Kombinationsfeld um Suche erweitern

Kombinationsfelder sind schon eine praktische Einrichtung: Sie erlauben nicht nur die Auswahl von Einträgen, die schon nach dem Alphabet voreingestellt sind, sondern auch noch die Eingabe der Anfangsbuchstaben der gesuchten Inhalte. Was aber, wenn Sie doch gezielter nach einem Eintrag im Kombinationsfeld suchen wollen – beispielsweise nach Einträgen, die bestimmte Vergleichstexte an beliebiger Stelle enthalten? Dieser Artikel zeigt, wie Sie für ein Kombinationsfeld nach Wunsch per Doppelklick oder über eine zusätzliche Schaltfläche einen Dialog öffnen, mit dem Sie schnell den gesuchten Eintrag auswählen können.

Beispieldatenbank

Die Beispiele dieses Artikels finden Sie in der Datenbank **1805_Kombisuche.accdb**.

Vorbereitung

Wir wollen uns die eingebauten Suchfeatures und usnere neuen Funktionen an einem ungebundenen Kombinationsfeld ansehen, das wir in einem neuen Formular namens **frmKombisuche** anlegen.

Das Kombinationsfeld soll **cboArtikel** heißen und als Datensatzherkunft die folgende SQL-Abfrage verwenden:

```
SELECT ArtikelID, Artikelname
FROM tblArtikel
ORDER BY Artikelname;
```

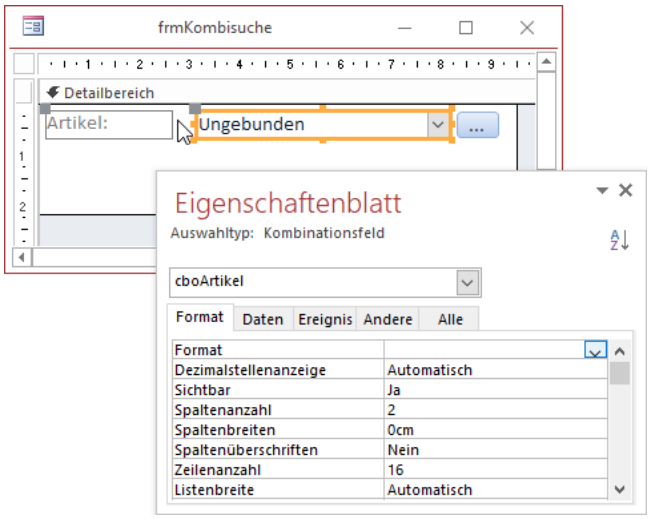

Bild 1: Entwurf des Formulars mit dem Kombinationsfeld

Es soll also die Einträge der Tabelle **tblArtikel** liefern, wobei wir nur die beiden Felder **ArtikelID** und **Artikelname** berücksichtigen. Diese sollen alphabetisch nach den Werten des Feldes **Artikelname** sortiert werden. Rechts neben dem Kombinationsfeld fügen wir noch eine Schaltfläche namens **cmdSuche** ein, der wir als Beschriftung schlicht drei Punkte hinzufügen.

Damit das Kombinationsfeld nur die Werte des Feldes **Artikelname** anzeigt, nicht jedoch die der gebundenen ersten Spalte **ArtikelID**, stellen wir die Eigenschaft **Spaltenanzahl** auf **2** und die Eigenschaft **Spaltenbreiten** auf **0cm** ein. Auf diese Weise wird die erste Spalte ausgeblendet (siehe Bild 1).

