XML Path Language (XPath)

http://www.w3.org/TR/xpath/ ,
http://www.w3.org/TR/xpath20/

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Prolog: XPath in APIs

Beispiele zum Zugriff auf XML-Daten mit XPath
und dem Ruby-API „REXML“

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: REXML-Beispiel

```
#!/usr/bin/env ruby
#
require "rexml/document"
include REXML        # Vermeidet Präfix „REXML::“

# XML-Dokument als Datenstruktur in den Speicher laden:
file = File.new( "08-bestell.xml" )
doc  = Document.new( file )

# XPath-Ausdrücke im Folgenden in rot.

# Ausgabe der Texte (Inhalte) aller Elemente „Beschreibung“:
XPath.each( doc, "//Beschreibung") { |element|
   puts element.text
}
```


© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: REXML-Beispiel

Array aller Elemente „ArtNr“, die eine ISBN enthalten:

```
articles = XPath.match( doc,"//ArtNr[@IdentArt='ISBN']")
```

Rollen der Handelspartner:

```
doc.elements.each("//Bestellkopf/Handelspartner") {|e|
      puts e.attributes["Rolle"]
}
```

Liste aller Belegnummern:

```
doc.elements.each
   ("/Bestellungen/Bestellung/Bestellkopf") { |element|
      puts element.elements["Belegnummer"].text
}
```


© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: Eine kleine Einführung

http://www.w3.org/TR/xpath

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Was ist XPath?

- Eine Sprache
  - zur Adressierung / Selektion von Teilen von XML-Infosets
  - zur Prüfung, ob ein Knoten des XML-Dokumentenbaumes bestimmte Bedingungen erfüllt (*pattern matching*)
  - zur Manipulation von Strings, Zahlen und booleschen Ausdrücken
  - mit ähnlicher Bedeutung für XML *information sets* wie SQL für relationale Datenbankmodelle/Schemata.
- Die Sprache
  - ist kompakt (verwendet nicht die XML-Syntax)
  - ist erweiterbar
- Der Name XPath
  - erklärt sich durch ihre Erweiterbarkeit und eine an Pfade in Dateisystemen erinnernde, allerdings stark verallgemeinernde Notation, z.B. `/doc/chapter[5]/section[2]`, `chapter//para`, aber auch: `para[@type="warning"] .`

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Was ist XPath?

- XPath ist die Grundlage
  - historisch von XSLT und XPointer
  - Inzwischen auch von weiteren XML-Technologien, z.B. von

    XML Schema (*identity constraints*)

    Schematron (alle *constraints*)

    DOM u.a. XML-APIs

    XQuery
- XPath bezieht sich auf logische XML-Dokumente, bestehend aus Knoten verschiedenen Typs, angeordnet in Baumform
  - analog zu (aber nicht identisch mit!) XML Information Sets
- XPath
  - unterstützt in Version 1.0 (1999) bereits XML-Namensräume
  - ist aber noch nicht auf XML Schema abgestimmt
  - Erscheint inzwischen zusammen mit XQuery & XSLT (XPath 2.0 + XQuery 1.0 + XSLT 2.0, REC 23. Januar 2007).

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Das XPath-Datenmodell

- XPath unterstellt
  - dass ein XML-Dokument als Baumstruktur vorliegt und
  - in bestimmter Weise seriell gelesen werden kann (*document order*).
- XPath definiert und verwendet 7 Knotentypen:
  - *R: root node* (Dokumentwurzelknoten)
    genau einer pro Dokument
  - *E: element nodes* (Element-Knoten)
  - *T: text nodes* (Textknoten)
  - *A: attribute nodes* (Attribut-Knoten)
  - *N: namespace nodes* (Namensraum-Knoten)
  - *P: processing instruction nodes* (PI-Knoten)
  - *C: comment nodes* (Kommentarknoten)
- Nur die Typen R und E können verkettet werden.
  - Bem.: Die anderen sollte man also als „Blätterknoten“ bezeichnen.
  - Typen N und A sind ihren E-Knoten assoziiert („separate“ Zweige).

