

## 2 Einführung in die VBA-Programmierung mit Access 2007

### 2.1 Grundlagen

Visual Basic for Applications (VBA) ist eine zu den Microsoft-Office-Programmen gehörende Skriptsprache. Sie wurde aus dem von Microsoft entwickelten BASIC-Dialekt Visual Basic (VB) abgeleitet und wurde zur Steuerung von Abläufen innerhalb der Microsoft-Office-Programme entwickelt. VBA ist seit Mitte der 1990er Jahre der Nachfolger der bis zu diesem Zeitpunkt in den Microsoft-Office-Anwendungen enthaltenen verschiedenen Makro-Sprachen.

Derzeit ist VBA in den Microsoft-Office-Programmen Word (seit Version 97), Excel (seit Version 95), Access (seit Version 95), Project, PowerPoint, FrontPage, Visio (seit Version 2000) und Outlook verfügbar. Darüber hinaus wird VBA von der Corel Corporation lizenziert und ist in Corel Draw und Corel Photo-Paint verfügbar. VBA wird auch als Makrosprache in AutoCAD, ArcGIS, ARIS, MindManager und vielen anderen Anwendungen eingesetzt.

VBA gilt als leistungsfähige Skriptsprache und ist die am stärksten verbreitete Möglichkeit, auf Microsoft-Office-Anwendungen (Excel, Word, Access) basierende Programme zu erstellen. VBA ist eine interpretierte Programmiersprache, deren Syntax der von Visual Basic entspricht. Die Möglichkeiten und die Leistungsfähigkeit von VBA sind allerdings gegenüber Visual Basic deutlich reduziert. Beispielsweise wird ein VBA-Skript zwar vorkompiliert, um Variablen- und Konstantentabellen aufzubauen und syntaktische Überprüfungen durchzuführen, ein Kompilieren bis hin zu ausführbarem Maschinencode ist jedoch nicht möglich.

VBA ist vor allem für **prozedurale Programmierung** konzipiert. Klassen und Objekte können zwar syntaktisch dargestellt und implementiert werden, unterliegen jedoch den auch in Visual Basic Classic bestehenden Einschränkungen (fehlende Implementierungsvererbung); andere Merkmale wie Datenkapselung, Interface-Vererbung und Laufzeitpolymorphie können hingegen verwendet werden.

