XPATH (XPATH)

Lernziele
• Sie finden sich im XPath-Datenmodell zurecht. • Sie kennen alle XPath-Achsen. • Sie können Knoten und Knotenmengen mit XPath-Ausdrücken auswählen. • Sie sind in der Lage, Lokalisierungspfade in ausführlicher und abgekürzter Syntax zu formulieren. • Sie setzen das erworbene Wissen praktisch in den Übungen ein.

XPath-Datenmodell

Knotentypen

XPath operiert wie XSLT nicht direkt auf einer XML-Datei, sondern auf einem Baum bzw. den Knoten dieses Baumes. XPath kennt folgende Arten von Knoten:

- Wurzelknoten
- Elementknoten
- Textknoten
- Attributknoten
- Namensraumknoten
- Processing-Instruction-Knoten
- Kommentarknoten

Jeder Elementknoten, der zu einem Namensraum gehört, besitzt auch einen zugehörigen Namensraumknoten. Dadurch ist es z.B. in XPath möglich, alle Elemente eines bestimmten Namensraums anzusteuern. Auch Kommentare und Verarbeitungsanweisungen werden in diesem abstrakten Modell repräsentiert. Nicht abgebildet sind aber z.B. Dokumenttyp-Deklarationen und CDATA-Abschnitte.

Eltern und Kinder

Elementknoten in XML können Kindknoten haben. In XPath ist es rein formal allerdings so, dass *jeder* Knoten eine Menge von Knoten als Kinder hat. Diese *Knotenmenge* ist jedoch - außer im Fall des Elementknotens - immer leer. D.h., wenn Sie mit XPath versuchen, die Kinder eines Attributknotens auszuwählen, erhalten sie eine leere Knotenmenge zurück. Ebenso besitzen alle Knoten außer dem Wurzelknoten einen Elternknoten.

Beachten Sie:
Die Eltern-Kind-Beziehung in XPath ist nicht symmetrisch! Attributknoten und Namensraumknoten sind keine Kinder ihrer Elementknoten. Aber trotzdem

gelten die Elementknoten als Elternknoten ihrer Namensraumknoten und Attributknoten.

Zeichenkettenwerte

Jeder Knoten im XPath-Datenmodell hat einen **Zeichenkettenwert**.

Folgende Tabelle stellt die Zeichenkettenwerte der Knotentypen zusammen:

Knotentyp	**Zeichenkettenwert**
Textknoten	Der Text, wie er im XML-Dokument steht.
Kommentarknoten	Kommentartext, wie er im XML-Dokument steht.
Attributknoten	Attributwert
Elementknoten	Zusammensetzung aus allen Zeichenkettenwerten der Kindknoten
Wurzelknoten	Zusammensetzung aus allen Zeichenkettenwerten der Kindknoten

Dokumentordnung

XPath verfügt über das Konzept der *Dokumentordnung*, worunter nichts anderes zu verstehen ist als die Anordnung der Konten in der Reihenfolge, wie sie im Dokument nacheinander vorkommen. Die Dokumentordnung spielt v.a. bei der Ausgabe und Sortierung von Elementen eine Rolle.

Folgende Abbildung zeigt die Reihenfolge der Abarbeitung eines Baumes in Dokumentordnung: Beginn bei der Wurzel, dann der erste Kindknoten, dann dessen Kindknoten, dann dessen erster Kindknoten bis zu einem Blatt, anschließend jeweils eins nach oben steigend, immer die Geschwister in der beschriebenen Leseweise.

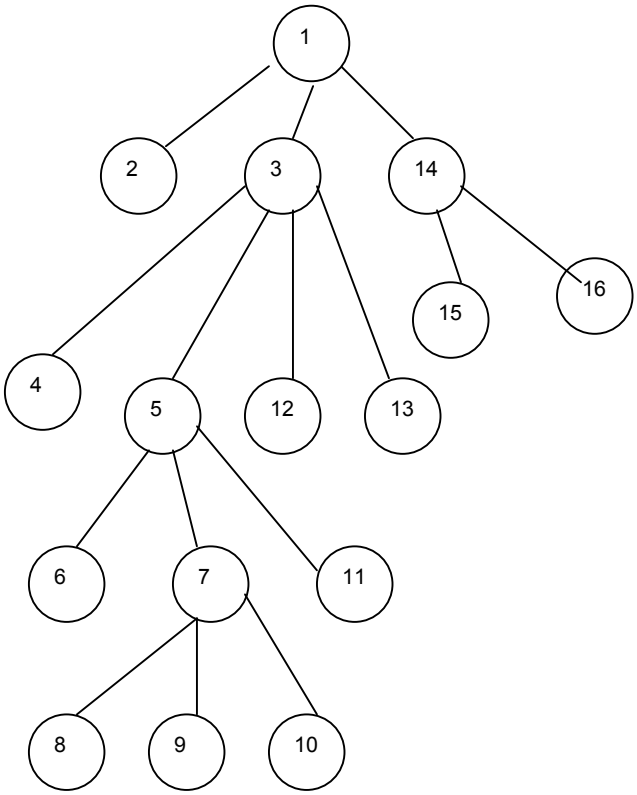


Abbildung XPATH-1: Dokumentordnung

Wurzelknoten und Wurzelelement

Sie haben schon im Zusammenhang mit den eingebauten Template Rules einiges über den Wurzelknoten erfahren.

XPath benötigt eine virtuelle Wurzel für das Dokument, den sogenannten **Wurzelknoten *(root node)***. Der Wurzelknoten selbst adressiert kein Element. Das **Wurzelelement** ist das Kind des Wurzelknotens.

Beachten Sie:
Als *Dokument-Element* (oder Wurzel) wird in der XML-Empfehlung das oberste Element bezeichnet, das in keinem anderen Element mehr enthalten ist. Wir haben hier die allgemein übliche und einleuchtendere Bezeichnung *Wurzelelement* dafür gewählt. Der Wurzelknoten in XPath entspricht der *Document Entity* in der XML-Empfehlung.