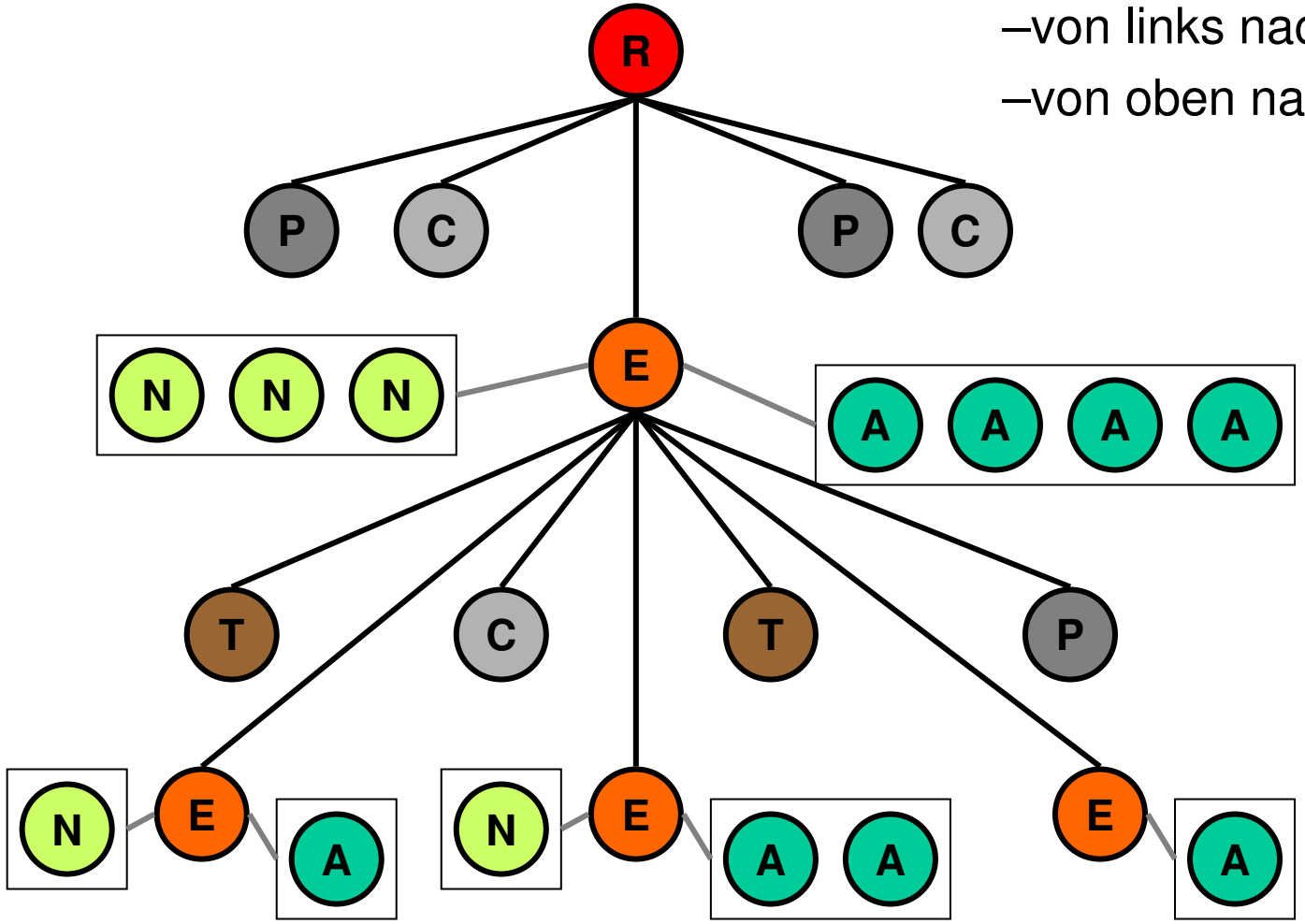
© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Das XPath-Datenmodell

Lesefolge

–von links nach rechts,

–von oben nach unten



© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Das XPath-Datenmodell

- Knoteneigenschaften
  - Für jeden Knoten(typ) läßt sich ein Stringwert ermitteln.
  - Viele Knoten besitzen einen expandierten Namen, bestehend aus

namespace URI	*null* oder ein String
local part	ein String

- Knotenreihenfolge (Dokumentenreihenfolge)
  - Erster Knoten ist stets der *root node* R, nicht zu verwechseln mit dem ersten Elementknoten, dem des *document element!*
  - Elementknoten erscheinen in der Reihenfolge ihrer *start tags*, also Eltern- vor Kind-Knoten.
  - Einem Elementknoten E folgen ggf. seine *namespace nodes*, dann seine *attribute nodes*, dann seine *child nodes*.
  - Die Reihenfolge der *namespace* und *attribute nodes* ist fest, aber unbestimmt – und damit implementierungsabhängig.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath und XML Infoset

- XPath ist „nur“ am Inhalt von Dokumentinstanzen interessiert
  - in möglichst „bequemer“ Form
  - Validierbarkeit interessiert nicht, Wohlgeformtheit reicht.
- Im Vergleich zu XML Infoset wurden daher Vereinfachungen vorgenommen:
  - Es gibt keine Entsprechung zu:

    Document type info items

    Unparsed entity reference info items

    Notation info items
  - Die Dokumententopologie wurde zu einem Baum vereinfacht
  - Einige *info item*-Eigenschaften werden ignoriert

    Beispiel: Angaben aus der XML-Deklaration
  - Folgen von *character info items* werden zu *text nodes* zusammengefasst, *whitespace*-Angaben gehen verloren.
- Vorsicht: XPath unterstellte eine inoffizielle Infoset-Spezifikation!

