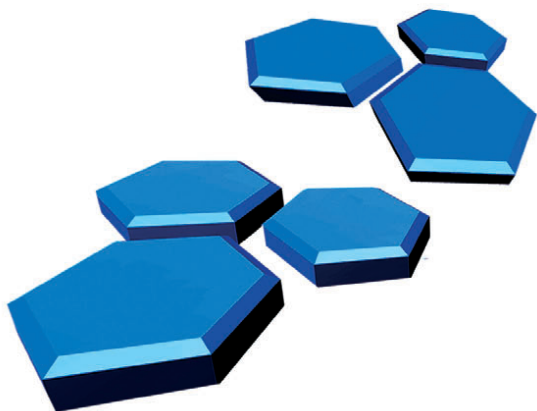




Premium Codebook

BERND HELD

Das **Access-VBA** Codebook



ADDISON-WESLEY

Die Access-Objekte

In diesem Kapitel können Sie Beispiele zu den wichtigsten Access-Objekten nachschlagen. Zu jedem Objekt stehen Ihnen zahlreiche Methoden und Eigenschaften zur Verfügung, die Sie bei der Programmierung einsetzen können.

Das AccessObject-Objekt

Mithilfe des `AccessObject`-Objekts können Sie auf Auflistungsobjekte zugreifen und diese auswerten. Dabei stehen Ihnen folgende Auflistungsobjekte zur Verfügung:

Access-Objekte

AccessObject	Auflistung	Enthält Informationen über
Datenzugriffsseite	AllDataAccessPages	gespeicherte Datenzugriffsseiten (Bem.: ab Access 2007 nicht mehr unterstützt)
Datenbankdiagramm	AllDatabaseDiagrams	gespeicherte Datenbankdiagramme
Form	AllForms	gespeicherte Formulare
Funktion	AllFunctions	gespeicherte Funktionen
Makro	AllMacros	gespeicherte Makros
Modul	AllModules	gespeicherte Module
Abfrage	AllQueries	gespeicherte Abfragen
Bericht	AllReports	gespeicherte Berichte
Gespeicherte Prozedur	AllStoredProcedures	gespeicherte Prozeduren
Table	AllTables	gespeicherte Tabellen
Sicht	AllViews	gespeicherte Ansichten

Tabelle 57: Alle AccessObjects im Überblick

130 Alle Module auflisten

Um alle in der Entwicklungsumgebung angelegten Module aufzulisten, können Sie auf die Auflistung `AllModules` zurückgreifen. Im Beispiel aus *Listing 145* werden die Namen sowie das Erstellungsdatum aller Module im Direktfenster der Entwicklungsumgebung ausgegeben.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1AccObj
'=====
```

```
Sub ModuleAuflisten()
    Dim obj As AccessObject
    Dim dbs As CurrentProject
```

Listing 145: Alle Module werden im Direktfenster dokumentiert.

```

Set dbs = Application.CurrentProject
For Each obj In dbs.AllModules
    Debug.Print "Name          : " & obj.Name
    Debug.Print "Erstellt am : " & obj.DateCreated & vbCrLf
Next obj
End Sub

```

Listing 145: Alle Module werden im Direktfenster dokumentiert. (Forts.)

Über die Eigenschaft `CurrentProject` haben Sie Zugriff auf die aktuelle Datenbank. Über die Eigenschaft `Name` können Sie die Namen der einzelnen Module ausgeben lassen, wenn Sie das Auflistungsobjekt `AllModules` einsetzen. Mithilfe der Eigenschaft `DateCreated` können Sie abfragen, wann das Modul erstellt wurde.

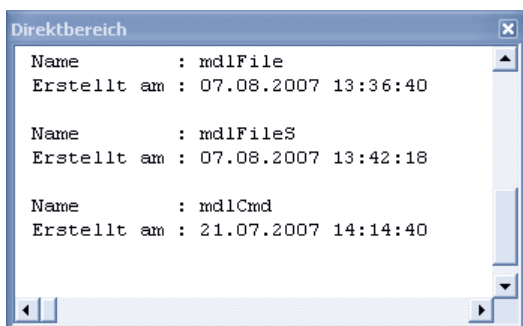


Abbildung 81: Die verfügbaren Module der Datenbank

Hinweis

Im Kapitel *VBE & Security* können Sie nachschlagen, wie Sie mithilfe der VBE-Programmierung noch weiter gehende Aufgaben schnell erledigen können.

131 Alle Tabellen auflisten

Über die Auflistung `AllTables` haben Sie Zugriff auf alle Tabellen, die sich in der Datenbank befinden. In der Prozedur aus *Listing 146* werden die Namen aller Tabellen in das Direktfenster der Entwicklungsumgebung geschrieben. Dabei wird geprüft, ob die jeweilige Tabelle momentan geöffnet ist.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1AccObj
'=====

```

Listing 146: Tabellennamen dokumentieren

```

Sub TabellenAuflisten()
    Dim obj As AccessObject
    Dim dat As CurrentData

    Set dat = Application.CurrentData
    For Each obj In dat.AllTables
        Debug.Print "Tabellenname: " & obj.Name
        Debug.Print "Geöffnet      : " & obj.IsLoaded & vbCrLf
    Next obj
End Sub

```

Listing 146: Tabellennamen dokumentieren (Forts.)

Über die Eigenschaft `CurrentData` haben Sie Zugriff auf einige Objekte, die sich in der geöffneten Datenbank befinden. Unter diesen Objekten befinden sich auch Tabellen, die Sie über die Auflistung `AllTables` ansprechen können. Mithilfe der Eigenschaft `Name` können Sie den Namen der jeweiligen Tabelle ermitteln. Die Funktion `IsLoaded` gibt Auskunft darüber, ob eine Tabelle momentan geöffnet ist. Wenn ja, dann meldet diese Funktion den Rückgabewert `True`.

