16.14.0 TextArea

Die Klasse TextArea (gb.qt4) implementiert ein mehrzeiliges Textfeld.

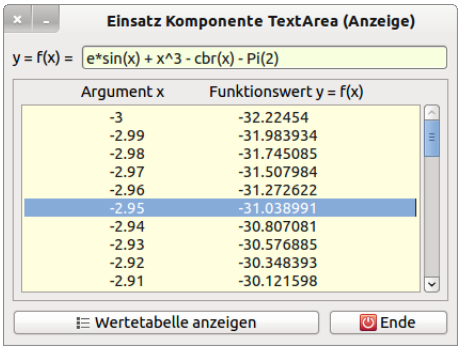

Abbildung 16.14.0.1: Wertetabelle in einer TextArea

16.14.0.1 Eigenschaften

Ausgewählte Eigenschaften der Klasse *TextArea* werden in der folgenden Tabelle beschrieben:

Eigenschaft	Typ	Beschreibung
Alignment	Integer	Gibt die Text-Ausrichtung zurück oder setzt diesen Wert. Folgende Konstanten sind einsetzbar: Align.Normal (0), Align.Left (1), Align.Right (2) oder Align.Center (3).
Background	Integer	Setzt die Hintergrund-Farbe oder liest den Farbwert aus.
Foreground	Integer	Setzt die Text-Farbe oder liest die aktuelle Text-Farbe aus.
Font	Font	Setzt die Text-Schrift oder liest die aktuelle Text-Schrift aus.
Column	Integer	Gibt die Text-Spalte in der aktuellen Text-Zeile zurück oder setzt die definierte Text-Spalte.
Line	Integer	Gibt die aktuelle Text-Zeile zurück oder setzt die definierte Text-Zeile.
Pos	Integer	Gibt die aktuelle Cursor-Position zurück oder setzt den Cursor an die definierte Position im Text. Wenn der Text in UTF-8 gespeichert wurde, so müssen Sie die String-Klasse verwenden, um die Position in den Byte-Index im String zu konvertieren.
Drop	Boolean	Ermittelt oder legt fest, ob die TextArea das Einfügen von Text per Drag & Drop akzeptiert.
Length	Integer	Gibt die Länge des Textes als Anzahl der Zeichen im Text zurück.
ReadOnly	Boolean	Ermittelt oder legt fest, ob der Benutzer den Text ändern kann.
Selected	Boolean	Gibt True zurück, wenn Text markiert ist.
Text	String	Gibt den Text zurück oder setzt den Text, der in der TextArea angezeigt wird.
Wrap	Boolean	Ermittelt oder legt fest, ob der Text – in Abhängigkeit von der Breite der TextArea – umgebrochen wird. Es erfolgt keine Silbentrennung!

Tabelle 16.14.0.1.1: Übersicht zu ausgewählten Eigenschaften der Klasse TextArea (gb.qt4)

Hinweise:

- Die Eigenschaften Pos, Column und Line erschließen sich Ihnen besser, wenn Sie sich den Text in einer TextArea als eine Zeichenkette vorstellen, in die mit "\n" oder *gb.NewLine* Zeilenende-Zeichen (EOL) eingefügt wurden.
- Die Eigenschaft *Column* ist eine relative Größe und bezieht sich immer auf die aktuelle Textzeile! Fassen Sie die Column-Eigenschaft als Cursor-Position in einer Text-Zeile auf, dann liegen Sie damit richtig.

- Die Zählung von Zeilen, Spalten und der Cursor-Position beginnt jeweils bei 0.

- Für die Auszeichnung des Textes in einer TextArea stehen Ihnen nur die zwei Eigenschaften *Foreground* und *Font* zur Verfügung. Sie gelten stets *global* für die gesamte TextArea!

```
  txaArea.Foreground = Color.RGB(0, 0, 0) ' Schrift-Farbe schwarz
  txaArea.Font = Font["Monospace, 10"]    ' dicktengleiche Schrift
' Zur Kontrolle:
' Print txaArea.Font.ToString()
```

- In der Übersicht zu den ausgewählten Methoden der Klasse TextArea finden Sie drei Methoden zur wechselseitigen Konvertierung von Werten der Eigenschaften Pos, Column und Line.

```
  Print "Zeile = "; txaTest.ToLine(91)
  Print "Spalte = "; txaTest.ToColumn(91)
  Print "Cursor-Position = "; txaTest.ToPos(1, 30)
```

- Die Eigenschaft *TextArea.ReadOnly* = *True* wird dann genutzt, wenn Text nur angezeigt werden soll und manuelle Text-Änderungen im Textfeld nicht zugelassen werden. Beispiele: Hilfe-Texte, Anzeige von Messwerten, Konsolen-Ausgaben o.ä..

