11.10.0 Weitergabe eines Gambas-Programms – Installationspaket

Oft besteht der Wunsch, ein Gambas-Programm weiterzugeben. Zwei Ansätze kommen dafür in Betracht:

- Weitergabe des Gambas-Projekts in einem Quelltext-Archiv. Notwendige Voraussetzung: Auch auf dem Ziel-Rechner ist Gambas in einer relevanten Version *vollständig* installiert.
- Weitergabe des Programms als Installationspaket (Archiv) für ausgewählte Linux-Distributionen. Ein Installationspaket besteht aus mehreren Dateien. Notwendige Voraussetzung: Auf dem Ziel-Rechner sind die Gambas-Pakete über den Paket-Manager in einer Version installierbar, deren Versionsnummer gleich oder größer ist als die auf dem Entwicklungsrechner.

Gambas stellt für beide Varianten in der IDE geeignete Export-Dialoge zur Verfügung. In diesem → Kapitel 11.10 erfahren Sie, wie Sie ein Installationspaket für eine Ziel-Distribution generieren.

11.10.0.1 Basis

Basis der Beschreibung, wie man ein Installationspaket schnürt und auf einem Ziel-Rechner installiert, ist eine *konkrete* Zielstellung mit eng gefassten *Randbedingungen* – umgesetzt im Projekt StructDB. Der Hinweis ist m.E. notwendig, weil es keine allgemeine, Erfolg versprechende Beschreibung für jede Linux-Distribution und für jede (stabile) Gambas-Version geben wird.

Zielstellung:

- Der Entwickler E entwickelt und testet das Projekt StructDB (Programm StructDB als einfache Datenbankanwendung) unter Ubuntu 12.04 LTS (Support bis 2017), setzt ein stabiles Gambas aus den Projekt-Quellen ein und schnürt abschließend das Gambas-Installationspaket für die Ziel-Distribution Ubuntu/Mint.
- Der Benutzer B setzt auf dem Ziel-Rechner Mint 17.1 (Rebecca) ein und es sind alle Gambas-Pakete aus dem (stabilen) Ubuntu-PPA installierbar. Die Gambas-Entwicklungsumgebung (IDE) ist dort nicht installiert, weil sie nicht benötigt wird, um das Gambas-Installationspaket – gespeichert im Ordner ~/IP_StructDB auf dem Zielrechner – von B (mit Root-Rechten) zu installieren und um das Programm StructDB zu verwenden.

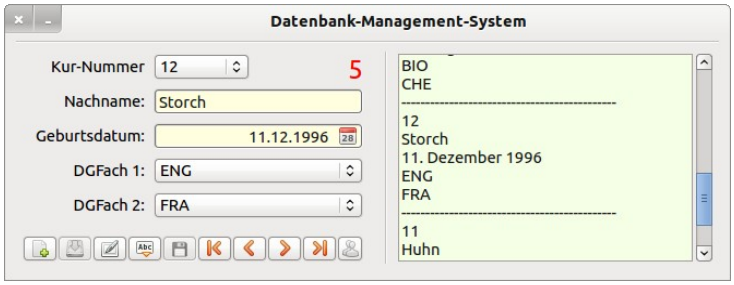

Abbildung 11.10.0.1.1: GUI für das Programm StructDB

11.10.0.2 Vorbetrachtungen

Diese Vorbetrachtungen sind das Resultat vieler Versuche, ein Installationspaket *erfolgreich* zu schnüren und widerspiegeln die Erfahrungen im Umgang mit dem Assistenten für das Generieren eines Installationspakets (→ Packer-Assistent). Als Erfolg gilt, wenn das Installationspaket vom Benutzer B mit Root-Rechten auf dem Ziel-Rechner ohne Probleme installiert werden kann und der Benutzer B das installierte Programm so nutzen kann, wie es die Programm-Spezifikation vorsieht.