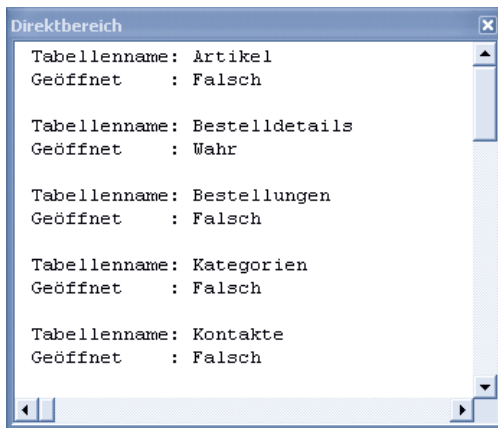


Abbildung 82: Tabellen auflisten und Status prüfen

132 Alle Abfragen auflisten

Über den Einsatz des `AccessObject` können Sie alle Abfragen der Datenbank anzeigen lassen. In der Prozedur aus *Listing 147* werden die Namen aller in der Datenbank befindlichen Abfragen dokumentiert.

```

' =====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1AccObj
' =====

```

Listing 147: Alle Abfragen dokumentieren

```

Sub AbfragenAuflisten()
    Dim obj As AccessObject
    Dim dat As CurrentData

    Set dat = Application.CurrentData

    For Each obj In dat.AllQueries
        Debug.Print obj.Name
    Next obj
End Sub

```

Listing 147: Alle Abfragen dokumentieren (Forts.)

In einer For Each Next-Schleife durchlaufen Sie Abfragen und ermitteln die Namen der Abfragen über die Eigenschaft `Name`.

Die Application-Objekte

Das `Application`-Objekt steht an oberster Stelle. Es bezieht sich auf die aktive Microsoft Access-Anwendung und beinhaltet alle darunter liegenden Objekte wie Formulare, Reports, Drucker und Bildschirm.

133 Name der aktuellen Datenbank abfragen

Soll der Name der aktuell geöffneten Datenbank abgefragt werden, dann können Sie diese Aufgabe über die Prozedur aus *Listing 148* durchführen.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====

```

```

Sub DatenbankName()
    MsgBox "Die aktuelle Datenbank heißt: " & _
        Application.CurrentProject.Name
End Sub

```

Listing 148: Dateiname der Datenbank ermitteln – Variante 1

Über die Eigenschaft `Name`, die Sie auf das Objekt `CurrentProject` anwenden, können Sie den Namen der aktuellen Datenbank ermitteln.

Eine zweite Variante, um den Namen der Datenbank zu ermitteln, sehen Sie in der Prozedur aus *Listing 149*. Gegenüber der vorigen Variante, die Ihnen lediglich den Dateinamen anzeigt, zeigt Ihnen diese Variante den kompletten Dateipfad der Datenbank an.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====

Sub DatenbankName2()
    MsgBox "Die aktuelle Datenbank heit: " & _
        Application.CurrentDb.Name
End Sub

```

Listing 149: Pfad der Datenbank ermitteln – Variante 2

Über die Eigenschaft `Name`, die Sie auf das Objekt `CurrentDb` anwenden, können Sie den Namen der aktuellen Datenbank ermitteln.

134 Pfad der aktuellen Datenbank abfragen

Um den Pfad der aktuell geöffneten Datenbank abzufragen, setzen Sie die Eigenschaft `Path`, wie in *Listing 150* gezeigt, ein.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====

Sub PfadAusgeben()
    MsgBox "Die aktuelle Datenbank heit: " & _
        Application.CurrentProject.Path
End Sub

```

Listing 150: Pfad der Datenbank ermitteln

Um beide Informationen, also den Pfad sowie den Namen der aktuellen Datenbank abzufragen, verwenden Sie die Eigenschaft `FullName`.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====

Sub PfadUndNameAusgeben()
    MsgBox "Die aktuelle Datenbank heit: " & _
        Application.CurrentProject.FullName
End Sub

```

Listing 151: Pfad und Dateiname der Datenbank ermitteln

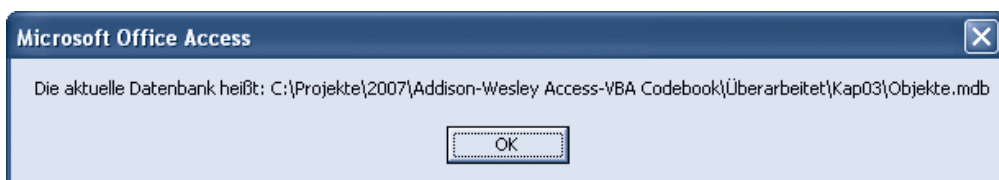


Abbildung 83: Pfad und Dateiname abfragen

135 Pfad der Anwendung abfragen

Über den Einsatz der Methode `SysCmd` können Sie den Pfadnamen der Anwendung ermitteln. Im Beispiel aus *Listing 152* wird der Installationspfad von Microsoft Access ermittelt.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====
```

```
Sub InstallationsPfadAusgeben()
    MsgBox "Das Installationsverzeichnis von Access lautet: " & _
        vbLf & SysCmd(acSysCmdAccessDir), vbInformation
End Sub
```

Listing 152: Den Pfad der Anwendung abfragen

Übergeben Sie der Methode `SysCmd` als Konstante `acSysCmdAccessDir`, um das Installationsverzeichnis von Microsoft Access abzufragen.

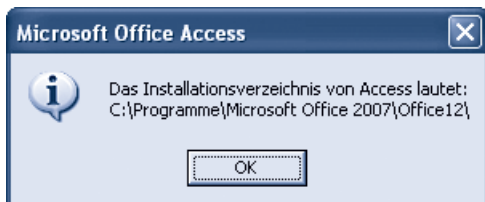


Abbildung 84: Den Installationspfad von Access ausgeben

136 Access-Version feststellen

Ebenso geeignet, um die eingesetzte Access-Version zu ermitteln, ist der Einsatz der Methode `SysCmd`. Im folgenden Beispiel aus *Listing 153* wird die Access Versionsnummer am Bildschirm ausgegeben.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====

Sub AccessVersionErmittleIn()
    MsgBox "Die eingesetzte Access-Version lautet: " & _
        vbCrLf & SysCmd(acSysCmdAccessVer), vbInformation
End Sub

```

Listing 153: Die eingesetzte Access-Version abfragen

Übergeben Sie der Methode `SysCmd` als Konstante `acSysCmdAccessVer`, um die Access-Versionsnummer zu ermitteln.

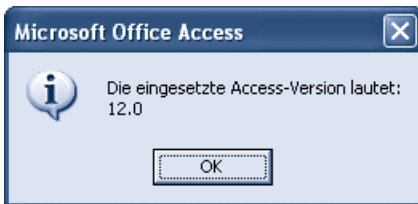


Abbildung 85: Die Versionsabfrage wurde durchgeführt.

Hinweis

Die Versionsnummer 12.0 steht für Microsoft Access 2007.

