Registry-Zugriff in Visual Basic

Alexander Greiner

A. Programmierprojekt: Autostart Manager

Motivation

Sehr viele Free- und Shareware-Programme schreiben einen Eintrag zum automatischen Starten des Programms beim Systemstart in den Autostart-Bereich der Windows-Registrierungsdatenbank (Registry). Ich weiß zwar schon lange, wo in der Registry diese Einträge gespeichert werden, konnte mir aber nie genau merken, wie der Schlüssel heißt und musste daher immer lange umhersuchen.

Also – sag' ich zu mir – schreib' dir doch ein Programm, das alle Einträge in den betreffenden Schlüsseln auflistet und mit dem man auch bestimmte Einträge editieren und löschen, sowie neue Einträge erstellen kann.

Das komische ist, dass ich die betreffenden Schlüssel jetzt – nach der Fertigstellung des Programms – auswendig weiß!

Lösung

Der Registry-Zugriff ist nicht direkt in Microsoft Visual Basic eingebettet. Man benötigt dafür verschiedene API-Funktionen von Windows. Die Programmierung mit API stellt sich leider in manchen Fällen als Herausforderung dar.

Daher wollte ich gleich eine Umgebung schaffen, mit der ein universeller Zugriff auf die Registrierungsdatenbank möglich ist. Dafür standen mehrere Lösungsvarianten zur Auswahl:

- Zusammenfassung der Zugriffsfunktionen in einem Modul (eigene .bas-Datei)
- Kapselung in eine Klasse
- Entwicklung eines ActiveX-Steuerelementes
- Programmierung einer ActiveX-DLL (Dynamic Link Library)

Die Zusammenfassung der Zugriffsfunktionen in eine Modul-Datei ist schon ein sehr guter Ansatz, aber diese Variante gefiel mir nicht besonders. Zu ActiveX-Steuerelementen kann ich nur sagen, dass ich sie nicht leiden kann. Im Falle dieser Implementierung hätte ich das Steuerelement unsichtbar gemacht. Unsichtbare Steuerelementen schaffen aber eine unheimliche Unordnung auf dem Form. Daher mag ich Steuerelemente nicht. ;-)

Die Idee der Programmierung einer ActiveX-DLL hat mir sehr gut gefallen. Und die Voraussetzung für ein ActiveX-DLL-Projekt in Visual Basic ist mindestens eine Klasse. Klassenprogrammierung soll aber nicht der Bestandteil dieses Artikels sein. Deshalb werde ich mich auf die essentielle Zugriffsfunktion und ein Beispiel der Anwendung beschränken.

Implementierung

Zuerst muss man die API-Zugriffsfunktion deklarieren. Dies passiert üblicherweise am Beginn der Klasse bzw. des Moduls. Im folgenden Codefragment sehen Sie nur die Deklaration der Funktion „`RegEnumValue`". (Die Auflistung aller für dieses Programm notwendigen Funktionen würde rasch den Rahmen dieses Artikels sprengen.) Zur angeführten Funktion: `RegEnumValue` liest einen beliebigen Eintrag in einem bestimmten Schlüssel der Registry anhand des übergebenen Index aus.

```
' Deklaration der API-Zugriffsfunktion
Private Declare Function RegEnumValue Lib "advapi32.dll" (
  ByVal hKey As Long, _
  ByVal dwIndex As Long,
  lpValueName As Any, _
  lpcbValueName As Any, _
  ByVal lpReserved As Long,
  lpType As Long, _
  lpData As Any,
  lpcbData As Any) _
  As Long
```

Das Schlüsselwort „`Private`" besagt in diesem Fall, dass die Funktion außerhalb dieser Klasse nicht sichtbar, und daher auch nicht aufrufbar ist. Mit „`Lib`" wird definiert, in welcher Windows-System-DLL sich der Code der besagten Funktion „`RegEnumValue`" befindet. Danach werden innerhalb der Klammern die Über- und Rückgabevariablen deklariert.

```
' Liefert einen beliebigen Eintrag in einem spezifizierten Schlüssel
Private Function pEnumValue(Root&, Key$, Index&, _
  Field As Variant, Value As Variant) As Boolean
Dim lResult&, keyhandle&, dwType&, zw&
Dim puffer$, puffergröße&, fieldpuffer$, fieldpuffergröße&

    'Schlüssel öffnen
    lResult = RegOpenKeyEx(Root, Key, 0, KEY_QUERY_VALUE, keyhandle)
    pEnumValue = (lResult = ERROR_SUCCESS)
    ' Schlüssel existiert nicht, aussteigen
    If lResult  ERROR_SUCCESS Then Exit Function
    fieldpuffergröße = 255 'Über-/Rückgabevariablen einstellen ...
    puffergröße = fieldpuffergröße
    fieldpuffer = String(fieldpuffergröße, " ")
    puffer = fieldpuffer
    ' Eintrag auslesen:
    lResult = RegEnumValue(keyhandle, Index, _
                           ByVal fieldpuffer, fieldpuffergröße, _
                           0&, dwType,
                           ByVal puffer, puffergröße)
    pEnumValue = (lResult = ERROR_SUCCESS)
    'Fehler beim Aufruf von RegEnumValue
    If lResult  ERROR_SUCCESS Then Exit Function
    'String mit dem Feldnamen anpassen
    Field = Left(fieldpuffer, fieldpuffergröße)
    'String mit dem Feldwert anpassen
    Value = Left(puffer, puffergröße)
End Function
```

Die Funktion „`pEnumValue`" habe ich – wie Sie sehen können – ebenfalls als „`private`" definiert. Denn die vorliegende Funktion greift in ihren Funktionsaufrufen direkt auf die Registry zu. Daher sollte man einer solchen Funktion auch alle Freiheiten geben, und alle verwendeten Parameter (Übergabevariablen) sollten einstellbar sein.

Wie gesagt, habe ich diese Funktionen in einer Klasse gespeichert. Das bedeutet, dass die Parameter „`Root`" (Wurzel; z.B. **`HKEY_LOCAL_MACHINE`**) und „`Key`" (Schlüssel; z.B. **`SOFTWARE\Microsoft\Windows`**) intern gespeichert werden, da sie sich sowieso nicht allzu oft ändern werden. (`Private pRoot As Long, Private pKey As String;` wobei ich an dieser Stelle vielleicht sagen sollte, dass ich im Zusammenhang mit öffentlichen und privaten Funktionen und Variablen den letzteren ein „`p`" vor den Namen setze.)

Und deshalb gibt es für den Aufruf der Funktion von außerhalb der Klasse die folgende Funktion:

```
' Liefert einen beliebigen Eintrag im aktuellen Schlüssel
Function EnumValue(Index As Long,
  ByRef Field As Variant, ByRef Value As Variant) As Boolean
  Dim RW As Boolean
  'Eintrag über dessen Index auslesen
  RW = pEnumValue(pRoot, pKey, Index, Field, Value)
  'Nullterminierung wegschneiden
  If RW Then Value = Mid(Value, 1, Len(Value) - 1)
  EnumValue = RW
End Function
```

Hier gibt es dann nur mehr die drei wirklich interessanten Übergabevariablen: Index des Eintrages, Feldname des Eintrages und Feldwert des Eintrages.

Anwendung

Der folgende Code wird beim Klicken auf den *CommandButton* „cmdEinträgeAuflisten“ aufgerufen, liest alle Einträge des Schlüssels „**HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**“ aus und listet deren Feldnamen in der *ListBox* „lstEinträge“ auf.

```
Private Sub cmdEinträgeAuflisten_Click()
  Dim reg As New Registry
  Dim RW As Boolean, fd As Variant, va As Variant, i As Long
  Dim SubKeys&, MaxSubKeyLen&, Values&, MaxValueNameLen&,
MaxValueLen&

  ' Registry-Objekt instanzieren und Variablen einstellen
  reg.SetProperties "HKEY_LOCAL_MACHINE",
   "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
  'Infos über den eingestellten Schlüssel einholen
  RW = reg.KeyInfo(SubKeys, MaxSubKeyLen, Values, _
       MaxValueNameLen, MaxValueLen)

  ' Liste löschen
  lstEinträge.Clear
  ' Daten aller Einträge holen
  For i = 0 To Values - 1
    'Daten des Eintrages mit dem Index i auslesen
    RW = reg.EnumValue(i, fd, va)
    'Name des Eintrages in die Liste aufnehmen
    lstEinträge.AddItem fd
  Next i

  Set reg = Nothing 'Registry-Objekt aus dem Speicher löschen
End Sub
```

5. Download und Kontakt

Das fertige Programm, mit dem Sie alle betreffenden Einträge aus den Bereichen **HKEY_LOCAL_MACHINE** und **HKEY_CURRENT_USER** aulesen, ändern und löschen, sowie neue Einträge hinzufügen können, finden Sie zum Download auf meiner Website unter in der Sektion „**Eigene Programme**“.

Wenn Sie noch fragen zum Programm haben, oder gerne den Quellcode haben würden, kontaktieren Sie mich bitte via E-Mail unter *greiner@pcnews.at* oder per ICQ unter 46304396.

B. Buchbesprechung: Jetzt lerne ich Visual Basic

Monadjemi, Peter: Jetzt lerne ich Visual Basic. Start ohne Vorwissen (Markt & Technik) 1. Auflage, 1999, 533 Seiten, ATS 365,--

Der Untertitel „Start ohne Vorwissen“ verrät schon den sehr einfachen Stil, in dem dieses Buch verfasst wurde. Einfach soll aber in diesem Zusammenhang nicht wenig qualitativ heißen!

Das Buch vermittelt einen unterhaltsamen und überaus leichtverständlichen Einstieg in die Programmierung mit Microsoft Visual Basic. Es richtet sich in erster Linie an alle diejenigen, die noch nie vorher programmiert haben. Spezielle Vorkenntnisse werden daher nicht vorausgesetzt.

Aufgrund der Fragen sowie der interessanten und spannenden Übungsaufgaben, die die einzelnen Kapitel abrunden, ist es besonders zum Selbststudium geeignet.

Im Kapitel 13 des Buches – und dies ist auch der Grund für den ersten Teil des Artikels – ist auch eine sehr schnelle Einführung in die Klassenprogrammierung mit Visual Basic und sogar der gesamte Code eines Beispielprogramms zum direkten Auslesen von bestimmten Werten aus der Registry enthalten.

Die mitgelieferte CD-ROM enthält sämtliche Übungsaufgaben des Buches, deren Lösungen sowie eine kostenlose Version des *Microsoft Visual Basic 6.0 Working Model*. Dieses sogenannte „Ablaufmodell“ benötigt einen Windows-PC mit installiertem Visual Basic 5 oder 6 und es ist nicht möglich, EXE-Dateien damit herzustellen.

Weiters findet man auf der CD auch die Visual Basic 5.0 CCE (*Control Creation Edition*), mit der man ActiveX-Steuerelemente programmieren kann. Leider ist die CCE hier nur in Englisch verfügbar.

Java lernen

Thomas Morawetz

Judy Bishop; Java lernen; ISBN: 3-8273-1605-7; Addison-Wesley; 437 öS

Zielgruppe des Buches sind Programmieranfänger bzw. Personen, die Java erlernen wollen. Die Autorin geht vom "blutigen" Anfänger aus, der nun mit dem Programmieren beginnen möchte und als erste Programmiersprache ist eben Java gewählt.

Es werden nicht nur alle wichtigen Besonderheiten und Aspekte der Java-Programmierung erklärt, sondern es wird auch auf die Prinzipien einer modernen Programmierung: von objektorientierter und strukturierter Programmierung über Softwareentwicklung, Algorithmen und Datenstrukturen, Benutzeroberflächen bis hin zu Multi-Threading und Arbeiten im Netzwerk eingegangen. Natürlich kommt auch die Web-Programmierung im Buch nicht zu kurz. Es gibt zahlreiche Beispiele, die aufeinander aufbauen, Übungen und Aufgaben am Ende jedes Kapitels um das Erlernte zu vertiefen.

Dem Titel Lehrbuch trägt das Werk ganz zu recht, es eignet sich gleichermaßen für das Selbststudium mit der Möglichkeit auch später wieder nachschlagen zu können, als auch als Lehrbuch um anderen die Programmierung oder Java oder beiden zu erklären und zu erlernen.

Es wird ein Gesamtwerk geboten, das zusätzlich noch durch die beiliegende CD mit JDK 1.1., den gesamten Beispielen und Vorlesungsmaterial, das an der Universität von Paderborn bei der Arbeit mit diesem Buch entstanden ist, sowie weiterer Freeware abgerundet wird.

Ein Link zum Projekt aus dem dieses Buch entstand und zur Autorin einer Professorin an der Universität von Pretoria Südafrika.
http://www.cs.up.ac.za/javagently/