

# Intelligente Systeme im WWW: Semantic Web

## XML – Extensible Markup Language

Dr. Raphael Volz  
volz@fzi.de

Forschungszentrum Informatik - FZI



Content licenced under Creative Commons  
<http://creativecommons.org/licenses/by-nc-sa/2.0/de/>

# XML Anfrage und Transformation

## Agenda

- XPath  
*Adressierung von Elementen*
- XQuery  
*Anfrage von Elementen*
- XSL(T)  
*Transformation von Elementen*

# XPath

- XPath dient zur Adressierung von Teilen in einem XML Dokument
- Nicht-XML Syntax  
Nutzbarmachung in URIs und XML Attributen
- Operiert auf der logischen Struktur eines XML Dokumentes [ > <http://www.w3.org/TR/query-datamodel/> ]
- Wird in mehreren „Host languages“ genutzt, z.B: XSLT , Xquery, Xpointer

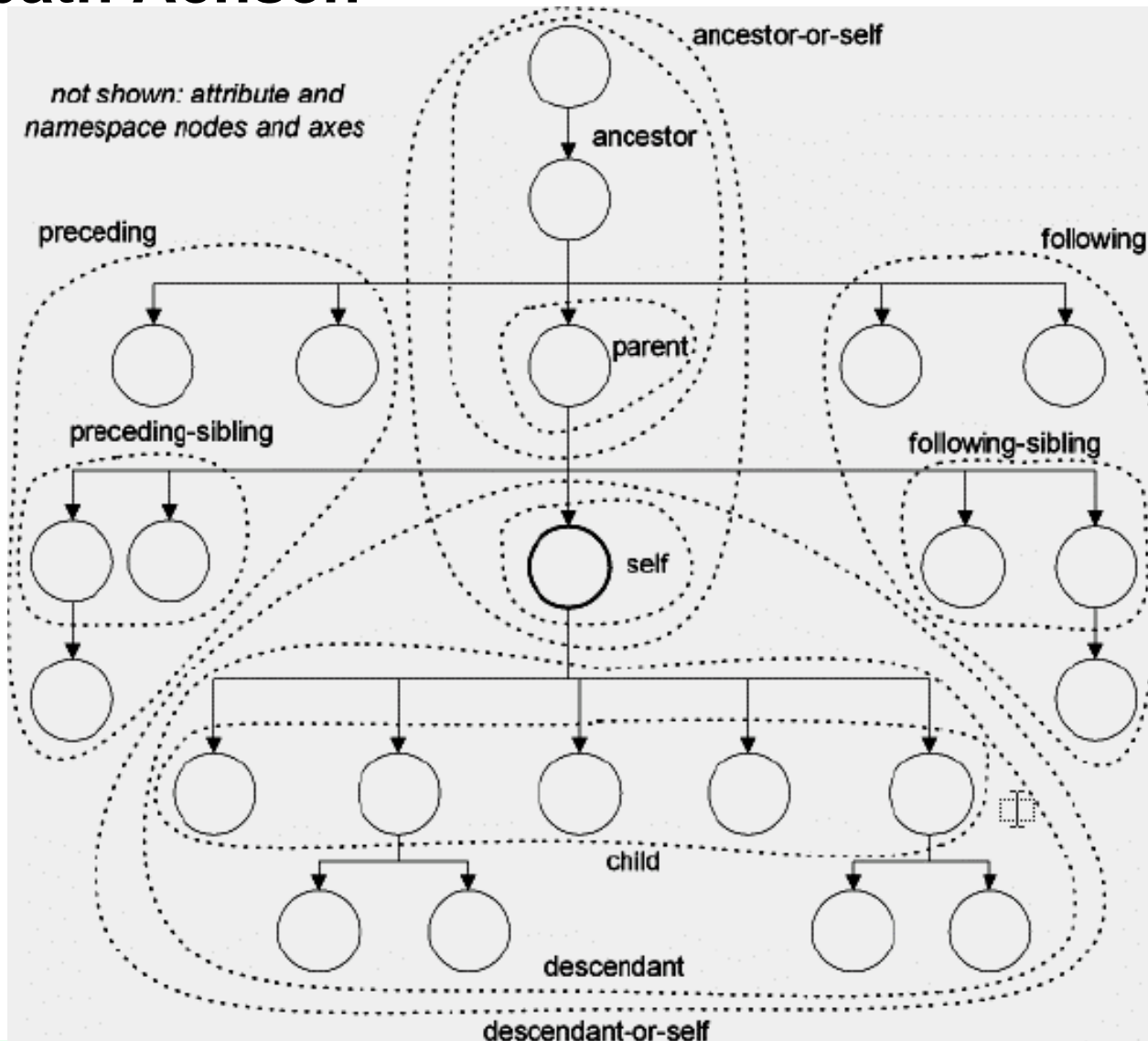
# XPath II

- Teilmenge kann für Mustererkennung genutzt werden
- Hat Formale Semantik  
[ > <http://www.w3.org/TR/query-semantics/> ]
- Gibt immer (flache) Sequenzen von XML Knoten zurück
- Sequenzen können Duplikate enthalten
- Umsortierung Sequenz möglich  
Voreinstellung: Reihenfolge in Dokument

# Adressierung in XPath

- Jedes Xpath Muster ist ein 'location path'
  - absoluter l.p. : Fängt mit ("/") an
  - relativer l.p. : sonst.
- Jeder l.p. besteht aus einer Serie von Schritten
- Schritte sind getrennt durch „/“
- Schritte haben 3 Teile:
  - Angabe der Achse  
*Beziehung zwischen dem **lokalen** Knoten und den **auszuwählenden** Knoten im XML Baum.*
  - Knoten Test  
*Typ des auszuwählenden Knoten*
  - Prädikate  
*Weitere Selektion der auszuwählenden Knoten*

# Xpath Achsen



# XPath Knotentests

K.T. testen XML Knoten (Elemente und Attribute)

- name  
*Adressiert alle <name> Elemente*
- \*  
*Adressiert beliebige Elemente*
- namespace:name  
*Adressiert <name> Elemente im entsprechenden Namespace*
- @name  
*Adressiert Attribut „name“ eines beliebigen Elementes*

