17.8 TableView

Gut zu wissen, dass eine TableView (gb.form) eine GridView mit Editier-Funktion ist. Im objekt-orientierten Sinn erbt eine TableView von einer GridView alle Konstanten, Eigenschaften, Methoden und Ereignisse. Daher werden in diesem Kapitel nur *die* Eigenschaften, Methoden und Ereignisse beschrieben, um die eine TableView gegenüber einer GridView erweitert worden ist und darauf ausgerichtet sind, eine Zelle im Gitter zu bearbeiten.

17.8.1 Beispiel

Die von Gambas verwendete (spezielle) MarkDown-Syntax für das Einfügen von Tabellen ist zwar sehr einfach, doch verliert man wegen der vielen notwendigen Zeilen mit den Zeilen- und Spalten-Trennzeichen sehr schnell die Übersicht.

Eleganter wäre es, wenn man die Inhalte der einzelnen Zellen direkt in ein TableView-Gitter einfügt und anschließend den Tabellen-Inhalt in MarkDown-Syntax konvertiert:

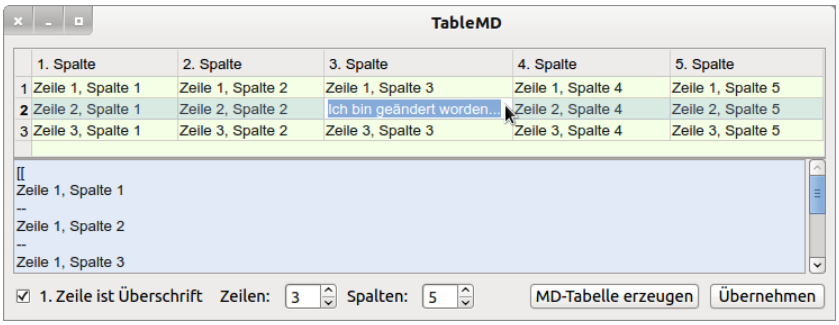

Abbildung 17.8.1.1: Tabellen in MarkDown-Syntax generieren

17.8.2 Eigenschaften

Die Klasse *TableView* verfügt über diese speziellen Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Editor	Control	Gibt die für die Bearbeitung der Zelle verwendete Komponente zurück. Wenn aktuell keine Zelle im Gitter bearbeitet wird, so wird NULL zurückgegeben.
NoKeyboard	Boolean	Ermittelt oder legt fest, ob man mit den Pfeil-Tasten durch die Tabelle navigieren kann, um die *aktuelle Zelle* zu markieren. Ist der Wert auf 'False' festgelegt, dann kann man eine Zelle im Gitter nur noch mit der Maus zum Bearbeiten auswählen. Die Auswahl einer Zeile ist immer möglich, so lange die Eigenschaft TableView.Mode nicht auf Select.None festgelegt ist.

Tabelle 17.8.2.1 : Spezielle Eigenschaften der Klasse TableView

17.8.3 Methoden

Die Klasse *TableView* besitzt u.a. diese speziellen Methoden:

Methode	Beschreibung
Cancel()	Verlässt den Editier-Modus sofort.
Edit ([List As String[], ReadOnly As Boolean])	Startet die Bearbeitung der aktuellen Zelle.
EditWith (Editor As Control)	Festlegung eines alternativen Editors zum Bearbeiten der aktuellen Gitter-Zelle.

Tabelle 17.8.3.1 : Spezielle Methoden der Klasse TableView

Hinweise:

- Beim Einsatz der Cancel-Methode werden alle Änderungen in der aktuellen Gitter-Zelle verworfen und der ursprüngliche Eintrag restauriert.

- Die Edit-Methode muss im Click-Ereignis aufgerufen werden. Wenn 'List' definiert ist, dann wird die aktuelle Zelle mit einer ComboBox versehen, aus der ein Eintrag aus dem List-Argument ausgewählt werden kann. Ist 'ReadOnly' definiert und TRUE, so können die Einträge in der ComboBox nur ausgelesen werden - jedoch nicht geändert werden. Ohne die optionalen Parameter wird eine Zelle mit dem Standard-Editor (TextBox) bearbeitet.
- Den Standard-Editor können Sie mit der EditWith-Methode ändern. Der alternative Editor muss jedoch im gleichen Formular eingefügt werden, in dem auch die TableView existiert.

17.8.4 Ereignisse

Die folgenden zwei speziellen Ereignisse werden benötigt, um editierten Text in einer TableView-Zelle im Gitter abzuspeichern:

- Insert
  Dieses Ereignis wird ausgelöst, wenn der Benutzer durch das Drücken der RETURN-Taste das Einfügen fordert.
- Save(Row As Integer, Column As Integer, Value As String)
  Dieses Ereignis wird ausgelöst, wenn ein editierter Text (String) in eine bestimmte Zelle(Row, Column) gespeichert werden soll.

  → 'Row' ist die aktuelle TableView-Zeile.
  → 'Column' ist die aktuelle TableView-Spalte.
  → 'Value' ist die Zeichenkette, die gespeichert werden soll.

17.8.5 Projekt 1

Das Projekt 1 zeigt Ihnen die Verwendung der speziellen Eigenschaften, Methoden und Ereignisse der *TableView*, um Text in einer ausgewählten TableView-Zelle mit einem bestimmten Editor zu editieren und abzuspeichern. Der Editier-Modus wird bei jedem Klick auf eine TableView-Zelle eingeschaltet. Es wird der vollständige Quelltext angegeben und kommentiert.

Nur in der IDE sehen Sie den eingefügten alternativen Editor 'Editor1', der jedoch im Projekt *nicht* eingesetzt wird:

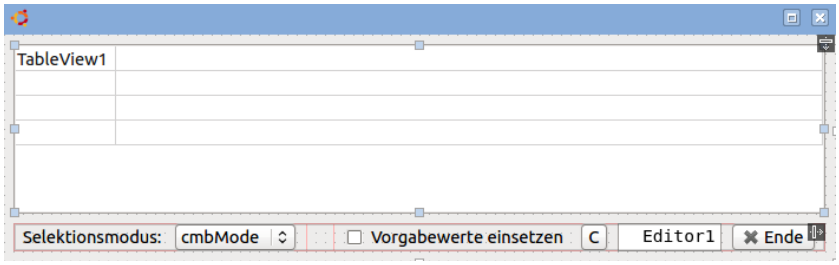

Abbildung 17.8.5.1: Formular

Vollständiger Quelltext:

```
[1]  ' Gambas class file
[2]
[3]  Public aMaxArray As New Integer[][]
[4]  Public aArray As Integer[]
[5]
[6]  Public Sub Form_Open()
[7]
[8]    FTable.Center
[9]    FTable.Arrangement = Arrange.Vertical
[10]   FTable.Margin = True
[11]   FTable.Spacing = True
[12]   FTable.Padding = 8
[13]   HBox1.Spacing = True
[14]   HBox1.H = 24
[15]   panSpace.Expand = True
[16]   cmbMode.Index = 1
[17]
[18]   TableView1.Expand = True
[19]   TableView1.Mode = cmbMode.Index
[20]   TableView1.Header = TableView1.Both ' Anzeige der beiden Header – wenn das erforderlich ist
[21]   TableView1.Resizable = True ' Die Spaltenbreite kann mit der Maus geändert werden
[22]   TableView1.AutoResize = True ' Die letzte Spalte verfügt über vorhandene (Rest-)Breite
```

```
[23]   TableView1.Background = &HF5FFE6
[24]
[25]   TableView1.Rows.Count = 5
[26]   TableView1.Columns.Count = 3
[27]
[28] ' Feste Werte für die Weite, die Überschriften und die Ausrichtung der drei Spalten
[29]   TableView1.Columns[0].Width = 200
[30]   TableView1.Columns[0].Title = "Betriebssytem"
[31]   TableView1.Columns[0].Alignment = Align.Center
[32]   TableView1.Columns[1].Width = 200
[33]   TableView1.Columns[1].Title = "Version"
[34]   TableView1.Columns[1].Alignment = Align.Center
[35]   TableView1.Columns[2].Title = "Computer"
[36]   TableView1.Columns[2].Alignment = Align.Center
[37]   TableView1.Rows[1].Selected = True ' Selektion der 2. Zeile
[38]   TableView1.NoKeyboard = True ' Pfeiltasten OHNE Wirkung zum Navigieren in Gitter-Spalten
[39]
[40] ' TableView1.EditWith(Editor1)
[41]
[42]   cbxEditModus.Value = False
[43]   FillArray(TableView1)
[44]
[45] End ' Form_Open()
[46]
[47] ' Nur für Testzwecke
[48] Public Sub TableView1_Data(Row As Integer, Column As Integer)
[49]   TableView1.Data.Text = "Zeile " & (Row + 1) & ", Spalte " & (Column + 1)
[50] End
[51]
[52] Public Sub TableView1_Click()
[53]    If cbxEditModus.Value = False Then
[54]       TableView1.Edit()
[55]    Else
[56]       Select Case TableView1.Column
[57]         Case 0
[58]           TableView1.Edit(["Linux", "Windows", "OS-X"])
[59]         Case 1
[60]           TableView1.Edit(["Ubuntu 12.04", "Mint 17", "Win 7", "Win 8.1", "X.10"], False)
[61]         Case 2
[62]           TableView1.Edit(["Desktop-PC", "NoteBook", "NetBook", "Tablet PC"], True)
[63]       End Select
[64]    Endif ' cbxEditModus.Value = False ?
[65] End ' TableView1_Click()
[66]
[67] Public Sub TableView1_Save(Row As Integer, Column As Integer, ValueEdit As String)
[68]   TableView1[Row, Column].Text = ValueEdit
[69]   If aMaxArray[Row][Column] < TableView1.Font.TextWidth(ValueEdit) Then
[70]      aMaxArray[Row][Column] = TableView1.Font.TextWidth(ValueEdit)
[71]      TableView1.Columns[Column].Width = TableView1.Font.TextWidth(ValueEdit) + 8
[72]   Else
[73]      TableView1.Columns[Column].Width = aMaxArray[Row][Column]
[74]   Endif
[75]
[76] End ' TableView1_Save(..)
[77]
[78] Public Sub TableView1_ColumnResize(Column As Integer)
[79]   FillArray(TableView1)
[80] End ' TableView1_ColumnResize(..)
[81]
[82] Private Sub FillArray(oTableView As TableView)
[83]   Dim r, c As Integer
[84]
[85]   aMaxArray.Resize(oTableView.Rows.Count)
[86]   For r = 0 To aMaxArray.Max
[87]     aMaxArray[r] = New Integer[]
[88]     aMaxArray[r].Resize(oTableView.Columns.Count)
[89]     For c = 0 To aMaxArray[r].Max
[90]       aMaxArray[r][c] = oTableView.Columns[c].Width
[91]     Next
[92]   Next
[93] End ' FillArray(..)
[94]
[95] Public Sub cmbMode_Click()
[96]   TableView1.Mode = cmbMode.Index
[97]   If cmbMode.Index <> 0 Then TableView1.Rows[1].Selected = True
[98] End ' cmbMode_Change()
[99]
[100]  Public Sub btnCancel_Click()
[101]    TableView1.Cancel()
[102]  End ' btnCancel_Click()
[103]
[104]  Public Sub btnExit_Click()
[105]    FTable.Close
[106]  End ' btnExit_Click()
```

Kommentar:

- In den Zeilen 8 bis 42 werden Initialisierungen einzelner Komponenten vorgenommen.
- In der Prozedur *TableView1_Click()* in den Zeilen 52 bis 65 wird – in Abhängigkeit vom Wert der CheckBox 'cbxEditModus' – der Edit-Modus zugeschaltet. Entweder wird die Edit-Methode ohne oder mit den 2 optionalen Parametern eingesetzt. Für jede Spalte getrennt werden im zweiten Fall die 3 (internen) ComboBoxen über ein Inline-Array (statisch) gefüllt. Beachten Sie, dass Sie für die Werte in der 3. Spalte ausschließlich die Werte aus der ComboBox verwenden können, da der zweite optionale Parameter den Wert *True* hat (Standard-Wert ist False). Wie Sie ComboBox-Listen dynamisch zur *Laufzeit* bearbeiten erfahren Sie im Kapitel 16.13.7 im Beispiel 4.

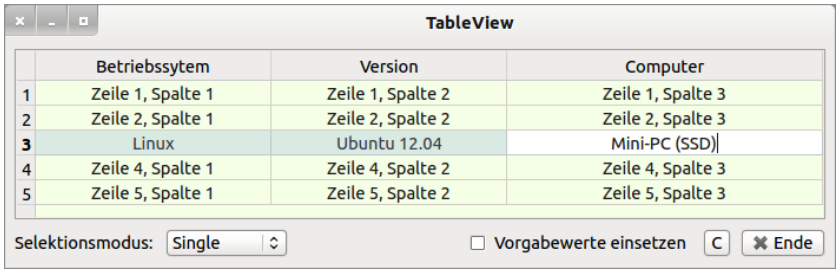

Abbildung 17.8.5.2: Edit-Methode ohne optionale Parameter

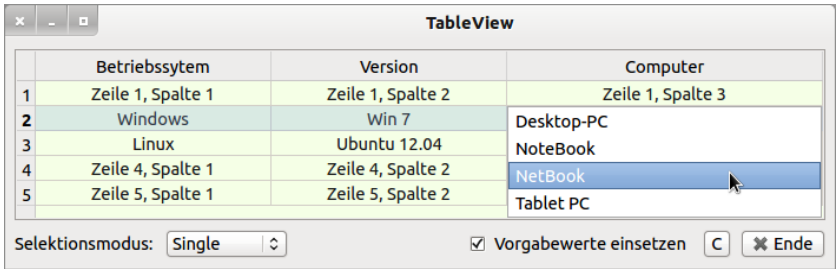

Abbildung 17.8.5.3: Edit-Methode mit optionalem Parameter 'List'

- Das Array *aMaxArray* dient dazu, die Spaltenbreite der bearbeiteten Zelle nach dem Speichern der Länge des Inhalts anzupassen (Zeilen 3, 46, 81-83, 85-98). Es wird stets die aktuelle *maximale Länge* der bearbeiteten Zelle gespeichert.
- Über die ComboBox 'cmbMode' haben Sie die Möglichkeit, die Selektion von einer Zeile, mehreren Zeilen (+CTRL) oder keiner Zeile zur Laufzeit festzulegen (Zeilen 100 bis 103).
- Mit einen Klick auf den *Cancel-Button* mit der Beschriftung 'C' können Sie die Bearbeitung einer Zelle sofort abbrechen – ohne den Inhalt zu verändern (Zeilen 105 bis 107).

17.8.6 Projekt 2

Das Projekt 2 demonstriert, wie Sie Daten in einer TableView mit einem Klick auf den Tabellenspalten-Kopf spaltenweise auf- oder absteigend sortieren können:

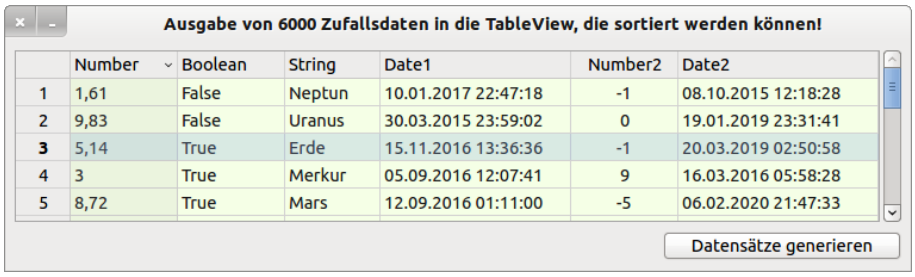

Abbildung 17.8.6.1: Sortieren von Daten in einer TableView