

MyTISM

next generation application framework

Das Entwickler-Handbuch

Version: Version 0.28.02 - 2011-05-18

Inhaltsverzeichnis

Vorwort	iii
1. Schema	1
1.1. Schema-Definition	1
1.2. Coredata-Generator	10
2. Sprachunterstützung und Internationalisierung	13
2.1. Einführung	13
2.2. Wo wird Mehrsprachigkeit unterstützt und wie benutze ich sie?	13
2.3. Wie wird die konkrete Zeichenkette für einen Schlüssel gefunden?	13
2.4. Welche L10nPacks gibt es und wie sind diese organisiert? Wie wird bestimmt, welche L10nPacks nach Texten durchsucht werden?	14
2.5. Welches sind die "beteiligten bzw. relevanten Objekte"?	14
2.6. Wo kommen die (Daten der) L10nPacks her?	15
2.7. Wichtige Klassen	15
2.8. Eingabe von L10n-Daten	16
3. Die Formularengine des Solstice Clients	17
3.1. de.ipcon.form	17
3.2. Fehler und Ursachen	18
4. BOMengen	20
4.1. Grundlagen	20
4.2. Gemeinsame Eigenschaften aller BOMengen	20
4.3. Typen	20
5. Synchronisation der Strukturelemente	22
5.1. Das Formular "DateiSystemSync"	22
6. Volltextsuche	23
6.1. Konfiguration im Schema	23

Vorwort

MyTISM ist plattformunabhängiges, objektorientiertes, dezentrales, multiuserfähiges, individuell anpassbares und quelloffenes 3-Tier-Datenbank- und Anwendungs-Framework incl. GUI und Web-Application-Server, entwickelt und betreut von OAshi s.a r.l.

In diesem Handbuch finden Sie alle Informationen, die Sie für die Programmierung von und mit MyTISM benötigen.



Anmerkung

Beachten Sie bitte, dass sich dieses Dokument noch im Aufbaustadium befindet und noch grosse Lücken aufweist, die wir natürlich nach und nach füllen werden.

Dieser Teil der Dokumentation ist im Wesentlichen dem Entwickler bzw. dem (sehr) interessierten Anwender gewidmet, der Einblick in die Interna bzw. Aufbau des Systems gewinnen will. Das Vorwissen besteht mindestens aus guten bis sehr guten Kenntnissen in Java und NetRexx sowie Datenbank-Kenntnisse in relationalen oder objektorientierten Systemen. Eine mindestens halbwegs intensive Beschäftigung mit der Bedienung des Systems sollte der Lektüre vorangegangen sein...

Bei Fragen, Problemen oder Anregungen, sei es bzgl. MyTISM selber oder dieser Dokumentation, wenden Sie sich bitte an uns; Kontaktinfos finden Sie im WWW unter <http://www.mytism.de/mytism/contact>.

Kapitel 1. Schema

Das Schema definiert/spezifiziert, wie die Datenobjekte bzw. die "Datenbankstruktur" einer Applikation aussieht. Aus der Schema-Definition wird automatisch der Netrexx-Quellcode für die benötigten Klassen generiert, die Datenbanktabellen etc. werden angelegt und für Solstice werden passende Automatik-Formulare, -Schablonen und Lesezeichen generiert.

1.1. Schema-Definition

FIXME: Besteht aus `Include`, `Entity` und `Generator`

Einleitender Tag: `<Schema BEFEHLE>`

Schliessender Tag: `</Schema>`

Tabelle 1.1. Schema

Befehl	Beschreibung	Beispiel
version	Freitext, der den Platzhalter <code>@BUILT@</code> enthalten sollte, welche beim Kompilieren durch eine Versionsnummer ersetzt wird; weitere Platzhalter sind: <code>@ProjectName@</code>	<code><Schema version="@ProjectName@ Schema built @BUILT@"></code>
defaultPackage	"Basis"-Package, welches bei der Entität im <code>extends</code> -Befehl eigentlich eigentlich noch vorangestellt werden müsste (also statt <code>extends="Artikel"</code> müsste man dann überall schreiben <code>extends="de.laeis.bo.Artikel"</code>); gleiches gilt für das Attribut und den <code>type</code> -Befehl (natürlich nur bei Entitäten). Das <code>defaultPackage</code> erspart einem also diese lästige Mehrarbeit. Schön, nicht?	<code><Schema version="@ProjectName@ Schema built @BUILT@" defaultPackage="@BOPACK@"></code>
defaultFolder	Hier kann zwecks geordneter Erstellung der nachfolgend definierten Entitäten ein <code>defaultFolder</code> definiert werden.	<code><Schema version="@ProjectName@ Schema built @BUILT@" defaultPackage="@BOPACK@" defaultFolder="Stammdaten/Konten"></code>

1.1.1. Include

FIXME: Einleitende, erklärende Worte

Tag: `<Include BEFEHLE />`

Tabelle 1.2. Include

Befehl	Beschreibung	Beispiel
file	Name der zu inkludierenden Datei	<Include file="/de/ipcon/schema/schema-core.xml">
child	FIXME	<Include file="/de/ipcon/schema/schema-core.xml" child="true">

1.1.2. Folder

FIXME: Einleitende, erklärende Worte

Tag: <Folder BEFEHLE />

Tabelle 1.3. Folder

Befehl	Beschreibung	Beispiel
path	Name der Pfads	<Folder path="Quertabellen">

1.1.3. Entity

FIXME: Einleitende, erklärende Worte: Definition einer Entität und ihrer Attribute

Einleitender Tag: <Entity BEFEHLE>

Schliessender Tag: </Entity>

Tabelle 1.4. Entity

Befehl	Beschreibung	Beispiel
name	Name der Entität	<Entity name="Lieferant">
plural	Plural-Bezeichnung der Entität	<Entity name="Lieferant" plural="Lieferanten">
extends	Von welcher Klasse sich die Entität ableitet	<Entity name="Lieferant" plural="Lieferanten" extends="XBuchungskonto">
abstract	Ob die Entität abstrakt sein soll (es können keine BOs von dieser Entität angelegt werden, da keine Formulare generiert werden - macht nur Sinn, wenn sich von dieser Entität weitere	<Entity name="XBuchungskonto" plural="XBuchungskonten" extends="CBO">

Befehl	Beschreibung	Beispiel
	Entitäten ableiten; Default: false	abstract="true">
ignoreReverse-Relations	FIXME; Default: false	<Entity name="Lieferant" plural="Lieferanten" extends="XBuchungskonto" ignoreReverseRelations="true">
discriminator	FIXME Irgendwas bzgl. BOT/Typ	<Entity name="Lieferant" plural="Lieferanten" extends="XBuchungskonto" discriminator="FIXME">
suid	FIXME SerialVersionUID (Klassenversionskennung)	<Entity name="Lieferant" plural="Lieferanten" extends="XBuchungskonto" suid="FIXME">
package	Hier kann ein zum defaultPackage (siehe obigen Abschnitt zum Schema) abweichendes Package angegeben werden.	package="de.laeis.bo"