In XPath wird der Wurzelknoten mit einem einfachen Schrägstrich selektiert. Die Template Rule zur Adressierung des Wurzelknotens lautet daher

Beispiel XPATH-1: Adressierung des Wurzelknotens

```
<xsl:template match="/">...</xsl:template>
```

Wenn Sie das Wurzelelement unseres Literaturlisten-Beispiels adressieren möchten, müssen Sie folgendes notieren:

Beispiel XPATH-2: Adressierung des Wurzelelements

```
<xsl:template match="/literaturliste">...</xsl:template>
```

Achsen

Achsen sind Wege oder Pfade, entlang derer Sie durch die Baumstruktur navigieren können. Sie beginnen bei einem bestimmten Knoten, dem **Kontextknoten *(context node)***, und folgen den Achsen zwischen den Knoten.

In der Regel entspricht der Kontextknoten in XPath dem Knoten, den der XSLT-Prozessor gerade bearbeitet (gegenwärtiger Knoten).

XPath kennt 13 Achsen:

- `ancestor`
- `ancestor-or-self`
- `attribute`
- `child`
- `descendant`
- `descendant-or-self`
- `following`
- `following-sibling`
- `namespace`
- `parent`
- `preceding`
- `preceding-sibling`
- `self`

In den folgenden Grafiken ist der Kontextknoten grau markiert. Die Reihenfolge, in der die Knoten abgearbeitet werden, ist durch ihre Nummerierung verdeutlicht.

ancestor

Diese Achse wählt alle Knoten aus, die Vorfahren des Kontextknotens sind (engl. *ancestor* = dt. *Vorfahren*). Die Knoten werden in umgekehrter Dokumentordnung aufgelistet.

Der erste Knoten in der Achse ist das Elternelement des Kontextknotens, der zweite ist das Großelternelement usw. Der letzte an der Achse ist die Dokumentwurzel selbst (Wurzelknoten). Der Wurzelknoten ist der einzige Knoten, der keine Vorfahren besitzt.

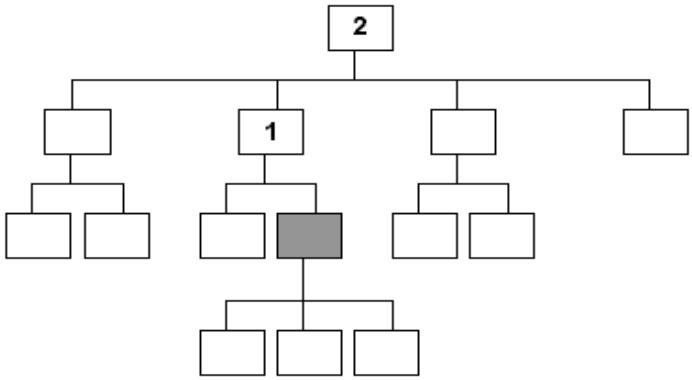


Abbildung XPATH-1: Die `ancestor`-Achse

ancestor-or-self

Diese Achse wählt dieselben Knoten wie die `ancestor`-Achse aus. Sie startet aber nicht beim Eltern-Element des Kontextknotens, sondern beginnt mit dem Kontextknoten selbst.

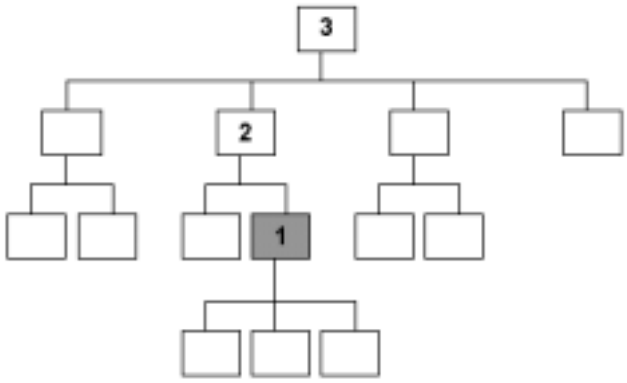


Abbildung XPATH-2: Die `ancestor-or-self`-Achse

attribute

Wenn der Kontextknoten ein Element ist, selektiert diese Achse alle seine Attribute in beliebiger Reihenfolge. Ansonsten wird nichts selektiert.

child

Diese Achse wählt alle Kinder (also die *direkten* Nachfahren) des Kontextknotens in Dokumentordnung aus.

Für alle Knoten, ausgenommen den Wurzelknoten und die Elementknoten, wird nichts ausgewählt, d.h., die selektierte Knotenmenge ist leer.

Attributknoten oder Namensraumknoten sind nämlich keine Kinder ihres Elementknotens. Lediglich Textknoten, Elementknoten, Processing-Instruction-Knoten und Kommentarknoten können als Kinder eines anderen Knotens auftreten.

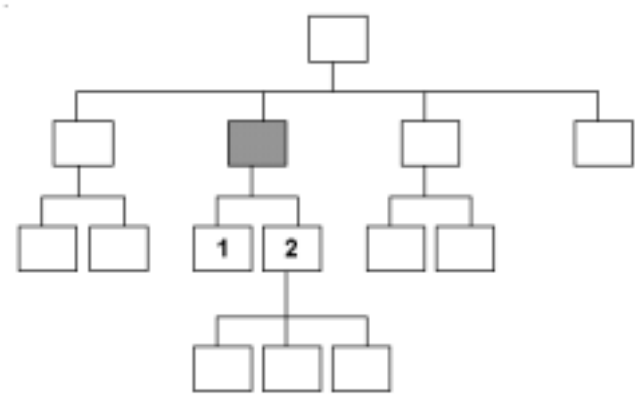


Abbildung XPATH-3: Die `child`-Achse

descendant

Diese Achse selektiert alle Nachfahren (engl. *descendant* = dt. *Nachfahre*) des Kontextknotens, d.h. alle Kinder des Kontextknotens und dessen Kinder und rekursiv so weiter. Die selektierte Knotenmenge besitzt Dokumentordnung.

