



XPath & Co.

Malgorzata Mochol
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mochol@inf.fu-berlin.de

Wie geht es weiter?

letzte Woche

- ☑ XML Parser: SAX, DOM & StAX
- ☑ Schema-Übersetzer

heutige Vorlesung

- Navigation & Verknüpfungen (XPath, XPointer, XLink)

- 4.Übung
 - heute (Mi.) 14:15 – 15:45, SR 005 und
 - Do. 10:15-11:45, SR 005
- Thema: XPath

Heute

- XML Path Language (XPath)
- XML Pointer Language (XPointer)
- XML Linking Language (XLink)
- XML Base (XBase)
- XML Query (XQuery)

nächste Woche

- XSLT

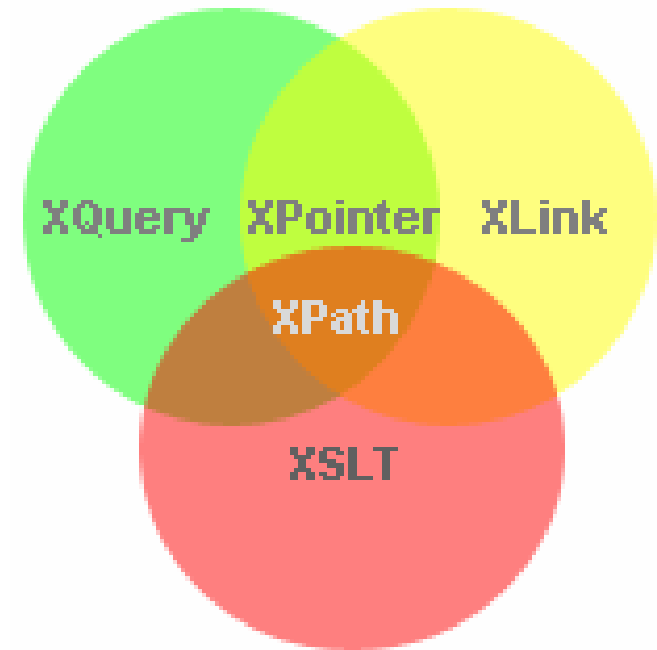


Bild-Quelle: <http://www.w3schools.com/xquery/default.asp>



XML Path Language (XPath)

- Standard zum Zugreifen beliebiger Teile eines XML-Dokuments
- keine XML-Anwendung
- wird von XSLT benutzt
- Adressierungspfad eines Dateisystems ähnlich aber wesentlich mächtiger:
z.B. /order/item
- XPath 1.0 – W3C-Recommendation seit Nov. 1999
 - <http://www.w3.org/TR/xpath>
- XPath 2.0 – W3C-Recommendation seit Jan. 2007
 - <http://www.w3.org/TR/xpath20/>

- ähnliches Modell wie in DOM
 - XML-Dokument als Baum mit Elementen, Attributen und PCDATA als Knoten
 - virtuelle Dokument-Wurzel (Wurzelknoten):
durch "/" repräsentiert (links von "/" steht nichts)
 - ⇒ Wurzel-Element immer Kind von "/":
 - ⇒ Dokument-Wurzel \neq Wurzel-Element
- z.B. /root

Knotentypen (I)

- **Wurzelknoten**

- oberster Knoten im Baum, dessen Kind der Elementknoten des Dokuments ist
- string-Wert: Verkettung der Zeichendaten aller Textknoten-Kinder in der Dokumentenreihenfolge

- **Elementknoten**

- Knoten für ein Element
- string-Wert: Verkettung der Zeichendaten aller Textknoten-Kinder des Elements

- **Textknoten**

- Knoten, der Zeichendaten enthält
- string-Wert: die Zeichendaten des Textknotens

Knotentypen (II)

- **Attributknoten**

- Knoten für jedes Element zugeordnete Attribut
- string-Wert: Normalisierter Attributwert