1.1.3.1. Unter-Element "ui" von Entity

FIXME: Einleitende, erklärende Worte

Tabelle 1.5. Unter-Element "ui" von Entity

Befehl	Beschreibung	Beispiel
niceName	FIXME "Schönerer" Name für Entität für Anzeige	<ui niceName="FIXME">
description	Beschreibung der Entität; definiert Ausgabe der describe()-Methode einer jeden Entität, Definition im CBOFormat (siehe dort "Wo kann man das CBOFormat nun überhaupt einsetzen?")	<ui description="Nachname(', Vorname)">
loadImmediate	Ob eine Übersicht (Liste) der BOs der Entität; direkt in einem geöffneten Lesezeichen (FTable) angezeigt werden soll (aus Performance-Gründen nur sinnvoll bei Entitäten mit einer überschaubaren Anzahl von BOs); Default: false	<ui loadImmediate="false">
linkOnly	Ob Objekte nicht im Kontext dieses BO angelegt, sondern sie lediglich mit diesem BO verknüpft werden können sollen; Default: false	<ui linkOnly="false">
tips	FIXME	<ui tips="FIXME">

Befehl	Beschreibung	Beispiel
defaultSorting	<p>Tabellen, die Objekte diesen Typs anzeigen, werden standardmäßig nach diesen Vorschriften sortiert, sofern keine explizite Sortierung in der Tabelle definiert worden ist. Subentities "erben" dabei das defaultSorting, falls sie selbst kein eigenes definiert haben. Format der Definition: Spaltenname:Sortierrichtung. Dabei impliziert die Reihenfolge der Einträge den sortLevel.</p> <p><i>Die Mehrfachsortierung erfordert eine zusätzliche Lizenz.</i></p>	<pre><ui defaultSorting="Name:ASC Beschreibung:DESC Position:ASC Eigenschaft:ASC Eigenschaft.Name:ASC"></pre>

1.1.3.2. Unter-Element "lookup" von Entity

FIXME

Tabelle 1.6. Unter-Element "lookup" von Entity

Befehl	Beschreibung	Beispiel
defaultProperty	In welchem Attribut der BOs vom entsprechenden Typ bei Eingabe eines Suchstrings in einem Popup gesucht werden soll, damit nach Eingabe von Enter direkt das Objekt (falls nur eines gefunden wurde) an das BO angehängt bzw. eine Liste von BOs mit diesem Wert im angegebenen Attribut angezeigt werden kann.	<pre><lookup defaultProperty="Name"></pre>
defaultSubstring	Ob die Suche exakt sein soll oder ein Substring-Match in der defaultProperty schon ausreicht; Default: true	<pre><lookup defaultSubstring="true"></pre>
defaultCaseSensitive	Ob die Suche Groß-/Kleinschreibung beachten soll; Default: false	<pre><lookup defaultCaseSensitive="true"></pre>

1.1.3.3. Unter-Element "code" von Entity

FIXME

Tabelle 1.7. Unter-Element "code" von Entity

Befehl	Beschreibung	Beispiel
package	Hier kann ein zum defaultPackage (siehe obigen Abschnitt zum Schema) abweichendes Package angegeben werden.	<pre><code package="de.laeis.bo"></pre>

Befehl	Beschreibung	Beispiel
generateAs	Unter welchem Namen die Basis-Klasse der Entität angelegt werden soll; es muss dann im "bo"-Verzeichnis von Hand eine Datei "Person.nrx" angelegt werden, welche die Klasse "Person" implementiert und sich von der generierten Klasse "PersonBase" ableitet	<code generateAs="PersonBase">
custom	Ob die Basis-Klasse der Entität mit Suffix "Base" angelegt werden soll; es muss dann im "bo"-Verzeichnis von Hand eine Datei mit dem Entity-Namen und Suffix ".nrx" angelegt werden, welche die Klasse "Entity-Name" implementiert und sich von der generierten Klasse "Entity-NameBase" ableitet	<code custom="true">
generate	Ob für die Entität Sourcecode generiert werden soll (eigentlich nur für "BO" mit "false" benutzt); Default: true	<code generate="false">
dependents	Welche Klassen von dieser Entität abhängen und die daher bei Änderung der Klasse der Entität ebenfalls neu gebaut werden müssen (z.B. damit die serialVersionUID aktuell ist).	<code dependents="GeschaeftsVorfallSchemaAspects">

1.1.3.4. Unter-Element "db" von Entity

FIXME

Tabelle 1.8. Unter-Element "db" von Entity

Befehl	Beschreibung	Beispiel
persistent	Ob die BOs der Entität persistiert (gespeichert) werden sollen, Default: true	<db persistent="false">
generateAs	Unter welchem Namen die Basis-Klasse der Entität angelegt werden soll; es muss dann im "bo"-Verzeichnis von Hand eine Datei "Person.nrx" angelegt werden, welche die Klasse "Person" implementiert und sich von der generierten Klasse "PersonBase" ableitet	<db generateAs="PersonBase">
streamRessource	FIXME Ob an diesen Entitäten ein BLOB (BinaryLargeObject) dranhängen kann; Default: false	<db streamRessource="true">
forbidDirectChanges	Ob direkte Änderungen an BOs diesen Typs vorgenommen werden dürfen oder ob nur der Server diese Objekte modifizieren darf; Default:	<db forbidDirectChanges="true">

Befehl	Beschreibung	Beispiel
	false	

1.1.3.5. Unter-Element "report" von Entity

FIXME

Tabelle 1.9. Unter-Element "report" von Entity

Befehl	Beschreibung	Beispiel
title	Titel für automatisch generierte Reports	<report title="Analyse">
orientation	Orientierung für automatisch generierte Reports (Portrait oder Landscape)	<report orientation="Portrait">
fontSizeNormal	Die Standard-Schriftgröße für automatisch generierte Reports; Default: 9	<report fontSizeNormal="10">
fontSizeBig	Die große Schriftgröße für automatisch generierte Reports; Default: 16	<report fontSizeBig="14">

Beispiel:

```
<Entity name="Person" extends="CB0" plural="Personen" folder="Kontakte">
  <code custom="true"/>
  <ui description="Nachname(' , 'Vorname)"/>
</Entity>
```

1.1.3.6. Attribut

FIXME: Einleitende, erklärende Worte

Tag für Attribute: <attr BEFEHLE />

Tag für virtuelle Attribute: <vattr BEFEHLE />

Tag für nicht-persistente Attribute: <npattr BEFEHLE />

Tabelle 1.10. Attribut

Befehl	Beschreibung	Beispiel
name	der Attribut-Name	name="Adressen"
backName	FIXME: Bei Relationen Name des Attributes "auf der anderen Seite".	backName="FIXME"
singular	FIXME: Wird kein singular angegeben wird der Singular auf den Wert von name gesetzt	singular="Adresse"

