24.9.8.1 Projekte – D-Bus-Objekt erzeugen, exportieren und verwenden

In diesem Kapitel wird Ihnen u.a. ein Gambas-Programm vorgestellt, das einen D-Bus-Server realisiert, der einen bestimmten Service für d-bus-fähige Programme anbietet. Der implementierte Service, den ein zum Session-D-Bus exportiertes D-Bus-Objekt bereitstellt, kann so beschrieben werden:

- Berechnung und Rückgabe des Produktes zwei reeller Zahlen, die als Parameter im Methoden-Aufruf 1 übergeben werden.
- Berechnung und Rückgabe der Summe von zwei ganzen Zahlen, die als Parameter im Methoden-Aufruf 2 übergeben werden.
- Berechnung und Rückgabe des aktuellen Wochentages als Text, wenn als Parameter im Methoden-Aufruf 3 der Funktionswert WeekDay(Now()) vom Datentyp Integer übergeben wird.
- Auslesen des Wertes der Eigenschaft PValue.
- Ändern der Eigenschaft PValue. Leider verhinderte ein Fehler in der Komponente gb.dbus, dass der Wert einer Eigenschaft auf einem D-Bus-Server geändert werden konnte. Der Fehler wurde mit https://gitlab.com/gambas/gambas/commit/be3375e0d29022f3e43adcfc38dafee9684060ba behoben.

Der Fehler in der Komponente gb.dbus machte sich so bemerkbar:

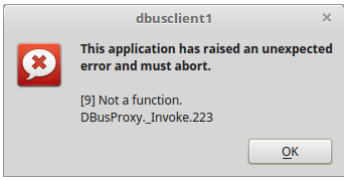

Abbildung 24.9.8.1.1: Fehler-Anzeige

24.9.8.1.1 Projekt Server

Der Quelltext für jeden D-Bus-Server, der einen Service zur Verfügung stellt, besteht aus mindestens 2 Klassen-Dateien. Der Grund liegt darin, das der Service in einem D-Bus-Objekt implementiert wird, das in einer speziellen Klassen-Datei beschrieben wird. Den Namen können Sie frei festlegen – in diesem Projekt Service.class. Das D-Bus-Objekt wird in der Startklasse FMain.class zum Session-D-Bus exportiert, damit der Service auch von anderen d-bus-fähigen Programmen genutzt werden kann.

Service

Für einen D-Bus-Service gilt generell, dass alle Methoden als Funktionen implementiert werden. Sind die Funktionswerte native Datentypen – im vorliegenden Service Integer, Float und String – dann brauchen Sie sich um die Konvertierung von Gambas-Datentypen in D-Bus-Datentypen nicht kümmern. Die Konvertierung erfolgt automatisch. Eigenschaften werden in üblicher Weise deklariert.

Der Quelltext der Klasse Service.class ist erfreulich kurz und übersichtlich:

```
' Gambas class file

Inherits DBusObject ' This instruction is required
Create Static       ' This instruction is required

Property PValue As Float         ' Public variable
Private $fPValue As Float = Pi() ' Local variable

Public Function ComputeAddInteger(Value1 As Integer, Value2 As Integer) As Integer
  Return Value1 + Value2
End

Public Function ComputeMulFloat(Value1 As Float, Value2 As Float) As Float
  Return Value1 * Value2
End

Public Function GetDayOfWeekText(iNumberOfDay As Integer) As String

  Dim aTagesListe As String[]

  aTagesListe = [("Sunday"),("Monday"),("Tuesday"),("Wednesday"),("Thursday"),("Friday"),("Saturday")]
  Return aTagesListe[iNumberOfDay]
```

```
End

Private Function PValue_Read() As Float
  Return $fPValue
End

Private Sub PValue_Write(Value As Float)
  $fPValue = Value
End
```

Im Quelltext von FMain.class ist die Registrierung des D-Bus-Objektes in der Zeile 12 die zentrale Anweisung, nachdem in der Zeile 3 ein neues D-Bus-Objekt vom Typ *Service* erzeugt wurde:

```
[1]  ' Gambas class file
[2]
[3]  Public hDBusObject As Service
[4]
[5]  Public Sub Form_Open()
[6]
[7]    FMain.Resizable = False
[8]    FMain.Caption = ("The data server is activated")
[9]    DBus.Unique = True
[10]
[11]   hDBusObject = New Service
[12]   Try DBus.Session.Register(hDBusObject, "/Service")
[13]   If Error Then
[14]      Message.Error("An instance of " & Application.Name & " already exists.")
[15]      FMain.Close()
[16]   Endif
[17]
[18] End
[19]
[20] Public Sub Form_Close()
[21]   If DBus.IsRegistered(hDBusObject) Then DBus.Session.Unregister(hDBusObject)
[22]   FMain.Close()
[23] End
```

24.9.8.1.2 Projekt Client

Der Quelltext für den Client dbusclient1 wird vollständig angegeben und anschließend kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Private $hDBusProxy As DBusProxy
[4]  Private $sApplication As String
[5]  Private $sObjectPath As String
[6]
[7]  Public Sub Form_Open()
[8]
[9]    Dim sMessage As String
[10]
[11]   FMain.Resizable = False
[12]   FMain.Caption = ("Remote data enquiry via D-Bus")
[13]   Application.MainWindow = FMain
[14]
[15]   $sApplication = "org.gambas.dbusserver1"
[16]
[17]   If Not DBus.Session.Applications.Exist($sApplication) Then
[18]      sMessage = ("There is no suitable data server on the session bus!")
[19]      sMessage &= "<center><font color='red'>"
[20]      sMessage &= ("The program is terminated.")
[21]      sMessage &= "</font></center>"
[22]      Message.Warning(sMessage)
[23]      FMain.Close()
[24]   Else
[25]     $sObjectPath = "/Service"
[26]     $hDBusProxy = DBus[$sApplication][$sObjectPath]
[27]     GetData()
[28]   Endif
[29]
[30] End
[31]
[32] Public Sub btnGetData_Click()
[33]   GetData()
[34] End
[35]
[36] Private Sub GetData()
[37]
[38]   Dim i As Integer
[39]   Dim fF1, fF2 As Float
```

