Ausgabe 06/2023

ACCESS BASICS

DAS ACCESS-MAGAZIN FÜR ALLE,
DIE VON 0 AUF 100 WOLLEN

In diesem Heft:

André Minhorst

ACCESS BASICS

DAS ACCESS-MAGAZIN FÜR ALLE, DIE VON 0 AUF 100 WOLLEN

Ausgabe 06/2023

Terminplanung mit Access auf einem neuen Level

In unserem Shop gibt es ein tolles, neues Produkt: den **amvCalendar**!

Diese Lösung kannst Du problemlos in alle Access-Datenbanken integrieren. Du kannst damit Termine anlegen und diese in Kalendern in verschiedenen Ansichten wie Tag, Arbeitswoche, Woche oder Monat darstellen.

Dazu sind keine externen ActiveX-Komponenten oder DLLs nötig. Du brauchst nur die Objekte der Lösung in Deine Datenbank zu importieren und kannst den Kalender direkt anzeigen und damit arbeiten.

Der Kalender hat eine für die Anzahl der angezeigten Steuerelemente nicht für möglich zu haltende Performance.

Überzeuge Dich selbst und hole Dir die Demo von unserer Seite unter folgendem Link:

https://www.amvcalendar.de

Viel Spaß beim Ausprobieren!

André Minhorst

IMPRESSUM

ACCESS [BASICS] WIRD HERAUSGEGEBEN VON:
Minhorst und Minhorst GbR - André Minhorst Verlag
Borkhofer Straße 17
47137 Duisburg

REDAKTION:
André Minhorst (V.i.S.d.P)
Telefon: 0203/4495577
E-Mail: info@access-basics.de
Internet: www.access-basics.de

Geschäftsführung, Herstellung, Text- und Schlussredaktion, Layout von Magazin und Webseite: André Minhorst
Autor: André Minhorst
Fach- und Sprachlektorat: Carsten Gromberg

ISSN: 2190-8761

Tabellen [basics]: Währungsfelder

Währungsfelder sind im Grunde Zahlenfelder des Datentyps **Dezimal**. Nur, dass wir den Datentyp **Dezimal** über die Benutzeroberfläche nicht direkt angeboten bekommen. Warum aber verwenden wir überhaupt einen anderen Zahlendatentyp als beispielsweise Single oder Double, die auch Nachkommastellen bieten? Der Grund ist die Genauigkeit. Single und Double sind Gleitkommazahlen, die bei Rechenoperationen Rundungsfehler produzieren können. Währung beziehungsweise Dezimal nehmen Festkommazahlen auf, bei denen die Position des Kommas für das Feld festgelegt wird. Dieser Artikel zeigt, wie wir Währungsfelder nutzen und was wir noch damit machen können – außer Geldbeträge darin zu speichern.

Währungsfelder

Ein Währungsfeld verwenden wir beispielsweise für die Angabe von Preisen. Im Beispiel aus Bild 1 legen wir ein neues Feld namens **Einzelpreis** an und wählen für dieses den Felddatentyp **Währung** aus.

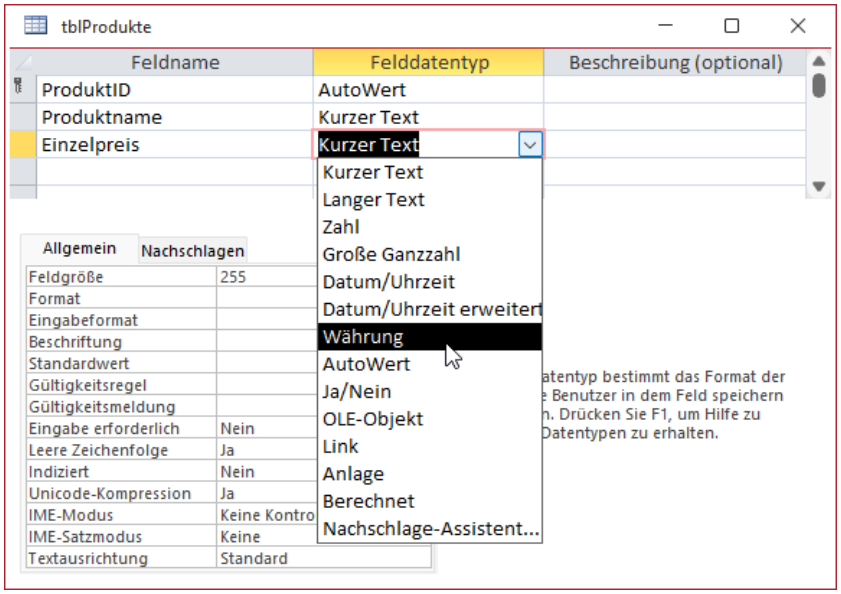

Bild 1: Anlegen eines Währungsfeldes

Danach finden wir in den Eigenschaften den Wert **Währung** für die Eigenschaft **Format** vor und **Automatisch** für die **Dezimalstellenanzeige** (siehe Bild 2)

Währung ist eines der eingebauten Formate in Access zu sein. Neben **Währung** gibt es auch noch das Format **Euro** (siehe Bild 3). Was ist genau der Unterschied zwischen diesen beiden Formatierungen?

Um dies zu ermitteln, legen wir neben dem Feld **Einzelpreis** noch ein Feld namens **EinzelpreisEuro** an und stellen für dieses die Eigenschaft **Format** auf **Euro** ein.

Wechseln wir in die Datenblattansicht der Tabelle und geben für beide Felder den gleichen Wert ein, sehen wir erst einmal keinen Unterschied (siehe Bild 4).

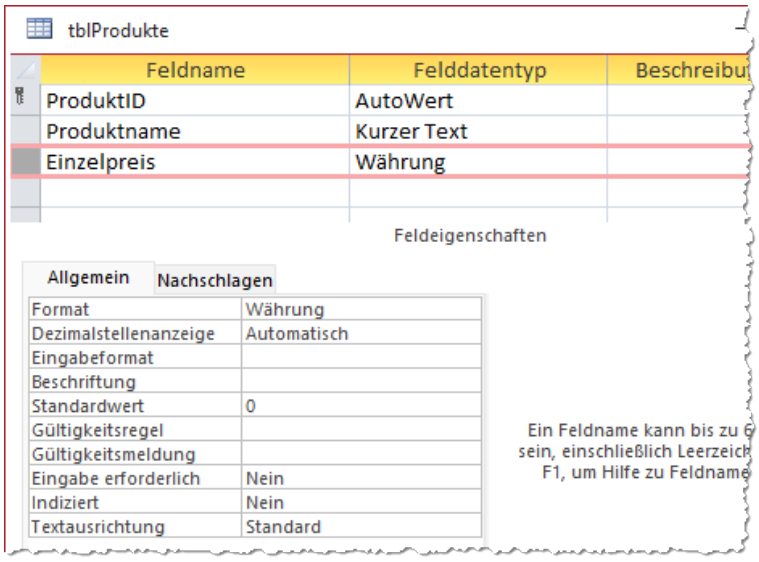

Bild 2: Eigenschaften eines Währungsfeldes

Dies ändert sich erst, wenn wir die Datenbank auf einem Rechner mit anderen regionalen Einstellungen öffnen – zum Beispiel würde auf einem System mit US-amerikanischen Einstellungen bei dem Wert **Währung** für die Eigenschaft **Format** die Währung **Dollar** ($) angezeigt werden.

Wenn Du also sicherstellen willst, dass die Währung immer in Euro angezeigt wird, wählst Du hier **Euro** statt allgemein **Währung**.

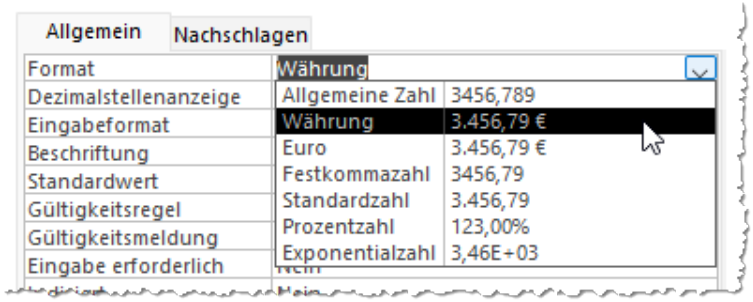

Bild 3: Neben **Währung** gibt es noch das Format **Euro**.

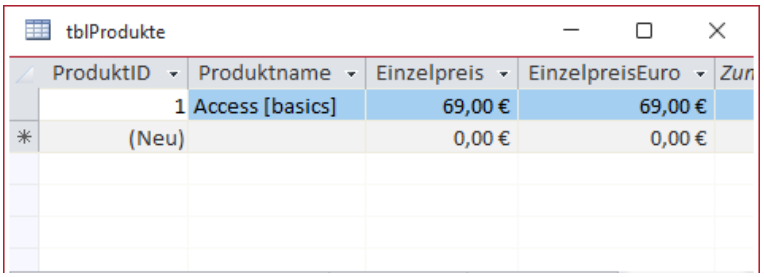

Bild 4: Währungsfeld in der Datenblattansicht

Einstellen der Dezimalstellenanzeige

Wie oben zu sehen, stellt Access für neue Währungsfelder standardmäßig den Wert **Automatisch** im Feld **Dezimalstellenanzeige** ein. Was bedeutet das? Dies bezieht sich allein auf die Anzeige der gespeicherten Daten. Im Fall des Wertes **Automatisch** werden zwei Stellen angezeigt. Aber woher bezieht Access diesen Wert?

Dazu werfen wir einen Blick in die Systemsteuerung (Windows-Suche, Eingabe **Systemsteuerung**). Hier klicken wir auf **Region** (siehe Bild 5).

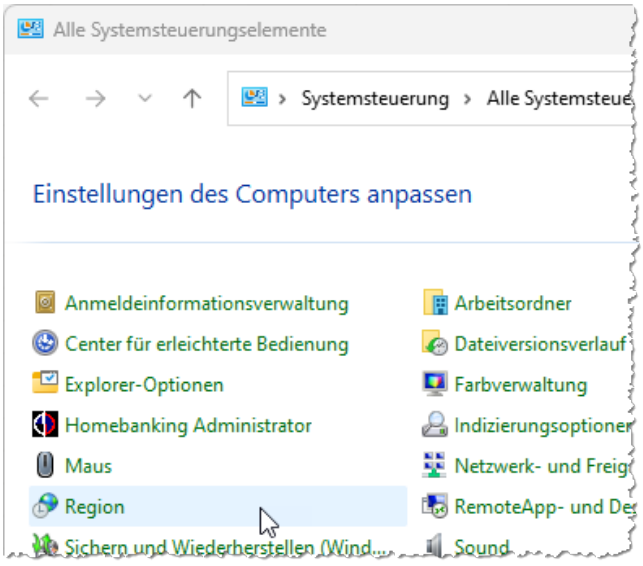

Bild 5: Öffnen der Regionseinstellungen der Systemsteuerung

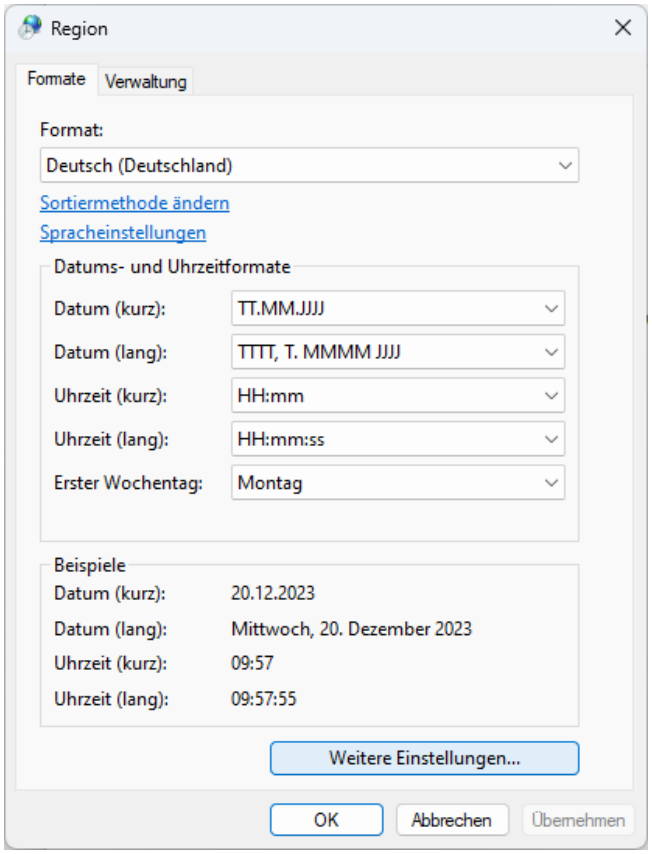

Bild 6: Region-Einstellungen der Systemsteuerung

Hier finden wir allerdings noch nicht die gewünschte Einstellung, sodass wir noch einen weiteren Klick auf **Weitere Einstellungen...** ausführen müssen (siehe Bild 6).