16.14.0.2 Methoden

Eine Beschreibung ausgewählter Methoden der Klasse *TextArea* finden Sie hier:

Methode	Beschreibung
Clear	Löscht den gesamten Text.
Copy	Der markierte Text wird in das ClipBoard kopiert.
Cut	Der markierte Text wird ausgeschnitten und in das ClipBoard eingefügt.
Paste	Fügt den (Text-)Inhalt des ClipBoards an der Cursor-Position ein.
Insert (Text As String)	Fügt ab der aktuellen Cursor-Position den spezifizierten Text in die TextArea ein.
Select ([Start As Integer, Length As Integer])	Markiert Text ab Cursor-Position *Start* in der Länge von *Length*. » Start ist die Position des ersten markierten Zeichens im Text. » Länge ist die Länge des zu markierenden Textes. Ist kein Argument angegeben, so wird der gesamte Text markiert.
SelectAll	Markiert den gesamten Text.
Unselect	Setzt die Markierung des Textes zurück.
ToColumn (Pos As Integer) As Integer	Konvertiert eine Cursor-Position in eine Spalten-Nummer in einer Text-Zeile → Methode ToLine(..).
ToLine (Pos As Integer) As Integer	Konvertiert eine Cursor-Position in eine Zeilen-Nummer.
ToPos (Line As Integer, Column As Integer) As Integer	Konvertiert eine Kombination aus Text-Zeilen-Nummer und Text-Spalten-Nummer (in dieser Text-Zeile!) in eine Cursor-Position.
Undo	Macht die letzte Aktion rückgängig.
Redo	Nimmt die letzte rückgängig gemachte Aktion zurück.

Tabelle 16.14.0.2.1 : Ausgewählte Methoden der Klasse TextArea (gb.qt4)

Hinweis:

- Zu den Methoden Copy(), Cut() und Paste() finden Sie interessante Informationen in den Kapiteln 20.4.0 'Clipboard' und 20.4.1 'Projekt 1 – Demonstrationsprogramm Clipboard Text'.

16.14.0.3 Ereignisse

Für die praktische Arbeit mit einer TextArea sind vor allem diese zwei Ereignisse wichtig:

Ereignis	Beschreibung
Change	Wird ausgelöst, wenn der Text geändert wurde.
Cursor	Das Ereignis wird ausgelöst, wenn sich die Cursor-Position geändert hat.

Tabelle 16.14.0.3.1: Übersicht zu ausgewählten Ereignissen der Klasse TextArea (gb.qt4)

Hinweise:

- Beachten Sie, dass eine Änderung des Textes *nicht notwendigerweise* auch eine Änderung der Cursorposition zur Folge hat.
- Das Gambas-Beispiel-Programm *Notepad* (Text-Editor) setzt die o.a. Eigenschaften, Methoden sowie Ereignisse sehr gut um, so dass es sich für Sie in jedem Fall lohnen wird, dieses Beispiel intensiv zu erproben.

16.14.0.4 Kontext-Menü

Das Kontext-Menü einer TextArea bietet zusammen mit den Short-Cuts – auch ohne Menüs und Toolbars – die Gewähr für ein flottes Arbeiten mit Texten:

Abbildung 16.14.0.4.1: Kontext-Menü

16.14.0.5 Einfügen von Text

Das Einfügen von Text in eine TextArea an einer beliebigen Cursor-Position können Sie auf mehrere Arten vornehmen:

(1) Eingeben von Text über die Tastatur
(2) Einfügen von Text aus dem ClipBoard → Kontext-Menü
(3) Einfügen von Text über Drag&Drop aus anderen, geeigneten (Text-)Quellen

(4) Einlesen des Inhalts einer Text-Datei (Text-Import)

(5) Änderung der Text-Eigenschaft *TextArea.Text*
(6) Einsatz der Methode TextArea.Insert("text")

Einfügen von Text über Drag&Drop

Das Einfügen von Text über Drag&Drop wurde erfolgreich bei den Editoren 'gedit', 'bluefish' und beim Internet-Browser 'firefox' getestet. Der markierte Text in der Quelle wird bei gedrückter Maustaste über das Textfeld gezogen. Dann wird die Maustaste losgelassen und der markierte Text befindet sich in der TextArea und kann dort weiter bearbeitet werden.

