Verschlagwortung mit Aspekten

[Stand: 01.07.2019]

Dieses Dokument beschreibt die Verschlagwortung mit Aspekten.

Inhalt

[1 Überblick 3](#_Toc13658981)

[1.1 Indexzeilen 3](#_Toc13658982)

[1.2 Visualisierung 3](#_Toc13658983)

[1.3 Vererbung 3](#_Toc13658984)

[1.4 Vererbung durch Workspace-ID 3](#_Toc13658985)

[2 Definitionen 4](#_Toc13658986)

[2.1 Aspekt, Aspekt-Objekt, Aspekt-Array 4](#_Toc13658987)

[2.2 Aspekt-Zuordnung 4](#_Toc13658988)

[2.3 Aspekt-Objekt-Tabelle 4](#_Toc13658989)

[2.4 Aspekt-Maske 4](#_Toc13658990)

[2.5 Workspace, Workspace-Sord, Workspace-Maske 4](#_Toc13658991)

[3 Datenstrukturen 6](#_Toc13658992)

[3.1 Übersicht 6](#_Toc13658993)

[3.2 Aspekt-Maske 7](#_Toc13658994)

[3.2.1 Workspace-Maske 7](#_Toc13658995)

[3.3 Vererbung unter Aspekt-Masken 7](#_Toc13658996)

[3.4 Aspekt-Zuordnung 7](#_Toc13658997)

[3.5 Aspect-Kardinalität 8](#_Toc13658998)

[3.6 Aspekt 9](#_Toc13658999)

[3.7 Formular zur Verschlagwortungsmaske, DocMaskForm 10](#_Toc13659000)

[3.8 Aspekt-Objekt-Tabelle 10](#_Toc13659001)

[3.9 Sord-Objekt 11](#_Toc13659002)

[3.9.1 Hilfsklasse AspectSord 12](#_Toc13659003)

[4 API-Funktionen 14](#_Toc13659004)

[4.1 createAspect 14](#_Toc13659005)

[4.2 checkinAspect 14](#_Toc13659006)

[4.3 checkoutAspect 14](#_Toc13659007)

[4.4 deleteAspect 14](#_Toc13659008)

[4.5 createAspectMask 14](#_Toc13659009)

[4.6 createAspectAssoc 14](#_Toc13659010)

[4.7 Beispiele 14](#_Toc13659011)

[4.7.1 Aspekt erstellen 14](#_Toc13659012)

[4.7.2 Aspekt-Maske erstellen 15](#_Toc13659013)

[5 Anhang 16](#_Toc13659014)

1. Überblick
   1. Indexzeilen

Zwischen Verschlagwortungsmasken und Indexzeilen soll eine neue Strukturierungsebene (Aspekt) eingeführt werden, die es erlaubt, von ELO vorgegebene Indexzeilen von projektspezifischen Indexzeilen zu trennen. Damit soll es einfacher werden, eine Verschlagwortungsmaske innerhalb eines ELO-Updates zu erweitern. Für diese neue Art von Verschlagwortungsmasken (Aspekt-Masken) entfällt das Konzept der Indexzeilenvorlagen.

* 1. Visualisierung

Eine Aspekt-Maske wird durch ein Formular in der Benutzeroberfläche dargestellt. Es werden keine Visualisierungsinformationen (z.B. Indexfeldpositionen usw.) gespeichert.

* 1. Vererbung

Aspekt-Masken können von beliebig vielen, anderen Aspekt-Masken erben.

* 1. Vererbung durch Region-ID

Ein Ordner, der mit einer Aspekt-Maske verschlagwortet ist, kann als Region-Ordner markiert werden. Seine ID wird zur Region-ID. Alle Einträge unterhalb dieses Ordners gehören zur gleichen Region, sie erben die ID automatisch (auch beim Verschieben wird sie automatisch aktualisiert).

Eine Suche kann über die individuelle Verschlagwortung wie auch über die Region- Verschlagwortung erfolgen.

1. Definitionen
   1. Aspekt, Aspekt-Objekt, Aspekt-Array

Der Begriff **Aspekt** wird hier als Schema für Daten, analog zu einer Klasse in der OOP, verwendet. Eine konkrete Realisierung im Speicher wird hier **Aspekt-Objekt** oder **Aspekt-Array** genannt.

