Kapitel 5
Navigation und Verknüpfung

Wie sind Teile eines XML-Dokuments für Anwendungen ansprechbar? Wie lassen sich Verknüpfungen herstellen? Spezielle Adressierungssprachen wie XPath und XLink liefern Antworten auf diese Fragen.

Die logische Struktur eines XML-Dokuments lässt sich, wie schon mehrfach angesprochen, in einer hierarchischen Baumstruktur darstellen, in der die unterschiedlichen Informationseinheiten als Knoten erscheinen.

Um Teile eines XML-Dokuments ansprechen und auswerten zu können, standardisierte das W3C schon sehr früh als Ergänzung zu XML eine spezielle Adressierungssprache unter dem Namen *XML Path Language – XPath*. Auf der Basis von XPath wurde mit der *XML Pointer Language – XPointer* – versucht, eine Möglichkeit zu schaffen, auch auf Fragmente in externen Dokumenten zuzugreifen.

Die *XML Linking Language – XLink* – definiert Möglichkeiten, Elemente in XML-Dokumenten zur Verknüpfung mit anderen Ressourcen zu verwenden.

5.1 Datenauswahl mit XPath

XPath ist selbst keine XML-Anwendung. Die Sprache enthält eine eigene Syntax zur Bildung von Zeichenketten, die sich zur Adressierung von Untermengen eines Dokuments verwenden lassen. Die Empfehlung finden Sie unter *http://www.w3.org/TR/xpath*.

Eine Teilmenge von XPath wird, wie schon beschrieben, in XML Schema für die Bildung von Werten für spezielle Auswahlattribute in den Elementen `<xsd:key>`, `<xsd:unique>` und `<xsd:keyref>` benutzt. XPath spielt vor allem aber in XSLT- oder XSL-Dokumenten eine wichtige Rolle – Thema von Kapitel 7 bzw. Kapitel 8 –, die ja wohlgeformte XML-Dokumente sind. Die XPath-Syntax wird dort zur Formulierung der Werte für die Attribute `match` und `select` benutzt, die in den Stylesheets zur Auswahl von Knotenmengen verwendet werden.

5.1.1 Baummodell und XPath-Ausdrücke

XPath arbeitet auf der Basis eines Baummodells, das die im XML-Dokument enthaltenen Informationseinheiten repräsentiert, die in der *Infoset*-Empfehlung definiert sind. Dieses Modell ist dem DOM-Modell – das in Kapitel 10, »Programmierschnittstellen für XML«, beschrieben wird – sehr ähnlich, aber doch nicht damit identisch. Das DOM-Modell ist ein Objektmodell mit Knoten, deren Eigenschaften und Methoden einem Anwendungsprogramm zur Verfügung stehen, während die Knoten im XPath-Modell verhaltenslos sind. Das Knotenmodell ist also eigentlich nur eine bestimmte Sicht auf ein Dokument.

Eine Instanz der XPath-Sprache wird *Ausdruck* genannt. Ein XPath-Ausdruck wird ausgewertet, um ein Objekt zu gewinnen, für das vier grundlegende Datentypen möglich sind.

- Der Datentyp `node-set` liefert eine ungeordnete Knotenmenge, die auch leer sein kann. Die Knotenmenge enthält keine Duplikate.
- `string` ist eine Zeichenfolge, die auch leer sein kann, wobei die Zeichen der XML-Empfehlung entsprechen.
- `number` ist immer eine Fließkommazahl, genauer eine Zahl im 64-Bit-Format – *Double-Precision* – nach IEEE 754.
- Der Datentyp `boolean` kann die Werte `true` und `false` annehmen.

Dabei sind zwischen den verschiedenen Datentypen Umwandlungen möglich, für die in der XPath-Empfehlung feste Regeln definiert sind.

XPath übernimmt nicht alle der in der Infoset-Empfehlung definierten Informationseinheiten, die Teile eines XML-Dokuments sein können. Entitäten sind in XPath beispielsweise nicht ansprechbar.

Das hängt damit zusammen, dass sich XPath-Ausdrücke immer auf das bereits von dem entsprechenden Prozessor geparste Dokument beziehen, in dem bereits alle Entitäten aufgelöst und durch die vollständigen Zeichenfolgen ersetzt sind. Auch CDATA-Abschnitte sind bereits in Text konvertiert, bevor ein XPath-Ausdruck angewendet wird.

5.1.2 Vom Dokument zum Knotenbaum

Zunächst ein Beispiel, wie ein XPath-Prozessor ein XML-Dokument in eine Baumstruktur übersetzt. Das Dokument gibt einen Auszug aus einem Weindepot:

```
<?xml version="1.0" encoding="UTF-8"?>
<weindepot>
  <anbaugebiet name="Mosel">
    <jahrgang jahr="2015">
```

```
      <art farbe="weiss">
        <wein>
          <bezeichnung>Enkircher Pastor</bezeichnung>
          <rebsorte>Riesling</rebsorte>
          <richtung qualitaet="Auslese">halbtrocken</richtung>
          <preis>8,00</preis>
        </wein>
        <wein>
          <bezeichnung>Cochemer Hospiz</bezeichnung>
          <rebsorte>Riesling</rebsorte>
          <richtung qualitaet="Auslese">trocken</richtung>
          <preis>18,00</preis>
        </wein>
      </art>
    </jahrgang>
  </anbaugebiet>
</weindepot>
```

Listing 5.1 weindepot.xml

Der XPath-Knotenbaum für dieses Dokument sieht vereinfacht so aus, wie in Abbildung 5.1 dargestellt.

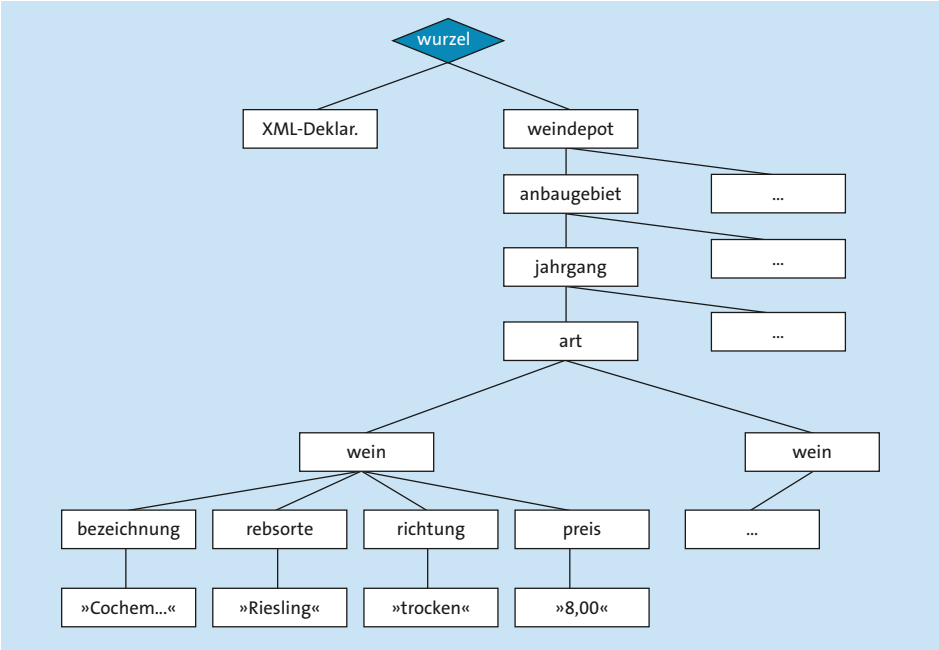


Abbildung 5.1 Baumstruktur des Beispieldokuments

Der Baum geht von einem Wurzelknoten aus, der nicht dem Dokumentelement entspricht, sondern ein logisches Konstrukt ist, von dem dann erst das Dokumentelement und eventuelle Knoten für Kommentare und Verarbeitungsanweisungen abzweigen.

5.1.3 Dokumentreihenfolge

Die Darstellung des XML-Dokuments als Baum wäre nicht viel mehr als eine Spielerei, wenn es für die Zuordnung der Knoten des Baumes zu den Komponenten des XML-Dokuments nicht eine strenge Regelung gäbe. Abbildung 5.2 zeigt eine abstrakte Struktur für einen in dieser Weise geordneten Baum.

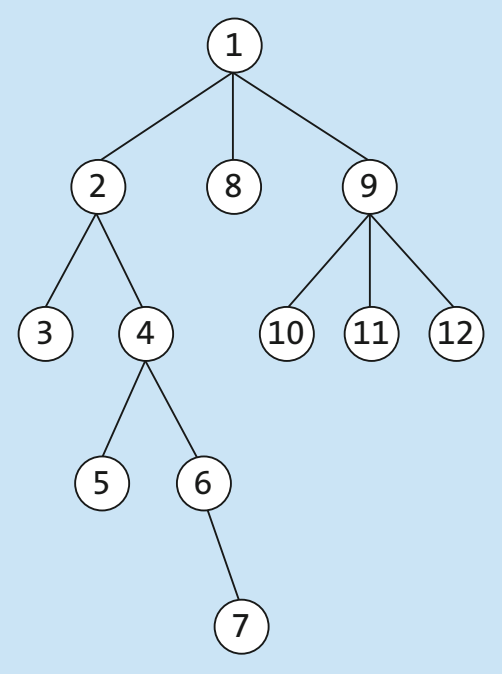


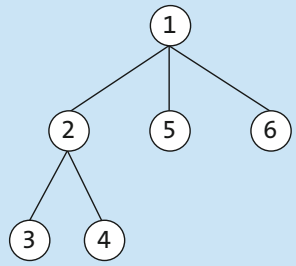
Abbildung 5.2 Die Zahlen geben die Reihenfolge der Knoten an, die der Dokumentreihenfolge entspricht.

Die Zahlen in den Knoten zeigen an, wie ein Prozessor die Knoten durchlaufen – traversieren – wird, wenn er den Auftrag erhält, sämtliche Knoten nacheinander zu besuchen.

Diese Anordnung wird *Dokumentreihenfolge* oder *Document Order* genannt. Sie entspricht exakt der Reihenfolge, in der die den Knoten entsprechenden Teile innerhalb des Dokuments erscheinen, wie das Beispiel in Abbildung 5.3 verdeutlicht.

Wie die Abbildung ebenfalls zeigt, findet dabei immer eine Tiefensuche statt. Es werden also nicht zunächst alle Knoten derselben Ebene nacheinander abgelaufen, sondern immer erst die Kinder und Kindeskinder des aktuellen Knotens abgearbeitet.

Wenn aus einem so geordneten Baum wieder ein sequenzielles Dokument erzeugt werden soll, muss er zunächst geplättet werden, und die Knoten müssen in der entsprechenden Reihenfolge in dem Dokument abgelegt werden. Dies wird *Serialisierung* genannt, während die Erzeugung eines Baumes aus einem Dokument umgekehrt als *Deserialisierung* bezeichnet wird.




```
1   <buch>
2      <autoren>
3         <name>Hanna Ergo</name>
4         <name>Peter Jahn</name>
       </autoren>
5      <titel>Out of Bonn</titel>
6      <preis>100</preis>
    </buch>
```

Abbildung 5.3 Die Dokumentreihenfolge entspricht der Reihenfolge der Start-Tags im Dokument.

5.1.4 Knotentypen

XPath unterscheidet unterschiedliche Knotentypen, deren Bedeutung in Tabelle 5.1 kurz erläutert wird. Zusätzlich wird jeweils der String-Wert angegeben. Damit ist die Zeichenkette gemeint, die die Auswertung eines XPath-Ausdrucks liefert, der den entsprechenden Knoten adressiert.

