27.1 Daten-Objekt-Modell

Das *Document Object Model* (DOM) repräsentiert Elemente, Attribute und Text innerhalb von Elementen als Knoten in einer Baum-Struktur.

27.1.1 Das Modell

Wenn ein XmlDocument-Objekt ein XML-Dokument in Stringform (XmlDocument.ToString() oder XmlDocument.ToString(True)) parst, erzeugt es einen Baum aus XmlNode-Objekten. Diese XmlNode-Objekte haben typischerweise einen Datentyp, der nicht exakt XmlNode ist, sondern von der XmlNode-Klasse erbt, wie zum Beispiel XmlTextNode für Text-Knoten oder XmlElement für Tags. Wenn ein Knoten k dieses Baumes zu einem Tag ("<tag>...</tag>" oder "<tag />" im Fall eines inhaltslosen Tags) gehört, nennt man k ein Element und es ist vom Typ XmlElement. In XML können nur Elemente Kind-Knoten haben. Wenn also ein Knoten in diesem Baum Kind-Knoten hat, ist er notwendigerweise vom Typ XmlElement. Die Kind-Knoten des Elements k sind dann genau die XML-Knoten, die innerhalb von "<tag>...</tag>" verschachtelt sind. Daher kommt dem XmlElement eine besondere Bedeutung zu und es gibt Extra-Versionen bestimmter Methoden (wie zum Beispiel FirstChildElement von FirstChild). Die XmlElemente sind Knoten, an denen der Baum in die Tiefe wächst und alle Nicht-Element-Knoten sind Blätter des Baumes.

Diese XML-Struktur

```
<tag>
 <tag2>abc</tag2>
   <tag3 />
   DEF
</tag>
```

würde formal den folgenden Baum produzieren, wobei die Einrückung die Eltern-Kind-Relation darstellt:

```
XmlElement: tag
  +-- XmlElement: tag2
      +-- XmlTextNode: abc
  +-- XmlElement: tag3
  +-- XmlTextNode: DEF
```

An XmlElement-Knoten kann der Baum tiefer verschachtelt sein als bei 'tag' und 'tag2' oder auch nicht – wie beim leeren 'tag3'. Es ist aber ausgeschlossen, dass unter XmlTextNode-Knoten noch Kind-Knoten kommen!

Der Schlüssel zur Arbeit mit der Klasse XmlDocument liegt darin, sich das Dokument als einen *Baum* von XML-Knoten vorzustellen. Wenn Sie ein Element hinzufügen wollen, dann hilft Ihnen die Wurzel des Baumes (also das XmlDocument-Objekt) an sich nicht weiter. Sie müssen zu dessen Knoten (XmlNode bzw. XmlElement) gehen. Nur dort können Sie dem Baum neue Knoten hinzufügen. Das spiegelt sich darin wider, dass XmlDocument so gut wie keine Methoden zur Modifikation des Baumes hat. Diese Methoden sind fast alle an XmlElement und XmlNode gebunden.

Das Verhältnis von XmlNode und XmlElement ist übrigens das Folgende: XmlNode ist die Elternklasse von XmlElement. Ein XmlNode kann zum Beispiel ein Kommentar, ein Text, ein CDATA oder ein Element im engeren Sinn sein. Mit XmlElement ist ein Tag "<name>inhalt</name>" gemeint. Ein XmlElement ist also insbesondere Etwas, das Kind-Knoten haben kann. Das erklärt, dass man im Zusammenhang mit der Modifikation des Baumes oft von XmlElement-Objekten spricht.

Jedes XmlElement ist ein XmlNode, aber nicht jeder Knoten (XmlNode) ist ein XmlElement. XmlElement ist nur eine Art XmlNode. Andere sind zum Beispiel XmlAttribute oder XmlText. Ein Element ist somit Teil der formalen Definition eines wohlgeformten XML-Dokuments, während ein Knoten als Teil des Dokument-Objekt-Modells für die Verarbeitung von XML-Dokumenten definiert ist. Ein Knoten ist ein Teil des DOM-Baums, ein Element ist eine bestimmte Art von Knoten.

