Neues in XSLT 2.0 und XPath 2.0

Wandlungsfähig

XML wird zunehmend zur strukturierten Ablage von Daten eingesetzt. Die Transformationssprache XSLT für die Verarbeitung solcher Datenstrukturen und XPath zur Lokalisierung von Inhalten gehen nun in den Versionen 2.0 an den Start. **Von Thomas Meinike**

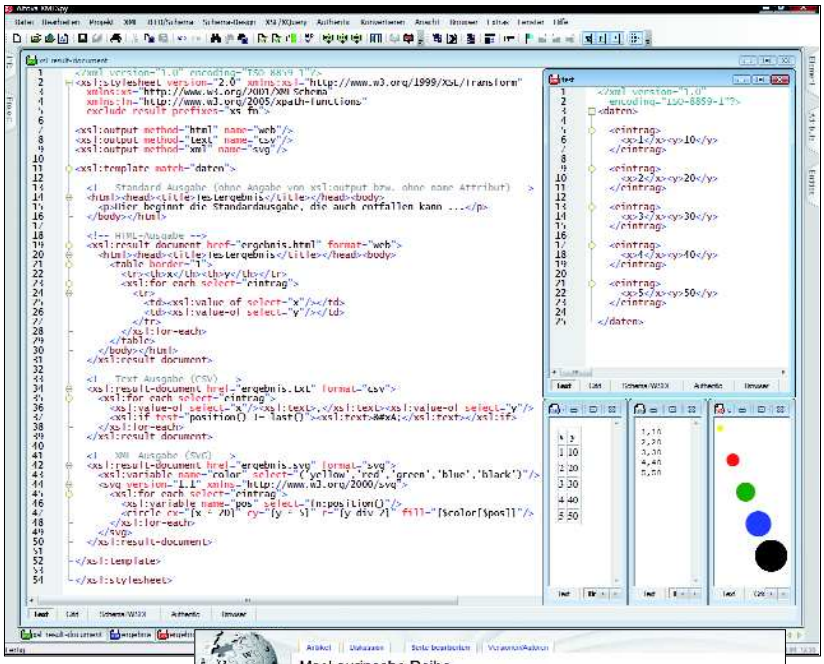
Info

Auf einen Blick

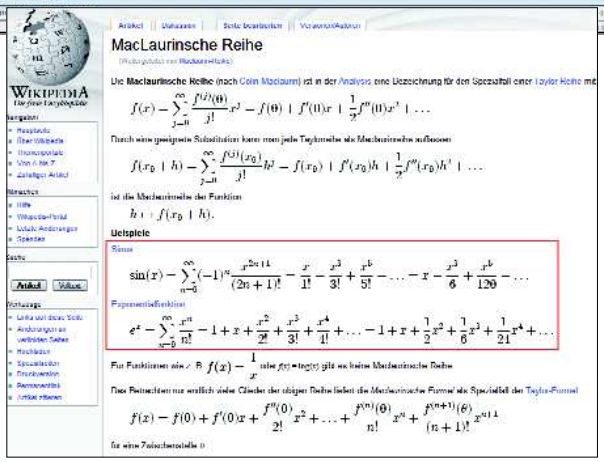
»Die wesentlichen Erweiterungen von XSLT 2.0 und XPath 2.0 werden vorgestellt. Kleine Beispiele demonstrieren die neuen Funktionen.

Das brauchen Sie

»XML-Editor wie XMLSpy
»XSLT-Prozessoren wie Altova XML 2007 oder Saxon 8.9
»Software auf Heft-CD
»Beispiele auf Heft-CD



In XMLSpy sind Beispielausgaben zu *xsl:result-document* zu sehen.



Mathematische Grundlagen zur Berechnung von Sinus und Cosinus liefert Wikipedia (de.wikipedia.org/wiki/MacLaurinsche_Reihe).

Im November 1999 hat das W3-Konsortium (W3C) die ersten Empfehlungen zu den Techniken XSLT und XPath veröffentlicht. An ihren Nachfolgern wurde in den folgenden Jahren intensiv gearbeitet, und am 23. Januar 2007 gab das W3C schließlich grünes Licht für die finalen Spezifikationen XSLT 2.0 und XPath 2.0. Hinzugekommen ist die Abfragesprache XQuery 1.0, die vor allem im Bereich der aufstrebenden XML-Datenbanken von Bedeutung ist. Insgesamt wurden acht neue Empfehlungen veröffentlicht (siehe Kasten).

