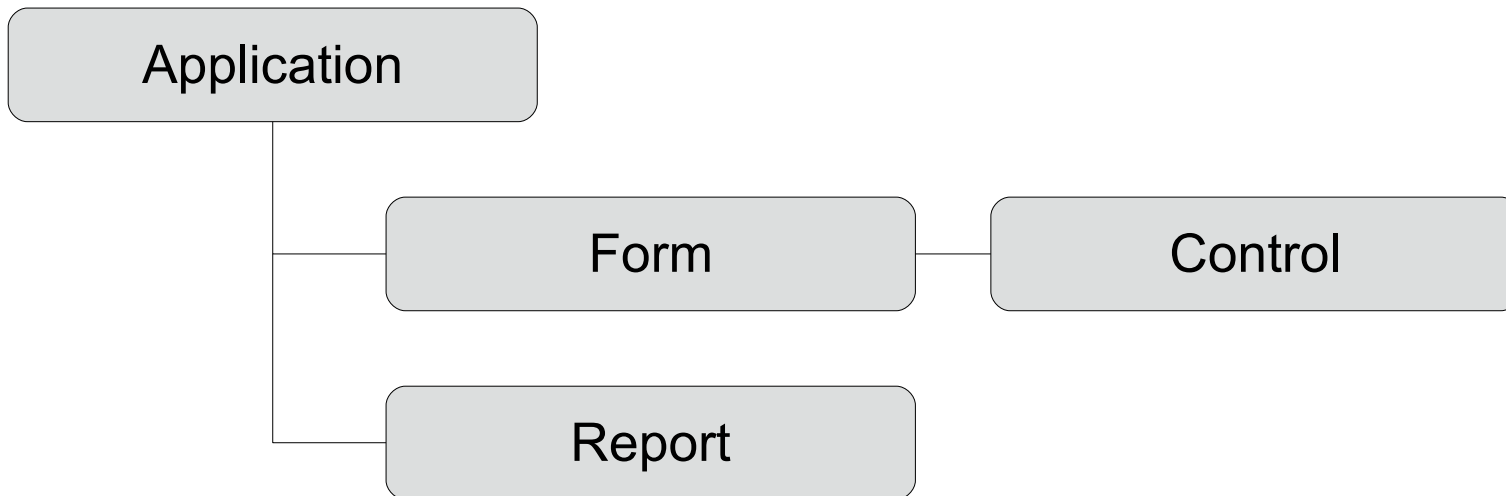
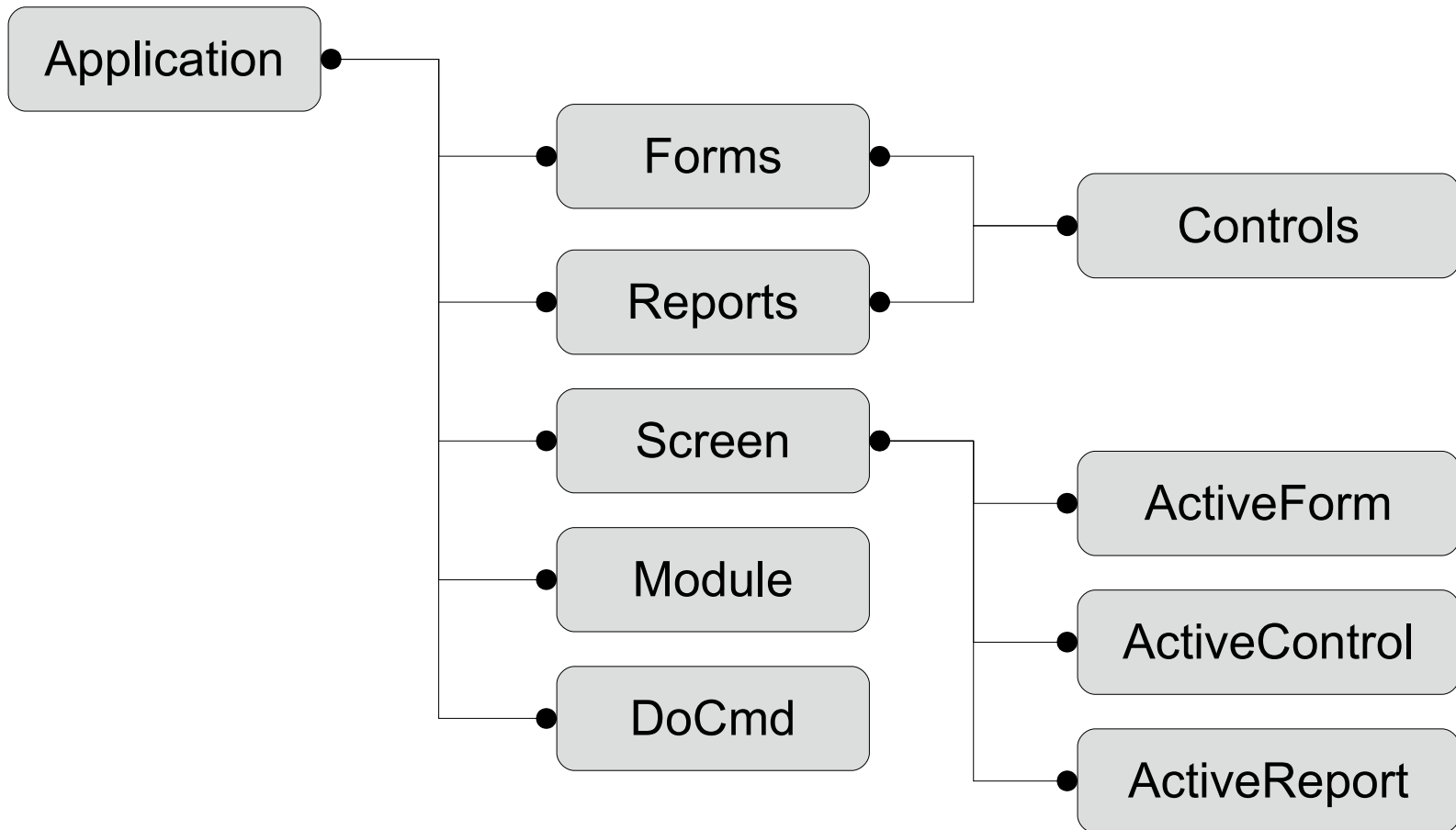


Access 2010 – Programmierung

Formulare, Berichte und Steuerelemente



Objektmodell in Access (Ausschnitt)



Anwendung « Application » ...

- beschreibt die gerade laufende Anwendung von Access.
- ist die Wurzel aller Objekte in Access.
- mit Hilfe der Eigenschaft « .Version » kann die Access-Version ermittelt werden.
- mit Hilfe der Methode « .Close » wird die Anwendung geschlossen.

Aktuelle Datenbank « Application.CurrentDB » ...

- ist ein Verweis auf die aktuell, geöffnete Datenbank.
- ist eine Methode des Objekts « Application ».
- wird zusammen mit DAO (Data Access Object) genutzt.
- « CurrentDb.Name » gibt den Speicherort sowie den Namen der aktuellen Datenbank zurück.

Aktuelles Projekt « CurrentProject » ...

- repräsentiert das aktuell geöffnete Access-Projekt.
- ist seit Microsoft Access 2000 vorhanden.
- verweist auf alle Objekte in der aktuell, geöffneten Datenbank.
- wird häufig mit ADO (ActiveX Data Object) genutzt.
- kann als Argument an ein Makro-Befehl übergeben werden.

Eigenschaften von « CurrentProject »

- « CurrentProject.Name » gibt den Namen der Access-Datenbank zurück.
- « CurrentProject.Path » gibt den Speicherort der Datenbank zurück.
- « CurrentProject.FullName » gibt den Speicherort und den Namen der Datenbank zurück.
- « CurrentProject.IsTrusted » ist wahr, wenn die Datenbank bearbeitet und der darin enthaltene Code ausgeführt werden kann.

« Application.DoCmd » ...

- bildet alle Aktionen aus der Entwurfsansicht eines Makros ab.
- Das Objekt « DoCmd » und die Methode (Makroaktion) werden durch ein Punkt getrennt.
- Jede Methode besitzt einen selbsterklärenden englischen Namen.
- Die Aktionen werden in der Hilfe von VBA unter *Access 2010-Entwicklerreferenz – Access-Objektmodellreferenz – DoCmd-Objekt* beschrieben.