Befehl	Beschreibung	Beispiel
type	Typ des Attributs; mögliche Werte: FIXME	type="Kontakt" (eine Adresse ist eine bestimmte Art einer Kontakt-Möglichkeit)
displayFormat	Standard-Display-Format des Attributs	displayFormat="Name"
relation	Relation des Attributs zu einer anderen Entität; mögliche Werte: "n-1", "1-n", "n-m"	relation="1-n"
dependent	Abhängigkeit: in unserem Beispiel werden beim Löschen der Person auch die zugehörigen Adressen gelöscht; ansonsten wie <code>createInDetailView</code> ; Default: false	dependent="true"
itemProperty	FIXME	itemProperty="Position"
shared	Wird die Referenz innerhalb der Entität geteilt; Default: false	shared="true"
default	Für diverse Attribut-Typen lässt sich ein Default-Wert vorgeben (z.B. für <code>Datetime: new java.util.Date(); Boolean: Boolean.TRUE; Decimal: #,##0.00</code>)	default="#,##0.00"
ignoreBackRelation	Es sollen keine Methoden für die Rückrelation generiert werden (z.B. n-1-Relation zu Core-Entität)	ignoreBackRelation="true"
readonly	Gibt an, ob das Attribut-Feld im Formular gesperrt ist für irgendwelche Eingaben; Default: false	readonly="true"
lazy	FIXME; Default: false	lazy="true"
omitOnCopy	Gibt an, ob dieses Attribut beim Kopieren eines Objekts diesen Typs übersprungen werden soll.	omitOnCopy="true"

1.1.3.6.1. Unter-Element "ui" von Attribute

FIXME

Tabelle 1.11. Unter-Element "ui" von Attribute

Befehl	Beschreibung	Beispiel
editMode	Gibt an, wie das Attribut in der UI editierbar sein soll, Mögliche Werte: linkonly, viewonly, locked, writenew, all.	<ui editMode="linkonly">
createInDetailView	Gibt an, ob im Formular der Entität automatisch Eingabefelder für die Many-Relation gebaut	<ui createInDetailView="true">

Befehl	Beschreibung	Beispiel
	werden sollen (normalerweise wird auf dem Formular-Reiter der Many-Relation nur eine FTable gebaut); Default: false	
mandatory	FIXME Definition als "Pflichtfeld". Z.Zt. nur partiell unterstützt, nicht z.B. von Solstice; Default: false	<ui mandatory="true">
tips	FIXME: mögliche Werte: StyledText, Area, comboBox: ATTRIBUTNAME, formRecursionDepth: INTEGER (Default: 3), createShared; werden mehrere Werte angegeben, so sind diese mit Leerzeichen zu trennen	<ui tips="FIXME">
visible	Ob das Attribut im Formular gebaut werden soll; Default: true	<ui visible="false">
expectedWidth	FIXME: Die erwartete Breite des Attributs im Formular	<ui expectedWidth="FIXME">
selectionFilter	FIXME	<ui selectionFilter="FIXME">

1.1.3.6.2. Unter-Element "lookup" von Attribute

FIXME

Tabelle 1.12. Unter-Element "lookup" von Attribute

Befehl	Beschreibung	Beispiel
property	In welchem Attribut der BOs vom entsprechenden Typ bei Eingabe eines Suchstrings in einem Popup gesucht werden soll, damit nach Eingabe von Enter direkt das Objekt (falls nur eines gefunden wurde) an das BO angehängt bzw. eine Liste von BOs mit diesem Wert im angegebenen Attribut angezeigt werden kann.	<lookup property="Name">
substring	Ob die Suche exakt sein soll oder ein Substring-Match in der property schon ausreicht; Default: true	<lookup substring="true">
caseSensitive	Ob die Suche Groß-/Kleinschreibung beachten soll; Default: false	<lookup caseSensitive="true">

1.1.3.6.3. Unter-Element "report" von Attribute

FIXME

Tabelle 1.13. Unter-Element "report" von Attribute

Befehl	Beschreibung	Beispiel
visible	Ob das Attribut in den Automatik-Reports angedruckt werden soll; Default: true	<report visible="false">
relativeWidth	Die relative Breite des Felds für das Attribut in den Automatik-Reports; Default: 1	<report relativeWidth="2">
position	Determiniert die Reihenfolge der Attribute in den Automatik-Reports.	<report position="100">
sort	Sollen die BOs im Automatik-Report nach diesem Attribut sortiert werden? Mögliche Werte: asc, desc	<report sort="asc">
manySort	Sollen die BOs im Automatik-Report nach diesen Attributen sortiert werden? Mögliche Werte: Komma-separierte Liste von Attribut-Namen der Entität mit Suffix :A oder :D.	<report manySort="Tid:A,Nummer:D">
alias	Ob im Automatik-Report für eine many-Relation eine Gruppe mit angegebenem Alias erzeugt und die Daten der many-Relation im Automatik-Report gedruckt werden sollen.	<report alias="P">

1.1.3.6.4. Unter-Element "virtual" von Attribute

FIXME

Tabelle 1.14. Unter-Element "virtual" von Attribute

Befehl	Beschreibung	Beispiel
writeable	Ob das virtuelle Attribut schreibbar sein soll.	<virtual writeable="true">
aggregate	Ob der Wert des virtuellen Attributs durch eine Aggregats-Funktion bestimmt werden soll.	<virtual aggregate="BO.Union:Gruppe.Benutzer">
cacheMode	Ob das virtuelle Attribute versioniert sein soll.	<virtual cacheMode="VERSIONED">

1.1.3.6.5. Unter-Element "db" von Attribute

FIXME

Tabelle 1.15. Unter-Element "db" von Attribute

Befehl	Beschreibung	Beispiel
indexed	Gibt an, ob die Werte eines Attributs indiziert werden sollen; Default: true	<db indexed="false">
unique	FIXME; Default: false	<db unique="true">

1.1.3.6.6. Restliche Unter-Elemente von Attribute

Tabelle 1.16. Restliche Unter-Elemente von Attribute

Unter-Element	Beschreibung	Beispiel
backRelation	Explizite Übersteuerung der automatisch generierten Rückrelation, um z.B. einen anderen Namen zu benutzen oder die Rückrelation mittels Unter-Elementen zu konfigurieren.	<backRelation name="BezugnehmendePosten"/>
comment	Um einen Kommentar zu diesem Attribut im Schema zu hinterlegen.	<comment>Dient zur Speicherung von XXX</comment>

Beispiel:

```
<attr name="Adressen" singular="Adresse" type="Kontakt" relation="1-n"
dependent="true" itemProperty="Position">
```

Die Entität *Person* hat ein Attribut namens *Adressen*. Für das Attribut ist eine 1-n-Relation definiert, d.h. eine *Person* kann mehrere *Adressen* haben. Wird die *Person* gelöscht, werden auch die angehängenen BOs der Entität *Adresse* gelöscht. Durch *itemProperty* besteht die Möglichkeit die jeweilige Position der Adressen innerhalb der Zuordnungsliste bequem mit Pfeil-rauf- und Pfeil-runter-Knöpfen im Formular zu bestimmen.