XSLT 2.0 stellt neue Elemente aus dem *xsl*-Namensraum und einige zusätzliche Funktionen zur Verfügung. Die aus XPath 1.0 bekannten Funktionen wie *sum()* oder *position()* existieren weiterhin, wurden jedoch dem neuen *fn*-Namensraum zugeordnet und sind nun in der Form *fn:sum()* beziehungsweise *fn:position()* zu verwenden. Eine Reihe neuer XPath-Funktionen hilft Entwicklern bei der Umsetzung von Lösungen, die bisher nur aufwendig oder überhaupt nicht realisierbar waren, zum Beispiel Datumsabfragen.

Als weitere Verfeinerung erweist sich die Integration von XML-Schema-Datentypen *(xs*-Namensraum). Somit lassen sich präzise Prüfungen von XML-Inhalten durchführen, etwa das Vorliegen ganzzahliger Werte.

Besonders nützlich ist für Programmierprofis die Einbeziehung von regulären Ausdrücken in die neuen Sprachkonstrukte. Ein typisches XSLT-2.0-Grundgerüst hat diesen Aufbau:

```
<?xml version="1.0" encoding="..."?>
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform"
xmlns:fn="http://www.w3.org/2005/
xpath-functions"
xmlns:xs="http://www.w3.org/2001/
XMLSchema"
exclude-result-prefixes="fn xs">
<xsl:template match="/">
<!-- weitere Inhalte -->
</xsl:template>
</xsl:stylesheet>
```

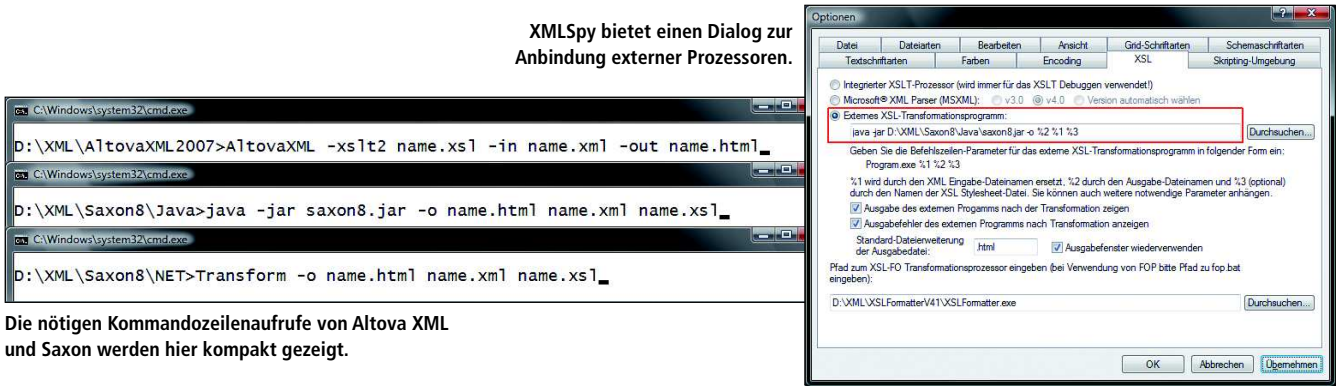
Neben den neuen Namensraum-Deklarationen ist die Versionsnummer 2.0 anzugeben, um XSLT-Prozessoren einzustellen. Die Angabe *exclude-result-prefixes* dient zur Vermeidung von nicht benötigten Namensraumangaben im Ergebnisdokument.

»Sequenzen überall«

XPath 1.0 kannte lediglich Knotenmengen der Typen *document, element, attribute, text, namespace, processing instruction* und *comment.* XPath 2.0 arbeitet dagegen durchgängig mit so genannten Sequenzen, die geordnete Listen von null oder mehreren Einheiten *(items)* darstellen. Dabei können atomare Werte aus dem Bereich der Schema-Typen sowie die genannten sieben Knotentypen vorkommen. Eie Abfrage der Art

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

gibt eine Sequenz aller *b*-Elementknoten unterhalb von *a*-Elementknoten zurück. Sequenzen lassen sich alternativ auch direkt formulieren und über ihren Positionsindex ansprechen.