- **Namensraumknoten**

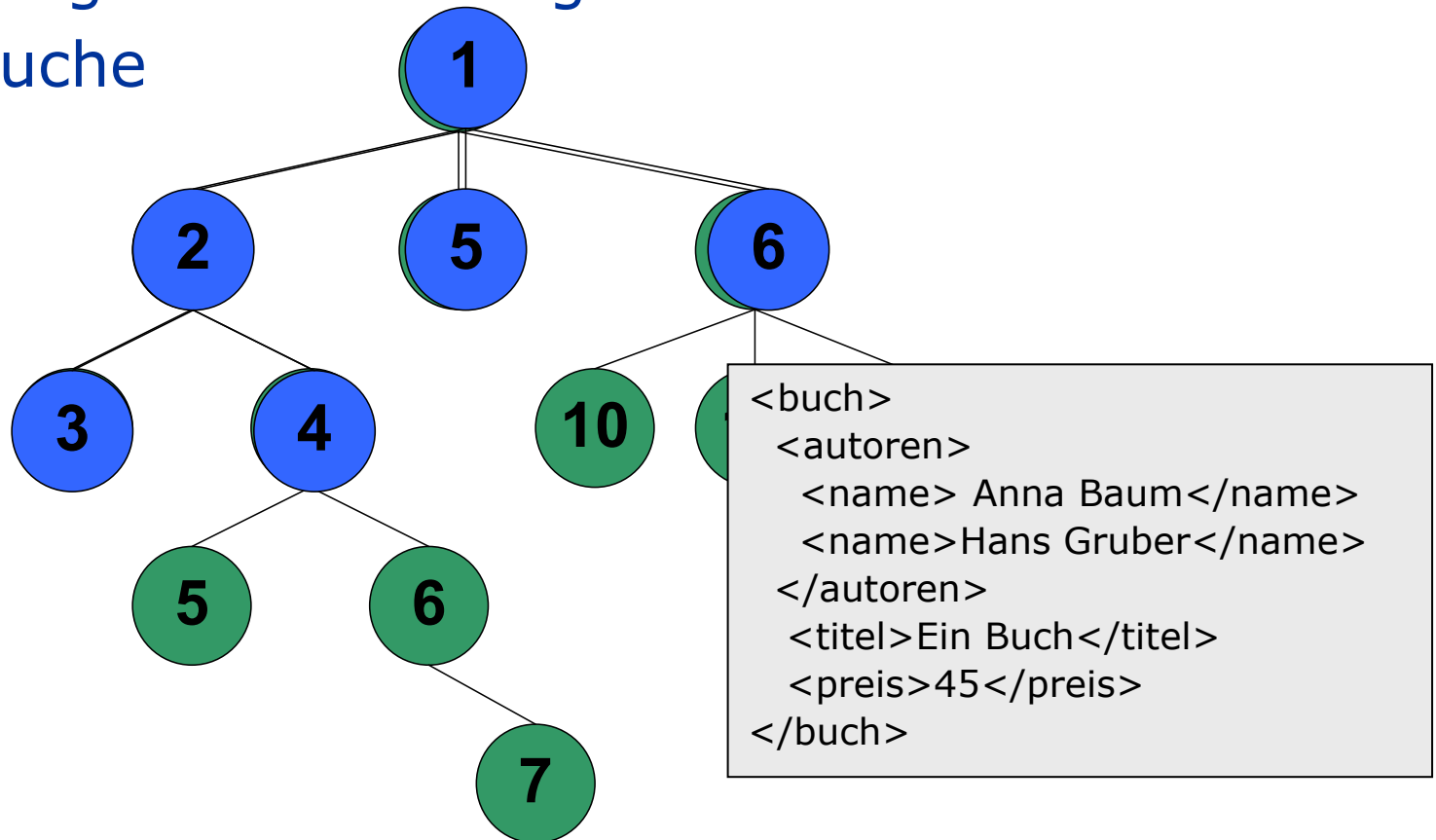
- der Namensraum ist jeweils einem Elementknoten als Elternknoten zugeordnet, ist aber nicht Kind dieses Elementknoten
- string-Wert: URI des Namensraum

- **Kommentarknoten**

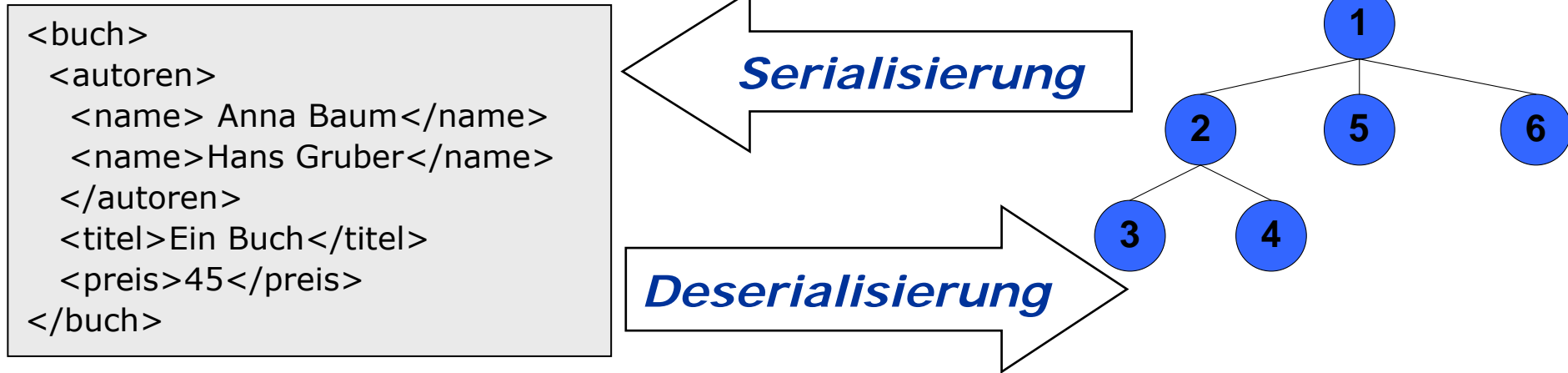
- Knoten für jeden einzelnen Kommentar
- string-Wert: Kommentarinhalt