Damit landen wir im Dialog aus Bild 7. Hier wählen wir den Bereich Währung aus und finden oben das aktuelle Währungssymbol und weiter unten die Option **Anzahl der Dezimalstellen**.

Hier wird also vorgegeben, wie viele Dezimalstellen wir sehen, wenn der Wert für das Feld auf **Automatisch** eingestellt ist. Wenn wir diesen Wert beispielsweise auf **4** ändern, erhalten wir in der Datenblattansicht der Tabelle die Ansicht aus Bild 8.

Experimente mit der Währung

Wenn wir schon in diesem Dialog sind, ändern wir gleich mal das Währungssymbol auf Dollar ($) und klicken auf **Übernehmen**. Damit wir die Änderung in der Tabelle sehen, müssen wir Access vollständig schließen und erneut öffnen. Danach sollte die Tabelle im Feld **Einzelpreis** das gewünschte Währungszeichen anzeigen (siehe Bild 9).

In unserem Fall (Office 365) funktionierte dies aber nach einigen Experimenten nicht mehr wie gewünscht. Auf einmal blieb die Dollar-Währungsbezeichnung stehen, obwohl wir in der Systemsteuerung die Währung wieder auf **Euro** umgestellt hatten. Also schauten wir nochmal in das Format des Feldes **Einzelpreis** und entdeckten etwas Interessantes. Die Eigenschaft **Format** enthielt nämlich nun den folgenden Ausdruck:

```
#.##0,00 $;-#.##0,00 $
```

Damit können wir nun in der Systemsteuerung ändern, was wir wollen – als Währungszeichen wird das Dollar-Symbol

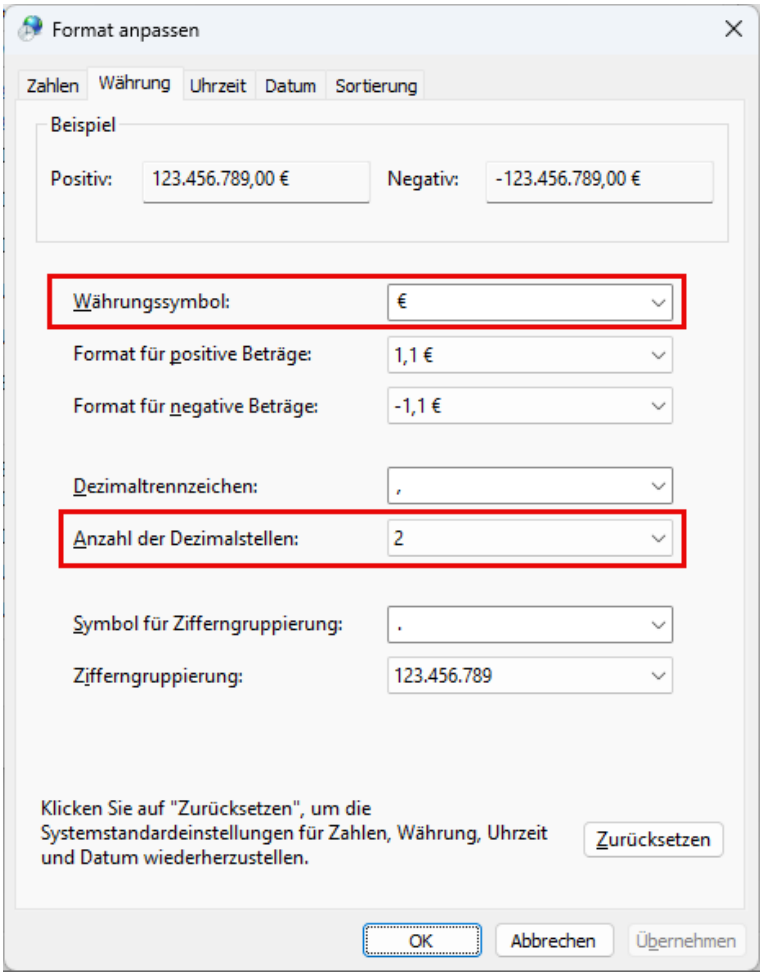

Bild 7: Währungseinstellungen

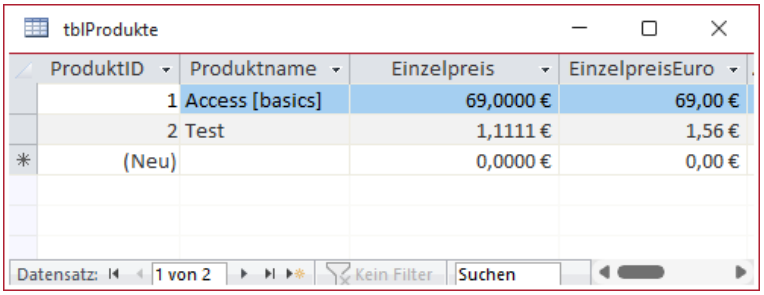

Bild 8: Währungsfeld mit vier Dezimalstellen aus der Systemsteuerung

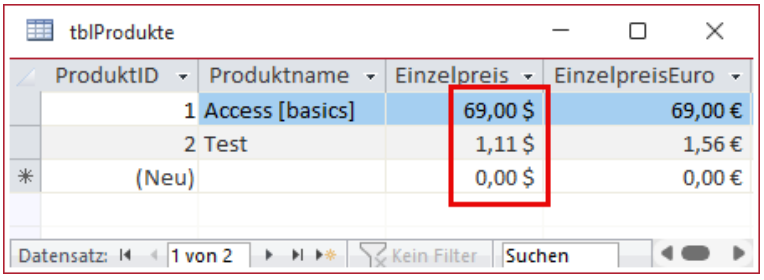

Bild 9: Währungsfeld mit dem Dollar-Zeichen

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst

Tabellen [basics]: Mehrwertige Felder

Wer mit Access arbeitet, kennt vermutlich die Nachschlagefelder, die das Auswählen von Daten aus einer per 1:n-Beziehung verknüpften Tabelle erleichtern. Es gibt aber noch eine weitere Möglichkeit unter Access, mit dem man in einem Feld mehrere Werte zur Auswahl anbieten kann. Solche Felder nennen wir mehrwertige Felder. Wie man ein solches anlegt, wie es funktioniert, warum man damit in Prinzip auch m:n-Beziehungen nachbilden kann und was die Vor- und Nachteile sind, erläutern wir in diesem Artikel.

Beispieldatenbank

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

Herkömmliche Feldlisten: Beispiel Anreden

Wir wollen zunächst an einem simplen Beispiel zeigen, wie man eine herkömmliche Feldliste anlegt: In diesem Fall ein Feld namens **Anrede**, das einen der Werte **Herr** oder Frau zur Auswahl anbietet. Um ein solches Feld als mehrwertiges Feld zu erstellen, gehen wir wie folgt vor:

- Wir legen eine neue Tabelle an und fügen die vier Felder **PersonID**, **Anrede**, **Vorname** und **Nachname** hinzu. Das Feld **Anrede** legen wir zunächst mit dem Felddatentyp **Kurzer Text** an.
- Nun wählen wir für das Feld **Anrede** den Felddatentyp **Nachschlage-Assistent...** aus (siehe Bild 1).
- Im ersten Schritt des Assistenten behalten wir nicht, wie üblich, die erste Option bei, sondern wählen die Option **Ich möchte selbst**

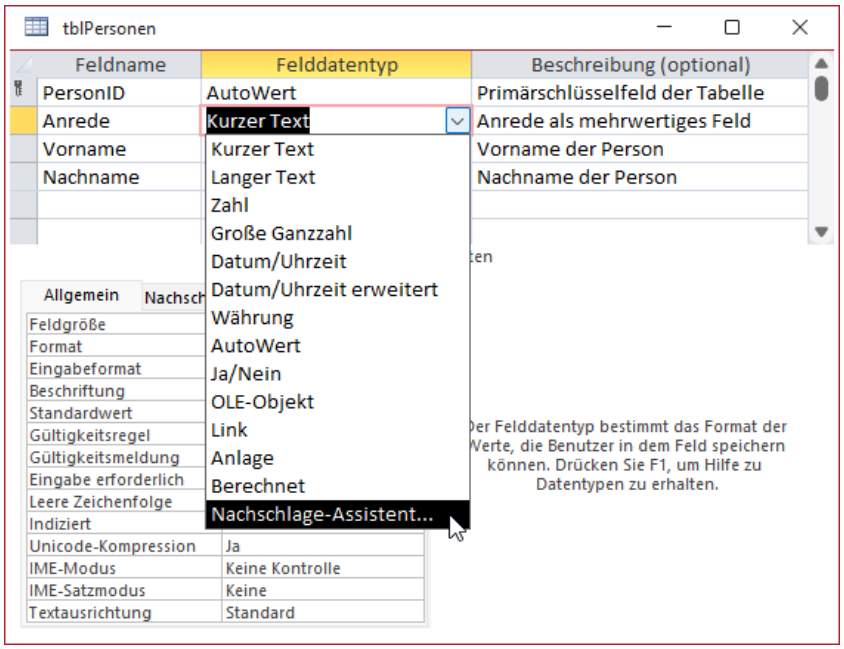

Bild 1: Die Tabelle **tblPersonen**

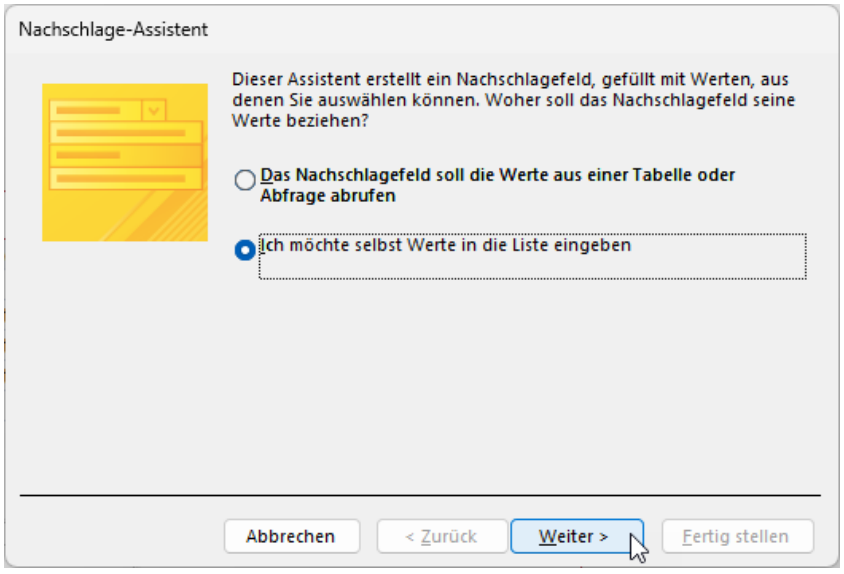

Bild 2: Erster Schritt des Nachschlage-Assistenten

Werte in die Liste eingeben aus (siehe Bild 2).

