

XSL Transformation (XSLT) Teil IV

Erik Wilde

26.6.2006

<http://dret.net/lectures/xml-fhnw-ss06/>

26.6.2006

XML Vorlesung FHA SS 2006

1

Übersicht

- Whitespace Handling
- Tips und Tricks
 - Mengenoperationen in XPath
- Keys

26.6.2006

XML Vorlesung FHA SS 2006

2

Probleme bei Textausgaben

- nicht immer erscheint alles wie gewünscht
 - insbesondere bei reinen Textausgaben
 - Ursache ist das *Whitespace Stripping* von XSLT
- Whitespace in XML ist der Normalfall
 - Text Nodes, die nur aus Whitespace bestehen
 - Whitespace ist Space, Tab, Newline und CR
 - XML ist oft lesbar strukturiert
 - Trennung mit Newline und/oder CR (je nach Plattform)
 - Einrückung mit Spaces oder Tabs
 - Whitespace ist oftmals semantisch leer
 - ausgenutzte syntaktische Freiheiten von XML
 - Whitespace Handling ist ein wichtiger Teil in XML

26.6.2006

XML Vorlesung FHA SS 2006

3

Whitespace Handling

- XSLT benutzt zwei XML Dokumente
 - Eingabe-Dokument und XSLT Stylesheet
- *Whitespace Stripping* vor der Verarbeitung
 - XSLT Prozessor erhält zwei Bäume
 - Resultat eines Parsers (u.U. bereits Modifikationen!)
 - Stripping findet auf beiden Bäumen statt
 - *Whitespace Text Nodes* im XSLT werden gelöscht
 - einzige Ausnahme ist das `<xsl:text>` Element
 - Whitespace Text Nodes im XML bleiben erhalten
 - ausser falls durch `xml:space` Attribut gesteuert
 - weitere Steuerung durch Stylesheet möglich
 - `<xsl:preserve-space>` und `<xsl:strip-space>`

26.6.2006

XML Vorlesung FHA SS 2006

4

Whitespace Handling der Eingabe

- Default ist das Erhalten von Whitespace Nodes
 - oftmals nicht erwünscht (viele solche Nodes!)
 - manche Parser liefern Whitespace nicht mit
 - konfigurierbar über Parser-Optionen
- XML unterscheidet die Signifikanz
 - Elemente mit `#PCDATA`: Whitespace signifikant
 - Elemente ohne `#PCDATA`: Whitespace insignifikant
 - Parser muss DTD interpretieren!
- XSLT definiert Whitespace Handling selber
 - Whitespace ist per Default signifikant
 - ausser falls von Dokument oder XSLT angegeben

26.6.2006

XML Vorlesung FHA SS 2006

5

Insignifikanter Whitespace

- `<xsl:strip-space>` listet Elementnamen
 - kann auch alle Elemente angeben ("*")
 - Whitespace Nodes in diesen Elementen verschwinden
 - sollte nicht für Elemente mit *mixed content* verwendet werden
 - Ausnahme: `xml:space="preserve"`
- `<xsl:preserve-space>` listet Elementnamen
 - Default-Verhalten in XSLT
 - Conflict Resolution mit `<xsl:strip-space>` Elementen
 - Whitespace Nodes in diesen Elementen bleiben
 - sollte für Elemente mit *mixed content* verwendet werden
 - Ausnahme: `xml:space="default"`

26.6.2006

XML Vorlesung FHA SS 2006

6

whitespace.xsl

Whitespace Beispiel (I)

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>

<xsl:template match="/">
  <xsl:apply-templates select="*" | @*"/>
  <xsl:with-param name="indent" select="1"/>
</xsl:apply-templates>
</xsl:template>

<xsl:template match="*">
<xsl:param name="indent"/>
<xsl:call-template name="spacing">
  <xsl:with-param name="indent" select="$indent"/>
</xsl:call-template>
<xsl:text>Element : </xsl:text>
<xsl:value-of select="local-name()"/>
<xsl:text>&#xa;</xsl:text>
<xsl:apply-templates select="*" | @*"/>
<xsl:with-param name="indent" select="$indent + 2"/>
</xsl:apply-templates>
</xsl:template>
```

26.6.2006 XML Vorlesung FHA SS 2006 7

whitespace.xsl

Whitespace Beispiel (II)

```
<xsl:template match="@*">
  <xsl:param name="indent"/>
  <xsl:call-template name="spacing">
    <xsl:with-param name="indent" select="$indent"/>
  </xsl:call-template>
  <xsl:text>Attribute : </xsl:text>
  <xsl:value-of select="local-name()"/>
  <xsl:text>&#xa;</xsl:text>
</xsl:template>

<xsl:template name="spacing">
  <xsl:param name="indent"/>
  <xsl:text> </xsl:text>
  <xsl:if test="$indent > 0">
    <xsl:call-template name="spacing">
      <xsl:with-param name="indent" select="$indent - 1"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>