Es ist bei der Analyse mit XML-Dateien von Vorteil, wenn man deren Struktur als Baum von XML-Knoten kennt oder Informationen zu den einzelnen Knoten besitzt. In diesem Kapitel werden Ihnen zwei Projekte vorgestellt, die genau das leisten:

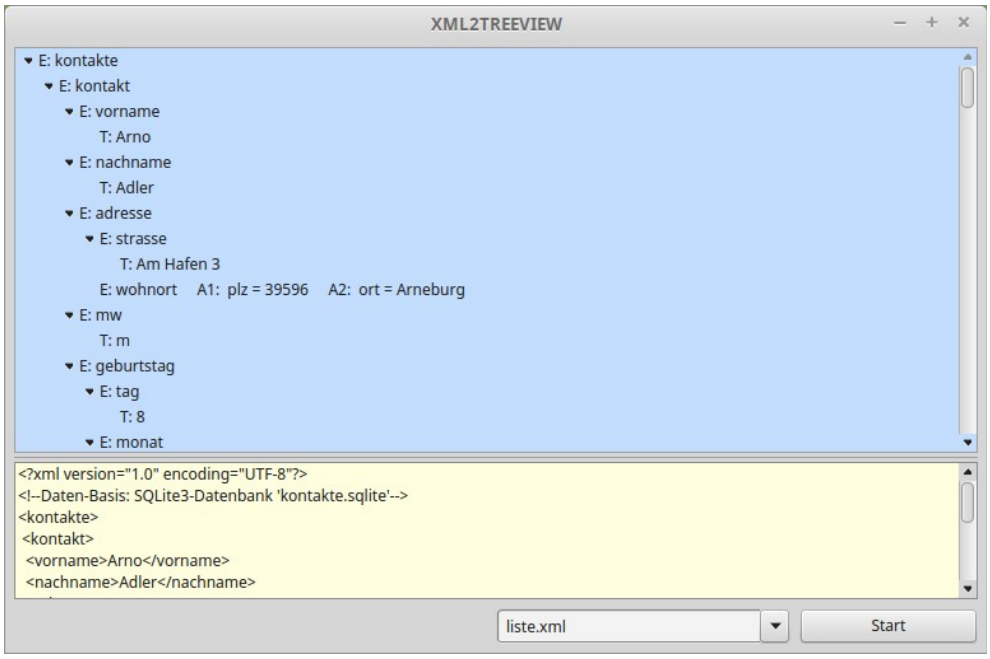

Abbildung 27.1.1.1: Projekt 1: Ansicht Knoten-Struktur – dargestellt in einer TreeView

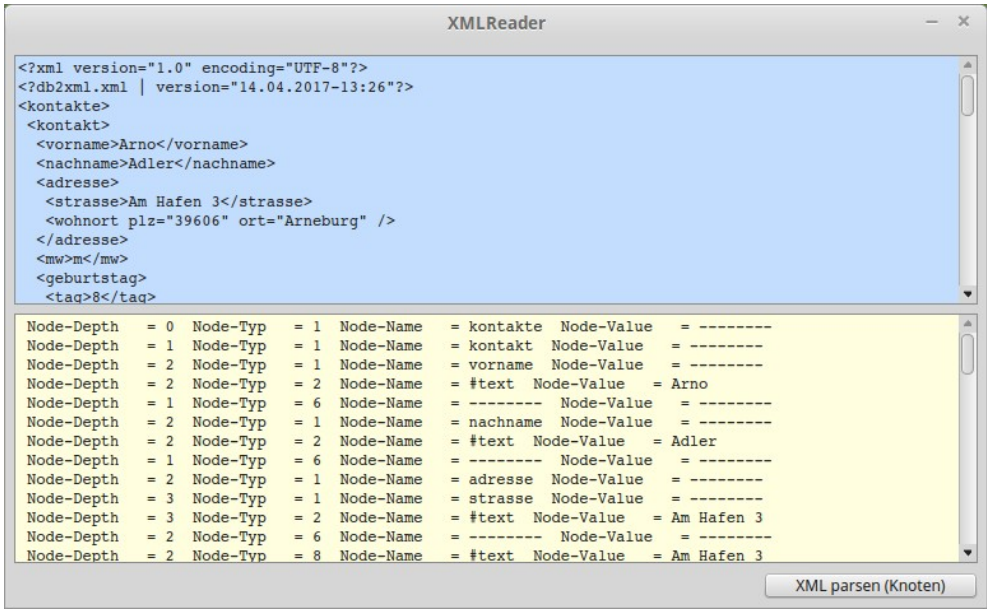

Abbildung 27.1.1.2: Projekt 2: Knoten-Informationen

27.1.2 Klasse XmlDocument

Objekte der Klasse XmlDocument können Sie erzeugen:

```
Dim hXmlDocument As XmlDocument
hXmlDocument = New XmlDocument([ FileName As String ])
```

Der folgende Quelltext

```
Public Sub btnXMLWriteDOM_Click()

  Dim hXMLContent As String

  hXMLDocument = New XmlDocument
```

```
  hXMLContent = hXMLDocument.ToString(True)
  txaXML.Text = hXMLContent
  File.Save(xmlFilePath, hXMLContent)

End
```

erzeugt den Inhalt eines XML-Dokuments mit einem *fest vorgegebenen Root-Element* <xml>..</xml>:

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

Wie Sie ein benutzerdefiniertes Root-Element erzeugen wird im Beispiel 27.1.4 beschrieben.