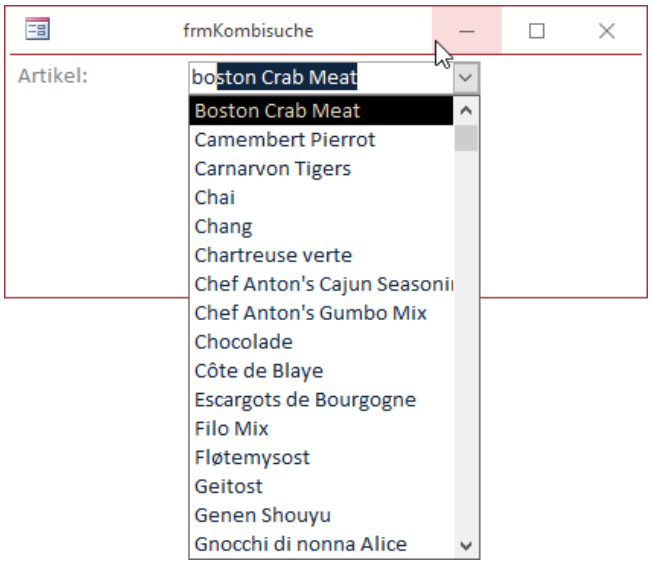

Bild 2: Markieren des Eintrags mit den eingegebenen Anfangsbuchstaben

Eingebaute Suchfeatures

Wenn wir nun in die Formularansicht des Formulars wechseln und das Kombinationsfeld aufklappen, zeigt dieses zunächst alle Einträge der Datensatzherkunft an. Geben Sie nun einen oder mehrere Buchstaben ein, springt das Kombinationsfeld automatisch zum ersten Eintrag, der mit den gleichen Buchstaben beginnt, und markiert diesen wie in Bild 2.

Das geschieht solange, bis sie einen Text eingegeben haben, der in den Werten des angezeigten Feldes nicht mehr gefunden wird – dann finden Sie nur noch den eingegebenen Text im Kombinationsfeld vor und die aufgeklappte Liste zeigt wieder alle Einträge der Datensatzherkunft an.

Diese Funktion ist also im Prinzip kein Filter, sondern nur ein Anspringen des ersten Eintrags, der den eingegebenen Zeichen entspricht. Damit wir noch gezielter suchen können, wollen wir nun noch ein zusätzliches Formular hinzufügen, dass per Doppelklick oder per Mausklick auf eine zusätzliche Schaltfläche neben dem Kombinationsfeld aufgerufen werden kann.

Suchformular anlegen

Das Suchformular wollen wir **frmArtikelsucheKombi** nennen. Es soll ein Textfeld enthalten, mir dem der Benutzer den Suchbegriff eingibt sowie ein Listenfeld, das die gefundenen Einträge der Datenherkunft liefert.

Da das Formular selbst nicht an eine Datenherkunft gebunden sein soll, können wir die Eigenschaften **Datensatzmarkierer**, **Navigationsschaltflächen**, **Trennlinien** und **Bildlaufleisten** auf den Wert **Nein** einstellen. Außerdem soll die Eigenschaft **Automatisch zentrieren** den Wert **Ja** erhalten.

Oben im Formular fügen wir das Textfeld namens **txtSuche** ein. Darunter legen wir ein Listenfeld namens **lstSuchergebnis** an. Dieses binden wir über die Eigenschaft **Datensatzherkunft** zunächst an die gleiche Abfrage, die wir bereits weiter oben für das Kombinationsfeld selbst genutzt haben:

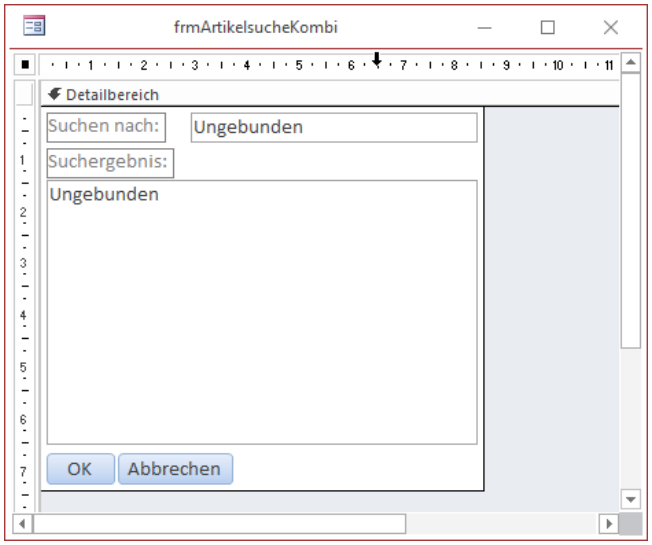

Bild 3: Entwurf des Suchformulars

```
SELECT ArtikelID, Artikelname
FROM tblArtikel
ORDER BY Artikelname;
```

Damit das Listenfeld, genau wie das Kombinationsfeld, nur die Artikelnamen anzeigt, aber nicht die Werte des Feldes **ArtikelID**, stellen wir auch hier die Eigenschaften **Spaltenanzahl** und **Spaltenbreiten** auf die Werte **2** und **0cm** ein.

