XPath

Udo Kelter

09.05.2007

Zusammenfassung dieses Lehrmoduls

XPath ist eine Abfragesprache, mit deren Hilfe Knotenmengen in einem XML-Syntaxbaum bestimmt bzw. selektiert werden können. Eine Besonderheit im Vergleich zu relationalen Sprachen besteht darin, mit Pfadausdrücken in der Baumstruktur navigieren zu können. Dieses Lehrmodul stellt die wichtigsten Konzepte von XPath (Version 1.0) an Beispielen vor.

obligatorisch:
- Transportdateien und die XML
- XSLT

Stoffumfang in Vorlesungsdoppelstunden: 1.1

Inhaltsverzeichnis

©2007 Udo Kelter **Stand: 09.05.2007**
Dieser Text darf für nichtkommerzielle Nutzungen als Ganzes und unverändert in elektronischer oder gedruckter Form beliebig weitergegeben werden und in WWW-Seiten, CDs und Datenbanken aufgenommen werden. Jede andere Nutzung, insb. die Veränderung und Überführung in andere Formate, bedarf der expliziten Genehmigung. Die jeweils aktuellste Version ist über http://kltr.de erreichbar.

1 Versionen

1. XML Path Language (XPath), Version 1.0
   W3C Recommendation 16 November 1999
   `http://www.w3.org/TR/xpath`
2. XML Path Language (XPath) 2.0
   W3C Candidate Recommendation 8 June 2006
   `http://www.w3.org/TR/xpath20`
   `http://www.w3.org/TR/2006/CR-xpath20-20060608/`

Dieses Lehrmodul basiert auf Version 1.0!
Unterschiede zwischen 1.0 und 2.0 erst bei intesiver Nutzung sichtbar.

2 Motivation

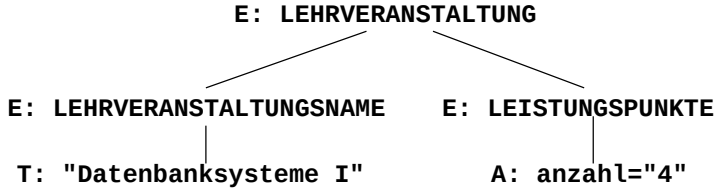
beim Verarbeiten eines Elements muß man ggf. selektiv auf innere Teile des Unterbaums zugreifen
Beispiel:

- s. HTML-Seite für Lehrveranstaltungsbeschreibung nach ECTS
- Beispieldaten:

```
<LEHRVERANSTALTUNG>
   <LEHRVERANSTALTUNGSNAME> Datenbanksysteme I
   </LEHRVERANSTALTUNGSNAME>

   <LEHRVERANSTALTUNGSKUERZEL>DBS1</LEHRVERANSTALTUNGSKUERZEL>
   <LEISTUNGSPUNKTE anzahl="4" />  ....
</LEHRVERANSTALTUNG>
```

Syntaxbaum (Auszug):



©2007 Udo Kelter

Beispiele für Selektionen von Knotenmengen:

1. `LEHRVERANSTALTUNGSNAME` ist direkter Kindknoten von `LEHRVERANSTALTUNG` →
   Pfad: `LEHRVERANSTALTUNGSNAME`

2. Attribut `anzahl` ist 2 Schritte entfernt → Notation ähnlich Pfadnamen in Dateisystemen:
   Pfad: `LEISTUNGSPUNKTE/@anzahl`

Verallgemeinerung: *ausgehend vom aktuell verarbeiteten Knoten beliebige andere Knoten lokalisieren* !
→ XPath

3 Lokalisierungspfade

Syntax (Nummern `[nn]` aus `www.w3.org/TR/xpath.html`):

```
[1] LocationPath   ::= RelativeLocationPath
                   | AbsoluteLocationPath
[2] AbsoluteLocationPath
                   ::= '/' RelativeLocationPath?
                   | AbbreviatedAbsoluteLocationPath
[3] RelativeLocationPath
                   ::= Step
                   | RelativeLocationPath '/' Step
                   | AbbreviatedRelativeLocationPath
```