1.2. Coredata-Generator



Anmerkung

vgl. [de/ipcon/schema/generators/CoreData.nrx](#) und Klassen in [de/ipcon/schema/generators/coredata](#)

Füllt die Datenbank mit grundlegenden, benötigten Daten/Objekten:

1. Füllt die BOT-Liste für alle Entities
2. Legt Admin-Benutzer und Admins-Gruppe an
3. Legt Sammelordner für Automatik-Objekte an (wobei diese im Normalfall direkt wieder gelöscht werden, da mittlerweile alle Entities explizit einen - anderen - Ordner angegeben haben sollten, und diese deshalb leer bleiben)

4. Legt Standard-Druckziele an
5. Erzeugt ein Standard-Formular für jede Entity
6. Erzeugt eine Standard-Schablone für jede nicht-abstrakte Entity
7. Erzeugt ein Standard-Lesezeichen für jede persistente Entity
8. Erzeugt Standard-Reports (Einzel und Liste) für jede Entity
9. Lädt und erzeugt zusätzliche, vorgebaute Strukturelemente
10. Löscht nicht mehr benutzte Automatik-Strukturelemente (also solche von mittlerweile wieder aus dem Schema entfernten Entities) und leere Ordner

1.2.1. Zusätzliche, vorgebaute Strukturelemente

Neben den automatisch erzeugten Strukturelementen können auch zusätzliche, vorgebaute Formulare, Schablonen und Lesezeichen automatisch in die Datenbank eingespielt werden. Diese müssen in Verzeichnis `de/ipcon/db/core/resources` abgelegt werden. Das Format und die Benamsung entspricht dem Format und der Benamsung der mittels Formularsynchronisation (siehe Nutzerdoku) exportierten Dateien, d.h. exportierte Dateien können direkt übernommen werden.

Beim Bauen von `MyTISM-Kernel.jar` werden die Dateien zusammengesucht und in eine Liste ("ResourceIndex") eingetragen und in das Jar aufgenommen. Beim Aufruf des Coredata-Generators wird diese Liste ausgelesen, die entsprechenden Dateien/Definitionen werden aus dem Jar geladen und die entsprechenden Objekte in der DB angelegt bzw. aktualisiert. Die Dateinamen sind übrigens frei wählbar, lediglich die Endung muss passend zum Typ der enthaltenen Daten sein (`*.tpl.xml` = Schablone, `*.frm.xml` = Formular, `*.bkm.xml` = Lesezeichen).

Folgende Angaben werden z.Zt. unterstützt:

<Root-Element>

Der Name des Root-Elements gibt den Typ der Daten an, also "Formular", "Schablone" oder "Lesezeichen"

Name

"Name"-Attribut des Root-Elements setzt den Namen des Objekts - an sich frei wählbar, Konvention aber "`<Entityname>` (Vorgebaut)" oder "`<Entityname>` (Vorgebaut; `<Kommentar>`)" wenn es mehrere Versionen gibt.

ElterPfad

"ElterPfad"-Attribut des Root-Elements gibt den Ordner an, in dem das Objekt abgelegt werden soll; durch "/" getrennte Ordner-Namen, sollte einer der Ordner (noch) nicht existieren, wird er automatisch angelegt.

Prioritaet

"Prioritaet"-Attribut des Root-Elements gibt die zuzuweisende Priorität an (je größer desto eher wird das Objekt benutzt); sollte im Allgemeinen aber nicht benutzt werden, es wird automatisch der Standard (-50, gegenüber -100 für Automatik-Strukturelemente) gesetzt.

Tid

"Tid"-Attribut des Root-Elements gibt den zu benutzenden Tid-Code ("Klartext-Identifizier") an; sollte im Allgemeinen aber nicht benutzt werden, es wird automatisch ein konsistenter Tid vergeben.

Beschreibung

Eigenes Kind-Element des Root-Elements; der Text wird als Beschreibung des Objektes benutzt.

BOTyp

Eigenes Kind-Element des Root-Elements; sein "Name"-Attribut gibt an, Objekte welchen Typs mit diesem Strukturelement bearbeitet bzw. angezeigt werden sollen.

Parameter

Eigenes Kind-Element des Root-Elements; sein Text wird als "Parameter"-Wert für das Strukturelement verwendet und enthält die weitergehende Definition, insb. für Formulare.

Formular

(Nur für Schablonen) Eigenes Kind-Element des Root-Elements; sein "Name"-Attribut gibt an, welches Formular von der Schablone für die Bearbeitung des neuen Objekts benutzt werden soll.

Gruppen, Polymorphic

Diese Elemente/Attribute werden z.Zt. noch nicht berücksichtigt. Alle Strukturelemente werden automatisch initial (nur) der Admins-Gruppe zugewiesen.

Beispiel für vorgebautes Formular (weitere finden sich im oben erwähnten Verzeichnis):

```
<Formular Name="$R{_Benutzer} (Vorgebaut)" ElterPfad="Admins/MyTISM/Benutzerverwaltung">
  <Beschreibung>Vorgebautes, aufger&auml;umteres Benutzer-Formular, mit Gruppierungen der Alarm-
    und Benachrichtigungsinfos und einfacherer, halbautomatischer
    Benachrichtigungskonfiguration.</Beschreibung>
  <Parameter>
    <TabbedView tabPlacement="TOP">
      <!-- ... mehr Definition ... -->
    </TabbedView>
  </Parameter>
  <BOTyp Name="Benutzer"/>
  <Gruppen>
    <Gruppe Name="Benutzer"/>
  </Gruppen>
</Formular>
```

Für spezifische Projekte gibt es (ausser der manuellen Synchronisation) z.Zt. noch keinen entsprechenden Mechanismus, um projektspezifische Strukturelemente automatisch zu laden. Bei Bedarf ließe sich das aber entsprechend ergänzen.



Anmerkung

Wenn ein vorgebautes Strukturelement geändert wurde oder neu hinzugekommen ist, muss vor dem Start des Servers die Datei .checked-initialdata gelöscht werden, damit die Änderungen wirksam werden (in Zukunft wird das wohl automatisch geschehen).