# Xpath Prädikate

Prädikate befinden sich immer in []

- `kt[1]` :  
*Adressiere ersten Knoten nach Knotentest*
- `kt[position()=last()]` :  
*Adressiere letzten Knoten nach Knotentest*
- `kt[@id='foo']`  
*Adressiere jene Elemente, die ein ID Attribut mit Wert „foo“ haben.*
- `element[not(@id)]`  
*Selektiere jene Elemente, die kein ID Att. haben*
- `author[firstname="Norman"]`  
*Nimm alle <author> Elemente, die ein firstname Subelement haben, dessen Text Norman ist*

## Xpath Beispiele (I)

- `para`  
*Nimm alle <para> Kinder im lokalen Kontext*
- `para/emphasis`  
*Alle <emphasis> Elemente mit direktem Obererelement <para>*
- `/`  
*Wurzel des Dokumentes*
- `para//emphasis`  
*Alle <emphasis> Elemente mit Obererelement <para>*
- `section/para[1]`  
*Erstes <para> Unterelement jener <section> Elemente im lokalen Kontext*

## Xpath Beispiele (II)

- `//title`
- `section/*/note`
- `stockquote[@symbol]`
- `stockquote[@symbol="XXXX"]`
- `emphasis|strong`  
*Addressiere <emphasis> oder <strong> Elemente*

# Operationen auf Sequenzen

- **For**

*Iteriert über Sequenzen, gibt neuen Wert für alle Elemente der Argumentsequenzen zurück:*

```
for $x in /order/item
return $x/price * $x/quantity
```

- **Bedingte Anweisungen**

*Wenn ... Dann:*

```
if ($part/@discounted)
  then $part/wholesale
  else $part/retail
```

# Operationen auf Sequenzen II

- **Quantifizierung**
  - (=) erlaubt Vergleich von Knotenmengen.
  - *Existenzquantifizierung*  
some \$x in /students/student/name satisfies \$x = "Fred"
  - *Allquantifizierung*  
every \$x in /students/student/name satisfies \$x = "Fred",
- **Mengenoperatoren**
  - Vereinigung: | bedeutet „oder“
  - Schnittmenge:  
\$x intersect /foo/bar
- **Except**  
*Erlaubt Auswahl bestimmter Knoten, ausser bestimmter Knoten*  
@\* except @exc:foo

