

XML Vorlesung ETHZ, Sommersemester 2006

## XML Path Language (XPath)

Erik Wilde

25.4.2006

<http://dret.net/lectures/xml-ss06/>

25.4.2006

XML Vorlesung ETHZ SS 2006

1

## Übersicht

- Motivation
- Analogie File System ↔ XML Dokumente
  - beides sind Baummodelle, aber mit Unterschieden
  - Navigieren in Bäumen
- XPath Übersicht
  - XPath Location Paths
  - XPath Funktionen
- XPath Anwendungen
- Zusammenfassung

25.4.2006

XML Vorlesung ETHZ SS 2006

2

## Motivation XPath

- Entwicklung von XSLT
  - Transformationssprache für XML Dokumente
  - Kernpunkt ist Selektion von Teilen eines Dokuments
- XPath als Teil von XSLT
  1. zunächst integraler Bestandteil
  2. Potential für die Wiederverwendung wurde erkannt
    - Selektion von Dokumententeilen ist vielerorts praktisch
  3. XPath als eigener Standard, auf den XSLT verweist
- XPath ist somit allgemein verwendbar
  - grundlegende XML-Technologie

25.4.2006

XML Vorlesung ETHZ SS 2006

3

## Bäume: Filesystem vs. XML

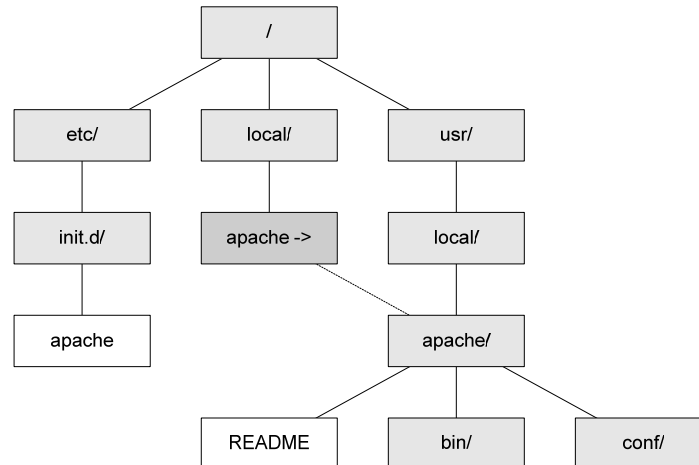
- Unterschiede zwischen der Navigation
    - Unix cd Kommando vs. XPath
  - 1. Knotentypen (XPath *Node Types*)
  - 2. Knotenanzahl (XPath *Node Sets*)
  - 3. Richtungen (XPath *Axes*)
  - 4. Filter (XPath *Predicates*)
- 
5. XPath Datenmodell (mehr als nur Knoten)
  6. XPath Funktionen

25.4.2006

XML Vorlesung ETHZ SS 2006

4

## Baumsicht eines Filesystems



25.4.2006

XML Vorlesung ETHZ SS 2006

5

## Eigenschaften eines Filesystems

- 3 verschiedene Arten von Knoten
  - Directories, Files, Symbolic Links
- durch das Filesystem garantierte Bedingungen
  - keine Directory-Einträge mit gleichem Namen
  - es gibt immer genau ein Root-Directory
  - Metainformationen werden im Directory gespeichert
    - Owner, Permissions, Creation Date, Modification Date
- verschiedene Sichten des Filesystems
  - *Raw Device vs. OS File System Support*
  - *OS File System vs. Network File System (NFS, SMB)*

25.4.2006

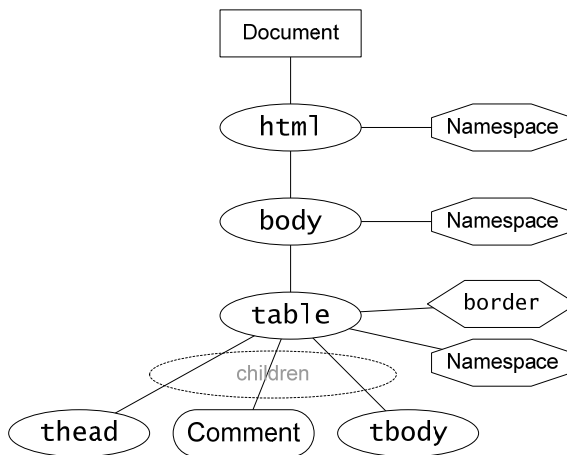
XML Vorlesung ETHZ SS 2006

6

## XML Dokument

```
<?xml version="1.0"?>  
<html xmlns="http://www.w3.org/1999/xhtml">  
  ...  
  <body>  
    <table border="0">  
      <thead> .... </thead>  
      <!-- Tabelleninhalt beginnt hier -->  
      <tbody> ... </tbody>  
    </table>  
  </body>  
</html>
```