Ausgangspunkt eines Lokalisierungspfads:
– bei relativen Pfaden: aktuell verarbeiteter Knoten (sog. "Kontextknoten")
– bei absoluten Pfaden: Dokumentwurzel

relativer Pfad: besteht aus mehreren (Navigations-) **Schritten**

Bedeutung eines Navigationsschritts: liefert zu *einem Ausgangsknoten* eine *Menge* von (Referenzen auf) Zielknoten

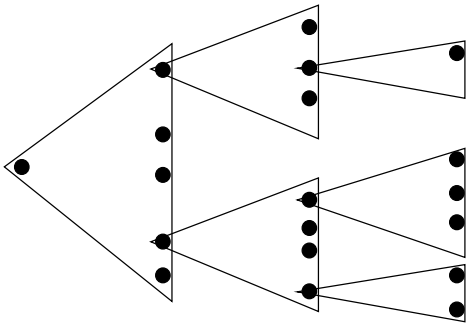
©2007 Udo Kelter

Bedeutung der Hintereinanderschaltung von Navigationsschritten (z.B. `schritt1/schritt2`):

- gegebene Ausgangsmenge: Menge der Zielknoten des vorherigen Schritts `schritt1`
- für *jeden einzelnen Knoten* `k` *der Ausgangsmenge*: bestimme Zielknoten gemäß nächstem Schritt `schritt2` ausgehend von `k`
- vereinige die Mengen der Zielknoten

Beispiel mit 3 Navigationsschritten:

`/schritt1/schritt2/schritt3`



Bedeutung ist analog zu *mengenwertiger Navigation in Dateisystemen*:

- Beispiel: `/home/propra/gruppe*/projekt1/*`
- Pfadnamen enthalten Navigationsschritte:
  - einzelne Dateinamen (Linknamen) [0 oder 1 Zielknoten]
  - oder Namensmuster [beliebig viele Zielknoten]
- Auswertung von links nach rechts
- Startmenge: Wurzel oder aktuelles Verzeichnis
- bei jedem Schritt:
  - für jedes Ausgangsverzeichnis Menge der Zielobjekte (Verzeichnisse oder Dateien) bestimmen,

©2007 Udo Kelter

– alle Zielobjektmengen vereinigen

Vorsicht: bei XPath sind viele Navigationsschritte, in denen kein `*` auftritt, trotzdem mengenwertig (entgegen der Erfahrung aus Dateisystemen)

bei relationaler Datenmodellierung: Navigationsschritte entsprechen Verbunden!

3.1 Navigationsschritte

einfachster Fall eines Navigationsschritts: Angabe eines Kindknotentyps oder eines Attributs
Bedeutung: ausgehend vom aktuellen Knoten, gehe zu allen Kindknoten mit dem angegebenen Typ
Beispielaufgabe:
ausgehend von einem LEHRVERANSTALTUNG-Element –
lokalisiere alle semester-Attribute in den DURCHFUEHRUNG-Elementen
Lösung: `'DURCHFUEHRUNG/@semester'`

3.2 Allgemeine Bestandteile eines Navigationsschritts

1. eine **Navigationsrichtung** (*axis*) in der Struktur des Syntaxbaums; Beispiel: direkte Kindknoten
   → ergibt ausgehend von einem Kontextknoten eine *Menge von (Referenzen auf) Zielknoten*
   = *Kandidaten* für Weiternavigation im nächsten Schritt oder (beim letzten Schritt) für das Endergebnis

2. ein **Knoten[typ]test**: Selektion der Kandidaten anhand der *Knotentypen*;
   Beispiel: Knoten muß vom Typ `DURCHFUEHRUNG` sein

3. **weitere Selektionsbedingungen**; sind Boolesche Ausdrücke; Details s.u.

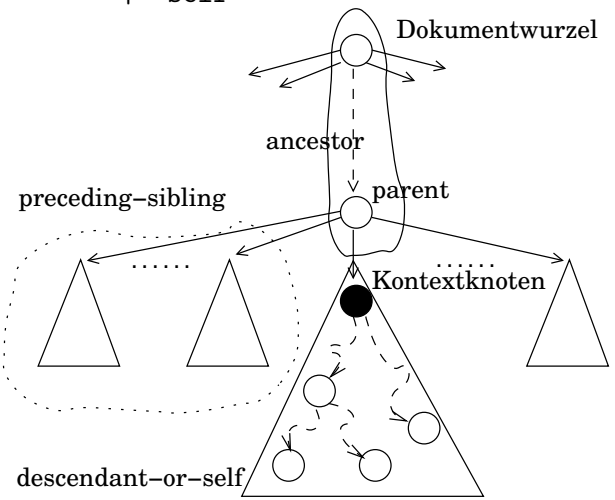
Syntax (Varianten: ausführliche und kompakte Notation):

```
[4] Step          ::= AxisSpecifier NodeTest Predicate*
                      | AbbreviatedStep
[5] AxisSpecifier ::= AxisName '::'
                      | AbbreviatedAxisSpecifier
```

