MyTISM - Ein Datenbank- und Anwendungs-Framework

Inhaltsverzeichnis

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.

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 https://mytism.de/#contact.

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.

Schema-Definition

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

Einleitender Tag: `<Schema BEFEHLE>`

Schliessender Tag: `</Schema>`

Table 1. Schema

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

Include

Der Include-Befehl bietet, analog zu dem aus manchen Programmiersprachen bekannten Konstrukten, die Möglichkeit an, eine existierende Schemadefinition aus einer weiteren Datei mit einzubinden.

So lassen sich z.B. in der Modellierung von Entities thematisch getrennte Elemente in verschiedenen Schemadefinitionsdateien ablegen, und über `include`-Befehle in einer Datei zusammenfügen.

Tag: `<Include BEFEHLE />`

Table 2. Include

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

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

Folder

FIXME: Einleitende, erklärende Worte

Tag: `<Folder Foldername />`

Table 3. Folder

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

Entity

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

Einleitender Tag: `<Entity BEFEHLE>`

Schliessender Tag: `</Entity>`

Table 4. Entity

Befehl	Beschreibung	Beispiel
name	Name der Entität, quasi unbegrenzter Länge	<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 Entitäten ableiten; Default: false	<Entity name="XBuchungsKonto" plural="XBuchungsKonten" extends="CBO" abstract="true">

Befehl	Beschreibung	Beispiel
noAbstractWarning	In Verbindung mit dem 'abstract'-Attribut verwendbarer Befehl; unterdrückt die Warnung, dass die Entität abstrakt ist und erlaubt (eingeschränkt) , dass **dennoch** BOs von dieser Entität angelegt werden können: Dies ist dann sinnvoll, wenn es sich um nicht für den Endnutzer bestimmte technische BOs handelt, die zwar programmatisch angelegt werden sollen, jedoch nicht via MyTISM-Formular. Diese BOs können in MyTISM per Lesezeichen abgefragt und per Formular angesehen werden, es gibt aber keine Schablone zum Erzeugen eines neuen BOs.	ignoreReverseRelations
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

Unter-Element "ui" von Entity

FIXME: Einleitende, erklärende Worte

Table 5. Unter-Element "ui" von Entity

Befehl	Beschreibung	Beispiel
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"/>
defaultSorting	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. Die Mehrfachsortierung erfordert eine zusätzliche Lizenz.	<ui defaultSorting="Name:ASC Beschreibung:DESC Position.ASC Eigenschaft:ASC Eigenschaft.Name:ASC">

Befehl	Beschreibung	Beispiel
autotrim	Nur sinnvoll für Entity-Attribute vom Typ "String". Whitespace am Anfang und Ende der Eingabe wird *in der GUI* automatisch entfernt. Standard für String-Attribute ist "true", mit `autotrim="false"` kann die automatische Entfernung aber falls gewünscht für einzelne Attribute deaktiviert werden. Hat *keine* Auswirkungen auf Werte die mittels Code gesetzt werden!	<ui autotrim="false"/>

Unter-Element "lookup" von Entity

FIXME

Table 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.	<lookup defaultProperty="Name"/>
defaultSubstring	Ob die Suche exakt sein soll oder ein Substring-Match in der defaultProperty schon ausreicht; Default: true	<lookup defaultSubstring="true"/>
defaultCaseSensitive	Ob die Suche Groß-/Kleinschreibung beachten soll; Default: false	<lookup defaultCaseSensitive="true"/>

Unter-Element "code" von Entity

FIXME

Table 7. Unter-Element "code" von Entity

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

Unter-Element "db" von Entity

FIXME

Table 8. Unter-Element "db" von Entity

Befehl	Beschreibung	Beispiel
persistent	Ob die BOs der Entität persistiert (gespeichert) werden sollen, Default: true	<db persistent="false"/>
name	Ein optionaler, abweichender, ggfs. kürzerer (Tabellen-)Name für diese Entität (case-insensitive). Wird hier keine Angabe gemacht, so wird als Default der Entitätsname in Kleinbuchstaben verwendet. Falls der Entitätsname länger als 63 Zeichen ist, sollte hier ein kürzerer Name angegeben werden, da aufgrund von Limits in der Datenbank für Tabellennamen in MyTISM max. 63 Zeichen erlaubt sind. Ferner sind auch keine Unterstriche in Tabellennamen erlaubt, da diese nur für interne junction tables für n-m-Relationen verwendet werden dürfen.	<db name="MyShorterTableNameNoUnderscore"/>
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"/>

Befehl	Beschreibung	Beispiel
streamResource	Gibt einen Tipp, ob mit Instanzen dieses Typs normalerweise ein BLOB (BinaryLargeOBject) verknüpft sein kann (verhindert bei "false" allerdings nicht, dass man ein BLOB verknüpfen kann); wird aktuell im Sync von BOs ins Dateisystem abgefragt, um zu entscheiden, ob zusätzlich auch ein Datei-Export für das verknüpfte Objekt durchgeführt werden muss, z.B. für Bilder an einem Report-Objekt; Default: false	<db streamResource="true"/>
noStreamResourceHistory	Legt fest, ob Sicherheitskopien der BLOBs beim Ersetzen derselben im Dateisystem des Servers angelegt werden sollen ("false") oder nicht ("true"); Default: false	<db noStreamResourceHistory="true"/>
forbidDirectChanges	Ob direkte Änderungen an BOs diesen Typs vorgenommen werden dürfen oder ob nur der Server diese Objekte modifizieren darf; Default: false; solche Objekte werden nicht zwischen verschiedenen MyTISM-Nodes synchronisiert/übertragen, sind also nur auf der erstellenden Instanz verfügbar	<db forbidDirectChanges="true"/>

Unter-Element "report" von Entity

FIXME

Table 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"/>

Befehl	Beschreibung	Beispiel
fontSizeNormal	Die Standard-Schriftgrüße für automatisch generierte Reports; Default: 9	<report fontSizeNormal="10"/>
fontSizeBig	Die große Schriftgrüße für automatisch generierte Reports; Default: 16	<report fontSizeBig="14"/>

Unter-Element "export" von Entity

Durch Angabe dieses Elements können ein oder mehrere automatisch bereitgestellte Exports konfiguriert werden.

Diese Funktionalität ist noch in der Entwicklung und wird später den Struktur-Sync, den Initialdaten-Import beim Serverstart und ggfs. auch manuell gebaute Exports in Fremdsysteme ersetzen.

Weiterhin sind neue Features wie etwa das Kopieren von Objekten via Clipboard von einem MyTISM-System in ein gänzlich anderes angedacht, sofern die Entitäten der zu kopierenden Objekte in beiden System existieren (z.B. für Core-Entitäten wie etwa Strukturelemente, aber ggfs. auch für andere).

Table 10. Unter-Element "export" von Entity

Befehl	Beschreibung	Beispiel
name	Der Name des Exports. Pflichtangabe.	<export name="Initialdaten"/>
primaryKey	Der primäre Schlüssel für den Export. Verwendet für Referenzen aus anderen Exports. Pflichtangabe.	<export primaryKey="ISOCode"/>
mode	Der Modus für den Export, der bestimmt, ob pro Objekt eine Datei erzeugt wird oder alle Objekte in eine einzige Datei geschrieben werden; erlaubte Werte: SINGLE, LIST. Pflichtangabe.	<export mode="SINGLE"/>

Beispiel:

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

Attribut

FIXME: Einleitende, erklärende Worte

Tag für Attribute: `<attr BEFEHLE />`

Tag für virtuelle Attribute: `<vattr BEFEHLE />`

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

Table 11. 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"
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 `createInDetailView`; Default: false	dependent="true"
itemProperty	Ermöglicht manuelle Sortierung per Pfeil-Buttons im Formular. Anzugeben ist das Attribut (der anhängenden Entität), nach dem sortiert werden soll.	itemProperty="Position"
shared	Wird die Referenz innerhalb der Entität geshared; Default: false	shared="true"

Befehl	Beschreibung	Beispiel
default	Für diverse Attribut-Typen lässt sich ein Default-Wert vorgeben (z.B. für `Datetime`: `new java.util.Date()`; `Boolean`: `Boolean.TRUE`; `Decimal`: `#,##0.00`	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"

Unter-Element "ui" von Attribute

FIXME

Table 12. 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 werden sollen (normalerweise wird auf dem Formular-Reiter der Many-Relation nur eine FTable gebaut); Default: false	<ui createInDetailView="true">
mandatory	FIXME Definition als "Pflichtfeld". Z.Zt. nur partiell unterstützt, nicht z.B. von Solstice; Default: false	<ui mandatory="true">

Befehl	Beschreibung	Beispiel
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	Hier kann direkt eine Filter-Clause (s. Bsp. 1) oder alternativ ein `vattr` definiert werden, dessen `getter()` eine Filter-clause definiert (s. Bsp. 2). Die Clause wirkt als nicht-interaktiver Filter innerhalb einer Query: Wird das Attribute z.B. als Popup-property verwendet, ist dieser Filter automatisch aktiv und schränkt die Auswahl ein. Die übergeordnete Query kann durch weitere, im Strukturelement definierte Filter erweitert werden.	• Bsp.1: `<ui selectionFilter="('BOTyp.Id='Bot.Id)"/>` • Bsp.2: `<ui selectionFilter="GeschaeftsbereichFilter" editMode="linkOnly"/>`

Unter-Element "lookup" von Attribute

FIXME

Table 13. 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">

Unter-Element "report" von Attribute

FIXME

Table 14. 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">

Befehl	Beschreibung	Beispiel
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">

Unter-Element "virtual" von Attribute

FIXME

Table 15. Unter-Element "virtual" von Attribute

Befehl	Beschreibung	Beispiel
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">

Befehl	Beschreibung	Beispiel
preCachingHook	Der Name einer Methode, die auf einen im Cache zu speichernden, skalaren Wert appliziert werden soll, bevor dieser in den Cache befördert wird. Die Methode wird typischerweise an der Klasse der Entität, die das vattr definiert, implementiert (alternativ relativ weit "oben", wenn sie mehrmals verwendet werden soll im Schema, also z.B. an der TBO-Klasse). Die Methodensignatur dieser Methode muss zwei Parameter akzeptieren, als erstes den Attributnamen und als zweites den zu cachenden Wert im richtigen, schema-definierten Typ, und ebendiesen Typ zurückgeben.	<virtual preCachingHook="compactBigDecimal"> sowie eine Methode `method compactBigDecimal(attribute = String, value = BigDecimal) returns BigDecimal`

Aggregats-Funktionen:

Aggregats-Funktionen dienen zum schnellen Definieren von virtuellen Attributen, in Fällen, bei deinen Beispielsweise der Skalar eines komplexen Attributes zurückgegeben, oder alle Elemente einer Relation summiert werden sollen.
Konkret kann die Angabe einer einer Funktion wie folgt aussehen:
`aggregate="Object.firstNonNull:Person.Name, Person.Nachname"`

Weitere verfügbare Funktionen sind, u.a.:

- Für String: `String.sortJoinCommaList`, `String.allNullOrEmpty`, `String.anyNullOrEmpty`, `String.allNotNullOrEmpty`, `String.anyNotNullOrEmpty`
- Für BigDecimal: `BigDecimal.min`, `BigDecimal.max`, `BigDecimal.sum`, `BigDecimal.diff`, `BigDecimal.absDiff`, `BigDecimal.prod`, `BigDecimal.div`, `BigDecimal.avg`, `BigDecimal.avgWithNullValues`, `BigDecimal.count`, `BigDecimal.countWithNullValues`, `BigDecimal.countTrueBools`, `BigDecimal.countFalseBools`, `BigDecimal.countNullOrFalseBools`
- Für Boolean: `Boolean.and`, `Boolean.or`, `Boolean.not`
- Für Date: `Date.min`, `Date.max`
- Für BOs: `BO.union`, `BO.firstNotDeleted`, `BO.newest`, `BO.oldest`, `BO.uniqueOrNull`
- Für beliebige Listen: `Object.firstNonNull`, `Object.allNull`, `Object.anyNull`, `Object.allNullOrEmptyRelations`, `Object.anyNullOrEmptyRelations`, `Object.allNotNullOrPopulatedRelations`, `Object.anyNotNullOrPopulatedRelations`

Unter-Element "np" von Attribute

Hier können weitere Optionen für nicht-persistente (npattr) Attribute einer Entität definiert werden.

