27.7 XML-EXPLORER

Die Klasse XmlExplorer (gb.xml) ist etwas Besonderes, da es sich um eine Kombination aus XmlReader und XmlDocument handelt: Die SAX-Schnittstelle ist dieselbe wie bei XmlReader, die Klasse funktioniert aber intern wie die DOM-API. Wenn Sie ein Dokument öffnen, so wird es geladen und über XmlDocument geparst. Rufen Sie die Read()-Methode auf, dann wird ein (interner) Zeiger auf den nächsten Knoten verschoben. Sie können daraufhin auf den aktuellen Knoten mit Hilfe der Node-Eigenschaft wie bei XmlReader zugreifen.

Stellen Sie während der Arbeit mit der XmlReader-API in einem Ihrer Projekte fest, dass Sie aus Leistungsgründen doch besser die DOM-API verwenden sollten, dann müssen Sie nur alle *XmlReader*-Vorkommen im Quelltext durch *XmlExplorer* ersetzen. Dadurch sind Sie nicht gezwungen Ihren Quelltext völlig neu zu schreiben, sondern ihn nur partiell zu ändern.

Wenn Sie aus irgendeinem Grund Ihre XML-Dokumente linear – vom ersten Knoten bis zum letzten – untersuchen möchten und das gesamte Dokument im Speicher halten wollen, dann könnte die Verwendung der Klasse XmlExplorer eine gute Lösung für Sie sein.

27.7.1 Eigenschaften und Methoden

Ein neues XmlExplorer-Objekt können Sie so erzeugen:

```
Dim hXmlExplorer As XmlExplorer
hXmlExplorer = New XmlExplorer ( [ Document As XmlDocument] )
```

Die Klasse XmlExplorer besitzt die folgenden Eigenschaften und Methoden, die bereits in den vorangegangenen Kapiteln beschrieben wurden:

- Property Document As XMLDocument
- Property Read Eof As Boolean
- Property Read Node As XmlNode
- Property Read ReadFlags As .XmlExplorerReadFlags
- Property Read State As Integer
- Sub Load (Document As XmlDocument)
- Sub Open (Path As String)
- Function Read ( ) As Integer

27.7.2 Beispiel XML-Parser (XmlExplorer)

Im vorgestellten Beispiel sollen aus Kontaktdaten – gespeichert in einer XML-Datei – ausgewählte Daten ausgelesen 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:

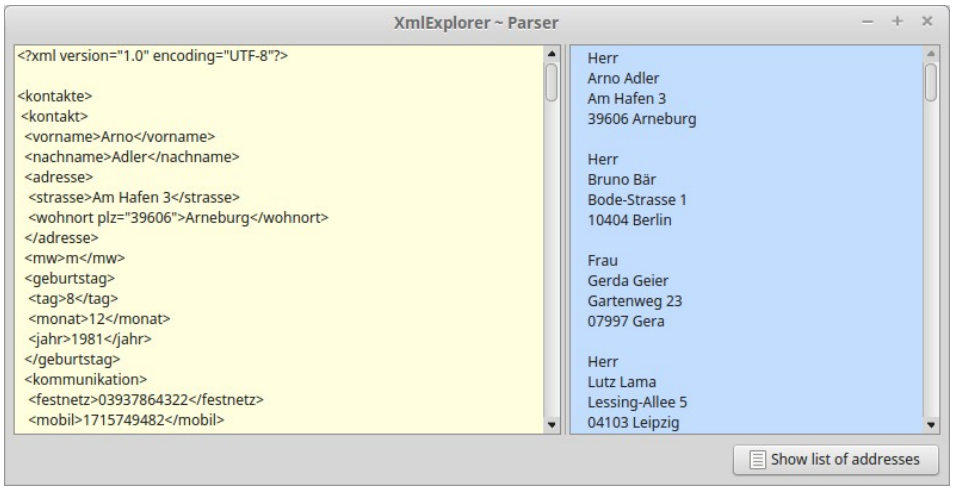

Abbildung 27.7.2.1: GUI des Beispiel-Projektes

Das Beispielprojekt ist eine Adaption eines Projektes → Kapitel 27.3 XMLReader mit gleicher Aufgabenstellung, jedoch unter Verwendung der Klasse XmlExplorer statt XmlReader.

