10 XML Verarbeitung

10.1 XML Namespaces

"Modulsystem für XML"

Vermeidung von Namenskonflikten bei Benutzung von

- mehreren Quellen für Elementdeklarationen oder
- mehreren Anwendungen mit dem gleichen Dokument

Definition: An XML namespace is a collection of names—identified by a URI reference RFC 2396—which are used in XML documents as element types and attribute names.

- Namen von Elementen und Attributen dürfen *qualifiziert* werden
- Qualifizierte Namen = $\langle \mathit{Namespace\text{-}Präfix} \rangle : \langle \mathit{lokaler\ Teil} \rangle$
- Namespace-Präfixe: *Abkürzungen* für absolute URIs
- Falls nspre steht für uri, dann wird
  $\mathit{nspre} : \mathit{lokalerName} \Rightarrow (\mathit{uri}, \mathit{lokalerName})$
- Identität von Element- und Attributnamen:
  Gleichheit dieser Paare
- die URI ist beliebig, muss nicht funktional sein

Deklaration eines Namespace

- Pseudoattribute für Namespace Deklarationen
  - `xmlns` (definiert Default-Namespace) und
  - `xmlns:`⟨*Namespace-Präfix*⟩ (bindet Namespace-Präfix)
- Deklaration ist jeweils gültig für das Eltern-Element des Pseudoattributs und sämtliche geschachtelten Elemente
- die Pseudoattribute können mehrfach auftreten, innere Vorkommen überdecken die äußeren

- Pseudoattribut `xmlns="`*`Default-Namespace`*`"`
  - unqualifizierte Elemente im Gültigkeitsbereich erhalten den Default-Namespace
  - Attribute müssen qualifiziert werden
  - *`Default-Namespace`* darf leer sein
- Pseudoattribut `xmlns:`$\langle$*`Namespace-Präfix`*$\rangle$`="`$\langle$*`URI`*$\rangle$`"`
  - bindet den Namespace $\langle$*`URI`*$\rangle$ an das $\langle$*`Namespace-Präfix`*$\rangle$
  - mehrere Präfixe für gleiche $\langle$*`URI`*$\rangle$ möglich
- Unqualifizierte Elemente außerhalb einer Default-Namespace Deklaration liegen im *leeren Namespace*, ebenso unqualifizierte Attribute

Beispiele

```
<x xmlns:edi='http://ecommerce.org/schema'>
   <!-- the "edi" prefix is bound to http://ecommerce.org/schema
        for the "x" element and contents -->
</x>
```

```
  <bk:BOOK xmlns:bk="urn:BookLovers.org:BookInfo"
           xmlns:money="urn:Finance:Money">
    <bk:TITLE>A Suitable Boy</bk:TITLE>
    <bk:PRICE money:currency="US Dollar">22.95</bk:PRICE>
  </bk:BOOK>
```

Beispiel: Äquivalentes Element mit Default Namespace

```
  <BOOK xmlns="urn:BookLovers.org:BookInfo"
        xmlns:money="urn:Finance:Money">
    <TITLE>A Suitable Boy</TITLE>
    <PRICE money:currency="US Dollar">22.95</PRICE>
  </BOOK>
```

10.2 XML Path Language (XPath)

Ziel von XPath: Spezifikation von Folgen von Knoten in XML-Dokument

- Verwendung in URIs und Attributen
- kompakte Syntax (nicht XML)
- operiert auf logischer Struktur des Dokuments
- einfache Berechnungen auf Ergebnismengen
- weitere Verwendung in anderen XML-Standards: XSLT, XQuery, XPointer, XLink, XForms, ...
- XPath 1.0 $\subseteq$ XPath 2.0

XPath-Sicht von XML

XML-Dokument ist ein Baum mit folgenden Knotenarten

- Wurzel (Dokumentenknoten)
- Element
- Attribut
- Text
- Namespace
- Verarbeitungsanweisungen <?*name daten*?>
- Kommentare

10.2.1 Pfadausdrücke

Ein Pfadausdruck ...

