19.1.3 Profile

Wollen Sie zum Beispiel bei einem Programm zur Datensicherung ein bestimmtes Verzeichnis auf unterschiedliche Datenträger oder unterschiedliche Verzeichnisse auf eine USB-Festplatte sichern, bietet sich die Verwendung von unterschiedlichen Profilen an. Für jeden dieser Fälle ist dann ein spezielles Sicherungsprofil nötig, das Sie in einer Konfigurationsdatei speichern können. Auch bei eMail- oder FTP-Programmen wird häufig mit unterschiedlichen Konten oder Profilen gearbeitet. Vielfach ist es erforderlich neue Profile anzulegen oder zu ändern oder ungenutzte Profile zu löschen.

Eine Lösung könnte darin bestehen:

- den Profilen einzelne Sektionen in der Konfigurationsdatei zuzuordnen und nur die unter einem ausgewählten Profil gespeicherten Schlüssel-Wert-Paare auszulesen, auszuwerten und dem Programm als (Start-)Profil zur Verfügung zu stellen. Dieser Ansatz erfordert Gambas3 mit erweiterten Eigenschaften und Methoden für die Klasse *gb.settings* oder eine spezielle Lösung für Gambas 2.
- für jedes benötigte Profil zur Laufzeit des Programms eine eigene Konfigurationsdatei anzulegen, um so geänderte Profildaten abzuspeichern oder die Konfigurationsdatei für ein bestehendes Profile zu löschen. Dieser Weg wird hier nicht verfolgt.

Hinweis:
Versuchen Sie nicht, eine Konfigurationsdatei *gleichzeitig* mit einem Settings-Objekt und mit einem seriellen Stream (Open, ...) zu bearbeiten. Ja, es ist möglich, weil Linux das gleichzeitige Öffnen einer Datei durch mehrere Prozesse zulässt, aber es gefährdet die Integrität Ihrer Daten. Sie können nämlich nie wissen, in welcher Reihenfolge die Lese-/Schreibzugriffe ausgeführt werden, weil alle diese Zugriffe gepuffert sind.

```
' Man sollte nicht in Dateien stöbern und Inhalte ändern, die bereits von einem Objekt genutzt werden.

  hFile = OPEN (Application.Path &/ "Profil/ftp.conf") FOR INPUT ' Profilnamen auslesen
  WHILE NOT Eof(hFile)
    LINE INPUT #hFile, sLine
    IF Mid(sLine, 1, 3) = "[P_" THEN
       aProfilMatrix.Add(Mid(sLine, 2, String.Len(sLine) - 2))
       cmbFTPProfilName.Add(Mid(sLine, 4, String.Len(sLine) - 4))
    ENDIF
  WEND
  CLOSE #hFile
```

19.1.3.1 Beispiel 3 – Profil-Manager GB2

Mit Gambas 3 lassen sich Profile auf komfortable Art in einer Konfigurationsdatei verwalten, weil die Klasse Settings um die Eigenschaft Keys erweitert wurde. Um auch unter Gambas 2 mit Profilen arbeiten zu können, wurde diese Klasse für Gambas 2 als SettingsP umgeschrieben. Das Umschreiben war erfolgreich – bis auf den Umstand, dass in den Profilnamen keine Umlaute und kein ß verwendet werden dürfen.
Die Klasse SettingsP benötigt die Klasse _Settings_Keys und das Modul Main. Alle drei Dateien werden Ihnen im Programm-Beispiel 3 zur Verfügung gestellt. Deshalb müssen Sie die folgenden drei Dateien in Ihr Gambas2-Projekt einbinden:

Abbildung 19.1.3.1.1: Modul und Klassen für die Arbeit mit Profilen unter Gambas 2

Es wurde auch die Methode Settings.Clear(Sektion) genutzt, um bestehende Profile zu löschen.

Methode	Beschreibung
Settings.Clear	Diese Methode löscht das Settings-Objekt und gibt den belegten Speicher frei.
Settings.Clear(Sektion)	Diese Methode löscht nur die angegebene Sektion in der Konfigurationsdatei.

Tabelle: 19.1.3.1 Beschreibung von Methoden der Klasse Settings

Mit dem Programm *Profil-Manager* kann man

- neue Profile anlegen,
- bestehende Profile ändern,
- neue oder geänderte Profile in der Konfigurationsdatei speichern oder
- bestehende Profile in der Konfigurationsdatei löschen.