Ein Aspekt fasst mehrere Indexzeilen zusammen.

Eine Indexzeile in einem Aspekt enthält höchstens einen Wert.

Beispiele:

* Aspekt Adresse mit Indexzeilen Name, Straße, Ort.
* Aspekt Rechnungsposition mit Indexzeilen Positionsnummer, Artikelnummer, Anzahl, Wert.
  1. Aspekt-Zuordnung

In einer Verschlagwortungsmaske können mehrere Aspekte enthalten sein. Eine solche Aggregation heißt hier **Aspekt-Zuordung**.

In einer Aspekt-Zuordnung wird u.a. die Kardinalität festgelegt: optional, erforderlich, optional-mehrfach, erforderlich-mehrfach.

* 1. Aspekt-Objekt-Tabelle

Einem Aspekt liegt in der Datenbank eine **Aspekt-Objekt-Tabelle** zugrunde. Jede Zeile stellt ein Aspekt-Objekt dar. In den Spalten der Zeile befinden sich die Indexwerte.

* 1. Aspekt-Maske

Eine Verschlagwortungsmaske, die Aspekte aggregiert, heißt hier **Aspekt-Maske**. Sie kann ihre Aspekte an andere Aspekt-Masken vererben. Ihre Visualisierung in der Client-Anwendung erfolgt durch ein Formular.

* 1. Region, Region-Sord, Region-Maske

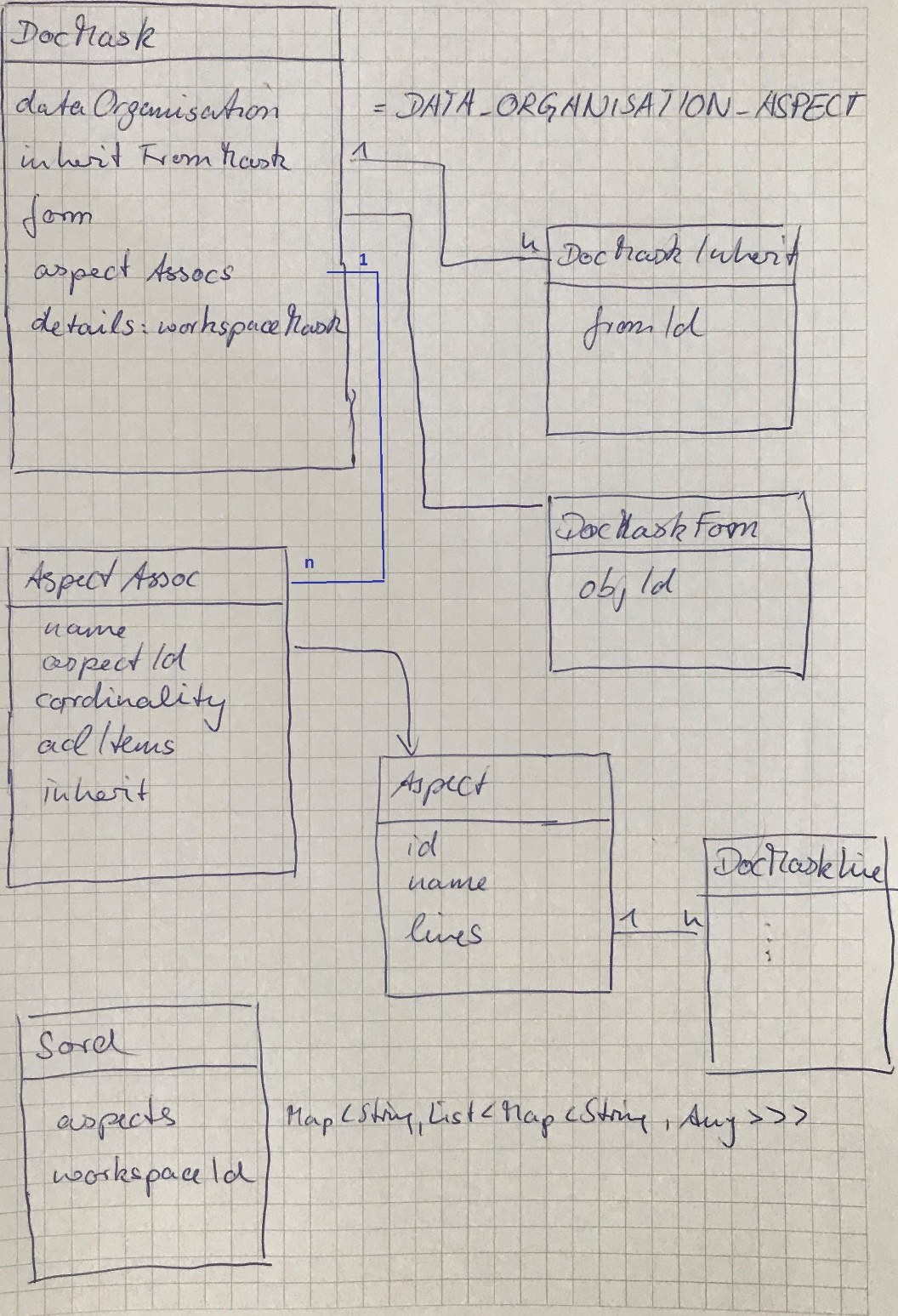
Ein Teilbaum im Archiv kann als **Region** gekennzeichnet werden. Dafür wird dem Startelement des Teilbaums, **Region-Sord**, eine Aspekt-Maske zugewiesen, die als **Region-Maske** vorgesehen ist.