Aktionen, die nicht in VBA implementiert sind

- « HinzufügenMenü », « AusführenCode », « StoppMakro » und « StoppAlleMakros » sind nicht vorhanden.
- « Warnmeldung » wird durch die Funktion « MsgBox() » ersetzt.
- « AusführenAnwendung » wird durch die Funktion « Shell » ersetzt.
- « TastaturBefehle » wird durch die Anweisung « SendKeys » ersetzt.
- « SetzenWert » wird durch Zuweisung eines Wertes an die Eigenschaft « [objekt].Value » ersetzt.

Beispiel

```
Private Sub cmdOpenFormular_Click()  
    DoCmd.Hourglass True  
  
    DoCmd.OpenForm FormName:="frmAutor"  
  
    DoCmd.GoToRecord ObjectType:=acDataForm, _  
        ObjectName:="frmAutor", _  
        Record:=acLast  
  
    DoCmd.Hourglass False  
  
    Me.Refresh  
End Sub
```

Erläuterung zu den Argumenten

- Jeder Marko-Befehl hat eine bestimmte Anzahl von Argumenten. Diese Argumente steuern die Aktion.
- Mit Hilfe des Operators « := » wird den benötigten Argumenten ein Wert zugewiesen. Die Werte werden in Abhängigkeit des Namens des Arguments vergeben. Die Reihenfolge der Argumente spielt keine Rolle.
- Mit Hilfe von « _ » (Leerzeichen + Unterstrich) können lange Argumentlisten getrennt werden.

Erläuterung zu den Makro-Aktionen

- « DoCmd.Hourglass » setzt die Sanduhr.
- « DoCmd.OpenForm » öffnet ein bestimmtes Formular. In diesem Beispiel wird das Formular „frmAutor“ in der aktuellen Datenbank geöffnet.
- « DoCmd.GoToRecord » wandert zu einem bestimmten Datensatz. In diesem Beispiel wird nach dem Öffnen des Formulars der letzte Datensatz angezeigt.

Daten nach Excel transferieren

```
Private Sub cmdToExcel_Click()  
    Dim quelle As String  
    Dim ziel As String  
  
    DoCmd.OpenForm FormName:="frmAutor"  
    quelle = Forms("frmAutor").RecordSource  
    ziel = CurrentProject.Path & "\autor.xls"  
    DoCmd.TransferSpreadsheet TransferType:=acExport, _  
        SpreadsheetType:=acSpreadsheetTypeExcel9, _  
        TableName:=quelle, _  
        FileName:=ziel, _  
        HasFieldNames:=True  
  
End Sub
```

Erläuterung

- « DoCmd.TransferSpreadsheet » transferiert ein Arbeitsblatt aus Access nach Excel.
- Folgende Angaben werden benötigt:
 - Werden die Daten exportiert, importiert oder verlinkt?
 - In welches Format werden die Daten transferiert?
 - Welche Daten werden transferiert?
 - In welche Datei werden die Daten transferiert?
 - Werden die Spaltenüberschriften der Tabelle übernommen?
- Weitere Argumente werden in der Hilfe erläutert.

Bericht in ein PDF konvertieren

```
Private Sub cmdToPDF_Click()  
    Dim ziel As String  
  
    ziel = CurrentProject.Path & "\autor.pdf"  
  
    DoCmd.OutputTo ObjectType:=acOutputReport, _  
        ObjectName:="repAutor", _  
        OutputFile:=ziel, _  
        OutputFormat:=acFormatPDF  
  
End Sub
```

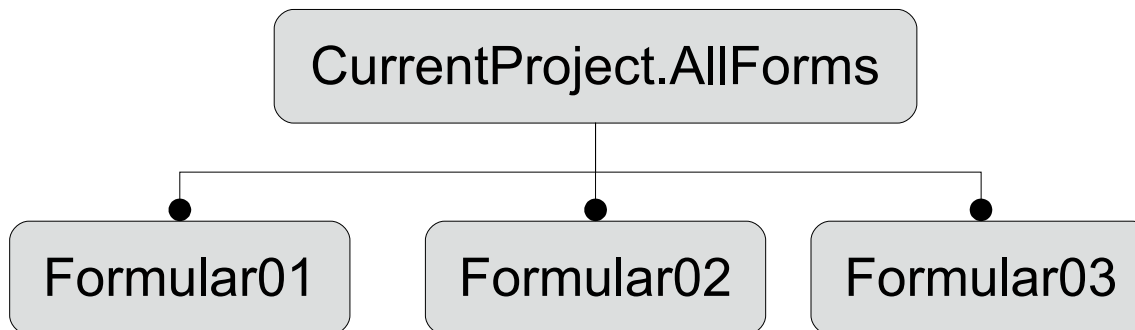
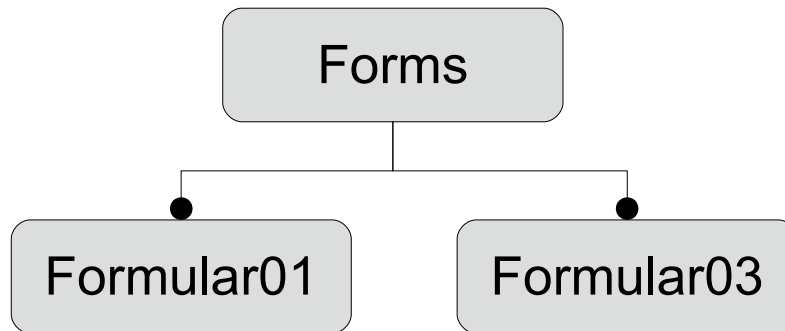
Erläuterung

- « DoCmd.OutputTo » konvertiert ein Access-Objekt in ein anderes Format.
- Folgende Angaben werden benötigt:
 - Welche Objektart wird konvertiert? Die verschiedenen Objektarten werden mit Hilfe von Intellisense oder in der Hilfe angezeigt.
 - Welches Access-Objekt wird transferiert?
 - In welche Datei werden die Daten transferiert?
 - In welches Format wird transferiert?
- Weitere Argumente werden in der Hilfe erläutert.

Informationen zu acFormat

- Die Konstante « acFormat » legt das Ausgabeformat fest.
- Das Format « acFormatPDF » ist seit Access 2007 implementiert.
- Möglichkeiten:
 - <http://msdn.microsoft.com/en-us/library/bb238050%28v=office.12%29.aspx>
 - <http://www.fmsinc.com/MicrosoftAccess/Email/SendObject.html>
 - Der Begriff „acFormat“ wird im Objektkatalog gesucht.

Objekt „Formular“



Formulare ...

- sind elektronische Eingabemasken.
- fragen Informationen vom Benutzer zu einem bestimmten Thema ab.
- zeigen Datensätze in einer für den Benutzer leicht lesbaren Form an.
- steuern die Eingabe von Daten.
- können verschachtelt werden.
- haben die Eigenschaft « .RecordSource » haben. Mit Hilfe dieser Eigenschaft wird das Formular an eine Tabelle oder Abfrage gebunden.

Mit Formulare in VBA arbeiten

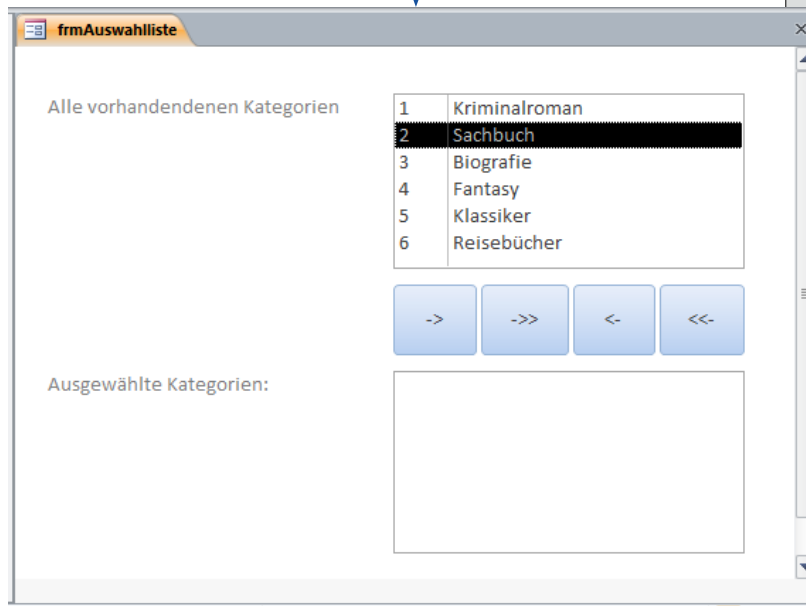
- Mit Hilfe von « DoCmd » können Formulare geöffnet und geschlossen werden.
- Eigenschaften von Formularen können mit Hilfe einer Anweisung gesetzt werden.
- Definierte Methoden können Eigenschaften des Formulars verändern.
- Auf Ereignisse kann der Entwickler reagieren, muss aber nicht.

