

Tutorial: Objektorientierte Erweiterung der Datentypen

Version: 1
Autor: Daniel Neumann

Inhaltsverzeichnis

1	Objektorientierte Erweiterung der Datentypen.....	2
1.1	Anlegen eines abgeleiteten Schemas.....	2
1.2	Erzeugen eines neuen Objekts.....	3
1.3	Anpassen des Editors.....	4
1.4	Anpassen der Transformation.....	5

Voraussetzungen

Für dieses Tutorial werden keine weiteren Voraussetzungen als für die vorhergehenden Tutorials benötigt.

- > [Informationsmodell festlegen](#)
- > [Transformationen entwickeln](#)
- > [Editor individualisieren](#)
- > [Strukturelle Integrität und Kindschemata](#)
- > [Module erstellen und verteilen](#)
- > [Erweiterung des Datenmodells und Datenkonsistenz](#)
- > [Einschränkung simpler Datentypen](#)

Beschreibung

In diesem Tutorial wird gezeigt, wie sich Ableitungen von Schemata erstellen und somit die Vorteile objektorientierter Modellierung nutzen lassen.

Zeichenerklärung



Text, der grün umrandet und mit dem Pfeil-Symbol gekennzeichnet ist, enthält konkrete Anweisungen, was als nächstes zu tun ist.



Texte in solchen Kästchen enthalten Tipps und Tricks.

Quellcode wird in solch blauen Boxen dargestellt.

1 Objektorientierte Erweiterung der Datentypen

Wir beginnen damit, den Datentyp für das Zitat zu verändern, dass der ComplexType nicht inline, sondern als benannter Datentyp vorliegt.



Wechseln Sie in das Schema „quotation“. Erweitern Sie das Element mit dem Namen „quotation“ um das Attribut „type='quotation'“ und machen Sie es zu einem self-closing Element. Löschen Sie das nun überflüssige schließende Tag „</xs:element>“. Fügen Sie dann dem complexType-Element ein Attribut „name“ mit dem Wert „quotation“ hinzu.

```
<xs:element name="quotation" type="quotation" />
<xs:complexType name="quotation">
  <xs:sequence>
    <xs:element name="author" type="xs:string" />
    <xs:element name="quote" type="xhtml:Flow" />
    <xs:element
      name="image"
      type="xlink:binaryReference"
      minOccurs="0"
    />
  </xs:sequence>
</xs:complexType>
```

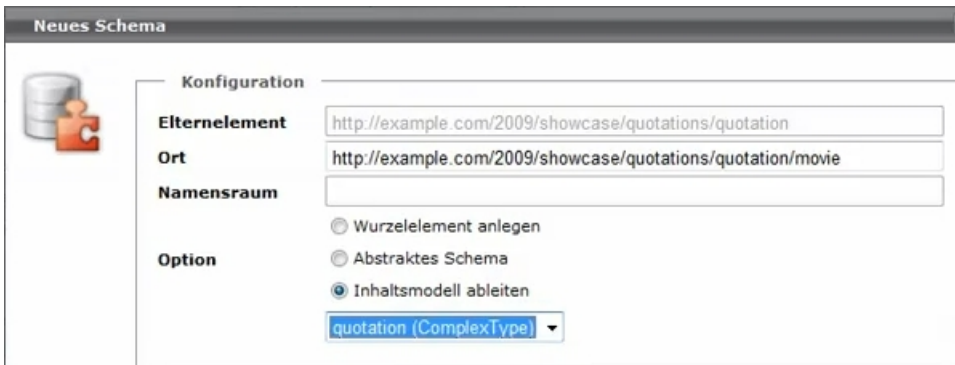


Speichern Sie das Schema.

1.1 Anlegen eines abgeleiteten Schemas

Jetzt legen wir ein neues Schema an, in welchem der ComplexType „quotation“ abgeleitet wird.

Erzeugen Sie ein neues Schema „movie“ unterhalb von „quotations/quotation“. Als Option wählen Sie nun „Inhaltsmodell ableiten“. Aus der Dropdown-Box wählen sie den ComplexType „quotation“ aus.



Das abgeleitete Schema erhält ein zusätzliches Attribut.



