ISA Dialog Manager

XML-Schnittstelle

A.06.03.b

Dieses Handbuch beschreibt die Schnittstelle zur Verarbeitung von XML-Daten (Extensible Markup Language) mit dem ISA Dialog Manager. Die dafür vorhandenen Objekte mit ihren Attributen und Methoden werden erklärt.

ISA Informationssysteme GmbH

Meisenweg 33

70771 Leinfelden-Echterdingen

Deutschland

Darstellungskonventionen

DM wird in diesem Handbuch synonym zu "Dialog Manager" verwendet.

Die Bezeichnung UNIX schließt generell alle unterstützten UNIX-Derivate ein - außer in den explizit angegebenen Fällen.

Dort wo für geläufige englische Fachbegriffe keine gängigen deutschen Übersetzungen existieren, wird zur Vermeidung von Unklarheiten der englische Begriff verwendet.

< >	muss durch einen entsprechenden Wert ersetzt werden
color	Schlüsselwort ("keyword")
.bgc	Attribut
{ }	optional (0 oder einmal)
[ ]	optional (0 oder n-mal)
<A> | <B>	entweder <A> oder <B>

Beschreibungsmodus

Alle Schlüsselwörter sind fett und unterstrichen, z.B.

variable **integer** **function**

Indizierung von Attributen

Syntax für indizierte Attribute:

[I]

[I,J] bzw. [row,column]

Identifikatoren

Identifikatoren müssen mit einem Großbuchstaben oder einem "Unterstrich" ('_') beginnen. Die weiteren Zeichen können Groß-, Kleinbuchstaben, Zahlen oder Unterstriche sein.

Der Bindestrich ('-') ist für die Benennung von Identifikatoren als Zeichen ***nicht*** zugelassen!

Die maximale Länge eines Identifikators beträgt *31* Zeichen.

Beschreibung der zugelassenen Identifikatoren in Backus-Naur-Form

<Identifikator> ::= <erstes Zeichen>{<Zeichen>}

<erstes Zeichen> ::= _ | <Großbuchstabe>

<Zeichen> ::= _ | <Kleinbuchstabe> | <Großbuchstabe> | <Ziffer>

<Ziffer> ::= 1 | 2 | 3 | … 9 | 0

<Kleinbuchstabe> ::= a | b | c | … x | y | z

<Großbuchstabe> ::= A | B | C | … X | Y | Z

Inhalt

Darstellungskonventionen ... 3

Inhalt ... 5

1 XML-Unterstützung ... 7

1.1 XML-Unterstützung unter Microsoft Windows ... 7
1.2 XML-Unterstützung unter Unix ... 7
1.3 Unterschiede zwischen Windows- und Unix-Implementierung ... 8

2 Verwendung von XML mit dem Datenmodell ... 9

2.0.1 Beispiel ... 10
2.0.2 Indexwert dopt_cache_data des Attributs dataoptions ... 12

3 Das XML-Dokument (document) ... 13

3.1 Attribute ... 14
3.2 Objektspezifische Attribute ... 15
3.3 Objektspezifische Methoden ... 16

4 Der XML-Cursor (doccursor) ... 18

4.1 Attribute ... 19
4.2 Objektspezifische Attribute ... 20
4.3 Objektspezifische Methoden ... 23
4.4 Muster für die Methoden :match() und :select() ... 24

5 Das transformer-Objekt ... 27

5.1 Attribute ... 29
5.2 Objektspezifische Attribute ... 30
5.3 Objektspezifische Methoden ... 31
5.4 Beispiel ... 32

6 Das mapping-Objekt ... 36

6.1 Attribute ... 36
6.2 Objektspezifische Attribute ... 37
6.3 Muster für .name ... 37
6.4 Objektspezifische Methoden ... 38

Index 39

1 XML-Unterstützung

Die XML-Unterstützung des IDM erlaubt die Verarbeitung von XML-Dokumenten in der Regelsprache. Kernbestandteil sind dabei die Objekt-Klassen ***document*** und ***doccursor***, die nachfolgend beschrieben sind.

Die Funktionalität der „XML-Transformation“ baut auf die XML-Schnittstelle auf und besteht aus dem Objekten ***transformer*** und ***mapping***. Sie dient zur Transformierung von XML-Daten nach IDM-Objekten/Attributen und umgekehrt.

XML und DOM sind Standards des World Wide Web Consortium (W3C). Die Spezifikationen können auf der Internetseite des W3C (**www.w3c.org/XML**, **www.w3c.org/DOM**) eingesehen werden.

1.1 XML-Unterstützung unter Microsoft Windows

Es werden die Microsoft XML Core Services (MSXML) Version 3.0 oder höher benötigt.

1.2 XML-Unterstützung unter Unix

Die XML-Schnittstelle ist ab IDM-Version A.05.02.d für Unix-Plattformen vorhanden, die über eine libxml2- und libxslt-Unterstützung verfügen.

Das sind die Plattformen:

- Solaris (ab Version 7)
- HP-UX 11.0 11.23 (32bit)
- AIX 5.1
- Red Hat 6.1, Red Hat 8
- RHEL4 i386 & x86_64
- Suse 9.1

Zu beachten ist, dass mindestens libxml2-Bibliotheken ab Version 2.7.2 (HPUX: 2.7.3) auf dem System installiert sein müssen. Für die Verwendung des ***Transformer***-Objekts muss zudem auch die libxslt-Bibliothek 1.1.24 verfügbar sein. Die Bibliotheken müssen nicht explizit hinzu gelinkt werden, sondern sollten über den Library-Suchpfad auffindbar sein (je nach Plattform LD_LIBRARY_PATH, SHLIB_PATH, LIBPATH) bzw. durch direktes Linken beim Bau der eigenen IDM-Applikation. Die Attribute *.idispatch* und *.ixmldomdocument2* liefern den xmlDocPtr und xmlNodePtr bzw. xmlAttrPtr entsprechend der angebundenen libxml2-Bibliothek. Eine Setzung ist nicht erlaubt.

1.3 Unterschiede zwischen Windows- und Unix-Implementierung

1. Die Processing-Instruction xml wird von libxml2 nicht als eigenständiger Knoten im DOM verwaltet. Dies hat Auswirkungen auf den Pfad, das Durchgehen mittels **:select** sowie beim Speichern des Dokuments.
2. Die Behandlung von Zwischenräumen ist unter Umständen unterschiedlich. Die libxml2-Bibliothek wird mit *keepBlanksDefault = 0* verwendet um unnötige Text-Knoten durch Knoten-Einrückungen oder Zeilenumschaltungen zu vermeiden und damit ein möglichst ähnliches Verhalten zur Windows-Implementierung zu erreichen.
3. Kleine Unterschiede zwischen MSXML und libxml2 im Hinblick auf Einrückung, Zeilenumschaltungen und Zwischenräume gibt es beim Auslesen des Attributs *.text* und bei der formatierten Speicherung.