Zugriff auf geöffnete Formulare

- « Forms!frmMitarbeiter ». Die Auflistung « Forms » enthält alle momentan geöffneten Formular im aktuellen Projekt. Nach dem Objekt « Forms » folgt der gewünschte Formularname. Das Objekt und der Name werden durch ein Ausrufezeichen getrennt.
- « Me » bezieht sich immer auf das zu dem Code gehörende Formular.

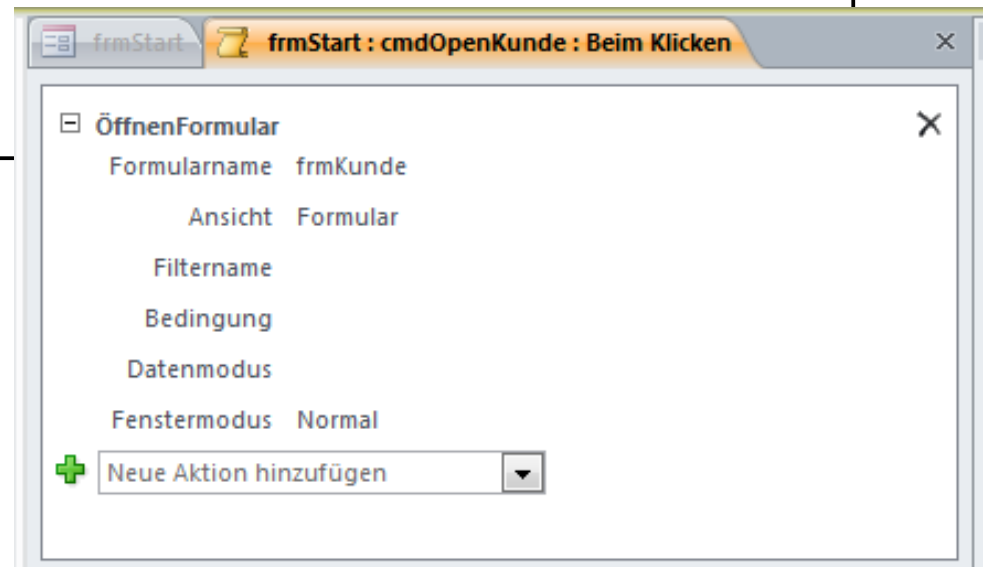
Beispiel für die Nutzung von Me

```
Sub ListeLeeren(listName As String)
  With Me.Controls(listName)
    Do While .ListCount > 0
      .RemoveItem (.ListCount - 1)
    Loop
  End With
End Sub
```



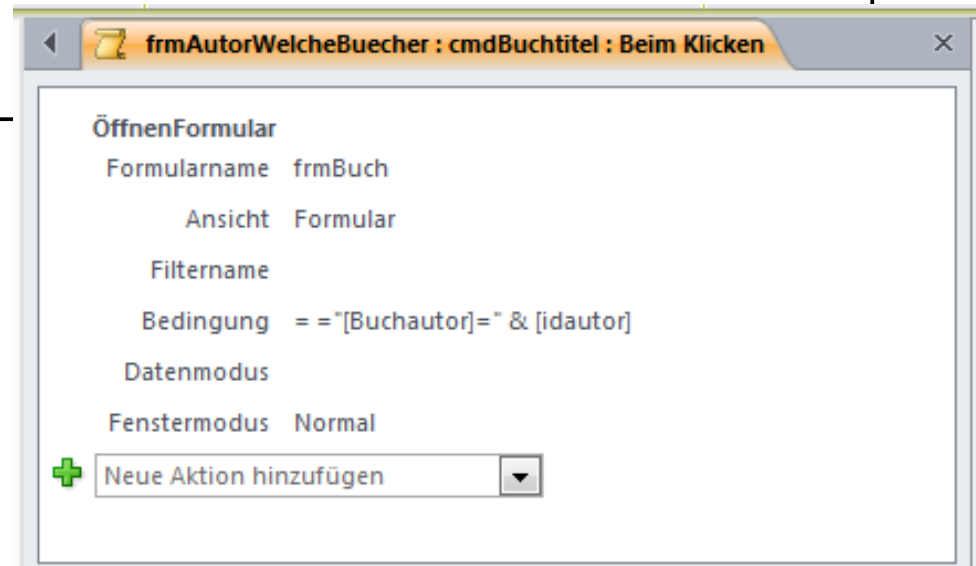
Formular öffnen

```
Private Sub cmdOpenKunde_Click()  
    DoCmd.OpenForm FormName:="frmKunde", _  
        View:=acNormal, _  
        DataMode:=acFormReadOnly, _  
        WindowMode:=acWindowNormal  
  
End Sub
```



Formular mit Hilfe einer Bedingung öffnen

```
Private Sub cmdBuchtitelVBA_Click()  
    DoCmd.OpenForm FormName:="frmBuch", _  
        View:=acNormal, _  
        DataMode:=acFormReadOnly, _  
        WhereCondition:="[Buchautor]=" & [idautor], _  
        WindowMode:=acWindowNormal  
  
End Sub
```



Leeres Formular schließen

```
Private Sub Form_Load()  
    Dim rs As dao.Recordset  
  
    Set rs = Me.Recordset  
  
    If (rs.BOF And rs.EOF) Then  
        DoCmd.Close ObjectType:=acForm, _  
            ObjectName:=Me.Name, _  
            Save:=acSaveNo  
        ' Formular ist leer  
        ' und schließen  
    End If  
End Sub
```

Zugriff auf geschlossene Formulare

- « CurrentProject.AllForms!frmMitarbeiter ».
- Die Auflistung « AllForms » enthält alle Formulare im aktuellen Projekt.
- Nach dem Objekt « AllForms » folgt der gewünschte Formularname.
- Das Objekt und der Name werden durch ein Ausrufezeichen getrennt.

Formulare und deren Daten aktualisieren

- « Me.Refresh » aktualisiert die angezeigten Datensätze. Nach Ausführung der Methode werden alle Änderungen an den Datensätzen angezeigt.
- « Me.Requery » fragt die Datenquelle des Formulars neu ab. Nach der Ausführung werden im Formular alle Änderungen an den Daten sowie die neuen Datensätze angezeigt. Datensätze, die als gelöscht markiert sind, werden aus der Anzeige entfernt.
- « Me.Recalc » aktualisiert alle berechnenden Steuerelemente in einem Formular neu.
- « Me.Repaint » zeichnet das Formular neu.

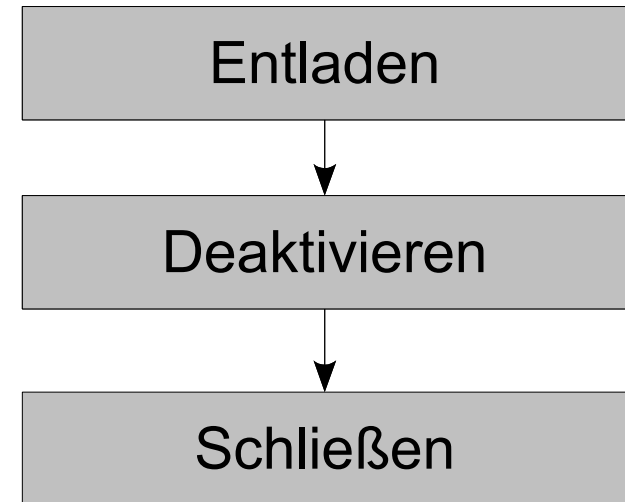
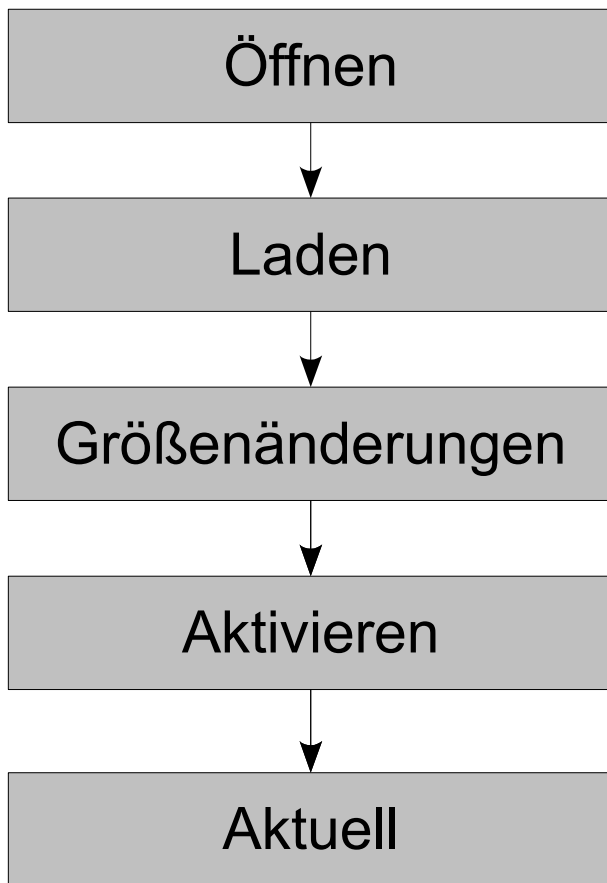
Ereignisprozeduren für ein Formular erstellen