XMLSpy bietet einen Dialog zur Anbindung externer Prozessoren.

Die nötigen Kommandozeilenaufrufe von Altova XML und Saxon werden hier kompakt gezeigt.

```
<xsl:variable name="seq"
select="(10,20,30)"/>
<xsl:value-of select="$seq[2]"/>
```

$seq[2] entspricht dem Wert 20. Statt Zahlenwerten können auch Zeichenketten oder sonstige XML-Knoten verwendet werden. Zudem gibt es noch das Element *xsl:sequence*, das ebenfalls Sequenzen erzeugt. Dazu sei auf die umfangreichen Beispiele auf der Heft-CD verwiesen.

Zur Bearbeitung von Sequenzen gibt es Funktionen, die an die Syntax anderer Programmiersprachen erinnern: *distinct-values(), empty(), exists(), index-of(), insert(), remove(), reverse()* und *subsequence()*. In der Praxis wird noch das genannte *fn*-Präfix vorgelagert. Die Abfrage des Wertes 30 bezogen auf die Variable *$seq*

```
<xsl:value-of select=
"fn:index-of($seq,30)"/>
```

ergibt entsprechend der Position 3 zurück. Das bisherige Konzept der statischen Ergebnisbaumfragmente (RTF) wurde übrigens zu Gunsten von Sequenzen aufgegeben.

»Erweiterte Abfragen«

Formal einfach erscheinende Aufgaben erforderten bisher mühevolle Umsetzungen, meistens unter Nutzung von *xsl:call-template* zum rekursiven Template-Aufruf. Das folgende XML-Fragment enthält ein Attribut *sterne*, das als HTML-Ausgabe in Form von *img*-Elementen umgesetzt werden soll.

```
<hotel sterne="3">
<ort>A</ort>
<name>B</name>
</hotel>
```

Eine korrekte Adressierung des *hotel*-Elements vorausgesetzt, reicht nun zur Produktion von drei *img*-Elementen diese Formulierung aus:

```
<xsl:for-each select="(1 to
@sterne)">
<img src="stern.gif" alt="*"/>
</xsl:for-each>
```

Auch hier wurde wiederum komfortabel auf eine dynamisch erzeugte Sequenz mit einer *from-to*-Syntax zugegriffen. Ähnliche Umstände bereitete bisher die Summierung von Produkten (hier wird die Summe der Einzelwerte *anzahl * preis* gesucht).

```
<bestellungen>
<auswahl>
<produkt>A</produkt>
<anzahl>2</anzahl>
<preis>14.50</preis>
</auswahl>
<auswahl>
<produkt>B</produkt>
<anzahl>5</anzahl>
<preis>9.20</preis>
</auswahl>
</bestellungen>
```

Summen lassen sich weiterhin mittels *fn:sum()* bilden, für Produkte ist die neue *for-in-return*-Anweisung hilfreich. Damit ergibt sich ein kompakter Ausdruck:

```
<xsl:value-of select="fn:sum(for
$a in auswahl return $a/anzahl *
$a/preis)"/>
```

Konditionale Abfragen waren unter XSLT 1.0 über die Elemente *xsl:if* sowie *xsl:choose, xsl:when* und *xsl:otherwise* möglich. XPath 2.0 stellt zusätzlich ein *if-then-else*-Konstrukt bereit.

```
<xsl:variable name="test"
select="99"/>
<xsl:value-of select="if($test lt
100) then 'günstig' else 'teuer'"/>
```

Dieses Beispiel gibt das Ergebnis *günstig* zurück. Die Schreibweise *lt* ist eine weitere Neuerung und steht für kleiner als (less than).

Der Unterschied von *lt* zum Zeichen < beziehungsweise der Umschreibung *<* im XML-Kontext besteht im Vergleich von Einzelwerten gegenüber Knotenmengen. Neben *lt* stehen die Bezeichner *eq, ge, gt, le* und *ne* zur Verfügung (e: equal, g: greater, l: less, n: not). Zum Vergleich auf Knotenebene dienen die neuen Operatoren << (linker Knoten vor rechtem Knoten), >> (linker Knoten nach rechtem Knoten) und *is* (Identität von Knoten).