Es gibt eine zweite Möglichkeit, wie Sie die Version von Access ermitteln können. Erledigen können Sie dies über die Eigenschaft `Version` des Objekts `Application`:

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====

Sub AccessVersionErmittleIn2()
    MsgBox "Sie arbeiten mit der Access-Version: " & _
        & Application.Version
End Sub

```

Listing 154: Die eingesetzte Access-Version ermitteln

137 Aktuellen Anwendernamen abfragen

Mithilfe der Methode `CurrentUser` können Sie den Namen des aktuellen Benutzers der Datenbank zurückgeben.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====
```

```
Sub AnwenderNameAusgeben()
    MsgBox "Der aktuelle Benutzer ist: " & _
        Application.CurrentUser, vbInformation
End Sub
```

Listing 155: Den aktuellen Anwendernamen abfragen

138 Aktuelle Datenbank schließen

Soll die aktuelle Datenbank geschlossen werden, dann können Sie für diese Aufgabe die Methode `CloseCurrentDatabase` einsetzen. Diese Methode kann unter anderem dann eingesetzt werden, wenn Sie eine Microsoft Access-Datenbank von einer anderen Anwendung aus über Automaten geöffnet haben.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====
```

```
Sub DatenbankBeenden()
    Application.CloseCurrentDatabase
End Sub
```

Listing 156: Aktuelle Datenbank schließen

139 Applikation beenden

Möchten Sie Microsoft Access beenden, dann setzen Sie die Methode `Quit` ein, die Sie auf das Objekt `Application` anwenden. In der Prozedur aus *Listing 157* wird die Anwendung beendet und alle Änderungen an der aktuellen Datenbank vorher gesichert.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====
```

```
Sub ApplikationBeenden()
    Application.Quit acQuitSaveAll
End Sub
```

Listing 157: Applikation beenden

Die Methode `Quit` kann mit folgenden Optionen eingesetzt werden:

- ▶ `acQuitPrompt`: Zeigt ein Dialogfeld an, in dem Sie gefragt werden, ob geänderte, aber nicht gesicherte Objekte vor dem Beenden von Microsoft Access gespeichert werden sollen.
- ▶ `acQuitSaveAll`: Speichert alle Objekte, ohne ein Dialogfeld anzuzeigen. (Standardeinstellung).
- ▶ `acQuitSaveNone`: Beendet Microsoft Access, ohne Objekte zu speichern.

140 Drucker auflisten

Um zu ermitteln, welche Drucker Sie im Einsatz haben und an welchem Anschluss diese hängen, können Sie das Auflistungsobjekt `Printers` nutzen, das Sie im Zusammenspiel mit dem Objekt `Application` einsetzen.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1App
'=====
```

```
Sub DruckerListen()
    Dim prt As Printer

    For Each prt In Application.Printers
        With prt
            Debug.Print "Druckername: " & .DeviceName & vbCrLf _
                & "Anschluss: " & .Port
        End With
    Next prt
End Sub
```

Listing 158: Drucker und Ports listen

Die Eigenschaft `DeviceName` zeigt den Druckernamen an. Mithilfe der Eigenschaft `Port` können Sie den Anschluss anzeigen, dem der Drucker zugeordnet ist.

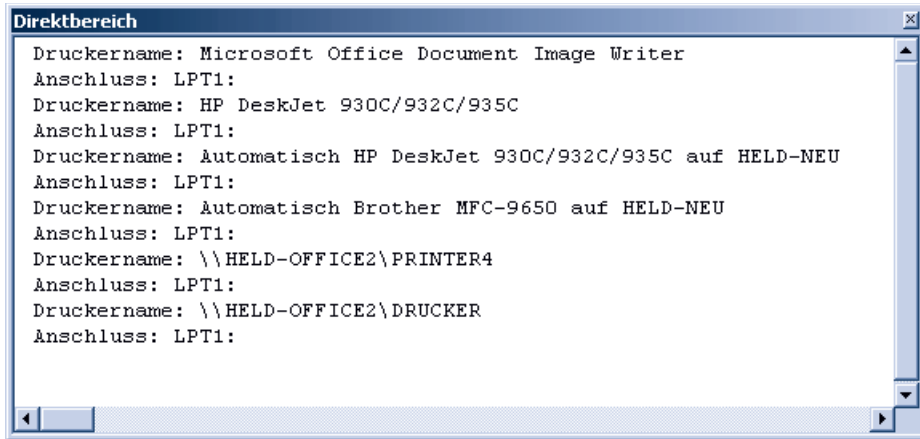


Abbildung 86: Die verfügbaren Drucker mit Anschlussbelegung

Das Control-Objekt

Das Objekt `Control` stellt ein Steuerelement dar, das sich in einem Formular, einem Bericht, einem Bereich oder innerhalb eines anderen Steuerelements befindet oder einem anderen Steuerelement beigelegt ist.

Bei dem folgenden Beispiel aus *Listing 159* werden alle Textfelder des Formulars `KUNDEN` im Direktfenster ausgegeben.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname   Objekte.mdb
' Modul       Md1Ctl
'=====

Sub SteuerelementeDokumentieren()
    Dim ctl As Control
    Dim frm As Form

    Set frm = Form_Kunden

    For Each ctl In frm.Controls
        If TypeName(ctl) = "Textbox" Then
            Debug.Print ctl.Name
        End If
    Next ctl
End Sub
```

Listing 159: Alle Textfelder in einem Formular auslesen

Über die Anweisung `Set` geben Sie zunächst einmal an, wie das Formular heißt. Danach setzen Sie eine `For Each Next`-Schleife auf, die alle Steuerelemente des Formulars abarbeitet. Innerhalb der Schleife prüfen Sie über den Einsatz der Funktion `TypeName`, ob es sich bei dem jeweiligen Steuerelement um ein Textfeld handelt. Wenn ja, dann meldet diese Funktion den Ausdruck `Textbox` zurück. In diesem Fall geben Sie den Namen des Steuerelementes über die Eigenschaft `Name` aus.

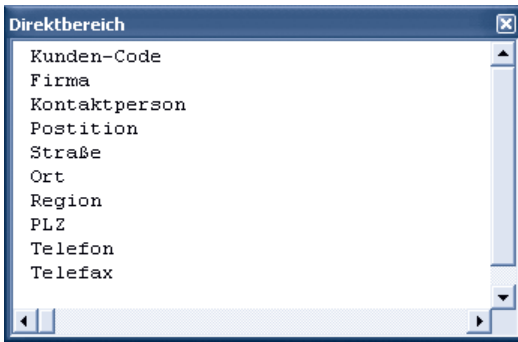


Abbildung 87: Die Textfelder eines Formulars wurden dokumentiert.

