert ist. Ist das nicht der Fall, so wird geprüft ob es am Obertyp eine View-Konfiguration gibt. Ist das ebenfalls nicht der Fall so wird in der Typenarchie jeweils eine Ebene nach oben gegangen und geprüft ob eine View-Konfiguration definiert ist. Es wird dann diejenige View-Konfiguration angewendet, die dem Objekttyp am nächsten steht. Wird keine View-Konfiguration an den Obertypen gefunden, wird für Administratoren die Default-Konfiguration verwendet.

**Was passiert wenn zwei gleichwertige View-Konfigurationen existieren?**

Gibt es zwei gleichwertige View-Konfigurationen, so wird keine View-Konfiguration angezeigt. Wurde bei einer View-Konfiguration die Anwendung oder der Objekttyp nicht definiert, zählt diese nicht zu den aktiven View-Konfigurationen. In diesem Fall wird die andere View-Konfiguration verwendet. Möchte man für unterschiedliche Benutzer jeweils andere Views anzeigen, kann im Detektorsystem eine Regel definiert werden. In diesem Fall wird dann die View-Konfiguration entsprechend der definierten Regel angewendet, solange die Regel abhängig von Benutzer nur eine gültige View-Konfiguration liefert.

**1.7.7 Knowledge-Builder-Konfiguration**

Die hier beschriebenen View-Konfigurationen betreffen ausschließlich den Knowledge-Builder. Weitere View-Konfigurationen, die den Knowledge-Builder betreffen, finden sich auch an anderen Stellen in Kapitel 7, können dann aber auch zusätzlich jeweils die Ausgabe in JSON betreffen.
1.7.7.1 Ordnerstruktur


1.7.7.1.1 Die Standard-Ordnerstruktur

Die Konfiguration der Standard-Ordnerstruktur stellt Ordner zur Verfügung, so dass im semantischen Modell navigiert und Inhalte abgelegt werden können. Für Administratoren werden drei Hauptbereiche zur Verfügung gestellt.

Der obere Hauptbereich "Ordner" stellt Ordner für die Anlage weiterer Ordner und das Verwalten von Inhalten zur Verfügung. Das sind der Arbeitsordner, der Privatordner, der Ordner "Zuletzt verwendete Objekte" und der Ordner "Suchergebnisse".

Der zweite Hauptbereich "Wissensnetz" ermöglicht die Navigation zu den Elementen über die Hierarchie der Typen. Die hier zu erreichenden Elemente sind Typen, Objekte und auch Attribute und Relationen. Dafür enthält der Bereich drei Ordner:

- Objekttypen für die Hierarchie der Objekttypen und ihrer konkreten Objekte
- Relationstypen für die Hierarchie der Relationen
- Attributtypen für die Hierarchie der Attribute


Die Konfiguration dieser Standard-Ordnerstruktur kann im Technik-Bereich >> View-Konfiguration überprüft, verändert und den Bedürfnissen der Anwender angepasst werden.

1.7.7.1.2 Konfiguration der Ordnerstruktur

Die Ordnerstruktur wird im Technik-Bereich unter View-Konfiguration >> Objekttypen >> Knowledge-Builder-Konfiguration >> Ordnerstruktur konfiguriert. Einen schnellen Zugriff auf die Konfigurationen erhält der Admin, wenn er im Technik-Ast den Knoten View-Konfiguration selektiert und im rechten Teilfenster auf dem Reiter Ordnerstruktur das Objekt Organizer auswählt.


<table>
<thead>
<tr>
<th>Ordnetyp (obligatorisch)</th>
<th>Parameter</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributtypen</td>
<td>Typ</td>
<td>Der angegebene Attributtyp und alle seine Untertypen werden in einem hierarchischen Baum angezeigt.</td>
</tr>
<tr>
<td>Privatordner</td>
<td>-</td>
<td>Anzeige des Ordners, den nur der Benutzer selbst sehen darf und der für jeden Benutzer unterschiedlich ist.</td>
</tr>
<tr>
<td>Relationstypen</td>
<td>Typ</td>
<td>Der angegebene Attributtyp und alle seine Untertypen werden in einem hierarchischen Baum angezeigt.</td>
</tr>
<tr>
<td>Strukturordner</td>
<td>Strukturordner</td>
<td>Ein beliebiger Strukturordner kann hier eingehängt werden.</td>
</tr>
<tr>
<td>Suchergebnisordner</td>
<td>-</td>
<td>Jeder Benutzer hat einen eigenen Suchergebnisordner, der die letzten Suchergebnisse des Benutzers speichert.</td>
</tr>
</tbody>
</table>
### Typbasierte Ordnerstruktur

| Ansicht "Ohne Vererbung", Typ | Der angegebene Typ und seine Untertypen werden tabellarisch aufgelistet. Ist der Parameter Ansicht "Ohne Vererbung" gesetzt, wird nur der angegebene Typ angezeigt. 
Anmerkung: Zur Steuerung welche Tabellenkonfigurationen auf der rechten Seite Anwendung finden, muss dort die Relation anwenden in mit diesem Ordnerstrukturelement verküpf werden. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtueller Ordner</td>
<td>-</td>
</tr>
<tr>
<td>Zuletzt verwendete Objekte</td>
<td>-</td>
</tr>
</tbody>
</table>

Nur der Konfigurationstyp *Virtueller Ordner* kann weitere Unterkonfigurationen enthalten bzw. nur beim ihm machen Unterkonfigurationen Sinn.

Anmerkung: Bei dem Ordnertyp Attributtypen, Relationstypen und Typenbasierte Ordnerstruktur dient der Parameter Typ zur Angabe des Attribut-, Relations- oder Objekttyp der und dessen Untertypen in dem Ordner angezeigt werden sollen.

### 1.7.7.2 Relationszielsuche

Relationszielsuche konfiguriert für die Suche nach Personen


1.7.7.3 Startansicht


Einstellungsmöglichkeiten

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hintergrundbild</td>
<td>Ein Bild</td>
</tr>
<tr>
<td>Farbwert für Schriftart einer Aktion</td>
<td>Je nach ausgewähltem Bild muss eine andere Farbe für die Beschreibung der Aktionen gewählt werden, um den Text lesen zu können.</td>
</tr>
</tbody>
</table>

Darüber hinaus lassen sich Aktionen definieren. Siehe dazu Kapitel Aktion. Zusätzlich kann eine Aktionsart festgelegt werden. Hier stehen folgende Einträge zur Verfügung:

<table>
<thead>
<tr>
<th>Aktionsart</th>
<th>Aktion</th>
</tr>
</thead>
</table>
Ein Web-Link muss vollständig konfiguriert sein, sonst wird er nicht angezeigt.
Abweichend dazu muss dies bei den drei oberen Aktionsarten (spezialisierte Web-Links) nicht so sein. Diese verwenden, falls eine Eigenschaft fehlt, ihre Standardwerte. Es besteht die Möglichkeit die Standardwerte zu überschreiben.

**Konfigurationsmöglichkeit Web-Link**

<table>
<thead>
<tr>
<th>Name</th>
<th>Wert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Anzeigenname hinter dem Icon</td>
</tr>
<tr>
<td>Symbol</td>
<td>Icon, welches vor der Beschriftung angezeigt wird</td>
</tr>
<tr>
<td>URL</td>
<td>URL die geöffnet werden soll</td>
</tr>
</tbody>
</table>

### 1.7.7.4 Suchfeld

Das Schnellsuchfeld findet sich in der linken oberen Ecke des Hauptfensters. Dieses Feld ermöglicht den schnellen Zugriff auf Abfragen. Diese werden vom Administrator zur Verfügung gestellt oder auch vom Anwender hinzugefügt. Alle Abfragen, die hier verwendet werden, dürfen nur eine Suchzeichenkette oder keine Sucheingabe erwarten.


#### 1.7.7.4.1 Suchfeldkonfiguration für Administratoren

Die "Suchfeld"-Konfiguration legt fest, welche Abfragen vom Administrator im Schnellsuchfeld des Knowledge-Builders zur Verfügung gestellt werden.

Neu angelegte Netze verfügen über eine Suchfeld-Konfiguration, die für alle Nutzer gleich ist. Der Administrator kann diese Suchfeld-Konfiguration erweitern, um allen Nutzern weitere Abfragen zugänglich zu machen. Zusätzlich kann jeder Nutzer seinem Schnellsuchfeld weitere Abfragen hinzufügen, die dann aber nur für ihn persönlich sichtbar sind.

### 1.7.7.4.2 Suchfeldkonfiguration für Anwender

Der Anwender kann Abfragen durch ziehen einer existierenden Abfrage auf das Schnellsuchfeld hinzufügen.

Das Hinzufügen kann ebenfalls über die Einstellungen erfolgen. Auf dem Reiter Persönlich befindet sich der Punkt Suchfeld. Im rechten Bereich im Abschnitt Benutzerdefiniert steht neben dem Hinzufügen auch die Operationen Entfernen und die Möglichkeit die Reihenfolge zu ändern zur Verfügung.

![Suchfeldkonfiguration für Anwender](image)

### 1.7.8 Style

Aufgabe der View-Konfiguration ist die strukturelle Aufbereitung von Elementen des semantischen Modells für die Anzeige. Geht es darüber hinaus um die Festlegung rein optischer Eigenschaften bzw. kontextloser Informationen, wird das sogenannte "Style"-Element verwendet.

#### 1.7.8.1 Style-Eigenschaften in Anwendungen

Es gibt eine Reihe von Style-Elementen, die bereits in i-views definiert sind. Um welche Elemente es sich handelt und wie diese Style-Elemente im Knowledge-Builder angelegt werden, sodass sie dann mit einzelnen Elementen der View-Konfiguration einer Anwendung verknüpft werden können, wird im Folgenden erläutert.

Zunächst muss das Element der View-Konfiguration ausgewählt werden, mit dem wir ein oder mehrere Style-Elemente verknüpfen wollen. Je nach Typ des View-Konfiguration-Elements stehen verschiedene Reiter zur Konfiguration der Styles zur Verfügung ("Aktionen und Styles" -> "Styles" oder direkt "Styles"). Diesen Reiter wählen wir aus und können dann entweder ein neues Style-Element definieren oder ein bereits vorhandenes Style-Element verknüpfen. Wenn wir ein neues Style-Element definieren, müssen wir diesem zuerst einen Konfigurationsnamen geben. Auf der rechten Seite des Editors kann daraufhin die Konfiguration...
vorgenommen werden.
Im Folgenden werden die einzelnen Konfigurationsmöglichkeiten für ein Style-Element erläutert:

<table>
<thead>
<tr>
<th>Name</th>
<th>Attributtyp</th>
<th>StylePropertyKey</th>
<th>Konfigurationsotyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionConfirmation</td>
<td>Verweis auf Skript</td>
<td>actionConfirmation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actionConfirmation</td>
<td>Zeichenkette</td>
<td>actionConfirmation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actionConfirmationTitle</td>
<td>Zeichenkette</td>
<td>actionConfirmationTitle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actionType</td>
<td>Zeichenkette</td>
<td>actionType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>Zeichenkette</td>
<td>class</td>
<td>Verweis auf CSS-Klasse</td>
<td>Verweist auf eine vordefinierte Klassendefinition in der CSS des Viewconfiguration-Mappers oder in Skript &quot;view-configmapper.config.GET&quot;</td>
</tr>
<tr>
<td>class (skript)</td>
<td>Verweis auf Skript</td>
<td>class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>collapsed</td>
<td>Boolean</td>
<td>collapsed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datumsformat</td>
<td>Zeichenkette</td>
<td>DatumFormat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do not print</td>
<td>Boolean</td>
<td>doNotPrint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Download request</td>
<td>Zeichenkette</td>
<td>download-Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element ID</td>
<td>Zeichenkette</td>
<td>elementId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extra</td>
<td>Zeichenkette</td>
<td>download-Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>group column grid</td>
<td>Zeichenkette</td>
<td>groupColumnGrid</td>
<td>Gruppe</td>
<td>Als Eingabe wird ein String mit Zahlen erwartet, die durch ein Leerzeichen oder ein Komma getrennt werden. Jede Zahl definiert die Anzahl der Spalten, wenn das Maximum 12 Spalten beträgt.</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-----------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hide filters</td>
<td>Boolean</td>
<td>hideFilters</td>
<td>Eigenschaften</td>
<td>Wenn angewählt, wird bei Tabellen die Anzeige von Tabellenspalten-Überschriften unterdrückt</td>
</tr>
<tr>
<td>hierarchyDetached</td>
<td>Boolean</td>
<td>hierarchyDetached</td>
<td></td>
<td>Erlaubt die Verwendung vordefinierter renderModes; wird u. a. benötigt zur Verwendung der VCM-Plugins</td>
</tr>
<tr>
<td>Label anzeigen</td>
<td>Boolean</td>
<td>hideLabel</td>
<td></td>
<td>global, none or page</td>
</tr>
<tr>
<td>Panel</td>
<td>Zeichenkette</td>
<td>panel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>render-Mode</td>
<td>Auswahl</td>
<td>renderMode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>render-Mode</td>
<td>Zeichenkette</td>
<td>renderMode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>schreibgeschützt</td>
<td>Boolean</td>
<td>readOnly</td>
<td>Eigenschaften</td>
<td></td>
</tr>
<tr>
<td>vcmStateContext</td>
<td>Auswahl</td>
<td>vcmStateContext</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmStateContext</td>
<td>Zeichenkette</td>
<td>vcmStateContext</td>
<td>vcmTruncate</td>
<td>Ganzzahl</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Zeichenkette für Konfigurationswert</td>
<td>Zeichenkette</td>
<td>style</td>
<td>CSS-Syntax</td>
<td>Erlaubt das Stylings des Web-Frontends mittels CSS-Syntax ohne Verwendung vordefinierter CSS-Klassen</td>
</tr>
<tr>
<td>Skript für Konfigurationswert</td>
<td>Verweis auf Skript</td>
<td>style</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zeilen markieren</td>
<td>Boolean</td>
<td>vcmMarkRowClick</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ziel</td>
<td>Zeichenkette</td>
<td>class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zusätzliche Datumsformate</td>
<td>Zeichenkette</td>
<td>extraDatumFormats</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


In der JSON-Ausgabe werden dann die Schlüssel-Wert-Paare (StylePropertyKey -> Style-Eigenschaft) als Array unter dem Schlüssel additionalConfig ausgeschrieben.
Auch ganz eigene Style-Eigenschaften können so definiert werden.

Beispiel

Konfiguration des Typs Zeichenkette für Style-Wert

<table>
<thead>
<tr>
<th>Eigenschaften des Typs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Begriffsname</td>
<td>Zeichenkette für Style-Wert</td>
</tr>
<tr>
<td>Farbe</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td></td>
</tr>
<tr>
<td>StylePropertyKey</td>
<td>jsonKey1</td>
</tr>
</tbody>
</table>

Konfiguration des Typs Noch eine Zeichenkette für Style-Wert

<table>
<thead>
<tr>
<th>Eigenschaften des Typs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Begriffsname</td>
<td>Noch eine Zeichenkette für Style-Wert</td>
</tr>
<tr>
<td>Farbe</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td></td>
</tr>
<tr>
<td>StylePropertyKey</td>
<td>jsonKey2</td>
</tr>
</tbody>
</table>

Konfiguration des Typs Banner anzeigen Attribut

<table>
<thead>
<tr>
<th>Eigenschaften des Typs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Begriffsname</td>
<td>Banner anzeigen Attribut</td>
</tr>
<tr>
<td>Farbe</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td></td>
</tr>
<tr>
<td>StylePropertyKey</td>
<td>Banner anzeigen</td>
</tr>
</tbody>
</table>

Konfiguration des Objektes Eine Style-Konfiguration vom Typ Style

JSON-Ausgabe:
1.7.8.2 Style-Eigenschaften im Knowledge-Builder

Im Knowledge-Builder bestimmen die Eigenschaften eines Styles die Darstellung einiger Benutzungsoberflächenelemente. Nicht alle möglichen Eigenschaften eines Styles ergeben für alle Konfigurationen Sinn. Die folgende Tabelle zeigt, welche Eigenschaften von welcher Konfiguration unterstützt werden und was der Effekt ist.

<table>
<thead>
<tr>
<th>Style-Eigenschaft</th>
<th>Konfigurationstyp</th>
<th>Effekt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banner anzeigen</td>
<td>Objektkonfiguration</td>
<td>Banner mit Namen des Objekts und Schaltflächen zur Bearbeitung anzeigen. Standard ist nein.</td>
</tr>
<tr>
<td>Baumsicht</td>
<td>Gruppe</td>
<td>Elemente der Gruppe als Baum darstellen. Standard ist nein, d.h. die Gruppenelemente werden nebeneinander oder untereinander angezeigt.</td>
</tr>
<tr>
<td>Editorbreite (Pixel)</td>
<td>Eigenschaft</td>
<td>Breite in Pixeln einer Eigenschaft</td>
</tr>
<tr>
<td>Höhe</td>
<td>Gruppe</td>
<td>Höhe eines Gruppenelements in Pixeln bei nebeneinander dargestellten Gruppenelementen.</td>
</tr>
<tr>
<td>Höhe</td>
<td>Eigenschaft</td>
<td>Höhe in Zeilen bei Zeichenkettenattributen</td>
</tr>
</tbody>
</table>
1.7.9 Detektorsystem zur Ermittlung der View-Konfiguration

Mit Hilfe des Detektorsystems können View-Konfigurationen an Bedingungen geknüpft werden. Das Detektorsystem bestimmt wann welche Konfiguration angezeigt werden soll. Im Folgenden wird die Funktionsweise des Detektorsystems und das Zusammenspiel mit View-Konfigurationen an einem Beispiel erläutert.


Benutzer, die Mitglied in der Band sind, auf die sie zugreifen wollen, sollen die "Band-Ansicht für Mitglieder" sehen. Alle Benutzer, die nicht Mitglied in der Band sind, auf die sie zugreifen wollen, sollen die "Band-Ansicht" sehen. Die Bedingungen, nach denen die Ansichten benutzt werden sollen, werden im Detektorsystem definiert.

Erstellung einer View-Konfigurations-Ermittlung

Das Detektorsystem befindet sich in der linken Ordnerhierarchie unter dem Abschnitt "Technik" und ist mit der Bezeichnung "Ermittlung der View-Konfiguration" unter "View-Konfiguration" versehen.
Im ersten Schritt muss über das Anlegen eines neuen Suchfilters (siehe Kapitel Suchfilter) der Ausgangspunkt definiert werden - das heißt, es muss definiert werden, wofür die noch folgenden Einstellungen gelten sollen. In diesem Beispiel ist unser Ausgangspunkt daher eine View-Konfiguration (hier: "Band-Ansicht für Mitglieder"), für die gleich eine Bedingung angelegt wird. Als Operationsparameter muss "View-Konfiguration" aus der Liste ausgewählt und eingetragen werden. Der Suchfilter sieht dann folgendermaßen aus:

Unter dem Suchfilter, der nach der View-Konfiguration "Band-Ansicht für Mitglieder" sucht, muss nun ein neuer Suchfilter angelegt werden, der die Bedingung für diese View-Konfiguration beschreibt: Die View-Konfiguration "Band-Ansicht für Mitglieder" soll nur für Benutzer sichtbar sein, die Mitglieder der Band sind, die sie sich gerade ansehen. Der zweite Suchfilter prüft also, ob der aktive Benutzer ein Mitglied der Band ist. Über einen Klick auf wird der Menge der Suchergebnisse dann erlaubt, die Konfiguration "Band-Ansicht für Mitglieder" einzusehen. Die folgende Abbildung zeigt, den Suchfilter nach Benutzern, die Mitglied der Band sind, die sie sich gerade ansehen und die Ordnerhierarchie, die auf der linken Seite bisher aufgebaut wurde.

Die View-Konfiguration "Band-Ansicht" wird automatisch für diejenigen Nutzer verwendet, die nicht Mitglied der Band sind, die sie sich gerade ansehen.
Gewichtung der Konfigurationen im Detektorsystem


1.8 JavaScript-API

1.8.1 Einführung

Die JavaScript-API ist eine serverseitige API für semantische Netze. Sie wird u.a. in Triggern, REST-Services und Berichten verwendet.

Mit der API kann man sowohl lesend auf das Wissensnetz zugreifen (Suchen ausführen, Eigenschaften abfragen usw.) als auch Änderungen vornehmen (neue Objekte anlegen, Attribute ändern usw.).

Beispiel: Neues Objekt vom Typ "Person" erzeugen

```javascript
var person = $k.Registry.type("Person").createInstance();
person.setAttributeValue("familyName", "Sinatra");
person.setAttributeValue("firstName", "Frank");
```

Beispiel: Artikel-Objekte über existierende Abfrage "articles" mit Parameter "tag" = "Jazz" suchen und als JSON ausgeben

```javascript
var articles = $k.Registry.query("articles").findElements({tag: "Jazz"});
var json = articles.map(function(element) {
    return {
        name: element.name(),
        type: element.type().name()
    }
});
return JSON.stringify(json)
```
1.8.1.1 API-Referenz

Die API-Referenz ist unter folgender Adresse erreichbar:
http://documentation.i-views.com/k-infinity-api/5.1/kinf-js-api

1.8.1.2 Der Namespace $k

Die meisten API-Objekte und -Funktionen sind im Namensraum $k definiert. Der Namensraum selbst definiert einige hilfreiche Funktionen, z.B.

$k.rootType()
das den Wurzeltyp des Netzes zurückgibt, oder

$k.user()
das den aktiven Benutzer zurückgibt.

1.8.1.3 Registry

Ein weiteres wichtiges Objekt ist das Registry-Objekt $k.Registry. Es erlaubt den Zugriff auf registrierte Ordnerelemente und Objekttypen.

Beispiele:

$k.Registry.type("Article")
gibt den Typ mit dem internen Namen "Article" zurück.

$k.Registry.query("articles")
gibt die Abfrage mit dem Schlüssel "articles" zurück.

Das Registry-Objekt ist ein Singleton, ähnlich wie das Math-Objekt von JavaScript.

1.8.1.4 Mit semantischen Elementen arbeiten

Auf Wissensnetzelemente greift man normalerweise über die Registry oder Abfragen zu.

// Personen-Typ anhand seines internen Namens ermitteln
var personType = $k.Registry.type("Person");

// Suche "articles" mit dem Suchparameter "tag" = "Vocal"
var sailingArticles = $k.Registry.query("articles").findElements({tag: "Vocal"});

Auf die Eigenschaften eines Elements kann man über ihren internen Namen zugreifen:

// Wert des Attributs "familyName"
var familyName = person.attributeValue("familyName");
// Ziel der Relation "bornIn"
var birthplace = person.relationTarget("bornIn");
Die Funktion `name()` liefert den Wert des Namensattributs:

```javascript
var name = birthplace.name();
```

Bei übersetzten Attributen kann die Sprache als 2- oder 3-stelliger ISO 639 Sprachcode angegeben werden. Ohne Sprachangabe wird die aktuelle Sprache der Umgebung verwendet.

```javascript
var englishTitle = book.attributeValue("title", "en");
var swedishTitle = book.attributeValue("title", "swe");
var currentTitle = book.attributeValue("title");
```

### 1.8.1.5 Transaktionen

Zum Anlegen, Ändern oder Löschen von Wissensnetzelementen wird eine Transaktion benötigt. Falls die Transaktionssteuerung durch das Script kontrolliert wird, kann man einen Block in eine Transaktion kapseln:

```javascript
$k.transaction(function() {
    return $k.Registry.type("Article").createInstance();
});
```

Man kann konfigurieren, ob ein Script die Transaktionssteuerung übernimmt, oder ob das gesamte Script in einer Transaktion ausgeführt werden soll. Ausgenommen sind davon Trigger-Skripte, die immer als Teil der schreibenden Transaktion ausgeführt werden.

Bei Nebenläufigkeitskonflikten wird eine Transaktion vom Server zurückgewiesen. In diesem Fall wird eine optionale Callback-Funktion aufgerufen, die man als Argument an `$k.transaction()` über gibt:

```javascript
$k.transaction(
    function() { return $k.Registry.type("Article").createInstance() },
    function() { throw "The transaction was rejected" }
);
```

Transaktionen, wie oben beschrieben, dürfen nicht geschachtelt werden. Es gibt allerdings Fälle, in denen eine Schachtelung nicht vermeidbar ist, weil beispielsweise eine Skriptfunktion sowohl von Funktionen aufgerufen wird, die bereits in einer Transaktion gekapselt sind als auch von Funktionen, für die das nicht gilt. Hier kann eine sogenannte "optimistische Transaktion" eingesetzt werden. Dieses Konstrukt verwendet die äußere Transaktion - sofern vorhanden, oder startet einen neue Transaktion.

```javascript
$k.optimisticTransaction(function() {
    return $k.Registry.type("Article").createInstance();
});
```

Solche Konstruktionen sollten vermieden werden, weil eine Transaktion eine sinnvolle Arbeitseinheit darstellt, welche ganz oder gar nicht ausgeführt wird. Entweder ist die eingebettete für sich alleine sinnvoll und vollständig oder nicht.
Achtung: Eine Fehlerbehandlungsfunktion bei Fehlschlag der optimistischen Transaktion steht nicht zur Verfügung. Sofern eine äußere Transaktion existiert, wird bei Fehlschlag deren Fehlerbehandlung ausgeführt.

1.8.1.6 Elemente modifizieren
1.8.1.6.1 Elemente anlegen

// Neues Objekt vom Typ "Person" erzeugen
var person = $k.Registry.type("Person").createInstance();

// Einen neuen Typ erzeugen
var blogType = $k.Registry.type("CommunicationChannel").createSubtype();
blogType.setName("Blog");

1.8.1.6.2 Attribute hinzufügen und ändern
Attributewerte können mit setAttributeValue() gesetzt werden. Es wird entweder der Wert des bestehenden Attributes geändert oder ein neues Attribut angelegt, falls noch kein Attribut vorhanden ist. Sollten bereits mehrere Attribute vorhanden sein wird eine Exception geworfen.

person.setAttributeValue("familyName", "Sinatra");
person.setAttributeValue("firstName", "Frank");
// Überschreibe den Attributwert "Frank" mit "Francis"
person.setAttributeValue("firstName", "Francis");

Mit createAttribute() können mehrere Attribute desselben Typs hinzugefügt werden, da bestehende Attribute nicht überschrieben werden:

// Zwei Attribute anlegen
person.createAttribute("nickName", "Ol' Blue Eyes");
person.createAttribute("nickName", "The Voice");

Die Attributwerte werden je nach Attributart durch unterschiedliche Objekte repräsentiert:

<table>
<thead>
<tr>
<th>Art des Attributs</th>
<th>Objekttyp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auswahl</td>
<td>$k.Choice</td>
</tr>
<tr>
<td>Boolesch</td>
<td>Boolean</td>
</tr>
<tr>
<td>Datei</td>
<td>$k.Blob</td>
</tr>
<tr>
<td>Datum</td>
<td>$k.Date</td>
</tr>
<tr>
<td>Datum und Uhrzeit</td>
<td>$k.DateTime</td>
</tr>
</tbody>
</table>
1.8.1.6.3 Relationen hinzufügen
Mit createRelation() kann eine Relation zwischen zwei Elementen angelegt werden:

```javascript
var places = $k.Registry.query("places").findElements({name: "Hoboken"});
if (places.length == 1)
    person.createRelation("bornIn", places[0]);
```

1.8.1.6.4 Elemente löschen
Elemente können mit der Funktion remove() gelöscht werden:

```javascript
person.remove();
```

Dabei werden auch alle Eigenschaften des Elements gelöscht.

1.8.2 Beispiele
1.8.2.1 Abfragen
Suche nach Elementen:

```javascript
// Abfrage "articles" mit dem Parameter tag = "Soccer" ausführen
var articles = $k.Registry.query("articles").findElements({tag: "Soccer"});
for (var a in articles)
    $k.out.print(articles[a].name() + "\n");
```

Suche nach Hits. Ein Hit kapselt ein Element und fügt einen Qualitätswert (zwischen 0 und 1) sowie weitere Metainformationen hinzu.
// Abfrage "mainSearch" mit dem Suchwert "Baseball" ausführen
var hits = $k.Registry.query("mainSearch").findHits("Baseball");
hits.forEach(function(hit) {
    $k.out.print(hit.element().name()+" (" + (Math.round(hit.quality() * 100))+ ")\n" );
});

Suchergebnis in JSON umwandeln:

var elements= $k.Registry.query("articles").findElements({tag: "Snooker"});
var json = elements.map(function(element) {
    return {
        name: element.name(),
        id: element.idString(),
        type: element.type().name()
    }
});
$k.out.print(JSON.stringify(json, undefined, "\t"));

1.8.2.2 Zur Laufzeit generierte Abfragen

Die Javascript-API erlaubt es auch, Abfragen dynamisch zu generieren. Hier einige Beispiele aus einem Filmnetz:

Suche nach Filmen mit Jahr + Name

var query = new $k.StructuredQuery("imdb_film");
query.addAttributeValue("imdb_film_year", "year");
query.addAttributeValue("name", "name");
query.findElements({year: "1958", name: "Vert*"});

Dem Constructor wird die Domain übergeben. Bei internen Namen wird automatisch nach Objekten dieses Types gesucht. Mehr Möglichkeiten bietet die Funktion setDomains()

Jahr + Anzahl Regisseure >= 3

var query = new $k.StructuredQuery("imdb_film");
query.addAttributeValue("imdb_film_year", "year");
query.addCardinality("imdb_film_regisseur", 3, ">=");
query.findElements({year: "1958"});

Jahr + Name des Regisseurs

var query = new $k.StructuredQuery("imdb_film");
query.addAttributeValue("imdb_film_year", "year", "=");
var directorQuery = query.addRelationTarget("imdb_film_regisseur").targetQuery();
directorQuery.addAttributeValue("name", "director");
query.findElements({year: "1950", director: "Hitchcock, Alfred"});
Alternativen (Oder-Bedingungen)

var query = new $k.StructuredQuery("imdb_film");
query.addAttributeValue("imdb_film_year", "year");
var alternatives = query.addAlternativeGroup();
alternatives.addAlternative().addAttributeValue("name", "name");
alternatives.addAlternative().addAttributeValue("imdb_film_alternativeTitel", "name");
query.findElements({year: "1958", name: "Vert*"});

Mögliche Operatoren sind:

<table>
<thead>
<tr>
<th>Operator-Name</th>
<th>Kurzbezeichnung</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>containsPhrase</td>
<td></td>
<td>Enthält Phrase</td>
</tr>
<tr>
<td>covers</td>
<td></td>
<td>enthält</td>
</tr>
<tr>
<td>distance</td>
<td></td>
<td>Abstand</td>
</tr>
<tr>
<td>equal</td>
<td>==</td>
<td>Gleich</td>
</tr>
<tr>
<td>equalBy</td>
<td></td>
<td>Entspricht</td>
</tr>
<tr>
<td>equalCardinality</td>
<td></td>
<td>Kardinalität gleich</td>
</tr>
<tr>
<td>equalGeo</td>
<td></td>
<td>Gleich (Geo)</td>
</tr>
<tr>
<td>equalMaxCardinality</td>
<td></td>
<td>Kardinalität kleiner gleich</td>
</tr>
<tr>
<td>equalMinCardinality</td>
<td></td>
<td>Kardinalität größer gleich</td>
</tr>
<tr>
<td>equalPresentTime</td>
<td></td>
<td>gleich jetzt (Gegenwart)</td>
</tr>
<tr>
<td>equalsTopicOneWay</td>
<td></td>
<td>filtern mit</td>
</tr>
<tr>
<td>fulltext</td>
<td></td>
<td>Enthält Zeichenkette</td>
</tr>
<tr>
<td>greater</td>
<td>&gt;</td>
<td>Grösser als</td>
</tr>
<tr>
<td>greaterOrEqual</td>
<td>&gt;=</td>
<td>Grösser/Gleich</td>
</tr>
<tr>
<td>greaterOverlaps</td>
<td></td>
<td>überschneidet von oben</td>
</tr>
<tr>
<td>greaterPresentTime</td>
<td></td>
<td>nach jetzt (Zukunft)</td>
</tr>
<tr>
<td>isCoveredBy</td>
<td></td>
<td>ist enthalten in</td>
</tr>
<tr>
<td>less</td>
<td>&lt;</td>
<td>Kleiner als</td>
</tr>
<tr>
<td>lessOrEqual</td>
<td>&lt;=</td>
<td>Kleiner/Gleich</td>
</tr>
<tr>
<td>lessOverlaps</td>
<td></td>
<td>überschneidet von unten</td>
</tr>
<tr>
<td>lessPresentTime</td>
<td></td>
<td>vor jetzt (Vergangenheit)</td>
</tr>
<tr>
<td>notEqual</td>
<td>!=</td>
<td>Ungleich</td>
</tr>
<tr>
<td>overlaps</td>
<td></td>
<td>überschneidet</td>
</tr>
</tbody>
</table>
### 1.8.2.3 Elemente anlegen und ändern

Eine Person anlegen

```javascript
// Personen-Typ über seinen internen Namen ermitteln
var personType = $k.Registry.type("Person");
// Neue Person anlegen
var person = personType.createInstance();
// Attribuwerte setzen
person.setAttributeValue("familyName", "Norris");
person.setAttributeValue("firstName", "Chuck");
```

Den vollständigen Namen einer Person setzen

```javascript
var familyName = person.attributeValue("familyName");
var firstName = person.attributeValue("firstName");
if (familyName && firstName)
{
    var fullName = familyName + ", " + firstName;
    person.setAttributeValue("fullName", fullName);
}
```

Attributwerte setzen

```javascript
// Boolean
element.setAttributeValue("hasKeycard", true);

// Auswahl - Internen Name des Auswahlwerts angeben
element.setAttributeValue("status", "confirmed");
// Auswahl - Auswahlwert angeben
var choiceRange = $k.Registry.attributeType("status").valueRange();
var choice = choiceRange.choiceInternalNamed("confirmed");
element.setAttributeValue("status", choice);

// Farbe
element.setAttributeValue("hairColor", "723F10");

// Datum / Uhrzeit
```
element.setAttributeValue("dateOfBirth", new $k.Date(1984, 5, 4));
element.setAttributeValue("lastModification", new $k.DateTime());
element.setAttributeValue("teatime", new $k.Time(15, 30, 0));

// Flexible Zeit - $k.FlexTime (ermöglicht ungenaue Zeitangaben)
element.setAttributeValue("start", new $k.FlexTime(1984, 6));
// Flexible Zeit - Date (fehlende Werte werden durch Standardwerte aufgefüllt)
element.setAttributeValue("start", new Date(1984, 5, 3));

// Zahl (Ganzzahl / Fließkomma)
element.setAttributeValue("weight", 73);

// Intervall
element.setAttributeValue("interval", new $k.Interval(2, 4));

// Zeichenkette - nicht übersetzt
element.setAttributeValue("familyName", "Norris");
// Zeichenkette - übersetzt
// Sprache wird als ISO 639-1 or 639-2b Code übergeben
element.setAttributeValue("welcomeMessage", "Welcome", "en");
element.setAttributeValue("welcomeMessage", "Bienvenue", "fre");