XPath-Knotentyp	Bedeutung
Wurzelknoten	Oberster Knoten des Baumes, dessen Kind der Elementknoten des Dokumentelements ist. Weitere Kinder können Processing-Instruction-Knoten und Kommentarknoten sein. String-Wert: Verkettung der Zeichendaten aller Textknoten-Kinder in der Dokumentreihenfolge
Elementknoten	Knoten für ein Element String-Wert: Verkettung der Zeichendaten aller Textknoten-Kinder des Elements

Tabelle 5.1 Knotentypen in XPath

XPath-Knotentyp	Bedeutung
Attributknoten	Knoten für jedes einem Element zugeordnete Attribut (Im Unterschied zu DOM wird in XPath dabei das betreffende Element als Elternelement betrachtet, obwohl die Attribute nicht als Kinder des Elements behandelt werden.) String-Wert: der normalisierte Attributwert
Textknoten	Knoten, der Zeichendaten enthält (Zeichen in Kommentaren, Verarbeitungsanweisungen und Attributwerte bilden keine Textknoten.) String-Wert: die Zeichendaten des Textknotens
Verarbeitungs-anweisungsknoten	Knoten für je eine Verarbeitungsanweisung String-Wert: der Inhalt der Instruktion, die dem Ziel der Verarbeitungsanweisung folgt (ohne den Begrenzer ?>)
Namensraumknoten	Der Namensraum ist jeweils einem Elementknoten als Elternknoten zugeordnet; er ist aber nicht Kind dieses Elementknotens. String-Wert: URI des Namensraums
Kommentarknoten	Knoten für jeden einzelnen Kommentar String-Wert: der Kommentarinhalt, der sich innerhalb der Kommentarbegrenzer befindet

Tabelle 5.1 Knotentypen in XPath (Forts.)

5.1.5 Lokalisierungspfade

Der wohl wichtigste Ausdruckstyp in XPath ist der Lokalisierungspfad. Nicht unähnlich den Techniken, die für die Pfadangaben von Dateiordnern verwendet werden, geben diese Ausdrücke an, welcher Teil der Knotenmenge eines Dokuments ausgewählt und bearbeitet werden soll. Dabei werden zwei Schreibweisen verwendet, eine ausführliche und eine Kurzform, die für viele Standardaufgaben meist am bequemsten ist.

Zugriff auf Elemente

Um in dem Weindepot-Beispiel den Knoten `<anbaugebiet>` zu erreichen, kann der folgende Ausdruck in der Kurzform verwendet werden:

```
/weindepot/anbaugebiet
```

Dieser Ausdruck liefert die entsprechende Knotenmenge, das *Node-Set*, eine ungeordnete Sammlung von Knoten. Wie bei Unix-Systemen werden die einzelnen Schritte mit einem Schrägstrich getrennt. Der erste Schrägstrich steht dabei für den Wurzelknoten.

Kontextknoten und relative oder absolute Pfade

Ähnlich wie bei Dateisystemen können absolute und relative Pfade verwendet werden. Ein absoluter Pfad geht immer vom Wurzelknoten aus, ein relativer dagegen von dem Knoten, der gerade der Kontextknoten ist. Das ist jeweils der Knoten, der durch die vorausgehenden Schritte erreicht worden ist.

```
anbaugebiet/jahrgang
```

wäre ein möglicher relativer Pfadausdruck, wenn `<weindepot>` der Kontextknoten ist.

```
/weindepot/anbaugebiet/jahrgang
```

ist der entsprechende Ausdruck mit einem absoluten Pfad. Um nicht immer die vollständige Hierarchie angeben zu müssen, kann mit dem Doppelschrägstrich gearbeitet werden.

```
//jahrgang
```

liefert die Knotenmenge der Elemente mit dem Namen `jahrgang`, falls diese Elemente auf irgendeiner Stufe zu den Nachkommen des Wurzelknotens gehören. Diese Abkürzung kann auch innerhalb eines Ausdrucks verwendet werden:

```
/weindepot//wein
```

Es muss dann nicht bekannt sein, wie viele Stufen zwischen den beiden Elementen liegen. Allerdings hat ein Prozessor mit einem so unbestimmten Ausdruck mehr Arbeit als mit den ersten beiden Ausdrücken, da er mehr Knoten des Dokuments prüfen muss.

Absolute Pfade haben den Vorteil, dass sie unabhängig vom gerade erreichten Kontextknoten verwendet werden können. Andererseits erschweren sie die Mehrfachverwendung von XPath-Ausdrücken.

Der Kontextknoten selbst kann in der Kurzform mit dem Punkt angesprochen werden. Die Eltern des Kontextknotens lassen sich mit `..` finden.

Anstelle konkreter Namen können auch Wildcards verwendet werden.

```
//anbaugebiet/*
```

liefert zum Beispiel alle Kinder des Elements `anbaugebiet`.

Zugriff auf Attribute

Obwohl XPath die Elementknoten als Eltern ihrer Attributknoten behandelt, werden die Attributknoten nicht als Kindknoten des Elementknotens angesprochen, wie dies für die Textknoten der Fall ist. Die Attributknoten müssen also in anderer Weise adressiert werden.

Um den Wert eines Attributknotens zu erhalten, kann in der Kurzform eine Kombination aus dem At-Zeichen `@` und dem Attributnamen verwendet werden.

```
anbaugebiet/@name
```

liefert zum Beispiel die Namensattribute der Elementknoten `<anbaugebiet>`.

Mit einem Ausdruck wie

```
anbaugebiet[@name="Mosel"]
```

dagegen lassen sich die Elementknoten auswählen, bei denen das Attribut `name` den angegebenen Wert hat.

Auch Attribute lassen sich über Wildcards suchen.

```
anbaugebiet/@*
```

liefert alle Attribute des Elements `<anbaugebiet>`.

5.1.6 Ausführliche Schreibweise

In der ausführlichen Schreibweise wird noch deutlicher, wie ein Lokalisierungspfad aus einzelnen Lokalisierungsstufen zusammengesetzt wird, die jeweils von links nach rechts ausgewertet werden.

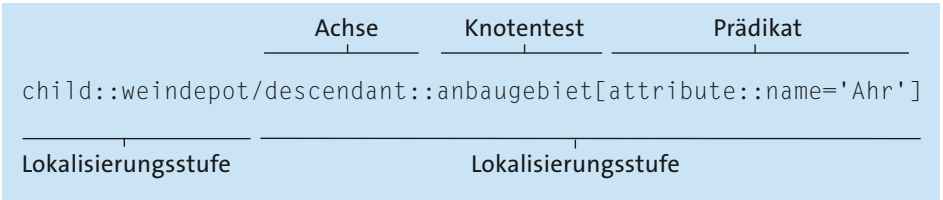


Abbildung 5.4 Die Bestandteile eines Lokalisierungspfads

5.1.7 Lokalisierungsstufen und Achsen

Jede Lokalisierungsstufe besteht aus drei Teilen: dem Achsenbezeichner, dem Knotentest und optional einem oder mehreren Prädikaten. Die allgemeine Syntax ist:

```
Achsenbezeichner::Knotentest[Prädikat1][Prädikat2]
```

Die doppelten Doppelpunkte werden als Trennzeichen zwischen Achsenbezeichner und Knotentest verwendet. Konkret sieht dies etwa so aus:

```
/child::weindepot
/child::anbaugebiet[attribute::name="Mosel"]
/child::jahrgang
```

Der Achsenbezeichner `child` gibt die Richtung an, in die der Knotendurchlauf erfolgen soll. In diesem Fall wird einfach vom Wurzelknoten zum Kind des Wurzelknotens, dem Wurzelelement `<weindepot>`, gesprungen.

Dieser erste Schritt liefert die Knotenmenge des Wurzelelements. Von dort geht es weiter zu dem Kind `<anbaugebiet>`, wobei durch das Prädikat ein bestimmtes Anbaugebiet ausgewählt wird. Das Prädikat bringt also die Attribute eines Elements ins Spiel. Wie zu sehen ist, kann der Lokalisierungspfad durch Leerräume gegliedert werden. Anstelle des beschriebenen Ausdrucks könnte auch die folgende Kurzform verwendet werden:

```
/weindepot/anbaugebiet[@name="Mosel"]/jahrgang
```

Mit dem |-Operator lassen sich auch gleich mehrere Pfade kombinieren:

```
child::anbaugebiet[attribute::name="Mosel"]|
child::anbaugebiet[attribute::name="Ahr"]
```

Der Knotentest kann entweder den Knotentyp oder direkt den erweiterten Namen des Knotens angeben. Wird ein qualifizierter Name, also ein Name mit einem Namensraumpräfix, verwendet, wird das Präfix durch den vollständigen Namensraumnamen ersetzt, weshalb von einem *erweiterten Namen* gesprochen wird. Ein positiver Effekt dieser Ersetzung ist, dass die Verwendung unterschiedlicher Präfixe für denselben Namensraum nicht zu Problemen führt.

Mit Hilfe der Achsenbezeichner kann gesteuert werden, ob von den Eltern aus die Kinder besucht werden, von den Kindern aus die Eltern oder die Geschwister. Mit jedem Schritt innerhalb des Knotengewirrs ändert sich dabei der Kontextknoten für den nächsten Schritt. Jede Achse definiert eine Knotenmenge relativ zum Kontextknoten. Tabelle 5.2 gibt einen Überblick über die möglichen Achsen.

Achse	Abgekürzt	Beschreibung
`self`	.	Liefert den Kontextknoten.
`child`	(Vorgabe)	Liefert die Kinder des Kontextknotens.

Tabelle 5.2 Achsen in XPath

Achse	Abgekürzt	Beschreibung
`parent`	..	Liefert die Eltern des Kontextknotens, falls vorhanden.
`descendant`	//	Liefert die Nachkommen des Kontextknotens (Kinder, Kindeskinder etc., also ausschließlich Elementknoten).
`descendant-or-self`		Liefert den Kontextknoten und die `descendant`-Achse.
`ancestor`		Liefert die Vorfahren des Kontextknotens bis hin zum Wurzelknoten.
`ancestor-or-self`		Liefert den Kontextknoten und die `ancestor`-Achse.
`following`		Liefert alle Elementknoten des Dokuments, die in der Dokumentreihenfolge dem Kontextknoten folgen, mit Ausnahme der Nachkommen des Kontextknotens.
`following-sibling`		Liefert alle folgenden Geschwister des Kontextknotens.
`preceding`		Liefert alle Elementknoten vor dem Kontextknoten, mit Ausnahme der Vorfahren.
`preceding-sibling`		Liefert alle vorhergehenden Geschwister des Kontextknotens.
`attribute`	@	Liefert die Attributknoten des Kontextknotens.
`namespace`		Liefert, falls vorhanden, den Namensraumknoten des Kontextknotens.

Tabelle 5.2 Achsen in XPath (Forts.)

Die meisten Achsen weisen vom Kontextknoten aus gesehen entweder vorwärts oder rückwärts, wobei die Knotenmengen der Achsen `self`, `ancestor`, `descendent`, `following` und `preceding` zusammen alle Elemente des Dokuments umfassen, ohne dass es zu Überschneidungen kommt.

In den folgenden Abbildungen (siehe Abbildung 5.5 bis Abbildung 5.8) sind an einem Beispiel die Knotenmengen zusammengestellt, die die verschiedenen Achsen in Bezug auf einen bestimmten Kontextknoten liefern.