Ist der Kontextknoten ein Elementknoten, so beinhaltet diese Achse alle Textknoten, Elementknoten, Processing-Instruction-Knoten und Kommentarknoten zwischen dem öffnenden und schließenden Tag des Elements, und zwar in Dokumentordnung.

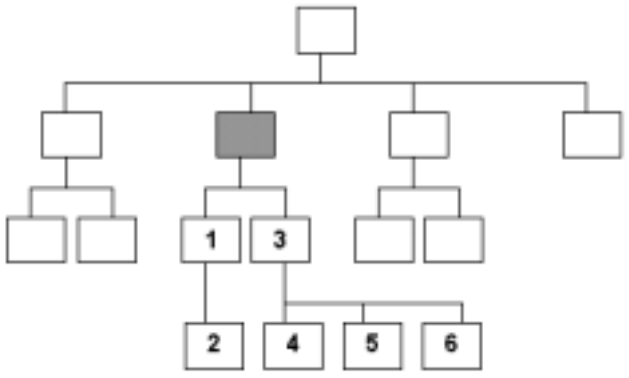


Abbildung XPATH-4: Die `descendant`-Achse

descendant-or-self

Diese Achse ist die selbe wie die `descendant`-Achse, schließt aber den Kontextknoten mit ein.

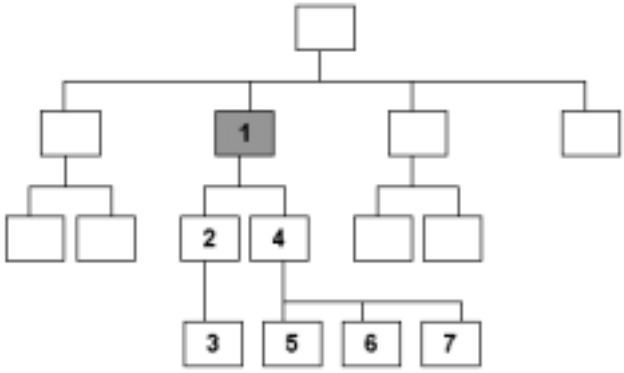


Abbildung XPATH-5: Die `descendant-or-self`-Achse

following

Diese Achse beinhaltet alle Nachbarn des Kontextknotens, die *nach* dem Kontextknoten folgen, ausgenommen die Nachfahren des Kontextknotens. Ist der Kontextknoten ein Elementknoten, so beinhaltet die Achse alle Textknoten, Elementknoten, Processing-Instruction-Knoten und Kommentarknoten, die zwischen dem öffnenden und schließenden Tag des Elements stehen, und zwar in Dokumentordnung.

Diese Achse enthält niemals Attribut- oder Namensraumknoten.

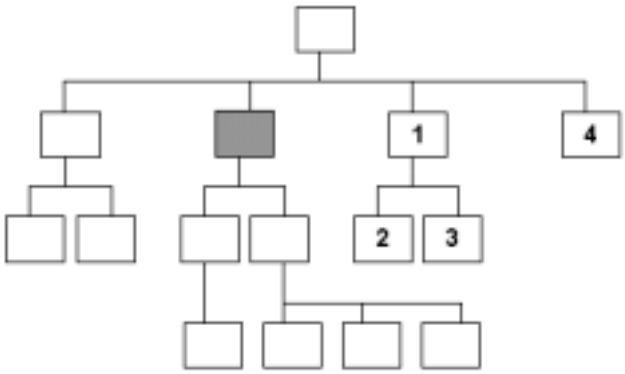


Abbildung XPATH-6: Die `following`-Achse

following-sibling

Diese Achse beinhaltet alle dem Kontextknoten folgenden Geschwister (engl. *sibling* = dt. *Geschwisterteil*) in Dokumentordnung, also alle nach dem Kontextknoten liegenden Knoten, die Kinder desselben Elternknotens sind wie der Kontextknoten.

Ist der Kontextknoten der Wurzelknoten, ein Attributknoten oder ein Namensraumknoten, bleibt diese Achse immer leer.

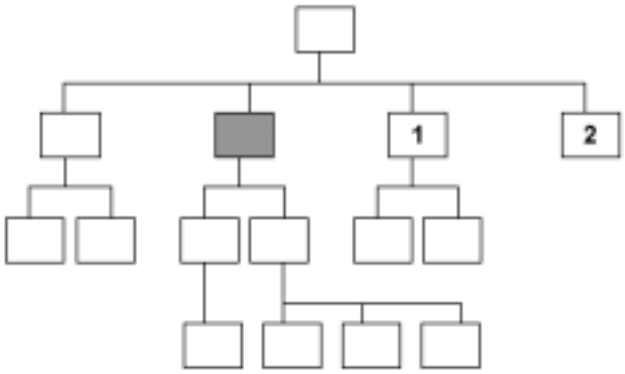


Abbildung XPATH-7: Die `following-sibling`-Achse

namespace

Ist der Kontextknoten ein Elementknoten, so selektiert diese Achse alle Namensraumknoten, die für dieses Element gültig sind. Ansonsten ist diese Achse leer. Die Namensraum-Knoten entsprechen einer Namensraumdeklaration im Element selbst oder innerhalb eines Vorgängerelements.

parent

Diese Achse wählt einen einzigen Knoten, nämlich den Elternknoten des Kontextknotens aus. Wenn der Kontextknoten der Wurzelknoten ist, ist diese Achse leer.