Fügen Sie innerhalb des extension-Elements folgendes Attribut hinzu und speichern Sie danach das Schema:

```
<xs:extension base="quotation">
  <xs:attribute
    name="movie"
    type="xs:string"
    use="required"
  />
</xs:extension>
```



Wechseln Sie in die Detailansicht und legen Sie unterhalb von „Zitate“ ein neues Objekt vom Typ „movie“ an.

Da der neue Datentyp vom Schema „quotation“ abgeleitet ist, darf er im Ordner „Zitate“ angelegt werden. Wie Sie sehen, wird nun im Editor ein weiteres Eingabefeld „movie“ angezeigt.

1.2 Erzeugen eines neuen Objekts



Legen Sie nun folgendes Zitat an und geben Sie es zurück:

Titel: Filmzitat Terminator

movie: Terminator

Autor: Terminator (Arnold Schwarzenegger)

Zitat:I'll be back.



Klicken Sie nun auf das Objekt „Zitate“.

Wie Sie sehen, wird unser neu angelegtes Objekt nur in der Baumansicht angezeigt, nicht jedoch in der Listenansicht.

1.3 Anpassen des Editors

Jetzt passen wir den Editor für den neuen Datentyp an.



Dazu wechseln Sie in das Editor-Modul „Zitatverwaltung“ und bearbeiten das Objekt „Icons“. An der Stelle, an der die Icons für das Schema „quotation“ definiert sind, aktivieren Sie die Checkbox „Ableitungen einbeziehen“.

Über das Häkchen erhält der abgeleitete Datentyp das gleiche Icon wie das Zitat.

Nach einem Refresh des Editors (mit F5) sehen Sie, dass das neue Zitat „Filmzitat Terminator“ in der Baumansicht das selbe Icon erhalten hat wie die Objekte vom Typ „quotation“.



Geben Sie nun das Objekt „Icons“ zurück und bearbeiten das Objektstrukturfenster.

Damit wir die Filmzitate auch in der Listenansicht des Editors sehen, können wir das Attribut „handleDerivations“ setzen.



Das Attribut fügen Sie im childType-Element hinzu:

```
<childType
  schemaLocation="http://example.com/2009/showcase/quotations/quotation"
  structureInvisible="false"
  handleDerivations="true"
/>
```

Wenn Sie nun Ihren Zitat-Container „Zitate“ anklicken, sehen Sie in der Listenansicht auch das neue Zitat.

Jetzt übersetzen wir noch die verbleibenden Elemente. Dies erledigen wir auch im Editor-Modul „Zitatverwaltung“ im Objekt „Ressourcen/de“.



Fügen Sie im type-Element ein neues item-Element hinzu. Als name-Attribut nutzen Sie „http://example.com/2009/showcase/quotations/quotation/movie“. Als Text soll „Filmzitat“ ausgegeben werden.

```
<item  
name="http://example.com/2009/showcase/quotations/quotation/movie">Filmzitat</item>
```

Zusätzlich soll das neue Attribut „movie“ eine Übersetzung bekommen.



Fügen Sie im namespace-Element „Editor“ ein neues type-Element für das Schema „quotations/quotation/movie“ hinzu. Im neuen item-Element nehmen Sie als name-Attribut „Attribute:movie#Label“. Als Text nehmen Sie „Film“:

```
<type name="http://example.com/2009/showcase/quotations/quotation/movie">  
  <item name="Attribute:movie#Label">Film</item>  
</type>
```

