27.6 Komponente XML.XSLT

Die Komponente gb.xml.xslt von Daniel Campos Fernández besitzt nur die Klasse Xslt, die zwei statische Methoden besitzt. Die Komponente basiert auf gb.xml.

27.6.1 Methoden

Die Klasse *Xslt* verfügt über diese beiden Methoden:

Methode	Rückgabetyp	Beschreibung
Transform (Document As XmlDocument, StyleSheet As XmlDocument)	XmlDocument	*Document* ist ein XML-Dokument (*.xml), das die zu transformierenden Daten enthält. *StyleSheet* ist ein XML-Dokument (*.xslt), das die Transformationsanweisungen enthält. Das Ergebnis der Transformation ist ein XML-Dokument, bei dem ein so genannter Quellbaum in einen Ergebnisbaum transformiert wurde.
TransformToString (Document As XmlDocument, StyleSheet As XmlDocument)	String	*Document* ist ein XML-Dokument (*.xml), das die zu transformierenden Daten enthält. *StyleSheet* ist ein XML-Dokument (*.xslt), das die Transformationsanweisungen enthält. Das Ergebnis der Transformation ist ein String, bei dem ein Quellbaum in einen Ergebnisbaum transformiert wurde.

Tabelle 27.6.1.1 : Methoden der Klasse Xslt

27.6.2 XSLT

Bevor Sie sich mit den vorgestellten Projekten beschäftigen, sollten Sie sich aus der Fülle der Dokumentationen zur Sprache XSLT (Extensible Stylesheet Language Transformations) im Internet die folgenden Seiten ansehen:

```
LINK 1: http://wiki.selfhtml.org/wiki/XML/XSL
Link 2: http://wiki.selfhtml.org/wiki/XML/XSL/XPath
```

Auf die Frage, was XSLT ist, kann nach https://www.data2type.de/xml-xslt-xslfo/xslt/ Folgendes genannt werden: "XSLT ist die Transformationssprache von XML. Genau wie XSL-FO und XPath ist XSLT eine Untermenge von XSL. In sogenannten Stylesheets lassen sich Regeln aufstellen, auf deren Basis Daten in ein Zieldokument überführt werden. Die Stylesheets werden von einem XSLT-Prozessor eingelesen, der mit diesen Regeln das XML-Dokument in das gewünschte Ausgabeformat umwandelt."

Die Umwandlung von XML-Dokumenten mit XSLT wird in diesem Kapitel für die Transformationen XML → CSV und XML → HTML beschrieben. Die verwendeten *Stylesheets als Transformationsregeln* sind einfach gehalten, um Ihnen das Prinzip der Transformation zu verdeutlichen. Für die ersten beiden Projekte wird eine *existierende* XML-Datei eingesetzt, während im dritten Projekt die XML-Datei vor der Transformation in HTML erzeugt wird, um aktuelle Daten zu verwenden. In allen drei Projekten wird die Methode 'Transform' eingesetzt. Auf die Angabe des Inhalts der XML-Dateien wird verzichtet.