# Weiteres zu XPath

- Xpath/Xquery Datenmodell  
[ > <http://www.w3.org/TR/query-datamodel/> ]  
*Beschreibt welche Informationen eines Dokuments durch Xpath verarbeitet werden können*
- Funktionen und Operatoren in Xpath 2.0  
[ > <http://www.w3.org/TR/query-operators/> ]  
*Beschreibt welche Funktionen und Operatoren es in Xpath gibt*
- Einführung in XPath 2.0  
[ > <http://www.xml.com/pub/a/2002/03/20/xpath2.html> ]
- Diverse Tutorials  
[ > Google : XPATH Tutorial ]
- Hier ausgelassen:
  - Sortierung von Knoten
  - Datentyp Operationen (z.B: Type Casting)

# XML Anfrage und Transformation

## Agenda

- XPath  
*Adressierung von Elementen*
- XQuery  
*Anfrage von Elementen*
- XSL(T)  
*Transformation von Elementen*

# XQuery

- Umfasst frühere Ansätze:  
XML-QL, XQL, SQL, OQL
- Entwurfsansatz getrieben durch „Use-cases“
- Queries können Daten aus mehreren Quellen umfassen
- Minimalismus , viele Convenience-Operationen zurückführbar auf Kernsprache
- Klare Semantik, klein, einfach zu verstehen
- Vorgänger: Quilt

# Xquery II

- XQuery ist eine funktionale Querysprache
- Jede Query ist ein Ausdruck
- Ausdrücke können flexibel kombiniert werden
- Innerhalb einer Query können definiert werden:
  - Namespace Bindungen (optional)
  - Definition von Benutzerfunktionen (optional)
  - Query selbst

# Xquery Grundelemente

- Xpath Ausdrücke: `/a//b[c = 5]`
- FLWR Ausdrücke:  
`FOR ... LET ... WHERE ... RETURN`
- Element Konstruktion: `<a> ... </a>`
- Variablen und Konstanten: `$x, 5`
- Operatoren und Funktionsaufrufe:  
`x + y, -z, foo(x, y)`
- Sortierungen:  
`expr SORTBY (expr ASCENDING , ... )`
- Entwurf für Updatesprache:  
`INSERT, REPLACE, DELETE`

# Element Konstruktoren

```
construct
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author>
      <last>Stevens</last>
      <first>W.</first>
    </author>
    <publisher>Addison-Wesley</publisher>
    <price> 65.95</price>
  </book>
</bib>
```

- Konstruktion von Elementen möglich
- Sehen so aus, wie das XML, welches konstruiert wird

# Erweitertes Xpath

- Alle XPath Ausdrücke nutzbar zur Selektion
- Erweiterungen:
  - Range expressions  
/bib/book/author[1 TO 2]
  - BEFORE und AFTER  
//book[ author[last="Stevens"] BEFORE  
author[last="Abiteboul"] ]
  - Namespace Unterstützung  
NAMESPACE rev = "www.reviews.com"  
//rev:rating
  - Dereferenzierung  
//publisher/web/@href->html

# FLWR - For, Let, Where, Return

- Ähnlich zur SELECT - FROM - WHERE Konstruktion in SQL

- Beispiel:

```
FOR $book IN document("bib.xml")//book
WHERE $book/publisher = "Addison-Wesley"
RETURN
    <book>
        {
            $book/title,
            $book/author
        }
    </book>
```

# FOR vs. LET

- FOR iteriert über eine Sequenz und bindet Variablen an jeden Knoten der Sequenz
- LET bindet Variablen an die gesamte Sequenz
- Beispiel:

```
FOR $book IN document("bib.xml")//book
LET $a := $book/author
WHERE contains($book/publisher, "Addison-Wesley")
RETURN
    <book>
        {
            $book/title,
            <count> Number of authors: { count($a) } </count>
        }
    </book>
```

# Inner Joins

```
FOR $book IN document("www.bib.com/bib.xml")//book,  
    $quote IN document("www.bookstore.com/quotes.xml")//listing  
WHERE $book/isbn = $quote/isbn  
RETURN  
    <book>  
        { $book/title }  
        { $quote/price }  
    </book>  
SORTBY (title)
```

# Outer Joins

```
FOR $book IN document("bib.xml")//book
RETURN
  <book>
    { $book/title }
    {
      FOR $review IN document("reviews.xml")//review
      WHERE $book/isbn = $review/isbn
      RETURN $review/rating
    }
  </book>
SORTBY (title)
```

# Funktionen

- Viele vordefinierte Funktionen, z.B:
  - max(), min(), sum(), count(), avg()
  - distinct(), empty(), contains()
- Endgültige Menge noch nicht fix
- Benutzer-definierte Funktionen
  - Definiert in XQuery Syntax
  - können recursiv sein (funktionale Sprache)
  - Können typisiert sein (z.B: Rückgabewert int)
- Erweiterungsmechanismen sind vorgesehen