Außerdem fügen wir noch eine Schaltfläche namens **cmdOK** und eine weitere Schaltfläche namens **cmdAbbrechen** zum Formularentwurf hinzu, der dann wie in Bild 3 aussieht.

Öffnen des Suchformulars

Bevor wir uns nun um die Funktionalität des Suchformulars kümmern, wollen wir erst einmal den Aufruf dieses Formulars realisieren. Dieser soll über zwei Möglichkeiten realisiert werden:

- einen Doppelklick auf das Kombinationsfeld und
- einen Klick auf die Schaltfläche **cmdSuche** rechts neben dem Kombinationsfeld.

Also hinterlegen wir eine Ereignisprozedur für die Ereigniseigenschaft **Beim Doppelklicken** des Kombinationsfeldes und eine für die Eigenschaft **Beim Klicken**

der Schaltfläche **cmdSuche**. Für beide tragen wir den Aufruf einer weiteren Prozedur namens Kombisuche ein:

```
Private Sub cboArtikel_DblClick(Cancel As Integer)
    Call Kombisuche
End Sub

Private Sub cmdSuche_Click()
    Call Kombisuche
End Sub
```

Der Hintergrund ist, dass beide die gleiche Aktion auslösen sollen – nämlich die Anzeige des Suchformulars und die Verarbeitung des dort gewählten Suchergebnisses. Also legen wir die dazu notwendigen Anweisung nur in einer einzigen Prozedur an, die dann von beiden Ereignisprozeduren aufgerufen wird. Auf diese Weise sparen wir uns redundanten Code.

Die durch diese beiden Ereignisprozeduren ausgelöste Prozedur sieht zunächst wie folgt aus:

```
Private Sub Kombisuche()

End Sub
```

Diese werden wir aber schon bald mit den notwendigen Anweisungen füllen. Vorher überlegen wir uns, was genau geschehen soll, wenn wir entweder einen Doppelklick auf dem Kombinationsfeld ausführen oder die Schaltfläche **cmdKombisuche** anklicken. Fest steht: Wir wollen das Formular **frmArtikelsucheKombi** öffnen, damit es uns die gezielte Suche nach einem der Einträge des Kombinationsfeldes erlaubt.

Was soll dann geschehen? Der Benutzer sucht nach einem Eintrag, der dann im Listenfeld markiert wird. Dann klickt er entweder auf die Schaltfläche **cmdOK** oder **cmdAbbrechen**. Klickt er auf **cmdOK** und es ist ein Eintrag markiert, soll dieser für das Kombinationsfeld ausgewählt werden. Das Formular muss dann noch geöffnet bleiben, damit wir den Wert auslesen können.

Klickt er auf **cmdOK** und es ist kein Eintrag markiert oder klickt er auf **cmdAbbrechen**, wird das Formular einfach geschlossen, denn dann brauchen wir ja auch keinen Wert auszulesen.

Wir nutzen also die bekannte Vorgehensweise: Wir öffnen das Formular als modalen Dialog, also in der Art, dass der Benutzer erst zu den übrigen Elementen der Benutzeroberfläche zurückkehren kann, wenn er das aufgerufene Formular entweder geschlossen oder es unsichtbar gemacht hat. In der aufrufenden Prozedur, die dann den Fokus zurückerhält und weiter ausgeführt wird, brauchen wir dann nur zu prüfen, ob das Formular noch geöffnet ist (allerdings unsichtbar) oder ob es nicht mehr geöffnet ist.

Dazu nutzen wir die folgende einfache Funktion, die wir in einem neuen Standardmodul namens **mdlTools** speichern:

```
Public Function IstFormularGeoeffnet(_
                    strFormular As String) As Boolean
    IstFormularGeoeffnet = SysCmd(_
       acSysCmdGetObjectState, acForm, strFormular) > 0
End Function
```

Die Prozedur, die durch unsere Ereignisprozeduren ausgelöst wird, gestalten wir dann wie in Listing 1.