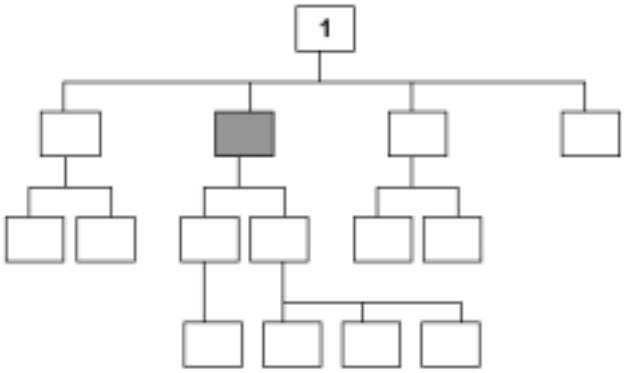


Abbildung XPATH-8: Die `parent`-Achse

preceding

Diese Achse wählt alle Knoten aus, die vor dem Kontextknoten im Dokument stehen (engl. *preceding* = dt. *vorhergehend*), ausgenommen die Vorfahren des Kontextknotens. Die Knoten werden in umgekehrter Dokumentordnung aufgelistet.

Ist der Kontextknoten ein Elementknoten, so beinhaltet diese Achse alle Textknoten, Elementknoten, Processing-Instruction-Knoten und Kommentarknoten, die zwischen dem öffnenden und schließenden Tag des Elements stehen.

Diese Achse enthält niemals Attribut- oder Namensraum-Knoten.

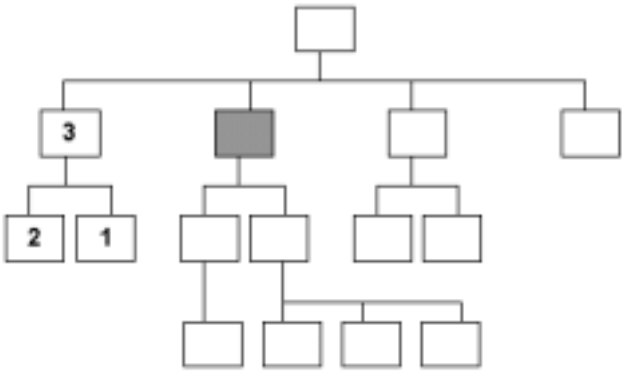


Abbildung XPATH-9: Die `preceding`-Achse

preceding-sibling

Diese Achse beinhaltet alle dem Kontextknoten vorausgehenden Geschwister (engl. *sibling* = dt. *Geschwisterteil*) in umgekehrter Dokumentordnung, also alle vor dem Kontextknoten liegenden Knoten, die Kinder desselben Elternknotens wie der Kontextknoten sind.

Ist der Kontextknoten der Wurzelknoten, ein Attributknoten oder ein Namensraumknoten, bleibt diese Achse immer leer.

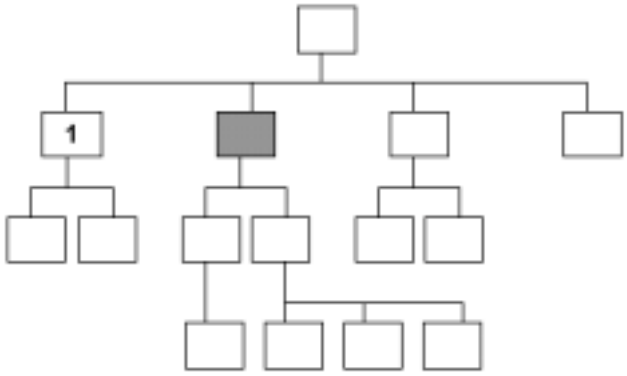


Abbildung XPATH-10: Die `preceding-sibling`-Achse

self

Diese Achse wählt einen einzigen Knoten aus und zwar den Kontextknoten selbst. Die `self`-Achse kann niemals leer sein.

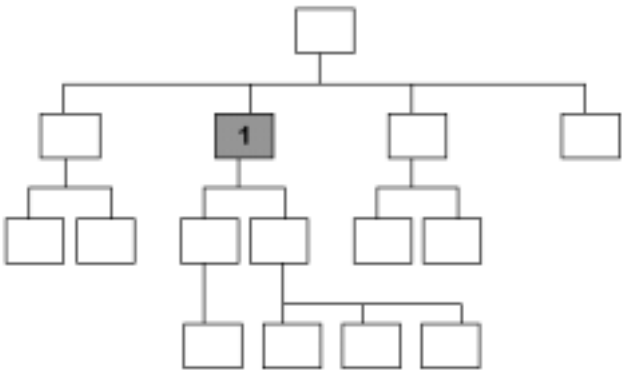


Abbildung XPATH-11: Die `self`-Achse

Knoten auswählen mit XPath

Das Herzstück von XPath sind die Konstrukte zur Auswahl von Knoten entlang der Achsen. Im Folgenden wollen wir die Navigationsausdrücke von XPath näher betrachten.

Ausdrücke

Das wichtigste syntaktische Konstrukt in XPath sind die so genannten **Ausdrücke (expressions)**. Ausdrücke verwenden Sie zum Beispiel, um Knoten im Quellbaum anzusteuern. Ausdrücke können aber auch mathematische Operationen, Vergleiche und Funktionen enthalten.

Ein XPath-Ausdruck liefert bei seiner Auswertung immer einen der folgenden **Datentypen** zurück:

- Eine Menge von Knoten, die als **Knotenmenge (*node set*)** bezeichnet wird. Sie enthält keinen, einen oder mehrere Knoten, aber keine Duplikate von Knoten; jeder Knoten kommt also genau einmal vor. Führt ein Lokalisierungspfad zu keinem Knoten, so bricht die Verarbeitung nicht ab und es wird auch kein Fehler gemeldet, sondern der XPath-Prozessor gibt einfach eine leere Knotenmenge zurück. Adressieren Sie einen einzigen Knoten, so liefert der XPath-Ausdruck eine Knotenmenge mit einem Knoten zurück.
- Einen **Wahrheitswert (*boolean*)**, der entweder wahr oder falsch (*true* / *false*) sein kann.
- Eine **Fließkommazahl (*floating point number*)**.
- Eine **Zeichenkette (*string*)**.