»Eigene Funktionen«

XSLT 2.0 bietet mit dem Element *xsl:function* endlich die Möglichkeit, benutzerdefinierte Funktionen einzusetzen. Dieses Element muss als direktes Kindelement von *xsl:stylesheet* notiert werden. Es benötigt einen Namen mit eigenem Namensraumpräfix. Argumente werden über *xsl:param*-Kindelemente übergeben und das ermittelte Ergebnis wird zurückgegeben. Der folgende Code dient zur Berechnung der Fakultät:

```
<xsl:function name="tm:fact"
as="xs:double">
<xsl:param name="n"
as="xs:integer"/>
<xsl:value-of select="if($n eq 0
or $n eq 1) then 1 else $n *
tm:fact($n - 1)"/>
</xsl:function>
```

Selbst definierte Funktionen lassen sich innerhalb von XPath-Ausdrücken wie jede andere vordefinierte Funktion aufrufen. Mit dem Argument 5 wird 120 erhalten.

```
<xsl:value-of select="tm:fact(5)"/>
```

Die Fakultät wird als Hilfsfunktion zur Ermittlung von Sinus- und Cosinus-Funktionswerten im Rahmen der XSLT-Umsetzung von SVG-Kreisdiagrammen benötigt (siehe Heft-CD).

Info

XML-Schema-Basistypen für XPath

xs: ...		
anyURI	double	gYearMonth
base64Binary	duration	hexBinary
boolean	float	NOTATION
date	gDay	QName
dateTime	gMonth	string
decimal	gMonthDay	time
	gYear	

Eine Einführung in die XML-Verarbeitung finden Sie bei SelfHTML (de.selfhtml.org).

Xqueryfunctions.com informiert über XPath- und XQuery-Funktionen.

Das neue *as*-Attribut ermöglicht die Zuordnung von Schema-Datentypen und kann neben *xsl:function* und *xsl:param* auch mit den Elementen *xsl:with-param* und *xsl:variable* kombiniert werden. Mögliche XML-Schema-Basistypen sind im Kasten auf Seite 85 aufgelistet.

»Mehrere Ausgabedokumente«

Unter XSLT 1.0 lassen sich Ausgaben über das Element *xsl:output* und dessen *method*-Attribut steuern. Mögliche Ziele waren bisher HTML, XML oder einfacher Text, etwa CSV-Formate. XSLT 2.0 definiert zusätzlich die XHTML-konforme Ausgabe.

```
<xsl:output method="xhtml"/>
```

Waren bisher für mehrere Ausgabeformate jeweils eigenständige Stylesheets notwendig, übernimmt das neue Element *xsl:result-document* diese Aufgabe innerhalb einer Transformationsvorlage. Zunächst werden mehrere benannte *xsl:output*-Elemente angelegt, wobei eines auch ohne Namen auskommt und dann die Standardausgabe festlegt.

```
<xsl:output method="html"
name="web"/>
<xsl:output method="text"
name="csv"/>
<xsl:output method="xml"
name="svg"/>
```

Innerhalb des jeweiligen Templates werden *xsl:result-document*-Blöcke gebildet, die über das *name*-Attribut auf die Formate referenzieren. Die Ausgabe in eine Datei steuert das *href*-Attribut.

```
<xsl:result-document href="
ergebnis.html" format="web">...
</xsl:result-document>
...
<xsl:result-document href=
"ergebnis.svg format="svg">...
</xsl:result-document>
```

»Gruppierungen«

Gerade bei strukturierten Dokumenten ergibt sich häufig die Anforderung einer Gruppierung von Inhalten nach einem bestimmten Kriterium. Die dabei bisher im Entwicklerkopf entstandenen Knoten werden durch das neue Element *xsl:for-each-group* aufgelöst.