Text-Import aus einer Text-Datei

Minisini sagte zu diesem Thema in einem Forum, dass man in eine TextArea nur UTF-8-Strings eingeben darf und betont: Wenn ein anderer Zeichensatz funktioniert … , dann ist das Kulanz von QT oder Zufall.

Mit diesem Quelltext-Ausschnitt sind Sie immer auf der richtigen Seite:

```
[1]    ...
[2]    Dim sMimeTypeCharSet, sCharSet, sMime As String
[3]    Dim sFilePath As String
[4]
[5]    Dialog.Title = "Wählen Sie eine Text-Datei aus!"
[6]    Dialog.Filter = ["*.ini", "INI-Dateien", "*.txt", "Text-Dateien", "*", "Alle Dateien"]
[7]    Dialog.Path = "/etc/odbcinst.ini"
[8]    If Dialog.OpenFile() Then Return
[9]    sFilePath = Dialog.Path
[10]
[11]   Exec ["file", "-bi", sFilePath] To sMimeTypeCharSet ' Ermittlung MimeTyp und Zeichensatz
[12]   Wait
[13]   Print sMimeTypeCharSet
[14]   sMime = Split(sMimeTypeCharSet, ";")[0]
[15]   Print "Mime-Typ = "; Trim(sMime)
[16]   sCharSet = Trim(Split(Split(sMimeTypeCharSet, ";")[1], "=")[1])
[17]   Print "CharSet  = "; Trim(sCharSet)
[18]
[19]   If sCharSet <> "utf-8" Then
[20]      Try txaArea.Text = Conv(File.Load(sFilePath), sCharSet, "UTF-8")
[21]    ' Try txaArea.Insert(Conv(File.Load(sFilePath), sCharSet, "UTF-8"))
[22]   Else
[23]      txaArea.Text = File.Load(sFilePath)
[24]    ' txaArea.Insert(File.Load(sFilePath))
[25]   Endif
[26]
[27]   txaArea.Pos = 0
[28]   ...
```

Kommentar:

- Die zu importierende Text-Datei wird in den Zeilen 5-9 in einem Dialog ausgewählt.
- In der Zeile 11 wird der kombinierte *MimeTyp-Zeichensatz-Wert* der importierten Text-Datei ausgelesen und in der Variablen *sMimeTypeCharSet* gespeichert.
- Die Zeile 16 übernimmt die Extraktion des Zeichensatzes, der in der Zeile 20 benötigt wird, um den Zeichensatz zu konvertieren – falls das notwendig sein wird.
- Achtung: Der importierte Text in der Zeile 20 oder 23 ersetzt den u.U. schon vorhandenen Text.
- Der importierte Text wird jedoch standardmäßig an das Text-Ende angehängt, wenn Sie die Zeilen 21 oder 24 verwenden.
- Die blau hervorgehobenen Zeilen dienen nur zur Kontrolle und können auskommentiert werden.

Die Konfigurationsdatei *odbcinst.ini* lieferte folgenden kombinierten Wert aus MimeTyp und Zeichensatz:

```
text/plain; charset=us-ascii
```

Der Zeichensatz der Datei wurde entsprechend der Forderung in der Zeile 19 deshalb in UTF-8 konvertiert. Notwendig war das im Falle von *us-ascii* nicht, weil ASCII in UTF-8 enthalten ist.

Textänderungen über die Text-Eigenschaft und die Insert(..)-Methode

In diesem Abschnitt wird einerseits Text in eine TextArea eingefügt, in dem über eine Änderung der Eigenschaft *TextArea.Text* = Text-String der alte Text komplett durch den neuen Text überschrieben wird und andererseits gezeigt, wie man Text an der aktuellen Text-Cursor-Position hinzufügt:

```
TextArea.Text = "Ich bin der neue Text!"
TextArea.Text = TextArea.Text & "Ich bin weiterer Text!"
TextArea.Text &= "Ich bin weiterer Text!"
TextArea.Text &= gb.NewLine ' erzwingt einen Zeilenwechsel
TextArea.Text &= gb.NewLine & gb.NewLine ' erzeugt eine Leerzeile nach dem Text

TextArea.Insert("Ich bin Text – an der aktuellen Cursor-Position eingefügt!")
```

Text-Export

Wenn Sie Text über das ClipBoard sichern wollen, dann erzeugt das nur eine scheinbare Sicherheit, denn der Inhalt des ClipBoards wird automatisch gelöscht, wenn Sie das Programm mit der TextArea beenden!