27.6.3 Projekt 1: XML → HTML

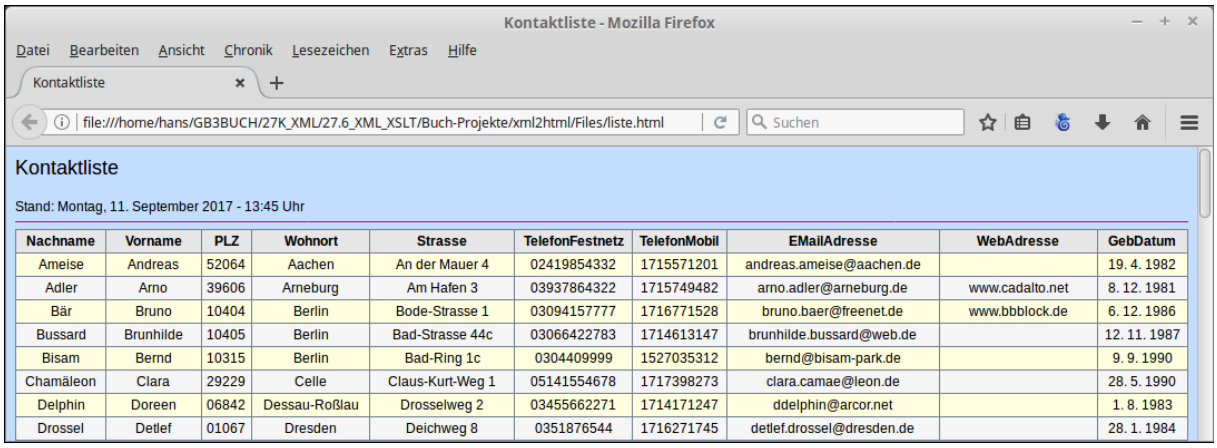
Der Quelltext für alle Beispiele ist erstaunlich kurz und bis auf die Angabe der XML-Datei (*.xml) und der Stylesheet-Datei (*.xsl) fast identisch:

```
[1]  ' Gambas class file
[2]
[3]  Public Sub Form_Open()
[4]    FMain.Center()
[5]    FMain.Resizable = False
[6]    FMain.Caption = "XML-TRANSFORMATION"
[7]  End
[8]
[9]  Public Sub btnConvertXML2HTML_Click()
[10]
[11]   Dim docXML, docXSL, docHTML As XmlDocument
[12]
[13]   docXML = New XmlDocument(Application.Path &/ "Files/liste.xml")
```

```
[14]   docXSL = New XmlDocument(Application.Path &/ "Files/liste.xsl")
[15]   docHTML = New XmlDocument
[16]
[17]   docHTML = Xslt.Transform(docXML, docXSL)
[18] ' Print Xslt.TransformToString(docXML, docXSL)
[19]   docHTML.Save(Application.Path &/ "Files/liste.html")
[20]   Desktop.Open("file://" & Application.Path &/ "Files/liste.html")
[21]
[22] End
```

Kommentar:

- In den Zeilen 13 und 14 werden die Pfade zur XML-Datei und zur Stylesheet-Datei angegeben.
- Die Transformation XML → HTML wird in der Zeile 17 ausgeführt.
- Für Test- oder Kontrollzwecke können Sie in der IDE die Zeile 18 auskommentieren, so dass Sie sich das Ergebnis der Transformation sofort in der Konsole der IDE ansehen können. Eingesetzt wird hier die TransformToString-Methode.
- Anschließend können Sie das Ergebnis der Transformation, das ja nur als XML-Dokument im Speicher liegt, als HTML-Datei abspeichern.
- Die Anweisung in der Zeile 21 zeigt den Inhalt der HTML-Datei im (System-)Webbrowser an:



Kontaktliste

Stand: Montag, 11. September 2017 - 13:45 Uhr

Nachname	Vorname	PLZ	Wohnort	Strasse	TelefonFestnetz	TelefonMobil	EMailAdresse	WebAdresse	GebDatum
Ameise	Andreas	52064	Aachen	An der Mauer 4	02419854332	1715571201	andreas.ameise@aachen.de		19. 4. 1982
Adler	Arno	39606	Arneburg	Am Hafen 3	03937864322	1715749482	arno.adler@arneburg.de	www.cadalto.net	8. 12. 1981
Bär	Bruno	10404	Berlin	Bode-Strasse 1	03094157777	1716771528	bruno.baer@freenet.de	www.bbblock.de	6. 12. 1986
Bussard	Brunhilde	10405	Berlin	Bad-Strasse 44c	03066422783	1714613147	brunhilde.bussard@web.de		12. 11. 1987
Bisam	Bernd	10315	Berlin	Bad-Ring 1c	0304409999	1527035312	bernd@bisam-park.de		9. 9. 1990
Chamäleon	Clara	29229	Celle	Claus-Kurt-Weg 1	05141554678	1717398273	clara.camae@leon.de		28. 5. 1990
Delphin	Doreen	06842	Dessau-Roßlau	Drosselweg 2	03455662271	1714171247	ddelphin@arcor.net		1. 8. 1983
Drossel	Detlef	01067	Dresden	Deichweg 8	0351876544	1716271745	detlef.drossel@dresden.de		28. 1. 1984

