

XML-Namensräume

Udo Kelter

31.03.2001

Zusammenfassung dieses Lehrmoduls

Sofern man XML-Daten, die aus verschiedenen Quellen stammen, mischt, können Namenskonflikte bei Elementtypen auftreten. Die Lösung dieses Problems besteht in Namensräumen. Dieses Lehrmodul stellt die entsprechenden Konzepte und Notationen vor.

obligatorisch: – Transportdateien und die XML

Stoffumfang in Vorlesungsdoppelstunden: 0.5

Inhaltsverzeichnis

1	Motivation	3
1.1	Namensräume in Programmen	3
1.2	Namenskonflikte in Transportdateien	4
1.3	Namensräume in XML	5
2	Bezeichner für Namensräume	5
3	Deklaration von Namensraumbezeichnern	6
4	Benutzung von Namensraumbezeichnern	7
5	Der vorgegebene Namensraum	9
6	Der namenlose Namensraum	10
	Literatur	11
	Index	11

1 Motivation

Bei den grundlegenden XML-Konzepten (s. [XML]) bilden u.a. die Namen von Elementen innerhalb einer Datei einen Namensraum, in dem jeder Name nur einmal deklariert werden kann. Wenn man nun mehrere XML-Dokumente oder einzelne Elemente, die aus verschiedenen Quellen stammen, zusammenmischen will, muß man die DTDs vereinigen, und es können Namenskollisionen auftreten. Dieses Problem ist mit den Konzepten, die in [XML00] enthalten sind, nicht lösbar, sondern ist Gegenstand einer ergänzenden Spezifikation [XMLNS99]. In dieser werden einige Grammatikregeln aus [XML00] erweitert¹.

1.1 Namensräume in Programmen

Das Konzept der Namensräume ist in allen üblichen Programmiersprachen vorhanden und daher wohlvertraut. Das ist einerseits hilfreich für ein erstes intuitives Vorverständnis. Andererseits unterscheidet sich der Begriff Namensraum in der XML in einigen Aspekten ganz wesentlich vom Begriff Namensraum in Programmiersprachen, was zu Mißverständnissen führen kann. Wir vergleichen daher zunächst die beiden Anwendungskontexte.

In üblichen Programmiersprachen sind Namensräume ein Konzept, das sich auf bestimmte Mengen von Deklarationen (z.B. die Menge aller Deklarationen von Variablen in einem Block oder die Menge der Attribute in einer Klasse) bezieht und das verbietet, daß in dieser Menge von Deklarationen der gleiche Bezeichner mehrfach deklariert wird. Ergänzt werden Namensräume durch Sichtbarkeitsbereiche und Sichtbarkeitsregeln, die bestimmen, wie Namenskollisionen bei überlappenden Sichtbarkeitsbereichen behandelt werden: eine Deklaration in einem inneren Block verdeckt i.d.R. eine Deklaration des gleichen Bezeichners in einem umgebenden Block².

¹Die Regeln in [XML00] haben eine eigene durchlaufende Numerierung. Die beiden Numerierungen sind nicht konsistent, d.h. für ein bestimmtes Nichtterminalzeichen können die zugehörigen Produktionen in den beiden Standards verschiedene Nummern haben. In diesem Text benötigen wir die Numerierungen nicht.

²Wobei in manchen Fällen die verdeckte Deklaration mit Hilfe einer qualifizie-

1.2 Namenskonflikte in Transportdateien

Transportdateien und speziell XML-Dateien bilden in einigen Punkten einen wesentlich anderen Kontext als Programme. Wenn Daten aus verschiedenen Quellen kommen und dabei Namenskollisionen auftreten, so kann i.a. nicht eine der Quellen generell bevorzugt werden (analog dazu, in Programmen die inneren Blöcke zu bevorzugen). Betrachten wir hierzu als Beispiel einen Auszug aus einem Vortragsverzeichnis:

```
<Kapitel>
  <Titel>Vorträge im Mai 2001</Titel>
  <Vortragstermin Datum="5.5.2001">
    <Vortragender Titel="Dr."
      Name="Klein" Vorname="Karl" />
    <Vortrag Dauer="60">
      <Titel>Einführung in XML</Titel>
    </Vortrag>
    <Videoaufzeichnung>
      <Titel>Vortrag von Herrn Klein am 5.5.2001</Titel>
      <Dauer Einheit="sek">2200</Dauer>
    </Videoaufzeichnung>
  </Vortragstermin>
</Kapitel>
```

