Mit unseren Übungsaufgaben zu *ACCESS* haben Sie sich bereits umfangreiche Kenntnisse zu dieser *Datenbankanwendung* angeeignet. Sie arbeiten mit *Tabellen, Abfragen, Formularen* und *Berichten.* Mit den Ausführungen in unserer Hilfedatei *Schaltfläche, Makro* haben wir Ihnen außerdem gezeigt, wie Sie *Steuerelementen* (*Schaltflächen* usw.) einfache, automatisierte Programmabläufe, sog. *Makros* zuweisen können.

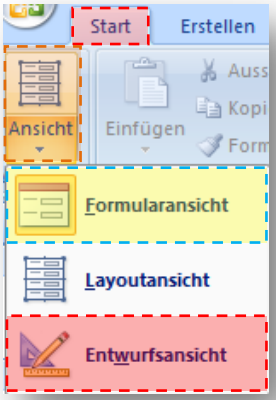

Abb. 1

Sie wissen, dass Sie – je nach *Datenbankobjekt* - zwischen unterschiedlichen Ansichten wechseln können und bei einem *Formular* beispielsweise eine *Formular-, Layout-* und *Entwurfsansicht* (*Abb. 1*) zur Verfügung steht.

In der *Formularansicht* werden die entsprechenden *Datensätze* eingegeben und in der *Entwurfsansicht* die Struktur des Objekts festgelegt und bearbeitet. Dabei beschränken sich die Möglichkeiten aber nicht nur auf das *Formular* selbst, sondern auch auf die *Steuerelemente*, wie z.B. *Textfelder, Kombinationsfelder, Schaltflächen* usw. die in das *Formular* eingefügt werden können.

Neben einer Vielzahl von ***Eigenschaften***[1] können den einzelnen Objekten auch unterschiedliche *Ereignisse* (*Abb. 2 – kleiner Screenshot*) zugeordnet werden - einer *Schaltfläche* beispielsweise das Ereignis *Beim Klicken*.

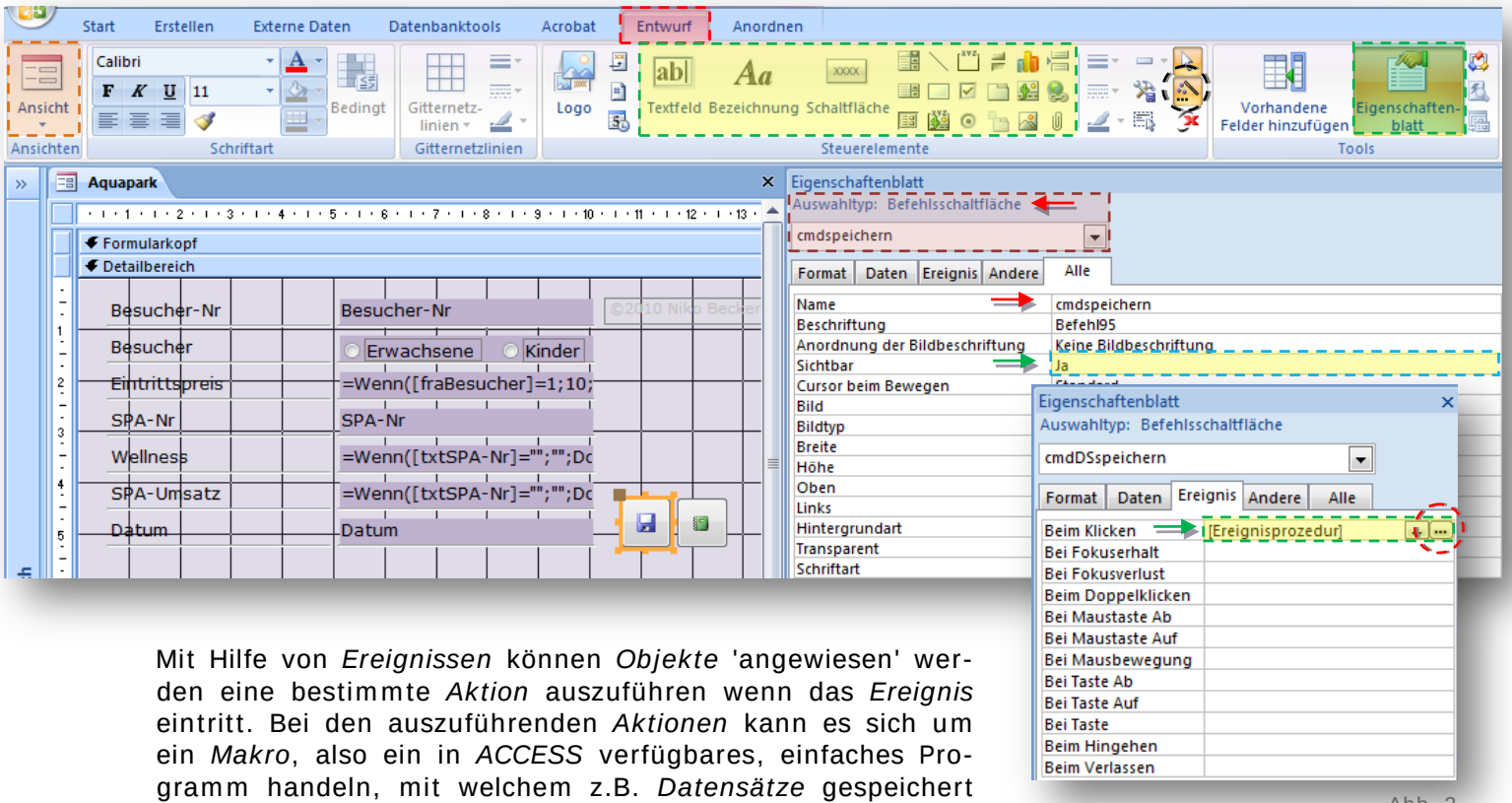

Abb. 2

Mit Hilfe von *Ereignissen* können *Objekte* 'angewiesen' werden eine bestimmte *Aktion* auszuführen wenn das *Ereignis* eintritt. Bei den auszuführenden *Aktionen* kann es sich um ein *Makro*, also ein in *ACCESS* verfügbares, einfaches Programm handeln, mit welchem z.B. *Datensätze* gespeichert oder andere *Formulare* geöffnet werden können, aber auch um eine benutzerdefinierte *Prozedur*, die manuell im *Visual*

---

[1] *...Eigenschaften* beschreiben das *Steuerelement*, hinsichtlich *Größe, Hintergrundfarbe, Textausrichtung, Rahmen* usw. *Eigenschaften* können abgefragt und auch neu festgelegt werden – z.B.:
Schaltfläche.sichtbar = ja/nein oder innerhalb des *Programmcodes* im *Visual Basic-Editor:*

```
Me.cmdspeichern.Visible = True/False
```

Eigenschaften werden keine *Argumente* übergeben.