</xsl:stylesheet>
```

26.6.2006 XML Vorlesung FHA SS 2006 8

Ausgabesteuerung im Detail

- `<xsl:output>` ermöglicht einige Steuerung
 - `method` definiert die Methode (XML, HTML, Text)
 - `version` definiert die Ausgabeformat-Version
 - `encoding` für Character Encoding der Ausgabe
 - `omit-xml-declaration` (nur bei XML)
 - `standalone` (nur bei XML)
 - `doctype-public` (bei XML und HTML)
 - `doctype-system` (bei XML und HTML)
 - `CDATA-section-elements`
 - `indent` für die Einrückung (bei XML und HTML)
 - `media-type` definiert den MIME-Type der Ausgabe

26.6.2006 XML Vorlesung FHA SS 2006 9

Identitäts-Test mit `generate-id()`

- Test auf Node Identität oft notwendig
 - z.B. Node in Variable gespeichert
 - Test auf Identität mit Variable in einer Schleife
- der direkte Vergleich ist nicht korrekt
 - aber: `$x = $y` ist ein korrekter XPath
 - er vergleicht jedoch die String Values der Knoten
 - oft richtig, aber nicht immer (schwer zu findender Fehler)
- Nodes müssen über ihre ID verglichen werden
 - `generate-id($x) = generate-id($y)`
 - vergleicht die generierten IDs
 - diese müssen identisch sein bei identischen Nodes

26.6.2006 XML Vorlesung FHA SS 2006 10

Zugriff auf Nodes über Keys

- Deklaration mit `<xsl:key>`
 - immer auf dem Top Level deklariert
 - Name, Match Pattern und Wert des Keys
 - falls Wert ein Node Set ergeben sich mehrere Keys
 - `<xsl:key name="" match="name" use="vorname"/>`
 - Hinweis an den XSLT Prozessor
 - sinnvollerweise wird ein Index aufgebaut
- Keys müssen nicht eindeutig sein
 - Eindeutigkeit muss separat sichergestellt werden
 - in XSLT: `count(key(...))` darf höchstens eins sein
 - im Schema: DTD IDs oder XML Schema Identity Constraints

26.6.2006 XML Vorlesung FHA SS 2006 11

key.xsl

Deklarieren und Verwenden eines Key

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:key name="topicid" match="topic" use="@TID"/>

  <xsl:template match="/">
    <xsl:value-of select="key('topicid','ascii')/name"/>
    <xsl:text> (</xsl:text>
    <xsl:value-of select="key('topicid','ascii')/alias"/>
    <xsl:text>), </xsl:text>
    <xsl:value-of select="key('topicid','md5')/name"/>
    <xsl:text> (</xsl:text>
    <xsl:value-of select="key('topicid','md5')/alias"/>
    <xsl:text>)</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

26.6.2006 XML Vorlesung FHA SS 2006 12

Multivalued Keys

- `<xsl:key>` ist sehr flexibel
 - ein Key kann mehrere Werte für einen Node haben
 - `<xsl:key name="" match="name" use="vorname"/>`
- wie findet man Schnittmengen von Keys?
 - generelles Problem in XSLT/XPath
 - auf Mengen gibt es nur den Union Operator (`|`)
- spezieller Code in XSLT


```
<xsl:variable name="a" select="key(...)" />
<xsl:variable name="b" select="key(...)" />
<xsl:variable name="result"
  select="$a[count(.|$b) = count($b)]"/>
```

26.6.2006

XML Vorlesung FHA SS 2006

13

Mengenoperationen

- XSLT kennt nur den Union Operator (`|`)
 - die am häufigsten gebrauchte Operation
 - aber nicht die einzig sinnvolle Operation
- Ausnutzung der Definition von XPath Prädikaten
- Verwendung von Schnittmengen
 - `$a[count(.|$b) = count($b)]`
 - selektiert Nodes, die in \$a und in \$b sind
- Verwendung von Differenzmengen
 - `$a[count(.|$b) != count($b)]`
 - selektiert Nodes in \$a, die nicht in \$b sind

26.6.2006

XML Vorlesung FHA SS 2006

14

Identifizieren von Knoten

- `generate-id()` erzeugt eine eindeutige ID
 - eine gültiger XML Name (DTD ID/IDREF)
 - kann jedem Knoten einen Namen zuweisen
- keine inverse Funktion
 - Abbildung ID auf Knoten nicht unterstützt
 - möglich durch `//*[generate-id()=$ID]`
 - extrem ineffizient
- Schlüsseldefinition für inverses Mapping
 - `<xsl:key name="id" match="*" use="generate-id()" />`
 - Benutzung einfach mit `key('id', $ID)`

26.6.2006

XML Vorlesung FHA SS 2006

15

Zusammenfassung

- XSLT Challenges
 - XPath als Grundlage der Sprache
 - XPath als eher komplexe *Expression Language*
 - funktionales Programmieren
 - Rekursion statt Iteration
 - viel Logik im Laufzeitsystem
 - Auswahl der Templates vom XSLT Prozessor

26.6.2006

XML Vorlesung FHA SS 2006

16