Tutorial zu Kapitel WT:III

III. Dokumentsprachen

- HTML
- CSS
- XML-Grundlagen
  - XML-Dokumentstruktur
  - Document Type Definition (DTD)
  - Namespaces
- XML-Schema
- XSL-Familie
  - XPath
  - XSLT

Die nach folgenden Erklärungen basieren auf W3C XPath Version 1.0. Vereinfachungen an verschiedenen Stellen sollen das grundsätzliche Verständnis erleichtern.

XPath Lokalisierungspfade

- Ein Lokalisierungspfad spezifiziert eine eventuell leere Liste (Menge) von Knoten in einem XML-Dokument.
- Ein Lokalisierungspfad setzt sich aus aufeinander folgenden Lokalisierungsschritten *(Location steps)* zusammen.
  Lokalisierungsschritte werden durch Schrägstriche *(Slashes)* getrennt:

  absoluter Pfad: /*Schritt_1*/ ... /*Schritt_i* / ... /*Schritt_k*

  relativer Pfad: *Schritt_1*/ ... /*Schritt_i* / ... /*Schritt_k*

- Die Auswertung eines absoluten Lokalisierungspfades beginnt beim Wurzelknoten (Dokumentknoten).

  Die Auswertung eines relativen Lokalisierungspfades beginnt bei einem vorgegebenen Knoten.
- Lokalisierungspfade werden Schritt für Schritt von links nach rechts ausgewertet.
- Die Ergebnisknoten des letzten Schrittes werden nacheinander als Kontextknoten des nächsten Schrittes verwendet, Die Teilergebnisse werden zusammengefasst.

XPath Lokalisierungspfade

Allgemeine Form eines Lokalisierungsschritts

. . . / *Achse* : : *Knotentest* [*Prädikat*] / . . .

- Achse

  legt den zu prüfenden Bereich fest ausgehend vom Kontextknoten,

- Knotentest

  schränkt die Knoten im zu prüfenden Bereich grob ein,

- Prädikat

  erlaubt eine detaillierte Beschreibung der gewünschten Eigenschaften der zu selektierenden Knoten und schränkt so die Knotenmenge weiter ein.

Wird die Achse nicht explizit angegeben, ist die Kindachse `child` gemeint. Wird kein Prädikat angegeben, so gehören alle Knoten zur Ergebnisliste, für die der Knotentest zutrifft.

XPath Lokalisierungspfade

Auswertungsstrategie für Lokalisierungspfade

- Ist der Lokalisierungspfad absolut, so wird der Dokumentknoten als Startknoten gewählt. Ist der Lokalisierungspfad relativ, so wird der vorgegebene Kontextknoten als Startknoten gewählt. Der Startknoten wird als Ergebnis des 0.ten Lokalisierungsschrittes aufgefasst.
- Der Reihe nach werden für jeden Knoten der Ergebnisliste des letzten Lokalisierungsschrittes die folgenden Schritte ausgeführt:
  - Der Knoten wird als Kontextknoten für die Auswertung des nächsten Lokalisierungsschrittes gewählt.
  - Ausgehend vom Kontextknoten wird die Achse ausgewertet. Die Achse legt eine Knotenliste (Menge) fest.
  - Für alle Knoten dieser Knotenliste wird der Knotentest durchgeführt. Besteht der Knoten den Knotentest nicht, so wird er verworfen.
  - Für alle verbleibenden Knoten wird das Prädikat geprüft. Trifft das Prädikat auf den Knoten nicht zu, so wird er verworfen.
  - Die übrig gebliebenen Knoten bilden die Ergebnisliste des Lokalisierungsschrittes für diesen Kontextknoten.
- Die Ergebnislisten des Lokalisierungsschrittes für die Kontextknoten werden zu einer Liste zusammengefasst.

Knoten, die in mehreren Teilergebnislisten auftreten, treten entsprechend mehrfach in der Ergebnisliste auf.