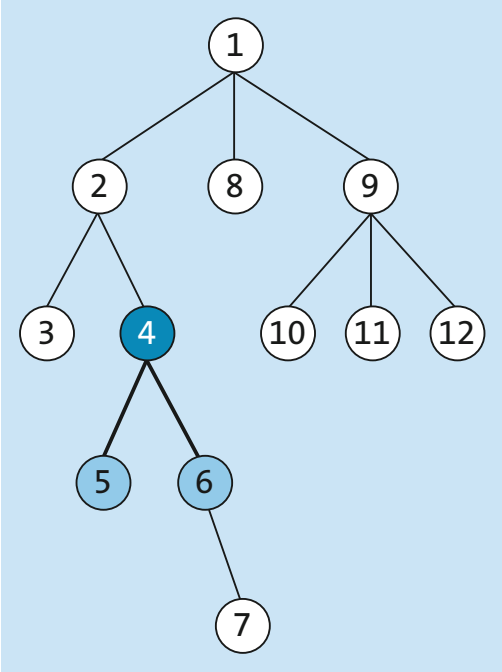


Abbildung 5.5 5 und 6 sind Kinder von 4.

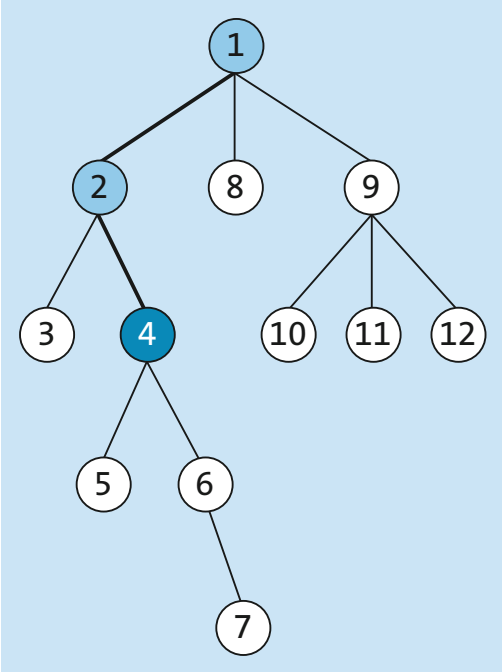


Abbildung 5.6 1 und 2 sind Vorfahren von 4.

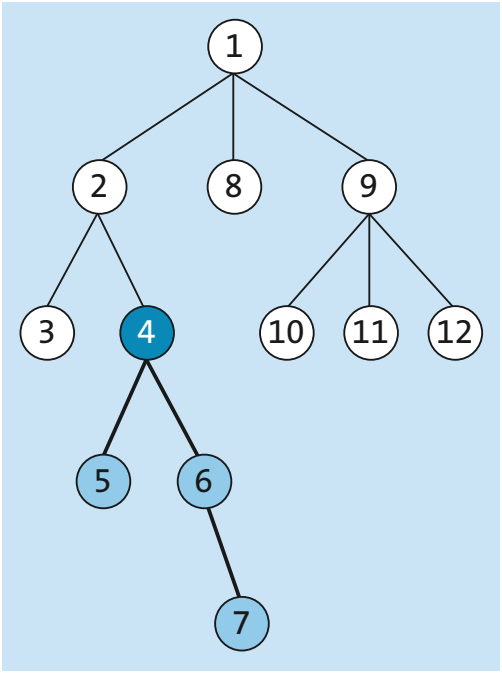


Abbildung 5.7 5, 6 und 7 sind Nachfahren von 4.

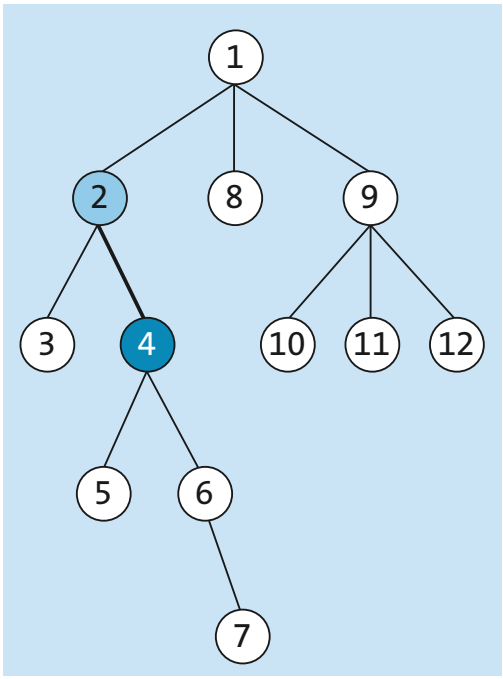


Abbildung 5.8 2 ist Elternteil von 4.

5.1.8 Knotentest

Die Knotenmenge, die durch die gewählte Achse ausgewählt wird, kann anschließend durch den sogenannten Knotentest weiter gefiltert werden. Dabei wird als Kriterium entweder ein bestimmter Knotenname oder ein bestimmter Knotentyp verwendet.

```
following::bezeichnung
```

wählt beispielsweise ausgehend vom jeweiligen Kontextknoten zunächst alle in der Dokumentreihenfolge folgenden Knoten aus und filtert daraus die Knoten mit dem Elementnamen `bezeichnung`. Dabei muss der Knotentyp des benannten Knotens mit dem grundsätzlichen Knotentyp der Achse übereinstimmen. Bei allen Achsen außer den Achsen `attribute` und `namespace` ist dies der Knotentyp `Element`. Knoten eines anderen Typs werden also ignoriert, auch wenn sie denselben Namen haben. Sollen Attributknoten zusammengestellt werden, wird der Attributname verwendet wie in

```
attribute::qualitaet
```

Sollen Knoten aufgrund ihres Knotentyps identifiziert werden, wird statt des Knotennamens eine entsprechende Funktion genutzt (siehe Tabelle 5.3).

Test per Knotentyp	Beschreibung
`text()`	Liefert alle Textknoten.
`comment()`	Liefert alle Kommentarknoten.
`processing-instruction()`	Liefert alle Knoten mit Verarbeitungsanweisungen.
`node()`	Liefert alle Knoten unabhängig vom jeweiligen Typ.

Tabelle 5.3 Knotentests

Um beispielsweise alle Textknoten, die Kinder des Kontextknotens sind, auszuwählen, kann

```
child::text()
```

verwendet werden. Der Ausdruck

```
descendant-or-self::comment()
```

stellt alle Kommentare des gesamten Dokuments zusammen. Der Funktion für `processing-instruction` kann auch ein Parameter übergeben werden, der das Ziel angibt.

5.1.9 Filtern mit Prädikaten

Mit Hilfe von Prädikaten lässt sich die Filterung der gewünschten Knotenmenge noch weiter verfeinern. Dabei wird durch einen logischen Ausdruck in eckigen Klammern eine Bedingung formuliert, die erfüllt sein muss, damit bestimmte Knoten ausgewählt werden. Im anderen Fall werden die entsprechenden Knoten aus der Knotenmenge entfernt. Werden mehrere Prädikate verwendet, bildet die Knotenmenge, die das Ergebnis des ersten Prädikats ist, den Ausgangspunkt für die Prüfung durch das zweite Prädikat etc.

Häufige Form eines Prädikatsausdrucks ist der Vergleich mit einer `string`-Konstanten. Der folgende Pfad

```
/child::weindepot/child::anbaugebiet[attribute::name="Ahr"]
```

findet zum Beispiel alle Weine aus dem Anbaugebiet »Ahr«. Findet ein solcher Vergleich mit einem String statt, zeigt der XPath-Ausdruck als Ergebnis den String-Wert der betroffenen Knoten an, der Tabelle 5.1 zu entnehmen ist.

Prädikatausdrücke unterstützen die üblichen logischen Operatoren `<`, `>`, `<=`, `>=`, `=` und `!=`. Allerdings muss dabei beachtet werden, dass Operatoren wie < oder > nicht unmittelbar in einem XML-Dokument erscheinen dürfen, sondern durch die entsprechenden Entitätsreferenzen `<` und `>` ersetzt werden müssen.

Auch mathematische Operatoren können in einem Prädikat eingesetzt werden. Die Folge ist, dass das Resultat des Ausdrucks von einem String-Wert in einen Zahlwert umgewandelt wird. Zur Verfügung stehen die numerischen Operatoren `+`, –, `*`, `div` und `mod`. Auch Klammern können verwendet werden.

5.1.10 Test von XPath-Ausdrücken

Wenn Sie die Formulierung von XPath-Ausdrücken etwas trainieren wollen, können Sie sich beispielsweise mit dem folgenden VBA-Makro behelfen, das sich in Excel anlegen lässt, vorausgesetzt, Sie binden durch einen Verweis im Visual-Basic-Editor die Microsoft XML-Bibliothek v6.0 – *msxml6.dll* – ein. Diese Bibliothek steht zur Verfügung, wenn der Microsoft XML-Parser installiert ist. Mehr dazu finden Sie in Kapitel 10, »Programmierschnittstellen für XML«. Während ältere Versionen des Parsers nur die abgekürzte Form der XPath-Ausdrücke unterstützen und deshalb auch nicht in der Lage sind, alle Achsen zu nutzen, gilt diese Einschränkung ab der Version 3 nicht mehr.

Mit Hilfe der `selectNodes`-Methode, einer von Microsoft bereitgestellten Erweiterung von DOM, können dann XPath-Ausdrücke für ein bestimmtes XML-Dokument ausgewertet werden. Sie liefern das aktuelle `node-set`, das dem jeweiligen Ausdruck entspricht.

```
Public Sub knotenabfrage()
  Dim XMLDokument As String
  Dim XPathexpression As String
  Dim i As Integer
  Dim menge As Integer
  Dim knotenliste As IXMLDOMNodeList
  Dim xml As New MSXML2.DOMDocument60

  XPathexpression = InputBox("XPath-Ausdruck: ")
  XMLDokument = FileSystem.CurDir + "\weindepot.xml"
  xml.Load XMLDokument
  xml.setProperty "SelectionLanguage", "XPath"
  Set knotenliste = xml.selectNodes(XPathexpression)
  menge = knotenliste.Length
  For i = 0 To menge - 1
    MsgBox (knotenliste.Item(i).xml)
  Next
End Sub
```

Listing 5.2 knotentest.xls

Die Angabe über die zu verwendende Abfragesprache

```
xml.setProperty "SelectionLanguage", "XPath"
```

ist wichtig, damit das Makro auch Ausdrücke in der ausführlichen Form akzeptiert. Wird diese Anweisung weggelassen, verhält sich die `selectNodes`-Methode so wie in den älteren Parser-Versionen, die nur die abgekürzten Ausdrücke verwerten konnten. Das Makro geht in diesem Fall davon aus, dass sich die XML-Datei in dem aktuellen Standardspeicherort von Excel befindet, der mit `Filesystem.CurDir` ausgelesen wird.

Die Ausgabe des Makros erscheint in diesem Fall einfach knotenweise in einem Meldungsdialog von Excel. Der XPath-Ausdruck `//wein` erzeugt zum Beispiel nacheinander die folgende Ausgabe:

```
<wein>
    <bezeichnung>Enkircher Pastor</bezeichnung>
    <rebsorte>Riesling</rebsorte>
    <richtung qualitaet="Auslese">halbtrocken</richtung>
    <preis>8,00</preis>
</wein>
<wein>
    <bezeichnung>Cochemer Hospiz</bezeichnung>
    <rebsorte>Riesling</rebsorte>
```

```
    <richtung qualitaet="Auslese">trocken</richtung>
    <preis>18,00</preis>
</wein>
```

XML-Entwicklungsumgebungen wie XMLSpy stellen Ihnen komfortable Werkzeuge für den Test von XPath-Ausdrücken zur Verfügung. Abbildung 5.9 zeigt ein Beispiel. Sie blenden dazu zunächst über XML • XPATH AUSWERTEN ein zusätzliches XPath-Fenster ein. In der ersten Eingabezeile geben Sie den XPath-Ausdruck ein. Mit den Schaltflächen darüber können Sie dazu im XML-Dokument markierte Elemente in das Eingabefeld übernehmen. Mehrere XPath-Abfragen lassen sich auf den durchnummerierten Registern ausführen.