- Folge von Schritten $\langle Step \rangle$ getrennt durch /.
- Absoluter Pfadausdruck beginnt mit /.
- definiert Folge von XML-Knoten.

$$
\begin{array}{lcl}
\langle LocationPath \rangle & ::= & \langle RelativeLocationPath \rangle \\
& | & \langle AbsoluteLocationPath \rangle \\
\langle AbsoluteLocationPath \rangle & ::= & /\langle RelativeLocationPath \rangle ? \\
& | & \langle AbbreviatedAbsoluteLocationPath \rangle \\
\langle RelativeLocationPath \rangle & ::= & \langle Step \rangle \\
& | & \langle RelativeLocationPath \rangle / \langle Step \rangle \\
& | & \langle AbbreviatedRelativeLocationPath \rangle
\end{array}
$$

Location ⟨*Step*⟩ besteht aus

- Achse (traversierte Beziehung zwischen Knoten)
- Knotentest (Typ und Name)
- optionale Prädikate

$$
\begin{array}{lcl}
\langle \mathit{Step} \rangle & ::= & \langle \mathit{AxisSpecifier} \rangle \ \langle \mathit{NodeTest} \rangle \ \langle \mathit{Predicate} \rangle^* \\
& | & \langle \mathit{AbbreviatedStep} \rangle \\
\langle \mathit{AxisSpecifier} \rangle & ::= & \langle \mathit{AxisName} \rangle \ \texttt{::} \\
& | & \langle \mathit{AbbreviatedAxisSpecifier} \rangle
\end{array}
$$

Ergebnis eines Pfadausdrucks: Folge der Knoten (in Dokumentenordnung), die entlang der Achse erreicht werden, den Knotentest erfüllen und sämtliche Prädikate erfüllen

Beispiel: Rezepte

```
child::rcp:recipe[attribute::id='117'] /
child::rcp:ingredient /
attribute::amount
```

```
<rcp:recipe id='116'>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
  <rcp:ingredient amount='42'>
  </rcp:ingredient>
  ...
  <rcp:ingredient amount='4711'>
  </rcp:ingredient>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
</rcp:recipe>
```

Beispiel: Rezepte

```
child::rcp:recipe[attribute::id='117'] /
child::rcp:ingredient /
attribute::amount
```

```
<rcp:recipe id='116'>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
  <rcp:ingredient amount='42'>
  </rcp:ingredient>
  ...
  <rcp:ingredient amount='4711'>
  </rcp:ingredient>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
</rcp:recipe>
```

Beispiel: Rezepte

```
child::rcp:recipe[attribute::id='117'] /
child::rcp:ingredient /
attribute::amount
```

```
<rcp:recipe id='116'>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
  <rcp:ingredient amount='42'>
  </rcp:ingredient>
  ...
  <rcp:ingredient amount='4711'>
  </rcp:ingredient>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
</rcp:recipe>
```

Beispiel: Rezepte

```
child::rcp:recipe[attribute::id='117'] /
child::rcp:ingredient /
attribute::amount
```

```
<rcp:recipe id='116'>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
  <rcp:ingredient amount='42'>
  </rcp:ingredient>
  ...
  <rcp:ingredient amount='4711'>
  </rcp:ingredient>
  ...
</rcp:recipe>
<rcp:recipe id='117'>
  ...
</rcp:recipe>
```

Beispiel: Paragraphen

```
child::para[position()=1]
```

```
<para>
  ...
</para>
<para>
  ...
</para>
...
<para>
  ...
</para>
```

Beispiel: Paragraphen

```
child::para[position()=1]

<para>
  ...
</para>
<para>
  ...
</para>
...
<para>
  ...
</para>
```

10.2.2 Alle XPath-Achsen

⟨*AxisName*⟩ ::=

- `child` — `parent`
- `descendant` — `ancestor`
- `following-sibling` — `preceding-sibling`
- `following` — `preceding`
- `attribute`
- `namespace`
- `self`
- `descendant-or-self` — `ancestor-or-self`

10.2.3 Knotentests

⟨*NodeTest*⟩	::=	⟨*NameTest*⟩	
	|	⟨*NodeType*⟩`()`	
⟨*NameTest*⟩	::=	`*`	jeder Knoten
	|	⟨*NCName*⟩`:*`	beliebiges Element in Namespace
	|	⟨*QName*⟩	benanntes Element
⟨*NodeType*⟩	::=	`comment`	true, falls Kommentarknoten
	|	`text`	falls Textknoten
	|	`processing-instruction`	falls Verarbeitungsanweisung
	|	`node`	immer true

10.2.4 Prädikate

$$\langle Predicate \rangle \quad ::= \quad [\langle Expr \rangle]$$

Für jeden durch Achse und $\langle NodeTest \rangle$ selektierten Knoten wird Ausdruck $\langle Expr \rangle$ im Kontext ausgewertet. Ergebnis $\rightarrow$ Boolean. Falls Ergebnis `false`, wird der Knoten verworfen.