Table 16. Unter-Element "np" von Attribute

Befehl	Beschreibung	Beispiel
calculationAuthority	Wo der Wert des nicht-persistenten Attributs berechnet werden soll. Bei Berechnung auf dem Server erfolgt zusätzlich eine Übertragung an den Client sowie eine in-place Ersetzung des clientseitigen Werts durch den Wert vom Server. Der Default ist "client".	<np calculationAuthority="server"/>

Unter-Element "db" von Attribute

FIXME

Table 17. Unter-Element "db" von Attribute

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

Restliche Unter-Elemente von Attribute

Table 18. 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 angehangenen 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.

Vordefinierte Datentypen für Attribute

FIXME Sehr unvollständig, viele Typen fehlen noch.

Timespan

Der Timespan-Typ wird verwendet, um eine *feste* Zeitspanne zu speichern. Intern wird diese als Anzahl von Sekunden (FIXME Sollte Millisekunden sein und wird ggf. irgendwann mal dahingehend umgebaut werden) abgespeichert, d.h. wenn ein Attribut mit type="Timespan" definiert wird, ist der Java-Typ ein `Long`.

In den meisten Anwendungsfällen bei denen es um Zeitspannen geht wird Timespan der richtige Typ sein. Wenn z.B. die Dauer eines Vorgangs abgespeichert werden soll ist die genaue, feste Zeitspanne, die der Vorgang gedauert hat, bekannt und soll auch genau so festgehalten werden.

Duration

Im Gegensatz zu Timespan handelt es sich bei Duration um eine (in gewissen Grenzen) *variable* Zeitspanne. Eine Duration wird als Kombination einer Anzahl von Jahren, Monaten, Tagen, Stunden, Minuten und Sekunden (inkl. Millisekunden) gespeichert. Abhängig von einem Referenzdatum kann daraus dann die genaue Zeitspanne ermittelt werden.

Intern wird der Wert als `javax.xml.datatype.Duration` abgespeichert.

Beispiel: `P1Y0M0DT0H0M0S` entspricht ausgehend vom 1.1.2015 365 Tagen; ausgehend vom 1.1.2012 (Schaltjahr) entspricht sie aber 366 Tagen.

Beispiel: `P0Y1M0DT0H0M0S` entspricht ausgehend vom 1.1.2015 31 Tagen; ausgehend vom 1.2.2015 aber nur 28 Tagen.

Die Behandlung von Durations im Code ist an vielen Stellen noch unsauber (z.B. in L10nTimespanFormat.nrx, welches auch zum Formatieren von Durations benutzt wird; oder im DurationType.nrx selbst). Oft wird die Duration einfach in eine feste Zeitspanne umgewandelt, ausgehend vom Date(0)/Epoch-Datum ohne zu berücksichtigen, dass ggf. ein anderes Referenzdatum benutzt werden müsste.

Schemapflege / Datenbankupdates

Einige Schemaänderungen können automatisch vom Schemagenerator verarbeitet werden. Ist das nicht der Fall, können die Änderungen per Datenbank Update-Script durchgeführt werden. FIXME TODO (Teile) der Schema-HowTo Datei hier einpflegen.

Liste der durch den UpdateHandler zur Verfügung gestellten Hilfsmethoden

Name	Parameter	Beschreibung	Beispielafuruf
checkTableExists	Tabellenname	Prüft, ob die Tabelle mit dem übergebenen Namen in der Datenbank existiert. Gibt einen boolschen Wert zurück.	checkTableExists('beleg')
checkColumnExists	table: Tabellenname, column: Spaltenname	Prüft, ob eine Spalte mit dem übergebenen Namen in der angegebenen Tabelle in der Datenbank existiert. Gibt einen boolschen Wert zurück.	checkColumnExists(table: 'beleg', column: 'belegnr')

Coredata-Generator

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

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-Identifier") an; sollte im Allgemeinen aber nicht benutzt werden, es wird automatisch ein konsistenter Tid vergeben.

Beschreibung

Eigenes Kind-Element des Root-Elementes; 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="RG_Solstice_Login"/>
  </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.

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).

Sprachunterstützung und Internationalisierung

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.

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.

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.

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

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

Die Benamsung 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.

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.

Web

Abweichend hiervon ist in Grails und Cauldron aktuell historisch bedingt noch eine andere Namensgebung üblich und es gibt kein eigentliches Mapping mit Automatismen für Klassen oder Entitäten.

In Grails gibt es oft nur ein einzelnes Bundle `Grails`, das aber auch aufgeteilt werden kann. Die Namensgebung lautet dann bspw. `Grails.checkout` oder `Grails.user.settings`.

In Cauldron besteht hier prinzipiell freie Namenswahl, neue Projekte sollten sich aber ebenfalls an das Paketnamensschema halten.

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. L10nPathCompilerI 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.

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.

Für jede im Schema definierte Entität, Beispiel "de.ipcon.db.core.Benannt", werden einige L10n-Daten automatisch angelegt und "gewartet". Diese sind im Beispiel zuerst das L10nBundle "de.ipcon.db.core" mit den L10nResourcen "_Benannt" sowie "_Benannt-s" (Name/Singular und Plural der Entität), 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.

Diese L10nBundles, bzw. deren L10nEntries und L10nResources, werden bei jedem Serverstart überprüft, ob Entitäten oder Attribute hinzugekommen sind - diese werden dann automatisch angelegt - bzw. weggefallen sind - dort werden dann für weggefallene Attribute automatisch entsprechende, ehemals gebrauchte L10nEntries und L10nResources entfernt.

In neueren Projekten werden Daten aus `nrx/[ Projektverzeichnis ]/resources/l10n/[ Projekt-Package ].bo_[ISO-Kürzel]` automatisch importiert. In diesen, von Hand angelegten Dateien, dürfen (bzw. zumindest sollten) sich nur Schlüssel-Text-Paare für im Schema definierte Entitäten und deren Attribute befinden. Ansonsten werden die "überzähligen" Texte bei jedem Serverstart erst angelegt, und dann, da es keine entsprechenden Entitäten bzw. Attribute (mehr) gibt, direkt wieder gelöscht.

"Freie" Texte (z.B. solche für Titel/Texte in Formularen) sollten - selbst wenn sie im Prinzip nur für eine Entität benutzt werden - in einem eigenen Paket `nrx/[ Projektverzeichnis ]/resources/l10n/[ Projekt-Package ]_[ISO-Kürzel]` untergebracht werden.

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

L10n und das Anführungszeichen bzw. Apostroph

Die Verwendung von Anführungszeichen oder Apostrophen in Übersetzungen (in `properties`- oder

Bundles-Dateien im Verzeichnis `resources/l10n/`) kann zu Problemen führen, insbesondere wenn diese in XML-Texten verwendet werden und unbeabsichtigt Abschnitte beenden, die nicht beendet werden sollten. Dies tritt häufig auf, wenn es um die Definition von Tabellenspalten geht. Besondere Vorsicht ist geboten, wenn es um Übersetzungen von Entitäts- und Attributnamen geht, da diese oft an solchen Stellen eingesetzt werden.

Um solche Probleme zu vermeiden, empfiehlt es sich, anstelle der „einfachen“ Zeichen `'` und `"`, die ursprünglich noch aus der Zeit der Schreibmaschinen stammen und lediglich eine vereinfachte Darstellung dieser Zeichen bieten, die schöneren und eigentlich korrekten typographischen Zeichen (erreichbar über AltGR+`'`) sowie (AltGr+`V`) und (AltGr+`B`) zu verwenden. Diese Zeichen werden in der Regel nicht als Steuerzeichen zur Abtrennung verwendet und tragen somit zur Vermeidung von unerwünschten Effekten bei.

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).

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 durchzustarten. 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=".

Die Formularengine des Solstice Clients

de.ipcon.form

Nachfolgende Dokumentation behandelt den Aufbau 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.

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.

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, Popuplisten und ähnlichem zu erzeugen und verwalten. Es stellt eigentlich das wichtigste und gleichzeitig das komplizierteste Objekt des Formularframeworks dar.

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.

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.

Definition

Fehler und Ursachen

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

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 duerfen nicht "abstract" sein

bi-Tabelle kann nicht erstellt werden (nachdem die Datenbank gedropped und recreated wurde)

".checked*"-Dateien im Projektverzeichnis löschen.

Compiler-Meldung "Object bla is null but shouldn’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 :-(

Synchronisation der Strukturelemente

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 - ...

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. 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.

Konfiguration im Schema

Berücksichtigte Daten

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

Berücksichtigte Entitäten

vgl. de/ipcon/db/fulltext/compass/SchemaMappingBuilder.isEntityIgnored()

Explizit *immer* ausgeschlossen werden die BOs nicht persistenter Entitäten, sowie die BOs der Entiäten `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="BO" plural="InterneEntitaeten">
  <fulltext indexed="no"/>
  <attr name="KryptischerString"/>
</Entity>
```

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

Berücksichtigte Attribute

vgl. 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="BO" 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="BO" plural="Eltern">
  <attr name="Name">
  <attr name="Kinder" type="Kind" relation="1-n">
     <fulltext indexed="yes"/>
  </attr>
</Entity>

<Entity name="Kind" extends="BO" 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.

Weitere Einstellungen im Schema

analyzed

vgl. de/ipcon/db/fulltext/compass/SchemaMappingBuilder.buildScalarConfig() sowie das entsprechende Kapitel in der Compass-Dokumentation

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

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. :toc: left :toc-title: Inhaltsverzeichnis :toclevels: 2 :icons: font

Formularelemente

Action

Name	Erlaubte Werte	Beschreibung
`acceleratorKey`	String: z. B. `"ENTER"`, `"control shift F5"`, ...	
`accKey`	siehe `acceleratorKey`	
`animation`	Boolean: **true**, false	Während die Action ausgeführt wird, eine Ladeanimation über das Formular legen.
`cmd`	String	Funktionsname, der z. B. von Buttons gerufen werden kann.
`contextMenu`	Boolean: true, **false**	
`formElementSync`	Boolean: **true**, false	
`icon`	String: icon="20x20/New.gif"	Pfad zum gewünschten icon.
`local`	Boolean: true, false	
`menu`	String	
`merge`	Boolean: true, **false**	
`mnemonicKey`		
`name`	String	Name der Action. Optional - wenn nicht angegeben, gleich `cmd`. Per default Button-Titel.
`offEDT`	Boolean: true, **false**	Action in neuem Thread ausführen.
`priority`	int: **0**	
`progressShowDelay`	int: **1000**	
`restoreFocus`	Boolean: **true**, false	
`shortDescription`		Kurze Beschreibung. Wird als Tooltip angezeigt.
`showLabel`	Boolean: true, **false**	Den Namen der Action unterhalb eines ggf. vorhandenen Icons anzeigen.
`smallIcon`		
`toolBar`	Leerer String. Bsp.: toolBar=""	Wird hinzugefügt, falls die Action in der Standard-ToolBar neben einer Table bzw. im Falle von topMdiOnly in der "obersten" Toolbar erscheinen soll.

Name	Erlaubte Werte	Beschreibung
`topMdiOnly`	Boolean: true, false	Bei `true` wird die Action auf die oberste Toolbar gedrückt, d.h. die des Clients (bzw. im SDI/native window manager mode die des Objektfensters).

availableOn

Liefert ein hier angegebenes Skript `true` zurück, wird die Action angezeigt, bei `false` nicht.

Die Bedingung für `availableOn` wird nur einmal und dann sehr früh evaluiert (bei der Entscheidung ob die Komponente überhaupt gebaut werden soll)

enabledOn

Die Action ist die ganze Zeit sichtbar und jenachdem ob ein hier angegebenes Skript `true` oder `false` zurückliefert, kann die Action angeklickt werden bzw. is ausgegraut.

`enabledOn` wird laufend, d.h. bei jedem Statuswechsel, evaluiert (nicht wie bei `availableOn`).

longDescription

Wenn man mal mehr (formatierten) Text zur Erklärung der Action anzeigen möchte (auch HTML ist möglich):