Basic-Editor (*Tastenkombination Alt+F11*) zu erstellen ist.

In *ACCESS* verfügbare *Makros* können einem *Steuerelement* mit Hilfe des *Makro-Generators* (*Abb. 3*) zugewiesen werden. Dieser kann beispielsweise in Zeile *Beim Klicken* (*Abb. 2 – kleiner Screenshot*), durch Anklicken der drei Punkte am rechten Zeilenrand und Aktivieren des entsprechenden Eintrags *geöffnet werden*. Sie erhalten ein weiteres *Dialogfenster* (*Abb. 4*) angezeigt und können nun in Spalte *Aktion* auswählen, welche Aktion dem *Steuerelement* zugewiesen werden soll. Legen Sie im unteren Fenster-Bereich ggf. noch die entsprechenden *Aktionsargumente,* wie z.B. *Berichtsnamen, Formularnamen* usw. fest und beachten Sie, dass diese *'Standard-Makros'* nicht editiert werden können.

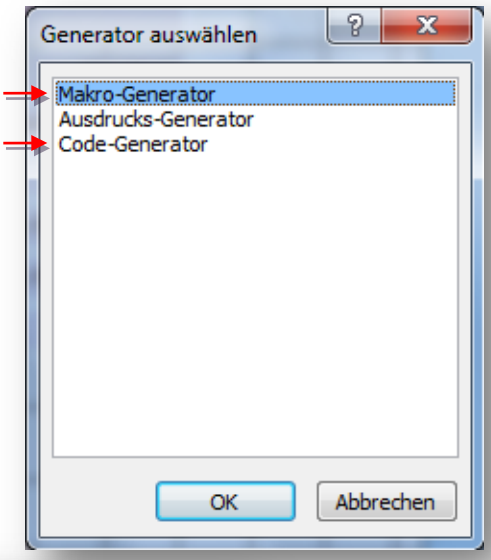

Abb. 3

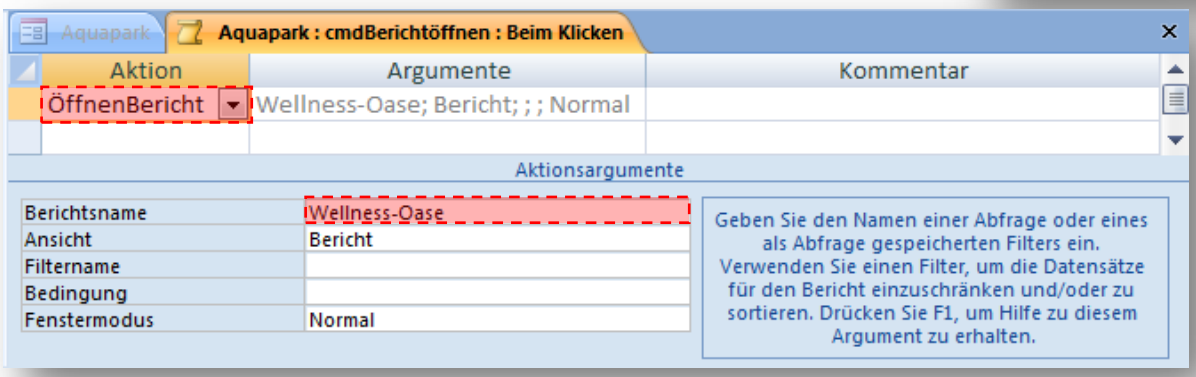

Abb. 4

Neben den vorgefertigten *Makros* kann allen *Steuerelementen* auch individuell erstellter Programmcode – sog. *Prozeduren* – zugewiesen werden. Diese 'automatisierten Programmabläufe' werden im *Visual Basic-Editor* erstellt der öffnet, wenn Sie in *Abb. 3* das Menü *Code-Generator* aktivieren. Alternativ können Sie den *Visual Basic-Editor* auch mit der *Tastenkombination Alt+F11* öffnen oder durch Anklicken des Icons *Visual Basic* in der *Multifunktionsleiste* - Register *Datenbanktools* – Gruppe *Makro,* wenn Sie *ACCESS 2007* nutzen.

Wie Sie dem *Screenshot* des Codefensters (*Abb. 5*) entnehmen können, besteht der *Visual Basic-Editor* aus mehreren *Fenstern* die - je nach Bedarf - im Menü *Ansicht* ein- und ausgeblendet werden können.

Links oben ist der *Projekt-Explorer* angeordnet. Hier wird nicht nur der Name der *Datenbank* angezeigt, sondern auch die angelegten *Formulare, Berichte, Module* und *Klassenmodule* aufgeführt. Die Vorsilben *Form_...* beim *Formular,* bzw. *Report_...* beim *Bericht* werden von *ACCESS* automatisch vorangestellt; *Module* und *Klassenmodule* können durch Anklicken des entsprechenden Eintrags im Menü *Einfügen* in das *Projekt* eingebunden werden.

Unterhalb des *Projekt-Explorers* ist das *Eigenschaften-Fenster* angeordnet. Hier werden alle verfügbaren *Eigenschaften* des jeweiligen Objekts und dessen aktuelle Einstellungen aufgeführt. Die einzelnen Werte können manuell und auch mittels Programmcode geändert werden.

Der große Bereich rechts daneben ist das *Code-Fenster*. Hier kann den einzelnen Objekten und *Ereignissen* der entsprechende Programmcode zugewiesen, und dieser auch editiert werden.