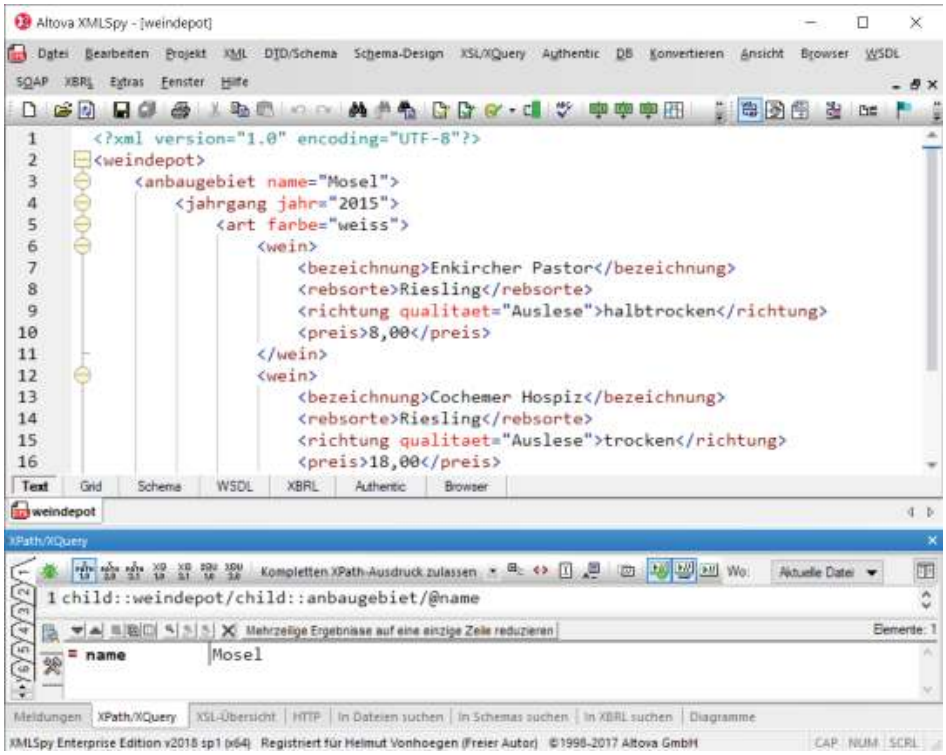


Abbildung 5.9 Testauswertung von XPath-Ausdrücken mit XMLSpy

5.1.11 XPath 1.0-Funktionen

XPath-Prozessoren müssen aufgrund der W3C-Empfehlung auch eine Bibliothek von Funktionen unterstützen. Diese Funktionen lassen sich insbesondere für die Formulierung ausdrucksstarker Prädikate einsetzen. Sie können entsprechend den von XPath 1.0-Ausdrücken gelieferten Datentypen in vier Gruppen eingeteilt werden:

- node-set
- string
- boolean
- number

Knotenmengenfunktionen

Häufig zum Einsatz kommen die in Tabelle 5.4 zusammengestellten Funktionen, die sich auf Knotenmengen beziehen, und zwar entweder auf die aktuelle Kontextknotenmenge oder auf eine explizit angegebene Knotenmenge.

In Bezug auf das Weindepot-Beispiel liefert die Funktion `count()` etwa mit

```
count(//wein)
```

die Anzahl der verschiedenen Weine. Durch Kombination der Funktionen `position()` und `last()` lässt sich der letzte Knoten in einer Knotenmenge auswählen:

```
//wein[position() = last()]
```

Um alle Elementknoten zu finden, die zu einem bestimmten Namensraum gehören, kann mit folgendem Ausdruck gearbeitet werden:

```
/descendent-or-self::*[namespace-uri() = "*"]
```

Knotenmengen-funktionen	Beschreibung
`last()`	Liefert eine Zahl, die die Größe der aktuellen Knotenmenge angibt – entspricht dem Wert, den auch `count()` liefert.
`position()`	Liefert die Position eines Mitglieds einer Knotenmenge.
`count()`	Liefert die Anzahl der Knoten in der Knotenmenge.
`id()`	Liefert die Knotenmenge, deren `ID`-Wert mit dem angegebenen von `object` übereinstimmt.
`local-name()`	Liefert den lokalen Teil des Knotennamens, der in der Knotenmenge an erster Position in der Dokumentreihenfolge steht, also der Element- oder Attributname ohne Präfix.
`namespace-uri()`	Liefert den URI des Namensraums des Knotens, der in der Knotenmenge an erster Position in der Dokumentreihenfolge steht.
`name()`	Liefert den qualifizierten Namen des Knotens, der in der Knotenmenge an erster Position in der Dokumentreihenfolge steht.

Tabelle 5.4 Knotenmengenfunktionen und ihre Bedeutung

String-Funktionen

Die `string`-Funktion wird verwendet, um ein Objekt in eine Zeichenkette umzuwandeln, also in ihren String-Wert. Was die String-Werte der verschiedenen Knotentypen sind, ist bereits in Tabelle 5.1 dargestellt worden.

Eine Reihe von Funktionen wird für die Verkettung und Zerlegung von Zeichenketten innerhalb von Lokalisierungspfaden angeboten. Damit können Knotenmengen aufgrund von String-Werten gefiltert werden. Um beispielsweise alle Weine zu markieren, deren Bezeichnung mit »En« beginnt, kann die folgende Funktion verwendet werden:

```
//child::wein[starts-with(bezeichnung, "En")]
```

Etwas ungewöhnlich ist die Funktion `translate`, die eine zeichenweise Ersetzung in einer Zeichenkette erlaubt. Das folgende Beispiel ergibt die Zeichenkette *BlauGelb*:

```
translate("blaugelb","bg","BG")
```

String-Funktionen	Beschreibung
`string()`	Wandelt ein Objekt in einen String um.
`concat()`	Verkettet die aufgeführten Zeichenketten.
`starts-with()`	Ergibt den Wert *wahr*, wenn die erste Zeichenkette mit der zweiten Zeichenkette beginnt.
`contains()`	Ergibt den Wert *wahr*, wenn die zweite Zeichenkette in der ersten enthalten ist.
`substring-before()`	Liefert aus der ersten Zeichenkette das, was vor dem Teil steht, der mit der zweiten Zeichenkette übereinstimmt.
`substring-after()`	Liefert aus der ersten Zeichenkette das, was hinter dem Teil steht, der mit dem ersten Auftreten der zweiten Zeichenkette übereinstimmt.
`substring()`	Liefert einen Teil der Zeichenkette, der an der mit dem zweiten Argument angegebenen Position beginnt und die mit dem dritten Argument angegebene Länge hat.
`string-length()`	Liefert die Anzahl der Zeichen in der Zeichenkette.
`normalize-space()`	Liefert die Zeichenkette mit normalisierten Leerzeichen. Führende und folgende Leerzeichen werden entfernt, interne Leerzeichen jeweils zu einem Leerzeichen zusammengefasst.

Tabelle 5.5 Liste der String-Funktionen

String-Funktionen	Beschreibung
`translate()`	Tauscht in der ersten Zeichenkette alle die Zeichen aus, die in der zweiten Zeichenkette vorkommen, und zwar jeweils durch die an der gleichen Position stehenden Zeichen der dritten Zeichenkette.

Tabelle 5.5 Liste der String-Funktionen (Forts.)

Logische Funktionen

Von den logischen Funktionen wird vor allem die `boolean`-Funktion häufig benutzt. Mit ihrer Hilfe kann sehr einfach getestet werden, ob eine Knotenmenge überhaupt Knoten enthält oder leer ist bzw. ob ein String-Wert tatsächlich eine Zeichenkette liefert oder leer ist.

Logische Funktionen	Beschreibung
`boolean()`	Wandelt das angegebene Objekt in einen logischen Wert um. Nichtleere Knotenmengen, nichtleere String-Werte und Zahlen größer als 0 ergeben *wahr*.
`not()`	Ergibt *wahr*, wenn das Objekt *falsch* ist.
`true()`	Gibt immer den Wert *wahr* zurück.
`false()`	Gibt immer den Wert *falsch* zurück.
`lang()`	Ergibt *wahr*, wenn der Wert von `xml:lang` mit dem Argument übereinstimmt, also derselben Sprache entspricht.

Tabelle 5.6 Liste der logischen Funktionen

Numerische Funktionen

XPath kennt auch einige wenige numerische Funktionen. Insbesondere wird die `number()`-Funktion genutzt, um ein Objekt in eine Zahl umzuwandeln. Das Ergebnis hängt vom Datentyp des betreffenden Objekts ab. Ein logisches Objekt liefert bei der Umwandlung den Wert `0` für *falsch* oder `1` für *wahr*; ein String-Wert liefert eine Zahl, wenn er als Zahl ausgewertet werden kann.

Praktisch ist auch die `sum()`-Funktion, mit der sehr einfach die Summe der Werte einer Knotenliste berechnet werden kann.

```
sum(//bestand)
```

liefert zum Beispiel die Summe aller Werte des Elements `<bestand>` in einer Lagerliste.

Numerische Funktionen	Beschreibung
`number()`	Wandelt das Objekt in eine Zahl um.
`sum()`	Liefert die Summe der Zahlenwerte der Knoten in der Knotenmenge, die sich aus der Umwandlung ihrer String-Werte ergeben.
`floor()`	Liefert die größte Ganzzahl, die nicht größer als die angegebene Zahl ist.
`ceiling()`	Liefert die kleinste Ganzzahl, die nicht kleiner als die angegebene Zahl ist.
`round()`	Rundet den Wert zur nächsten Ganzzahl.

Tabelle 5.7 Liste der numerischen Funktionen

5.2 XPath 2.0

Im Januar 2007 hat das W3C die Empfehlung für die nächste Version von XPath veröffentlicht. XPath 2.0 wird sowohl von XSLT 2.0 als auch von XQuery 1.0 verwendet. Alle Empfehlungen wurden zeitgleich verabschiedet. Während XSLT weiterhin vornehmlich zum Transformieren von XML-Daten eingesetzt wird, soll XQuery der Standard bei der Abfrage von XML-Dokumenten werden. XQuery 1.0 ist im Grunde eine Erweiterung von XPath 2.0. Die neue Abfragesprache für XML-Daten, die in Kapitel 9, »Abfragen mit XQuery«, vorgestellt wird, beansprucht, ähnlich leistungsfähig wie die weitverbreitete Datenbankabfragesprache SQL zu sein.

XPath 2.0 ist als Erweiterung von XPath 1.0 angelegt. Es ist dafür gesorgt, dass bis auf ganz wenige Ausnahmen jeder XPath 1.0-Ausdruck dasselbe Ergebnis liefert wie der entsprechende XPath 2.0-Ausdruck. Ein XPath 2.0-Prozessor kann dazu bei Bedarf einen entsprechenden Kompatibilitätsmodus nutzen, falls XPath 1.0-Ausdrücke zu verarbeiten sind.

