
ZMS Professional Publishing

Einleitung

ZMS ist das professionelle Publishing-System für den elektronischen Wissenschafts-Verlag. Durch den Einsatz von kostenfreier Open Source Technologie, die dauerhafte und intensive Analyse der Autoren-Produktivität sowie das einzigartige Bedienkonzept hat sich ZMS etabliert als solides Werkzeug für Fachgesellschaften, Bildungsorganisationen und verteilte Unternehmen: von der Website über die technische Dokumentation auf CD bis hin zu Buch- und Zeitschriften-Produktionen ist ZMS das effektive Produktionswerkzeug für unternehmensrelevante elektronische Inhalte. ZMS passt sich flexibel den vorgegebenen Produktionsbedingungen an und unterstützt den Anwender mit so entscheidenden Funktionsmerkmalen wie

- Workflow
- Versionierung
- XML-Filter für Im- und Export
- Leicht bedienbares Redaktions-Interface

Das vorliegende Dokument beschreibt schwerpunktmäßig diejenigen System-Funktionen, die ZMS für größere Produktionen sowie die Integration von Datenbanken anbietet. Ziel dieser Darstellung ist es, dem fortgeschrittenen ZMS-Anwender mit Erfahrung in den Bereichen Python, Zope-Entwicklung, SQL und XML/XSL konkrete Lösungsansätze anzubieten und Perspektiven für die breitere Nutzung des Systems aufzuzeigen.

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Die ZMS-Entwicklung verläuft parallel zum 2er-Branch der Zope-Entwicklung und zeichnet sich in den Versionen ZMS 2.9 und 2.10 durch folgende System-Neuerungen aus:

1. Versionen-Management mit expliziter Versionierung
2. Reduktion der Datenbankgröße bei deaktiviertem Workflow um bis zu 40% durch Aufhebung der primären Objekt-Doppelung
3. Neue Default-Objekte für den optionalen Import:
 - 3.1. Neue Objekttypen: ZMSReference (erlaubt inline), ZMSLibrary
 - 3.2. Beispiele für ZMSReference: Literaturstelle, Fussnote, Glossar
 - 3.3. Anwender-Komentierungen für Content-Objekte (*LGreview*)
 - 3.4. Wikipage-Objekt
 - 3.5. Formular-Generator
 - 3.6. Double-Opt-In Newsletter
4. ZMI-Optimierungen
 - 4.1. AJAX-basierte Action-List für spürbar verbesserte Performance
 - 4.2. AJAX-basierte ZMI-Sitemap
 - 4.3. ZMI-Quicknavigation (basiert auf objMap-Methode)
 - 4.4. Inteface-Optimierung für geschachtelte Special Objects
5. ZMSelC
 - 5.1. SCORM 2004 Kompatibilität
 - 5.2. Optimierte Dokument-Navigation

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Inhalt

Einleitung	1
Optimierung der System-Performance	5
Allgemeine Maßnahmen zur Systemoptimierung	5
Hilfs-Methoden zur Identifikation von BLOBs.....	6
robots.txt.....	8
Caching-Konzepte: intern vs. extern	8
Literatur	15
SQL-Integration mit dem ZMS-Datenquelle-Objekt.....	16
Installation/Voraussetzungen.....	16
Übersicht Datenquelle-Objekt ZMSSqldb.....	18
query()-Funktion: SQL-Statements an die Datenbank senden	20
Konfiguration des Datenmodells: SQL-Tabelle(n) per XML-Code.....	21
Einsatz von Primär- und Fremdschlüssel: Relationen abbilden	24
Modellierung von n:m-Relationen: <i>intersection</i> -Tabelle	26
Multimultiselect: kombinierte Relation von n:m und 1:n.....	28
Clientseitige Validierung der Dateneingabe.....	31
Serverseitige Validierung während der Dateneingabe	32
Daten-Export	33
Literatur	33
ZMS-Filter: Konfiguration von XSL-Pipelines	34
Ablauf des Datenexportes.....	35
Systemvoraussetzungen.....	37
Interface für die Konfiguration von ZMS-Filtern.....	39
DocBook-basierte Export-Filter: PDF, Word, Windows-Help	41
Literatur/Links	52
Workflow	53
Workflow initialisieren.....	54
Modell: Aktive Workflow-Zustände und deren Übergänge	54
Workflow erstellen	55
Code-Beispiele	57
Workflow im Redaktionsinterface.....	59
Tipps und Tricks.....	60
Dokument-Versionierung mit ZMS	61
Workflow und Versionierung.....	61
System-Konfiguration: <i>Historie</i> aktivieren	62
Technisches Prinzip: Attribute-Set für Versionsinformationen	63
Versionen im Redaktions-Interface darstellen.....	64

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Master-Version: Versionen vereinheitlichen, Überflüssiges löschen.....	66
Einsatz der ZMS.netbook Formatvorlage.....	68
Übersicht.....	68
Einrichtung der Dokumentvorlage	68
Arbeiten mit der Dokumentvorlage.....	69
Die Formate im Detail.....	74
Detail-Einstellungen	83
Wichtige Tipps im Überblick.....	84
ZMS-Reference-Objekt: inline-Referenzen verwalten und einsetzen	86
Anwendungsbeispiel Literatur-Verwaltung	87
Anwendungsbeispiel Glossar	90
XML-Prozessierung von Literaturdatensätzen	95
Literaturreferenzen am Beispiel KPOH	95
Import der Daten nach ZMS.....	99
XSL-Code für die Transformation von MODS nach ZMS	101
Export von Literaturstellen in das Word2003-Format.....	104
LDAP integrieren mit LDAPUserFolder	107
Systemvoraussetzung: python-LDAP.....	107
Systemvoraussetzung: Zope-Produkt LDAPUserFolder.....	108
Anlegen des LDAPUserFolders	108
Konfiguration des LDAPUserFolders	109
Userrechte in ZMS konfigurieren.....	112
Literatur/Links	114
ZMS-Formulator	115
Content-Modell.....	116
Pre- und Postconditions für die Eingabe-Validierung	120
Prüfziffern-Check	124
Tipps, Tricks mit API-Funktionen.....	125
RSS-Newsfeed erzeugen	125
List-Mapping mit ZMS-API-Funktionen	127
Metaobjekt-Struktur iterieren	129
scalePix: Bild-Ausmaße für die Druckausgabe optimieren.....	130
Google Sitemap per ZMS-Aktion statisch erzeugen	133
onChangeObjEvt().....	135
ANHANG: Allgemeine Literatur-Empfehlungen.....	136

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

SQL-Integration mit dem ZMS-Datenquelle-Objekt

Um Datenbank-Inhalte in der ZMS-Website zu publizieren, bietet ZMS drei unterschiedliche Ansätze:

1. **ZMS-Datentabelle (*ZMSDatatable*):** generische ZMS-Lösung für die Archivierung kleinerer Datenmengen; die Datentabelle hält die Daten in Form eines python-typischen XML-Stroms komplett in einem einfachen Textfeld innerhalb der Zope/ZMS-Objekthierarchie. Das Objekt eignet sich auch für einfache relationale Datenmodelle.
2. **ZMS-Systemfolder (*ZMSSystemfolder*):** der „Systemorder“ kapselt beliebige Zope-Objekte und kann im Gegensatz zu diesen mit den Standard-Methoden in der Navigation erscheinen bzw. ist über das Redaktionsinterface leicht erreichbar. Hier können z.B. die ZOPE-SQL-Methoden zum Einsatz kommen.
3. **ZMS-Datenquelle (*ZMSsqldb*):** ist eine Schnittstelle für externe SQL-Datenbanken; *ZMSsqldb* erzeugt (halb-)automatisch ein Bearbeitungs-Interface, auch für relationale Datenmodelle.