Lokalisierungspfade

Die zentralen XPath-Ausdrücke sind die **Lokalisierungspfade (*location paths*)**. Über Lokalisierungspfade steuern Sie bestimmte Knoten eines XML-Baums an, genauer gesagt: Sie wählen diese Knoten aus.

Einen Lokalisierungspfad kann man als Wegweiser durch das XML Dokument verstehen, der etwa sagt: "Du bist hier, jetzt musst du hier nach rechts, dann die dritte Strasse nehmen und dort das fünfte Haus auf der linken Seite ...".

Es gibt zwei Typen von Lokalisierungspfaden:

- absolute Lokalisierungspfade,
- relative Lokalisierungspfade.

Ein **absoluter Lokalisierungspfad** wird durch einen Schrägstrich (/) eingeleitet, auf den ein relativer Lokalisierungspfad folgt. In der Regel beginnt ein absoluter Lokalisierungspfad somit beim Wurzelknoten des Dokuments.

Ein **relativer Lokalisierungspfad** besteht aus einem oder mehreren *Lokalisierungsschritten* und beginnt beim Kontextknoten.

Lokalisierungsschritte

Lokalisierungsschritte (*location steps*) sind Teilwegbeschreibungen, d.h., man kann mit ihnen Schritt für Schritt durch eine Baumstruktur wandern.

Ein Lokalisierungspfad setzt sich aus mehreren Lokalisierungsschritten zusammen, die durch einen Schrägstrich voneinander getrennt sind:

Beispiel XPATH-3: Lokalisierungsschritte

```
/child::literaturliste/child::eintrag/child::titel
```

Hier gehen Sie Schritt für Schritt vom Wurzelknoten ausgehend zum `<literaturliste>`-Element, dann weiter zum `<eintrag>`-Element und letztlich zum `<titel>`-Element.

Sie können so einen Lokalisierungspfad von rechts nach links lesen. In obigem Beispiel bedeutet der Ausdruck Folgendes: "Selektiere jedes `<titel>`-Element, das Kind eines `<eintrag>`-Elements ist, welches selbst wiederum Kindelement von einem `<literaturliste>`-Element ist".

Die Auswahl von Attributen in XPath erfolgt über die `attribute`-Achse. Der folgende Ausdruck wählt z.B. das `datum`-Attribut des `<eintrag>`-Elements aus:

Beispiel XPATH-4: Adressierung eines Atributs

```
child:eintrag/attribute::datum
```

Aufbau eines Lokalisierungsschritts

Der Lokalisierungsschritt ist der wichtigste Teil des Lokalisierungspfades. Für ihn gelten bestimmte Regeln.

Zuerst wird angegeben, welche Achse man – vom Kontextknoten ausgehend – ansprechen will. Über einen der Achsennamen wird die Richtung gewählt. Danach folgt ein Kriterium zur Vorauswahl der Knoten auf dem Weg. Das ist der so genannte **Knotentest (*node test*)**.

Der Knotentest wird mit einem doppelten Doppelpunkt (`::`) an den Achsennamen angehängt:

Beispiel XPATH-5: Knotentest

```
child::autor
```

Optional können Sie weitere Auswahlkriterien mit Hilfe von so genannten **Prädikaten** angegeben. Prädikate werden in eckigen Klammern an den Knotentest angehängt:

Beispiel XPATH-6: Prädikat

```
child:autor[attribute::name='Shakespeare']
```

Der Achsenname zusammen mit dem doppelten Doppelpunkt wird **Achsenbezeichner *(axis specifier)*** genannt.

Ein Lokalisierungsschritt ist also folgendermaßen aufgebaut: Achse + Knotentest + optionales Prädikat, oder abstrakter ausgedrückt:

```
AchsenName::KnotenTest( ' [ ' Prädikat ' ] ' )
```

Beispiele:

Lokalisierungsschritt	Bedeutung
`child::eintrag`	Der Ausdruck selektiert alle <eintrag>-Elemente, die Kinder des Kontextknotens sind.
`attribute::name`	Der Ausdruck selektiert das name - Attribute des Kontextknoten.
`child:autor[attribute::name='Shakespeare']`	Der Ausdruck selektiert alle <autor>- Elemente, die ein name-Attribut mit dem Wert Shakespeare besitzen.

Knotentests

Wie schon erwähnt, ist der zweite Teil eines Lokalisierungsschritts (nach dem Achsenbezeichner) der Knotentest.

Der Knotentest wird in einem Lokalisierungsschritt genutzt, um den Typ oder den Namen des Knotens zu bestimmen, der durch den Lokalisierungsschritt ausgewählt werden soll.

Jede der 13 Achsen hat einen sogenannten *Hauptknotentyp (principal node type)*. Für die `attribute`- und `namespace`-Achse sind diese Knotentypen der Attribut- bzw. Namensraumknoten. Für alle anderen Achsen ist dieser Hauptknotentyp der Elementknoten.

Ein Knotentest überprüft nun, ob ein Knoten einen bestimmten Namen hat (mit oder ohne Namensraumpräfix) und ob er mit dem Hauptknotentyp der eingeschlagenen Achse übereinstimmt.

Ein besonderer Knotentest ist der Stern (`*`), mit dem Sie alle Knoten auswählen, die dem Hauptknotentyp der gewählten Achse entsprechen.

Beispiel XPATH-7: *-Knotentest

```
child::buch/child::*
```

wählt daher alle *Element*knoten aus, die Kinder des <buch>-Elements sind, nicht jedoch Textknoten. Für das XML-Fragment:

```
<buch>Mein
  <titel>Ein Sommernachtstraum</titel>
  <autor>Shakespeare</autor>
</buch>
```

selektiert der Ausdruck also das `<titel>`-Element und das `<autor>`-Element. Für das XML-Fragment:

```
<buch>Ein Sommernachtstraum</buch>
```

bleibt die selektierte Knotenmenge jedoch leer.