Das DoCmd-Objekt

Über den Einsatz des Objekts `DoCmd` können Sie Aktionen wie das Schließen von Fenstern, das Öffnen von Formularen und das Festlegen der Werte von Steuerelementen ausführen.

Auf den nächsten Seiten erfolgt nun eine Auswahl von typischen Aufgaben, die in Verbindung mit diesem Objekt durchgeführt werden können.

141 Filter in Tabellen setzen

Mithilfe der Methode `ApplyFilter` können Sie einen Filter hinzufügen. Diesen Filter können Sie gewöhnlich in Tabellen oder Formularen einsetzen. Die Syntax dieser Methode lautet:

```
Ausdruck.ApplyFilter(Filtername, Bedingung)
```

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein <code>DoCmd</code> -Objekt zurückgibt.
Filtername	Optional. Legt den Namen eines Filters oder einer Abfrage in der aktuellen Datenbank fest.
Bedingung	Optional. Legt die Bedingung fest.

Tabelle 58: Die Argumente der Methode `ApplyFilter`

Im folgenden Beispiel aus *Listing 160* wird die Tabelle `ARTIKEL` geöffnet. Danach werden alle Artikel ausgefiltert, die einen Lagerbestand kleiner als 10 aufweisen.

```

' =====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
' =====

```

```

Sub FilterSetzen()
    DoCmd.OpenTable "Artikel"
    DoCmd.ApplyFilter , "Lagerbestand < 10"
End Sub

```

Listing 160: Einen Tabellenfilter setzen

Über die Methode `OpenTable` öffnen Sie die Tabelle. Danach wenden Sie die Methode `ApplyFilter` an, um den Filter zu setzen. Das Filterkriterium wird dabei in Anführungszeichen angegeben.



Arti	Artikelname	Lieferant	Kategorie	Liefereinheit	Cinzelpreis	Lagerbestand
8	Northwoods Cranberry S	Grandma Kelly's Homes	Gewürze	12 x 12-oz-Gläser	41,38 €	6
21	Sir Rodney's Scones	Specialty Biscuits, Ltd.	Süßwaren	24 Packungen x 4 Stk	10,35 €	3
32	Mascarpone Fabboli	Formaggi Fortini s.r.l.	Milchprodukte	24 x 200-g-Packunge	33,11 €	9
45	Røgede sild	I ynghysild	Meeresfrüchte	1-kg-Paket	9,83 €	5
66	Louisiana Hot Spiced Okr	New Orleans Cajun Deli	Gewürze	24 x 8-oz-Gläser	17,59 €	4
68	Scottish Longbreads	Specialty Biscuits, Ltd.	Süßwaren	10 Kartons x 8 Stück	12,93 €	6
74	Longlife Tofu	Tokyo Traders	Naturprodukte	5-kg-Paket	10,35 €	4
* (Neu)					0,00 €	0

Abbildung 88: Alle Artikel mit einem Lagerbestand unter 10 Stück werden angezeigt.

Sollen beispielsweise gleich mehrere Artikel ausgefiltert werden, dann können Sie diese Aufgabe über die Prozedur aus *Listing 161* lösen. Dabei werden alle Artikel mit dem Namen TOFU oder KONBU in der Tabelle ARTIKEL herausgefiltert.

```

' =====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
' =====

```

```

Sub FilterSetzen2()
    DoCmd.OpenTable "Artikel"
    DoCmd.ApplyFilter , "Artikelname = 'Tofu' OR Artikelname= 'Konbu'"
End Sub

```

Listing 161: Mehrere Bedingungen beim Filtern angeben



Arti	Artikelname	Lieferant	Kategorie	Liefereinheit	Einzelprei	Lagerbestand
13	Konbu	Mayumi's	Meeresfrüchte	2-kg-Karton	6,21 €	24
14	Tofu	Mayumi's	Naturprodukte	40 x 100-g-Packung	24,05 €	35
*	(Neu)				0,00 €	0

Abbildung 89: Die Artikel Konbu und Tofu wurden herausgefiltert.

Hinweis

Durch die Methode ShowAllRecords werden alle vorhandenen Filter entfernt, die für die aktuelle Tabelle, die aktuelle Abfrage oder das aktuelle Formular eingerichtet wurden. Diese Methode hat keine weiteren Argumente.

142 Objekte umbenennen

Mithilfe der Methode Rename können Sie Objekte wie Tabellen, Formulare und sonstige umbenennen. Die Syntax dieser Methode lautet:

Ausdruck.Rename(NewName, ObjectType, OldName)

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein DoCmd-Objekt zurückgibt.
NewName	Erforderlich. Legt den neuen Namen des umzubenennenden Objekts fest.
ObjectType	Optional. Gibt den Typ des umzubenennenden Objekts an.
OldName	Optional. Gibt den alten Namen des Objekts an.

Tabelle 59: Die Argumente der Methode Rename

Im folgenden Beispiel aus Listing 162 wird eine Tabelle umbenannt.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====

Sub TabelleUmbenennen()
    On Error GoTo Fehler
    DoCmd.Rename "ArtikelNeu", acTable, "ArtikelAktuell"
Exit Sub
Fehler:
    MsgBox "Fehler bei TabelleUmbenennen: " & Err.Description
End Sub
```

Listing 162: Eine Tabelle umbenennen

Fehler:

```
MsgBox Err.Number & vbCrLf & Err.Description
End Sub
```

Listing 162: Eine Tabelle umbenennen (Forts.)

Sollte die Umbenennung nicht erfolgreich sein, dann können Sie statt eines Prozedurabsturzes über die Anweisung `On Error GoTo` zu einer Fehlerbehandlung übergehen. Dabei werten Sie das Objekt `Err` aus, indem Sie über die Eigenschaften `Number` und `Description` die Fehlernummer sowie eine Beschreibung des Fehlers am Bildschirm anzeigen.

143 Objekte kopieren

Mithilfe der Methode `CopyObject` können Sie Objekte wie Tabellen und Berichte kopieren. Die Syntax dieser Methode lautet:

```
Ausdruck.CopyObject(Zieldatenbank, NeuerName, Quellobjektyp, Quellobjektname)
```

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein <code>DoCmd</code> -Objekt zurückgibt.
Zieldatenbank	Optional. Gibt den gültigen Pfad und Dateinamen für die Datenbank an, in die das Objekt kopiert werden soll.
NeuerName	Optional. Legt den neuen Namen für das zu kopierende Objekt fest. Sie können beim Kopieren in eine andere Datenbank denselben Namen verwenden, indem Sie dieses Argument nicht angeben.
Quellobjektyp	Optional. Gibt an, um welchen Objekttyp es sich handelt.
Quellobjektname	Optional. Gibt den Namen des Quellobjekts an.

Tabelle 60: Die Argumente der Methode `CopyObject`

Beim folgenden Beispiel aus *Listing 163* wird die Tabelle `ARTIKEL` kopiert und der Name `ARTIKELAKTUELL` vergeben.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====

Sub TabelleKopieren()
    DoCmd.CopyObject , "ArtikelAktuell", acTable, "Artikel"
End Sub
```

Listing 163: Tabelle in aktueller Datenbank kopieren

Soll die Tabelle ARTIKEL in eine andere Datenbank kopiert werden, die momentan nicht geöffnet ist, dann wenden Sie die Prozedur aus *Listing 164* an.

```

'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====

Sub TabelleKopierenInExterneDB()
    On Error GoTo Fehler
    DoCmd.CopyObject "C:\Eigene Dateien\Backup.mdb", _
        "ArtikelAktuell", acTable, "Artikel"
    Exit Sub

Fehler:
    MsgBox Err.Number & vbTab & Err.Description
End Sub

```

Listing 164: Tabelle in andere Datenbank kopieren

Sollte der Kopiervorgang nicht erfolgreich sein, dann können Sie statt eines Prozedurabsturzes über die Anweisung `On Error GoTo` zu einer Fehlerbehandlung übergehen. Dabei werten Sie das Objekt `Err` aus, indem Sie über die Eigenschaften `Number` und `Description` die Fehlernummer sowie eine Beschreibung des Fehlers am Bildschirm anzeigen.

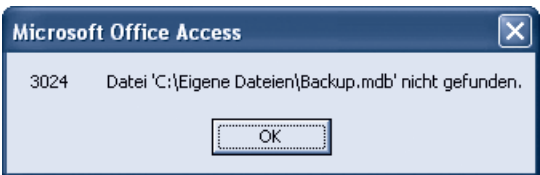


Abbildung 90: Den Fehlerfall über das Objekt Err abfangen

144 Objekte löschen

Über die Methode `DeleteObject` können Sie ein Objekt wie beispielsweise eine Tabelle oder ein Formular aus einer Datenbank löschen. Die Syntax dieser Methode lautet:

`Ausdruck.DeleteObject(Objecttyp, Objektname)`

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein <code>DoCmd</code> -Objekt zurückgibt.
Objecttyp	Gibt den Objecttyp an.
Objektname	Optional. Gibt den Namen des Objekts bekannt, das gelöscht werden soll.

Tabelle 61: Die Argumente der Methode DeleteObject

Im folgenden Beispiel aus *Listing 165* wird die Tabelle ARTIKELAKTUELL aus der aktuellen Datenbank gelöscht.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====

Sub TabelleLöschen()
    On Error GoTo Fehler
    DoCmd.DeleteObject acTable, "ArtikelAktuell"
    Exit Sub

Fehler:
    MsgBox Err.Number & vbTab & Err.Description
End Sub
```

Access-Objekte

Listing 165: Eine Tabelle aus der aktuellen Datenbank löschen

145 Objekte exportieren

Mit der Methode `OutputTo` können Sie die Daten in einem bestimmten Microsoft Access-Datenbankobjekt (einem Datenblatt, einem Formular, einem Bericht, einem Modul oder einer Datenzugriffsseite) in verschiedenen Formaten ausgeben. Die Syntax dieser Methode lautet:

`OutputTo(ObjectType, ObjectName, OutputFormat, OutputFile, AutoStart, TemplateFile)`

Argument	Beschreibung
ObjectType	Erforderlich. Legt die Art des Access-Objekts fest, dessen Daten Sie exportieren möchten. Dabei haben Sie folgende Möglichkeiten: ▶ <code>acOutputForm</code>: Export der Daten eines Formulars ▶ <code>acOutputFunction</code>: Export einer Funktion ▶ <code>acOutputModule</code>: Export eines kompletten Moduls inkl. aller Funktionen und Prozeduren ▶ <code>acOutputQuery</code>: Export der Ergebnisse eine Abfrage ▶ <code>acOutputReport</code>: Export eines Berichts ▶ <code>acOutputServerView</code>: Export einer Serveransicht ▶ <code>acOutputStoredProcedure</code>: Export einer gespeicherten Prozedur ▶ <code>acOutputTable</code>: Export einer Tabelle
ObjectName	Optional. Gibt den Namen des Objekts an, das Sie exportieren möchten.

Tabelle 62: Die Argumente der Methode `OutputTo`

Argument	Beschreibung
OutputFormat	Optional. Legt fest, in welchem Datenformat Sie die Daten transferieren möchten. Die bekanntesten Formate heißen dabei wie folgt: ▶ <code>acFormatHTML</code>: Konvertiert die Daten in das HTML-Format ▶ <code>acFormatRTF</code>: Konvertiert die Daten in das Rich-Text-Format. Dieses Format kann beispielsweise problemlos in Microsoft Word eingelesen werden. ▶ <code>acFormatTXT</code>: Mit diesem Format ist das Textformat gemeint. ▶ <code>acFormatXLS</code>: Konvertiert die Daten in das Microsoft Excel-Format.
OutputFile	Optional. Der Pfad sowie der Dateiname der Datei, in die Sie die Daten transferieren möchten. Dabei muss die Datei noch nicht vorhanden sein. Access legt diese bei Bedarf selber an.
AutoStart	Optional. Damit haben Sie die Möglichkeit, die so erstellte Exportdatei gleich zu öffnen. Verwenden Sie den Wert <code>True</code> , um die entsprechende auf Windows basierende Anwendung sofort zu starten. Setzen Sie das Argument auf den Wert <code>False</code> oder lassen Sie es weg, wenn Sie die Exportdatei nicht öffnen möchten.
TemplateFile	Optional. Dieses Argument ist dann von Interesse, wenn Sie eine Vorlage beispielsweise für die HTML-Datei verwenden möchten. In diesem Fall ist dann der komplette Pfad dieser Vorlagendatei anzugeben.

Tabelle 62: Die Argumente der Methode `OutputTo` (Forts.)

Im folgenden Beispiel aus *Listing 166* wird die Tabelle `ARTIKEL` in eine Textdatei geschrieben.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====
```

```
Sub TabelleExportieren()
    On Error GoTo Fehler
    DoCmd.OutputTo acOutputTable, "Artikel", _
        acFormatTXT, "C:\Eigene Dateien\Artikel.txt", True
    Exit Sub
```

```
Fehler:
    MsgBox Err.Number & vbTab & Err.Description
End Sub
```