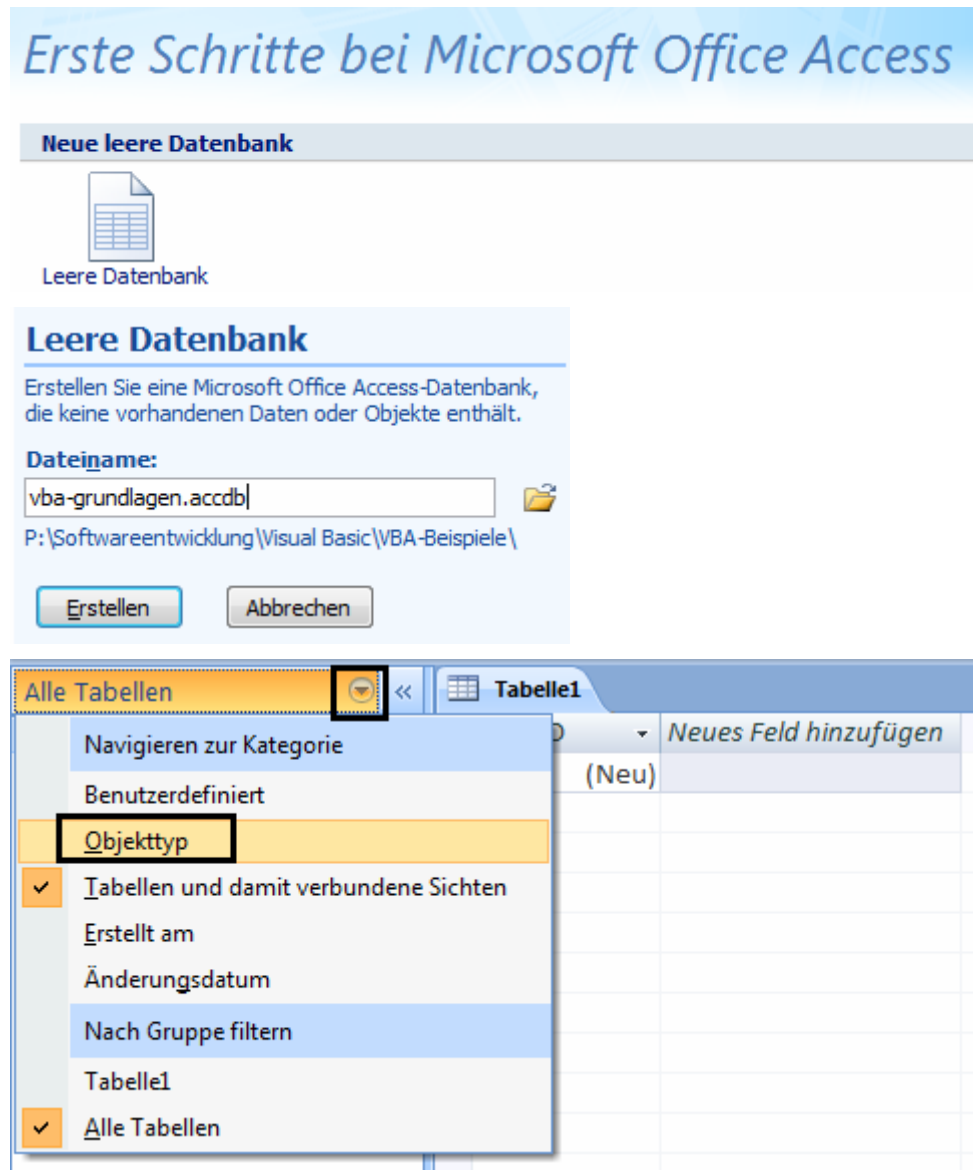
Der Zugriff über VBA auf das jeweilige Wirtsprogramm (Word, Excel etc.) erfolgt über eine meist gut dokumentierte **Programmierschnittstelle**. In den jeweiligen Anwendungen stehen neben dem VBA-Kern (Kontrollstrukturen, Datentypen, mathematische Funktionen, Dateisystem) spezielle Objekte des Wirtsprogramms zur Verfügung. Diese Objekte ermöglichen es, Abläufe des jeweiligen Wirtsprogramms zu automatisieren. Da diese Schnittstelle über das **Component Object Model (COM)** realisiert wird, kann eine VBA-Anwendung auch auf andere COM-Komponenten als die vom Wirtsprogramm zur Verfügung gestellten zugreifen.

#### Sicherheitsaspekte:

Die Anwendungsmöglichkeiten von VBA sind nicht auf die Automatisierung von Wirtsanwendungen beschränkt; der Leistungsumfang umfasst auch wesentliche Funktionen von Visual Basic. Daher können VBA-Anwendungen, ebenso wie alle anderen Windows-Anwendungen, schädlichen Code enthalten. Statt eines a-priori-Sicherheitskonzeptes, bei dem möglicherweise gefährliche Funktionen schon beim Sprachentwurf eingeschränkt oder die Makroausführung mittels einer Sandbox vom restlichen System abgeschirmt wird, stellt Microsoft-Office lediglich im Nachhinein Methoden zur Absicherung der Programmausführung wie etwa eine Prüfungsmöglichkeit eventuell vorhandener Makrozertifizierung zur Verfügung. Diese Faktoren, kombiniert mit der leichten Erlernbarkeit von VBA, führten zur Verbreitung von Word-Makroviren. Neuere MS-Office-Versionen fragen vor der Ausführung von VBA-Programmen nach und empfehlen, nur vertrauenswürdige und mit überprüfbaren Zertifikaten ausgestattete Makros zuzulassen.