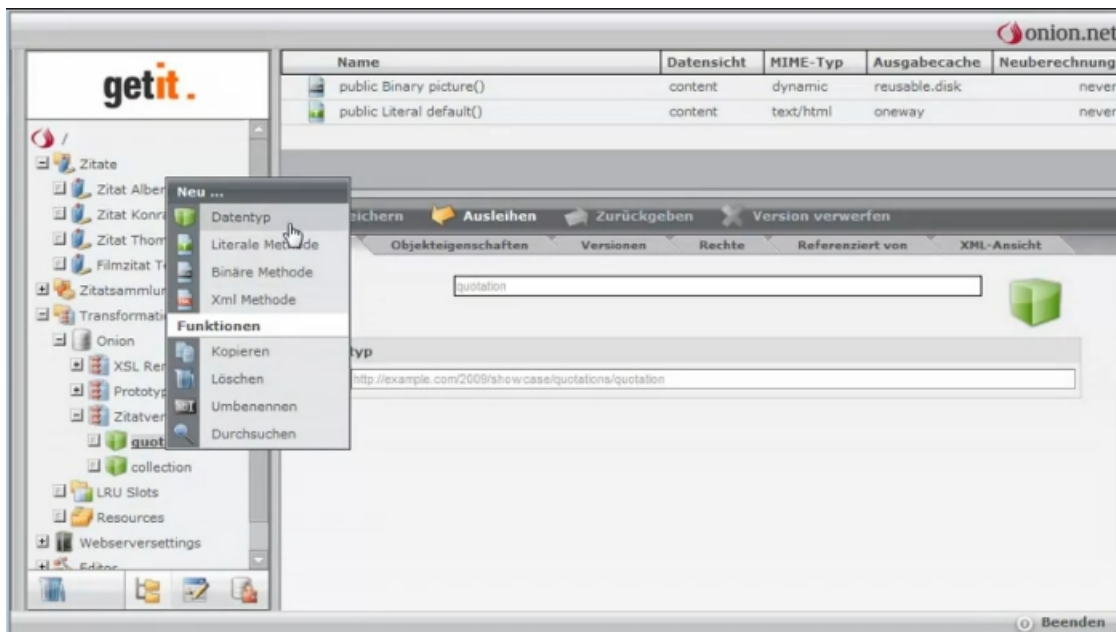
Geben Sie das Objekt zurück und wechseln in Ihr Filmzitat. Sie sehen nun die Ausgabe „Film“ für das Attribut „movie“.

1.4 Anpassen der Transformation

Jetzt müssen wir in der Transformation unterscheiden, ob es sich um ein normales oder ein Filmzitat handelt. Bislang wird der Film natürlich noch nicht berücksichtigt.



Wechseln Sie in Ihre Transformationsgruppe „Zitatverwaltung“ und dort in den Datentyp „quotation“. Legen Sie darunter einen neuen Datentyp „movie“ an. In das Feld „name“ ziehen Sie das Schema „quotations/quotation/movie“.



In der Transformation für den Datentyp „movie“ legen wir in der literalen Methode „default“ zunächst die Ausgabe für den Film fest.



Legen Sie dazu eine neue literale Methode „default“ an. Wählen Sie als Zugriffsschutz „öffentlich“ und als Datensicht „content“. Als Inhalt der Methode schreiben Sie „<xml:stylesheet“ und nutzen Sie die Codeervollständigung, um den Standard-Inhalt anzulegen, ohne einen weiteren Namensraum auszuwählen. Tragen Sie nur im output-Element in das Attribut method den Wert „xml“ ein. Erweitern Sie das Template-Element wie folgt:

```
<xsl:template match="/quotation"> Zitat aus dem Film:
  <xsl:value-of select="@movie" />
  <br/>
</xsl:template>
```

Schauen Sie sich nun eine Vorschau des Objekts „Filmzitat Terminator“ an. Sie sehen nun die Ausgabe, die wir eben in die literale Methode geschrieben haben. Jetzt erweitern wir die „default“-Methode, dass die restlichen Daten eines Zitats von der übergeordneten Methode transformiert werden.

Dazu muss die übergeordnete literale Methode importiert werden. Dies geschieht über das import-Element direkt an erster Stelle im Stylesheet:

```
<xsl:import href="../default" />
```

Damit die importierte Transformation auch tatsächlich angewendet wird, muss diese noch konkret angewendet werden. Dazu fügen Sie innerhalb des template-Elements ein apply-imports-Element ein:

```
<xsl:template match="/quotation"> Zitat aus dem Film:  
  <xsl:value-of select="@movie" />  
  <br/>  
  <xsl:apply-imports/>  
</xsl:template>
```

Öffnen Sie nun erneut die Vorschau des Filmzitats, werden Sie nun auch die restlichen Ausgaben der übergeordneten Transformation sehen. Zum Vergleich öffnen Sie die Vorschau eines anderen Zitats vom Typ „quotation“. Hier ist die Ausgabe wie gehabt ohne den Zusatz „Zitat aus dem Film.“

Objektorientierung macht das Komplexe einfach; vorhandene Logik nutzen statt sie zu kopieren, Modelle ableiten statt sie neu zu erfinden. So wird der Pflegeaufwand minimiert.