27.1.2.1 Eigenschaften XmlDocument

Die Klasse XmlDocument verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
All	XmlNode[]	Gibt ein Node-Array mit allen Knoten des DOM-Baums zurück.
Content	String	Gibt den kompletten Inhalt des Dokuments zurück oder setzt den Inhalt.
Root	XmlElement	Gibt das Root-Element zurück oder setzt das Root-Element.

Tabelle 27.1.2.1.1 : Eigenschaften der Klasse XmlDocument

27.1.2.2 Methoden XmlDocument

Die Klasse *XmlDocument* besitzt diese Methoden:

Methode	Rückgabetyp	Beschreibung
Open(FileName As String)	.	Öffnet ein XML-Dokument mit dem Pfad 'FileName'.
Save(FileName As String [, Indent As Boolean])	-	Speichert den Inhalt des XML-Dokuments unter dem Datei-Pfad 'FileName'. Wenn der optionale Parameter 'Indent' mit dem Wert True angegeben wird, dann werden die Elemente passend eingerückt.
CreateElement(TagName As String)	XmlElement	Die Funktion erzeugt ein neues Element mit dem Namen 'TagName' und gibt es als XmlElement zurück.
ToString([Indent As Boolean])	String	Die Funktion gibt den Inhalt des XML-Dokuments als String zurück. Wenn der optionale Parameter 'Indent' mit dem Wert True angegeben wird, dann werden die Elemente eingerückt.
FromString(Data As String)	-	Das XML-Dokument wird mit dem Inhalt 'Data' gefüllt. Bestehender Inhalt wird überschrieben.
HtmlFromString(Data As String)	-	Das XML-Dokument wird mit dem Inhalt 'Data' gefüllt. Bestehender Inhalt wird überschrieben.
GetElementsByNamespace(Namespace As String [, Mode As Integer, Depth As Integer])	XmlElement[]	Gibt alle Elemente des Dokuments in einem Array zurück, deren Element-Name mit dem Parameterwert 'Namespace' übereinstimmt. Der Mode-Parameter definiert die verwendete Vergleichsmethode. Unterstützt werden GB.Binary, GB.IgnoreCase und GB.Like. Das Depth-Argument definiert, wo die Suche gestoppt werden soll: Bei einem negativen Wert wird nur am Ende des Baums gestoppt (Standard), 1 überprüft nur das Root-Element, 2 überprüft nur die direkten Kinder des Root-Elements und so weiter.
GetElementsByTagName(TagName As String [, Mode As Integer, Depth As Integer])	XmlElement[]	Gibt alle Elemente des Dokuments in einem Array zurück, deren Element-Name mit dem Parameterwert 'TagName' übereinstimmt. Der Mode-Parameter defi-

Methode	Rückgabetyp	Beschreibung
		niert die verwendete Vergleichsmethode. Unterstützt werden GB.Binary, GB.IgnoreCase und GB.Like. Das Depth-Argument definiert, wo die Suche gestoppt werden soll: Bei einem negativen Wert wird nur am Ende des Baums gestoppt (Standard), 1 überprüft nur das Root-Element, 2 überprüft nur die direkten Kinder des Root-Elements und so weiter.