Abbildung 27.6.3.1: Anzeige der HTML-Datei

Quelltext für die Stylesheet-Datei:

```
[1]  <?xml version="1.0" encoding="utf-8"?>
[2]  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[3]  <xsl:template match="/">
[4]  <!-- <!DOCTYPE html> darf hier NICHT stehen -->
[5]  <html>
[6]    <head>
[7]      <title>Kontaktliste</title>
[8]      <meta charset="utf-8"></meta>
[9]      <link rel="stylesheet" type="text/css" href="liste.css"></link>
[10]     <script type="text/javascript" src="datetime.js" charset="utf-8"></script>
[11]   </head>
[12]   <body>
[13]     <p>Kontaktliste</p>
[14]     Stand: <span class="id" id="datetime"></span>
[15]     <hr class="line"></hr>
[16]     <table>
[17]     <tr>
[18]       <th>Nachname</th>
[19]       <th>Vorname</th>
[20]       <th>PLZ</th>
[21]       <th>Wohnort</th>
[22]       <th>Strasse</th>
[23]       <th>TelefonFestnetz</th>
[24]       <th>TelefonMobil</th>
[25]       <th>EMailAdresse</th>
[26]       <th>WebAdresse</th>
[27]       <th>GebDatum</th>
[28]     </tr>
[29]     <xsl:for-each select="kontakte/kontakt">
[30]     <!-- Sortierung nach dem Wohnort! -->
[31]     <xsl:sort select="adresse/wohnort/@ort" order="ascending"/>
[32]       <xsl:if test="position() mod 2 = 1">
[33]         <tr class="Row1">
```

```
[34]                <td><xsl:value-of select="nachname"/></td>
[35]                <td><xsl:value-of select="vorname"/></td>
[36]                <td><xsl:value-of select="adresse/wohnort/@plz"/></td>
[37]                <td><xsl:value-of select="adresse/wohnort/@ort"/></td>
[38]                <td><xsl:value-of select="adresse/strasse"/></td>
[39]                <td><xsl:value-of select="kommunikation/festnetz"/></td>
[40]                <td><xsl:value-of select="kommunikation/mobil"/></td>
[41]                <td><xsl:value-of select="kommunikation/internet/email"/></td>
[42]                <td><xsl:value-of select="kommunikation/internet/web"/></td>
[43]                <td>
[44]                  <xsl:value-of select="geburtstag/tag"/>.
[45]                  <xsl:value-of select="geburtstag/monat"/>.
[46]                  <xsl:value-of select="geburtstag/jahr"/>
[47]                </td>
[48]          </tr>
[49]        </xsl:if>
[50]        <xsl:if test="position() mod 2 = 0">
[51]          <tr class="Row2">
[52]                <td><xsl:value-of select="nachname"/></td>
[53]                <td><xsl:value-of select="vorname"/></td>
[54]                <td><xsl:value-of select="adresse/wohnort/@plz"/></td>
[55]                <td><xsl:value-of select="adresse/wohnort/@ort"/></td>
[56]                <td><xsl:value-of select="adresse/strasse"/></td>
[57]                <td><xsl:value-of select="kommunikation/festnetz"/></td>
[58]                <td><xsl:value-of select="kommunikation/mobil"/></td>
[59]                <td><xsl:value-of select="kommunikation/internet/email"/></td>
[60]                <td><xsl:value-of select="kommunikation/internet/web"/></td>
[61]                <td>
[62]                  <xsl:value-of select="geburtstag/tag"/>.
[63]                  <xsl:value-of select="geburtstag/monat"/>.
[64]                  <xsl:value-of select="geburtstag/jahr"/>
[65]                </td>
[66]          </tr>
[67]        </xsl:if>
[68]      </xsl:for-each>
[69]      </table>
[70]    </body>
[71] </html>
[72] </xsl:template>
[73] </xsl:stylesheet>
```