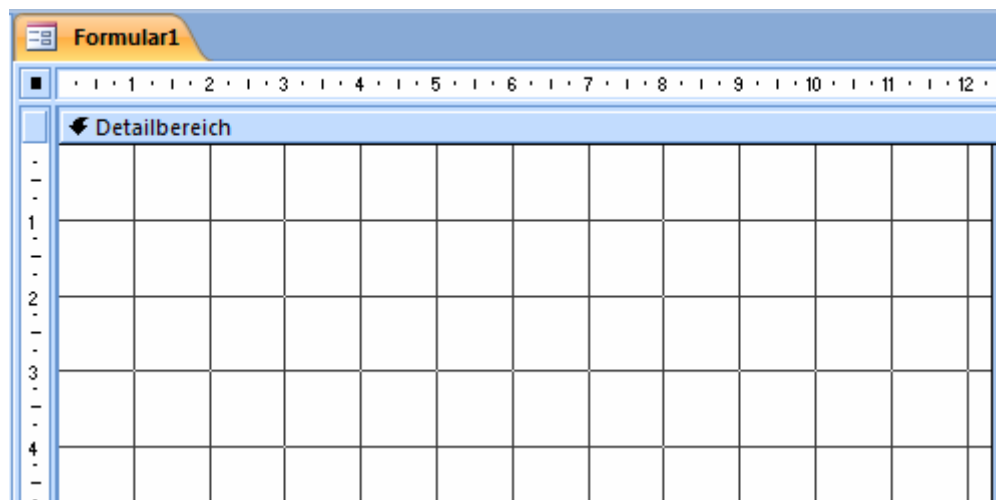
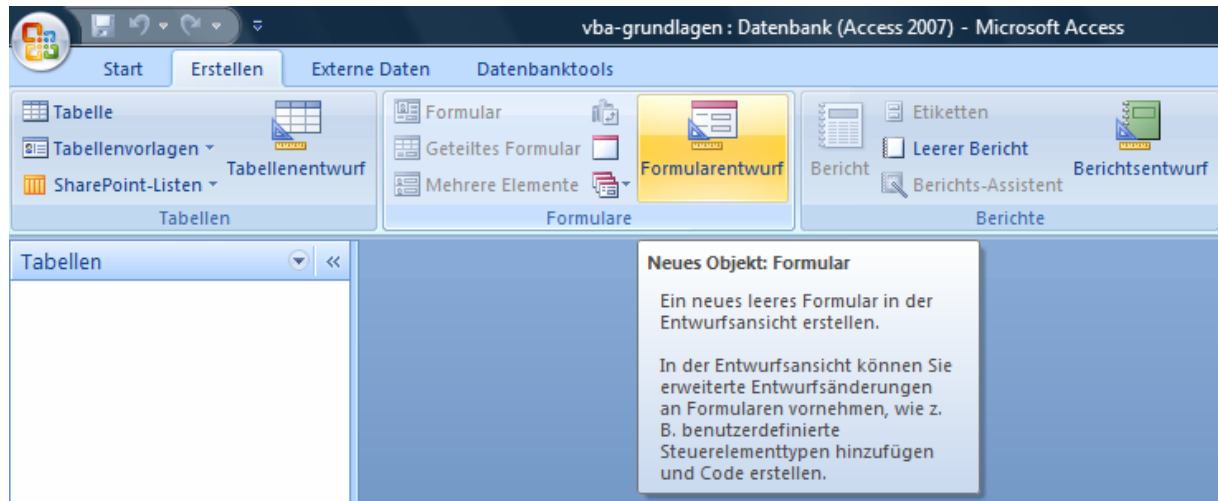
## 2.2 Erste Schritte

Legen Sie zunächst eine leere Access-Datenbank an und speichern Sie sie:

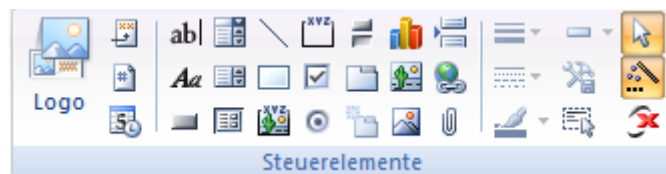



Obwohl in einer Access-Datenbank Tabellen die wichtigsten Objekte darstellen (alle Daten werden in Tabellen gespeichert), wollen wir bei der Einführung in die Programmierung diesen Aspekt beiseite lassen.

Wir schließen die von Access 2007 erzeugte Tabelle und legen stattdessen ein neues leeres Formular (engl. Form) an:




Jede Form enthält eine Reihe von **Steuerelementen** (engl. „Controls“): Bildlaufleisten, Befehlschaltflächen (z.B. „OK“ oder „Abbrechen“) usw. Diese Elemente können alle mit Hilfe der Werkzeug-Leiste in die Form eingefügt werden.




 PictureBox (Bild). Kann Grafiken (Bitmap \*.BMP, Icon \*.ICO, Metafile \*.WMF, \*.GIF, \*.JPG) darstellen. Kann auch andere Steuerelemente enthalten und Text anzeigen.

 Label (Bezeichnungsfeld). Für Texte, die nicht vom User geändert werden sollen (Überschriften etc.).

 Text Box (Eingabefeld, „Textfeld“). Hierin befindet sich Text, der vom User geändert werden kann/soll.

 Frame (Rahmen). Dient zur Gruppierung von Steuerelementen.

 CommandButton (Befehlschaltfläche). Erzeugt eine Schaltfläche, deren Anklicken ein Unterprogramm auslöst.



CheckBox (Auswahlfeld). Hier kann eine wahr/falsch-Möglichkeit ausgewählt werden bzw. eine Auswahl aus mehreren Punkten vorgenommen werden. (Das Anklicken keines oder mehrerer Felder ist möglich.)



OptionButton (Optionsfeld). Hier kann der User nur eine Möglichkeit auswählen.



ComboBox (Kombinationslistenfeld) = Kombination aus einem Listenfeld und einem Textfeld. Der User kann entweder eine Eingabe in den Textfeldteil schreiben oder aus dem Listenteil eine Möglichkeit wählen.



ListBox (Listenfeld). Scrollbare Liste mit Elementen, aus denen der User eines auswählen kann.



Line (Linie). Zur Entwicklungszeit können Linien verschiedener Stärke, Farbe und Art gezeichnet werden.

An dieser Stelle soll erwähnt werden, dass alle Steuerelemente und auch das Formular selbst sogenannte **Objekte** darstellen. Visual Basic ist ein spezieller Typ der sogenannten „objektorientierten Programmiersprachen“.

Jedes Objekt ist durch zwei Dinge charakterisiert:

- **Eigenschaften = Properties** (Variablen)
- **Verhalten** (Prozeduren, Funktionen)

So hat etwa eine Form die Eigenschaft „Caption“. (In dieser Variable ist die „Überschrift“ abgespeichert, die man als Titelzeile einer Form sieht.) Man kann diese Eigenschaft entweder über das „Eigenschaften“-Fenster ändern oder aber vom Code aus:

```
Form1.Caption = "Mehrwertsteuerberechnung"
```

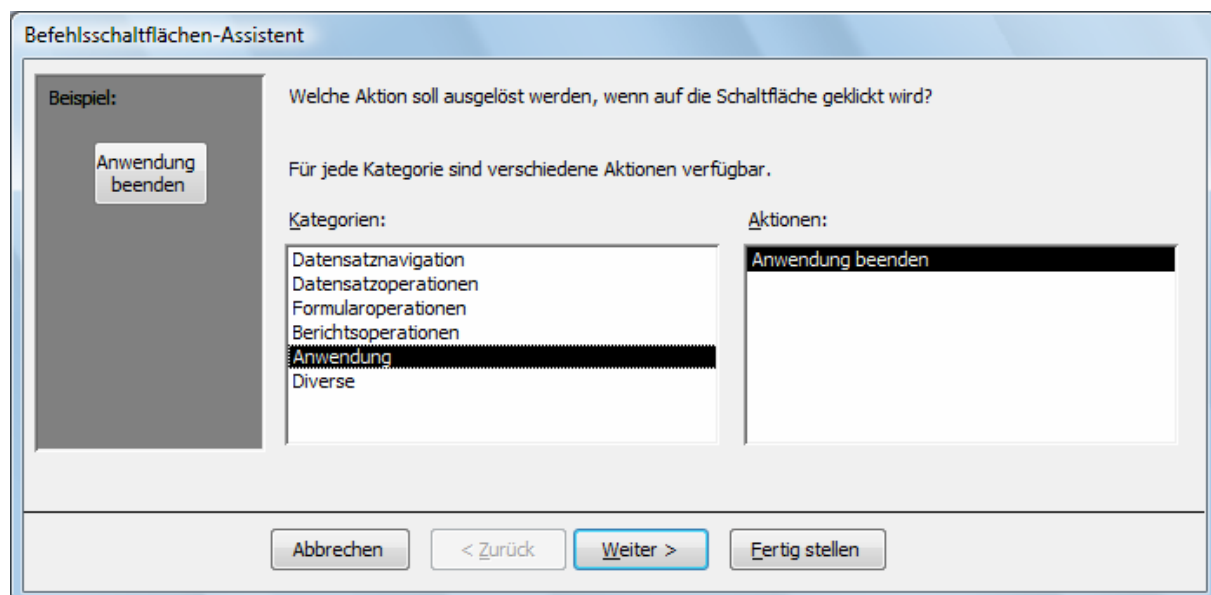
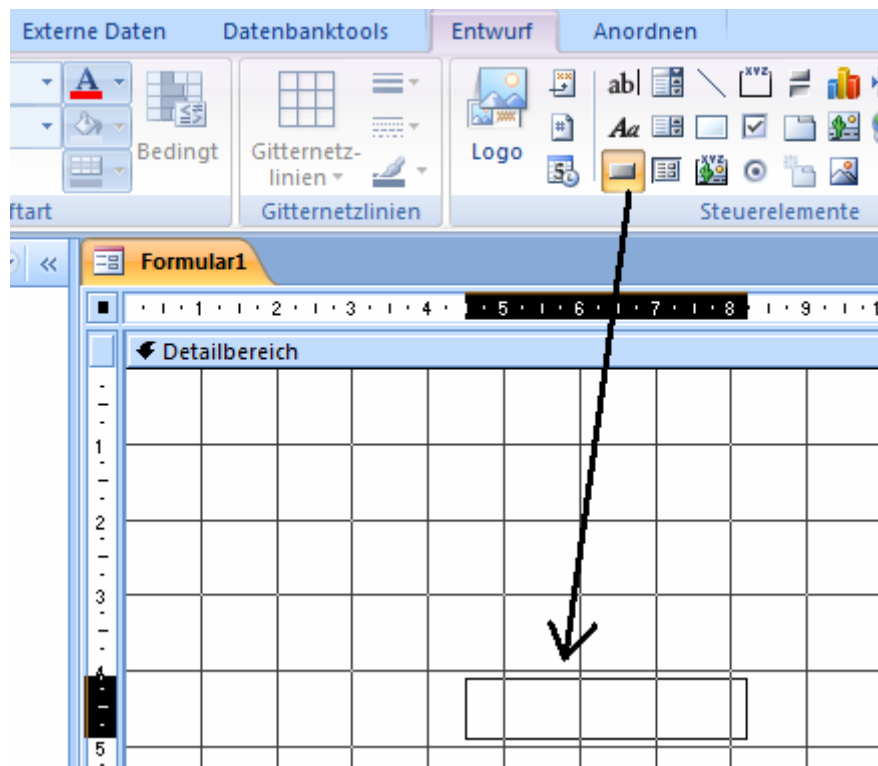
Die Variablen eines Objekts werden oft als **Attribute** bezeichnet. Jede Form hat aber auch die Möglichkeit, angezeigt zu werden. Dazu gibt es die Prozedur „Show“, die ebenfalls auf diese Art und Weise aufgerufen werden kann:

```
Form1.Show
```

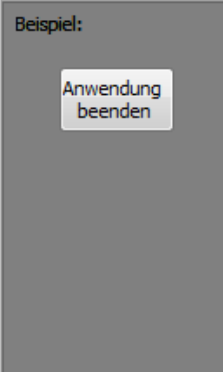
Die Prozeduren und Funktionen eines Objekts werden meist als **Methoden** bezeichnet.