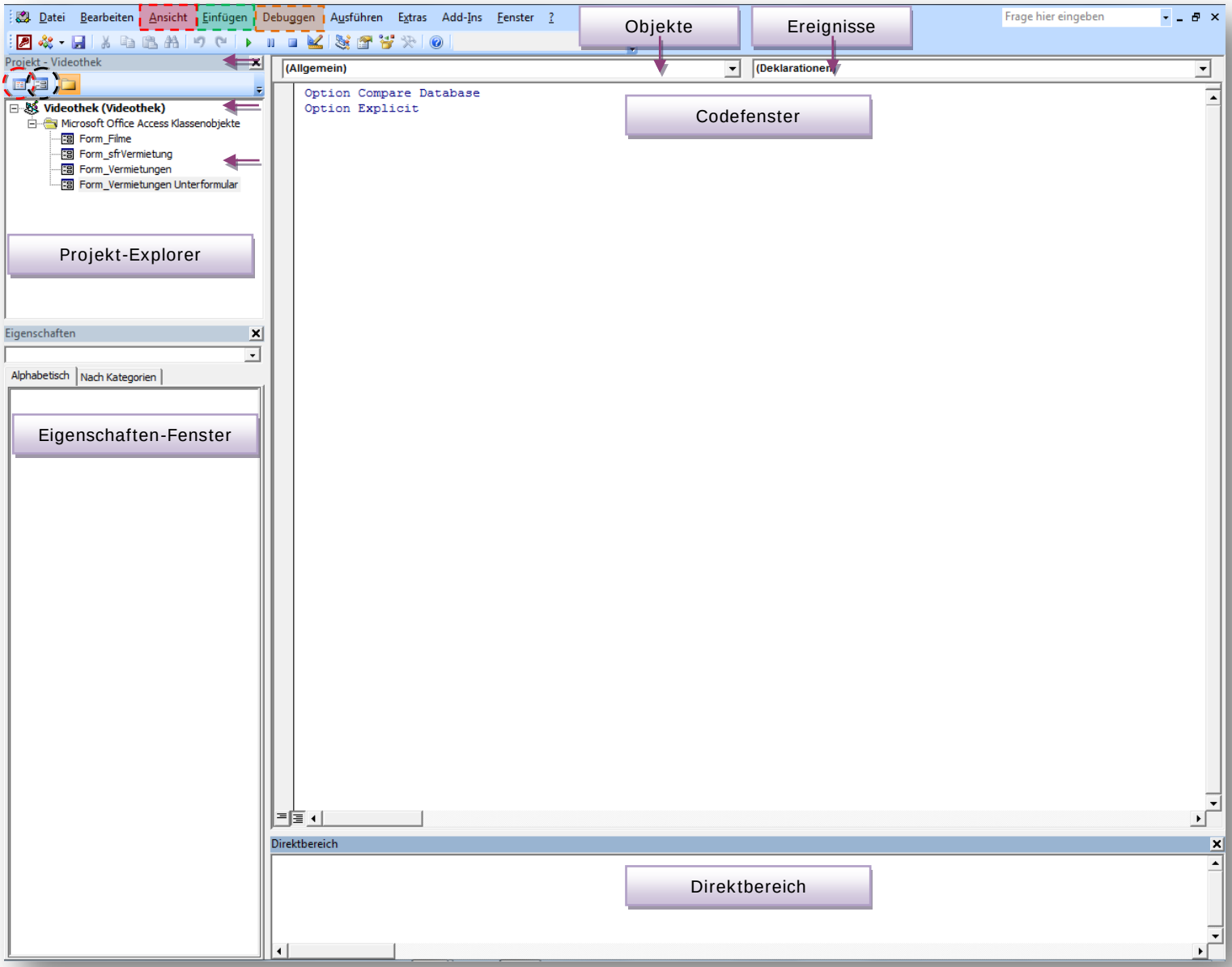

Abb. 5

Unterhalb dieses Fensters ist der *Direktbereich* eingebunden. Hier können Teile des Programmcodes getestet, und auch Befehle eingegeben werden.

Neben den hier dargestellten Bearbeitungsbereichen können im Menü *Ansicht* noch zwei weitere Fenster aktiviert werden: das *Lokalfenster* – zur Anzeige des Inhalts von *Variablen* und das *Überwachungsfenster* – zur Überwachung von *Variablen*.

Am oberen Rand des *Codefensters* befinden sich zwei *Kombinationsfelder*. Im linken *Kombinationsfeld* werden alle Objekte der *Datenbank* angezeigt und im rechten *Kombinationsfeld* die dazugehörenden *Ereignisse.* Durch Aktivieren der entsprechenden Einträge werden Objekt und *Ereignis* in das *Codefenster* eingebunden, damit der übrige Programmcode erstellt werden kann.

Mit Hilfe der beiden *Icons* - links oben - können Sie wahlweise in das *Formular* (*schwarzer* Kreis), bzw. in den *Programmcode* des *Objekts* (*roter* Kreis) wechseln.

Bevor Sie den einzelnen *Objekten* allerdings Programmcode zuweisen, sollten Sie diese eindeutig benennen!

Allen *Steuerelementen* wird zwar von *ACCESS* ein entsprechender Name, z.B. *Bezeichnungsfeld22* oder *Textfeld5* zugewiesen - allerdings wird das einfache Durchnummerieren schnell unübersichtlich wenn Sie mehrere gleiche *Steuerelemente* angelegt haben und diese in einer *Prozedur* 'ansprechen' wollen.

Üblicherweise wird dem Namen des *Steuerelements* zunächst ein *Typkürzel* vorangestellt damit erkennbar ist, um welches Objekt es sich handelt. Das *Präfix* für ein *Textfeld* ist beispielsweise *txt* und für ein Bezeichnungsfeld *lbl* (*Label*). In Verbindung mit einem entsprechenden Namen können die unterschiedlichen Objekte der *Datenbank* eindeutig identifiziert werden.
Ein *Formular* erhält das Präfix *frm* und kann so beispielsweise den Namen *frmFilme* oder *frmVermietungen* erhalten; der Name eines *Textfeldes* zur Eingabe von Filmtiteln kann z.B. *txtTitel* und das dazugehörende Bezeichnungsfeld *lblTitel* lauten usw. Weitere *Typkürzel* können Sie unserer *Online*-Hilfedatei (*Typkürzel für Objekte*) entnehmen.

Beachten Sie beim Erstellen des Programmcodes, dass Sie sich immer auf den *Namen* des Objekts beziehen müssen und nicht auf dessen *Beschriftung* (*Abb. 6*)!

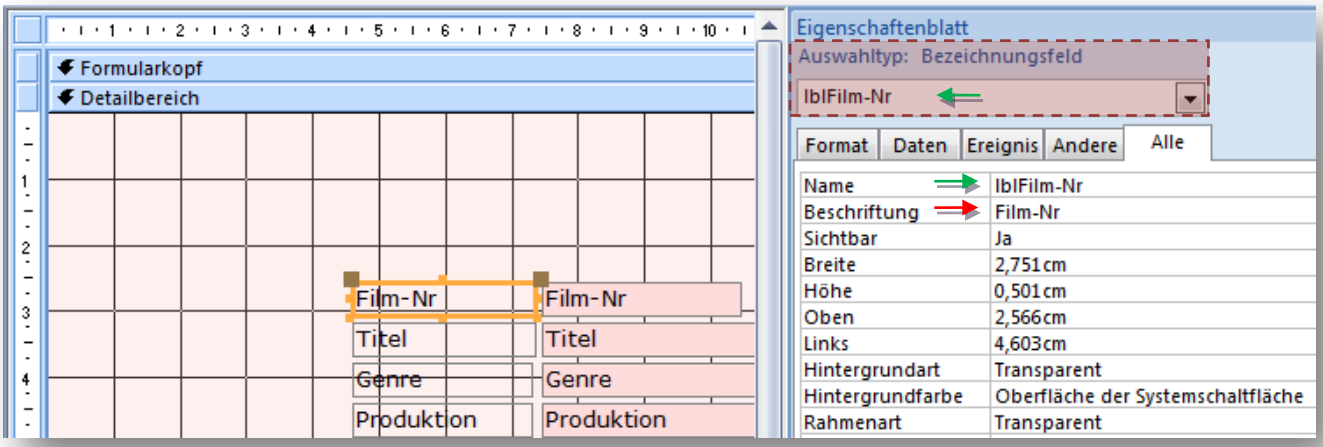

Abb. 6

Das obige *Bezeichnungsfeld* hat den Namen: *lblFilm-Nr* und die Beschriftung lautet: *Film-Nr*. Beziehen Sie sich in einer *Prozedur* nur auf den Namen *lblFilm-Nr*.