Listing 166: Tabelle in eine Textdatei exportieren

Geben Sie bei der Methode `OutputTo` den Objekttyp, den Namen des Objektes, das gewünschte Ausgabeformat sowie den Namen der Zieldatei an. Dabei wird die Zieldatei, sofern sie noch nicht existiert, neu angelegt. Im letzten Argument der Methode geben Sie den Wert `True` an, was bedeutet, dass die Zieldatei nach dem Export angezeigt werden soll.

Artikel-ID	Artikelname	Lieferant	Kategorie
1	Chai	Exotic Liquids	Getränk
6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Gewürze
7	Uncle Bob's organic dried	Grandma Kelly's Homestead	Naturpr
8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Gewürze
10	Ikura	Tokyo Traders	Meeresf
11	Queso Cabrales	Cooperativa de Quesos 'Las	Milchpr
13	Konbu	Mayumi's	Meeresf
14	Tofu	Mayumi's	Naturpr

Abbildung 91: Die Tabelle liegt nun als Textdatei vor.

146 Objekte drucken

Über die Methode `PrintOut` können Sie einen Drucken-Befehl in Access ausführen. Die Syntax dieser Methode lautet:

`Ausdruck.PrintOut(Druckbereich, Von, Bis, Druckqualität, Exemplare, ExemplareSortieren)`

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein DoCmd-Objekt zurückgibt.
Druckbereich	Optional. <code>acPages</code> , <code>acPrintAll</code> (Standard), <code>acSelection</code> .
Von	Optional. Gibt die erste Seite des Bereichs an, ab der gedruckt werden soll. Dieses Argument ist erforderlich, wenn Sie für das Argument Druckbereich die Konstante <code>acPages</code> angeben.
Bis	Optional. Gibt die letzte Seite an, bis zu der gedruckt werden soll. Dieses Argument ist erforderlich, wenn Sie für das Argument Druckbereich die Konstante <code>acPages</code> angeben.
Druckqualität	Optional. Dabei gelten folgende Konstanten: ▶ <code>acDraft</code> (Entwurfsqualität), ▶ <code>acHigh</code> (bestmögliche Druckqualität) ▶ <code>acLow</code> (Konzeptdruckqualität) ▶ <code>acMedium</code> (mittlere Druckqualität)
Exemplare	Optional. Gibt die Anzahl der Kopien an.
ExemplareSortieren	Optional. Verwenden Sie <code>True</code> , um die Exemplare zu sortieren, und <code>False</code> , um die Exemplare während des Druckvorgangs nicht zu sortieren.

Tabelle 63: Die Argumente der Methode `PrintOut`

Im Beispiel aus *Listing 167* wird ein Bericht in der aktuell geöffneten Datenbank geöffnet und einige Seiten davon ausgedruckt.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====

Sub BerichtDrucken()
    Application.Echo False
    DoCmd.OpenReport "Rechnung", acViewPreview
    DoCmd.PrintOut acPages, 1, 1, , 2
    DoCmd.Close
    Application.Echo True
End Sub
```

Listing 167: Bericht drucken

Über die Methode `Echo` können Sie festlegen, ob die Bildschirmaktualisierung während des Prozedurablaufs eingeschaltet bzw. ausgeschaltet wird. Wenn Sie diese Methode mit dem Argument `False` aufrufen, dann erfolgt der Druckvorgang völlig im Hintergrund. Nach dem Drucken der Seiten wenden Sie die Methode `Close` an, um den Bericht wieder zu schließen, und schalten die Bildschirmaktualisierung wieder ein.

147 Objekte versenden

Für das Versenden von E-Mails können Sie in Access die Methode `SendObject` einsetzen. Dabei können Sie ganz genau festlegen, welchen Bestandteil einer Access-Datenbank Sie versenden möchten. Ferner übergeben Sie dieser Methode die Adressaten sowie den Begleittext der E-Mail. Die Syntax dieser Methode lautet:

`SendObject(Objekttyp, Objektname, Ausgabeformat, An, Cc, Bcc, Betreff, Nachricht, NachrichtBearbeiten, Vorlagedatei)`

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein <code>DoCmd</code> -Objekt zurückgibt.
Objekttyp	Optional. Dabei geben Sie in einer Konstanten an, welchen Bestandteil der Datenbank Sie per E-Mail versenden möchten. Folgende Konstanten stehen Ihnen dabei zur Verfügung: ▶ <code>acSendDataAccessPage</code>: Eine Access-Datenzugriffseite wird einem E-Mail-Empfänger zugestellt. ▶ <code>acSendForm</code>: Ein Formular soll über eine E-Mail versendet werden. ▶ <code>acSendModule</code>: Ein Modul wird per E-Mail versendet.

Tabelle 64: Die Argumente der Methode `SendObject`

Argument	Beschreibung
Objektyp	▶ <code>acSendNoObject</code>: Es wird lediglich eine Text-E-Mail ohne Anhang versendet. Es handelt sich dabei um die Standardeinstellung. ▶ <code>acSendQuery</code>: Hierbei soll eine Abfrage per E-Mail versendet werden. ▶ <code>acSendReport</code>: Bei dieser Angabe wird ein Bericht versendet. ▶ <code>acSendTable</code>: Diese Konstante steht für das Versenden einer bestimmten Tabelle aus einer Datenbank.
Objektname	Optional. Hier muss der Name des Objekts angegeben werden, der per E-Mail versendet werden soll.
Ausgabeformat	Optional. Hier können Sie festlegen, in welcher Form das Access-Objekt versendet werden soll. Dabei haben Sie unter anderem die Auswahl zwischen folgenden Konstanten: ▶ <code>acFormatHTML</code>: Ausgabe des Access-Objektes über das HTML-Format, das Sie mit jedem Browser ansehen können. ▶ <code>acFormatRTF</code>: Beim RTF-Format handelt es sich um ein Textformat, das Sie mit nahezu jedem Textverarbeitungsprogramm öffnen können. ▶ <code>acFormatTXT</code>: Dieses Textformat ist mit jedem Text-Editor, beispielsweise im Zubehör-Programm von Windows Notepad, zu lesen. ▶ <code>acFormatXLS</code>: Dabei handelt es sich um das Excel-Tabellenformat. ▶ <code>AcFormatDAP</code>: Bei dieser Konstante handelt es sich um Datenzugriffsseiten.
An	Optional. Hier müssen Sie die Empfänger auflisten, deren Namen in die AN-Zeile der E-Mail-Nachricht aufgenommen werden sollen. Die Empfängernamen in diesem Argument müssen durch Semikola (;) voneinander getrennt werden.
Cc	Optional. Gibt an, an welche E-Mail-Empfänger Sie die E-Mail als Kopie schicken möchten. Es gelten dabei dieselben Optionen wie auch beim Argument An.
Bcc	Optional. Hier können Sie E-Mail-Empfängern eine »blinde Kopie« der E-Mail schicken, ohne dass der eigentliche Empfänger der E-Mail, der unter dem Argument AN angegeben wurde, etwas davon erfährt.
Betreff	Optional. Repräsentiert die BETREFF-Zeile der E-Mail-Nachricht. Geben Sie dort einen Betreff in doppelten Anführungszeichen an.
Nachricht	Optional. Geben Sie den Text an, der in die E-Mail eingefügt werden soll. Wenn Sie dieses Argument nicht angeben, wird nur das Objekt, jedoch kein Text, in die E-Mail-Nachricht aufgenommen.

Tabelle 64: Die Argumente der Methode `SendObject` (Forts.)

Argument	Beschreibung
NachrichtBearbeiten	Optional. Hier können Sie entscheiden, ob Sie die E-Mail direkt absenden oder zur weiteren Bearbeitung vorher öffnen möchten. Setzen Sie dieses Argument auf den Wert <code>False</code> , um die Nachricht direkt zu versenden. Setzen Sie das Argument auf den Wert <code>True</code> , um die E-Mail zur weiteren Bearbeitung zu öffnen.
Vorlagedatei	Optional. Es handelt sich um einen optionalen Wert, der den vollständigen Namen und Pfad der Datei angibt, die als Vorlage für eine HTML-Datei verwendet werden soll.

Tabelle 64: Die Argumente der Methode `SendObject` (Forts.)

Im folgenden Beispiel aus *Listing 168* wird eine E-Mail ohne Dateianhang versendet.