Dokumentenreihenfolge

- Baummodell als Basis
- feste Dokumentreihenfolge (document order) = Reihenfolge der Start-Tags im Dokument
- Tiefensuche



(De)Serialisierung



- **Serialisierung** – Erzeugung eines Dokuments aus einem Baum
- **Deserialisierung** – Erzeugung eines Baums aus einem Dokument

- Elemente werden einfach **über ihren Namen** identifiziert:
z.B. **order** oder **order/item**
- Attribute werden mit "**@name**" identifiziert:
z.B. **@id** oder **order/@id**

```
<?xml version="..." encoding="..."?>  
<order id="O56">  
  <item item-id="E16-2">  
    <name>buch</name>  
  </item>  
</order>
```

Absolute und relative Pfade

- **absolute Pfade**

- beginnen mit "/",
z.B. /order/item

lesen: (➔) Folge dem Pfad von dem Wurzelknoten zu einem Kind-Element `order` und von dort aus zu einem Kind-Elementen `item`!

- **relative Pfade**

- beginnen mit einem Element oder Attribut

z.B. `order/item`

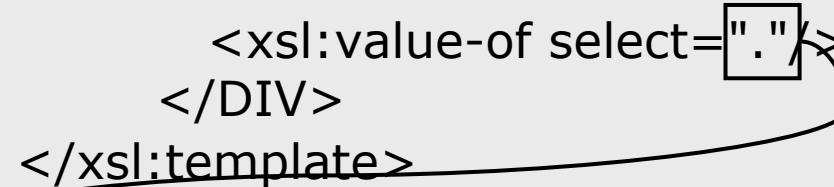
lesen: (➤) `item`-Elemente, die Kind eines Elementes `order` sind

- Element `order` kann an beliebiger Stelle des XML-Dokumentes

Kontext-Knoten

- XPath-Pfade werden in XSLT immer bzgl. Eines bestimmten **Kontext-Knotens** ausgewertet: Element-, Attribut- oder Text-Knoten
- Beispiel:

```
<xsl:template match="p">
  <DIV>
    <xsl:value-of select="."/ >
  </DIV>
</xsl:template>
```



- Was bedeutet hier aktueller Knoten "." ?
- "." = Kontext-Knoten
- Kontext-Knoten = Knoten, auf den das Template angewandt wird (hier ein p-Element)

Lokalisierungsstufe



- besteht aus:
 - einem Achsenbezeichner
 - einem Knotentest
 - einem oder mehreren Prädikat (optional)
- ...in der Form:

Achsenbezeichner::Knotentest[Prädikat1][Prädikat2]

- :: Trennzeichen zwischen Achsenbezeichner und Knotentest

Aschen (I)