2 Verwendung von XML mit dem Datenmodell

Verfügbarkeit

Ab IDM-Version A.06.01.b

XML lässt sich als Datamodel nutzen, wobei Knotentexte und Knotenattribute die Daten – in der Regel Strings – enthalten können. Ein XML-Dokument wird mit einem ***Doccursor*** an eine View gekoppelt. Dafür werden am ***Doccursor*** Datamodel-Attribute und Selektionsmuster für Knoten des XML-Dokuments definiert. Weiterhin lässt sich festlegen, ob das Datamodel-Attribut an den Inhalt oder einen Attributwert des Knotens gekoppelt ist. Bei Operationen, die das XML-Dokument ändern, werden die Datenänderungen an alle Datamodel-Attribute weitergereicht.

Der ***Doccursor*** besitzt folgende Attribute um ihn als Datamodel zu verwenden:

Tabelle 1: *Datenmodellattribute des Doccursors*

Attribut	Bedeutung
.dataselect	Definiert ein Datamodel-Attribut mit zugehörigem Selektionsmuster.
.dataselectattr	Ordnet ein Datamodel-Attribut einem Attribut der selektierten Knoten zu.
.dataselecttype	Datentyp-Konvertierung eines Datamodel-Attributs.
.dataselectcount	Definiert die Kardinalität eines Datamodel-Attributs.
.dataoptions	Steuert das Caching über den Index *dopt_cache_data*.

Dabei ist das Attribut *.dataselect* von zentraler Bedeutung. Es definiert die Datamodel-Attribute des ***Doccursors*** sowie deren Verknüpfung mit Knoten des XML-Dokuments mithilfe von Selektionsmustern. Die Syntax der Selektionsmuster entspricht den Musterdefinitionen der **:select**-Methode. Das Selektionsmuster eines *.dataselect*-Attributs ohne Index – also ohne Datamodel-Attribut – wird **vor** den Selektionsmustern aller Datamodel-Attribute angewendet. Dies ermöglicht relative Selektionsmuster für die Datamodel-Attribute, ausgehend von den Knoten, die vom *.dataselect*-Attribut ohne Index vorselektiert worden sind. Gleichzeitig lässt sich dadurch der Aufwand für das Sammeln der Daten reduzieren.

Beim Sammeln der Daten für ein Datamodel-Attribut wird das Dokument anhand des Selektionsmusters in einer Schleife durchlaufen. Dabei wird entweder der Knoteninhalt über das *.text*-Attribut des ***Doccursors*** oder der Wert des über *.dataselectattr* gesetzten Knotenattributs geholt.

Für die mit *.dataselect* definierten Datamodel-Attribute können mit den Attributen *.dataselecttype* und *.dataselectcount* der Datentyp und die Kardinalität geändert werden. Standardmäßig enthalten die Datamodel-Attribute Vektoren mit String-Werten (Datentyp *vector[string]*). Mit dem Attribut *.dataselecttype* kann ein Datentyp (z. B. *integer*, *boolean*) definiert werden, in den die Werte konvertiert werden. Die Kardinalität (Vektor oder Skalar) von Datamodel-Attributen kann über das Attribut *.dataselectcount* gesteuert werden. Wenn der Datentyp des Datamodel-Attributs eine Sammlung oder seine Kardinalität der Datentyp *integer* ist, dann enthält das Datamodel-Attribut einen *vector*, ansonsten nur einen Skalar mit dem ersten Wert.

Einmal geholte Werte eines Datamodel-Attributs werden zwischengespeichert bis sich entweder das XML-Dokument oder die Datenmodellattribute des ***Doccursors*** ändern. Dieses Zwischenspeichern („Caching“) kann durch Setzen von *.dataoptions[dopt_cache_data] = false* am ***Doccursor*** unterbunden werden.

Das Speichern von Daten in einem XML-Dokument erfolgt analog mit sinngemäß umgedrehten Operationen. Allerdings können im XML-Dokument keine Knoten automatisch angelegt oder gelöscht werden.

Hinweise

- Das Sammeln der Daten kann eine „teure“ Operation sein, da im gesamten XML-Dokument nach dem Selektionsmuster gesucht werden muss. Der Aufwand lässt sich möglicherweise reduzieren, indem die Suche mithilfe eines *.dataselect*-Attributs ohne Index auf vorselektierte Teilbäume beschränkt wird.
- Ein ***Doccursor***, der als Datenmodell verwendet wird, sollte nicht für andere Zwecke genutzt werden, da er typischerweise ständig seinen Selektionspfad ändert.

2.0.1 Beispiel

Eine Liste von Nobelpreisträgern soll aus folgender XML-Datei ausgelesen und in einer Tabelle dargestellt werden:

```
<?xml version="1.0"?>
<nobelprizes>
  <category id="p">Physics</category>
  <category id="c">Chemistry</category>
  <winner year="1" category="p">Wilhelm Conrad Röntgen</winner>
  <winner year="11" category="c">Marie Curie</winner>
  <winner year="18" category="p">Max Planck</winner>
  <winner year="70" category="c">Luis Leloir</winner>
</nobelprizes>
```

Die Namen der Preisträger werden aus dem Inhalt der XML-Knoten „winner“ geholt und im Datamodel-Attribut „.Winner“ abgelegt. Die Jahreszahlen der Auszeichnung stehen im Attribut „year“ der XML-Knoten und werden in der Datei ab 1900 gezählt. Sie werden zu *integer* gewandelt und als *vec-*

tor in das Datenmodell-Attribut „.Year“ eingelesen. In der überschriebenen **:represent**-Methode werden die Jahreszahlen für die Anzeige zu vierstelligen Zahlen ergänzt.

```
dialog D

document Doc
{
  doccursor DocCur
  {
    .dataselect[.Winner]    "..winner";
    .dataselect[.Year]      "..winner";
    .dataselectattr[.Year]  "year";
    .dataselecttype[.Year]  integer;
    .dataselectcount[.Year] integer;
  }
}

window Wi
{
  .title "Nobel prize winners";
  .width 300;  .height 220;

  tablefield Tf
  {
    .xauto 0;  .yauto 0;
    .datamodel DocCur;
    .colcount 2;  .rowcount 1;
    .rowheader 1;  .colheader 1;
    .rowheight[0] 25;
    .colwidth[0] 180;  .colwidth[1] 60;
    .content[1,1] "Year";
    .content[1,2] "Winner";
    .dataget[.field]     .Winner;
    .dataget[.userdata]  .Year;
    .dataindex[.userdata] [0,1];

    :represent()
    {
      variable integer I;
      if Attribute=.userdata then
        for I := 1 to itemcount(Value) do
          Value[I] := 1900 + (Value[I] % 100);
        endfor
        setvector(this, .content, Value,[2,1],
                  [1 + itemcount(Value),1]);
        return;
      endif
      pass this:super();
```

```
    }
  }

  on close { exit(); }
}

on start
{
  Doc:load("nobelprizes.xml");
}
```

Dieser Dialog erzeugt folgendes Fenster:

Abbildung 1: *Tabelle mit XML-Daten als Datenmodel*

2.0.2 Indexwert *dopt_cache_data* des Attributs *dataoptions*

Attribut-Index	Default	Komponente	Bedeutung
dopt_cache_data	***true***	Model	Dieser Indexwert ist nur für den ***doccursor*** verfügbar. **true** Die vom ***doccursor*** selektierten Daten werden für weitere Zugriffe zwischengespeichert („Caching“). **false** Der ***doccursor*** selektiert die Daten bei jedem Zugriff neu aus dem XML-Dokument.