Darüber hinaus gibt es **Knotenfunktionen**, die bestimmte Knoten*typen* selektieren:

Diese Funktionen sind:

- `comment()` für Kommentarknoten
- `text()` für Textknoten
- `processing-instruction()` für Processing-Instruction-Knoten
- `node()` für beliebige Knotentypen

Beispiele:

Lokalisierungspfad mit Knotentest	**Bedeutung**
`autor`	Dieser Ausdruck selektiert alle `<autor>`-Elemente.
`attribute::autor` oder `@autor`	Dieser Ausdruck selektiert das `autor`-Attribut des Kontextknotens.
`buecher:autor`	Dieser Ausdruck selektiert alle Knoten mit dem Namen `autor` (diese können Attribut- oder Elementknoten sein), die zum Namensraum `buecher` gehören. Im Stylesheet muss eine Namenraumdeklaration der Form `xmlns:buecher="uri"` vorhanden sein.
`child::*`	Dieser Ausdruck selektiert *alle* Elementknoten , die Kinder des Kontextknotens sind aus.
`attribute::*`	Dieser Ausdruck selektiert alle Attribute des Kontextknotens

	Kontextknotens.
`child::text()`	Dieser Ausdruck selektiert alle Textknoten, die Kinder des Kontextknotens sind.
`/child::comment()`	Dieser Ausdruck selektiert alle Kommentarknoten direkt unterhalb des Wurzelknotens.
`parent::node()`	Dieser Ausdruck selektiert den Elternknoten des Kontextknotens (oder den Wurzelknoten).
`self::node()`	Dieser Ausdruck selektiert den Kontext-Knoten.

Prädikate

Prädikate sind Ausdrücke, mit deren Hilfe Sie eine weitere Spezifizierung der Knotenauswahl vornehmen können, also so etwas wie ein zusätzlicher Filter. Prädikate werden in eckigen Klammern angegeben (`[Prädikat]`). So können Sie z.B. eine Knotenmenge unter Berücksichtigung der Achsen herausfiltern und eine neue Knotenmenge erzeugen, die die zusätzlichen Bedingungen des Prädikats erfüllt.

Das Ergebnis von Prädikaten ist immer ein boolescher Wert oder es wird in einen solchen konvertiert. Nur wenn das Ergebnis des Prädikatsausdrucks `true` als Wert liefert, passiert der gegenwärtige Knoten den Test.

Der Ausdruck

Beispiel XPATH-8: Prädikate

```
child::absatz[attribute::type='beispiel'][position()=3]
```

enthält zwei Prädikate.

Das erste `[attribute::type='beispiel']` selektiert das Attribut `type`, wenn dessen Wert `beispiel` ist.

Das zweite Prädikat `[position()=3]` selektiert das dritte `<absatz>`-Element.

Dieser Ausdruck selektiert also wenn überhaupt nur ein Element, und zwar dasjenige `<absatz>`-Element, das ein Kind des Kontextknotens ist und ein Attribut namens `type` mit dem Wert `beispiel` besitzt sowie zusätzlich an dritter Position (nach Dokumentordung gezählt) von allen ausgewählten `<absatz>`-Kindern des Kontextknotens steht.

Beispiele:

Lokalisierungspfad mit Prädikat	Bedeutung
`child::buch/child::autor[child::name]/child::titel`	Der Ausdruck selektiert das <titel>-Element eines <autor>-Elements in einem <buch>-Element. Aber nur dann, wenn das <autor>-Element auch ein <name>-Element als Kindelement besitzt.
`child::autor[attribute::name='J.R.R. Tolkien']/child::titel`	Der Ausdruck selektiert das <titel>-Element eines <autor>-Elements, aber nur dann, wenn dessen name Attribut den Wert J.R.R. Tolkien aufweist.
`child::kapitel[position()=1]`	Der Ausdruck selektiert das erste <kapitel>-Element (in Dokumentordnung gezählt) des Kontextknotens.
`child::autor[*]`	Der Ausdruck selektiert alle <autor>-Elemente des Kontextknotens, die ein Kindelement besitzen.

Logische Vergleichsoperatoren in Prädikaten

Operator	Bedeutung	Beispiel
`and`	Logisches UND	`child::autor[attribute::id and attribute:name]`
`or`	Logisches Oder	`child::autor[attribute::id and attribute:name]`
`not()`	Logische Negation	`child::autor[not(position()=1)]`
`=`	gleich	`child::autor[child::name='Grass']`
`!=`	ungleich	`child::autor[child::name!='Grass']`
`<`	kleiner als	`child::autor[attribute::id<5]`
`<=`	kleiner oder gleich	`child::autor[attribute::id<=5]`
`>`	größer als	`child::autor[attribute::id>5]`
`>=`	größer oder gleich	`child::autor[attribute::id>=5]`

Zusätzlich bietet XPath folgende mathematische Operatoren:

mathematische Operatoren

Operator	Bedeutung	Beispiel
`+`	Addition	`child::autor[position()=5+1]`
`-`	Subtraktion	`child::autor[position()=last()-1]`
`*`	Multiplikation	`child::autor[position()=last()*2]`
`div`	Divison	`child::autor[position()=last() div 2]`
`mod`	Liefert den Rest einer ganzzahligen Division	`child::autor[position()=last() mod 2]`

Abgekürzte XPath-Syntax

Je nach Knoten, die Sie auswählen möchten, können Ihre Lokalisierungspfade sehr lange Ausdrücke produzieren:

Beispiel XPATH-9: Komplexer Lokalisierungspfad

```
/child::buch/child::kapitel[position()=5]/child::sektion
[position()=2]
```

Dieser Ausdruck wählt das zweite `<sektion>`-Element des fünften `<kapitel>`-Elements des Wurzelelements `<buch>` aus.