3.3 Navigationsrichtungen ("Achsen")

Syntax:

```
[6] AxisName ::= 'ancestor'   | 'ancestor-or-self'
               | 'attribute'
               | 'child'
               | 'descendant' | 'descendant-or-self'
               | 'following'  | 'following-sibling'
               | 'namespace'
               | 'parent'
               | 'preceding'  | 'preceding-sibling'
               | 'self'
```



zugehörige Menge der Zielknoten ausgehend vom aktuell bearbeiteten Knoten (Kontextknoten):

`attribute` alle Attribute des Kontextknotens; = leere Menge, falls der Kontextknoten kein Element ist

`child` alle direkt enthaltenen Kindknoten außer den den Attributen!!

`descendant` die Nachfahren, d.h. alle Element-Kindknoten und deren Nachfahren-Knoten (keine Attribute)

`descendant-or-self` alle `descendant`-Knoten + Kontextknoten

`parent` der Elternknoten des Kontextknotens, sofern vorhanden (d.h. Kontextknoten ungleich Dokumentwurzel)

`ancestor` alle Vorfahren; enthält immer die Dokumentwurzel, sofern nicht leer

`ancestor-or-self` alle `ancestor`-Knoten zzgl. Kontextknoten

`following-sibling` nachfolgende Geschwister
leere Menge, falls der Kontextknoten ein Attribut-Knoten oder Namensraum-Knoten ist

`preceding-sibling` vorhergehende Geschwister
leere Menge, falls der Kontextknoten ein Attribut-Knoten oder Namensraum-Knoten ist

`following` alle "späteren" Knoten: alle Knoten, die in der Dokumentordnung (= bei einem preorder-Durchlauf) *nach* dem Kontextknoten kommen; ohne Attributknoten und ohne die Nachfahren

`preceding` alle "früheren" Knoten: alle Knoten, die in der Dokumentordnung *vor* dem Kontextknoten kommen; ohne Attributknoten und ohne die Nachfahren

`namespace` alle Namensraumknoten; leere Menge, falls der Kontextknoten kein Element ist

`self` der Kontextknoten

3.3.1 Abkürzungen für die Navigationsrichtung

`.` `self::node()`

`..` `parent::node()`

keine Angabe = `child::` (= Vorgabewert)
Beispiel: `LEHRVERANSTALTUNG/DURCHFUEHRUNG` =
`child::LEHRVERANSTALTUNG/child::DURCHFUEHRUNG`

`@` = `attribute::`
Beispielpfad: `LEISTUNGSPUNKTE/@anzahl`; darin:
`@anzahl` = `attribute::anzahl`

`//` = `/descendant-or-self::node()/` = alle Elementknoten im Teilbaum, dessen Wurzel der Kontextknoten ist
Beispiel: `.//@semester`

3.4 Knotentests

Details abhängig von der Navigationsrichtung; zu unterscheiden sind folgende Fälle:

1. Navigationsrichtung Attribute:
   alle Zielknoten sind **Attribut-Knoten**
2. Navigationsrichtung Namensräume:
   alle Zielknoten sind **Namensraum-Knoten**
3. restliche Fälle: als Zielknoten können **Element-Knoten** auftreten

Fettdruck: **dominierender Knotentyp** (*principal node type*)

3.4.1 Knotentests für Attribut-Knoten

Beispiele:

– einzelne Namen:
  selektiert Attribut mit diesem Namen, falls vorhanden
  andernfalls: ergibt leere Menge
  Beispiel: `DURCHFUEHRUNG/@semester`

– "`*`" für alle Attribute

3.4.2 Knotentests für Element-Knoten

`*` selektiert alle Elemente (nicht alle Knoten!)