- Im nächsten Schritt geben wir dann direkt die Werte ein, die das Nachschlagefeld zur Auswahl bereitstellen soll – in diesem Fall **Herr** und **Frau** (siehe Bild 3). Hier können wir auch noch eine Anzahl von Spalten definieren, aber wir kommen in diesem Beispiel mit einer einzigen Spalte aus.

- Danach folgt bereits der letzte Schritt (siehe Bild 4). Hier legen wir die Beschriftung für das Nachschlagefeld fest, wobei wir die Bezeichnung **Anrede** beibehalten.

- Die Option **Nur Listeneinträge** gibt an, ob der Benutzer eigene Werte in die Liste eintragen können soll. Das wollen wir aus Beispielzwecken in diesem Fall ermöglichen.

- Die Option **Mehrere Werte zulassen** legt fest, ob wir nicht nur einen, sondern auch mehrere Werte auswählen können wollen. Bei der Anrede ist das nicht sinnvoll, deshalb wählen wir diese Option nicht aus.

Bevor wir uns die Dateneingabe mit diesem Feld ansehen, werfen wir einen Blick in die Eigenschaften des Feldes im Tabellenentwurf (siehe Bild 5). Hier wechseln wir zur Registerseite **Nachschlagen**. Es wurde ein Kombinationsfeld mit einer Wertliste als Datensatzherkunft definiert. Die Datensatzherkunft ist die Liste der

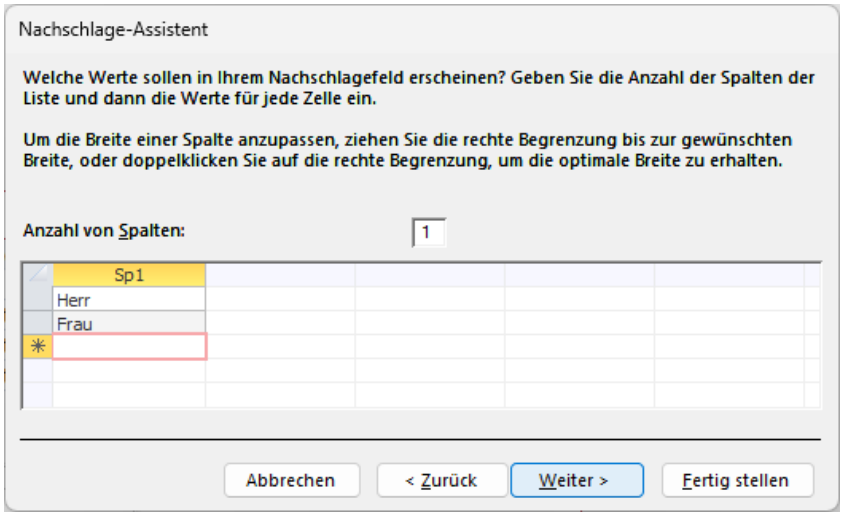

Bild 3: Eingeben der gewünschten Werte

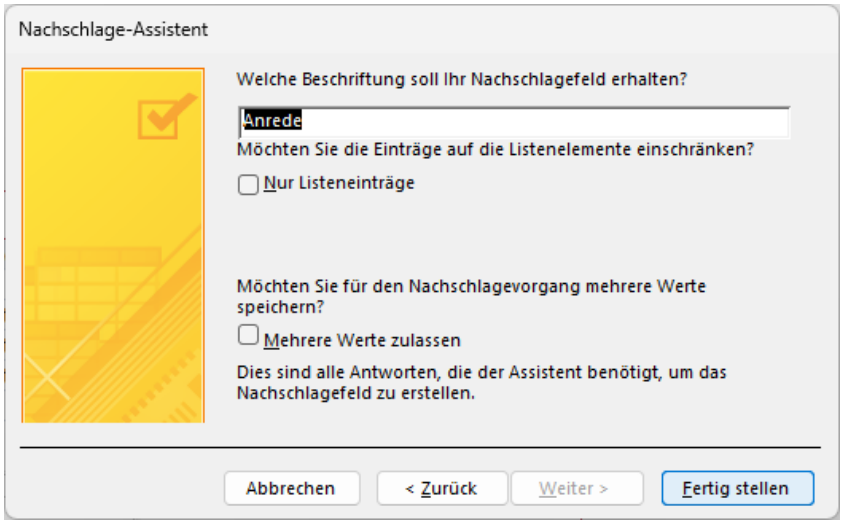

Bild 4: Letzter Schritt im Nachschlage-Assistent

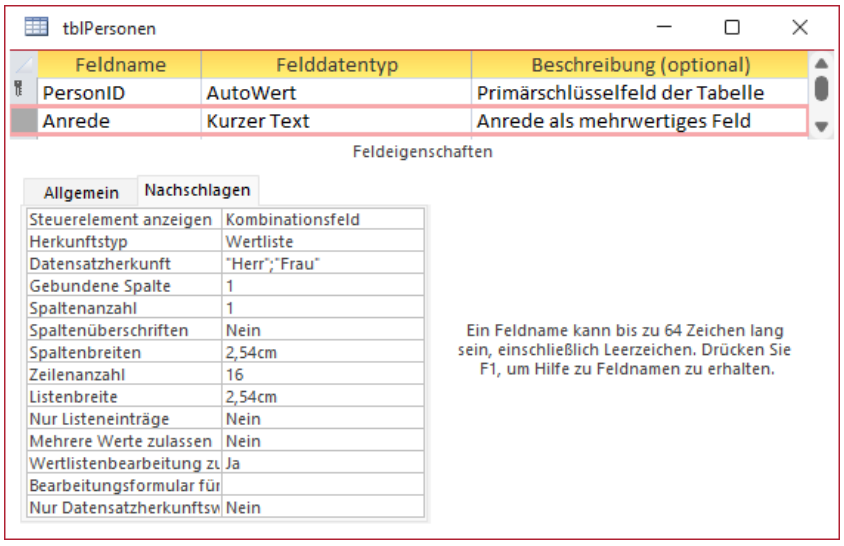

Bild 5: Entwurf des Feldes **Anrede**

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst

Abfragen [basics]: Mit Zeiträumen rechnen

Im Artikel **Eingebaute Funktionen: Rund um Datum und Uhrzeit** (**www.access-basics.de/632**) haben wir uns die eingebauten Funktionen von Access zum Arbeiten mit Datums- und Uhrzeitwerten beschäftigt. Im vorliegenden Artikel nutzen wir diese, um verschiedene Zeiträume zu ermitteln. Tage, Wochen, Monate zwischen zwei Datumsangaben, das Gleiche für Stunden, Minuten und Sekunden, Berechnen des Alters von Personen, Tage seit und bis zu bestimmten Daten und vieles mehr untersuchen wir in diesem Artikel.

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **AbfragenBasics_MitZeitraeumenRechnen.accdb**.

Beispieltabellen

Im Rahmen der Beispiele zu diesem Artikel werden wir mit verschiedenen Tabellen arbeiten. Für die Beispiele, in denen es sich um das Berechnen von Abständen zwischen zwei Datumsangaben dreht, nutzen wir die Tabelle **tblAbwesenheiten**. Hier können wir mit den beiden Feldern **AbwesenheitVon** und **AbwesenheitBis** arbeiten, um verschiedenste Zeiträume zu berechnen. Die Tabelle enthält bereits einige Beispieldatensätze (siehe Bild 1).

Für andere Zwecke nutzen wir die Tabelle **tblGeburtstage**.

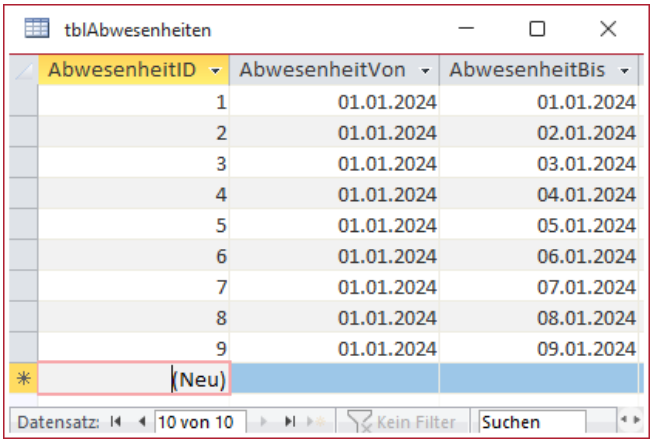

Bild 1: Die Tabelle **tblAbwesenheiten**

Anzahl Tage zwischen zwei Datumsangaben

Für die Berechnung von Abwesenheiten wie Fehlzeiten können wir die Anzahl der Tage zwischen zwei Datumsangaben mit der **DatDiff**-Funktion ermitteln.

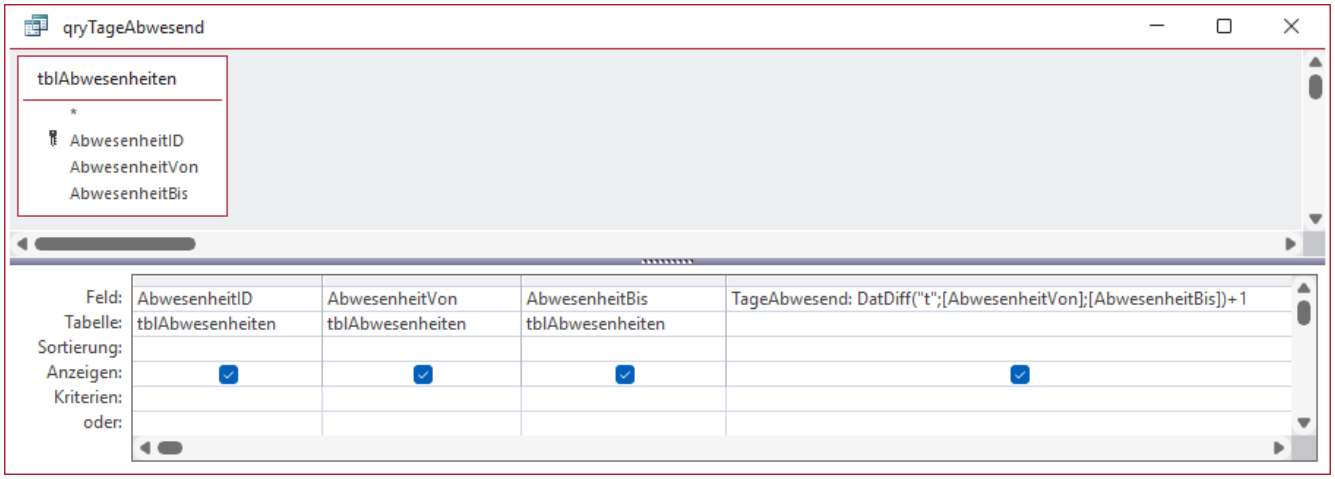

Bild 2: Die Abfrage **qryTageAbwesend**

Dazu nutzen wir den folgenden Ausdruck:

```
TageAbwesend: DatDiff("t";[AbwesenheitVon];[AbwesenheitBis]) + 1
```

Diesen fügen wir in eine Abfrage ein, die auf der Tabelle **tblAbwesenheiten** basiert und alle Felder dieser Tabelle enthält. Der Entwurf dieser Abfrage ist in Bild 2 dargestellt.