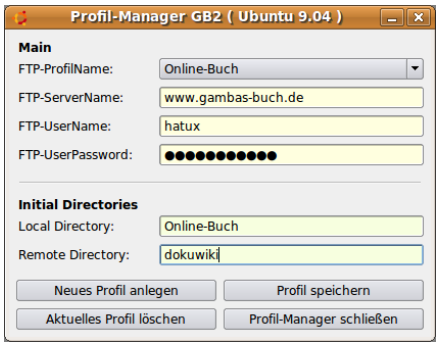

Abbildung 19.1.3.1.2: Profil-Manager – Gambas 2

Die Einträge zum Profilnamen, zum Server, zum User sowie dem Password sind notwendig, während die Angaben zum lokalen (Start-)Verzeichnis und zum Remote-Verzeichnis optional sind.

Da nicht jede Sektion in der Konfigurationsdatei einem Profil zugeordnet ist, werden den *echten* Profilen die beiden Zeichen P_ als Kennung vorangestellt. In der Sektion *[LastProfil]* befindet sich nur ein Schlüssel-Wert-Paar: *LastProfilName="Online-Buch"*, in dem die Information zum zuletzt genutzten Profil steht, wie Sie dem Ausschnitt aus der verwendeten Konfigurationsdatei entnehmen können:

```
# Profile für einen FTP-Client
[LastProfil]
LastProfilName="Online-Buch"

[P_Online-Buch]
FTPServerName="www.gambas-buch.de"
FTPUserName="hatux"
FTPUserPassword="vdo#bs12"
FTPInitialDirLocal="Online-Buch"
FTPInitialDirRemote="dokuwiki"

[P_Gymnasium]
FTPServerName="www.gymnasium.de"
FTPUserName="w00xyz"
FTPUserPassword="f1#skkk"
FTPInitialDirLocal="Bild&Ton"
FTPInitialDirRemote="Projekttage"
...
```

