

XML-Praxis
Mit XSLT arbeiten

Jörn Clausen
joern@TechFak.Uni-Bielefeld.DE

Übersicht

- Kontrollstrukturen
 - Bedingungen
 - Schleifen
- named templates
- Variablen und Parameter
- Rekursion

Bedingungen

- Datum nur ausgeben, falls date-Attribut gesetzt:

```
<title>
  <xsl:value-of select="title"/>
  <xsl:if test="@date!=''">
    (<xsl:value-of select="@date"/>)
  </xsl:if>
</title>
```

- kürzer, aber etwas andere Semantik:

```
<xsl:if test="@date">
```

- kein else-Zweig

Die Bedingung ist wahr wenn der Ausdruck

- zu einem wahren Booleschen Wert
- zu einem nicht-leeren node set
- zu einem nicht-leeren String
- zu einer von Null verschiedenen Zahl

evaluiert.

Aufgaben

- Verwende das Stylesheet `cd2html.xsl` aus der letzten Stunde, oder aus dem Übungs-Archiv.
- Falls keine Jahreszahl angegeben ist, wird ein leeres Paar Klammern ausgegeben. Ändere das Stylesheet so ab, daß die Klammern nur um eine tatsächlich vorhandene Jahreszahl gesetzt werden.

```
<h1>  
<xs1:value-of select="artist"/>  
<xs1:text /> / <xs1:text />  
</xs1:value-of select="title"/>  
<xs1:if test="@year">  
<xs1:text /> (</xs1:text />  
<xs1:value-of select="@year"/>  
</xs1:if />  
<xs1:text />  
</h1>
```

Verzweigungen

```
<body>
  <xsl:attribute name="bgcolor">
    <xsl:choose>
      <xsl:when test="@status='draft'">
        <xsl:text>red</xsl:text>
      </xsl:when>
      <xsl:when test="@status='final'">
        <xsl:text>blue</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text>white</xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
  <xsl:apply-templates/>
</body>
```

Aufgaben

- Färbe die Überschriften der CDs entsprechend ihrem Erscheinungsjahr ein:

vor 1970	rot
1971–1990	blau
ab 1991	grün
sonst	schwarz

Verwende das (nicht mehr zeitgemäße) `font`-Element:

```
<h1><font color="red">The Beatles ...</font></h1>
```

```
<h1>  
<font  
...  
</font>  
</h1>  
  
<xs:attribute name="color">  
<xs:choose>  
<xs:when test="@year<1970">  
<text>rot</text>  
</xs:when>  
<xs:when test="@year<1970 and @year>1991">  
<text>blau</text>  
</xs:when>  
<xs:when test="@year>1991">  
<text>grün</text>  
</xs:when>  
<xs:otherwise>  
<text>schwarz</text>  
</xs:otherwise>  
</xs:choose>  
</xs:attribute>  
</h1>
```

Schleifen

- Inhaltsverzeichnis für Präsentation

```
<table>
  <xsl:for-each select="//slide[title/@toc='yes']">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="position()"/></td>
    </tr>
  </xsl:for-each>
</table>
```

- Schleife iteriert über node set
- verarbeiteter Knoten wird zum *current node*
- `position()` relativ zum node set

Aufgaben

- Ändere die templates für `song` und `songlist` so ab, daß die Liste der Lieder als nummerierte Tabelle ausgegeben wird:

```
<tr>
  <td align="right">1</td>
  <td>Help!</td>
</tr>
```

- Wie werden die Lieder nummeriert? Warum? Ändere das Stylesheet so ab, daß korrekte Nummern vergeben werden.
- Erzeuge die Tabelle mit Hilfe einer `xsl:for-each`-Schleife.

```
<table>
  <xsl:for-each select="song">
    <tr>
      <td align="right"><xsl:value-of select="position()" /></td>
      <td><xsl:value-of select="." /></td>
    </tr>
  </xsl:for-each>
</table>
```

- template `songlist` mit Schleife, template `song` entfällt:

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

für `songlist` nur die Lieder weiterzuarbeiten:

- Die Lieder werden in 2er-Schritten nummeriert. `position()` zählt auch die Text-Knoten mit whitespaces zwischen den `song`-Knoten mit. Die Lösung ist, im template

```
<xsl:template match="song">
  <tr>
    <td align="right"><xsl:value-of select="position()" /></td>
    <td><xsl:apply-templates /></td>
  </tr>
</xsl:template>
```

- Neue Version des `song`-templates:

push vs. pull templates

- *push templates:*

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

- *pull templates:*

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

- Stylesheet kann aus einem einzigen pull template bestehen
- Stylesheets haben sehr unterschiedlichen Aufbau
- Wahl hängt von der Struktur der Quell- und Zieldatei ab
- *data centric vs. document centric*

templates aufrufen

- Wie wird das Inhaltsverzeichnis in die Ausgabe eingebunden?
- *named templates*

```
<xsl:template name="maketoc">
  <table>
    <xsl:for-each select="//slide[title/@toc='yes']">
      ...
    </xsl:for-each>
  </table>
</xsl:template>
```

- Aufruf an gewünschter Stelle:

```
<xsl:apply-templates select="title|author"/>
<xsl:call-template name="maketoc"/>
<xsl:apply-templates select="slide"/>
```

Aufgaben

- Verlagere die Ausgabe der Statistik (Anzahl CDs, Anzahl Lieder) in ein eigenes, benanntes template. Rufe es an der passenden Stelle auf.

```
<body>
  <xsl:call-template name="stats" />
</body>
<xsl:apply-templates />
</xsl:template>
</p>
songs.
<xsl:value-of select="count(//song)" />
CDs with a total of
<xsl:value-of select="count(/collection/cd)" />
I have
<p>
<h1>Statistics</h1>
<xsl:template name="stats">
```

Variablen

- Aufgabe: wandle URL

```
<url>http://www.w3c.org</url>
```

in aktiven Link um

```
<a href="http://www.w3c.org">http://www.w3c.org</a>
```

- mit Hilfe einer Variablen:

```
<xsl:template match="url">
  <xsl:variable name="url" select="."/>
  <a href="{url}"><xsl:value-of select="{url}" /></a>
</xsl:template>
```

- alternative Zuweisung:

```
<xsl:variable name="today">
  today is <xsl:value-of select="@date" />
</xsl:variable>
```

Variablen, cont.

- Variablen nicht nachträglich änderbar
- Platzhalter wie in funktionalen Sprachen
- keine Seiteneffekte
- imperative Verfahren funktionieren nicht
- FAQ: „Wie zähle ich eine Variable in einer Schleife hoch?“
- stattdessen: z.B. Rekursion

Sichtbarkeit von Variablen

- scope: aktueller „Block“
- falsch:

```
<xsl:if test="@status='draft'">  
  <xsl:variable name="color" select="'red'"/>  
</xsl:if>
```

- richtig:

```
<xsl:variable name="color">  
  <xsl:if test="@status='draft'">  
    <xsl:text>red</xsl:text>  
  </xsl:if>  
</xsl:variable>
```

Aufgaben

- Die XSLT-Anweisungen zum Einfärben der CD-Überschriften „zerreißen“ den HTML-Code sehr stark. Das Stylesheet kann mit Hilfe einer Variablen etwas übersichtlicher gestaltet werden. Schreibe das `cd-template` so um, daß die Textfarbe auf diese Weise festgelegt wird:

```
<h1>
  <font color="{ $color }">
    ...
  </font>
</h1>
```

```
<xsl:template match="cd"
  <xsl:variable name="color"
  <xsl:choose>
    ...
  </xsl:choose>
  <xsl:choose>
    ...
  </xsl:choose>
  <xsl:variable>
  </xsl:variable>
  <h1>
    <font color="{ $color }">
      ...
    </font>
  </h1>
  <xsl:apply-templates select="songlist" />
</xsl:template>
```

Parameter

- Parameter an named template übergeben:

```
<xsl:template name="maketoc">
  <xsl:param name="heading"/>
  <xsl:param name="color"/>
  <h1><font color="{ $color }">
    <xsl:value-of select="$heading"/>
  </font></h1>
  ...
</xsl:template>
```

- Aufruf:

```
<xsl:call-template name="maketoc">
  <xsl:with-param name="heading" select="Inhaltsverzeichnis"/>
  <xsl:with-param name="color" select="yellow"/>
</xsl:call-template>
```


Parameter, cont.