# Benutzer definierte Funktionen

```
FUNCTION depth(ELEMENT $e) RETURNS integer
{
  -- An empty element has depth 1
  -- Otherwise, add 1 to max depth of children
  IF empty($e/*)
    THEN 1
    ELSE max(depth($e/*)) + 1
}
```

- **Beispiel Anfrage:**  
`depth(document("partlist.xml"))`

# Xquery Beispiele I

## Daten:

```
<bib>
<book year="1994">
<title>TCP/IP Illustrated</title>
  <author>
    <last>Stevens</last>
    <first>W.</first>
  </author>
<publisher>Addison-Wesley</pu
<price> 65.95</price> </book>
<book year="1992">
<title>Advanced Programming in
  <author>
    <last>Stevens</last>
    <first>W.</first>
  </author>
<publisher>Addison-Wesley</pu
<price>65.95</price> </book>
...
</bib>
```

## Antwort:

```
<bib>
<book year="1994"> <title>TCP/IP
Illustrated</title> </book> <book
year="1992">
<title>Advanced Programming in the
Unix environment</title>
</book>
</bib>
```

## Qu

```
<bib> {
for
$b in document(„ueb1.xml“)/bib/book
where
$b/publisher = "Addison-Wesley" and
$b/@year > 1991
return
<book year="{ $b/@year }">
{
$b/title
}
</book>
}
</bib>
```

## Xquery Beispiele II

### Daten II:

```
<reviews>
<entry>
<title>Data on the Web</title> <price>34.95</price>
<review> A very good discussion of...
</entry>
<entry>
<title>Advanced Programming in...
<price>65.95</price>
<review> A clear and detailed dis...
</entry>
<entry>
<title>TCP/IP Illustrated</title> <
<review> One of the best books...
</reviews>
```

### Antwort:

```
<books-with-prices>
<book-with-prices> <title>TCP/IP
Illustrated</title>
<price-amazon>65.95</price-amazon>
<price-bn> 65.95</price-bn>
</book-with-prices>
....
</books-with-prices>
```

### Query:

```
<books-with-prices> {
for
$b in document("www.bn.com/bib.xml")//book,
$a in
document("www.amazon.com/reviews.xml")//entry
where
$b/title = $a/title
return <book-with-prices> {
$b/title
}
<price-amazon>{ $a/price/text() }</price-amazon>
<price-bn>{ $b/price/text() }</price-bn>
</book-with-prices>
}
```

## XQuery Beispiele III

- Viele mehr siehe XQuery Use Cases  
[ > <http://www.w3.org/TR/xmlquery-use-cases> ]
- Freie Implementierung  
*Galax* [ > <http://www.galaxquery.org/> ]

# Mehr zu XQuery

- W3C XQuery  
[ > <http://www.w3.org/TR/xquery/> ]
- W3C XML Query Use Cases  
[ > <http://www.w3.org/TR/xquery-use-cases/> ]
- W3C XML Query Requirements  
[ > <http://www.w3.org/TR/xmlquery-req/> ]
- W3C XML Query Data Model  
[ > <http://www.w3.org/TR/query-datamodel/> ]
- W3C XML Query Algebra  
[ > <http://www.w3.org/TR/query-algebra/> ]

# XSLT (XSL Transformations)

- XSL (Extensible Stylesheet Language) = XSLT (inkl. XPath) + Formatting Objects (FO)
- XSLT ist regel-basiert
- Ergebnis einer Transformation zumeist XML, eventuell aber auch nicht XML
- Transformationen alternativ
  - auf Server (z.B: Apache's Cocoon)
  - Beim Client (z.B: Netscape 6 or IE 5)
  - Offline



# XSL Fähigkeiten

- A stylesheet specifies the presentation of an XML document. It allows
  - the transformation of the input document into another structure,
  - the description of how to present information.
- Transformation capabilities include:
  - generation of constant text
  - suppression of content
  - moving text
  - duplicating text
  - sorting
  - more complex transformations that compute new from existing information.