`:einzelnerName:` selektiert Element mit Typ mit diesem Namen, falls vorhanden
Beispiel: `LEHRVERANSTALTUNG/DURCHFUEHRUNG`

`text()` selektiert alle Textknoten

`comment()` selektiert alle Kommentarknoten

`processing-instruction()` selektiert alle Verarbeitungsanweisungen

`node()` "selektiert" alle Knoten

Auszug aus der Syntax:

```
[7]  NodeTest ::= NameTest
                | NodeType '(' ')'
                | 'processing-instruction' '(' Literal ')'
[37] NameTest ::= '*'
                | NCName ':' '*'
                | QName
[38] NodeType ::= 'comment'
                | 'text'
                | 'processing-instruction'
                | 'node'
```

3.4.3 Weitere Selektionsbedingungen

Effekt: reduzieren die nach dem Knotentest vorhandene Knotenmenge; können sich beziehen auf:

- Werte von Attributen, Elementen, ...
- Struktur des Syntaxbaums, z.B. Kontextknoten ist erster von mehreren Geschwisterknoten

insg. komplex und deutlich unsystematischer als das relationale Modell
hier nur auszugsweise Darstellung, komplette Spezifikation s. `www.w3.org/TR/xpath`

Syntax - grobe Züge:

- einfache oder komplexe Bedingung in `[...]`

```
[4] Step  ::= AxisSpecifier NodeTest Predicate* | ...
[8] Predicate     ::= '[' PredicateExpr ']'
[9] PredicateExpr ::= Expr
```

- Boolesche Ausdrücke bestehend aus elementaren Booleschen Ausdrücken, die wie üblich mit Booleschen Operatoren verbunden werden
  Schlüsselworte: `or, and`
  vordefinierte Boolesche Funktion `not(...)`;
- komplizierte Syntax infolge diverser Abkürzungen (100* 2 Sekunden Tippzeit sparen und dafür 1* 1 Std. Fehlersuche....)

3.4.4 Einfache Boolesche Ausdrücke mit Wertvergleich

Vergleichsoperatoren für Texte bzw. numerische Werte: `=, !=, <, <=, <, >=, >`
(Vorsicht: das Zeichen `<` muß i.d.R. als `<` codiert werden!!)

Beispiel: gesucht sind die `DURCHFUEHRUNG`en einer `LEHRVERANSTALTUNG`, bei denen das Attribut `dozentId` den Wert "nn" hat. Lösung:

- Kontextknoten sei ein `LEHRVERANSTALTUNG`-Knoten
- der Pfad `child::DURCHFUEHRUNG` liefert Referenzen auf *alle* zugehörigen `DURCHFUEHRUNG`-Knoten;
  diese sind Kandidaten für das Endergebnis

– das Datum, anhand dessen wir einen Kandidaten selektieren, steht nicht im Kandidatenknoten selbst, sondern muß von dort aus *durch einen Pfad lokalisiert werden*;

  hier passender Pfad: `attribute::dozentId`

  oder Kurzschreibweise: `@dozentId`

– Lösung somit: `DURCHFUEHRUNG [ @dozentId = 'nn' ]`

Numerische Vergleiche und arithmetische Ausdrücke sind erlaubt, Beispiele:

`LEHRFORM [ @sws_umfang > 0 ]`

selektiert `LEHRFORM`-Elemente, bei denen das Attribut `sws_umfang` einen numerischen (!!) Wert > 0 hat.

`LEHRFORM [ @sws_umfang = 0.5 * .. / LEISTUNGSPUNKTE / @anzahl ]`

selektiert die `LEHRFORM`-Elemente, die zwei Leistungspunkte pro SWS haben

Besonderheit bei dieser Selektion: ein innerhalb und ein außerhalb des Elements liegendes Attribut werden verglichen und es wird ein numerischer Ausdruck benutzt

`LEHRFORM [ . = 'Vorlesung' ]`

selektiert `LEHRFORM`-Elemente, deren Inhalt (= enthaltene Textknoten inkl. Leerraum!) = 'Vorlesung' ist

3.4.5 Selektion anhand der Position

Kandidaten können auch anhand der Position im aktuellen Kontext selektiert werden, Beispiele:

`DURCHFUEHRUNG [ position() = 1 ]`

selektiert das 1. von mehreren `DURCHFUEHRUNG`-Elementen innerhalb eines umgebenden Elements