Warum verwenden wir nicht einfach nur **DatDiff**, sondern addieren noch den Wert **1** hinzu? Weil **DatDiff** die Differenz liefert und nicht beide Tage mit in den Abwesenheitszeitraum einschließt.

Das Ergebnis liefert die erwarteten Werte (siehe Bild 3).

qryTageAbwesend

AbwesenheitID	AbwesenheitVon	AbwesenheitBis	TageAbwesend
1	01.01.2024	01.01.2024	1
2	01.01.2024	02.01.2024	2
3	01.01.2024	03.01.2024	3
4	01.01.2024	04.01.2024	4
5	01.01.2024	05.01.2024	5
6	01.01.2024	06.01.2024	6
7	01.01.2024	07.01.2024	7
8	01.01.2024	08.01.2024	8
9	01.01.2024	09.01.2024	9
(Neu)			

Datensatz: 1 von 9 | Kein Filter | Suchen

Bild 3: Die Abfrage **qryTageAbwesend** in der Datenblattansicht

Anzahl Tage zwischen zwei Datumsangaben ohne Wochenenden

Wenn man Abwesenheiten berechnet, sollen dabei oft nur die Tage von Montag bis Freitag berücksichtigt werden, nicht die Wochenenden. Die vorliegende Abfrage kümmert sich nicht um Wochentage.

Wie können wir diese erweitern, damit sie nur die Tage von Montag bis Freitag berücksichtigt?

Der dazu benötigte Ausdruck ist wesentlich komplexer als der vorherige und er sieht so aus:

```
AbwesenheitOhneWochenenden: DatDiff("t";[AbwesenheitVon];[AbwesenheitBis])-DatDiff("ww";[AbwesenheitVon];[AbwesenheitBis])*2+1+(Wochentag([AbwesenheitVon])=1)+(Wochentag([AbwesenheitBis])=7)
```

Das nehmen wir nun einmal auseinander:

- **DatDiff("t";[AbwesenheitVon];[AbwesenheitBis])**: Berechnet die Differenz in Minuten zwischen dem Abwesenheitsbeginn (**AbwesenheitVon**) und dem Abwesenheitsende (**AbwesenheitBis**).
- **DatDiff("ww";[AbwesenheitVon];[AbwesenheitBis])*2**: Berechnet die Anzahl der Wochen zwischen dem Abwesenheitsbeginn und dem Abwesenheitsende und multipliziert sie mit **2**, um die Anzahl der Wochenendtage zu erhalten.
- **+(Wochentag([AbwesenheitVon])=1)**: Addiert **1**, wenn der Wochentag des Abwesenheitsbeginns ein Sonntag ist (Wochentag **1** entspricht **Sonntag**).
- **+(Wochentag([AbwesenheitBis])=7)**: Addiert **1**, wenn der Wochentag des Abwesenheitsendes ein Samstag ist (Wochentag **7** entspricht **Samstag**).
- **+1**: Addiert **1**, um den ersten Tag der Abwesenheit zu berücksichtigen.

Den Ausdruck geben wir wieder in einer ähnlichen Abfrage wie der vorherigen ein. Um Ausdrücke wie diesen etwas komfortabler zu bearbeiten, als dies in der einzeiligen Darstellung im Entwurfsraster möglich ist, markieren wir den Text und betätigen die Tastenkombination **Umschalt + F2**.

Danach können wir den Text wie in Bild 4 bearbeiten.

Wechseln wir anschließend in die Datenblattansicht, finden wir die korrekte Anzahl der Abwesenheitstage ohne Wochenenden vor (siehe Bild 5).

Anzahl Wochen zwischen zwei Datumsangaben

Auf die gleiche Weise wie beim ersten Beispiel können wir auch die Anzahl der Wochen zwischen zwei Datumsangaben berechnen. Dazu verwenden wir wieder die **DatDiff**-Funktion, diesmal mit der Einheit **ww**. Klar ist, dass wir erst dann den Wert 1 erhalten, wenn die Differenz sieben Tage oder mehr aufweist:

```
AnzahlWochen: DatDiff("ww";[AbwesenheitVon];[AbwesenheitBis])
```

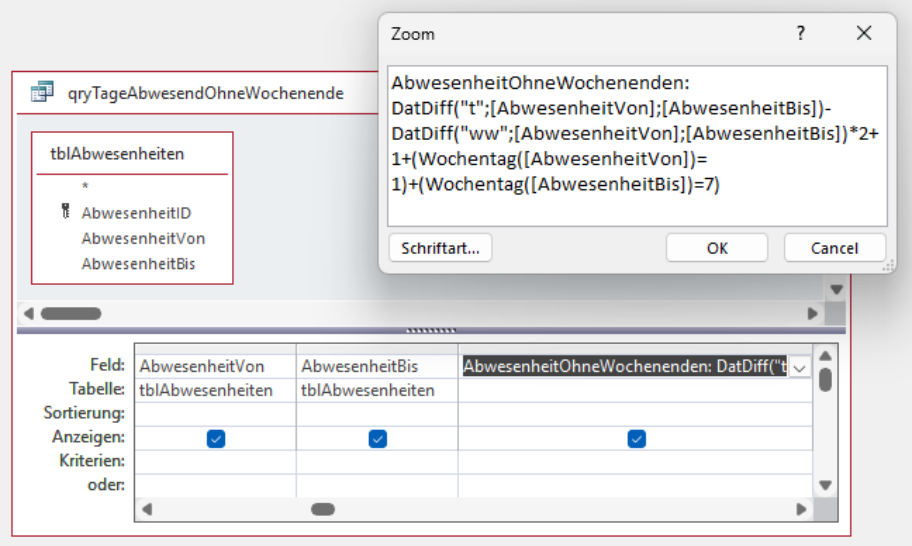

Bild 4: Die Abfrage **qryTageAbwesendOhneWochenende**

Das Berechnen der vollständigen Wochen ist dabei relativ einfach. Was aber, wenn wir neben den Wochen den Anteil der Tage ermitteln, die keine weitere volle Woche ausmachen? Dazu verwenden wir einen Ausdruck wie den Folgenden:

```
AnzahlWochenDezimal: DatDiff("ww";[AbwesenheitVon];[AbwesenheitBis])+(((DatDiff("t";[AbwesenheitVon];[AbwesenheitBis])+1) Mod 7)/7)
```

qryTageAbwesendOhneWochenende

AbwesenheitID	AbwesenheitVon	AbwesenheitBis	Abwesenhei
1	01.01.2024	01.01.2024	1
2	01.01.2024	02.01.2024	2
3	01.01.2024	03.01.2024	3
4	01.01.2024	04.01.2024	4
5	01.01.2024	05.01.2024	5
6	01.01.2024	06.01.2024	5
7	01.01.2024	07.01.2024	5
8	01.01.2024	08.01.2024	6
9	01.01.2024	09.01.2024	7
(Neu)			

Datensatz: 1 von 9 Kein Filter Suchen

Bild 5: Die Abfrage **qryTageAbwesendOhneWochenende** in der Datenblattansicht

Auch dieser Ausdruck bedarf einiger Erläuterungen:

- **DatDiff("ww";[AbwesenheitVon];[AbwesenheitBis])**: Dieser Teil des Ausdrucks berechnet die ganze Anzahl der Wochen zwischen dem Startdatum (**AbwesenheitVon**) und dem Enddatum (**AbwesenheitBis**).
- **DatDiff("t";[AbwesenheitVon];[AbwesenheitBis])+1**: Berechnet die Gesamtdifferenz in Tagen zwischen dem Start- und

qryWochenAbwesend

AbwesenheitVon	AbwesenheitBis	AnzahlWochen	AnzahlWochenDezimal
01.01.2024	01.01.2024	0	0,14
01.01.2024	02.01.2024	0	0,29
01.01.2024	03.01.2024	0	0,43
01.01.2024	04.01.2024	0	0,57
01.01.2024	05.01.2024	0	0,71
01.01.2024	06.01.2024	0	0,86
01.01.2024	07.01.2024	1	1,00
01.01.2024	08.01.2024	1	1,14
01.01.2024	09.01.2024	1	1,29

Datensatz: 1 von 9 Kein Filter Suchen

Bild 6: Die Abfrage **qryWochenAbwesend** in der Entwurfsansicht

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst

Handyverwaltung VIII: 64-Bit-fähig machen

Als mich neulich ein Kunde fragte, ob es nicht eine 64-Bit-Version der Handyverwaltung aus unserer Artikelreihe von 2017/2018 gäbe, habe ich mir angeschaut, wie groß der Aufwand ist, um diese Anwendung 64-Bit-kompatibel zu machen. Der Aufwand war nicht so groß, aber beim Umbau ist mir aufgefallen, dass noch ein paar Dinge fehlen. Diese reichen wir der Vollständigkeit halber in weiteren Artikeln nach. In diesem Artikel starten wir mit den Schritten, die nötig sind, um die Anwendung 64-Bit-fähig zu machen.

Beispieldatenbank
Die Beispiele dieses Artikels findest Du in der Datenbank **Handyverwaltung.accdb**.

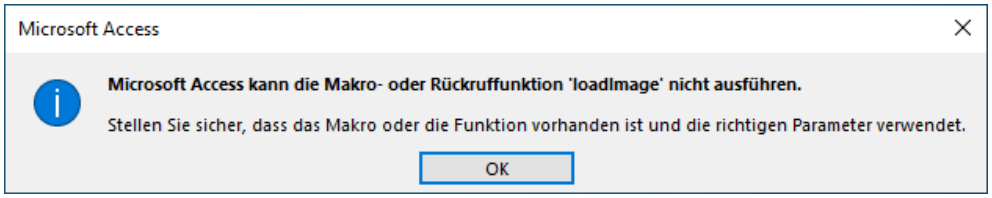

Bild 1: Fehler beim Öffnen der Handyverwaltung mit 64-Bit-Access

Fehler beim Aufruf mit 64-Bit-Access

Als Erstes schauen wir uns an, woran Du erkennen kannst, dass Du die 32-Bit-Version der Handyverwaltung mit 64-Bit-Access öffnen möchtest. Hier sollte zuerst der Fehler aus Bild 1 auftreten. Dieser Fehler tritt mehrmals auf – genau genommen für jedes Bild, das im Ribbon angezeigt werden soll.

Anschließend erscheint zwar das Ribbon der Anwendung, aber zeigt keine Bilder an (siehe Bild 2). Wir sehen also gleich, dass wir irgendein Problem mit den Bildern haben.

Das ist die Ausgangssituation für diesen Artikel – wir werden uns ansehen, wir wie die Anwendung auch unter 64-Bit-Access nutzbar machen und in weiteren Artikeln führen noch einige weitere Optimierungen aus.

Handyverwaltung 64-Bit-fähig machen

Wenn wir eine Datenbank programmieren, ist diese üblicherweise für 32-Bit- und für 64-Bit-Access nutzbar. Es gibt jedoch einige Dinge, die nicht zu den Bordmitteln gehören, die hier einen Unterschied ausmachen können.