```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====

Sub TextMailVersenden()
    On Error GoTo Fehler
    DoCmd.SendObject , , , _
        "Fritz.Mueller@online.de", _
        "Machero@aol.com", , "Feedback erwünscht", _
        "Sehr geehrter Herr XY," & vbCrLf & vbCrLf & _
        "bitte senden Sie mir ein Feedback zu meiner Anfrage vom 20.02.2008" & _
        vbCrLf & vbCrLf & "Mit freundlichem Gruß," & vbCrLf & _
        "Bernd Held", True
Exit Sub

Fehler:
    MsgBox Err.Number & vbCrLf & Err.Description
End Sub
```

Listing 168: Eine einfache Text-E-Mail versenden

Da bei dieser Art von E-Mail kein Dateianhang entsteht, lassen Sie die ersten drei Argumente der Methode `SendObject` einfach leer. Danach geben Sie den Adressaten sowie den Kopieempfänger und den Titel sowie den eigentlichen Text der E-Mail an. Über den Einsatz der Konstanten `vbCrLf` können Sie im E-Mail-Text einen Zeilenumbruch erzeugen. Über das letzte Argument, das Sie auf `True` oder `False` setzen können, bestimmen Sie, ob die E-Mail direkt versendet oder vor dem Versand noch einmal zur Kontrolle angezeigt werden soll.

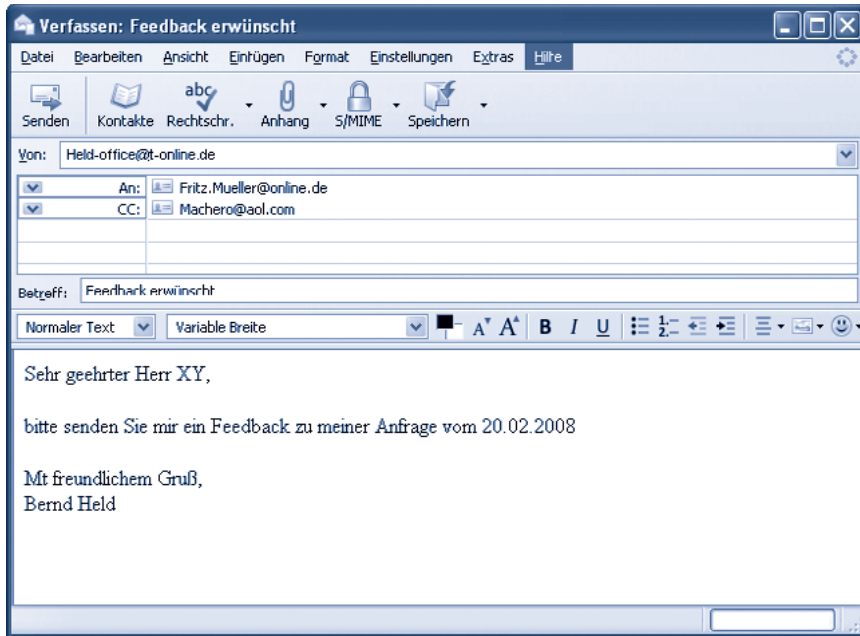


Abbildung 92: Eine Textmail ohne Anhang erstellen

Im nächsten Beispiel aus *Listing 169* wird die Tabelle ARTIKEL gleich an mehrere Anwender per E-Mail im Excel-Format verschickt.

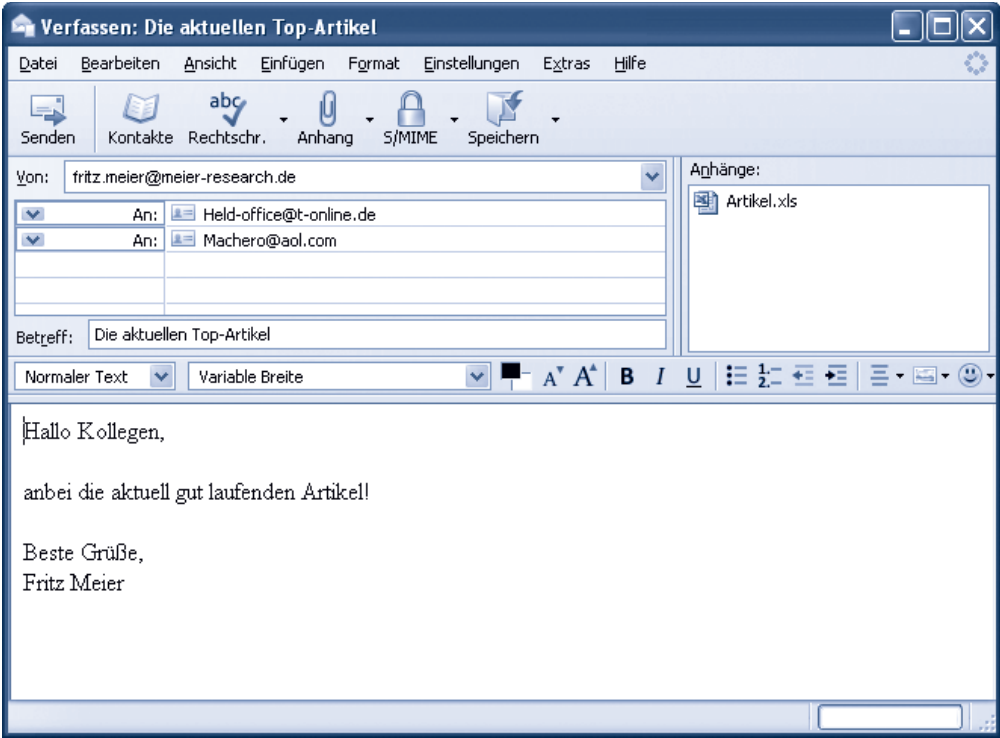
```
'=====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
'=====
```