Tabelle 27.1.2.2.1 : Methoden der Klasse XmlDocument

27.1.3 Klasse XmlNode

Die Node-Klasse beschreibt den Basis-Typ für das gesamte Dokument-Objekt-Modell (DOM). Sie repräsentiert einen einzelnen Knoten im Dokument-Baum. Die Klasse XmlNode hat die folgenden Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Attributes	.XmlElementAttributes	Gibt ein Node-Array mit allen Knoten des DOM-Baums zurück. Zurückgegeben wird eine virtuelle Collection aller Attribut-Knoten des aktuellen Knotens. Der Daten-Typ wird durch die Klasse .XmlElementAttributes beschrieben, die über die Eigenschaften Count, Name und Value verfügt.
Children	XmlNode[]	Gibt ein Array zurück, das alle Kind-Knoten enthält. Wenn es keine Kind-Knoten gibt, wird ein leeres Array zurück gegeben.
ChildNodes	XmlNode[]	Synonym für die Children-Eigenschaft.
Element	XmlElement	Wenn der aktuelle Knoten ein Element ist, dann wird das Element zurück gegeben. Im anderen Fall wird NULL zurück gegeben. Wenn Sie nur wissen wollen, ob der aktuelle Knoten ein Element ist, dann kann das mit der Eigenschaft *XmlNode.IsElement* geprüft werden.
IsCDATA	Boolean	Diese Eigenschaft gibt True zurück, wenn der aktuelle Knoten ein CDATA-Knoten ist, sonst False.
IsComment	Boolean	Diese Eigenschaft gibt True zurück, wenn der aktuelle Knoten ein Kommentar-Knoten ist, sonst False.
IsElement	Boolean	Diese Eigenschaft gibt True zurück, wenn der aktuelle Knoten ein Element-Knoten ist, sonst False.
IsText	Boolean	Diese Eigenschaft gibt True zurück, wenn der aktuelle Knoten ein Text-Knoten ist, sonst False.
Name	String	Es wird der Name des Knotens zurückgegeben. Achtung: Der Wert dieser Eigenschaft kann je nach Knoten-Typ variieren: * Element-Knoten: Zurückgegeben wird der Element-Name. * Text-Knoten: Zurückgegeben wird #text. * CDATA-Knoten: Zurückgegeben wird #cdata. * Kommentar-Knoten: Zurückgegeben wird #comment. * Attribut-Knoten: Zurückgegeben wird der Name des Attribut-Knotens.
TextContent	String	Gibt den Text-Inhalt eines Knotens zurück oder legt diesen fest. Für Textknoten, Kommentare, CDATA-Abschnitte oder Attribute ist es der Wert des Knotens. Für Knoten, die Kinder haben können, ist dies eine Verkettung des TextContent-Eigenschaftswertes jedes untergeordneten Knotens. Das gilt nicht für Kommentare. Dies ist die leere Zeichenfolge, wenn der Knoten keine Kinder hat. Für Knoten, die Kinder haben können, wird die Festlegung dieser Eigenschaft alle Kinder des Knotens entfernen! Das Setzen dieser Eigenschaft mit einem Nullstring hat keinen Effekt.
Value	String	Synonym für die TextContent.Eigenschaft.
Parent	XmlElement	Gibt das übergeordnete Element des aktuellen Knotens zurück. Alle Knoten können ein Elternteil haben. Wenn jedoch ein Knoten

Eigenschaft	Datentyp	Beschreibung
		gerade erzeugt wurde und noch nicht dem Baum hinzugefügt wurde oder wenn er aus dem Baum entfernt wurde, so gibt diese Eigenschaft NULL zurück.
Previous	XmlNode	Gibt den Knoten unmittelbar vor dem aktuellen Knoten zurück. Wenn es keinen solchen Knoten gibt, gibt diese Eigenschaft NULL zurück.
PreviousSibling	XmlNode	Synonym für die Previous-Eigenschaft.
Next	XmlNode	Gibt den Knoten unmittelbar nach dem aktuellen Knoten zurück. Wenn es keinen solchen Knoten gibt, gibt diese Eigenschaft NULL zurück.
NextSibling	XmlNode	Synonym für die Next-Eigenschaft.
OwnerDocument	XmlDocument	Gibt das mit dem aktuellen Knoten verknüpfte Dokument-Objekt zurück. Wenn es kein mit dem Knoten assoziiertes Dokument-Objekt gibt, gibt diese Eigenschaft NULL zurück. Anders als nach der W3C-Spezifikation können Knoten existieren, ohne dass ein Dokument zugeordnet ist, da sie einfach mit dem NEW-Befehl oder einem XmlReader erzeugt werden können. Jedoch wird ein Dokument automatisch dem Knoten zugeordnet, wenn die AppendChild-(oder Geschwister-)Methode von einem Element verwendet wird, dem bereits ein Dokumentobjekt zugeordnet ist.