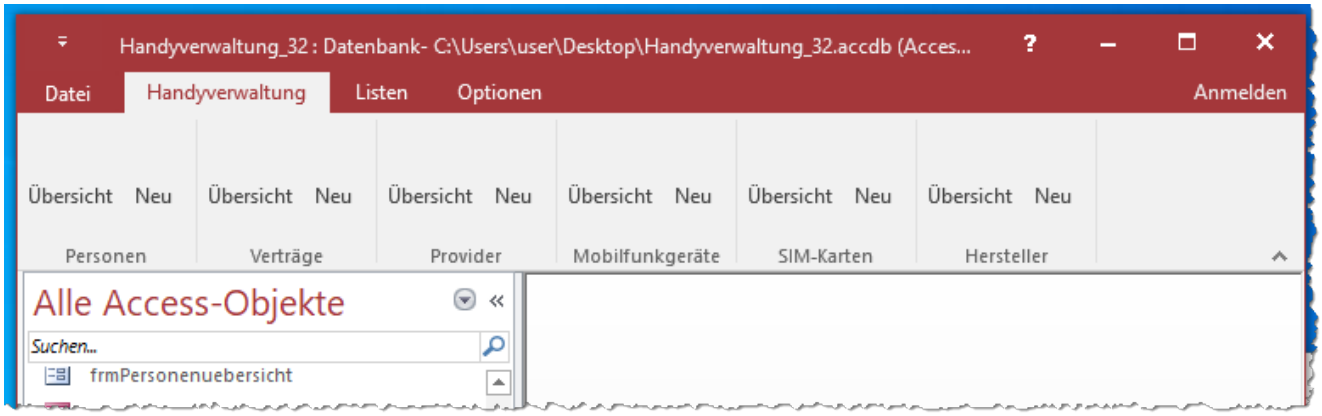

Bild 2: Das Ribbon zeigt keine Bilder an.

Dazu gehören bestimmte Steuerelemente, die nur unter 32-Bit-Office funktionieren und vor allem API-Befehle. API-Befehle sind Befehle, mit denen wir direkt auf Systemfunktionen von Windows zugreifen können. Diese deklarieren wir in einem VBA-Modul und können diese dann aus Prozeduren und Funktionen heraus aufrufen.

Wo aber haben wir in der Handyverwaltung mit solchen Elementen gearbeitet? Wissentlich haben wir diese gar nicht eingesetzt. Wir haben jedoch ein Modul namens **mdlRibbonImages** hinzugefügt, das einige Funktionen enthält, mit denen wir die Bilddateien aus der Tabelle **MSysResources** im Ribbon anzeigen können.

Das diese zumindest Teil des Problems sind, können wir schnell ermitteln. Wenn der Aufruf der Callbackfunktionen des Ribbons nicht funktioniert, lohnt es sich immer, einen Blick in den VBA-Code zu werfen – genau genommen: diesen zu kompiliere und zu schauen, ob dabei Kompilierfehler auftreten.

VBA-Code kompilieren

Das gelingt wie folgt: Wir wechseln mit der Tastenkombination **Alt + F11** oder **Strg + G** zum VBA-Editor von Access. Hier führen wir den Menübefehl **Debuggen|Kompilieren von Database** aus (**Database** ist der aktuelle Name des VBA-Projekts, den wir noch nicht geändert haben).

Dies ruft die Fehlermeldung aus Bild 3 hervor. Hier finden wir genau den entscheidenden Punkt – Teile des Codes sind noch nicht für 64-Bit vorbereitet. Es gibt bereits einen Hinweis – wir müssen die **Declare**-Anweisungen des Projekts um das Schlüsselwort **PtrSafe** ergänzen. Außerdem muss man bei einigen API-Funktionen die **Long**-Parameter und -Rückgabewerte in **LongPtr** umwandeln. Keine Angst, das machen wir nicht im Detail – diese Arbeit haben wir

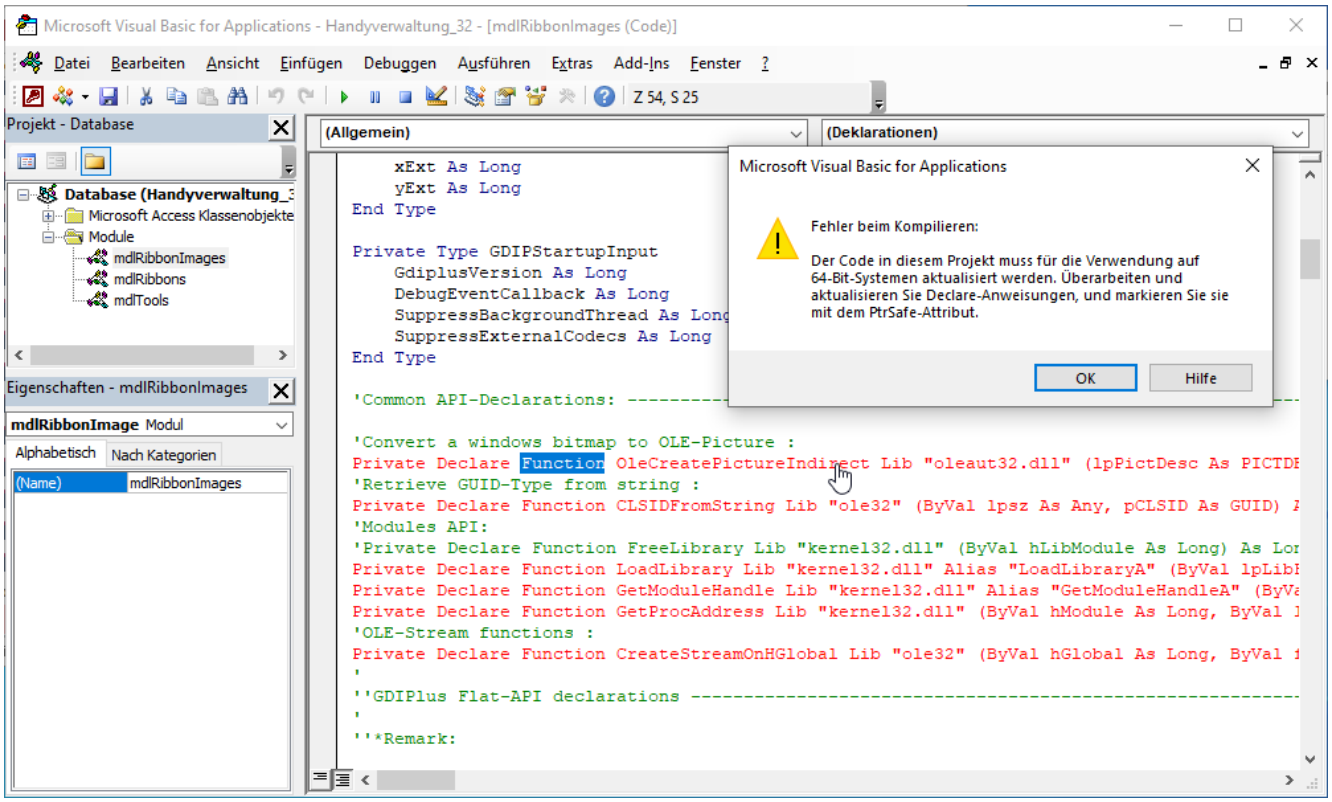

Bild 3: Kompilierfehler in den API-Funktionen

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst

Handyverwaltung IX: Fehlende Formulare

Bei der Durchsicht unserer Lösung aus dem Jahr 2017/2018 ist aufgefallen, dass noch ein Formular fehlt beziehungsweise der Ribbon-Eintrag Verträge|Übersicht auf das falsche Formular verweist. Das fehlende Formular zur Anzeige der Übersicht der Verträge der Handyverwaltung fügen wir in diesem Artikel hinzu. Außerdem passen wir die Ribbon-Programmierung so an, dass dieses Formular beim Anklicken des entsprechenden Befehls geöffnet wird.

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **Handyverwaltung_IX_Fehlende-Formulare.accdb**.

Fehlende Elemente in der Lösung

Die längst abgeschlossene Lösung zum Thema Handyverwaltung aus dem Jahr 2017/2018 habe ich noch einmal herausgeholt, weil ein Leser sich wünschte, dass er diese unter Access in der 64-Bit-Version einsetzen kann. Diesen Wunsch konnte ich leicht erfüllen, es musste nur das Modul **mdlRibbonImages** gegen eine neuere Version ausgetauscht werden – siehe Artikel **Handyverwaltung VIII: 64-Bit-fähig machen** (**www.access-basics.de/634**).

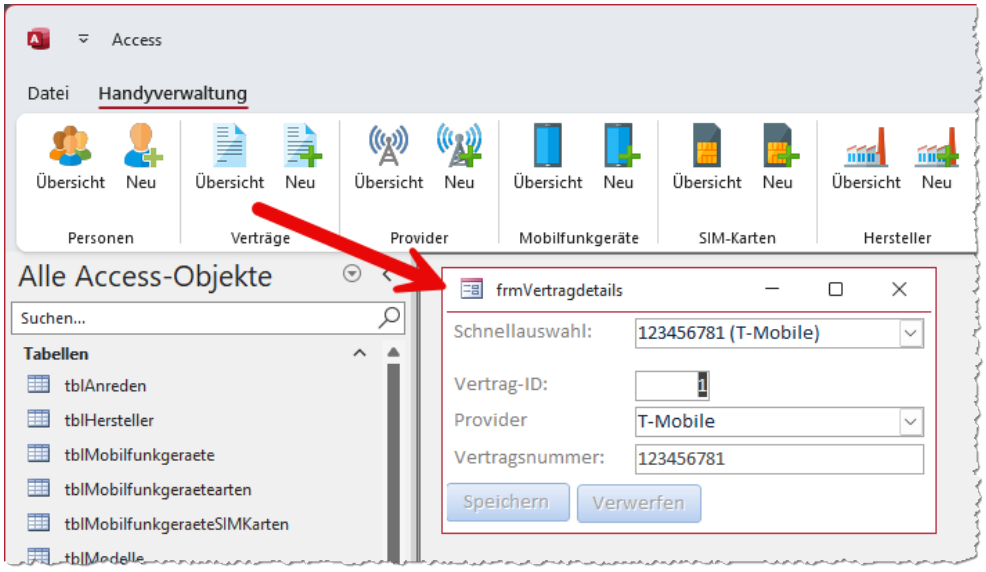

Bild 1: Das ist das falsche Formular.

Dabei ist allerdings aufgefallen, dass der Ribbon-Button **Verträge|Übersicht** beim Anklicken genau das gleiche Formular geöffnet hat wie der Befehl **Verträge|Neu** (siehe Bild 1). Das Formular, das diese Schaltfläche eigentlich öffnen sollte, war sogar noch nicht einmal vorhanden. Um die Lösung zu vervollständigen, wollen wir das Formular noch hinzufügen und die Ribbon-Programmierung anpassen. Das Formular wollen wir ähnlich gestalten wir die Formulare **frmPersonenuebersicht** und **frmMobilfunkgeraeteUebersicht**.

Da wir schon gezeigt haben, wie man solche Formulare von Grund auf anlegt, wollen wir diesmal einen alternativen Weg einschlagen: Wir kopieren das Haupt- und Unterformular zur Darstellung der Übersicht der Personen und passen diese so an, dass sie anschließend die Verwaltung der Verträge erlauben.

Datenbank zum Entwickeln öffnen

Für die Datenbank ist das Ribbon namens **Main** in den Access-Optionen unter **Name des Menübands** eingestellt. Diese Ribbondefinition blendet alle eingebauten Elemente des Ribbons aus, was es kompliziert macht, die Elemente der Datenbank zu entwerfen und zu programmieren. Damit die eingebauten Ribbonelemente nicht ausgeblendet werden, öffnen wir die Anwendung bei gedrückter **Umschalt**-Taste. Auf diese Weise wird die Ribbondefinition nicht

angewendet und es erscheinen die zur Entwicklung notwendigen Ribbonelemente.

