17.3 ListBox

Die Komponente ListBox (gb.qt4) implementiert eine Liste mit auswählbaren Textelementen:

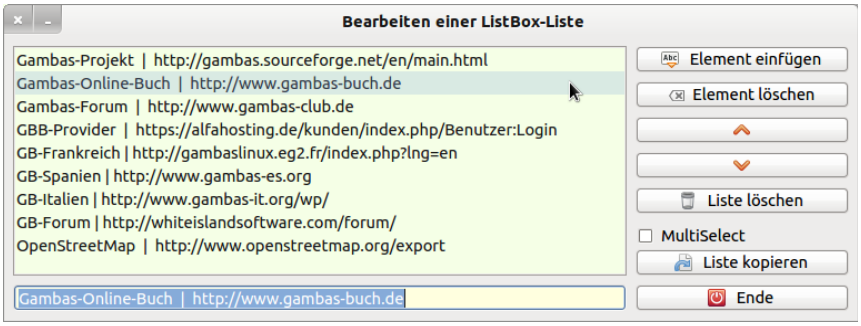

Abbildung 17.3.1: Ein Doppelklick auf das Textelement startet die ausgewählte WebSite

17.3.1 Eigenschaften und Methoden ListBox

Ausgewählte Eigenschaften und Methoden einer ListBox werden jeweils in einer Tabelle aufgeführt und beschrieben:

Eigenschaft	Typ	Beschreibung
Count	Integer	Gibt die Anzahl der Elemente in der Listbox an.
Current	ListBox.Item	Liefert das aktuelle, ausgewählte Element in der Listbox.
Index	Integer	Gibt den Index des aktuell ausgewählten Elements in der Listbox an.
List	Array	Speichert den Inhalt der Listbox in einem String-Array oder füllt die Listbox-Liste mit dem Inhalt eines String-Array.
Mode	Integer	Ermittelt den Selektions-Modus oder legt ihn fest (Konstanten: Select.None (0), Select.Single (1) oder Select.Multiple (2).
Sorted	Boolean	Legt fest, ob die Elemente in der Listbox sortiert angezeigt (!) werden oder nicht. Es erfolgt keine automatische Sortierung der Liste der (Text-)Elemente!
Text	String	Gibt den markierten Text in der Listbox zurück oder setzt den Text.

Tabelle 17.3.1.1: Ausgewählte Eigenschaften der Komponente ListBox

Die Klasse *ListBox.Item* repräsentiert:

- einen Eintrag in der Listbox mit der Eigenschaft Text (Datentyp String) oder
- einen Eintrag in der Listbox mit der Eigenschaft Selected (Datentyp Boolean).

Beispiel 1: Einer TextBox wird der markierte Text in einer ListBox zugewiesen:

```
txtElement.Text = listBoxURL.Current.Text
```

Beispiel 2: Nur selektierte Textelemente aus einer ListBox werden in ein Array kopiert:

```
lsbListe.Mode =  Select.Multiple
…
For iIndex = 0 To lsbListe.Count - 1
  If lsbListe[iIndex].Selected = True Then arrayListe.Add(lsbListe[iIndex].Text)
Next ' iIndex
```

- Die Eigenschaft *.Editable* – wie sie bei einer ListView existiert – fehlt bei einer ListBox, so dass Sie die Liste *nicht direkt in der ListBox* pflegen können. Es werden aber alle Eigenschaften und Methoden bereitgestellt, um die Liste zur Laufzeit zu bearbeiten. Das vorgestellte Projekt zeigt ein erprobtes Vorgehen.
- Sie können einer ListBox ein selbst definiertes Kontext-Menü zuordnen, das Sie über einen Klick mit der rechten Maustaste (RMT) aufrufen.

Eine Beschreibung ausgewählter Methoden einer ListBox finden Sie in der folgenden Tabelle:

Methode	Beschreibung
.Add(Item As String [,Index As Integer])	Fügt ein Element (Item) in die ListBox-Liste ein. Ist der optionale Wert *Index* gesetzt, dann erfolgt das Einfügen an der durch den Index definierten Position, sonst am Ende der Liste.
.Clear()	Löscht den Inhalt der ListBox-Liste.
.Find(Item As String)	Findet ein Element in der ListBox und gibt den entsprechenden Index (Typ Integer) zurück oder eine -1, wenn das Element in der ListBox nicht gefunden wird.
.Popup()	Ordnet einer ListBox ein selbst definiertes Popup-Menü zu.
.Remove(Index As Integer)	Löscht das indizierte Element aus der ListBox-Liste.
.SelectAll()	Selektiert und markiert alle Elemente in der ListBox in Abhängigkeit von der .Mode-Eigenschaft.
.Unselect()	Hebt die Auswahl der markierten Elemente in der ListBox auf.

Tabelle 17.3.1.2: Übersicht zu ausgewählten Methoden der Klasse ListBox

17.3.2 Ereignisse ListBox

Spezielle Ereignisse der Komponente *ListBox* und ergänzende Kommentare finden Sie hier:

Ereignis	Beschreibung
Activate	Wird ausgelöst, wenn der Benutzer auf ein Text-Element in der ListBox doppelt klickt.
Click	Wird ausgelöst, wenn ein Element in der ListBox angeklickt wird und somit ausgewählt wird.
Select	Wird ausgelöst, wenn sich die Selektion ändert.

Tabelle 17.3.2.1: Übersicht zu den 3 Ereignissen der Komponente ListBox

17.3.3 Projekt ListBox

Die im vorgestellten Projekt benutzte ListBox dient dazu, nach einem Klick auf ein Text-Element eine Website anzuzeigen. Das Projekt zeichnet sich durch folgende Merkmale aus:

- Bearbeitung der ListBox-Liste – Element hinzufügen
- Bearbeitung der ListBox-Liste – Element löschen
- Bearbeitung der ListBox-Liste – Element ändern
- Manuelle Sortierung der ListBox-Liste
- Löschen der gesamten ListBox-Liste
- Änderung des Modus der Selektion
- Markierte Text-Elemente der ListBox-Liste in ein Array kopieren
- Import-Funktion für die ListBox-Liste
- Export-Funktion für die ListBox-Liste
- Die ListBox besitzt ein frei definiertes Kontext-Menü mit zwei Einträgen

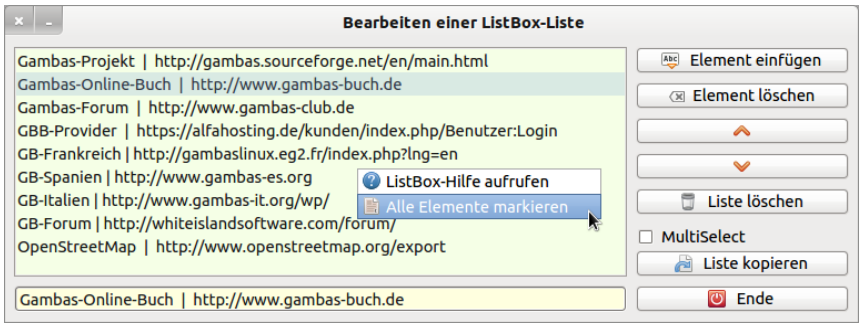

Abbildung 17.3.3.1: ListBox mit Kontext-Menü

Nur wenn Sie die Eigenschaft *ListBox.Mode* auf *Select.Multiple* gesetzt haben, gelingt es mehrere

Text-Elemente bei gedrückter CTRL-Taste mit der Maus zu markieren. Oder Sie nutzen im Kontext-Menü den 2. Eintrag → Abbildung 17.3.3.1, um alle Elemente in der ListBox zu markieren.

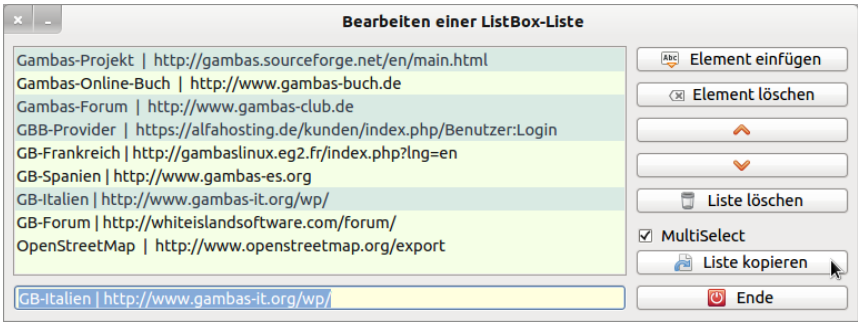

Abbildung 17.3.3.2: ListBox mit mehreren markierten Einträgen

Der Quelltext wird vollständig angegeben. Sie erkennen beim Lesen sicher, dass fast alle Eigenschaften, Methoden und Ereignisse aus den o.a. drei Tabellen benutzt worden sind.

```
[1] ' Gambas class file
[2]
[3] Private $hFile As File
[4] Private $sFilePath As String
[5] Private $sProjectPath As String = Application.Path
[6] Private $iCount As Integer = 1
[7]
[8] Public Sub Form_Open()
[9]
[10]  FMain.Center
[11]  FMain.Resizable = False
[12]
[13]  $sFilePath = $sProjectPath &/ "url.lis"
[14]  If Exist($sFilePath) Then
[15]     ImportLBList($sFilePath, lsbURL)
[16]  Else
[17]     lsbURL.Add("http://www.gambas-buch.de")
[18]  Endif ' Exist($sFilePath) ?
[19]
[20]  lsbURL.Sorted = False ' Notwendig wegen UP und DOWN
[21]  lsbURL[0].Selected = True
[22]  Refresh
[23]
[24]End ' Form_Open()
[25]
[26]Public Sub btnInsertElement_Click()
[27]  lsbURL.Add("Text | Link" & CStr($iCount))
[28]  Inc $iCount
[29]  Refresh
[30]  lsbURL.Index = lsbURL.Count - 1
[31]  Object.Lock(txtElement)
[32]    txtElement.Text = lsbURL.Current.Text
[33]  Object.Unlock(txtElement)
[34]  txtElement.SetFocus
[35]  txtElement.SelectAll
[36]End ' btnInsertSenderURL_Click()
[37]
[38]Public Sub btnDeleteElement_Click()
[39]  Dim iCurrentIndex As Integer
[40]
[41]  iCurrentIndex = lsbURL.Index
[42]  If lsbURL.Index < 0 Then Return
[43]
[44]  If lsbURL.Count > 1 Or lsbURL.Count <= lsbURL.Count - 2 Then
[45]     If Message.Question("Element löschen?", "Löschen", "Abbrechen") <> 1 Then Return
[46]     lsbURL.Remove(lsbURL.Index)
[47]  Else
[48]  If Message.Question("Letztes Element löschen?", "Löschen", "Abbrechen") <> 1 Then Return
[49]     lsbURL.Remove(lsbURL.Index)
[50]  Endif ' lsbURL.Count > 1 ?
[51]
```

```
[52]  If lsbURL.Index >= lsbURL.Count Then Dec lsbURL.Index
[53]  txtElement.Text = lsbURL[lsbURL.Index].Text
[54]  Refresh
[55]
[56]End '  btnDeleteRadioURL_Click()
[57]
[58]Public Sub btnUp_Click()
[59]  Dim iCurrentIndex As Integer
[60]
[61]  iCurrentIndex = lsbURL.Index
[62]  If iCurrentIndex > 0 Then
[63]     Swap lsbURL[iCurrentIndex].Text, lsbURL[iCurrentIndex - 1].Text
[64]     lsbURL.Index = iCurrentIndex - 1
[65]  Endif ' iCurrentIndex > 0 ?
[66]
[67]End ' btnUp_Click()
[68]
[69]Public Sub btnDown_Click()
[70]  Dim iCurrentIndex As Integer
[71]
[72]  iCurrentIndex = lsbURL.Index
[73]  If iCurrentIndex < (lsbURL.Count - 1) Then
[74]     Swap lsbURL[iCurrentIndex].Text, lsbURL[iCurrentIndex + 1].Text
[75]     lsbURL.Index = iCurrentIndex + 1
[76]  Endif ' iCurrentIndex < (lstValue.Count – 1) ?
[77]
[78]End ' btnDown_Click()
[79]
[80]Public Sub btnDeleteList_Click()
[81]  If Message.Question("Komplette Liste löschen?", "Löschen", "Abbrechen") <> 1 Then Return
[82]  lsbURL.Clear
[83]  Refresh
[84]End ' btnDeleteList_Click()
[85]
[86]Public Sub cboxMultiSelect_Click()
[87]  If cboxMultiSelect.Value = cboxMultiSelect.True Then
[88]     lsbURL.Mode = Select.Multiple
[89]  Else
[90]     lsbURL.Mode = Select.Single
[91]     If lsbURL.Count > 0 Then lsbURL[0].Selected = True
[92]  Endif ' cboxMultiSelect.Value = cboxMultiSelect.True ?
[93]End ' cboxMultiSelect_Click()
[94]
[95]Public Sub btnCopyList_Click()
[96]  Dim iIndex As Integer
[97]  Dim sElement As String
[98]  Dim aListe2 As New String[]
[99]
[100]   If cboxMultiSelect.Value = cboxMultiSelect.True Then
[101]      For iIndex = 0 To lsbURL.Count - 1
[102]        If lsbURL[iIndex].Selected = True Then
[103]           aListe2.Add(lsbURL[iIndex].Text)
[104]        Endif ' lsbURL[iIndex].Selected = True ?
[105]      Next ' iIndex
[106]   Else
[107]      Message.Info("MultiSelect nicht markiert?\nKein Element markiert?")
[108]      Return
[109]   Endif ' cboxMultiSelect.Value = cboxMultiSelect.True ?
[110]
[111] '  For Each sElement In aListe2 ' Nur zur Kontrolle ...
[112] '      Print sElement
[113] '  Next ' sElement
[114]
[115] End ' btnCopyList_Click()
[116]
[117] Public Sub lsbURL_Click()
[118]   If Not lsbURL.Current Then Return
[119]   Object.Lock(txtElement)
[120]     txtElement.Text = lsbURL.Current.Text
[121]   Object.Unlock(txtElement)
[122]   txtElement.SetFocus
[123]   txtElement.SelectAll
[124] End ' lsbList_Click()
[125]
```

```
[126] Public Sub lsbURL_DblClick()
[127]   Dim aMatrix As String[]
[128]
[129]   aMatrix = Split(lsbURL.Current.Text, "|")
[130]
[131] If Trim(Lower(aMatrix[1])) Begins "http" Or Trim(Lower(aMatrix[1])) Begins "https" Then
[132]    Try Desktop.Open(Trim(aMatrix[1]))
[133] Else
[134]    Message.Error("In der URL fehlt das Protokoll 'http' oder 'https'!")
[135]    Return
[136] Endif ' Protokoll ok ?
[137]
[138] End ' lsbURL_DblClick()
[139]
[140] Public Sub btnEnde_Click()
[141]   FMain.Close(True)
[142] End ' btnEnde_Click()
[143]
[144] '*****************************************************************************************
[145]
[146] Public Sub txtElement_Change()
[147]   If lsbURL.Current Then
[148]      If lsbURL.List.Exist(txtElement.Text) Then
[149]         Message.Error("Dieser Eintrag existiert bereits.")
[150]      Else
[151]         lsbURL.Current.Text = txtElement.Text
[152]      Endif
[153]   Endif ' lsbCBList.Current ?
[154] End ' txtElement_Change()
[155]
[156] Private Sub Refresh()
[157]   Dim bEnabled As Boolean
[158]
[159]   bEnabled = lsbURL.Count
[160]   txtElement.Enabled = bEnabled
[161]   btnDeleteElement.Enabled = bEnabled
[162]   btnUp.Enabled = bEnabled
[163]   btnDown.Enabled = bEnabled
[164]   btnDeleteList.Enabled = bEnabled
[165]   If Not bEnabled Then txtElement.Clear
[166]
[167] End ' Refresh()
[168]
[169] ' Diese Routine lädt die ListBox-Liste aus einer binären, gambas-spezifischen Datei.
[170] ' Parameter:
[171] ' sPath - Datei-Pfad zur ausgewählten Import-Datei.
[172] ' lsb_URL - Referenz auf die ausgewählte ListBox auf der Form.
[173] Public Sub ImportLBList(sPath As String, lsb_URL As ListBox)
[174]   $hFile = Open sPath For Read
[175]     lsb_URL.List = Read #$hFile As Array
[176]   Close #$hFile
[177] Catch
[178] Message.Error("Der Daten-Import war fehlerhaft!" & gb.NewLine & "Fehler: " & Error.Text)
[179] End ' ImportLBList(...)
[180]
[181] ' Diese Routine speichert eine ListBox-Liste in eine binäre, gambas-spezifische Datei.
[182] ' Parameter:
[183] ' sPath - Datei-Pfad zur ausgewählten Export-Datei.
[184] ' lsb_URL - Referenz auf die ausgewählte ListBox auf der Form.
[185] Public Sub ExportLBList(sPath As String, lsb_URL As ListBox)
[186]   If lsb_URL.Count = 0 Then
[187]      Return
[188]   Else
[189]      $hFile = Open sPath For Write Create
[190]      Write #$hFile, lsb_URL.List As Array
[191]      Close #$hFile
[192] Catch
[193] Message.Error("Der Daten-Export war fehlerhaft!" & gb.NewLine & "Fehler: " & Error.Text)
[194]   Endif ' lsb_URL.Count = 0 ?
[195] End ' ExportLBList(...)
[196]
[197] Public Sub lsbURL_Menu()
[198]   Dim mnuContextLB As Menu
[199]   Dim mnuMenuItem1, mnuMenuItem2 As Menu
```

```
[200]
[201] ' Es wird ein neues Menü-Objekt für die Liste erzeugt:
[202]   mnuContextLB = New Menu(FMain, False)
[203] ' 1. Unter-Menü im Menü mnuContextLB
[204]   mnuMenuItem1 = New Menu(mnuContextLB) As "mnuEditLB"
[205]   mnuMenuItem1.Text = "ListBox-Hilfe aufrufen"
[206]   mnuMenuItem1.Picture = Stock["help"]
[207] ' 2. Unter-Menü im Menü mnuContextLB
[208]   mnuMenuItem2 = New Menu(mnuContextLB) As "mnuSelectAll"
[209]   mnuMenuItem2.Text = "Alle Elemente markieren"
[210]   mnuMenuItem2.Picture = Stock["select-all"]
[211]   mnuContextLB.Popup ' Das mnuContextLB wird der Liste als PopUp-Menü zugewiesen
[212]
[213] End ' lsbURL_Menu()
[214]
[215] ' 2 Aktionen, wenn das Kontextmenü der ListBox ausgewählt wurde:
[216] Public Sub mnuEditLB_Click()
[217]   Message.Info("Ich bin die kleine ListBox-Hilfe ...")
[218] End ' mnuEditLB_Click()
[219]
[220] Public Sub mnuSelectAll_Click()
[221]   If cboxMultiSelect.Value = cboxMultiSelect.True Then
[222]      lsbURL.SelectAll
[223]      lsbURL.Refresh
[224]      Wait
[225]   Else
[226]      Return
[227]   Endif ' cboxMultiSelect.Value = cboxMultiSelect.True ?
[228] End ' mnuSelectAll()
[229]
[230] Public Sub Form_Close()
[231]   $sFilePath = $sProjectPath &/ "url.lis"
[232]   ExportLBList($sFilePath, lsbURL)
[233] End ' Form_Close()
```

Im Kapitel 17.4 ListView wird Ihnen das Projekt *WebRadio* vorgestellt, bei dem Sie ein Text-Element nicht mehr durch ein Trennzeichen – im o.a. Projekt war es das Pipe-Zeichen – in 2 Teile trennen müssen, sondern die Link-Adresse als Key nutzen und nur den Namen des Web-Radio-Senders in der ListView sehen. Der Aufruf des Web-Radio-Senders erfolgt nach der Auswahl des angezeigten Namens des Web-Radio-Senders in der ListView über den korrespondierenden Key. Nur 2 informatische Fliegen mit einer Klappe – aber immerhin … .