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath und XML Infoset

Knoten	Stringwert	Expandierter Name
• R	Stringwert von E	-
• E	Konkatenierung der Stringwerte aller enthaltenen T	*ii: namespace URI, local name* (ii = *information item*)
• A	ii: *normalized value*	*ii: namespace URI, local name*
• T	Konkatenierung aller *character values* aufeinanderfolgender *character info items*	-
• N	*ii: namespace URI*	*null, ii: prefix*
• P	*ii: content*	*null, ii: target*
• C	*ii: content*	-

Bem.: Mozilla zeigt bei leerer CSS-Datei offenbar den Stringwert von R an!

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke

- XPath-Ausdrücke sind die zentralen Objekte der Sprache. Sie werden ausgewertet und liefern ein Objekt zurück, etwa:
  `/mydoc/selection` liefert alle „selection“-Elemente von „mydoc“
- Es gibt 4 derartige Objekttypen:
  - Knotenmenge (*node set*) bzw. Feld (*sequence, XPath 2.0*) - der mit Abstand wichtigste Fall!
  - Boolescher Ausdruck (*true* oder *false*),
  - Zahl (*floating-point*) und
  - String (UCS)
- Auswertungen finden stets in einem Kontext statt:
  - dem Kontext-Knoten (per *default* ist das zunächst R)
  - der Kontext-Position und -größe (zwei positive ganze Zahlen)
  - einem Satz Variablen mit ihren aktuellen Inhalten
  - einem Satz Funktionen (*core functions* + *extensions*)
  - einem Satz Namensraum-Deklarationen
- Den Kontext definieren XPath-Anwendungen wie z.B. XSLT

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke

- Knotenmenge vs. Sequenz (Folge)
  - Die Knotenmengen von XPath 1.0 entsprechen der Begriffsbildung für mathematische Mengen:
    - Reihenfolge ist nicht spezifiziert (oft jedoch Dokumentenreihenfolge)
    - Keine Doppelten (jeder Knoten kann nur einmal auftreten, auch wenn der Ausdruck ihn mehrfach selektiert)
  - Die Sequenz von XPath 2.0 bricht mit dieser Sichtweise
    - Reihenfolge wird stets eingehalten
    - Mehrfaches Vorkommen eines Knotens ist daher möglich
- Analogie bei Datenstrukturen
  - Hash/Dictionary vs. Array

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke

Rückgabetyp „Knotenmenge/Feld“:

- Der XPath-Ausdruck soll alle Knoten des Dokumentbaumes zurückliefern,
  - auf die der übergebene Ausdruck passt
  - und zwar im aktuellen Kontext.

- Wichtigster Spezialfall eines Ausdrucks, der Knotenmengen erzeugt: *location path* (etwa: „Suchpfad“).

- Unterscheide relative und absolute Suchpfade:
  - Relative: `section/para,` `./customer`
  - Absolute: `/mydoc/section/para,` `/book/title`

(in Analogie zu Dateisystem-Pfaden)

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke

Rückgabetyp „Knotenmenge/Feld“ (Forts.):

- Suchpfade setzten sich aus Suchschritten (*location steps*) zusammen, diese bestehen wiederum aus drei Teilen:
  - einer „Achse“ (*axis*)
  - einem Knotentest (*node test*)
  - optionalen Prädikaten (*predicates*)
- Beispiel:

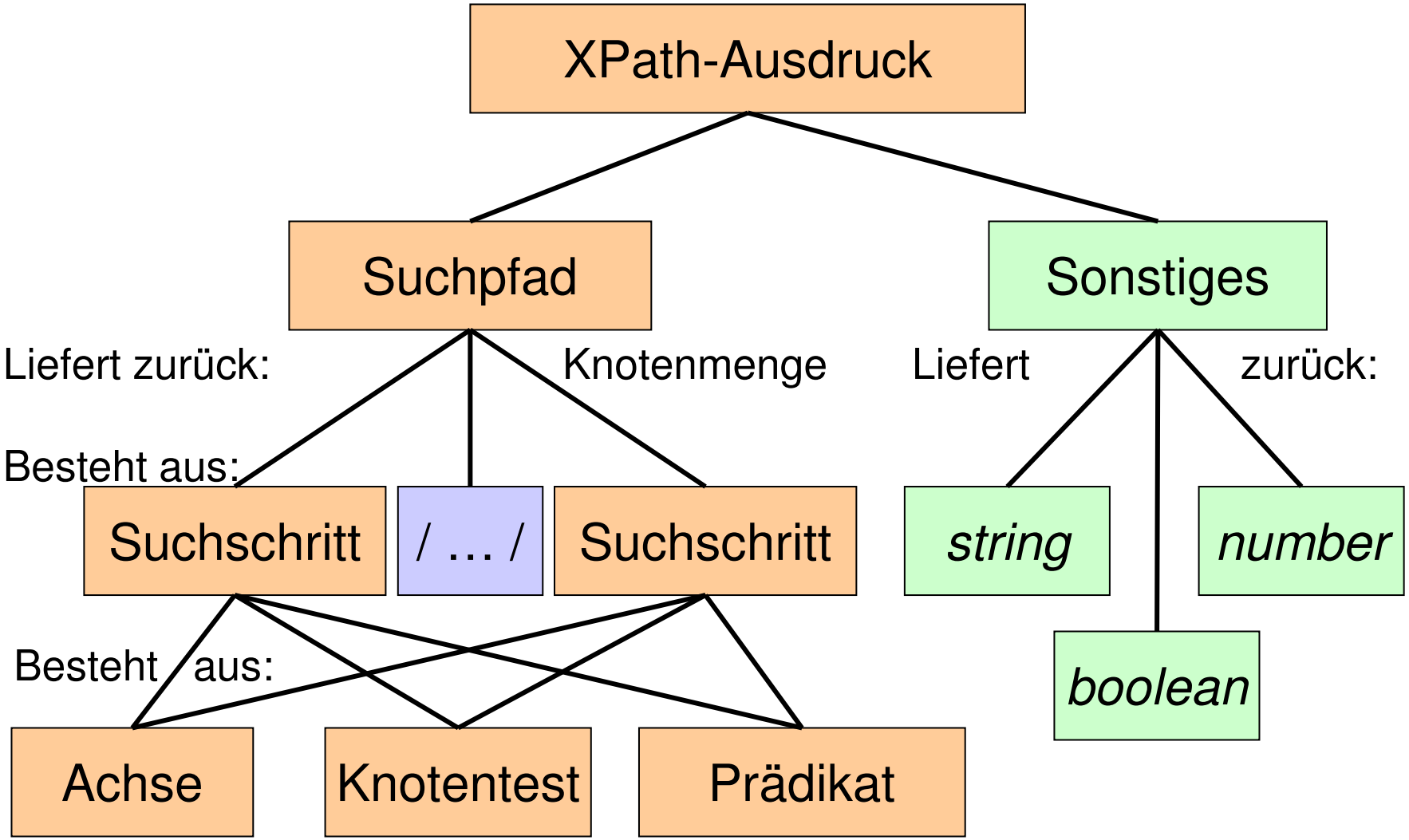
```
child::para[position()=1]
```

Ausgehend vom Kontextknoten, das erste Kind-Element namens „para“.
Hinweis auf Kurzschreibweise: `para[1]`

- Suchschritte werden durch „/“ getrennt.

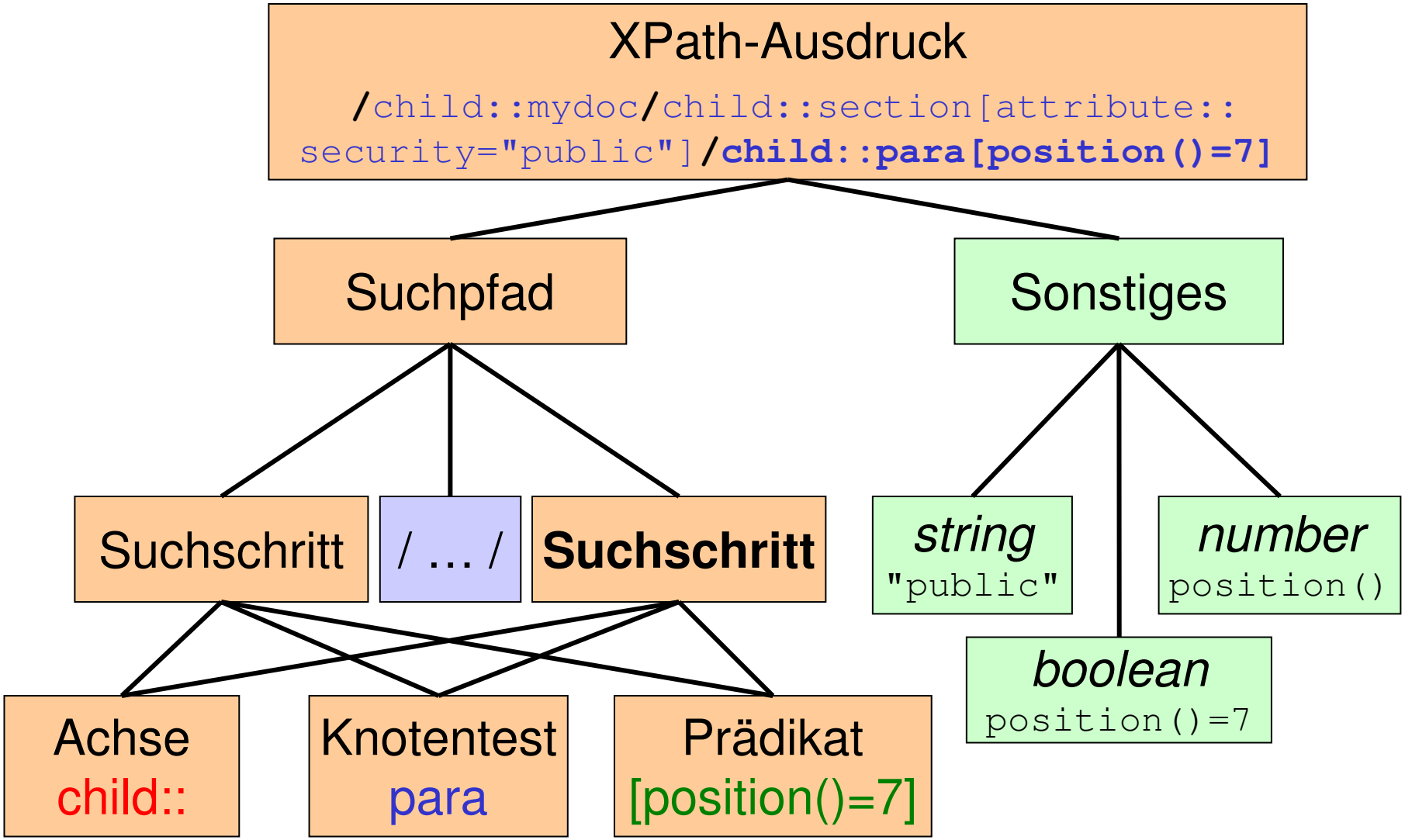
© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke



© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke: Beispiel



© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Bildung der Knotenmenge

- Prinzip der Teilmengenbildung:
  - Die Auswertung erfolgt immer von links nach rechts.
  - Suchschritt n+1 arbeitet mit der Knotenmenge K_n von Suchschritt n, ergibt also höchstens eine Teilmenge $K_{n+1} \subseteq K_n$

    ```
    / step1 /  step2 / …
    ```
  - Innerhalb eines Suchschritts reduziert ein Prädikat die Knotenmenge des Knotentests weiter (Filter):

    ```
    / … / step[expression] / …
    ```
  - Mehrere Prädikate hintereinander erzeugen ebenfalls verschachtelte Teilmengen. Reihenfolge beachten!

    ```
    … /step2[expr1][expr2][expr3]/ step3 / …
    ```

- Komplexe Ausdrücke:
  - Ausdrücke in Prädikaten können ihrerseits ganze Suchpfade enthalten!

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Ausdrücke

- Beispiele für einen Suchschritt:

`child::para`	Alle para-Kindelemente des Kontextknotens K
`child::*`	Alle Kind-Elementknoten des Kontextknotens
`child::text()`	Alle Kind-Knoten vom Texttxp von K.
`child::node()`	Alle Kind-Knoten von K, unabhängig vom Typ
`descendant::*`	Alle Abkömmlinge des Kontextknotens K
`/`	*document root* (R)
`child::para[position()=1]`	Der erste Kind-Elementknoten namens „para“ des Kontextknotens.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Achsen

- XPath-Ausdrücke suchen ihre Knoten entlang sogenannter „Achsen“. Es gibt **13 Achsen**.

- Die wichtigsten lassen sich in intuitiver Weise abkürzen:

self	Nur der Kontextknoten K selbst
child	Default-Achse, daher darf der Achsenteil child:: eines Suchpfades einfach fehlen.
descendant	Alle Nachkommen-Knoten von K
descendant-or-self	K und alle Nachkommen
parent	Der Elternknoten von K
ancestor	Alle Vorfahren-Knoten von K
ancestor-or-self	K und alle Vorfahren-Knoten

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Achsen

- Weitere Achsen:

following-sibling	Alle nachfolgende „Geschwister“-Knoten von K. Leer falls K vom Typ A oder N.
following	Alle nachfolgenden Knoten, außer Kindknoten, Attribut- sowie Namensraumknoten.
preceding-sibling	Alle vorangehenden „Geschwister“-Knoten von K. Leer falls K vom Typ A oder N.
preceding	Alle vorangehende Knoten, außer Ahnenknoten, Attribut- sowie Namensraumknoten.
attribute	Abgekürzt einfach durch Präfix „@“ child::para[attribute::type="warning"] entspricht para[@type="warning"].
namespace	Alle Namensraumknoten des Kontextknotens. Leer falls K selbst vom Typ A oder N

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Knotentests

- Jede Achse besitzt einen Hauptknotentyp (*principal node type*)
  - Fall A: *attribute*
  - Fall N: *namespace*
  - sonst: *element*

- Namespace-Behandlung:
  - Besteht ein Knotentest aus einem *QName,* ist er genau dann wahr,

    wenn der aktuelle Knoten vom Hauptknotentyp der Achse ist

    und sein Name (*expanded-name*) mit dem expandierten (!) *QName* übereinstimmt.