Tabelle 27.1.3.1 : Eigenschaften der Klasse XmlNode

Die Klasse XmlNode verfügt u.a. über diese Konstanten:

```
Const ElementNode As Integer = 1
Const TextNode As Integer = 2
Const CommentNode As Integer = 3
Const CDATASectionNode As Integer = 4
Const AttributeNode As Integer = 5
```

27.1.3.1 Methoden

Die Klasse *XmlNode* besitzt diese Methoden:

Methode	Rückgabetyp	Beschreibung
NewElement (EName As String [, EValue As String])	.	Erzeugt einen neuen Element-Knoten, setzt (optional) den Element-Namen und den Wert mit den Argumenten 'EName' und 'EValue' und fügt es dem aktuellen Knoten hinzu. Wenn der Knoten kein Element ist, hat diese Methode keine Wirkung.
NewAttribute (AName As String, AValue As String)	-	Erzeugt einen neuen Attribut-Knoten, setzt den Attribut-Namen und den Wert mit den Argumenten 'AName' und 'AValue' und fügt es dem aktuellen Knoten hinzu. Wenn der Knoten kein Element ist, hat diese Methode keine Wirkung.
ToString ([Indent As Boolean])	String	Gibt eine String-Darstellung des aktuellen Knotens zurück, da es sich um eine reguläre XML-Datei handelt. Wenn die Einzugseigenschaft auf True gesetzt ist, wird die Ausgabe eingerückt. Standardmäßig ist dieses optionale Argument 'Indent' auf False gesetzt.
SetUserData (Key As String, Value As Variant)	-	Verknüpft ein Objekt mit einem Schlüssel auf diesen Knoten.
GetUserData (Key As String)	Variant	Ruft das Objekt ab, dem ein Schlüssel auf diesen Knoten zugeordnet ist.

Tabelle 27.1.3.1.1 : Methoden der Klasse XmlNode

Im folgenden Abschnitt finden Sie Hinweise zu den Methoden SetUserData (Key As String, Value As Variant) und GetUserData (Key As String):

Sub SetUserData (Key As String, Value As Variant)
Speichert ein Objekt mit einem Schlüssel 'Key' und dem Wert 'Value' - der benutzerdefinierte Daten enthält - zu diesem Knoten. Das Objekt kann später vom Knoten abgerufen werden, indem die Methode 'GetUserData(Key As String)' mit dem selben Schlüssel aufgerufen wird.

Function GetUserData (Key As String) As Variant
Ruft das Objekt ab, dem ein Schlüssel 'Key' auf diesen Knoten zugeordnet ist. Das Objekt muss zuerst mit diesem Knoten verbunden werden, indem die Methode 'SetUserData' mit dem selben Schlüssel aufgerufen wird. Wenn keine Daten mit dem angegebenen Schlüssel verknüpft wurden, gibt diese Methode NULL zurück.

Anmerkungen:

- An jedem XmlNode können Sie als Programmierer optional mit den Methoden SetUserData(..) und GetUserData(..) Benutzerdaten verwalten. Der Einsatz von Benutzerdaten (Typ und Wert) ist ganz dem Programmierer überlassen. Die Verwendung der Konstanten, Eigenschaften, Methoden und Ereignisse der XmlNode-Klasse ist von diesen Benutzerdaten völlig unabhängig.
- Das Prinzip des Einsatzes von benutzerdefinierten Daten kennen Sie bereits von der Tag-Eigenschaft graphischer Steuerelemente (Control-Klasse).
- Sie können die benutzerdefinierten Daten verwenden, um zum Beispiel Knoten zu markieren, ihnen eine eindeutige ID zu geben oder anwendungsspezifische Daten direkt am Knoten zu speichern und später ereignisorientiert wieder auslesen!
- Während die Eigenschaft *Control.Tag* in den graphischen Steuerelementen einen beliebigen Variant-Wert speichert, funktionieren die benutzerdefinierten Daten in XmlNode anders. Hier können Sie beliebig viele Variants speichern, indem Sie jeden Variant-Wert mit einem Schlüssel verknüpfen. Der Benutzer-Datenspeicher eines Xml-Knoten-Objekts arbeitet genau so wie der Daten-Typ Collection, denn SetUserData(..) und GetUserData(..) verhalten sich wie ein Objekt vom Typ Collection.

27.1.4 Projekte

Es werden Ihnen in den folgenden Abschnitten unterschiedliche Projekte vorgestellt. Dabei geht es um das Schreiben, das Lesen (Parsen) und das Modifizieren des Inhalts von XML-Dokumenten, vorwiegend unter Verwendung der Klasse XmlDocument. Der Inhalt der XML-Dokumente ist entweder statisch oder wird erst zur Laufzeit erzeugt.