Sord-Objekte in der Region erhalten zur Unterstützung von Suchfunktionen auf logischer Ebene implizit bestimmte Aspekt-Objekte aus der Region-Maske des Startelements. Insbesondere werden die Aspekt-Objekte zu allen Sord-Objekten der Region in der VT-Datenbank ergänzt. Die „geerbten“ Region-Aspekt-Objekte werden jedoch nicht in die Sord-Objekte kopiert.

**Relationen**

Zu prüfen wäre die Integration von vererbbaren Aspekten aus Relationen.

1. Datenstrukturen
   1. Übersicht



* 1. Aspekt-Maske

Eine Aspekt-Maske wird wie gewöhnliche Verschlagwortungsmasken durch ein DocMask-Objekt gespeichert. Das Element dataOrganisation legt fest, dass es sich um eine Aspekt-Maske handelt. Die Liste der beerbten Aspekt-Masken wird im Element inheritFromMaskIds. angegeben. Die zugeordneten Aspekte sind in aspectAssocs gespeichert. Es sind hierin auch die Aspekte der Masken eingetragen, von denen diese Maske erbt. ~~Das Formular zur Visualisierung ist in form enthalten.~~

class DocMask {  
…

int dataOrganisation; // = DocMaskC.DATA\_ORGANISATION\_ASPECT  
 List<DocMaskInherit> inheritFromMasks;  
 Map<String, AspectAssoc> aspectAssocs;

~~DocMaskForm form;~~

…

}

* + 1. Region-Maske

Das Element DocMask.details.regionMask kennzeichnet eine Aspekte-Maske als Region-Maske.

* 1. Vererbung unter Aspekt-Masken

class DocMaskInherit {

int parentMaskId;

//{{elm: protected ChildMaskId int childmaskid}}

}

Die Vererbungsbeziehungen werden in der Tabelle docmaskinherit gespeichert.

|  |  |  |
| --- | --- | --- |
| docmaskinherit | | |
| Spaltenname | Typ | Beschreibung |
| parentmaskid | int | ID der Maske, von der geerbt wird. |
| childmaskid | int | ID der Maske, die Aspektzuordnungen erbt. |

* 1. Aspekt-Zuordnung

Die Aspekt-Zuordnung ist wie folgt definiert:

class AspectAssoc {

String name;  
 String displayName;  
 String translationKey;  
 int aspectId;  
 Cardinality cardinality;  
 AclItem[] aclItems;   
 boolean regionInherit;

// {{ elm: protected maskId int maskid }};  
  
}

Das Element name legt einen technischen Namen fest – analog zum Gruppennamen einer Indexzeile. Der displayName ist der zum translationKey ermittelte Anzeigename aus der Übersetzungstabelle. Die aspectId verweist auf den zugeordneten Aspekt. In cardinality wird angegeben, ob das Objekt z.B. optional ist. Die Zugriffsberechtigungen legt aclItems fest.