- Erfüllt kein Knoten entlang der aktuellen Achse den Test, wird die leere Menge (von Knoten) zurückgegeben.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Knotentests

- Eingebaute Knotentests:

*	alle Knoten des Hauptknotentyps der Achse
text()	alle Knotentypen T der Achse
comment()	alle Knotentypen C der Achse
processing-instruction()	alle Knotentypen P der Achse
node()	alle Knoten der Achse.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Prädikate

- Mit Achse und Knotentest eines XPath-Ausdrucks wurde bereits eine Knotenmenge gebildet. Diese kann mit einem Prädikat weiter eingeschränkt werden (**Filter**).
- Dazu wird für jeden Knoten der Knotenmenge der Ausdruck des Prädikats ermittelt. Ist er wahr, gelangt der Knoten in die neue Knotenmenge, sonst nicht.
- Dabei ist jeweils *context size* die Anzahl Knoten der (alten) Knotenmenge, *context position* die „Entfernung“ des Knotens entlang der aktuellen Achse zum Kontextknoten.
- Numerische Angaben im Ausdruck werden zu „wahr“, wenn sie mit *context position* übereinstimmen. Das erklärt folgende Kurzschreibweise:

  para[3] ist äquivalent zu para[position()=3]

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Kurzschreibweisen

Explizite Angabe	Kurzform
child::	(kann entfallen)
self::node()	.
parent::node()	..
/descendant-or-self::node()/	//
attribute::	@

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Kurzschreibweisen

Beispiele:

../title Alle „title“-Knoten des Elternknotens von K

.//para Alle „para“-Abkömmlinge von K
(kann K selbst einschließen)

para[5][@type="warning"]
Das fünfte „para“-Kindelement von K,falls es ein Attribut „type“ mit Wert „warning“ besitzt.

para[@type="warning"][5]
Das fünfte „para“-Kindelement von K,
das ein Attribut „type“ mit Wert „warning“ besitzt.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Operatoren in XPath-Ausdrücken

- Ausdrücke in XPath können zusammengesetzt sein, verknüpft mit Operatoren.
- Operator **|**
  - Vereinigungsmenge der Ergebnisse zweier Ausdrücke, die eine Knotenmenge als Ergebnis haben.
- Operatoren **and**, **or**
  - verknüpfen Ausdrücke mit booleschem Ergebnis
- Operatoren **=**, **!=**, **<**, **<=**, **>**, **>=**
  - Unterschiedliche Vergleiche sind möglich, mit impliziten Typkonvertierungsregeln, die im Detail in der XPath-Spezifikation nachzulesen sind (Kap. 3.4)

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Vorbemerkungen:*
  - Ziel an dieser Stelle ist eine Zusammenstellung der vorhandenen Funktionen und ihre ungefähre Verwendung, damit eine rasche Orientierung bzw. ein gezieltes Nachschlagen möglich wird.
  - Hier ist nicht genug Raum für alle Einzelheiten der Spezifikationen. Lesen Sie bei Bedarf die Details in den Spezifikationen nach! (Kap. 4)
  - Auf den Mechanismus der XPath-Erweiterungen wird hier nicht eingegangen. Er erfolgt grundsätzlich auf der Ebene der XPath nutzenden Anwendung wie etwa XSLT und besteht etwa aus dem Hinzufügen weiterer Funktionen.

    VORSICHT: Erweiterungen führen leicht zu Plattform-abhängigen Implementierungen!

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:* Funktionen auf Knotenmengen (*node set functions*)

`number last()`

Rückgabewert = *context size*

`number position()`

Rückgabewert = *context position*

`number count(node-set)`

Rückgabewert = Anzahl Knoten in der Knotenmenge

`node-set id(object)`

Liefert alle Knoten, deren ID-Attribute einem der Stringwerte des übergebenen Objekts entsprechen. Dieses Objekt kann ein einfacher String sein, eine Liste von Tokens oder auch eine Knotenmenge, wobei die Stringwerte der Knoten die Liste der Tokens ergeben.

Bemerkung: id() funktioniert ohne DTD nicht, denn nur aus der DTD erhält der Parser die Information, welche Attribute „ID“s sind.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:* Funktionen auf Knotenmengen (*node set functions*)

```
string local-name(node-set?),
string namespace-uri(node-set?),
string name(node-set?)
```

Liefert *local-name* bzw. *namespace URI* bzw. QName (I.d.R. gleich *expanded name*) des *expanded-name* des ersten Knoten der übergebenen Knotenmenge (in Dokumentenreihenfolge) bzw. des Kontextknotens.

