Hello, Library Calling

von Stefan Schnell

Oftmals besteht der Wunsch oder sogar die Notwendigkeit mit einem ABAP®-Programm von einem SAP®-Applikationsserver auf dem Präsentationsserver mit dem Betriebssystem Microsoft® Windows® Operationen auszuführen. Zu diesem Zweck bieten ABAP® und der SAP® GUI für Windows® einige Möglichkeiten an, die jedoch nicht alle Wege bereitstellen. So ist beispielsweise ein direkter Zugriff auf dynamische Bibliotheken, auch als DLLs bekannt, und eine Kommunikation mit dem GUI, sobald das ABAP®-Programm im Hintergrund ausgeführt wird, nicht vorgesehen. Im Folgenden wird beschrieben wie mit der Programmiersprache FreeBASIC eine DLL erstellt und diese mittels DynamicWrapperX mit einem ABAP®-Programm verwendet werden kann. Weiterhin wird aufgezeigt wie dieses Szenario mit Background Light auch in Hintergrundprozessen verwendet werden kann.

Die Wahl der Programmiersprache zur Erstellung einer DLL ist frei. Prinzipiell spielt es keine Rolle welche Sprache verwendet wird. Hier fiel die Wahl exemplarisch auf FreeBASIC, jedoch hätte ebenso ein anderer BASIC-, Pascal- oder C-Dialekt gewählt werden können.

Wir sprechen FreeBASIC

FreeBASIC ist ein frei verfügbarer 32-bit x86-Compiler für Windows® und Linux. Die ursprüngliche Idee hinter dieser Sprache ist die Fortführung eines Ansatzes mit hoher Microsoft® QuickBASIC-Kompatibilität. Daneben bietet FreeBASIC aber auch alle Möglichkeiten moderner Programmiersprachen bis hin zur objektorientierten Programmierung.

Programmierung einer DLL

Eine dynamische Bibliothek mit FreeBASIC zu erstellen ist keine besondere Schwierigkeit. Die Sub-Routinen und Funktionen müssen lediglich mit dem Bezeichner `Export` deklariert und die Kompilierung mit dem Parameter `-dll` durchgeführt werden. Für das folgende Beispiel mit dem Name `Test.dll` wurden drei Routinen mit unterschiedlichen Schnittstellen und Typen ausgeprägt:

1. Die Sub-Routine `DebugPrint` erhält einen Text und übergibt diesen an einen Debugger ohne einen Rückgabewert zu liefern.
2. Die Funktion `MsgBox` erhält zwei Texte und eine Ganzzahl, übergibt diese der Windows® API-Funktion `MessageBox` und liefert als Rückgabewert ebenfalls eine Ganzzahl.
3. Die Funktion `CircleArea` erhält einen Durchmesser als Kommazahl, berechnet die Fläche eines entsprechenden Kreises und liefert diese ebenfalls als Kommazahl zurück.

```
'-Begin------------------------------------------

  '-Includes---------------------------------------
    #Include Once "Windows.bi"

  Extern "Windows-MS"

  '-Sub DebugPrint---------------------------------
    Sub DebugPrint Alias "DebugPrint" _
      (ByVal Text As String) Export
      OutputDebugString Text
    End Sub

  '-Function MsgBox--------------------------------
    Function MsgBox Alias "MsgBox" _
      (ByVal Text As String, _
       ByVal Caption As String, _
       ByVal Style As Integer) As Integer Export
      MsgBox = MessageBox(NULL, Text, Caption, Style)
    End Function

  '-Function CircleArea----------------------------
    Function CircleArea Alias "CircleArea" _
      (ByVal Diameter As Single) _
      As Single Export
      Const Pi As Double = 3.1415926535897932
      CircleArea = Pi * (Sqr(Diameter) / 4)
    End Function

  End Extern

'-End--------------------------------------------
```

DynamicWrapperX COM-Server

DynamicWrapperX ist ein frei verfügbarer COM-Server für Windows®, der die Möglichkeit bietet Funktionen von dynamischen Bibliotheken über die COM-Schnittstelle aufzurufen. Da ABAP® die direkte Verwendung von COM-Bibliotheken unterstützt, können auf diesem Wege DLL-Funktionen auf dem Präsentationsserver aufgerufen werden.

Verwendung dynamisch

Der Einsatz der im Vorherigen erstellten DLL gestaltet sich mit ABAP® und DynamicWrapperX ebenfalls nicht besonders aufwendig. Neben der Erzeugung des Objektes muss jede Funktion erst-einmal registriert werden. Dabei werden mit den Argumenten `'i=...'` die Eingabe- und mit `'r=...'` die Ausgabeparameter spezifiziert, z.B. steht `s` für String, `l` für Long und `f` für Float. Nach der erfolgreichen Registrierung kann nun die DLL-Funktion einfach aufgerufen werden.

```
"-Begin------------------------------------------
  Report zDynWrapX.

    "-Includes---------------------------------------
      Include OLE2INCL.

    "-Variables--------------------------------------
      Data DynWrapX Type OLE2_OBJECT.
      Data ret Type Integer Value 0.
      Data Area Type Float Value '0.0'.
      Data Diameter Type Float Value '1.0'.
```

```
    "-Main-----------------------------------------
      Create Object DynWrapX 'DynamicWrapperX'.
      If sy-subrc = 0 And DynWrapX-Handle <> 0.

        "-DebugPrint-----------------------------------
          Call Method Of DynWrapX 'Register' Exporting
            #1 = 'Test.dll' #2 = 'DebugPrint'
            #3 = 'i=s'.
          If sy-subrc = 0.
            Call Method Of DynWrapX 'DebugPrint'
              Exporting #1 = 'Dies ist ein Test'.
          EndIf.

        "-MsgBox---------------------------------------
          Call Method Of DynWrapX 'Register' Exporting
            #1 = 'Test.dll' #2 = 'MsgBox'
            #3 = 'i=ssl' #4 = 'r=l'.
          If sy-subrc = 0.
            Call Method Of DynWrapX 'MsgBox' = ret
              Exporting #1 = 'Dies ist ein Test'
              #2 = 'Test' #3 = 1.
            Write: / ret.
          EndIf.

        "-CircleArea-----------------------------------
          Call Method Of DynWrapX 'Register' Exporting
            #1 = 'Test.dll' #2 = 'CircleArea'
            #3 = 'i=f' #4 = 'r=f'.
          If sy-subrc = 0.
            Call Method Of DynWrapX 'CircleArea' =
              Area Exporting #1 = Diameter.
            Write: / Area.
          EndIf.

        Free Object DynWrapX.
      EndIf.

"-End------------------------------------------------
```

Hintergrundbeleuchtung

Im Normalfall ist eine Kommunikation zwischen dem Präsentationsserver und einem im Hintergrund laufenden ABAP®-Programm nicht möglich. Background Light ist eine ABAP®-COM-Brücke die es erlaubt COM-Bibliotheken auch in ABAP®-Hintergrundprozessen zu verwenden und somit trotzdem eine Kommunikation zu erlauben. Bei Background Light handelt es sich um ein SAP® Server-Programm auf Basis des SAP® NetWeaver® RFC SDK mit dem jeder COM-Server angesprochen werden kann. Darüber hinaus werden noch weitere Möglichkeiten angeboten z.B. um das Windows® Management Instrumentaion (WMI) zu nutzen.

Blick in den Rückspiegel

Die Umsetzung eines ABAP®-Programmes mit Background Light und DynamicWrapperX sieht ähnlich dem vorherigen Ansatz aus. Neben der zusätzlichen Initialisierung und dem zusätzlichen Rückgabewert hResult sticht die Befehlsübergabe als Zeichenkette ins Auge. Während bei ABAP® diese mit den #-Parametern spezifiziert werden, findet hier die Übergabe als eine Zeichenkette statt. Dieser Ansatz an sich ist zwar etwas gewöhnungsbedürftig, bietet aber den Vorteil der besseren Lesbarkeit.

```
"-Begin----------------------------------------------
  Report zBLight.

    "-Includes-------------------------------------
      Include ZBLIGHTINCL.

    "-Variables------------------------------------
      Data DynWrapX Type Integer Value 0.
      Data ret Type Integer Value 0.
      Data Area Type Float Value '0.0'.
      Data Diameter Type String Value '1.0'.
      Data Cmd Type String Value ''.

    "-Main-----------------------------------------
      BackgroundLightExists DestExists.
      If DestExists = 1.
        CreateObject 'DynamicWrapperX' DynWrapX
          hResult.
        If sy-subrc = 0 And DynWrapX <> 0 And
          hResult = S_OK.

          "-DebugPrint-----------------------------
            CallMethod DynWrapX
              'Register(''Test.dll'', ''DebugPrint'',
''i=s'')'
              hResult.
            If sy-subrc = 0 And hResult = S_OK.
              CallMethod DynWrapX
                'DebugPrint(''Dies ist ein Test'')'
                hResult.
              EndIf.

          "-MsgBox---------------------------------
            CallMethod DynWrapX
              'Register(''Test.dll'', ''MsgBox'',
''i=ssl'', ''r=l'')'
              hResult.
            If sy-subrc = 0 And hResult = S_OK.
              GetPropertyInteger DynWrapX
                'MsgBox(''Dies ist ein Test'',
''Test'', 1 As Long)'
                ret hResult.
              Write: / ret.
            EndIf.

          "-CircleArea-----------------------------
            CallMethod DynWrapX
              'Register(''Test.dll'', ''CircleArea'',
''i=f'', ''r=f'')'
              hResult.
            If sy-subrc = 0 And hResult = S_OK.
              Concatenate 'CircleArea(' Diameter ')'
                Into Cmd.
              GetPropertyFloat DynWrapX Cmd Area
                hResult.
              Write: / Area.
            EndIf.

          FreeObject DynWrapX.
        EndIf.
      EndIf.

"-End------------------------------------------------
```

Zusätzliche Hinweise

- Der Zusatz .dll beim Methodenaufruf Register ist optional.
- Das Verzeichnis system32 ist am besten geeignet als Speicherort der DLL. Aufgrund eingeschränkter Rechte kann dies jedoch zu Problemen führen.

Fazit

Wie gezeigt ist die Verwendung von lokalen Bibliotheken auf dem Präsentationsserver mit ABAP®-Programmen vom Applikationsserver kein Hexenwerk oder Magie. DynamicWrapperX bietet hier einen einfachen Ansatz dies zu realisieren. Damit können auch die Funktionen des WinAPI genutzt werden. Ebenso kann der Aufruf von DLL-Funktionen aus ABAP®-Hintergrundprozessen mit Background Light und DynamicWrapperX simpel

durchgeführt werden, so dass nicht einmal ein Dialogprozess zur Verfügung gestellt werden muss. Auch die Verwendung von Zeigern, die bei DLL-Funktionen öfter üblich sind, können mit einer weiteren COM-Bibliothek ermöglicht werden.

Haben Sie Fragen, Anregungen oder Anmerkungen, so senden Sie einfach eine E-Mail an mail@stschnell.com.

Warenzeichen

- SAP, ABAP und NetWeaver sind eingetragene Warenzeichen der SAP AG
- Microsoft, Windows, und QuickBASIC sind eingetragene Warenzeichen der Microsoft Corporation, USA
- FreeBASIC ist Freeware von Andre Victor und steht unter GPL-Lizenz
- PureBasic ist das Eigentum von Frédéric Laboureur, Fantaisie Software
- DynamicWrapperX ist Freeware und das Eigentum von Yuri Popov
- Background Light ist das Eigentum von Stefan Schnell

Links

- www.freebasic.net
- www.purebasic.com
- DynamicWrapperX
- www.stschnell.com

Anhang A

Ergänzend die identische Ausprägung der dynamischen Bibliothek in einem anderen BASIC-Dialekt – PureBasic. Diese Bibliothek kann mit dem ausgeführten ABAP®-Beispiel verwendet werden.

```
;-Begin------------------------------------------

  ;-Sub DebugPrint-----------------------------------
    ProcedureDLL DebugPrint(Text.s)
      OutputDebugString_(Text)
    EndProcedure

  ;-Function MsgBox----------------------------------
    ProcedureDLL.l MsgBox(Text.s, Caption.s, Style.l)
      ProcedureReturn MessageBox_(0, Text, Caption,
Style)
    EndProcedure

  ;-Function CircleArea------------------------------
    ProcedureDLL.f CircleArea(Diameter.f)
      ProcedureReturn #PI * (Sqr(Diameter) / 4)
    EndProcedure

;-End--------------------------------------------
```

Anhang B

Abschließend noch ein Beispiel mit dem der Zugriff auf das Windows® API demonstriert wird. Es wird die Funktion MessageBox der Bibliothek user32 verwendet, ausgeführt und der Rückgabewert abgefragt.

```
"-Begin------------------------------------------
  Report zDynWrapX.

    "-Includes---------------------------------------
      Include OLE2INCL.

    "-Constants--------------------------------------
      Data IDYes Type i Value 6.
      Data IDNo Type i Value 7.

    "-Variables--------------------------------------
      Data Win32 Type OLE2_OBJECT.
      Data ret Type i.

    "-Main-------------------------------------------
      Create Object Win32 'DynamicWrapperX'.
      If sy-subrc = 0 And Win32-Handle <> 0.

        Call Method Of Win32 'Register'
          Exporting
            #1 = 'user32' #2 = 'MessageBoxW'
            #3 = 'i=hwwu' #4 = 'r=l'.
        If sy-subrc = 0.

          "-Call MessageBox from WinAPI---------------
          Call Method Of Win32 'MessageBoxW' = ret
            Exporting
              #1 = 0 #2 = 'Hello World'
              #3 = 'Test' #4 = 4.

          If ret = IDYes.
            Write: / 'Yes'.
          ElseIf ret = IDNo.
            Write: / 'No'.
          Else.
            Write: / '?'.
          EndIf.

        EndIf.

        Free Object Win32.
      EndIf.

"-End--------------------------------------------
```