```
<Action cmd="resetData" name="Daten resetten" shortDescription="Daten zuruecksetzen"
toolBar="" accKey="control R">
  <onAction language="groovy">rootBO.doMagic()</onAction>
  <longDescription><![CDATA[<html>
    Folgende Voraussetzungen müssen erfüllt sein, damit die Daten der selektierten
Zeilen zurückgesetzt werden können:
    <ul>
      <li>Der Benutzer ist Mitglied der "Verwaltung" oder ein ADMIN</li>
      <li>Es ist mindestens eine Zeile ausgewählt</li>
    </ul>
  </html>]]></longDescription>
</Action>
```

onAction

Hier wird das Script definiert, welches für die jeweilige `Action` ablaufen soll.

BooleanInputComponent

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`class`		
`displayProperty` DEPRECATED		siehe `property`
`falseText`		
`nullText`		
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`text`		
`triState`	Boolean: **true**, false	Wenn hier true gesetzt ist, kann das Element drei Zustände annehmen: true, false und null. Ansonsten nur true und false.
`trueText`		

Border

Attribute:

Name	Erlaubte Werte	Beschreibung
`bevel-highlight`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	
`bevel-highlightInner`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
`bevel-highlightOuter`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
`bevel-shadow`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
`bevel-shadowInner`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	

Name	Erlaubte Werte	Beschreibung
bevel-shadowOuter	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
bevel-type	LOWERED, RAISED	Abschrägung.
beveled	highlight, highlightInner, highlightOuter, shadow, shadowInner, shadowOuter	
debug		
editable		
etched-highlight	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	Hervorhebung. Muss immer zusammen mit etched-shadow verwendet werden.
etched-shadow	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	Schattierung. Muss immer zusammen mit etched-highlight verwendet werden.
etched-type	String: **LOWERED**, RAISED	Begrenzung als Vertiefung oder Erhöhung darstellen.
etched	Boolean: true, **false**	
fontSize	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
fontStyle	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
implied		
line-color	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
line	Boolean: true, false	
maximumSize		
maxSize		
minimumSize		Alias. Siehe minSize

Name	Erlaubte Werte	Beschreibung
minSize	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
missingPropertiesPolicy		
name	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
preferredSize		Alias. Siehe prefSize
prefSize	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
title-justification	String: **LEFT**, CENTER, RIGHT	
title-position	String: NORTH, SOUTH, **WEST**, EAST	
title	String	Überschrift für die durch die Border abgegrenzten Inhalte.
toolBar-floatable		
toolBar-layout		
toolBar-orientation	String: HORIZONTAL, **VERTICAL**	
toolBar-position	String: NORTH, SOUTH, **WEST**, EAST	
toolBar-rollover		
toolBar		
topMdi		

Button

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`action`	String	`cmd`-Attribut der Action, die bei Klick auf den Button ausgeführt werden soll.
`background`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
`defaultButton`	Boolean: true, **false**	
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`fontStyle`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
`foreground`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	
`hAlign`		
`icon`	String: icon="20x20/New.gif"	Pfad zum gewünschten icon.
`multiClickThreshold`	Integer (750ms)	Spezifiziert innerhalb welchen Zeitraums mehrfaches Klicken des Buttons als ein einfaches Klicken interpretiert wird
`text`	String	Beschriftung des Buttons.
`vAlign`	String: TOP, CENTER, BOTTOM, z. B. vAlign="TOP"	Bestimmt die vertikale Ausrichtung des Textes innerhalb des Elements. Die Höhe des Elements muss größer sein als eine normale Zeilenhöhe, was z. B. durch Setzen des Attributs prefSize erreicht werden kann.

Canvas

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`toolTipText`	String.	Text, der angezeigt wird, wenn man den Mauszeiger über das Element hält.

Chart

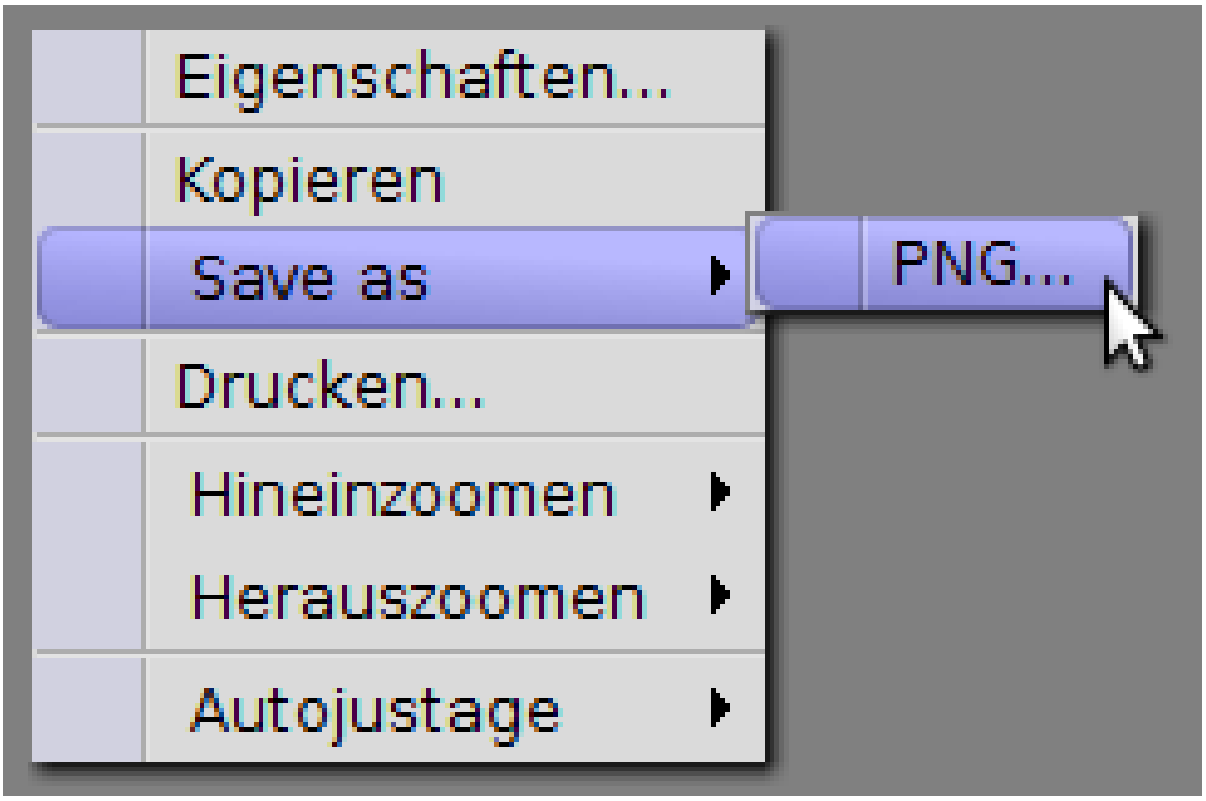
Komponente zur Darstellung von Diagrammen in Formularen. Die zugrundeliegende Bibliothek ist "JFreeChart".

Derzeit werden von unserer direkten Implementierung folgende Diagramme unterstützt:

- timeSeriesChart
- stackedXYAreaChart
- scatterPlot
- xYAreaChart
- xYLineChart
- xYStepAreaChart
- xYStepChart

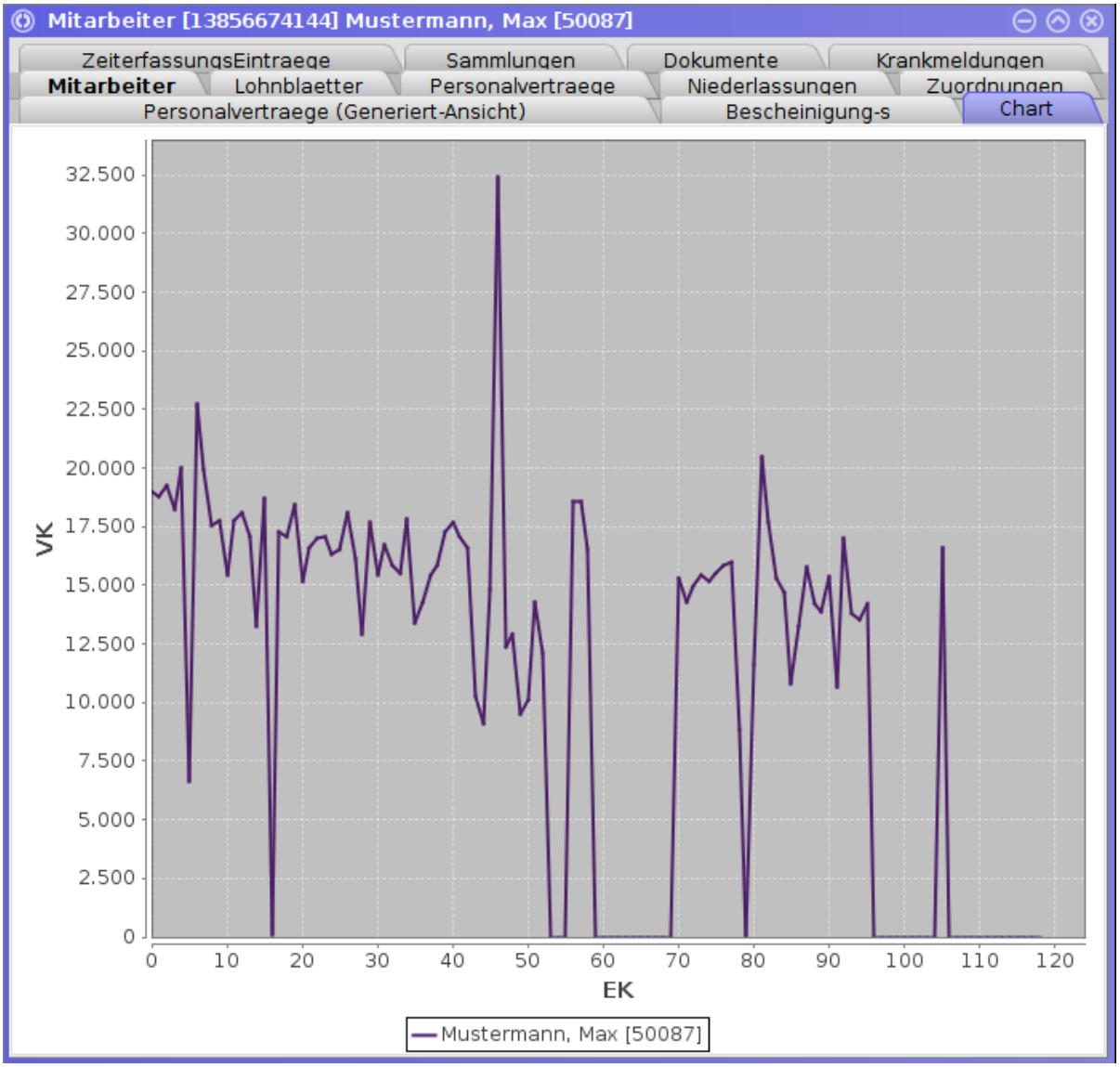
Darüber hinaus kennt JFreeChart noch einige andere Diagramme (z.B. Torten-Diagramme).

Über das Kontextmenü stehen weitere Funktionen wie Export als Bild, Drucken, Einstellungsmöglichkeiten, etc zur Verfügung.



Beispiel unter Verwendung unserer direkten Implementierung:

Das folgende Beispiel dient der Erklärung des Aufbaus. Es werden die ZeiterfassungsEinträge eines Mitarbeiters dargestellt, mit der jeweiligen Anwesenheitsdauer in y-Richtung.
Der JFreeChartBuilder wird als `builder` an das buildScript übergeben. Über diesen können dann die verschiedenen Diagramme gezeichnet werden. Innerhalb der Closure steht das `anchor`-Objekt zur Verfügung, welches das im Formular geöffnete BO referenziert.


Mitarbeiter [13856674144] Mustermann, Max [50087]
ZeiterfassungsEintraege
Sammlungen
Dokumente
Krankmeldungen
Mitarbeiter
Lohnblaetter
Personalvertraege
Niederlassungen
Zuordnungen
Personalvertraege (Generiert-Ansicht)
Bescheinigung-s
Chart
VK
32.500
30.000
27.500
25.000
22.500
20.000
17.500
15.000
12.500
10.000
7.500
5.000
2.500
0
0
10
20
30
40
50
60
70
80
90
100
110
120
EK
Mustermann, Max [50087]