Nach dem Öffnen des Formulars **frmArtikelsucheKombi** als modalen Dialog wird die Prozedur unterbrochen, bis das Formular **frmArtikelsucheKombi** entweder geschlossen oder ausgeblendet wurde. Im letzteren Fall ermitteln wir aus dem Listenfeld den Primärschlüsselwert des gewählten Eintrags und tra-

```
Private Sub Kombisuche()
    DoCmd.OpenForm "frmArtikelsucheKombi", WindowMode:=acDialog
    If IstFormularGeoeffnet("frmArtikelsucheKombi") Then
        Me!cboArtikel = Forms!frmArtikelsucheKombi!lstSuchergebnis
        DoCmd.Close acForm, "frmArtikelsucheKombi"
    End If
End Sub
```

Listing 1: Öffnen des Formulars für die Suche und Einstellen des gefundenen Wertes, wenn das Formular unsichtbar gemacht wurde

gen diesen für das Kombinationsfeld **cboArtikel** ein. Außerdem müssen wir in diesem Fall auch das noch geöffnete Formular **frmArtikelsucheKombi** schließen.

Formular frmArtikelsucheKombi mit Funktion versehen

Nun ist das Formular **frmArtikelsucheKombi** allerdings noch komplett funktionslos. Es sucht weder nach Artikeln, wenn der Benutzer einen Suchbegriff eingibt, noch wird es geschlossen oder unsichtbar gemacht, wenn er auf eine der Schaltflächen **cmdOK** oder **cmdAbbrechen** klickt.

Also statten wir es nun mit den notwendigen Ereignisprozeduren aus. Was soll genau geschehen, wenn der Benutzer ein Zeichen in das Textfeld **txtSuche** eingibt? Wir gehen an dieser Stelle davon aus, dass es sich um eine Datenbank mit lokalen Daten handelt und wollen die angezeigten Einträge nach der Eingabe eines jeden Zeichens aktualisieren. Aber auch hier müssen wir uns noch überlegen, ob und wie wir mit Platzhaltern umgehen, also etwa mit dem Sternchen (*) als Platzhalter für beliebige Zeichen in beliebiger Anzahl:

- Wir können entweder fest definieren, dass der eingegebene Suchbegriff immer an beliebiger Stelle im Artikelnamen gesucht wird.
- Oder wir legen fest, dass wir nur nach Artikelnamen suchen, die mit dem eingegebenen Text beginnen.
- Wir könnten auch festlegen, dass immer genau nach dem eingegebenen Text gesucht wird und der Benutzer somit benötigte Platzhalter selbst eingeben muss.
- Oder wir bieten dem Benutzer die Möglichkeit, einzustellen, ob er die eine oder andere Variante nutzen möchte.

Fürs erste legen wir einfach fest, dass wir nach Artikelnamen suchen, die mit den Buchstaben beginnen, die der Benutzer in das Suchfeld eingibt – also hängen wir jeweils das Sternchen an den Suchbegriff an.

Suchen nach Artikelnamen mit bestimmten Anfangsbuchstaben

Wenn wir nach der Eingabe eines jeden Zeichens neu filtern wollen, können wir nur das Ereignis **Bei Änderung** nutzen. Dieses legen wir wie in Listing 2 an.

Die Prozedur ermittelt über die **Text**-Eigenschaft des Textfeldes **txtSuche** den aktuell eingegebenen Suchbegriff und speichert diesen in der Variablen **strSuchtext**.

Anschließend prüft sie mit der Len-Funktion, ob **strSuchtext** überhaupt einen Suchbegriff enthält. Ist dies der Fall, wird ein Kriterium zusammengestellt, das für den Suchbegriff **A** beispielsweise wie folgt aussieht:

```
WHERE Artikelname LIKE 'A*'
```

```
Private Sub txtSuche_Change()
    Dim strSuchtext As String
    Dim strFilter As String
    strSuchtext = Me!txtSuche.Text
    If Len(strSuchtext) > 0 Then
        strFilter = "WHERE Artikelname LIKE '" & strSuchtext & "*'"
    Else
        strFilter = ""
    End If
    Me!lstSuchergebnis.RowSource = "SELECT ArtikelID, Artikelname FROM tblArtikel " & strFilter & " ORDER BY Artikelname"
End Sub
```