3 Das XML-Dokument (document)

Das document Objekt ist der Behälter für ein XML-Dokument. Ein XML-Dokument wird als DOM-Baum gespeichert. Dieser DOM-Baum kann mit Hilfe eines Doccursor-Objekts, das ein Kind des Document-Objekts sein muss, durchwandert werden.

Definition

```
{ export | reexport } { model } document { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

Ereignisse

keine

Kinder

doccursor

document

record

transformer

Vater

application

canvas

checkbox

dialog

doccursor

document

edittext

groupbox

image

import

layoutbox

listbox

menubox

menuitem

menusep

messagebox

module

notebook

notepage

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

statusbar

tablefield

timer

toolbar

transformer

treeview

window

Menü

keins

Methoden

:load()

:save()

:transform()

:validate()

3.1 Attribute

doccursor[I]

document[I]

external

external[I]

firstrecord

idispatch

ixmldomdocument2

label

lastrecord

model

parent

real_version[enum]

record[I]

recordcount

scope

transformer[I]

userdata

version[enum]

xml

3.2 Objektspezifische Attribute

doccursor[I]

Über das doccursor Attribut kann auf den XML-Cursor des XML-Dokuments zugegriffen werden. Das Attribut wird mit dem Objektindex indiziert (ähnlich zu child).

idispatch

Über das idispatch Attribut kann unter Microsoft Windows auf den IDispatch COM Interface Pointer des XML-Dokuments zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

ixmldomdocument2

Über das ixmldomdocument2 Attribut kann unter Microsoft Windows auf den IXMLDOMDocument2 COM Interface Pointer des XML-Dokuments zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen. Der Dialog Manager wird ebenso abstürzen, wenn der angegebene Zeiger nicht auf ein COM Interface zeigt.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

xml

Über dieses Attribut kann man die String-Darstellung des XML-Dokuments erfragen. Wenn ein neuer Wert gesetzt wird, wird der gespeicherte DOM-Baum gelöscht und ein neuer DOM-Baum aus dem gesetzten Wert aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

3.3 Objektspezifische Methoden

:load()

Lädt ein XML-Dokument von der angegebenen Datei oder URL. Der gespeicherte DOM-Baum wird gelöscht und ein neuer DOM-Baum aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

:save()

Speichert XML-Dokument in Datei oder URL ab.

:transform()

Transformiert das XML-Dokument mit dem angegebenen Schema. Wenn das Ziel ein XML-Dokument ist, wird der gespeicherte DOM-Baum gelöscht und ein neuer DOM-Baum aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

Alternativ kann das Ziel der Transformation auch ein Text oder eine Datei sein. Ist das Resultat der Wandlung kein legales XML-Format, dann muss direkt in einen Text oder eine Datei gewandelt werden, da das Resultat nicht einem XML-Dokument zugewiesen werden kann. Dies ist zum Beispiel der Fall, wenn zu HTML gewandelt wird.

:validate()

Überprüft das XML-Dokument, ob es dem im XML-Dokument angegebenen Dokumenttyp entspricht. Ist kein Dokumenttyp enthalten, wird ein Fehler gemeldet.

4 Der XML-Cursor (doccursor)

Das ***doccursor***-Objekt ist immer ein Kind eines XML-Dokuments. Es verweist auf einen Knoten des DOM-Baums, der im XML-Dokument (dem Vater) gespeichert ist.

Mittels Methoden kann der Verweis auf einen anderen Knoten des DOM-Baumes gesetzt werden. Verständlicher formuliert bedeutet dies, dass der XML-Cursor mittels Methoden im DOM-Baum bewegt werden kann. Er bleibt natürlich ein Kind des XML-Dokuments.

Definition

```
{ export | reexport } { model } doccursor { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

Außer den „normalen“ Dialog Manager Attributen, besitzt der XML-Cursor Attribute, um auf Eigenschaften wie Name, Wert oder Attribute des DOM-Knotens zuzugreifen. Da dies Laufzeitattribute sind, sind diese nicht vererbbar. Außerdem sind viele dieser Attribut auch nur lesbar, da die entsprechende Eigenschaft des DOM-Knotens nicht geändert werden kann.

Der XML-Cursor ist zunächst ungültig, wird jedoch beim ersten Zugriff automatisch auf die Wurzel des DOM-Baums positioniert. Achtung, dies geschieht auch, wenn der XML-Cursor durch irgendeine Aktion ungültig geworden ist. Das Attribut *.mapped* liefert Auskunft darüber, ob der XML-Cursor gültig ist.

Ereignisse

keine

Kinder

document

record

transformer

Vater

document

Menü

keins

Methoden

:add()

:delete()

:match()

:reparent()

:select()

:transform()

4.1 Attribute

attribute[I]

attribute[string]

data

dataselect[attribute]

dataselectattr[attribute]

dataselectcount[attribute]

dataselecttype[attribute]

document[I]

external

external[I]

firstrecord

idispatch

ixmldomnode

ixmldomnodelist

label

lastrecord

mapped

model

name

nodetype

parent

path

publicid

record[I]

recordcount

scope

specified

systemid

target

text

transformer[l]

userdata

value

xml

4.2 Objektspezifische Attribute

attribute[l]
attribute[string]

Das Attribut attribute dient je nach Indizierung zum Abfragen des Namens oder des Wertes eines Attributs des DOM-Knotens.

Ist der Index eine Zahl, dann wird der Name des entsprechenden Attributs des DOM-Knotens geliefert. Es ist zu beachten, dass Attribute eines DOM-Knotens primär unsortiert sind.

