22.5.7 Klasse DataView (gb.db.form)

Die Klasse DataView präsentiert eine DB-Gridview, die den Inhalt eines DB-Results anzeigt. Mit den Cursortasten oder mit der Maus können Sie durch die Datensätze navigieren. Alternativ können Sie eine Navigationsleiste in die Form einfügen, deren Funktionalität Sie jedoch selbst programmieren müssen. Über die Eigenschaft 'Columns' (Datentyp String-Array) können Sie festlegen, welche DB-Felder angezeigt werden. Wenn Sie mit Header = True eine Kopfzeile einfügen, so legen Sie über die Eigenschaft 'Labels' (Datentyp String-Array) die Bezeichner des einzelnen Spalten fest. Die DB-Daten der verwendeten DB-Tabelle können direkt als DB-Result als auch über das Steuerelement 'Datasource' bereitgestellt werden (Standard).

22.5.7.1 Eigenschaften

Die Klasse *DataView* verfügt über die folgenden Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Columns	String[ ]	Legt die Felder fest, deren Inhalt in der DataView in den Spalten angezeigt werden. Sie können den Wert der Eigenschaft auch auslesen. Ist die Eigenschaft nicht gesetzt, dann fehlt diese Beschränkung und es werden automatisch *alle* Felder angezeigt.
Count	Integer	Liefert die aktuelle Anzahl der Datensätze.
Current	Variant[ ]	Gibt die Werte aller Primärschlüssel des aktuellen Datensatzes als Array vom Typ Variant zurück.
Data	_GridView_Data	Verwenden Sie diese Eigenschaft, um den Inhalt einer Zelle während des Data-Ereignisses anzuzeigen (Event Data (Row As Integer, Column As Integer, Value As Variant)).
Editable	Boolean	Ist der Wert True, können die Inhalte *aller* Zellen in der TableView editiert werden. Sie können den Wert der Eigenschaft auch auslesen.
Grid	Boolean	Ist der Wert True, wird ein Gitter angezeigt. Sie können den Wert der Eigenschaft auch auslesen.
Header	Boolean	Ist der Wert True, wird eine Kopfzeile im Gitter angezeigt. Sie können den Wert der Eigenschaft auch auslesen.
Index	Integer	Liefert den Index des aktuellen Datensatzes.
Labels	String[ ]	Ersetzt die Feld-Namen in der Kopfzeile im Gitter durch den Inhalt des String-Arrays in der vorgegebenen Reihenfolge. Sie können den Wert der Eigenschaft auch auslesen.
Sorted	Boolean	Zeigt an, ob durch Anklicken der Spaltenüberschriften im Header die Feldwerte sortiert werden können oder setzt die Eigenschaft.
View	TableView	Gibt die für die Anzeige/Bearbeitung der Daten verwendete Tableview zurück.

Tabelle 22.5.7.1.1 : Eigenschaften der Klasse DataView

Eigenschaften, Methoden und Ereignisse einer TableView finden Sie im Kapitel '17.8 TableView'.

Hier einige Beispiele für den Einsatz der View-Eigenschaft bei einer DataView:

```
DataView1.View.Columns[0].Alignment = Align.Center
DataView1.View.Columns[0].Width = 36
DataView1.View.Rows[DataSource1.Index].Selected = True
```

Am interessantesten sind die beiden Methoden View.Edit ([List As String[], ReadOnly As Boolean]) und View.EditWith(Editor As Control). Diese beiden Methoden müssen im Click-Ereignis oder im Activate-Ereignis aufgerufen werden. Wenn der *optionale* Parameter 'List' definiert ist, dann wird die aktuelle Zelle mit einer ComboBox versehen, aus der ein Eintrag aus der angegebenen Liste 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-(Zellen-)Editor können Sie mit der EditWith-Methode ändern. Der alternative Editor muss jedoch als Steuerelement in das Formular eingefügt werden, in dem auch die DataView existiert.

Beispiel für die Methode ViewEdit([List As String[ ], ReadOnly As Boolean]):

```
Public Sub DataView1_Activate()

'-- The event is triggered when the current record changes.

    If DataSource1.Index > -1 Then
       DataSource1.MoveTo(DataView1.Index)
    Else
       Return
    Endif

    If ComboboxEditMode.Value = False Then
       DataView1.View.Edit()
    Else
       Select Case DataView1.View.Column
         Case 0
           DataView1.View.Edit(["Linux", "Windows", "OS-X"], True)
         Case 1
           DataView1.View.Edit(["Ubuntu 12.04", "Mint 17", "X.10"], False)
         Case 2
           DataView1.View.Edit(["Desktop-PC", "NoteBook", "Tablet-PC"], True)
       End Select
    Endif
End
```

22.5.7.2 Methoden

Die Klasse *DataView* verfügt über diese Methoden:

Methode	Rückgabetyp	Beschreibung
Cancel	-	Bricht die Bearbeitung des aktuellen Datensatzes ab.
Create	-	Wechselt in den Erzeugungsmodus. Am Ende der Tabellenansicht wird eine neue Zeile angezeigt, die den neu erzeugten leere Datensatz darstellt.
Find(Where As String, ...)	-	Verschiebt die aktuelle Zeile zum nächsten Datensatz, der dem angegebenen Filter entspricht. Where ist eine SQL WHERE-Klausel, der der nächste Datensatz entsprechen muss. Zusätzliche Argumente werden innerhalb der Where-Zeichenkette ersetzt, wie bei der DB.Subst-Methode. Die Suche beginnt mit der Zeile, die auf die aktuelle Zeile folgt oder mit der ersten Zeile, wenn es keine aktuelle Zeile gibt.
MoveFirst()	-	Verschiebt den aktuellen Datensatz Datensatz-Zeiger zum ersten Datensatz, der markiert wird.
MoveLast()	-	Verschiebt den aktuellen Datensatz-Zeiger zum letzten Datensatz.
MovePrevious()	-	Verschiebt den aktuellen Datensatz-Zeiger zum vorherigen Datensatz.
MoveNext()	-	Verschiebt den aktuellen Datensatz-Zeiger zum nächsten Datensatz.
MoveTo(Index As Integer)	-	Verschiebt den aktuellen Datensatz-Zeiger zum Datensatz mit dem angegebenen Index, der dann markiert wird.
Remove()	Boolean	Löscht die ausgewählten Datensätze und gibt den Funktionswert TRUE zurück, wenn dies nicht möglich war.
Save()	Boolean	Speichert den aktuellen Datensatz und gibt den Funktionswert TRUE zurück, wenn dies nicht möglich war.

Tabelle 22.5.7.2.1 : Methoden der Klasse DataView

22.5.7.3 Ereignisse

Für die Klasse *DataView* sind zwei Ereignisse von Bedeutung:

(1) Das Ereignis *Activate* wird ausgelöst, wenn sich der aktuelle Datensatz geändert hat.

(2) Das Ereignis *Data (Row As Integer, Column As Integer, Value As Variant) wird ausgelöst*, wenn die DataView eine bestimmte Zelle anzeigen muss. Verwenden Sie die *Eigenschaft* Data, um den Inhalt einer Zelle zu definieren:

- Row ist der Zeilenindex,
- Column ist der Spaltenindex und
- Value ist der Wert, der von der Datenbank-Tabelle erhalten wurde und in der Zelle angezeigt wird.

Beispiele:

```
Public Sub DataView1_Data(Row As Integer, Column As Integer, Value As Variant)

    If IsNull(Value) Then Return

'-- For each line …
'-- ID
    If Column = 0 Then
       DataView1.Data.Font.Bold = True
    Endif

'-- Name
    If Column = 1 Then
       DataView1.Data.Text = Value
    Endif

'-- Date
    If Column = 2 Then
       If TypeOf(Value) = gb.Date Then
          DataView1.Data.Text = Format(Value, gb.MediumDate)
       Endif
    Endif

End
```

22.5.7.4 Beispiel

In der folgenden Datenbank-Anwendung werden die DB-Daten in einer DataView und ausgewählte DB-Daten (Name, Datum und Bild) in drei DB-Steuerelementen angezeigt. Der Bild-Inhalt des BLOB-Feldes im ausgewählten Datensatzes wird in einer Picture-Box gezeigt.

In der Abbildung sehen Sie unter der DataView rechts die vier Buttons zur Navigation in der DataView und links die Buttons, um einen neuen, leeren Datensatz zu erzeugen, um die Anzeige zu aktualisieren, um neue oder geänderte Daten zu speichern oder um einen Datensatz komplett zu löschen:

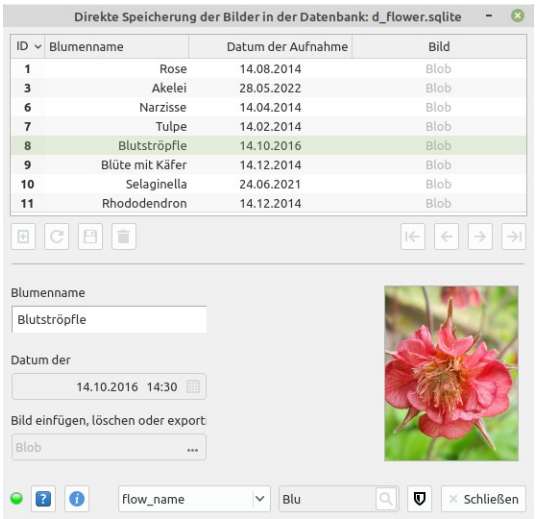

Abbildung 22.5.7.4.1: DB-Anwendung mit DataView und drei DB-Steuerelementen

Im unteren Teil ist neben einer Suche nach den Inhalten eines bestimmten Feldes mit einem LIKE-Filter auch die Erzeugung einer Datenbank-Sicherung (DB-Dump) implementiert.

Die komplette Beschreibung des DB-Projektes finden Sie im Kapitel '22.13 Blob-Projekte'.

Hinweise

- Die Felder können mit einem Klick auf den (Anzeige-)Feldnamen sortiert werden. Dabei gilt: Erster Klick: absteigende Sortierung (▾), zweiter Klick: aufsteigende Sortierung (▴) und dritter Klick: ohne Sortierung – es wird kein Symbol angezeigt.
- Die Anzeige aller Datensätze – die von einem gesetzten Daten-Filter nicht ausgefiltert werden – übernimmt das Steuer-Element DataView.

Das Container-Steuerelement 'DataSource' stellt den Steuerelementen alle Daten der ausgewählten Datenbank zur Verfügung, zu der verbunden wurde. Welche DB-Daten tatsächlich angezeigt werden sollen, können Sie über die Eigenschaften *Table* und *Filter* auswählen:

```
    DataSource1.Connection = DBCS.DBConnection
    DataSource1.Table = "contacts"

'-- Filter (Where-Klausel; optional):
'   DataSource1.Filter = "id > 4 AND city <> 'Berlin' AND city NOT LIKE 'G%'"
```

22.5.7.5 Quelltext-Auszüge

Die folgenden Quelltext-Auszüge zeigen den Einsatz der Methoden MoveFirst(), MoveNext(), Move-Last() und MovePrevious() in der Navigationsleiste:

```
Public Sub ButtonGoToFirst_Click()

'-- At least one data set exists ...
    If DataSource1.Index > -1 Then
       DataSource1.MoveFirst()
  '--  Mark current line
       DataView1.View.Rows[0].Selected = True
       DataView1.SetFocus()
    Endif

End

Public Sub ButtonGoToLast_Click()

    If DataSource1.Index > -1 Then
       DataSource1.MoveLast()
       DataView1.View.Rows[DataSource1.Count - 1].Selected = True
       DataView1.SetFocus()
    Endif

End

Public Sub ButtonGoToPrevious_Click()

    If DataSource1.Index > -1 Then
       DataSource1.MovePrevious()
       DataView1.View.Rows[DataSource1.Index].Selected = True
       DataView1.SetFocus()
    Endif

End

Public Sub ButtonGoToNext_Click()

    If DataSource1.Index > -1 Then
       DataSource1.MoveNext()
       DataView1.View.Rows[DataSource1.Index].Selected = True
       DataView1.SetFocus()
    Endif

End
```