- Das Formular ist in der Entwurfsansicht geöffnet.
- Das Eigenschaftenblatt ist geöffnet und das Register Ereignis ist eingeblendet.
- Mit Hilfe des Pfeils nach unten am rechten Rand des gewünschten Ereignisses wird eine Liste geöffnet. Aus der Liste wird das Element [Ereignisprozedur] ausgewählt.
- Mit Hilfe der Schaltfläche *Drei Punkte* wird der VBA-Editor geöffnet. Passend zum Ereignis wird eine Prozedur automatisch generiert. Die Anweisungen innerhalb der Prozedur muss der Entwickler eingeben.

Ereignisse für Formulare

Ereignis	Erläuterung
Beim Laden	Das Formular wird erstmalig geöffnet. Aber das Formular ist für den Benutzer nicht sichtbar. In dieser Prozedur können Standardwerte oder Eigenschaften gesetzt werden.
Bei Entladen	Das Formular wird aus dem Speicher entfernt. In dieser Prozedur können Daten validiert werden.
Beim Anzeigen	Ein neuer Datensatz wird angezeigt.
Bei Aktivierung	Das Formular erhält den Focus. Das Formular wird in den Vordergrund geschoben.
Bei Deaktivierung	Das Formular verliert den Focus. Das Formular wird in den Hintergrund geschoben.

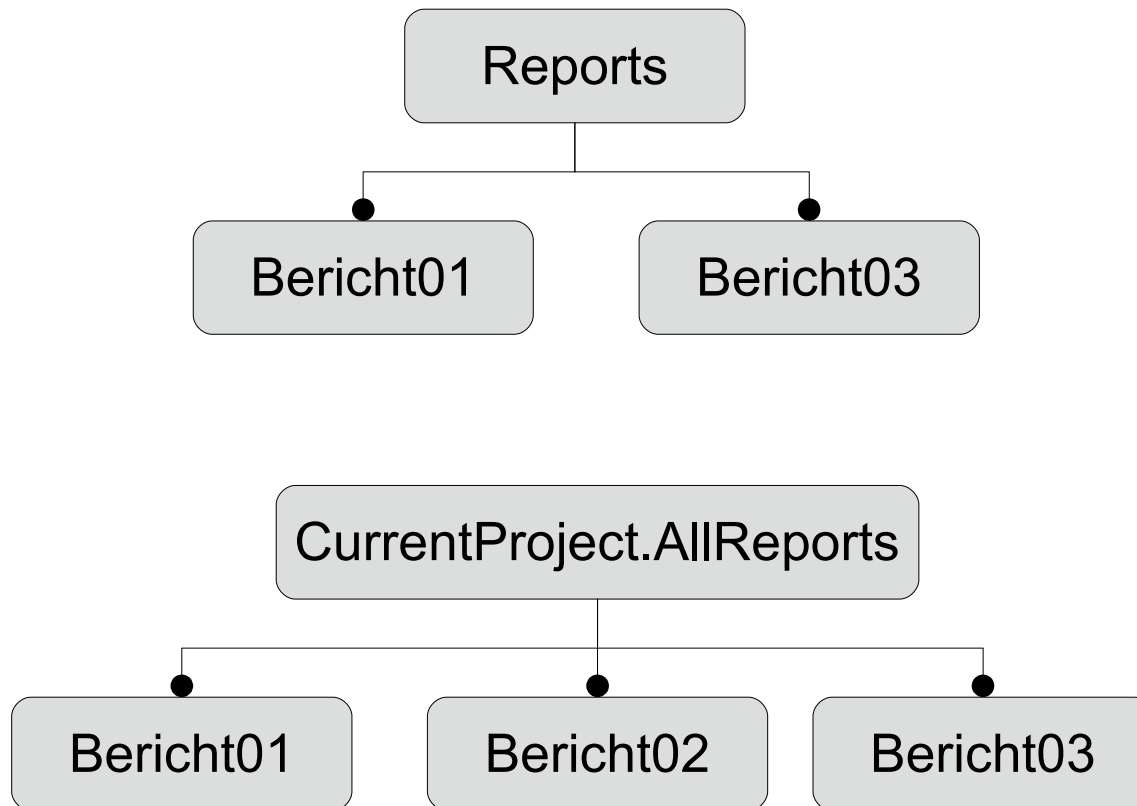
Reihenfolge beim Öffnen und Schließen



Beispiel: On Load

```
Private Sub Form_Load()  
    Me.dateKundeSeit.ForeColor = RGB(0, 0, 0) ' Schriftfarbe  
  
    Me.lblBankname.Visible = False           ' Sichtbar?  
    Me.lblBankleitzahl.Visible = False  
    Me.lblKontonummer.Visible = False  
    Me.txtBankname.Visible = False  
    Me.txtBankleitzahl.Visible = False  
    Me.txtKontonummer.Visible = False  
End Sub
```

Objekt „Bericht“



Berichte ...

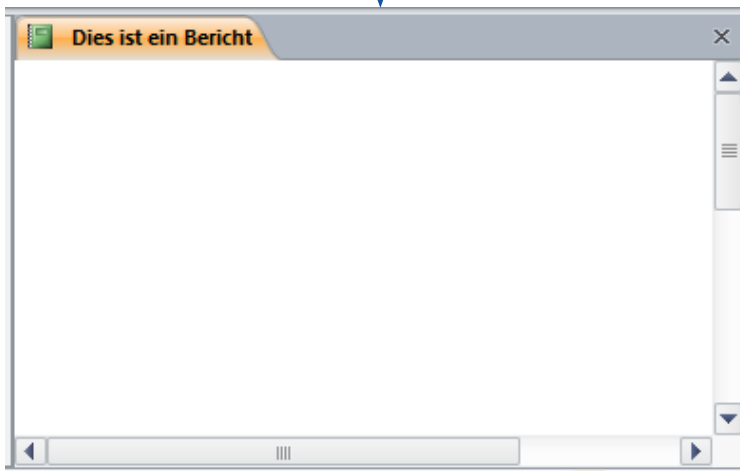
- zeigen Datensätze in einer für den Benutzer lesbaren Form an.
- werden für den Ausdruck von Daten genutzt.
- können für die Präsentation von Daten genutzt werden.
- haben als Grundlage immer eine Tabelle oder Abfrage.
- können verschachtelt werden.
- haben die Eigenschaft « .RecordSource » haben. Mit Hilfe dieser Eigenschaft wird das Formular an eine Tabelle oder Abfrage gebunden.

Zugriff auf geöffnete Berichte

- « Reports!IstMitarbeiter ». Die Auflistung « Reports » enthält alle momentan geöffneten Berichte im aktuellen Projekt. Nach dem Objekt « Reports » folgt der gewünschte Berichtsname. Das Objekt und der Name werden durch ein Ausrufezeichen getrennt.
- « Me » bezieht sich immer auf das zu dem Code gehörende Formular.