Kommentar:

- Wie Sie sehen, ist die XSL-Datei in XML codiert, denn XSL ist selbst eine XML-basierte Auszeichnungssprache.
- Die Zeilen 1, 2 und 73 sind in jeder XSL-Datei vorhanden. Dazwischen befinden sich die Transformationsanweisungen.
- Sie können im XSLT-Stylesheet beispielsweise angeben, dass der Wert für ein Element 'nachname' in den HTML-Code <td><xsl:value-of select="nachname"/></td> transformiert werden soll (Zeile 34 und Zeile 52). Aus dem Eintrag <nachname>Adler</nachname> in der XML-Datei wird nach der zutreffenden Transformationsregel der Zellen-Inhalt <td>Adler</td> in der HTML-Datei erzeugt.
- Selbst Java-Skripte können Sie einsetzen – wie in den Zeilen 10 und 14 notiert.
- Die Anweisungen in den Zeilen 32 und 50 – in Verbindung mit den CSS-Anweisungen in den Zeilen 33 und 51 – sorgen für alternative, unterschiedliche Hintergrundfarben der Tabellenzeilen. Im Gegensatz zu einer normalen CSS-Datei können Sie mit Kontrollstrukturen wie zum Beispiel Verzweigungen und Wiederholungen arbeiten, was XSLT in die Nähe von Programmiersprachen rückt.
- Besonders interessant ist die Zeile 31, in der eine Sortierung nach dem Attribut 'ort' vorgenommen wird. Auch die Sortierreihenfolge ist einstellbar. Den Attributnamen wird stets ein @-Zeichen vorangestellt.
- In den Zeilen 44 bis 46 sowie 62 bis 64 wird aus den einzelnen Angaben zum Geburtstag (Tag, Monat und Jahr) in der Tabellen-Zelle ein Datum erzeugt, formatiert und auch so angezeigt. Achten Sie auf den Punkt am Ende der Zeilen 44, 45, 62 und 63 (→ Abbildung 27.6.3.1)!

27.6.4 Projekt 2: XML → CSV

Im Beispiel 2 wird die Transformation XML → CSV umgesetzt. Die CSV-Datei wird danach einer leeren Tabellenkalkulationsdatei (LibreOffice Calc) als Argument übergeben.