Im Folgenden wird die Objektklasse *ZMSsqldb* genauer vorgestellt. Die Objektinstanz verhält sich im ZMS wie ein Block-Element und kann analog jedem Standard-Objekt an beliebiger Stelle in den ZMS-Objekt-Baum eingesetzt werden. Es liefert dem Redakteur den direkten Zugriff auf externe SQL-Datentabellen. Als Unter-Element innerhalb Spezieller Objekte mit spezifischer *manage_main*-Methode kann man auch komplexere Interfaces bzw. Maskenflüsse abbilden, wie sie z.B. bei der Pflege von Userdaten oder radiologische Falldaten nötig sein können.

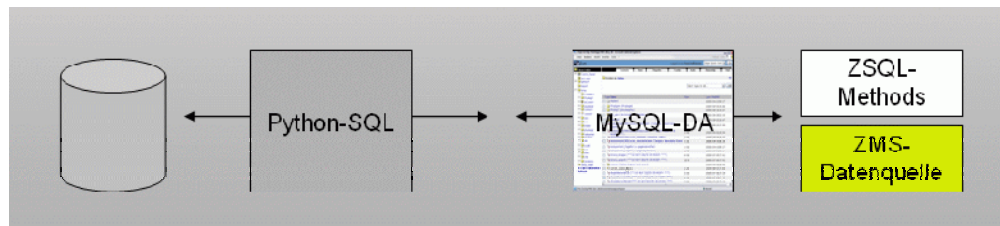
Installation/Voraussetzungen

Zope muss über den direkten Zugriff auf die Datenbank verfügen; dafür sind zwei Voraussetzungen zu erfüllen:

1. Zope braucht eine abstrakte, durch Scripting-Code ansprechbare Kommunikations-Schnittstelle für SQL; dies wird von einem Zope-Produkt (z.B. ZMySQLDA für MySQL oder ZpopyDA für Postgres) erledigt
2. Die Datenbank braucht ein entsprechendes Schnittstellen-Gegenstück, um die von Zope gesendeten Requests zu verarbeiten und wiederum mit Zope zu kommunizieren. Dafür wird eine datenbankspezifische Python API eingesetzt

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

(Python Postgres API, POPY für Postgres oder MySQL-Python-Adapter für MySQL)



Architektur der Datenbank-Connection

Folgende Websites liefern die Sourcen und Installations-Hinweise für den MySQL-Python-Adapter und den Einsatz des DA-Adapters in Zope:

1. Schritt für Schritt-Anleitung Suse 8.1
<http://alt.dzug.org/Members/wheider/howto/inst-Zope-MySQL-SuSE>
2. Download MySQL-Python-Adapter
<http://sourceforge.net/projects/mysql-python>
3. Binaries u.a. für Windows
http://www.zope.org/Members/isalsberg/Binaries/Useful_binary_collection
4. Connecting Zope to External Relational Databases
<http://www.sampublishing.com/articles/article.asp?p=24698&rl=1>

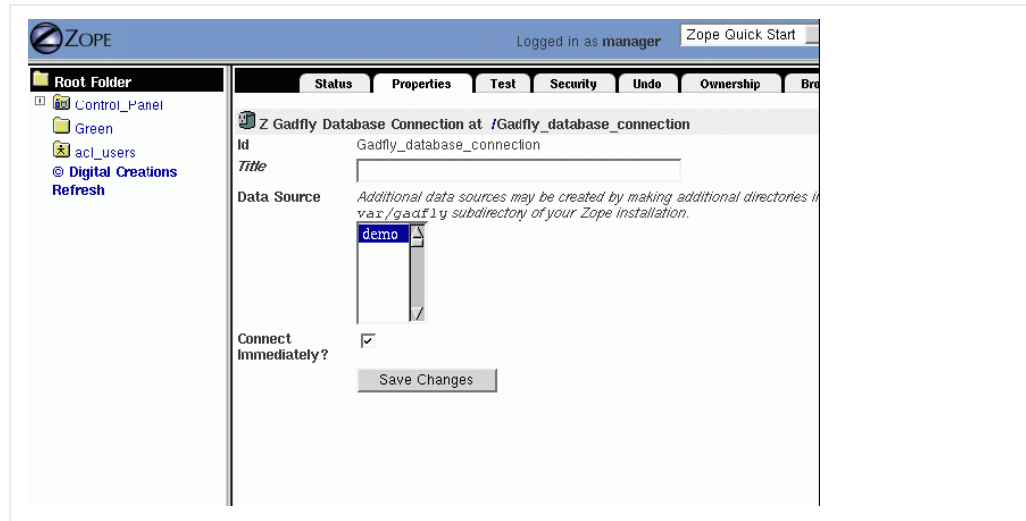
TIPP: Mit der Zope-eigenen Gadfly-Datenbank experimentieren

Im Installationsverzeichnis des ZMS-Produktes befindet sich der Ordner `./zms/import/zmssqldb/`. Dort ist ein SQL-Beispiel für die Zope-interne SQL-Datenbank *Gadfly* angelegt. Mit Hilfe von `zgadfly.sql` lässt sich über eine ZSQL-Methode ein kleiner Satz von Datenbank-Tabellen erzeugen.

Um eine Gadfly-Datenbank nutzen zu können, ist es lediglich erforderlich, im Zope-Instanz-Verzeichnis `./var/gadfly/instance` ein Verzeichnis für die Datenbank anzulegen. Die dort befindlichen Verzeichnisnamen erscheinen in einer Auswahl des Gadfly-Connectors. Über das Test-Menü lässt sich der SQL-Code zur Erzeugung der Tabellen ausführen.

Sobald eine GadflySQL-Connectors konfiguriert ist, kann die entsprechende Datenbank über jedes `ZMSSqlDb`-Objekt angesprochen werden. *Wichtig:* Gadfly ist nicht in der Lage auto-incrementelle Datenfelder zu füllen. Das kann bei der Neuanlage von Datensätzen einen SQL-Fehler bedingen, sofern dieser Datentyp im Datenmodell definiert ist.