XPath Lokalisierungspfade

Evaluation von Lokalisierungspfaden

Algorithm: `evaluate-xpath`

Input: `path`. Auszuwertender Lokalisierungspfad.
`node`. Kontextknoten für relativen Pfad, sonst `null`.

Output: Liste selektierter Knoten.

```
evaluate-xpath(path, node)

1.  IF (path is absolute )
      THEN resultList = [rootNode]; // Wurzelknoten des Dokumentes
      ELSE resultList = [node]; // angegebener Kontextknoten aus Dokument
2.  FOREACH step IN path DO // von links nach rechts
      oldResultList = resultList; // letzte Ergebnisliste verwenden
      resultList = [ ]; // Ergebnisliste initialisieren: leere Liste
      FOREACH contextNode IN oldResultList DO
        resultList =append(resultList, evaluate-xstep(step, contextNode));
    ENDDO;
3.  RETURN(resultList);
```

XPath Lokalisierungspfade

Evaluation von Lokalisierungsschritten: Algorithmisierung

Algorithm: `evaluate-xstep`

Input: `step`. Auszuwertender Lokalisierungsschritt.
`contextNode`. Kontextknoten für Lokalisierungsschritt.

Output: Liste selektierter Knoten.

```
evaluate-xstep(step, contextNode)
  1.   resultList = [ ]; // Ergebnisliste initialisieren: leere Liste
  2.   FOREACH node IN nodesInAxis(step, contextNode) DO
                          // in (bzw. in umgek.) Dokumentreihenfolge
         IF ( nodeTest(step, node) ) // Knotentest durchführen
           THEN
             IF ( predicateTest(step, node) ) // Prädikattest durchführen
               THEN resultList =append(resultList, [node]);
       ENDDO;
  3.   RETURN(resultList);
```

XPath Lokalisierungspfade

Achsen in Lokalisierungsschritten: Algorithmisierung

Insgesamt gibt es 13 Achsen:

- Vorwärtsachsen

  Knoten, die zu diesen Achsen gehören, werden in Dokumentreihenfolge betrachtet.

- Rückwärtsachsen

  Knoten, die zu diesen Achsen gehören, werden in umgekehrter Dokumentreihenfolge betrachtet.

Achsen, die immer höchstens einen Knoten enthalten, also `parent` und `self`, können als Vorwärtsachsen oder Rückwärtsachsen gesehen werden.

XPath Lokalisierungspfade

Achsen in Lokalisierungsschritten: wichtige Vorwärtsachsen

- `self` Achse

  Kontextknoten selbst

- `child` Achse

  Kindknoten des Kontextknotens in Dokumentreihenfolge

- `descendant` Achse

  Nachfolgerknoten (Kindknoten, deren Kindknoten,...) des Kontextknotens in Dokumentreihenfolge

- `descendant-or-self` Achse

  Nachfolgerknoten (Kindknoten, deren Kindknoten,...) des Kontextknotens und der Kontextknoten in Dokumentreihenfolge

- `attribute` Achse

  Attributknoten des Kontextknotens in Dokumentreihenfolge

XPath Lokalisierungspfade

Achsen in Lokalisierungsschritten: weitere Vorwärtsachsen

- `following-sibling` Achse

  nachfolgende Geschwisterknoten des Kontextknotens in Dokumentreihenfolge

- `following` Achse

  Knoten, die dem Kontextknoten im Dokument folgen (nicht geschachtelt), in Dokumentreihenfolge

- `namespace` Achse (deprecated as of XPath 2.0)

  Namensraumdeklarationsknoten des Kontextknotens in Dokumentreihenfolge

XPath Lokalisierungspfade

Achsen in Lokalisierungsschritten: wichtige Rückwärtsachsen

- `parent` Achse

  Elterknoten des Kontextknotens

- `ancestor` Achse

  Vorgängerknoten (Elterknoten, dessen Elterknoten,...) des Kontextknotens in umgekehrter Dokumentreihenfolge

- `ancestor-or-self` Achse

  Vorgängerknoten (Elterknoten, dessen Elterknoten,...) des Kontextknotens und der Kontextknoten in umgekehrter Dokumentreihenfolge