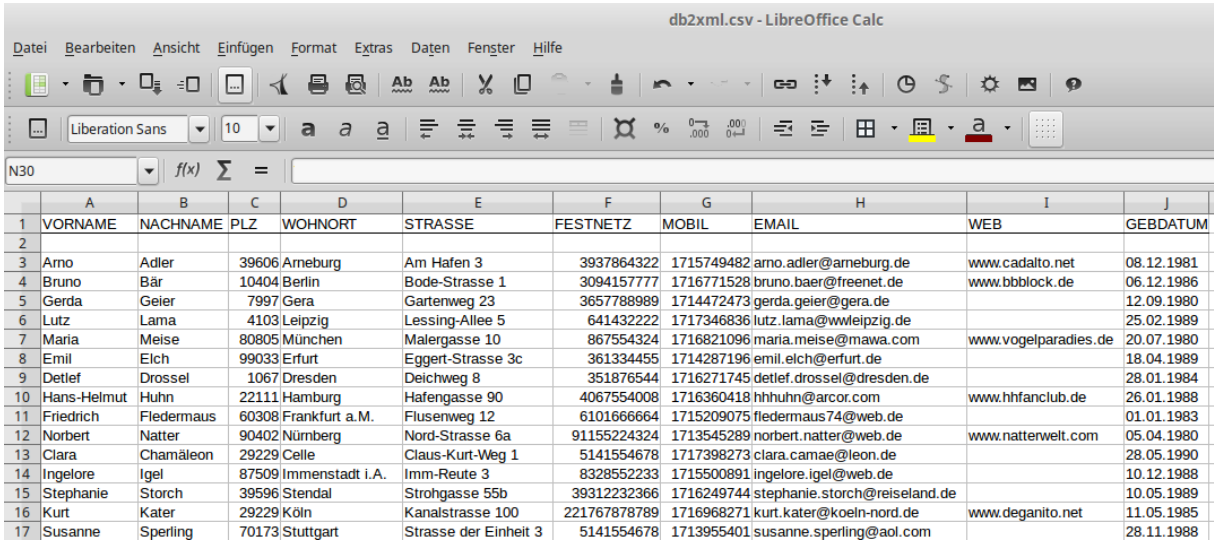
Quelltext für die Stylesheet-Datei:

```
[1]  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[2]  <xsl:output method="text" omit-xml-declaration="no" encoding="utf-8"/>
[3]
[4]    <xsl:strip-space elements="*" />
[5]
[6]    <!--  Erzeugen CSV-Kopfzeile  -->
[7]    <xsl:template match="/child::*">
[8]      <xsl:for-each select="*[1]/child::*">
[9]        <xsl:if test="position() != last()"><xsl:value-of select="translate(name(),
           'abcdefghijklmnopqrstuvwxyz', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ')"/>,</xsl:if>
[10]       <xsl:if test="position()  = last()"><xsl:value-of select="translate(name(),
           'abcdefghijklmnopqrstuvwxyz', ABCDEFGHIJKLMNOPQRSTUVWXYZ')"/><xsl:text>&#xA;</xsl:text></xsl:if>
[11]     </xsl:for-each>
[12]     <xsl:text>&#xA;</xsl:text>
[13]     <xsl:apply-templates/>
[14]   </xsl:template>
[15]
[16]   <!--  Erzeugen CSV-Inhalt  -->
[17]   <xsl:template match="/*/child::*">
[18]     <xsl:for-each select="child::*">
[19]       <xsl:if test="position() != last()"><xsl:value-of select="normalize-space(.)"/>,</xsl:if>
[20]       <xsl:if test="position()  = last()"><xsl:value-of select="normalize-
       space(.)"/><xsl:text>&#xA;</xsl:text></xsl:if>
[21]     </xsl:for-each>
[22]   </xsl:template>
[23]
[24] </xsl:stylesheet>
```

Kommentar:

- Die Feldnamen werden in den Zeilen 9 und 10 in Großbuchstaben transformiert.
- In der Zeile 9 wird jedem Feld-Namen das Feldtrennzeichen Komma nachgestellt. In der Zeile 10 wird am Ende jeder Zeile in der CSV-Datei kein Komma gesetzt – dafür aber ein Zeilenumbruch erzeugt.
- In der Zeile 19 wird jeder Feld-Wert von "-Zeichen umgeben. In der Zeile 20 wird am Ende jeder Zeile ein Zeilenumbruch erzeugt.

Das Ergebnis der Transformation XML → CSV kann sich sehen lassen:



	A	B	C	D	E	F	G	H	I	J
1	VORNAME	NACHNAME	PLZ	WOHNORT	STRASSE	FESTNETZ	MOBIL	EMAIL	WEB	GEBDATUM
2										
3	Arno	Adler	39606	Arneburg	Am Hafen 3	3937864322	1715749482	arno.adler@arneburg.de	www.cadalto.net	08.12.1981
4	Bruno	Bär	10404	Berlin	Bode-Strasse 1	3094157777	1716771528	bruno.baer@freenet.de	www.bbblock.de	06.12.1986
5	Gerda	Geier	7997	Gera	Gartenweg 23	3657788989	1714472473	gerda.geier@gera.de		12.09.1980
6	Lutz	Lama	4103	Leipzig	Lessing-Allee 5	641432222	1717346836	lutz.lama@wwleipzig.de		25.02.1989
7	Maria	Meise	80805	München	Malergasse 10	867554324	1716821096	maria.meise@mawa.com	www.vogelparadies.de	20.07.1980
8	Emil	Elch	99033	Erfurt	Eggert-Strasse 3c	361334455	1714287196	emil.elch@erfurt.de		18.04.1989
9	Detlef	Drossel	1067	Dresden	Deichweg 8	351876544	1716271745	detlef.drossel@dresden.de		28.01.1984
10	Hans-Helmut	Huhn	22111	Hamburg	Hafengasse 90	4067554008	1716360418	hhhuhn@arcor.com	www.hhfanclub.de	26.01.1988
11	Friedrich	Fledermaus	60308	Frankfurt a.M.	Flusenweg 12	6101666664	1715209075	fledermaus74@web.de		01.01.1983
12	Norbert	Natter	90402	Nürnberg	Nord-Strasse 6a	91155224324	1713545289	norbert.natter@web.de	www.natterwelt.com	05.04.1980
13	Clara	Chamäleon	29229	Celle	Claus-Kurt-Weg 1	5141554678	1717398273	clara.camae@leon.de		28.05.1990
14	Ingelore	Igel	87509	Immenstadt i.A.	Imm-Reute 3	8328552233	1715500891	ingelore.igel@web.de		10.12.1988
15	Stephanie	Storch	39596	Stendal	Strohgasse 55b	39312232366	1716249744	stephanie.storch@reiseland.de		10.05.1989
16	Kurt	Kater	29229	Köln	Kanalstrasse 100	221767878789	1716968271	kurt.kater@koeln-nord.de	www.deganito.net	11.05.1985
17	Susanne	Sperling	70173	Stuttgart	Strasse der Einheit 3	5141554678	1713955401	susanne.sperling@aol.com		28.11.1988

Abbildung 27.6.4.1: Kalkulationstabelle in LibreOfficeCalc

27.6.5 Projekt 3: XML → HTML

Die Transformation im Beispiel 3 wird erst dann vorgenommen, nachdem eine XML-Datei mit *aktuellen Daten aus einem RSS-Feed* erzeugt wurde. Das Besondere für dieses dritte Projekt besteht weiterhin darin, dass in die HTML5-Datei Bilder eingefügt werden, denen ein Link zugeordnet wird. Der Inhalt der XSLT-Datei ist überschaubar kurz und wird vollständig angegeben:

```
[1]  <?xml version="1.0" encoding="utf-8"?>
[2]  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
[3]  <xsl:output method="html" encoding="utf-8" indent='yes'/>
[4]  <xsl:template match="/">
[5]    <html>
[6]      <head>
[7]        <title>ARD RSS-FEED</title>
[8]        <meta charset="utf-8"></meta>
[9]        <link rel="stylesheet" type="text/css" href="feeds.css"></link>
[10]     </head>
[11]     <body>
[12]     <!-- Content - Begin -->
[13]       <xsl:for-each select="feeds/feed">
[14]         <p><xsl:value-of select="title"/></p>
[15]         <a href="{link}"><img src="{imagelink}"/></a>
[16]         <br />
[17]         <xsl:value-of select="description"/>
[18]         <a class="arrow" href="{link}">mehr ...</a>
[19]         <br />
[20]         <hr class="redline"></hr>
[21]       </xsl:for-each>
[22]     <!-- Content - End -->
[23]     </body>
[24]   </html>
[25] </xsl:template>
[26] </xsl:stylesheet>
```

Kommentar:

- In der Zeile 9 wird eine CSS-Datei eingebunden.
- In einer For-Each-Kontrollstruktur (Zeilen 13 bis 21) werden die aus jedem Feed-Eintrag ausgewählten Daten in ein HTML-Konstrukt aus Bild, Text und Hyperlinks transformiert.
- Beachten Sie: Hyperlinks werden in XSLT stets mit geschweiften Klammern notiert.
- Jeder Feed-Eintrag in der HTML5-Datei schließt nach einer Leerzeile mit einer horizontalen roten Linie ab.

Eine Bildschirmkopie des Programm-Fensters kann an dieser Stelle nicht geboten werden, da die ARD auf Nachfrage des Autors angab, dass es Probleme mit den Bildrechten geben könnte, da auch die ARD einige Bilder aus anderen Quellen zukauft. Deshalb gilt: Starten Sie das Programm 3 und sehen Sie selbst!