Beispiel für die Nutzung von Me

```
Private Sub Report_Load()  
    Me.Caption = "Dies ist ein Bericht"  
    Reports!rep_Objekt.Requery  
End Sub
```



Zugriff auf geschlossene Berichte

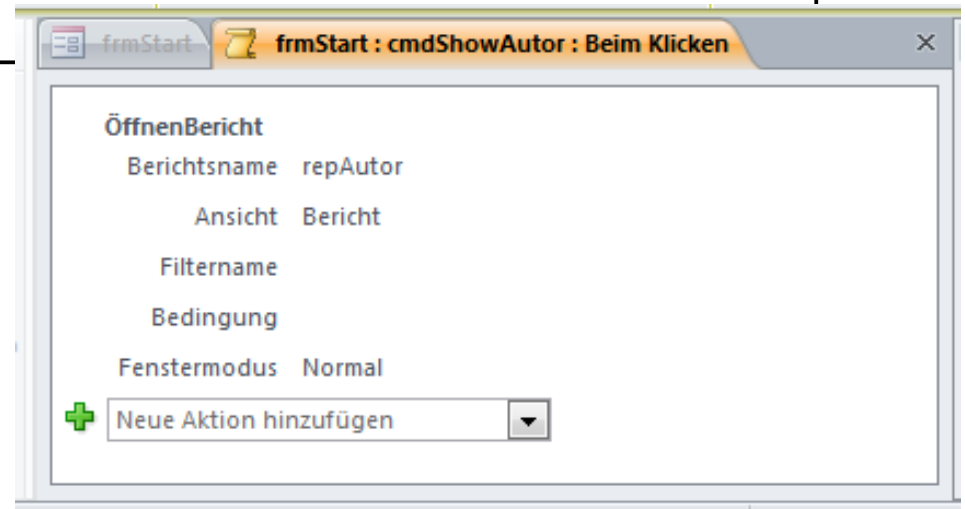
- « CurrentProject.AllReports!IstMitarbeiter ».
- Die Auflistung « AllReports» enthält alle Berichte im aktuellen Projekt.
- Nach dem Objekt « AllReports » folgt der gewünschte Berichtsname.
- Das Objekt und der Name werden durch ein Ausrufezeichen getrennt.

Bereiche eines Berichts

- « Reports!IstMitarbeiter.Section(acDetail) » ermöglicht den Zugriff auf Eigenschaften und Methoden des Detailbereichs.
- « Reports!IstMitarbeiter.Section(acHeader) » beschreibt den Berichtskopf.
- « Reports!IstMitarbeiter.Section(acFooter) » beschreibt den Berichtsfuß.
- « Reports!IstMitarbeiter.Section(acPageHeader) » beschreibt den Seitenkopf eines Berichts.
- « Reports!IstMitarbeiter.Section(acPageFooter) » beschreibt den Seitenfuß eines Berichts.

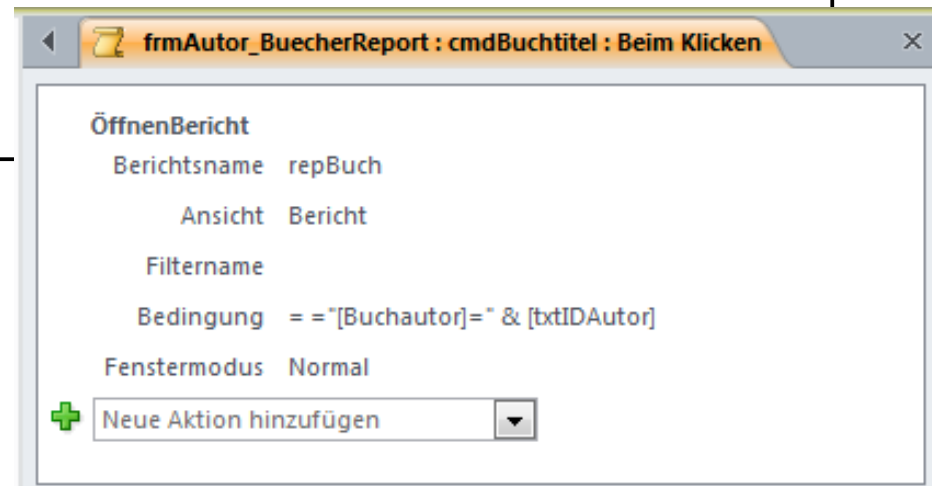
Bericht öffnen

```
Private Sub cmdShowAutorVBA_Click()  
    DoCmd.OpenReport ReportName:="repAutor", _  
        View:=acViewReport, _  
        WindowMode:=acWindowNormal  
  
End Sub
```



Bericht mit Hilfe einer Bedingung öffnen

```
Private Sub cmdBuchtitelVBA_Click()  
    DoCmd.OpenReport ReportName:="repBuch", _  
        View:=acViewReport, _  
        WindowMode:=acWindowNormal, _  
        WhereCondition:="[Buchautor]=" & [txtIDAutor]  
End Sub
```



Leeren Bericht schließen

```
Private Sub Report_NoData(Cancel As Integer)
    DoCmd.Close ObjectType:=acReport, _
        ObjectName:=Me.Name, _
        Save:=acSaveNo
End Sub
```

Ereignisprozeduren für ein Bericht nutzen

- Der Bericht ist in der Entwurfsansicht geöffnet.
- Das Eigenschaftensblatt ist geöffnet und das Register Ereignis ist eingeblendet.
- Mit Hilfe des Pfeils nach unten am rechten Rand des gewünschten Ereignisses wird eine Liste geöffnet. Aus der Liste wird das Element [Ereignisprozedur] ausgewählt.
- Mit Hilfe der Schaltfläche *Drei Punkte* wird der VBA-Editor geöffnet. Passend zum Ereignis wird eine Prozedur automatisch generiert. Die Anweisungen innerhalb der Prozedur muss der Entwickler eingeben.

Ereignisse für Berichte

Ereignis	Erläuterung
Beim Laden	Der Bericht wird erstmalig geöffnet.
Beim Schließen	Der Bericht wird geschlossen. Das Ereignis kann rückgängig gemacht werden.
Beim Anzeigen	Ein neuer Datensatz wird angezeigt.
Bei Aktivierung	Der Bericht erhält den Focus. Der Bericht wird in den Vordergrund geschoben.
Bei Deaktivierung	Der Bericht verliert den Focus. Der Bericht wird in den Hintergrund geschoben.
Bei Ohne Daten	Der Bericht enthält keine Daten.

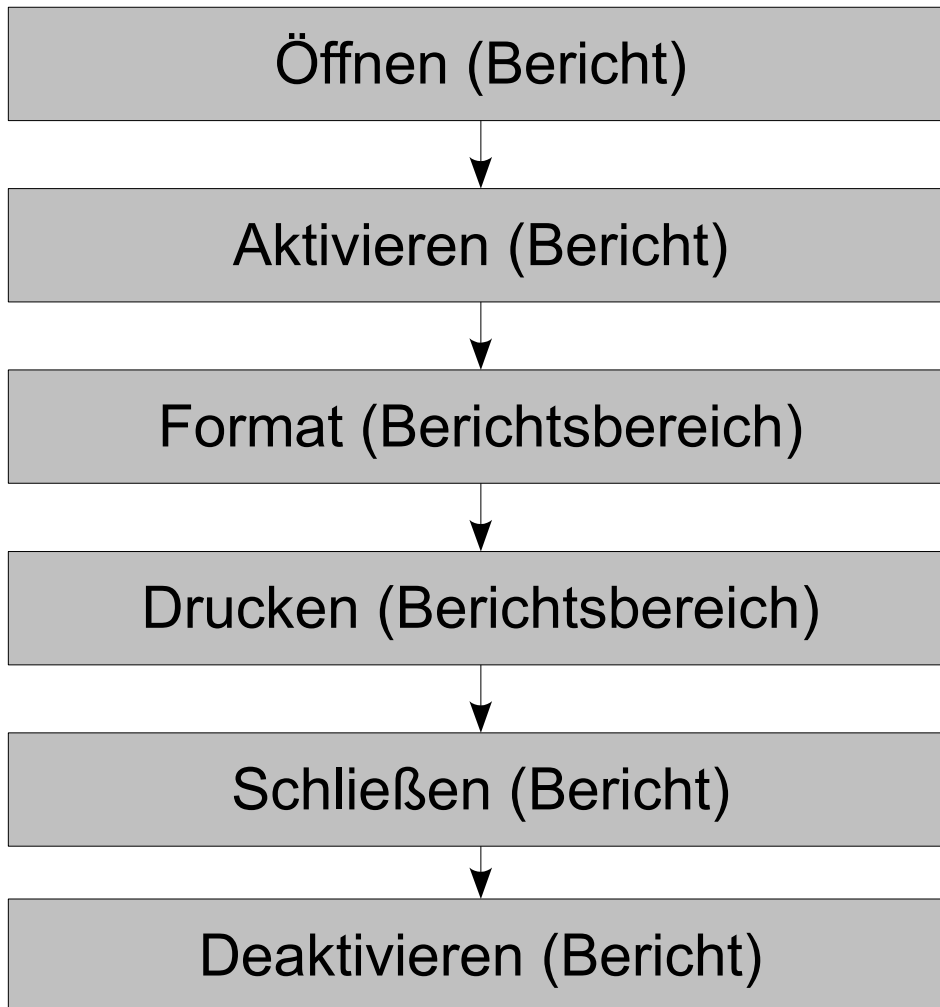
Beispiel: On Unload

```
Private Sub Report_Unload(Cancel As Integer)
    Dim result As VbMsgBoxResult

    result = MsgBox("Möchten Sie den Bericht schließen?", vbYesNo)

    If result = vbNo Then
        Cancel = True      ' Ereignis wird abgebrochen
    End If
End Sub
```

