XML-Praxis

XSLT – XSL Transformations

Teil 2

Jörn Clausen

joern@TechFak.Uni-Bielefeld.DE

Übersicht

- weitere XSLT-Techniken:
  - Bedingungen, Verzweigungen
  - Schleifen
  - templates aufrufen
  - Variablen, Parameter
- Anwendungsbeispiel

Bedingungen

- Datum ausgeben, falls `date`-Attribut gesetzt

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

- XPath-Ausdrücke, relativ zum Kontext-Knoten
- auf korrekte Schachtelung der quotes im Test achten
- kein `if-then-else`, nur `if-then`

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>
```

Schleifen

- Inhaltsverzeichnis für Präsentation

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

- Schleife liefert *node set*
- `position()` relativ zum node set

Schleifen, cont.

- Sortierung ändern

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

- `xsl:sort` unmittelbar nach `xsl:for-each`
- Reihenfolge umkehren

```
<xsl:sort select="name" order="descending"/>
```

- numerisch sortieren

```
<xsl:sort select="zipcode" data-type="number"/>
```

templates aufrufen

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

```
<xsl:template name="maketoc">
  <table>
    <xsl:for-each select="/presentation/slide">
      ...
</xsl:template>
```

- Aufruf an gewünschter Stelle

```
<xsl:call-template name="maketoc"/>
```

template-Aufrufe steuern

- bisher

```
<xsl:template match="/presentation">
   ...
   <xsl:apply-templates/>
   ...
```

- ein `xsl:apply-templates` für `title`, `author` und `slide`
- `maketoc` soll dazwischen

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

- Nachteil: Stylesheet nicht mehr so allgemein

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

Variablen

- Aufgabe: erzeuge Hyperlink

```
<link>http://www.TechFak.Uni-Bielefeld.DE</link>
```

- verwende Variable

```
<xsl:template match="link">
  <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 Programmierverfahren funktionieren nicht
- 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>
```

Parameter

- parametrisierte templates

```
<xsl:template name="tocrow">
  <xsl:param name="entry"/>
  <xsl:param name="page"/>
  <tr>
    <td><xsl:value-of select="$entry"/></td>
    <td><xsl:value-of select="$page"/></td>
  </tr>
</xsl:template>
```

- Aufruf

```
<xsl:call-template name="tocrow">
  <xsl:with-param name="entry" select="title"/>
  <xsl:with-param name="page" select="position()"/>
</xsl:call-template>
```

Parameter, cont.

- default-Werte

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

- globale Parameter

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

- Übergabe beim Aufruf des XSLT-Prozessors

```
$ xsltproc --param email '"juser"' tohtml.xsl page.xml
```

rekursive templates

```
<xsl:template name="square">
  <xsl:param name="value">1</xsl:param>
  <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 &lt; $maxval">
    <xsl:call-template name="square">
      <xsl:with-param name="value" select="$value + 1"/>
      <xsl:with-param name="maxval" select="$maxval"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>
```

rekursive templates, cont.

- Verwendung

```
<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*)

weitere Eigenschaften von XSLT

- XSLT kann noch mehr:
  - modulare Stylesheets mit `xsl:import` und `xsl:include`
  - Import weiterer XML-Dateien mit `document()`
  - Kopie von Elementen mit `xsl:copy` und `xsl:copy-of`
- XSLT Standard Library (http://xsltsl.sourceforge.net)
- Erweiterungen
  - prozessorabhängige
  - „Extensions to XSLT“ (http://www.exslt.org)

ein Beispiel aus der Praxis

- Aufgabe: Perl-Module installieren
- Problem: Abhängigkeiten zwischen verschiedenen Modulen
- beschreibe Daten in XML

```
<module name="XML-Parser">
  <instance version="2.31" installer="joern" date="2002-08-19"/>
  <depends>expat 1.95.4</depends>
</module>

<module name="XML-Twig">
  <instance version="3.05" installer="joern" date="2002-08-21"/>
  <uses module="XML-Parser"/>
</module>
```

ein Beispiel aus der Praxis, cont.

- erzeuge Web-Seite mit tabellarischer Übersicht
  - alphabetisch sortiert
  - aktuelle Version hervorheben, ältere Versionen dokumentieren
  - Abhängigkeiten als Hyperlinks
  - „A verwendet B“ $\rightarrow$ „B wird von A verwendet“
- visualisiere Abhängigkeitsgraph
  - Softwarepaket „GraphViz“
  - erzeuge Graphbeschreibung im ASCII-Format „dot“

ein Beispiel aus der Praxis, cont.