```
<Chart>
  <buildScript><![CDATA[
  builder.xYLineChart(title:'oneTitle', xAxisLabel:"EK", yAxisLabel:'VK') {
  def zes = anchor.ZeiterfassungsEintraege
  antiAlias=true
  borderVisible=false
  borderPaint='#c0c0c0'
  plot {
    tableXYDataset() {
      rows = {
        (0..zes.size()-1).each{ it }
      }
      x = {
        it
      }
      series(name: anchor.kontakt.describe()) {
        values = {
          zes.values().getAt(it).Anwesenheitsdauer
        }
        stroke = 2
        paint = '#4c1e67'
      }
    }
  }
}
  ]]></buildScript>
</Chart>
```

Beispiel der generischen Verwendung (Prüfserie hat Einzelprüfungen mit deren Messwerten):

```
<Chart>
  <buildScript><![CDATA[
    import org.jfree.chart.ChartFactory
    import org.jfree.chart.ChartPanel
    import org.jfree.chart.JFreeChart
    import org.jfree.chart.plot.PlotOrientation
    import org.jfree.data.category.DefaultCategoryDataset

    def pserie = builder.anchor

    // dataset definieren
    def dataset = new DefaultCategoryDataset()
    pserie.getEinzelpruefungen().values().sort{ it.getPosition() }.each{
      dataset.addValue(it.getFeinheitNm(), 'FeinheitNm', it.getPosition())
      dataset.addValue(it.getElastizitaet(), 'Elastizitaet', it.getPosition())
      dataset.addValue(it.getHeissluftschrumpfS130(), 'HLS130', it.getPosition())
    }

    // Chart generieren
    JFreeChart chart = ChartFactory.createLineChart(
            'Messwerte der Einzelpruefungen',
            'Position','Skala',
            dataset,
            PlotOrientation.VERTICAL,
            true, true, false)
return chart
  ]]></buildScript>
</Chart>
```

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`closed`	Boolean: true, **false**	
`print`	Boolean: **true**, false	
`property`		
`props`	Boolean: **true**, false	
`save`	Boolean: **true**, false	
`toolTips`	Boolean: **true**, false	
`zoom`	Boolean: **true**, false	

CheckBox

Die Checkbox kann benutzt werden, um Boolean-Werte zu beeinflussen.

Attribute:

Name	Erlaubte Werte	Beschreibung
`class`		
`debug`		
`displayProperty` DEPRECATED		siehe `property`
`editable`	Boolean: **true**, false	false verhindert das Editieren des Feldes. Das Feld WIRD NICHT ausgegraut.
`implied`		
`maximumSize`		
`maxSize`		
`minimumSize`		Alias. Siehe `minSize`
`minSize`	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`missingPropertiesPolicy`		
`name`	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
`preferredSize`		Alias. Siehe `prefSize`
`prefSize`	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.

Name	Erlaubte Werte	Beschreibung
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`text`	String	Text, der zusätzlich hinter der Checkbox angezeigt wird.
`triState`	Boolean: **true**, false	Wenn hier true gesetzt ist, kann das Element drei Zustände annehmen: true, false und null. Ansonsten nur true und false. Wenn false, kann Null nicht mehr ausgewählt werden. Kann durch eine Angabe im Schema bereits definiert werden.

ComboBox

Die API-Dokumentation enthält ebenfalls weitere Informationen zu dieser Komponente.

Wie in dem Bereich über Widgets bereits gezeigt, kann eine ComboBox direkt auf ein property zeigen, woraufhin alle möglichen BOs angezeigt werden.

Beispiel:

Will man programmatisch eigene Auswahlmöglichkeiten anbieten kann man das über das choiceSkript machen:

```
...
<ComboBox property="FilterZeitraum" e-label="$R{Zeitraum}" chooseOnly="true"
nullable="false">
  <choiceScript>
    model.addEntry("2Tage", "letzten 2 Tage")
    model.addEntry("7Tage", "letzten 7 Tage")
    model.addEntry("30Tage", "letzten 30 Tage")
    model.addEntry("60Tage", "letzten 60 Tage")
    model.addEntry("100Tage", "letzten 100 Tage")
    model.addEntry("HalbesJahr", "letztes halbes Jahr")
    model.addEntry("LetztesJahr", "letztes Jahr")
    model.addEntry("Alle", "Alle")
  </choiceScript>
</ComboBox>
...
```

Attribute

Name	Erlaubte Werte	Beschreibung
autoSelect	String: first/last/firstNonNull	Gibt an welches Element als default ausgewählt werden soll.
chooseOnly	Boolean: true, **false**	Wenn true, dürfen keine eigenen Werte in das Feld eingetragen werden.
displayFormat DEPRECATED		siehe format
displayProperty DEPRECATED		siehe property
format	String (CBOFormat).	Legt fest, wie das BO für die Anzeige im Dropdown-Menü und im Feld formatiert ist. Bsp: "Id': 'Name"
nullable	Boolean: **true**, false	Wenn false wird null nicht als mögliche Auswahl angeboten.

Name	Erlaubte Werte	Beschreibung
nullChoiceTitle	String	Wird angezeigt, wenn nichts ausgewählt wurde. Gilt sowohl für das Feld als auch für den Eintrag im Dropdown-Menü, der den leeren Wert repräsentiert.
property		
relationName DEPRECATED		
selectEntity	String	Name der Entität, deren Objekte hier zur Auswahl angeboten werden sollen.
selectOutOf	String	Name der Relation, aus der Objekte zur Auswahl angeboten werden sollen.
sortBy	String	Name des Attributes, nach dem sortiert werden soll.
whereClause	String (OQL)	Optionaler Filter für im Attribut selectEntity gesetzte Entität im OQL-Format.
showId	Boolean: true, **false**	Wenn true, wird die Id jedes (BO-)Elements in eckigen Klammern hinter dem Element angezeigt.
suppressDuplicatesInNonRelationMode	Boolean: true, **false**	Wenn true, werden für Comboboxen, die keine Relation anzeigen, Duplikate in der Auswahlliste verhindert, indem die Id jedes (BO-)Elements in eckigen Klammern hinter dem Element angezeigt wird, sofern Duplikate auftreten.

DateChooser

Attribute:

Name	Erlaubte Werte	Beschreibung
autoHideButton		
columns		
debug		
displayFormat DEPRECATED		siehe format
displayProperty DEPRECATED		siehe property
editable	Boolean: **true**, false	false verhindert das Editieren des Feldes. Das Feld WIRD NICHT ausgegraut.
enabled	Boolean: **true**, false	
fontSize	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
fontStyle	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
format	String: LONG_, MEDIUM_, SHORT_, dd/MM/YYYY	Bestimmt die Formatierung des angezeigten Datums. Beispiele: LONG_: 7. September 2016 MEDIUM_: 07.09.2016 SHORT_: 07.09.16
implied	Boolean: true, false	
maximumSize		
maxSize		
minimumSize		Alias. Siehe minSize
minSize	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
missingPropertiesPolicy		
name	String	Um das Form-Element innerhalb des Formulars referenzieren zu können.

Name	Erlaubte Werte	Beschreibung
popupHeight	Integer mit Einheit px, c, em oder dlu	Die Angabe in Character "c" ist die zu favorisierende Bestimmt die Höhe des Fensters, in dem der Kalender zur Datumsauswahl angezeigt wird.
popupWidth	Integer mit Einheit px, c, em oder dlu	Die Angabe in Character "c" ist die zu favorisierende Bestimmt die Breite des Fensters, in dem der Kalender zur Datumsauswahl angezeigt wird.
preferredSize		Alias. Siehe prefSize
prefSize	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
property	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
selectAllWhenFocused	Boolean: true, *false*	Wenn das Form-Element den Fokus bekommt wird der gesamte Inhalt selektiert

Editor

Der Editor kann für String-Attribute angezeigt werden. Durch die Angabe des Modus kann die Darstellungsart beeinflusst werden. Es werden dazu intern JEdit-Klassen benutzt.

Beispiel:

```
...
<Editor property="Bemerkung" mode="patch"/>
...
```

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`columns`	int: **60**	Gibt an wieviele Spalten angezeigt werden sollen.
`electricScroll`	int: **3**	Wird der Cursor an eine bestimmte Stelle bewegt, wird sichergestellt, dass x Zeilen über & unter der Position angezeigt werden.
`focusable`	Boolean: **true**, false	Ist focusable false, kann das Textfeld nicht fokussiert werden...
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`maxUndos`	int: **500**	Gibt die Anzahl der gespeicherten Schritte und damit der möglichen Undos an
`mode`	String: **xml**, groovy, log, patch	Definiert die Formatierung des Editor-Fensters.
`rows`	int: **10**	Gibt an wieviele Zeilen angezeigt werden sollen.
`text`	String	Füllt das Attribut automatisch mit dem eingegebenen Text

Element

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`autoCreate`	Boolean: true, false	
`autoHide`	Boolean: true, false	
`displayProperty` DEPRECATED		siehe `property`
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`fontStyle`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
`hideForNullBO`	Boolean: **true**, false	Bei **true** wird das eingebettete GUI-Element nicht angezeigt, wenn das BO des angegebenen Attributs null ist, ansonsten ist es ausgegraut. Das ist nützlich im Fall von Attributketten, z.Bsp. bei `property="Person.Name"` wird über das Flag gesteuert, ob das Eingabefeld des Namens angezeigt wird, selbst wenn `Person` nicht gesetzt ist. Das Setzen auf **false** ist ratsam, wenn durch das Wegfallen des Eingabefelds die Anordnung anderer Elemente ungewollt beeinflusst wird.
`hSpaceDist`	Double **-1**	Was soll mit dem restlichen Platz von rightFill geschehen, sofern != 0 (default: FView.defaultCellHAlign); Beispiel: siehe ipcon/db/core/Report.nrx Beschriftung, die vor dem Inhalt des Element Tags angezeigt wird.
`label`	String	Beschriftung, die vor dem Inhalt des Element Tags angezeigt wird.

Name	Erlaubte Werte	Beschreibung
`labelBackground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Bestimmt die Hintergrundfarbe des Labels.
`labelForeground`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	Bestimmt die Schriftfarbe des Labes.
`mode`	String: `FREE_FIELD`, `FREE_LABEL`, `LABEL_ON_TOP`, `DEFAULT`	`FREE_FIELD`: Das im Element beinhaltete Feld füllt den gesamten Raum zwischen Ende des Labels und rechtem Rand des Elternelements. `FREE_LABEL`: Das längste Label innerhalb innerhalb eines gemeinsamen Elternelements bestimmt die Breite der "Labelspalte". Alle Labels und die beinhalteten Felder werden am rechten Rand des längsten Labels ausgerichtet. `LABEL_ON_TOP`: Das Label wird über dem beinhalteten Feld angezeigt. `DEFAULT`: siehe `FREE_LABEL`.
`no-label`	Boolean: true, **false**	

Name	Erlaubte Werte	Beschreibung
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt. Nähere Informationen hierzu im GUI-Kochbuch
`rightFill`		
`rows`		
`transactionControl`		
`x`		
`y`		

Email

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`align`	String: `LEFT`, `CENTER`, `RIGHT`, `LEADING`, `TRAILING`	
`class`		
`columns`		
`debug`		
`disabled`	Boolean: true, **false**	false verhindert das Editieren des Feldes. Das Feld WIRD ausgegraut.
`displayFormat` DEPRECATED		siehe `format`
`displayProperty` DEPRECATED		siehe `property`
`editable`	Boolean: true, **false**	false verhindert das Editieren des Feldes. Das Feld WIRD NICHT ausgegraut.
`font`		
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`fontStyle`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
`foreground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	

Name	Erlaubte Werte	Beschreibung
format		
implied		
lineWrap	Boolean: **true**, false	Bricht Text an der rechten Kante um, statt eine Scrollbar anzuzeigen.
maxSize		
maximumSize		
minimumSize		Alias. Siehe minSize
minSize	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
missingPropertiesPolicy		
name		
password	Boolean: true, **false**	Passwortfeld statt normalem Text.
prefSize	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
preferredSize		Alias. Siehe prefSize
property	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
roundingFormat		
rows	int: **4**	

Name	Erlaubte Werte	Beschreibung
`selectAllWhenFocused`	Boolean: true, *false*	Wenn das Form-Element den Fokus bekommt wird der gesamte Inhalt selektiert
`syncOnWait`		
`syncOnWaitDelay`		
`tabSize`	int: **4**	
`translationAvailable`		
`wrapStyleWord`	Boolean: **true**, false	Wenn Text umgebrochen wird, dann möglichst an Whitespace-Grenzen.

Image

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`displayProperty` DEPRECATED		siehe `property`
`prefSize`	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`scaleToFit`	Boolean: true, **false**	

Label

Name	Erlaubte Werte	Beschreibung
`asyncRefresh`	Boolean: true, false, **null**	Explizit synchrones oder asynchrones Verhalten erzwingen. Synchrones Verhalten ist hilfreich, falls das Label variierenden Platzbedarf hat. (Üblicherweise bei mehrzeiliger HTML-Ausgabe.)
`background`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'
`class`		
`disabledIcon`		
`displayFormat` DEPRECATED		siehe `format`
`displayProperty` DEPRECATED		siehe `property`
`font`		
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`fontStyle`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	

Name	Erlaubte Werte	Beschreibung
`foreground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'
`format`	String (CBOFormat). Bsp.: `property="." format="'Auflage: 'Auflage.Nr"`	Ermöglicht dynamisches Erzeugen des Labeltextes durch CBOFormat. Voraussetzung, um, wie im Beispiel, auf einen Attributswert zugreifen zu können, ist es, dass die Property gesetzt ist (siehe `property`).
`gradientStartColor`	Farbangabe. Bsp: `gradientStartColor="160 160 255"`	Hinterlegt das Label mit einem Farbverlauf von links nach rechts. Kann zusammen mit `gradientStopColor` verwendet werden.
`gradientStartPosition`	String: `NORTH`, `SOUTH`, `WEST`, `EAST`	Verlegt den Anfang des Farbverlaufs.
`gradientStopColor`	Farbangabe. Bsp: `gradientStopColor="160 160 255"`	Hinterlegt das Label mit einem Farbverlauf von rechts nach links. Kann zusammen mit `gradientStartColor` verwendet werden.
`gradientStopPosition`	String: `NORTH`, `SOUTH`, `WEST`, `EAST`	Verlegt das Ende des Farbverlaufs.
`hAlign`	String: `LEFT`, `CENTER`, `RIGHT`	Horizontale Ausrichtung.
`hTextPosition`	String: `LEFT`, `CENTER`, `RIGHT`, `LEADING`, `TRAILING`	

Name	Erlaubte Werte	Beschreibung
`html`	Boolean: true, **false**	Erlaubt es, im `text`-Attribut HTML zu benutzen, ohne auf spitze Klammern verzichten zu müssen.
`icon`	String: icon="20x20/New.gif"	Pfad zum gewünschten icon. Das icon erscheint vor dem in `text` angegebenen Text.
`iconTextGap`	Bsp: `iconTextGap="10"`	Angabe des Abstandes zwischen dem Icon und Text (als Ganzzahl)
`padding`	Vier positive ganze Zahlen, getrennt durch Komma oder Leerzeichen, die den jeweiligen Abstand in Pixeln zum oberen, linken, unteren und rechten Rand spezifizieren: `padding="20,30,20,30"`	Außenabstand in Pixeln.
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`text`	String	Text des Labels. Kann HTML beinhalten, wenn `html` Attribut `true` ist.
`textWhileLoading`		
`toolTipText`	String.	Text, der angezeigt wird, wenn man den Mauszeiger über das Element hält.
`vAlign`	String: TOP, CENTER, BOTTOM, z. B. vAlign="TOP"	Bestimmt die vertikale Ausrichtung des Textes innerhalb des Elements. Die Höhe des Elements muss größer sein als eine normale Zeilenhöhe, was z. B. durch Setzen des Attributs prefSize erreicht werden kann.
`vTextPosition`	String: `TOP`, `CENTER`, `BOTTOM`	

FPanel (abstrakt)

Superklasse für viele Elemente. Bringt die unten stehenden Attribute und Sub-Elemente mit.

Skriptvariablen

In den Skripten sind diese Bindings verfügbar:

Variablenname	Klasse/Interface	Definition	Beschreibung
`ctx`	ClientContextI	ftx.getCtx()	Client-weiter Kontext, wird z.Bsp. zum Öffnen von Dialogen oder Formularen benutzt
`user`	Benutzer	ftx.getCtx().getCurrentUser()	der im Client angemeldete Benutzer
`ftx`	FormContextI	ftx	Kontext des Formulars, in dem das FPanel eingesetzt ist. Wird gebraucht, um andere Formularelemente anzusprechen.
`bo`	BO	ftx.getBO()	Das zugrundeliegende BO des FPanels. (Kann vom rootBO abweichen)
`tx`	Transaction	ftx.getRoot().getTransaction()	Die Transaction, mit der das rootBO geladen wurde, wenn `bol instanceof Transaction`
`fe`	FormElementI	fe	Das FPanel (als FormElementI)
`bol`	BOLoaderI	ftx.getRoot().getBOLoader()	Der Loader, mit dem die Daten des Strukturelements geladen wurde. Im Fall von Formularen gilt: `bol instanceof Transaction`
`rootBO`	BO	ftx.getRoot().getBO()	Das zugrundeliegende BO des Formulars.

Subelemente

onAfterSelectValue

Ermöglicht es, ein Skript zu definieren, das nach dem Setzen eines Werts ausgeführt wird.

```
<Text property="Beschreibung">
  <onAfterSelectValue
language="groovy">ftx['v_lbl_beschreibung_fehlt'].ftx.refreshForms()</onAfterSelectVal
ue>
</Text>
```

Attribute

Name	Erlaubte Werte	Beschreibung
`language`	String: groovy, **beanshell**	Die zu verwendende Skriptsprache.

editableIf

Ergebnis des Ausdrucks (hier `bo.kannEditiertWerden`) bestimmt, ob das Elternelement editiert werden kann.

```
<Text property="Beschreibung">
  <editableIf language="groovy">bo.kannEditiertWerden</editableIf>
</Text>
```

Attribute

Name	Erlaubte Werte	Beschreibung
`language`	String: groovy, **beanshell**	Die zu verwendende Skriptsprache.

visibleIf

Ergebnis des Ausdrucks (hier `bo.istSichtbar`) bestimmt, ob das Elternelement angezeigt wird.

```
<Text property="Beschreibung">
  <visibleIf language="groovy">bo.istSichtbar</visibleIf>
</Text>
```

Attribute

Name	Erlaubte Werte	Beschreibung
`leftEntity`	String: Entitätsname	Die Entität, an der das `property` des Elternelements hängt, muss eine Subklasse der hiermit angegebenen Entität sein. Kann Skript-Inhalt dieses Tags ersetzen oder ihn als Konjunktion ergänzen.
`language`	String: groovy, **beanshell**	Die zu verwendende Skriptsprache.
`leftClass`	DEPRECATED	siehe `leftEntity`
`never`	Boolean: true, **false**	Wenn **true**, wird das Elternelement niemals angezeigt.
`notForLeftEntity`		
`notForRightEntity`		
`property`		
`rightEntity`		

OnDrop

OnDrop ist ein Tag, dass benutzt werden kann um Dateien zu behandeln die auf das Element gedropt wurden:

Beispiel:

```
<onDrop language="groovy">
  def result = Datei.importFileFromDnD(ftx, tx, files, rootBO)
  ftx.toast("${result.size()} Dateien importiert.")
  ftx.getRoot().refreshForms()
</onDrop>
```

Attribute: FIXME

onFocusGained

Eventhandler. Wird beim Setzen des Eingabe-Focus auf das Elternelement ausgeführt.

```
<Text property="Beschreibung">
  <onFocusGained language="groovy">ftx.toast('Das Textfeld "Beschreibung" hat jetzt
den Fokus.')</onFocusGained>
</Text>
```

onFocusLost

Eventhandler. Wird ausgeführt, wenn das Elternelement den Fokus verliert.

```
<Text property="Beschreibung">
  <onFocusLost language="groovy">ftx.toast('Tschüss!')</onFocusLost>
</Text>
```

onRefresh

Eventhandler. Wird beim Übertragen von Model-Daten in die View aufgerufen.

```
<Text property="Beschreibung">
  <onRefresh language="groovy">ftx.toast('Lade Beschreibung.')</onRefresh>
</Text>
```

onSync

Eventhandler. Wird beim Übertragen von View-Daten ins Model aufgerufen.

```
<Text property="Beschreibung">
  <onSync language="groovy">ftx.toast('Speichere Beschreibung.')</onSync>
</Text>
```

FInputPanel (abstrakt)

Leitet sich von FPanel ab.
Alle Komponenten, die einen Input erwarten leiten sich daraufhin von FInputPanel ab.

Zentralisiert die Mandatory-Berechnung, sodass alle Subklassen ein gleiches Verhalten besitzen.
Außerdem wird noch ein neues Skript-Tag - `alsoMandatoryIf` - hinzugefügt.

alsoMandatoryIf

Kann benutzt werden um ein Eingabeelement zum Pflichtfeld zu machen.

```
<alsoMandatoryIf language="groovy">
   ctx.currentUser.istMitgliedVon("Pflichtgruppe")
</alsoMandatoryIf>
```

Per default ist das angegebene Skript gecached (wird also nur beim Öffnen des Formulars ausgewertet), da man von komplexerem Code ausgegangen ist.

Möchte man aber, dass bei einem Refresh das Skript erneut evaluiert wird, kann man das Caching mittels `cached="false"` deaktivieren.