XPath Lokalisierungspfade

Achsen in Lokalisierungsschritten: weitere Rückwärtsachsen

- `preceding-sibling` Achse

  vorhergehende Geschwisterknoten des Kontextknotens in umgekehrter Dokumentreihenfolge

- `preceding` Achse

  Knoten, die dem Kontextknoten im Dokument vorausgehen (nicht geschachtelt), in umgekehrter Dokumentreihenfolge

XPath Lokalisierungspfade

Knotentests in Lokalisierungsschritten: wichtige Tests

- `node()`

  trifft für jeden Knoten zu

- `text()`

  trifft für jeden Textknoten zu

- *NCName*

  trifft für jeden Knoten mit dem angegebenen Namen zu (Elementknoten oder Attributknoten je nach Achse, Name ohne Doppelpunkt)

- `*`

  trifft für jeden Knoten mit beliebigem Namen (Elementknoten oder Attributknoten je nach Achse) zu

XPath Lokalisierungspfade

Knotentests in Lokalisierungsschritten: weitere Tests

- `comment()`

  trifft für jeden Kommentarknoten zu

- `processing-instruction'()`

  trifft für jeden Processing-Instruction Knoten zu

- *QName*

  trifft für jeden Knoten mit dem Namen zu (Elementknoten oder Attributknoten je nach Achse, Name auch mit Qualifizierung durch Präfix für Namensraum)

- *NCName*`:*`

  trifft für jeden Knoten mit beliebigem lokalem Namen und angegebenem Präfix zu (Elementknoten oder Attributknoten je nach Achse)

- `*:`*NCName*

  trifft für jeden Knoten mit diesem lokalem Namen und beliebigem, auch fehlendem Präfix zu (Elementknoten oder Attributknoten je nach Achse)

XPath Lokalisierungspfade

Abkürzungen in Lokalisierungsschritten

- `child::`

  Angabe für `child`-Achse kann weggelassen werden

- `@`

  Abkürzung für `attribute::`
  (Nur in Verbindung mit Namentest: `@*` oder `@`*QName* erlaubt.)

- `//`

  Abkürzung für `/descendant-or-self::node()/`
  (Nicht am Ende eines Lokalisierungspfades erlaubt.)

- `.`

  Abkürzung für `self::node()`

- `..`

  Abkürzung für `parent::node()`

XPath Lokalisierungspfade

Prädikate in Lokalisierungsschritten

- Ergebnisse aus Zahlenvergleichen mit `<`, `>`, `=`, `<=`, `>=`, `!=`

  Vergleiche zwischen Zahlausdrücken, d.h. Zahlen und Funktionen mit Zahlenwert als Ergebnis
  Achtung:
  In XML-Dokumenten muss statt `<` und `<=` eine Entity verwendet werden, also `<` und `<=`

- Ergebnis von Ausdrücken, die Wahrheitswerte liefern

  z.B. Funktion mit Wahrheitswert als Ergebnis, z.B. `contains(@color,'red')`

- Ergebnisse aus Verknüpfung von Ausdrücken mit *and, or*

  Verknüpfung von Prädikaten (Tests) mittels *and* und *or* ggf. mit Klammerung

- Mehrfach-Prädikate

  Liste von Prädikaten (Tests) jeweils in eigener Klammerung [..], z.B.
  `//node[contains(@color,'red')][position()>=3]`

XPath Lokalisierungspfade

Funktionen mit Zahlenwert als Ergebnis

- `position()`

  liefert die Position des aktuell betrachteten Knotens in der Liste aller der Knoten, die beim aktuellen Kontextknoten für den Lokalisierungsschritt zu untersuchen sind. (Vorwärts- / Rückwärtsachsen unterscheiden.)
  Beispiel:

  `//service/*[position()=4]`

  `//service/*[4]` (Abkürzung)
  liefert die vierten Kindelemente von `service`-Knoten.