## Baumsicht eines XML Dokuments



## 1. Knotentypen

- XPath definiert 7 Knotentypen
  - *Root Node* (repräsentiert das gesamte Dokument)
  - *Element* und *Attribute Nodes*
  - *Text Nodes* für Text innerhalb von Elementen
  - *Namespace Nodes* für gültige Namespaces
  - *Processing Instruction (PI)* und *Comment Nodes*
- Elemente haben nicht nur Kinder
  - Kinder sind Text, Elemente, PIs und Kommentare
  - Attribute und Namespaces sind keine Kinder
    - aber das Element ist das Elternteil dieser Knoten!

25.4.2006

XML Vorlesung ETHZ SS 2006

9

## 2. Knotenanzahl

- `cd` verlangt immer genau ein Directory
  - dieses wird danach zum *Current Working Directory*
- in vielen Fällen löst die Shell Wildcards auf
  - `ls -laF *.ppt`
  - das Kommando muss mehrere Parameter akzeptieren
- spezielle Kommandos erzeugen Listen
  - können per Pipe übermittelt werden
  - `find . -name README -ls | wc`
- Unix hat kein allgemeines Modell für *File Sets*

25.4.2006

XML Vorlesung ETHZ SS 2006

10

## Abarbeitung Filesystem Pfad

- jeder Schritt selektiert genau ein Directory
  - Wildcards im Pfad sind nicht erlaubt
  - Directorynamen sind eindeutig

cd /usr/local/apache/bin/  
1 → 1 → 1 → 1 → 1

## Abarbeitung XPath

- jeder Schritt selektiert eine Knotenmenge
  - der nächste Schritt operiert auf dieser Menge
  - pro Schritt kann die Menge wachsen oder schrumpfen

/html/body/table/thead/tr  
1 → 1 → 1 → 3 → 2 → 4

### 3. Richtungen

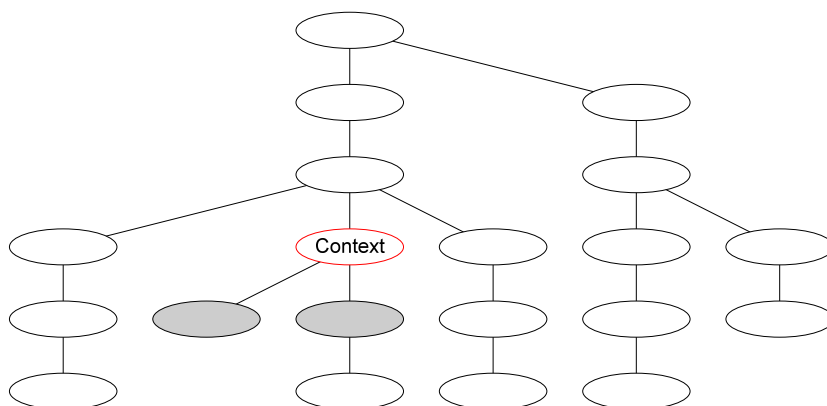
- absolute und relative Pfade
  - Pfad beginnt mit Slash oder mit einem Namen
- Unix Pfade gehen immer einen Schritt abwärts
  - implizite Semantik des Slashes im Pfad
  - . und .. sind keine Ausnahmen, sondern Hacks
    - auf spezielle Directories umgebogene Directory-Einträge
- XPath unterstützt verschiedene Richtungen
  - der Slash trennt die Schritte (keine implizite Richtung)
  - die Richtung wird durch die XPath Axis angegeben
    - wird sie weggelassen, so ist der Default child

25.4.2006

XML Vorlesung ETHZ SS 2006

13

### child Axis



25.4.2006

XML Vorlesung ETHZ SS 2006

14

### descendant Axis

The diagram shows a tree structure with a root node at the top. The root has two children. The left child has three children. The middle child of this level is highlighted in red and labeled 'Context'. Below it, several nodes are shaded gray, representing its descendants.