Im folgenden Beispiel sollen die CD-Titel *(t)* nach den mehrfach vorkommenden Interpreten *(i)* gruppiert werden.

```
<cds>
<cd><i>Nick Cave &amp;amp; The Bad
Seeds</i><t>The Good Son</t></cd>
<cd><i>Einstürzende Neubauten</i>
<t>Silence Is Sexy</t></cd>
<cd><i>Nick Cave &amp;amp; The Bad
Seeds</i><t>Murder Ballads</t></cd>
<cd><i>Einstürzende Neubauten</i>
<t>Haus der Lüge</t></cd>
</cds>
```

Das folgende Code-Fragment ist auf die Ausgabe innerhalb einer HTML-Tabelle ausgelegt.

```
<xsl:for-each-group select="cds/cd"
group-by="i">
<tr>
<td><xsl:value-of select="i"/></td>
<td><xsl:value-of select="current-
group()/t" separator=", "/></td>
</tr>
</xsl:for-each-group>
```

Wesentlich sind das Attribut *group-by* und die XSLT-Funktion *current-group()*, die die gruppierte Ausgabe der Daten bewerkstelligen. Interessant ist an dieser Stelle noch das neue *separator*-Attribut, mit dem sich einfache Auflistungen von Inhalten erzeugen lassen. Weitere Attribute zum Gruppieren sind *group-adjacent, group-starting-with* und *group-ending-with.*

»Textanalyse«

Zum Arbeiten mit Zeichenketten standen bereits unter XPath 1.0 einige Funktionen wie *substring-after()* und *substring-before()* zur Verfügung. XPath 2.0 hat neue Funktionen im Gepäck, die auf der Basis von Perl-kompatiblen regulären Ausdrücken arbeiten: *fn-matches()* sucht Übereinstimmungen und gibt *true* oder *false* zurück, *fn:replace()* ersetzt Teilzeichenketten und *fn:tokenize()* zerlegt Zeichenketten nach einem Muster.

```
<xsl:variable name="txt"
select="'D 06217'"/>
<xsl:value-of
select="fn:matches($txt,'(D
)?\d{5}')"/><!-- true -->
<xsl:value-of select=
"fn:replace($txt,'\d{5}','xxxxx')"/>
<!-- D xxxxx -->
```

Als nützlich erweist sich die neue XSLT-Funktion *unparsed-text()*. Diese ermöglicht den Aufbau von XML-Strukturen auf der Basis einfacher Textdateien. Ausgehend vom Inhalt der Datei *buecher.txt*

```
Charles Bukowski,Barfly
George Orwell,1984
```

erzeugt dieses XSLT-Fragment eine XML-Ausgabe unter Verwendung der Funktion *fn:tokenize()* zur Erkennung von beliebigen Zeilenumbrüchen:

```
<buecher>
<xsl:for-each select="fn:tokenize
($text,'(\r)|(\n)|(\r\n)')">
<xsl:if test="fn:string-length(.)
gt 0">
<buch>
<autor><xsl:value-of select="fn:
substring-before(.,',')"/></autor>
<titel><xsl:value-of select="fn:
substring-after(.,',')"/></titel>
</buch>
</xsl:if>
</xsl:for-each>
</buecher>
```

Links

Sprachwächter

W3C-Empfehlungen vom 23. Januar 2007
▶**www.w3.org/News/2007#item8)**

XSL Transformations (XSLT) Version 2.0
▶**www.w3.org/TR/xslt20**

XML Path Language (XPath) 2.0
▶**www.w3.org/TR/xpath20**

XQuery 1.0 und XPath 2.0: Funktionen und Operatoren
▶**www.w3.org/TR/xpath-functions**

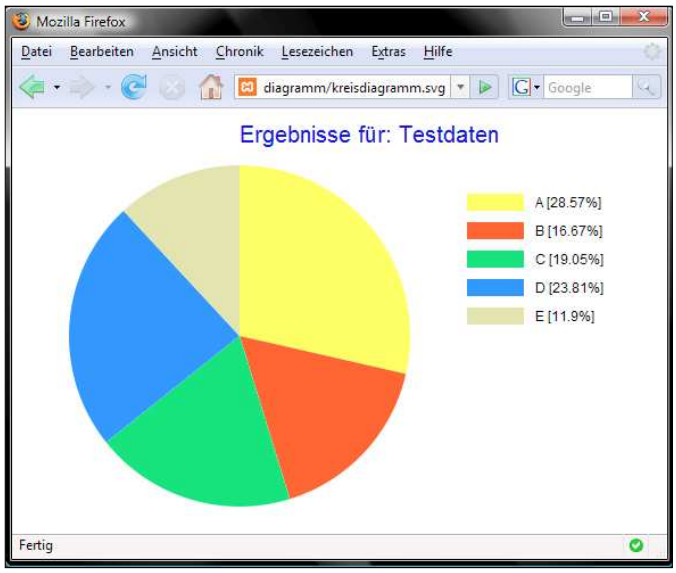
XQuery 1.0: XML-Abfrage-Sprache
▶**www.w3.org/TR/xquery**