- Schon bei der Anlage des Projekts, dessen Programm weitergegeben werden soll, sollten Sie sich vor allem beim *Projekt-Typ* und bei den auszuwählenden *Optionen* sicher sein und den vorgesehenen Typ und die passenden Optionen auswählen. Bei den Programmen mit einer GUI können Sie wählen:

  - Grafische Anwendung
  - QT-GUI-Anwendung

- GTK+ Grafische Anwendung

Wenn Sie 'Grafische Anwendung' wählen, so wird zur Entwicklungszeit die Komponente *gb.gui* geladen und der Packer-Assistent wird sowohl die gb.gtk-Pakete als auch die gb.qt4-Pakete in die Liste der Abhängigkeiten aufnehmen. Das hätte zur Folge, dass Sie dann u.a. 3 verschiedene *.deb-Dateien im Paket finden:

StructDB_1.2.12-0ubuntu1_all.deb
StructDB_gtk_1.2.12-0ubuntu1_all.deb
StructDB_qt4_1.2.12-0ubuntu1_all.deb

und führt zu der Frage: Welches Debian-Paket wird auf dem Ziel-Rechner verwendet?

Sie können das *StructDB_gtk_1.2.12-0ubuntu1_all.deb-Paket* oder das *StructDB_qt4_1.2.12-0ubuntu1_all.deb-Paket* installieren oder beide. Da diese beiden Pakete aber vom Debian-Paket *StructDB_1.2.12-0ubuntu1_all.deb* abhängen, welches das Projekt ohne GUI-Abhängigkeiten ist, wird das *StructDB_1.2.12-0ubuntu1_all.deb-Paket* in jedem Fall auch (mit-)installiert.

- Wählen Sie beim Anlegen eines neuen Projekts bei den Projekt-Informationen gleiche Bezeichnungen für den Projektnamen und den Projekttitel. Wählten Sie zum Beispiel als Projektname 'StructDB' und als Projekttitel 'StructDataBase', dann erscheint im Menü auf dem Ziel-Rechner der Projekttitel 'StructDataBase', während Sie das Programm in der Konsole mit 'StructDB' aufrufen müssen.
- Da später auf dem Entwicklungsrechner im Packer-Assistenten im Schritt 1 wesentliche Angaben aus den Grund-Einstellungen von Gambas *automatisch* in das Installationspaket übernommen werden und bei der Programm-Installation auf dem Ziel-Rechner auch angezeigt werden, sollte man die Angaben zum Bearbeiter unter *Tools> Einstellungen> Identity* bereits rechtzeitig eintragen.
- Der Packer-Assistent fordert Sie im 9. Schritt auf den Ordner anzugeben, in dem der Assistent das generierte Installationspaket ablegen soll. Hier ist es von Vorteil, wenn dieser (leere) Ordner bereits existiert.

11.10.0.3 Projektbeschreibung

Das getestete Projekt StructDB setzt einige Forderungen um, die im Zusammenhang mit der Weitergabe von Gambas-Programmen häufig gestellt werden:

(F1) Festlegung von Programm-Konfigurationen (Konfigurationsdatei)
(F2) Mehrsprachige Anwendung
(F3) Einsatz einer Programm-Hilfe (→ Funktionstaste F1)
(F4) Weitergabe von Dateien, die von allen Nutzern nur gelesen werden können
(F5) Weitergabe von Dateien, die von einem Benutzer Schreib- und Lese-Rechte erfordern
(F6) Programmstart mit und ohne Argument
(F7) Erzeugen eines eigenen MIME-Typs für programm-spezifische Daten-Dateien

In den folgenden Abschnitten und weiteren Kapiteln wird am Beispiel des Projekts StructDB beschrieben, wie Sie die o.a. Forderungen erfolgreich umsetzen. An geeigneten Stellen unterstützen einige Ausschnitte aus dem Quelltext des Projekts die Beschreibung.

11.10.0.4 Festlegung von Programm-Konfigurationen

Für die Festlegung von Programm-Konfigurationen in einer Konfigurationsdatei wird die Komponente *gb.settings* eingesetzt (→ Kapitel 19.1). In der Konfigurationsdatei im *Standardpfad* werden im Projekt StructDB die Fenster-Koordinaten des Programmfensters gespeichert.