In diesem Beispiel kommen Daten zusammen, die u.U. aus verschiedenen Beständen stammen, z.B. einer Personaldatenbank und einem Archiv von Videokassetten. Der Bezeichner `Titel` kommt wiederholt vor, dreimal als Name eines Elementtyps und einmal als Name eines Attributs. Die Verwendung als Attributname stellt in diesem Beispiel kein Problem dar, da die Menge der Attribute eines Elementtyps einen eigenen Namensraum bildet. Problematisch sind die Elementtypen. Die Menge aller Elementtypen in einer XML-Datei bildet einen einzigen Namensraum, eine unterschiedliche Definition von `Titel` im Kontext von `Vortrag` bzw. `Videoaufzeichnung` (mit anderen Attri-

renden Notation doch noch benutzt werden kann.

buten bzw. Komponentelementtypen) ist nicht möglich.

1.3 Namensräume in XML

Der grundlegende Ansatz in XML zur Bildung und Trennung von Namensräumen besteht darin,

1. Namensräume explizit zu benennen (in Programmen sind Namensräume “anonym”)
2. universell eindeutige Namen für Namensräume zu benutzen
3. die lokalen Typnamen innerhalb eines Namensraums universell eindeutig zu machen, indem man sie mit dem universell eindeutigen Namen des Namensraums kombiniert. Der Bezeichner für den Namensraum wird dem lokalen Namen vorangesetzt, beide Namen werden durch einen Doppelpunkt als Trennzeichen getrennt.

Die universell eindeutigen Typnamen ähneln der qualifizierten Angabe von Namen in Programmen, bis auf folgende Unterschiede:

- Auch in der Deklaration (!) müssen die qualifizierten Namen verwendet werden (eine implizite Zuordnung einer Deklaration zu einem Sichtbarkeitsbereich bzw. Namensraum wie in Programmen ist in der XML nicht möglich; die Elementtypdeklarationen bilden eine “flache” Menge)
- Letztlich gibt es deswegen bzgl. der Deklarationen keine echten Namenskonflikte mehr; die Verwendung unqualifizierter Namen ist möglich, stellt von der Systematik (nicht von der Häufigkeit) her aber eine Ausnahme dar.

2 Bezeichner für Namensräume

In der Denkwelt des WWW sind URIs (*Uniform Resource Identifiers* [RFC2396]) die naheliegendste Technik, Namensräume eindeutig zu identifizieren. Der URI könnte z.B. eine Datei angeben, die eine DTD enthält, die die Definitionen der Elemente des Namensraums enthält. URIs haben aber zwei Nachteile:

- Sie sind zu lang. URIs können ohne weiteres 30 - 50 Zeichen lang sein; da sie in einer Transportdatei oft auftreten und da Verbindungen im Internet vielfach langsam sind, stört schon der Platzbedarf. Da die Transportdateien teilweise auch von Hand ediert werden, stört ferner der Schreibaufwand und die resultierende Fehlerquelle.
- URIs können Zeichen enthalten, die in XML-Namen unzulässig sind (z.B. /), d.h. man müßte diesbezüglich die Syntax von Namen in recht komplizierter Weise erweitern.

Die Lösung des Problems besteht darin, statt eines vollen URIs in der Transportdatei nur einen (kompakten) Platzhalter zu verwenden. Diese Platzhalter nennen wir i.f. **Namensraumbezeichner**; sie sind als Bezeichner für Konstanten ganz analog dazu, wie man in der Programmierung Konstanten für komplizierte Werte deklariert (`int pi = 3.14....`), zu verstehen.

3 Deklaration von Namensraumbezeichnern

Der Sichtbarkeitsbereich eines Namensraumbezeichners ist an ein Element (incl. dessen innere Elemente) gekoppelt. Wenn dieses Element das Wurzelement ist, gilt der Bezeichner im gesamten Dokument. Deklariert wird der Bezeichner im öffnenden *tag* des Elements. Wenn das Element den Typ X hat, sieht die Deklaration eines Namensraumbezeichners folgendermaßen aus:

```
<X xmlns:meinNR='http://www.musterfirma.de/schema' ... >
  ...
  <!-- innerhalb dieses Elements ist der
        Bezeichner meinNR an die Zeichenkette
        'http://www.musterfirma.de/schema' gebunden -->
  ...
</X>
```

Syntaktisch sind diese Deklarationen leider ein bißchen kompliziert geraten. Nach den grundlegenden syntaktischen Strukturen für *tags* wird hier eigentlich einem Attribut namens `xmlns:meinNR` ein Wert

zugewiesen (obwohl die Definition des Elementtyps `X` gar kein Attribut dieses Namens enthält und obwohl der Name `xmlns:meinNR` nicht zulässig ist, denn er fängt mit `xml` an).