Kapitel 2. Sprachunterstützung und Internationalisierung

2.1. Einführung

MyTISM bietet eine durchgehende Unterstützung für verschiedene Locales, sowohl für die Übersetzung von Texten und Namen als auch für Ein- und Ausgabe von Zahlen, Daten, etc.

2.2. Wo wird Mehrsprachigkeit unterstützt und wie benutze ich sie?

Neben der direkten Benutzung in Programmcode mittels der Methoden `L10n.msg()` bzw. `L10n.applyL10n()` gibt es weitere Stellen an denen die Mehrsprachigkeit unterstützt wird.

Im GUI-Client Solstice, an vom Benutzer bearbeitbaren Stellen:

- In den Benutzer-Login-Scripten.
- In den XML-Definitionen für Plugins in den Benutzer-Voreinstellungen.
- In den XML-Definitionen für Defaults in Benutzer-Voreinstellungen.
- In Report-Definitionen und in den Parametern von Formularen, Schablonen und Lesezeichen.

Im GUI-Client Solstice, interne Funktionalität:

- Ausgabe von Name und ElterPfad von Benannts im Navigationsbaum bzw. in `PolymorphicTemplateSelectionTreeModel`.

In diesen "Texten" können Platzhalter `$_R{key}` eingefügt werden. Diese werden dann automatisch vor der "Benutzung" durch zum aktuellen Locale passende Texte ersetzt.

2.3. Wie wird die konkrete Zeichenkette für einen Schlüssel gefunden?

`L10nPackProviderI` (z.Zt. `de/ipcon/tools/L10n` und `de/ipcon/db/AbstractClient`) halten benannte `L10nPacks` bereit, welche die diversen Textbausteine in unterschiedlichen Sprachen enthalten, gruppiert mittels entsprechender Schlüssel für jeden Textbaustein.

Beim Auflösen von `$_R{key}`-Platzhaltern z.B. im Formularcode (und ebenso beim direkten Aufruf der `L10n.msg()`-Methoden) wird eine - je nach Aufrufart bzw. -ort unterschiedliche, und bei den im vorherigen Abschnitt aufgeführten Stellen, automatisch zusammengestellte - Liste der an dieser Stelle beteiligten bzw. relevanten Objekte übergeben (Z.B. für Formulare das Formular-Objekt selbst).

Mittels dieser übergebenen Objekte wird nun eine Liste von relevanten `L10nPacks` erstellt, in denen

mittels des Schlüssels "key" nach den angeforderten Textbausteinen gesucht wird. Wird ein zum Schlüssel passender Textbaustein gefunden, wird seine zur gewünschten Sprache passende Version zurückgeliefert.



Anmerkung

Schlüsselnamen dürfen nur Buchstaben, Zahlen, '_', '-', '.', '~' und '/' enthalten.

2.4. Welche L10nPacks gibt es und wie sind diese organisiert? Wie wird bestimmt, welche L10nPacks nach Texten durchsucht werden?

Die Benennung bzw. Hierarchie der L10nPacks folgt in der Regel der Klassen- bzw. Paketstruktur der Java-Klassen. In den meisten Fällen bestimmen die Java-Klassen der beteiligten Objekte und die Java-Pakete denen diese angehören die zu durchsuchenden L10nPacks.



Anmerkung

L10nPack-Namen dürfen nur Buchstaben, Zahlen, '_', '-' und '.' enthalten, wobei '.' das Trennzeichen zum Aufsplitten der Namen ist.

Beispiel: Ein Objekt der Klasse "de.ipcon.form.FText" will den Textbaustein mit Schlüssel "eineNachricht" in der aktuellen Sprache ausgeben. In diesem Fall werden - sofern vorhanden - die L10nPacks "de.ipcon.form.FText", "de.ipcon.form.FPanel" (FText leitet sich von FPanel ab), "de.ipcon.form", "de.ipcon" und "de" in dieser Reihenfolge nach einer zum Schlüssel passenden Version des Textes durchsucht.

2.5. Welches sind die "beteiligten bzw. relevanten Objekte"?

Dies ist unterschiedlich und hängt davon ab, wo und wie die L10n-Funktionalität genutzt wird, z.B.:

- Beim direkten Aufruf von `L10n.msg()` wird normalerweise automatisch die aufrufende Klasse ermittelt und als das (einzige) beteiligte Objekt übergeben.
- Bei Benutzung von `$_R{key}`-Platzhaltern im Parameter von Formularen, etc. wird die Klasse des im Formular, der Schablone oder im Lesezeichen dargestellten BOs sowie die Klasse des Strukturelements (Formular.class, etc.) selbst übergeben.

Im Normalfall wird die Liste der L10nPacks für eine Klasse einfach nach der im vorherigen Abschnitt beschriebenen Methode (Klassen + Pakete) erstellt. Ist für eine Klasse aber ein sog. `L10nPathCompiler` beim L10n registriert, so bestimmt dieser, welche L10nPacks für die entsprechende Klasse in die Liste aufgenommen werden (siehe z.B. "FormularPathCompiler" in `Formular.nrx`).

Genauere Informationen finden sich bei den entsprechenden Aufrufen von `L10n.applyL10n()` bzw. `L10n.msg()` (bei letzterem nur selten, da dort fast nie ein expliziter "path" übergeben wird und fast immer die oben erwähnte Automatik benutzt wird) im Quellcode von MyTISM. `L10n.compilePath()`

enthält weitere Informationen darüber, wie die Liste der zu durchsuchenden L10nPacks zusammengestellt wird.

2.6. Wo kommen die (Daten der) L10nPacks her?

Die Texte/Daten für die L10nPacks werden z.Zt. aus zwei Quellen gezogen. Die L10n-Klasse lädt ihre Daten aus Dateien `.../resources/*.properties` die in den Quellcode-Verzeichnissen liegen und beim Bauen ebenfalls in die JARs eingebunden werden. Daneben lädt der Server noch die in der Datenbank befindlichen L10nBundles in seinen L10nCache.

Die Texte der `*.properties`-Dateien werden alle "von Hand" angelegt und bearbeitet. Die L10nBundles, etc. werden teilweise automatisch generiert, können aber auch von Hand angelegt und bearbeitet werden

(FIXME wobei dabei zu beachten ist, dass automatisch angelegte L10nEntries und L10nResources - aber nicht L10nBundles - teilweise auch automatisch gelöscht werden, wenn z.B. die entsprechenden Entities oder Attribute weggefallen sind).

Für jede im Schema definierte Entität, Beispiel `"de.ipcon.db.core.Benannt"`, werden einige L10n-Daten automatisch angelegt. Diese sind im Beispiel zuerst das L10nBundle `"de.ipcon.db.core"` mit den L10nResources `"_Benannt"` sowie `"_Benannt-s"`, L10nResources für die Ordner, also hier `"Interna"` (FIXME ggf. weitere) sowie L10nResources für jedes Attribut der Entität also hier `"Name"`, `"Beschreibung"` usw.