Weitere Ausdrücke Boolesche Operationen, arithmetische Operationen, Vergleichsoperationen, Stringoperationen (über Funktionsaufrufe), Operationen auf Knotenmengen

10.2.5 XPath-Kontexte

Die Auswertung aller Ausdrücke erfolgt in einem *Kontext* bestehend aus

- Kontextknoten
- Paar von natürlichen Zahlen (Kontextposition, Kontextgröße)
- Variablenbindungen
- Bibliotheksfunktionen
- Namespace-Deklarationen

Der initiale Kontext

- bestimmt durch Anwendung, die XPath startet
- bei absolutem Pfad: Dokumentenknoten des XML-Dokuments mit Kontextposition und Kontextgröße 1

Zwischen-Kontexte

- Ein Pfadausdruck wird $\langle \mathit{Step} \rangle$-weise ausgewertet beginnend mit aktuellem Kontext.
- Jeder Knoten der Ergebnisfolge Z wird Kontextknoten für den Rest des Ausdrucks
- Neue Kontextposition aus Position des Knotens in Z
- Neue Kontextgröße ist $|Z|$

10.2.6 Beispiele

- `child::para`
  selects the para element children of the context node
- `child::*`
  selects all element children of the context node
- `child::text()`
  selects all text node children of the context node
- `child::node()`
  selects all the children of the context node, whatever their node type
- `attribute::name`
  selects the name attribute of the context node
- `attribute::*`
  selects all the attributes of the context node

- `self::para`
  selects the context node if it is a para element, and otherwise selects nothing
- `child::chapter/descendant::para`
  selects the para element descendants of the chapter element children of the context node
- `child::*/child::para`
  selects all para grandchildren of the context node
- `/`
  selects the document root (which is always the parent of the document element)
- `child::para[position()=1]`
  selects the first para child of the context node
- `/descendant::figure[position()=42]`
  selects the forty-second figure element in the document

10.2.7 Mehr über Prädikate

Der Ausdruck in einem Prädikat kann auch andere Typen annehmen:

- eine Zahl: zählt als `true`, falls = Kontextposition
- ein String: zählt als `true`, falls nicht-leer
- eine Folge: zählt als `true`, falls nicht-leer

Die üblichen arithmetischen und logischen Operatoren (`and`, `or`, `not()`), sowie Vergleichsoperatoren sind vorhanden.

Pfadausdruck als Prädikat

Testen der Umgebung, ohne dorthin zu gehen.

Rezepte mit Zutat sugar:

```
/descendant::rcp:recipe
    [descendant::rcp:ingredient[attribute::name='sugar']]
```

Zutaten, deren Name sugar ist:

```
/descendant::rcp:recipe/
    descendant::rcp:ingredient[attribute::name='sugar']
```

Prädikate sind sequentiell

Prädikate können i.a. nicht vertauscht werden, da manche Funktionen kontextabhängig sind. Beispiel: `position()`.

Die dritte Zutat:

```
/descendant::rcp:ingredient[position()=3][position()=1]
```

```
/descendant::rcp:ingredient[3][1]
```

Die leere Folge

```
/descendant::rcp:ingredient[position()=1][position()=3]
```

Prädikate sind sequentiell, II

d.h., zwei Prädikate können i.a. nicht durch ihre Konjunktion ersetzt werden.

Die dritte Zutat

```
/descendant::rcp:ingredient[position()=3][position()=1]
```

Die leere Folge

```
/descendant::rcp:ingredient[position()=3 and position()=1]
```

10.2.8 Abkürzungen

In Location ⟨*Step*⟩s gelten folgende Abkürzungen

`child::`		ist optional
`//`	für	`/descendant-or-self::node()/`
`@`	für	`attribute::`
`.`	für	`self::node()`
`..`	für	`parent::node()`

⟨*AbbreviatedAbsoluteLocationPath*⟩ ::= `//`⟨*RelativeLocationPath*⟩

⟨*AbbreviatedRelativeLocationPath*⟩ ::= ⟨*RelativeLocationPath*⟩`//`⟨*Step*⟩

⟨*AbbreviatedStep*⟩ ::= `.` | `..`

⟨*AbbreviatedAxisSpecifier*⟩ ::= `@`?

Beispiele mit Abkürzungen

`child::para`	`para`
`child::*`	`*`
`child::text()`	`text()`
`child::node()`	`node()`
`attribute::name`	`@name`
`attribute::*`	`@*`
`child::chapter/descendant-or-self::para`	`chapter//para`
`child::*/child::para`	`*/para`
`child::para[position()=1]`	`para[1]`

```
//rcp:nutrition[@calories=349]/../rcp:title/text()
```