Formulare kopieren

Der erste Schritt zum Erstellen der neuen Formulare ist das Kopieren der benötigten Formulare. Die Anzeige der Übersicht der Personen erfolgt mit dem Hauptformular **frmPersonenUebersicht** und **sfmPersonenUebersicht**. Diese beiden kopieren wir wie folgt:

- Markieren des Formulars **frmPersonenUebersicht** im Navigationsbereich
- Betätigen der Tastenkombination **Strg + C** zum Kopieren in die Zwischenablage
- Bestätigen der Tastenkombination **Strg + V** zum Einfügen aus der Zwischenablage

Es erscheint der Dialog aus Bild 2, wo wir den Namen des neuen Formulars eingeben. Dieser soll **frmVertraegeuebersicht** heißen.

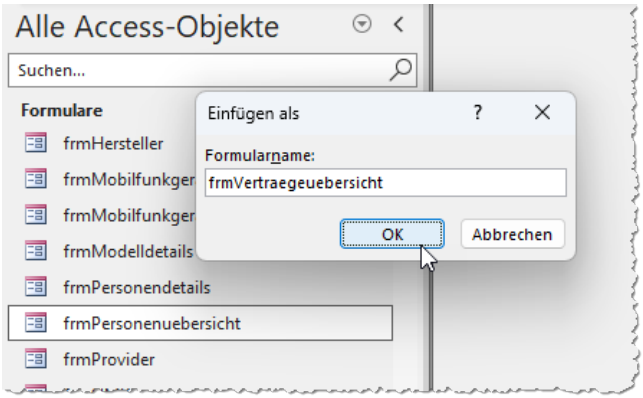

Bild 2: Einfügen eines kopieren Formulars unter einem neuen Namen

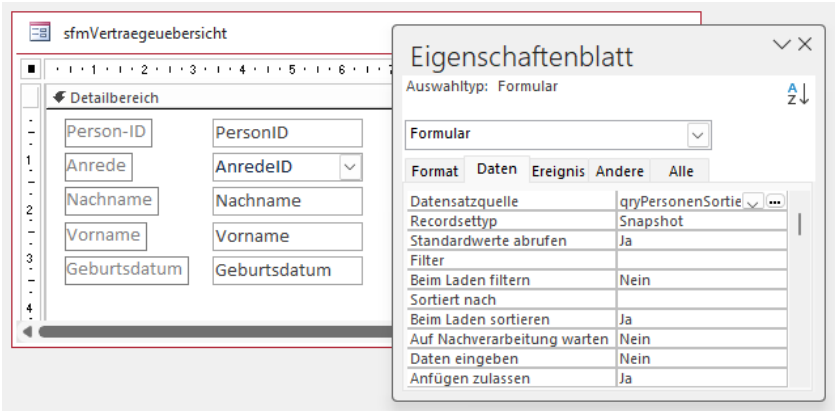

Bild 3: Anpassen des Unterformulars

Auf die gleiche Weise verfahren wir mit dem Formular **sfmVertraegeuebersicht**.

Unterformular anpassen

Als Erstes passen wir das Unterformular an. Dazu öffnen wir das soeben kopierte Formular **sfmVertraegeuebersicht** in der Entwurfsansicht. Hier stellen wir fest, dass die Datensatzquelle auf die Abfrage **qryPersonenSortiert** eingestellt ist und das Formular auch die darin enthaltenen Felder anzeigt (siehe Bild 3).

Datenherkunft des Unterformulars erstellen

Also ändern wir zuerst die Datensatzquelle. Dazu erstellen wir eine neue Abfrage namens **qryVertraegeSortiert**. Hier stellt sich nur noch die Frage, wonach wir initial sortieren wollen – nach dem Provider oder der Vertragsnummer? Vermutlich kennt man eher den Provider als die Vertragsnummer, also würde man eher nach dem Provider suchen und dann nach der Vertragsnummer. Also sortieren wir nach dem Provider.

Wir legen also eine neue Abfrage in der Entwurfsansicht an und ziehen die Tabelle **tblVertraege** aus dem Navigationsbereich in den Abfrageentwurf. Dann ziehen wir alle Felder aus der Feldliste in das Entwurfsraster. Schließlich fügen wir noch eine aufsteigende Sortierung nach dem Feld **ProviderID** hinzu (siehe Bild 4).

Schauen wir uns das Ergebnis an, sehen wir, dass wir noch einmal nacharbeiten müssen. Dieses wird nach dem tatsächlich im Feld **ProviderID** gespeicherten

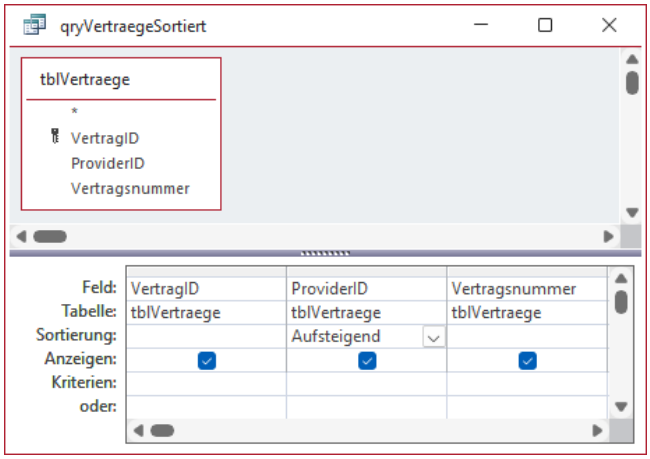

Bild 4: Abfrage sortiert nach dem Feld **ProviderID** im Entwurf

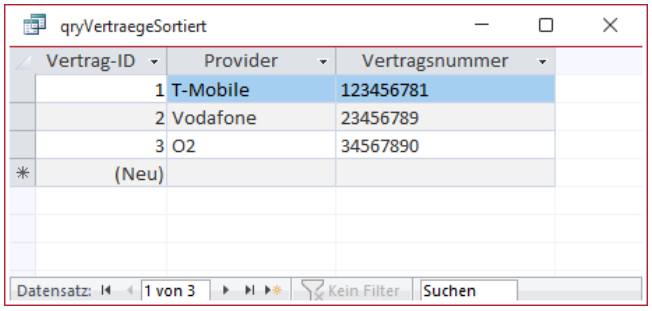

Bild 5: Abfrage sortiert nach dem Provider

Wert sortiert und nicht nach dem angezeigten Providernamen aus der Tabelle **tblProvider** (siehe Bild 5). Um tatsächlich eine Sortierung nach dem Namen des Providers zu erhalten, müssen wir noch die Tabelle **tblProvider** in die Abfrage aufnehmen und dieses Feld anstelle des Feldes **ProviderID** angeben.

Der Entwurf sieht nun wie in Bild 6 aus. Da das Feld mit dem Namen des Providers Bezeichnung lautet, stellen wir für dieses noch die Eigenschaft **Beschriftung** auf den Wert **Provider** ein. Auf diese Weise erscheint diese Bezeichnung als Spaltenüberschrift für dieses Feld.

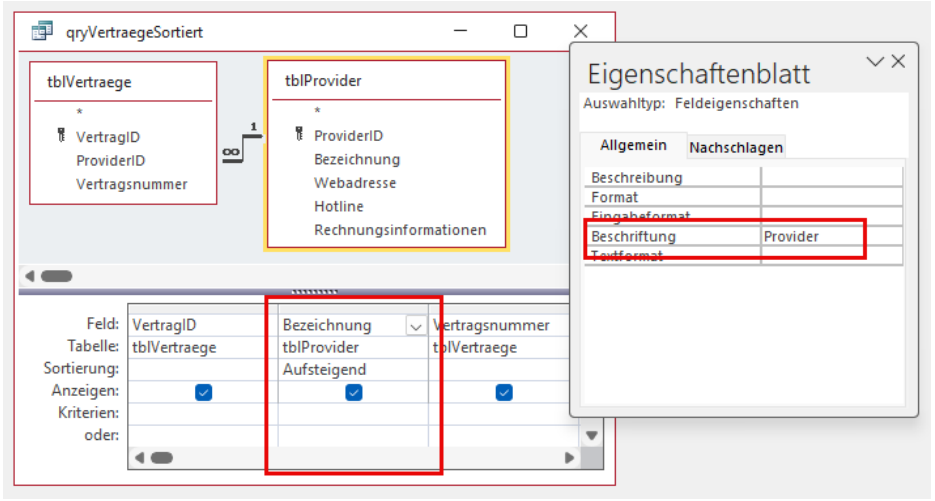

Bild 6: Abfrage sortiert nach dem Provider im Entwurf

Wechseln wir erneut in die Datenblattansicht, finden wir nun die korrekte Sortierung vor (siehe Bild 7).

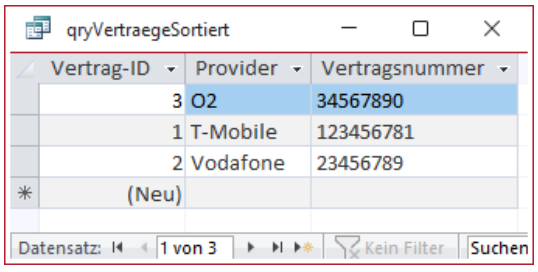

Bild 7: Sortierung nach der Providerbezeichnung

Damit können wir uns wieder dem Unterformular **sfmVertraegeuebersicht** zuwenden. Hier stellen wir nun zunächst die Eigenschaft Datensatzquelle auf die neu erstellte Abfrage **qryVertraegeSortiert** ein.

Da wir die somit die alte Datenquelle entfernt haben, fehlen nun die Felder dieser Datenquelle als Steuerelementinhalt der vorhandenen Textfelder (siehe Bild 8).

Da die Feldbeschriftungen auch nicht mehr stimmen, löschen wir einfach alle Felder und ziehen diese aus der Feldliste erneut in den Entwurf des Formulars (siehe Bild 9). Hier ist gut zu erkennen, dass für das Feld **Bezeichnung** die Beschrif-

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst

Handyverwaltung X: Titel und Icons

Die Handyverwaltung ist nun fast vollständig. Was allerdings noch fehlt, sind einige optische Elemente. Zum Beispiel gibt es noch kein Anwendungsicon und auch keinen Anwendungstitel. Und auch die einzelnen Formulare könnten wir noch mit Icons ausstatten – und die Anzeige des Formularnamens in der Titelleiste ist auch nicht gerade sexy. Also kümmern wir uns noch um diese Verfeinerungen und schauen uns an, wie uns die Anwendung anschließend gefällt.

Beispieldatenbank
Die Beispiele dieses Artikels findest Du in der Datenbank **Handyverwaltung_X_TitelUndIcons.accdb**.

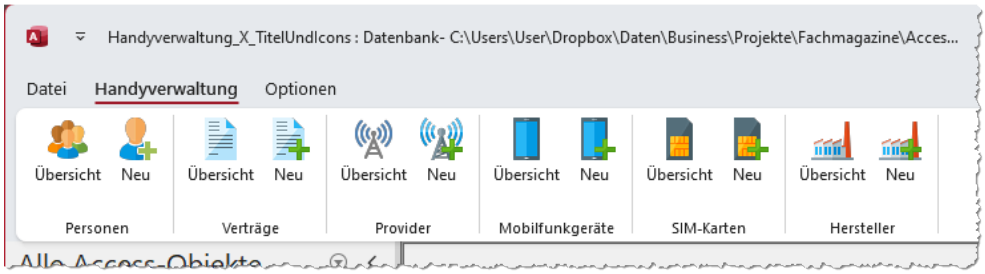

Bild 1: Aussehen des Titelbereichs der Anwendung

Anwendungstitel festlegen und Icon hinzufügen
Schauen wir uns an, wie der obere Teil unserer Anwendung aussieht, können wir zumindest mit dem Ribbon schon einmal zufrieden sein (siehe Bild 1).