- `last()`

  liefert die Position letzten zu betrachteten Knotens in der Liste aller der Knoten, die beim aktuellen Kontextknoten für den Lokalisierunbgsschritt zu untersuchen sind.
  Beispiel:

  `//service/*[position()=last()]`

  `//service/*[last()]` (Abkürzung)
  liefert die letzten Kindelemente von `service`-Knoten.

XPath Lokalisierungspfade

Funktionen mit Zahlenwert als Ergebnis

- `count(`*node-set*`)`

  liefert die Anzahl der Knoten in der Knotenmenge. Eine Knotenmenge kann z.B. durch einen relativen Lokalisierungspfad angegeben werden.

  Beispiel:

  ```
  //service[count(./*)=4]
  ```

  liefert die `service`-Knoten, die genau vier Kindelemente haben. Für die Auswertung des Pfades `./*` werden als Kontextknoten jeweils die Knoten benutzt, die den Knotentest im Lokalisierungsschritt bestanden haben.

XPath Lokalisierungspfade

Funktionen mit Wahrheitswert als Ergebnis

- `contains(`*string1*`,` *string2*`)`

  prüft, ob die Zeichenkette `string2` in der Zeichenkette `string1` enthalten ist. Eine Zeichenkette kann z.B. durch einen Elementknoten oder eine Knotenmenge angegeben werden. Gemeint ist in dem Fall die Zeichenkette, die durch Aneinanderhängen der Nachfolger-Textknoten (Descendants) in Dokumentreihenfolge entsteht.

  Beispiel:

  ```
  //text()[contains(.,"XML"]
  ```

  liefert die Textknoten, die die Zeichenkette XML enthalten. Für die Auswertung des Pfades `./*` werden als Kontextknoten jeweils die Knoten benutzt, die den Knotentest im Lokalisierungsschritt bestanden haben.

Tutorial zu Kapitel WT:III

III. Dokumentsprachen

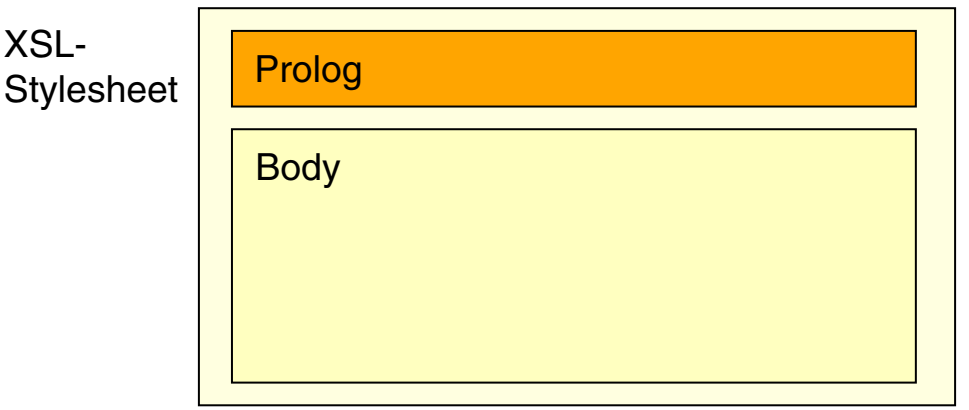
- CSS
- DTD
- Namespaces
- XML-Schema
- XSL-Familie
  - XPath
  - XSLT

Die nach folgenden Erklärungen basieren auf W3C XSLT Version 1.0. Vereinfachungen an verschiedenen Stellen sollen das grundsätzliche Verständnis erleichtern.

