20.1.1 Klasse Message – MessageBox

Meldungen in einem Programm unterstützen Programmentwickler beim Entwurf und Test sowie Nutzer beim erfolgreichen Umgang mit dem Programm. Die Klasse *Message* ist geeignet, Meldungen in einer Message-Box anzuzeigen. Die Klasse ist statisch und kann wie eine Funktion genutzt werden, die über maximal 4 Argumente verfügt. Das erste Argument – der Mitteilungstext – ist notwendig, die weiteren drei sind optional. Die Argumente sind vom Typ Zeichenkette und der Funktionswert ist eine Integer-Zahl aus der Menge {1,2,3}. Die Meldungen können als einfacher Text oder im HTML-Format ausgegeben werden. In der Message-Box wird in Abhängigkeit vom Typ ein aussagekräftiges Icon angezeigt. Die Meldungen kann man auch in die beiden Kategorien „Statischer Text der Meldung“ und „Dynamischer Text der Meldung“ einteilen, wenn man als Kriterium den strukturellen Aufbau des Textes der Meldung heranzieht. Bei einem dynamischen Text der Meldung werden zum Beispiel die aktuellen Werte von Programm-Variablen oder Systemmeldungen in den Text der Meldung eingefügt. Da die Message-Box nur über eine geringe Standardbreite verfügt, müssen Sie den Text der Meldung in geeigneter Weise formatieren, um die beabsichtigte Wirkung zu erzielen. Sie können die Message-Box zur Programmlaufzeit in der Größe verändern.

Gambas verfügt über 5 verschiedene Message-Box-Typen:

- Information-Box (message.info(...) = message(...)) als Standard
- Frage-Box (message.question(...))
- Warnung-Box (message.warning(...))
- Fehler-Box (message.error(...))
- Löschen-Box (message.delete(...))

Das ist die Syntax für die Standard-Message-Box als Information-Message-Box und für die Warnung-Message-Box:

```
FUNCTION Info (Message AS String [, Button AS String ]) AS Integer
FUNCTION Warning (Message AS String [, Button1 AS String,Button2 AS String,Button3 AS String]) AS Integer
```

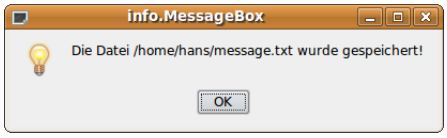

Abbildung 20.1.1.1: Info-Messagebox

Abbildung 20.1.1.2: Warning-Messagebox

Der Auslöser für die Anzeige einer Message-Box ist entweder ein besonderer Zustand des Programms der intern ausgewertet wird oder eine Reaktion des Nutzers (Beispiele: Druck auf einen Button oder eine Mausreaktion oder das Aufrufen eines Menüeintrags). Auf jeden Fall bewirkt das Anzeigen einer Message-Box eine *Unterbrechung des Programms*, weil der Aufruf der Message-Box *modal* erfolgt. Erst wenn der Nutzer auf die Meldung reagiert, wird das Programm fortgesetzt, wobei es maximal drei verschiedene Möglichkeiten dazu gibt.

Die Antwort auf die folgenden Fragen erleichtern Ihnen die Entscheidung für den Einsatz einer Message-Box:

- An welcher Stelle im Programmablauf scheint es *notwendig*, das Programm für die Anzeige einer Meldung in einer Message-Box zu *unterbrechen*?
- Welcher Typ der 5 möglichen Typen der Message-Box soll verwendet werden?
- Welches Format soll der Text der Meldung besitzen?
- Wie viel Button sollen in der Message-Box angezeigt werden und welchen Text erhalten diese Button?
- In welcher Form soll der Funktionswert, der vom Objekt Message-Box zurückgegeben wird, erfasst werden? Wie werden die möglichen Funktionswerte ausgewertet und wie wird auf die Funktionswerte im Programm reagiert?
- Sind für das Programm Übersetzungen in andere Sprachen vorgesehen?

Die folgenden Beispiele werden hinreichend kommentiert, so dass Sie Besonderheiten gut erkennen können und es werden auch alternative Lösungen angeboten.

20.1.1.1 Beispiel 1

Die Meldung informiert den Nutzer hier über eine (erfolgreich) abgeschlossene Programm-Aktion. Er muss nur mit einem Klick auf den OK-Button die Programmunterbrechung aufheben.