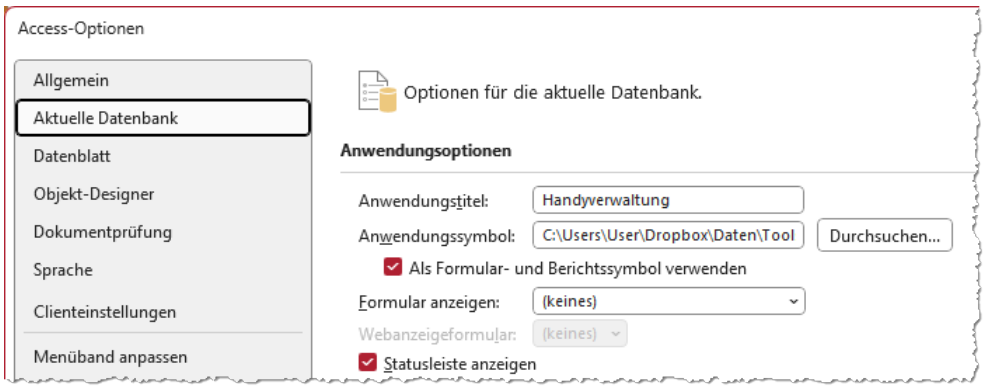

Bild 2: Einstellungen für die aktuelle Datenbank

Hübsche Icons, ordentliche Beschriftungen – was will man mehr. Darüber sieht es allerdings wüst aus. Erstens wollen wir nicht das Access-Icon sehen, und der längliche Titeltext gefällt uns auch nicht wirklich.

Also steigen wir in die Access-Optionen ein, die wir mit einem Klick auf den Ribboneintrag **Datei|Optionen** öffnen. Hier finden wir unter **Aktuelle Datenbank** einige verheißungsvolle Eigenschaften:

- Die erste lautet **Anwendungstitel** und sorgt dafür, dass der hier angegebene Text in der Titelleiste der Access-Anwendung erscheint.
- Die zweite heißt Anwendungssymbol und wir hoffen, dass wir damit ein Icon festlegen können, das links oben in der Titelleiste von Access erscheint.
- Außerdem können wir mit der Option **Als Formular- und Berichtssymbol** noch festlegen, dass wir das Anwendungssymbol auch als Symbol für Formulare und Berichte sehen wollen.

Also wählen wir ein passendes Icon aus und bearbeiten die Einstellungen wie in Bild 2.

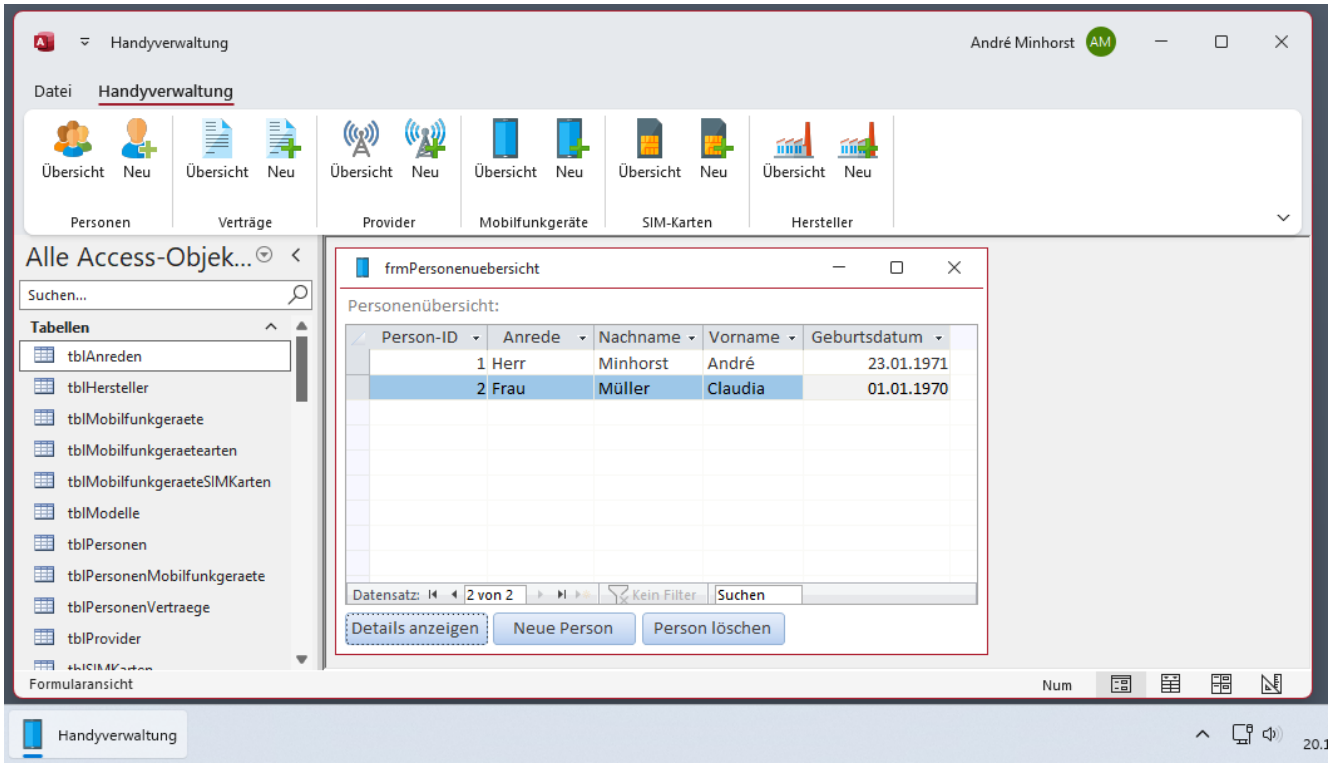

Bild 3: Der aktuelle Stand

Zwischenstand

Schauen wir uns an, wie die Anwendung nach diesen Änderungen aussieht, so haben wir unser Ziel nur teilweise erreicht (siehe Bild 3). Der Anwendungstitel ist nun stimmig, allerdings zeigt das Anwendungsfenster nicht das gewünschte Icon an. Diese erscheint wiederum unten in der Taskleiste. Das ist immerhin etwas. Außerdem wird das Icon auch als Icon der Formulare angezeigt. Auch das sieht bereits viel besser und individueller aus als das übliche Icon von Formularen und Berichten.

Formulartitel per Eigenschaft definieren

Schauen wir uns nun die Formulartitel an. Standardmäßig zeigt Access den Namen des Formulars als Titel an. Wenn wir einen eigenen Titel anzeigen wollen, erledigen wir das am einfachsten durch das Einstellen der Eigenschaft Beschriftung des Formulars selbst. Diese finden wir im Bereich **Format** des Eigenschaftenblatts (siehe Bild 4).

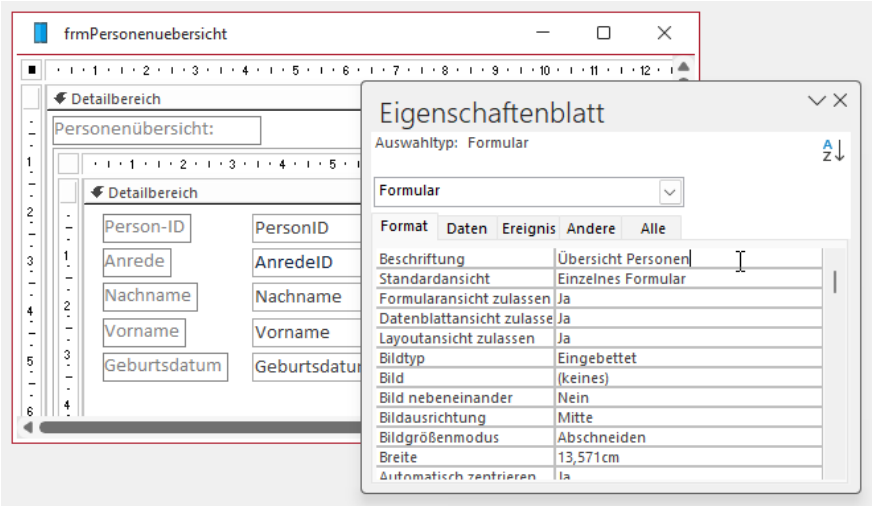

Bild 4: Einstellen der Formularbeschriftung

Wechseln wir danach in die Formularansicht, sehen wir die gewünschte Überschrift in der Titelleiste (siehe Bild 5).

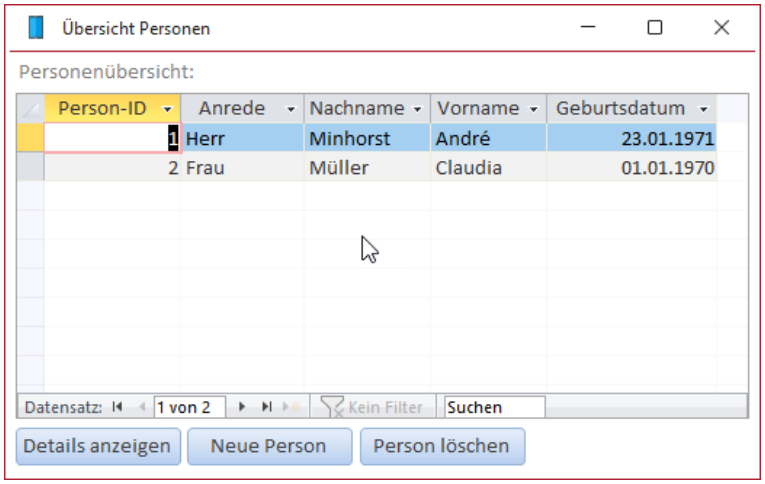

Bild 5: Formular mit individueller Beschriftung

Formulartitel per Code definieren

Bei den Übersichtsformularen reicht ein statischer Text als Formulartitel aus. Wenn wir jedoch die Detailformulare betrachten, stellen wir fest, dass diese für verschiedene Aufgaben geöffnet werden können:

- zum Anlegen neuer Datensätze und
- zum Anzeigen vorhandener Datensätze.

In diesem Fall möchten wir unterschiedliche Titeltexte anzeigen. Dazu müssen wir beim Öffnen des Formulars herausfinden, ob das Formular zum Anlegen eines neuen Datensatzes geöffnet wurde oder zum Bearbeiten eines vorhandenen Datensatzes.

Dies gelingt mit einer Eigenschaft namens **DataEntry**. Hat diese den Wert **True**, wurde das Formular mit der **DoCmd.OpenForm**-Methode unter Verwendung des Wertes **acFormAdd** des Parameters **DataMode** geöffnet. Im Falle des Wertes **False** wurde der Wert **acFormEdit** verwendet, das Formular also zum Anzeigen/Bearbeiten des Datensatzes geöffnet.

Wir nutzen also eine kleine Prozedur, die durch das Ereignis **Beim Laden** ausgelöst wird, und die den Wert der Eigenschaft **DataEntry** prüft. Wenn **DataEntry** den Wert **True** hat, soll die Beschriftung der Titelleiste des Formulars auf **Neue Person anlegen** lauten, falls der Wert **False** vorliegt, soll der Text **Person bearbeiten** angezeigt werden:

```
Private Sub Form_Load()
    If Me.DataEntry = True Then
        Me.Caption = "Neue Person anlegen"
    Else
        Me.Caption = "Person bearbeiten"
    End If
End Sub
```

Das Ergebnis sieht wie in Bild 6 aus.