Einzelheiten ggf. in den Spezifikationen nachlesen, Kap. 4.1.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:* String-Funktionen

`string string(object?)`

node-set: Stringwert des ersten Knoten in Dokumentenreihenfolge

number: *NaN*, *Infinity*, *-Infinity*, *0*, bzw. die Dezimaldarstellung

boolean: *true* bzw. *false* (als Strings)

andere: erfordert Umwandlung in *string*, kontextabhängig.

default: Stringwert des Kontextknotens.

`string concat(string, string, string*)`

Selbsterklärend

`boolean starts-with(string, string)`

true wenn das erste Argument mit dem zweiten beginnt

`boolean contains(string, string)`

true wenn das erste Argument das zweite enthält

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:* String-Funktionen

`string substring-before(string, string),`

`string substring-after(string, string)`

Liefert den Teilstring des ersten Arguments vor bzw. nach dem ersten Auftreten des zweiten Arguments. Beispiel:

`substring-before("1999/04/01","/")` liefert `"1999"`

`string substring(string, number, number?)`

Liefert den Teilstring des ersten Arguments ab der vom zweiten Argument angegebenen Position bis zum Ende bzw. mit der vom dritten Argument angegebenen Länge. Beispiel:

`substring("12345",2,3)` liefert `"234"`

VORSICHT: Hier wird ab 1 gezählt, nicht ab 0 wie in einigen anderen Sprachen.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:* String-Funktionen

`number string-length(string?)`

Selbsterklärend, wirkt per default auf den Stringwert des Kontextknotens

`string normalize-space(string?)`

Normiert den Argument-String analog zur XML-Normierungsregel für die Werte von NMTOKEN-Attributtypen, wirkt per default auf den Stringwert des Kontextknotens.

`string translate(string, string, string)`

Ersetzt Zeichen des ersten Arguments bzw. entfernt sie. Kommt ein Zeichen im zweiten Argument vor, wird es ersetzt durch das an gleicher Stelle im dritten Argument stehende Zeichen. Fehlt dieses, wird das Zeichen entfernt. Analog Unix-Tool bzw. Perl-Funktion „tr“.
Beispiel: `translate("abCdE","abE","AB")` liefert `"ABCd"`

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:* Boolesche Funktionen

boolean boolean(object)

number: *true*, wenn Zahl <> 0 und <> NaN.

string: *true*, wenn seine Länge > 0 ist.

node-set: *true*, wenn die Menge nicht leer ist.

andere: erfordert Umwandlung in *boolean*, kontextabhängig.

boolean not(boolean),

boolean true(),

boolean false()

selbsterklärend

boolean lang(string)

true, wenn der übergebene Sprachschlüssel zu dem des Attributs `xml-lang` des Kontextknotens „passt“.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:*
  Funktionen auf numerischen Variablen

number number(object?)

string: Umwandlung wie üblich, ggf. in „NaN“

boolean: Umwandlung in 1 bzw. 0 (0 entspricht *false*)

node-set: Behandlung des Stringwertes der Knotenmenge wie im Fall *string.*

andere: Erfordert kontextabhängige Typumwandlung.

number sum(node-set)

Summe der Umwandlungsergebnisse aller Stringwerte der Knoten in der Knotenmenge in Zahlen gemäß Funktion *number*.

number floor(number),

number ceiling(number),

number round(number):

Die üblichen Rundungsoperationen.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath-Funktionen

- *Core function library:*
  Funktionen auf numerischen Variablen

```
number floor(number),
number ceiling(number),
number round(number)
```

  Die üblichen Rundungsoperationen, *integer*-wertig.

- Operatoren für Zahlen
  - Zahlen entsprechen „double“ gemäß IEEE 754

    `+, -, *` Addition, Subtraktion, Multiplikation

    `div, mod` Division, Modulo
- Bemerkung

  „div“ statt „/“ vermeidet „path“-Verwechlungen.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Variablen in XPath

- Variablen?
  - XPath selbst besitzt keine Möglichkeiten zur Definition von Variablen, aber von „außen“ vorgegebene Variablen lassen sich in XPath-Ausdrücken verwenden.
- Beispiel:

```
/mydoc//title[@myattr=$mysample]
```

  Selektiert nur "title"-Elemente, deren Attribut "myattr" einen in der Variablen $mysample enthaltenen Wert besitzt.

- XML-Standards wie XSLT, die XPath verwenden, gestatten es, Variablen zu definieren.