Dieser Abschnitt konzentriert sich auf die Neuerungen, die mit XPath 2.0 angeboten werden. Für Tests kann beispielsweise die Entwicklungsumgebung XMLSpy oder der Saxon-Prozessor von Michael Kay, der selbst an der Erstellung des neuen Standards maßgeblich beteiligt war, genutzt werden. Beispiele sind in dem Abschnitt zu XSLT 2.0 zu finden. Auch das Microsoft .NET Framework unterstützt seit der Version 3.5 mit den Klassen in dem Namensraum `System.Xml.XPath` das Datenmodell von XPath 2.0 und XQuery 1.0.

5.2.1 Erweitertes Datenmodell

Die wichtigste Erweiterung in XPath 2.0 gegenüber XPath 1.0 betrifft zunächst das Datenmodell. Neben dem in Abschnitt 5.1.1, »Baummodell und XPath-Ausdrücke«, beschriebenen Baummodell, das die im XML-Dokument vorhandenen Informationseinheiten repräsentiert, berücksichtigt XPath 2.0 auch einzelne Werte, die als *atomic values* bezeichnet werden. Dabei kann es sich um Daten unterschiedlichen Typs handeln: Zeichenfolgen, Zahlen, logische Werte, Datums- oder Zeitwerte, aber auch qualifizierte Namen oder URIs. Der dritte Bereich sind Sequenzen oder Listen, die sowohl aus Einzelwerten als auch aus Bezügen auf Knoten in einem XML-Dokument bestehen können. Dabei handelt es sich um einfache Sequenzen, die nicht geschachtelt werden dürfen.

Die mit XPath 2.0 formulierbaren Ausdrücke sind in der Folge durch zusätzliche Operatoren und zahlreiche neue Funktionen dahingehend erweitert worden, dass sie geeignet sind, alle drei Bereiche des Datenmodells abzudecken. Während XPath 2.0 in Bezug auf den Knotenbaum nur Daten auslesen kann, besteht bei den Einzelwerten und bei den Sequenzen nun auch die Möglichkeit, neue Werte oder Sequenzen zu erzeugen. Im Ergebnis kann dann ein XPath 2.0-Ausdruck eine Auswahl von Knoten, einen Einzelwert oder eine Sequenz liefern.

Während XPath 1.0-Ausdrücke eine ungeordnete Knotenmenge zurückgeben, liefert ein XPath 2.0-Ausdruck immer eine Sequenz. Die Sequenz besteht aus einer geordneten Gruppe von Knoten oder atomaren Werten.

5.2.2 Neue Konstrukte für Ausdrücke

Für Operationen mit Sequenzen kann insbesondere der `for`-Ausdruck verwendet werden. Er erlaubt es, eine Operation für alle Datenelemente in einer Sequenz vorzunehmen, etwa um daraus eine neue Sequenz zu erzeugen.

Schleifen

Ein einfaches Beispiel ist etwa der Ausdruck:

```
for $i in 1 to 3 return $i*$i
```

Das Resultat dieses Ausdrucks ist die Sequenz `1, 4, 9`. Der Ausdruck arbeitet zunächst mit einer Variablen `i`, die durch das `$`-Zeichen als solche gekennzeichnet wird. Diese Variable wird als *Bereichsvariable* bezeichnet. Hinter dem Schlüsselwort `in` wird die Bindungssequenz angegeben und hinter dem Schlüsselwort `return` der `return`-Ausdruck. Das Ergebnis des `for`-Ausdrucks wird dadurch erzielt, dass für jedes Element in der Bindungssequenz jeweils einmal der `return`-Ausdruck geliefert wird.

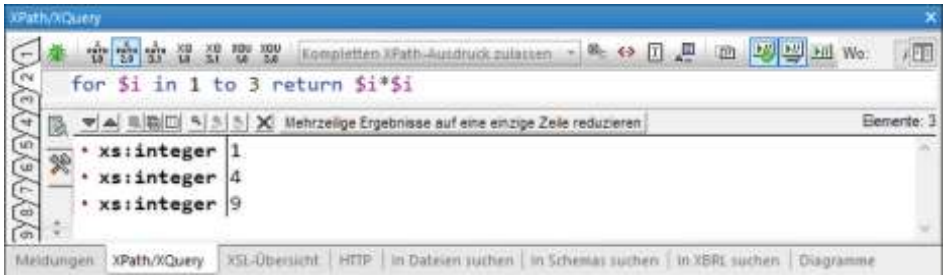


Abbildung 5.10 Wenn die Schaltfläche »XPath 2.0« aktiviert ist, liefert der XPath-Ausdruck das gewünschte Ergebnis.

Besteht die Sequenz aus Knotenreferenzen, liefert beispielsweise ein Ausdruck wie

```
for $n in child::* return name($n)
```

eine Liste aller Kindelemente des aktuellen Knotens in einem Dokument.

Bedingte Ausdrücke

Mit `if` lassen sich nun auch bedingte Ausdrücke formulieren, etwa:

```
if @menge > 1000 then "gut" else "weniger gut"
```

Nützlich sind auch die quantifizierenden Ausdrücke mit `some` oder `every`.

```
some $a in $lager/artikel satisfies  $lager/artikel/menge = 0
```

ist wahr, wenn die Menge wenigstens bei einem Artikel gleich null ist.

```
every $a in $lager/artikel satisfies  $lager/artikel/menge > 0
```

ist wahr, wenn von allen Artikeln wenigstens einer vorhanden ist.

5.2.3 Neue Datentypen

Für XPath 2.0 und XQuery 1.0 gilt ein neues Datentypsystem, das insbesondere darauf ausgelegt ist, die durch XML Schema gegebenen Datentypen zu unterstützen.

Während XPath 1.0 nur einen einzigen numerischen Datentyp – `number` – unterstützt, stellt XPath 2.0 auch die Typen `integer`, `decimal`, `float` und `double` zur Verfügung. Zahlreiche neue Datentypen sind für Datums-, Zeit- und Dauerwerte vorhanden. Zudem werden nun auch benutzerdefinierte Datentypen unterstützt, die über ein entsprechendes XML-Schema definiert werden können.

Was den Teil des Datenmodells betrifft, der das XML-Dokument repräsentiert, sind die Änderungen in XPath 2.0 eher gering. Hinzugekommen ist im Wesentlichen die

Möglichkeit, Element- und Attributknoten in Bezug auf den verwendeten Datentyp entsprechende Typkennzeichnungen zuzuordnen, die durch das benutzte Schema definiert sind. Ist ein XML-Dokument nicht über ein Schema validiert, wird die Typkennzeichnung auf die Werte `untyped` für Elemente und `untypedAtomic` für Attribute gesetzt. Mehr zur Typhierarchie für XPath 2.0 und XQuery 1.0 finden Sie in Kapitel 9, »Abfragen mit XQuery«.

5.2.4 Neue Operatoren

Um die Möglichkeiten des erweiterten Datenmodells nutzen zu können, bietet XPath 2.0 zahlreiche neue Operatoren an. Für den Knotenvergleich können beispielsweise die Operatoren `is`, `<<` und `>>` eingesetzt werden. Der erste Operator prüft, ob zwei Ausdrücke denselben Knoten liefern, die beiden anderen Operatoren stellen fest, welcher von zwei Knoten in der Dokumentreihenfolge früher oder später erscheint.

Für die Kombination von Knotensequenzen können die Operatoren `union`, `intersect` und `except` verwendet werden. Mit `union` vereinigen Sie zwei Knotensequenzen zu einer Sequenz, wobei Duplikate verhindert werden. Der Operator `intersect` dagegen erzeugt aus zwei Sequenzen eine Sequenz, die genau die Knoten enthält, die in beiden Sequenzen vorkommen. Mit `except` wird eine Sequenz aus zwei Knotensequenzen erzeugt, die nur Knoten enthält, die in der ersten Sequenz vorkommen, aber nicht in der zweiten.

Für den Vergleich von Einzelwerten werden die Operatoren `eq`, `ne`, `lt`, `le`, `gt` und `ge` angeboten, die den XPath 1.0-Operatoren `=`, `!=`, `<`, `<=`, `>` und `>=` entsprechen.

Für die Division von Ganzzahlen wird der Operator `idiv` angeboten. Um einen Bereich von Ganzzahlen zu erzeugen, kann mit `to` gearbeitet werden, etwa `0 to 100`.

Außerdem sind zahlreiche Operatoren hinzugekommen, die mit speziellen Datentypen verknüpft sind, beispielsweise den zahlreichen Daten-, Zeit- oder Dauerdatentypen. Die komplette Liste der neuen Operatoren finden Sie in Tabelle 5.9.

5.2.5 Die erweiterte Funktionenbibliothek

Die Zahl der in XPath 2.0 verwendbaren Funktionen ist vervielfacht worden. Zahlreiche Funktionen stehen jetzt für die Behandlung von Zeichenketten zur Verfügung, insbesondere können jetzt auch reguläre Ausdrücke verwendet werden.

Neu sind vor allem die Funktionen für die Arbeit mit Sequenzen. Sehr umfangreich ist auch die Gruppe der Funktionen, die bei Operationen mit Datums- und Zeitwerten eingesetzt werden können.

In Tabelle 5.8 sind die in XPath 2.0 verfügbaren Funktionen nach Gruppen geordnet zusammengestellt.

Funktion	Beschreibung
Zugriffsfunktionen	
fn:node-name	Liefert den kompletten xs:QName.
fn:nilled	Liefert einen xs:boolean-Wert, der anzeigt, ob für den Knoten das Attribut xs:nil="true" gilt.
fn:string	Liefert den Argumentwert als xs:string.
fn:data	Liefert eine Sequenz atomarer Werte.
fn:base-uri	Liefert den Basis-URI des Arguments.
fn:document-uri	Liefert den Dokument-URI des Arguments.
Fehlerfunktion	
fn:error	Ruft einen Fehler auf.
Trace-Funktion	
fn:trace	Liefert für das Debuggen die Ausführungsabfolge.
Spezielle Konstruktorfunktion	
fn:dateTime	Setzt einen Wert aus einer Datums- und einer Zeitangabe zusammen.
Funktionen für numerische Werte	
fn:abs	Liefert den absoluten Wert des Arguments.
fn:ceiling	Liefert die kleinste Ganzzahl, die größer oder gleich dem Argument ist.
fn:floor	Liefert die größte Ganzzahl, die kleiner oder gleich dem Argument ist.
fn:round	Rundet auf die nächstliegende Ganzzahl.
fn:round-half-to-even	Liefert eine Zahl mit der angegebenen Präzision.
String-Funktionen	
fn:codepoints-to-string	Bildet einen xs:string aus einer Sequenz von Unicode-Codes.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0

Funktion	Beschreibung
`fn:string-to-codepoints`	Liefert die Sequenz der Unicode-Codes, die dem `Xs:string` entsprechen.
`fn:compare`	Liefert `-1`, `0`, oder `1`, je nachdem, ob der erste Wert kleiner, gleich oder größer als der zweite ist.
`fn:codepoint-equal`	Liefert `true`, wenn die beiden Argumente nach Anwendung einer Unicode-Kollation gleich sind.
`fn:concat`	Verkettet zwei oder mehr `xs:anyAtomicType`-Argumente zu einem `xs:string`.
`fn:string-join`	Liefert einen `xs:string` durch Verknüpfung einer Sequenz von Zeichenfolgen, wobei ein Separator verwendet werden kann.
`fn:substring`	Liefert den `xs:string`, der an der angegebenen Stelle im Argument zu finden ist.
`fn:string-length`	Liefert die Länge des Arguments.
`fn:normalize-space`	Liefert den `whitespace-normalized`-Wert des Arguments.
`fn:normalize-unicode`	Liefert den normalisierten Wert in der mit dem zweiten Argument angegebenen Form.
`fn:upper-case`	Gibt den Wert des Arguments in Großbuchstaben an.
`fn:lower-case`	Gibt den Wert des Arguments in Kleinbuchstaben an.
`fn:translate`	Ersetzt die im ersten Argument vorkommenden Zeichen durch die an der entsprechenden Stelle im zweiten Argument vorkommenden Zeichen.
`fn:encode-for-uri`	Liefert das `xs:string`-Argument mit maskierten Zeichen, damit es im URI verwendet werden kann.
`fn:iri-to-uri`	Liefert das `xs:string`-Argument mit maskierten Zeichen, damit der resultierende String als (Teil eines) URI verwendet werden kann.
`fn:escape-html-uri`	Liefert das `xs:string`-Argument mit maskierten Zeichen, so dass es unter HTML für Attributwerte in einem URI verwendet werden kann.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0 (Forts.)