Formularicons hinzufügen

Weiter oben haben wir bereits die Möglichkeit aufgezeigt, das Anwendungsicon als Formular- und Berichtsicon anzuzeigen. Allerdings möchten wir für

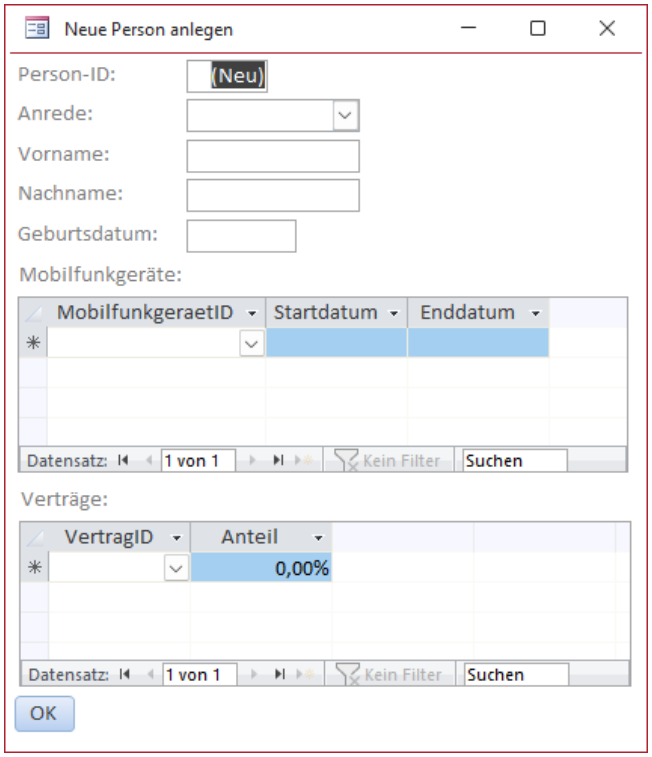

Bild 6: Detailformular mit kontextabhängigem Titel

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst

Handyverwaltung XI: Datenblatt-Ribbon unterbinden

Wenn wir ein Formular in der Datenblattansicht anzeigen, erscheint automatisch das Ribbon-Tab **Formulardatenblatt**. Dabei handelt es sich um kontextabhängiges Tab-Element, das nur im Kontext mit Datenblättern in Formularen erscheint. Wenn wir dieses nicht anzeigen wollen, können wir dies verhindern. Dazu brauchen wir nur für das Unterformular einzustellen, dass auch für dieses keine eingebauten Ribbon-Elemente angezeigt werden sollen. Wie das gelingt und welche zwei Möglichkeiten wir dazu haben, zeigen wir in diesem Artikel.

Beispieldatenbank

Die Beispiele dieses Artikels findest Du in der Datenbank **Handyverwaltung_XI_RibbonDatenblatt.accdb**.

Problem: Kontextabhängiges Datenblatt-Tab

Wenn wir die Lösung der Artikelreihe Handyverwaltung öffnen und die Personenübersicht anzeigen, wird der Fokus direkt auf das Unterformular in der Datenblattansicht verschoben. Für Formulare in der Datenblattansicht sieht Access vor, dass ein spezielles Ribbon-Tab mit einigen Befehlen für die Verwaltung von Daten in der Datenblattansicht eingeblendet wird. Dieses sieht wie in Bild 1 aus.

Ganz davon abgesehen, dass wir lieber die ganze Zeit das Anwendungsribbon aus Bild 2 sehen würden, um jederzeit auch andere Funktionen der Anwendung aufrufen zu können, wollen wir dem Benutzer bestimmte Elemente des Ribbon-Tabs **Formulardatenblatt** auch gar nicht anzeigen – zum Beispiel soll dieser nicht in eine andere Ansicht wechseln können.

Einfache Lösung des Problems

Um das zu erledigen, können wir einen einfachen Schritt durchführen: Wir stellen einfach den Wert der Eigenschaft **Name des Menübands** des Unterformulars auf den Namen des Ribbons ein, das in den Access-Optionen eingestellt wurde – in diesem Fall heißt dieses **Main**.

Dazu öffnen wir entweder direkt das betroffene Unterformular, hier **sfmPersonenuebersicht**, in der Entwurfsansicht und stellen die Eigenschaft im Be-

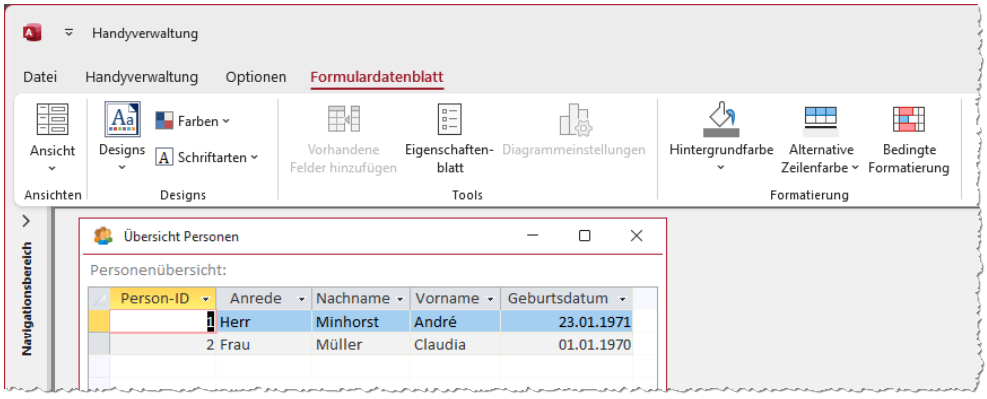

Bild 1: Ribbon bei Aktivierung eines Datenblatts

Bild 2: Dieses Ribbon wollen wir lieber sehen.

reich **Andere** des **Eigenschaftenblatts** ein. Oder wir wechseln direkt in die Entwurfsansicht des Hauptformulars, klicken einmal auf das Unterformular und dann nochmal auf die obere, linke Ecke des Unterformulars, um das Unterformular zu markieren. Die Eigenschaft **Name des Menübands** ändern wir dann wie zuvor beschrieben durch Auswahl des Eintrags **Main** (siehe Bild 3).

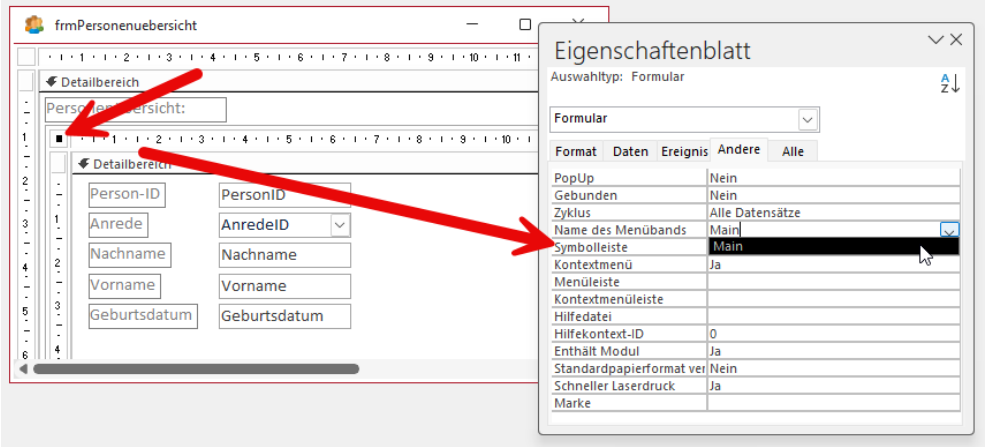

Bild 3: Einstellen des Menübands auf das Menüband der Anwendung

Nach einem Wechsel in die Formularansicht sehen wir immerhin, dass nun direkt das gewünschte Ribbon-Tab angezeigt wird. Allerdings sehen wir immer noch das Tab **Formulardatenblatt** (siehe Bild 4). Und wir möchten dieses immer noch ausblenden, damit der Benutzer es nicht nutzen kann.

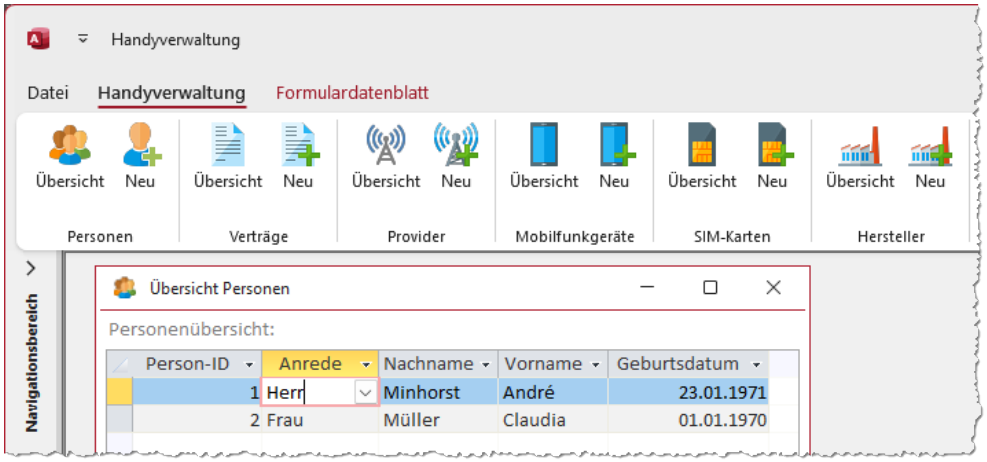

Bild 4: Nun wird das gewünschte Ribbon auch bei aktiviertem Datenblatt angezeigt.

Kontextabhängiges Tab-Element ausblenden

Also müssen wir einen Umweg gehen, um das Ribbon-Tab Formulardatenblatt endgültig auszublenden. Wie im Artikel **Ribbon für die Handyverwaltung** (**www.access-basics.de/385**) beschrieben, haben wir bereits eine Ribbon-Tabelle namens **USysRibbons** zur Datenbank hinzugefügt. In dieser ist auch das Ribbon **Main** gespeichert, sodass wir dieses als Anwendungsribbon oder auch als Ribbon für Formulare und Berichte auswählen können.

Um das kontextabhängige Ribbon-Tab Formulardatenblatt vollständig auszublenden, wenn ein Unterformular in der Datenblattansicht angezeigt wird, müssen wir die vorhandene Ribbondefinition für diesen Zweck erweitern.

In diesem Fall fügen wir keine weiteren Elemente zum Ribbon hinzu, sondern wir ändern die Eigenschaft **visible** des kontextabhängigen Tab-Elements **Formulardatenblatt** auf **false**.

Dazu müssen wir prinzipiell den Pfad zu diesem Element durch eine Ribbondefinition nachbauen und dann seine Eigenschaft anpassen. Dazu brauchen wir das **customUI**-Element, das **ribbon**-Element und direkt darunter fügen wir die benötigten neuen Elemente an. Das erste Element heißt **contextualTabs** und

ACCESS BASICS

Schade! Du hast die letzte Seite der Leseprobe dieses Artikels erreicht. Wenn Du den Rest lesen möchtest oder sogar alle über 400 Artikel, die bisher in Access [basics] erschienen sind, hol Dir doch das Jahresabonnement!

Zum Jahresabo ... hier klicken

André Minhorst