Hier ist der vollständige Quellcode für das Programm *Profil-Manager*:

```
' Gambas class file ---> Gambas 2

PUBLIC pSettings AS SettingsP
PUBLIC aProfilMatrix AS NEW String[]

PUBLIC SUB Form_Open()
  FProfilManager.Center
  FProfilManager.Border = 1
  aProfilMatrix.Clear
  pSettings = NEW SettingsP(Application.Path &/ "Profils/profils.conf", "Profile für einen FTP-Client")
  cmbProfilName.SetFocus
  txtFTPUserPassword.Password = TRUE
  cmbProfilName.ReadOnly = TRUE

  IF pSettings["LastProfil/LastProfilName"] = NULL THEN
     Message.Info("Es existiert kein Profil für das Client-Programm!") ' optional
     cmbProfilName.ReadOnly = FALSE
  ELSE
     LoadSettings(pSettings["LastProfil/LastProfilName"])
  ENDIF ' = NULL?

END ' Form_Open()
```

```
PUBLIC SUB LoadSettings(OPTIONAL sProfilName AS String)
  DIM sSection AS String

  cmbProfilName.Clear()
  aProfilMatrix.Clear
  pSettings.Reload()

  FOR EACH sSection IN pSettings.Keys ' Profile (beginnen mit P_) auslesen und in Matrix speichern
      IF Left(sSection, 2) = "P_" THEN
         aProfilMatrix.Add(Mid(sSection, 3))
      ENDIF
  NEXT

  IF sProfilName THEN
     aProfilMatrix.Remove(aProfilMatrix.Find(sProfilName)) ' Das Profil suchen und in der Matrix löschen
     aProfilMatrix.Add(sProfilName, 0)  ' Das Profil an die 1. Stelle in der Matrix setzen
     GetProfilValues(sProfilName) ' Die Profil-Daten zum Profil anzeigen
  ELSE
     GetProfilValues(aProfilMatrix[0])
  ENDIF ' sProfilName

  cmbProfilName.List = aProfilMatrix
END ' LoadSettings()

PRIVATE SUB SetProfil(sProfilName AS String)
  SetProfilValues(sProfilName)
  LoadSettings(sProfilName)
  cmbProfilName.ReadOnly = FALSE
END ' SetProfil

PRIVATE SUB Reset()
  cmbProfilName.Clear
  cmbProfilName.ReadOnly = FALSE
  txtFTPServerName.Clear
  txtFTPUserName.Clear
  txtFTPUserPassword.Clear
  txtInitialDirLocal.Clear
  txtInitialDirRemote.Clear
  txtFTPUserPassword.Password = FALSE
END ' Reset

PRIVATE SUB ProfilClear(sProfilName AS String)
  IF Message.Question("Soll das Profil " & sProfilname & " gelöscht werden?", "Ja", "Nein") = 1 THEN
     pSettings.Clear("P_" & sProfilName)
     pSettings.Save
  ELSE
     RETURN
  ENDIF
  LoadSettings()
END ' ProfilClear(..)

PUBLIC SUB GetProfilValues(sProfilName AS String)
  sProfilName = "P_" & sProfilName
  txtFTPServerName.Text = pSettings[sProfilName & "/FTPServerName"]
  txtFTPUserName.Text = pSettings[sProfilName & "/FTPUserName"]
  txtFTPUserPassword.Text = pSettings[sProfilName & "/FTPUserPassword"]
  txtInitialDirLocal.Text = pSettings[sProfilName & "/FTPInitialDirLocal"]
  txtInitialDirRemote.Text = pSettings[sProfilName & "/FTPInitialDirRemote"]
END ' GetProfilValues(..)

PUBLIC SUB SetProfilValues(sProfilName AS String)
  sProfilName = "P_" & sProfilName
  pSettings[sProfilName & "/FTPServerName"] = txtFTPServerName.Text
  pSettings[sProfilName & "/FTPUserName"] = txtFTPUserName.Text
  pSettings[sProfilName & "/FTPUserPassword"] = txtFTPUserPassword.Text
  pSettings[sProfilName & "/FTPInitialDirLocal"] = txtInitialDirLocal.Text
  pSettings[sProfilName & "/FTPInitialDirRemote"] = txtInitialDirRemote.Text
  pSettings.Save
END ' SetProfilValues(..)

PUBLIC SUB Form_Close()
  pSettings["LastProfil/LastProfilName"] = cmbProfilName.Text
' Auf den Save-Befehl kann verzichtet werden. Er wird automatisch ausgeführt beim Schließen des Programms.
  pSettings.Save
END ' Form_Close()

PUBLIC SUB cmbProfilName_Click()
  LoadSettings(cmbProfilName.Current.Text)
END ' cmbProfilName_Click()

PUBLIC SUB btnProfilClear_Click()
  ProfilClear(cmbProfilName.Text)
END ' ProfilClear_Click()
```

```
PUBLIC SUB btnProfilCreate_Click()
  IF Message.Question("Soll ein neues Profil angelegt werden?", "Ja", "Nein") = 1 THEN
     Reset()
  ELSE
     RETURN
  ENDIF
END ' ProfilCreate

PUBLIC SUB btnProfilSave_Click()
  IF Message.Question("Das neue oder geänderte Profil <br><b>" & cmbProfilName.Text & \
     "</b><br>speichern?", "Ja", "Nein") = 1 THEN
     IF txtFTPServerName.Text <> "" AND txtFTPUserName.Text <> "" AND \
        txtFTPUserPassword.Text <> "" AND cmbProfilName.Text <> "" THEN
        SetProfil(cmbProfilName.Text)
        cmbProfilName.ReadOnly = TRUE
        txtFTPUserPassword.Password = TRUE
     ELSE
        Message.Error("Die Profil-Daten sind nicht vollständig oder fehlerhaft!")
        RETURN
     ENDIF ' KontoDaten?
  ELSE
     RETURN
  ENDIF ' Question?
END ' Profil speichern

PUBLIC SUB btnManagerClose_Click()
  FProfilManager.Close
END ' btnManagerClose
```

19.1.3.2 Beispiel 4 – Profil-Manager GB3

Mit Gambas 3 lassen sich Profile mit der Klasse Settings anlegen, auslesen sowie auswerten, ändern, speichern und löschen, ohne dem Projekt weitere Dateien wie im Beispiel 3 hinzuzufügen.

