XML : XSLT

In den vorhergehenden Ausgaben haben wir uns mit den Grundlagen, dem Export und dem Import von Informationen mit Hilfe von XML in Notes/Domino beschäftigt. In dieser Ausgabe möchte ich Ihnen weitere wichtige Grundregeln in dem Umgang mit XML und Domino nennen sowie einige wichtige Begriffe, die oft im Zusammenhang mit XML genannt werden, erläutern.

In den letzten Artikeln haben wir uns mit den Import- und Exportfunktionen beschäftigt. Dabei wurden Daten von Notes (oder in Notes) mit Hilfe von XML verarbeitet. Betrachtet man den Aufbau solcher XML-Dateien, dann stellt man sehr schnell fest, dass diese zunächst recht unübersichtlich sind und zum anderen eine Fülle an Informationen enthalten, die für die eigentliche Verarbeitung gar nicht benötigt werden.

Solche XML-Dokumente, wie wir sie in den vorhergehenden Ausgaben erstellt haben, können durchaus noch weiter überarbeitet und für die eigentliche Weiterverarbeitung optimiert werden.
Dabei können die Informationen in den XML-Dateien verändert und an die jeweilige Struktur des Ziel-Systems angepasst werden.

Für solche Anpassungen bietet Lotus die Verwendung von XSLT an. Bei XSLT handelt es sich um einen Standard, der im Umfeld von XML in den letzten Jahren mehr und mehr an Bedeutung gewonnen hat und ein Bestandteil der XSL (Extensible Stylesheet Language) ist. Mit XSLT können XML-Dokumente – genauer DXL-Dateien – zwischen unterschiedlichen strukturellen XML-Modellen konvertiert werden. Aber nicht nur die Konvertierung zu unterschiedlichen XML-Modellen macht die Verwendung von XSLT so interessant, sondern auch die Möglichkeit, aus XML mit Hilfe von XSLT HTML zu generieren. Damit können die XML-Informationen zum Beispiel auch in einem Browser verarbeitet werden.

Sicher ist es nicht die Weiterverarbeitung der XML-Informationen in einem Browser, sondern vielmehr die Möglichkeit, mittels XSLT Domino-spezifische Informationen für die unterschiedlichsten Weiterverarbeitungen – wie zum Beispiel die Verwendung für eine Web-Verarbeitung – umzuformatieren.

Die Aufgaben von XSLT in Verbindung mit Domino/Notes-Informationen sind folglich:

- Umsetzung von XML-Daten in HTML-Daten
- Umsetzung von Domino XML-Daten in andere XML-Formate

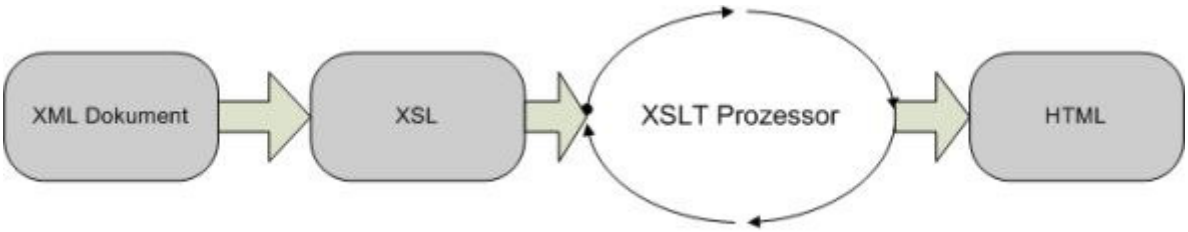

Abb. 001: XSLT-Ablaufschema

Das nachfolgende Beispiel zeigt, wie solche Informationen (wir verwenden zur Veranschaulichung einen einfachen Kundenstamm) in Domino, in XML und anschließend in einem XSLT Style-Sheet aussehen.

Die enthaltenen Daten erscheinen folgendermaßen:

KdNr	Name	Vorname	PLZ	Ort
1	Winzig	Willi	40000	Düsseldorf
2	ITP Verlags GmbH			Kaufering
3	Müller	Jens	40000	Düsseldorf
4	Weber	Waldemar	50000	Köln
5	Müller	Michael	50000	Köln

Abb.002: Notes-Kundenstamm-Daten

In XML (DXL) umgesetzt und entsprechend vorselektiert erhalten wir die folgenden Informationen:

Kunden.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xml" href="kunden.xsl"?>