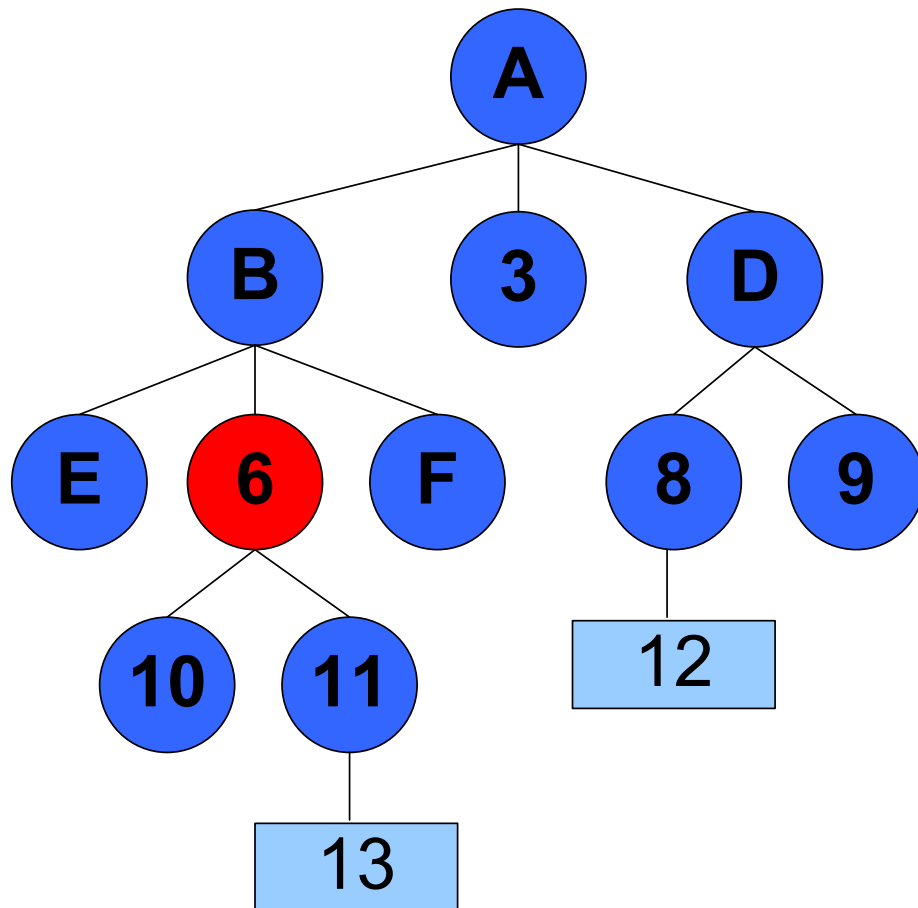
- `self::` Kontextknoten (.)
- `child::` Kinder des Kontextknoten (/)
- `parent::` Eltern des Kontextknoten (..)
- `descendant::` Nachkommen des Kontextknoten (//)
- `descendant-or-self::` Kontextknoten & descendant-Achse
- `ancestor::` Vorfahren des Kontextknoten bis zum Wurzel
- `ancestor-or-self::` Kontextknoten & ancestor-Achse

Aschen (II)



- preceding-sibling:: vorhergehende Geschwisterknoten
- preceding:: Elementknoten vor dem Kontextknoten ohne vorfahren
- following-sibling:: alle folgende Geschwisterknoten
- following:: Elementknoten, die in der Dokumentenreihenfolge dem Kontextknotenfolgen
- attribute:: Attribute des Kontextknoten (@)
- namespace:: Namensraumknoten, falls vorhanden

- . aktueller Knoten
- .. Eltern-Knoten
- * beliebiges Kind-Element
- @* beliebiges Attribut
- // überspringt ≥ 0 Hierarchie-Ebenen nach unten
- [] Prädikatbeschreibung (Ziel \rightarrow genauere Element-Spezifikation)
- | Auswahl (Vereinigung)
- Beispiel: *|@*
„Kind-Element oder Attribut des aktuellen Knotens“



- self:: 6
- child:: 10, 11
- parent:: B
- descendant:: 10, 11, 13
- descendant-or-self:: 6, 10, 11, 13
- ancestor:: B, A
- ancestor-or-self:: 6, B, A
- preceding-sibling:: E
- preceding:: E, B, A
- following-sibling:: F
- following:: 10, 11, 13, F, 3, D, 8, 12, 9

Ausführliche vs. **gekürzte** Schreibweise

Achse Knotentest Achse Knotentest Prädikat

`/child::buch/descendant::autoren[attribute::id='E16-2']`

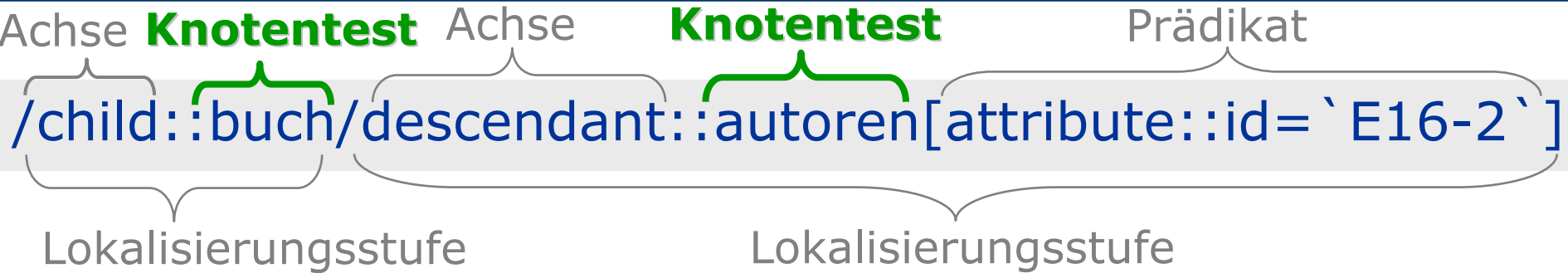
Lokalisierungsstufe

Lokalisierungsstufe

```
<?xml version="..." encoding="..."?>
<buch>
  <autoren id="E16-2">
    <name> Anna Baum</name>
    <name>Hans Gruber</name>
  </autoren>
  <titel>Ein Buch</titel>
  <preis>45</preis>
</buch>
```