25.4.2006 XML Vorlesung ETHZ SS 2006 15

### parent Axis

The diagram shows a tree structure with a root node at the top. The root has two children. The left child has three children. The middle child of this level is highlighted in red and labeled 'Context'. Above it, the parent node is shaded gray, representing its parent.

25.4.2006 XML Vorlesung ETHZ SS 2006 16

### ancestor Axis

The diagram shows a tree structure with a central node circled in red and labeled "Context". This node has three children. The left child has three children of its own. The middle child of the "Context" node has two children. The right child of the "Context" node has two children. The node "Context" and all its ancestors (the three nodes above it) are shaded gray. All other nodes in the tree are white.

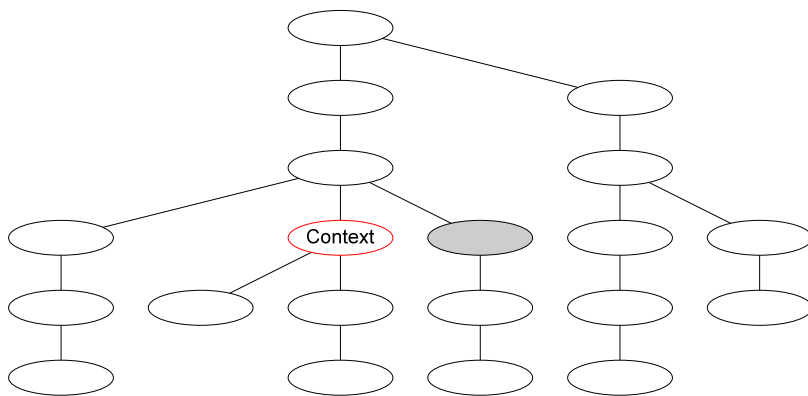
25.4.2006 XML Vorlesung ETHZ SS 2006 17

### preceding-sibling Axis

The diagram shows a tree structure with a central node circled in red and labeled "Context". This node has three children. The left child has three children of its own. The middle child of the "Context" node has two children. The right child of the "Context" node has two children. The node "Context" and all its preceding siblings (the left child and its three children) are shaded gray. All other nodes in the tree are white.

25.4.2006 XML Vorlesung ETHZ SS 2006 18

## following-sibling Axis

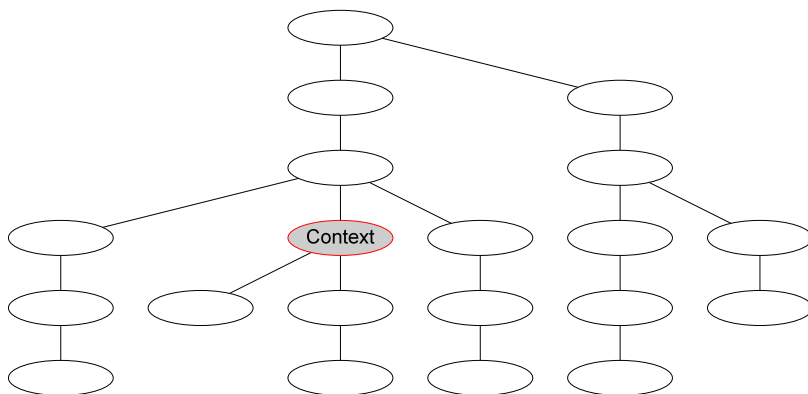


25.4.2006

XML Vorlesung ETHZ SS 2006

19

## self Axis



25.4.2006

XML Vorlesung ETHZ SS 2006

20

### ancestor-or-self Axis

The diagram shows a tree structure with a root node (grey) and two main branches. The left branch has three levels of nodes, with the middle node at the second level labeled 'Context' and highlighted with a red circle. The right branch has two levels of nodes. The 'Context' node is the starting point for the ancestor-or-self axis.