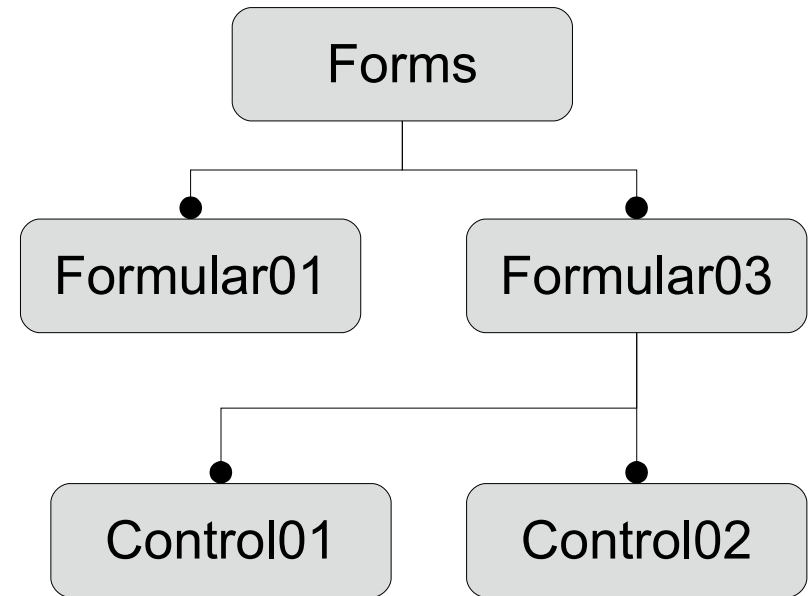
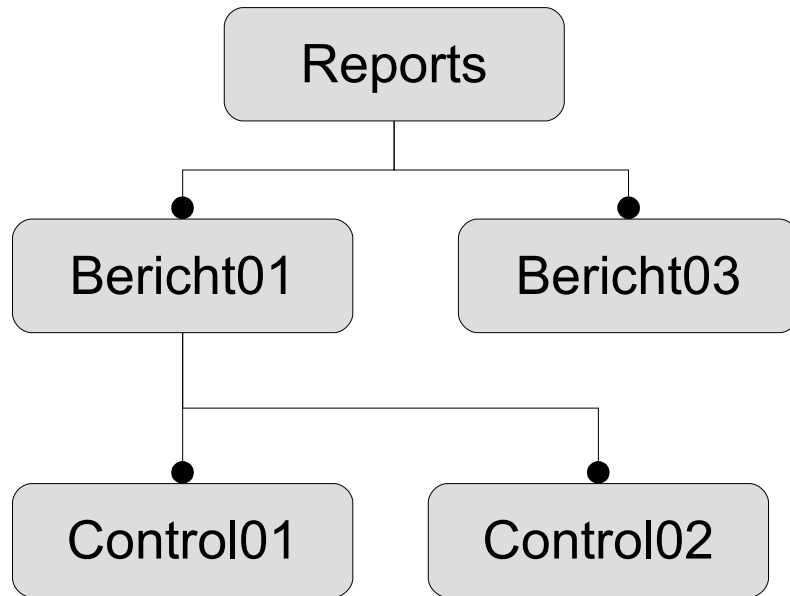
Reihenfolge beim Drucken



Bericht drucken

```
Private Sub cmdPrintAutorVBA_Click()  
    On Error GoTo ErrorHandler  
  
    DoCmd.OpenReport "repAutor", acViewPreview  
    DoCmd.RunCommand acCmdPrint  
  
ErrorHandler:  
    If Err.Number <> 0 And Err.Number <> 2501 Then  
        MsgBox "Error: " & Err.Number & vbNewLine & Err.Description  
        Exit Sub  
    End If  
  
End Sub
```

Objekt „Steuerelement“



Steuerelemente (Controls) ...

- befinden sich immer innerhalb von Formularen oder Berichten.
- können Daten aus der Datenquelle des Formulars oder Berichts anzeigen.
- können Aktionen starten.
- können ein Formular oder Bericht grafisch gestalten.

Steuerelemente auf einem Bericht / Formular

- « Reports!IstMitarbeiter!txtNachname » bezieht sich auf ein Textfeld auf dem Bericht „IstMitarbeiter“.
- « Forms!frmMitarbeiter!IstAbteilung » bezieht sich auf ein Listenfeld auf dem Formular „frmMitarbeiter“.
- « Me!IstAbteilung » bezieht sich auf ein Listenfeld in dem aktuellen Objekt. Das aktuell zu dem Code gehörende Objekt kann ein Formular oder Bericht sein. « Me » kann auch weggelassen werden.

Hinweise

- Der Formular- oder Berichtsname wird von dem Steuerelementnamen durch ein Ausrufezeichen getrennt.
- Falls der Steuerelementname Leer- oder Sonderzeichen enthält, muss dieser in eckige Klammern geschrieben werden.
Beispiel: « Reports!IstMitarbeiter![Vorname Mitarbeiter]».

Die Auflistung Controls nutzen

- « Forms!frmMitarbeiter.Controls!IstAbteilung » ist ein Verweis auf das Steuerelement „IstAbteilung“ in dem Formular „frmMitarbeiter“.
- « Forms!frmMitarbeiter.Controls("IstAbteilung") » nutzt als Index den Namen des Steuerelements,
- « .Controls » ist eine Auflistung, die alle Steuerelemente in einem Formular oder Bericht enthält.

Auflistung (Collection) ...

- sind Controls (Steuerelemente), Forms (Formulare) oder Reports (Berichte)
- haben Namen, die mit einem „s“ enden.
- sind eine Liste von Objekten. Jedes Element in der Liste hat ein Index von 0 bis n.
- entsprechen einem dynamischen Feld.

IstKategorie
IstAuswahl
cmdAuswahlAlle
cmdAuswahl
cmdRuecksetzen

Index in einer Collection

- gibt die Position eines Elements innerhalb der Liste an.
- verändert sich in Abhängigkeit der Anzahl der Elemente in der Liste.
- kann eine Ganzzahl in Abhängigkeit der Position des Elements sein.
- kann ein String sein. Der Name des Elements kann für den Zugriff auf das einzelne Objekt genutzt werden.

IstKategorie
IstAuswahl
cmdAuswahlAlle
cmdAuswahl
cmdRuecksetzen

Variablen als Index nutzen

- « Me.Controls("IstKategorie") ». Der Aufzählung wird der Name eines Steuerelements als Strings fest übergeben. In diesem Beispiel wird das Steuerelement „IstKategorie“ in dem zu dem Code gehörenden Formular / Bericht angesprochen.
- « Me.Controls(listName) ». Als Index wird in diesem Beispiel eine String-Variable genutzt werden. Die Variable ist ein Platzhalter für jeden vorhandenen Steuerelementnamen in dem aktuellen Formular oder Bericht.

Punkt oder Trennzeichen?

- Punkt als Trennzeichen:
 - Vor Auflistungen.
 - Vor Eigenschaften oder Methoden.
 - Dem Trennzeichen folgt ein in Access fix eingebautes Element wie zum Beispiel « .Controls ».
- Ausrufezeichen als Trennzeichen:
 - Vor Objekten.
 - Dem Trennzeichen folgt ein benutzerdefinierter Name wie zum Beispiel « !IstAbteilung ».
 - Intellisense wird nicht angeboten.

Ereignis im Eigenschaftenblatt nutzen

- Das Formular / Bericht ist in der Entwurfsansicht geöffnet.
- Das zu bearbeitende Steuerelement ist aktiv.
- Das Eigenschaftenblatt ist geöffnet und das Register Ereignis ist eingeblendet.
- Mit Hilfe des Pfeils nach unten am rechten Rand des gewünschten Ereignisses wird eine Liste geöffnet. Aus der Liste wird das Element [Ereignisprozedur] ausgewählt.
- Mit Hilfe der Schaltfläche *Drei Punkte* wird der VBA-Editor geöffnet. Passend zum Ereignis wird eine Prozedur automatisch generiert. Die Anweisungen innerhalb der Prozedur muss der Entwickler eingeben.