Funktion	Beschreibung
Substring-Funktionen	
`fn:contains`	Zeigt an, ob ein `xs:string`-Wert einen anderen `xs:string`-Wert enthält.
`fn:starts-with`	Zeigt an, ob ein `xs:string`-Wert mit dem anderen angegebenen `xs:string`-Wert beginnt.
`fn:ends-with`	Zeigt an, ob ein `xs:string`-Wert mit dem anderen angegebenen `xs:string`-Wert endet.
`fn:substring-before`	Liefert den vorausgehenden Teilstring.
`fn:substring-after`	Liefert den nachfolgenden Teilstring.
Pattern-Matching-Funktionen	
`fn:matches`	Zeigt an, ob ein Wert zu dem mit dem zweiten Argument angegebenen regulären Ausdruck passt.
`fn:replace`	Liefert den Wert des ersten Arguments, wobei jeder Teilstring, der zu dem regulären Ausdruck des zweiten Arguments passt, durch die im dritten Ausdruck angegebenen Strings ersetzt wird.
`fn:tokenize`	Liefert eine Sequenz von Strings, deren Werte Teilstrings des ersten Arguments sind, getrennt durch Separatoren, die zu dem regulären Ausdruck im zweiten Argument passen.
URI-Funktion	
`fn:resolve-uri`	Liefert einen absoluten URI aus einem Basis-URI und einem relativen URI.
Logische Funktionen	
`fn:true`	Liefert den `xs:boolean`-Wert `true`.
`fn:false`	Liefert den `xs:boolean`-Wert `false`.
`fn:not`	Kehrt den `xs:boolean`-Wert um.
Funktionen zur Zeitzonen-Anpassung	
`fn:adjust-dateTime-to-timezone`	Passt den `xs:dateTime`-Wert an die Zeitzone an.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0 (Forts.)

Funktion	Beschreibung
fn:adjust-date-to-timezone	Passt den xs:date-Wert an die Zeitzone an.
fn:adjust-time-to-timezone	Passt den xs:time-Wert an die Zeitzone an.
Funktionen zu Dauer, Datum, Zeit und Zeitzonen	
fn:years-from-duration	Liefert die Jahre von xs:duration.
fn:months-from-duration	Liefert die Monate von xs:duration.
fn:days-from-duration	Liefert die Tageszahl von xs:duration.
fn:hours-from-duration	Liefert die Stunden von xs:duration.
fn:minutes-from-duration	Liefert die Minuten von xs:duration.
fn:seconds-from-duration	Liefert die Sekunden von xs:duration.
fn:year-from-dateTime	Liefert das Jahr von xs:dateTime.
fn:month-from-dateTime	Liefert den Monat von xs:dateTime.
fn:day-from-dateTime	Liefert den Tag von xs:dateTime.
fn:hours-from-dateTime	Liefert die Stunden von xs:dateTime.
fn:minutes-from-dateTime	Liefert die Minuten von xs:dateTime.
fn:seconds-from-dateTime	Liefert die Sekunden von xs:dateTime.
fn:timezone-from-dateTime	Liefert die Zeitzone xs:dateTime.
fn:year-from-date	Liefert das Jahr von xs:date.
fn:month-from-date	Liefert den Monat von xs:date.
fn:day-from-date	Liefert den Tag von xs:date.
fn:timezone-from-date	Liefert die Zeitzone von xs:date.
fn:hours-from-time	Liefert die Stunden von xs:time.
fn:minutes-from-time	Liefert die Minuten von xs:time.
fn:seconds-from-time	Liefert die Sekunden von xs:time.
fn:timezone-from-time	Liefert die Zeitzone von xs:time.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0 (Forts.)

Funktion	Beschreibung
QName-Funktionen	
`fn:resolve-QName`	Liefert den entsprechenden `xs:QName`.
`fn:QName`	Liefert den `xs:QName` mit dem Namensraum des ersten Arguments und dem lokalen Namen und dem Präfix des zweiten Arguments.
`fn:prefix-from-QName`	Liefert den `xs:NCName` für das angegebene Präfix.
`fn:local-name-from-QName`	Liefert den `xs:NCName` mit dem lokalen Namen.
`fn:namespace-uri-from-QName`	Liefert den Namensraum-URI für das `xs:QName`-Argument.
`fn:namespace-uri-for-prefix`	Liefert den Namensraum-URI für das gegebene Element.
`fn:in-scope-prefixes`	Liefert das Präfix für das gegebene Element.
Knotenfunktionen	
`fn:name`	Liefert den Knotennamen als `xs:string`.
`fn:local-name`	Liefert den lokalen Namen des betreffenden Knotens als `xs:NCName`.
`fn:namespace-uri`	Liefert den Namensraum-URI des Knotens als `xs:anyURI`.
`fn:number`	Liefert den Wert des Arguments.
`fn:lang`	Gibt an, ob die Sprache des Knotens der mit `xml:lang` angegebenen Sprache entspricht.
`fn:root`	Liefert den Wurzelknoten.
Allgemeine Sequenzfunktionen	
`fn:boolean`	Liefert den logischen Wert der Sequenz.
`fn:index-of`	Liefert eine Sequenz von Integer-Werten, wobei jeder der Index eines Mitglieds der Sequenz des ersten Arguments ist, das gleich dem Wert des zweiten Arguments ist.
`fn:empty`	Gibt an, ob die betreffende Sequenz leer ist.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0 (Forts.)

Funktion	Beschreibung
fn:exists	Gibt an, ob die betreffende Sequenz nicht leer ist.
fn:distinct-values	Entfernt Duplikate aus einer Sequenz.
fn:insert-before	Fügt ein einzelnes Datenelement oder eine Sequenz an eine Stelle einer Sequenz ein.
fn:remove	Entfernt ein Datenelement von der angegebenen Position einer Sequenz.
fn:reverse	Kehrt die Reihenfolge der Datenelemente in einer Sequenz um.
fn:subsequence	Liefert die Teilsequenz einer gegebenen Sequenz, identifiziert über die Position.
fn:unordered	Liefert die Datenelemente in einer Reihenfolge, die von der Implementierung abhängig ist.
Testfunktionen für Sequenzen	
fn:zero-or-one	Liefert die Sequenz, wenn sie kein oder zumindest ein Datenelement enthält.
fn:one-or-more	Liefert die Sequenz, wenn sie ein oder mehrere Datenelemente enthält.
fn:exactly-one	Liefert die Sequenz, wenn sie genau ein Datenelement enthält.
fn:deep-equal	Ergibt true, wenn die beiden Argumente Datenelemente enthalten, die an den entsprechenden Positionen übereinstimmen.
Aggregatfunktionen	
fn:count	Liefert die Zahl der Datenelemente einer Sequenz.
fn:avg	Liefert den Durchschnitt einer Wertesequenz.
fn:max	Liefert den Höchstwert einer Wertesequenz.
fn:min	Liefert den Tiefstwert einer Wertesequenz.
fn:sum	Liefert die Summe einer Wertesequenz.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0 (Forts.)

Funktion	Beschreibung
Sequenzgenerierende Funktionen	
`fn:id`	Liefert die Sequenz der Elementknoten, die einen `ID`-Wert haben, der zu einem `IDREF`-Wert passt.
`fn:idref`	Liefert die Sequenz der Element- oder Attributknoten, die einen `ID`-Wert haben, der zu einem `IDREF`-Wert passt.
`fn:doc`	Liefert einen `document`-Knoten über den angegebenen URI.
`fn:doc-available`	Ergibt `true`, wenn der `document`-Knoten über den URI gefunden werden kann.
`fn:collection`	Liefert eine Knotensequenz über den angegebenen URI oder die Knoten der vorgegebenen Kollektion.
Kontextfunktionen	
`fn:position`	Liefert die Position des Kontext-Datenelements in der Sequenz.
`fn:last`	Liefert die Zahl der Datenelemente in der Sequenz, die gerade verarbeitet wird.
`fn:current-dateTime`	Liefert das aktuelle `xs:dateTime`.
`fn:current-date`	Liefert das aktuelle `xs:date`.
`fn:current-time`	Liefert die aktuelle `xs:time`.
`fn:implicit-timezone`	Liefert den Wert der impliziten Zeitzonen-Eigenschaft des dynamischen Kontextes.
`fn:default-collation`	Liefert den Wert der vorgegebenen `collation`-Eigenschaft des statischen Kontextes.
`fn:static-base-uri`	Liefert den Wert der `Base URI`-Eigenschaft des statischen Kontextes.

Tabelle 5.8 Liste der neuen Funktionen in XPath 2.0 (Forts.)

Fast ebenso umfangreich ist die Liste der neuen Operatoren, die mit XPath 2.0 im Zusammenhang mit den neuen Funktionen und Datentypen zur Verfügung stehen. Tabelle 5.9 gibt einen Überblick.

Operator	Beschreibung
Operatoren für numerische Werte	
op:numeric-add	Addition
op:numeric-subtract	Subtraktion
op:numeric-multiply	Multiplikation
op:numeric-divide	Division
op:numeric-integer-divide	Ganzzahldivision
op:numeric-mod	Modulus
op:numeric-unary-plus	positives Vorzeichen
op:numeric-unary-minus	negatives Vorzeichen
Vergleichsoperatoren für numerische Werte	
op:numeric-equal	Prüft die Gleichheit numerischer Werte.
op:numeric-less-than	Prüft, ob der erste Wert kleiner ist.
op:numeric-greater-than	Prüft, ob der erste Wert größer ist.
Operatoren für logische Werte	
op:boolean-equal	Prüft die Gleichheit logischer Werte.
op:boolean-less-than	Prüft, ob der erste Wert kleiner ist.
op:boolean-greater-than	Prüft, ob der erste Wert größer ist.
Vergleichsoperatoren für Datum-, Zeit- und Dauerwerte	
op:yearMonthDuration-less-than	Prüft, ob das Jahr-Monats-Intervall kleiner ist.
op:yearMonthDuration-greater-than	Prüft, ob das Jahr-Monats-Intervall größer ist.
op:dayTimeDuration-less-than	Prüft, ob das Tag-Uhrzeit-Intervall kleiner ist.
op:dayTimeDuration-greater-than	Prüft, ob das Tag-Uhrzeit-Intervall größer ist.
op:duration-equal	Prüft auf Übereinstimmung der Dauer.