Einzig beim ersten Start wird das Programm-Fenster *zentriert* ausgegeben. Danach gelten die Werte, die beim Schließen des Programmfensters in der Konfigurationsdatei gespeichert wurden:

```
Public Sub Form_Open()
…
  If IsNull(structSettings["Window/Top"]) And IsNull(structSettings["Window/Left"]) Then
     FMain.Center
  Else
     FMain.Top = structSettings["Window/Top", FMain.Top]
```

```
    FMain.Left = structSettings["Window/Left", FMain.Left]
  Endif
…
End ' Form_Open()
Public Sub Form_Close()
  structSettings["Window/Top"] = FMain.Top
  structSettings["Window/Left"] = FMain.Left
  If FHelp.Closed = False Then FHelp.Close
End ' Form_Close()
```

11.10.0.5 Mehrsprachige Anwendung

In den Projekteinstellungen wird im Reiter *Optionen* als Standardsprache *Englisch (UK)* ausgewählt, nachdem die Frage *Projekt kann übersetzt werden* mit *Ja* beantwortet wurde. Weitere Informationen zur Internationalisierung (I18N) eines Projekts finden Sie im → Kapitel 20.1.3 Message und I18N und im → Kapitel 11.4 Projekt I18N.

Die Übersetzung der einzelnen Texte in die deutsche Sprache starten Sie am schnellsten mit CTRL+T oder starten im Menü unter Projekt> Übersetzen – nur übersetzen müssen Sie dann selbst.

11.10.0.6 Programm-Hilfe

Spendieren Sie dem Programm *StructDB* und damit allen zukünftigen Nutzern eine Programm-Hilfe. Die Hilfe im Hauptprogramm wird mit der Funktionstaste F1 aufgerufen und im Hauptprogramm das Ereignis *Form_KeyPress()* ausgewertet. Als Reaktion wird ein weiteres Formular – hier FHelp – mit dem Hilfetext angezeigt, das mit der Escape-Taste (ESC) geschlossen wird:

```
Public Sub Form_KeyPress()
  If Key.Code = Key.F1 Then
     FHelp.Show
  Endif
  If Key.Code = Key.ESC And FHelp.Closed = False Then
     FHelp.Delete
  Endif
End ' Form_KeyPress()

Public Sub Form_Show()
  FHelp.Border = 1
  FHelp.Arrangement = Arrange.Fill
  TextArea1.Background = &H00FFFFDF&
  TextArea1.Text = File.Load("help.txt") ' → Relativer Pfad!
End ' Form_Show()
```

Der Hilfetext wird aus einer Datei *help.txt* nur ausgelesen, da er für die aktuelle Programm-Version nicht geändert werden muss. Die Hilfe-Datei liegt in der IDE im logischen Ordner "Daten", was dem physischen Projekt-Ordner entspricht.

Beachten Sie: Alle Dateien im physischen Projektverzeichnis werden beim Generieren der ausführbaren Datei *StructDB.gambas* in die "Gambas-Executable" eingebunden. Sie sind nirgendwo im Dateisystem des Ziel-Rechners zu finden; aber auch nicht unter "Application.Path" erreichbar! Details zu dieser frei gewählten Klassifizierung von Ordnern (logisch/physisch) in einem Gambas-Projekt finden Sie im nächsten Absatz.

11.10.0.7 Verzeichnispfade

Die folgenden Ausführungen sind generell – auch für das Verständnis der Weitergabe von Dateien im Installationsprojekt – notwendig:

Auf einem Datenträger, das wird häufig die Rechner-Festplatte sein, liegt das Gambas-Projekt *StructDB* in einer bestimmten Verzeichnis-Struktur vor. Schalten Sie in Ihrem Datei-Browser die Anzeige versteckter Dateien/Ordner ein, so sehen Sie neben einigen versteckten Dateien auch versteckte Ordner, die Gambas für Steuerdateien verwendet sowie Dateien und Ordner (rot markiert), die Sie selbst in das Projekt-Verzeichnis kopiert oder dort angelegt haben.