Der einfachste Weg, den Inhalt einer TextArea zu sichern, führt über einen Datei-Speichern-Dialog:

```
Public Sub btnTextToFile_Click()
  Dialog.Filter = ["*.txt", "Text-Dateien"]
  If Dialog.SaveFile() Then Return
  File.Save(Dialog.Path, txaArea.Text)
  CATCH
    Message.Info(Error.Text)
END
```

Nützliches

Die folgenden kommentierten Quelltext-Abschnitte sind hilfreich bei der Arbeit mit einer TextArea:

```
Print "Anzahl der Zeichen = "; txaTest.Length
Print "Anzahl der Zeilen = "; Split(txaTest.Text, gb.NewLine).Count
Print "Anzahl der Zeilen = "; txaTest.ToLine(txaTest.Length) + 1
```

```
Public Sub btnSetWrapping_Click()
  txaTest.Wrap = Not txaTest.Wrap ' Alternative Änderung der Wrap-Eigenschaft (Zeilenumbruch)
End
```

```
TextArea.Text = "" ' löscht den gesamten Text
TextArea.Clear ' löscht den gesamten Text
```

```
Public Sub txaTest_Cursor()
  Print "Zeile = "; txaTest.Line
  Print "Spalte = "; txaTest.Column
  Print "Position = "; txaTest.Pos
  Print "Cursor in Zeile "; txaTest.Line; " auf Spalte "; txaTest.Column; " auf Position "; txaTest.Pos
End
```

```
txaTest.Pos = 0 ' Sprung in die erste Zeile (physisch 1) auf Spalte 0
txaTest.Pos = txaTest.Length ' Sprung in die letzte Zeile
```

```
Public Sub txaTest_Cursor()
  lblAktuelleZeile.Text = "Z: " & txaTest.Line     ' Anzeige der aktuellen Zeile
  lblAktuelleSpalte.Text = "S: " & txaTest.Column  ' Anzeige der aktuellen Spalte (in der Zeile)
  lblCursorPosition.Text = "P: " & txaTest.Pos     ' Anzeige der aktuellen Cursor-Position
End
```

Limitierter Text

Ein SMS umfasst im Normalfall maximal 160 Zeichen. Mit diesem Quelltext-Ausschnitt können Sie diese Anzahl der Zeichen garantieren. Auch der Fall, dass sich durch Kopieren und Einfügen zu viel Text in der TextArea befindet wurde berücksichtigt:

```
Public Const MAX_CHAR As Integer = 160

Public Sub txaTest_Change()

' If Len(Last.Text) < MAX_CHAR Then ' Alternative
  If txaTest.Length < MAX_CHAR Then
     txaTest.Foreground = &00007FFF&
     If MAX_CHAR - txaTest.Length = 1 Then
        FMain.Text = Subst$("Es ist noch genau 1 Zeichen verfügbar!")
     Else
        FMain.Text = Subst$("Es sind noch &1 Zeichen verfügbar!", MAX_CHAR - txaTest.Length)
     Endif
  Else If txaTest.Length = MAX_CHAR
     txaTest.Foreground = &00007FFF&
     txaTest.ReadOnly = True
     FMain.Text = Subst$("Die maximale Anzahl von Zeichen " & Str(MAX_CHAR) & " ist erreicht!")
  Else
    txaTest.Foreground = Color.Red
    FMain.Text = Subst$("Es sind &1 Zeichen zu viel eingefügt.", - MAX_CHAR + txaTest.Length)
  Endif

End ' txaTest_Change()

Public Sub txaTest_KeyPress()
  If Key.Code = Key.BackSpace Or Key.Code = Key.Delete Then txaTest.ReadOnly = False
End ' txaTest_KeyPress()
```

Das Thema 'Suchen und ersetzen von Text in einer TextArea' wird im nächsten Kapitel behandelt.