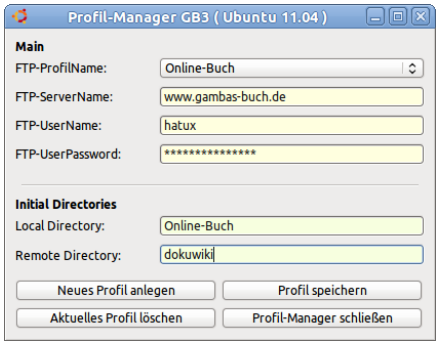

Abbildung 19.1.3.2.1: Profil-Manager – Gambas 3

Die Änderungen im Quelltext bei Gambas 3 im Vergleich zu dem o.a. GB2-Quelltext sind sehr gering und betreffen nur die 2 farbig markierten Zeilen, in denen *SettingsP* gegen *Settings* ausgetauscht wird:

```
' Gambas class file
PUBLIC pSettings AS Settings
PUBLIC aProfilMatrix AS NEW String[]

PUBLIC SUB Form_Open()
  FProfilManager.Center
  FProfilManager.Border = 1
  aProfilMatrix.Clear
  pSettings = NEW Settings(Application.Path &/ "Profils/profils.conf", "Profile für einen FTP-Client")
  cmbProfilName.SetFocus
  txtFTPUserPassword.Password = TRUE
  cmbProfilName.ReadOnly = TRUE
  ...
```

19.1.3.3 Einsatz Profil-Manager

Der Profil-Manager ist ein eigenständiges Programm, das nur für die Verwaltung von Profilen in einer Konfigurationsdatei einer Anwendung eingesetzt wird. Die Anwendung liest die Daten eines im Profil-Manager ausgewählten Profils oder des zuletzt benutzten Profils ein, wertet diese aus und weist sie intern Variablen oder Komponenten-Eigenschaften als Wert zu. Damit greift nur ein Programm auf die Konfigurationsdatei mit den Profilen zu. Der Profil-Manager muss immer im Kontext mit der Anwendung gesehen werden, die ihn nutzt. Einen Manager für alle Programme wird es deshalb nicht geben.

Die Anwendung muss mit Hilfe der Klassen Settings oder SettingsP (für Gambas 2) in der Lage sein, beim *ersten Start* den Profil-Manager aufzurufen und ein erstes Profil anzulegen oder die Daten des zuletzt genutzten Profils auszulesen. Deswegen können Sie mit der Klasse SettingsP/Settings so arbeiten, wie es Ihnen in den Beispielen 1 und 2 vorgestellt wurde.

Im folgenden Abschnitt wird Ihnen ein FTP-Client präsentiert, der den beschriebenen Profil-Manager nutzt. Der FTP-Client ist geeignet, Ihnen den Einsatz des Profil-Managers zu veranschaulichen! Bitte beachten Sie, dass der FTP-Client nur das Einlesen des Remote-Verzeichnisses unter dem ausgewählten Profil von 2 öffentlich zugänglichen FTP-Servern demonstriert:

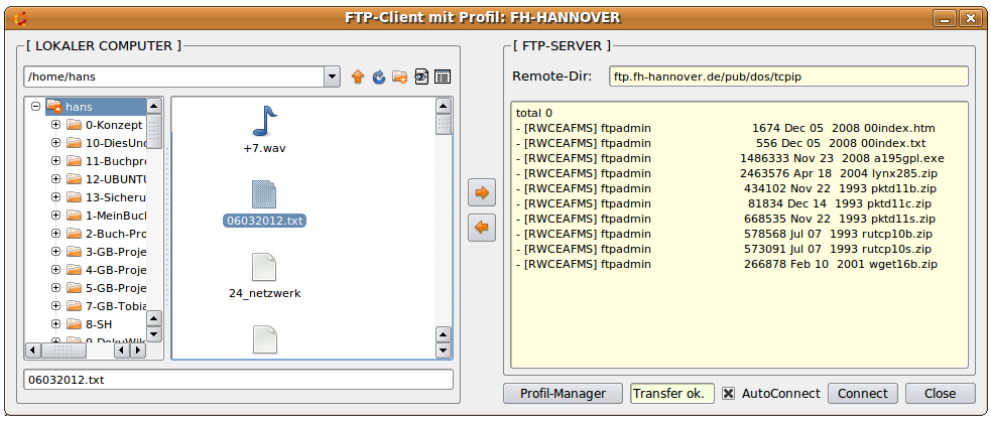

Abbildung 19.1.3.3.1: FTP-Client mit Profil-Manager

Für Ihr Testprogramm können Sie selbstverständlich auch Ihre Daten in ein neues Profil eintragen, wenn Sie über ein eigenes FTP-Konto verfügen.