`/buch//autoren[@id='E16-2']`

Knotentest



- Filterung der Knotenmenge
- Filterungs-Kriterium:
 - Knotenname
z.B.: `child::buch`
 - Knotentyp
z.B.: `child::text()`
`child::node()`

Prädikate



- Verfeinerung der Filterung durch Prädikate
- Anzahl der Prädikate ≥ 0
- [] Bedienung
- Prädikatausdruck unterstützen
 - logische Operatoren: $<$, $>$, \leq , \geq , $=$, \neq
wobei $>$, $<$ müssen als Entity-Referenzen $>$ und $<$ benutzt werden
 - numerische Operatoren: $+$, $-$, $*$, div , mod

- `order/item[@item-id = 'E16-2']`
 - item-Elemente, die Kind von order sind und Attribut item-id mit Wert 'E16-2' haben

```
<order id="4711">  
  <item item-id="E16-2">  
    <name>buch</name>  
  </item>  
</order>
```

- Randbedingungen können an beliebiger Stelle in einem Pfad vorkommen:

- `order[@order-id = '4711']/item`

```
<orders>  
  <order ored-id="4711">  
    <item item-id="E16-2">  
      <name>buch</name>  
    </item>  
  </order>  
  <order id="4711">  
    <item item-id="E16-3"/>  
  </order>  
</orders>
```

/child::buch/descendant::autoren[attribute::id=`E16-2`]

→ /buch//autoren[@id=`E16-2`]

/descendent::name

→ //name

/child::buch/child::titel

→ /buch/titel

wenn buch der Kontextknoten ist:

child::preis/child::zahl

→ preis/zahl

```
<?xml version="..." encoding="..."?>
<buch>
  <autoren item-id="E16-2">
    <name> Anna Baum</name>
    <name>Hans Gruber</name>
  </autoren>
  <titel>Ein Buch</titel>
  <preis>
    <zahl>45</zahl>
    <währung>Euro</währung>
  </preis>
</buch>
```



XPath Funktionen

Grundlegende Datentypen

- node-set – liefert eine ungeordnete Knotenmenge
- string – ist eine Zeichenfolge
- number – Fließkommazahl
- boolean – Werte true und false

Knotenmenge-Funktionen

- Funktionen:

- *number* last() – eine Zahl, die die Größe der aktuellen Knotenmenge entspricht
- *number* position() – Position eines Knotens
- *number* count(node-set) – Anzahl der Knoten in der Knotenmenge

- Beispiele:

order/item[position() = 1]

order/item[position()=last()]

- Funktionen:

- *string* string(object) – interpretiert ein übergebenes Argument als Zeichenkette und gibt die ermittelte Zeichenkette zurück
- *string* string-length(string) – Länge von String (Anzahl der Zeichen)
- *boolean* starts-with(string, string) – true wenn die erste Zeichenkette mit der zweiten Zeichenkette anfängt

Boolesche Funktionen

- Funktionen:

- *boolean* `boolean(object)` – nimmt ein Objekt und liefert booleschen Wert zurück
- *boolean* `not(object)` – nimmt booleschen Ausdruck und liefert `true` wenn Argument `false` ist
- *boolean* `true()` – immer `true`
- *boolean* `false()` – immer `false`

- Beispiel:

`order/item[not(position)=last()]`

- Funktionen:

- *number* number(object) – versucht eine Zeichenkette als Zahl zu interpretieren und gibt die ermittelte Zahl zurück
- *number* sum(node-set) – Gesamtsumme der Zahlenwerte des Ausgangsknotens (nach Umwandlung des String-Werte)
- *number* round(number) – rundet den Wert zur nächsten Ganzzahl

- Beispiel:

`number(3xy) → 3`



XPath 2.0

XPath 2.0

- Januar 2007 – W3C Recommendation für neue Version von XPath
- zeitgleich mit XQuery 1.0 & XSLT 2.0
- rückwärts kompatibel zu XPath 1.0

Was hat sich geändert? Was ist neu?