Listing 2: Verarbeiten des Vergleichswertes und Filtern der Listeneinträge

Das Kriterium soll also alle Datensätze ermitteln, deren Artikelname mit dem Buchstaben **A** beginnt. Das Kriterium landet anschließend in der Variablen **strFilter**. Ist **strSuchtext** leer, wird auch **strFilter** auf eine leere Zeichenfolge eingestellt. Der Inhalt von **strFilter** wird dann in die neu zusammengestellte Datensatzherkunft des Listenfeldes integriert. Diese sieht für unser Beispiel dann so aus:

```
SELECT ArtikelID, Artikelname FROM tblArtikel WHERE
Artikelname LIKE 'A*' ORDER BY Artikelname
```

Wenn wir das Formular **frmArtikelsucheKombi** geöffnet haben und den Suchbegriff eingeben, sieht das Ergebnis beispielsweise wie in Bild 4 aus. Hier werden wie gewünscht nur die Einträge angezeigt, die mit dem Buchstaben **A** beginnen.

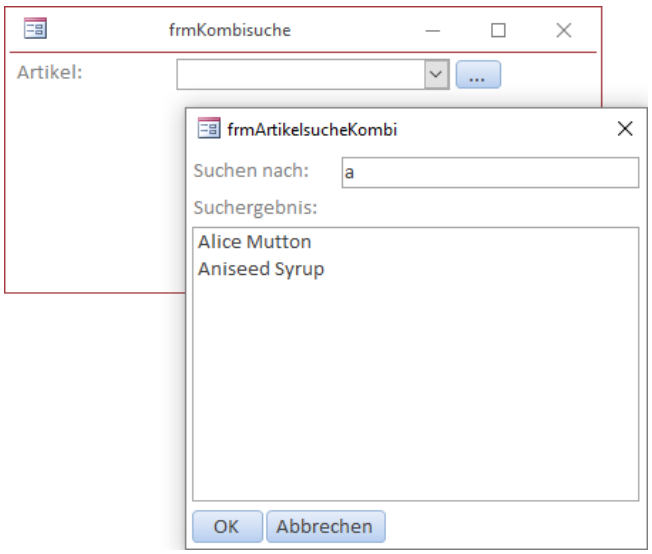

Bild 4: Eingabe eines Suchbegriffs

Ersten gefundenen Eintrag markieren

Nun soll direkt nach dem Einstellen des Filters der erste gefundene Eintrag im Listenfeld markiert werden, damit das aufrufende Formular einen gefundenen Eintrag vorfindet.

Dazu erweitern wir die Prozedur noch um die folgende Anweisung:

```
Private Sub txtSuche_Change()
    ...
    Me!lstSuchergebnis = Me!lstSuchergebnis.ItemData(0)
End Sub
```

Damit auch gleich beim Öffnen des Formulars der erste Eintrag im Listenfeld markiert ist, fügen wir dem Formular eine Ereignisprozedur hinzu, die durch das Ereignis **Beim Laden** ausgelöst wird und die wie folgt aussicht:

```
Private Sub Form_Load()
    Me!lstSuchergebnis = Me!lstSuchergebnis.ItemData(0)
End Sub
```

Wenn der Benutzer das Formular über die Schaltfläche **cmdOK** unsichtbar machen soll, müssen wir dafür die folgende Ereignisprozedur hinterlegen:

```
Private Sub cmdOK_Click()
    Me.Visible = False
End Sub
```

Die Ereignisprozedur, die durch die Schaltfläche **cmdAbbrechen** ausgelöst wird, soll das Formular hingegen direkt vollständig schließen:

```
Private Sub cmdAbbrechen_Click()
    DoCmd.Close acForm, Me.Name
End Sub
```

Dadurch erkennt die aufrufende Prozedur dann, dass der Benutzer die **Abbrechen**-Schaltfläche angeklickt hat und kein gewählter Eintrag übernommen werden soll.

Auswahl per Doppelklick übernehmen

Eine weitere praktische Möglichkeit, den gewählten Eintrag zu übernehmen, wäre ein Doppelklick auf diesen.