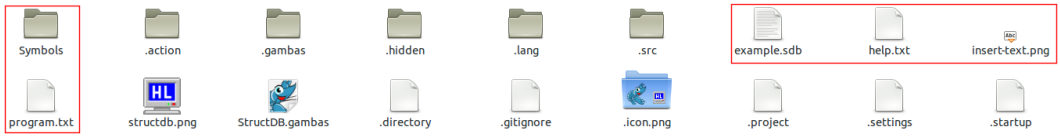

Abbildung 11.10.0.7.1: Verzeichnis-Struktur Projekt-Verzeichnis (Datei-Browser)

Bearbeiten Sie das Projekt *StructDB* in der IDE von Gambas, so fällt Ihnen dort sicher der Unterschied zur Verzeichnis-Struktur in der → Abbildung 11.10.0.7.1 auf:

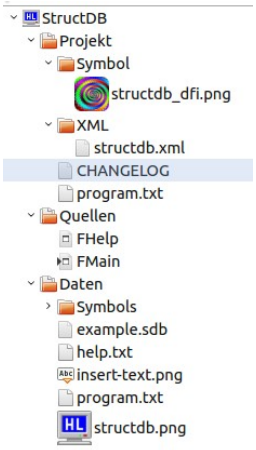

Abbildung 11.10.0.7.2: Verzeichnis-Struktur (Projekt-Ordner) Gambas-IDE

Um den Unterschied klar herauszuarbeiten, wird ein Ordner im folgenden "physisch" genannt, wenn er auf einem Datenträger liegt und Sie ihn mit dem Datei-Browser betrachten können. Ein Ordner dagegen ist vom Typ "logisch", wenn die *Gambas-IDE* einem bestimmten physischen Ordner eine besondere Bedeutung beimisst. Im Dateibrowser sehen Sie nur physische Ordner, in der Gambas-IDE vorwiegend logische. Der physische Ordner *.hidden* entspricht dem logischen Ordner "Projekt". Nur Dateien, die Sie in den versteckten Ordner *.hidden* legen, werden in der IDE im Ordner "Projekt" angezeigt und einzig Dateien aus diesem Ordner können Sie im Schritt 8 des Packer-Assistenten auswählen und in einen Ordner Ihrer Wahl auf dem Ziel-Rechner kopieren. Dateien und Ordner (→ Abbildung 11.10.0.7.1; rot markiert), die Sie in das physische Projekt-Verzeichnis kopiert oder dort angelegt haben, sehen Sie in der IDE im logischen Ordner "Daten". Diese Ordner und Dateien werden beim Kompilieren in die ausführbare Datei *StructDB.gambas* aufgenommen.

11.10.0.8 Weitergabe von Dateien, die von allen Benutzern nur gelesen werden können

Wenn Sie das Installationspaket auch nutzen möchten, um Dateien auf dem Ziel-Rechner system-weit zu installieren, aber vor schreibendem Zugriff regulärer (nicht Root-)Benutzer zu schützen, dann kopieren Sie diese Dateien zur Entwicklungszeit in den physischen Ordner .hidden oder legen diese zum Beispiel in der IDE im logischen Ordner "Projekt" an!