1. erweitertes Datenmodell
2. neue Konstrukte für Ausdrücke
3. neue Datentypen
4. neue Operatoren
5. erweiterte Funktionsbibliothek

1. Erweitertes Datenmodell

- Berücksichtigung einzelner Werte (**atomic values**)
- Daten unterschiedlichen Typs:
 - Zeichenfolgen, Zahlen, logische Werte, Datums- und Zeitwerte
 - qualifizierte Namen & URIs
 - einfache Sequenzen & Listen
- Ergebnis eines XPath Ausdrucks: Auswahl von Knoten, Einzelwert oder Sequenz
- XPath 2.0 auf Knotenbaum
 - nur Daten auslesen
- XPath 2.0 auf Einzelwerten & Sequenzen
 - neue Werte/Sequenzen erzeugen

2. Neue Konstrukte für Ausdrücke

- für Operationen mit Sequenzen Verwendung des *for*-Ausdrucks
- Beispiel

```
for $i in 1 to 3 return $i*$i
```

Variable
(Bereichsvariable)

binding sequenz

return-Ausdruck

Ergebnis: 1, 4, 9

- bedingte Ausdruck *if*

```
if @menge > 1000 then "gut" else "weniger gut"
```

- quantifizierende Ausdrücke *some* und *every*

```
some $a in $lager/artikel satisfies $lager/artikel/menge=0
```

- wahr, wenn die Menge mind. bei einem Artikel = 0

```
every $a in $lager/artikel satisfies $lager/artikel/menge>0
```

- wahr, wenn von allem Artikel mind. einer vorhanden ist.

3. Neue Datentypen

- Unterstützung der XML-Schema Datentypen
- XPath 1.0
 - number – Fließkommazahl
- XPath 2.0
 - integer
 - decimals
 - single precision
 - Datums-, Zeit- und Dauerwerte

4. Neue Operatoren

- **Knotenvergleiche**
 - *is* – prüft, ob zwei Ausdruck den selben Knoten liefern
 - *<<, >>* – prüfen, welcher von zwei Knoten in der Dokumentreihenfolge früher oder später erscheint

- **Kombination von Knotensequenzen**
 - *union* – Vereinigung zwei Knotensequenzen zu einer Sequenz
 - *intersect* – erzeugt aus zwei Sequenzen eine Sequenz, die Knoten enthält, die in beiden vorkommen
 - *except* – erzeugt aus zwei Sequenzen eine Sequenz, die Knoten enthält, die in der ersten Sequenz aber nicht in der zweiten vorkommen

5. Erweiterte Funktionsbibliothek

- Behandlung von Zeichenketten
- Verwendung regulärer Ausdrücke
- Funktionen für Arbeit mit Sequenzen
- Funktionen für Datums- und Zeitwerte



XML Linking Language (XLink)

XLink – Ziele & Eigenschaften

- Ziel:
 - innerhalb von XML-Dokumenten Links zu erzeugen
 - Vokabular für Beschreibung von Links
- Eigenschaften
 - definiert, wie Dokumente miteinander verknüpft werden
 - erlaubt Links zwischen Ressourcen zu erzeugen & zu beschreiben
 - nutzt XML-Syntax
 - bietet ein minimales Link-Verhaltensmodell
 - bietet Link-Datenstrukturen
- Rahmen für die Erzeugung
 - grundlegender unidirektionaler Links
 - komplexerer Link-Strukturen

- Entwicklung beeinflusst von vieler etablierter Hypermedia-Systeme und –Standards:
 - **HTML** definiert mehrere Elementtypen, die Links repräsentieren
 - **HyTime** definiert Strukturen u.a. für eingebettete, eingehende und Third-Party-Links
 - **Text Encoding Initiative Guidelines** bietet Strukturen zum Erzeugen von Links, aggregierten Objekten und Linksammlungen.

XLink – Features

- Juni 2001 W3C Recommendation
- Anforderungen:
 - Links, die in beide Richtungen begangen werden können
 - Links, die mehrere Ziele ansteuern können
 - Links, die mit Metadaten versehen sind
 - Links, die unabhängig von den Dokumenten gespeichert sind

XLink – Verwendung

- keine eigenen Elementen → Benutzung vorhandener Elemente
- Eigenschaften eines XLinks-Element über Attribute
- Globale Attribute für die Definition von XLink-Elementtypen
- XLink Namensraum → <http://www.w3.org/1999/xlink/>
 - Elementen werden XLink-Eigenschaften mitgeben

Einfache vs. erweiterte Links

- **einfache Links**

- einfache Verallgemeinerung von HTML-Links
- können einem Link und seinem Ziel einen Namen geben (title und role)
- können das Verhalten eines Links spezifizieren (actuate und show)
- werden bisher nur von wenigen Browsern unterstützt