Um das Erfassen der Daten effizient zu gestalten können Sie einzelnen *Steuerelementen* in deren *Eigenschaften* einen *SteuerelementTip-Text* (*Quick-Info*) und/oder einen *Statusleistentext* (*Abb. 7*) zuweisen. Damit erreichen Sie, dass auch wenig erfahrene Anwender die richtigen Werte in die entsprechenden *Felder* eintragen. Der benutzerdefinierte Hinweis wird entweder beim Bewegen des Mauszeigers über das *Feld* als *Quick-Info* oder beim Anklicken des *Feldes* - am unteren Bildrand - angezeigt (*Abb. 8*).

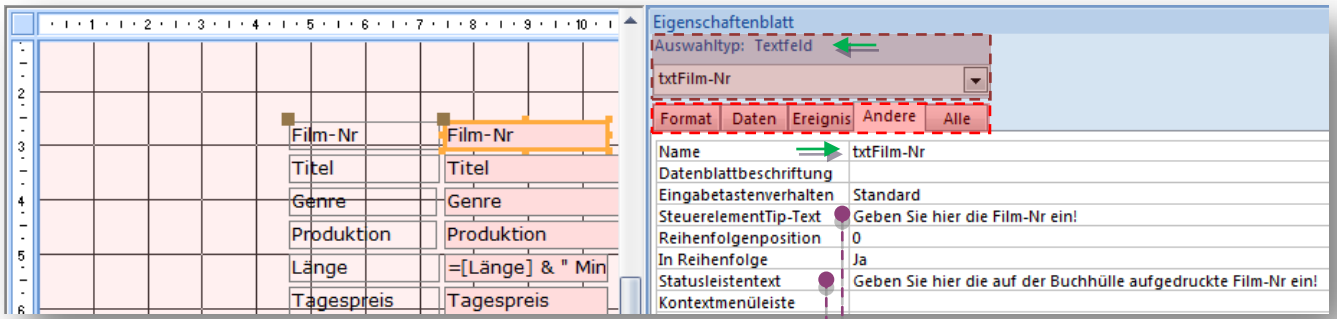

Abb. 7

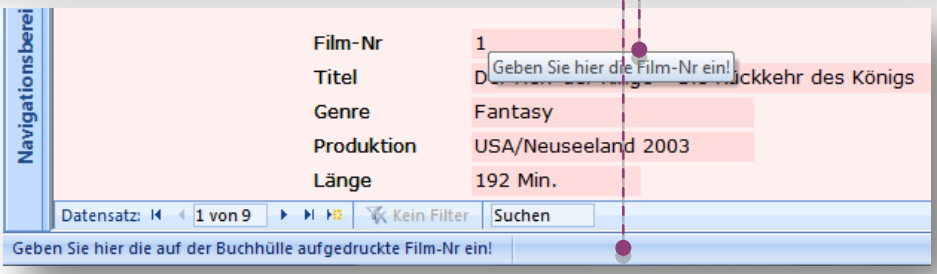

Abb. 8

Nach Markieren eines *Objekts* und Anklicken des Icons *Eigenschaftenblatt* (*Abb. 2 – grüne* Markierung) in der *Multifunktionsleiste – Formularentwurfstools* – Register *Entwurf* - Gruppe *Tools* werden dessen verfügbare *Eigenschaften* angezeigt. Die Auswahl der einzelnen Objekte kann auch durch Aktivieren eines Eintrags im Kombinationsfeld *Auswahltyp* (*Abb. 9 – braune* Markierung) erfolgen.

Klicken Sie beispielsweise in der *Entwurfsansicht* auf das *Formular*, werden die *Eigenschaften* des *Detailbereichs* eingeblendet und Sie können u.a. *Größe, Hintergrundfarbe* und *Spezialeffekte* usw. ändern.

Klicken Sie den Bereich rechts neben dem *Formular* an, werden Ihnen die *Eigenschaften* des *Formulars* selbst angezeigt.

Allen Objekten gemeinsam ist, dass Sie mit Hilfe von 5 Registern eine Vorauswahl beim Anzeigen der *Eigenschaften* (*Properties*) treffen können: *Format, Daten, Ereignis, Andere, Alle* (*Abb. 7 – rote* Markierung). Beachten Sie auch, dass - je nach Objekt - die Anzahl der verfügbaren *Eigenschaften* differiert.

Wenn Sie in den *Eigenschaften* eines *Formulars* beispielsweise das Register *Ereignis* anklicken sehen Sie, welche unterschiedlichen *Ereignisse* diesem Objekt zugewiesen werden können. *Ereignisse* dienen bekanntlich dazu, bestimmte *Aktionen* auszulösen wenn das entsprechende *Ereignis* eintritt. In *Abb. 9* sehen Sie, dass das auslösende *Ereignis* bei einem *Formular* u.a. beim *Anzeigen, Laden* oder *Schließen* des *Formulars* eintreten kann.

Einen entsprechenden Hinweistext erhalten Sie außerdem in der *ACCESS-Statusleiste* angezeigt wenn Sie ein *Ereignis* auswählen.