```
<alsoMandatoryIf cached="false" language="groovy">
   !rootBO.KundeWillDatenNichtNennen
</alsoMandatoryIf>
```

Falls das Schema vorgibt, dass ein Element mandatory ist, kann dies NICHT über `alsoMandatoryIf` ausgehebelt werden.

PDFViewer

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
annotationColorEdit	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH Standard: **#000000**	
annotationColorSaved	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH Standard: **FF0000**	
annotationFont	String: **Helvetica-18**	
bufferedImage	Boolean: true, **false**	**false** = ein volatiles Bild wird direkt in den Grafik(karten)-Speicher geladen, was je nach Treiber(/Karte/System) länger dauern kann. **true** = ein gepuffertes (aber speicher-intensiveres) Bild wird in den Grafik(karten)-Speicher geladen. Diese Vorgehensweise ist mitunter unter Windows etwas schneller
e-no-label	Boolean: true, **false**	Unterdrückt die Anzeige des zugehörigen Labels
enableAnnotations	Boolean: true, **false**	
fitOnPage	Boolean: true, **false**	
fitWidth	Boolean: **true**, false	
forceAntialiasing	Boolean: **true**, false	
lowQuality	Boolean: true, **false**	
onAnnotationAdded		
onAnnotationChanged		
onAnnotationRemoved		
onAnnotationSelected		

Name	Erlaubte Werte	Beschreibung
`prefSize`	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`scaleBicubic`	Boolean: true, **false**	
`scaleToFit`	Boolean: true, **false**	Alias für `fitOnPage`.
`zoomValue`	Double: **0.05**	

Popup

Attribute:

Name	Erlaubte Werte	Beschreibung
`align`	String: `LEFT`, `CENTER`, `RIGHT`, `LEADING`, `TRAILING`	
`autoEdit`	Boolean: true, **false**	Attribute in der Detailview eines Popup Elements sind normalerweise schreibgeschützt, d.h. Wert 'false'. Grund ist, dass ein Popup mit Detailview oft allgemeinere Informationen im Formular anzeigt, das BO des Popups jedoch nicht nur von dem aktuellen `RootBO` abhängt. Eine Änderung eines Attributs im Kontext dieses RootBOs könnte demnach nicht allgemein genug sein um überall zu passen. Dies hindert den Benutzer daran, ausversehen einen solchen Wert zu ändern. Es gibt jedoch Fälle, in denen es im Formular explizit gewünscht ist, die Attribute ändern zu dürfen. Zum Beispiel wenn das BO 'dependent' vom RootBO ist. In diesem Fall kann dem Benutzer das Editieren per Definition von `autoEdit="true"` erlaubt werden. Soll der Benutzer die Attribute des BOs ändern dürfen, jedoch das eigentliche BO nicht sehen können, dann ist eine Umsetzung über eine Attributkette in der property-Definition bzw. ein Wrapper-`<Element..></Element>` vielleicht sinnvoller.
`columns`		
`debug`		

Name	Erlaubte Werte	Beschreibung
`displayFormat` DEPRECATED		siehe `format`
`displayFormatDivider`		
`displayFormatPostfix`		
`displayFormatPrefix`		
`displayProperty` DEPRECATED		siehe `property`
`displaySort`		
`editable`	Boolean: **true**, false	`false` verhindert Editieren des Feldes. Das Feld wird ausgegraut.
`fallBackProperty`		
`font`		
`fontsize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`fontStyle`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
`fontStyle`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
`foreground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'

Name	Erlaubte Werte	Beschreibung
format	String (CBOFormat).	Legt fest, wie das BO für die Anzeige im Feld formatiert ist. Bsp: "Id': 'Name"
implied		
lazy		
lookupCaseSensitive	Boolean: true, **false**	Definiert ob die Eingabe mit oder ohne Berücksichtigung der Gross-/Kleinschreibung gesucht werden soll
lookupProperty	String	Komma-separierte Liste von Attributen/Attribut-Ketten, die bei der "Schnelleingabe" durchsucht werden sollen; bei einem eindeutigen Ergebnis wird automatisch selektiert
lookupStartingWith		
lookupSubstring	Boolean: **true**, false	
maximumSize		
maxSize		
minimumSize		Alias. Siehe minSize
minSize	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
missingPropertiesPolicy		
name	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
nullChoiceTitle		
offerCopyBeforeEdit		
openFormTid	String	Tid des Formulars, das benutzt werden soll, um Objekte aus diesem Popup zu öffnen.
openProperty		
popupAlign		

Name	Erlaubte Werte	Beschreibung
`popupHeight`	Integer mit Einheit **px**, c, em oder dlu. Bsp.: popupHeight="40c"	Die Höhe des resultierenden Popups. Wird keine Einheit angegeben, wird von einem px-Wert ausgegangen.
`popupSize`	Komma-separierte Integer mit Einheit **px**, c, em oder dlu. Bsp.: popupSize="960, 20c"	Kurzform für `popupWidth` und `popupHeight`. Der erste Wert bestimmt die Breite, der zweite Wert die Höhe. Einheiten können gemischt angegeben werden. Wird keine Einheit angegeben, wird von einem px-Wert ausgegangen.
`popupWidth`	Integer mit Einheit **px**, c, em oder dlu. Bsp.: popupWidth="40c"	Die Breite des resultierenden Popups. Wird keine Einheit angegeben, wird von einem px-Wert ausgegangen.
`preferredSize`		Alias. Siehe `prefSize`
`prefSize`	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`showEntityName`	Boolean: **true**, false	
`showNewAction`	Boolean: true, **false**. Bsp.: `showNewAction="true"`	Bestimmt, ob im Feld ein Icon (Blatt mit Stern) angezeigt wird, das beim Anklicken ein neues Fenster zum Erstellen eines neuen Objektes für die jeweilige Relation öffnet.

Name	Erlaubte Werte	Beschreibung
`showSelectAction`	Boolean: **true**, false. Bsp.: `showSelectAction="false"`	Bestimmt, ob im Feld ein Icon (Blatt Papier) angezeigt wird, das beim Anklicken eine Tabelle bereits existierender Objekte öffnet, die für die Relation ausgewählt werden können.
`subentitiesToExclude`		
`templateSource`		
`usePolymorphySelectionTree`		

Tab

Attribute:

Name	Erlaubte Werte	Beschreibung
`background`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'
`debug`		
`editable`	Boolean: **true**, false	Bei false kann innerhalb dieses Elements kein Feld mehr editiert werden.
`foreground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'

Name	Erlaubte Werte	Beschreibung
grabFocus	Boolean: **true**, false	
implied		
lazy	Boolean: **true**, false	Daten dieses Tabs erst beim Öffnen des Tabs laden, nicht schon beim Öffnen des Formulars.
maximumSize		
maxSize		
minimumSize		Alias. Siehe minSize
minSize	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
missingPropertiesPolicy		
name	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
preferredSize		Alias. Siehe prefSize
prefSize	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
scrollable	Boolean: true, false; Oder Richtungsbeschränkung, z. B. scrollable="VERTICAL_ONLY"	Scrollbar für dieses Element ein- oder ausschalten bzw. auf eine Richtung beschränken.
title	String	Wird als Titel des Tabs angezeigt.
toolTipText	String.	Text, der angezeigt wird, wenn man den Mauszeiger über das Element hält.

Subelemente:

Name	Erlaubte Werte	Beschreibung
autoSelectObject		
Column		
Columns		
DetailView		
onAfterSelectValue		
onShowingTab		Wird ausgeführt, sobald der Tab aktiviert (angezeigt) wird
onHidingTab		Wird ausgeführt, sobald der Tab deaktiviert (versteckt) wird
Query		
View		

TabbedView

Attribute:

Name	Erlaubte Werte	Beschreibung
`antiAlias`	Boolean: true, **false**	
`conflictPolicy`	String: '**IGNORE**', 'ASK'	Gibt an, wie Konflikte beim Speichern durch veraltete BOs behandelt werden sollen. Nur in der äussersten TabbedView / View erlaubt. IGNORE: Ignoriere Konflikte und übernehme/merge die Änderungen, ASK: Prüfe auf Konflikte und Frage den Benutzer wie damit umgegangen werden soll.
`debug`		
`editable`	Boolean: **true**, false	Bei false kann innerhalb dieses Elements kein Feld mehr editiert werden.
`height`	Integer mit Einheit px, c, em oder dlu	Initiale Höhe der View in Pixeln oder der angegebenen Einheit.
`ignoreOtherLocalTransactionSaves`	Boolean: true, **false**	Nur für Formular-Roots. Änderungen aus lokalen Speichervorgängen im Client in diesem Formular nicht nachziehen
`implied`		
`l10nBundle`		
`maximumSize`		
`maxSize`	Tupel: (horizontal, vertikal). Beispiele: maxSize="4c, 5c"; maxSize="4c,"; maxSize=",5c"	
`minimumSize`		Alias. Siehe `minSize`

Name	Erlaubte Werte	Beschreibung
`minSize`	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`missingPropertiesPolicy`		
`name`	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
`preferredSize`		Alias. Siehe `prefSize`
`prefSize`	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`rotateLabels`	Boolean: **true**, false	Gibt an, ob bei tabPlacement `LEFT` oder `RIGHT` die Beschriftung der Tabs gedreht werden soll oder nicht.
`tabLayoutPolicy`		
`tabPlacement`	String: `BOTTOM`, `TOP`, `LEFT`, `RIGHT`	Die Reiter der TabbedView unten, oben, links oder rechts anzeigen.
`useMaximumHeight`	Boolean: true, **false**, z. B.: useMaximumHeight="true"	
`useMaximumWidth`	Boolean: true, **false**, z. B.: useMaximumWidth="true"	
`width`	Integer mit Einheit px, c, em oder dlu	Initiale Breite der View in Pixeln oder der angegebenen Einheit.

Subelemente:

Name	Erlaubte Werte	Beschreibung
Tab		

Table

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
`asyncModel`	Boolean: **true**, false	
`autoRefresh`	Boolean: true, **false**	Änderungen an Objekten des Typs, welche in der Tabelle angezeigt werden, führen zu einer Aktualisierung der Tabelle
`additionalAutoRefreshEntities`	String	komma-separierte Liste von Entitätsnamen; Änderungen an Objekten dieser Typen führen zusätzlich zum in der Tabelle angezeigten Typ zu einer Aktualisierung der Tabelle; funktioniert nur zusammen mit `autoRefresh="true"`; Unter-Entitäten werden automatisch mitüberwacht
`autoSelectFirst`	Boolean: **true**, false	
`autoSelectLast`	Boolean: true, **false**	
`autoSelectNone`	Boolean: true, **false**	

Name	Erlaubte Werte	Beschreibung
`columns`	String	Definition der Spalten einer Tabelle im Kurzformat. Bsp.: `columns="Attribut1, 25c, desc3 | Attribut2, desc | Attribut3, desc2"` Einzelne Spaltendefintionen werden durch `|` getrennt. `desc` und `desc2` bestimmen eine absteigende Sortierung. Die Numerierung legt die Priorität der Spalte beim Sortieren fest, wobei `desc` vor `desc2` berücksichtigt wird und `desc2` vor höheren Ziffern. Durch Verwendung von `asc` kann auch aufsteigend sortiert werden. Spalten werden ausgeblendet, indem sie einfach nicht deklariert werden. Die Reihenfolge kann geändert werden. `25c` definiert hier eine Spaltenbreite von 25 Buchstaben. Weitere erlaubte Einheiten sind `px`, `em` und `dlu`. Bei polymorphem Tabelleninhalt kann es hilfreich sein, den Typ des jeweiligen Objektes mit `Bot.Name '$R{BOTyp}'` anzuzeigen.
`columnSelectionAllowed`	Boolean: **true**, false	False: Bei Mausklick (oder `.selectObject()`) wird die komplette Zeile ausgewählt
`createInDetailView`	Boolean: true, **false**	
`dependent`	Boolean: true, false	Die anzuzeigenden Elemente sind nicht eigenständig und leben nur in diesem View.
`displayClass` DEPRECATED		

Name	Erlaubte Werte	Beschreibung
`displayProperty` DEPRECATED		siehe `property`
`easyEdit`	Boolean: **true**, false	Selektion per Doppelklick
`editableDetailView`	Boolean: true, **false**	Bewirkt, dass die DetailView der Tabelle das Bearbeiten der dort angezeigten Werte zulässt. Nützlich, wenn es sich bei diesen Werten bspw. um Attribute aus Subrelationen eines virtuellen Attributs handelt, die eigentlich gesperrt wären.
`editable`	Boolean: true, **false**	
`entity`	String: Entitätsname	Bestimmt die Entität, deren Objekte diese Tabelle darstellen soll.
`explicitStart`	Boolean: true, **false**	Die Daten im Lesezeichen werden erst vom Server geladen, wenn man das explizit mit F5 oder RETURN "anfordert". Man kann also erstmal alle nötigen Filter setzen und das Lesezeichen beginnt nicht bereits nach dem ersten gesetzten Filter mit dem Laden. Nützlich bei Entitäten mit sehr vielen Objekten, bei denen es angeraten ist erst mal ein bisschen vorzufiltern.
`filter`	String	
`fontSize`	String: +X%	Gibt an, um wieviel Prozent die Schrift vergrößert werden soll.
`freeSearch`	Boolean: **true**, false	
`height`	Integer	

Name	Erlaubte Werte	Beschreibung
`horizontalScrollBarPolicy`	String: `ALWAYS`, `AS_NEEDED`, `NEVER`	`ALWAYS`: Zeigt eine horizontale Scrollbar an, sobald ein Objekt in der Tabelle nicht mehr vollständig in das aktuelle Fenster passt. `AS_NEEDED`: Zeigt eine horizontale Scrollbar an, sobald ein Objekt in der Tabelle nicht mehr vollständig in das aktuelle Fenster passt. `NEVER`: Es wird niemals eine Scrollbar angezeigt, auch nicht dann, wenn nicht alle Objekte der Tabelle in das aktuelle Fenster passen.
`ignoreInitialFocus`	Boolean: true, **false**	
`implied`		
`intercellSpacingX`	Integer	Zusätzlicher freier Raum in Zellen horizontal.
`intercellSpacingY`	Integer	Zusätzlicher freier Raum in Zellen vertikal.
`itemProperty`	String	Zeiger auf die Property, die in den Elementen die Postennummer darstellt. Posten-Modus. Fuehrt dazu, dass zwei Aktionen, naemlich Posten-rauf und Posten-runter gebaut werden, und die ITable standardmaessig die Postenspalte aufsteigend sortiert.
`linkOnly`	Boolean: true, **false**	Es können nur bestehende Objekte verknüpft werden und keine neue erstellt
`loadImmediate`	Boolean: true, **false**	Bestimmt, ob die Objekte in der Tabelle sofort geladen werden sollen. Wenn `true`, ist ein Drücken von Enter im Suchbalken zur Anzeige der Inhalte nicht erforderlich.
`maxRowHeight`	Integer	

Name	Erlaubte Werte	Beschreibung
maxRows	Integer: **100000**	Definiert die maximale Anzeige von Objekten (vergleichbar mit dem von SQL bekannten "LIMIT 100000")
maximumSize	siehe maxSize	
maxSize	Tupel: (horizontal, vertikal). Beispiele: maxSize="4c, 5c"; maxSize="4c,"; maxSize=",5c"	
minimumSize	siehe minSize	
minSize	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	
missingPropertiesPolicy	String: **error**, ignore, log	Wie soll mit nicht existierenden oder falsch geschriebene Attributen in der Spaltendefinition umgegangen werden
name	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
openFormTid	String	Tid des Formulars, das benutzt werden soll, um Objekte aus dieser Tabelle zu öffnen.
openProperty	String	Name der Property für welche beim Doppelklick das Standardformular geöffnet wird.
parentClass DEPRECATED		
parentEntity	String	
preferredSize		Alias. Siehe prefSize
prefSize	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
preferredVisibleRows	Integer: **5**	Definiert die bevorzugte Anzahl angezeigter Zeilen

Name	Erlaubte Werte	Beschreibung
property	String	
readOnly	Boolean: true, **false**	
reloadBOsWhenOpening	Boolean: true, **false**	True: Beim Öffnen von BOs über dieses Lesezeichen werden die BOs immer neu vom Server neu geladen, anstatt die Instanz aus dem Lesezeichen zu verwenden.
resizeMode	String: ALL_COLUMNS, LAST_COLUMN, NEXT_COLUMN, SUBSEQUENT_COLUMNS, OFF	Bestimmt das Verhalten der Spaltenbreite beim Vergrößern/Verkleinern der Tabelle.
rowSelectionAllowed	Boolean: **true**, false	False: Bei Mausklick wird die komplette Spalte ausgewählt
showEntityName	Boolean: **true**, false	
showHorizontalLines	Boolean: **true**, false	
showVerticalLines	Boolean: **true**, false	
singleClickEdit	Boolean: true, **false**	
sortForSelected	Boolean: true, **false**	
subentitiesToExclude	String: Kommaseparierte Liste von Entitätsnamen	Subentitäten der in der Tabelle dargestellten Entität, die nicht angezeigt werden sollen.
templateSource	String	
useMaximumHeight	Boolean: true, **false**, z. B.: useMaximumHeight="true"	
useMaximumWidth	Boolean: true, **false**, z. B.: useMaximumWidth="true"	
usePolymorphySelectionTree	Boolean: true, **false**	

Name	Erlaubte Werte	Beschreibung
`verticalScrollBarPolicy`	String: `ALWAYS`, `AS_NEEDED`, `NEVER`	`ALWAYS`: Zeigt eine vertikale Scrollbar an, sobald ein Objekt in der Tabelle nicht mehr vollständig in das aktuelle Fenster passt. `AS_NEEDED`: Zeigt eine vertikale Scrollbar an, sobald ein Objekt in der Tabelle nicht mehr vollständig in das aktuelle Fenster passt. `NEVER`: Es wird niemals eine Scrollbar angezeigt, auch nicht dann, wenn nicht alle Objekte der Tabelle in das aktuelle Fenster passen.
`viewOnly`	Boolean: true, **false**	Es darf nur geschaut werden, verlinken bestehender Objekte, etc ist nicht möglich
`width`	Integer	

Column

Attribute:

Name	Erlaubte Werte	Beschreibung
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`displayProperty` DEPRECATED		siehe `property`
`format`	String (CBOFormat).	Legt fest, wie das BO aus der property für die Anzeige in der Zelle formatiert wird. Bsp: "Id': 'Name"; funktioniert nur, wenn das letzte Attribut in der property auf eine Relation verweist
`displayFormat` DEPRECATED		siehe `format`

Name	Erlaubte Werte	Beschreibung
`toolTipFormat`	String (CBOFormat).	Legt fest, wie das BO der aktuellen Zeile als ToolTip für diese Spalte formatiert wird. Bsp: "Id': 'Name"
`title`	String	Überschrift für die Spalte.
`width`	Integer mit Einheit px, c, em oder dlu	Initiale Breite der Spalte in Pixeln oder der angegebenen Einheit.
`vAlign`	String: TOP, CENTER, BOTTOM, z. B. vAlign="TOP"	Bestimmt die vertikale Ausrichtung des Textes innerhalb des Elements. Die Höhe des Elements muss größer sein als eine normale Zeilenhöhe, was z. B. durch Setzen des Attributs prefSize erreicht werden kann.
`cellBackground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Bestimmt die Hintergrundfarbe der Spalte für ungerade Zeilennummern.

Name	Erlaubte Werte	Beschreibung
`alternateCellBackground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	Bestimmt die alternative Hintergrundfarbe der Spalte für gerade Zeilennummern.
`cellForeground`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	Bestimmt die Schriftfarbe der Spalte für ungerade Zeilennummern.
`alternateCellForeground`	Farbangabe. #rrggbbaa, r,g,b,a, r g b a. Alpha optional. Beispiele: #14f900, 255,255,0,0, 0.5 0.4 0.3 0.2, GREEN, YELLOWISH	Bestimmt die Schriftfarbe der Spalte für gerade Zeilennummern.
`deferredRendering`	Boolean: true, **false**	**FIXME** aktuell ohne Funktion?
`debug`	Boolean: true, **false**	Aktiviert Info-Level Logging beim Rendern von Zellen.
`caching`	Boolean: **true**, false	Aktiviert oder deaktiviert das Caching von angezeigten Werten.
`style`	String: z. B. fontStyle="bold", fontStyle="italics" oder fontStyle="BOLD", fontStyle="italics"	
`justification`	String: `LEFT`, `CENTER`, `RIGHT`	
`sort`	`ASC`, `DESC`, `NONE`	Gibt an, in welcher Richtung die Spalte sortiert werden soll. NONE verbietet das Sortieren.

Name	Erlaubte Werte	Beschreibung
`sortLevel`	int: **0**	Gibt die Sortier-Reihenfolge der Spalten an.
`rendererClass`	String	Hier kann der Name einer TableCellRenderer-Klasse angegeben werden, die statt des Default Renderers benutzt werden soll.

headerRenderer und renderer

Hier können spezielle Renderer konfiguriert werden um z.B. die Farbe abhängig vom Inhalt der Zelle zu ändern. Sie müssen auf Spaltenebene definiert werden:

```
<Column property="Enabled">
   <renderer>
      import java.awt.Color
      renderer.setHorizontalAlignment(javax.swing.SwingConstants.CENTER)
      if (!value) { ①
         renderer.setBackground((row&amp;amp;1) == 0 ? new Color(234, 176, 176) : new Color(240, 200, 200)) ②
         renderer.setText('\u2717') ③
      } else {
         renderer.setBackground((row&amp;amp;1) == 0 ? new Color(199, 234, 176) : new Color(207, 226, 186))
         renderer.setText('\u2714')
      }
   </renderer>
</Column>
```

① value beinhaltet den Wert aus dem Property. In diesem Fall also `Enabled` als Boolean.

② Falls der Wert von `Enabled` false/null ist wird der Hintergrund abhängig von der Zeilen-Nummer in einem unterschiedlichen Rot-Ton eingefärbt.

③ Der Inhalt wird auf das Unicode-Zeichen gesetzt.

Wenn man am Background rumfuhrwerkt, muss man natürlich auch den "isSelected"-Status abfragen und den Background selbst entsprechend setzen (z.B. das gesetzte Rot bei markierten Zellen etwas bläulich einfärben).

DetailView

Die DetailView wird als Subelement der Table verwendet, um Details zum jeweils gerade innerhalb der Tabelle angeklickten Objekt anzuzeigen bzw. um das Bearbeiten der Attribute des Objekts zu ermöglichen. Sind von den Änderungen innerhalb der DetailView Inhalte der Tabelle betroffen, werden diese automatisch aktualisiert.

Achtung: Falls die DetailView für eine Tabelle in einem Lesezeichen verwendet wird, muss im `<table>` Element das Attribut `viewOnly="true"` gesetzt sein.

Attribute: FIXME

Name	Erlaubte Werte	Beschreibung
adjustableSplit	Boolean: true, **false**	Falls Wert auf **true**, kann der Bereich der DetailView mit der Maus verschoben, sprich vergrößert/verkleinert werden.
position	String: NORTH, **SOUTH**, WEST, EAST	
resizeWeight	Float: **0.0**	
scrollable	Boolean: true, false; Oder Richtungsbeschränkung, z. B. scrollable="VERTICAL_ONLY"	Scrollbar für dieses Element ein- oder ausschalten bzw. auf eine Richtung beschränken.

Sollen Attribute von Objekten aus Relationen des Tabellenobjekts geändert werden, müssen für diese jedoch onAfterSetValue-Hooks implementiert werden, der die Version des eigentlichen Objekts anstößt, um ein Aktualisieren der Tabelle zu erzwingen.

Beispiel mit onAfterSetValue:

```
<Table property="Buecher" columns="Titel | Erscheinungsjahr | Autor.Familienname">
  <DetailView name="autor">
    <Border etched="true" title="Details zum Autor">
      <View scrollable="true">
        <Element label="Familienname">
          <Text property="Autor.Familienname">
            <onAfterSetValue
language="groovy">ftx['autor'].getBO().bumpVersion()</onAfterSetValue>
          </Text>
        </Element>
      </View>
    </Border>
  </DetailView>
</Table>
```

MultipleChoiceFilterGUI

Hierbei handelt es sich um einen Filter, der via `<filter type="multipleChoice">` in eine FTable (Lesezeichen, Popup) eingebaut werden kann.
Für Hilfe beim Anlegen eines Filters die Solstice User-Dokumentation lesen.

Name	Erlaubte Werte	Beschreibung
preselectIdx	int	Gibt an, welcher Index in dem Dropdown standardmäßig ausgewählt ist.
sort	`ASC`, `DESC`, `NONE`	Gibt an, in welcher Reihenfolge die Queryergebnisse sortiert werden sollen

Text

Attributes:

Name	Erlaubte Werte	Beschreibung
`align`	String: `LEFT`, `CENTER`, `RIGHT`, `LEADING`, `TRAILING`	
`class`		Angabe der Klasse zur Textdarstellung, z.B. `ITextArea`
`disabled`	Boolean: true, **false**	Das Feld wird ausgegraut und kann nicht editiert werden.
`displayFormat` DEPRECATED		siehe `format`
`displayProperty` DEPRECATED		siehe `property`
`font`		
`foreground`	Farbangabe. Bitte entweder als "#rrggbbaa" oder "r,g,b,a", "r g b a" oder eine Farbkonstante der java.awt.Color, z.B. YELLOW angeben. Farbnamen mit Postfix "ISH" werden in Richtung weiss verschoben (Mittelwert der einzelnen Farbwerte und 255). Der Alphawert ist optional. Die einzelnen Werte sind bei den beiden letzteren Varianten entweder Float-Werte von 0.0..1.0 (bei 1 bitte 1.0 angeben!) oder Integer-Werte von 0..255. bitte nur die eine Sorte Werte verwenden. Oder fuer Random-Farbe: random.'	
`format`		
`lineWrap`	Boolean: **true**, false	Bricht Text an der rechten Kante um, statt eine Scrollbar anzuzeigen.
`password`	Boolean: true, **false**	Passwortfeld statt normalem Text.
`roundingFormat`		
`rows`	int: **4**	

Name	Erlaubte Werte	Beschreibung
`selectAllWhenFocused`	Boolean: true, *false*	Wenn das Form-Element den Fokus bekommt wird der gesamte Inhalt selektiert
`syncOnWait`		
`syncOnWaitDelay`		
`tabSize`	int: **4**	
`translationAvailable`		
`wrapStyleWord`	Boolean: **true**, false	Wenn Text umgebrochen wird, dann möglichst an Whitespace-Grenzen.

TimeSelector

Eingabe-/Bearbeitungsmöglichkeit für ein oder mehrere Zeitspannen innerhalb eines Zeitraums.

FIXME TT 2019-05-08: Wird praktisch nie benutzt.

Name	Erlaubte Werte	Beschreibung
`startingHour`	int: **0**-23	Erste mögliche auszuwählende Stunde des Tages
`rangeHours`	int: 1-n **(24)**	Anzahl der von `startingHour` ausgehend anzuzeigenden Stunden
`interval`	int: 5,10,**15**,20,30,60	Länge der Intervall-Aufteilung innerhalb einer Stunde
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`onRelease`	Element FIXME ???	FIXME ???

ToggleButton

Knopf der zwischen zwei Zuständen wechselt; Eingabe-/Bearbeitungsmöglichkeit für Boolean-Werte.

FIXME TT 2019-05-08: Wird praktisch nie benutzt.

Name	Erlaubte Werte	Beschreibung
`text`	String: ???	TT 2019-05-08: Scheint nicht genutzt/angezeigt zu werden.
`trueText`	String	Anzuzeigender Text wenn Attribute den Wert "true" hat
`falseText`	String	Anzuzeigender Text wenn Attribute den Wert "true" hat
`nullText`	String	Anzuzeigender Text wenn Attribute den Wert "null" hat
`displayProperty` DEPRECATED		siehe `property`
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`class`	FIXME ???	FIXME ???
`triState`	Boolean: **true**, false	Ob der Button Null-Werte unterstützen/anzeigen soll. FIXME TT 2019-05-08: Scheint nicht zu funktionieren.
`textExpression`	Element FIXME ???	FIXME ???

Tree

Auswahl von Einträgen für eine Relation, deren mögliche Einträge hierarchisch in einer Baumstruktur angeordnet sind.

Attributes:

Name	Erlaubte Werte	Beschreibung
`childrenProperty`	String. Bsp.: `childrenProperty="Kinder"`	Kinder-Attribut. Bei Subentitäten von Querbaum einfach "Kinder."
`displayClass` DEPRECATED		
`displayProperty` DEPRECATED		siehe `property`
`entity`	String: Entitätsname. Bsp.: `entity="Verzeichnis"`	Die Entität, deren Instanzen als Baum dargestellt werden sollen. Obligatorisch.
`filter`		ACHTUNG: Dieses Attribut wird im Moment noch nicht vom Tree unterstützt.
`format`	String: Attributname, z. B. `format="Bezeichnung"`	Das Attribut, das die Beschriftung der Knoten im Baum stellen soll. Default: `ui description`
`height`	Integer. Bsp.: `height="3"`	
`parentProperty`	String. Bsp.: `parentProperty="Elter"`	Eltern Attribut. Bei Subentitäten von Querbaum einfach "Elter."
`property`	String: Property accessor. Beispiele: property="Buch.Autor.Alter"; property="."	Das Attribut der aktuell betrachteten Entität, das im Kontext dieses Elementes verwendet werden soll. Im zweiten Beispiel wird das BO des aktuellen Formkontexts als Property gesetzt.
`restrictToEntity`		
`usePolymorphySelectionTree`	Boolean: true, **false**	
`width`	Integer. Bsp.: `width="15"`	

Uri

Attributes:

Name	Erlaubte Werte	Beschreibung
`autoaddProtocol`	Boolean: **true**, false	

View

Attributes:

Name	Erlaubte Werte	Beschreibung
`autoHideElements`	Boolean: **true**, false	
`border`		
`columns`	Integer, z. B.: `columns="3"`	Teilt die Inhalte der View auf die angegebene Anzahl Spalten auf.
`conflictPolicy`	String: '**IGNORE**', 'ASK'	Gibt an, wie Konflikte beim Speichern durch veraltete BOs behandelt werden sollen. Nur in der äussersten TabbedView / View erlaubt. IGNORE: Ignoriere Konflikte und übernehme/merge die Änderungen, ASK: Prüfe auf Konflikte und Frage den Benutzer wie damit umgegangen werden soll.
`debug`		
`defaultRightFill`	Double: **1**. Wert zwischen `0` und `1`. `defaultRightFill="1"`: maximale Breite; 0-1: Prozentuale Streckung, bevor die Streckung aufgegeben wird. Fallback ist `0`. Bsp.: `defaultRightFill="0.5"`	Verhindert Streckung von Feldern und benutzt `preferredSize`.
`delegateToParent`	Boolean: true, **false**	Die View übernimmt die Anzeigeeinstellungen (z.B. Anzahl columns) der übergeordneten View. Dadurch kann z.B. einheiltiche Anzeige der Formularelemente in verscheidenen Views erreicht werden.
`editable`	Boolean: **true**, false	Wenn false werden Kindelemente gesperrt, sodass diese nicht editiert werden können.
`externalHGap`	Integer mit Einheit px, c, em oder dlu. Bsp.: `externalHGap="3c"`	Erzeugt links und rechts der View den angegebenen Abstand.

Name	Erlaubte Werte	Beschreibung
`externalVGap`	Integer mit Einheit px, c, em oder dlu. Bsp.: `externalVGap="3c"`	Erzeugt über und unter der View den angegebenen Abstand.
`height`		
`hideElementsForNullBO`	Boolean: **true**, false	
`ignoreOtherLocalTransactionSaves`	Boolean: true, **false**	Nur für Formular-Roots. Änderungen aus lokalen Speichervorgängen im Client in diesem Formular nicht nachziehen
`implied`		
`internalHGap`	Integer mit Einheit px, c, em oder dlu. Bsp.: `internalHGap="3c"`	Fügt den angegebenen Abstand zwischen Labels und Feldern ein.
`internalVGap`	Integer mit Einheit px, c, em oder dlu. Bsp.: `internalVGap="3c"`	Fügt den angegebenen Abstand zwischen Zeilen ein.
`l10nBundle`		
`maximumSize`		Alias. Siehe `maxSize`
`maxSize`	Tupel: (horizontal, vertikal). Beispiele: maxSize="4c, 5c"; maxSize="4c,"; maxSize=",5c"	
`minimumSize`		Alias. Siehe `minSize`
`minSize`	Tupel: (horizontal, vertikal). Beispiele: minSize="4c, 5c"; minSize="4c,"; minSize=",5c"	Gibt die minimale Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`missingPropertiesPolicy`		
`name`	String	Um das Form-Element innerhalb des Formulars referenzieren zu können
`preferredSize`		Alias. Siehe `prefSize`

Name	Erlaubte Werte	Beschreibung
`prefSize`	Tupel: (horizontal, vertikal). Beispiele: prefSize="8c, 6c"; prefSize="8c,"; prefSize=",6c"	Gibt die bevorzugte Größe des Elements mit horizontalem und vertikalem Wert an und ersetzt die ansonsten automatische Größenberechnung. Diese würde das Element auf das Minimum an Platz für dessen Inhalt setzen. Beide Werte sind jeweils optional.
`scrollable`	Boolean: true, false; Oder Richtungsbeschränkung, z. B. scrollable="VERTICAL_ONLY"	Scrollbar für dieses Element ein- oder ausschalten bzw. auf eine Richtung beschränken.
`tint`	<Farbwert>[@<Intensität>], Intensität ist default 0.1; Beispiele: tint="#ff0000 @ 0.1"; tint="GREENISH"	Färbt den Hintergrund des Views in der angegebenen Farbe ein
`useMaximumHeight`	Boolean: true, **false**, z. B.: useMaximumHeight="true"	
`useMaximumWidth`	Boolean: true, **false**, z. B.: useMaximumWidth="true"	
`width`		

Datenaustausch

MyTISM bietet diverse Möglichkeiten des Imports als auch Export von Daten.

Import

FIXME

Export

Excel

Für den Datenaustausch nach Excel bzw. kompatiblen Tabellenkalkulationsprogrammen kommt intern die Apache POI Bibliothek zum Einsatz (https://poi.apache.org).

Etwas schöner benutzen lässt sich das mit dem XlsHandler, der in dem ein oder anderen Projekt bereits zur Anwendung gekommen ist (einfach über das Repository nach XlsHandler.nrx suchen). Dort gibt es drei Methoden: . Bauen von Header . Befüllen des Sheets . Den eigentlichen Export erstellen

Die Features, die man nutzen kann, findet man auf der Seite von Apache POI beschrieben. Grob gesagt geht "fast" alles:

- mehrere Sheets
- Styles
- Formeln
- Bilder
- ...

(Siehe die Tabelle unten hier: https://poi.apache.org/components/spreadsheet/index.html)

Bekannte Einschränkungen sind:

- wenige Charts
- keine Makros
- kein Pivot
- gewisse Größenbeschränkungen der Files

(Siehe auch https://poi.apache.org/components/spreadsheet/limitations.html)

Für das, was normalerweise benötigt wird, sollte es aber hoffentlich ausreichend sein.

Ein Beispiel für den Export findet man im Repository, wenn man nach der Klasse AngebotsgruppeGas.nrx sucht (Methode "exportSummenlastgaenge").

Natürlich kann man das auch in einem Groovy-Skript direkt im Client / einem Lesezeichen / einem Formular machen.

Zu überlegen ist, ob bei entsprechend häufiger Verwendung (u.a. auch direkt durch den Kunden), wir die interne API möglichst benutzerfreundlich gestalten, damit die Skripte nicht zu kompliziert werden.

Lokale autoritative sowie synchronisierende Instanzen zum Entwickeln aufsetzen

Manchmal benötigt man beim Entwickeln bestimmter Features - meist bei Core-Arbeiten - eine Konstellation von autoritativem Server und synchronisierendem/n Servern.

Hier eine stichwortartige Anleitung wie man solche eine Konstellation auf seinem lokalen Entwickersystem aufsetzen kann. Ausgegangen wird davon, dass eine Instanz - die autoritative Instanz - bereits existiert. Außerdem wird davon ausgegangen, dass beide Instanzen immer denselben Codestand nutzen sollen - falls nicht muss für die synchronisierende Instanz ein eigenes "deploy"-Verzeichnis angelegt werden, statt das der autoritativen Instanz zu verlinken.

Im folgenden wurde überall der Zusatz "syncnode" für Namen von zur synchronisierende Instanz gehörigen Dingen gewählt. Dieser ist aber prinzipiell beliebig bzw. das Verzeichnis könnte generell theoretisch beliebig benannt werden. Der besseren Zuordnung halber sollte man allerdings die hier beschriebene Variante wählen.

Folgendes ist dann zum Aufsetzen der zugehörigen synchronisierende Instanz zu tun:

- Verzeichnis `/.<projektkuerzel>-syncnode` anlegen (muss normalerweise als root/via sudo gemacht werden, um das /-Verzeichnis modifizieren zu dürfen)
- Besitzer korrekt setzen (wie vom Verzeichnis der autoritativen Instanz `/.<projektkuerzel>`)
- `/.<projektkuerzel>/deploy` nach `/.<projektkuerzel>-syncnode/deploy` verlinken
- `/.<projektkuerzel>/filesRoot` nach `/.<projektkuerzel>-syncnode/filesRoot` verlinken
- `/.<projektkuerzel>/lib/mytism` nach `/.<projektkuerzel>-syncnode/lib/mytism` verlinken
- Dateien `/.<projektkuerzel>mytism.ini` etc. nach `/.<projektkuerzel>-syncnode` kopieren
- `/.<projektkuerzel>-syncnode/mytism.ini` anpassen:
  - `nodeNumber` anpassen, falls Id der zugehörigen BN bekannt; oder Zeile komplett löschen, dann wird automatisch eine neue angelegt
  - (DB-)`url` anpassen: `<dbname>` → `<dbname>-syncnode`
  - `filesRoot` anpassen → `/.<projektkuerzel>-syncnode/filesRoot`
  - `authoritative=1` → `authoritative=0`
  - SyncService*-Einträge aktivieren:

```
[Sync]
url=socket://localhost:4242?compress=zlib9
user=Admin
pass=
```

Statt "Admin" und leerem Passwort bitte die korrekten Zugangsdaten aus der Datenbank eintragen.

  - `port` und `tlsPort` anpassen → z.B. statt 4242 und 4243 Ports 14242 und 14243 nutzen
  - `DeploySite`-Einstellungen anpassen oder neu eintragen, falls noch nicht vorhanden:

```
[DeploySite]
host=localhost
port=18080
--tlsHost=
--tlsPort=
requireAuthentication=0
--reauthEveryXDays=90
```

- Skript `mytism` anpassen:
  - `export PRJDIR="/.<projektkuerzel>-syncnode"`
  - `export DBNAME="<dbname>-syncnode"`

Danach kann, wie in der Admin-Doku unter "Aufsetzen eines syncenden MyTISM-Servers aus einem Backup des autoritativen Servers" beschrieben, ein Datenbank-Backup der autoritativen Instanz eingespielt und die beiden Instanzen genutzt werden.