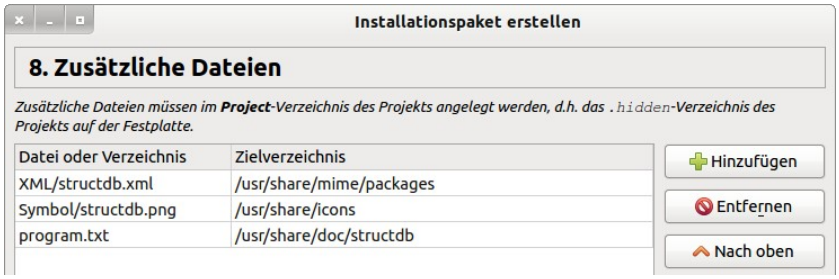

Abbildung 11.10.0.8.1: Dateien im versteckten Ordner .hidden

- Als Ziel-Ordner wählen Sie zum Beispiel */usr/share/doc/StructDB* im Packer-Assistenten im Schritt 8. Unter /usr/share/ befinden sich normalerweise die read-only-(Daten)-Dateien installierter Programme.
- Beim Installieren des Installationspakets auf dem Ziel-Rechner werden die ausgewählten Dateien aus dem logischen Ordner "Projekt" automatisch in die angegebenen Ziel-Pfade kopiert!
- Beachten Sie, dass Sie die Pfade rechtzeitig festlegen, denn diese werden Sie beim Kodieren und Erproben der Anwendung nicht nutzen. Achten Sie deshalb darauf, dass die Datei *program.txt* auch im logischen Ordner "Daten" liegt. Das ist notwendig, wenn Sie das Programm auf dem Entwicklungsrechner erproben. Von der Idee, beim ersten Programm-Start einen Ord-

ner */usr/share/doc/StructDB* anzulegen und dorthin die Datei *program.txt* zu kopieren, sollten Sie sich trennen, denn Sie benötigen Root-Rechte für das Anlegen des Ordners im angegebenen System-Pfad!

- Neben der (Kurz-)Dokumentation *program.txt* werden im Projekt StructDB auch die Icon-Datei *structdb.png* aus dem Ordner *Symbol* und die Datei *structdb.xml* aus dem Ordner *XML* exportiert.

11.10.0.9 Weitergabe von Dateien, die vom Benutzer Schreib- und Lese-Rechte erfordern

Nutzer-spezifische Dateien dagegen sind im Home-Verzeichnis des Benutzers B auf dem Ziel-Rechner bestens aufgehoben. Dort kann der Inhalt nur vom Benutzer B gelesen und bearbeitet werden; im Gegensatz zu allen anderen Benutzern. Diese nutzer-spezifischen Dateien müssen in der IDE-Order-Struktur im logischen Ordner "Daten" liegen – also im physischen Projekt-Ordner. So eine Datei ist im Projekt StructDB *example.sdb* (→ Abbildung 11.10.0.7.1), die als Muster-Datenbank dem Programm StructDB mitgegeben wird und vom Benutzer B Lese- sowie Schreibrechte erfordert.

Aber aufgepasst: Sie können die nutzer-spezifische Datei *example.sdb* nicht mit einem Debian-Paket installieren, da der Debian-Installer immer als Root gestartet wird und nicht erkennen kann, dass Sie beispielsweise der System-Benutzer B mit dem klangvollen Namen *gambanix* sind. Deshalb wird beim *ersten* Start des Programms die ausgewählte nutzer-spezifische Datei *example.sdb* in den Ordner 'StructDB' im Home-Verzeichnis von Benutzer B kopiert. Der Ordner, dessen Namen Sie frei wählen können, wird angelegt, wenn er nicht existiert - was zu prüfen ist:

```
Public Sub Form_Open()
…
  If Not Exist(User.Home &/ "StructDB") Then
     Try Mkdir User.Home &/ "StructDB"

     If Error Then
        Message.Error(Subst$(("Couldn't create the directory &1"), User.Home &/ "StructDB" & Error.Code))
        Quit
     Else
        Try Copy "example.sdb" To User.Home &/ "StructDB/example.sdb"
        If Error Then
           Message.Error(Subst$(("Couldn't copy the file &1"), User.Home &/ "StructDB/example.sdb"))
        Endif
        Wait
     Endif

  Endif ' Not Exist(User.Home &/ "StructDB")
…

End ' Form_Open()
```

11.10.0.10 Programmstart mit und ohne Argument

Der Programmstart von *StructDB* auf einem Ziel-Rechner (Mint 17.1 mit deutschem Sprachpaket) erfolgt entweder über das Menü oder durch einen angelegten Starter auf dem Desktop oder durch den Aufruf in der Konsole – im ersten Fall ohne Argument:

```
hans@linuxmint:~$ /usr/bin/StructDB
hans@linuxmint:~$ StructDB
```