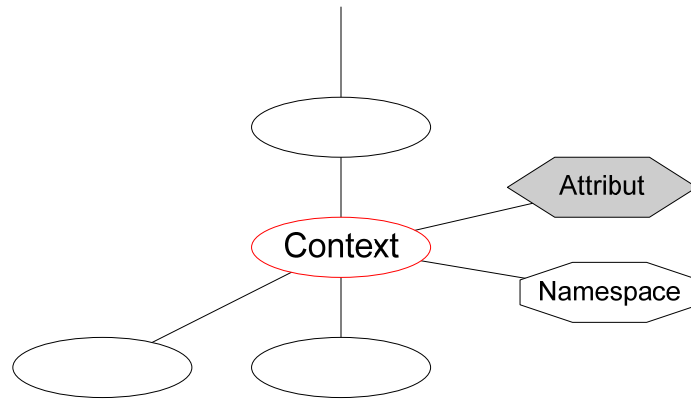
25.4.2006 XML Vorlesung ETHZ SS 2006 21

### descendant-or-self Axis

The diagram shows a tree structure with a root node (white) and two main branches. The left branch has three levels of nodes, with the middle node at the second level labeled 'Context' and highlighted with a red circle. The right branch has two levels of nodes. The 'Context' node is the starting point for the descendant-or-self axis.

25.4.2006 XML Vorlesung ETHZ SS 2006 22

## attribute Axis

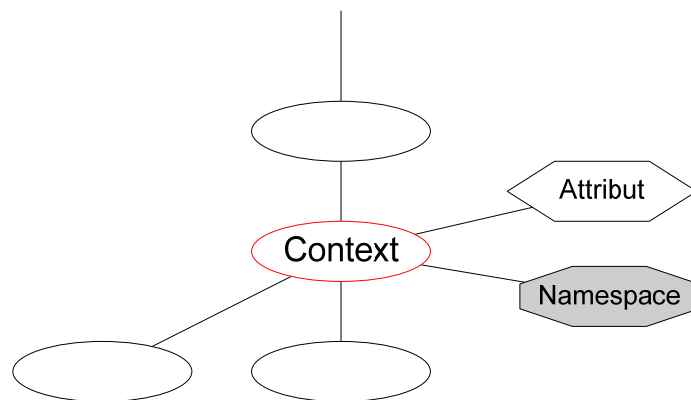


25.4.2006

XML Vorlesung ETHZ SS 2006

23

## namespace Axis



25.4.2006

XML Vorlesung ETHZ SS 2006

24

## XPath Location Path Syntax

- Folge von *Location Steps*, getrennt mit /
  - erste Komponente: *Axis Specifier* (gefolgt von ::)
  - zweite Komponente: *Node Test*
  - dritte Komponente: 0-n *Predicates* (in [ ])
- Abkürzungen in Location Paths
  - `child::` ist die *Default Axis*
  - `attribute::` kann abgekürzt werden als `@`
  - `//` ist kurz für `/descendant-or-self::node()/`
  - `.` ist kurz für `self::node()`
  - `..` ist kurz für `parent::node()`

25.4.2006

XML Vorlesung ETHZ SS 2006

25

## XPath Node Tests

- Namenstest (für Elemente und Attribute)
  - testet einen Node auf einen bestimmten Namen
    - `/html/head/table`
  - \* kann verwendet werden als Wildcard Character
- Test auf den Nodetype
  - markiert mit nachfolgenden Klammern
  - Test auf `text()`, `comment()` oder `node()`
- Test auf Processing Instruction
  - anwendbar auf spezifische Processing Instruction
    - `processing-instruction("myapplication")`

25.4.2006

XML Vorlesung ETHZ SS 2006

26

## 4. Filter (Prädikate)

- jeder Schritt kann zusätzliche Filter enthalten
  - Auswertung jedes Filters für jeden selektierten Node
    - daraus resultiert eine logisches *and* der Filter
  - Filter sind XPath Expressions
    - Auswertung relativ zum selektierten Node
    - Interpretation als boolean (ausser für number)
- Reduktion der Resultatmenge eines Schrittes
  - `/descendant::table[not(thead)]`
  - können in einigen Fällen vermieden werden
    - `/descendant::table[thead]`
    - `/descendant::table/thead/..`