Eine “Wertzuweisung an ein Attribut” wird dann, wenn der Attributname mit `xmlns:` beginnt, nicht als solche interpretiert, sondern als Deklaration eines Bezeichners für einen Namensraum. Der Bezeichner ist der Rest des Attributnamens, wenn man vorne das `xmlns:` entfernt hat. Gleichzeitig wird diesem Bezeichner der in der Zuweisung angegebene Wert zugeordnet; dieser Wert wird auch als **Namensraumname** (*namespace name*) bezeichnet. In unserem vorstehenden Beispiel wird also der Namensraumbezeichner `meinNR` deklariert, und es wird ihm der Namensraumname `'http://www.musterfirma.de/schema'` zugeordnet.

Obwohl Namensraumnamen die Syntax von URIs haben müssen, werden sie nicht als solche interpretiert, d.h. es muß weder der in der URI angegebene Rechner noch die angegebene Datei auf dem Rechner existieren. An dieser Stelle sei daran erinnert, daß Transportdateien völlig unabhängig von Rechnernetzen benutzbar sein müssen.

4 Benutzung von Namensraumbezeichnern

Einem lokalen Typnamen, der zu einem bestimmten Namensraum gehören soll, wird der Namensraumbezeichner des Namensraums mit einem Doppelpunkt als Trennzeichen vorangesetzt; diese Form der Typnamen bezeichnen wir als **qualifizierte Typnamen**. Im folgenden Beispiel geschieht dies für die Elementtypen `E` und `X` und das Attribut `B`:

```
<meinNR:X
  xmlns:meinNR='http://www.musterfirma.de/schema' ... >
  ...
  <meinNR:E ... A='aaa'> .... </meinNR:E>
  ...
  <F ... meinNR:B='bbb'> .... </F>
</meinNR:X>
```

Das Beispiel `meinNR:X` zeigt, daß ein Namensraumbezeichner innerhalb des *tags*, in dem er deklariert wird, auch schon textuell vorher bei der Angabe des Elementtyps benutzt werden kann.

Innerhalb des Elements müssen immer dann, wenn man Typen aus dem Namensraum benennen will, *qualifizierte Typnamen verwendet werden* (auch in einer DTD!), man kann (zumindest mit den bisher vorgestellten Konzepten) nicht allein einen lokalen Namen verwenden³.

Es können auch mehrere Namensraumbezeichner gleichzeitig deklariert werden, etwa wie folgt:

```
<X xmlns:meinNR='http://www.musterfirma.de/schema'
    xmlns:htmlNR='http://www.w3.org/TR/REC-html40' .. >
    ...
</X>
```

In dem Element sind dann beide Namensraumbezeichner (zusätzlich zu den von umgebenden Elementen stammenden) verwendbar. Doppeldeutigkeiten können nicht entstehen, denn die für Namensräume in Programmen typische Problematik, einem nichtqualifizierten Typnamen einen global eindeutigen Typ zuzuordnen, existiert hier nicht.

Man kann für einen Namensraum mehrere Namensraumbezeichner deklarieren, indem man den gleichen Namensraumnamen angibt. Zwei Namensraumnamen werden nur dann als **identisch** angesehen, wenn sie buchstäblich gleich sind, wobei Groß-/Kleinschreibung unterschieden wird. Im folgenden Beispiel bezeichnen `meinNR:E` und `nocheinNR:E` den gleichen Elementtyp:

```
<X xmlns:meinNR='http://www.musterfirma.de/schema'
    xmlns:nocheinNR='http://www.musterfirma.de/schema'>
    ...
    <meinNR:E> .... </meinNR:E>
    <nocheinNR:E> .... </nocheinNR:E>
</X>
```

³Hier liegt ein wesentlicher Unterschied zur Wirkung von Namensräumen in Programmen.

5 Der vorgegebene Namensraum

Nach den bisherigen Konzepten ist es definitionstechnisch der Normalfall, qualifizierte Typnamen innerhalb des Dokuments und auch innerhalb der DTD zu verwenden. Die ständige Wiederholung der Namensraumbezeichner ist lästig. Falls weit überwiegend ein bestimmter oder sogar nur ein einziger Namensraum verwendet wird, würde man gerne auf die Angabe seines Namens verzichten; in diesem Fall können auch die Namensraumbezeichner entfallen. Hierzu kann ein **vorgegebener Namensraum** (*default namespace*) – oder, wenn man so will, ein vorgegebener Namensraumname – deklariert werden, und zwar nach folgendem Muster:

```
<X xmlns='http://www.musterfirma.de/schema' ... >
  ... <E>.....</E> ...
</X>
```

Hinter dem `xmlns` fehlen hier der Doppelpunkt und der Namensraumbezeichner. Der “Attributname” `xmlns` ist ein weiterer Sonderfall in der Syntax, der so interpretiert wird, daß hier ein vorgegebener Namensraum festgelegt wird.