# XSLT Example – Input

```
<addresses>
```

```
<address>
```

```
<name>Xaver M. Linde</name>
```

```
<street>Wikingerufer 7</street>
```

```
<town>10555 Berlin</town>
```

```
</address>
```

```
<address>
```

```
<name>John Doe</name>
```

```
<street>42 Gary Cooper Street</street>
```

```
<town>Stanwyck City</town>
```

```
</address>
```

```
</addresses>
```



# XSLT Example – Stylesheet

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:template match="/">  
  <html>  
    <head><title>Addresses</title></head>  
    <body bgcolor="white">  
      <xsl:apply-templates/>  
    </body>  
  </html>  
</xsl:template>
```

template for  
document root

```
<xsl:template match="address">  
  <p>  
    <i><xsl:value-of select="name"/></i><br/>  
    <xsl:value-of select="street"/><br/>  
    <b><xsl:value-of select="town"/></b>  
  </p>  
</xsl:template>
```

template for  
address  
elements

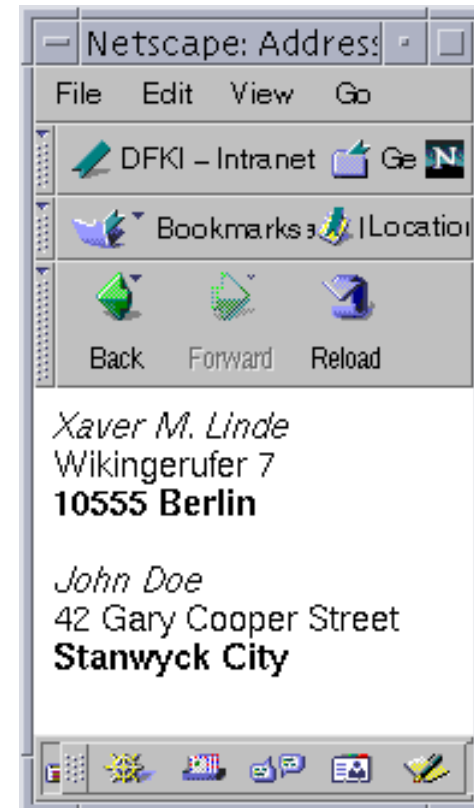
```
</xsl:stylesheet>
```

# XSLT Example – Output

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
```

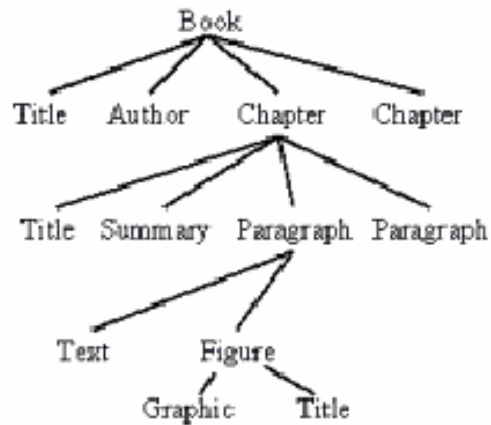
```
<html>
<head><title>Addresses</title></head>
<body bgcolor="white">
  <p><i>Xaver M. Linde</i><br>
    Wikingerufer 7<br>
    <b>10555 Berlin</b>
  </p>
  <p><i>John Doe</i><br>
    42 Gary Cooper Street<br>
    <b>Stanwyck City</b>
  </p>
</body>
</html>
```

*(The HTML code was produced with Apache's Cocoon in HTML mode, hence <br/> became <br>)*

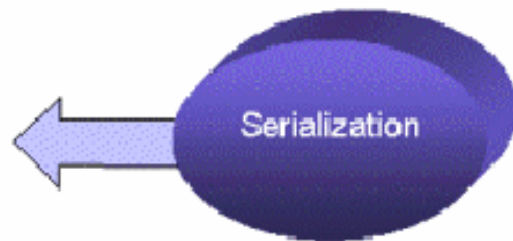
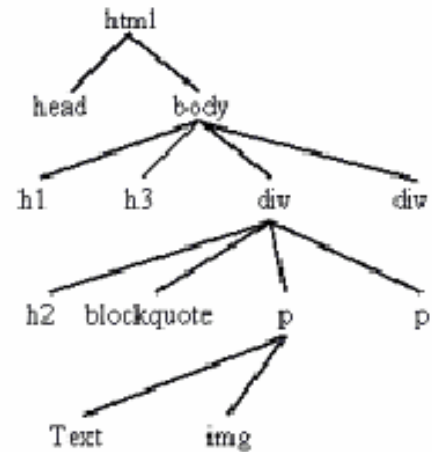


# XHTML Output

## XML Source Tree



## XHTML Result Tree



```

<html>
<head>...</head>
<body>
<h1></h1>
<h3></h3>
....
</body>
</html>
  
```