Tabelle 5.9 Neue Operatoren in XPath 2.0

Operator	Beschreibung
op:dateTime-equal	Prüft auf Gleichheit von Datum und Uhrzeit.
op:dateTime-less-than	Prüft, ob der Wert für Datum und Uhrzeit kleiner ist.
op:dateTime-greater-than	Prüft, ob der Wert für Datum und Uhrzeit größer ist.
op:date-equal	Prüft auf Gleichheit des Datums.
op:date-less-than	Prüft, ob der Wert für Datum kleiner ist.
op:date-greater-than	Prüft, ob der Wert für Datum größer ist.
op:time-equal	Prüft auf Gleichheit der Uhrzeit.
op:time-less-than	Prüft, ob der Uhrzeitwert kleiner ist.
op:time-greater-than	Prüft, ob der Uhrzeitwert größer ist.
op:gYearMonth-equal	Prüft auf Gleichheit eines gregorianischen Jahr-Monats-Werts.
op:gYear-equal	Prüft auf Gleichheit eines gregorianischen Jahreswerts.
op:gMonthDay-equal	Prüft auf Gleichheit eines gregorianischen Monat-Tag-Werts.
op:gMonth-equal	Prüft auf Gleichheit eines gregorianischen Monatswerts.
op:gDay-equal	Prüft auf Gleichheit eines gregorianischen Tageswerts.
Arithmetische Operatoren für Dauerwerte	
op:add-yearMonthDurations	Addiert ein Jahr-Monats-Intervall.
op:subtract-yearMonthDurations	Subtrahiert ein Jahr-Monats-Intervall.
op:multiply-yearMonthDuration	Multipliziert ein Jahr-Monats-Intervall.
op:divide-yearMonthDuration	Dividiert ein Jahr-Monats-Intervall.

Tabelle 5.9 Neue Operatoren in XPath 2.0 (Forts.)

Operator	Beschreibung
op:divide-yearMonthDuration-by-yearMonthDuration	Dividiert ein Jahr-Monats-Intervall durch ein Jahr-Monats-Intervall.
op:add-dayTimeDurations	Addiert ein Tag-Uhrzeit-Intervall.
op:subtract-dayTimeDurations	Subtrahiert ein Tag-Uhrzeit-Intervall.
op:multiply-dayTimeDuration	Multipliziert ein Tag-Uhrzeit-Intervall.
op:divide-dayTimeDuration	Dividiert ein Tag-Uhrzeit-Intervall.
op:divide-dayTimeDuration-by-dayTimeDuration	Dividiert ein Tag-Uhrzeit-Intervall durch ein Tag-Uhrzeit-Intervall.
Arithmetische Operatoren für Datum-, Zeit- und Dauerwerte	
op:subtract-dateTimes	Subtrahiert einen Datum- und Uhrzeitwert.
op:subtract-dates	Subtrahiert einen Datumswert.
op:subtract-times	Subtrahiert einen Zeitwert.
op:add-yearMonthDuration-to-dateTime	Addiert ein Jahr-Monats-Intervall zu einem Wert mit Datum und Uhrzeit.
op:add-dayTimeDuration-to-dateTime	Addiert ein Tag-Uhrzeit-Intervall zu einem Wert mit Datum und Uhrzeit.
op:subtract-yearMonthDuration-from-dateTime	Subtrahiert ein Jahr-Monats-Intervall von einem Wert mit Datum und Uhrzeit.
op:subtract-dayTimeDuration-from-dateTime	Subtrahiert ein Tag-Uhrzeit-Intervall von einem Wert mit Datum und Uhrzeit.
op:add-yearMonthDuration-to-date	Addiert ein Jahr-Monats-Intervall zu einem Datum.
op:add-dayTimeDuration-to-date	Addiert ein Tag-Uhrzeit-Intervall zu einem Datum.

Tabelle 5.9 Neue Operatoren in XPath 2.0 (Forts.)

Operator	Beschreibung
op:subtract-yearMonthDuration-from-date	Subtrahiert ein Jahr-Monats-Intervall von einem Datum.
op:subtract-dayTimeDuration-from-date	Subtrahiert ein Tag-Uhrzeit-Intervall von einem Datum.
op:add-dayTimeDuration-to-time	Addiert ein Tag-Uhrzeit-Intervall zur Uhrzeit.
op:subtract-dayTimeDuration-from-time	Subtrahiert ein Tag-Uhrzeit-Intervall von der Uhrzeit.
Operatoren in Bezug auf QNamen	
op:QName-equal	Prüft auf Gleichheit bei QNamen.
Operatoren für base64Binary und hexBinary	
op:hexBinary-equal	Prüft hexadezimal-binäre Werte auf Gleichheit.
op:base64Binary-equal	Prüft base64-binäre Werte auf Gleichheit.
Operator für NOTATION	
op:NOTATION-equal	Prüft die Übereinstimmung von Namen in Namensräumen.
Operatoren für Knoten	
op:is-same-node	Prüft, ob es derselbe Knoten ist.
op:node-before	Prüft, ob es der vorige Knoten ist.
op:node-after	Prüft, ob es der folgende Knoten ist.
Allgemeiner Operator für Sequenzen	
op:concatenate	Verknüpft zwei Sequenzen.
Operatoren für die Bildung von Sequenzen aus Knotenmengen	
op:union	Vereinigung ohne Duplikate
op:intersect	Schnittmenge ohne Duplikate
op:except	Differenzmenge ohne Duplikate

Tabelle 5.9 Neue Operatoren in XPath 2.0 (Forts.)

Operator	Beschreibung
Operator zum Generieren von Sequenzen	
`op:to`	Liefert eine Sequenz von Ganzzahlen zwischen zwei Werten.

Tabelle 5.9 Neue Operatoren in XPath 2.0 (Forts.)

Mehr zur praktischen Arbeit mit XPath 2.0 finden Sie in Kapitel 7, »Umwandlungen mit XSLT«, und in Kapitel 9, »Abfragen mit XQuery«.

5.3 XPath 3.0 und XPath 3.1

Im April 2014 hat das W3C die Empfehlung für XPath 3.0 verabschiedet. Die neue Version der Adressierungssprache erlaubt die Verarbeitung von Werten, die dem in der gleichzeitig veröffentlichten Empfehlung XQuery und XPath Model 3.0 definierten Datenmodell entsprechen, das auch als zweite Version des bisherigen Datenmodells bezeichnet wird.

Außerdem wurde mit der ebenfalls gleichzeitig erschienenen Empfehlung XPath and XQuery Functions and Operators 3.0 die Zahl der Funktionen, Datentypen und Operatoren noch einmal wesentlich erweitert.

Im März 2017 wurde die Empfehlung für XPath 3.1 verabschiedet, die das Datenmodell noch um die Unterstützung von Maps und Arrays erweitert und dafür entsprechende Ausdrücke und Funktionen zur Verfügung stellt.

Maps enthalten Zuordnungen; Abbildung 5.11 zeigt ein einfaches Beispiel:

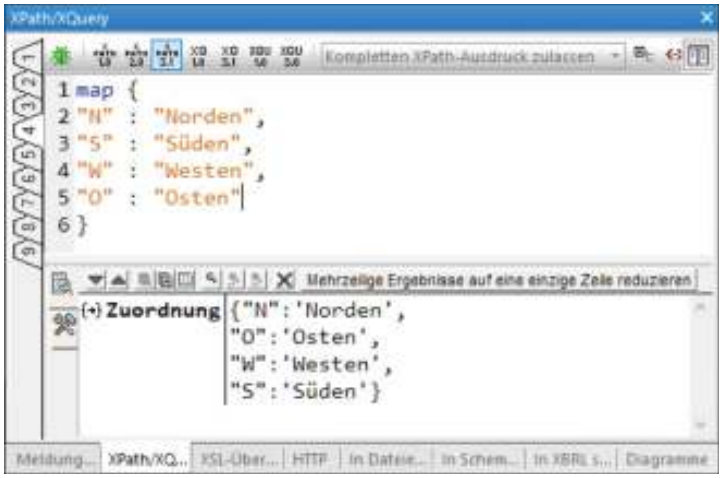


Abbildung 5.11 Mit Maps werden Schlüssel-Werte-Paare gebildet.

Ein einfaches Beispiel für einen Ausdruck mit einem Array ist

```
[1, 4, 7, 10](3)
```

und liefert 7.

Außerdem lässt sich XPath nun auch für die Auswertung von *JSON* (*JavaScript Object Notation*) verwenden, dem textbasierten Datenformat, das für eine einfachere Kommunikation zwischen Server und Client entwickelt wurde.

Funktionserweiterungen

Neu sind seit der Version 3.0 dynamische Funktionsaufrufe. Dabei werden Funktionswerte aufgerufen, ohne den Funktionsnamen direkt zu verwenden.

```
$f[3] ("Hallo Welt")
```

holt beispielsweise das dritte Element aus der Sequenz `$f`, ruft es als Funktion auf und übergibt das in der Klammer angegebene Argument. Der erste Teil des Ausdrucks muss also immer eine Funktion liefern, der zweite Teil enthält die Argumente, die an diese Funktion übergeben werden.

XPath 3.0 erlaubt nun auch die Erweiterung der Funktionsbibliothek durch selbst programmierte Inline-Funktionsausdrücke. Diese Funktionen bleiben namenlos, es handelt sich also um anonyme Funktionen. Sie werden direkt innerhalb eines gültigen XPath-Ausdrucks gebildet, Variablen gelten deshalb auch nur innerhalb dieses Ausdrucks. Hier ein einfaches Beispiel:

```
function ($netto) {$netto * 1,19}
```

Die Funktion, die immer mit dem allgemeinen Namen `function` beginnt, deklariert in der einfachen Klammer, wenn nötig, ein oder mehrere Argumente für die Funktion. Die geschweiften Klammern enthalten jeweils den Body der Funktion, hier wird also angegeben, was die Funktion liefern soll.

Eine bedeutende Neuerung seit XPath 3.0 ist die Möglichkeit, mit höherrangigen Funktionen zu arbeiten. Sie sind dadurch definiert, dass sie entweder als Argumente selbst wieder Funktionen verwenden können oder als Ergebnis Funktionen liefern. Solche Funktionen können als anonyme Funktionen definiert werden. Als höherrangig sind aber auch einige vorgegebene Funktionen eingestuft, die Sie am Ende von Tabelle 5.10 finden.

Abbildung 5.12 zeigt den Test einer solchen Funktion in XMLSpy. Über die Schaltfläche XPath 3.1 müssen Sie dazu zunächst die entsprechenden Funktionen aktivieren.

Der Ausdruck

```
fn:for-each(1 to 6, function($a) {$a * $a}
```

verwendet als Argument eine Inline-Funktion. Das Ergebnis wird auf den Ergebnisregistern des Fensters ausgegeben.

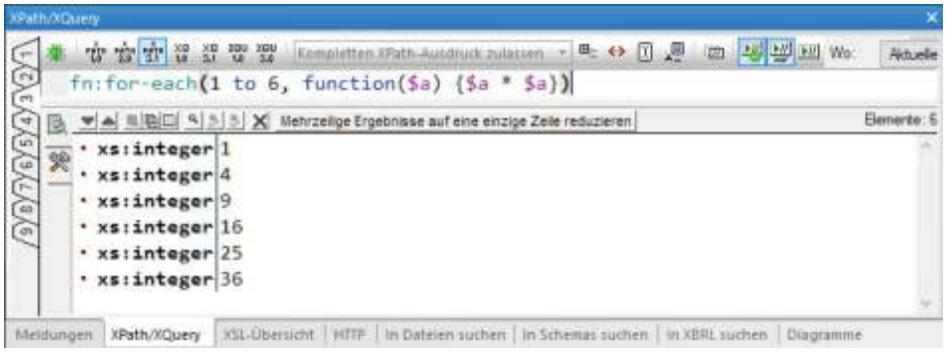


Abbildung 5.12 Test einer übergeordneten Funktion in XMLSpy

In Tabelle 5.10 sind die neuen Funktionen und Datentypen von XPath 3.1 nach Sachgebieten zusammengestellt.