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing



Im instance-Verzeichnis `./var/gadfly/` befindet sich ein Ordner „demo“; dieser erscheint dann in der Auswahl für die data source. Die damit verfügbare Datenbank kann nun mit Datentabellen gefüllt werden. Hinweis: Weitere Details zur Anlage einer Gadfly-Datenbank befinden sich in der der ZoPe-Dokumentation bzw. unter

<http://www.sampublishing.com/articles/article.asp?p=24698&seqNum=8&rf=1>

Übersicht Datenquelle-Objekt ZMSSqldb

Das Datenquellen-Objekt ermöglicht eine einfache Integration von SQL-Datenbank-Inhalten in das ZMS-Interface, indem die Datentabellen als filter- und editierbares *Grid* dargestellt werden. Darüber hinaus kann das Objekt per DTML SQL-Ausdrücke an die Datenbank weiterleiten bzw. Ergebnislisten zurückliefern. Um das Datenmodell korrekt abzubilden, wird dieses im Konfigurationsmenü der SQL-Datenquellen nachgebildet. ZMSSqldb erwartet eine generische Minimalkonstruktion der Datenbank (Tabellen- und Feld-Namen, sog. Entitäten) in Form einer *entities*-Liste. Tatsächlich kann die per SQL vermittelte Abfrage nach den *entities* einer Datenbank je nach Datenbank-Typ variieren; sofern also die Standard-Methode (die abgestimmt auf MySQL ist) die Entitäten nicht korrekt zurückliefert, kann man eine eigene DTML-Methode `getEntities<connection_id>` in den Objektbaum legen, die dann vom ZMSSqldb-Objekt akquiriert wird. Die Namensergänzung `<connection_id>` ist die ID des jeweiligen Datenbank-Connectors. Hiesse die Connection `odbcda`, so muß die Methode `getEntitiesodbcda` heißen. Beispiele für `getEntities`-Methoden existieren für Oracle und Postgres im Produkt-Folder:

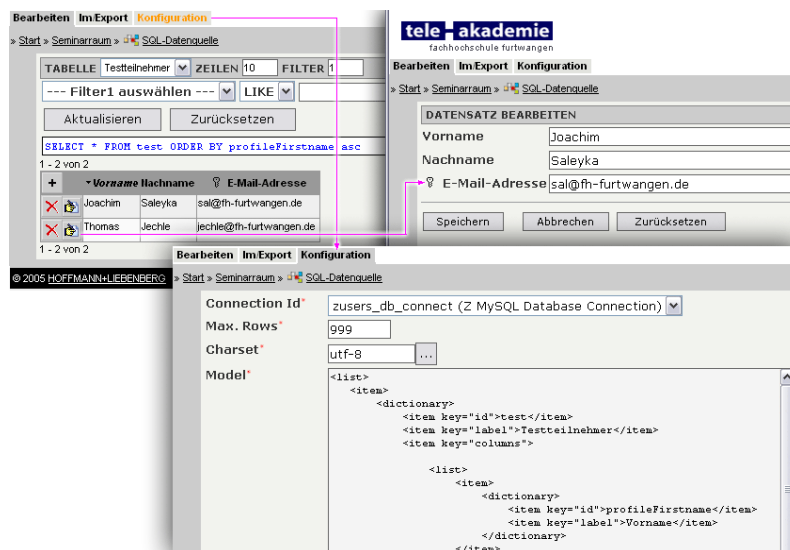
Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Name	Größe	Zuletzt geändert
zms		2007-03-22 10:10
conf		2007-03-22 10:10
db		2007-03-22 10:10
getEntities.Oracle.dtml	2203	2007-03-07 07:49
getEntities.Postgres.dtml	2651	2007-03-07 07:49

Beispiel-Code für die Schnittstellen-Methode `getEntities` (Wichtig: es handelt sich nur um Beispiel-Codes; für die lokale Methode unbedingt den Punkt weglassen, z.B. `getEntitiesoracle`, wenn die DB-Connection die ID `oracle` hat).

Anlegen eines neuen ZMSSqldb-Objekts

In der Regel ist das ZMSSqldb-Objekt in der System-Konfiguration nicht als aktives „insertable Object“ konfiguriert; das muss unbedingt nachgeholt werden, damit in jedem Knoten ein Datenquellen-Objektes angelegt werden kann. Bei der Anlage einer Objekt-Instanz ist zunächst aus der Liste der verfügbaren SQL-Connection die relevante ID auszuwählen.



ZMS generiert ein Pflege-Interface auf externe relationale Datenbanken. Im Konfigurationsmenü des ZMSSqldb-Objektes wird das Datenmodell per XML abgebildet. Die so formulierten Tabellen erscheinen dann (in verknüpfter Weise) im Bearbeiten-Menü in einem entsprechenden Daten-Grid.

Interface

Das Einstiegs-Interface liefert einen Überblick über die durch den Datenbank-Connector zugänglichen Daten-Tabellen und erlaubt eine tabellarische Darstellung der Datensätze. Die Menge der Datensätze und deren Auswahl kann durch Angabe der Zeilenzahl und mit einer beliebigen Anzahl von Filtern beeinflusst werden. Für die Filterung der Datensätze lassen sich die üblichen SQL-Wildcards einsetzen:

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

- „%“ für eine beliebige Anzahl beliebiger Zeichen und
- „_“ für ein einziges Zeichen.

Beispiel für den Einsatz von zwei Filtern (bezogen auf die Felder ‚Term‘ und ‚Parent‘) und Wildcard-Matching: das Ergebnis zeigt alle die Level-1-Elemente eines medizinischen Thesaurus an. Wichtig: Das Symbol „Datendatz editieren“ erscheint nur, wenn per Konfigurations-Menü ein Datenfeld als PRIMARY KEY deklariert wurde.

query()-Funktion: SQL-Statements an die Datenbank senden

Das ZMSSqlDb-Objekt kann mit seiner API-Funktion `query()` im DTML-Code SQL-Statements ausführen. Es kann per SQL aus einer Datenbank-Tabelle Listen bzw. Tupel generieren und erlaubt z.B. Iterationen, wie sie im Folgenden dargestellt sind:

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

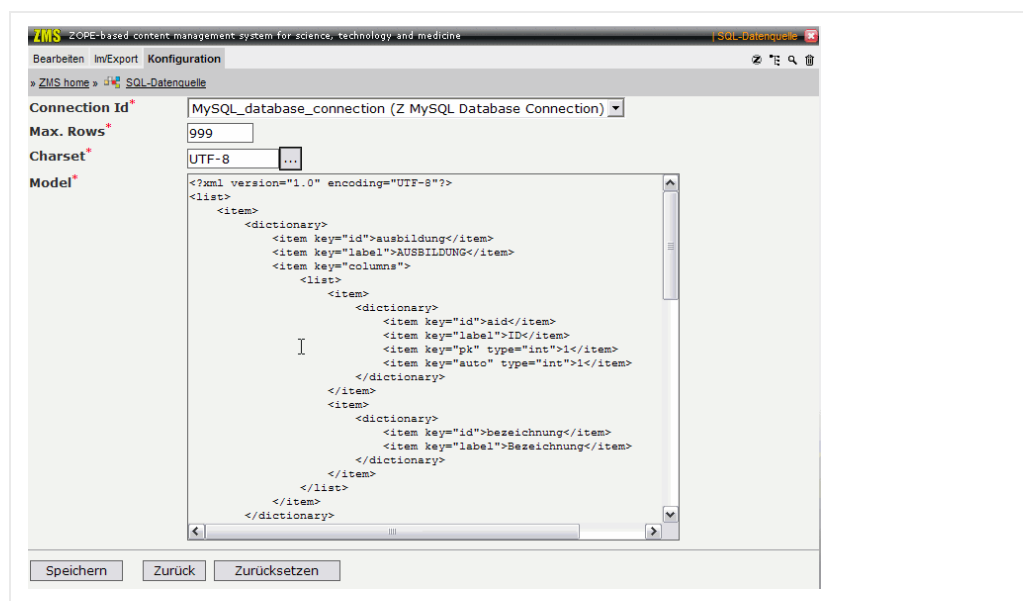
```

1 <dtml-let res="e2.query('SELECT * FROM TEST') ">
2   <dtml-in "res['columns'][0:2]" mapping>
3     <dtml-var "_['sequence-item']['id']" >
4   </dtml-in>
5   <hr/>
6   <dtml-in "res['records']">
7     <dtml-var "_['sequence-item']['NAME']">,
8     <dtml-var "_['sequence-item']['VORNAME']"><br />
9   </dtml-in>
10 </dtml-let>

```

In Zeile 1 wird das zu iterierende Objekt über ein SQL Statement erzeugt (in diesem Fall die *gesamte* Tabelle). Die erste Iteration (Zeile 2) erzeugt eine Liste der ersten drei Spaltenbezeichner (*columns*) und gibt deren IDs aus. Die zweite Iteration (Zeile 6) durchläuft die Datenfelder (*records*) und gibt in diesem Beispiel die Felder mit den IDs „Name“ und „Vorname“ aus.

Konfiguration des Datenmodells: SQL-Tabelle(n) per XML-Code



ZMSSqldb-Konfiguration: die Verbindung zur Datenbank wird über die Auswahlliste der in der Hierarchie verfügbaren Zope-Datenbank-Connection-Objekt hergestellt. Die maximale Anzahl der Datensätze (max. Rows) dient bei großen Datenmengen dazu, den (ungefilterten) Datenstrom zwischen ZMS/Zope einzuschränken. Die Darstellung der Tabellen im ZMI (Re-Modellierung des Datenbankmodells) wird gesteuert über eine XML-basierten Konfigurationstext, die man hier frei editieren kann (XML- Struktur siehe weiter unten).

Die *Re-Modellierung* des SQL-Datenmodells in ZMS erfolgt python-typisch in Form einer Liste. Diese Liste repräsentiert das Datenmodell der gesamten Datenbank bzw. derjenigen Daten, für die das ZMSSqldb-Objekt ein Interface generieren soll. Im XML-Code schachtelt das primäre *list*-Element eine Reihe von *item*-

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Elementen, welche die Tabellen bzw. das Datenmodell im Detail beschreiben. Eine Tabelle ist modelliert als *dictionary*; jeder Eintrag ist – ebenso wie bei der Liste - ein *item*-Element, allerdings besitzt jedes *item* einen Schlüssel (*key*-Attribut) und einen Wert (Inhalt des *item*-Elements).

Eine Tabelle wird anhand von fünf Eigenschaften (*items*) spezifiziert:

1. id (ID)
2. label (Feldbezeichner) [string]
3. type (Deklaration einer intersection-Tabelle für n:m-Relationen)
4. interface (Java-Script-Element für clientseitige Eingabe-Validierung)
5. columns (Spalten der darzustellenden Daten-Grids)

Die Spalten der Tabelle (*columns*-item) werden wiederum als Liste dargestellt, die für jedes Element der Tabelle (Spalte) ein *item*-Element definiert, das die Spalten-Eigenschaften (Datentypen) in Form von Schlüssel-Wert-Paaren enthält. Für die Beschreibung der Datentypen einer Spalte stehen folgende Attribute zur Verfügung:

1. id (ID)
2. auto (automatischer, DB-generierter Wert) [int: 0|1]
3. label (Feldbezeichner) [string]
4. pk (primary key, Primärschlüssel) [int: 0|1]
5. fk (foreign key, Fremdschlüssel) [string]
bildet Container für tablename, fieldname, displayfield
6. password [int: 0|1]. maskiert Inhalt des Eingabefeldes
7. hide [int: 0|1], verhindert Anzeige im Übersichts-Grid
8. checkbox [int: 0|1]
9. multiselect [dictionary] Mehrfachauswahl
schachtelt fk-Element für (intersection-basierte) Mehrfachauswahl
10. options (Einfachauswahl) [dictionary]
schachtelt Liste von Options-Werten
11. index (Position des Elements in der Darstellung-Reihenfolge) [int]
12. displayfield (Feldname, der statt des fk-Feldes angezeigt wird) [string]
13. type [string: text | date | int etc.]
14. multimultiselect (Container für Auswahlkombinationen, s.u.)
15. tables (Container für eine Liste von Tabellenreferenzen, die für die Kombination von Auswahlfeldern, multimultiselect herangezogen werden).

Damit eine Tabelle per ZMI überhaupt editierbar ist, muss zumindest ein Element mit eindeutigem Primärschlüssel (*pk*) definiert sein. Beispielsweise sieht eine einfache Tabelle „persons“ aus ID (primary key), Name und Vorname in ihrer XML-Modellierung folgendermaßen aus:

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

```

<list>
<!-- #####
      ### persons
      ##### -->
<item>
<dictionary>
<item key="id">persons</item>
<item key="label">Personen</item>
<item key="columns">
<list>
<item>
<dictionary>
<item key="id">id</item>
<item key="label">ID</item>
<item key="pk" type="int">1</item>
</dictionary>
</item>
<item>
<dictionary>
<item key="id">name</item>
<item key="label">Name</item>
</dictionary>
</item>
<item>
<dictionary>
<item key="id">vorname</item>
<item key="label">Vorname</item>
</dictionary>
</item>
</list>
</item>
</dictionary>
</item>
</list>

```

Die Darstellung des Konfigurations-Codes in Form der Eingabe-Maske übernimmt die Produkt-Methode `\zms\dtml\zmssql\input_form.dtml`. Das Masken-Element wird in der Regel automatisch gewählt (z.B. Datum-Selektor für DATE-Felder). Bei Bedarf lässt sich mit dem `type`-Element ein Feldtyp explizit deklarieren, z.B. text-Feld bei `varchar(255)`.

```

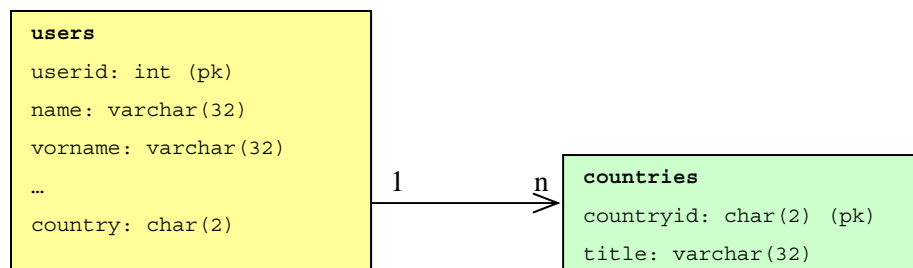
<item>
<dictionary>
<item key="id">comments</item>
<item key="label">Comments</item>
<item key="hide" type="int">1</item>
<item key="type">text</item>
</dictionary>
</item>

```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Einsatz von Primär- und Fremdschlüssel: Relationen abbilden

Im relationalen Datenbankmodell sind Tabellen über „Schlüssel“ verbunden; der Primärschlüssel ist eine innerhalb der Tabelle für jeden Datensatz eindeutige ID; diese eindeutige, *primäre* ID kann von einer anderen Tabelle referenziert werden. Diese Referenz zu einer ID in eine anderen Tabelle nennt sich „Fremdschlüssel“.



ER-Diagramm

Beispiel: In einer Datenbank für die Verwaltung von Benutzern soll die Zuordnung zu einem Land über einen zweistelligen Länder-Code (z.B. ‚DE‘ für Deutschland) abgebildet werden. Die Länder werden mit Code und Bezeichnung in einer eigenen Tabelle gespeichert und haben den Länder-Code als eindeutige ID (Primärschlüssel). Die Nutzerdaten-Tabelle referenziert nun diesen Primärschlüssel der Länder-tabelle mit einem diesem Schlüssel entsprechenden Wert. Für den Eintrag dieses referenzierten, also entfernten bzw. fremden Schlüssels in die Nutzerdaten-Tabelle ist in der Nutzerdaten-Tabelle eine entsprechende Fremdschlüssel-Spalte vorgesehen. Damit stehen beide Tabellen in einer „Relation“.

Diese Relationen lassen sich auch über die XML-Konfiguration des *ZMSSqldb*-Objektes abbilden: Dazu wird der zu referenzierende Tabellename (tablename) und diejenige Tabellenespalte bezeichnet, die als Fremdschlüssel (fieldname) referenziert wird. Das Beispiel zeigt ein Tabellen-Item (Spalte, columns-item), das als Select-Liste abgebildet wird und seine Werte aus einer referenzierten Länder-Tabelle erhält:

So sieht das entsprechende Redaktions-Interface aus:

Place	Berlin
Country	Deutschland
Tel	Tschechische Republik
Fax	Deutschland
	Djibouti

Konfigurations-Code für das Datenmodell der oben dargestellten Einfachauswahl-Liste „Länder“:

```

...

```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

```

<item>
  <dictionary>
    <item key="id">country</item>
    <item key="label">Country</item>
    <item key="fk">
      <dictionary>
        <item key="tablename">countries</item>
        <item key="fieldname">countryid</item>
        <item key="displayfield">title</item>
      </dictionary>
    </item>
  </dictionary>
</item>
...

```

Alternativ ist es auch möglich bei fehlender Bezugstabelle willkürliche Auswahlwerte vorzugeben. Über das option-Element lassen sich Werte explizit vorgeben, ohne dass diese in einer Datenbank-Tabelle stehen müssen:

```

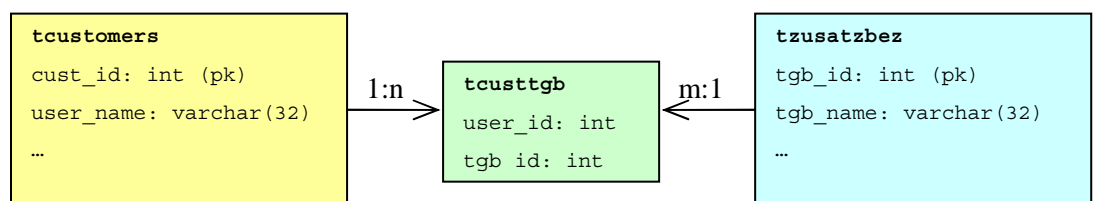
<item>
  <dictionary>
    <item key="id">lang</item>
    <item key="label">Language</item>
    <item key="hide" type="int">1</item>
    <item key="fk">
      <dictionary>
        <item key="options">
          <list>
            <item>
              <list>
                <item type="int">1</item>
                <item>English</item>
              </list>
            </item>
            <item>
              <list>
                <item type="int">2</item>
                <item>German</item>
              </list>
            </item>
          </list>
        </item>
      </dictionary>
    </item>
  </dictionary>
</item>

```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

Modellierung von n:m-Relationen: *intersection*-Tabelle

Am Beispiel einer Ärzte-Datenbank lässt sich ein weiteres wichtiges Prinzip von Relationen erläutern: ein Arzt kann mehrere fachliche Zusatzqualifikationen (sog. Zusatzbezeichnungen) besitzen. Die Zahl der möglichen Zusatzbezeichnungen ist begrenzt. Um dieses n:m-Verhältnis abzubilden, wird im Datenbankmodell üblicherweise eine Zwischentabelle (*intersection*) eingeführt. Dort kann der Datensatz eines Arztes in Form seines Fremdschlüssels mehrfach vorkommen (bzw. referenziert werden) und diese Datensätze referenzieren wiederum (auch per Fremdschlüssel) jeweils einen Eintrag in der Tabelle der Zusatzqualifikationen:



Beispiel eines ER-Diagramms für Ärzte mit Zusatzbezeichnungen

Das Interface zeigt die Konfiguration der Tabellenfelder:

- Kennung:** HoffmEdna
- Passwort:** [maskiert]
- Fachgebiete:**
 - vorhanden: Heilpraktiker, Zahnarzt, Facharzt für Biochemie, Facharzt für Orthopädie..., Facharzt für Physikalisc...
 - ausgewählt: Praktischer Arzt
- Zusatzbezeichnung:**
 - vorhanden: Allergologie, Balneologie und Medizinis..., Chirotherapie, Homöopathie, Naturheilverfahren
 - ausgewählt: [leer]

Das Interface präsentiert die Tabellen in einem integrierten Interface (Ausschnitt):

Der folgende Konfigurations-Code für das hier dargestellte Interface auf dem weiter oben abgebildeten Datenmodell zeigt ausschnittshaft definiert die relevanten Sektionen für die Abbildung einer n:m-Beziehung:

- das Mehrfachauswahl-Element „Zusatzbezeichnung“ wird in die Liste der Datenfelder der Tabelle *tcustomers* ergänzt; hier mit der (willkürlichen) ID *zusatzbez*; über ein Element *index* bekommt dieses zusätzliche Element eine Platzierung in der Reihenfolge aller Interface-Elemente der Tabelle zugewiesen (im Beispiel-Code die Position 4). Den Bezug zur *intersection*-Tabelle ge-

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

schieht durch ein per *multiselect* geschachteltes *fk*-Elements bzw. die dort im *tablename*-Element genannten intersection-Tabelle *tcusttgb*.