<KUNDEN>

<KUNDE>
<Name>Winzig</Name>
<Vorname>Willi</Vorname>
</KUNDE>

<KUNDE>
<Name>ITP Verlags GmbH</Name>
<Vorname></Vorname>
</KUNDE>

<KUNDE>
<Name>Müller</Name>
<Vorname>Jens</Vorname>
</KUNDE>

<KUNDE>
<Name>Weber</Name>
<Vorname>Waldemar</Vorname>
</KUNDE>

<KUNDE>
<Name>Müller</Name>
<Vorname>Michael</Vorname>
</KUNDE>

</KUNDEN>
```

Abb. 003: XML-Dokument

Diese Informationen können mittels des XSLT-Prozessors (in diesem Fall verwenden wir den LotusXSL) transformiert werden. LotusXSL ist in dem Standard-Lieferumfang von Domino enthalten. LotusXSL liest die DXL-Datei und erzeugt aus ihr eine neue XML-Datei oder eine Datei in einem anderen Format.

Grundlage für die Umsetzung sind unterschiedliche XSLT-Schablonen, welche die Angaben für die jeweilige Umsetzung enthalten. In der Fachliteratur findet man hier auch oft die Bezeichnungen: „Template", „Styling" oder „Vorlagen".

Eine solche Schablone ist in Abb. 004 enthalten. Mit dieser Schablone erzeugen wir aus der DXL-Datei eine HTML-Aufbereitung der Domino-Informationen zur weiteren Verarbeitung. Dabei werden bestimmte Informationen aus dem XML-Dokument selektiert. Das Ergebnis stellt dann eine HTML-Datei dar.

Kunden.xml KUNDEN.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0"
    xmlns:xalan="http://xml.apache.org/xslt">

    <xsl:template match="KUNDEN">
    <HTML>
    <xsl:apply-templates/>
    </HTML>
    </xsl:template>

    <xsl:template match="KUNDE">
    <P>
    <xsl:value-of select="NAME"/>
    </P>
    </xsl:template>


</xsl:stylesheet>
```

Abb. 004: XSL Stylesheet

In der zweiten Zeile befindet sich die Angabe der verwendeten XSL-Version (klassifiziert durch den Eintrag: „http://www.w3.org/1999/XSL/Transform“).

Zunächst werden dabei die Einträge „KUNDE“ verarbeitet. Diese befinden sich in den dafür vorgesehenen TAGS .

Die Informationen nach diesen TAG-Einträgen werden zwischen den Anweisungen und als HTML-Inhalte definiert und sind somit für eine Standardverarbeitung als HTML verfügbar.

Im nächsten Schritt wird nun auf die Informationen der Elemente „NAME“ zugegriffen. Dabei verwenden wir jeweils den Eintrag „NAME“, der zwischen den Tags

und
eingebunden ist.

Als Ergebnis dieses Ablaufes erhält man die folgenden Informationen:

```
<HTML>
<P>WINZIG</P>
<P>ITP Verlags GmbH</P>
<P>Müller</P>
<P>Weber</P>
<P>Müller</P>
</HTML>
```

Abb. 005: HTML-Dokument

Der Inhalt dieser Datei kann nun mit jedem beliebigen Browser angesehen oder mit jeder Anwendung verarbeitet werden, die HTML-Code verarbeiten kann.

WINZIG

ITP Verlags GmbH

Müller

Weber

Müller

Abb. 006: HTML-Browser-Anzeige

Nachdem wir uns nun grob die Funktionsweise von XSLT angesehen haben, wird es Zeit, die Strukturen von XSLT näher zu betrachten.

Wie man an den Beispielen oben erkennen kann, arbeitet XSLT mit strukturierten Dokumenten. Betrachtet man diese Dokumente genau, dann fällt die Baumstruktur der darin enthaltenen Elemente auf (einige XML-Editoren bereiten die Dokumente ebenso auf!). Die einzelnen Baumbereiche bestehen aus so genannten Knoten, welche die nächste wesentliche Stufe der XML-Elemente darstellen.

Es gibt eine Reihe verschiedener Knoten, die für bestimmte Verwendungen herangezogen werden können:

- Document Root

-- Stellt den Anfang jedes Dokumentes dar.

- Attribute

-- Enthält den Wert eines Attributes.

- Comment

-- Ein Kommentar. Besonderheit: Kommentare können auch verarbeitet werden!

- Element

-- Besteht aus allen Zeichen inklusive der Elemente der untergeordneten Stufe (Kinderelemente).

- Namespace

-- Enthält die URL des Namensraumes.

- Processing Instruction

-- Ist eine Anweisung zur Verarbeitung weiterer Informationen.