- Methodische Grenzen:
  - Elementnamen o.ä. in XPath dürfen keine Variablen sein.
  - descendant::$myname // nicht zulässig
  - descendant::*[name()=$myname] // ok, ohne *namespace*

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Erweiterbarkeit von XPath

- Erfolgt indirekt über die Erweiterbarkeit der XPath verwendenden Sprachen.
- Beispiele:
  - XSLT
  - XPointer

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath 2.0 und XQuery

- Ausblick
  - XPath 2.0 unterstützt XML Schema-Datentypen und erleichtert damit Vergleiche und Berechnungen etwa mit Datum/Zeit-Angaben erheblich …
  - … und bietet Mengenoperationen auf Knotenmengen wie Vereinigung, Schnittmenge, Mengendifferenz.
  - XQuery 1.0 gibt nicht nur Knotenmengen, sondern ganze XML-Fragmente zurück, z.B. gebildet aus verschiedenen Bestandteilen der XML-Quelle.
  - XPath 2.0 wurde gemeinsam mit XQuery 1.0 und XSLT 2.0 entwickelt.
  - XPath 2.0 wird nicht mehr von einem einzigen Dokument beschrieben, sondern von mehreren „Baustein“-Dokumenten.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

Ausblick: XPath 2.0

- Neues in XPath 2.0
  - Sehr viel – zu viel für diese Vorstellung!
- Highlights:
  - Neuer Begriff „Sequenz“ bzw. „Feld“ (nicht mehr nur „Knotenmenge“)
  - Mengenoperatoren auf Sequenzen: union, intersect, except, …
  - Bedingte Selektionen

    for … in …

    if … then … else
  - Generalisierte Vergleiche

    instance of, treat as, cast as, castable as

    Vergleiche von Knoten, Werten, Knotenmengen
  - Neue Knotentypen: **item(), document-node()**
  - Neue Tests, incl. Selektion nach Datentyp: **element(), attribute()**
  - Datentypen: Von XML Schema, plus eigene

Quelle: http://nwalsh.com/docs/tutorials/extreme04/slides/index.html

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath 1.0: Beispiele

Kommentierte Beispiele zur Vertiefung
des Theorie-Teils

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: Beispiele

`/`

Der root-Knoten R des Dokuments.

`/mydoc`

Der Dokumentenelement-Knoten (namens „mydoc“)

`/mydoc/section`

Alle (!) `section`-Kindelemente von `mydoc`.

`/child::mydoc/child::section`

Dito, aber mit expliziten Achsenangaben.

`/mydoc/section[@security="public"]/para[7]`

Das siebte Element `para` jedes `section`-Elements, dessen Attribut `security` den Wert `public` hat.

`/child::mydoc/child::section[attribute::security="public"]/child::para[position()=7]`

Dito, aber mit expliziten Achsenangaben und ohne implizite Kurzformen.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: Beispiele

`/mydoc//*`

Alle Nachkommen von `mydoc`, einschließlich (?!) `mydoc`

`/mydoc//para`

Alle `para`-Nachkommen (Elementknoten) von `mydoc`, unabhängig von ihrer Tiefe im Dokumentenbaum.

`/mydoc//@type`

Alle Attributknoten des gesamten Dokuments namens `type`.

`/mydoc//comment()`

Alle Kommentarknoten, die von `mydoc` abstammen.

`/mydoc//text()`

Alle Textknoten, die von `mydoc` abstammen, d.h. alle *character data* des Dokuments!

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: Beispiele

`../@*`

Alle Attributknoten des Elternknotens des Kontextknotens.

`//para[footnote]/[@important]`

Alle `para`-Elementknoten mit `footnote` Kindelementen und `important` Attributen.

`section[author/qualifications[@professional]`
`[@affordable]]`

`section` Kindelemente des Kontextknotens von Autoren mit mehreren durch Attribute benannte Qualifikationen - ein Beispiel für die Möglichkeit, auch Prädikate aus eigenen Ausdrücken mit mehreren Schritten (*steps*) aufbauen zu können.

`id('A12345')/title | /mydoc/title`

Vereinigungsmenge: Alle `title`-Kindknoten des Knotens mit der angegebenen ID und alle `title`-Kindknoten des Dokumentknotens.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden

XPath: Beispiele

`//ordered-list[item[@type]/para[2]]//para`

Etwas zum Nachdenken: `para`-Nachfahren von `ordered-list` Knoten, die `item`-Kindknoten besitzen, welche ein `type` Attribut und mindestens 2 `para`-Kindknoten aufweisen.

`a//b[last()-2]`

Der drittletzte Nachfahre `b` von `a`, in Dokumentenreihenfolge.

`//list[@type='ordered']/item[1]/para[1]`

Der erste `para`-Kindknoten des ersten `item`-Kindknotens aller `list`-Knoten des Dokuments, die ein Attribut `type` mit Wert `ordered` besitzen.

© 2004, 2009 H. Werntges, SB Informatik, FB DCSM, FH Wiesbaden