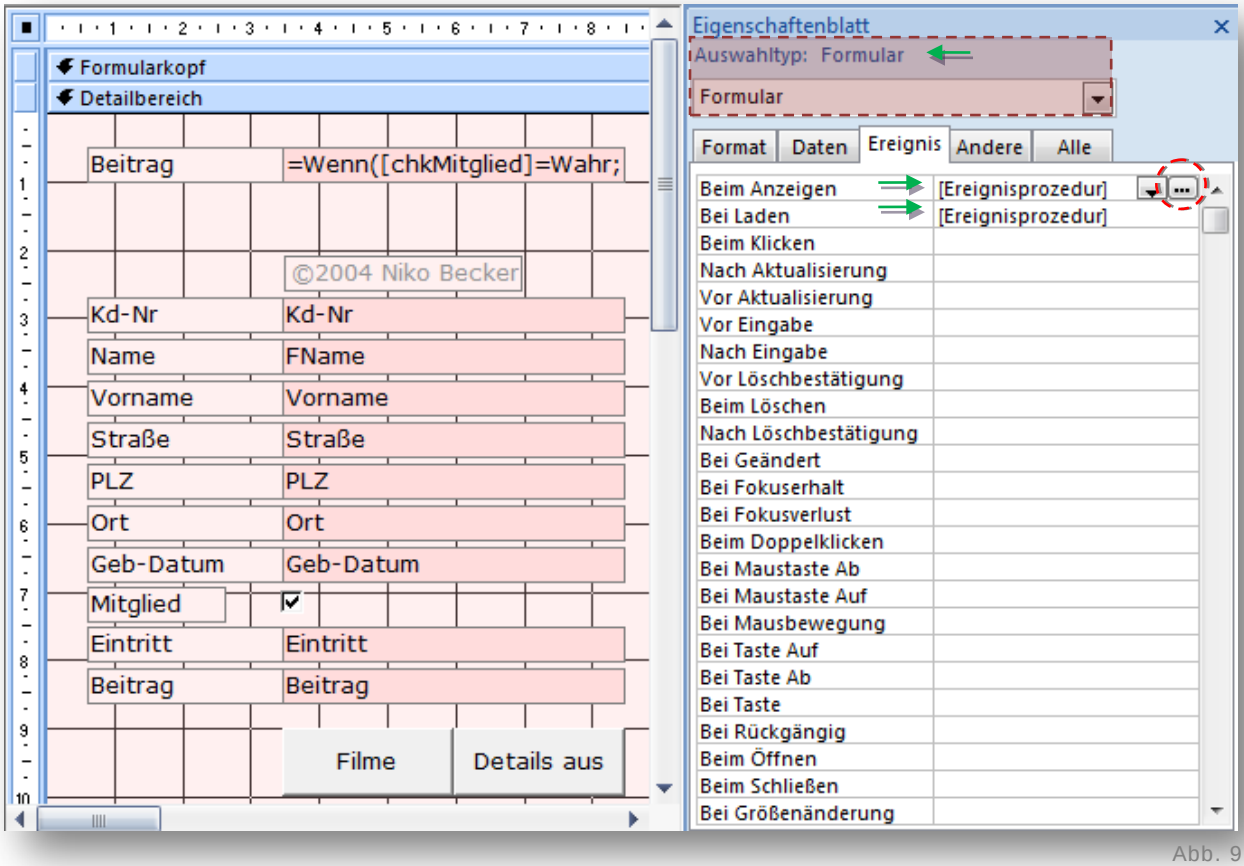

Abb. 9

Wie bereits erwähnt, kann eine auszuführende Aktion mittels vorgefertigtem *ACCESS-Makro* erfolgen, mit Hilfe einer im *Visual Basic-Editor* erstellten *Prozedur* oder auch durch eine *Function* - eine benutzerdefinierte *Funktion*.

Die entsprechende Auswahl kann bei allen Objekten - im Register *Ereignis* - nach Anklicken der drei Punkte am rechten Zeilenrand getroffen werden (*Abb. 9*). Soll dem ausgewählten *Steuerelement* ein vorgefertigtes *Makro* zugewiesen werden klicken Sie im öffnenden Fenster den Eintrag *Makro-Generator* (*Abb. 3*) an, ansonsten das Menü *Code-Generator*.

Bei allen *Ereignissen* werden beim Öffnen des *Code-Generators* zwei Codezeilen automatisch generiert und Sie müssen nur noch Ihre Anweisungen dazwischen schreiben – im folgenden Beispiel beim Ereignis: *Beim Anzeigen.*

```
Private Sub Form_Current()
…
End Sub
```

Der komplette Programmcode lautet in unserem Beispiel:

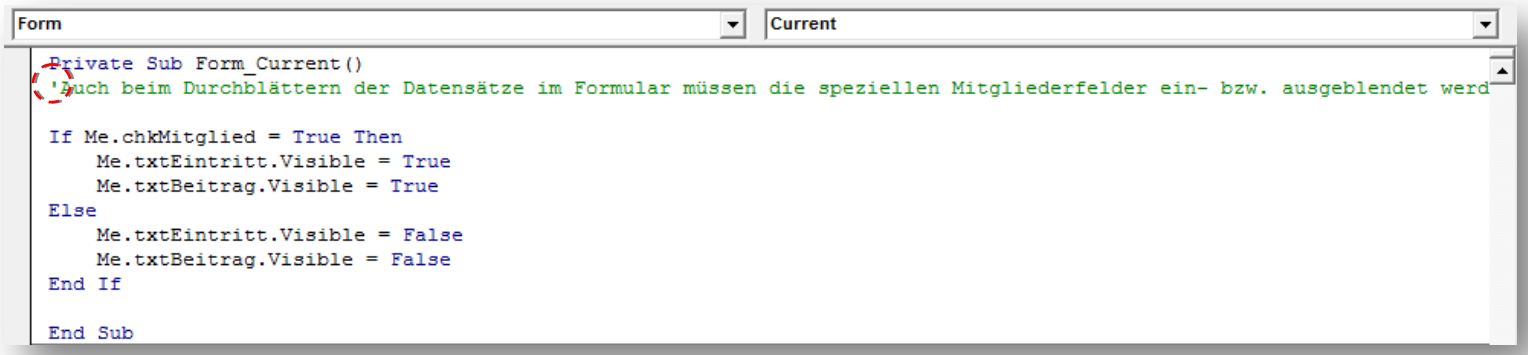

Abb. 10

Wie Sie *Abb. 10* entnehmen können formatiert *ACCESS* automatisch verschiedene Wörter im Programmcode in Schriftfarbe *blau* und *grün*. Bei den *blau* formatierten Texten handelt es sich um *Schlüsselwörter* des Programms. Sie sind vorbelegt und in unserem Beispiel Teil einer `If…Then…Else` (*WENN…DANN…SONST-Bedingung*). Bei dem *Grün* formatierten Text handelt es sich um einen benutzerdefinierten Kommentar, der zum besseren Verständnis des Programmcodes eingefügt werden kann und vom Programm nur angezeigt, aber nicht abgearbeitet wird. Kommentare werden immer mit einem Hochkomma (') eingeleitet und automatisch in Schriftfarbe *Grün* dargestellt.

Wenn Sie beim Eintippen des Programmcodes *Schlüsselwörter* klein schreiben, werden diese automatisch von *ACCESS* in Groß- und Kleinbuchstaben umformatiert, so dass Sie bereits beim Eingeben der Codezeilen erkennen können, ob die *Syntax* richtig geschrieben wurde. Außerdem können Sie das komplette *Projekt* auf Syntaxfehler hin überprüfen, indem Sie im Menü *Debuggen* den Eintrag *Kompilieren* aktivieren (*Abb. 11*).