Desweiteren wird für jede Entität noch ein eigenes Bundle, im Beispiel `"de.ipcon.db.core.Benannt"`, angelegt und dort werden ebenfalls noch einmal wie oben für alle Attribute L10nResources angelegt.

Genauere Informationen finden sich in `L10nBundle.initEnvironment()` und den dort benutzten Methoden.

2.7. Wichtige Klassen

`de.ipcon.tools.L10n`

Die zentrale Klasse. Enthält u.A. die `msg()`-Methoden die zu einem gegebenen Key die zum gewünschten/aktuellen Locale passende Version der entsprechenden Zeichenkette liefern. Enthält auch diverse Methoden um Format-Objekte zum formatieren von Zahlen, Daten, etc. zu erhalten.

`de.ipcon.db.core.L10nBundle`

Sammlung von L10nResources.

`de.ipcon.db.core.L10nResource`

Entspricht grob einer Zeichenkette welche in unterschiedlichen Sprachen ausgegeben werden können soll. Hat einen oder mehrere L10nEntries.

`de.ipcon.db.core.L10nEntry`

Konkrete Version der Zeichenkette für ein bestimmtes Locale (grob: eine Sprache).

2.8. Eingabe von L10n-Daten

FIXME Stichworte

neues Bundle anlegen (de.venice) bzw. in einem bestehenden Bundle (de.venice.bo) was hinzufuegen => schauen, dass das Bundle auch Preloaded wird und auch die PfadPos setzen (0). Wenn trotzdem ein neuer Eintrag nicht direkt gefunden wird, dann kann es noetig sein den Server durchzustrarten. Das kann der Fall sein, wenn das Bundle erstmalig auf Preload und/oder PfadPos gesetzt wird.

Einfaches Hochkomma muss "escaped" werden => doppelt schreiben

\$R-Tags in Formularen, etc.: Suche nach title=", label=", text=".

Kapitel 3. Die Formularengeine des Solstice Clients

3.1. de.ipcon.form

Nachfolgende Dokumentation behandelt den Aufbau der eines sogenannten Formulars im de.ipcon.form Package des MyTISM Frameworks. Sie soll den Entwickler in die Lage versetzen, Wünsche des Anwenders an die grafische Oberfläche umzusetzen. Im Gegensatz zu Web-Oberflächen sind diese für den Anwender viel effektiver und schneller bedienbar als die etwas generischeren und graphisch meist viel ansprechenderen, aber dennoch umständlich zu bedienenden Web-Oberflächen. Allerdings ist das Abstraktionsniveau im grafischen Client etwas geringer, um schneller zum Ergebnis zu kommen - manchmal ist es besser, den ein oder anderen Wunsch eines Anwenders zugunsten einer saubereren oder aber auch für eingeschränkte Benutzer (sei es technisch (niedrige Farbe, langsamer Rechner) oder auch körperlich (Farbenblindheit, Kurzsichtigkeit)) bedienbaren Lösung abzulehnen.

3.1.1. Hintergrund

Das Formularframework bzw. die Engine, die die Formulare aufbaut, hat zwei Ziele: Flexible Formulare und eine flexible Verwaltung dieser Formulare. Das HTML-Format bzw. dessen Vorgänger SGML bzw. XML haben mit ihrem Markup-Konzept einen entscheidenden Denkanstoß zur Entwicklung der jetzigen Implementation geliefert. Angereichert mit einer Meta-Ebene, die aus dem Datenbank-Backend MyTISM kommt und somit einen schnellen und effizienten Zugriff auf die Formulare gestattet sowie die Synchronisation der Formulare realisiert. Im folgenden werden die Objekte einzeln ausführlich vorgestellt, ihre Eigenschaften und ihre Verwendung dokumentiert. Alle Objekte außer den Lesezeichen haben die Eigenschaft, in bestimmten Kontexten als Auswahl zur Verfügung zu stehen, sei es, um ein neues Objekt zu erzeugen oder ein bestehendes anzuzeigen. Das Verfahren, diese Auswahl zu generieren, wird ebenfalls beschrieben.

3.1.2. Das Formular-Objekt

Das Formular-Objekt hat die Aufgabe, eine Eingabemaske für ein Objekt einer bestimmten BO-Klasse zu beschreiben. Das de.ipcon.form Package enthält die notwendigen Methoden, um eine solche Eingabemaske bestehend aus Oberflächenelementen wie Textfelder, Populisten und ähnlichem zu erzeugen und verwalten. Es stellt eigentlich das wichtigste und gleichzeitig das komplizierteste Objekt des Formularframeworks dar.

3.1.2.1. Eigenschaften

Das Formular-Objekt hat im wesentlichen folgende Eigenschaften:

1. Name: Eine klar abgrenzende Bezeichnung, die auch im Kontextmenu des jeweiligen Objektes erscheint.
2. Beschreibung: Eine etwas ausführlichere Beschreibung, durchaus als Platz für Bemerkungen wie den Verweis auf spezielle Versionen oder Spezifika. Sie wird dem Benutzer nicht in der GUI präsentiert und ist ausschließlich den Entwicklern vorbehalten.

3. Elter: Ein Verweis auf die ID des Strukturelements (meist ein Ordner oder eine Gruppe), unter dessen Repräsentation im Menubaum dieses Element absortiert wird. Wird gesetzt beim Drag'n'Drop im NavigationTree in der GUI.
4. IstAutomatik: Ein Wahrheitswert, der anzeigt, ob das Formular direkt aus dem Formulargenerator stammt. Falls Sie ein Formular abändern, achten Sie bitte darauf, daß dieser Wert false ist, sonst wird beim nächsten Schema-Update das Formular neu erstellt; der NavigationTree setzt es beim 'Move' dieses Flag automatisch auf false, um eine Fehlbedienung zu vermeiden.
5. Parameter: Hier steckt der Source des eigentlichen Formulars. Im folgenden wird der Inhalt dieser Eigenschaft ausführlich beschrieben.
6. BOTyp: Ein Verweis auf den Typ des BO (selbst natürlich ebenfalls ein BO), welches mit diesem Formular angezeigt werden kann. [fixme: Polymorphie?]
7. Gruppen: Ein Mehrfach-Verweis auf die Gruppen, die dieses Formular benutzen sollen.
8. Schablonen: Ein Mehrfach-Verweis auf die Schablonen-Objekte, die direkten Gebrauch von diesem Formular machen.
9. Priorität: ein 32bit signed Integer, der die Priorität des Formulars im Falle einer mehrfachen Auswahl von Formularen für ein Objekt festlegt und damit die Präferenz in diesem Fall festlegt.