Diese lange Ausdrucksweise wird in XPath als **ausführliche Syntax (*unabbreviated syntax*)** bezeichnet.

Sie sehen: Es kann wirklich unübersichtlich und auch mühsam sein, solche Ausdrücke immer wieder notieren zu müssen. XPath kennt deshalb auch eine weniger lange Variante, die als **abgekürzte Syntax (*abbreviated syntax*)** bezeichnet wird und die wir in unseren Literaturlisten-Beispielen auch bereits intuitiv genutzt haben.

Obiges Beispiel wird in der abgekürzten Syntax folgendermaßen notiert:

Beispiel XPATH-10: Abgekürzte Syntax

```
/buch/kapitel[5]/sektion[2]
```

Wie Sie sehen, ist der Ausdruck nun um einiges kürzer und vor allen Dingen lesbarer.

Folgende Abkürzungen, die Ihnen die Notation von Lokalisierungspfaden erleichtern, sollten Sie sich angewöhnen:

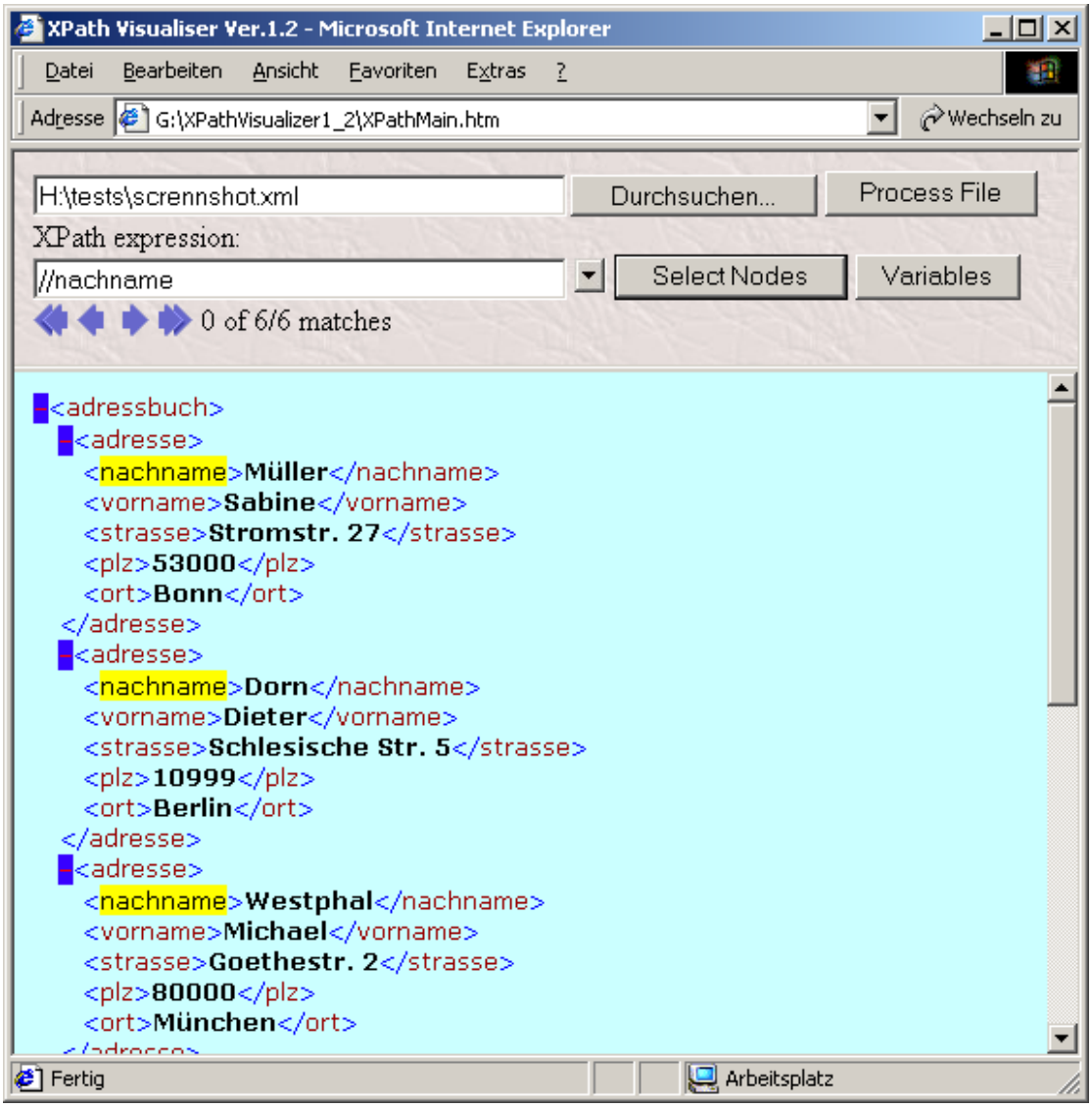
- Der Achsenbezeichner `child::` kann in einem Lokalisierungsschritt einfach ausgelassen werden, so wie wir es in den meisten Beispielen ja bereits getan haben. Statt `child::autor/child::titel` können Sie einfach `autor/titel` schreiben.
- Über zwei doppelte Schrägstriche (`//`) gefolgt von einem Knotentest (also z.B. `//autor`) am Beginn eines absoluten Lokalisierungspfades, können Sie alle Elemente in einem Dokument (unabhängig von der Strukturtiefe!) selektieren, die dem Knotentest genügen. Formal gesehen ist `//` also eine Abkürzung für `/descendant-or-self::node()/`.
- Die doppelten Schrägstriche können Sie auch relativ nutzen, d.h. innerhalb eines Lokalisierungspfades. So selektiert der Ausdruck `autor//titel` alle Nachfahren des `<autor>`-Elements, die `<titel>`-Elemente sind, und ist somit eine Abkürzung für