Neues Attribut anlegen

person.createAttribute("nickName", "Ground Chuck");

Neue Relation anlegen

var places = $k.Registry.query("places").findElements({name: "Oklahoma"});
if (places.length == 1)
    person.createRelation("bornIn", places[0]);

Ein Element samt Eigenschaften löschen

person.remove()

Eine Zeichenkette in einen Attributwert konvertieren. Der ValueRange eines Attributtyps kennt die erlaubten Werte und kann die Zeichenkette einlesen. Bei ungültigen Zeichenketten wird ein Fehler geworfen.

var statusRange = $k.Registry.type("status").valueRange();
var statusConfirmed = statusRange.parse("Confirmed", "eng");

Änderungs-Metadaten ändern

element.setAttributeValue("lastChangeDate", new Date());
var userInstance = $k.user().instance();
// Verknüpfung mit Benutzer?
if (element.relationTarget("lastChangedBy") !== userInstance)
    { // bestehende Relationen löschen
        var relations = element.relations("lastChangedBy");
        }
for (var r in relations)
    relation[r].remove();
    // Relation zum Benutzer-Objekt ziehen
    element.createRelation("lastChangedBy", userInstance);

1.8.2.4 REST

Ein REST-Script muss eine Funktion `respond()` definieren, die als Argumente die HTTP-Anfrage, die geparsen Anfrage-Parameter und eine leere HTTP-Antwort entgegennimmt. Das Script füllt dann Header und Inhalt der Antwort.

```javascript
function respond(request, parameters, response)
{
    response.setText("REST example");
}
```

Einen Blob herunterladen

```javascript
function respond(request, parameters, response)
{
    var name = parameters["name"];
    if (name)
    {
        var images = $k.Registry.query("rest.image").findElements({"name": name});
        if (images.length == 1)
        {
            // Inhalt und content type aus dem Blob übernehmen
            response.setContents(images[0].value());
            // Inline anzeigen
            response.setContentDisposition("inline");
        }
        else
        {
            response.setCode($k.HttpResponse.BAD_REQUEST);
            response.setText(images.length + " images found");
        }
    }
    else
    {
        response.setCode($k.HttpResponse.BAD_REQUEST);
        response.setText("Name not specified");
    }
}
```

Neues Objekt mit einem hochgeladenen Blob anlegen

```javascript
function respond(request, parameters, response)
{
    var formData = request.formData();
    var name = formData.name;
```
var picture = formData.picture;
if (name && picture)
{
    var city = $k.Registry.type("City").createInstance();
    city.setAttributeValue("image", picture);
    city.setName(name);
    response.setText("Created city " + name);
}
else
{
    response.setCode($k.HttpResponse.BAD_REQUEST);
    response.setText("Parameters missing");
}

1.8.2.5 XML
Suchergebnisse in XML transformieren

function respond(request, parameters, response)
{
    var name = parameters["name"]; 
    if (name)
    {
        // Find points of interest
        var pois = $k.Registry.query("rest.poi").findElements({name: name});
        // Write XML
        var document = new $k.TextDocument();
        var writer = document.xmlWriter();
        writer.startElement("result");
        for (var p in pois)
        {
            writer.startElement("poi");
            writer.attribute("name", pois[p].name());
            writer.endElement();
        }
        writer.endElement();
        response.setContents(document);
        response.setContentType("application/xml");
    }
    else
    {
        response.setCode($k.HttpResponse.BAD_REQUEST);
        response.setContents("Name not specified");
    }
}

XML-Ausgabe

<result>
    <poi name="Plaza Mayor"/>
    <poi name="Plaza de la Villa"/>
}
Qualifizierte Namen verwenden

```javascript
var document = new $k.TextDocument();
var writer = $k.out.xmlWriter();
writer.setPrefix("k", "http://www.i-views.de/kinfinity");
writer.startElement("root", "k");
writer.attribute("hidden", "true", "k");
writer.startElement("child", "k").endElement();
writer.endElement();
```

XML-Ausgabe

```xml
<k:root xmlns:k="http://www.i-views.de/kinfinity" k:hidden="true">
  <k:child/>
</k:root>
```

Standard-Namespace definieren

```javascript
var document = new $k.TextDocument();
var writer = $k.out.xmlWriter();
writer.startElement("root");
writer.defaultNamespace("http://www.i-views.de/kinfinity");
writer.startElement("child").endElement();
writer.endElement();
```

XML-Ausgabe

```xml
<root xmlns="http://www.i-views.de/kinfinity">
  <child/>
</root>
```

### 1.8.2.6 HTTP-Client

In einem Script können auch HTTP-Requests abgeschickt werden.
Beispiel: Bild über HTTP laden und als Blob im Wissensnetz speichern

```javascript
var http = new $k.HttpConnection();
var imageUrl = "http://upload.wikimedia.org/wikipedia/commons/e/e7/2007-07-06_GreatBritain_Portree.jpg";
var imageResponse = http.getResponse(new $k.HttpRequest(imageUrl));
if (imageResponse && imageResponse.code() == $k.HttpResponse.OK ) {
  var portree = $k.Registry.type("City").createInstance();
  portree.setAttributeValue("image", imageResponse);
  portree.setName("Portree");
}
```
Wetterberichte aller Städte aktualisieren

```javascript
var instances = $k.Registry.type("City").instances();
var http = new $k.HttpConnection();
for (var i in instances) {
    var city = instances[i];
    var weatherUrl = "http://api.openweathermap.org/data/2.5/weather";
    var weatherRequest = new $k.HttpRequest(weatherUrl);
    weatherRequest.setQueryData({q: city.name()});
    try {
        var weatherResponse = http.getResponse(weatherRequest);
        if (weatherResponse.code() == $k.HttpResponse.OK) {
            var json = JSON.parse(weatherResponse.text());
            var weather = json.weather[0].description;
            city.setAttributeValue("weather", weather);
        }
    } catch (e) { }
}
```

Um zu verhindern, dass Skripte Requests zu beliebigen Hosts schicken, kann man in der Konfigurationsdatei einer Anwendung eine Whitelist definieren:

```javascript
allowedOutgoingDomains=*.i-views.de,*.intelligent-views.com,ivinternal:8080
```


### 1.8.2.7 E-Mails versenden


```javascript
var mail = new $k.MailMessage();
mail.setSubject("Hello from " + $k.volume());
mail.setText("This is a test mail");
mail.setSender("kinfinity@example.org");
mail.setReceiver("developers@example.org");
mail.setUserName("kinf");
mail.send();
```

Das Benutzerkonto "kinf" wird für die Authentifizierung verwendet. Das Passwort wird in den SMTP-Einstellungen hinterlegt.
1.8.2.8 Views

JSON-Strukturen können auch anhand der View-Konfiguration generiert werden, sowohl für einzelne Objekte als auch Objektlisten.

Im einfachsten Fall wird ein Objekt anhand der Standard-Konfiguration ohne weiteren Kontext in JSON umgewandelt:

```javascript
var data = element.renderJSON();
```

Es werden dann alle durch die Konfiguration definierten Strukturen in JSON umgesetzt:

```json
{
    "viewType": "fieldSet",
    "label": "Bern",
    "elementType": "instance",
    "modNum": 26,
    "elementId": "ID17361_141538476",
    "type": {
        "elementType": "instance",
        "typeId": "ID10336_319205877",
        "internalName": "City",
        "typeName": "Stadt"
    },
    "properties": [{
        "values": [{
            "value": "Bern",
            "propertyId": "ID17361_137824032"
        }],
        "schema": {
            "label": "Name",
            "elementType": "attribute",
            "internalName": "name",
            "maxOccurrences": 1,
            "attributeType": "string",
            "viewId": "ID20838_426818557",
            "typeId": "ID4900_317193164",
            "minOccurrences": 0
        }
    }, {
        "values": [{
            "typeId": "ID4900_79689320"
        }],
        "schema": {
            "label": "Alternativname/Synonym",
            "elementType": "attribute",
            "internalName": "alternativeName",
            "attributeType": "string",
            "rdf-id": "alternativeName",
            "viewId": "ID20839_64952366",
            "typeId": "ID4900_79689320",
            "minOccurrences": 0
        }
    }]
}
```
"minOccurrences" : 0
},

"values" : [
  {
    "target" : {
      "typeId" : "ID10336_493550611",
      "label" : "Kunstmuseum Bern",
      "elementId" : "ID17362_205182965"
    },
    "propertyId" : "ID17361_395925739"
  },
  {
    "target" : {
      "typeId" : "ID10336_493550611",
      "label" : "Schweizerische Nationalbibliothek",
      "elementId" : "ID20401_126870015"
    },
    "propertyId" : "ID17361_9264966"
  }
],

"schema" : {
  "targetDomains" : [
    {
      "elementType" : "instance",
      "typeId" : "ID10336_493550611",
      "internalName" : "point_of_interest",
      "typeName" : "Sehenswürdigkeit"
    }
  ],
  "label" : "beherbergt Sehenswürdigkeit",
  "elementType" : "relation",
  "internalName" : "contains_poi",
  "viewId" : "ID20840_182208894",
  "typeId" : "ID2052_332207092",
  "minOccurrences" : 0
}
}

Zusätzlich kann ein Kontext in Form eines Anwendungs- oder Konfigurationsobjekts angegeben werden. Es wird dann ein zu diesem Kontext passende Konfiguration gewählt. Im folgenden Beispiel wird die Anwendung "Android" vorgegeben:

var application = $k.Registry.elementAtValue("viewconfig.configurationName", "Android");
var data = element.renderJSON(application);

Es ist aber auch möglich, eine Konfiguration vorzugeben und diese das Element umwandeln zu lassen. Dazu erzeugt man eine $k.ViewConfiguration aus dem Konfigurationsobjekt.

var configurationElement = $k.Registry.elementAtValue("viewconfig.configurationName", "Android Article Configuration");
var data = $k.ViewConfiguration.from(configurationElement).renderJSON(element);
Da die JSON-Struktur recht umfangreich ist, kann man auch bestimmte Properties bei der Umwandlung weglassen, indem man die Schlüssel als zusätzlichen Parameter angibt:

```javascript
var application = $k.Registry.elementAtValue("viewconfig.configurationName", "Android");
var data = element.renderJSON(application, ["rdf-id", "viewId", "typeId", "propertyId", "modNum", "maxOccurrences"]);

{
  "viewType": "fieldSet",
  "label": "Bern",
  "elementType": "instance",
  "elementId": "ID17361_141538476",
  "type": {
    "elementType": "instance",
    "internalName": "City",
    "typeName": "Stadt"
  },
  "properties": [
    {
      "values": [
        {"value": "Bern"}
      ],
      "schema": {
        "elementType": "attribute",
        "label": "Name",
        "internalName": "name",
        "attributeType": "string",
        "maxOccurrences": 1
      }
    },
    {
      "schema": {
        "elementType": "attribute",
        "label": "Alternativname/Synonym",
        "internalName": "alternativeName",
        "attributeType": "string"
      }
    },
    {
      "values": [
        {
          "target": {
            "label": "Kunstmuseum Bern",
            "elementId": "ID17362_205182965"
          }
        },
        {
          "target": {
            "label": "Schweizerische Nationalbibliothek",
            "elementId": "ID20401_126870015"
          }
        }
      ]
    }
  ]
}
```
1.8.2.9 Mustache-Templates

Die folgende Funktion erzeugt ein Dokument mit Hilfe der Mustache Template-Bibliothek. Es erwartet folgendes Schema:

- ein Zeichenketten-Attribut (interner Name "template.id"), um das Template zu identifizieren
- ein Dateiattribut (interner Name "template.file") mit dem Template, z.B. ein HTML-Dokument
- Eine Relation zu einem MediaType_objekt (interner Name "template.contentType")

Die Abfrage "rest.articles" gibt alle Element zurück die dargestellt werden sollen. Die Mustache-Bibliothek ist unter "mustache.js" registriert.

```javascript
function respond(request, parameters, response) {
    // Mustache einbinden
    $k.module("mustache.js");

    // Template ermitteln
    var templateId = parameters["templateId"];
    var template = $k.Registry.elementAtValue("template.id", templateId);
    var templateText = template.attributeValue("template.file").text("utf-8");

    // Darzustellende Elemente suchen
    var elements = $k.Registry.query("rest.articles").findElements(parameters);

    // Template-Parameter vorbereiten
    var templateParameters = {
        data: elements.map(function(element) {
            return {
                name: element.name(),
            };
        })
    }

    // Mustache erzeugen
    var renderedTemplate = mustache.render(templateText, templateParameters);

    // Prüfen, ob das Template korrekt ist
    if (renderedTemplate) {
        response.writeHead(200, { "Content-Type": "text/html; charset=utf-8" });
        response.end(renderedTemplate);
    } else {
        response.writeHead(500, { "Content-Type": "text/plain; charset=utf-8" });
        response.end("Mustache Template konnte nicht erzeugt werden.");
    }
}
```
1.8.2.10 Java Native Interface

Mit Hilfe von JNI (Java Native Interface) können Java-Bibliotheken eingebunden werden.

**Warnung:** JNI ist ein experimentelles Feature und hat einige Einschränkungen:

- JNI kann nicht in Triggern verwendet werden
- Es ist nicht möglich, Klassen zu definieren (beispielsweise für Callbacks)
- Generics werden nicht unterstützt
- JNI erlaubt den Zugriff auf Systemressourcen (z.B. Dateien)

JNI muss in den Konfigurationsdateien aller Anwendungen eingerichtet werden, die diese Skripte verwenden. Der Klassenpfad kann nicht zur Laufzeit geändert werden.

```
[JNI]
classPath=tika\tika-app-1.5.jar
libraryPath=C:\Program Files\Java\jre7\bin\server\jvm.dll
```


Beispiel

```
// Import the StringBuilder class, without namespace
$jni.use("[java.lang.StringBuilder]", false);
// Create a new instance
var builder = new StringBuilder();
// Javascript primitives and Strings are automatically converted
builder.append("Welcome to ");
builder.append($k.volume());
// toJS() converts Java objects to Javascript objects
$k.out.print(builder.toString().toJS());
```

Text/Metadaten-Extraction mit Apache Tika
$jni.use(
    "java.io.ByteArrayInputStream",
    "java.io.BufferedInputStream",
    "java.io.StringWriter",
    "org.apache.tika.parser.AutoDetectParser",
    "org.apache.tika.metadata.Metadata",
    "org.apache.tika.parser.ParseContext",
    "org.apache.tika.sax.BodyContentHandler"
], false);

// Get a blob
var blob = $k.Registry.elementAtValue("uuid", "f36db9ef-35b1-48c1-9f23-1e10288fddf6").attributeValue("ebook");

var bufferedInputStream = new BufferedInputStream(new ByteArrayInputStream($jni.toJava(blob)));

// Parse the blob
try {
    var parser = new AutoDetectParser();
    var writer = new StringWriter();
    var metaData = new Metadata();
    parser.parse(bufferedInputStream, new BodyContentHandler(writer), metaData, new ParseContext());

    var string = writer.toString().toJS();

    // Print extracted metadata
    var metaNames = metaData.names().toJS().sort(
        function(a,b) { return a.localeCompare(b) });
    for (n in metaNames)
        $k.out.print(metaNames[n] + " = " + metaData.get(metaNames[n])).cr();

    // Print extracted text (first 100 chars)
    $k.out.cr().cr().print(string.substring(1, 100) + " [...]

} catch (e) {
    $k.out.print("Extraction failed: " + e.toString());
} finally {
    bufferedInputStream.close();
}

1.8.3 Module

1.8.3.1 Module definieren

Ein Modul wird mit der Funktion define() definiert. Als Argument übergibt man entweder ein
Modulobjekt oder eine Funktion, die ein Modulobjekt zurückgibt. Ein Script sollte nur ein
Modul definieren.

Beispiel: Modul mit einer Funktion jsonify()

$k.define({
    // Create a JSON object array for the elements
    jsonifyElements: function(elements) {
        return elements.map(function(element) {
            return {
                name: element.name(),
            }
        });
    }
});
id: element.idString(),
type: element.type().name()  
});
});
}
}

Module können auch von anderen Modulen abhängig sein. Das folgende Script definiert ein Modul, das ein anderes ("rest.common") verwendet.

$k.define(["rest.common"], function(common) {
    return {
        stringifyElements: function(elements) {
            return JSON.stringify(common.jsonifyElements(elements), undefined, "\t")
        }
    }
});

### 1.8.3.2 Module verwenden

Ein Modul kann entweder mit `require()` oder `module()` verwendet werden. `require()` erwartet einen Array von Modulnamen und eine Callback-Funktion. Die Callback-funktion wird mit den Modulen ausgeführt.

```javascript
var elements = $k.Registry.query("rest.poi").findElements({name: "Madrid");
var json = $k.require(["rest.common"], function(common) {
    return common.jsonifyElements(elements);
});
```

`module()` erwartet den Namen eines Moduls und gibt das Modulobjekt zurück.

```javascript
var json = $k.module("rest.common").jsonifyElements(elements);
```

`module()` kann auch mit Skripten verwendet werden, die keine Module definieren. Das Script wird ausgeführt und alle deklarierten Funktionen instanziert. Diese Funktionen können anschließend aufgerufen werden.

### 1.8.3.3 AMD

Um JavaScript-Bibliotheken einzubinden, die den AMD-Standard unterstützen, muss man vorher `define()` und `require()` global definieren:

```javascript
this.define = $k.define;
this.define.amd = {};
this.require = $k.require;
```

Falls eine Bibliothek ein Modul mit einer bestimmten ID definiert und man diese Bibliothek unter einem anderen Namen registrieren möchte, kann man Module-IDs auf Registratur-IDs abbilden:
$k.mapModule("underscore", "lib.underscore");

Nun kann man underscore.js als "lib.underscore" registrieren und das fort definierte Modul "underscore" verwenden.

1.8.4 Debugger

1.8.4.1 Debuggen im Knowledge-Builder


// Request und Response vorbereiten
var testRequest = new $k.HttpRequest("http://localhost");
var testResponse = new $k.HttpResponse();
// Restlet-Funktion respond() mit den Test-Parametern ausführen,
// die als Property k.testbenchParameters verfügbar sind
respond(testRequest, $k.testbenchParameters, testResponse);
// Response ausgeben
$k.out.print(testResponse.debugString());

Auf dem Reiter "Debug" kann man Breakpoints setzen und das Script in Einzelschritten ausführen.

1.8.4.2 Remote Debugger

Restlet-Skripte können auch mit einem Remote Debugger analysiert werden.

1. Debug-Port der Konfigurationsdatei der Bridge aktivieren

[KRemoteDebuggerBridge]
port=9000

2. Eclipse und JavaScript Development Tools installieren

4. Debugger in Eclipse starten
5. Im Kontextmenü "Show all scripts" auswählen, damit alle Skripte der Bridge angezeigt werden.

6. "Open source" im Kontextmenü eines Skripts auswählen (Doppelklick scheint dagegen keine Funktion zu haben)
7. Im Skript können nun Breakpoints gesetzt werden

1.8.5 API-Erweiterungen
1.8.5.1 Zusätzliche Funktionen

Die API kann erweitert werden, indem eigene Funktionen bei Prototyp-Objekten hinzufügt. Im folgenden Beispiel werden einige Prototypen erweitert, um Schemeinformation mit printSchema() auszugeben.

```javascript
// Print the schema of the instances and subtypes of a type
$k.Type.prototype.printSchema = function() {
  this.typeDomain().printSchema("Type schema of "+ this.name() + \\
  this.domain().printSchema("Instance schema of "+ this.name() + \\
  this.subtypes().forEach(function(subtype) {
    subtype.printSchema();
  });
}

// Print information about a property type
$k.PropertyType.prototype.logPropertySchema = function() {
  $k.out.print("t" + this.name() + \\
  this.valueRange().valueType() + \
}

// Attribute types print their type
$k.AttributeType.prototype.logPropertySchema = function() {
  $k.out.print("t" + this.name() + " (Attribute of type " + this.valueRange().valueType() + \\
}

// Relation types print their target domains
$k.RelationType.prototype.logPropertySchema = function() {
  $k.out.print("t" + this.name());
  var inverse = this.inverseRelationType();
  if (inverse) {
    var inverseDomains = inverse.domains();
    if (inverseDomains.length > 0 ) {
      $k.out.print(" (Relation to ");
      var separate = false;
      inverseDomains.forEach(function(inverseDomain) {
        if (separate)
          $k.out.print("", "); else
            separate = true;
        $k.out.print("\n + inverseDomain.type().name() + "\n";
      });
      $k.out.print("\n");
    }
  }
  $k.out.cr();
}

// Print all properties defined for a domain
$k.Domain.prototype.printSchema = function(label) {
  var definedProperties = this.definedProperties();
  if (definedProperties.length > 0)
  {
```
1.8.5.2 Eigene Prototypen definieren

Der Prototyp eines Wissensnetzelements ist normalerweise vordefiniert (Instance, Relation usw.). Es ist aber auch möglich, eigene Prototypen zu definieren und Objekten von bestimmten Typen mit der Funktion mapInstances(internalName, prototype) zuzuordnen.

Beispiel: Ein Warenkorb-Prototyp

// Definiere einen Prototyp mit der Funktion totalPrice()
function Basket() {}

Basket.prototype.totalPrice = function() {
    return this.relationTargets("contains").reduce(
        function(sum, item) {
            return sum + item.attributeValue("price");
        },
        0);
};

// Den Prototyp allen Objekten vom Typ "Basket" zuordnen
$k.mapInstances("Basket", Basket);

Für die Verwendung in anderen Skripten muss zunächst das Modul geladen werden:

$k.module('myBasketSkript');
var basket = $k.Registry().elementWithID('ID_123');
$k.out.print(basket.totalPrice() + "\n");

1.9 REST-Services

Über die REST-Schnittstelle kann lesend und schreibend auf das Wissensnetz zugegriffen werden. Dazu definiert man im Wissensnetz Ressourcen (beschreiben das Verhalten der Schnittstelle beim Zugriff auf eine Ressource) und Services (fassen mehrere Ressourcen zusammen).

Das Verhalten einer Ressource wird über Skripte gesteuert. Zusätzlich können auch vordefinierte
Ressourcen verwendet werden.
Der Zugriff erfolgt über HTTP-Requests, die nach dem Muster

```
https://<hostname>:<port>/<service-id>/<resource-pfad-und-parameter>
```
aufgebaut sind.

### 1.9.1 Konfiguration

Im Wissensnetz muss die REST-Komponente hinzugefügt werden. Diese definiert das notwendige Schema, das man im Knowledge-Builder im Bereich "Technik "-> "REST" findet.


Der Bridge-Dienst benötigt eine passende Konfigurationsdatei (bridge.ini). In diese trägt man die Namen des Servers (host), des Wissensnetzes (volume) und der REST-Service-IDs ein. Die Zeile mit "services" kann auch komplett weggelassen werden, dann werden die Ressourcen aller vorhandenen Service-Objekte automatisch aktiviert.

```
[Default]
host=localhost
loglevel=10

[KHTTPRestBridge]
volume=demo
port=8086
services=core,extra
```

### 1.9.2 Services

Services fassen mehrere Ressourcen zusammen. Ressourcen können in mehreren Services enthalten sein.

Der Services-Editor im Knowledge-Builder zeigt in seiner Strukturansicht die Ressourcen an. Mit "Neues verknüpfen" wird eine neue Ressource angelegt und zum Service hinzugefügt. Mit "Bestehendes verknüpfen" wird eine bereits definierte Ressource zum Service hinzugefügt.

### 1.9.3 Ressourcen

Ressourcen beschreiben das Verhalten bei einer HTTP-Anfrage an die Schnittstelle. Es gibt folgende Arten von Ressourcen:

<table>
<thead>
<tr>
<th>Ressource</th>
<th>Beschreibung</th>
</tr>
</thead>
</table>

...
Eine Ressource hat folgende konfigurierbare Eigenschaften:

<table>
<thead>
<tr>
<th>Eigenschaft</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path pattern</td>
<td>Definiert die URL der Ressource relativ zur Adresse des Services. Der Pfad kann parametrisiert werden, indem man Parameter in geschweiften Klammern hinzufügt: albums/{genre}\n</td>
</tr>
<tr>
<td>Part of service</td>
<td>Services, die diese Ressource verwenden</td>
</tr>
<tr>
<td>Description</td>
<td>Beschreibung zu Dokumentationszwecken</td>
</tr>
<tr>
<td>Requires authentication</td>
<td>Für den Zugriff auf die Ressource ist eine Authentifizierung notwendig</td>
</tr>
</tbody>
</table>

**1.9.3.1 Methoden**

Eine Ressource wird mit einer oder mehreren Methoden verknüpft. Diese definiert das Verhalten sowie die unterstützten Ein- und Ausgabetypen (Content-Type). Anhand der Methoden und Typen der HTTP-Anfrage wird eine passende konfigurierte Methode ausgewählt.

In der Strukturansicht werden Methoden als Unterelemente von Ressourcen angezeigt und können dort angelegt/gelöscht werden.
<table>
<thead>
<tr>
<th>Methode</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input media type</td>
<td>Nur POST/PUT: Erwarteter Content-Type des Inhalts der Anfrage.</td>
</tr>
<tr>
<td>Output media type</td>
<td>Content-Type der Antwort. Falls die Anfrage per &quot;Accept&quot; erwartete Content-Typen vorgibt, muss der Output media type dazu passen.</td>
</tr>
<tr>
<td>Script</td>
<td>Registriertes Skript zur Definition des Verhaltens (nur bei Script-Ressourcen relevant)</td>
</tr>
<tr>
<td>Transaction</td>
<td>Transaktionssteuerung (nur bei Script-Ressourcen relevant)</td>
</tr>
</tbody>
</table>

Die Transaktionssteuerung ist für schreibenden Zugriffe auf das Wissensnetz relevant, da diese nur innerhalb einer Transaktion möglich sind.

<table>
<thead>
<tr>
<th>Transaktionssteuerung</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>Bei GET nur lesender Zugriff, bei POST/PUT/DELETE wird das Skript in einer Transaktion ausgeführt. Dies ist die Standardinstellung.</td>
</tr>
<tr>
<td>Controlled by script</td>
<td>Keine Transaktion, das Skript muss diese selbst steuern.</td>
</tr>
<tr>
<td>Read</td>
<td>Nur lesender Zugriff, das Skript kann keine Transaktion starten.</td>
</tr>
<tr>
<td>Write</td>
<td>Das Skript wird in einer Transaktion ausgeführt.</td>
</tr>
</tbody>
</table>

1.9.3.2 Script-Ressource

Durch ein Skript wird bei einer Methode einer Script-Ressource die Antwort auf eine HTTP-Anfrage definiert. Von der Schnittstelle wird dazu die Funktion respond(request, parameters, response) aufgerufen, die im Skript definiert werden muss.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Typ</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>$k.HttpRequest</td>
<td>Anfrage (URL, Header, usw.)</td>
</tr>
<tr>
<td>parameters</td>
<td>Object</td>
<td>Aus dem Request extrahierte Parameter</td>
</tr>
<tr>
<td>response</td>
<td>$k.HttpResponse</td>
<td>Antwort</td>
</tr>
</tbody>
</table>
Die Funktion füllt dann Header und Inhalt der Antwort. Einen Rückgabewert gibt es nicht.
Wenn für einen Parameter ein Typ definiert wurde (z.B. xsd:integer), wird der konvertierte Wert übergeben, ansonsten eine Zeichenkette. Bei Parametern, die laut Definition mehrfach vorkommen können, werden diese immer als Array übergeben.
Wenn in der Methode ein Output Content-Type für die Antwort definiert wurde, wird dieser automatisch gesetzt. Alternativ kann der Content-Type auch im Skript definiert werden.
Das folgende Skript sucht Alben und wandelt diese in JSON-Objekte um. Die Parameter der Ressource werden an als Suchparameter an die Abfrage weitergereicht.

```javascript
function respond(request, parameters, response) {
    var albums = $k.Registry.query("albums").findElements(parameters);
    var albumData = albums.map(function(album) {
        return {
            name: album.name(),
            id: album.idString(),
        });
    response.setText(JSON.stringify(albumData, undefined, "\t");
    response.setContentType("application/json");
}

Diese Skript könnte man z.B. mit einer Ressource
albums/{genre}/{year}
verwenden und in der Abfrage "albums" die Suchparameter "genre" und "year" als Suchbedingungen verwenden.

1.9.3.3 Built-In Resourcen


<table>
<thead>
<tr>
<th>Rest resource ID</th>
<th>Methode</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlobResource</td>
<td>GET</td>
<td>Gibt den binären Inhalt eines bestehenden Blob-Attributes zurück.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Das Blob-Attribut wird über den Query-Parameter &quot;blobLocator&quot; identifiziert.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional kann über den Parameter &quot;allowRedirect&quot; festgelegt werden, das Blobs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nicht direkt vom Blobservice geholt werden dürfen (Fixed-Value: false).</td>
</tr>
</tbody>
</table>
Ab i-views 4.1 kann noch ein Java-Script (*rest.preprocessScript*) an die Ressource angebracht werden. Die darin enthaltene Funktion `preprocessParameters ( parameters, request )` kann die Parameter ergänzen. Aus den übergebenen Parametern kann so etwa der noch fehlende blobLocator (bzw. das zugehörige Blobattribut) ermittelt werden, was sonst einen zusätzlichen Script-Ressource-Aufruf benötigen würde.

**BlobResource**


**Download**

Über die Methode "GET" kann man den binären Inhalt eines bestehenden Datei-Attributes herunterladen. Das Datei-Attribut wird über den Query-Parameter "blobLocator" identifiziert.

**Upload**

Beim Upload identifiziert der Parameter "blobLocator" entweder ein existierendes Datei-Attribut oder ein potentielles (d.h. neu anzulegendes) Datei-Attribut. Die Syntax für ein potentielles Attribut hat die Form: "PP--ID1_115537458--ID36518_344319903", wobei die erste ID das Wissensnetzelement und die zweite ID den Attribut-Prototyp repräsentiert.

Die Binärdaten können wahlweise als Multi- oder Singlepart übertragen werden. Bei Multi-
part können potentiell mehrere Dateien gleichzeitig hochgeladen werden, was natürlich nur Sinn macht, wenn jede Datei in ein neu anzulegendes Datei-Attribut geschrieben wird. In jedem Fall ist zu jeder übertragenden Datei der Dateiname zu setzen.

Der optionale Parameter "binaryKey" definiert den form-key, unter dem die Binärdaten im MultiPart übertragen werden.

Setzt man den optionalen booleschen Parameter "uploadOnly" auf "true", dann werden nur die Binärdaten hochgeladen jedoch nicht ins Datei-Attribut geschrieben. Dieser Modus wird im Zusammenspiel mit dem ViewConfig-Mapper verwendet. Rückgabe ist in diesem Fall der JSON-Wert (fileName, fileSize, binaryContainerId), der dann in einem zweiten Schritt über den Mapper in das Attribut geschrieben werden kann. Der Content-Type der Rückgabe des JSON-Werts ist normalerweise "application/json", kann aber über den Parameter "overrideContentType" auf einen anderen Wert gesetzt werden, falls der Browser (z.B. IE) Probleme damit hat.

**Topic Icon**

Über den folgenden Pfad kann man die Bilddatei zu einem gegebenen Topic laden. Wenn ein Individuum keine eigene Bilddatei hat, wird auf die Bilddatei des Typs zurückgegriffen, welche wiederum vererbar ist. Über den optionalen Parameter "size" kann man die Bilddatei mit der am ehesten passenden Größe selektieren, vorausgesetzt im Wissensnetz sind mehrere Bildgrößen hinterlegt.