Ereignisse im Codebereich nutzen

- Im Codebereich des VBA-Editors wird das gewünschte Steuerelement mit Hilfe der Liste *Objekt* ausgewählt.
- Anschließend wird automatisch das Gerüst der dazugehörigen Standard-Ereignisprozedur im Code-Bereich eingefügt.
- Mit Hilfe der Liste *Prozedur* kann aber jedes andere mögliche Ereignis ausgewählt werden.
- Nicht gewünschte Ereignisprozeduren werden markiert und gelöscht.

Ereignisse für alle Steuerelemente

Ereignis	Steuerelemente	Erläuterung
Beim Hineingehen	alle	Das Steuerelement erhält den Focus.
Beim Verlassen	alle	Das Steuerelement verliert den Focus.
Bei Taste	alle	Eine Taste wird gedrückt.
Beim Klicken	Schaltflächen	Mit der linken Maustaste wird auf die Schaltfläche geklickt.
Bei nicht in Liste	Kombinationsfeld	Der eingegebene Wert ist nicht in der Liste vorhanden.

Ereignisse für Datensätze

Ereignis	Formular	Textfeld	Erläuterung
Bei Geändert	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Der Datensatz in einem Formular oder Textfeld ändert sich.
Bei Rückgängig	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Beim Anzeigen	<input checked="" type="checkbox"/>		
Vor Löschbestätigung Nach Löschbestätigung	<input checked="" type="checkbox"/>		Der Datensatz wird gelöscht.
Vor Eingabe Nach Einfügung	<input checked="" type="checkbox"/>		Ein neuer Datensatz wird eingefügt.
Vor Aktualisierung Nach Aktualisierung	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Nach einer Änderung werden die Daten aktualisiert.

Beispiel: Taste gedrückt

```
Private Sub txtOperandLinks_KeyPress(KeyAscii As Integer)

    ' Zahlen 0 (Ascii-Code 48) bis 9 (Key-Ascii-Code 57)
    If (KeyAscii < 48) Or (KeyAscii > 57) Then
        KeyAscii = 0
    End If

End Sub
```

Reihenfolge „Drücken einer Taste“



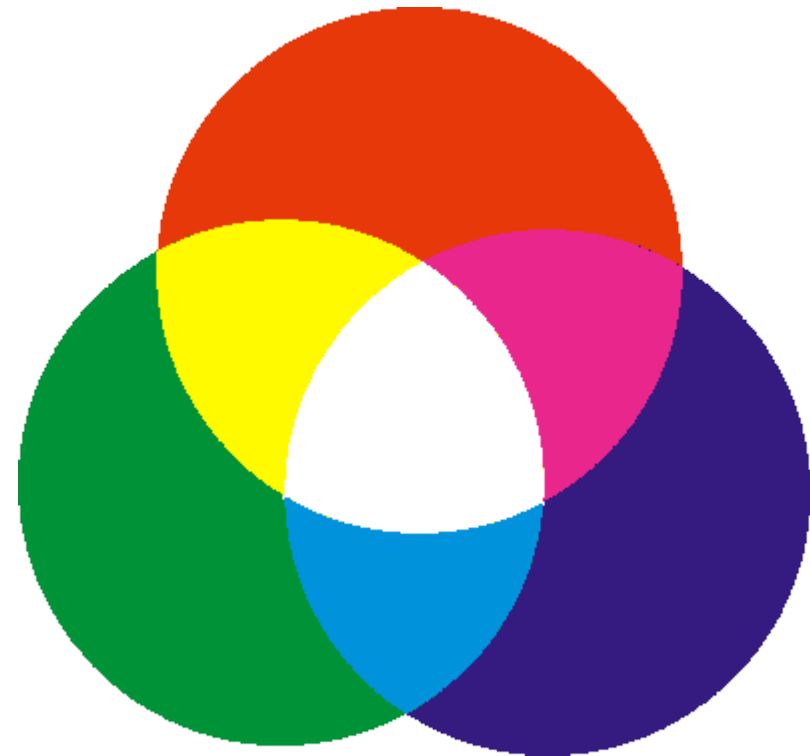
Farben für Eigenschaften angeben

```
Private Sub Form_Load()  
    Me.dateKundeSeit.ForeColor = RGB(0, 0, 0)  
End Sub
```

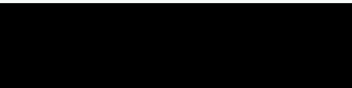







- Die Funktion RGB() mischt eine Farbe für den Bildschirm.
- Der Funktion wird der Rotanteil, der Grün-Anteil und der Blau-Anteil der Farbe übergeben.
- Farben werden zum Beispiel für die Eigenschaften « .ForeColor » und « .BackColor » genutzt.

RGB-Farben

- ... werden für die Darstellung von Farben am Bildschirm genutzt.
- Das RGB-Farbsystem addiert (mischt) Licht in verschiedenen Farben.
- Jede Farbe (Rot, Grün, Blau) wird in 256 Helligkeitsstufen unterteilt.
- Um so mehr sich eine Farbe Weiß annähert, um so heller wird diese.



Beispiele

Farbe	RGB	Farbkonstante
	(0, 0, 0)	vbBlack
	(255, 255, 255)	vbWhite
	(255, 0, 0)	vbRed
	(0, 255, 0)	vbGreen
	(0, 0, 255)	vbBlue
	(255, 255, 0)	vbYellow
	(255, 0, 255)	vbMagenta
	(0, 255, 255)	vbCyan

Angezeigter und vorheriger Wert

- Die Eigenschaft « .Value » liefert bei ...
 - einem Textfeld einen String.
 - einer Optionsgruppe den Wert der gewählten Option.
 - einem Kontrollkästchen einen booleschen Wert.
 - einem Listenfeld die gebundene Spalte der markierten Zeile.
- Die Eigenschaft « .OldValue » liefert den letzten gespeicherten Wert eines gebundenen Textfeldes.

Beispiel

```
Private Sub dateKundeSeit_AfterUpdate()  
    Const fehlermeldung As String = "Das Eintrittsdatum liegt in der Zukunft."  
    Dim result As VbMsgBoxResult  
  
    If dateKundeSeit.Value > Now Then  
        result = MsgBox(fehlermeldung, vbOKOnly + vbInformation, _  
            "Hinweis auf eine falsche Eingabe")  
  
        Me.dateKundeSeit.ForeColor = RGB(255, 0, 0)  
        Me.dateKundeSeit.SetFocus  
  
    End If  
End Sub
```

Steuerelemente ein- und ausblenden

```
Private Sub fraZahlung_AfterUpdate()  
  Select Case fraZahlung.Value  
    Case 1:  
      Me.lblBankleitzahl.Visible = True  
      Me.lblKontonummer.Visible = True  
      Me.txtBankleitzahl.Visible = True  
      Me.txtKontonummer.Visible = True  
    Case Else:  
      Me.lblBankleitzahl.Visible = False  
      Me.lblKontonummer.Visible = False  
      Me.txtBankleitzahl.Visible = False  
      Me.txtKontonummer.Visible = False  
  End Select  
End Sub
```

Textfeld (Textbox) ...

- zeigt Werte ...
 - in Abhängigkeit eines bestimmten Datenfeldes an.
 - an, der mit Hilfe von VBA berechnet oder zugewiesen wurden.
- hat die Eigenschaft « .Value ». Diese Eigenschaft gibt immer einen String zurück.
- hat die Eigenschaft « .Text ». Der Wert kann nur abgefragt werden, wenn das Textfeld aktiv ist.
- kann an ein Feld in einer Tabelle oder Abfrage gebunden (Eigenschaft: « .ControlSource ») sein.

Textfelder sperren

```
Private Sub fraZahlung_AfterUpdate()  
    Select Case fraZahlung.Value  
        Case 1:  
            txtBankleitzahl.Locked = False           ' Nicht gesperrt  
            txtKontonummer.Locked = False  
            txtBankname.Locked = False  
        Case 2 To 3:  
            txtBankleitzahl.Locked = True           ' Gesperrt  
            txtKontonummer.Locked = True  
            txtBankname.Locked = True  
    End Select  
End Sub
```

Textfelder aktivieren

```
Private Sub fraZahlung_AfterUpdate()  
    Select Case fraZahlung.Value  
        Case 1:  
            txtBankleitzahl.Enabled = True           ' Aktivieren  
            txtKontonummer.Enabled = True  
            txtBankname.Enabled = True  
        Case 2 To 3:  
            txtBankleitzahl.Enabled = False         ' Deaktivieren  
            txtKontonummer.Enabled = False  
            txtBankname.Enabled = False  
    End Select  
End Sub
```

Optionsgruppe ...