XSLT

Aufbau eines XSL-Stylesheets

XSL-Stylesheets sind XML-Dokumente:



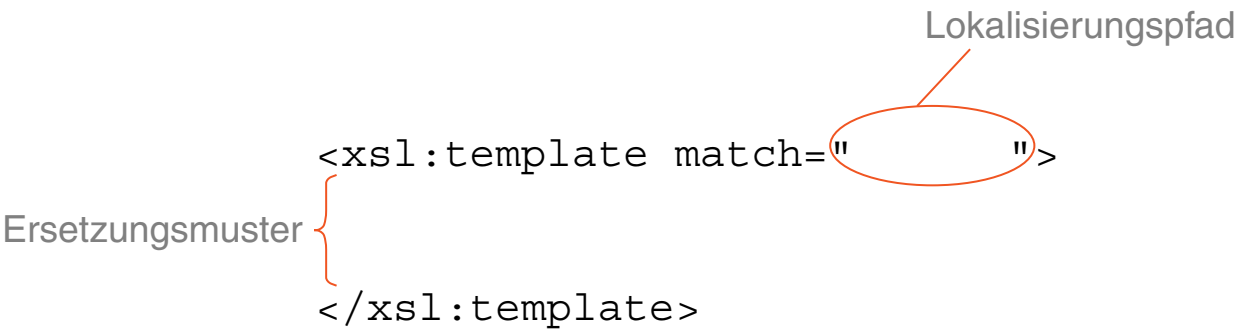

```
<?xml version="1.0"?>


<xsl:stylesheet xmlns:xsl=...>
  <xsl:template match=...>
      ...
  </xsl:template>
  ...
</xsl:stylesheet>
```

- Wurzelelement jedes XSL-Stylesheets ist das Element `<xsl:stylesheet>` oder synonym `<xsl:transform>`.
- Die Kindelemente von `<xsl:stylesheet>` bzw. `<xsl:transform>` sind Template-Regeln.
- Die Attribute von `<xsl:stylesheet>` bzw. `<xsl:transform>` sind `version="1.0"`, das die genutzte Version von XSL angibt, und die Deklaration des XSL-Namensraumes `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"`.

XSLT

Elemente eines XSL-Stylesheets

Wichtigstes Stylesheet-Element ist die Template-Regel *(Template)*:



- Eine Template-Regel matched einen Knoten n, falls n bei passender Wahl eines Kontextknotens in der vom `match`-Attribut spezifizierten Knotenmenge M enthalten ist.
- Wird ein Template auf einen Knoten n angewandt, werden die im Ersetzungsmuster vorgegebenen Aktionen durchgeführt. Dies kann eine Bearbeitung des gesamten Teilbaumes des XML-Dokuments bedeuten, der den Knoten n als Wurzel hat.
  Der Knoten n ist damit abgearbeitet.

XSLT

XSLT-Prozessor: Verarbeitungsstrategie

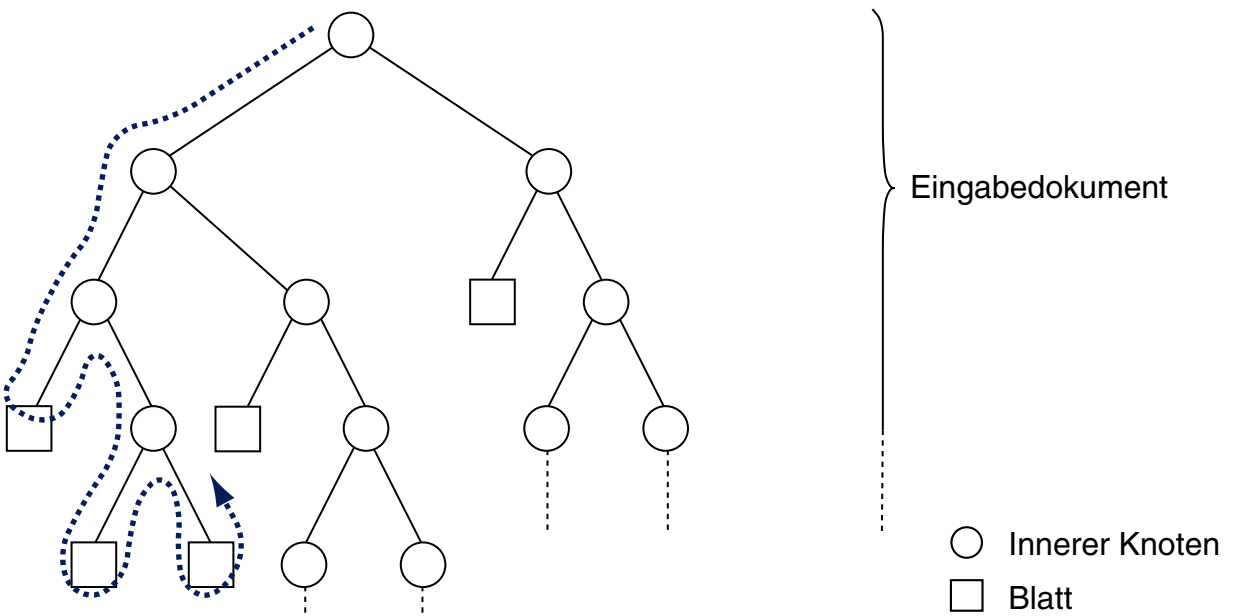
- Die Bearbeitung eines XML-Dokumentenbaumes wird durch Verarbeitung des Wurzelknotens (Dokumentknoten) gestartet.
- Für eine Liste von Knoten, die verarbeitet werden soll, werden nacheinander für jeden Knoten der Liste die folgenden Schritte ausgeführt:
  - Im XSL-Stylesheet wird nach einer für den Knoten am besten passende Template-Regel ausgewählt.
  - Ist keine solche Regel im XSL-Stylesheet vorhanden, so wird nach einer passenden Built-in-Template-Regel gesucht.
  - Wenn eine Template-Regel gefunden wurde, so wird sie angewendet. Wird keine Template-Regel gefunden, so erfolgt keine Aktion.
  - Die Bearbeitung des Knotens ist abgeschlossen.