Ist der Index ein String, dann stellt der Index den Namen eines Attributs dar und es wird der Wert des Attributs zurückgeliefert. Eine Zuweisung auf das mit einem String indizierte Attribut attribute legt ein entsprechendes Attribut am DOM-Knoten an. Eine Zuweisung eines Leerstrings löscht das entsprechende Attribute des DOM-Knotens.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

data

Dient zum Setzen und Abfragen der Daten des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_cdata_section* oder *nodetype_processing_instruction* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

dataselect[attribute] (ab IDM A.06.01.b)

Mit diesem Datenmodellattribut werden gleichzeitig das als Index angegebene Datamodel-Attribut und ein als Wert zugewiesenes Selektionsmuster für Knoten eines XML-Dokuments definiert.

dataselectattr[attribute] (ab IDM A.06.01.b)

Dieses Datenmodellattribut definiert, mit welchem Knotenattribut das als Index angegebene Datamodel-Attribut verknüpft ist.

dataselectcount[attribute] (ab IDM A.06.01.b)

Dieses Datenmodellattribut definiert die Kardinalität des als Index angegebenen Datamodel-Attributs.

dataselecttype[attribute] (ab IDM A.06.01.b)

Dieses Datenmodellattribut definiert den Datentyp, in den die Werte des als Index angegebenen Datamodel-Attributs konvertiert werden.

idispatch

Über das idispatch Attribut kann unter Microsoft Windows auf den IDispatch COM Interface Pointer des XML-Cursors zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

ixmldomnode

Über das ixmldomnode Attribut kann unter Microsoft Windows auf den IXMLDOMNode COM Interface Pointer des XML-Cursors zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

ixmldomnodelist

Über das ixmldomnodelist Attribut kann unter Microsoft Windows auf den IXMLDOMNodeList COM Interface Pointer des XML-Cursors zugegriffen werden. Über diesen Interface Pointer kann auf die direkten Kinder des XML-Cursors zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

mapped

Ist *true*, wenn der XML-Cursor auf einen Knoten im DOM-Baum zeigt. Es ist zu beachten, dass ein XML-Cursor, der auf keinen Knoten im DOM-Baum verweist, automatisch auf die Wurzel des DOM-Baums positioniert wird, wenn auf eines der objektspezifischen Attribute zugegriffen wird oder eine der objektspezifischen Methoden aufgerufen wird.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

name

Name oder auch Tag des DOM-Knotens.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

nodetype

Dient zur Abfrage des Typs des DOM-Knoten.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

path

Liefert eine String-Repräsentation für die Position des XML-Cursors im DOM-Baum. Mit diesem String kann die select Methode aufgerufen werden, um eine XML-Cursor auf den Knoten im DOM-Baum zu positionieren.

Wird der Wert des path Attributs an anderer Stelle gespeichert (zum Beispiel im userdata Attribut), dann ist zu beachten, dass dieser gespeicherte Wert nicht angepasst wird, wenn die Struktur des DOM-Baums verändert wird. Ein anschließender Aufruf der Methode select mit diesem gespeicherten Wert, wird demzufolge den XML-Cursor auf einen falschen DOM-Knoten zeigen lassen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

publicid

Öffentliche Kennung des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_entity* oder *nodetype_notation* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

specified

Gibt an, ob ein Attribut eines DOM-Knotens explizit angegeben wurde oder von einem Standardwert ererbt wurde. Das Attribut ist immer *true*, außer für den nodetype *nodetype_attribute*.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

systemid

Systemkennung des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_entity* oder *nodetype_notation* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

target

Name der Instruktion eines DOM-Knotens. Der Wert entspricht dem Wert des name Attributes. Das Attribut ist nur verfügbar, wenn der nodetype *nodetype_processing_instruction* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

text

Werte aller Unterknoten des DOM-Knotens. Es wird ein String geliefert, der den Text aller Unterknoten repräsentiert. Das Attribut ist hauptsächlich hilfreich, wenn man den Text eines XML-Elements benötigt und nicht erst zu dem Kindknoten, der den Text enthält, navigieren möchte.

Es ist zu beachten, dass das Setzen dieses Attributs alle Kindknoten löscht und einen neuen Textknoten anfügt.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

value

Wert des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_attribute*, *nodetype_text*, *nodetype_cdata_section*, *nodetype_processing_instruction* oder *nodetype_comment* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

xml

String-Darstellung des DOM-Knotens und all seiner Unterknoten.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

4.3 Objektspezifische Methoden

:add()

Fügt einen Kindknoten als letzten Knoten an den aktuellen DOM-Knoten an. Der XML-Cursor wird auf das neue Element gesetzt.

:delete()

Löscht den DOM-Knoten mit allen Kindknoten. Der XML-Cursor wird auf den Vaterknoten positioniert. XML-Cursor, die in den Unterbaum des gelöschten DOM-Knotens zeigen, werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

Werden die Werte des path Attributs an anderer Stelle gespeichert (zum Beispiel im userdata Attribut), dann ist zu beachten, dass diese gespeicherten Werte nicht angepasst werden, wenn die Struktur des DOM-Baums verändert wird. Ein anschließender Aufruf der Methode select mit einem dieser gespeicherten Werte, kann demzufolge den XML-Cursor auf einen falschen DOM-Knoten zeigen lassen.

:match()

Testet, ob der DOM-Knoten dem angegebenen Muster entspricht (siehe Kapitel „Muster für die Methoden :match() und :select()“).

:reparent()

Hängt den DOM-Knoten mit allen Kindknoten um.

Werden die Werte des path Attributs an anderer Stelle gespeichert (zum Beispiel im userdata Attribut), dann ist zu beachten, dass diese gespeicherten Werte nicht angepasst werden, wenn die Struktur des DOM-Baums verändert wird. Ein anschließender Aufruf der Methode select mit einem dieser gespeicherten Werte, kann demzufolge den XML-Cursor auf einen falschen DOM-Knoten zeigen lassen.

:select()

Bewegt den XML-Cursor in der angegebene Richtung oder bewegt den XML-Cursor auf den ersten DOM-Knoten, der dem angegebenen Muster entspricht (siehe Kapitel „Muster für die Methoden :match() und :select()“).

:transform()

Transformiert den XML-Cursor mit dem angegebenen Schema. Wenn das Ziel ein XML-Dokument ist, wird der gespeicherte DOM-Baum gelöscht und ein neuer DOM-Baum aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

Alternativ kann das Ziel der Transformation auch ein Text oder eine Datei sein. Ist das Resultat der Wandlung kein legales XML-Format, dann muss direkt in einen Text oder eine Datei gewandelt werden, da das Resultat nicht einem XML-Dokument zugewiesen werden kann. Dies ist zum Beispiel der Fall, wenn zu HTML gewandelt wird.

4.4 Muster für die Methoden :match() und :select()

Ein Muster ist ähnlich zu einem Dialog Manager Bezeichner. Das Muster bildet einen Pfad von Elementnamen. Der Pfad beginnt bei der Wurzel des DOM-Baums. Jede Hierarchiestufe wird mit dem entsprechenden Teil des Pfades verglichen. Zudem können noch bestimmte Eigenschaften wie Vorhandensein eines Attributs oder die Position innerhalb der Kinder bzw. des Vaters angegeben werden.