```
http://{server:port}/baseService/topicIcon/{topicId}?size=10
```

**Objektliste**

Über den folgenden Pfad kann eine Objektliste im JSON-Format anfordert werden:

```
http://{server:port}/baseService/{conceptLocator}/objectList
```

Der Typ der Objektliste wird über den Parameter "conceptLocator" referenziert, dem Form- für Topic-Referenzen in der Rest-URL folgt. (siehe Verknüpfung)

Alternativ kann der "conceptLocator" auch den einen Prototyp (Individuum oder Typ) des zu verwendenden Typs referenzieren.

Der optionale Parameter "name" bestimmt die Objektliste, die für die Ausgabe verwendet werden soll.

**Filter**

Über den optionalen und mehrwertigen Query-Parameter "filter" kann die Objektliste gefiltert werden. Ein Filter hat zwei mögliche Formen:

1. `<Spalten-Name/Spalten-Nr.> ~ <Operator> ~ <Wert>
2. `<Spalten-Name/Spalten-Nr.> ~ <Wert>

Mögliche Operatoren sind: equal, notEqual, greater, less, greaterOrEqual, lessOrEqual, equalCardinality, containsPhrase, covers, isCoveredBy, distance, fulltext, equalGeo, equalPresentTime, greaterOverlaps, greaterPresentTime, lessOverlaps, lessPresentTime, equalMaxCardinality, overlaps, unmodifiedEqual.

**Sortierung**

Über den optionalen und mehrwertigen Query-Parameter "sort" kann die Objektliste sortiert werden. Die Reihenfolge der Sortierparameter bestimmt die Sortierungspriorität. Die Angabe der Sortierung hat zwei mögliche Formen:
1. <Spalten-Name>
2. {:<Spalten-Nr.>}

Stellt man in Variante 2 ein Minus vor, wird absteigend sortiert, sonst aufsteigend.

**Startmenge der Liste setzen**

Über den optionalen Query-Parameter "elements" kann eine Komma-separierte Liste von Topic-Referenzen übergeben werden, die als Listenelemente verwendet werden sollen.

Da die Liste der Element ggf. sehr lang ist, kann der Request auch als POST geschickt und die Parameter als Form-Parameter übergeben werden.

**Startmenge der Liste über KPath setzen**

Über die optionalen Query-Parameter "elementsPath" und "startTopic" können die initialen Elemente der Liste berechnet werden. Sind diese Parameter nicht gegeben, besteht die Ausgangsmenge aus allen Individuen bzw. allen Untertypen (im Falle einer Typ-Objektliste) des per "conceptLocator" festgelegten Typs.

Dabei ist "elementsPath" ein KPath-Ausdruck und "startTopic" eine Referenz auf das Topic, mit dem die Auswertung des KPath gestartet werden soll. Die Form des Parameters "startTopic" entspricht der des "conceptLocator".

**Vererbung**

Über den optionalen Query-Parameter "disableInheritance" kann die Vererbung unterdrückt werden. Der Parameter macht nur Sinn, wenn kein "elementsPath" gesetzt ist.

**JSON-Ausgabeformat (Beispiel)**

```json
{
  rows: [{
    topicID: "ID123_987654321",
    row: ["MM",
      "Mustermann",
      "Max",
      "111",
      "m.mustermann@email.net",
      "10",
      "6",
      "2000-01-01",
      "Projekt A, Projekt B"]
  },
  topicID: "ID987_123456789",
  row: ["MF",
    "Musterfrau",
    "Maxine",
    "222",
    "m.musterfrau@email.net",
    "10",
    "8",
    "2000-01-01",
    "Projekt X, Projekt Y, Projekt Z"]
}]
```
Objektlistentendruckvorlage
Über den folgenden Pfad kann eine Objektliste in eine 'Druckvorlage für Liste' gefüllt und das Resultat heruntergeladen werden:

http://{server:port}/baseService/{conceptLocator}/objectList/printTemplate/{templateLocator}/{filename}
Der Service funktioniert genau wie der Abruf einer Objektliste, trägt aber als zusätzlichen Parameter eine Referenz auf das Individuum des Typs 'Druckvorlage für Liste' im Wissensnetz.

"templateLocator" muss eines der unter "Allgemeines" beschriebenen Formate haben

Der optionale Pfad-Parameter "filename" wird nicht ausgewertet und dient dem besseren Browser-Handling.


Über den optionalen Query-Parameter "targetMimeType" kann der Wert des "Accept" Header-Fields überschrieben werden. Dies ist notwendig, wenn man den Request aus einem Browser aufrufen möchte und dort keinen Einfluss auf die Header-Fields hat.

**Topic drucken**

Über den folgenden Pfad kann ein Topic in ein Drucklistentemplate gefüllt und das Resultat heruntergeladen werden:

http://{server:port}/baseService/{topicLocator}/printTemplate/
{templateLocator}/{filename}

"templateLocator" muss eines der unter "Allgemeines" beschriebenen Formate haben

Der optionale Pfad-Parameter "filename" wird nicht ausgewertet und dient dem besseren Browser-Handling.


Über den optionalen Query-Parameter "targetMimeType" kann der Wert des "Accept" Header-Fields überschrieben werden. Dies ist notwendig, wenn man den Request aus einem Browser aufrufen möchte und dort keinen Einfluss auf die Header-Fields hat.

**Dokumentformatkonvertierung**

Über den folgenden Pfad kann ein Dokument in ein anderes Format umgewandelt werden (z.B. odt in pdf):

http://{server:port}/baseService/jodconverter/service

Der Service bildet den JOD-Konverter (siehe http://sourceforge.net/projects/jodconverter/) ab und dient der Abwärtskompatibilität für Installationen, die bisher mit dem JOD-Konverter betrieben wurden.

Damit der Service funktioniert muss Open/LibreOffice (ab Version 4.0) installiert sein und die Konfigurationsdatei "bridge.ini" muss einen Eintrag enthalten, der auf die Datei "soffice" verweist:

[file-format-conversion] sofficePath="C:\Program Files (x86)\LibreOffice 4.0\program\soffice.exe"

1.9.3.4 **Static File Resource**

Liefert Dateien aus dem Dateisystem aus.

Bei dieser Art von Ressource legt man lediglich per *Path pattern* das Verzeichnis fest, unterhalb dessen Dateien ausgeliefert werden. Die Adressierung des Verzeichnisses erfolgt relativ zum Installationsverzeichnis der REST-Bridge.
Beispiel:
Gegeben sei ein Verzeichnis *icons* mit der Datei *bullet.png*. Das Path-Pattern der Ressource lautet *icons*, der dazugehörige Service hat die Service ID *test*. Der Zugriff auf die Datei *bullet.png* lautet dann:

http://localhost:8815/test/icons/bullet.png

1.9.3.5 Ressourcen Parameter

Unterhalb von Methoden kann man die Parameter der Ressource definieren. Dies ist nicht zwingend erforderlich, hat aber einige Vorteile:

- Durch Typangaben kann man Parameter prüfen und konvertieren (z.B. in Zahlen oder Objekte)
- Dokumentation für Kunden

Folgende Parameter-Eigenschaften können konfiguriert werden:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Name des Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style</td>
<td>Art des Parameters</td>
</tr>
<tr>
<td></td>
<td>- path (Teil des Pfads der URL)</td>
</tr>
<tr>
<td></td>
<td>- query (Query-Parameter der URL)</td>
</tr>
<tr>
<td></td>
<td>- header (HTTP-Header)</td>
</tr>
<tr>
<td>Type</td>
<td>Datentyp des Parameters. Parameter werden validiert und</td>
</tr>
<tr>
<td></td>
<td>umgewandelt an das Skript übergeben.</td>
</tr>
<tr>
<td>Repeating</td>
<td>Parameter darf mehrfach vorkommen.</td>
</tr>
<tr>
<td></td>
<td>Wenn aktiviert wird immer eine Array von Werten an das Skript</td>
</tr>
<tr>
<td></td>
<td>übergeben, selbst wenn nur ein Parameterwert in der Anfrage</td>
</tr>
<tr>
<td></td>
<td>vorhanden ist.</td>
</tr>
<tr>
<td>Required</td>
<td>Parameter muss angegeben werden</td>
</tr>
<tr>
<td>Fixed value</td>
<td>Standardwert, falls kein Parameter angegeben wurde.</td>
</tr>
</tbody>
</table>

1.9.4 CORS

Bei OPTIONS-Requests antwortet die REST-Schnittstelle standardmäßig mit

Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept

In der Konfigurationsdatei (bridge.ini) können diese Header konfiguriert werden:

[HTTPRestBridge]
accessControlAllowOrigin=http://*.i-views.de
accessControlAllowHeaders=Origin, X-Requested-With,Content-Type, Accept

1.10  Berichte und Drucken

Mit Hilfe der Druckkomponente kann man Dokumentvorlagen (ODT/DOCX/XLSX/RTF-Dateien) mit KPath-Ausdrücken auf Objekten oder Objektlisten anwenden und daraus eine angepasste Ausgabe-Datei generieren, die entweder gedruckt oder gespeichert werden kann.


1.10.1  Druckvorlagen erstellen


Das folgende Beispiel zeigt eine ODT-Druckvorlage für Objekte des Typs "Task".

Die folgenden Kapitel erläutern das Erstellen der Druckvorlagen-Dokumente.

1.10.1.1  RTF-Vorlagen erstellen

Die RTF-Vorlagedateien können auswertbare KPath-Ausdrücke mit den Schlüsselworten KPATH_EXPAND und KPATH_ROWS sowie Aufrufe registrierter KSkripte mit den Schlüsselworten KSCRIPT_EXPAND
und **KSCRIPT_ROWS** enthalten. Die Pfadausdrücke bzw. der Name des aufzurufenden Skriptes stehen immer zwischen spitzen Klammern und nach dem Schlüsselwort durch ein Leerzeichen getrennt.

**KPATH_EXPAND**


**Beispiel:**

Die Vorlage sei:

Absender:

\$KPATH_EXPAND\ $adresse$/rawValue()$

Nach Auswertung steht in der Ausgabedatei:

Absender:

intelligent views gmbh
Julius-Reiber-Str. 17
64293 Darmstadt

**KSCRIPT_EXPAND**


**Beispiel:**

Die Vorlage sei:

\$KSCRIPT_EXPAND\ einSkriptDas1bis9Ausgibt$

Nach Auswertung steht in der Ausgabedatei:

123.456.789

**KPATH_ROWS**


**Beispiel:**

Die Vorlage sei:

<table>
<thead>
<tr>
<th>Teile (&lt;KPATH_EXPAND topic()~/$hatTeile$/size()&gt; Stück)</th>
<th>Bemerkung</th>
</tr>
</thead>
</table>

Nach Auswertung in der Ausgabedatei:

<table>
<thead>
<tr>
<th>Teile (3 Stück)</th>
<th>Bemerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTF-Druck</td>
<td></td>
</tr>
<tr>
<td>ODT-Druck</td>
<td>Ersetzt den RTF-Druck</td>
</tr>
<tr>
<td>Konvertierungsservice</td>
<td>Optionaler Dienst</td>
</tr>
</tbody>
</table>

**KSCRIPT_ROWS**

Bei KSCRIPT_ROWS werden die Objekte für die Tabellenzeilen durch ein registriertes KSekript ermittelt. Der Name des registrierten Skriptes wird direkt hinter KSCRIPT_ROWS angegeben. Das Skript muss vom Typ KSkript sein und die auszugebenden Objekte zurückgeben.

**Beispiel:**
Die Vorlage sei:

<table>
<thead>
<tr>
<th>Spalte1</th>
<th>Spalte2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;KSCRIPT_ROWS @$nachname$&gt;</td>
<td>&lt;KPATH_EXPAND @Vorname$&gt;</td>
</tr>
</tbody>
</table>

Nach Auswertung in der Ausgabedatei:

<table>
<thead>
<tr>
<th>Spalte1</th>
<th>Spalte2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meier</td>
<td>Peter</td>
</tr>
<tr>
<td>Schulze</td>
<td>Helmut</td>
</tr>
</tbody>
</table>

**1.10.1.2 ODT-Dokumente (OpenOffice) erstellen**

Der Druck über das ODT-Format (Open Document Text, offener Standard) hat viele Vorteile gegenüber dem RTF-Format:

- Das ODT-Format unterstützt eine große Menge an Formatanweisungen (vergleichbar
mit MS-Word), die RTF nicht kennt.

- RTF hat als Format keine einheitliche Normierung (MS-Word kann z.B. "mehr" RTF als der Standard).
- Die Bearbeitung der RTF-Vorlagen ist sehr fragil. Vor allem MS-Word neigt dazu, die Vorlagen mit Steuerelementen (wie z.B. die aktuelle Cursorposition bei der letzten Bearbeitung) zu ergänzen, sodass die Skripte nicht mehr verlässlich erkannt werden können.


**In LibreOffice 5 lässt sich das Skriptfeld nicht länger einbinden.** Als Alternative hierzu kann das Feld "Eingabefeld" benutzt werden:

Einfügen > Feldbefehl > Weitere Feldbefehle (alternativ Tastenkombination Strg+F2)

Dort findet sich im Reiter "Funktionen" das Eingabefeld.
"Hinweis" entspricht dem vorherigen "Script-Typ"; nach einem Klick auf Einfügen öffnet sich ein weiteres Fenster, in das das Skript eingetragen werden kann.

Verfügbare Skript-Typen
Als Skript-Typen gibt es:

- **KPath** : analog zu KPATH_EXPAND
- **KScript** : analog zu KSCRIPT_EXPAND
- **KPathRows** : analog zu KPATH_ROWS
- **KPathImage** : zur Einbettung von Bildern
- **ScriptFunction** : Aufruf einer Funktion eines registrierten Scripts. Als Text wird eine Zeichenkette mit folgendem Format erwartet:

  ScriptID->Funktionsname()

Der Funktionsaufruf wird automatisch um zwei Argumente erweitert: das semantisch Elemente und die durch die Umgebung vorgegebenen Variablen
Ein Beispiel für ein aufgerufenes Skript:

```javascript
function headerLabel(element, variables)
{
    return element.name().toLocaleUpperCase();
}
```

- **ScriptRowsFunction**: Analog zu ScriptFunction. Für die zurückgegebenen Objekte werden analog zu KPathRows Tabellenzeilen erzeugt.
- **ScriptImageFunction**: zum Einfügen von Bitmap-Images
- **ScriptSVGImageFunction**: zum Einfügen von SVG-Zeichnungen

Beispiel zum Erzeugen des JSON-Objekts:

```javascript
function templateData(element)
{
    return {
        name: element.name(),
        idNumber: element.idNumber(),
        someData: { idString: element.idString() }
    }
}
```

Um zum Beispiel auf den Wert idString zugreifen, muss als Text `someData.idString` gesetzt sein.

- **DataRowsPath**: Analog zu DataPath. Als Text wird ein Schlüssel erwartet, dessen Wert im JSON ein Array von Objekten ist. Nachfolgende DataPath-Felder beziehen sich auf die Objekte in diesem Array.


Eingebettete Bilder werden immer auf ihre Originalgröße (bei 96 dpi) gezogen. Möchte man im Ausdruck eine andere Größe erhalten, muss man einen Rahmen mit den gewünschten Ausmaßen (unbedingt absolute Maße in cm verwenden!) um das Skript-Element bauen. Das resultierende eingebettete Bild wird dann so in den Rahmen eingepasst, dass das Rahmenmaß unter Beibehaltung der Bild-Seitenverhältnisse nicht überschritten wird.

### 1.10.1.3 DOCX-Dokumente (Microsoft Word) erstellen

DOCX-Vorlagen können mit Microsoft Word 2007 oder neuer erstellt werden.

Die Erstellung erfolgt analog zu der Erstellung von RTF-Vorlagen mit dem Unterschied, dass die Path-/Skript-Anweisungen in Text-Inhaltssteuerelementen abgelegt werden.

Zum Einfügen der Steuerelement müssen in Word zuerst die Entwicklertools aktiviert werden. Dazu im Office-Menü die **Word-Optionen** öffnen und in der Kategorie **Häufig verwendet** die Option **Entwicklerregisterkarte in der Multifunktionsleiste anzeigen** aktivieren.
Nun aktiviert man auf der Registerkarte **Entwicklertools** den **Entwurfsmodus**.

1.10.2 Druckvorlagen für Listen erstellen


Wenn man das Attribut "Dokument (Druckvorlage)" nicht angelegt hat, so wird bei der Dokumentgenerierung eine Excel-Datei erzeugt, die ein Arbeitsblatt mit den Daten der Objektliste und den Spaltenüberschriften aus der Objektlistenkonfiguration enthält, d.h. man muss nicht
zwangsläufig eine Excel-Datei als Druckvorlage angeben.

Das folgende Beispiel zeigt eine Druckvorlage für Listen mit Objekten des Typs "Task".


**Erstellen der Excel-Datei**

Als Vorlage dient eine gewöhnliche Excel-Datei, die ein zusätzliches Arbeitsblatt namens "data" enthalten muss. Die Objektlistendaten werden später dann in dieses Arbeitsblatt gefüllt und zwar ohne Überschriften und beginnend mit der Zelle A1.
Die anderen Arbeitsblätter können Daten aus dem Blatt "data" in Formeln referenzieren. i-views sorgt dafür, dass alle Formeln neu berechnet werden, sobald die ausgefüllte Excel-Datei das nächste Mal mit Excel geöffnet wird.

1.10.3 Dokumentformatkonvertierung mit Open/LibreOffice

Das Ausgabeformat des Druckvorgangs entspricht dem des verwendeten Templates. Möchte man ein anderes Ausgabeformat erhalten, muss man einen Konverter einrichten.

Dazu benötigt man eine Installation von Libre- oder OpenOffice ab Version 4.0 auf dem Rechner, der die Konvertierung durchführen soll - gewöhnlich dort, wo die Bridge oder der Jobclient läuft, der auch den Druckvorgang durchführt.

Zusätzlich muss in der Konfigurationsdatei (bridge.ini, jobclient.ini, etc.) der Pfad zum "soffice"-Programm angegeben werden, welches Teil der Libre/OpenOffice-Installation ist und sich dort im Unterverzeichnis "program" befindet. Diese Angabe muss als absoluter Pfad erfolgen, relative Pfade (..\LibreOffice\etc.) sind hier nicht möglich.

```
[file-format-conversion]
sofficePath="C:\Program Files (x86)\LibreOffice 4.0\program\soffice.exe"
```

Konvertierungsservice

Möchte oder kann man nicht an allen Arbeitsplätzen bzw. Serverinstallationen, von denen aus die Formatkonvertierung durchgeführt werden soll, eine Libre/Office-Installation vorhalten, kann eine entsprechend konfigurierte REST-Bridge die Konvertierung durchführen.

Das .ini-File der REST-Bridge muss dazu folgendes Format aufweisen:

```
[Default]
host=localhost

[KHTTPRestBridge]
port=3040
volume=kartenverwaltung
```
services=jodService

[file-format-conversion]
sofficePath="C:\Program Files (x86)\LibreOffice 4.0\program\soffice.exe"

Im Admin-Tool trägt man unter Systemkonfiguration/Komponenten/Konvertierungsservice ein, über welche Adresse der Konvertierungsservice erreichbar ist.
Beispiel:

http://localhost:3040/jodService/jodconverter/service

Dokument-Formate
Damit die Ausgabeformate verfügbar sind, müssen entsprechend konfigurierte Objekte vom Typ "Konverter-Dokumentformat" im Netz vorhanden sein.
Wichtig ist, dass nicht alle Formate ineinander Konvertiert werden können. Die wichtigsten sind:

<table>
<thead>
<tr>
<th>Name</th>
<th>Extension</th>
<th>Mime-Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portable Document Format</td>
<td>pdf</td>
<td>application/pdf</td>
</tr>
<tr>
<td>OpenDocument Text</td>
<td>odt</td>
<td>application/vnd.oasis.opendocument.text</td>
</tr>
<tr>
<td>Microsoft Word</td>
<td>doc</td>
<td>application/msword</td>
</tr>
</tbody>
</table>

1.11 Entwicklungsunterstützung

1.11.1 Dev-Tools
Es stehen verschiedene Tools zur Verfügung um die Entwicklung zu erleichtern.


1.11.2 Dev-Service
Hierfür muss im Knowledge-Builder der Dev-Service gestartet werden. Dazu ruft man zunächst die Einstellungen auf und geht dann im Reiter Persönlich auf Dev-Tools. Hier lässt sich nun ein Port angeben unter denen der Dienst erreichbar sein soll. Über die nebentstehenden Schaltflächen kann der Dienst manuell gestartet und angehalten werden. Ist die Checkbox "Au-
tomatisch starten" gesetzt, wird der Service automatisch mit dem Knowledge-Builder gestartet.

Hat der Knowledge-Builder eine INI-Datei (der Standardname ist "kb.ini") kann er die Einstellungen dauerhaft speichern. In der INI-Datei können die Einstellungen aber auch von Hand eingetragen werden:

```
[DevService]
autostart=true
port=3050
```

## 2 Admin-Tool

Mit dem Admin-Tool können neue Wissensnetze angelegt, alle Wissensnetze eines Mediators verwaltet und einzelne Wissensnetze konfiguriert werden.

### 2.1 Startfenster


![Startfenster](image)

#### 2.1.1 Server


Läuft der Mediator, über den Wissensnetze administriert werden sollen, auf dem gleichen Rechner wie das Admin-Tool, kann er auch über den Rechnernamen localhost angesprochen werden.

Bleibt das Feld leer, wird auf die Wissensnetze zugegriffen, die relativ zur Position des Admin-Tools im direkten Unterordner volumes liegen. Für diese Art des Zugriffes ist kein Mediator notwendig.

Einmal eingegebene Einträge im Freitextfeld werden gespeichert. Über die Schaltfläche ...
können sie in einem separaten Fenster aus einer Liste ausgewählt werden.
Mit der Schaltfläche Verwalten gelangt man zur Serververwaltung, bei der eine Authentifizierung mit dem Serverpassword benötigt wird.

2.1.2 Wissensnetz

Im Freitextfeld Wissensnetz wird das Wissensnetz angegeben, das administriert werden soll. Einmal eingegebene Einträge im Freitextfeld werden gespeichert. Über die Schaltfläche ... können sie in einem separaten Fenster aus einer Liste ausgewählt werden. Zur Anzeige aller Wissensnetze wird man gegebenenfalls aufgefordert, das Serverpassword einzugeben.

2.1.3 Info

Über die Schaltfläche Info lassen sich in einem eigenen Fenster versionsspezifische Informationen über das Admin-Tool abrufen.

Konkret handelt es sich dabei um

- die Versionsnummer des Admin-Tools (Build),
- den Veröffentlichungsstatus des Admin-Tools (Release State),
- die vom Admin-Tool maximal nutzbare Menge an Systemarbeitsspeicher in Byte (Speicherbegrenzung),
- die Versionsnummer und der digitale Fingerabdruck der vom Admin-Tool verwendeten Ausführungsumgebung (VM Version),
- die im Betriebssystem aktive Spracheneinstellung (Locale),
- die im Admin-Tool verwendeten, mitgelieferten Schriftarten (Fonts),
• die mit dem Admin-Tool ausgelieferten Wissensnetzkomponenten inklusive Versionsnummer (Softwarekomponenten) und
• die im Admin-Tool verwendeten Smalltalk-Pakete inklusive Versionsnummer (Pakete).

Die Angaben zur \textit{Netzversion} und zur \textit{Volume-Information} sind hierbei nicht einschlägig.

Die Informationen werden in einem unsichtbaren Textfeld ausgegeben, welches über ein Kontextmenü verfügt, das per Rechtsklick aktiviert werden kann:

• \textbf{Select All} markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.
• \textbf{Copy} kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.
• \textbf{Find Again} sucht nach dem gewählten Textbereich und findet sein nächstes Auftreten gemäß der Leserichtung.
• \textbf{Find} erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.

Über die Schaltfläche \textit{Kopieren} werden alle Informationen in die Zwischenablage des Betriebssystems kopiert.


Die Schaltfläche \textit{Ok} ermöglicht eine Rückkehr zum Startfenster.

\subsection*{2.1.4 Verwalten, Neu und Weiter}

\textbf{Neu} leitet weiter zur Wissensnetzerzeugung.

\textbf{Verwalten} leitet weiter zur Serververwaltung.

\textbf{Weiter} leitet weiter zur Einzelnetzverwaltung. Hierfür werden die Einträge \textit{Benutzername} und \textit{Passwort} zur Anmeldung mit einem Administratorkonto verwendet.

\subsection*{2.1.5 Ende}

Die Schaltfläche \textit{Ende} schließt das Admin-Tool.

\subsection*{2.2 Wissensnetzerzeugung}

2.2.1 Server

Im Freiertextfeld Server wird der Name oder die IP-Adresse des Rechners angegeben, auf dem der Mediator läuft, über den das neue Wissensnetz angelegt werden soll. Sollte dieser nicht über den Standard-Port erreichbar sein, muss außerdem die korrekte Port-Nummer genannt werden. Die Eingabeform hierzu lautet [Rechnername oder IP-Adresse]:[Portnummer].

Läuft der Mediator, über den das neue Wissensnetz angelegt werden soll, auf dem gleichen Rechner wie das Admin-Tool, kann er auch über den Rechnernamen localhost angesprochen werden.

Bleibt das Feld leer, wird das Wissensnetz im relativ zur Position des Admin-Tools direkten Unterordner volumes erzeugt.

2.2.2 Neues Wissensnetz

Im Freiertextfeld Neues Wissensnetz wird der Name des Wissensnetzes festgelegt. Die dafür erlaubten Zeichen werden über das Dateisystem des Betriebssystems vorgegeben, auf dem das Wissensnetz gespeichert werden soll. Damit die Daten auch auf unterschiedlichen Dateisystemen gespeichert werden können, gilt:

- maximal 64 Zeichen
- keine Leerzeichen am Anfang oder Ende
- erlaubte Zeichen: große und kleine lateinische Buchstaben, Ziffern, Leerzeichen, !@#$%^&'()-_.\_/~Œœ sowie ASCII-Zeichen 160-255
- nicht erlaubte Zeichenfolgen sind: AUX, CON, NUL, PRN sowie COM0-COM9 und LPT0-LPT9

Die Vergabe eines Namens ist zwingend.
Der Name lässt sich später nur bei Kopiervorgängen des Wissensnetzes oder über Umbenennungen der Datei- und Verzeichnisnamen ändern. Bei einer Änderung ist zu beachten, dass der Name des Wissensnetzes eventuell in Initialisierungsdateien verwendet wird und möglicherweise die Lizenz darauf angepasst wurde.

2.2.3 Passwort (Mediator)


2.2.4 Lizenz

Ein Wissensnetz muss eine gültige Lizenz besitzen, damit der Knowledge-Builder und andere Software-Komponenten (mit Ausnahme des Admin-Tools) damit arbeiten können. Über die Schaltfläche ... kann auf das Dateisystem des Betriebssystems zugegriffen werden, um einen Lizenzschlüssel (Dateiname: `[Lizenzname].key`) zu laden.

2.2.5 Benutzername


2.2.6 Passwort (Benutzer)

Im Freitextfeld **Passwort** kann ein Passwort für den ersten im Wissensnetz registrierten Nutzer vergeben werden. Dieses Passwort wird später gebraucht, wenn dieser Nutzer sich im Knowledge-Builder oder im Admin-Tool für das Wissensnetz anmelden will.

2.2.7 OK und Abbrechen

Die Schaltfläche **Ok** erzeugt das Wissensnetz unter Einbeziehung der eingegebenen Daten. Die Schaltfläche **Abbrechen** bricht den Vorgang ab. In beiden Fällen erfolgt eine Rückkehr zum **Startfenster**.

2.3 Serververwaltung

Die Gesamtnetzverwaltung erlaubt die Administration aller Wissensnetze eines Mediators beziehungsweise des lokalen Unterordners volumes. Sie kann über die Schaltfläche **Verwalten** im **Startfenster** erreicht werden. Erforderlich ist hierzu eine entsprechende Eingabe im Feld **Server** des **Startfensters**. Etwaige Eingaben im Feld **Wissensnetz** des **Startfensters** werden ignoriert. Werden die zu administrierenden Wissensnetze über einen Mediator angesprochen, muss außerdem das korrekte Mediator-Passwort in einem eigenen Fenster angegeben werden.
Das Gesamtnetzverwaltungsfenster besteht aus einer tabellarischen Netzübersicht, einem Nachrichtenfeld und einer Menüzeile.

2.3.1 Netzübersicht

Die tabellarische Netzübersicht gibt Aufschluss über

- den Namen (Volume),
- die Anzahl an gegenwärtig aktiven Nutzern (Clients),
- das Datum und die Uhrzeit der letzten Sicherung (letztes Backup) sowie
- die letzte Statusmeldung (Status) des jeweiligen Netzes.

Die einzelnen Spalten sind über einen Klick auf den Spaltenkopf sortierbar.


2.3.2 Nachrichtenfeld

2.3.3 Menüzeile

Die Menüzeile besteht aus den folgenden Menüreitern:

2.3.3.1 Datei


Abmelden schließt die Serververwaltung und öffnet wieder das Anmeldefenster.

Beenden schließt die Serververwaltung

2.3.3.2 Server

Aktualisieren ermittelt die in der Netzübersicht im Gesamtverwaltungsfenster erhobenen Daten neu.


2.3.3.3 Verwalten

Volume herunterladen erzeugt eine Kopie des in der Netzübersicht ausgewählten Wissensnetzes und speichert sie lokal im relativen zur Position des Admin-Tools liegenden Unterordner volumes. In einem separaten erscheinenden Freitextfeld kann ein neuer Name für diese Kopie vergeben werden.

Volume kopieren erzeugt eine Kopie des in der Netzübersicht ausgewählten Wissensnetzes und speichert sie im gleichen Ordner wie das Originalnetz. In einem separat erscheinenden Freitextfeld muss ein neuer Name für diese Kopie vergeben werden.

Volume hochladen erzeugt eine Kopie eines ausgewählten lokalen Wissensnetzes und speichert sie im relativ zum verbundenen Mediator liegenden Unterordner volumes. In einem separat erscheinenden Freitextfeld kann ein neuer Name für diese Kopie vergeben werden. Die Auswahl des lokalen Wissensnetzes, das im relativ zur Position des Admin-Tools liegenden Unterordner volumes abgelegt sein muss, erfolgt über ein separates Auswahlfenster.

Volume austauschen erzeugt eine Kopie eines ausgewählten lokalen Wissensnetzes und überschreibt damit das in der Netzübersicht ausgewählte Wissensnetz. Die Kopie erhält
dabei den Namen des überschriebenen Wissensnetzes. Die Auswahl des lokalen Wissensnetzes, das im relativ zur Position des Admin-Tools liegenden Unterordner *volumes* abgelegt sein muss, erfolgt über ein separates Auswahlfenster.

In Folge der über Transfer-Operationen ausgelösten Kopierprozesse wird die Blockbelegung der Cluster und Blobs innerhalb der Wissensnetzkopien neu festgelegt und dabei deren Platzverbrauch optimiert. Der hierdurch bewirkte Komprimierungseffekt ist der gleiche wie über die Operation *Verwalten -> Volume komprimieren*.

Mit Ausnahme der Operation *Volume kopieren* setzen alle diese Operationen voraus, dass das Admin-Tool mit einem Mediator verbunden ist.

### 2.3.3.4 Verwalten

**Admintool öffnen** meldet sich im ausgewählten Volume mit dem Admintool an. Hiefür ist keine Authentifizierung im Volume nötig - die Mediator-Authentifizierung genügt.

Auf diese Art kann bei einem vergessenen Administrator-Passwort auf die Benutzerverwaltung des Volumes zugegriffen werden.


Vor Erzeugung der Sicherungskopie wird über ein eigenes Fenster erfragt, ob der Nutzer bis zum Abschluss des Kopiervorganges warten will. Gegebenenfalls wird bis zu diesem Zeitpunkt die weitere Nutzung des Admin-Tools blockiert. Andernfalls startet der Kopiervorgang im Hintergrund und eine Nachricht über den Fortgang des Kopiervorganges oder dessen Fertigstellung unterbleibt.

**Backup wiederherstellen** erzeugt eine Kopie einer ausgewählten Sicherungskopie und speichert sie im gleichen Ordner wie die in der Netzübersicht dargestellten Wissensnetze. In einem separat erscheinenden Freitextfeld muss ein neuer Name für diese Kopie vergeben werden. Die Auswahl der Sicherungskopie, die in einem Unterordner des Ordners *backup* abgelegt sein muss, der parallel zur Position der in der Netzübersicht dargestellten Wissensnetze liegt, erfolgt über zwei separate Auswahlfenster, in denen zuerst das Wissensnetz und danach die nach Erstellungszeiten sortierte Version ausgewählt werden.

**Backup löschen** löscht eine ausgewählte Sicherungskopie. Die Auswahl dieser Sicherungskopie, die in einem Unterordner des Ordners *backup* abgelegt sein muss, der parallel zur Position der in der Netzübersicht dargestellten Wissensnetze liegt, erfolgt über zwei separate Auswahlfenster, in denen zuerst das Wissensnetz und danach die nach Erstellungszeiten sortierte Version ausgewählt werden.


**Volume löschen** löscht das in der Netzübersicht ausgewählte Wissensnetz.


**Volume-Storage aktualisieren** aktualisiert die Version des Blockdateisystems des in der

2.3.3.5 Garbage Collection


Start startet eine neue Speicherbereinigung für das in der Netzübersicht ausgewählte Wissensnetz oder setzt eine pausierte Speicherbereinigung fort. Es erfolgt keine Rückmeldung, wann der Vorgang abgeschlossen ist. Der Stand des Fortschritts kann über den Menüpunkt Status in Erfahrung gebracht werden.

Pause unterbricht die Durchführung der aktiven Speicherbereinigung für das in der Netzübersicht ausgewählte Wissensnetz.

Anhalten bricht die Durchführung der aktiven Speicherbereinigung für das in der Netzübersicht ausgewählte Wissensnetz ab.

Status schreibt den aktuellen Zustand der Speicherbereinigung für das in der Netzübersicht ausgewählte Wissensnetz in die Statusspalte der Netzübersicht und in das Nachrichtenfeld. Ist eine Speicherbereinigung aktiv, erfolgt zusätzlich eine Rückmeldung über den Stand des Fortschritts in Form einer Prozentangabe.

2.4 Einzelnetzverwaltung


2.4.1 Nutzerauthentifizierung

Für den Zutritt zum Einzelnetzverwaltungsfenster ist die Anmeldung eines Nutzers mit Administratorenrechten notwendig.

Falls man keinen Zugang mehr zum Wissensnetz mehr hat, kann man mit Hilfe der Anmeldung in der Serververwaltung über die Authentifizierung am Server Zugang zum Wissensnetz bekommen.
2.4.2 Einzelnetzverwaltungsfenster

Das **Einzelnetzverwaltungsfenster** verfügt über eine mehrstufig gegliederte Menüliste links und ein Operationsfenster rechts. Der Inhalt des Operationsfensters richtet sich nach dem gewählten Menüpunkt in der Menüliste.

Die Schaltfläche **Zurück** wechselt zum Startfenster zurück.

Die Schaltfläche **Beenden** schließt das Admin-Tool.

Wird das zu administrierende Wissensnetz ohne Mediator angesprochen, ist ein Zutritt anderer Nutzer zum Wissensnetz über den Knowledge-Builder oder eine weitere Instanz des Admin-Tools nicht möglich, solange das **Einzelnetzverwaltungsfenster** geöffnet ist.
2.4.2.1 Datenbestand verwalten


Vor Erzeugung der Sicherungskopie wird über ein eigenes Fenster erfragt, ob der Nutzer bis zum Abschluss des Kopiervorganges warten will. Gegebenenfalls wird bis zu diesem Zeitpunkt die weitere Nutzung des Admin-Tools blockiert. Andernfalls startet der Kopiervorgang im Hintergrund und eine Nachricht über den Fortgang des Kopiervorganges oder dessen Fertigstellung unterbleibt.

**Backup wiederherstellen** ersetzt das aktuelle Wissensnetz durch eine Sicherungskopie (danach wird man automatisch abgemeldet). Die Auswahl der Sicherungskopie erfolgt über die Zeitpunkte der jeweiligen Backups.

**Backup löschen** löscht eine einzelne Sicherungskopie dieses Wissensnetzes.


**Download** erzeugt eine Kopie des Wissensnetzes und speichert sie lokal im relativ zur Posi-
tion des Admin-Tools liegenden Unterordner volumes. In einem separat erscheinenden Freitextfeld kann ein neuer Name für diese Kopie vergeben werden.

**Volume hochladen** überträgt ein lokal gespeichertes Netz und ersetzt das aktuelle Wissensnetz durch dieses (danach wird man automatisch abgemeldet)

### 2.4.2.2 Information

#### 2.4.2.2.1 Job-Client


Die tabellarische **Job-Clients-Übersicht** zeigt für jeden aktuell laufenden Job-Client

- seinen Namen im Format [Job-Client-Name]@[Mediator-Name] (*Name*),
- seine Job-Client-Nummer (*ID*),
• seine IP-Adresse (IP),
• den Namen des mit ihm verbundenen Mediators (Server),
• seine vom Betriebssystem vergebene Prozessnummer (Prozess),
• die ihm zugeordneten Aufgabentypen (Pool),
• seinen Arbeitsstatus (Status) und
• die Anzahl von ihm erledigter Aufgaben (Erledigt).


Die einzelnen Spalten der **Job-Clients-Übersicht** sind über einen Klick auf den Spaltenkopf sortierbar. Per Rechtstastenzug auf eine Zeile kann außerdem ein Kontextmenü geöffnet werden:

- **Informationen anzeigen** stellt alle in der ausgewählten Zeile gelisteten Daten mit Ausnahme des Aufgabentyps und der erledigten Aufgabenanzahl in einem neuen Fenster dar. Ergänzt werden
  - Datum und Uhrzeit des letzten Startzeitpunkts des Job-Clients (startUpTime),
  - die ihm zur Verfügung stehende maximal nutzbare Menge an Systemarbeitspeicher in Byte (max Memory),
  - der Name seiner Protokolldatei (logFileName) und
  - sein spezieller Name, unter dem er zum Beenden gezwungen werden kann (eine Verkettung der Zeichenkette „jobclient“ und der Job-Client-Nummer) (shutDown-String).

  Die Daten können dort in die Zwischenablage des Betriebssystems kopiert (Schaltfläche **In Zwischenablage kopieren**) oder über einen Speicherdialog als frei benennbare Textdatei an eine beliebige Stelle exportiert werden (Schaltfläche **Speichern**).

  Die über den Menüpunkt **Informationen anzeigen** ausgelöste Operation kann alternativ über einen Doppelklick auf eine Zeile in der Job-Clients-Übersicht erwirkt werden.

- **Job-Client entfernen** beendet den in der **Job-Clients-Übersicht** ausgewählten Job-Client.

- **Alle Job-Clients** entfernen beendet alle in der **Job-Clients-Übersicht** gelisteten Job-Clients.

Die tabellarische **Job-Pools-Übersicht** listet alle Aufgabentypen, die in der **Job-Clients-Übersicht** mindestens einmal einem Job-Client zugeordnet sind. Für jeden Aufgabentyp werden

- seine Bezeichnung (Name),
- seine in der Job-Clients-Initialisierungdatei verwendete technische Bezeichnung (JobPool),
- die Anzahl unerledigter Aufgaben dieses Aufgabentyps (ToDo),
- die Anzahl fehlgeschlagener Aufgaben dieses Aufgabentyps (Fehlgeschlagen) und
- die Anzahl der ihm zur Verfügung stehender Job-Clients (job-Clients) genannt.
Die einzelnen Spalten der **Job-Pools-Übersicht** sind über einen Klick auf den Spaltenkopf sortierbar. Per Rechtsklick auf eine Job-Client-Zeile kann außerdem ein Kontextmenü geöffnet werden:

- **Job-Pool leeren** löscht alle unerledigten und fehlgeschlagenen Aufgaben des in der **Job-Pools-Übersicht** ausgewählten Aufgabentyps. Diese Operation ist nur möglich, wenn kein Job-Client läuft.


- Die Verwaltung der zu unterdrückenden Fehlermeldungen erfolgt in einem separaten Fenster:
  - In der alphabetisch sortierten **Fehlermeldungsliste** werden alle zu unterdrückenden Fehlermeldungen gelistet. Eine Fehlermeldung wird unterdrückt, wenn ihr Ausgabetext mit einem Text in der **Fehlermeldungsliste** übereinstimmt.
  - + erlaubt die Eingabe einer zu unterdrückenden Fehlermeldung über ein eigenes Fenster. Die eingegebene Fehlermeldung erscheint in der **Fehlermeldungsliste**.
  - ... erlaubt die Änderung der in der **Fehlermeldungsliste** ausgewählten Fehlermeldung.
  - - löscht die in der **Fehlermeldungsliste** ausgewählte Fehlermeldung.

### 2.4.2.2.2 Leistung

**Client**
**Client-Leistungsdaten aufzeichnen** startet und beendet die Erhebung diverser Leistungskennzahlen, die an Aktivitäten der mit dem Wissensnetz verbundenen Software-Komponenten gekoppelt sind. Diese Leistungskennzahlen können zur Performanzanalyse verwendet werden.

**Interval** setzt die notwendige Zeitspanne in Sekunden, bis eine Software-Komponente erneut ein Datenpaket mit Leistungskennzahlen an das Admin-Tool sendet. Es kann nach dem Start der Aufzeichnung nicht mehr verändert werden. Die Vorerinstellung liegt bei 10 Sekunden.

In der **Leistungskennzahlenübersicht** werden die Leistungskennzahlen in geschachtelten Listenpunkten ausgegeben. Per Klick auf die links neben den Rubriken befindlichen Dreiecksymbole können Listenunterpunkte ein- und ausgeklappt werden. Alternativ lässt sich dies über ein Kontextmenü realisieren, das über einen Klick mit der rechten Maustaste auf einen Listenpunkt aufgerufen werden kann:

- **Expand** klappt alle direkten Listenunterpunkte des gewählten Listenpunkts aus.
- **Expand fully** klappt alle direkten und alle indirekten Listenunterpunkte des gewählten Listenpunkts aus.
- **Contract fully** klappt alle Listenunterpunkte des gewählten Listenpunkts wieder ein.

Mit einem Doppellklick auf einen Listenpunkt lassen sich alle darunter abgelegten Leistungskennzahlen in einem separaten Fenster auf einen Blick darstellen. Dort können sie in die Zwischenablage des Betriebssystems kopiert (Schaltfläche **In Zwischenablage kopieren**) oder über einen Speicherdialog als frei benennbare Textdatei an eine beliebige Stelle ex-
portiert werden (Schaltfläche Speichern).

Aktualisieren aktualisiert die in der Leistungskennzahlenübersicht dargestellten Leistungskennzahlen.

Zurücksetzen löscht die in der Leistungskennzahlenübersicht dargestellten Leistungskennzahlen.

In Zwischenablage kopieren kopiert die in der Leistungskennzahlenübersicht dargestellten Leistungskennzahlen in die Zwischenablage des Betriebssystems.

Server

Leistung prüfen startet einen Testvorgang, der die Performance des angeschlossenen Mediators auswertet. Dabei werden vier Anfragen an den Mediator geschickt und die an das Admin-Tool gesendeten Antworten ausgewertet. Gemessen werden:

- die Zeiten bis zur Rücksendung einer kleinen Datei (Roundtrip: Blob) und
- des Ergebnisses einer Indexsuchanfrage (Roundtrip: RPC) sowie
- die durchschnittliche Übertragungsgeschwindigkeiten beim Versand mehrerer Dateien der Größe 1 MB (Throughput: Blob (1.0 MB)) und
- die durchschnittliche Übertragungsgeschwindigkeiten beim Versand mehrerer Dateien der Größe 100 KB (Throughput: Blob (100.0 KB)).

Die Testergebnisse werden in die bereitgestellte Ergebnisliste geschrieben. Die einzelnen Spalten der Tabelle sind über einen Klick auf den Spaltenkopf sortierbar.
In Zwischenablage kopieren kopiert die Testergebnisse der Ergebnisliste als Reintext in die Zwischenablage des Betriebssystems.

2.4.2.2.3 Versionsinformation

Über diesen Menüpunkt lassen sich diverse versionsspezifische Informationen über das Wissensnetz und das Admin-Tool abrufen.

![Screenshot von Versioneninformationen](image)

Konkret handelt es sich dabei um

- die Versionsnummer des Admin-Tools (Build),
- den Veröffentlichungsstatus des Admin-Tools (Release),
- die Versionsnummer des Wissensnetzes (Netzversion), die Namen des Wissensnetzes und des verwendeten Mediators (Volume-Information),
- die vom Admin-Tool maximal nutzbare Menge an Systemarbeitsspeicher in Byte (Speicherbegrenzung),
- die Versionsnummer und der digitale Fingerabdruck der vom Admin-Tool verwendeten Ausführungsumgebung (VM Version),
- die im Betriebssystem aktive Spracheinstellung (Locale),
- die im Admin-Tool verwendeten, mitgelieferten Schriftarten (Fonts),
- die im Wissensnetz installierten Wissensnetzkomponenten inklusive Versionsnummer (Softwarekomponenten) und
• die im Admin-Tool verwendeten Smalltalk-Pakete inklusive Versionsnummer (Pakete).

Die Informationen werden in einem unsichtbaren Textfeld ausgegeben, welches über ein Kontextmenü verfügt, das per Rechtsklick aktiviert werden kann:

• **Select All** markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.

• **Copy** kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.

• **Find Again** sucht nach dem gewählten Textbereich und findet sein nächstes Auftreten gemäß der Leserichtung.

• **Find** erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.

Über die Schaltfläche **Kopieren** werden alle Informationen in die Zwischenablage des Betriebssystems kopiert.


### 2.4.2.3 Systemkonfiguration

#### 2.4.2.3.1 Benutzer

Die Benutzerverwaltung gleicht derjenigen im Knowledge-Builder mit der Ausnahme, dass keine Verknüpfungen zwischen Nutzern und Objekten des nutzergenerierten Teilnetzes bearbeitet werden können.
Die tabellarische **Nutzerübersicht** zeigt für jeden im Wissensnetz registrierten Nutzer

- seinen Benutzernamen (*Benutzer*),
- mit welchem Objekt des nutzergenerierten Teilnetzes er verknüpft ist (*Verknüpft mit*),
- welchen Status er momentan besitzt (*Status*),
- an welchem Datum und zu welcher Uhrzeit er sich über den Knowledge-Builder im Wissensnetz angemeldet hat (*Anmeldedatum*), sofern er noch angemeldet ist, und
- mit welchem Verfahren das Passwort verschlüsselt ist (*Passworttyp*).

Die einzelnen Spalten der Tabelle sind über einen Klick auf den Spaltenkopf sortierbar.


**Erstellen** legt einen neuen Nutzer an. Benutzernamen (verpflichtend) und Passwort (optional) werden in einem eigenen Fenster festgelegt. Die Art und Menge der dafür erlaubten Zeichen ist nicht beschränkt.

**Passwort ändern** ändert das Passwort des in der **Nutzerübersicht** ausgewählten Nutzers. In zwei aufeinanderfolgenden Fenstern wird zweimal das neue Passwort eingegeben.

**Abmelden** meldet den in der **Nutzerübersicht** ausgewählten Nutzer nach einer Sicherheitsbestätigung aus dem Wissensnetz ab. Damit diese Operation eine Wirkung entfaltet, muss
dieser Nutzer aktuell über den Knowledge-Builder im Wissensnetz angemeldet sein.

**Löschen** löscht den in der **Nutzerübersicht** ausgewählten Nutzer nach einer Sicherheitsbestätigung. Mindestens ein Nutzer mit Administratorrechten muss verbleiben.

**Umbenennen** erlaubt über ein Freitextfeld in einem separaten Fenster die Vergabe eines neuen Benutzernamens für den in der **Nutzerübersicht** ausgewählten Nutzer. Bleibt das Freitextfeld leer, erfolgt keine Umbenennung.

**Mitteilung** sendet über ein Freitextfeld in einem separaten Fenster eine Nachricht an den in der **Nutzerübersicht** ausgewählten Nutzer. Die Nachricht wird im Wissensnetz zwischengespeichert und erscheint dem adressierten Nutzer in einem separaten Fenster im Knowledge-Builder, sobald er sich damit am Wissensnetz anmeldet. Der Nutzer kann auf diese Nachricht nicht antworten.


**Operationen** öffnet ein neues Fenster, in dem für den in der **Nutzerübersicht** ausgewählten Nutzer aus einer Liste von Operationen, namentlich

- Backup erstellen,
- Backup löschen,
- Backup wiederherstellen,
- Garage Collection,
- Kopieren,
- Log herunterladen,
- Volume herunterladen,
- Volume hochladen,
- Volume löschen,

diejenigen gewählt werden können, die dieser Nutzer im Rahmen der Einzelnetzverwaltung in Zukunft ohne Eingabe des Mediator-Passworts ausführen darf. Zur Bestätigung der Auswahl muss in das Freitextfeld **Server-Passwort für Operationen** das korrekte Mediator-Passwort eingegeben werden.

Die Operation **Operationen** ist nur wählbar für einen Nutzer mit Administratorrechten. Ihre Verwendung setzt außerdem voraus, dass ein Mediator-Passwort gesetzt wurde.

Das Feld **Administratoren** gibt die Anzahl aller im Wissensnetz registrierten Nutzer mit Administratorenrechten an.

Das Feld **Benutzer** gibt die Anzahl aller im Wissensnetz registrierten Nutzer ohne Administratorenrechte an.

Das Feld **Aktive** gibt die Anzahl aller gegenwärtig über den Knowledge-Builder im Wissensnetz angemeldeten Nutzer an.

### 2.4.2.3.2 Blob-Speicherung

Attributewerte von Attributdatentyp **Datei** (sogenannte Blobs) können auch wissensnetzextern in einem Blob-Speicher gespeichert werden. Dies hat den Vorteil, dass sie unabhängig vom Wissensnetz und damit bei Bedarf in einer anderen Systemumge-
bung verwaltet werden können. Um Blobs in einem Blob-Speicher zu sichern, muss der Blob-Speicher eingerichtet und mit einem konfigurierten Blob-Service (einem Software-Dienst) verbunden werden.


**Löschen** löscht den in der Blob-Speicher-Gesamtübersicht ausgewählten Blob-Speicher. Das Zahlenfeld **Löschbare Dateien** zeigt die Anzahl der nicht mehr gebrauchten Blobs in dem in der Blob-Speicher-Gesamtübersicht ausgewählten Blob-Speicher. Blobs werden nicht mehr gebraucht, wenn ihre jeweiligen Attribute im Wissensnetz gelöscht wurden oder wenn die Verbindung zwischen Blob-Service und Blob-Speicher per Admin-Tool aufgehoben wurde.

**Löschen** löscht alle nicht mehr gebrauchten Blobs in dem in der Blob-Speicher-Gesamtübersicht ausgewählten Blob-Speicher.

werden. Das rechts neben dem Freitextfeld URLs positionierte Kontrollkästchen Intern indiziert, ob der integrierte Blob-Service oder ein externer Blob-Service angesprochen wird. Die Konfiguration des im Mediator integrierten Blob-Services erfolgt nicht über die Mediator-Initialisierungsdatei, sondern über eine separate Initialisierungsdatei (Standarddateiname: blobservice.ini).


**Aktualisieren** aktualisiert die Übersicht angemeldeter Blob-Speicher. Dazu muss ein Blob-Speicher in der Blob-Speicher-Gesamtübersicht ausgewählt sein.


### 2.4.2.3.3 Komponenten

Wissensnetze bestehen aus Wissensnetzkomponenten. Neben den Basisfunktionalitäten gewähren sie den Wissensnetzen im Wesentlichen zusätzliche Schnittstellen und im Browser darstellbare Bedienoberflächen für die Nutzdaten (Webfrontends).


Wird eine Wissensnetzkomponente mit der rechten Maustaste angeklickt, erscheint ein Kontextmenü. Der dort verfügbare Menüpunkt **Standardkomponente hinzufügen** verfügt über die gleiche Funktionalität wie die gleichnamige Schaltfläche.


In der **Wissensnetzliste** sind alle im Wissensnetz installierten Wissensnetzkomponenten mit

Die Textfelder Name und Version zeigen den Namen respektive die dreistellige Versionsnummer der in der Wissensnetzliste ausgewählten installierten Wissensnetzkomponente.


**Boost Libraries 1.18.0**

Dieses Konfigurationsmenü erscheint nur, wenn die Wissensnetzkomponente Boost Libraries installiert ist.


Die Wissensnetzkomponente Boost Libraries ermöglicht die Sicherstellung, dass auf die Boost.Regex-Bibliothek zugegriffen werden kann.
Ist die Option **Boost-Libraries für alle inkl. Admins erforderlich** ausgewählt, können alle Software-Komponenten außer dem Admin-Tool auf das Wissensnetz nur zugreifen, wenn sie auf die Bibliothek Boost.Regex zugreifen können.


**Knowledge Portal**

Dieses Konfigurationsmenü erscheint nur, wenn die Wissensnetzkomponente Knowledge-Portal installiert ist.

Im Frontend wird eine Bearbeitungssicht und eine Präsentationssicht mit jeweils exklusiven Darstellungs- und Bedienelementen unterschieden. Für beide Sichten werden separate DTD-Schemata geführt. Die nachfolgend erläuterten Bedienelemente existieren jeweils für jede Sicht.

Über die Schaltfläche **Wählen** kann auf das Dateisystem des Betriebssystems zugegriffen werden, um eine DTD-Schema-Datei für die jeweilige Sicht zu laden und im Wissensnetz zu installieren. Der Standarddateiname für Bearbeitungssicht-DTDs lautet `editConfig.dtd`, der Standarddateiname für Präsentationssicht-DTDs lautet `viewConfig.dtd`.

**Zurücksetzen** löscht das für die jeweilige Sicht installierte DTD-Schema aus dem Wissensnetz.

**Anzeigen** stellt das für die jeweilige Sicht installierte DTD-Schema in einem eigenen Fenster dar. Dort kann es in die Zwischenablage des Betriebssystems kopiert (Schaltfläche **In Zwischenablage kopieren**) oder über einen Speicherdialog als frei benennbare Textdatei an eine beliebige Stelle exportiert werden (Schaltfläche **Speichern**). Das Fenster verfügt außerdem über ein eigenes Kontextmenü, welches mit einem rechten Mausklick geöffnet werden kann:

- **Suche** erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.
- **Alles markieren** markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.
- **Kopieren** kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.

**Konvertierungsservice**

Dieses Konfigurationsmenü erscheint nur, wenn die Wissensnetzkomponente Druckkomponente installiert ist.


Die Verknüpfung der Bridge mit der Druckkomponente erfolgt über das Freitextfeld **URL**. Dort wird die Netzadresse der Bridge im Format `http://[Bridge-IP-Nummer]:[Bridge-Port]/jodService/jodconverter/service` eingetragen. Der Pfadabschnitt `/jodService/jodconverter/service` ist historisch bedingt und aktiviert den vordefinierten jodService.

**Überprüfen** startet einen Testprozess. Der Testprozess schickt über REST ein Testdokument an die über die Netzadresse festgelegte Bridge und erwartet, dass ein ordnungsgemäß konvertiertes Testdokument zurückgeschickt wird. Das Testergebnis wird in einem separaten Fenster ausgegeben.

Im Freitextfeld **Timeout** wird festgelegt, wie viele Sekunden lang auf die Rücksendung des konvertierten Testdokuments gewartet wird, bevor eine Fehlermeldung generiert wird. Die Voreinstellung liegt bei 20 Sekunden.

### 2.4.2.3.4 Lizenz

Ein Wissensnetz muss eine gültige Lizenz besitzen, damit der Knowledge-Builder und andere Software-Komponenten (mit Ausnahme des Admin-Tools) damit arbeiten können.
Das Feld **Status** gibt an, ob die Lizenz gegenwärtig gültig oder ungültig ist. Falls sie ungültig ist, wird außerdem ein Grund genannt. Gründe für eine ungültige Lizenz können die Überschreitung des Gültigkeitsdatums oder der maximalen Anzahl erlaubter registrierter Nutzer sein.

Das Feld **Kunde** beschreibt den Kunden, für den die Lizenz ausgestellt wurde. Neben dem Namen können auch die Adresse und die Abteilung genannt sein.

Das Feld **Komponenten** stellt den Inhalt der für die Lizenzschlüsselgenerierung verwendeten Komponentenlizenzkonfigurationsdatei `[Wissensnetz].componentLicenseTemplate.ini` dar. Dort werden Festlegungen über

- die lizensierten Versionen einzelner Komponenten (version),
- die maximale Anzahl registrierter Nutzer mit Administratorrechten (maxAdminUsers) und
- die maximale Anzahl registrierter Nutzer ohne Administratorrechte (maxUsers) getroffen.

Das Feld **Partner** enthält den Namen des Partners, über den die Lizenz weitergegeben wird.

Das Feld **gültig bis** enthält das Datum, nach dessen Ablauf die Lizenz erlischt.

Das Feld **gültig für Netze** enthält eine Liste der Namen aller Netze, auf die die Lizenz beschränkt ist. Möglich ist eine Erfassung über einen regulären Ausdruck.
Das Feld gültig für Server enthält eine Liste aller IP-Adressen und Port-Nummern, über die ein an das Wissensnetz angeschlossener Mediator erreicht werden darf.

Die Felder Partner, gültig bis, gültig für Netze und gültig für Server können leer sein.

Alle Felder verfügen über ein Kontextmenü, das per Rechtstaste aktiviert werden kann:

- **Select All** markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.
- **Copy** kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.
- **Find Again** sucht nach dem gewählten Textbereich und findet sein nächstes Auftreten gemäß der Leserichtung.
- **Find** erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.

**Hinzufügen / Erneuern** erlaubt das Laden eines neuen Lizenzschlüssels (Dateiname: [Lizenzname].key) über das Dateisystem des Betriebssystems.

### 2.4.2.4 Wartung

#### 2.4.2.4.1 Client-Caches


2.4.2.4.2 Garbage Collection

Start startet eine neue Speicherbereinigung für das Wissensnetz oder setzt eine pausierte Speicherbereinigung fort. Es erfolgt keine Rückmeldung, wann der Vorgang abgeschlossen ist. Der Stand des Fortschritts kann über den Menüpunkt Aktualisieren in Erfahrung gebracht werden.

Pause unterbricht die Durchführung der aktiven Speicherbereinigung für das Wissensnetz.

Anhalten bricht die Durchführung der aktiven Speicherbereinigung für das Wissensnetz ab.

Aktualisieren schreibt den aktuellen Zustand der Speicherbereinigung für das Wissensnetz in das nebenstehende Textfeld. Ist eine Speicherbereinigung aktiv, erfolgt zusätzlich eine Rückmeldung über den Stand des Fortschritts in Form einer Prozentangabe.
2.4.2.4.3  Wartung

**Wartung jetzt ausführen** überprüft

- die Lizenz (*Lizenz*),
- Indizes (*Indizes*),
- registrierte Objekte (*die Registratur*),
- Rechte (*Zugriffsrechte*),
- Trigger (*Trigger*) und
- installierte Wissensnetzkomponenten (*aktive Komponenten*)

auf Mängel. Im Zuge der Prüfung wird auch die über den Knowledge-Builder einsehbare Statistik von Eigenschaftshäufungen pro Objekt (Metriken) aktualisiert.

Gefundene Mängel werden in einer tabellarischen **Mangelübersicht** gesammelt. Für jeden Mangel wird dort

- eine kurze Beschreibung, falls einschlägig inklusive der Cluster-ID und der Frame-ID (Format *Cluster-ID/Frame-ID*) des mangelhaften Objekts (in programmierterminologisch-er Lesart) (*Mitteilung*),
- das vom Mangel betroffene übergeordnete Wissensnetzelement (*Objekt*),
- dessen Typ (*Typ*),
• die Schwere des Mangels (Priorität) und
• dessen ersten Feststellungszeitpunkt in Form eines Datums (Datum)

ausgegeben. Die einzelnen Spalten der Tabelle sind über einen Klick auf den Spaltenkopf sortierbar.

Details stellt alle in der Mangelübersicht des ausgewählten Mangels gelisteten Daten in einem neuen Fenster dar. Ergänzt werden die Uhrzeit des ersten Feststellungszeitpunkts sowie Datum und Uhrzeit des letzten Feststellungszeitpunkts. Die Daten können dort in die Zwischenablage des Betriebssystems kopiert (Schaltfläche in Zwischenablage kopieren) oder über einen Speicherdialog als frei benennbare Textdatei an eine beliebige Stelle exportiert werden (Schaltfläche Speichern). Die über die Schaltfläche Details ausgelöste Operation kann alternativ über einen Doppelklick auf einen Mangel in der Mangelübersicht er- wirkt werden.


2.4.2.4.4 Wartungsinformation

Die **Wartungshistorie** verfügt über ein Kontextmenü, das per Rechtsklick aktiviert werden kann:

- **Select All** markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.
- **Copy** kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.
- **Find Again** sucht nach dem gewählten Textbereich und findet sein nächstes Auftreten gemäß der Leserichtung.
- **Find** erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.

**In Zwischenablage kopieren** kopiert die gesamte **Wartungshistorie** in die Zwischenablage des Betriebssystems.

**Kommentar hinzufügen** erlaubt die Eingabe einer Anmerkung über ein Freitextfeld in einem separaten Fenster. Sie wird mit einem Zeitstempel versehen und in die **Wartungshistorie** aufgenommen. In die **Wartungshistorie** aufgenommene Anmerkungen lassen sich nicht löschen.

### 2.4.2.4.5 Wartungsnachricht

Die Schaltfläche **Setzen** aktiviert eine Wartungssperre, die jedem Nutzer den Zutritt zum
Wissensnetz über den Knowledge-Builder verwehrt. Dafür muss eine Wartungsnachricht formuliert werden.

Die Wartungsnachricht wird im Freitextfeld **Wartungsnachricht** formuliert. Sie wird jedem Nutzer, der das Wissensnetz bei aktiver Wartungssperre über den Knowledge-Builder betreten will, in Form einer Fehlermeldung angezeigt.

Die Schaltfläche **Zurücksetzen** hebt die zuvor gesetzte Wartungssperre auf und löscht die Wartungsnachricht.

### 2.4.2.4.6 Wartungsskript

Über **Wartungsskript auswählen** kann auf das Dateisystem des Betriebssystems zugriffen werden, um ein Wartungsskript (Dateiname: *[Wartungsskript].kss* ) zu laden. Wartungsskripte werden fallspezifisch in der Programmiersprache Smalltalk angefertigt und erlauben Operationen, die sich nicht über die vordefinierten Funktionen des Admin-Tools oder über die KEM- oder JS-Schnittstellen realisieren lassen.

Verfügt das Wartungsskript über eine Beschreibung, wird diese nach dem Laden des Wartungsskripts in einem unsichtbaren Textfeld unterhalb der Schaltfläche **Wartungsskript auswählen** ausgegeben. Dieses Textfeld verfügt über ein Kontextmenü, das per Rechtsklick aktiviert werden kann:

- **Select All** markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.
- **Copy** kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.
- **Find Again** sucht nach dem gewählten Textbereich und findet sein nächstes Auftreten gemäß der Leserichtung.
- **Find** erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.

**Wartungsskript ausführen** startet das Wartungsskript. Ein separat erscheinendes Fenster gibt Auskunft, wenn das Wartungsskript vollzogen wurde, und bietet je nach Skript zusätzliche Ausführungsinformationen oder erlaubt skriptspezifische Ausführungsoptionen.

### 2.4.2.5 XML-Import/-Export

#### 2.4.2.5.1 Schema und Konfiguration

Ein Wissensnetz im weiteren Sinne besteht neben den nutzergenerierten und über Komponenten eingebrachten Teilnetzen (Schemata mit Nutzdaten) noch aus diversen weiteren Bausteinen (Konfigurationen), die dieses Teilnetz funktional erweitern, konfigurieren oder damit arbeiten. Im Rahmen dieses Menüpunkts werden Schemata und Konfigurationen zusammenfassend als Konfigurationen bezeichnet.

Zahlreiche Konfigurationen eines Wissensnetzes lassen sich gezielt exportieren und importieren.
Die Konfigurationsübersicht bietet einen listenartigen Überblick über alle mittels der im Folgenden beschriebenen Operationen prinzipiell transferierbaren Konfigurationstypen eines Wissensnetzes. Prinzipiell transferierbar sind

- einzelne registrierte Abbildungen von Datenquellen (Abbildungen von Datenquellen),
- einzelne von Administratoren konfigurierte und benutzerdefinierte Suchfelder (Abfragen),
- einzelne Datenquellenzugriffseinstellungen zur Nutzung für Abbildungen von Datenquellen (Datenquellen),
- die Druckkonfiguration (Druckkonfiguration),
- die Menge aller innerhalb der Rubrik Ermittlung der View-Konfiguration definierten Bausteine (Ermittlung der View-Konfiguration),
- einzelne Indexfilter (Indexfilter),
- einzelne Indexerkonfigurationen (Indizes),
- die LDAP-Authentifizierung (LDAP),
- die Lizenz des Wissensnetzes (Lizenz),
- einzelne registrierte Sammlungen semantischer Objekte (Sammlung von Wissensnetzelementen),
- einzelne registrierte Skripte (Skripte),
- den Arbeitsordner (Strukturordner),
- die Menge aller innerhalb der Rubrik Trigger definierten Bausteine (Trigger),
- einzelne Teilnetze (Wissensnetz) und
die Menge aller innerhalb der Rubrik Rechte definierten Bausteine (Zugriffsrechte).


Das Ein- und Ausklappen von Listenunterpunkten in der Konfigurationsübersicht geschieht per Klick auf die sich links neben den Listenpunkten befindlichen Dreiecksymbole. Alternativ lässt sich dies über ein Kontextmenü realisieren, das über einen Klick mit der rechten Maustaste auf einen Listenpunkt aufgerufen werden kann:

- Expand klappt alle direkten Listenunterpunkte des gewählten Listenpunkts aus.
- Expand fully klappt alle direkten und alle indirekten Listenunterpunkte des gewählten Listenpunkts aus.
- Contract fully klappt alle Listenunterpunkte des gewählten Listenpunkts wieder ein.

Hinzufügen fügt der Konfigurationsübersicht eine Konfiguration des dort ausgewählten Konfigurationstyps hinzu. Existiert mehr als eine Konfiguration für den ausgewählten Konfigurationstyp im Wissensnetz, schließt sich eine Auswahlmöglichkeit in einem separaten Fenster an. Die Auswahl erfolgt dort entweder einzeln per Klick auf die jeweiligen Konfiguratio-
nen in einer Liste oder pauschal über die Schaltfläche **Alles aus-/abwählen**.

**Entfernen** löscht entweder alle Konfigurationen des in der **Konfigurationsübersicht** ausgewählten Konfigurationstyps oder die in der **Konfigurationsübersicht** ausgewählte Konfiguration.

**Alle hinzufügen** fügt der **Konfigurationsübersicht** alle im Wissensnetz existierenden Konfigurationen hinzu und verteilt diese auf die jeweils passenden Konfigurationstypen.

Über die Schaltflächen ... kann auf das Dateisystem des Betriebssystems zugegriffen werden, um ein Wartungsskript (Dateiname: `[Wartungsskript].kss` zu laden. Wartungsskripte werden fallspezifisch in der Programmiersprache Smalltalk angefertigt und erlauben Operationen, die sich nicht über die vordefinierten Funktionen des Admin-Tools oder über die KEM- oder JS-Schnittstellen realisieren lassen.


**Export** exportiert die in der **Konfigurationsübersicht** ausgewählten Konfigurationen. Zur Auswahl stehen der Export in eine einzige Archivdatei im Archivformat `tar` oder in einzelne Dateien in einem Ordner. Die Auswahl der Exportmethode vollzieht sich in einem separaten Fenster:

- In den Freitextfeldern **Datei** oder **Verzeichnis** kann der Name der Archivdatei (Dateiname: `[Wissensnetz].tar` respektive des Ordners (kein Standardname) angegeben werden. Die Archivdatei respektive der Ordner wird im gleichen Ordner wie das Admin-Tool angelegt. Alternativ kann über **Wählen** ein Speicherdialog aufgerufen werden, um Name und Speicherort der Archivdatei respektive des Ordners frei festzulegen.

**Import** importiert nach einer Bestätigungsfrage Konfigurationen in das Wissensnetz. Zur Auswahl stehen der Import aus einer einzigen Archivdatei im Archivformat `tar` oder aus einzelnen Dateien in einem Ordner. Die Auswahl der Importmethode vollzieht sich in einem separaten Fenster:

- In den Freitextfeldern **Datei** oder **Verzeichnis** kann der Name der Archivdatei (Dateiname: `[Wissensnetz].tar` respektive des Ordners (kein Standardname) angegeben werden. Die Archivdatei respektive der Ordner wird im gleichen Ordner wie das Admin-Tool gesucht. Alternativ kann über **Wählen** auf das Dateisystem des Betriebssystems zugegriffen werden, um eine Archivdatei respektive einen Ordner an einer beliebigen Stelle auszuwählen.

- Ist die zu importierende Archivdatei respektive der zu importierende Ordner gewählt, erscheint in einem weiteren Fenster eine Übersicht über alle darin enthaltenen Konfigurationen. Diese Übersicht kann dort in die Zwischenablage des Betriebssystems kopiert (Schaltfläche **In Zwischenablage kopieren**) oder über einen Speicherdialog als freiblebbare Textdatei an eine beliebige Stelle exportiert werden (Schaltfläche **Speichern**). Die Schaltfläche **Import** startet den Importprozess. Das Fenster verfügt außerdem über ein eigenes Kontextmenü, welches mit einem rechten Mausklick geöffnet werden kann:
  - **Suche** erlaubt die Eingabe einer zu suchenden Zeichenkette in einem separaten Fenster und findet ihr nächstes Auftreten gemäß der Leserichtung relativ zur Position der per Mausklick setzbaren Schreibmarke. Bei der Suche wird Groß- und Kleinschreibung unterschieden.
- **Alles markieren** markiert den gesamten Text. Alternativ kann mit dem Mauszeiger ein beliebiger Textausschnitt markiert werden.
- **Kopieren** kopiert den gewählten Textbereich in die Zwischenablage des Betriebssystems.

**Speichern** speichert die in der Konfigurationsübersicht für dieses Wissensnetz aktuell getroffene Auswahl an Konfigurationen als XML-Datei. Über einen Speicherdialog werden Name und Ort der XML-Datei (Standarddateiname: `instruction.xml`) festgelegt.

**Laden** greift auf das Dateisystem des Betriebssystems zu, um eine zuvor gespeicherte Auswahl an Konfigurationen für dieses Wissensnetz aus einer XML-Datei (Standarddateiname: `instruction.xml`) zu laden.

**Aktualisieren** fügt dem Wissensnetz die mit dem Attributdatentyp Boolesch ausgestatteten Attributtypen

- **XML: Alle Objekte exportieren**, 
- **XML: Direkte Objekte exportieren**, 
- **XML: Typ und alle Untertypen nicht exportieren und** 
- **XML: Untertypen nicht exportieren**


Soweit nicht anders über diese Attributwerte konfiguriert, gilt für jeden Objekttyp, dass er selbst exportiert wird, nicht aber seine Objekte. Wird ein Objekt oder Objekttyp exportiert, werden alle direkt mit ihm verbundenen Attribute und Relationen sowie deren Attribut- und spezifische Relationstypen ebenfalls exportiert.

## 3 ViewConfig-Mapper

### 3.1 Einführung


Der Viewconfig Mapper ist eine Single-Page-Applikation, die client-seitig im Web-Browser läuft. Sie verwendet reactive (reactive.js.org) für eine interaktive und reaktive Anwendung, die auf mustache-Templates (mustache.github.io/) basiert.

### 3.2 Konfiguration

Zunächst muss die Viewkonfig-Mapper-Komponente im Netz hinzugefügt werden. Dabei werden REST-Ressourcen, die spezielle für den Viewkonfig-Mapper zur Verfügung stehen, angelegt und einige speziell auf das Frontend abgestimmte Viewkonfigurationsoptionen angelegt.

3.2.1 View-Konfigurationen für den Viewconfig Mapper

Der Viewconfig-Mapper interpretiert alle View-Konfigurationen, die in i-views erstellt wurden. Dabei gibt es jedoch ein paar Unterschiede zwischen der Verarbeitung im Knowledge-Builder und im Viewconfig-Mapper, auf die in diesem Kapitel eingegangen wird.

3.2.1.1 Panel-Konfiguration


3.2.1.2 anwenden in


Mögliche Ziele von "anwenden in" sind Anwendung, Panel und Aktion.

Bei Panel-Konfigurationen kommt diese Technik immer dann zur Anwendung, wenn bei einem Panel "Flexible Ansicht" konfiguriert wurde.

3.2.1.3 Style/additionalConfig

Grundsätzlich wird die Darstellung in HTML über die Templates bestimmt, so dass dies nicht Teil der View-Konfiguration ist. Um aber dem Ersteller der View-Konfigurationen die
Möglichkeit zu geben, in einigen Aspekten doch Einfluss auf die Darstellung zu haben, gibt es sogenannte "Styles" in der View-Konfiguration.

Im Reiter "Styles" können beliebig viele Styles hinzugefügt werden. Nach dem Ablegen eines Styles hat man die Möglichkeit, diesen genauer zu definieren. Unter dem Reiter "StylePropertyKey" findet man alle bereits konfigurierten Styles und kann die entsprechenden Werte für das Frontend dort konfigurieren.


3.2.1.4 Style

Grundsätzlich wird die Darstellung in HTML über die Templates bestimmt, so dass dies nicht Teil der View-Konfiguration ist. Um aber dem Ersteller der View-Konfigurationen die Möglichkeit zu geben, in einigen Aspekten doch Einfluss auf die Darstellung zu haben, gibt es sogenannte "Styles".

Die Einstellungen zu den Styles für die Darstellung im Web-Frontend durch den Viewkonfiguration-Mapper sind unter dem Reiter "Viewkonfiguration-Mapper" verfügbar. Voraussetzung hierfür ist eine installierte Viewkonfiguration-Mapper-Komponente im KB.

Insgesamt sind mehrere Einstellungsmöglichkeiten zu den Styles vorhanden (siehe Abbil-
Es gibt eine Reihe von Style-Elementen, die bereits in i-views definiert sind. Um welche Elemente es sich handelt und wie diese Style-Elemente im Knowledge-Builder angelegt werden, sodass sie dann mit einzelnen Elementen der View-Konfiguration einer Anwendung verknüpft werden können, wird im Folgenden erläutert.


Wenn wir ein neues Style-Element definieren, müssen wir diesem zuerst einen Konfigurationsnamen geben. Auf der rechten Seite des Editors kann daraufhin die Konfiguration vorgenommen werden.

Im Folgenden werden die einzelnen Konfigurationsmöglichkeiten für ein Style-Element erläutert:

<table>
<thead>
<tr>
<th>Name</th>
<th>Attributtyp</th>
<th>Konfigurationstyp</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class (skript)</td>
<td>Verweis auf Skript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>collapsed</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Beschreibung</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>dateFormat</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>datetimepickerOptions</td>
<td>Verweis auf Skript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>downloadRequest</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>editCustomButtons</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>editStageToggle</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extra</td>
<td>Verweis auf Skript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extra</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extraDateFormats</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>groupColumnGrid</td>
<td>Zeichenkette</td>
<td>Gruppe</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Als Eingabe wird ein String mit Zahlen erwartet, die durch ein Leerzeichen oder ein Komma getrennt werden. Jede Zahl definiert die Anzahl der Spalten, wenn das Maximum 12 Spalten beträgt.</td>
<td></td>
</tr>
<tr>
<td>hideFilters</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hideLabel</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>href</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>localAction</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>numberFormat</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>readOnly</td>
<td>Boolean</td>
<td>Eigenschaften</td>
<td></td>
</tr>
<tr>
<td>renderMode</td>
<td>Auswahl</td>
<td>• breadcrumb</td>
<td>• calendar</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>target</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tooltip</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmDetailed</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmMarkRowClick</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmPluginCalendarOptions</td>
<td>Verweis auf Skript</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmPluginChartDataColumns</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmPluginChartDataMode</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmPluginChartHeight</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmPluginChartLabelColumn</td>
<td>Zeichenkette</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vcmPluginChartOptions</td>
<td>Verweis auf Skript</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 3.2.1.4.1 RenderModes

Mithilfe von renderModes können zusätzliche, vordefinierte Style-Eigenschaften angewendet werden.

RenderModes sind in der View-Konfiguration in den Styles unter dem Reiter "Viewkonfiguration-Mapper" verfügbar, einmal per Dropdown-Menü und zusätzlich per Eingabezeile. Die im Dropdown-Menü verfügbaren renderModes sind wie folgt:

<table>
<thead>
<tr>
<th>render-Mode</th>
<th>Erläuterung</th>
<th>Anwendbarkeit</th>
</tr>
</thead>
<tbody>
<tr>
<td>bread-crum</td>
<td>Zeigt die Hierarchie mit Pfadnavigation an</td>
<td>Hierarchie</td>
</tr>
<tr>
<td>calendar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chart</td>
<td>Darstellung der Daten einer Tabelle in einem Diagramm. Unter vcmPluginChartType kann die Art des Diagramms ausgewählt werden. Unter vcmPluginChartOptions kann mittels Skript eine genauere Formatierung des Diagramms vorgenommen werden, bspw. Achsen-Skalierung, Anzeige von Legenden etc.</td>
<td>Tabelle</td>
</tr>
<tr>
<td>download</td>
<td>Link auf Dateidownload</td>
<td>Aktion</td>
</tr>
<tr>
<td>edit</td>
<td>Untergeordnete Eigenschaften werden editierbar</td>
<td>Gruppe</td>
</tr>
<tr>
<td>external</td>
<td>Erzeugt in Verbindung mit href einen externen Link; lässt sich bspw. in Kombination mit Symbol und Tooltip verwenden</td>
<td>Aktion</td>
</tr>
<tr>
<td>render-Mode</td>
<td>Erläuterung</td>
<td>Anwendbarkeit</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>email</td>
<td>Erzeugt einen Link auf die Email-Adresse</td>
<td>Zeichenketten-Eigenschaft</td>
</tr>
<tr>
<td>image</td>
<td>Zeigt ein Icon an der Aktion an</td>
<td>Aktion</td>
</tr>
<tr>
<td>jumbotron</td>
<td>Hervorgehobene Darstellung. Siehe getbootstrap.com/docs/4.1/components/jumbotron/</td>
<td></td>
</tr>
<tr>
<td>well</td>
<td>Erzeugt eine Box mit eingedrücktem Effekt. Siehe getbootstrap.com/docs/3.3/components/#wells</td>
<td></td>
</tr>
</tbody>
</table>
3.2.1.5 ausführen in

Beim Anlegen einer benutzerdefinierten Aktion kann auch die Relation "ausführen in" gezogen werden. Dies bewirkt, dass die zurückgegebenen Daten nicht auf alle vcm-Inhalte angewendet werden, sondern sich die Änderung nur auf eine bestimmte View bezieht. Diese View muss als Relationsziel von "ausführen in" gesetzt werden.

3.2.1.6 Globaler Kontext


3.2.2 Login-Konfiguration

3.2.2.1 JWT Authentifizierung

3.2.2.1.1 Login Formular anpassen

Das Loginformular kann über folgende Übersetzungsschlüssel angepasst werden:

<table>
<thead>
<tr>
<th>Schlüssel</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>login.form.title</td>
<td>Überschrift des Formulars</td>
</tr>
<tr>
<td>login.form.message</td>
<td>Beschreibungs-/Willkommenstext</td>
</tr>
<tr>
<td>login.form.username.label</td>
<td>Label des Benutzernamenfeldes</td>
</tr>
</tbody>
</table>
3.3 Aktionen

In diesem Kapitel werden alle VCM-spezifischen Aktionseinstellungsmöglichkeiten und Skript-Parameter (context, actionResponse) beschrieben.

vcm unterstützt Standard-Interaktionen wie das Editieren von Inhalten, ohne dass dies extra konfiguriert werden muss. Es ist aber möglich, in der View-Konfiguration benutzerdefinierte Aktionen zu definieren, die dann auch in vcm ausgeführt werden können.

Es gibt zwei Sorten von benutzerdefinierten Aktionen:

- Standard-Aktion mit benutzerdefiniertem Rückgabewert
- Skript-Aktion

Die Auswahl geschieht über ein Dropdown-Menü.

Für eine Skript-Aktion muss in diesem Menü "Skript" ausgewählt werden und in der Liste unter dem Eintrag "Skript" ein Skript angelegt werden.
Um den Rückgabewert einer Aktion zu bestimmen, kann für beide Aktionsarten unter dem Eintrag "Skript (Action Response)" ein Skript angelegt werden.

Ein typisches Rückgabeskript:

```javascript
function actionResponse(element, context, actionResult){
  var actionResponse = new $k.ActionResponse();
  actionResponse.setFollowup("show");
  actionResponse.setData({
    elementId: actionResult.idString(),
    viewMode: "edit"
  });
  return actionResponse;
}
```

Mögliche Follow-Ups:

- show - Die übergebenen Inhalte werden geladen und auf der ganzen Seite dargestellt.
- refresh - Alle Komponenten aktualisieren ihre Inhalte.
- reload - Die Seite wird neu geladen.
- update - Die übergebenen Inhalte werden geladen und in der View dargestellt, die in "ausführen in" bestimmt wurde. Es kommt zu einem Fehler, wenn "ausführen in" nicht gesetzt ist.

Außerdem ist es möglich, eigene Follow-Ups zu definieren. Die Interaktion mit dem Frontend muss dann aber darauf angepasst werden, da vcm dafür standardmäßig nicht ausgelegt ist. Auf der anderen Seite ist es aber auch möglich, die derzeitigen Reaktionen auf einen Follow-Up im eigenen Anpassungsprojekt zu überschreiben.

Weitere mögliche Rückgabewerte:

- elementId - ID des anzuzeigenden Elements
- viewMode - Derzeit gibt es nur die Unterscheidung zwischen Lese- und Schreibmodus, wobei standardmäßig der Lesemodus angenommen wird und nur bei viewMode: "edit"
der Schreibmodus angezeigt wird.

Auch hier ist es möglich, eigene Werte zu definieren und sein Frontend darauf anzupassen.

### 3.4 View-Konfigurationselemente

#### 3.4.1 Allgemeines

Die einzelnen Views können über Styles weiter angepasst werden. Die für den VCM relevanten Styles befinden sich auf dem Reiter *Viewkonfiguration-Mapper*:

**Allgemeine Styles**

<table>
<thead>
<tr>
<th>Style Eigenschaft</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>class</code></td>
<td>Zuständliche CSS Style Klassen die am View angebracht werden</td>
</tr>
<tr>
<td><code>class (script)</code></td>
<td>Wie <code>class</code>, mit dem Unterschied dass die Klassen dynamisch per Skript berechnet werden können</td>
</tr>
</tbody>
</table>

### 3.4.2 Alternative

Bei einer Alternative-View handelt es sich um eine Sammel-View für andere Views, d.h. mithilfe dieses Viewtyps lassen sich Views zusammenfassen, die Daten zu einem gemeinsamen Objekt angeben (z.B. eine Eigenschaften-View mit den Lebensdaten eines Künstler und eine Tabelle-View, in der die Werke des Künstlers aufgelistet werden). Im Gegensatz zur Gruppen-View werden die zusammengefassten Views aber nicht gleichzeitig, sondern abwechselnd (z.B. über Reiter) angezeigt.

Um die Views zusammenzufassen, werden die entsprechenden Views als Unterviews an die Alternative-View angehängt. Ihre Position entscheidet dabei über die Reihenfolge der Anzeige. Deswegen kann die Position über die Pfeilbuttons geändert werden.
Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Liste:

<table>
<thead>
<tr>
<th>Beschreibung</th>
<th>Default</th>
<th>Konfigurationsname</th>
<th>Skript für &quot;Beschriftung&quot;</th>
<th>Skript für Sichtbarkeit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Der hier eingegebene Wert erscheint als Überschrift der Alternative</td>
<td>Default wird die erste angehängt View angezeigt. Falls man z.B. möchte, dass die Alternative auf dem dritten Reiter zuerst angezeigt wird, kann man diese View hier angeben. Im Frontend wird die zuletzt angezeigte View innerhalb einer Session vermerkt, so dass der Nutzer, wenn er innerhalb einer Session eine Alternative-View mehrfach betrachtet, auf den zuletzt von ihm betrachteten Reiter landet.</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der Titel der Alternative in einem Skript bestimmt werden.</td>
<td>Über dieses Skript kann festgelegt werden, ob und unter welchen Bedingungen die Alternative angezeigt werden soll.</td>
</tr>
</tbody>
</table>
Im Reiter "Menüs" lassen sich Aktionen zu der Alternative konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich konfigurieren, für welche Objekttypen die Alternative-View verwendet werden soll und in welchen Anwendungskontexten.

Eine Alternative-View sollte dann verwendet werden, wenn mehrere Views auf den Daten eines Objekts oder Typs basieren, aber nicht gleichzeitig sondern alternativ angezeigt werden sollen.

3.4.3 Gruppe

Bei einer Gruppen-View handelt es sich um eine Sammel-View für andere Views, d.h. mithilfe dieses Viewtyps lassen sich Views gruppieren, die Daten zu einem gemeinsamen Objekt anzeigen (z.B. eine Eigenschaften-View mit den Lebensdaten eines Künstler und eine Tabelle-View, in der die Werke des Künstlers aufgelistet werden). Um die Views zu gruppieren, werden die entsprechenden Views als Unterviews an die Gruppe-View angehängt. Ihre Position entscheidet dabei über die Reihenfolge der Anzeige. Deswegen kann die Position über die Pfeilbuttons geändert werden.

Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Liste:

<table>
<thead>
<tr>
<th>Beschriftung</th>
<th>Der hier eingegebene Wert erscheint als Überschrift der Gruppe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der Titel der Gruppe in einem Skript bestimmt werden.</td>
</tr>
</tbody>
</table>
Skript für Sichtbarkeit

Über dieses Skript kann festgelegt werden, ob die Gruppe angezeigt werden soll.

Im Reiter "Menüs" lassen sich Aktionen zu der Gruppe konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich konfigurieren, für welche Objekttypen die Gruppen-View verwendet werden soll und in welchen Anwendungskontexten.

Eine Gruppen-View sollte dann verwendet werden, wenn mehrere Views gleichzeitig gruppiert angezeigt werden sollen, die auf den Daten eines Objekts oder Typs basieren. Im Gegen- satz dazu steht die Alternative, die die enthaltenen Views zu einem Objekt alternierend (z.B. als Reiter) anzeigt.

Im Web-Frontend gibt es unterschiedliche Möglichkeiten, die gruppierten Views anzuzeigen. Falls nicht anders konfiguriert, werden die Views untereinander angeordnet. Mit einem Style kann man eine horizontale oder Grid-Anordnung ermöglichen:

![Screenshot]

Dazu wird ein Style-Objekt an der Gruppe-View angelegt und im Reiter Viewkonfiguration-Mapper wird als "renderMode" der Wert "grid" ausgewählt und bei "groupByColumnGrid" die gewünschte Grid-Konfiguration eingetragen.
Die Beispiel-View hat das Grid "4 8 0". Die Summe der Summanden muss immer zwölf ergeben.

Wählt man den "renderMode" "panel" aus, erhält man eine aufklappbare Gruppe.

Auch die aus Bootstrap bekannten "renderMode"-Werte "jumbotron" und "well" werden bei der Gruppe unterstützt.

3.4.4 Hierarchie

Bei einer Hierarchie-View handelt es sich um eine hierarchische Ansicht zu konfigurierbaren Aspekten eines Objekts.
Die Konfiguration im Knowledge-Builder erfolgt über das Anlegen einer Hierarchie-View.
Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Hierarchie:

<table>
<thead>
<tr>
<th>Beschreibung</th>
<th>Der hier eingegebene Wert erscheint als Überschrift der Hierarchie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skript für Beschreibung</td>
<td>Alternativ zu &quot;Beschreibung&quot; kann der Titel der Hierarchie in einem Skript bestimmt werden.</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
</tbody>
</table>
### Relation (aufsteigend) / Relation (absteigend) / Strukturabfrage (aufsteigend) / Strukturabfrage (absteigend) / Skript (aufsteigend) / Skript (absteigend)


### Absteigend sortieren

Per default wird die Hierarchie aufsteigend sortiert. Durch Aktivieren der Checkbox wird diese Sortierreihenfolge umgekehrt.

### Aktion (Auswahl)

An dieser Stelle kann die Aktion konfiguriert werden, die erfolgen soll, wenn ein Nutzer ein Element in der Hierarchie selektiert.

### Banner der Hierarchiewurzel anzeigen

Über diese Checkbox kann festgelegt werden, ob das Banner des Wurzelelements in der Hierarchie-View angezeigt werden soll. Diese Konfiguration wird nur im Knowledge-Builder berücksichtigt.

### Detailansicht ausblenden

Standardmäßig wird zu einem selektierten Knoten (siehe auch Aktion (Auswahl) eine Detailansicht angezeigt, dies kann über diese Option deaktiviert werden.

### Primäres Sortierkriterium

Über das Sortierkriterium wird bestimmt, nach welchem Aspekt die Sortierung der Hierarchieelemente auf eine Ebene erfolgen soll.

### Sekundäres Sortierkriterium

Wie "Primäres Sortierkriterium", wird aber nur genutzt, falls die errechnete Position aus "Primäres Sortierkriterium" bei zwei oder mehr Attributen gleich ist.

### Skript für Sichtbarkeit

Über dieses Skript kann festgelegt werden, ob die Liste angezeigt werden soll.

### Skript für Sortierung

Dieses Skript wird genutzt, wenn als primäres oder sekundäres Sortierkriterium "Skript für Sortierung" ausgewählt wurde.
| Verbiete manuelles Sortieren | Über diese Option wird die Möglichkeit eine Hierarchie durch den Nutzer umsortieren zu lassen unterbunden. Diese Option wird nur im Knowledge-Builder angewendet. |


### 3.4.5 Eigenschaften

Bei einer Eigenschaften-View handelt es sich um eine Liste von Attributen und Relationen zu einem Objekt.

![Eigenschaften-View](image)

Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Liste:
<table>
<thead>
<tr>
<th>Absteigend sortieren</th>
<th>Grundsätzlich werden die enthaltenen Attribute/Relationen in der Reihenfolge angezeigt, wie es die Reihenfolge der eingehängten Eigenschaft-Views vorgibt. Da es bei diesen aber möglich ist, übergeordnete Typen (z.B. &quot;Benutzerrelation&quot;) anzugeben, werden die so zusammengefassten Eigenschaften nach ihrem Namen aufsteigend sortiert. Durch Aktivieren der Checkbox &quot;Absteigend sortieren&quot; kann diese Reihenfolge geändert werden.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Der hier eingegebene Wert erscheint als Überschrift der Liste</td>
</tr>
<tr>
<td>Initial ausgeklappt</td>
<td>Im Knowledge-Builder werden ab einer großen Anzahl Eigenschaften nicht direkt angezeigt, sondern ausklappbar. Durch das Aktivieren dieser Option werden sie direkt ausgeklappt.</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
<tr>
<td>Sekundäres Sortierkriterium</td>
<td>Wie &quot;Primäres Sortierkriterium&quot;, wird aber nur genutzt, falls die errechnete Position aus &quot;Primäres Sortierkriterium&quot; bei zwei oder mehr Attributen gleich ist.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der Titel der Liste in einem Skript bestimmt werden.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>Über dieses Skript kann festgelegt werden, ob die Liste angezeigt werden soll.</td>
</tr>
<tr>
<td>Skript für Sortierung</td>
<td>Dieses Skript wird genutzt, wenn als primäres oder sekundäres Sortierkriterium &quot;Skript für Sortierung&quot; ausgewählt wurde.</td>
</tr>
</tbody>
</table>

Im Reiter "Menüs" lassen sich Aktionen zu der Liste konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich konfigurieren, für welche Objekttypen die Eigenschaften-View verwendet werden soll und in welchen Anwendungskontexten.  

Für die Lese-Ansicht kann die Eigenschaften-View alleinstehend verwendet werden, wird aber oft auch in Gruppe- oder Alternative-Views verwendet. Um die Bearbeitung der Objekteigenschaften zu ermöglichen, muss eine Eigenschaften-View in eine Edit-View eingehängt werden.  

Die Attribute und Relationen, die zu einem Objekt angezeigt werden sollen, sind konfigurabel. Dazu werden einer Eigenschaft-View Eigenschaft-Views hinzugefügt, an denen das/die entsprechende Attribut/Relation ausgewählt werden und bestimmt wird, wie diese im einzelnen dargestellt werden sollen.
3.4.6 Eigenschaft

Bei einer Eigenschaft-View handelt es sich um die Darstellungskonfiguration eines Attributs oder einer Relation zu einem Objekt. Eine Eigenschaft-View kann nur innerhalb einer Eigenschaften-View verwendet werden.

Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Attribute/Relationen:

<table>
<thead>
<tr>
<th>Option</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aktion (Auswahl)</td>
<td>Die hier konfigurierte Aktion wird ausgeführt, wenn der Eigenschaftswert im Frontend ausgewählt wird (z.B. per Klick).</td>
</tr>
<tr>
<td>anwenden auf/anwenden in</td>
<td>An dieser Stelle kann der Kontext genauer bestimmt werden, so dass unterschiedliche Konfigurationen einer Eigenschaft in einer Eigenschaften-View genutzt werden können.</td>
</tr>
<tr>
<td>Beschreibung</td>
<td>Eine Eigenschaft wird als Name-Wert-Paar in den Frontends dargestellt. Als default wird der Name der konfigurierten Eigenschaft genommen. Über &quot;Beschriftung&quot; kann ein anderer Wert als Name angezeigt werden.</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Einblendung zusätzlicher Eigenschaften</td>
<td>Falls das konfigurierte Attribut mehrfach vorkommen darf, kann man über diese Option festlegen, dass automatisch ein neuer Platzhalter für einen weiteren Attributwert angezeigt wird.</td>
</tr>
<tr>
<td>EinblendungsfILTER</td>
<td>Über dieses Skript können die vorhandenen Attribute dieses Typs gefiltert werden, sodass in der Darstellung nur bestimmte von ihnen berücksichtigt werden.</td>
</tr>
<tr>
<td>Konfiguration für eingebettete Metaeigenschaften/Konfiguration für Metaeigenschaften</td>
<td>Falls das Attribut Metaeigenschaften enthält, kann an dieser Stelle eine View angebracht werden, um die Darstellung dieser Eigenschaften zu bestimmen. Der einzige Unterschied bei den eingebrachten Metaeigenschaften ist, dass sie eingerückt unter dem Attribut erscheinen.</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der dargestellt Attributsname in einem Skript bestimmt werden.</td>
</tr>
<tr>
<td>Skript für Sichtbarkeit</td>
<td>Über dieses Skript kann festgelegt werden, ob diese Eigenschaft angezeigt werden soll.</td>
</tr>
<tr>
<td>Skript für Sortierung</td>
<td>Dieses Skript kann genutzt werden, wenn mehrere Ausprägungen einer Eigenschaft zu einem Objekt existieren.</td>
</tr>
</tbody>
</table>

Für Relationen gibt es noch weitere Optionen:
<table>
<thead>
<tr>
<th>Eigenschaftssicht</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Einblendung des Relationsziels</td>
<td>Per Default wird als Attributwert der Name des Relationszielobjekts als Link angezeigt. Über diese Option werden konfigurierte Eigenschaften des Relationsziels angezeigt. Die dafür verwendete View kann über ihren Kontext bestimmt werden, d.h. bei &quot;anwenden in&quot; der entsprechenden View sollte die Eigenschaft-View angegeben werden.</td>
</tr>
<tr>
<td>Relationszielfilter</td>
<td>Um den Benutzer in seiner Auswahl eines geeigneten Relationsziels zu unterstützen, kann hier eine Filterungsabfrage angebracht werden.</td>
</tr>
<tr>
<td>Relationszieltypfilter</td>
<td>Falls mehrere Objekttypen als Ziel einer Relation definiert wurden, kann an dieser Stelle ein Filter auf die angezeigten Typen konfiguriert werden.</td>
</tr>
<tr>
<td>Skript für Relationszielbezeichner</td>
<td>Per Default wird der Name des Relationszielobjekts angezeigt. Dies kann per Skript an dieser Stelle angepasst werden.</td>
</tr>
</tbody>
</table>
Im Reiter "Menüs" lassen sich weitere Aktionen zu der Eigenschaft konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich nachvollziehen, in welchen Views die Eigenschaft-View angewendet wird.

3.4.7 Edit

Eine Edit-View dient zur Pflege von Attributen oder Relationen.


Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Edit-View:

<table>
<thead>
<tr>
<th>Editiermodus umschaltbar</th>
<th>Über dieses Option ist der Formularmodus &quot;Umschaltbar&quot;. D.h. Eigenschaften werden zuerst im nur lesen Modus angezeigt. Über einen Umschaltbutton kann dann in den Pflegemodus geschaltet werden.</th>
</tr>
</thead>
</table>
**Benutzerdefinierte Schaltflächen**

Ist die Option *Nur benutzerdefinierte Schaltflächen* aktiv, können eigene Buttons zum Umschalten und Speichern konfiguriert werden. Dabei muss an der jeweiligen Aktion eine eigene ActionResponse mit definierten Followups konfiguriert werden:

<table>
<thead>
<tr>
<th>Aktion</th>
<th>Followup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speichern</td>
<td>edit-save</td>
</tr>
<tr>
<td>Umschalten Lesemodus</td>
<td>edit-state-read</td>
</tr>
<tr>
<td>Umschalten Pflegemodus</td>
<td>edit-state-edit</td>
</tr>
</tbody>
</table>

Beispiel für einen eigenen Speichern Button:

```javascript
function actionResponse(element, context, actionResult) {
    var actionResponse = new $k.ActionResponse();
    actionResponse.setFollowup("edit-save");
    return actionResponse;
}
```

Ab 5.2 ist dies für den Speichern-Button nicht mehr nötig. Dann genügt es, eine Update-Aktion zu konfigurieren und bei "ausführen in View" die Edit-View auszuwählen, in der das Speichern erfolgen soll.

**3.4.8 Tabelle**

Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung und des Verhaltens.

<table>
<thead>
<tr>
<th>Aktion (Auswahl)</th>
<th>Die hier konfigurierte Aktion wird ausgeführt, wenn eine Zeile im Frontend ausgewählt wird (z.B. per Klick).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anzahl Zeilen (Page size)</td>
<td>Hier wird die maximale Anzahl Zeilen angegeben, die auf einer Seite angezeigt werden.</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Eine Tabelle wird in den Frontends mit Überschrift dargestellt. Als default wird der Name aus dem Kontext erzeugt. Über &quot;Beschriftung&quot; kann ein anderer Wert als Name angezeigt werden.</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
<tr>
<td>Ohne Spaltenfilter</td>
<td>Hier kann bestimmt werden, ob zwischen Tabellen-Kopfleiste und Tabelleninhalt ein Spaltenfilter dargestellt werden soll. Anhand des Spaltenfilters kann für die jeweilige Spalte das Suchergebnis durch Eingabe eines Begriffs gefiltert werden.</td>
</tr>
</tbody>
</table>
Im Reiter "Sortierung" kann das Sortierverhalten anhand der Spalten konfiguriert werden. Der Reiter "Tabelle" enthält zwei Unterpunkte: "Menüs" und "Styles". Im Reiter "Menüs" lassen sich weitere Aktionen zu der Tabelle konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen, die sich auf die gesamte Tabelle auswirken ausgewählt werden. Im nächsten Reiter, "Spalten" > "Styles" werden dann entsprechend die Darstellungsoptionen für Spalten ausgewählt.

Die Spalten der Tabelle werden durch Subkonfigurationen definiert, siehe nächster Abschnitt. Die Reihenfolge der Spalten kann in der Baumansicht auf der linken Seite mittels der Pfeil-Buttons darüber verändert werden.

Die Spalten-View repräsentiert die Konfiguration einer gesamten Spalte. Hier lässt sich Einfluss auf die Darstellung und das Verhalten (z.B. Filterung) nehmen.

Der Inhalt der Zellen ("Spaltenelement") wird wiederum durch eine Subkonfiguration festgelegt, wie im folgenden Abschnitt beschrieben.
<table>
<thead>
<tr>
<th>Beschreibung</th>
<th>Wertung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breite der Spalte (%)</td>
<td>Breite der Spalte in Prozent, bezogen auf die Breite der Tabelle</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
<tr>
<td>Nicht anzeigen</td>
<td>Hiermit wird eine Spalte ausgeblendet. Sie wird trotzdem im Hintergrund berechnet und kann z.B. für die Sortierung verwendet werden.</td>
</tr>
<tr>
<td>Nicht sortierbar</td>
<td>In der Standardeinstellung lassen sich Spalten mit Klick auf den Header sortieren. Hier lässt sich diese Funktionalität deaktivieren.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der dargestellte Attributsname in einem Skript bestimmt werden.</td>
</tr>
<tr>
<td>Skript für Vorverarbeitung von Eingabefeldern</td>
<td>Der Text, der im Spaltenfilter angegeben wurde, lässt sich hier mittels Skript beeinflussen.</td>
</tr>
<tr>
<td>Suchtext</td>
<td>Hier lässt sich der Text für den Spaltenfilter im Voraus festlegen.</td>
</tr>
</tbody>
</table>

Die Spaltenelement Sub-Konfiguration bestimmt den Inhalt der Spalte. Der Inhalt wird typischerweise von den Elementen abgeleitet, auf die sich die Tabelle bezieht.
<table>
<thead>
<tr>
<th>Eigenschaft</th>
<th>Die in dieser Spalte anzuzeigende Eigenschaft des Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strukturabfrage-Baustein</td>
<td>Alternativ zu &quot;Eigenschaft&quot; kann der anzuzeigende Inhalt auch über eine Strukturabfrage bestimmt werden</td>
</tr>
<tr>
<td>Skript</td>
<td>Alternativ zu den ersten beiden Möglichkeiten, kann der anzuzeigende Inhalt auch über ein Skript vom Element abgeleitet werden</td>
</tr>
<tr>
<td>Nicht anzeigen</td>
<td>Hiermit wird das Spaltenelement ausgeblendet. Es wird trotzdem im Hintergrund berechnet und kann z.B. für die Sortierung oder Filterung verwendet werden.</td>
</tr>
<tr>
<td>Hervorhebung</td>
<td>Hier kann ausgewählt werden, ob der Inhalt des Spaltenelements durch Unterstreichung hervorgehoben werden soll.</td>
</tr>
<tr>
<td>Hits verwenden</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der dargestellte Attributname in einem Skript bestimmt werden.</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
</tbody>
</table>
3.4.9 Suche

Mit einer Suche-View lassen sich Suchseiten erstellen, auf denen die Suchabfrage und die Ergebnisse gleichzeitig angezeigt werden. Sollte die Suche keine oder nur optionale Parameter haben, wird die Suche sofort ausgeführt und die Ergebnisse direkt angezeigt. Gibt es obligatorische Parameter, wird die Suche erst nach einer Benutzereingabe ausgeführt.

<table>
<thead>
<tr>
<th>Name</th>
<th>Wahlberechtigte</th>
<th>Wähler</th>
<th>Gültige Stimmen</th>
<th>Ungültige Stimmen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aarbergen</td>
<td>4.588</td>
<td>1.999</td>
<td>1.983</td>
<td>16</td>
</tr>
<tr>
<td>24.02.2013</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abtsteinach</td>
<td>2.040</td>
<td>1.412</td>
<td>1.389</td>
<td>23</td>
</tr>
<tr>
<td>27.03.2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ahnental 09.11.2014</td>
<td>6.657</td>
<td>2.839</td>
<td>2.790</td>
<td>49</td>
</tr>
<tr>
<td>Allendorf (Eder) 14.08.2011</td>
<td>4.212</td>
<td>1.335</td>
<td>1.329</td>
<td>6</td>
</tr>
</tbody>
</table>


Für eine einfache Suchseite legt man im Knowledge-Builder eine Suche-View an.
Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Suche:

Als erstes wird die Suche konfiguriert, die beim Ausführen der Suche ausgeführt werden soll.

- Der Name eines Suchparameters. Alle Parameter, die in der Suche konfiguriert wurden, müssen an dieser Stelle auch konfiguriert werden, damit es nicht zu Fehlern in der Suche kommt.

- Skriptfall der Parameterwert über ein Skript ermittelt werden soll, muss dieses hier konfiguriert werden.
Wertier wird angegeben, wie der Parameterwert bestimmt werden soll.
- "Skript" (Wertermittlung per Skript)
- "Skript, überschreibbar" (Wertermittlung per Skript, wird aber über Benutzereingaben im Frontend überschrieben
  - "Benutzbereingabe (optional)" (Der Parameterwert wird aus der Benutzereingabe übernommen, falls er gesetzt wird. Er wird dem Nutzer als optional im Frontend angezeigt. Bitte beachten, dass die Suche dann auch so konfiguriert ist, dass dieser Parameter nicht gesetzt sein muss)
  - "Benutzereingabe (obligatorisch)" (Der Nutzer muss im Frontend einen Wert eingeben, ansonsten wird die Suche nicht ausgeführt)
  - "Benutzereingabe (deaktiviert, wenn leer)" (Der Parameter wird für die Suche gesetzt, wenn es eine Benutzereingabe gab. Ansonsten wird der Parameter beim Ausführen der Suche deaktiviert)

<table>
<thead>
<tr>
<th>Typ</th>
<th>Datentyp des Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Bezeichnung des Parameters im Frontend</td>
</tr>
<tr>
<td>Rei</td>
<td>Die Reihenfolge der Parameter, wie sie im Frontend angezeigt werden kann</td>
</tr>
<tr>
<td>Bestätigung</td>
<td>Eingabewerte erscheinen als Überschrift der Suche</td>
</tr>
<tr>
<td>Konfiguration</td>
<td>Konfigurationen können Views und Panels identifiziert werden</td>
</tr>
<tr>
<td>Skript</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der Titel der Gruppe in einem Skript bestimmt werden</td>
</tr>
<tr>
<td>Sichtbarkeit</td>
<td>Skript über dieses Skript kann festgelegt werden, ob die Gruppe angezeigt werden soll</td>
</tr>
<tr>
<td>Tabelle</td>
<td>Alternativ zu &quot;Tabelle&quot; kann an dieser Stelle über ein Skript die dargestellte Tabelle bestimmt werden</td>
</tr>
</tbody>
</table>
Im Reiter "Menüs" lassen sich Aktionen zu der Suche konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich konfigurieren, für welche Objekttypen die Suche-View verwendet werden soll und in welchen Anwendungskontexten.

3.4.9.1 Suchfeld-Ansicht

Eine Suchfeld-Ansicht wird verwendet, wenn z.B. nur ein Suchschlitz an einer Stelle angezeigt werden soll, nicht aber die Suchergebnisse. Die Konfiguration erfolgt wie bei der Suche-View, jedoch ohne die Konfiguration zur Ergebnisdarstellung.

Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung des Suchfeldes:

Abfrage: wird die Suche konfiguriert, die beim Ausführen der Suche ausgeführt werden soll.
Der Name eines Suchparameters. Alle Parameter, die in der Suche konfiguriert wurden, müssen an dieser Stelle auch konfiguriert werden, damit es nicht zu Fehlern in der Suche kommt.

<table>
<thead>
<tr>
<th>Skript</th>
<th>der Parameterwert über ein Skript ermittelt werden soll, muss dieses hier konfiguriert werden.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werte</td>
<td>wird angegeben, wie der Parameterwert bestimmt werden soll:</td>
</tr>
<tr>
<td></td>
<td>- &quot;Skript&quot; (Wertermittlung per Skript)</td>
</tr>
<tr>
<td></td>
<td>- &quot;Skript, überschreibbar&quot; (Wertermittlung per Skript, wird aber über Benutzeingaben im Frontend überschrieben)</td>
</tr>
<tr>
<td></td>
<td>- &quot;Benutzereingabe (optional)&quot; (Der Parameterwert wird aus der Benutzereingabe übernommen, falls er gesetzt wird. Er wird dem Nutzer als optional im Frontend angezeigt. Bitte beachten, dass die Suche dann auch so konfiguriert ist, dass dieser Parameter nicht gesetzt sein muss)</td>
</tr>
<tr>
<td></td>
<td>- &quot;Benutzereingabe (obligatorisch)&quot; (Der Nutzer muss im Frontend einen Wert eingeben, ansonsten wird die Suche nicht ausgeführt)</td>
</tr>
<tr>
<td></td>
<td>- &quot;Benutzereingabe (deaktiviert, wenn leer)&quot; (Der Parameter wird für die Suche gesetzt, wenn es eine Benutzereingabe gab. Ansonsten wird der Parameter beim Ausführen der Suche deaktiviert)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Typ</th>
<th>Datentyp des Parameters</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bezeichnung</th>
<th>der Parameters im Frontend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reihenfolge</td>
<td>Die Reihenfolge der Parameter, wie sie im Frontend angezeigt werden</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bestätiger eingeggebene Wert erscheint als Überschrift der Suche</th>
</tr>
</thead>
<tbody>
<tr>
<td>Konfigurationenamen können Views und Panels identifiziert werden.</td>
</tr>
</tbody>
</table>

Im Reiter "Menüs" lassen sich Aktionen zur Suchfeld-Ansicht konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich konfigurieren, für welche Objekttypen die Suchfeldansicht verwendet werden soll und in welchen Anwendungskontexten.

3.4.9.2 Suchergebnis-Ansicht


Sie kann im Knowledge-Builder angelegt werden.
Im Reiter "Konfiguration" gibt es Möglichkeiten zur Bestimmung der allgemeinen Darstellung der Suche:

<table>
<thead>
<tr>
<th>Einstellung</th>
<th>Erklärung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abfrage</td>
<td>Hier wird die Suche konfiguriert, die beim Ausführen der Suche ausgeführt werden soll.</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Der hier eingegebene Wert erscheint als Überschrift der Suche</td>
</tr>
<tr>
<td>Konfigurationname</td>
<td>Über den Konfigurationsnamen können Views und Panels identifiziert werden.</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Alternativ zu &quot;Beschriftung&quot; kann der Titel der Gruppe in einem Skript bestimmt werden.</td>
</tr>
<tr>
<td>Skript für Tabellekonfiguration</td>
<td>Alternativ zu &quot;Tabelle&quot; kann an dieser Stelle über ein Skript die dargestellte Tabelle bestimmt werden.</td>
</tr>
<tr>
<td>Tabelle</td>
<td>Die Suchergebnisse werden im Frontend in der Tabellenkonfiguration angezeigt, die hier konfiguriert wird.</td>
</tr>
</tbody>
</table>

Im Reiter "Menüs" lassen sich Aktionen zu der Suche konfigurieren, und im Reiter "Styles" können bestimmte Darstellungsoptionen ausgewählt werden. Der "KB"-Reiter enthält Optionen, die nur für den Knowledge-Builder gelten und im Web-Frontend keine Anwendung finden. Über den "Kontext"-Reiter lässt sich konfigurieren, für welche Objekttypen die Suche-View verwendet werden soll und in welchen Anwendungskontexten.
3.4.9.3 Facetten-Ansicht

<table>
<thead>
<tr>
<th>Name</th>
<th>Wahltag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad Arlosen, St. 09.02.2014</td>
<td>09.02.2014</td>
</tr>
<tr>
<td>Bad Camberg, St. 07.11.2010</td>
<td>07.11.2010</td>
</tr>
<tr>
<td>Bad Emstal 25.03.2012</td>
<td>25.03.2012</td>
</tr>
<tr>
<td>Bad Endbach 14.08.2011</td>
<td>14.08.2011</td>
</tr>
<tr>
<td>Bad Hersfeld, Kreisstadt 07.11.2010</td>
<td>07.11.2010</td>
</tr>
<tr>
<td>Bad Hersfeld, Kreisstadt 21.11.2010</td>
<td>21.11.2010</td>
</tr>
<tr>
<td>Bad Homburg v. d. Höhe, St 14.06.2015</td>
<td>14.06.2015</td>
</tr>
<tr>
<td>Bad Homburg v. d. Höhe, St 28.06.2015</td>
<td>28.06.2015</td>
</tr>
<tr>
<td>Bad Karlsfeld, St. 27.03.2011</td>
<td>27.03.2011</td>
</tr>
<tr>
<td>Bad König, St. 09.09.2012</td>
<td>09.09.2012</td>
</tr>
<tr>
<td>Bad Nauheim, St. 10.04.2011</td>
<td>10.04.2011</td>
</tr>
<tr>
<td>Bad Nauheim, St. 27.03.2011</td>
<td>27.03.2011</td>
</tr>
</tbody>
</table>

Eine Facetten-Ansicht lässt sich im Knowledge-Builder konfigurieren.
Um Facetten zu konfigurieren, werden Facette-Views erstellt und an die Facetten-Ansicht gehängt. Diese können mehrfach hierarchisch angeordnet werden.

<table>
<thead>
<tr>
<th>Abfrage</th>
<th>Die Suche, mit der die Ansicht verknüpft ist. Wenn die Ansicht mit anderen verwandten Ansichten wie z.B. der Suchfeld-Ansicht verknüpft ist, muss hier keine Abfrage konfiguriert werden.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschreibung</td>
<td>Der Titel, der über der Facetten-Ansicht im Frontend erscheinen soll.</td>
</tr>
<tr>
<td>Konfigurationsname</td>
<td>Über Konfigurationsnamen lassen sich Views und Panels identifizieren</td>
</tr>
<tr>
<td>Skript für Beschreibung</td>
<td>Alternativ zu einer festen Beschreibung kann der Titel auch über ein Skript gesetzt werden.</td>
</tr>
</tbody>
</table>
Abfrage zur Elterntermermittlung

Abfrage zur Termermittlung

Ausblenden ab Anzahl von Termen

Beschriftung

Kind terme initial anzeigen

Konfigurationsname

<p>| <strong>Abfrage zur Elterntermermittlung</strong> | Strukturabfrage, über die die Facette gebildet wird. Die Struktur einer solchen Abfrage wird im nächsten Abschnitt erklärt. |
| <strong>Abfrage zur Termermittlung</strong> |
| <strong>Ausblenden ab Anzahl von Termen</strong> | Die Facette wird ausgeblendet, wenn die der Facette zugrundeliegenden Suchergebnisse diese Anzahl übersteigen. |
| <strong>Beschriftung</strong> | Der Titel der Facette |
| <strong>Kind terme initial anzeigen</strong> | Falls die Facette hierarchisch aufgebaut ist, kann über diese Option definiert werden, ob die Unter-Facetten initial angezeigt werden sollen. |
| <strong>Konfigurationsname</strong> | Views und Panels können über einen Konfigurationsnamen identifiziert werden. |</p>
<table>
<thead>
<tr>
<th>Leere Term anzeigen</th>
<th>Falls zu der Facette keine Ergebnisse gefunden werden, wird sie per Default ausgeblendet. Über diese Option wird sie trotzdem angezeigt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximale Anzahl von Termen</td>
<td>Beschreibt die maximale Anzahl der Termen, die die Facette haben kann.</td>
</tr>
<tr>
<td>Term-Anzahl nicht anzeigen</td>
<td>Im Frontend wird hinter dem Facettentitel die Anzahl der gefundenen Termen angezeigt. Über diese Option kann das deaktiviert werden.</td>
</tr>
<tr>
<td>Term-Operator</td>
<td>An dieser Stelle kann konfiguriert werden, wie die Termen miteinander verknüpft werden. Möglich ist eine &quot;Und&quot; oder eine &quot;Oder&quot;-Verknüpfung.</td>
</tr>
<tr>
<td>Termart</td>
<td>Dynamisch oder statisch.</td>
</tr>
<tr>
<td>Terme absteigend sortieren</td>
<td>Per Default werden die für eine Facette gefundenen Termen aufsteigend sortiert. Über diese Option dreht sich die Sortierreihenfolge.</td>
</tr>
<tr>
<td>Terme nach Anzahl sortieren</td>
<td>Die Facetten-Termarten werden grundsätzlich alphabetisch sortiert, über diese Option werden sie nach der gefundenen Anzahl Ergebnisse sortiert.</td>
</tr>
</tbody>
</table>

Die für die Termermittlung genutzte Strukturabfrage muss sich an ein paar Vorgaben halten, damit eine Facette ermittelt werden kann.
Im Prinzip ist alles möglich, was sonst auch in Strukturabfragen möglich ist. Wichtig ist nur, dass die Ausprägungen einer Facette über den Bezeichner "term" ermittelt werden. Es ist auch möglich, dass der Bezeichner mehrfach in einer Strukturabfrage verwendet wird. Dann wird er über den bei "Term-Operator" definierten Wert verknüpft.

### 3.4.10 Graph-Konfiguration

Eine Graph-Konfiguration dient dazu, Objekte in einem Graphen darstellen zu können. Eine erste Einführung zur Nutzung des Graphen im Knowledge-Builder ist unter Knowledge-Builder > Grundlagen > Graph-Editor zu finden.

Details zu den Einstellungsmöglichkeiten der verschiedenen Views, die für das Einbinden eines Graphen in das Frontend benötigt werden, sind unter Knowledge-Builder > View-Konfiguration > View-Konfigurationselemente > Graph erläutert.

Es wird eine **Graph-View** sowie eine **Graph-Konfigurations-View** zur Darstellung benötigt. Das Panel, in dem der Graph angezeigt werden soll, erhält eine Graph-View ("V:Graph"). Die Angabe des Kontextelements ist optional und wird beim Start der Applikation im Graphen angezeigt.

Die Graph-View muss nur eine Verknüpfung zur Graph-Konfiguration enthalten. Optional, aber meist vorhanden, ist die Einstellung der Größe des Graph-Feldes über die Felder Breite und Höhe, sowie die Strukturabfrage für Startelemente.
In der Regel sieht die Strukturabfrage so aus:

Über die Angabe der Anzahl der Wiederholungen an der Benutzerrelation kann festgelegt werden, wie viele Schritte vom Ausgangsknoten aus angezeigt werden sollen, wenn ein Objekt angezeigt wird. (Achtung: Kann zu langen Rechenzeiten führen!)

Die Graph-View sorgt dafür, dass der Graph insgesamt angezeigt wird. Um festzulegen, welche Knoten und Relationen angezeigt werden sollen, wird die Graph-Konfiguration verwendet.
Für jeden Typen, dessen Objekte (oder auch Typen) angezeigt werden sollen, muss eine Knotenkategorie angelegt werden. Diese werden standardmäßig als Legende im Graphen angezeigt.

Im Graphen werden Objekte angezeigt, die direkt am Typen oder an dessen Untertypen hängen. Über An konkreten Typ anpassen werden die Untertypen gesondert in der Legende angezeigt, ohne dass diese einzeln als Knotenkategorien angelegt werden müssen.

Um Typen statt Objekte anzuzeigen, muss im Reiter Kontext der Haken bei anwenden auf Untertypen gesetzt sein.

Im Reiter Knoten kann unter Menüs ein Satellitenmenü vergeben werden, um weiter im Graphen zu arbeiten (siehe Knowledge-Builder > View-Konfiguration > Aktionen > Aktionen für den Viewconfiguration-Mapper > NN-Expand/NN-Hide/NN-Pin Aktionen).

Um die Relationen zwischen den Knoten anzuzeigen, wird eine Verknüpfung unter jeder Knotenkategorie benötigt. Hier wird festgelegt, welche Relationen für diesen Typen angezeigt werden sollen. Die Relationen können über eine Abfrage, ein Skript oder über die direkte
Bestimmung der Relation festgelegt werden. *Benutzerrelation* kann vergeben werden, wenn alle Relationen (außer Systemrelationen) angezeigt werden sollen.

Für weitere Details siehe Kapitel *vcm-plugin-net-navigator*

### 3.4.11 Text

Mit der Text-View kann Text dargestellt werden, der entweder statisch vorgegeben oder durch ein Skript berechnet wird.

<table>
<thead>
<tr>
<th>Text</th>
<th>Statischer, mehrsprachiger Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skript für Text</td>
<td>Skript zur Berechnung des Textes</td>
</tr>
<tr>
<td>Beschriftung</td>
<td>Optionale Überschrift</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Optionales Skript zur Berechnung der Überschrift</td>
</tr>
</tbody>
</table>

Beispiel für ein Text-Skript:

```javascript
function text(element) {
    return "Durch ein Skript im Netz " + $k.volume() + " generierter Text";
}
```
### 3.4.12 Bild

Stellt ein im Wissensetz gespeichertes Bild dar, das entweder statisch vorgegeben oder durch ein Script berechnet wird.

<table>
<thead>
<tr>
<th>Bild</th>
<th>Statisches Bild</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beschriftung</td>
<td>Optionale Überschrift</td>
</tr>
<tr>
<td>Skript für Beschriftung</td>
<td>Optionales Skript zur Berechnung der Überschrift</td>
</tr>
<tr>
<td>Breite / Höhe</td>
<td>Fest vorgegebene Breite / Höhe des Bildes</td>
</tr>
</tbody>
</table>

---

### 3.4.13 Skriptgeneriertes HTML

Dieser View erzeugt per Skript HTML. Sowohl Knowledge-Builder als auch ViewConfigMapper zeigen dies ungefiltert dar, d.h. es ist Aufgabe des Script-Entwicklers, Nutzerinhalte nicht ungefiltert auszugeben. Im Knowledge-Builder sind die Anzeigemöglichkeiten stark eingeschränkt (u.a. kein CSS).

Bei umfangreichem HTML sollte man besser einen scriptgenerierten View verwenden. Als Parameter werden an das Skript folgende Argumente übergeben:

<table>
<thead>
<tr>
<th>element</th>
<th>$k.SemanticElement</th>
<th>Das Element, in dessen Kontext der View dargestellt wird</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>$k.TextDocument</td>
<td>Dokument, auf das HTML ausgegeben wird</td>
</tr>
</tbody>
</table>

Zur Ausgabe des HTML kann man zwei Ansätze wählen:

- Ausgabe des HTML-Quellcodes per Funktion print() des Dokuments
- Strukturierte Ausgabe per XMLWriter

Das nachfolgende Beispiel zeigt die Verwendung eines XMLWriters, um eine Überschrift auszugeben:

```/**```
3.4.14 Skriptgenerierte View

Ein scriptgenerierte View ermöglicht es, eigene View-Komponenten zu definieren. Die Daten werden durch ein Skript erzeugt und per JSON weitergegeben. Es ist Aufgabe des Frontends, diese darzustellen.

An das Skript werden zwei Parameter übergeben:

<table>
<thead>
<tr>
<th>Element</th>
<th>$k.SemanticElement, in dessen Kontext der View angezeigt wird</th>
</tr>
</thead>
<tbody>
<tr>
<td>view object</td>
<td>Vorbefülltes Objekt mit den Viewdaten. Konfigurationselemente wie z.B. Styles sind hier bereits enthalten.</td>
</tr>
</tbody>
</table>

Nachfolgendes Skript liefert die Daten für einen View, der im Plugin `vcm-plugin-timeline` enthalten ist:

```javascript
/**
 * Get json object to modify.
 * @function
 * @this $k.View
 * @param {$k.SemanticElement} element
 * @param {object} json object
 * @returns {object} modified json object
 **/
function customizeView (element, view) {
    view.options = {
```
3.5 Plugins

3.5.1 vcm-plugin-calendar

Mit dem vcm-plugin-calendar können Daten in einem Kalender dargestellt werden.

Um die Daten als Kalender anzuzeigen, muss an der Tabellenkonfiguration ein Style-Element hinzugefügt werden, das den renderMode calendar enthält. Der Wert unter Anzahl Zeilen (Page Size) gibt an, wie viele Kalendereinträge maximal pro Ansicht (in diesem Fall pro Monat) angezeigt werden können. Die Tabelle muss die folgenden Spalten haben:

- **start**: Ein Datum an dem der Kalendereintrag startet.
- **end**: Enddatum des Eintrags (optional)
- **title**: Der Titel des Eintrags
- **allDay**: Boolean-Wert der angibt, ob der Eintrag für den gesamten Tag gilt (optional)
- Weitere Möglichkeiten für Spalten sind in der fullcalendar.io Event_Object Dokumentation zu finden.

Es kann auch eine Auswahlaktion auf die Spalten der Tabelle konfiguriert werden. Diese wird dann beim Klick auf einen Kalendereintrag ausgeführt.

Außerdem lassen sich mit dem Style Attribut vcmPluginCalendarOptions weitere Konfigurationen vornehmen.

Weiterführende Informationen zum Plugin sind unter fullcalendar.io abrufbar.
### 3.5.2 vcm-plugin-chart

Das vcm-plugin-chart dient dazu, Daten aus einer Tabellenkonfiguration in Form eines Diagramms im Web-Frontend anzeigen zu können. Es stehen verschiedene Diagrammtypen zur Verfügung: Linien-, Balken-, Torten-, Ring- und Radardiagramme.

Beispiel eines Balkendiagramms:

![Balkendiagramm Beispiel](image)

Beispiel eines Tortendiagramms:

![Tortendiagramm Beispiel](image)

#### 3.5.2.1 Konfiguration

Das vcm-plugin-chart dient dazu, Daten aus einer Tabellenkonfiguration in Form eines Diagramms im Web-Frontend anzeigen zu können. Hierbei muss ein Style angelegt werden, der als renderMode die Option "chart" hat.

Wird beispielsweise in der zugrundeliegenden Tabellenkonfiguration eine Aktion mit der Option "Graphisch darstellen" angefügt, so kann durch Klick auf Teile des Diagramms der jeweilige Datensatz zlasszlich im Net-Navigator angezeigt werden.
Für den vcm-plugin-chart gibt es mehrere Optionen zur Darstellungsanpassung:

- **vcmPluginChartDataColumns**
- **vcmPluginChartDataMode**
- **vcmPluginChartHeight:**
  Angabe der Diagrammhöhe in Pixel
- **vcmPluginChartLabelColumn**
- **vcmPluginChartOptions:**
  JavaScript für Anpassung von Legenden-Anzeige und Skalierung von Achsen
- **vcmPluginChartType:**
  Angabe zu Diagrammtyp: bar, doughnut, line, pie, polar, radar
- **vcmPluginChartWidth:**
  Angabe der Diagrammbreite in Pixel

### 3.5.3 vcm-plugin-html-editor

**Web-Frontend**

Das vcm-plugin-html-editor ermöglicht es, HTML-formatierten Text zu bearbeiten. Es verwendet hierfür den WYSIWYG-Editor summernote.

**Konfiguration**

Für die View-Konfiguration des HTML-Editors wird eine Gruppe benötigt: Im Reiter "Kontext" wird hierzu unter "anwenden auf" der Eintrag "vcm-plugin-html-editor" benötigt.
Nach dem der Konfiguration der Gruppe muss eine Eigenschafts-Konfiguration angelegt werden. Diese ist mit einem Style zu versehen, welchen den renderMode "htmleditor" enthält:

Die Eigenschafts-Konfiguration benötigt darüber hinaus ein Zeichenketten-Attribut, welches den Text enthalten soll.

3.5.4 vcm-plugin-markdown

Das vcm-plugin-markdown ermöglicht die HTML Ausgabe von Markdown Texten. Es lässt sich verwenden indem man einen Style mit dem renderMode `markdown` an eines der folgenden Konfigurationselemente anbringt:

- **Statischer Text**: Hierbei wird die Viewconfig-Eigenschaft Text des Konfigurationselements als Markdown interpretiert.

Für die Anwendung des Plugins muss im Reiter "Style" der RenderMode mit der Bezeichnung "markdown" eingegeben werden:

- **Eigenschaft**: hierbei wird der Wert des Attributes als Markdown interpretiert.
Die Konfiguration der View für das Zeichenketten-Attribut "Markdown" findet mithilfe einer Eigenschafts-View statt:

Die Eigenschaft erhält wie ein Text-Objekt gleichermaßen den RenderMode "markdown".

Nach dem Rendern erhält der Text im Web-Frontend die optischen Hervorhebungen:

Auch hier aktiviert der *renderMode* `markdown` die Umwandlung von Markdown in HTML.

Weitere Konfiguration des Plugins kann über das Style-Attribut `vcmPluginMarkdownOptions` vorgenommen werden.

Das Plugin verwendet das Modul markdown-it
3.5.5 **vcm-plugin-timeline**

Mit dem Plugin vcm-plugin-timeline lassen sich Ereignisse chronologisch auf einem Zeitstrahl anzeigen.

Der Zeitstrahl kann horizontal oder vertikal ausgerichtet werden. In der horizontalen Variante bietet die Timeline zusätzlich zwei Knöpfe zum scrollen an, sollte die Zeitleiste breiter als der zur Verfügung stehende Platz sein. Bei der vertikalen Variante sollte in diesem Fall vom Browser eine Scrollbar angeboten werden.

![Zeitstrahl mit Ereignissen](image)

### 3.5.5.1 Konfiguration

Es muss zunächst eine "Skriptgenerierte View" angelegt werden und ihr viewType-Attribut auf "timeline" gesetzt werden. Außerdem muss ein Skript an der View angebracht werden, welches die Daten für die Timeline bereit stellt, beispielsweise:

```javascript
function customizeView (element, view) {
    //other content ....
    view.options = {
        layout: 'horizontal',
        // layout: 'vertical',
        itemHeight: 130
    }
    view.events = element.relationTargets('hasAlbum').map(function (album) {
        var obj = {
            name: album.name(),
            date: album.attributeValue('releaseDate')
            if (date) { date = date.toString() } else { date = '' }
            return obj = {name: name, date: date, elementId: album.idString()}
        }
    return view
}
```

Über das Skript können mit den folgenden Parameter unter 'view.options' das Erscheinungsbild der Timeline angepasst werden:

- 'layout': Bestimmt die Richtung des Zeitstrahls, entweder 'horizontal' oder 'vertical'.
- 'itemHeight': Höhe der Elemente auf der Zeitleiste in Pixeln. Falls nicht gesetzt, erhalten alle Elemente die Höhe des Elements mit dem größten Platzbedarf.

Unter 'view.events' muss ein Array angelegt werden, welches die Ereignisse als Objekte enthält. Diese benötigen jeweils die Attribute 'name', 'date' und 'elementId'.
3.5.5.2 Styling

Mittels CSS-Regeln lässt sich der Default-Style der Timeline anpassen.

Hierfür steht, je nach konfigurierter Ausrichtung der Zeitleiste, die folgende Klassenhierarchie zur Verfügung:

Die Textfelder der Ereignisse lassen sich mit den folgenden Selektoren anpassen:

```css
.timelineVertical ul li
.timelineHorizontal ul li
```

Die Markierungspunkte der Ereignisse lassen sich über folgenden Selektor anpassen:

```css
.timelineVertical ul li::after
.timelineHorizontal ul li::after
```

3.5.6 vcm-plugin-page

3.5.7 vcm-plugin-net-navigator

Das vcm-plugin-net-navigator visualisiert Elemente in einer graphartigen Ansicht.

3.5.7.1 Konfiguration

Das Plugin kann über Styles konfiguriert werden.

Styles der View
<table>
<thead>
<tr>
<th>Style</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcmPluginNetNavigatorOptions</td>
<td>Ein JSON Objekt für die View Optionen. Details s. unten</td>
</tr>
<tr>
<td>extra</td>
<td>Alternativ zu vcmPluginNetNavigatorOptions</td>
</tr>
</tbody>
</table>

**Optionen**

<table>
<thead>
<tr>
<th>Option</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcmPluginNetNavigatorOptions.categories.hideLabel</td>
<td>Ein-/Ausblenden der Kategorielabels</td>
</tr>
<tr>
<td>vcmPluginNetNavigatorOptions.categories.embeddedActions</td>
<td>Konfiguration wo die Aktionen angezeigt werden sollen. Bei true werden sie neben den Kategorien angezeigt</td>
</tr>
<tr>
<td>vcmPluginNetNavigatorOptions.categories.compactActions</td>
<td>Aktionen in einem Menü zusammenfassen</td>
</tr>
<tr>
<td>vcmPluginNetNavigatorOptions.history.enabled</td>
<td>Aktiviert/Deaktiviert die Navigationshistorie</td>
</tr>
<tr>
<td>vcmPluginNetNavigatorOptions.enableEditing</td>
<td>Aktiviert-/Deaktiviert die Möglichkeit Elemente im Graph neu zu verknüpfen</td>
</tr>
<tr>
<td>vcmPluginNetNavigatorOptions.nnOptions</td>
<td>Optionen für die Net-Navigator Komponente</td>
</tr>
</tbody>
</table>

**Styles der Knoten**

| extra | Ein JSON Objekt für die Knoten Optionen. Details s. unten |

**Optionen der Knoten**

<p>| color | Überschreibt die Hintergrundfarbe des Knotens |</p>
<table>
<thead>
<tr>
<th>label</th>
<th>Überschreibt das Label des Knotens</th>
</tr>
</thead>
<tbody>
<tr>
<td>icon</td>
<td>Überschreibt das Icon des Knotens</td>
</tr>
</tbody>
</table>

**Styles der Kanten**

| extra | Ein JSON Objekt für die Kanten Optionen. Details s. unten |

**Optionen der Kanten**

<table>
<thead>
<tr>
<th>color</th>
<th>Überschreibt die Hintergrundfarbe der Kante</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>Überschreibt das Label der Kante</td>
</tr>
</tbody>
</table>

**3.5.7.2 Aktionen**

Knoten und Relationen können um Aktionen erweitert werden. Diese werden Kreisförmig um einen Knoten bzw. die Relation angeordnet.

Aktionen werden in der Graph-Konfiguration innerhalb von einer Knotenkategorie bzw. Verknüpfung konfiguriert.
Vorkonfigurierte Aktionen

<table>
<thead>
<tr>
<th>Aktionssart</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN-Expand</td>
<td>Über ein kleines Plus Symbol können benachbarte Knoten (für die es eine Konfiguration gibt) angezeigt werden</td>
</tr>
<tr>
<td>NN-Hide</td>
<td>Ausblenden eines Knotens</td>
</tr>
<tr>
<td>NN-Pin</td>
<td>Festpinnen eines Knotens</td>
</tr>
</tbody>
</table>

Eigene Aktionen
Für die Darstellung wird immer ein Symbolbild benötigt

3.5.7.3 Followups
Die Graphansicht reagiert auf folgende Followups:

<table>
<thead>
<tr>
<th>Followup</th>
<th>Data</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph-show</td>
<td>{elementId: [&quot;ID123_456&quot;]}</td>
<td>Zeigt die Elemente im Graph an. Bereits angezeigte Elemente werden ausgeblistet</td>
</tr>
</tbody>
</table>
Beispiel ActionResponse Skript um den Wurzelbegriff der Graphansicht hinzuzufügen:

```javascript
function actionResponse (element, context, actionResult) {
  var actionResponse = new $k.ActionResponse()

  actionResponse.setFollowup('graph-join')
  actionResponse.setData({
    elementId: [$k.rootType().idString()]
  })

  return actionResponse
}
```

### 3.6 Installation

- ViewConfig-Mapper als ZIP-Datei über Static-REST-Ressource bereitstellen
- Verweis auf VCM-Demo mit Bezugsmöglichkeit (Link)
- Über (andere) Web-Server
- Produktivbetrieb/Testbetrieb

### 3.7 Anpassungsprojekt

#### 3.7.1 Entwicklungsumgebung

- Node.js/Webpack/etc.

#### 3.7.2 Technische Details

- Schaubild Informationsfluss bei Aktionen
- Komponenten-State
4 i-views-Dienste

4.1 Allgemeines

4.1.1 Kommandozeilen-Parameter

Falls es zu einem Aufrufparameter auch einen Eintrag der Ini-Datei gibt, hat der der Aufrufparameter höhere Priorität.

-inifile <Dateiname>, -ini < Dateiname >
Name der Ini-Datei, die statt dem Standard-Ini-Datei verwendet wird.

4.1.2 Konfigurationsdatei

Einige Einstellungen können über eine Konfigurationsdatei (*.ini) festgelegt werden. Der Aufbau der Datei sieht folgendermaßen aus:

[Default]
parameterName1=parameterWert1
parameterName2=parameterWert2
...

Im folgenden sind Konfigurationen aufgeführt, die für jeden Dienst verwendet werden können. Für dienstspezifische Einstellungen siehe den Abschnitt "Konfigurationsdatei" des entsprechenden Dienstes.

Logging-Einstellungen

loglevel = <LogLevel>

Konfiguriert, welche Meldungen im Log erscheinen sollen:

- FATAL ERROR: Nur kritische Fehlermeldungen
- ERROR: Nur Fehlermeldungen
- WARNING: Nur Warnungen und Fehlermeldungen
- NORMAL (Standardwert): Alle Meldungen außer Debug-Ausgaben
- NOTIFY: Alle Meldungen inklusive einiger Debug-Ausgaben
- DEBUG: Alle Meldungen inklusive aller Debug-Ausgaben

debug = true/false

Veraltet. Setzt den Log-Level bei true auf DEBUG, bei false auf NORMAL. Wird nur noch ausgewertet, wenn logLevel nicht gesetzt wird

nolog = true/false
Veraltet. Entspricht bei true einem logtargets=null. Wird nur noch ausgewertet, wenn logtargets nicht gesetzt wird

channels = <Channel1> [,<Channel2>,...]


channelLevels = <Channel1>:<Level1> [,<Channel2>:<Level2>,...]
Gezielte Konfiguration des Loglevels für den jeweiligen Channel.

logTargets = <Name1> [,<Name2>,...]
Namen von Log-Targets. Für die Konfiguration siehe Abschnitt „Log-Targets“.

logprefix = <Prefix1> [, <Prefix2>,... ]
Zusätzliche Daten, die bei jeder Log-Ausgabe hinzugefügt werden:

- $pid$: Prozess-ID der Anwendung
- $proc$: ID des aktuellen Smalltalk-Threads
- $alloc$: belegter Speicher der VM (in Megabyte)
- $free$: Freier Speicher der VM (in Megabyte)
- $incGC$: Status inkrementelle GCs
- $os$: Information über OS
- $cmd$: Kommandozeile
- $build$: Build-Version
- $coast$: COAST-Version

Bei einem Präfix, der nicht in dieser Liste enthalten ist, wird der Präfix unverändert ausgegeben.

logTimestampFormat = <FormatString>
Formatierungsangabe für den Timestamp des Log-Eintrags, z.B. "hh:mm:ss".

exceptionLogSize = <Integer>
Setzt die maximale Größe des bei einer Fehlermeldung mitgelieferten StackTrace.

**Log-Targets**

Über Log-Targets lassen sich verschiedene Ziele für das Logging festlegen, für die sich jeweils Log-Level, Channels, Formatierung und mehr konfigurieren lassen. Für jeden angegebenen Namen aus der logtargets-Liste muss eine Konfiguration im Abschnitt [<Konfigurationsname>] angegeben werden:
logTargets=errorausgabe

type=stderr
format=json
loglevel=ERROR


Eine Ausnahme stellt das Log-Target null da: bei einer Konfiguration von logtargets=null muss kein Konfigurationsabschnitt erstellt werden. Fehlt dieser, so ist dies gleichbedeutend mit folgender Konfiguration

logTargets=null

type=null

Es ist jedoch möglich, null als bezeichner für eine beliebige Log-Target Konfiguration zu verwenden.

Grundsätzlich lassen sich genau wie in der allgemeinen Konfiguration loglevel, debug, channels, channelLevels, logprefix und logTimestampFormat festlegen (siehe oben). Die Konfiguration am Log-Target hat immer Vorrang, wenn keine angegeben ist wird jedoch auf die allgemeine Konfiguration zurückgegriffen.

Zusätzlich gibt es noch einige weitere Konfigurationsmöglichkeiten:

format = <Format>

legt das Ausgabeformat fest. Mögliche Werte sind:

- **plain**: Die Standardformatierung in möglichst menschenlesbarer Form
- **json**: einzeilige Ausgabe als JSON-String, vor allem für Maschinenverarbeitung

**type = <Zieltyp>**

legt den Typ der Ausgabe fest. Diese Konfiguration MUSS angegeben werden, sonst wird das Log-Target ignoriert. Im folgenden sind Beschreibung und weitere Konfigurationsmöglichkeit der verschiedenen Typen angegeben:

**file**

Ausgabe in eine Log-Datei.

file = <Dateiname>

legt den Dateinamen der Ziel-Datei fest.

maxLogSize = <size>
Die maximale Größe des Logfiles, ab der die alte Logdatei archiviert wird und eine neue angebrochen wird. Bei Werten kleiner als 1024 wird die Angabe als in MB verstanden.

maxBacklogFiles = <amount>

Die maximale Anzahl an archivierten Logdateien. Beim Anbruch einer neuen wird die letzte gelöscht.

**transcript**

Ausgabe in das Transcript, kann außerdem in eine Log-Datei umgeleitet werden und akzeptiert daher die gleichen Konfigurationen wie **file**.

**stdout**

Ausgabe auf den Standard-Out-Stream.

**stderr**

Ausgabe auf den Standard-Error-Stream.

**mail**

Versendet die Log-Ausgabe per Mail.

```ini
[errorMail]
type = mail
loglevel = ERROR
;Absender-Adresse:
sender = mail@example.org
;Empfänger-Adresse:
recipient = rec@example.org
;Mail-Server:
smtphost = smtp.example.org
;Port des Mail-Servers:
smtport = 465
;Aktiviert bei true die gesicherte Verbindung (TLS/SSL).
;Bei true muss username und password gesetzt sein.
tls = true
username = mail@example.org
password = 12345abc
;Anzahl der Versuche, die Mail bei einem Fehlschlag erneut zu senden:
retries = 3
;Wartezeit zwischen den Versuchen in Sekunden:
retryDelay = 5
```

**mailfile**

Wie mail, allerdings werden Ausgaben mit niedrigem Log-Level zunächst angesammelt und erst per Mail versendet, wenn ein Eintrag mit hohem Level geloggt wird.

mailSendLevel = <LogLevel>

setzt das Log-Level, ab dem die Mail gesendet wird.

**syslog**

Ausgabe als UDP-Datagramm an einen Syslog-Client.
format = <Format>

Anders als bei anderen Log-Targets werden json und plain als Formatierung nicht unterstützt, stattdessen kann hier die Syslog-Version angegeben werden:


facility = <Integer>


targetHostname = <Hostname>

Der Hostname des Zielsystems. Falls nicht angegeben wird localhost verwendet.

targetPort = <Integer>

Der Port, an den gesendet werden soll. Falls nicht angegeben, wird der Syslog-Standard-Port 514 verwendet.

hostname = <Hostname>

Der Hostname des Senders. Falls nicht angegeben, wird des Hostname des Systems ausgelesen.

appname = <Name>

Name der sendenden Anwendung. Falls nicht angegeben, wird der Name der EXE verwendet.

maxMessageSize = <Integer>


null

Zum Unterdrücken der Logausgaben. Es werden keinerlei Optionen ausgelesen.