Der Effekt dieser Festlegung besteht darin, daß bei allen *nichtqualifizierten* Typnamen innerhalb dieses Elements (im vorstehenden Beispiel: `E`) der vorgegebene Namensraum benutzt wird. Man kann sich den Effekt mit einem implizit vorhandenen Namensraumbezeichner, z.B. `VorgabeNRB`, veranschaulichen, der ansonsten nicht als Namensraumbezeichner benutzt wird. `VorgabeNRB` wird implizit in der Deklaration und bei den nichtqualifizierten Typnamen ergänzt. Unser vorstehendes Beispiel sähe dann folgendermaßen aus:

```
<VorgabeNRB:X
  xmlns:VorgabeNRB='http://www.musterfirma.de/schema'>
  ... <VorgabeNRB:E>.....</VorgabeNRB:E> ...
</VorgabeNRB:X>
```

6 Der namenlose Namensraum

Im nachhinein mag es jetzt erstaunlich erscheinen, daß wir in [XML] völlig ohne Namensräume auskommen konnten. In der Tat kann man dann, wenn man nur mit den Kernkonzepten von XML arbeitet, auf Namensräume völlig verzichten.

Ermöglicht wird dies durch die Festlegung, daß die Typnamen, die für die Kernkonzepte von XML benötigt werden, zu einem *implizit vorhandenen Namensraum*, der *keinen Namensraumnamen* hat, gehören. Für diesen Namensraum kann man keine Namensraumbezeichner deklarieren. Benutzen kann man die Typen aus diesem Namensraum nur, indem man nichtqualifizierte Typnamen (solche ohne Doppelpunkt) verwendet.

Sofern in einem Element ein vorgegebener Namensraumname deklariert ist, gilt dieser für alle nichtqualifizierten Typnamen; die Typen aus dem namenlosen Namensraum sind dann *nicht* mehr benutzbar. Hierzu ein Beispiel:

```
<root>
  <E1>
    <E2>.....</E2>
  </E1>

  <E1 xmlns='http://www.xxx.de'>
    <E2>.....</E2>
  </E1>
</root>
```

Die beiden Auftreten des nichtqualifizierten Typnamens E2 bezeichnen *verschiedene* Typen: der erste liegt im namenlosen Namensraum, der zweite im Namensraum mit Namen `http://www.xxx.de`.

Wenn man an einer Stelle, an der bereits ein vorgegebener Namensraumnamen wirksam ist, trotzdem einen Typ aus dem namenlosen Namensraum benutzen will, kann man den vorgegebenen Namensraumnamen durch die Pseudo-Attributzuweisung `xmlns=''` wie-

der abschalten, etwa wie folgt:

```
<root>
  <E1   xmlns='http://www.xxx.de'>
    <E2   xmlns=''>
      <E3>.....</E3>
    </E2>
    <E4>.....</E4>
  </E1>
</root>
```

Die Typnamen E2 und E3 gehören hier zum namenlosen Namensraum, während E4 zum Namensraum mit Namen `http://www.xxx.de` gehört.

Literatur

- [RFC2396] Berners-Lee, T.; Fielding, R.; Masinter, L. (eds.): IETF RFC 2396 Uniform Resource Identifiers (URI): Generic Syntax; IETF (Internet Engineering Task Force); 1998-08
- [XML00] Extensible Markup Language (XML) 1.0 (Second Edition); World Wide Web Consortium; <http://www.w3.org/TR/2000/REC-xml-20001006.html>; 2000-10-06
- [XMLNS99] Namespaces in XML; World Wide Web Consortium; <http://www.w3.org/TR/REC-xml-names>; 1999-01-14
- [XML] Kelter, U.: Lehrmodul "Transportdateien und die SGML"; 2001

Index

default namespace, 9

Namenskonflikt, 4, 5

Namensraum

in Programmiersprachen, 3, 8

in XML, 5

mehrere Bezeichner, 8

namenloser, 10

vorgegebener, 9

Namensraumbezeichner, 5, 6

Benutzung, 7

Deklaration, 6

Sichtbarkeitsbereich, 6

Namensraumname, 7

Sichtbarkeit, 3

Typname

lokaler, 7

nichtqualifizierter, 9, 10

qualifizierter, 5, 7, 8

URI, 5, 7

vorgegebener Namensraum, 9