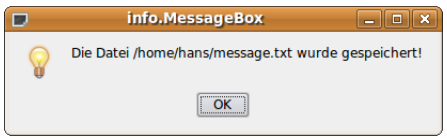

Abbildung 20.1.1.1.1: Info-Messagebox

```
  ...
  Message.Info("Die Datei " & User.Home &/ "message.txt" & " wurde gespeichert")
  ...
```

Alternative:

```
  ...
  Message.Info("Die Datei " & User.Home &/ "message.txt" & " wurde gespeichert", "OK")
  ...
```

20.1.1.2 Beispiel 2

Der Text der Meldung ist vom Typ *Text/Plain* und es werden 2 Button verwendet:

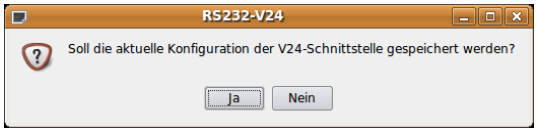

Abbildung 20.1.1.2.1: Info-Messagebox

```
PUBLIC SUB btnAktuelleKonfigurationSpeichern_Click()
  DIM iIndexMBQ AS Integer

  iIndexMBQ = Message.Question("Soll die aktuelle Konfiguration der V24-Schnittstelle \
              gespeichert werden?", "Ja", "Nein")

  IF iIndexMBQ = 1 THEN
     v24Settings["V24Konfiguration/Port-Name"] = cmbRS232PortName.Text
     ...
     v24Settings["V24Konfiguration/Datenflusskontrolle"] = cmbFlow.Text
     v24Settings.Save
  ELSE
     RETURN
  ENDIF ' Message.Question("...?")
END ' btnAktuelleKonfigurationSpeichern_Click()
```

Alternative:

```
PUBLIC SUB btnAktuelleKonfigurationSpeichern_Click()
  IF Message.Question("Soll die aktuelle Konfiguration der V24-Schnittstelle \
     gespeichert werden?", "Ja", "Nein") = 1 THEN
     v24Settings["V24Konfiguration/Port-Name"] = cmbRS232PortName.Text
     ...
     v24Settings["V24Konfiguration/Datenflusskontrolle"] = cmbFlow.Text
     v24Settings.Save
  ELSE
     RETURN
  ENDIF ' Message.Question("...?")
END ' btnAktuelleKonfigurationSpeichern_Click()
```

Mit dem Aufruf der Message-Box wird das Programm unterbrochen. Wenn der Nutzer den „Ja“-Button gedrückt hat, dann wird die aktuelle Konfiguration der V24-Schnittstelle abgespeichert und danach das Programm fortgesetzt. Im alternativen Fall „Nein“ wird das Programm *sofort* weiterarbeiten. Hätten Sie folgenden Text verwendet:

```
IF Message.Question("Soll die aktuelle Konfiguration der V24-Schnittstelle gespeichert werden?", \
   "Na klar...", "Nö – jetzt nicht!") = 1 THEN
```

dann hätte das keinen Einfluss auf den Funktionswert gehabt, der nur durch den gedrückten Button

bestimmt wird, weil deren Reihenfolge in der Message-Box eine Rangfolge impliziert:

- Button1 „Na klar“ → Funktionswert = 1,
- Button2 „Nö – jetzt nicht!“ → Funktionswert = 2,
- *Button3 hier nicht genutzt; ergäbe aber als Funktionswert = 3.*

20.1.1.3 Beispiel 3

Abbildung 20.1.1.3.1: Frage-Messagebox

Im diesem Fall wird als Funktionswert entweder 1 oder 2 oder 3 zurückgegeben. Die Auswertung des Funktionswerts sollten Sie über die Kontrollstruktur *Select … Case* vornehmen:

```
SELECT message.question("Soll die Datei message.txt gesichert werden?", "Ja", "Nein", "Abbruch")
   CASE 1
      File.Save(dateipfad &/ message.txt , txaTextFeld.Text)
   CASE 2
      TRY KILL dateipfad &/ message.txt
      IF ERROR THEN
         Message.Error("Die Datei konnte nicht gelöscht werden!")
         RETURN
      ENDIF
   CASE 3
      RETURN
END SELECT
```