Funktion	Beschreibung
Neue Funktionen, um Zahlen zu formatieren	
`fn:format-integer`	Formatiert eine Ganzzahl in einer als Zeichenfolge angegebenen Form.
`fn:format-number`	Liefert eine Zeichenfolge, die eine Zahl wiedergibt, deren Format durch eine Zeichenfolge beschrieben ist.
Neue trigonometrische und andere mathematische Funktionen	
`math:pi`	Liefert einen Näherungswert für die Konstante π.
`math:exp`	Liefert den Wert von e^x.
`math:exp10`	Liefert den Wert von 10^x.
`math:log`	Liefert den natürlichen Logarithmus des Arguments.
`math:log10`	Liefert den Logarithmus zur Basis 10.
`math:pow`	Erhebt das erste Argument in die mit dem zweiten Argument angegebene Potenz.
`math:sqrt`	Liefert die Wurzel des angegebenen Arguments.
`math:sin`	Liefert den Sinuswert des Arguments, angegeben im Bogenmaß.

Tabelle 5.10 Neue Funktionen und Operatoren in XPath 3.1

Funktion	Beschreibung
math:cos	Liefert den Kosinuswert des Arguments, angegeben im Bogenmaß.
math:tan	Liefert den Tangenswert des Arguments, angegeben im Bogenmaß.
math:asin	Liefert für den angegebenen Sinuswert den Winkel im Bogenmaß.
math:acos	Liefert für den angegebenen Kosinuswert den Winkel im Bogenmaß.
math:atan	Liefert für den angegebenen Tangenswert den Winkel im Bogenmaß.
math:atan2	Liefert direkt den Steigungswinkel für die beiden Koordinaten.
Eine neue String-Funktion, die reguläre Ausdrücke verwendet	
fn:analyze-string	Analysiert eine Zeichenfolge mit Hilfe eines regulären Ausdrucks, um zu prüfen, ob es Übereinstimmungen gibt.
Zusätzliche Datentypen zur Dauer	
xs:yearMonthDuration	Dieser von xs:duration abgeleitete Datentyp berücksichtigt nur die Monats- und Jahresanteile.
xs:dayTimeDuration	Dieser von xs:duration abgeleitete Datentyp berücksichtigt nur die Tages-, Stunden-, Minuten-, und Sekundenanteile.
Neue Funktionen zur Formatierung von Daten und Zeitangaben	
fn:format-dateTime	Liefert eine Zeichenfolge für einen xs:dateTime-Wert in dem als Zeichenfolge angegebenen Format.
fn:format-date	Liefert eine Zeichenfolge für einen xs:date-Wert in dem als Zeichenfolge angegebenen Format.
fn:format-time	Liefert eine Zeichenfolge für einen xs:time-Wert in dem als Zeichenfolge angegebenen Format.

Tabelle 5.10 Neue Funktionen und Operatoren in XPath 3.1 (Forts.)

Funktion	Beschreibung
Neue Funktionen für die Behandlung von Knoten	
`fn:path`	Liefert einen Pfadausdruck, mit dem ein angegebener Knoten relativ zum Wurzelknoten des Dokuments ausgewählt werden kann.
`fn:has-children`	Prüft, ob der angegebene Knoten Kindknoten hat.
`fn:innermost`	Liefert jeden Knoten innerhalb der Eingabesequenz, der nicht ein Vorfahre eines anderen Mitglieds der Eingabesequenz ist.
`fn:outermost`	Liefert jeden Knoten innerhalb der Eingabesequenz, der kein Vorfahre eines anderen Mitglieds der Eingabesequenz ist.
Neue Funktionen für Sequenzen	
`fn:head`	Liefert das erste Element einer Sequenz.
`fn:tail`	Liefert alle Elemente einer Sequenz außer dem ersten.
Neue Funktionen zur Identifizierung von Knoten	
`fn:element-with-id`	Liefert die Sequenz der Elementknoten, die einen `ID`-Wert haben, der mit einem oder mehreren der `IDREF`-Werte übereinstimmt, die als Argument angegeben sind.
`fn:generate-id`	Die Funktion liefert eine Zeichenfolge, die einen gegebenen Knoten eindeutig identifiziert.
Neue Funktionen für den Zugriff auf externe Informationen	
`fn:uri-collection`	Liefert eine Sequenz von `xs:anyURI`-Werten, die die URIs in einer Kollektion von Ressourcen repräsentieren.
`fn:unparsed-text`	Die Funktion liest eine externe Ressource, wie etwa eine Datei, und liefert eine Zeichenfolge, die diese Daten repräsentiert.
`fn:unparsed-text-lines`	Die Funktion liest eine externe Ressource, wie etwa eine Datei, und liefert diese Daten zeilenweise als Zeichenfolgen.
`fn:unparsed-text-available`	Prüft beim Einsatz der Funktion `fn:unparsed-text`, ob tatsächlich ungeparste Texte vorhanden sind.

Tabelle 5.10 Neue Funktionen und Operatoren in XPath 3.1 (Forts.)

Funktion	Beschreibung
`fn:environment-variable`	Liefert den Wert einer Umgebungsvariablen des Systems, falls es solche Werte gibt.
`fn:available-environment-variables`	Liefert eine Liste der verfügbaren Umgebungsvariablen des Systems.
Neue Funktionen zum Parsen und Serialisieren	
`fn:parse-xml`	Die Funktion verwendet ein XML-Dokument in Form einer Zeichenfolge als Input und liefert den Dokumentknoten an der Wurzel des XDM-Baumes, der das geparste Dokument repräsentiert.
`fn:parse-xml-fragment`	Die Funktion verwendet eine externe Entität in Form einer Zeichenfolge als Input und liefert den Dokumentknoten an der Wurzel des XDM-Baumes, der das geparste Dokumentfragment repräsentiert.
`fn:serialize`	Die Funktion serialisiert die mit dem Argument angegebene Input-Sequenz und liefert eine serialisierte Repräsentation der Sequenz als Zeichenfolge.
Höherrangige Funktionen	
`fn:function-lookup`	Liefert die Funktion mit dem angegebenen Namen und der angegebenen Argumentanzahl.
`fn:function-name`	Liefert den Namen der Funktion, die über ein Funktion-Item identifiziert ist.
`fn:function-arity`	Liefert die Argumentanzahl der Funktion, die über ein Funktion-Item identifiziert ist.
`fn:for-each`	Wendet die angegebene Funktion auf jedes Element der angegebenen Sequenz an und verknüpft die Ergebnisse in der Reihenfolge der Sequenz.
`fn:filter`	Liefert die Elemente der Sequenz, für die die angegebene Funktion den Wert *wahr* liefert.
`fn:fold-left`	Verarbeitet die angegebene Sequenz von links nach rechts und wendet dabei wiederholt die angegebene Funktion an, um ein kumuliertes Ergebnis zu liefern.

Tabelle 5.10 Neue Funktionen und Operatoren in XPath 3.1 (Forts.)

Funktion	Beschreibung
`fn:fold-right`	Verarbeitet die angegebene Sequenz von rechts nach links und wendet dabei wiederholt die angegebene Funktion an, um ein kumuliertes Ergebnis zu liefern.
`fn:for-each-pair`	Wendet die angegebene Funktion paarweise jeweils auf ein Element der ersten und der zweiten Sequenz an und verknüpft die Ergebnisse in der Reihenfolge der Sequenzen.
Neue Funktionen in XPath 3.1 für Maps, Arrays, JSON etc.	
`fn:apply`	Wendet die angegebene Funktion dynamisch auf Argumente an, die als Array vorliegen.
`fn:collation-key`	Generiert einen Key für ein kollationsbasiertes Matching.
`fn:contains-token`	Prüft, ob ein Token in einer Liste vorhanden ist.
`fn:default-language`	Liefert die vorgegebene Sprache des Kontextes.
`fn:json-doc`	Lädt und parst ein JSON-Dokument und liefert Maps und Arrays.
`fn:json-to-xml`	Konvertiert JSON in XML.
`fn:load-xquery-module`	Lädt dynamisch ein XQuery-Modul und liefert den Zugriff auf globale Variablen und Funktionen.
`fn:parse-ietf-date`	Parst Datums- und Zeitdaten im IETF-Format.
`fn:parse-json`	Parst eine Zeichenkette im JSON-Format und liefert Maps und Arrays.
`fn:random-number-generator`	Generiert Zufallszahlen.
`fn:sort`	Sortiert eine Sequenz.
`fn:transform`	Führt eine XSLT-Transformation aus.
`fn:xml-to-json`	Konvertiert XML-Daten in das JSON-Format

Tabelle 5.10 Neue Funktionen und Operatoren in XPath 3.1 (Forts.)

Ganz praktisch ist auch ein neuer Operator für die Verknüpfung, den Sie anstelle der String-Funktion `fn:concat` einsetzen können. Statt

```
fn:concat("Olympia ", 2024)
```

schreiben Sie einfach

```
"Olympia " || 2024.
```

Neu ist auch der `map`-Operator in Form eines Ausrufezeichens. Er erlaubt ein einfaches Mapping und kann Ausdrücke vereinfachen, die bisher mit `for` gearbeitet haben. Der Ausdruck

```
//name ! ("Hallo " || .)
```

liefert beispielsweise für alle Elemente von `<name>` die Verknüpfung mit der Begrüßung.

5.4 Verknüpfungen mit XLink

Der weltweite Erfolg von HTML hat sehr viel damit zu tun, dass es nicht einfach nur Seiten in einem Browser anzeigen kann, sondern dass diese Seiten Verknüpfungen – Hyperlinks – enthalten, mit denen sich der Webbesucher ganz nach Belieben im Meer der Informationen bewegen kann.

5.4.1 Mehr als Anker in HTML

Die Architekten von XML hatten von Anfang an den Ehrgeiz, auch in puncto Links etwas bereitzustellen, was deutlich über die Möglichkeiten von HTML hinausgehen sollte. Die Unzufriedenheit mit dem erreichten Status bezog sich im Wesentlichen auf folgende Punkte:

- Es gibt in HTML nur zwei Elementtypen, die für die Herstellung von Links genutzt werden können: `<a>` und in einem eingeschränkten Sinne auch `<img>`.
- HTML-Links haben nur eine Richtung; sie führen immer von der Stelle, an der der Link im Quelldokument steht, zu dem Ziel, das der Link über eine URI-Referenz angibt. Sie sind also unidirektional.
- HTML-Links können nur ein Ziel ansteuern, also nicht mehrere gleichzeitig.
- Die Links müssen tatsächlich in dem Quelldokument eingefügt werden. Es ist also nicht möglich, Verknüpfungen einzurichten, die von einem Dokument ausgehen, das nicht bearbeitet werden darf, weil nur Lesezugriff eingeräumt ist.

Beeinflusst von Hypermediasystemen wie *HyTime – www.hytime.org* – und den Erfahrungen der *Text Encoding Initiative – www.tei-c.org –*, wurde das ehrgeizige Projekt auf den Weg gebracht, eine Sprache zu entwickeln, die sich von diesen Begrenzungen löst. Ziel war es, ein Vokabular für die Beschreibung von Links zu entwickeln, das im Wesentlichen folgende Erweiterungen ermöglichen sollte: