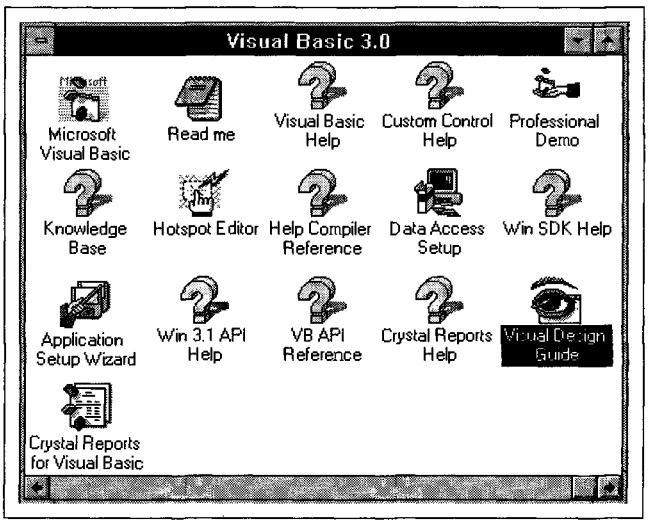


Programmieren für Juristen: Visual Basic – Lektion III

Maximilian Herberger

Ein Sucb-Interface

Am Ende der letzten Folge war demonstriert worden, wie man mit Hilfe eines SQL-Befehls einen Suchfilter über die Sätze einer Datenbank legen kann. Dabei war darauf hingewiesen worden, daß dieses Verfahren in großen Datenbeständen zeitkritisch ist. Aus diesem Grunde ist es notwendig, auch andere Suchverfahren vorzusehen. Um das dabei anzuwendende Prinzip zu veranschaulichen, reichern wir in einem ersten Anlauf unser "Formular" um eine Fundstellensuche an *(vgl. Abb. 1)*.

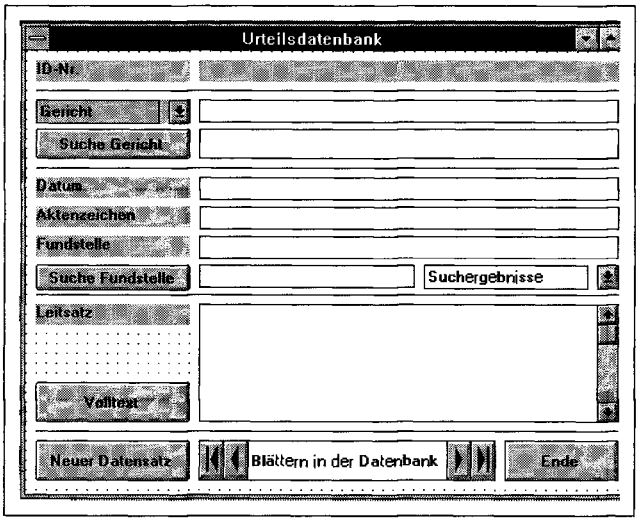
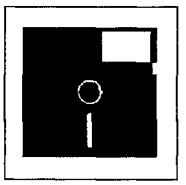


Abb. 1: "Formular" mit Fundstellensuche

Dies geschieht über einen Befehlsknopf "Suche Fundstelle" unterhalb des Anzeige-/Bearbeitungsfeldes "Fundstelle" und eine rechts daneben plazierte Combo-Box, die die Suchergebnisse aufnehmen soll.
Hinter "Suche Fundstelle" liegt der folgende Code:

```
Sub Command5_Click ()
    combo2.Clear
    combo2.Text = "Suchergebnisse"
    Dim kriterium As String
    kriterium = "Fundstelle like " + "'" + text5.Text + "*'"
    datal.Recordset.FindFirst kriterium
    combo2.Addltem datal.Recordset.Fields("az")
    While Not datal.Recordset.NoMatch
```




```
        datal.Recordset.FindNext kriterium
        If Not datal.Recordset.NoMatch Then
           combo2.Addltem datal.Recordset.Fields("az")
        End If
    Wend
End Sub
```

Erläuterung:
"combo2.clear" löscht den Inhalt der Combo-Box. Das ist notwendig, weil sonst bei jeder neuen Suche die neuen Suchergebnisse mit den alten kumuliert würden.
"combo2.text" stellt den Text in der Combo-Box wieder her, der bei der vorherigen Löschoperation natürlich mit gelöscht wurde.
Als Suchkriterium wird dann der Inhalt des Textfeldes neben dem Button "Suche Fundstelle" übernommen, in das die gewünschte Suche eingegeben wird. Die Syntax muß dabei dem "Where"-Teil eines SQL-Befehls genügen. (Erläuterungen und Beispiele finden sich in der Online-Hilfe unter "where"). Bei Eingabe etwa von "jur-pc" lautet die "where"-Klausel: "Fundstelle like 'jurpc*'". Dies bedeutet, daß sich alle Datensätze qualifizieren, die im Feld "Fundstelle" mit der Zeichenfolge "jur-pc" beginnen. "FindFirst Kriterium" findet dann den ersten Datensatz, der dem Kriterium genügt.
Es ist nun noch erforderlich, die Suche solange weiterzuführen, bis alle dem Kriterium entsprechenden Datensätze gefunden wurden. Dies geschieht in der While-Wend-Schleife, die abläuft, solange die Eigenschaft "NoMatch" (= "Kein Treffer") nicht wahr ist. Bei jedem Treffer wird der Combo-Box mit "AddItem" das Aktenzeichen des gefundenen Datensatzes hinzugefügt.
Will man erreichen, daß in der Combo-Box die Ergebnisliste sortiert erscheint, muß man die Eigenschaft "sorted" auf "true" stellen *(vgl. Abb. 2)*. Diese Eigenschaft kann beim Ablauf des Programms nicht beeinflußt werden, sie muß also vorher gesetzt sein.
Starten wir nun das Programm und führen eine Fundstellensuche mit "jur-pc" durch, so ergibt sich das aus Abb. 3 ersichtliche Bild: In der "aufgeklappten Combo-Box erscheinen die Aktenzeichen der einschlägigen Entscheidungen in sortierter Form. Selbstverständlich kann man, wenn man die Aktenzeichen in einer Ergebnisliste nicht für besonders aussagekräftig hält, durch Abändern des Ausdrucks *"combo2.AddItem datal.Recordset.Fields("az")"* jeden Feldinhalt in die Combo-Box befördern.

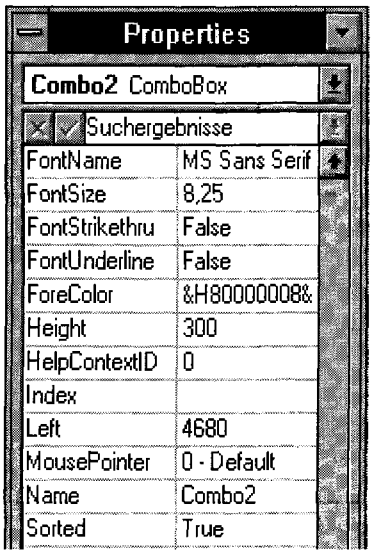


Abb. 2:
Eigenschaft "sortiert" für die Ergebnisliste

Sortierte Ergebnisliste

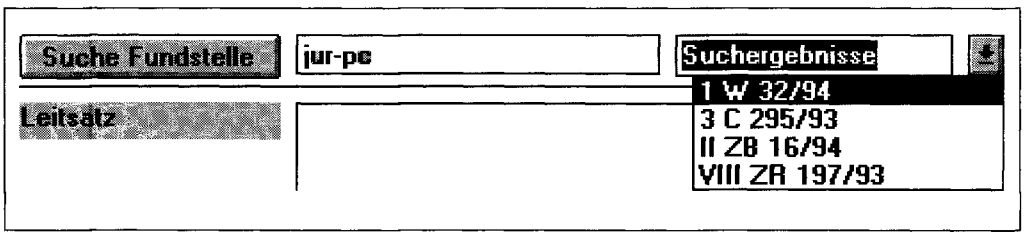


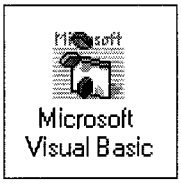
Abb. 3:
Fundstellensuche mit "jur-pc"

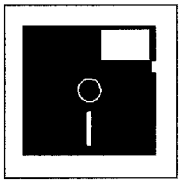
Von der Ergebnisliste zum Dokument

Die Ergebnisliste muß auch noch dazu dienen, durch "Click" auf den gewünschten Eintrag das zugehörige Zieldokument aufzurufen. In einem ersten Anlauf könnte man versuchen, das dadurch zu erreichen, daß man der Combo-Box für das Ereignis "Click" den folgenden Code hinterlegt:

```
Sub Combo2_Click ()
    Dim suchaz As String
    suchaz = "az like " + "'" + Combo2.Text + "*'"
    Datal.Recordset.FindFirst suchaz
End Sub
```

Hier wird das Aktenzeichen, das der Anwender per Click auswählt (es befindet sich in Combo2.Text) nach der eben beschriebenen Methode für eine Suche benutzt. Dieses Verfahren enthält aber einen schwerwiegenden Mangel: Es arbeitet nur solange fehlerlos, wie das gewählte Suchkriterium singulär ist. Dies ist schon bei Aktenzeichen auf der Ebene der Instanzgerichte nicht der Fall. Es kann also bei dieser Lösung nicht bleiben. Hier kommt nun unsere ID-Nr. ins Spiel.
Um "punktgenau" auf einen Datensatz zugreifen zu können, benötigt man eine individuelle Identifikationsmöglichkeit. Jede Datenbank sollte deswegen eine derartige eigene Ken-





"Parallele Buchführung": Die Eigenschaft "ItemData"

nung für jeden Datensatz vorsehen, bei der sichergestellt ist, daß sie jeweils nur einmal vorkommt. In der vorliegenden Datenbank erfüllt die ID-Nr. (im Feld "lfdnr") diese Aufgabe. Wenn wir über die ID-Nummer auf die Datensätze zugreifen wollen, die Ergebnis einer Suche sind, müssen wir die ID-Nr. bei den Suchergebnissen mitnotieren. Zu diesem Zweck greifen wir auf die für Listen (und mithin auch Combo-Boxen) verfügbare Eigenschaft "ItemData" zurück. Was hat man darunter zu verstehen?
"ItemData" konstruiert zu einer Liste (und allen listenartigen Objekten, also auch Combo-Boxen) eine zweite parallele Liste, in die man Elemente aufnehmen kann, die mit einem Listeneintrag mitgeführt werden sollen. In unserem Fall ist das die ID-Nr. Allerdings ist das Vorhaben nicht ganz trivial, weil unsere Liste ja sortiert ist, d.h. wir wissen nicht auf Anhieb, an welcher Stelle ein neuer Eintrag eingefügt werden wird. Indessen nimmt uns Visual Basic diese Arbeit ab. Mit "NewIndex" kann man – wie aus den folgenden Zeilen ersichtlich – an der korrekten Sortierstelle einfügen:

```
combo2.AddItem Data1.Recordset.Fields("az")
combo2.ItemData(combo2.NewIndex) = Data1.Recordset.Fields("lfdnr")
```

Die erste Zeile ist uns bereits aus unseren bisherigen Bemühungen bekannt (s.o.) Die zweite Zeile besagt, daß an der jeweiligen Einfügestelle (durch *NewIndex* bestimmt) in dem zur Liste parallelen Array *Combo2.Itemdata* der Inhalt des Datenbankfeldes "lfdnr" eingefügt werden soll – und das ist unsere Id-Nr. Fügt man diesen Code "hinter" unserem Befehls-Button "Suche Fundstelle" ein, so wird bei jeder Suche parallel zur Ergebnis-Combo-Box die Liste mit den ID-Nummern für die Treffer aufgefüllt.
Es liegt auf der Hand, daß wir nun auch noch den Code für die Auswahl aus der Ergebnisliste modifizieren müssen. Dies geschieht folgendermaßen:

```
Sub Combo2_Click ()
    ziel = "lfdnr = " + Combo2.ItemData(Combo2.ListIndex)
    Data1.Recordset.FindFirst ziel
End Sub
```

Der Ausdruck *Combo2.ItemData(Combo2.ListIndex)* liefert für das Aktenzeichen, das "angeclickt" wurde, die parallel mitgeführte ID-Nummer. Damit wird dann in der bereits erläuterten Weise ein Suchstring aufgebaut, der im Feld "lfdnr" nach der ID-Nummer sucht. Da diese singulär ist, wird – wie erforderlich – "punktgenau" der einschlägige Datensatz gefunden.