Das neue Element *xsl:analyze-string* mit seinen Kindelementen *xsl:matching-string* und *xsl:non-matching-string* operiert ebenfalls mit regulären Ausdrücken und kann über die XSLT-Funktion *reg ex-group()* auf gefundene Untergruppen reagieren.

»Datum und Uhrzeit«

Mit den eingeführten XPath-Funktionen *fn:current-date()*, *fn:current-time()* und *fn:current-dateTime()* können aktuelle Werte von Datum und Uhrzeit abgefragt werden. XSLT-Funktionen wie *format-date()*, *format-time()* und *format-dateTime()* helfen bei der Formatierung.

```
<xsl:variable name="datum"
select="fn:current-date()"/>
<xsl:value-of select="format-
date($datum, '[D].[M].[Y]')"/>
```



Das Beispiel des SVG-Kreisdiagramms wird im Browser angezeigt.

»Sonstiges«

Beim Studium der Spezifikationen lassen sich weitere Techniken entdecken:

- Speziellere Elemente *xsl:document, xsl:import-schema, xsl:namespace, xsl:next-match, xsl: perform-sort*
- Aggregatfunktionen *fn:avg(), fn:min(), fn:max()*
- Elemente *xsl:character-map* / *xsl:output-charac ter* zum Ersetzen von Zeichen
- Quantifikatoren *some/every-in-satisfies*
- Mengenoperatoren *except* (Differenz), *intersect* (Durchschnitt), *union* (Vereinigung)
- Funktionen für Klein- und Großschreibung *fn:lower-case()* und *fn:upper-case()*
- Typoperationen *cast as, castable as, treat as*
- Operator *idiv* für Ganzzahldivision
- *(: … :)* für Kommentare in XPath-Ausdrücken

Viele Anregungen zur praktischen Anwendung sind in den Beispielen auf der Heft-CD zu finden.

»Software und Fazit«

Die XSLT- und XQuery-Engine und XML-Parser Altova XML 2007 *(altova.com/altovaxml.html)* und der XSLT- und XQuery-Prozessor Saxon 8.9 *(saxon.sourceforge.net)* unterstützen die finalen Spezifikationen von XSLT 2.0 beziehungsweise XPath 2.0 bereits. Beide werden über die Kommandozeile gesteuert oder direkt in XML-Editoren wie XMLSpy eingebunden.

Weitere XSLT-Prozessoren sollen in Kürze auf dem aktuellen Stand sein. Für PHP sind gegenwärtig noch keine Pläne bekannt, vermutlich werden aber auch die Bibliotheken angepasst. Microsoft beabsichtigt die Unterstützung im Rahmen des .NET Frameworks und im nächsten Release von Visual Studio (Orcas).

XSLT und XPath stellen in den neuen Versionen erhebliche Verbesserungen bereit, die Entwicklern das Leben erleichtern und vielleicht auch von diesen Techniken noch nicht überzeugte Anwender zu einem Umdenken veranlassen. **[jp]**

Verlosung

Mitmachen & gewinnen

Internet Professionell verlost drei Exemplare des Buches »XSLT 2.0 und XPath 2.0« in der kommenden, frisch aktualisierten 2. Auflage aus dem Galileo-Verlag im Wert von je 59,90 Euro. Das Buch bietet eine umfassende Einführung in die komplexe Materie der Transformation von XML-Dateien mittels XSLT. Dabei werden die Sprachelemente und Instruktionen von XSLT sowie die Elemente und Funktionen der Pfadbeschreibungssprache XPath beschrieben. Das Fachbuch wurde aktuell zur neuen Version von XSLT 2.0 und XPath 2.0 erweitert und aktualisiert. Die zweite Auflage dieses Buches wird voraussichtlich im August erscheinen. Um an der Verlosung teilzunehmen, füllen Sie bitte den kurzen Fragebogen unter www.ipro-leser.de aus.