Wenn die Aspekt-Zuordnung in einer Region-Maske enthalten ist, kann regionInherit gesetzt sein, um anzuzeigen, dass das Region-Sord das Aspekt-Objekt an alle Untereinträge weitergibt.

Aspekt-Zuordnungen speichert die Tabelle aspectassoc:

|  |  |  |
| --- | --- | --- |
| Aspectassoc | | |
| Spaltenname | Typ | Beschreibung |
| maskId | int | ID der Verschlagwortungsmaske |
| name | varchar | Name der Zuordnung |
| translationKey | varchar | Übersetzungsschlüssel |
| aspectId | int | ID des zugeordneten Aspekts |
| cardinality | int | Numerische Repräsentation der Kardinalität |
| acl | varchar | ACL |

* 1. Aspect-Kardinalität

Der Aufzählungstyp Cardinality legt fest, wie oft eine Aspekt-Zuordnung in der Verschlagwortung vorkommen kann bzw. muss.

enum Cardinality {

OPTIONAL, MANDATORY, OPTIONAL\_MANY, MANDATORY\_MANY

}

* 1. Aspekt

class Aspect {

int id;

String name;

String displayName;

String translationKey;  
 Map<String, DocMaskLine> lines;  
 String guid;  
 String TStamp;

boolean deleted; // status

String TStampSync;  
}

Das Element id kennzeichnet einen Aspekt eindeutig. (Eine ID wird benötigt, damit das Aspect ggf. umbenannt werden kann). In AspectAssoc.aspectId wird auf diese ID verwiesen. Einen technischen Bezeichner findet man in name. Der displayName ist der zum translationKey ermittelte Anzeigename aus der Übersetzungstabelle. In lines befinden sich die Indexzeilendefinitionen. Die Elemente guid, TStamp, TStampSync, status werden für die Replikation benötigt.

Die Definition eines Aspekts speichert die Tabelle aspect.

|  |  |  |
| --- | --- | --- |
| aspect | | |
| Spaltenname | Typ | Beschreibung |
| id | int | ID des Aspekts |
| name | varchar | Technischer Name |
| lockId | int | ID des Users, der das Objekt gerade blockiert |
| translationKey | varchar | Übersetzungsschlüssel |
| guid | varchar | GUID |
| tstamp | varchar | Zeitstempel |
| status | int | 0 für verwendbar, 1 für gelöscht. |
| tstampsync | varchar | Zeitstempel |
| acl | varchar | ACL |

Die Indexzeilen eines Aspekts liegen in der Tabelle aspectlines. Diese Tabelle ist wie masklines definiert.

Eine separate Tabelle hat den Vorteil, dass die Mechanismen für Indexzeilenvorlagen nicht greifen.

Die Tabelle aspectlines wird voraussichtlich um ein Flag ergänzt werden, dass als Entscheidung für die Erstellung eines DB-Indexes auf die betreffende Indexzeile (d.h. Spalte in der Aspekt-Objekte-Tabelle) dient.

* 1. ~~Formular zur Verschlagwortungsmaske, DocMaskForm~~

~~Das Formular zur Visualisierung der Verschlagwortungsdaten wird mit der Klasse DocMaskForm dargestellt.~~

~~class DocMaskForm {~~

~~String objId;~~

~~}~~

~~Mit objId wird ein Dokument im Archiv angegeben werden, das das Formular enthält. Es ist eine numerische ID, eine GUID oder ein Archivpfad.~~

* 1. Aspekt-Objekt-Tabelle

Eine Aspekt-Objekt-Tabelle wird mit „asp\_<aspectid>“ benannt, wobei „<aspectid>“ für die ID des Aspekts steht.

Eine Zeile speichert ein Aspekt-Objekt.

Die Aspekt-Objekt-Tabelle hat die folgenden Spalten:

|  |  |  |
| --- | --- | --- |
| aspect\_<id> | | |
| Spaltenname | Typ | Beschreibung |
| objid | int | Objekt-ID (Sord-ID) |
| regionobjid | int | Region-Sord-ID |
| ordinal | int | Wenn es sich um ein Aspekt-Array handelt, dann gibt diese Spalte den Array-Index an. Andernfalls ist der Wert 0. |
| assoc | varchar | Name der Aspekt-Zuordnung |
| maskId | int | ID der Ablagemaske. Dieser Wert kann für die Partitionierung der Tabelle hilfreich sein. |
| Indexwert…e |  | Für jede Indexzeile einen Wert. Der Typ wird von der Indexzeile festgelegt. |
| tstamp | String | Zeitstempel der letzten Änderung |

Die Spalte „regionobjid“ ist notwendig um später bei Verschlagwortungssuchen einen Join über a.regionobjid = b.objid bilden zu können. Sie wird gefüllt mit der objid des Region-Sord.