  `child::autor/descendant-or-self::node()/child::titel`.
- Der Punkt (`.`) ist die Abkürzung für den Ausdruck `self::node()`, der den Kontextknoten selektiert. So selektiert der Ausdruck `./autor` alle `<autor>`-Elemente, die Kinder des Kontextknotens sind.
- Zwei Punkte sind die Abkürzung für `parent::node()`. Mit `../autor` klettern Sie also ausgehend vom Kontextknoten eine Ebene nach oben (zum Elternknoten des Kontextknotens) und von dort wieder hinunter zum Kindelement `<autor>`. Dadurch haben Sie z.B. innerhalb der Struktur `<buch><autor/></titel></buch>` Zugriff auf das `<autor>`-Element, wenn der Kontextknoten der `<titel>`-Knoten ist.
- Das `@` ist eine Abkürzung für den Achsenbezeichner `attribute::`. Somit ist `autor[@name='Grass']` die Kurzform für `child::autor[attribute::name='Grass']`. Mit `autor/@*` wählen Sie alle Attribute des `<autor>`-Elements.
- Statt `autor[position()=5]` können Sie auch einfach `autor[5]` verwenden.

TOOL-TIPP

XPath Visualizer *ist ein in den IE integrierbares Tool, mit dem sie sich obige XPath-Ausdrücke hervorragend visualisieren lassen können. Zudem hilft es Ihnen beim Ausprobieren und Testen komplexerer XPath-Ausdrücke.*

URL:
`http://www.vbxml.com/xpathvisualizer/`



Beispiel XPATH-11: XPath Visualizer

Funktionen

XPath und XSLT verfügen zusammen über eine Reihe von sogenannten Kernfunktionen, die sehr nützlich und teilweise unerlässlich sind.

In der folgenden Tabelle bezeichnet ? ein optionales Argument einer Funktion und * ein Argument, das beliebig oft auftauchen kann.

Funktionen

Name	Rückgabewert	Rückgabetyp
Knotenmengen-orientierte Funktionen		
`last()`	Anzahl (üblicherweise von Elementen in einem Kontext)	`string`
`position()`	Kontext-Position	`number`
`count(node-set)`	Anzahl der Knoten innerhalb des `node-set`	`number`
`id(object)`	Element mit dem Attribut `id` vom Wert `object`	`node-set`
`local-name(node-set?)`	Lokaler Bestandteil des ersten Knotennamens in `node-set` (ohne Namensraum-Präfix)	`string`
`namespace-uri(node-set?)`	Namensraum-Bezeichner des ersten Knotennamens in `node-set` (ohne lokalen Bestandteil)	`string`
`name(node-set?)`	Namensraum-Bezeichner (falls vorhanden) und lokaler Bestandteil des ersten Knotennamens in `node-set`; normalerweise der im XML-Dokument verwendete Name	`string`
Zeichenorientierte Funktionen		
`string(object?)`	Konvertiertes Objekt als Zeichenkette	`string`
`concat (string, string, string*)`	Die Aneinanderreihung der Argumente	`string`
`starts-with(string, string)`	Wahr, wenn die erste Zeichenkette mit der zweiten beginnt (sonst unwahr).	`boolean`
`contains(string, string)`	Wahr, wenn die erste Zeichenkette die zweite beinhaltet (sonst unwahr)	`boolean`

`substring-before(string, string)`	Der Teil der ersten Zeichenkette, der dem Teil vorausgeht, den die zweite bezeichnet	`string`
`substring-after(string, string)`	Der Teil der ersten Zeichenkette, der dem Teil folgt, den die zweite bezeichnet	`string`
`substring(string, number, number?)`	Der Teil der ersten Zeichenkette, der an der Stelle beginnt, die das zweite Argument bezeichnet, bis zu der Stelle, die das dritte angibt; das erste Zeichen wird durch 1 bezeichnet	`string`
`string-length(string?)`	Anzahl der Zeichen einer Zeichenkette	`string`
`normalize-space(string?)`	Die Zeichenkette ohne "überflüssigen" Leerraum: Tabulator plus Leerzeichen plus doppelter Return werden zu einem Leerzeichen	`string`
`translate(string, string, string)`	Die Zeichenkette des ersten Arguments, mit den Zeichen des zweiten durch die des dritten ersetzt	`string`
Boolesche Funktionen		
`boolean(object)`	Das in einen booleschen Wert konvertierte Argument	`boolean`
`not(object)`	Wahr, wenn das Argument unwahr ist (sonst unwahr)	`boolean`
`true()`	Wahr	`boolean`
`false()`	Unwahr	`boolean`
`lang(string)`	Wahr, wenn die Sprache des gegenwärtigen Knotens (oder des nächsten Vorfahren, in dem die Sprache definiert ist) mit dem Wert des Arguments übereinstimmt (sonst unwahr)	`boolean`
Numerische Funktionen		
`number(object?)`	Der in `number` konvertierte Wert des Objekts (Zeichenkette, boolescher Wert, Knotenmenge oder Objekt); fehlt das Argument, gilt der gegenwärtige Knoten	`number`
`sum (node-set)`	Die Summe aus den Werten, die sich daraus ergeben, die Zeichenkettenwerte der einzelnen Knoten des `node-set` zu `number` zu konvertieren (auch Kommentare und Verarbeitungsanweisungen, als	`number`

	Kindelemente eines Elements)	
`floor(number)`	Die nächstkleinere Ganzzahl von `number` (untere Gauss-Klammer)	`number`
`ceiling(number)`	Die nächstgrößere Ganzzahl von `number` (obere Gauss-Klammer)	`number`
`round(number)`	Die dem Argument nächstgelegene Ganzzahl (wenn das zwei sind: die größere)	`number`