Die Anwendung einer Template-Regel kann (z.B. durch `<xsl:apply-templates>`) die Anwendung von Template-Regeln auf weitere Listen von Knoten auslösen.

XSLT

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)

Wenn nur die Built-in-Template-Regeln vorhanden sind, durchläuft der XSLT-Prozessor den aus dem Eingabedokument erzeugten Baum ausgehend vom Wurzelknoten in Preorder-Reihenfolge.



Es werden nur Knoten in der Kindachse (Dokumentknoten, Elementknoten, Textknoten, Verarbeitungsanweisungsknoten, Kommentarknoten) bearbeitet.

XSLT

XSLT-Prozessor: Built-in-Templates

1. Built-in-Template, das die rekursive Verarbeitung der Knoten der Kindachse garantiert, falls kein matchendes Template im Stylesheet existiert:

```
<xsl:template match="*|/">
  <xsl:apply-templates select="child::node()"/>
</xsl:template>
```

2. Built-in-Template zur Ausgabe von Text- und Attributknoten:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="./>
</xsl:template>
```

3. Built-in-Template, das die Kommentare matched und ignoriert (d.h. keine Aktion für diese Knoten ausführt):

```
<xsl:template match="processing-instruction()|comment()"/>
```

XSLT

Auflösung von Konflikten für Template-Regel

1. Für jede Regel mit passendem `match` Attribut wird ihre Priorität bestimmt:
   (a) Ist der Pfad im `match` Attribut

   `child::name` oder `child::prefix:name` oder
   `attribute::name` oder `attribute::prefix:name`
   `child::processing-instruction`

   ist die Priorität 0.

   (b) Ist der Pfad im `match` Attribut

   `child::prefix:*` oder `attribute::prefix:*`

   ist die Priorität $-0,25$.

   (c) Ist der Pfad im `match` Attribut

   `child::*` oder `attribute::*` oder
   ein Knotentyp-Test in Kind- oder Attributachse (z.B. `child::node()`)

   ist die Priorität $-0,5$.

   (d) Sonst ist die Priorität $0,5$.

2. Bei mehreren Template-Regeln mit höchster Priorität wird die letzte Regel (Reihenfolge im Stylesheet) gewählt.

XSLT

Elemente in der Ersetzungsliste einer Template-Regel

0. Beliebige Textfragmente mit Auszeichnungen, deren Namen nicht aus dem XSL-Namensraum stammen, werden in das Ausgabedokument kopiert.

   Die Ersetzungsliste muss ein wohlgeformtes XML-Fragment bilden. Insbesondere müssen eventuell verwendete Namensraum-Präfixe deklariert sein.