27.1.4.1 Projekt 1

Im Projekt 1 (xmldom_write) wird Ihnen gezeigt, wie ein XML-Dokument unter Verwendung der Klasse XmlDocument neu geschrieben wird. Die Daten werden in einer Struktur (Datentyp Struct) programmintern zur Verfügung gestellt. Dieser Datentyp sichert u.a. den Vorteil, die einzelnen Daten über ihren Namen anzusprechen statt über einen anonymen Index. Es wird Ihnen nur die wichtigste Prozedur vorgestellt:

```
Private Sub WriteXMLDOM()
  Dim hXMLComment As XmlCommentNode
  Dim hXMLElement As XmlElement
  Dim hXMLContent As String
  Dim i As Integer

  hXMLDocument = New XmlDocument
' Statt <xml>...</xml> wird ein korrektes, aber frei definiertes Root-Element erzeugt!
  hXMLDocument.FromString("<?xml version=\"1.0\" encoding=\"utf-8\" ?><kontakte />")
  hXMLComment = New XmlCommentNode("Daten-Basis: Daten-Array aDataSet (DataSet[])")
  hXMLDocument.Root.AppendChild(hXMLComment)

  For i = 0 To aDataSet.Max
  ' Das Element <kontakt> zum Baum hinzufügen und hXMLElement auf dieses Element setzen.
    hXMLElement = New XmlElement("kontakt")
    hXMLDocument.Root.AppendChild(hXMLElement)
  ' <vorname> hat nur einen Wert und verschachtelt nicht weiter.
```

```
    hXMLElement.NewElement("vorname", aDataSet[i].Vorname)
  ' <nachname> hat nur einen Wert und verschachtelt nicht weiter.
    hXMLElement.NewElement("nachname", aDataSet[i].Nachname)
  ' Das Element <adresse> hat Kind-Elemente --> absteigen
    hXMLElement.NewElement("adresse")
    hXMLElement = hXMLElement.LastChildElement
    hXMLElement.NewElement("strasse", aDataSet[i].Strasse)
    hXMLElement.NewElement("wohnort")
  ' Zwei Attribute sind zu setzen --> absteigen.
    hXMLElement = hXMLElement.LastChildElement
    hXMLElement.NewAttribute("plz", aDataSet[i].PLZ)
    hXMLElement.NewAttribute("ort", aDataSet[i].Ort)
  ' Das Element <wohnort> ist geschrieben --> aufsteigen
    hXMLElement = hXMLElement.Parent ' </wohnort>
    hXMLElement = hXMLElement.Parent ' </adresse>
    hXMLElement.NewElement("tag", Day(aDataSet[i].GebDatum))
    hXMLElement.NewElement("monat", Month(aDataSet[i].GebDatum))
    hXMLElement.NewElement("jahr", Year(aDataSet[i].GebDatum))
    hXMLElement.NewElement("kommunikation")
    hXMLElement = hXMLElement.LastChildElement
    hXMLElement.NewElement("festnetz", aDataSet[i].TFestnetz)
    hXMLElement.NewElement("mobil", aDataSet[i].TMobil)
    hXMLElement.NewElement("internet")
    hXMLElement = hXMLElement.LastChildElement
    hXMLElement.NewElement("email", aDataSet[i].EMail)
    hXMLElement.NewElement("web", aDataSet[i].Web)
  ' Die folgenden 3 Zuweisungen sind nicht notwendig, da hXMLElement nicht mehr
  ' bearbeitet wird. Der DOM-Baum ist auch ohne diese 3 folgenden Zuweisungen korrekt.
  ' Es wird nur gezeigt, entlang welcher Tags man wieder zur Wurzel am DOM-Baum kommt.
    hXMLElement = hXMLElement.Parent ' </internet>
    hXMLElement = hXMLElement.Parent ' </kommunikation>
    hXMLElement = hXMLElement.Parent ' </kontakt>
  Next
  hXMLElement = hXMLElement.Parent ' </kontakte>
  hXMLContent = hXMLDocument.ToString(True) ' Das komplette XML-Dokument wird erzeugt
  txaXML.Text = hXMLContent ' Der XML-Dokument-Inhalt wird angezeigt
  File.Save(xmlFilePath, hXMLContent) ' Der XML-Dokument-Inhalt wird gespeichert
End
```

Während das XmlDocument im Speicher liegt, werden alle Eigenschaften und Inhalte der Knoten in den Eigenschaften der verwendeten XML-Gambas-Objekte gespeichert. Erst eine Methode wie XmlDocument.ToString() geht diesen Baum ab und produziert (rekursiv) aus allen Attributen, Eigenschaften und Kind-Knoten ein *wohlgeformtes* XML-Dokument vom Daten-Typ String.