Im Muster ist prinzipiell jedes Zeichen mit Ausnahme der Zeichen **.**, **[**, **"**, **]**, **<**, **>**, **=**, ****, Tab, Leerzeichen und Zeilenumbruch zulässig. Soll eines der oben erwähnten Zeichen, .mit Ausnahme des Zeilenumbruchs, verwendet werden, dann muss ein **** vorangestellt werden. Zwischen zwei Anführungszeichen (**"**), also innerhalb eines Strings, sind zusätzlich auch die Zeichen **.**, **[**, **]**, **<**, **>**, **=**, Tab und Leerzeichen erlaubt. Es ist zu beachten, dass im Dialog Skript das Zeichen **** in einem String schon Escape-Zeichen ist, so dass im Dialog Skript immer **** anzugeben ist, wo ein **** benötigt wird. Ebenso muss im Dialog Skript **\"** angegeben werden, wo ein **"** benötigt wird.

```
{ <Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} }
  [ .<Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} ]
```

<Name>

Wird mit dem name Attribut des XML-Cursors verglichen. Der XML-Cursor muss den nodetype *nodetype_element* besitzen. Alternativ kann hier auch * angegeben werden, dann wird das name Attribute nicht beachtet.

Der Name eines XML-Elements beginnt mit einem Buchstaben oder einem Unterstrich und er kann Buchstaben, Ziffern, Bindestriche, Unterstriche und Punkte enthalten. Die genaue Definition für einen XML-Elementnamen kann in der XML-Spezifikation (**www.w3c.org/XML**) nachgelesen werden.

<Attr>

Der DOM-Knoten, auf den der XML-Cursor zeigt, muss das angegebene Attribut besitzen.

Der Name eines XML-Attributs beginnt mit einem Buchstaben oder einem Unterstrich und er kann Buchstaben, Ziffern, Bindestriche, Unterstriche und Punkte enthalten. Die genaue Definition für einen XML-Attributnamen kann in der XML-Spezifikation (**www.w3c.org/XML**) nachgelesen werden.

<Op>

Vergleichsoperator, um das in <Attr> angegebene Attribut mit dem <Value> zu vergleichen. Es kann auf gleich **=** und ungleich **<>** verglichen werden.

<Value>

Der Wert gegen den das <Attr> verglichen wird.

<Idx>

Der DOM-Knoten muss an dieser Position innerhalb der Kindknoten seines Vaterknotens stehen. Das erste Kind besitzt die Position 1.

Der <Idx> besteht nur aus Ziffern (**0** – **9**), die als Zahl interpretiert werden.

Besonderheiten

.

Beginnt das Muster mit einem Punkt, dann ist es relativ zum aktuellen DOM-Knoten. Der Pfad beginnt also beim aktuellen DOM-Knoten.

..

Zwei aufeinander folgende Punkte überspringen beliebig viele Hierarchiestufen.

[<Idx>]

Ein Index ohne weitere Angaben selektiert den DOM-Knoten an dieser Position. Der nodetype des Knotens bleibt dabei unberücksichtigt. Jeder DOM-Knoten kann somit durch einen Ausdruck der Form {**[**<Idx>**]**[.**[**<Idx>**]**]} eindeutig referenziert werden (path Attribut).

Erweiterung unter Microsoft Windows

Es kann auch XPath als Mustersyntax verwendet werden. Hierzu muss das Muster entweder mit / oder ./ beginnen. Die Verwendung von XPath Mustern ist wird nicht auf jeder Plattform unterstützt und ist somit nicht portabel.

5 Das transformer-Objekt

Das Transformer-Objekt ermöglicht den Durchlauf eines XML-Dokuments oder einer IDM-Objekthierarchie, wobei während dieses Durchlaufs an einzelnen Knoten semantische Aktionen ausgeführt werden können. Auf diese Weise kann leicht eine Transformation von Daten implementiert werden, bei der z.B. entweder XML-Daten in IDM-Objekte übertragen werden oder umgekehrt mit Daten, die in IDM-Objekten stehen, ein XML-Dokument generiert wird. Da die semantischen Aktionen durch benutzerdefinierte Codes beschrieben werden, ist die Art der Transformation prinzipiell beliebig.