- default-Werte

```
<xsl:template name="phone">
  <xsl:param name="prefix">0521/106-</xsl:param>
  <xsl:param name="extension"/>
  ...
</xsl:template>
```

- globale Parameter:

```
<xsl:stylesheet>
  <xsl:param name="email">webmaster</xsl:param>
```

- Übergabe beim Aufruf des XSLT-Prozessors:

```
$ xsltproc --param email 'juser' src2html.xsl index.xml
```

Aufgaben

- Erweitere das „Statistik-template“ um einen Parameter `withquot`. Wenn dieser Parameter auf `yes` gesetzt ist, soll zusätzlich die durchschnittliche Anzahl Lieder pro CD angezeigt werden:

```
<p>
  I have 6 CDs with a total of 70 songs.
  That are 11.6666666666667 songs per CD.
</p>
```

(Hinweis: Siehe Übung zur Veranstaltung „XPath“)

```
<xsl:call-template name="stats"
  <xsl:with-param name="withquot" select="'yes'" />
</xsl:call-template>
<p>
  I have
  <xsl:value-of select="$cdcount" />
  CDs with a total of
  <xsl:value-of select="$songcount" />
  songs.
  <xsl:if test="$withquot" >
    That are
    <xsl:value-of select="$songcount div $cdcount" />
    songs per CD.
  </xsl:if>
</p>
```

rekursive templates

```
<xsl:template name="square">
  <xsl:param name="value" select="1"/>
  <xsl:param name="maxval"/>
  <tr>
    <td><xsl:value-of select="$value"/></td>
    <td><xsl:value-of select="$value * $value"/></td>
  </tr>
  <xsl:if test="$value < $maxval">
    <xsl:call-template name="square">
      <xsl:with-param name="value" select="$value + 1"/>
      <xsl:with-param name="maxval" select="$maxval"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>
```

rekursive templates, cont.

- Aufruf:

```
<table>
  <xsl:call-template name="square">
    <xsl:with-param name="maxval" select="5"/>
  </xsl:call-template>
</table>
```

- kann genauso effizient sein wie Schleife (*tail recursion*)

Variablen und Parameter

- Datentypen in XPath: node set, Boolean, String, Zahl

- node set in Variable ablegen:

```
<xsl:variable name="tocsl" select="slide[title/@toc='yes']"/>
```

- XPath-Ausdruck mit Variable:

```
<xsl:for-each select="$tocsl">
```

```
<xsl:for-each select="$tocsl/title">
```

- node set als Parameter:

```
<xsl:call-template name="mktoc">
```

```
<xsl:with-param name="tocsl" select="slide[title/@toc='yes']"/>
```

```
</xsl:call-template>
```

Aufgaben

- Die Datei order.xml enthält eine Warenliste:

```
<order>
  <item status="legal">
    <name>Food</name><quant>10</quant><price>4.4</price>
  </item>
  ...
</order>
```

Schreibe ein Stylesheet, das diese Liste tabellarisch ausgibt und dabei den Gesamtpreis für jeden einzelnen Posten und den Gesamtbetrag der ganzen Bestellung berechnet.

```
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Quantität</th>
      <th>Preis</th>
      <th>Gesamtpreis</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Food</td>
      <td>10</td>
      <td>4.4</td>
      <td>44.0</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="3">Gesamt</td>
      <td>44.0</td>
    </tr>
  </tfoot>
</table>
```