```
[40]   Dim iS1, iS2 As Integer
[41]
[42]   fF1 = Val(txbFactor1.Text)
[43]   fF2 = Val(txbFactor2.Text)
[44]   tboxMul.Text = CStr($hDBusProxy.ComputeMulFloat(fF1, fF2))
[45]
[46]   iS1 = Val(txbSummand1.Text)
[47]   iS2 = Val(txbSummand2.Text)
[48]   tboxSum.Text = CStr($hDBusProxy.ComputeAddInteger(iS1, iS2))
[49]
[50]   i = WeekDay(Now())
[51]   lblDOW.Text = $hDBusProxy.GetDayOfWeekText(i) & "!"
[52]
[53]   tboxPValue.Text = Round($hDBusProxy.PValue, -7)
[54]
[55] End
[56]
[57] Public Sub sboxChange_Change()
[58]
[59]   Dim fCurValue As Float
[60]
[61]   fCurValue = $hDBusProxy.PValue
[62]   Try $hDBusProxy.PValue = fCurValue + sboxChange.Value
[63]   tboxPValue.Text = Round($hDBusProxy.PValue, -7)
[64]
[65] End
[66]
[67] Public Sub btnIntrospection_Click()
[68]   FIntrospection.Show()
[69] End
[70]
[71] Public Sub Form_Close()
[72]   FMain.Close()
[73] End
```

Kommentar:

- In Zeile 17 wird geprüft, ob es die Anwendung mit dem D-Bus-Namen org.gambas.dbusserver1 auf dem Session-Bus gibt. Ist das der Fall, wird nach der Zuweisung in der Zeile 25 in der Zeile 26 ein Proxy erzeugt und konsequent nur mit diesem gearbeitet.
- In der Prozedur GetData() werden alle drei implementierten Methoden mit den erforderlichen Argumenten aufgerufen und die vom Server gelieferten Rückgaben (Funktionswerte) in den vorgesehenen Steuerelementen angezeigt. Zusätzlich wird auch die Eigenschaft PValue ausgelesen und angezeigt.
- Die Prozedur zur Änderung des Wertes der Eigenschaft PValue konnte bisher nicht erprobt werden.

24.9.8.1.3 Einsatz von Server und Client

Server

Abbildung 24.9.8.1.2: Server-GUI

Der Server wird gestartet, exportiert ein D-Bus-Objekt zum Session-D-Bus mit dem implementierten Service und wartet auf Anfragen von d-bus-fähigen Clients.

Der vorgestellte Client nutzt den angebotenen Service des Servers:

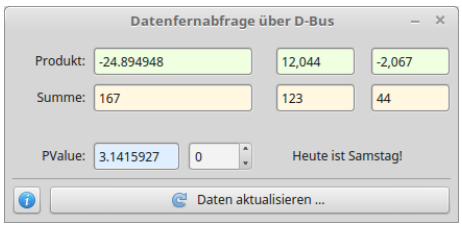

Abbildung 24.9.8.1.3: Berechnung 1 mit Startwerten

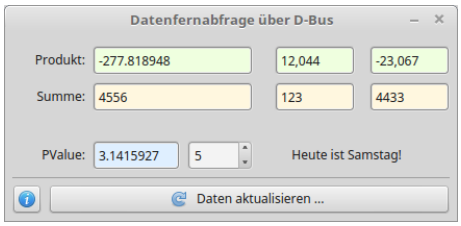

Abbildung 24.9.8.1.4: Berechnung 2

Sie können die Summanden und die Faktoren über Eingabefelder ändern. Den Wert für PValue modifizieren Sie über eine SpinBox in Ganzzahl-Schritten. Anschließend werden das Produkt, die Summe und der aktuelle Tagesname über drei unterschiedliche Methoden-Aufrufe aktualisiert und angezeigt sowie der Wert der PValue-Eigenschaft ausgelesen und ebenso in einer Text-Box angezeigt.

Jederzeit können Sie sich über die implementierten Methoden, deren Signatur und Eingabeparameter sowie über die definierte Eigenschaft 'PValue' über eine Introspection informieren, die über einen Klick auf den i-Button realisiert wird:

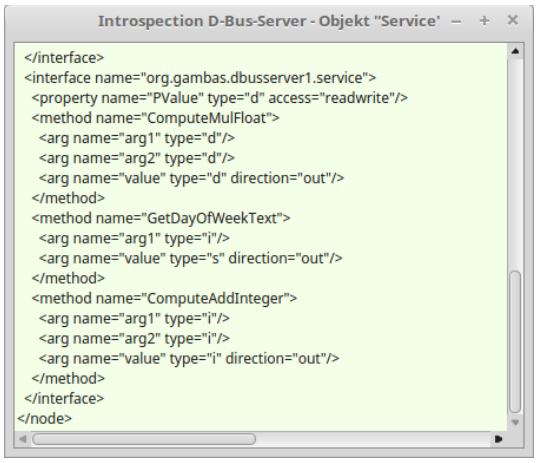

Abbildung 24.9.8.1.5: Erfolgreiche Introspection des Objektes "/Service"