Definition

```
{ export | reexport } { model } transformer { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

Zur Definition einer Transformation stehen dem IDM-Programmierer folgende Mittel zur Verfügung.

- Eine Transformation wird durch den Aufruf der :apply() Methode eines Transformer-Objekts gestartet. Als Parameter werden hier die Quelle und das Ziel der Transformation übergeben (z.B.: ein Doccursor-Objekt als Quelle und ein IDM-Objekt als Ziel, das dann die Daten aufnehmen soll oder diese weiter delegiert).
- Während jeder Transformation möchte man durch eine festgelegte Menge von Knoten, ob es IDM-Objekte oder Knoten eines XML-Baums sind, in einer bestimmten Reihenfolge durchlaufen. Die apply Methode realisiert einen solchen Durchlauf, indem sie ausgehend vom Startknoten (Quelle) in einer Schleife die select_next Methode des Transformers aufruft, die den Nachfolger des aktuellen Knotens bestimmt. Der Durchlauf wird beendet, sobald die select_next Methode *null* zurückliefert. Die Default-Implementierung der select_next Methode definiert eine Pre-Order-Reihenfolge. Der IDM-Programmierer kann in diesem Prozess an zwei Stellen eingreifen. Zum einen kann die apply Methode überdefiniert werden, um so den Startknoten festzulegen oder die Abbruchbedingung zu verändern. Zum anderen kann die select_next Methode überdefiniert werden und somit die gesamte Reihenfolge, in der die Knoten besucht werden.
- Nachdem sichergestellt ist, dass jeder Knoten irgendwann besucht wird, muss es eine Möglichkeit geben um festzulegen, an welchen Knoten welche semantische(n) Aktion(en) ausgeführt wird (werden). Zu diesem Zweck werden am Transformer-Objekt Mapping-Objekte als Kinder definiert. Jedes dieser Objekte definiert in seinem Attribut name ein Muster, das zur Entscheidung herangezogen wird, ob der gerade besuchte Knoten eine Aktion auslösen soll. Eine solche Aktion wird durch die action Methode des Mapping-Objekts festgelegt, die vom IDM-Programmierer überdefiniert wird. Diese Methode bekommt als ersten Parameter den gerade besuchten Knoten und als zweiten Parameter das Ziel-Objekt, das von der apply Methode des Transformers herrührt.

» Um das Bild zu vervollständigen: Das Transformer-Objekt hat ebenfalls eine action Methode. Diese Methode wird in der oben erwähnten Schleife der apply Methode für jeden besuchten Knoten aufgerufen und untersucht dabei, ob ein Muster eines der Mapping-Kinder auf den Knoten passt. Die Reihenfolge, in der das passiert, entspricht der Definitionsreihenfolge der Mappings beim Vater-Transformer. Die geerbten Mapping-Objekte werden als letzte untersucht. Wird bei der Untersuchung ein erstes passendes Mapping-Objekt gefunden, so ruft die action Methode des Transformers die action Methode dieses Mapping-Objekts auf. Diese kann dann die Daten vom aktuellen Knoten zum Ziel-Objekt übertragen. Als Rückgabewert kann diese Methode ein *true* zurückliefen. In diesem Fall wird angenommen, dass der Knoten komplett abgearbeitet worden ist und keine weiteren Mapping-Objekte untersucht werden sollen. Anderenfalls werden die übrigen Mappings untersucht und gegebenenfalls deren action Methoden aufgerufen, bis entweder alle Mappings abgearbeitet worden sind oder die action Methode eines von diesen *true* zurückgeliefert hat.

Ereignisse

keine

Kinder

document

mapping

record

transformer

Vater

application

canvas

checkbox

dialog

doccursor

document

edittext

groupbox

image

import

layoutbox

listbox

mapping

menubox

menuitem

menusep

messagebox

module

notebook

notepage

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

statusbar

tablefield

timer

toolbar

transformer

treeview

window

Menü

keins

Methoden

:action()

:apply()

:select_next()

5.1 Attribute

document[I]

external

external[I]

firstrecord

label

lastrecord

mapping[I]

model

module

parent

record[I]

recordcount

root

scope

userdata

5.2 Objektspezifische Attribute

mapping[I]

Über das .mapping Attribut kann auf die Mapping-Kinder zugegriffen werden. Das Attribut wird mit dem Objektindex indiziert (ähnlich zu child). Die Reihenfolge der Mappings in diesem Vektor bestimmt die Reihenfolge, in der während einer Transformation ein Knoten mit einzelnen Mappings verglichen wird. Die vererbten Mappings werden in diesen Vektor nicht übernommen. Während einer Transformation wird mit solchen Mappings erst zum Schluss verglichen, die direkten Instanzen haben also Vorrang. Das ist anders, als bei anderen vererbten Kindobjekten im IDM, die im Kindvektor der Vaterinstanz vorne eingefügt werden.

root

Nach dem Aufruf der apply Methode wird in diesem Attribut der Ausgangspunkt der Transformation gespeichert. Damit kann während einer Transformation entschieden werden, ob der Ausgangspunkt wieder erreicht worden ist und die Transformation beendet werden kann.

Dabei sind folgende Fallunterscheidungen zu beachten.

- Ist der Src-Parameter der apply Methode ein Document- oder Doccursor-Objekt so wird in .root ein String abgespeichert, der die entsprechende Position im XML-Baum beschreibt. Die Syntax dieses Strings entspricht der Konvention, die im .path Attribut des Doccursor-Objekts verwendet wird (siehe Objektbeschreibung zu doccursor). Vergleiche mit dem .path Attribut von Doccursor sind deswegen besonders einfach.
- Ist der Src-Parameter der apply Methode ein IDM-Objekt, so wird im .root dieses Objekt gespeichert. Folglich ist in diesem Fall der Typ dieses Attributs object.

Anhand des Wertes im .root Attribut entscheiden die action und select_next Methoden des Transformers, was diese tun müssen (siehe hierzu die Beschreibung dieser Methoden).

Am Ende der apply Methode wird *.root* wieder auf *void* gesetzt.

Default-Wert ist *void*.

5.3 Objektspezifische Methoden

:apply()

Mit dieser Methode wird die Transformation angestoßen. Die Default-Implementierung des Algorithmus sieht folgendermaßen aus:

» Ist der Src-Parameter ein Document- oder Doccursor-Objekt, so wird angenommen, dass Daten aus einem XML-Baum zum IDM übertragen werden sollen. Im Falle eines Document-Objekts wird ein temporäres Doccursor-Objekt erzeugt, das zur Navigation im XML-Baum benutzt wird. Im folgenden Pseudocode wurde der Einfachheit halber angenommen, dass in Src immer ein Doccursor übergeben wird.

```
:apply(anyvalue Src, anyvalue Dest)
{
  variable object NextObj;

  this.root ::= Src.path;
  NextObj := Src;
  while NextObj <> null do
    this:action(Src, Dest);
    NextObj := this:select_next(NextObj);
  endwhile
  this.root ::= null;
  return true;
}
```

» Ist der Src-Parameter ein IDM-Objekt (außer Document- oder Doccursor-Objekt), so wird angenommen, dass Daten vom IDM nach XML bzw. sonst wohin übertragen werden sollen. Der zugrundeliegende Code ist mit dem oben aufgeführten Codefragment identisch, außer das hier in .root nicht Src.path sondern Src selbst gespeichert wird. Also:

```
this.root ::= Src;
```

Die apply Methode kann überdefiniert werden (ähnlich zu init).

:action()

Diese Methode wird von der apply Methode des Transformers benutzt (siehe oben). Damit wird festgestellt, ob der aktuelle Knoten auf eines der Mapping-Kinder passt und deswegen die action Methode dieses Mapping-Objekts aufgerufen werden muss.

Die action Methode kann überdefiniert werden (ähnlich zu init).

:select_next()

Diese Methode wird von der apply Methode des Transformers zum Durchlaufen aller Knoten eines XML-Baums oder der IDM-Objekthierarchie benutzt. (siehe oben). Die Default-Implementierung dieser Methode sieht so aus, dass das wiederholte Aufrufen der Methode einen Pre-Order-Durchlauf realisiert.

Die select_next Methode kann überdefiniert werden (ähnlich zu init).

5.4 Beispiel

Dieses Beispiel ist im Verzeichnis *examples/xml* zu finden.

Als XML-Dokument dient folgende Datei mit dem Namen „CD-Katalog.xml"":

```
<?xml version="1.0" ?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  <CD>
    <TITLE>Maggie May</TITLE>
    <ARTIST>Rod Stewart</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Pickwick</COMPANY>
    <PRICE>8.50</PRICE>
    <YEAR>1990</YEAR>
  </CD>
</CATALOG>
```

Dann können die Daten aus dieser Datei beispielsweise wie folgt ausgelesen werden.