```
Sub MailMitAnhangVersenden()
    On Error GoTo Fehler
    DoCmd.SendObject acSendTable, _
        "Artikel", acFormatXLS, _
        "Held-office@t-online.de; Machero@aol.com", _
        , , "Die aktuellen Top-Artikel", _
        "Hallo Kollegen, " & vbCrLf & vbCrLf & _
        "anbei die aktuell gut laufenden Artikel!" & _
        vbCrLf & vbCrLf & "Beste Grüße," & vbCrLf & _
        "Fritz Meier", True
    Exit Sub
```

```
Fehler:
    MsgBox Err.Number & vbCrLf & Err.Description
End Sub
```

Listing 169: E-Mail mit Dateianhang versenden

Legen Sie in den ersten drei Argumenten der Methode `SendObject` den Objekttyp, den Objektnamen und das gewünschte Ausgabeformat fest. Danach geben Sie den Adressaten sowie den Kopieempfänger und den Titel sowie den eigentlichen Text der E-Mail an.



Access-Objekte

Abbildung 93: Die Tabelle Artikel wird im Excel-Format verschickt.

148 Feld im Formular aktivieren

Über die Methode `GoToControl` können Sie den Fokus in einem Formular auf ein bestimmtes Feld setzen. Die Syntax dieser Methode lautet:

`Ausdruck.GoToControl(Steuerelementname)`

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein <code>DoCmd</code> -Objekt zurückgibt.
Steuerelementname	Erforderlich. Gibt den Namen des Steuerelements im aktiven Formular oder Datenblatt an.

Tabelle 65: Die Argumente der Methode `GoToControl`

Im Beispiel aus *Listing 170* wird das Formular `KUNDEN` geöffnet und der Fokus auf das Feld `FIRMA` gesetzt.

```

' =====
' Auf CD      Buchdaten\Beispiele
' Dateiname  Objekte.mdb
' Modul      Md1DoCmd
' =====

```

```

Sub FeldAnspringen()
    DoCmd.OpenForm "Kunden"
    DoCmd.GoToControl "Firma"
End Sub

```

Listing 170: Ein bestimmtes Feld im Formular anspringen

Über die Methode `OpenForm` öffnen Sie das Formular **KUNDEN**. Danach wenden Sie die Methode `GoToControl` an, um das gewünschte Feld zu aktivieren.

Abbildung 94: Der Fokus wurde auf das Feld **Firma** gesetzt.

149 Datensatz in Tabelle aktivieren

Über die Methode `GoToRecord` können Sie einen bestimmten Satz in einem Formular oder einer Tabelle voreinstellen. Die Syntax dieser Methode lautet:

```
Ausdruck.GoToRecord(Objekttyp, Objektname, Datensatz, Offset)
```

Argument	Beschreibung
Ausdruck	Erforderlich. Ein Ausdruck, der ein DoCmd-Objekt zurückgibt.
Objekttyp	Gibt den Objekttyp an. Es gelten dabei: acActiveDataObject (Standard), acDataForm, acDataFunction, acDataQuery, acDataServerView, acDataStoredProcedure oder acDataTable.
Objektname	Optional. Gibt den Namen des Objekts an.
Datensatz	Optional. Dabei gelten folgende Konstanten: acFirst, acGoTo, acLast, acNewRec, acNext (Standard), acPrevious.
Offset	Optional. Eine Zahl, die die Anzahl der Datensätze angibt, um die vorwärts- oder rückwärtsgeblättert werden soll, wenn Sie für das Argument Datensatz acNext oder acPrevious angegeben haben, oder der Datensatz, zu dem Sie wechseln möchten, wenn Sie für das Argument Datensatz acGoTo angegeben haben. Der Ausdruck muss eine gültige Datensatznummer ergeben.

Tabelle 66: Die Argumente der Methode GoToRecord

Im Beispiel aus Listing 171 wird die Tabelle ARTIKEL geöffnet und der letzte Datensatz ausgewählt.

```
' =====  
' Auf CD      Buchdaten\Beispiele  
' Dateiname  Objekte.mdb  
' Modul      Md1DoCmd  
' =====
```

```
Sub LetztenDatensatzAuswählen()  
    DoCmd.OpenTable "Artikel"  
    DoCmd.GoToRecord acDataTable, "Artikel", acLast  
End Sub
```

Listing 171: Den letzten Datensatz einer Tabelle auswählen



Abbildung 95: Der letzte Datensatz der Tabelle wurde ausgewählt.