- Die intersection-Tabelle muss gesondert definiert werden; hier werden die beiden *fk*-Beziehungen wie üblich abgebildet. Der type *intersection* verhindert, dass die Tabelle im Interface in der Tabellenliste erscheint bzw. direkt bearbeitet werden kann.

```
<list>
<!-- #####
      ### TCUSTOMER
      ##### -->
<item>
  <dictionary>
    <item key="id">tcustomers</item>
    <item key="label">Kunden</item>
    <item key="columns">
      <list>
<!--... [1]...-->
      <item>
        <dictionary>
          <item key="hide" type="int">1</item>
          <item key="id">zusatzbez</item>
          <item key="index" type="int">4</item>
          <item key="label">Zusatzbezeichnung</item>
          <item key="multiselect">
            <dictionary>
              <item key="fk">cust_id</item>
              <item key="tablename">tcusttgb</item>
            </dictionary>
          </item>
        </dictionary>
      </item>
    </list>
  </item>
<!-- ... [1]...-->
  </list>
</item>
</dictionary>
</item>
<!-- ... [2]...-->
<!-- #####
      ### tcusttgb INSECTIION
      ##### -->
<item>
  <dictionary>
    <item key="id">tcusttgb</item>
    <item key="type">intersection</item>
    <item key="columns">
      <list>
        <item>
          <dictionary>
            <item key="id">user_id</item>
            <item key="label">Kunde</item>
            <item key="fk">
              <dictionary>
                <item key="tablename">tcustomers</item>
                <item key="fieldname">cust_id</item>
                <item key="displayfield">user_name</item>
              </dictionary>
            </item>
          </dictionary>
        </item>
      </list>
    </item>
    <item>
      <dictionary>
        <item key="id">tgb_id</item>
        <item key="label">Zusatzbezeichnung</item>
        <item key="fk">
          <dictionary>
            <item key="tablename">tzusatzbez</item>
            <item key="fieldname">tgb_id</item>
          </dictionary>
        </item>
      </dictionary>
    </item>
  </list>
</item>
</dictionary>
</item>
</list>
```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

```

        <item key="displayfield">tgb_name</item>
      </dictionary>
    </item>
  </dictionary>
</item>
</list>
</item>
</dictionary>
</item>
<!-- .. [2] ....-->
</list>

```

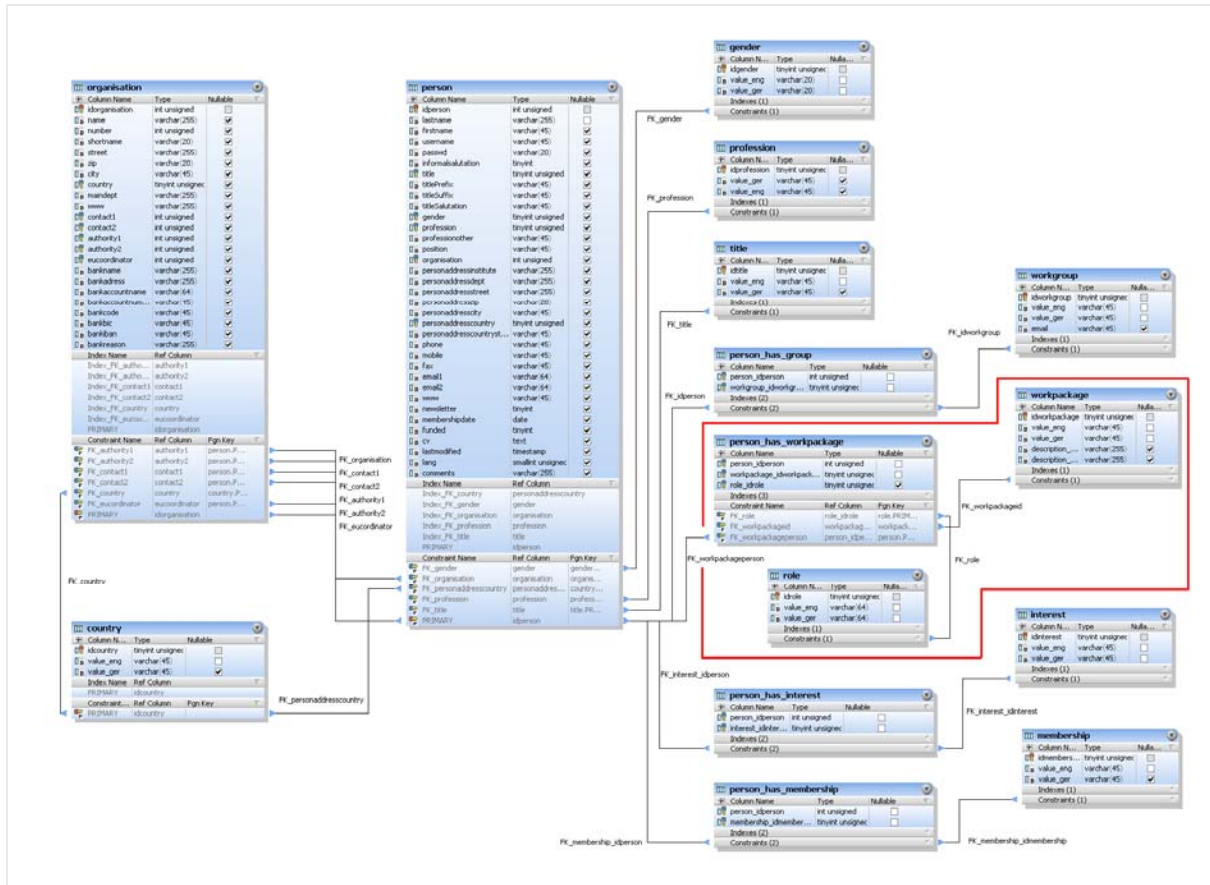
Multiselect: kombinierte Relation von n:m und 1:n

Inbesondere Userdatenbank aus dem wissenschaftlichen Umfeld besteht oftmals die Situation, die Mitarbeit einer Person an diversen Arbeitspaketen (*workpackages*) und dessen für das jeweilige Arbeitspaket spezifische Rolle abzubilden.