- Text

-- Enthält den Text eines Knotens.

Neben diesen Knoten besteht XSLT aus einer Reihe von Anweisungen. Die wichtigste dieser Anweisungen ist die Anweisung „match". Diese Auswahl wird für das Ausführen von Auswahlen verwendet. Schauen Sie sich nochmals die Abbildung 004 an. Dort verwenden wir die Anweisung „match" für die Selektion aller Einträge „KUNDEN" in einem XML-Dokument. Mit der „match"-Anweisung wird exakt angegeben, auf welchen Knoten der Quell-Datei die Anweisung angewandt werden soll.

```
<xsl:template match="KUNDEN">
```

Eine Stufe tiefer finden wir eine weitere „match"-Anweisung, welche die nächste Elementstufe selektiert:

```
<xsl:template match="KUNDE">
```

Diese Form der Anweisungen kann in unendlich vielen Elementstufen ausgeführt werden.

In den hier gezeigten Beispielen wird die Anweisung „match" auf bestimmte Elemente angewendet. Der Einsatzbereich dieser Anweisung beschränkt sich nicht nur auf diese Elemente, sondern kann auch auf Attribute und andere Informationen angewendet werden; sie kann durch die Verwendung von „AND"/„OR"-Anweisungen entsprechend erweitert werden.

Betrachten wir die Abb. 004 weiter: Wir finden hier den Eintrag:

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

Bei dem Eintrag „value-of" handelt es sich um einen Konstruktor, mit dem der Inhalt eines Tags ausgegeben wird.

Mit der Anweisung

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

wird das aktuelle Element „KUNDEN" (die übergeordnete Ebene) mit dem Wert des darunter enthaltenen Elements „KUNDE" gelesen und in das neue Dokument geschrieben.

Es gibt verschiedene Varianten bei der Verwendung von „value-of":
Hier wird der aktuelle Tag-Inhalt ausgegeben.

```
<xsl:value-of />
```

Hat dieselbe Funktion wie

Damit die Inhalte, die verarbeitet werden sollen, selektiert werden können, gibt es entsprechende Bedingungsanweisungen, mit denen die zu verarbeitenden Informationen bedingt werden können.

xsl:if

Mit „xsl.if" können Sie überprüfen, ob Variablen, Funktionen, Tags oder Attribute einen bestimmten Wert haben.

```
<xsl:if match="@name=Müller">
```

Mit dieser Bedingung wird eine Selektion auf das Element „Name" vorgenommen.

Bei der Verwendung von „xsl:if" können unterschiedliche Verschachtelungen realisiert werden.

Wie bei der Verwendung von „if" in anderen Programmiersprachen auch, gibt es eine ganze Anzahl von weiteren Operatoren, die in Verbindung mit „if" eingesetzt werden können:

- Choose

-- Vergleichbar mit „case" in anderen Programmiersprachen. Ist die Basis für die Verwendung von „when" und „otherwise"-Anweisungen.

- When

-- Verwendung ausschließlich in choose-Tags.

- Otherwise

-- Verwendung ausschließlich in choose-Tags – entspricht der Verwendung von „else" in anderen Programmiersprachen.

Neben diesen Bedingungsangaben sind natürlich auch Schleifenprogrammierungen möglich.

xsl:for-each

„xsl:for-each" ist ein sehr praktisches Tag, weil es erlaubt, alle Tags eines bestimmten Typs im aktiven Template auszugeben und zu bearbeiten.

Die Verwendung von „xsl:for-each“ könnte in unserem Beispiel – angewendet auf alle Elemente „KUNDE“ – so aussehen:

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

xsl:sort

„xsl:sort“ ist als Kindelement zu „for-each“ zugelassen. Es sortiert die Ausgabe nach dem in „select“ angegebenem Attribut.

Dieses einfache Beispiel sollte dazu dienen, einen kleinen Einblick in die Möglichkeiten von XSLT zu bekommen, um Domino-spezifischen Informationen mit anderen XML-Versionen oder auch HTML-Anwendungen auszutauschen. Im Umfeld von Domino und XML gibt es noch weitere Möglichkeiten, die Verbindung zwischen Domino- und Nicht-Domino-Daten herzustellen. Entsprechende Anpassungen von DOM und der Einsatz von SAX stellen hier einige weitere interessante Möglichkeiten dar.

Den Autor Jörg Zeig erreichen Sie unter ZeBIS Beratung/Informationssysteme - Im Bühlsgarten 3 - 57223 Kreuztal - Tel. (+49) 2732/892491 - e-Mail: HJZeig@ZeBIS.de