```
dialog D {}

window Wi
{
```

```
  .title "XML-CD-CATALOG";

  on close { exit(); }

  child treeview Tv
  {
    .xauto 0;
    .yauto 0;
    .style[style_lines]   true;
    .style[style_buttons] true;
    .style[style_root]    true;
    integer CdIdx := 0;

    rule NewCatalog() {
      if this.itemcount = 0 then
        this.itemcount ::= this.itemcount + 1;
      endif
      this.content[this.itemcount] := "CD-Catalog";
      this.open[this.itemcount] := true;
    }

    rule AddCD() {
      this:insert(this.itemcount+1, 4);
      this.CdIdx := this.CdIdx + 1;
      this.content[this.itemcount-3] := ""+this.CdIdx+". CD";
      this.level[this.itemcount-3] := 2;
    }

    rule AddTitle(string S input) {
      this.content[this.itemcount-2] := "Title: " + S;
      this.level[this.itemcount-2] := 3;
    }

    rule AddArtist(string S input) {
      this.content[this.itemcount-1] := "Artist: " + S;
      this.level[this.itemcount-1] := 3;
    }

    rule AddPrice(string S input) {
      this.content[this.itemcount] := "Price: " + S;
      this.level[this.itemcount] := 3;
    }
  }
}

transformer Tr
{
  !! transformer ist zur Übernahme der Daten aus einem XML-Dokument gedacht
```

```
  !! in Src ist deswegen immer ein doccursor zu erwarten

  child mapping MCatalog {
    .name "..CATALOG";

    :action() {
      Dest:NewCatalog();
      return true;
    }
  }

  child mapping MCD {
    .name "..CD";

    :action() {
      Dest:AddCD();
      return true;
    }
  }

  child mapping MTitle {
    .name "..CD.TITLE";

    :action() {
      Dest:AddTitle(Src.text);
      return true;
    }
  }

  child mapping MArtist {
    .name "..CD.ARTIST";

    :action() {
      Dest:AddArtist(Src.text);
      return true;
    }
  }

  child mapping MPrice {
    .name "..CD.PRICE";

    :action() {
      Dest:AddPrice(Src.text);
      return true;
    }
  }
}
```

```
document Doc {}

on dialog start
{
  Doc:load("CD-Katalog.xml");

  Tv.visible := false;
  Tr:apply(Doc, Tv);
  Tv.visible := true;
}
```

6 Das mapping-Objekt

Das Mapping-Objekt dient dem Zweck, eine semantische Aktion zu definieren, die während einer Transformation für einen bestimmten Knoten (in einem XML-Baum oder in der IDM-Objekthierarchie) aufgerufen werden soll, wenn eine Übereinstimmung zwischen diesem Mapping-Objekt und dem Knoten gefunden wird. Die Mapping-Objekte werden als Kinder eines Transformer-Objekts definiert und beschreiben zusammen mit diesem eine Transformation von Daten.

Definition

```
{ export | reexport } { model } mapping { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

Ereignisse

keine

Kinder

document

record

transformer

Vater

transformer

Menü

keins

Methoden

:action()

6.1 Attribute

document[I]

external

external[I]

firstrecord

label

lastrecord

name

model

parent

record[I]

recordcount

scope

transformer[I]

userdata

6.2 Objektspezifische Attribute

name

Hier wird ein Muster (ein XPath ähnlicher Ausdruck) angegeben, das definiert, auf welche Knoten in einem XML-Baum oder in der IDM-Objekthierarchie das Mapping-Objekt passt. Dieses bestimmt, ob während einer Transformation die action Methode für den Knoten aufgerufen wird oder nicht.

6.3 Muster für .name

Ein Muster ist ähnlich zu einem Dialog Manager Bezeichner. Das Muster bildet einen Pfad von Elementnamen. Der Pfad beginnt bei der Wurzel des Dokuments oder IDM-Objekthierarchie (also Dialog oder Modul). Jede Hierarchiestufe wird mit dem entsprechenden Teil des Pfades verglichen. Zudem können noch bestimmte Eigenschaften wie Vorhandensein eines Attributs oder die Position innerhalb der Kinder bzw. des Vaters angegeben werden:

```
{ <Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} }
  [ .<Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} ]
```

<Name>

XML: Wird mit dem name Attribut des Cursors verglichen. Der Cursor muss den nodetype *nodetype_element* besitzen. Alternativ kann hier auch * angegeben werden, dann wird das name Attribute nicht beachtet.

IDM: Wird mit dem Label eines IDM-Objekts verglichen. Alternativ kann hier auch * angegeben werden, dann ist jedes Label passend.

<Attr>

XML: Der XML-Knoten, auf den der Cursor zeigt, muss das angegebene Attribut besitzen.

IDM: Das IDM-Objekt muss das angegebene Attribut besitzen.

<Op>

XML und **IDM:** Vergleichsoperator, um das in <Attr> angegebene Attribut mit dem <Value> zu vergleichen. Es kann auf gleich **=** und ungleich **<>** verglichen werden.

<Value>

XML und **IDM:** Der Wert gegen den das <Attr> verglichen wird.

<Idx>

XML: Der XML-Knoten muss an dieser Position innerhalb der Kinder seines Vaters stehen.

IDM: Das IDM-Objekt muss an dieser Position innerhalb der Kinder seines Vaters stehen. Dabei werden alle Kinder aus hierarchischen Attributen zusammengefasst betrachtet.

Besonderheiten

.

XML: Beginnt das Muster mit einem Punkt, dann ist es relativ zum aktuellen Knoten. Der Pfad beginnt also beim aktuellen Knoten.

..

XML und **IDM:** Zwei aufeinander folgende Punkte überspringen beliebig viele Hierarchiestufen.

[<Idx>]

XML: Ein Index ohne weitere Angaben selektiert den XML-Knoten an dieser Position. Der nodetype des Knotens bleibt dabei unberücksichtigt. Jeder Knoten kann somit durch einen Ausdruck der Form {**[**<Idx>**]**[**.[**<Idx>**]**]} eindeutig referenziert werden (path Attribut).

Bei allen Vergleichen wird zwischen Groß- und Kleinschreibung unterschieden.

Beispiel

Das Muster „..CD[.Title = Yellow]“ referenziert innerhalb eines XML-Baums alle Knoten (egal wo in der Hierarchie) mit dem Tag „CD“, die ein Attribut „.Title“ mit dem Wert *Yellow* haben.

Das gleiche Muster auf IDM-Objekte angewandt wird alle Objekte mit dem Label „CD“ auswählen, die das benutzerdefinierte Attribut „.Title“ mit dem Wert *Yellow* haben.

6.4 Objektspezifische Methoden

:action()

Diese Methode wird von der **:action**-Methode des Vater-***Transformers*** aufgerufen, wenn eine Entsprechung zwischen einem Knoten in einem XML-Baum bzw. einer IDM-Objekthierarchie und dem Muster im *.name*-Attribut des ***Mapping***-Objektes gefunden wurde. Die Default-Implementierung dieser Methode tut nichts und liefert immer *true* zurück.

Da diese Methode überdefiniert werden kann, kann der IDM-Programmierer hier festlegen, was mit den Daten aus dem Knoten geschehen soll.