Abbildung 27.1.4.1.1: Projekt 1: Hinweis zur Daten-Basis

Abbildung 27.1.4.1.2: Projekt 1: Ausschnitt Inhalt XmlDocument

27.1.4.2 Projekt 2

Im vorgestellten Projekt (xmldom_read_addresses) sollen aus Kontaktdaten – gespeichert in einer XML-Datei – *ausgewählte Daten* ausgelesen, bearbeitet werden und dann in einer TextArea angezeigt werden. Die ausgelesenen, aufbereiteten und in einer Text-Datei abgespeicherten Daten könnten zum Beispiel als Basis für den Druck von Adress-Aufklebern dienen:

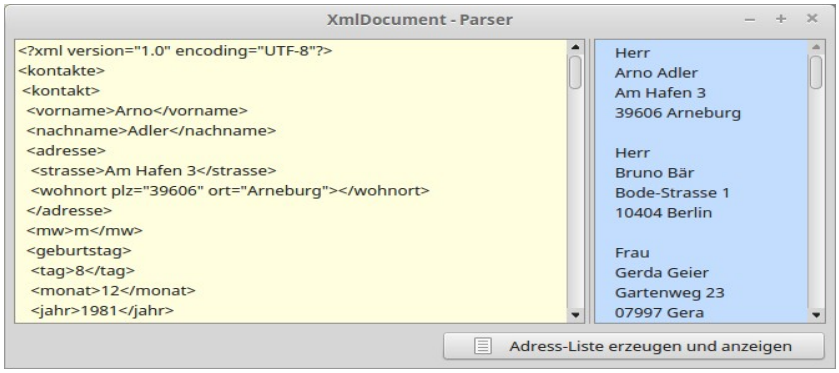

Abbildung 27.1.4.2.1: Projekt 2: XML-DOM-Parser

Der Quelltext wird vollständig angegeben und kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Public hXMLDocument As XmlDocument
[4]  Public sXMLPath As String = "files/list.xml"
[5]
[6]  Public Sub Form_Open()
[7]    ShowXMLContent()
[8]    txaXML.Pos = 0
[9]    HSplit1.Layout = [5, 2]
[10] End
[11]
[12] Public Sub btnShowRecords_Click()
[13]
[14]   Dim i As Integer
[15]   Dim asMatrix As String[]
[16]   Dim avRecords As New Variant[]
[17]   Dim sSpace As String = String$(4, " ")
[18]
```