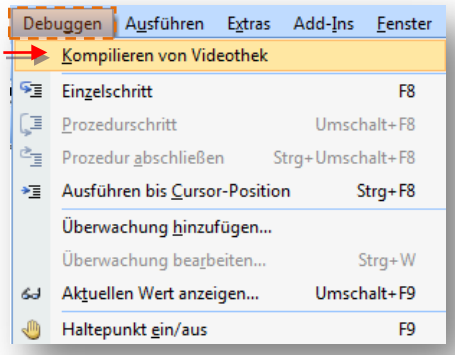

Abb. 11

Nun zur Bedeutung der obigen Codezeilen: Mit `Sub` und `End Sub` werden Anfang und Ende einer *Prozedur* definiert – dazwischen stehen Ihre Anweisungen. Mit `Private` wird festgelegt, dass die *Prozedur* nur in diesem ***Modul***[2] oder *Formular* zur Verfügung steht - wogegen mit dem Schlüsselwort `Public` eröffnete *Prozeduren* immer in der gesamten *Datenbank* zur Verfügung stehen.

[2] …wenn Sie eine neue *Prozedur* erstellen oder eine *Function* in das *Projekt* einbinden wollen, schreiben Sie den entsprechenden Programmcode in ein *Modul* (*VBA-Editor: Einfügen – Modul*). Diese Anweisungen können dann in *Prozeduren* oder im Programmcode von *Steuerelementen* aufgerufen werden.
Im Unterschied zu den bisher beschriebenen *Prozeduren* werden diese nicht bestimmten *Ereignissen* zugewiesen.

Wenn Sie z.B. erreichen wollen, dass beim Öffnen eines *Formulars* bestimmte *Textfelder* und deren *Bezeichnungsfelder* nicht angezeigt werden, fügen Sie zwischen die beiden vorgenannten Schlüsselzeilen folgende Codezeilen ein:

```
Me.[txtEintritt].Visible = False
Me.[lblEintritt].Visible = False
```

Den Bezug auf ein *Feld* in Ihrem *Formular* stellen Sie dadurch her, dass Sie den *Feldnamen* in eckige Klammern [] schreiben – Sie kennen das bereits aus *Abfragen*.

Sollen die beiden *Felder* im obigen Beispiel wieder angezeigt werden schreiben Sie:

```
Me.[txtEintritt].Visible = True
Me.[lblEintritt].Visible = True
```

Mit `Me.` beziehen Sie sich auf das aktuelle *Formular* in welchem die *Prozedur* erstellt wird. Nach Eingeben des Punktes nach `Me` wird eine Liste aller Objekte des *Projekts* angezeigt und Sie können das gewünschte Objekt einbinden indem Sie es anklicken. Der Bezug auf das *Formular* verkürzt sich dadurch von der Schreibweise: `Forms!Formularname.Element.Eigenschaft` auf `Me.Element.Eigenschaft`

Beim Anlegen des Programmcodes wird auch nach Eingeben des Punktes hinter dem Feldnamen `[txtEintritt]` immer die *Direkthilfe* mit den verfügbaren *Eigenschaften* eingeblendet und Sie können durch Anklicken eines Eintrags die gewünschte *Eigenschaft* auswählen – im aktuellen Beispiel: `visible` (sichtbar). Beim Öffnen eines *Formulars* besitzen diese *Eigenschaften* immer den Wert `True` (Wahr); Änderungen können manuell oder durch Aktivieren des entsprechenden Eintrags in der *Direkthilfe* (nach =) vorgenommen werden.

Wenn Sie beim Erstellen des Programmcodes Punkte, Kommas, Klammern oder Gleichheitszeichen schreiben bietet *ACCESS* permanent *Direkthilfen* an um das Vervollständigen der *Syntax* zu erleichtern.

Da es sich auch bei den Einträgen `True` und `False` um *Schlüsselwörter* handelt, werden sie automatisch *blau* formatiert dargestellt.

Beachten Sie, dass die Namen von *Prozeduren, Variablen* und *Konstanten* immer mit einem Buchstaben beginnen müssen. Grundsätzlich dürfen nur *Buchstaben, Ziffern* und der *Unterstrich (_)* verwendet werden. Eine Bezeichnung kann maximal 200 Zeichen lang sein und sie darf keine *Satzzeichen* und *Leerzeichen* enthalten. Die von Ihnen verwendeten Begriffe dürfen außerdem nicht mit *Schlüsselwörtern* des Programms identisch sein.

Wenn Sie mit Hilfe des *Befehlsschaltflächen-Assistenten* (*Abb. 12*) eine *Schaltfläche* in ein *Formular* einfügen um z.B. ein Formular *Filme* zu öffnen, wird automatisch von *ACCESS* ein entsprechender Programmcode generiert, den Sie -

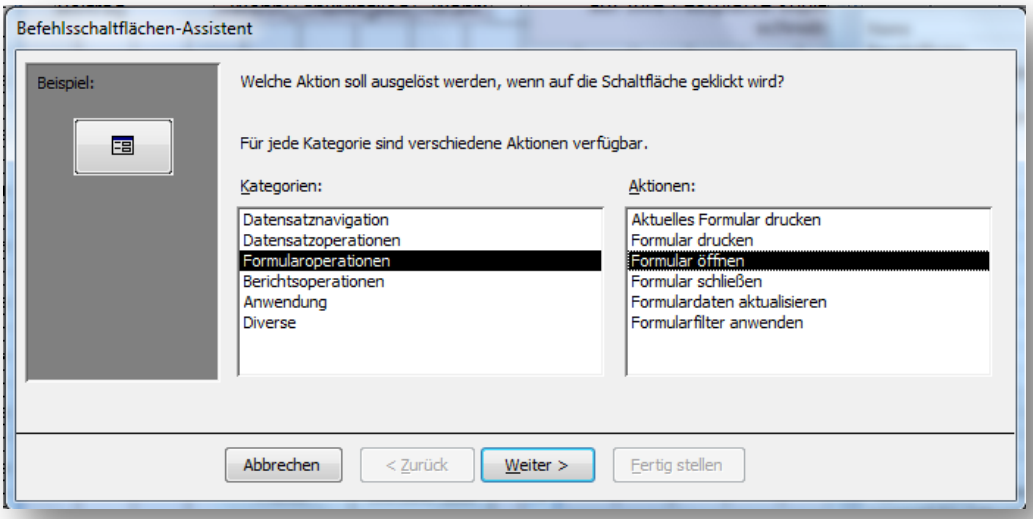

Abb. 12

nach Markieren des Objekts - im *Visual Basic-Editor* einsehen können (*Abb. 14*).