Auf bestimmte Indexzeilen (d.h. Spalten in der Aspekt-Objekt-Tabelle) werden wir in der Datenbank einen Index legen müssen. Hierfür müssen wir für jede Indexzeile entscheiden, ob ein Index notwendig ist. Voraussichtlich als Flag in der Tabelle aspectlines.

* 1. Sord-Objekt

Das Sord-Objekt wird erweitert um eine Map von Aspekt-Objekten.

class Sord {  
…   
 Map<String, List<Map<String, Any>>> aspects;

int regionId;  
…  
}

Die äußere Map liefert eine Liste von Aspekt-Objekten für eine gegebene Aspekt-Zuordnung. Die innere Map gibt den Indexwert zu einer Indexzeile zurück.

Beispiel in JavaScript:

var plz = sord.aspects.KUNDE[0].PLZ.intValue;

Beispiel in Java:

int plz = Sord.getAspects().get("KUNDE").get(0).get("PLZ").getIntValue();

Wenn das Sord-Objekt in einer Region enthalten ist, gibt regionId die numerische ID des Region-Sord an. Ist es das Region-Sord selbst, dann gilt Sord.id = Sord.regionId. Das Element regionId ist read-only und kann nur durch Verschieben geändert werden.

Die vom Region-Sord „geerbten“ Aspekt-Objekte werden nicht in der Sord.aspects zurückgegeben. (Weil sie in der Maske des Sords nicht definiert sind).

* + 1. Hilfsklasse AspectSord

Weil der Umgang mit Sord.aspects insbesondere beim Hinzufügen von Indexwerten unhandlich ist, schlage ich eine Art „Builder“-Klasse vor, mit der wie folgt auf die Indexwerte zugegriffen werden kann.

AspectSord aspectSord = new AspectSord(sord);

// Zuweisen mehrerer Indexwerte

aspectSord

.aspectObject("Kundenadresse")

.set("NAME", "Fritz Baum")

.set("STRASSE", "Wiesenweg 3")

.set("PLZ", 20030)

.set("ORT", "Hamburg")

.build()

.aspectArray("Rechnungspositionen")

.add()

.set("POS", 1)

.set("A\_NUMMER", 123)

.set("A\_NAME", "Schraube")

.set("ANZAL", 3)

.set("PREIS", 12.33)

.build()

.build()

.build();

// Zugriff auf einen Integer-Indexwert

int plz = 70101;

aspectSord.aspectObject("Kundenadresse").set("PLZ", plz);

plz = aspectSord.aspectObject("Kundenadresse").getInt("PLZ");

// Zugriff auf einen Integer-Indexwert im Aspekt-Array

int pos = 1;

aspectSord.aspectArray("Rechnungspositionen").get(0).set("POS", pos);

plz = aspectSord.aspectArray("Kundenadresse").get(0).getInt("POS");

// Schleife über Indexwerte eines Aspekt-Objekts

AspectObject<?> aspectObject =   
 aspectSord.aspectObject("Kundenadresse");

for (String lineKey : aspectObject.keySet()) {

Object value = aspectObject.get(lineKey);

if (value instanceof String) {

// ...

}

else if (value instanceof Integer) {

// ...

}

}

* + 1. Serialisierung der Hilfsklasse AspectSord <-> JSON

Wir haben relativ häufig den Anwendungsfall eine vereinfachte JSON-Syntax zu transformieren. Im einfachsten Fall betrifft das Templating-Engines oder Platzhalten in Word. Aktuell haben wir hierzu eine extra Klasse TemplateSord erstellt, die eine Transformation vornimmt. Schöner wäre es wenn das direkt vom AspectSord übernommen werden könnte.

Es wäre praktisch zwei Hilfsfunktionen toJson und fromJson zu haben.