3.1.2.2. Auswahl

Die Auswahl eines Formulars wird aufgrund folgender Regeln getroffen:

1. Zunächst werden alle passenden (gleicher BOTyp [fixme: Polymorphie]) Formulare erfragt, deren Priorität gesetzt ist und einer der eigenen Gruppen zugeordnet ist. Beim Benutzer Admin werden als Ausnahme auch diejenigen Formulare mit einbezogen, die keine Priorität haben (diese Mechanismus wird in einer der nächsten Versionen ausgebaut und ist damit als obsolet deklariert!).
2. Diese passenden Formulare werden der Priorität nach absteigend geordnet und dem Benutzer ggfs. per Kontextmenu zur Verfügung gestellt. Ein Doppelklick oder adäquate Aktion öffnet das nach dieser Sortierung am höchsten priorisierte Element. Die momentane Implementation ersetzt Formulare gleicher Priorität ohne eine deterministische (oder vielmehr eine dokumentierte Deterministik) oder vorhersehbare Präferenz. Daher bitte ich unbedingt auf eine klare Priorisierung zu achten. Der Zahlenraum der Priorität bietet genügend Spielraum: ca. -2 bis 2 Milliarden.

3.1.2.3. Definition

3.2. Fehler und Ursachen

Einige Fehler denen ich bei der MyTISM-Entwicklung schon begegnet sind und deren Ursachen bzw. Lösungsmöglichkeiten:

3.2.1. Compiler-Meldung "Object cannot be null"

BOs brauchen einen Konstruktor (ohne Argumente); es wird nicht automatisch einer gebaut oder der der Superklasse benutzt. BO-Klassen dürfen nicht "abstract" sein

3.2.2. bi-Tabelle kann nicht erstellt werden (nachdem die Datenbank gedroped und recreated wurde)

".checked*" -Dateien im Projektverzeichnis löschen.

3.2.3. Compiler-Meldung "Object bla is null but shoudn't" (sic)

Beispiel: "Zustellversuch" hängt an "Sendeauftrag". Neuer Zustellversuch wurde angelegt, in Transaction included und an Sendeauftrag angehängt. Dumm nur: Sendeauftrag war nicht in Transaction included! FIXME: Hmm ... das war aber wohl doch nicht das Problem :-)

Kapitel 4. BOMengen

4.1. Grundlagen

BOMengen bieten, ähnlich wie BOMasken, eine Möglichkeit, BOs zu Gruppen bzw. Mengen zusammenzufassen.

4.2. Gemeinsame Eigenschaften aller BOMengen

Alle BOMengen haben bestimmte Eigenschaften gemeinsam:

Name

Der Name oder Titel einer BOMenge sollte die BOMenge kurz und prägnant benennen. Der Name kann frei gewählt werden und kann z.B. bei der Anzeige der BOMengen im zugehörigen Lesezeichen benutzt werden. Es ist sehr sinnvoll, jedoch keineswegs zwingend, dass unterschiedliche BOMengen unterschiedliche Namen haben :-)

Beschreibung

Die Beschreibung kann einen längeren Kommentar bzw. eine längere Beschreibung der BOMenge beinhalten.

Entitaet

BOMengen umfassen immer nur BOs einer bestimmten Entitaet, d.h. eines bestimmten Typs bzw. Klasse von BOs. Welche dieses sind, kann hier angegeben werden.

KomplementBilden

Durch Setzen von KomplementBilden kann auf einfache Weise die Komplementärmenge gebildet werden, d.h. die Menge soll alle BOs umfassen, die sonst *nicht* zur Menge gehören.

BOs

Diese virtuelle Relation beinhaltet alle BOs die in der BOMenge enthalten sind.

OQLString

Die Auswahl der BOs erfolgt originär mit der hier angezeigten OQL-Abfrage. Das Attribut ist ebenfalls virtuell, die Anfrage wird dynamisch vom BOMengen-Objekt generiert.

WhereClause

Die WhereClause beinhaltet die Bedingung, die die BOMenge zur obengenannten OQL-Abfrage "beisteuert". Auch dieses Attribut ist virtuell.

BOMengenVerknuepfungen

Durch Verknüpfung unterschiedlicher BOMengen ist es möglich, komplexere Abfragen und damit spezifischere Mengen zu bauen; diese Relation beinhaltet alle `BOMengenVerknuepfungen`, an denen die BOMenge beteiligt ist. Dieses Thema wird weiter unten noch ausführlicher behandelt.

4.3. Typen

BOMenge

Die Basisklasse und einfachste mögliche BOMenge; umfasst einfach alle BOs der angegebenen Entität ohne irgendwelche speziellen Kriterien zu berücksichtigen; Ist KomplementBilden aktiviert enthält die Menge *keine* BOs, d.h. entspricht der leeren Menge.

ExpliziteBOMenge

Eine BOMenge, der ein fester Satz von BOs "manuell" (was natürlich auch z.B. durch ein Programm erfolgt sein kann) zugeordnet wurde. Die Menge der umfassten BOs ändert sich bei diesem Typ nur durch direktes Hinzufügen oder Entfernen von BOs.

OQLBOMenge

Diese BOMenge hat eine fest gesetzte OQL-Where-Clause, die angibt, welche BOs enthalten sind.

AttributBedingungBOMenge

Diese BOMenge beinhaltet alle BOs, bei denen der Wert des angegebenen Attributes (def. durch Attributname) einen zur angegebenen Bedingung (definiert durch einen Operator und möglicherweise einen Vergleichswert) passt. Hierbei kann es sich um Bedingungen wie "beginnt mit A" oder ">= 5" oder "Flag ist gesetzt" handeln. Für die verschiedenen Datentypen existieren Unterklassen dieser Klasse.

BOMengenVerknuepfung

Dieser "Meta-Typ" erlaubt es, die Elemente zweier vorhandener BOMengen zu kombinieren. Standardmässig gehört ein BO zur Menge, wenn es zu mindestens einer der beiden angegebenen Mengen gehört (d.h. mindestens die Kriterien einer Menge auf das BO zutreffen; "Vereinigung"). Ist MitUndVerknuepfen gesetzt, muss das BO in beiden Mengen enthalten sein (d.h. die Kriterien beider Mengen müssen zutreffen; "Schnitt") damit es aufgenommen wird.

Kapitel 5. Synchronisation der Strukturelemente

5.1. Das Formular "DateiSystemSync"

Das Benutzerhandbuch enthält bereits einen Abschnitt zu diesem Thema; ggf. sollten diese zusammengeführt werden.

FIXME! (werde später noch ein bisschen was dazu schreiben - sw) - anhand was wird rein- bzw. rausgesynct - Konventionen Dateiname - wofuer Tid - ...