Klicken Sie dazu in den *Eigenschaften* des *Steuerelements*, Zeile *Beim Klicken* auf die drei Punkte am rechten Zeilenrand (*Abb. 13*) und beachten Sie, dass hier von *ACCESS* automatisch der Eintrag *Ereignisprozedur* eingefügt wurde. Nachdem Sie den *Code-Generator* geöffnet haben wird der folgende Programmcode angezeigt:

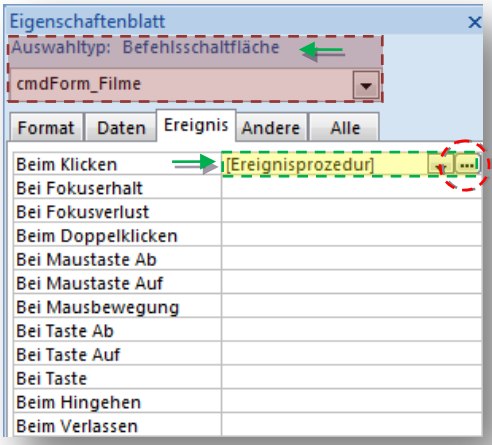

Abb. 13

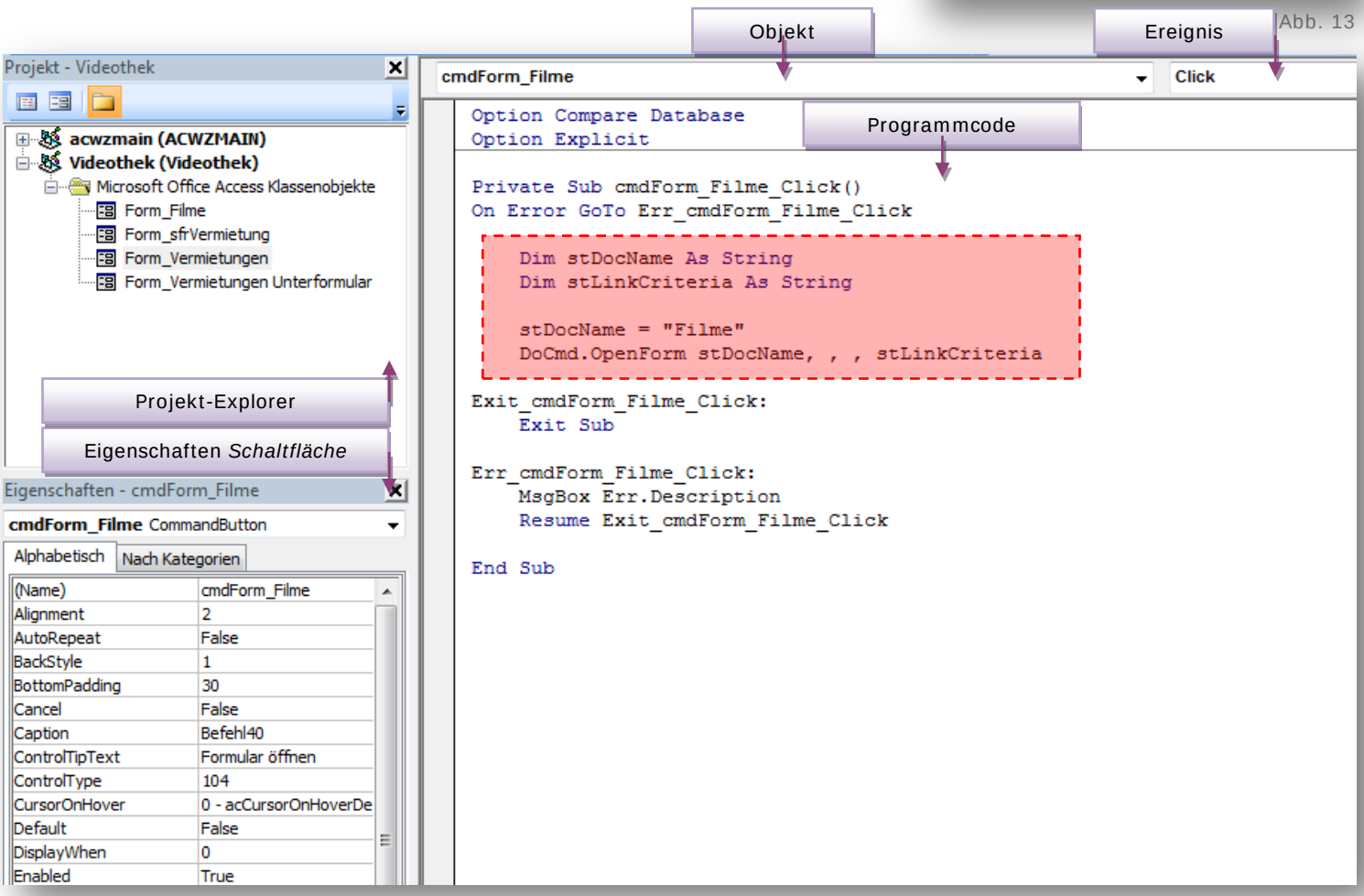

Abb. 14

Das *Präfix* für *Schaltflächen* (*CommandButton*) lautet: *cmd*, der Name des Objekts somit: *cmdForm_Filme*.

Der vorstehende Programmcode ist deshalb etwas umfangreicher, weil *ACCESS* bei einem automatisch generierten Code zusätzlich eine Fehlerroutine einbindet und festlegt was passieren soll wenn beispielsweise ein *Laufzeitfehler* auftritt.

Für das Funktionieren der Anweisung sind allerdings nur die markierten Zeilen relevant:

```
Dim stDocName As String
Dim stLinkCriteria As String

stDocName = "Filme"
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

Mit dem *Schlüsselwort* `Dim` werden in den beiden ersten Zeilen zwei *Variablen* deklariert, wobei `stDocName` der Name der *Variablen* ist und `As String` der Variablentyp *Text*.

ACCESS legt hier eine *Variable* an weil immer dann wenn Sie Werte speichern, seien es Ergebnisse aus Berechnungen oder Namen die sich durch neue Berechnungen oder durch Namensänderungen ergeben, diese in *Variablen* gespeichert werden.

Variablen können *explizit* deklariert werden; d.h. *ACCESS* überwacht, dass die *Variablen* in einer *Prozedur* deklariert werden, dass ihnen der deklarierte Wert (*Variablentyp*) zugewiesen wird, und dass der Name der *Variablen* innerhalb der *Prozedur* richtig geschrieben wird.

Die Überwachung können Sie im *Visual Basic-Editor*, Menü - *Extras – Optionen* – Register *Editor* durch Aktivieren des Kontrollkästchens *Variablendeklaration erforderlich* festlegen. Sie erkennen diese Festlegung am Eintrag `Option Explicit` in der ersten Zeile im Codefenster.

Bei größeren *Prozeduren* und bei Verwendung mehrerer *Variablen* ist diese Vorgehensweise empfehlenswert, da sie ein größeres Maß an Sicherheit bietet.

Variablen können innerhalb einer *Prozedur* auch *implizit* deklariert werden - beispielsweise durch den Eintrag:

```
Mehrwertsteuer = [Nettobetrag]*0.19
```

In diesem Fall muss die vorstehend beschriebene Festlegung (`Option Explicit`) allerdings wieder aufgehoben werden, die *Variable* erhält den Variablentyp *Variant* und ist auch nur in der aktuellen *Prozedur* gültig.

Variablen können unterschiedliche Werte aufnehmen; feststehende Werte müssen in *Konstanten* deklariert werden. Die Deklaration erfolgt aber nicht mit dem Schlüsselwort `Dim` sondern mit `Const` und die entsprechende *Syntax* lautet in diesem Fall beispielsweise:

```
Const MwStSatz = 0.19
```

Beachten Sie, dass Sie in einer *Prozedur* die Nachkommastellen einer Dezimalzahl mit einem **Punkt** darstellen müssen und das Tausender-Trennzeichen mit einem **Komma**.

Nun zurück zu den vier Codezeilen am Ende von *Seite 8*. Wie Sie der 3. Codezeile entnehmen können wird der *Variablen* `stDocName` der Wert *Filme* zugewiesen (`stDocName = "Filme"`). Im weiteren Verlauf der *Prozedur* wird im generierten Code dann nur noch Bezug auf die *Variable* (s. *Zeile 4* - `stDocName`) genommen, die als *Platzhalter* für den jeweiligen Formularnamen (*Filme*) steht.

Die Codezeile enthält noch eine weitere Anweisung: `DoCmd` - was mit *Do Command* - *Kommando ausführen* - zu übersetzen wäre. Nach einem Punkt folgen die ***Methoden***[3] des `DoCmd`-Objekts - in unserem Beispiel: `OpenForm` (Formular öffnen) – und danach der Name des *Formulars* – in Form der *Variablen*: `stDocName`, sowie die *Argu-*

[3] …*Methoden* sind Aktionen die von oder mit einem Objekt ausgeführt werden können – z.B.: `SetFocus`, `OpenForm` usw.
Verschiedene *Methoden* übergeben auch *Argumente* um die Ausführung zu präzisieren: `DoCmd.OpenForm "Filme"` übergibt beispielsweise das *Argument* `"Filme"` um die Ausführung des Befehls *Öffnen Formular* zu präzisieren.

mente [,Ansicht] [,Filtername] [,Bedingung] [,Datenmodus] [,Fenstermodus] [,Öffnungsargumente].

Wenn Sie diese *Syntax* manuell erstellen wird Ihnen nach jedem Komma wieder die *Direkthilfe* eingeblendet, damit Sie den Programmcode vervollständigen können. Die Kommas stehen stellvertretend für nicht benannte *Argumente* innerhalb der *Syntax*. Ohne die *Argumente* wird das Formular *Filme* in der *Formularansicht* geöffnet.

Soll ein *Formular* geschlossen werden binden Sie in Ihre Prozedur die folgende Codezeile ein:

```
DoCmd.Close acForm, "Filme", acSaveNo
```

Die *Syntax* beginnt wieder mit `DoCmd.Close` (Kommando ausführen: schließen) danach die *ACCESS-Konstante* für *Formulare* (`acForm`) und anschließend der Name des Objekts `"Filme"`, sowie die Speichern-Anweisung (`acSaveNo`) – Formular schließen - nicht speichern.

Wenn Sie erreichen wollen, dass ein *Formular* beim Öffnen im *Vollbildmodus* angezeigt wird, müssen Sie in dessen *Eigenschaften* das Ereignis *Beim Öffnen* auswählen und nach Anklicken der drei Punkte am rechten Zeilenrand den *Code-Generator* öffnen. Ergänzen Sie die beiden generierten Codezeilen, indem Sie die folgende Anweisung dazwischen schreiben:

```
Private Sub Form_Open(Cancel As Integer)
DoCmd.Maximize
End Sub
```

Die Anweisung wird mit `DoCmd` eröffnet, ergänzt mit der *Methode* `Maximize`, die das *Formular* maximiert - also im *Vollbildmodus* anzeigt.

Hinweis:

Sollten Sie *ACCESS* in Verbindung mit anderen Office-Anwendungen, wie z.B. *EXCEL* oder *WORD* nutzen, werden Sie gelegentlich versuchen auf *EXCEL*-Funktionen zurückzugreifen. Das ist allerdings nur bedingt möglich: *ACCESS* ist nicht *EXCEL*! In unseren Übungsaufgaben zeigen wir Ihnen in welchen Fällen das möglich ist.

Wenn Sie unterschiedliche Office-Versionen nutzen sollten Sie darauf achten, dass in *ACCESS* die entsprechenden Bibliotheken eingebunden sind. Dazu muss im *Visual Basic-Editor* - Menü *Extras – Verweise* ein Verweis auf die entsprechende *EXCEL*- oder *WORD*-Library hergestellt werden. Dieser Verweis ist nur in der aktuellen *Datenbank* gültig und wird durch Anklicken des entsprechenden *Kontrollkästchens* aktiviert oder deaktiviert.

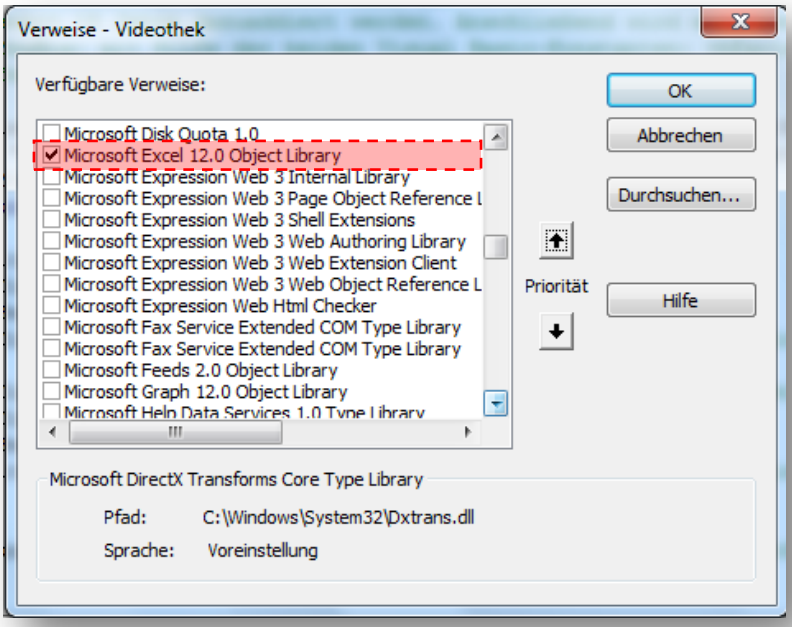

Abb. 15

Weitere Informationen zu diesem Thema finden Sie in der *VBA-Offline-Hilfe* - nach Eingabe des entsprechenden Suchbegriffs - und in den ausführlichen Lösungshinweisen zu unseren Übungsaufgaben.