```
[19]   avRecords = GetRecords()
[20]
[21]   For i = 0 To avRecords.Max
[22]     asMatrix = New String[]
[23]     asMatrix = avRecords[i]
[24]     txaList.Insert(sSpace & asMatrix[0] & gb.NewLine)
[25]     txaList.Insert(sSpace & asMatrix[1] & " " & asMatrix[2] & gb.NewLine)
[26]     txaList.Insert(sSpace & asMatrix[3] & gb.NewLine)
[27]     txaList.Insert(sSpace & asMatrix[4] & " " & asMatrix[5] & gb.NewLine)
[28]     txaList.Insert(gb.NewLine)
[29]   Next
[30]
[31]   File.Save(Application.Path &/ "files/addresslist.txt", txaList.Text)
[32]
[33] End
[34]
[35] Private Sub ShowXMLContent()
[36]   hXMLDocument = New XmlDocument
[37]   hXMLDocument.Open(sXMLPath)
[38]   txaXML.Insert(hXMLDocument.Content)
[39] End
[40]
[41] Private Function GetRecords() As Variant[]
[42]
[43]   Dim i As Integer
[44]   Dim xeElements As XmlElement[]
[45]   Dim asRecord As String[]
[46]   Dim avRecords As New Variant[]
[47]
[48]   txaList.Clear()
[49]
[50]   hXMLDocument = New XmlDocument
[51]   hXMLDocument.Open(sXMLPath)
[52]
[53]   xeElements = New XmlElement[]
[54]   xeElements = hXMLDocument.GetElementsByTagName("kontakt")
[55]   For i = 0 To xeElements.Max
[56]     asRecord = New String[]
[57]     asRecord.Add(xeElements[i].GetChildrenByTagName("vorname")[0].Value)
[58]     asRecord.Add(xeElements[i].GetChildrenByTagName("nachname")[0].TextContent)
[59]     asRecord.Add(xeElements[i].GetChildrenByTagName("strasse")[0].TextContent)
[60]     asRecord.Add(xeElements[i].GetChildrenByTagName("wohnort")[0].Attributes["plz"])
[61]     asRecord.Add(xeElements[i].GetChildrenByTagName("wohnort")[0].Attributes["ort"])
[62]     If xeElements[i].GetChildrenByTagName("mw")[0].Value = "w" Then
[63]        asRecord.Add("Frau", 0)
[64]     Else
[65]        asRecord.Add("Herr", 0)
[66]     Endif
[67]     avRecords.Add(asRecord)
[68]   Next
[69]
[70]   Return avRecords
[71]
[72] End
```

Kommentar:

- Der vorgestellte Parser liest den Inhalt der XML-Datei komplett und filtert nur die Daten heraus, die der Anwender weiterverarbeiten möchte. Diese Daten werden in Datensätzen erfasst und jeweils in einem Variant-Array gespeichert (Zeile 53 und Zeilen 55 bis 68).
- Dabei werden diese Rohdaten für den Anwendungszweck 'Adress-Aufkleber' aufbereitet. So wird je nach Wert für das mw-Element entschieden (Zeilen 62 bis 66), ob die Anrede Mann oder Frau gelten soll.
- Die so aufbereiteten Daten werden anschließend in einer TextArea angezeigt (Zeilen 21 bis 29).
- In der Zeile 31 wird die erzeugte Adressliste in der Datei *addresslist.txt* abgespeichert.
- Ein Besonderheit zeigt sich beim Filtern der Original-Daten nach Ort und Postleitzahl, da diese in Attributen mit den Attributnamen 'plz' und 'ort' stecken.

27.1.4.3 Projekt 3

Im Gegensatz zu den beiden ersten Projekten werden im Projekt 3 (xmldom_read_call) die Daten über einen HTTP-Client dynamisch aus einer speziellen Datenbank für Amateurfunkrufzeichen in ein XML-Dokument vom Typ XmlDocument importiert und in einer TextArea angezeigt. Ein Parser extrahiert in 10 unterschiedlichen Varianten jeweils ausgewählte Daten aus dem XML-Dokument und fügt diese in die TextArea ein. Das dritte Projekt wurde von Claus Dietrich entwickelt, kodiert und erprobt.

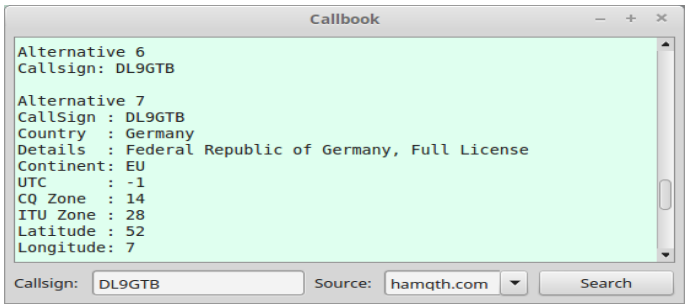

Abbildung 27.1.4.3.1: Projekt 3: XML-DOM-Parser - Callbook

27.1.4.4 Projekt 4

Im Projekt xmldom_read_write_modification wird eine XML-Datei vorgegeben, deren Inhalt gezielt geändert werden soll. Anschließend wird gegen ein XML-Schema – gespeichert in der XSD-Datei magazin_m.xsd – geprüft, ob die geänderte XML-Datei ein gültiges Dokument ist. Zur Prüfung wird das Programm *xmllint* in einer Shell-Instruktion eingesetzt:

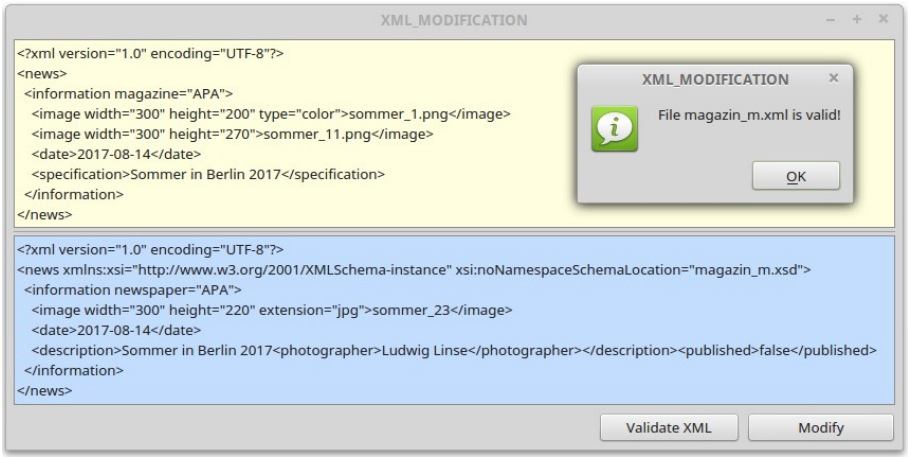

Abbildung 27.1.4.4.1: Projekt 4: XML-DOM-Update

Alle o.a. Projekte sowie weitere Projekte finden Sie als Archive im Download-Bereich.