Im Folgenden wird ein reales Datenbank als Beispiel herangezogen; es handelt sich um eine Userdatenbank für ein internationales Forschungsnetz. Das Übersichtsbild (übernächste Grafik, ER-Modell) veranschaulicht dieses Beziehungsgefüge im rot markierten Bereich. Um ein möglichst einfaches und effizientes Interface zu erhalten, erzeugt das *ZMSSqldb*-Objekt auf Basis einer entsprechenden Modellierung per XML folgendes Eingabe-Interface:

Für jedes Arbeitspaket kann eine Rolle definiert und per Klick auf den [+]-Button in die obere Übersichtsliste transferiert werden. Auf diese Weise lassen sich einer Person viele Arbeitspakete samt Rolle, welche die Person im jeweiligen Arbeitspaket innehat, in Form entsprechender Wertepaare festlegen. Mit dem [-]-Button können einzelne Wertepaare wieder entfernt werden.

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing



Datebank-Modell für das Nutzer-Management eines Forschungsnetzes. Der rot umrandete Bereich markiert diejenigen Tabellen, welche die n:m-Beziehung einer ‚person‘ per intersection-Tabelle (person_has_workpackages) zu multiplen ‚workpackages‘ vermittelt. Jede dieser Beziehungen ist wiederum durch eine Relation zur ‚role‘-Tabelle erweitert, um die Rolle, welche die Person im Arbeitspaket inne hat, zu referenzieren.

```

<list>
<!-- ##### Table person ##### -->
<item>
  <dictionary>
    <item key="id">person</item>
    <item key="label">Person</item>
    <item key="columns">
      <list>
        <item>
          <dictionary>
            <item key="id">workpackages</item>
            <item key="index" type="int">36</item>
            <item key="hide" type="int">1</item>
            <item key="label">Workpackages</item>
            <item key="multiselect">
              <dictionary>
                <item key="fk">person_idperson</item>
                <item key="tablename">person_has_workpackage</item>
                <item key="tables">
                  <list>
                    <item>
                      <dictionary>
                        <item key="fk">workpackage_idworkpackage</item>

```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

```

        <item key="tablename">workpackage</item>
        <item key="fieldname">idworkpackage</item>
        <item key="displayfield">value_eng</item>
        <item key="label">Workpackage</item>
    </dictionary>
</item>
<item>
    <dictionary>
        <item key="fk">role_idrole</item>
        <item key="tablename">role</item>
        <item key="fieldname">idrole</item>
        <item key="displayfield">value_eng</item>
        <item key="label">Role</item>
    </dictionary>
</item>
</list>
</item>
</dictionary>
</item>
</dictionary>
</item>
...
</list>
</item>
</dictionary>
</item>
...
<!-- #####
    ### Table person_has_workpackage INSECTION
    ##### -->
<item>
    <dictionary>
        <item key="id">person_has_workpackage</item>
        <item key="type">intersection</item>
        <item key="columns">
            <list>
                <item>
                    <dictionary>
                        <item key="id">person_idperson</item>
                        <item key="label">Person</item>
                        <item key="fk">
                            <dictionary>
                                <item key="tablename">person</item>
                                <item key="fieldname">idperson</item>
                                <item key="displayfield">lastname</item>
                            </dictionary>
                        </item>
                    </dictionary>
                </item>
            </list>
        </item>
        <item>
            <dictionary>
                <item key="id">workpackage_idworkpackage</item>
                <item key="label">Workpackage</item>
                <item key="fk">
                    <dictionary>
                        <item key="tablename">workpackage</item>
                        <item key="fieldname">idworkpackage</item>
                        <item key="displayfield">value_eng</item>
                    </dictionary>
                </item>
            </dictionary>
        </item>
    </list>
</item>
</dictionary>
</item>
...
</list>

```

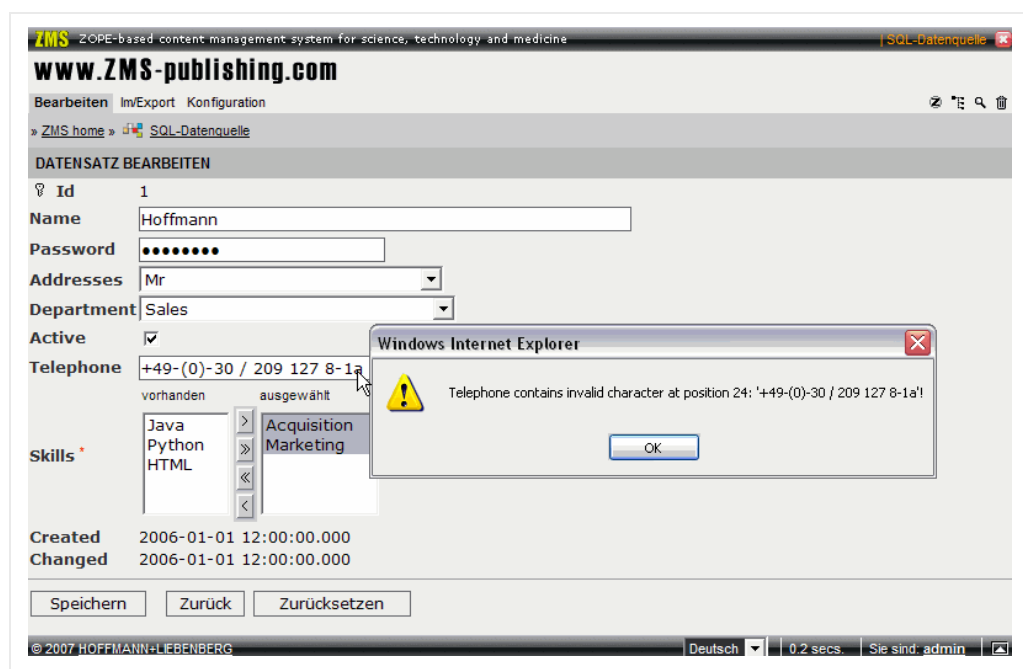
Ausschnitt aus der XML-Konfiguration des Datenmodells: für die Tabelle ‚person‘ schachtet ein multimultiselect-Element die Liste zweiter Tabellen ‚person_has_workpackage‘ und ‚role‘. Im Anschluss wird die intersection-Tabelle person_has_workpackage im Detail mo-

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

deklariert. Hierbei werden die beiden Datenspalten der Bezugstabellen, also die Fremdschlüssel (jk) adressiert und diejenige Datenspalten deklariert, die im Userinterface angezeigt werden soll (displayfield)

Clientseitige Validierung der Dateneingabe

Mit ZMS 2.10.1 wurde das Interface-Element für die Konfiguration des Datenmodells eingeführt; per JavaScript (JS) oder AJAX lässt client- oder serverseitig eine Daten-Validierung durchführen. Es folgt ein JS-Beispiel für die Validierung einer Eingabe als Telefonnummer; dieser Code ist Teil einer Konfigurationsdatei, die sich im ZMS-Product-Folder ../zms/import/zmssqldb/ befindet.