25.4.2006

XML Vorlesung ETHZ SS 2006

27

## Filter in XML Bäumen

- Abarbeitung eines Location Paths
  1. Node Set ist die Basis jedes Schrittes
    - Context/Node: Node, Context Size, Context Position, ...
  2. Selektion der Nodes (Axis und Node Test)
  3. Auswertung aller Filter für alle selektierten Nodes
  4. die verbleibenden Nodes sind das neue Node Set
- Aufgabe: Finden Sie den 6. Link im HTML
  - Ansatz 1: `//a[position()=6] ≡ //a[6]`
    - Problem: `/descendant-or-self::node()/a[6]`
  - korrekter XPath: `/descendant::a[6]`

25.4.2006

XML Vorlesung ETHZ SS 2006

28

## XPath Expressions

- Location Paths sind ein Sonderfall von XPath
- allgemein ist ein XPath eine *Expression*
  - Auswertung gemäss definierter Grammatik
    - XPath  $(2+2)*10$  ergibt die Zahl 40
  - Expressions treten oft in Filtern auf
    - `//table[count(tbody/tr) > 10]`
- wichtig für die Auswertung von Expressions
  - der Context (durch Umgebung oder Location Path)
- 5. der Umgang mit Datentypen
- 6. die Verfügbarkeit von Funktionen

25.4.2006

XML Vorlesung ETHZ SS 2006

29

## 5. XPath Datenmodell

- XPath unterscheidet 4 Datentypen
  1. node-set (Mengen von Knoten)
  2. boolean (true oder false)
  3. number (Floating Point Numbers)
  4. string (Unicode Character strings)
- Beispiele für XPaths dieser Datentypen
  1. `//table[thead]`: alle Tabellen mit einem thead
  2. `1=2`: ergibt den Boolean Wert `false`
  3. `27 div 16`: ergibt 1.6875
  4. `substring('abcd', 2, 2)`: ergibt den String `bc`

25.4.2006

XML Vorlesung ETHZ SS 2006

30

## String-value eines Nodes

- wichtiges Konzept für XPath Resultate
  - jeder Node Type hat einen String-value
  - lokal oder rekursiv definiert
- Definition der String-values
  - *Root*: Konkatenation aller Text Node Descendants
  - *Element*: Konkatenation aller Text Node Descendants
  - *Attribute*: normalisierter Attributwert
  - *Namespace*: Namespace URI
  - *Processing Instruction*: Inhalt (nicht das Target!)
  - *Comment* und *Text*: Inhalt

25.4.2006

XML Vorlesung ETHZ SS 2006

31

## XPath Umgang mit Datentypen

- XPath definiert viele implizite Konvertierungen
  - keine gute Grundlage für eine typensichere Sprache
  - Umformungsregel müssen bekannt sein
    - es gibt explizite Funktionen, die diese Regeln definieren
    - diese werden u.U. implizit angewandt
- viele XPath sind "korrekt"
  - häufiger Effekt beim Lernen von XPath
    1. Typ-Probleme (oder anderes...) im XPath
    2. Resultat ist ein leeres Node Set
    3. kann als Boolean, String oder Number interpretiert werden

25.4.2006

XML Vorlesung ETHZ SS 2006

32

## 6. XPath Funktionen

- Funktionen mit dem Resultattyp `boolean`
  - `boolean`, `contains`, `false`, `lang`, `not`, `starts-with`, `true`
- Funktionen mit dem Resultattyp `number`
  - `ceiling`, `count`, `floor`, `last`, `number`, `position`, `round`, `string-length`, `sum`
- Funktionen mit dem Resultattyp `string`
  - `concat`, `local-name`, `name`, `namespace-uri`, `normalize-space`, `string`, `substring`, `substring-after`, `substring-before`, `translate`
- Funktionen mit dem Resultattyp `node-set`
  - `id`

25.4.2006

XML Vorlesung ETHZ SS 2006

33

## Location Paths und Funktionen

- Location Paths können Funktionen benutzen
  - kommen dann immer in Prädikaten vor
  - `//a[substring(@href, string-length(@href)-2)='pdf']`
- Funktionen können Location Paths benutzen
  - falls die Funktion Node Sets akzeptiert
    - `count(//a)` zählt die Anzahl der Links auf einer Seite
    - andernfalls findet eine implizite Konvertierung statt
- dieses Prinzip kann geschachtelt werden
  - die Übersichtlichkeit kann leiden
    - `//*[ @* ]` oder `//*[ count(@*) = count(*) ]`
  - sehr komplexe Aufgabenstellungen lösbar