`DURCHFUEHRUNG [1]`
: dito, abgekürzte Schreibweise

`DURCHFUEHRUNG [ position() = last() ]`
: selektiert das letzte von mehreren `DURCHFUEHRUNG`-Elementen

Kontext = (vereinfacht) Geschwister des Kontextknotens, soweit der aktuellen Selektion entsprechend

Positionsfunktionen:

`last()` liefert Nummer des letzten Elements des Kontextes (= Anzahl der Knoten im Kontext)

`position()` liefert Positionsnummer des gerade behandelten Elements im Kontext

1. Element hat Nr. 1 (nicht 0).

Zählrichtung hängt ab von der Navigationsrichtung:

- *rückwärts*: andere Knoten liegen in Dokumentordnung vorher (Richtungen `ancestor`, `ancestor-or-self`, `preceding`, `preceding-sibling`)
- *vorwärts*: andere Knoten liegen in Dokumentordnung dahinter (alle anderen Richtungen; bei `self` ist die Unterscheidung gegenstandslos)

3.4.6 Existenz-Quantor

Selektion der Kandidaten danach, ob ein anderer Knoten existiert; Beispiele:

`LERNZIELE [ BEWERTUNGSKOMPETENZ ]`
: selektiert die `LERNZIELE`-Elemente, in denen wenigstens ein Element `BEWERTUNGSKOMPETENZ` enthalten ist

nur Elementtyp als Boolescher Ausdruck = impliziter Existenzquantor

`LEHRVERANSTALTUNG [ LERNZIELE/BEWERTUNGSKOMPETENZ ]`

selektiert die `LEHRVERANSTALTUNG`en, die wenigstens ein `LERNZIELE`-Element enthalten, in dem ein Element `BEWERTUNGSKOMPETENZ` enthalten ist

3.4.7 Zählfunktion

Funktion `count` liefert die Zahl der Knoten in der Knotenmenge, die als Argument angegeben wird

Beispiel: lokalisiere alle Lehrveranstaltungen, die mehr als zwei Mal durchgeführt wurden bzw. werden

```
// LEHRVERANSTALTUNG [ count (DURCHFUEHRUNG) > 2 ]
```

4 Vergleich mit relationalen Abfragesprachen

i.f. unterstellt: typische Modellierung im jeweiligen Datenbankmodell, Abbildung des Syntaxbaums in Form von Fremdschlüsseln von den Kindknoten auf die Elternknoten

Formulierung von Suchaufträgen mit XPath:

- typisch für XPath (und generell für navigierende Datenbankmodelle): Navigation
- deutlich umfangreicher wegen des komplexen Datenbankmodells
  jedes Konzept des Datenbankmodells führt zu einem Konzept (+ zug. Syntax) der Abfragesprache
- jeder Navigationsschritt = 1 Verbund
  ist bei "nativer" Speicherung effizienter berechenbar als ein Verbund in relationalen Systemen
  (aber auch nur dieser spezielle Verbund! hängt außerdem vom DVS ab)
- keine Projektionen, Aggregationen, Sortierungen, allgemeinen Verbunde usw. → weniger leistungsfähig

Formulierung von Suchaufträgen mit XPath und XSLT:

- Projektionen: OK
- Sortierungen: OK
- Verbunde: besser als mit XPath alleine, aber immer noch eingeschränkt; sehr umständlich
- Aggregationen: eingeschränkt; umständlich und fehlerträchtig

Softwaretechnische Beurteilung von XPath 1.0 und XSLT:

- XPath: kryptische Syntax, Buchstabenknauserei
- Gefahr der Vermischung von Layoutgenerierung und reinen Suchproblemen (Fachlogik)
- für "kleine" Anwendungen vertretbar,
  für komplexe Anwendungen softwaretechnisch fragwürdig (kein Modulkonzept, keine vernünftigen Abstraktionsmechanismen, kryptische Syntax, ...)

Literatur

[XPath] XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999; 1999; `http://www.w3.org/TR/xpath`

[XSLT] XSL Transformations (XSLT), Version 1.0, W3C Recommendation 16 November 1999; 1999; `http://www.w3.org/TR/xslt`

[XML] Kelter, U.: Lehrmodul "XML"; 2004

[XSLT05] Kelter, U.: Lehrmodul "XSLT"; 2004

©2007 Udo Kelter