Der Quelltext wird an dieser Stelle nur für die Prozeduren angegeben, welche das *Zusammenspiel von Profil-Manager und FTP-Client als Anwendung* verdeutlichen. Die komplette Implementierung finden Sie im Projekt *FTPC-Profile* für Gambas 2. Die Änderungen für Gambas 3 sind gering und bereits vorgestellt worden. Sie können für die Anwendung auch weitere Daten – hier die Angaben zum Auto-Connect – in die Konfigurationsdatei schreiben und auslesen, wenn Sie das für notwendig halten, wie ein Blick in die verwendete Konfigurationsdatei und ein zweiter in den Quelltext-Ausschnitt zeigen:

```
# Profile für einen FTP-Client
[AutoConnect]
AutoConnect=-1

[LastProfil]
LastProfilName="FH-HANNOVER"

[P_FH-HANNOVER]
FTPServerName="ftp.fh-hannover.de"
FTPUserName="anonymous"
FTPUserPassword="wer@ist.da"
FTPInitialDirRemote="pub/dos/tcpip"

[P_UNI-ERLANGEN]
FTPServerName="ftp.uni-erlangen.de"
FTPUserName="anonymous"
FTPUserPassword="wer@ist.da"
FTPInitialDirRemote="pub"
```

Im folgenden Abschnitt sehen Sie den Quelltext-Ausschnitt der Anwendung *FTPC-Profile*:

```
' Gambas class file

PUBLIC ftpSettings AS SettingsP ' ---> Gambas 2
...

PUBLIC SUB Form_Open()
  FMain.Center
  FMain.Border = 1
  btnFileUpLoad.Enabled = FALSE
  btnFileDownLoad.Enabled = FALSE
  bFinishedFlag = FALSE
  oFTPClient.Timeout = 5
```

```
' ---> GB2; GB3 ---> ... AS NEW Settings
  ftpSettings = NEW SettingsP(Application.Path &/ "Profils/profils.conf", "Profile für einen FTP-Client")

  IF ftpSettings["LastProfil/LastProfilName"] = NULL THEN
     Message.Info("Es existiert kein Profil für das Client-Programm!")
     FProfilManager.ShowModal
  ELSE
     GetProfilValues(ftpSettings["LastProfil/LastProfilName"])
     FileChooser1.Dir = FileChooser1.Dir &/ InitialDirLocal
     cboxAutoConnect.Value = ftpSettings["AutoConnect/AutoConnect", FALSE]
     IF cboxAutoConnect.Value = TRUE THEN
        ConnectToFTPServer()
     ENDIF
  ENDIF ' = NULL?
END ' Form_Open()

PUBLIC SUB GetProfilValues(sProfilName AS String)
  sProfilName = "P_" & sProfilName
  FTPServerName = ftpSettings[sProfilName & "/FTPServerName"]
  FTPUserName = ftpSettings[sProfilName & "/FTPUserName"]
  FTPUserPassword = ftpSettings[sProfilName & "/FTPUserPassword"]
  InitialDirLocal = ftpSettings[sProfilName & "/FTPInitialDirLocal"]
  InitialDirRemote = ftpSettings[sProfilName & "/FTPInitialDirRemote"]
  FMain.Text = "FTP-Client mit Profil: " & Mid(sProfilName, 3)
  txtRemoteDirectory.Text = FTPServerName &/ InitialDirRemote
END ' GetProfilValues(..)

...

PUBLIC SUB btnProfilManagerShow_Click()
  FProfilManager.ShowModal
  ftpSettings.Reload
  IF ftpSettings["LastProfil/LastProfilName"] = NULL THEN
     FProfilManager.ShowModal
  ELSE
     GetProfilValues(ftpSettings["LastProfil/LastProfilName"])
     FileChooser1.Dir = User.Home &/ InitialDirLocal
     txtRemoteDirectory.Text = FTPServerName &/ InitialDirRemote
     txaRemote.Clear
  ENDIF ' = NULL?
END ' ProfilManagerShow

PUBLIC SUB Form_Close()
   ftpSettings["AutoConnect/AutoConnect"] = cboxAutoConnect.Value
END ' Form_Close()

PUBLIC SUB btnClose_Click()
  IF oFTPClient.Status > 0 THEN oFTPClient.Close
  FMain.Close
END ' btnClose_Click()
```

So lange Sie das Programm *FTF-Client mit Profilmanager* (Projekt FTPC-Profile) in der IDE testen, sehen Sie in der Konsole das Verbindungsprotokoll zwischen FTP-Server und FTP-Client, das interessante Details zeigt. Es kann dann gut interpretiert werden, wenn Sie die RFC 959 zum File Transfer Protocol (FTP) – veröffentlicht auf der Website *http://www.faqs.org/rfcs/rfc959.html* – gelesen haben.