Dafür legen wir für das Ereignis **Beim Doppelklicken** des Listenfeldes die folgende Ereignisprozedur an:

```
Private Sub lstSuchergebnis_DblClick(Cancel As Integer)
    Me.Visible = False
End Sub
```

Auch dies blendet das Formular nur aus und sorgt dafür, dass die aufrufende Prozedur den aktuell markierten Eintrag auswählt und diesen in das Kombinationsfeld übernimmt.

```
Private Sub Kombisuche()
    DoCmd.OpenForm "frmArtikelsucheKombi", WindowMode:=acDialog
    If IstFormularGeoeffnet("frmArtikelsucheKombi") Then
        If Not IsNull(Forms!frmArtikelsucheKombi!lstSuchergebnis) Then
            Me!cboArtikel = Forms!frmArtikelsucheKombi!lstSuchergebnis
        End If
        DoCmd.Close acForm, "frmArtikelsucheKombi"
    End If
End Sub
```

Listing 3: Die Bedingung sorgt dafür, dass das Kombinationsfeld nicht geleert werden kann.

Damit kann nun nichs mehr schiefgehen – das Listenfeld führt das Doppelklick-Ereignis nicht aus, wenn der Benutzer auf einen leeren Bereich des Listenfeldes klickt, also kann dadurch kein Fehler auftauchen. Dies ist auch der Fall, wenn der Benutzer einen Suchbegriff eingegeben hat, der das Listenfeld komplett leert. Was dann hingegen noch geschehen kann, ist, dass der Benutzer einen Suchbegriff eingibt, der das Listenfeld komplett leert und dieser dann die **OK**-Schaltfläche betätigt.

In diesem Fall ist kein Eintrag markiert und dementsprechend übernimmt die Prozedur Kombisuche den Wert **Null** aus dem Listenfeld in das Kombinationsfeld. Das ist kein Problem, wenn vorher auch noch kein Eintrag im Kombinationsfeld ausgewählt war, aber wenn der Benutzer bereits einen Wert selektiert hat, möchte er diesen vermutlich nicht über die Suchen-Funktion löschen, sondern beibehalten. Also entschärfen wir die Situation, in dem wir der Prozeder Kombisuche noch eine Bedingung hinzufügen, welche dies prüft und in diesem Fall den Wert von **cboArtikel** nicht neu setzt (siehe Listing 3).

Zusammenfassung und Ausblick

Schön wäre nun noch, wenn das Formular zur Suche nach den Kombinationsfeldeinträgen auch unmittelber neben oder unter dem Kombinationsfeld erscheinen würde. Dazu müsste man die Position des öffnenden Formulars ermitteln und gegebenenfalls auch die des Kombinationsfeldes und dann das zu öffnende Formular nach dem Öffnen entsprechend positionieren. Dies würde jedoch den Rahmen dieses Artikels sprengen. Die grundlegenden Informationen dazu finden Sie jedoch im Artikel **Formularpositionen lesen und setzen** in Ausgabe 11/2016 von **Access [basics]**.

Davon abgesehen haben wir hier eine praktische Lösung geschaffen, mit der wir die Auswahl für Kombinationsfelder mit vielen Einträgen stark vereinfachen können. Es gibt noch weitere Erweiterungsmöglichkeiten, die wir weiter oben schon beschrieben haben. So könnten Sie dem Benutzer weitere Suchoptionen anbieten, damit dieser beispielsweise nicht nur nach Einträgen suchen kann, die mit dem eingegebenen Suchbegriff beginnen. Das würde allerdings auch die Einfachheit des Suchformulars zunichte machen, also verzichten wir an dieser Stelle darauf. Letztlich würden Sie dafür aber auch ohnehin nur eine Optionsgruppe benötigen, um die verschiedenen Suchoptionen wie etwa die folgenden zur Auswahl anzubieten:

- Suche nach Wortanfang
- Suche im ganzen Wort
- Nur exakte Treffer

Man könnte, für den Fall, dass zum Zeitpunkt des Aufrufs des Suchformulars bereits ein Eintrag im Kombinationsfeld vorliegt, diesen auch an das aufgerufene Formular übergeben und den vorhandenen Wert als Suchausdruck in das Suchfeld eingeben. Das ließe sich einfach durch das **OpenArgs**-Argument der **DoCmd.OpenForm**-Methode realisieren, das im aufgerufenen Formular mit der Eigenschaft **Me.OpenArgs** abgefragt werden kann.