**4.2 Mediator**

**4.2.1 Allgemeines**

Der i-views-Server sorgt für konsistente und persistente Datenhaltung und für die Aktualität der Daten auf den angeschlossenen i-views-Clients.
Die Datenhaltung erfolgt in einer objektorientierten Datenbank, die durch ein optimistisches Transaktionssystem kooperatives Arbeiten auf dem Wissensnetz ermöglicht.

In seiner Funktion als Kommunikationszentrale sorgt der i-views-Server für die Synchronisation von Clients und Services. Als Basismechanismus stellt er hierfür einen geteilten Objektraum und aktive Updates zur Verfügung.

Technische Daten:

- Konfigurierbarer TCP/IP Server-Port für die Kommunikation mit den Clients, Standard bei i-views 5.1 ist 30063.

Der i-views-Server kann in drei Modi betrieben werden:

1. Klassisch / kompakt: In diesem Modus startet der Server als einzelner Prozess - dem sogenannten "mediator".

4.2.2 Systemvoraussetzungen

Der i-views-Server ist Plattform-unabhängig und läuft auf allen gängigen Betriebssystemen, z. B. Windows, Solaris und Linux (ab Kernel 2.2).

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>Prozessor</th>
<th>Unterstützt</th>
<th>64 Bit VM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2003</td>
<td>IA64</td>
<td>x86 Emulation</td>
<td>nein</td>
</tr>
<tr>
<td>AIX</td>
<td>5.3</td>
<td></td>
<td>ja</td>
<td>nein</td>
</tr>
<tr>
<td>HP-UX</td>
<td>11.x</td>
<td>PA-RISC</td>
<td>ja</td>
<td>nein</td>
</tr>
<tr>
<td></td>
<td>11.x</td>
<td>IA64</td>
<td>nein</td>
<td>nein</td>
</tr>
<tr>
<td>Solaris</td>
<td>8+</td>
<td>Sparc</td>
<td>ja</td>
<td>ja</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>x86</td>
<td>nein</td>
<td>nein</td>
</tr>
</tbody>
</table>
4.2.3 Betriebsmodi


- **stock**
  Startet die Serverkomponente "Stock", die für die persistente Datenhaltung verantwortlich ist.

- **dispatcher**
  Startet die Serverkomponente "Dispatcher", die für die Synchronisation der Clients bzw. für die Verteilung der "active updates" verantwortlich ist.

- **server**
  Startet den vollständigen Server im Multiprozess-Modus.

4.2.3.1 Multi-Prozess Modus (-server)


**Achtung: Es ist wichtig, dass alle Clients (Knowledge-Builder, Bridge, BatchTool etc) Zugriff auf Stock und Dispatcher haben.**

Falls ein dies nur für bestimmte Ports möglich ist, muss eine explizite Konfiguration von Stock und Dispatcher erfolgen. Es werden die gleichen Konfigurations-Dateien im lokalen Verzeichnis verwendet wie im echten verteilten Modus

- **dispatcher.ini** konfiguriert den Dispatcher-Prozess
- **stock.ini** konfiguriert den Stock-Prozess

Es ist aktuell nicht möglich, andere Konfigurations-Dateien zu verwenden.
4.2.3.2 Konfiguration des Stock

Der Stock ist für die Speicherung der Daten auf der Festplatte verantwortlich. Ein einfaches Beispiel für die Konfigurationsdatei `stock.ini` ist

```
[Default]

interfaces=cnp://0.0.0.0:4998
```

Diese Konfiguration sorgt dafür, dass der Stock auf Port 4998 horcht und über das Native Coast-Protokoll kommuniziert.

Die Konfigurations-Datei kann folgende Einträge enthalten:

```
[Default]

parameterName1=parameterWert1
parameterName2=parameterWert2
...
```

Folgende Parameter sind an dieser Stelle einsetzbar:

```
port=<portnummer>
```

Startet den Stock mit der Portnummer `<num>`. Ohne diese Angabe wird Port 30063 verwendet.

Dieser Parameter ist veraltet. Er wird durch den Parameter "interfaces" ersetzt. Dabei entspricht ein Eintrag von "port=1234" dem Eintrag "interfaces=cnp://0.0.0.0:1234". Im Unterschied zum Startparameter sind hier mehrere Werte zulässig, die, durch Komma getrennt, hintereinander aufgezählt werden.

```
interfaces=<interface-1>,<interface-2>,...,<interface-n>
```


Die Protokolle "http" und "https" lassen sich über Proxies umleiten, so dass man den Server bspw. über einen auf Port 443 laufenden IIS erreichen kann.

```
baseDirectory=<Verzeichnis>
```

Setzt das Verzeichnis, in dem sich das Verzeichnis volumes befindet. Sollte dieser Wert auf volumes enden, so wird dieses Verzeichnis direkt verwendet, ohne noch zusätzlich darunter ein Verzeichnis volumes anzugrenzen.

```
volumesDirectory=<Verzeichnis>
```

In diesem Verzeichnis liegen die Wissensnetze. Als Standardwert ist an dieser Stelle 'volumes' eingetragen.

```
backupDirectory=<Verzeichnis>
```

`networkBufferSize=<Größe in Bytes>`


`flushJournalThreshold=<Anzahl der Cluster>`

gibt den Maximalwert an, den "veränderte Cluster" + "Indexcluster" in einem Speichervorgang erreichen dürfen. Wenn der Wert für "veränderte Cluster" bereits überschritten ist, werden keine "Indexcluster" gespeichert sondern diese werden mit Journal geführt.

Ein niedriger Wert (z.B. 50) garantiert schnelle Speicherzeiten, baut aber potentiell ein großes Journal auf.

Ein Wert von "0" deaktiviert Journaling. Standardwert ist "2000". Anmerkung: Ein "flush" des Journals wird immer spätestens bei einer vollständigen Speicherung ausgeführt. Diese wiederum wird ausgelöst, wenn:

- der Mediator beendet wird
- der letzte Client des entsprechenden Volumes abgemeldet wird
- eine Speicherung durch einen full-save-job (siehe jobs.ini) ausgelöst wird

`autoSaveTimeInterval=<Warteintervall in Sekunden>`

gibt an in Sekunden, wie lang nach dem letzten Cluster-Speichern maximal gewartet wird, bis wieder automatisch gespeichert wird. Standardwert ist 15.

`clientTimeout=<Timeout in Sekunden>`

gibt die Zeit in Sekunden an, die ein verbundener Client maximal keine Alive-Nachricht geschickt haben darf, bevor der Mediator ihn als inaktiv erachtet und ihn ausschließt.

`password.flavour=1901332930715229280018647198055591376361`
`password.hash=11199545182458660705495599802052624171734965791427080638694954247035513239844`


den Server veranlassen, beim Start einen neuen Flavour und einen passenden Hashwert zu berechnen und in die ini-Datei zurückzuschreiben. Der Eintrag password.update wird dabei entfernt.

`password=<String>`

Geändert

skipVolumesCheck=<true|false>

gibt an, ob die normalerweise nach dem Start des Mediators durchgeführte Überprüfung der vorhandenen Volumes ausgelassen wird

Geändert

Logging-Einstellungen:
Zu den Konfigurationsmöglichkeiten des Loggings siehe Logging-Einstellungen im Kapitel 11.1.2 Konfigurationsdatei.

Speicher-Einstellungen:
Die folgenden drei Parameter dienen zur Konfiguration der Speicherzuteilung und -nutzung. Erlaubt ist die Angabe von Werten entweder in Megabyte oder in tatsächlichen Byte, wobei die Annahme gilt, dass sich Werte kleiner als 1048576 auf Megabyte-Angaben beziehen.

maxMemory=<Integer, in MB>
Maximal erlaubte Hauptspeicherbelegung. Minimal 50 MB, standardmäßig gesamter physikalisch vorhandener Hauptspeicher (unter Windows) bzw. 512 MB.

baseMemory=<Integer, in MB>
Hauptspeicherbelegung ab der verstärkt versucht wird, Speicher freizugeben. Standardmäßig 0.6 * maxMemory. (alias: "growthRegimeUpperBound")

freeMemoryBound=<Integer, in MB> [10]
Falls belegt, aber nicht mehr benötigter Speicher diese Grenze überschreitet, wird er wieder freigegeben.

BLOB-Service-Konfiguration
Soll der Mediator mit einem integrierten BLOB-Service gestartet werden, damit die BLOBs getrennt von der Datenbank auf der Festplatte gespeichert werden, so muss die folgende Einstellung in der "mediator.ini" eingetragen werden:

startBlobService=true

Nähere Informationen dazu finden Sie in der Dokumentation des BLOB-Service (siehe Link unten).

4.2.3.3 Konfiguration des Dispatchers
Der Dispatcher ist verantwortlich für Transaktionssteuerung und Koordination mehrere Clients. Eine einfacher Konfigurations-Datei ist

[Default]

interfaces=cnp://0.0.0.0:5000
stockAddress=cnp://localhost:4998
stockAuthentication=dsfkhvqw3m9485z432504

Diese Konfiguration öffnet einen Server auf Port 5000 zu dem sich Clients verbinden können. Den Stock sucht der Dispatcher unter localhost:4998. Diese Adresse ist auch die Adresse, die von den Clients verwendet wird um Daten vom Stock zu
Falls Dispatcher und Stock auf dem gleichen Server laufen, teil der Dispatcher seinen Clients eigenen Hostname mit, damit auch Verbindungen über das Netzwerk funktionieren.
Für die Authentifizierung des Dispatchers beim Stock wird das Token dsfkhvqw3n9485z432504 verwendet. Dieses Token muss in der Stock-Konfiguration über die "password."-Schlüssel eingestellt sein.

4.2.4 Installation

Der i-views-Server benötigt prinzipiell keine spezielle Installation, d.h. er ist ad hoc aus einem beliebigen Verzeichnis startbar.
Es ist dabei darauf zu achten, dass die notwendigen Zugriffsrechte (lesen/schreiben/erzeugen) für das Arbeitsverzeichnis des Servers und alle Unterverzeichnisse gesetzt sind.

4.2.4.1 Startparameter

Die komplette Liste der möglichen Startparameter gibt der Mediator beim Aufruf mit dem Parameter ".?" aus.

- interface <interface-1>

Die Protokolle "http" und "https" lassen sich über Proxies umleiten, so dass man den Server bspw. über einen auf Port 443 laufenden IIS erreichen kann.

- clientTimeout <sec>

Setzt die Zeit, innerhalb der sich ein Client automatisch melden muss, auf <sec> Sekunden. Der Wert sollte mindestens auf 600 gesetzt werden (was auch der Standardwert ist).
-baseDirectory <directory>

Setzt das Verzeichnis, in dem sich das Verzeichnis "volumes" befindet. Neben dem Unter-
verzeichnis "volumes" werden hier auch standardmäßig die Verzeichnisse für Backups und
Downloads angelegt. Dieser Parameter hieß früher "-volumes".

Die folgenden Parameter stellen Kommandos an das Mediator Executable, um bestimmte
Aufgaben auszuführen, ohne danach als Server für Wissensnetze zu fungieren.

-quickRecover <volume> -recover <volume>

Falls der Mediator unordnungsgemäß beendet wird (z.B. Absturz des Rechners), bleiben in
Volumes, die in Benutzung waren, Lock-Dateien stehen. Das Volume kann dann nicht mehr
betreten werden. Um das Lock aufzuheben, kann man mit dem Aufruf von -quickRecover
<volume> das Lock entfernen. Der Aufruf schlägt fehl, wenn (mögliche) Inkonsistenzen ge-
fundenen wurden. In diesem Fall muss der Startparameter -recover verwendet werden.

Achtung:

Das Arbeitsverzeichnis beim Aufruf muss hier das Verzeichnis sein, welches das „volumes“-
Verzeichnis enthält. Der „-volumes“-Parameter wirkt hier also nicht.

-bfscommand <volume> <command>

Führt Kommandos aus, die vom BlockFileSystem erkannt werden.

Kommandozeilen-Parameter für das Logging:

-nolog

Schaltet Logging ab

-loglevel <Integer>

Konfiguriert, welche Meldungen im Log erscheinen sollen:

- 0: Alle Meldungen inklusive Debug-Ausgaben
- 10 (Standardwert): Alle Meldungen außer Debug-Ausgaben
- 20: Nur Warnungen und Fehlermeldungen
- 30: Nur Fehlermeldungen

-logfile <Dateiname>, -log <Dateiname>

Name der Log-Datei, die statt der Standard-Log-Datei verwendet wird. Dieser Parameter
muss auf jeden Fall verändert werden, wenn mehrere Clients im selben Arbeitsverzeichnis
gestartet werden sind.

-debug

Schaltet das Logging auf debug-mode

-log <logname>

Setzt die Logdatei auf <logname>.

4.2.4.2 Konfigurationsdatei "mediator.ini"

Einige Mediator-Einstellungen können auch in der Konfigurationsdatei mediator.ini fest-
gelegt werden. Der Aufbau der Datei sieht folgendermaßen aus:
Folgende Parameter sind an dieser Stelle einsetzbar:

**Netzwerkkommunikation**

`port=<portnummer>`


Dieser Parameter ist veraltet. Er wird durch den Parameter "interfaces" ersetzt. Dabei entspricht ein Eintrag von "port=1234" dem Eintrag "interfaces=cnp://0.0.0.0:1234". Im Unterschied zum Startparameter sind hier mehrere Werte zulässig, die, durch Komma getrennt, hintereinander aufgezählt werden.

`interfaces=<interface-1>,<interface-2>,...<interface-n>`


Die Protokolle "http" und "https" lassen sich über Proxies umleiten, so dass man den Server bspw. über einen auf Port 443 laufenden IIS erreichen kann.

Für ssl-Kommunikation (cnps:// bzw. https://) müssen in der Konfigurationsdatei zusätzlich die Dateipfade für Zertifikat und Private Key angegeben werden:

`certificate=Name der .crt-Datei privateKey=Name der .key-Datei`

**Verzeichnisse**

`baseDirectory=<Verzeichnis>`

Setzt das Verzeichnis, in dem sich das Verzeichnis volumes befindet. Sollte dieser Wert auf volumes enden, so wird dieses Verzeichnis direkt verwendet, ohne noch zusätzlich darunter ein Verzeichnis volumes anzulegen.

`volumesDirectory=<Verzeichnis>`

In diesem Verzeichnis liegen die Wissensnetze. Als Standardwert ist an dieser Stelle ‘volumes’ eingetragen.

`backupDirectory=<Verzeichnis>`


`networkBufferSize=<Größe in Bytes>`

Gibt die Größe des Puffers an, der für das Senden/Empfangen von Daten verwendet wird. Der Standardwert ist 20480. In manchen Infrastrukturen kann man durch Angabe von `networkBufferSize=4096`
einen höheren Durchsatz erreichen.

\texttt{journalMaxSize=Maximale\ Grö\ße\ des\ Journals} \\

Mit \texttt{journalMaxSize=0} kann man das normalerweise aktive Journalling deaktivieren. Der Standardwert ist 5242880 (5 MB).

\texttt{autoSaveTimetInteral=Warteintervall\ in\ Sekunden} \\

gibt an in Sekunden, wie lang nach dem letzten Cluster-Speichern maximal gewartet wird, bis wieder automatisch gespeichert wird. Standardwert ist 15.

\texttt{clientTimeout=Timeout\ in\ Sekunden} \\

gibt die Zeit in Sekunden an, die ein verbundener Client maximal keine Alive-Nachricht geschickt haben darf, bevor der Mediator ihn als inaktiv erachtet und ihn ausschließt.

\texttt{password.flavour=0\ Flavor\ der\ Passwörter} \\
\texttt{password.hash=0\ Hash\ des\ Passwörter} \\

Das Mediator-Passwort wird zusammen mit einem zufälligem flavour zu einen (SHA256) hashwert berechnet. Diese beiden Informationen genügen dann, um dem Mediator eine Authentifizierungsanfrage zu überprüfen. Beim Authentifizieren am Server muss der Benutzername mit "Server:admin" angegeben werden. Um diese Werte zu ermitteln kann man mit \texttt{password.update=neues\_passwort} \\

den Server veranlassen, beim Start einen neuen Flavour und einen passenden Hashwert zu berechnen und in die ini-Datei zurückzuschreiben. Der Eintrag password.update wird dabei entfernt.

\texttt{password=String} \\


Geändert \texttt{skipVolumesCheck=true|false} \\

gibt an, ob die normalerweise nach dem Start des Mediators durchgeführte Überprüfung der vorhandenen Volumes ausgelassen wird

**Logging**

Zu den Konfigurationsmöglichkeiten des Loggings siehe Logging-Einstellungen im Kapitel 11.1.2 Konfigurationsdatei.

**Arbeitsspeicher**

Die folgenden drei Parameter dienen zur Konfiguration der Speicherverteilung und -nutzung. Erlaubt ist die Angabe von Werten entweder in Megabyte oder in tatsächlichen Byte, wobei die Annahme gilt, dass sich Werte kleiner als 1048576 auf Megabyte-Angaben beziehen.

\texttt{maxMemory=Integer,\ in\ MB} \\

Maximal erlaubte Hauptspeicherbelegung. Minimal 50 MB, standardmäßig gesamter physikalisch vorhandener Hauptspeicher (unter Windows) bzw. 512 MB.

\texttt{baseMemory=Integer,\ in\ MB} \\

Hauptspeicherbelegung ab der verstärkt versucht wird, Speicher freizugeben. Standard-
mäßige 0.6 \times \text{maxMemory}. (alias: "growthRegimeUpperBound")

\text{freeMemoryBound} = \langle \text{Integer, in MB} \rangle [10]

Falls belegt, aber nicht mehr benötigter Speicher diese Grenze überschreitet, wird er wieder freigegeben.

**BLOB-Service-Konfiguration**

Soll der Mediator mit einem integrierten BLOB-Service gestartet werden, damit die BLOBs getrennt von der Datenbank auf der Festplatte gespeichert werden, so muss die folgende Einstellung in der "mediator.ini" eingetragen werden:

\begin{verbatim}
startBlobService=true
\end{verbatim}

Nähere Informationen dazu finden Sie in der Dokumentation des BLOB-Service (siehe Link unten).

### 4.2.4.3 Sicherheitskonzept des Mediators

Der i-views-Server ist eine generische Komponente, die nicht nur für i-views verwendet werden kann. Neben der Einschränkungen über die Authentifizierungen am Server oder in der Datenbank kann man auch kontrollieren, welche Anwendungen sich verbinden dürfen.

Jede Anwendung (Client und Server) enthält ein RSA-Schlüsselpaar, das je ausgelieferter Anwendung eindeutig ist. Den öffentlichen Schlüssel kann man über die Information erhalten (KB: Menü „Werkzeuge“, „Info“, dann die Schaltfläche „RSA-Key kopieren“) bzw. für Konsolen-Anwendungen per Aufruf mit dem Parameter -showBuildID. Die hierdurch exportierte Build-Information enthält den öffentlichen RSA-Exponenten (rsa.e_1) und RSA-Modul (aufgeteilt auf mehrere rsa.n_x) sowie eine MD5 Prüfsumme dieser Informationen (buildID).

Beispiel einer Build-Information:

\begin{verbatim}
[buildID.90A1203EFB957A58C2268AD8FE3CC5A3] rsa.n_1=93D516DF61395258AA21A91B33E8EE67 rsa.n_2=B07C6...
\end{verbatim}

Möchte man nun, dass sich nur eine bestimmte Menge Client-Anwendungen mit dem Server verbinden kann, so muss man im Server die jeweiligen Abschnitte in die mediator.ini übertragen. Beim Verbindungsaufbau überträgt der Client seine buildID. Wenn der Mediator einen passenden Eintrag enthält, so wird er die Client-Authentizität prüfen. Andernfalls wird er eine Verbindung nur aufbauen, wenn es gar keine Einträge zu Build-Informationen in seiner Ini-Datei gibt. Somit kann beispielsweise verhindert werden, dass sich veraltete Client-Anwendungen oder modifizierte Client-Anwendungen mit dem Mediator verbinden.

Umgekehrt können auch in der Client-Anwendung entsprechende buildIDs für die Mediatoren in die jeweilige ini-Datei eingetragen werden, um eine Verbindung zu einem kompromittierten oder veralteten Server zu verhindern.

So kann man eine Umgebung einrichten, in der nur mit der aktuellsten Software auf die Produktivdaten zugreifen werden kann, aber auf die Server mit den Testdaten auch von einer Entwicklungsumgebung aus. Die Anwendersoftware wiederum kann nur auf den Produktservor oder auf den Testserver zugreifen.

Konfiguriert man weder Server noch Client, so verhält sich die Installation wie in den Vorgängerversionen: Jede Anwendung kann sich mit jedem Server verbinden (sofern die Protokollversion übereinstimmt).

Seit der Version 5.4 des Servers benötigt man zum Durchführen administrativer Befehle das
Server-Passwort als Parameter (über die Rest-Schnittstelle oder über die Verwaltung per Administrationswerkzeug). Für Aktionen, die sich auf eine existierende Datenbank beziehen (backup, download, garbage collection usw) genügt hierfür seit Version 6.2 eine Authen-
tifizierung als Administrator im Volume.

Umgekehrt ist es mit dem Serverpasswort möglich, sich in einem Volume anzumelden. De-
tails hierzu finden sich im Admin-Tool.

Ist am Server kein Passwort konfiguriert, so kann man sich mit einem beliebigen Passwort
am Server anmelden. Die Anmeldung im Volume ist dann jedoch nicht möglich.

4.2.4.4 Audit-Log konfigurieren

In einigen Anwendungsszenarien kann es gefordert sein, alle Zugriffe auf ein Wissensnetz
in einem Zugriffs- oder Audit-Log zu protokollieren. Dieses Audit-Log enthält Einträge für
alle An- und Abmeldevorgänge, schreibende und lesende Zugriffe auf Wissensnetz-Inhalte,
gestellte Suchanfragen, Ausdrucke, Exporte etc.

Damit das Zugriffslog aktiviert werden kann, muss eine Konfigurationsdatei 'log.ini' im Verzei-
chnis des zu überwachenden Wissensnetzes angelegt werden:

```
[Default]
applicationLog=CoastJSONApplicationLogger

[CoastJSONApplicationLogger]
backupInterval=14
maxLogSize=5
```

Zusätzlich muss das Log im Admin-Tool in der Rubrik 'Systemkonfiguration / Audit-Log' ak-
tiviert werden. Die Aktivierung bzw. Deaktivierung des Logs resultiert wiederum in einem
Eintrag im Audit-Log.

Für die Auswertung der Zugriffslogs steht Administratoren eine Funktion im Administrations-
Menue des Knowledge-Builders zur Verfügung.

4.2.5 Betrieb

4.2.5.1 Herunterfahren des Servers

Der i-views-Server lässt sich lokal durch das Strg-C Abbruchsignal herunterfahren.

Bei der Installation als Windows-Dienst muss der Server mit der Diensteverwaltung gestoppt
werden.

Unter UNIX sowie beim Betrieb als Windows-Dienst, wird der Server beim Herunterfahren
des Betriebssystems ordnungsgemäß beendet.

4.2.5.2 Speicherung und Backup von Wissensnetzen

Verzeichnisstruktur

Das Basisverzeichnis des i-views-Servers weist folgende Struktur auf:
volumes/
    wissensnetzName/
        wissensnetzName.cbf
        wissensnetzName.cdr
        wissensnetzName.cfl
        wissensnetzName.lock (wenn das Wissensnetz geöffnet ist)

backup/
    wissensnetzName/
        <zehnstellige Nummer>/
            wissensnetzName.cbf
            wissensnetzName.cdr
            wissensnetzName.cfl

Speicherung von Wissensnetzen


Backup von Wissensnetzen


Der Ort, an dem die Backups angelegt werden kann über den Eintrag

backupDirectory=<Verzeichnis>

in der Datei "mediator.ini" festgelegt werden. Ohne diese Angabe wird das Unterverzeichnis "backup" des Basisverzeichnisses verwendet.

Der Backup-Service des K-Infinity-Servers kann auf zwei Arten angestoßen werden:

1. Durch einen direkten Request an den Serverprozess (z.B. vom Administrationstool aus)
2. Durch Einträge in der Datei jobs.ini im Arbeitsverzeichnis des Servers. Diese Datei kann pro Wissensnetz eine Rubrik [Name_des_Netzes] mit folgenden Einträgen enthalten:

Beispiel jobs.ini

[volume1]
;Backup des Netzes "volume1"

;Uhrzeit, zu dem das Backup gestartet wird
backupTime=00:45

;Turnus in Tagen - hier täglich
backupInterval=1

;Die letzten 5 Backups dieses Wissensnetzes aufheben
backupsToKeep=5
'backupsToKeep' gibt die Anzahl der aufzuhebenden Backups an. Dies beinhaltet auch Backups, die manuell erstellt worden. Der Standardwert ist 3.

Bei der Angabe der Netznamen in eckigen Klammern ist die Verwendung der Platzhalter "*" und "?" erlaubt, Groß- und Kleinschrift wird ignoriert.

4.2.5.3 Garbage Collection

Ohne Garbage Collection wächst das Wissensnetz kontinuierlich mit der der Verwendung. Folglich ist es sinnvoll, von Zeit zu Zeit eine Bereinigung (Garbage Collection) durchzuführen. Wie die Datensicherung kann die Garbage Collection jederzeit manuell (z.B. mit einem speziellen Administrationswerkzeug) oder automatisch gestartet werden.


Automatische Garbage Collection: Aufbau der Datei jobs.ini

Die automatische Garbage Collection wird durch einen Eintrag in der Datei 'jobs.ini' konfiguriert, z.B.

[volume1] garbageCollectTime=00:55 garbageCollectInterval=7

Dieser Eintrag in der jobs.ini sorgt dafür, dass das Netz mit Namen "volume1" im Abstand von "7" Tagen jeweils um "00:55" Uhr garbage-collected wird. Für das Intervall ist der Standardwert "1" (also täglich), der Zeitpunkt muss angegeben werden.

Beider Angabe der Netznamen in eckigen Klammern ist die Verwendung der Platzhalter "*" und "?" erlaubt, Groß- und Kleinschrift wird ignoriert.

Manueller Start der Garbage Collection

Alternativ kann die Garbage Collection auch durch spezielle Aufrufparameter des i-views-Servers gesteuert werden:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>-infoGC &lt;volume&gt; -host &lt;hostname&gt;</td>
<td>Informiert über den aktuellen Stand der Garbage Collection.</td>
</tr>
</tbody>
</table>
Diese Kommandos werden mit Hilfe eines Mediator-Executables an einen anderen bereits laufenden Mediator übermittelt. 

Als weitere Möglichkeit bietet sich das Starten der Garbage Collection über das Admin-Tool an. 

Zum Ausführen dieser Befehle muss mittels Parameter -password das richtige Serverpasswort übermittelt werden. 

### 4.2.5.4 Betrieb unter Unix 

Unter UNIX reagiert der Server auf folgende Signale:

- SIGTERM/SIGHUP
  - Beendet den Server

- SIGUSR2
  - Der Server startet einen sofortigen Backup aller Wissensnetze, die in der jobs.ini-Datei für Backup spezifiziert sind (siehe auch Abschnitt über Backup). 

### 4.2.5.5 Betrieb im Cluster 


Bei Ausfall des ersten Mediators kann es passieren, dass der Mediator keine Zeit mehr hat, das Wissensnetz in einen konsistenten Zustand zu bringen, das Netz damit eine Inkonsistenz aufweist und die "lock"-Datei des alten Mediators noch im entsprechenden Verzeichnis bestehen bleibt. Damit der neue Mediator in der Lage ist, die "lock"-Datei zu löschen, muss folgender Parameter in die mediator.ini hinzugefügt werden.

```ini
host=NameDesClusters
```

In diesem Fall können alle Mediatoren mit diesem ini-Eintrag auch gesperrte Volumes anderer Mediatoren, die beim Start den selben Wert in der mediator.ini ausgelesen hatten, entsperren. "NameDesClusters" ist frei wählbar, muss aber den Regeln entsprechen, die für Hostnamen gelten (keine Leerzeichen, Doppelpunkte, o.ä.)


### 4.2.5.6 Problembehebung 


Falls der Mediator eine Inkonsistenz erkennt, kann in der Kommandozeile durch den Aufruf
des Mediators mit den Parametern -quickRecover / -recover das Wissensnetz geprüft und
Inkonsistenzen soweit möglich repariert werden.
Sollte eine Auflösung der Inkonsistenzen wider Erwarten nicht möglich sein, muss auf eine
Sicherungskopie zurückgegriffen werden.

4.2.5.7 Kommandos des BlockFileSystems

Die Befehle hinter -bfscommand ermögliche Oerationen auf dem BlockFileSystem und sind
für Supportfälle vorgesehen. Ein solcher Befehl könnte zBsp so aussehen:

-bfscommand quickCheck {target volume}

Die mit {target volume} adressierte Datenbank wird einer schnellen Strukturanalyse unter-
zogen. Analog kann mit deepCheck eine komplette Analyse ausgeführt werden.

4.3 Bridge

4.3.1 Allgemeines

Die Bridge ermöglicht den externen Zugang zu Wissensnetzenn auf drei Arten/Betriebsmodi:

- Über eine RESTful Services-Architektur (REST-Bridge). Die Schnittstelle steht als HTTP-
oder HTTPS-Version zur Verfügung (KHTTPRestBridge)
- Über KEM-RPC (KEMBridge): Zugang über KEM. Falls Binärdaten im Wissensnetz gespe-
ichert werden ist zusätzlich eine REST-Bridge erforderlich, die einen REST-Service mit
einem Blob-Resource-Handler bereitstellt.
- Betriebsmodus "Lastverteiler für andere Bridges" (KLoadBalancer).

ACHTUNG: KLoadBalancer und KEMBridge/KHTTPRestBridge dürfen nicht gleichzeitig in ein-
er Bridge aktiviert werden, weil diese sich gegenseitig behindern.

Die Bridge und alle in ihr zu aktivierenden Zugänge lassen sich über eine ini-Datei konfig-
urieren. Einstellungen für die Zugänge sind dabei in Abschnitten gebündelt. Die wichtigsten
dieser Parameter lassen sich aber auch über die Kommandozeile spezifizieren. Ist dies der
Fall, so haben die Werte des Kommandozeilenaufrufs Vorrang vor denen in der ini-Datei. Die
einzelnen Parameter werden nun erläutert.

4.3.2 Gemeinsame Kommandozeilen-Parameter

Wird die Bridge ohne jegliche Parameter gestartet, so werden die erforderlichen Param-
eter aus der Ini-Datei bridge.ini gelesen und die Fehlermeldungen in die Datei bridge.log
geschrieben.

Falls es zu einem Aufrufparameter auch einen Eintrag der Ini-Datei gibt, hat der der Aufruf-
parameter höhere Priorität.

-inifile <Dateiname>, -ini < Dateiname >
Name der Ini-Datei, die statt dem Standard-Ini-Datei verwendet wird. Standard ist bridge.ini

-host <hostname:port>, -hostname <hostname:port>

Name des Mediators, der als Datenserver fungiert. Dieser gilt für alle aktivierten Bridgeclients

-port |<ClientName> |<portnumber>

Der Parameter -port ist eigentlich für jeden Klienten in der ini-Datei zu setzen. Will man dieses aber bereits in der Komandozeile tun, so lassen sich die unterschiedlichen Klienten durch Voranstellen des Klientennamens vor die Portnummer spezifizieren. Die obige Zeile gilt für einen Klienten, entsprechend muss der Parameter -port wiederholt werden, sollen mehrere Klienten konfiguriert werden.

Beispiele für den Aufruf der Bridge:

bridge -host server01:30000 -port KEMBridge 4713 -port KEMStreamingBridge 4714
bridge -ini bridge2.ini -port KMultBridge 3030

Kommandozeilen-Parameter für das Logging:

-nolog
Schaltet Logging ab
-loglevel <Integer>
Konfiguriert, welche Meldungen im Log erscheinen sollen:

• 0: Alle Meldungen inklusive Debug-Ausgaben
• 10 (Standardwert): Alle Meldungen außer Debug-Ausgaben
• 20: Nur Warnungen und Fehlermeldungen
• 30: Nur Fehlermeldungen

-logfile <Dateiname>, -log <Dateiname>
Name der Log-Datei, die statt der Standard-Log-Datei verwendet wird. Dieser Parameter muss auf jeden Fall verändert werden, wenn mehrere Clients im selben Arbeitsverzeichnis gestartet werden sind.

-debug
Schaltet das Logging auf debug-mode
-log <logname>
Setzt die Logdatei auf <logname>.

-stop <hostname>
Ruft man die Bridge mit dem obigen Parameter auf, so wird die auf dem angegebenen Host laufende Bridge zum Beenden aufgefordert. Alle in ihr gestarteten Klienten werden heruntergefahren und die Bridge beendet.
4.3.3 Konfigurationsdatei "bridge.ini"

Alle der folgenden Einträge befinden sich unterhalb des ini-Datei-Abschnitts [Default]. Die Einträge für die einzelnen Klienten schließen daran an. Durch das Einfügen klientspezifischer Konfigurationsabschnitte wird zusätzlich definiert, welche Klienten in der zu konfigurierenden und zu startenden Bridge aktiviert sind. Im Moment mögliche Klienten sind dabei:

- KEMBridge
- KHTTPRestBridge

Zusätzlich kann noch der KLoadBalancer als Klient der Bridge gestartet werden, dann enthält die ini-Datei nur den Abschnitt

- KLoadBalancer

host = <hostname:portnumber>
siehe Kommandozeilenparameter -host

**Speicher-Einstellungen:**

Die folgenden drei Parameter dienen zur Konfiguration der Speicherzuteilung und -nutzung. Erlaubt ist die Angabe von Werten entweder in Megabyte oder in tatsächlichen Byte, wobei die Annahme gilt, dass sich Werte kleiner als 1048576 auf Megabyte-Angaben beziehen.

maxMemory=<Integer, in MB>
Maximal erlaubte Hauptspeicherbelegung. Minimal 50 MB, standardmäßig gesamter physikalisch vorhandener Hauptspeicher (unter Windows) bzw. 512 MB.

baseMemory=<Integer, in MB>
Hauptspeicherbelegung ab der verstärkt versucht wird, Speicher freizugeben. Standardmäßig 0.6 * maxMemory. (alias: "growthRegimeUpperBound")

freeMemoryBound=<Integer, in MB> [10]
Falls belegt, aber nicht mehr benötigter Speicher diese Grenze überschreitet, wird er wieder freigegeben.

minAge=<Integer> [30]

unloadInterval=<Integer> [10]
Mindestdauer (in Sekunden) zwischen zwei Cluster-Auszagerungen

unloadSize=<Integer> [4000]
Mindestanzahl an geladenen Cluster, ab der ausgelagert wird

keepSize=<Integer> [3500]
Zahl der Cluster, die beim Auslagern behalten werden

useProxyValueHolder=true/false
Um den Mediator bei Suchen zu entlasten, kann die Option useProxyValueHolder=false verwendet werden. Der Client lädt dann Indizes in den Hauptspeicher, statt per RPCs den Mediator abzufragen. Der Nachteil dieser Option ist, dass dann nur noch lesender Zugriff möglich ist.

loadIndexes=true/false


**Logging-Einstellungen:**

Zu den Konfigurationsmöglichkeiten des Loggings siehe Logging-Einstellungen im Kapitel 11.1.2 Konfigurationsdatei.