Der Quelltext wird vollständig angegeben:

```
' Gambas class file

Public hXMLExplorer As XmlExplorer
Public sXMLPath As String = "files/list.xml"

Public Sub Form_Open()
  ShowXMLContent()
  txaXML.Pos = 0
  HSplit1.Layout = [3, 2]
End

Public Sub btnShowRecords_Click()

  Dim i, j As Integer
  Dim asMatrix As String[]
  Dim avRecords As New Variant[]
  Dim sSpace As String = String$(4, " ")

  avRecords = GetRecords()

  For i = 0 To avRecords.Max
    asMatrix = New String[]
    asMatrix = avRecords[i]
    txaList.Insert(sSpace & asMatrix[0] & gb.NewLine)
    txaList.Insert(sSpace & asMatrix[1] & " " & asMatrix[2] & gb.NewLine)
    txaList.Insert(sSpace & asMatrix[3] & gb.NewLine)
    txaList.Insert(sSpace & asMatrix[5] & " " & asMatrix[4] & gb.NewLine)
    txaList.Insert(gb.NewLine)
  Next

  File.Save(Application.Path &/ "files/addresslist.txt", txaList.Text)

End

Private Sub ShowXMLContent()
  hXMLExplorer = New XmlExplorer
  hXMLExplorer.Open(sXMLPath)
  txaXML.Text = hXMLExplorer.Document.Content
End

Private Function GetRecords() As Variant[]

  Dim asRecord As String[]
  Dim avRecords As New Variant[]

  txaList.Clear()
  asRecord = New String[]

  hXMLExplorer = New XmlExplorer
  hXMLExplorer.Open(sXMLPath)

  While Not hXMLExplorer.Eof
    Select Case hXMLExplorer.Node.Type
      Case XMLReaderNodeType.Element
        If hXMLExplorer.Node.Name = "mw" Then
           If hXMLExplorer.Node.Value = "w" Then
              asRecord.Add("Frau", 0)
           Else
              asRecord.Add("Herr", 0)
           Endif
           avRecords.Add(asRecord)
           asRecord = New String[]
        Endif
        If hXMLExplorer.Node.Name = "vorname" Then
           asRecord.Add(hXMLExplorer.Node.Value, 1)
        Endif
        If hXMLExplorer.Node.Name = "nachname" Then
           asRecord.Add(hXMLExplorer.Node.Value, 2)
        Endif
        If hXMLExplorer.Node.Name = "strasse" Then
           asRecord.Add(hXMLExplorer.Node.Value, 3)
        Endif
        If hXMLExplorer.Node.Name = "wohnort" Then
           asRecord.Add(hXMLExplorer.Node.Value, 5)
```

```
          For Each hXMLExplorer.Node.Attributes
            Select Case hXMLExplorer.Node.Attributes.Name
              Case "plz"
                asRecord.Add(hXMLExplorer.Node.Attributes.Value, 4)
            End Select
          Next
        Endif
    End Select
    hXMLExplorer.Read()
  Wend

  Return avRecords

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 jeweils in Datensätzen erfasst und in einem Variant-Array gespeichert.
- Dabei werden diese Rohdaten für den Anwendungszweck aufbereitet. So wird je nach Wert für das mw-Element entschieden, ob die Anrede Mann oder Frau gelten soll.
- Die so aufbereiteten Daten werden anschließend → Abbildung 27.7.2.1 in einer TextArea angezeigt.
- Die Adressliste wird in der Datei *addresslist.txt* abgespeichert.
- Ein Besonderheit zeigt sich beim Filtern der Original-Daten nach Wohnort und Postleitzahl, da die Postleitzahl in einem Attribut mit dem Attributnamen 'plz' steckt, während der Ort als Element-Wert (Text-Knoten) ausgelesen wird.