- **erweiterte Links**

- verbinden mehr als zwei Ressourcen
- beschreiben eine Menge von Ressourcen zusammen mit ihrer Link-Struktur (Netzwerk)
- werden von keinem Browser unterstützt

Einfaches Link – Beispiel

```
<?xml version="1.0" encoding="..."?>
<image xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="bild.gif"
xlink:actuate="onRequest"
xlink:show="new">Zeige das Bild
</image>
```

Ressource

W3C XLink
Namensraum

Ladevorgang
(Zeitpunkt)

Darstellung

- XML Element verwendbar als XLink-Element wenn es mind. ein Attribut `xlink:type` mit einem gültigen Wert enthält
- Problem: nur weniger/eingeschränkte Implementierungen von XLink

XLink – Attribute

- **xlink:href** gibt an, wo eine Ressource zu finden ist (URI)
- **xlink:actuate** gibt an, wann die Ressource geladen wird
 - "onLoad" gleichzeitig mit dem Dokument
 - "onRequest" nur nach Aufforderung
- **xlink:show** gibt an, wo die Ressource dargestellt wird
 - "new": in einem neuen Fenster
 - "replace": im aktuellen Fenster und dieses ersetzen
 - "embeded": ins aktuelle Fenster integrieren

- beschreiben eine Menge von Ressourcen zusammen mit ihrer Link-Struktur (Netzwerk)
 - Ressourcen existieren unabhängig von Links
 - extra Linking-Elemente definieren Beziehungen
 - erlaubt many-to-many-Beziehungen
- werden von keinem Browser unterstützt
- stoßen bisher auf wenig Akzeptanz
- mögliche Alternative:
 - RDF (*Resource Description Framework*)



XML Base (XBase)

- in HTML 4.01 `<base>` für Basisadresse für URIs
- XBase
 - Spezifikation für XML-Attribut `xml:base`
 - Bestimmung einer anderen Basisadresse als die, die ein XML Dokument selbst verwendet
 - Präfix `xml` gebunden an den Namensraum <http://www.w3.org/XML/1998/namespace>
- Juni 2001 W3C Recommendation

XBase mit XLink – Beispiel

```
<?xml version="1.0" encoding="..."?>
<image xml:base="http://www.mybase.de/mylinking"
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="bild.gif"
xlink:actuate="onRequest"
xlink:show="new">Zeige das Bild
</image>
```

xml:base

+

relativer URI

=

vollständiger URI

<http://www.mybase.de/mylinking/bild.gif>

- XLink stellt flexibles Linking für XML bereit
- XML Base liefert einen bewährten Ansatz zur Nutzung von Base URI für XML
- XLink, XML Base bringt Tiefe und Leistungsfähigkeit in die XML-Familie

Quelle: <http://www.w3c.de/Press/link-base-pressrelease.html>



XML Pointer Language (XPointer)

- 1997 – Grundlagen für XPointer in Zusammenhang mit XLink
- März 2003 – W3C Recommendation
 - grundlegendes Framework
 - Schemas, die dem Framework entsprechen
- basiert auf & erweitert XPath in einigen Bereichen
- Ziel
 - Zeigen auf bestimmte Elemente innerhalb eines XML Dokumentes (engl. to point ~ zeigen)

XPointer – Eigenschaften

- XPointer nur in wohlgeformten XML-Dokumenten möglich
- XPointer ermöglicht
 - Punkte, Bereiche und Knoten anzusprechen
 - Informationen durch String-Vergleiche zu lokalisieren
 - Adressen in URI-Referenzen als Identifikationspunkte zu nutzen
 - Adressierung ohne anchor-Markup in Zieldokument
 - Adressierung von Knoten (Elemente, Attribute, Text)
- Extension von XPath: zusätzlich auch Adressierung von Dokumentteilen, die keine Knoten sind
→ Adressierung von Punkten (points) und Bereichen (range)

XPointer – Aufbau

- XPointer Framework

- <http://www.w3.org/TR/xptr-framework/>

- element() Scheme

- <http://www.w3.org/TR/xptr-element/>

- xmlns() Scheme

- <http://www.w3.org/TR/xptr-xmlns/>

W3C Recommendations

- xpointer() Scheme

- <http://www.w3.org/TR/xptr-xpointer/>

W3C Working Draft

Zeigertyp element()

- zeigt auf ein einzelnes Element innerhalb des Dokuments durch
 - Element-ID
 - Indexsequenz – besteht aus Indexnummern der einzelnen Elemente
- Beispiel
 - im Dokument zeige auf Element "D"

```
<A id="12345" >
  <B></B>
  <C>
    <D/>
  </C>
</A>
```