Kapitel 6. Volltextsuche

Die Volltextsuche erlaubt die einfache und schnelle Suche nach gegebenen Suchbegriffen über alle in der MyTISM-Datenbank gespeicherten Objekte. Informationen zur allgemeinen Konfiguration und Bedienung finden sich in der MyTISM-Benutzerdokumentation [<http://www.mytism.de/docs/user.html#volltextsuche>]. In diesem Kapitel befinden sich noch einige nur für Entwickler interessante Informationen, insb. zur Konfiguration der Suche im Schema und der Benutzung von Volltextsuche-Queries in Programm- oder Skriptcode.

6.1. Konfiguration im Schema

6.1.1. Berücksichtigte Daten

Standardmässig werden, mit wenigen Ausnahmen, die BOs aller Entitäten für die Volltextsuche aufbereitet. Von diesen BOs werden standardmässig die Inhalte alle Attribute, wiederum mit einigen Ausnahmen, in den Suchindex aufgenommen.

6.1.1.1. Berücksichtigte Entitäten



Anmerkung

vgl. [de/ipcon/db/fulltext/compass/SchemaMappingBuilder.isEntityIgnored\(\)](http://de/ipcon/db/fulltext/compass/SchemaMappingBuilder.isEntityIgnored())

Explizit *immer* ausgeschlossen werden die BOs nicht persistenter Entitäten, sowie die BOs der Entitäten `BT`, `BP` und `BX`.

Durch explizite Angabe von `indexed="no"` im Schema ist es möglich, weitere Entitäten von der Indexierung auszunehmen.

Beispiel:

```
<Entity name="InterneEntitaet" extends="B0" plural="InterneEntitaeten">
  <fulltext indexed="no"/>
  <attr name="KryptischerString"/>
</Entity>
```

FIXME Infos zum "Cascading", (Nicht-)Indexierung von Unterklassen

6.1.1.2. Berücksichtigte Attribute



Anmerkung

vgl. [de/ipcon/db/fulltext/compass/SchemaMappingBuilder.isAttributeIgnored\(\)](http://de/ipcon/db/fulltext/compass/SchemaMappingBuilder.isAttributeIgnored())

Standardmässig ausgeschlossen werden die Daten von

- Relationen-Attributen (sowohl Single als auch Many)
- Attributen mit (Java-)Typ `Boolean`, `Date` oder `Number`

- virtuelle Attribute

Durch explizite Angabe im Schema ist es jedoch möglich, entsprechende Attribute von bestimmten Entitäten doch in den Index aufzunehmen. Andererseits können auch normalerweise indexierte Attribute explizit von der Indexierung ausgenommen werden.

Beispiel:

```
<Entity name="InterneEntitaet" extends="B0" plural="InterneEntitaeten">
  <attr name="Name">
  <attr name="InteressantesDatum" type="DateTime">
    <fulltext indexed="yes"/>
  </attr>
  <attr name="KryptischerString">
    <fulltext indexed="no"/>
  </attr>
</Entity>
```

Wird für Relationen-Attribute (sowohl Single- als auch Many-Relationen) `indexed="yes"` angegeben, ist das Resultat, dass Objekte der "Elter"-Klasse (die, die das Relationen-Attribut enthält) auch als Suchtreffer gefunden werden, wenn ein Suchbegriff "nur" auf eines der "Kind"-Objekte (die in der Relation enthaltenen Objekte) zutrifft.

Beispiel:

```
<Entity name="Elter" extends="B0" plural="Eltern">
  <attr name="Name">
  <attr name="Kinder" type="Kind" relation="1-n">
    <fulltext indexed="yes"/>
  </attr>
</Entity>

<Entity name="Kind" extends="B0" plural="Kinder">
  <attr name="Name">
</Entity>
```

Es existiert ein Elter "Elter1" mit Kindern "Kind1" und "Kind2". Wird jetzt z.B. im "Eltern"-Lesezeichen nach "Kind1" gesucht, so wird das Objekt "Elter1" als Ergebnis geliefert, obwohl der Suchbegriff "Kind1" eigentlich nur in einem der "Kinder"-Objekte vorkommt.

6.1.2. Weitere Einstellungen im Schema

6.1.2.1. analyzed



Anmerkung

vgl. [de/ipcon/db/fulltext/compass/SchemaMappingBuilder.buildScalarConfig\(\)](http://www.compass-project.org/docs/2.2.0/reference/html/core-searchengine.html#core-searchengine-analyzers) sowie das entsprechende Kapitel in der Compass-Dokumentation [<http://www.compass-project.org/docs/2.2.0/reference/html/core-searchengine.html#core-searchengine-analyzers>]

Für einzelne Attribute kann im Schema definiert werden, ob die entsprechenden Inhalte bei der Indexierung "analysiert" werden sollen oder nicht.

Ein Text wie "Dies ist ein Attributwert" wird normalerweise nicht in dieser Form im Index abgelegt,

sondern in seine einzelnen Bestandteile (normalerweise "Wörter", d.h. durch Whitespace abgetrennte Tokens) aufgeteilt. Auch werden einige sehr häufig vorkommende Wörter (sog. "Stopwords") entfernt.

Durch diese Behandlung ist es möglich, dass bei der Suche nach z.B. "Dies" das Objekt mit dem obigen Attributwert gefunden wird.

Wäre der Wert nicht "analysiert" worden, so wäre nur die gesamte Zeichenkette genau in dieser Form im Index abgelegt und das Objekt würde nur bei Eingabe genau von "Dies ist ein Attributwert" (oder ggf. noch bei Benutzung von Platzhaltern oder Ähnlichkeitssuche) gefunden, nicht aber nur bei Eingabe von "Dies".

Standardmässig werden alle Attribute mit (Java-)Typ `String` "analysiert"; alle anderen Attribute (insb. z.B. Zahlen) nicht. Im Normalfall ist diese Einstellung wohl sinnig; in Einzelfällen (z.B. vielleicht wenn es sich um eine Bezeichnung handelt, die nur genau in der eingegebenen Form gefunden werden soll) kann das Verhalten aber durch eine explizite Angabe beim Attribut geändert werden.

Beispiel:

```
<Entity name="InterneEntitaet" extends="B0" plural="InterneEntitaeten">
  <attr name="Name">
    <attr name="Typenbezeichnung">
      <fulltext analyzed="no"/>
    </attr>
  </attr>
</Entity>
```

6.1.2.2. boost

Sowohl für Entitäten als auch für einzelne Attribute kann ein "Boost"-Wert im Schema angegeben werden. Dieser dient dazu, einen Treffer für die entsprechende Entität oder das entsprechende Attribut höher oder niedriger zu bewerten und damit im Ranking der Suchergebnisse weiter nach vorne oder hinten zu plazieren. Da jedoch z.Zt. in MyTISM kein Ranking von Suchergebnissen benutzt wird, ist die Angabe dieses Wertes z.Zt. weder erforderlich noch sinnvoll.