20.1.1.4 Beispiel 4

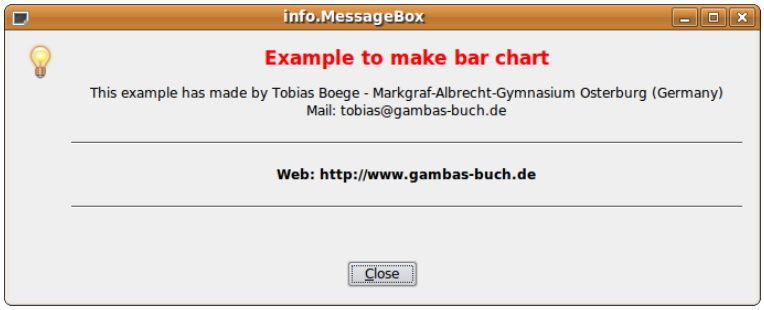

Abbildung 20.1.1.4.1: Info-Messagebox (HTML)

Der Text der Meldung kann in geeigneter Weise formatiert werden, wenn man Text vom Typ text/html wählt:

```
PUBLIC SUB btnInfo_Click()
  DIM text, leerzeile AS String
  DIM iCount AS Integer

  FOR iCount = 1 TO 10 * 15
      leerzeile &= " "
  NEXT
  text = "<center><h2><font color='red'>Example to make bar chart</h2></font></center>"
  text &= "<center><p>This example has made by Tobias Boege - Markgraf-Albrecht-Gymnasium \
           Osterburg (Germany)<br>"
  text &= "Mail: tobias@gambas-buch.de</center></p>"
  text &= "<hr>"
  text &= "<center><b>Web: http://www.gambas-buch.de</b></center>"
  text &= "<hr>"
  text &=  leerzeile

  Message.Info(text, "&Close") ' ---> Schließen mit [ALT]+[C]

END ' btnInfo_Click()
```

Die Besonderheiten bei dieser Meldung sind das Einfügen einer Zeile aus geschützten Leerzeichen,

um die erforderliche *Breite der Box zu erzwingen* und die zwei horizontalen Linien sowie die Auszeichnung von Teilen des Textes mit dem Attribut „fett“ sowie die Verwendung von Farben. So eine Meldung ist informativ und fällt auf.

20.1.1.5 Beispiel 5

Abbildung 20.1.1.5.1: Info-Messagebox (HTML)

Sie können diese Meldung zum Beispiel aus einem Menü heraus aufrufen:

```
...
  DIM leerzeile AS String
  DIM iCount AS Integer

  FOR iCount = 1 TO 10 * 1
      leerzeile &= " "
  NEXT

  Message.Info(File.Load("help.htm") & leerzeile)
...
```

oder über die F1-Taste:

```
PUBLIC SUB Form_KeyPress()
  DIM leerzeile AS String
  DIM iCount AS Integer

  FOR iCount = 1 TO 10 * 1
      leerzeile &= " "
  NEXT

  IF Key.Code = Key.F1 THEN Message.Info(File.Load("help.htm") & leerzeile)

END
```

Speichern Sie folgenden HTML-Quelltext in der Datei *help.html* im Projektverzeichnis ab, dann können Sie den Text in der HTML-Datei nachträglich nicht mehr ändern, weil er mit compiliert wurde. Die Style-Anweisungen werden hier als Inline-CSS in der HTML-Datei verankert. Die Alternative, diese Style-Anweisungen in einer separaten CSS-Datei zu pflegen, lohnt sicher nur für größere Projekte.

```
<html>
  <head>
    <title>Hilfetext</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF8">
    <style type="text/css">
    html {background-color:#FFFFDF; color:#000000; font-family:Verdana,Helvetica,Arial,sans-serif; \
          font-size:12px;}
    H1 {font-size:12px;}
    <!-- Alternative: <link rel="stylesheet" type="text/css" href="help.css" /> -->
    </style>
  </head>
  <body>
    <h1><font color='Blue'>Ich bin die kleine Hilfe ...</font></h1>
    <hr>
    <p><b>F1HilfeW</b> ist eine einfache Anwendung mit einer Programm-Hilfe im HTML-Format:</p>
    <p>
    Die <b><font color='Blue'>F1-Taste</font></b> ruft das Hilfefenster der Anwendung F1HilfeW auf.<br>
    Die <b><font color='Blue'>Escape-Taste (ESC)</font></b> beendet das Hilfefenster.<br>
```

```
    Das <b><font color='Blue'>Hilfefenster</font></b> kann in der Größe verändert werden.<br>
    </p>
    <hr>
    <p>
    Das Programm (Version 0.0.20) wurde von Hans Lehmann 2012 entwickelt.
    <br />
    Weitere Hilfe finden Sie unter www.gambas-buch.de.<br />
    Anfragen bitte nur an hans@gambas-buch.de.
    </p>
  </body>
</html>
```