AspectSord aspectSord = new AspectSord(sord);

aspectSord.toJson()

{

"name": "Sordname",

"aspects": {

"PARTNER": {

"$aspectId": 4123,

"$aspectType": "Adress",

"CITY": "Stuttgart",

"PLZ": "70179"

},

"ITEMS": [{

"$aspectId": 4123,

"$aspectType": "Adress",

"LINE\_DESC": "Artikel 1"

}, {

"$aspectId": 4123,

"$aspectType": "Adress",

"LINE\_DESC": "Artikel "

}]

}

}

1. API-Funktionen
   1. createAspect

Die Funktion initialisiert ein Aspekt. Der Aspekt wird nicht persistiert.

* 1. checkinAspect

Speichere einen Aspekt in der aspect-Tabelle. Lege die Aspekt-Objekt-Tabelle an, wenn sie noch nicht existiert oder ändere sie.

* 1. checkoutAspect

Lies einen Aspekt aus der DB.

* 1. deleteAspect

Die Funktion benennt den Aspekt in der Datenbank um und setzt ein Lösch-Flag. Die zugehörige Aspekt-Objekt-Tabelle wird ebenfalls nur umbenannt – nicht gelöscht.

Vorschlag: als neuer Name wird eine Zufallszahl genommen, die in der aspect-Tabelle als „status“ eingetragen wird.

* 1. createAspectMask

Initialisiere eine neue Aspekt-Maske. Die Aspekt-Maske wird nicht persistiert.

* 1. createAspectAssoc

Initialisiere eine Aspekt-Zuordnung. Die Aspekt-Zuordnung wird nicht persistiert.

* 1. Beispiele
     1. Aspekt erstellen

Aspect aspectAddress = insertAspect("Adresse",

createAspectLine("NAME", DocMaskLineC.TYPE\_TEXT),

createAspectLine("STRASSE", DocMaskLineC.TYPE\_TEXT),

createAspectLine("PLZ", DocMaskLineC.TYPE\_INTEGER),

createAspectLine("ORT", DocMaskLineC.TYPE\_TEXT));

log.info("aspectAddress={}", aspectAddress);

Aspect aspectInvoicePosition = insertAspect("Rechnungsposition",

createAspectLine("POS", DocMaskLineC.TYPE\_INTEGER),

createAspectLine("A\_NUMMER", DocMaskLineC.TYPE\_INTEGER),

createAspectLine("A\_NAME", DocMaskLineC.TYPE\_TEXT),

createAspectLine("ANZAL", DocMaskLineC.TYPE\_INTEGER),

createAspectLine("PREIS", DocMaskLineC.TYPE\_DOUBLE));

log.info("aspectInvoicePosition={}", aspectInvoicePosition);  
  
  
 private Aspect insertAspect(String name, DocMaskLine ...docMaskLines)  
 throws Exception {

Aspect aspect = conn.ix().createAspect(name,   
 Arrays.asList(docMaskLines));

int aspectId = conn.ix().checkinAspect(aspect, LockC.NO);

return conn.ix().checkoutAspect(Integer.toString(aspectId), LockC.NO);

}  
  
 private DocMaskLine createAspectLine(String key, int type)   
 throws Exception {

DocMaskLine dml = new DocMaskLine();

dml.setKey(key);

dml.setType(type);

return dml;

}

* + 1. Aspekt-Maske erstellen

AspectAssoc assocAddress =   
 conn.ix().createAspectAssoc("Kundenadresse", "Adresse",   
 Cardinality.MANDATORY);

AspectAssoc assocInvoicePosition =   
 conn.ix().createAspectAssoc("Rechnungspositionen",   
 "Rechnungsposition", Cardinality.OPTIONAL\_MANY);

DocMask dm = conn.ix().createAspectMask("Aspektmaske",   
 Arrays.asList(assocAddress, assocInvoicePosition));

int maskId = conn.ix().checkinDocMask(dm, DocMaskC.mbAll, LockC.NO);

1. Anhang

|  |  |
| --- | --- |
|  | Hinweis: beispiel. |

|  |  |  |  |
| --- | --- | --- | --- |
| Revisionsgeschichte dieses Dokuments | | | |
| Version | Datum | Bearbeiter | Änderungen |
|  |  |  |  |
|  |  |  |  |