### 4.3.4 REST-Bridge

#### 4.3.4.1 Einführung

Die REST-Bridge Application ermöglicht den lesenden und schreibenden Zugriff auf i-views über eine RESTful Services-Architektur. Die Schnittstelle steht als HTTP- oder HTTPS-Version zur Verfügung.

Die REST-Bridge läuft innerhalb der Standard-Bridge von i-views (bridge.exe).

Die Schnittstelle wird vollständig durch Konfigurations-Individuen im Wissensnetz konfiguriert. Der Rückgabewert eines REST-Aufrufs ist eine beliebige Zeichenkette, in der Regel in einem Format, das der aufrufende Client gut weiterverarbeiten kann (z.B. XML oder JSON).

#### 4.3.4.2 Installation

##### 4.3.4.2.1 Volume vorbereiten

Durch das Hinzufügen der Softwarekomponente "KRestServiceComponent" im Admin-Tool wird im Wissensnetz das benötigte Schema angelegt.
Das Schema wird in einem Teilnetz des Wissensnetzes namens "Servicekonfiguration" angelegt:
4.3.4.2.2 Bridge konfigurieren

Die REST-Schnittstelle wird durch die Standard-Bridge-Komponente von i-views bereitgestellt, sofern in der zugehörigen Konfigurationsdatei bridge.ini eine Kategorie HTTPRestBridge bzw. HTTPSRestBridge eingetragen ist:

[KHTTPRestBridge] volume = name des Wissensnetzes port = port, unter dem der Service erreichbar sein soll

Für die HTTPS-Version müssen in der Konfigurationsdatei zusätzlich die Dateipfade für Zertifikat und Private Key angegeben werden:

[KHTTPSRestBridge] volume = name des Wissensnetzes port = port, unter dem der Service erreichbar sein soll

Im Konfigurationsabschnitt "KHTTPRestBridge" oder "KHTTPSRestBridge" können außerdem noch die folgenden speziellen Konfigurationsoptionen eingetragen werden:

<table>
<thead>
<tr>
<th>Name</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>echo</td>
</tr>
<tr>
<td>PathPattern</td>
<td>echo</td>
</tr>
<tr>
<td>Report GET</td>
<td>&lt;Script&gt; &lt;Output&gt; &lt;path path=1&gt;</td>
</tr>
<tr>
<td>Response content type</td>
<td>application/text</td>
</tr>
</tbody>
</table>
**realm**

Name, der bei aktivierter Authentifizierung als Realm-Name an den Client zurückgegeben wird. Web-Browser zeigen den Realm-Namen typischerweise in Dialogfenstern zur Authentifizierung als Applikationsnamen an, damit der Benutzer weiß, wer die Authentifizierung fordert. Standardwert: REST

---

### 4.3.5 KEM-Bridge

**KEMBridge**

Abschnittsname:

[KEMBridge]

port = <portnumber>

Angabe des Ports, unter welchem die KEMBridge reagiert. Bei Nichtangabe gilt der Defaultwert von 4713.

ldapHost = <hostname:portnumber>

Angabe des LDAP-Hostes, welcher kontaktiert werden soll, für den Fall, dass die Authentifizierung über LDAP stattfinden soll. Ist dieser Parameter angegeben, so muss die Authentifizierung über LDAP abgewickelt werden.

maxLoginCount = <number>


Um ein Sperren des Benutzers im Wissensnetz zu ermöglichen, muss für Individuen des Personenkonzepts ein boolesches Attribut mit internen Namen userlock und Defaultwert false definiert sein.

KEMrestrictToIP=Address = <IP-Adresse>

Wenn dieser Parameter gesetzt ist, werden nur Verbindungen von dem hier angegebenen Host akzeptiert.

trustedLoginEnabled = <true/false>

Erlaubt ein Einloggen ohne Passwort mittels des Requests: „newAuthenticatedUser(username)“. preventSessionReplay=<true/false>

[default=false]

Dieser Parameter gibt an, dass jede schreibende Session ihren eigenen geschützten Wissensnetzzugriff erhält, so dass der sonst übliche Mechanismus, die Aktionen einer deaktivierte Session beim Reaktivieren erneut auszuführen, um den letzten aktuellen Editorstand zu erhalten, unnötig wird.
KEMStreamingBridge

Abschnittsname:
[KEMStreamingBridge]
port = <portnumber>

Angabe des Ports, unter welchem die KEMStreamingBridge reagiert. Bei Nichtangabe gilt der Defaultwert von 4714.

4.3.6 KLoadBalancer

Der KLoadBalancer kann eingesetzt werden, um die Services und Verfügbarkeit der KEM-Bridge und KEMStreamingBridge zu skalieren.

Im Abschnitt [KLoadBalancer] können/müssen die folgenden Angaben gemacht werden, um den gewünschten Betriebsmodus zu erreichen:

- allowRemoteShutdown (Default-Wert false)
- autoRestart (Default-Wert true)
- directory (Default-Wert aktuelles Arbeitsverzeichnis, in dem der KLoadBalancer gestartet wurde)
- executable (Default-Wert 'bridge.exe')
- image (Default-Wert 'bridge.im')
- vm (Default-Wert 'visual')
- hostname (Default-Wert Localhost)
- configNames (benötigter Wert, nicht optional)
- parameters (Default-Wert leer)

Der Parameter #configNames dient der weiteren Konfiguration der zu startenden KEM-Bridges und KEMStreamingBridges, je Einzel-Konfiguration wird ein Bridge-Typ gesteuert. Die Konfigurationsnamen sind durch Komma zu trennen.

Hier ein Beispiel für eine KLoadBalancer-ini-Datei:

[Default] [KLoadBalancer] hostname=ws01 port=30003 directory=C:\3.2\balancing executable=bridge.exe ;vm=visual ;image=bridge.im configNames=KEM,Streaming

Beim Start werden gemäß der beiden Konfigurationen KEMBridges und KEMStreaming-Bridges gestartet. Da zum Betrieb dieselbe Software wie für den Betrieb des KLoadBalancers verwendet wird, sind in diesem Abschnitt die Angabe der Parameter #executable, #image und #vm (für Linux-Betrieb), #hostname, #directory und #parameters nötig.

eexecutable / image, vm; directory: Angaben, wie die einzelnen Bridges gestartet werden können. Unter Windows wird die Angabe von #executable und #directory benötigt, unter Linux die Angabe von #image, #vm und #directory.

hostname / port: Der Hostname, der den zu startenden Bridges als für Verwaltungszwecke zu kontaktierender KLoadBalancer genannt wird. Falls hier keine Angabe gemacht wird, wird der Rechnername ermittelt und dieser verwendet. Der Port gibt an, unter welchem Port die Bridges den Balancer ansprechen, Default-Wert ist 4715.

Vorsicht: Der Name des jeweiligen Mediators, den die Bridges zum Abrufen von Daten kontaktieren, ist in den jeweiligen ini-Dateien gemäß Konfigurationsabschnitt einzutragen!
parameters: Ein Feld, mit dem zusätzliche Angaben in die Kommandozeile der zu startenden Bridges eingefügt werden können, ist für alle zu startenden Bridges gleich.

allowRemoteShutdown: Parameter, der angibt, ob der KLoadBalancer per shutdown-Request per remote-Zugriff zu beenden ist.

autoRestart: Parameter, der angibt, ob eine gestoppte KEMBridge nach dem shutdown erneut zu starten ist, mit neuer ID.

In jedem Konfigurationsabschnitt müssen zusätzliche Angaben gemacht werden:

- bridgeClientClassName (nicht optional, nur eine Angabe je Abschnitt möglich. Bitte obige Schreibweise beachten!)
- inifile (ini-Datei mit Einstellungen für diesen Typ zu startende Bridge)
- bridgeLogFile (Muster eines Logfile-Namens, in den ein Platzhalter eingefügt wird, <id>, über den sich die Log-Dateien der einzelnen Bridges auseinanderhalten lassen; wird mit der laufenden Nummer der gestarteten Bridge ersetzt)
- maxBridges (Anzahl der maximal zu startenden Bridges des angegebenen Typs, nicht optional!)
- sslEnabled (Angabe, ob die Bridges dieses Typs SSL für den Verbindungsaufbau verwenden sollen, Default-Wert false)


Die ini-Dateien der jeweiligen Bridges müssen wie gewohnt aufgebaut werden. Ein Beispiel für die im obigen Konfigurationsabschnitt KEM referenzierte ini-Datei ist hier angefügt:

[Default] host=mediator-hostname:30053 [KEMBridge] trustedLoginEnabled=true preventSessionReplay=true

Für Details sei auf Kapitel 5 "Konfigurationsdatei bridge.ini" verwiesen.

4.4 Jobclient

4.4.1 Allgemeines

Der Job-Client erbringt zum einen Dienste für andere i-views-Clients, um diese von rechenzeit- oder datenintensiven Aufgaben zu entlasten. Zum anderen dient er als Brücke zwischen i- views-Clients und externen Systemen.

Zu seinen wichtigsten Aufgaben gehört die Ausführung aller Arten von Suchen sowie die Auslieferung der Suchergebnisse an die Clients (Sortierung, textuelle Aufbereitung, Rechtefilterung).

Im Normalfall wartet der Client auf die Fertigstellung eines Auftrags (Synchronbetrieb).

Für die Ausführung komplexer Suchen, das Erstellen von Statistiken, Batch-Abgleiche, Datei- aufbereitungen, Datenbereinigungen, etc. muss der Client nicht auf die Fertigstellung warten (Asynchronbetrieb). Das Ergebnis wird vom Service bereitgestellt und der Client wird benachrichtigt. Das Ergebnis kann dann beliebige Zeit später eingesehen werden. Da das Ergeb- nis auch persistent gemacht wird, ist es auch nach einem Neustart des Systems bzw. im Falle eines Fail-Overs weiterhin verfügbar.

Funktionsweise:
In dem vom i-views-Mediator bereitgestellten geteilten Objektraum werden die Aufträge der

Technische Daten:
Multi-Platform Executable auf Basis der VisualWorks Smalltalk Virtual Machine (jobclient.exe bzw. jobclient.im)
Benötigt eine TCP/IP-Verbindung zum i-views-Server
Automatische Lastverteilung zwischen den Services
Job-Clients können zu jeder Zeit zugeschaltet oder heruntergefahren werden
Standby-Modus bei zeitweiliger Nicht-Verfügbarkeit benötigter Ressourcen

4.4.2 Konfiguration des Job-Clients

4.4.2.1 Konfigurationdatei "jobclient.ini"

Die Konfiguration des Job-Clients wird in der Ini-Datei vorgenommen. Falls diese nicht durch den Aufrufparameter "-ini" beim Start des JobClients spezifiziert ist, wird "jobclient.ini" als Konfigurationsdatei verwendet.

4.4.2.1.1 Allgemeine Parameter

Die folgenden Parameter können konfiguriert werden:

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>Beschreibung:</th>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Name / IP-Adresse und Port des Servers.</td>
<td>host=<a href="">Hostname:Portnummer</a></td>
</tr>
<tr>
<td>volume</td>
<td>Der Name des Wissensnetzes, auf dem gearbeitet werden soll.</td>
<td>volume=&lt;Volumename&gt;</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>jobPools={Jobname1}[,{Jobname2}, ...]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Beispiel: jobPools=KScriptJob, query</td>
<td></td>
</tr>
<tr>
<td>cacheDir</td>
<td>Beschreibung des Ortes, an dem der Cache für den Job-Client angelegt wird.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cacheDir=&lt;Verzeichnis&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Beispiel: volumeAccessor=CatBSBlockFileVolumeAccessor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>oder</td>
<td></td>
</tr>
<tr>
<td></td>
<td>volumeAccessor=CatCSVolumeFileStorageAccessor</td>
<td></td>
</tr>
<tr>
<td>maxCacheSize</td>
<td>Zielgröße des Caches</td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxCacheSize=&lt;Größe in MB&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shutDownTimeout=&lt;Sekunden&gt;</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Beschreibung</td>
<td>Beispiel</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>enableLowSpaceHandler</td>
<td>Mit dieser Option wird der LowSpaceHandler eingeschaltet. Dieser sollte auf jeden Fall bei großen Netzabgriffen eingeschaltet werden.</td>
<td>enableLowSpaceHandler=true/false</td>
</tr>
<tr>
<td>useProxyValueHolder</td>
<td>Mit dieser Option kann gesteuert werden, ob der Job-Client Indexzugriffe per RPC durchführt (true), oder Indizes in den Speicher lädt (false). Diese Option sollte ausgeschaltet werden, wenn der Mediator entlastet werden soll. Dabei sollte allerdings darauf geachtet werden, dass der Job-Client genug Speicher zur Verfügung hat. Falls der Job-Client für schreibende Jobs konfiguriert wurde, hat diese Option keinen Effekt, da dann der Indexzugriff immer per RPC durchgeführt wird. Es wird beim Start im Log eine Meldung ausgegeben, falls man den Wert auf false gesetzt hat.</td>
<td>useProxyValueHolder=true/false</td>
</tr>
<tr>
<td>loadIndexes</td>
<td>Seit 4.2 gibt es die Option loadIndexes=true. Indizes werden dann ebenfalls in den Speicher geladen. Im Gegensatz zur Option useProxyValueHolder ist aber auch weiterhin schreibender Zugriff möglich. Die Option kann bei allen Clients inkl. Knowledge-Builder aktiviert werden.</td>
<td>loadIndexes=true/false</td>
</tr>
<tr>
<td>name</td>
<td>Dieser Name wird verwendet, um den Job-Client im Admin-Tool in der Übersichtsliste aller Job-Clients zu identifizieren.</td>
<td>name=&lt;Job-Client-Name&gt;</td>
</tr>
</tbody>
</table>

```
scheduledJobs=<Job-Name1>, <Job-Name2>, ...
```

Speicher-Einstellungen:
Die folgenden drei Parameter dienen zur Konfiguration der Speicherzuteilung und -nutzung. Erlaubt ist die Angabe von Werten entweder in Megabyte oder in tatsächlichen Byte, wobei die Annahme gilt, dass sich Werte kleiner als 1048576 auf Megabyte-Angaben beziehen.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Beschreibung</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxMemory</td>
<td>Maximal erlaubte Hauptspeicherbelegung. Minimal 50 MB, standardmäßig gesamter</td>
<td>maxMemory=&lt;Integer, in MB&gt;</td>
</tr>
<tr>
<td></td>
<td>physikalisch vorhandener Hauptspeicher (unter Windows) bzw. 512 MB.</td>
<td></td>
</tr>
<tr>
<td>baseMemory</td>
<td>Hauptspeicherbelegung der verstärkt versucht wird, Speicher freizugeben.</td>
<td>baseMemory=&lt;Integer, in MB&gt;</td>
</tr>
<tr>
<td></td>
<td>Standardmäßig 0.6 * maxMemory. (alias: &quot;growthRegimeUpperBound&quot;)</td>
<td></td>
</tr>
<tr>
<td>freeMemoryBound</td>
<td>Falls belegt, aber nicht mehr benötigter Speicher diese Grenze überschreitet,</td>
<td>freeMemoryBound=&lt;Integer, in MB&gt; [10]</td>
</tr>
<tr>
<td></td>
<td>wird er wieder freigegeben.</td>
<td></td>
</tr>
<tr>
<td>minAge</td>
<td>Mindestdauer (in Sekunden), die ein Cluster im Speicher bleibt. Ein Cluster</td>
<td>minAge=&lt;Integer&gt; [30]</td>
</tr>
<tr>
<td></td>
<td>ist eine Menge von Objekten, die immer zusammen am Stück geladen werden (z.B.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ein Individuum mit all seinen (Meta)eigenschaften. Cluster, die längere Zeit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nicht mehr verwendet werden, werden bei Bedarf ausgelagert.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Beschreibung</td>
<td>Beispiel</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>unloadInterval</td>
<td>Minstdauer (in Sekunden) zwischen zwei Cluster-Auslagerungen</td>
<td>unloadInterval=&lt;Integer&gt; [10]</td>
</tr>
<tr>
<td>unloadSize</td>
<td>Mindestanzahl an geladenen Cluster, ab der ausgelagert wird</td>
<td>unloadSize=&lt;Integer&gt; [4000]</td>
</tr>
<tr>
<td>keepSize</td>
<td>Zahl der Cluster, die beim Auslagern behalten werden.</td>
<td>keepSize=&lt;Integer&gt; [3500]</td>
</tr>
</tbody>
</table>

**Job Konfiguration:**


Beispiel:

```
[Job-Name1]
<Parameter>=<Wert>
...

[Job-Name2]
...
```

**Logging-Einstellungen:**

Zu den Konfigurationsmöglichkeiten des Loggings siehe Logging-Einstellungen im Kapitel 11.1.2 Konfigurationsdatei.

**Lucene-Server-Konfiguration:**

Die Einbindung von Lucene erfolgt über einen Job Client, dessen jobclient.ini-Datei dafür entsprechend konfiguriert werden muss. Nachfolgend eine Beispielkonfiguration:

```
[lucene]
directory=lucene-index
port=5100
pageSize=100
; Wildcards am Wortanfang sind standardmäßig verboten, da sehr langsam
; In dieser Konfiguration erlauben
allowLeadingWildcards=true

[JNI]
classPath=lucene-6.4.1\core\lucene-core-6.4.1.jar;lucene-6.4.1\analysis\common\lucene-analyzers-common.jar
```

Das Verzeichnis `lucene-6.4.1` enthält die Binaries von Lucene. Im Verzeichnis `lucene-index` wird der Index gespeichert.
### 4.4.2.1.2 Job-spezifische Parameter

#### Allgemein:

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>Beschreibung:</th>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobPool</td>
<td>JobPool für die Ausführung des Jobs.</td>
<td>jobPool=&lt;Job-Pool-Name&gt;</td>
</tr>
<tr>
<td>...?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### scheduledJobs:

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>Beschreibung:</th>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>Zeitpunkt an dem zum ersten mal der Job ausgeführt werden soll. (d=Tage, h=Stunden, m=Minuten, s=Sekunden)</td>
<td>time=&lt;Uhrzeit&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Beispiel:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time=22:15</td>
</tr>
<tr>
<td>interval</td>
<td>Angabe wie häufig der Job ausgeführt werden soll. (d=Tage, h=Stunden, m=Minuten, s=Sekunden)</td>
<td>interval=&lt;Zeitangabe&gt;</td>
</tr>
<tr>
<td>command</td>
<td><strong>Nur bei KExternalCommandJob.</strong> Name einer externen Batchdatei, die vom Job ausgeführt werden soll.</td>
<td>command=&lt;Dateiname.cmd&gt;</td>
</tr>
<tr>
<td>scriptName</td>
<td><strong>Nur bei KScriptJob.</strong> Registrierungsschlüssel eines internen Skripts, das vom Job ausgeführt werden soll.</td>
<td>command=&lt;Skriptressource&gt;</td>
</tr>
<tr>
<td>unique</td>
<td>(?)</td>
<td>unique=true/false</td>
</tr>
<tr>
<td>:user</td>
<td>(Nur ?) Interne Name einer Benutzerinstanz unter der der Job ausgeführt werden soll.</td>
<td>user=&lt;Username&gt;</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>arguments</td>
<td>(Nur KEternalCommandJob?) Argumente die beim Skriptaufruf übergeben werden.</td>
<td>arguments=&lt;Argument1 [Argument2 ...]&gt;</td>
</tr>
</tbody>
</table>

### 4.4.2.2 JobPool Typen

Die folgenden Typen an JobPools stehen zur Verfügung:

#### 4.4.2.2.1 Indexjobs

- Kategorie(n): index

Werden für den Jobpool die unten angezeigten Jobklassen oder index angegeben, dann werden die Indexierungsaufträge vom Job-Client ausgeführt. Die Indexierungsaufträge sollten nur von einem einzigen Job-Client durchgeführt werden. Statt alle Jobklassen einzeln im Job-Pool aufzuzählen, kann auch der symbolische Name index verwendet werden.

**KAddAllToListIndexJob**

- Bezeichnung: Attribute zum Index hinzufügen

**KLightweightIndexJob**

- Bezeichnung: Externen Index aktualisieren

    Ein externer Index wird über den KLightweightIndexJob gepflegt.

**KLuceneAdminJob**

- Bezeichnung: Lucene Verwaltungsaufgabe

    Der KLuceneIndexJob verwaltet einen extern aufgebauten Lucene-Index.

**KRemoveIndexJob**

- Bezeichnung: Attribute aus dem Index entfernen

**KSincIndexJob**

- Bezeichnung: Index synchronisieren
**KAddAllToIndexJob, KRemoveIndexJob und KSyncIndexJob** werden benötigt, um die internen Indizes zu pflegen.

### 4.4.2.2 KBrainbotJob

- **Kategorie(n):** <keine>
- **Bezeichnung:** KBrainbotJob

Der KBrainbotJob führt Aktionen zur Pflege eines Brainbot-Indexes aus.

Falls innerhalb der Konfiguration im Admin-Tool angegeben wird, dass Pflegeaktionen von einem Jobclient ausgeführt werden sollen ("Jobclient benutzen"), so muss ein Jobclient gestartet werden, damit die Pflege des externen Index ausgeführt wird.

Der KBrainbotJob hat keine weiteren Konfigurationsparameter in der ini-Datei, da die gesamte Konfiguration im Admin-Tool stattfindet.

### 4.4.2.3 KExternalCommandJob

- **Kategorie(n):** <keine>
- **Bezeichnung:** Externer Aufruf


Das Hauptelement des Skriptauftragdes ist das Element **ExternalCommandJob**. Mit dem Attribut execution kann eingestellt werden, ob der Job lokal ohne JobClient (Wert: local) oder mit JobClient (Wert: remote) ausgeführt werden soll. Der Standardwert ist remote.

**Anmerkung zur Remote-Ausführung:**


Das notwendige, erste Unterelement im Skript:

- **Command:** gibt an welche Batchdatei aufgerufen werden soll

\[\text{<Command>convert.bat</Command>}\]

*In dem Element Command wird der Name der Batchdatei angeben. In der Batchdatei ist das Verzeichnis und das auszuführende Programm selbst angegeben. **Wichtig:** Die Batchdatei muss auf der gleichen Ebene wie das Programm (z.B. JobClient oder KB) liegen. Verzeichnisangaben im Element Command werden ignoriert.*

Die weiteren Unterelemente werden von oben nach unten abgearbeitet. Falls die Reihenfolge der Parameter im externen Programm eine Rolle spielt, sollte dies berücksichtigt werden.
Skriptelemente, die die Parameter für den Aufruf bilden:

  
  `<OptionString>-size 100x100</OptionString>`

- **OptionPath**: der angegebene Path-Ausdruck wird ausgewertet und als Zeichenkette in den Kommandoaufruf eingebaut.
  
  `<OptionPath path="./topic()/concept()/$size$"/>`

Skriptelemente, die sich mit dem Handling von Attributen beschäftigen

- **SourceBlob**: Angabe des Blobattributes, das als Datenquelle verwendet wird.
  
  `<SourceBlob><Path path="$bild$"/></SourceBlob> <SourceBlob path="$bild$"/>`

- **ResultAttribute**: Angabe der Parameter für die Erzeugung eines neuen oder die Veränderung eines bestehenden Blobattributes mit dem Inhalt der Datei bzw. der Datei selbst, die das Ergebnis des extern aufgerufenen Programms ist.
  
  Attributwerte:
  
  - **name**: Name bzw. interner Name des anzulegenden Attributes
  - **topic**: Zielindividuum des anzulegenden Attributes
  - **modifyExisting**: verändern (true) oder neu anlegen (false, Standardwert)
  - **filename**: Dateiname des anzulegenden Blobattributes
  
  `<ResultAttribute name="$bild2$" topic="./topic()" modifyExisting="true" filename="convert_ +./valueString()"/>
  
  `<Path path="$bild2$"/></ResultAttribute>`

Beispiel 01:

Skript:

```xml
<Script>
  <ExternalCommandJob execution="local">
    <Command>convert.bat</Command>
    <OptionString>-size 100x100</OptionString>
  </ExternalCommandJob>
</Script>
```

Inhalt der Batchdatei unter Windows:

"C:\Program Files\ImageMagick-6.2.6-Q16\convert.exe" %*
exit /B %ERRORLEVEL%

Inhalt der Batchdatei unter Linux:

```
#!/bin/bash
convert $*
```

Beispiel 02:

Skript:

```xml
<Script>
  <ExternalCommandJob execution="local">
    <Command>convert2.bat</Command>
  </ExternalCommandJob>
</Script>
```

Inhalt der Batchdatei unter Windows:

"C:\Program Files\ImageMagick-6.2.6-Q16\convert" -size 100x100 %1
-geometry +5+10 %2 -geometry +35+30 -composite %3
exit /B %ERRORLEVEL%

Inhalt der Batchdatei unter Linux:

```
#!/bin/bash
```

Hier noch ein Beispiel für ein erweitertes Konvertierungsskript, welches mit den Parametern "Quelldatei", "Bildbreite" und "Zieldatei" aufgerufen werden kann und welches nur breitere Bilder auf die angegebene Breite verkleinert. Das Script schreibt außerdem eine Protokolldatei über die Konvertierung wobei auch Fehlermeldungen von Image Magick in die Logdatei geschrieben werden:

```bash
set MONTH_YEAR=%DATE:~8%
echo Converting %1 to %3 (width: %2) >> convert%MONTH_YEAR%.log
custom.exe %1 -resize "%2" %3 2 >> convert%MONTH_YEAR%.log
echo Conversion finished with exit code %ERRORLEVEL% >> convert%MONTH_YEAR%.log
exit /B %ERRORLEVEL%
```

Und hier noch die Version für Linux (Bash):

```bash
#!/bin/bash
FULLDATE=`date +%c`
MONTH_YEAR=`date +%m.%Y`
LOGFILE="convert.$MONTH_YEAR.log"
echo "$FULLDATE: Converting $1 to $3 (width: $2)" >>$LOGFILE
custom "$1" -resize "$2" "$3" >>$LOGFILE
EXITCODE="?"
echo $FULLDATE: Conversion finished with exit code $EXITCODE>>$LOGFILE
exit $EXITCODE
```

4.4.2.2.4 KExtractBlobTextJob

- Kategorie(n): <keine>

4.4.2.2.5 KQueryJob

- Kategorie(n): query
- Bezeichnung: Suche
Dient der ausgelagerten Ausführung von einfachen und Expertensuchen auf einem Jobclient. Wird je nach Bedürfnissen der betrachteten Suche ausgestattet und ausgeführt.

4.4.2.6 KScriptJob

- Kategorie(n): script
- Bezeichnung: KScriptJob


Der Parameter "scriptName" muss hierfür auf den Registrierungsschlüssel eines im Netz hinterlegten Skripts verweisen. Das Skript wird automatisch in einer Transaktion gekapselt.

4.4.2.3 Beispiel für eine Ini-Datei

```plaintext
volume=MeinNetz
host=localhost
jobPools=query, index
cacheDir=jobcache
logfile=jobclient01.log
maxMemory=400
name=jobclient01
```

4.4.2.4 Performance-Optimierungen

Vorab laden


In die Ini-Datei des JobClients muss der Eintrag `keepClusterIDs` angegeben werden. Mögliche Werte für diesen Eintrag sind:

- `index` - Bei den Einstellungen der zusammensteckbaren Indexern gibt es die Möglichkeit, das Häkchen bei `jobclient soll Index in den Hauptspeicher laden` zu setzen. Für die aktivierten Indexer wird ein Teil ihrer Indexstruktur geladen.
- `protoOfSizes` - Die Anzahl der Individuen für jedes Konzept werden bereits beim Start ermittelt.
- `accessRights` - Das Root-Objekt des Rechtesystems wird in den Speicher geladen.

**Wichtig:** Für den Eintrag `useProxyValueHolder` muss der Wert `false` gesetzt sein. Sonst versucht der JobClient RPCs (Anfragen, die der Mediator beantworten kann) an den Mediator
abzusetzen. Der Client soll jedoch die Cluster selber laden und unter Umständen auch im Speicher behalten.

**Anmerkung:** Es ist ebenfalls von Vorteil, für eine Performanceverbesserung den Festplatten-cache für den JobClient einzuschalten.

Beispiel für die Einträge in der INI-Datei:

```ini
[Default]
...
useProxyValueHolder=false
keepClusterIDs=index,protoOfSizes,accessRights
cacheDir=jobcache
maxCacheSize=1000
...
```

### 4.5 BLOB-Service

#### 4.5.1 Einführung

Der Blob-Service dient der Datenhaltung von großen Dateien außerhalb des Wissensnetzes, aber verknüpft mit den Datei-Attributen, in denen diese Dateiinhalte abgelegt werden sollen. Dies hat mehrere Vorteile:

- Das Wissensnetz enthält dadurch nur noch die semantischen Informationen, die auf den Dateien aufsetzen und bleibt gut sicher- und übertragbar.
- Speicherorte von Wissensnetz und Dateiinhalten können unterschiedlich konfiguriert werden.
- Es lassen sich mehrere Blob-Services an ein Wissensnetz anschließen, so dass theoretisch je Attributdefinition ein Speicherort vorgehalten werden kann.

Im folgenden Kapitel wird das Einrichten und der Betrieb von Blob-Services erläutert.

#### 4.5.2 Konfiguration

Um festzulegen, unter welcher Netzwerk-Adresse (Host und Port) der Blobservice erreichbar sein soll, muss in der Datei "blobservice.ini" die Option "interfaces" eingetragen werden. Prinzipiell gibt es dabei zwei Möglichkeiten:

1. Der BLOB-Service soll nur von dem Rechner aus erreichbar sein, auf dem der BLOB-Service installiert ist
2. Der BLOB-Service soll über das Netzwerk auch von anderen Rechnern aus erreichbar sein.

Hier ein Konfigurationsbeispiel für Variante 1, wobei der BLOB-Service-Port (30000) auch frei wählbar ist:

```ini
interfaces=http://localhost:30000
```

Zur Konfiguration von Variante 2 muss man anstelle von "localhost" die IP-Adresse des Netzwerk-Adapters eintragen, über den der BLOB-Service aus dem Netzwerk ansprechbar
sein soll. Möchte man, dass der BLOB-Service über alle Netzwerk-Adapter erreichbar ist, die auf dem Rechner aktiv sind, so muss man als IP-Adresse "0.0.0.0" eintragen. Beispiel:

```
interfaces=http://0.0.0.0:30000
```

Wird der BLOB-Service über das Netzwerk angesprochen, so sollte die Kommunikation verschlüsselt werden. Die verschlüsselte Kommunikation über HTTPS kann ebenfalls in der Option "interfaces" konfiguriert werden, indem "http://" durch "https://" ersetzt wird. Beispiel:

```
interfaces=https://0.0.0.0:30000
```

Für den verschlüsselten Fall siehe auch das nachfolgende Kapitel SSL Zertifikate.

Um den Betrieb zu gewährleisten, muss zusätzlich im Arbeitsverzeichnis die DLL des SQLite Frameworks "sqlite3.dll" vorhanden sein. Ohne diese DLL kann die intern benötigte Verwaltungsstruktur nicht aufgebaut und gepflegt werden.

Danach kann der Blobservice gestartet werden und steht ab sofort zur Verfügung.

Um den Blobservice mit einem Blobstore in der semantischen Graph-Datenbank zu verknüpfen, bietet das Admin-Tool unter "Systemkonfiguration -> Blob-Speicherung" die nötigen Werkzeuge:

Durch Klicken auf "Anlegen" (1) wird eine neuer logischer Store erzeugt. Danach muss in das Eingabefeld "URL" (2) die in der ini-Datei angegebene URL des Blobservices eingetragen werden und dann auf "Hinzufügen" (3) geklickt werden. Der neu gebaute Blobstore für externe Haltung von Dateiattributen ist danach mit dem Blobservice verknüpft, was durch Klicken auf "Aktualisieren" (4) im unteren Darstellungsbereich kontrolliert werden kann.
Im Bereich "URLs" (2) kann auch eine Liste alternativer URLs per Komma getrennt angegeben werden. i-views bevorzugt bei alternativen URLs wenn möglich eine Verbindung über ein Loopback-Device.

Der Bereich "Löscharbare Dateien" (7) zeigt die Zahl der aus Sicht des Wissensnetzes nicht mehr benötigten Dateien an. Mit "Löschen" (8) werden diese im Blob-Service dereferenziert und ggf. entfernt.

Der Indikator "Intern" (9) zeigt an, dass es sich um einen in einen Mediator integrierten Store handelt. Interne Stores werden bei Volume-Transfers (upload, download, copy, backup, recovery) automatisch mit dem Volume transferiert.

Will man die Verknüpfung eines Blobstores zu einem Blobservice aufheben, so selektiert man den gewünschten Blobstore in der Liste "Externe Speicher im Blob-Service" und klickt "Entfernen" (5). Danach kann man den Blobstore im oberen Bereich "Externe Speicher für Dateiattribute" selektieren und durch Klicken auf "Löschen" (6) ganz entfernen, oder man kann durch Angabe einer neuen URL den Blobstore mit einem anderen Blobservice wie oben beschrieben neu verknüpfen.

ACHTUNG!

Durch das Auflösen der Verknüpfung eines Blobstores zu einem Blobservice gehen alle dort gespeicherten Dateien verloren!

### 4.5.3 SSL Zertifikate

Zur Konfiguration der HTTPS-Verbindung müssen das Zertifikat und der Private-Key abgelegt werden.

Das Zertifikat muss unter certificates/server.crt liegen.

Der Private-Key muss unter private/server.key liegen. Es ist darauf zu achten, dass server.key als RSA-Key vorliegt, d.h. die erste Zeile der Datei muss

```
——BEGIN RSA PRIVATE KEY——
```

lauten. Wenn der Key in einem anderen Format vorliegt, muss er konvertiert werden. Mittels OpenSSL ist dies bspw. mittels "openssl rsa -in input.key -out private/server.key -outform PEM" möglich.

### 4.6 Als OS-Dienst installieren

Die Dienst-Programme können unter den verschiedenen unterstützten Betriebssystemen die Möglichkeit als OS-Dienst eingerichtet zu werden.

Für Unix-artige Betriebssysteme sind die auf der jeweiligen Plattform unterstützten Mechanismen zu verwenden, einige Beispiele finden sich im versions-unabhängigen Handbuch von i-views.

Für MS-Windows bieten die Dienste die Parameter -installAsService NAME und -deinstallService NAME an, um aus einer administrativen Shell heraus einen von Windows verwalteten Dienst einzurichten bzw. zu entfernen. Beim Installieren werden alle Parameter, die nach dem Servicenamen angegeben werden, dem installierten Dienst als Kommandozeilenparameter übergeben. Z.B. richtet

```
bridge -installAsService iviews-bridge-rest -inifile bridge-rest.ini
```

einen Dienst mit dem Namen "iviews-bridge-rest" ein, der als Aufrufzeile
PFAD\bridge.exe bridge.exe -serviceName iviews-bridge-rest -ini bridge-rest.ini

erhält.