- ist eine Sammlung von Optionsfeldern zu einem bestimmten Thema.
- hat die Eigenschaft « .Value », die einen ganzzahligen Wert zurückgibt. Diese Eigenschaft identifiziert eindeutig ein Optionsfeld in der Optionsgruppe. Der zurückgegebene Wert wird in der Eigenschaft « .OptionValue » des Optionsfeldes definiert.
- kann an ein Feld in einer Tabelle oder Abfrage gebunden (Eigenschaft: « .ControlSource ») sein.

Kontrollkästchen ...

- bieten eine Mehrfachauswahl zu einem Thema.
- hat die Eigenschaft « .Value », die einen booleschen Wert zurückgibt. Der Wert „True“ symbolisiert ein aktives Kontrollkästchen. Der Wert „False“ symbolisiert ein nicht aktives Kontrollkästchen.
- kann an ein Feld in einer Tabelle oder Abfrage gebunden (Eigenschaft: « .ControlSource ») sein.

Listenfelder ...

- bieten eine Liste von Möglichkeiten an.
- Der Benutzer wählt mit Hilfe der linken Maustaste ein oder mehrere Elemente aus der Liste aus.
- Standardmäßig kann aus der Liste nur ein Element ausgewählt werden.

Anzahl der Spalten und Zeilen

- Die Eigenschaft « [Liste].ColumnCount » gibt die Anzahl der Spalten zurück oder setzt diese.
- Die Eigenschaft « [Liste].ListCount » gibt die Anzahl der Zeilen zurück. Die Anzahl der Zeilen ergibt sich aus der Anzahl der Elemente.

Spalten-Eigenschaften definieren

```
With Me.Controls("IstKategorie")
```

```
    .ColumnCount = 2           ' Anzahl der Spalten  
    .ColumnWidths = "1 cm; 5 cm" ' Breite der einzelnen Spalten  
    .BoundColumn = 1         ' Gebundene Spalte
```

```
End With
```

- Die Eigenschaft « .ColumnWidths » ist eine Auflistung von Breiten für jede, in der Liste vorhandenen Spalte. Standardmäßig werden die Spaltenbreiten in Zentimeter angegeben. Die einzelnen Spaltenbreiten werden durch Kommata getrennt.
- Die Eigenschaft « .BoundColumn » legt fest, welcher Spaltenwert der aktuellen Zeile gespeichert wird.

Werte in einer Liste

- Die Eigenschaft « [Liste].Value » gibt den Wert der gebundenen Spalte der markierten Zeile zurück.
- Die Eigenschaft « [Liste].Column(spalte, zeile) » gibt den Wert einer bestimmten Spalte und Zeile zurück.
- Die Eigenschaft « [Liste].ListIndex » gibt die Nummer der gewählten Zeile in einem Listenfeld zurück.
- Die Eigenschaft « [Liste].Column(spalte, [Liste].ListIndex) » gibt den Wert einer bestimmten Spalte in der gewählten Zeile zurück.

Gebundenes Listenfeld ...

- zeigt die Daten aus einem Feld in einer Tabelle oder Abfrage an.
- Die Eigenschaft « .RowSourceType » hat den Wert "Table/Query".
- Die Eigenschaft « .RowSource » zeigt mit Hilfe einer SQL-Anweisung an, woher die angezeigten Daten kommen.

Werteliste ...

- zeigt benutzerdefinierte Werte.
- Die Eigenschaft « .RowSourceType » hat den Wert "Value List".
- Die Eigenschaft « .RowSource » zeigt eine Liste von Werte, getrennt durch Semikolon an.

Elemente einer Werteliste hinzufügen

```
Private Sub cmdClearAuswahl_Click()  
    Dim aktuellesElement As Integer  
    Dim newElement As String  
  
    aktuellesElement = IstKategorieAuswahl.ListIndex  
    newElement = IstKategorieAuswahl.Column(0, aktuellesElement) & ";" &  
        & IstKategorieAuswahl.Column(1, aktuellesElement)  
  
    IstKategorie.AddItem newElement  
  
    Me.Refresh  
End Sub
```

Erläuterung

- Mit Hilfe der Eigenschaft « .AddItem » wird ein Element der Wertliste hinzugefügt.
- « .AddItem Item:=newElement, Index:=0 » fügt das neue Element in der ersten Zeile an. Die Zeilen eines Listenfeldes werden von 0 bis n nummeriert.
- Wenn keine Zeilennummer angegeben ist, wird das neue Element am Ende der Liste angefügt.
- Die Elemente einer Zeile werden als String übergeben. Falls die Liste mehrspaltig ist, werden die Elemente für die einzelnen Spalten durch ein Semikolon getrennt. Das Element des Strings wird der ersten Spalte von links zugeordnet und so weiter.

Löschen von Elementen in einer Werteliste

```
Sub ListeLeeren(listName As String)
  With Me.Controls(listName)

    Do While .ListCount > 0
      .RemoveItem (.ListCount - 1) ' Liste von hinten nach vorne leeren
    Loop

  End With
End Sub
```

Erläuterung

- Mit Hilfe der Eigenschaft « `.RemoveItem(index)` » wird eine bestimmte Zeile aus der Wertliste gelöscht.
- Die Anzahl der Elemente « `.ListCount` » wird nach jeder Löschung neu angepasst.
- Wenn der angegebene Index nicht vorhanden ist, wird eine Fehlermeldung ausgegeben.
- Falls die Liste vollständig gelöscht werden soll, muss die Liste von hinten nach vorn gelöscht werden. Andernfalls kann es zu Fehlern kommen.

Mehrfachauswahl ...

- muss im Eigenschaftenblatt der Liste eingestellt werden.
- wird mit Hilfe der Eigenschaft Mehrfachauswahl auf der Registerkarte *Andere* eingestellt. Die Einstellung Keine ist Standard. Mit Hilfe der linken Maustaste kann nur ein Element aktiviert oder deaktiviert werden.
- irritiert meist die Anwender. Anwender erwarten bei einer Liste die Auswahl von einem Element.

Einstellungsmöglichkeiten

- Einzel. Mit Hilfe der Pfeiltasten nach oben oder unten kann ein Element aus der Liste gewählt werden. Mit Hilfe der Leertaste oder der linken Maustaste kann ein oder mehrere Elemente aktiviert oder deaktiviert werden. In der Hilfe wird diese Einstellung als „Einfach“ bezeichnet.
- Erweitert.
 - Mit Hilfe der linken Maustaste wird ein Element markiert.
 - Mit der linken Maustaste plus der <Umschalt>-Taste wird das letzte Element einer Gruppe aktiviert. Alle dazwischen liegenden Elemente werden automatisch aktiviert.
 - Falls die <STRG>-Taste statt der <Umschalt>-Taste genutzt wird, können nicht zusammenhängende Elemente einer Gruppe aktiviert werden.

Welche Elemente sind in der Liste ausgewählt?

```
For count = 0 To IstOptionen.listCount - 1
    newElement = IstOptionen.Column(0, count) & "; "
    newElement = newElement & IstOptionen.Column(1, count) & "; "
    newElement = newElement & IstOptionen.Column(2, count)

    If IstOptionen.Selected(count) = True Then

End If

Next
```

Erläuterung

- Die Methode « Selected(index) » gibt true (wahr) zurück, wenn die Zeile ausgewählt wurde. Andernfalls gibt die Methode false zurück.
- Hinweis: Durch das Löschen eines Elements in der Werteliste wird die Auswahl der Elemente aufgehoben.

Liste der ausgewählten Elemente

```
Private Sub cmdNoAuswahl_Click()  
    Dim zeile As Variant  
    Dim ausgabe As String  
  
    For Each zeile In IstAuswahl.ItemsSelected  
        ausgabe = IstAuswahl.ItemData(zeile) & ";"  
        ausgabe = ausgabe & IstAuswahl.Column(1, zeile) & ";"  
        ausgabe = ausgabe & IstAuswahl.Column(2, zeile)  
        ausgabe = ausgabe & ";" & IstOptionen.Column(3, zeile)  
        IstOptionen.AddItem ausgabe  
    Next  
  
End Sub
```

Erläuterung

- Die Auflistung « .ItemsSelected » ist eine Liste aller ausgewählten Elemente einer Liste.
- Die Auflistung kann mit Hilfe einer For-Each-Schleife durchlaufen werden.
- Die Methode « ItemData(zeile) » gibt den Inhalt des gebundenen Feldes in der angegebenen Zeile aus.