```
<xsl:template match="/">
  <html> <head><title>Test</title></head>
  <body>
    <h2>HTML-Test-Output</h2>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisici elit,
      sed eiusmod tempor incidunt ut labore et dolore magna aliqua.
      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquid ex ea commodi consequat. Quis aute
      iure reprehenderit in voluptate velit esse cillum dolore eu
      fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non
      proident, sunt in culpa qui officia deserunt mollit anim id
      est laborum.
    </p>
  </body>
  </html>
</xsl:template>
```

XSLT

Elemente in der Ersetzungsliste einer Template-Regel (Fortsetzung)

1. Das `<xsl:text>`-Element erzeugt einen Textknoten im Ausgabedokument.

   ```
   <xsl:text> ... </xsl:text>
   ```

   Ein Text muss nicht in `<xsl:text>` eingeschlossen werden. Es ändert sich ggf. die Behandlung von Leerzeichen.

2. Das `<xsl:value-of>`-Element (vereinfacht) erzeugt Textknoten im Ausgabedokument für die Information in Knoten des Eingabedokumentes:

   ```
   <xsl:value-of select="..."/>
   ```

   Ein Textknoten wird im Ausgabedokument erzeugt, der eine textuelle Repräsentation des ersten im `select`-Attribut spezifizierten Knotens enthält. (Nur der erste Knoten führt zu einer Ausgabe, keiner der weiteren.)

3. Das `<xsl:copy-of>`-Element (vereinfacht) kopiert Knoten des Eingabedokumentes in das Ausgabedokument:

   ```
   <xsl:copy-of select="..."/>
   ```

   Die im `select`-Attribut spezifizierten Knoten werden in das Ausgabedokument kopiert, d.h. die gesamten Teilbäume des Eingabedokumentes, die die im `select`-Attribut spezifizierten Knoten als Wurzel haben.

XSLT

Elemente in der Ersetzungsliste einer Template-Regel (Fortsetzung)

4. Das `<xsl:apply-templates>`-Element (vereinfacht) dient der expliziten Verarbeitungssteuerung:

   ```
   <xsl:apply-templates select="..."/>
   ```

   Für die im `select`-Attribut spezifizierten Knoten (Kontextknoten ist Knoten auf den die Template-Regel gerade angewendet wird.) werden die Template-Regeln des XSL-Stylesheets bzw. die Built-in-Template-Regeln angewendet.

5. Das `<xsl:for-each>`-Element (vereinfacht) dient der expliziten Verarbeitungssteuerung:

   ```
   <xsl:for-each select="...">
      ...
   </xsl:for-each>
   ```

   Für die im `select`-Attribut spezifizierten Knoten (Kontextknoten ist Knoten auf den die Template-Regel gerade angewendet wird.) werden die zwischen den `<xsl:for-each>`-Tags stehenden Instruktionen angewendet.

XSLT

Elemente in der Ersetzungsliste einer Template-Regel (Fortsetzung)

6. Das `<xsl:sort>`-Element (vereinfacht) erlaubt eine Sortierung in der expliziten Verarbeitungssteuerung:

```
<xsl:apply-templates select="...">
   <xsl:sort select="..."/>
   ...
</xsl:apply-templates>
...
<xsl:for-each select="...">
   <xsl:sort select="..."/>
   ...
</xsl:for-each>
```

   Die im `select`-Attribut des `<xsl:apply-templates>`-Elementes bzw. des `<xsl:for-each>`-Elementes spezifizierten Knoten werden vor der Bearbeitung sortiert nach dem Kriterium, das im `select`-Attribut des `<xsl:sort>`-Elementes angegeben ist.

   Es können mehrere `<xsl:sort>`-Elemente verwendet werden. Das erste gibt das primäre Sortierkriterium an, das nächste das sekundäre Kriterium, usw.