Index

*

* 25, 37

.

. 25, 38

.. 25, 38

A

action 31, 37
- Mapping 38
- überdefinieren 31

add 23

AddRef 15-16, 21

apply 27-28, 31
- doccursor 30-31
- document 30-31
- IDM-Objekt 30-31
- Src 30-31
- überdefinieren 27, 31

Attribut 20, 22, 25, 37
- attribute 20
- data 20
- dataselect 20
- dataselectattr 20
- dataselectcount 20
- dataselecttype 21
- doccursor 15
- idispatch 15, 21
- ixmldomdocument2 16
- ixmldomnode 21
- ixmldomnodelist 21
- mapped 16, 22-24
- mapping 30
- name 22-23, 25, 27, 37
- Name 20
- nodetype 22
- path 22-24
- publicid 22
- root 30
- specified 22
- Standardwert 22
- systemid 22
- target 23
- text 8, 23
- value 23
- Wert 20
- xml 16, 23

attribute 20
- Index 20

Attributname 25

B

Beispiel
- Datenmodell 10

C

Caching
- doccursor 12
- Doccursor 10

COM Interface Pointer 15-16, 21
COM Methode
AddRef 15-16, 21
Release 15-16, 21

D

data 20
Datamodel-Attribut
Datentyp 10
Kardinalität 10
dataoptions 12
dataselect 20
dataselectattr 20
dataselectcount 20
dataselecttype 21
Datenmodell
Beispiel 10
Doccursor 9
XML 9
Datentyp
Datamodel-Attribut 10
delete 23
doccursor 15, 18
add 23
attribute 20
Caching 12
data 20
dataselect 20
dataselectattr 20
dataselectcount 20
dataselecttype 21
delete 23
idispatch 21
ixmldomnode 21
ixmldomnodelist 21
mapped 22
match 24
name 22
nodetype 22
path 22
publicid 22
reparent 24
select 24
specified 22
systemid 22
target 23
text 23
transform 24
value 23
xml 23
Doccursor
Caching 10
Datenmodell 9
document 13
doccursor 15
idispatch 15
ixmldomdocument2 16
load 16
save 16
transform 16
validate 17
xml 16
Dokumenttyp 17

DOM-Baum 13, 16, 18, 24
Knoten 22
Wurzel 22, 24
DOM-Knoten 18, 20
Attribut 20, 22
Daten 20
Index 25, 38
Instruktion 23
Kennung 22
Kindknoten 23
löschen 23
Muster 24
Name 22
Position 25, 38
String-Darstellung 23
Systemkennung 22
Tag 22
transformieren 24
Typ 22
umhängen 24
Unterknoten 23
Wert 23
dopt_cache_data 10, 12

E

Einrückung 8
Elementname 24-25, 37
Escape-Zeichen 24

H

Hierarchiestufe
überspringen 25, 38

I

Identifikator 3
idispatch 7, 15, 21
IDM-Objekt 37
Label 37
Position 38
Index 25, 38
Instruktion 23
ixmldomdocument2 7, 16
ixmldomnode 21
ixmldomnodelist 21

K

Kardinalität
Datamodel-Attribut 10
keepBlanksDefault 8
Kennung, öffentliche 22
Kindknoten 23
hinzufügen 23
löschen 23
umhängen 24
Knoten 18, 22
Vergleich mit mapping 31
Knotentyp 20

L

Label 37
libxslt 7
liibxml2 7
load 16

M

mapped 16, 18, 22-24

mapping 30, 36

- Muster 37
- name 37

Mapping-Objekt 27

- action 27-28
- name 27
- Vergleich mit Knoten 31

Mapping-Objekte

- geerbte 28, 30
- Reihenfolge 30

match 24

- Muster 24

Methode

- action 27-28, 31, 37-38
- add 23
- apply 27, 31
- delete 23
- load 16
- match 24
- reparent 24
- save 16
- select 8, 22-24
- select_next 27, 32
- transform 16, 24
- validate 17

MSXML 7

Muster 24, 27, 37

- * 25, 37
- . 25, 38
- .. 25, 38
- Beispiel 38
- Escape-Zeichen 24
- mapping 37
- match 24
- name 37
- select 24
- Syntax 24, 37
- XPath 26

N

name 22-23, 25, 37

- Muster 37

Name 22

nodetype 20, 22-23, 25, 37-38

nodetype_attribute 22-23

nodetype_cdata_section 20, 23

nodetype_comment 23

nodetype_element 25, 37

nodetype_entity 22

nodetype_notation 22

nodetype_processing_instruction 20, 23

nodetype_text 23

O

Objekt

- doccursor 18
- document 13
- mapping 36

Objekthierarchie 36

- durchlaufen 27

öffentliche Kennung 22

P

path 22-24

Pfad 24, 37

  relativ 25, 38

Position 22, 25, 38

  String-Repräsentation 22

Pre-Order-Durchlauf 32

Pre-Order-Reihenfolge 27

Processing-Instruction 8, 23

publicid 22

R

Referenzzähler 15-16, 21

Release 15-16, 21

reparent 24

root 30

  Typ 30

S

save 16

select 8, 22-24

  Muster 24

select_next 27, 32

  Default-Implementierung 27, 32

  überdefinieren 27, 32

Selektionsmuster 9

semantische Aktion 27, 36

  definieren 36

specified 22

Src 30-31

String-Darstellung 16, 23

  Unterknoten 23

String-Repräsentation 22

systemid 22

Systemkennung 22

T

Tag 22

target 23

text 23

Textknoten 23

transform 16, 24

Transformation 16, 24, 27, 31, 36-37

  Ausgangspunkt 30

  HTML 16, 24

  Quelle 27

  Text 16, 24

  Ziel 27

transformer 27

  action 31

  apply 31

  mapping 30

  root 30

  select_next 32

Transformer

  Beispiel 32

Transformer-Objekt 7, 27

  action 28

  apply 28

Typ 22

U

Unterknoten 23
- hinzufügen 23
- löschen 23
- String-Darstellung 23
- umhängen 24

V

validate 17

value 23

Vergleichsoperator 25, 38

W

Wert 23, 25, 38

Wurzel 22

X

xml 16, 23

XML 17
- Datenmodell 9

XML-Attribut 25

XML-Baum 36

XML-Cursor 16, 18, 23-24
- bewegen 18, 24
- gültig 18
- name 25, 37
- Position 22

XML-Dokument 13
- durchlaufen 27
- laden 16
- speichern 8, 16
- String-Darstellung 16
- transformieren 16

XML-Element 25

XML Core Services 7

xmlAttrPtr 7

xmlDocPtr 7

xmlNodePtr 7

XPath 26

Z

Zeilenumschaltung 8

Zwischenräume 8
- Behandlung 8