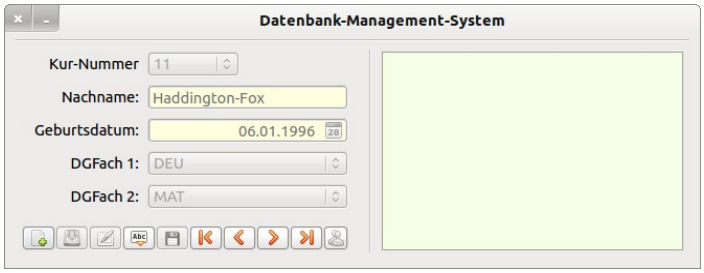

Abbildung 11.10.0.10.1: Programm-Start ohne Argument

Im 2. Fall wird beim Programm-Start die Daten-Datei *example.sdb* als Argument übergeben:

```
hans@linuxmint:~$ StructDB -- ~/Kursdaten/example.sdb
```

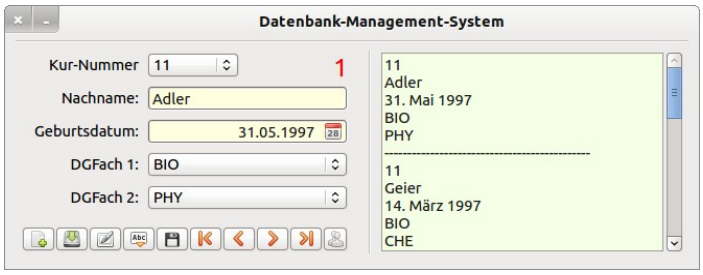

Abbildung 11.10.0.10.2: Programm-Start mit Datei als Argument

Im Quelltext findet sich in der Prozedur *Form_Open()* ein kleiner Abschnitt, der beim Programm-Start das Programm sicher *mit* und *ohne* Argument öffnet:

```
Public sOpenProjectPath As String

Dim sArgsString As String
Dim bOpenByArgs As Boolean

  Args.Begin()
  Args.End()
  For Each sArgsString In Args.End()
    If Not IsNull(sArgsString) Then
       sOpenProjectPath = sArgsString
       Import(sOpenProjectPath)
       tlblRecords.Foreground = Color.Red
       bOpenByArgs = True
    Endif
  Next

  If bOpenByArgs = False Then
     cmbDWC1.Text = cmbDWC1[0].Text
     cmbDWC2.Text = cmbDWC2[1].Text
     tlblRecords.Visible = False
     tlblRecords.Foreground = Color.Red
     Status(False)
     btnUpdate.Enabled = False
     btnSave.Enabled = False
     txbLastname.SetFocus
  Endif ' bOpenByArgs = False ?
```

Eingesetzt werden 2 Methoden der Komponente *gb.args*. Eine detaillierte Beschreibung dieser Komponente finden Sie im → Kapitel 5.8.0 Gambas-Programme mit Optionen und Argumenten.

11.10.0.11 MIME-Typ für die Daten-Dateien

Eleganter wäre die Möglichkeit, dass sich nach einem Doppel-Klick auf eine *.sdb-Datei das Datenbank-Programm *StructDB* öffnet, die Datensätze aus der ausgewählten *.sdb-Datei eingelesen und angezeigt werden. Um das zu realisieren, müssen Sie

- entweder den MIME-Typ der *.sdb-Dateien erkunden und diesen MIME-Typ im Packer-Assistenten im 6. Schritt dem Installationspaket mitgeben
  oder
- vom Installationspaket einen programm-spezifischen MIME-Typ auf dem Ziel-Rechner erzeugen und registrieren lassen → Kapitel 11.10.1. Dazu sind Eintragungen in den Schritten 6 und 8 im Packer-Assistenten notwendig.