Client-seitige Eingabe-Validierung per JavaScript: mithilfe des interface-Elements in der Datenquellen-Konfiguration lässt sich JS-Code in das automatische Interface einschleusen. Damit kann das Absenden einer fehlerhaften Eingabe (z.B. Buchstabe innerhalb einer Telefonnummer) abgefangen werden.

```

...
<item key="interface">
<![CDATA[<script language="JavaScript">
<!--
/**
 * OnSubmit-Event: form0
 */
function form0Submit( fm)
{
    var b = true;
    var emp_phone = fm.elements['new_EMP_PHONE'].value;
    for ( var i = 0; i < emp_phone.length && b; i++) {
        var ch = emp_phone.charAt( i)
        if (! ( ch == '1' || ch == '2' || ch == '3' || ch == '4' || ch == '5' || ch == '6' || ch == '7' || ch

```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

```

== '8' ||
    ch == '9' || ch == '0' || ch == ' ' || ch == '-' || ch == '(' || ch == ')' || ch == '/' || ch
== '+' || ch == '-') {
    alert( "Telephone contains invalid character at position " + i + ": '" + emp_phone.substr( 0, i+1) +
"!");
    b = false;
    }
}
return b;
}

var fm_form0 = document.forms['form0'];
fm_form0.onsubmit = function() { return form0Submit(this) };

-->
</script>]]>
</item>
...

```

Code-Beispiel für das Einführen von JavaScript mit Hilfe des interface-Elements in der Datenmodell-Konfiguration.

Serverseitige Validierung während der Dateneingabe

Analog dem obigen Beispiel kann das Interface-Element AJAX-Code in das Formular einführen; das Absetzen von AJAX-Requests sieht im Prinzip wie folgt aus: im Code-Beispiel werden über die serverseitige Methode *ajaxValidatePlzOrt* die neuen Anwender-Eingaben von Ort ('new_ORT*') und PLZ ('new_PLZ') *onchange* verrechnet und ein Feedback in das Eingabe-Formular zurückgegeben.

```

var fm_form0 = document.forms['form0'];
var el_plz = fm_form0.elements['new_PLZ'];
var el_ort = fm_form0.elements['new_ORT'];

function el_plz_ort_Change(el)
{
    document.getElementsByTagName('body')[0].style.cursor = "wait";
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    var url = "<dtml-var URL>/ajaxValidatePlzOrt?plz="+escape(el_plz.value)+"&ort="+escape(el_ort.value);
    req.open("GET",url,true);
    req.onreadystatechange = processResponseValidatePlzOrt;
    req.send(null);
}

function processResponseValidatePlzOrt()
{
    if (req.readyState == 4 && req.status == 200)
    {
        document.getElementsByTagName('body')[0].style.cursor = "auto";
        [...]
    }
}

// Event-Handler dynamisch anpassen
el_plz.onchange = function() { return el_plz_ort_Change(this) };
el_ort.onchange = function() { return el_plz_ort_Change(this) };

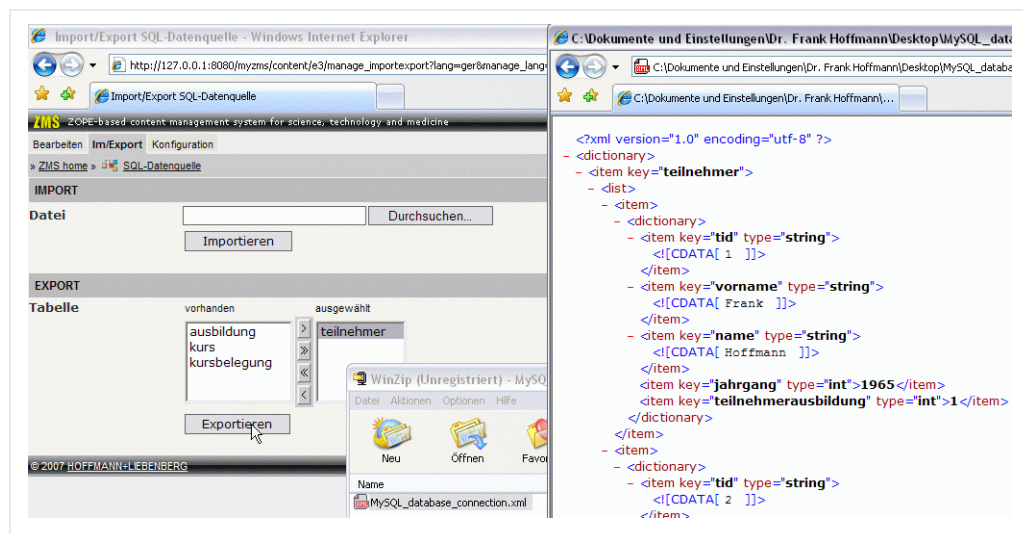
```

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing

* *Hinweis:* das von ZMSQldb automatische Darstellungsformular setzt dem Namen eines jeden Datenfeldes den prefix „new_“ voran; die Bestandsdaten sind allesamt in Hidden- Feldern gedoppelt (prefix für den Feldnamen: „old_“).

Daten-Export

Auf Grundlage des konfigurierten Datenmodells ist es möglich, die entsprechenden Datentabellen einzeln oder im Gesamten als gezippten XML-Strom zu exportieren und damit der externen Weiterverarbeitung zuzuführen.



Export einer Datentabelle aus einem Set von vier Tabellen, die im Konfigurationsmenü des SQLdb-Objektes modelliert worden sind.

Literatur

Punkt für Punkt Anleitung zur Installation des Zope-Servers mit Anbindung des

MySQL-Servers auf SuSE Linux 8.1 Systemen

<http://alt.dzug.org/Members/wheider/howto/inst-Zope-MySQL-SuSE>

Connecting Zope to External Relational Databases

Gadfly—Zope's Integrated Demo Relational Database

<http://www.sampublishing.com/articles/article.asp?p=24698&seqNum=8&rl=1>

Erstellung	J. Klein, F. Hoffmann	Gültigkeitsbeginn	04.06.2007	ZMS Professional Publishing
Prüfung	F. Hoffmann, J. Schönfeldt	Version	0.73	Datei: ZMSProfessionalPublishing.xml
Freigabe	F.Hoffmann	Letzte Bearbeitung	05.06.07 19:15	© 2007 SNTL Publishing