Eine Nummer kleiner in der Textauszeichnung geht es natürlich auch, wie diese zwei Beispiele zeigen:

Abbildung 20.1.1.5.2: Warning-Messagebox (HTML)

```
...
Message.Warning("Die Datei <b>message.txt</b> ist gesperrt!", "Die Datei entsperren", "Abbruch")
...
```

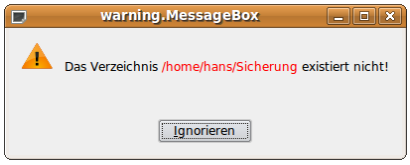

Abbildung 20.1.1.5.3: Warning-Messagebox (HTML)

```
...
  DIM text, s AS String
  DIM iCount AS Integer

  FOR iCount = 1 TO 10 * 8 ' auch mal mit Werten < 8 probieren...
      s &= " "
  NEXT

  text = s & "<br />"
  text &= "Das Verzeichnis <font color='red'>Home.User &/ Sicherung</font> existiert nicht!"
  Message.Warning(text, "&Ignorieren")

...
```

20.1.1.6 Beispiel 6

Meldungen sind besonders informativ, wenn Sie zum Beispiel in einem Programm an einer bestimmten, als zeitkritisch eingeschätzten Stelle, einen Fehler abfangen, wenn er auftrat und Sie in der Fehlermeldung für die Fehleranalyse dann auf angezeigte Fehler-Codes zurückgreifen könnten oder auf Informationen zu dem Programmabschnitt oder Modul, das einen Fehler auslöste. Zusätzlich können Sie eigene Texte den Meldungen hinzufügen. Mit der folgenden Fehlermeldung können Sie als *Programm-Entwickler* sicher etwas anfangen. Vor allem dann, wenn mehrere Entwickler am Projekt beteiligt sind:

Abbildung 20.1.1.6.1: Error-Messagebox mit Fehlercode und Hinweisen

Sie erfahren, das ein Fehler auftrat, bekommen zum Fehlercode 43 den entsprechenden englischen Fehlertext und sehen auch den Namen der Prozedur sowie die Zeilennummer im Programm, bei der ein Fehler auftrat.

Der Quelltext – mit dem entsprechenden Programmabschnitt für Programm-Entwickler – wird vollständig angegeben:

```
PUBLIC SUB btnCreatingFile_Click()
  DIM hFile AS Stream
  DIM sErrorText AS String
  IF NOT Exist(TextBox1.Text) THEN
     hFile = OPEN TextBox1.Text FOR CREATE
    ' Den Benutzer über den Erfolg der Operation informieren
     Message.Info("Die Datei wurde angelegt.")
     CLOSE #hFile
  ELSE
    ' Wenn die Datei existiert, eine Fehlermeldung ausgeben
     Message.Error("Datei existiert bereits.")
  ENDIF
  CATCH
  SELECT CASE Error.Code
    CASE 43 'Access forbidden
    CASE 44 'File name is too long
    CASE 45 'File or Directory does not exist
    CASE 46 'File is a Directory
    CASE 48 'Write Error
    CASE ELSE
  ' NOOP
  END SELECT

  Message.Error("FehlerCode: " & Error.Code & Chr(10) & "FehlerText: " & Error.Text & Chr(10) & \
                "FehlerQuelle: " & Error.Where)

END ' btnCreatingFile_Click()
```

In der getesteten Programmversion werden Sie wohl nur den folgenden Anweisungsblock verwenden:

```
...
SELECT CASE Error.Code
    CASE 43
     sErrorText = "Sie besitzen nicht die Rechte,\num diese Datei anzulegen."
    CASE 44
     sErrorText = "Der Dateiname ist zu lang."
    CASE 45
     sErrorText = "Die Datei existiert nicht."
    CASE 46
     sErrorText = "Der Pfad führt zu einem Ordner."
    CASE 48
     sErrorText = "Fehler beim Schreiben in die Datei."
    CASE ELSE
     sErrorText = "Fehler?"
  END SELECT

  Message.Error(sErrorText)
...
```

Die Fehlermeldung ist für den Programmnutzer gedacht, besitzt einen selbst verfassten deutschen Text in der Message-Box und benennt den Fehler eindeutig:

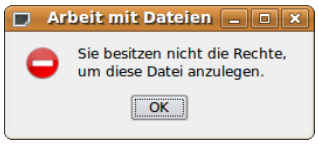

Abbildung 20.1.1.6.2: Error-Messagebox mit eigenem Text