#element(12345/2/1)

#element(/1/2/1)

Pfad vom root-Element aus: erstes Kind des zweiten Kindes des ersten Kindes

Verknüpfung xmlns()

- um eine XML-Resource an einen Namespace zu binden
- Beispiel

```
<A xmlns="http://elementA">
  <B xmlns="http://elementB">
    </B>
  </A>
```

Dokument

```
xmlns(NameA=http://elementA) xmlns(NameB=http://elementA)
xpointer(/NameA:A/NameB:B)
```

XPointer

→ Zugriff auf Elemente über die Namensräume hinweg

Zeigertyp xpointer()

- entspricht einem XPath-Ausdruck
- Funktionen und Schreibweisen von XPointer möglich
- Beispiele

```
meinequelle.xml#xpointer(id("a1"))
```

→ wählt aus dem Dokument `meinequelle.xml` Element mit ID-Attribut `id="a1"`

```
meinequelle.xml#a1
```

→ einfache Schreibweise

Erweiterungen zu XPath

- Punkt (point)

`meinequelle.xml#xpather(/A/[0])`

Position vor dem ersten Kinderknoten von A

`meinequelle.xml#xpather(/A/[3])`

Position nach dem dritten Kinderknoten von A

- Bereich (range)

`meinequelle.xml#xpather(/A/[3] range-to /A/[5])`

Startpunkt

Endpunkt

Bereich zwischen dem dritten und fünften Zeichen



XML Query

Was ist XQuery?

XQuery ...

- ist **die Abfragesprache** für XML-Daten
 - XML-Dateien &
 - alles was in XML darstellbar ist (auch DBs)
- ist für XML das, was SQL für Datenbanken
- basiert auf **XPath-Ausdrücken**
- wird bei fast allen DB-Engines unterstützt (IBM, Oracle, Microsoft, etc.)
- ist sein Januar 2007 eine **W3C Recommendation**
→ <http://www.w3.org/TR/xquery/>

XQuery kann benutzt werden um ...

- Informationen zu extrahieren, um sie in Web Services zu nutzen
- Reports zu generieren
- XML ins XHTML zu transformieren
- in Web-Dokumenten nach relevanten Informationen zu suchen

Schlüsselwörter & Variablen

- **for** – zur (Schleifen-)Verarbeitung von einzelnen Elementen innerhalb eines XML-Dokuments
- **let** – zur Erstellung von Variablen und Zuweisung von Werten
- **where** – konditionale Anweisung, wird zusammen mit dem Schlüsselwort *for* verwendet
- **return** – zur Rückgabe von Werten an die aufrufende Instanz des Ausdrucks
- **order** – sortier die Ausgabe
- **if-then-else** – bedingte Abfragen

- **\$** - kennzeichnet eine Variable
- **doc("file.xml")** – öffnet das spezifizierte XML-Dokument (hier: file.xml)

for – Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
</bookstore>
```

Dokument

```
<title lang="en">Everyday Italian</title>
<title lang="en">Learning XML</title>
```

**XQuery-
Anfrage an
das
Dokument**

**Ergebnis der
Anfrage**

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return $x
```

Quelle: http://www.w3schools.com/xquery/xquery_flwor_html.asp

for – Beispiel mit HTML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
</bookstore>
```

Dokument

```
<ul>
  <li>Everyday Italian</li>
  <li>Learning XML</li>
</ul>
```

**Ergebnis der
Anfrage**

**XQuery-
Anfrage an
das
Dokument**

```
<ul>
{
  for $x in doc("books.xml")/bookstore/book/title
  order by $x
  return <li>{data($x)}</li>
}
</ul>
```

Quelle: http://www.w3schools.com/xquery/xquery_flwor_html.asp

if-then-else – Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
  </book>
</bookstore>
```

Dokument

```
<various>Everyday Italian</various>
<web>Learning XML</web>
```

Ergebnis der Anfrage

XQuery-Anfrage an das Dokument

```
for $x in doc("books.xml")/bookstore/book return
  if ($x/@category="WEB")
    then <web>{data($x/title)}</web>
    else <various>{data($x/title)}</various>
```

Quelle: http://www.w3schools.com/xquery/xquery_flwor_html.asp

Wie geht es weiter?

heutige Vorlesung

- ☑ XPath 1.0 & 2.0
- ☑ XLink
- ☑ XBase
- ☑ XPointer
- ☑ XQuery

Vorlesung nächste Woche

- XLST

Übung nächste Woche

- 5. Übung: XSLT
- 4. Übungsblatt