25.4.2006

XML Vorlesung ETHZ SS 2006

34

## XPath ist keine Query-Sprache

- Query-Sprachen kennen zwei Aspekte:
  1. *Adressierung* von gesuchten Informationen
  2. *Rekombination* der gesuchten Informationen
- XPath behandelt nur den ersten Aspekt!
- XQuery als neue Query-Sprache für XML
  - wird auf XPath aufbauen
  - verwendet eine erweiterte Version (XPath 2.0)
- XPath als Grundlage für viele Standards
  - XSLT, XML Schema, XPointer, XQuery
  - aber auch als Grundlage für Tools (z.B. Editoren)

25.4.2006

XML Vorlesung ETHZ SS 2006

35

## XPath in XSLT

- XSLT verwendet XPath an vielen Stellen
  - *Match Patterns* für Templates
  - Konditionen in Bedingungen
  - Ausführen von Schleifen (Iterationen über Node Sets)
  - Ausgaben von Werten
- XPath ist die *Expression Language* von XSLT
- XSLT ist ohne XPath nicht denkbar
  - oftmals Design-Frage beim XSLT-Schreiben
    - komplexere XPaths oder mehr XSLT Code?
  - XPath wichtigste Grundlage für Umgang mit XSLT

25.4.2006

XML Vorlesung ETHZ SS 2006

36

## XPath in XML Schema

- XML Schema als DTD++
  - viele neue Konzepte
- Verallgemeinerung des ID/IDREF Konstrukts
  - Identity Constraints
  - Uniqueness, Keys, Key References
- Selektion von Knoten für Constraints
  - Selektion der Knoten des Constraints
    - z.B. alle person Elemente eines Dokuments
  - Selektion der auszuwertenden Felder
    - z.B. müssen name & vorname eindeutig sein

25.4.2006

XML Vorlesung ETHZ SS 2006

37

## XPath in XPointer

- XPointer definiert XML Fragment Identifier
  - `http://...xml#xpointer(id(x99))`
- erweitert XPath sehr massiv
  - führt neue Datentypen ein
  - führt neue Funktionen ein
- Unterstützung von Ranges
  - Idee der Selektion wie mit einer Maus
    - Anfangspunkt und Endpunkt
    - dafür braucht man ein Konzept für Punkte...
    - ...und die Bereiche zwischen ihnen

25.4.2006

XML Vorlesung ETHZ SS 2006

38

## Ausblick auf XPath 2.0

- vollständig neues Datenmodell
  - Sequenzen als Grundlage
    - Sequenzen sind geordnet (und erlauben Duplikate)
    - Sequenzen können verschiedene Datentypen enthalten
    - über Sequenzen kann iteriert werden
  - Unterstützung der XML Schema Datentypen
- konditionale Ausdrücke
  - `if-then-else` innerhalb des XPath
- verbesserte Mengenoperationen
  - XPath 1.0 kennt nur die Vereinigung von Mengen

25.4.2006

XML Vorlesung ETHZ SS 2006

39

## XPath in XQuery

- XQuery wird die Query-Sprache für XML
  - XPath 2.0 ist ein grosser Teil der Sprache
  - XQuery hat zusätzliche Konstruktoren

```
<bib>
  { for $b in doc("http://www.bn.com/bib.xml")/bib/book
    where $b/publisher = "Addison-wesley" and $b/@year > 1991
    return
      <book year="{ $b/@year }">
        { $b/title }
      </book>
    }
</bib>
```

25.4.2006

XML Vorlesung ETHZ SS 2006

40

## Zusammenfassung

- XPath als Selektion von XML Teilen
  - strukturell orientiert an XML
    - basiert (indirekt...) auf dem Infoset
  - spartanischer Support für Text-Handling
    - nicht das Ziel von XPath
- XPath als Grundlage für XML Standards
- XPath als Tool
  - Unterstützung in Editoren als *Find* Syntax