Es gibt meist mehrere Objekte vom gleichen „Typ“ (also z.B. mehrere Formen). Beide Objekte sind eigentlich nur spezielle „Variablen“, deren „Objekttyp“ generell festgelegt ist. Man sagt: Jedes Objekt ist eine **Instanz** seiner **Klasse**.

Erzeugen Sie durch „Drag and Drop“ (Ziehen und Ablegen) eine Instanz eines CommandButton-Objekts:



**Befehlsschaltflächen-Assistent**

Beispiel: 

Möchten Sie Text oder ein Bild auf Ihrer Schaltfläche?


Wenn Text auf der Schaltfläche angezeigt werden soll, geben Sie diesen bitte ein. Möchten Sie ein Bild anzeigen, so können Sie auf die Schaltfläche 'Durchsuchen' klicken, um nach dem entsprechenden Bild zu suchen.

Text:

Bild:

Alle Bilder anzeigen

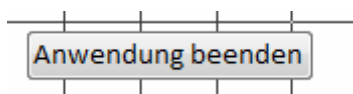
**Befehlsschaltflächen-Assistent**

Beispiel: 

Wie möchten Sie Ihre Schaltfläche nennen?


Wenn Sie später auf die Schaltfläche verweisen möchten, kann ein sinnvoller Name nützlich sein.

Dies sind alle Informationen, die der Assistent zur Erstellung Ihrer Befehlsschaltfläche benötigt. Hinweis: Dieser Assistent erstellt eingebettete Makros, die in Access 2003 und früheren Versionen nicht ausgeführt oder bearbeitet werden können.



Speichern Sie das Formular:

Microsoft Office Access

 Möchten Sie die am Entwurf von Formular 'Formular 1' vorgenommenen Änderungen speichern?

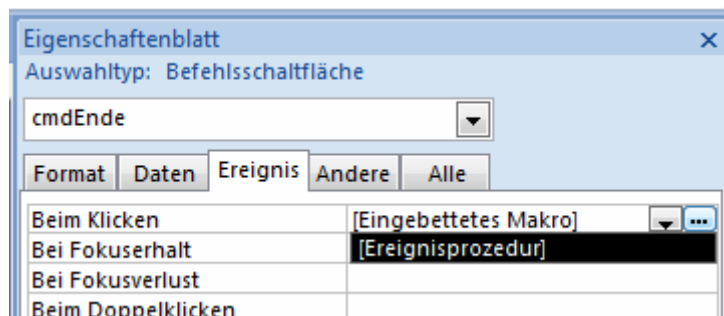
Speichern unter

Formularname:

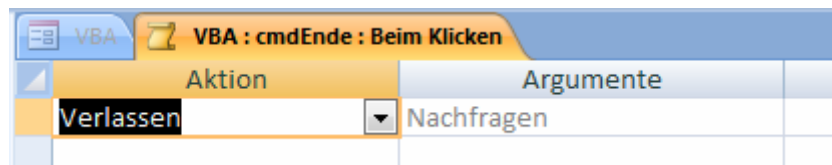
Wechseln Sie in die Formularansicht und testen Sie die Befehlsschaltfläche.

Ergebnis: Klicken auf die Befehlsschaltfläche beendet das Programm und schließt Access.

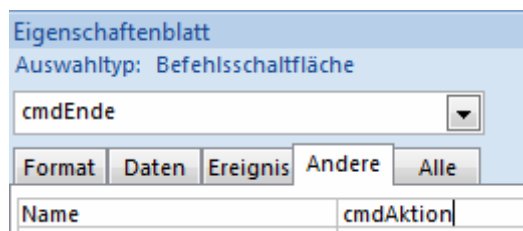
Klicken Sie in der Entwurfsansicht mit der rechten Maustaste auf die eben erstellte Befehlsschaltfläche und wählen Sie [Eigenschaften]:



Sie sehen: Der Assistent hat die Befehlsschaltfläche mit einem **eingebetteten Access-Makro** gekoppelt. Das Makro können Sie mit einem Klick auf die drei Punkte ansehen:

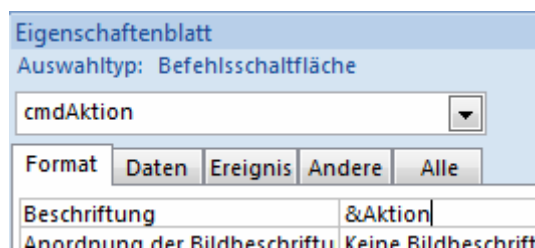


Zunächst ändern wir den internen Objektname der Schaltfläche:



Unter diesem Namen muss die Befehlsschaltfläche beim Programmieren angesprochen werden.

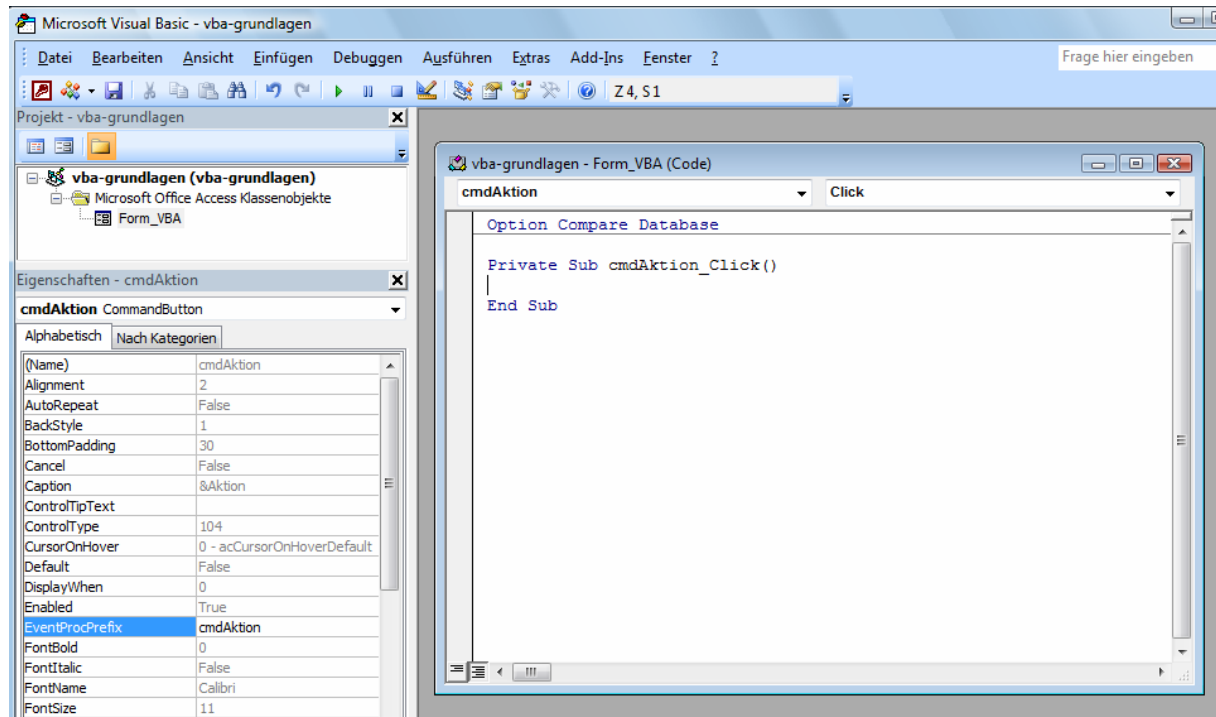
Dann ändern wir die Beschriftung:



Das vorangestellte &-Zeichen bewirkt, dass das folgende Zeichen (hier: das A) unterstrichen dargestellt wird und die Befehlsschaltfläche auch mit der Tastenkombination ALT-A ausgeführt werden kann.

Wir wollen das Verhalten dieser Schaltfläche ändern. Wir wählen als Aktion, die beim Klicken ablaufen soll, den Eintrag [Ereignisprozedur] aus.

Es öffnet sich erstmals die integrierte **VBA-Entwicklungsumgebung**:



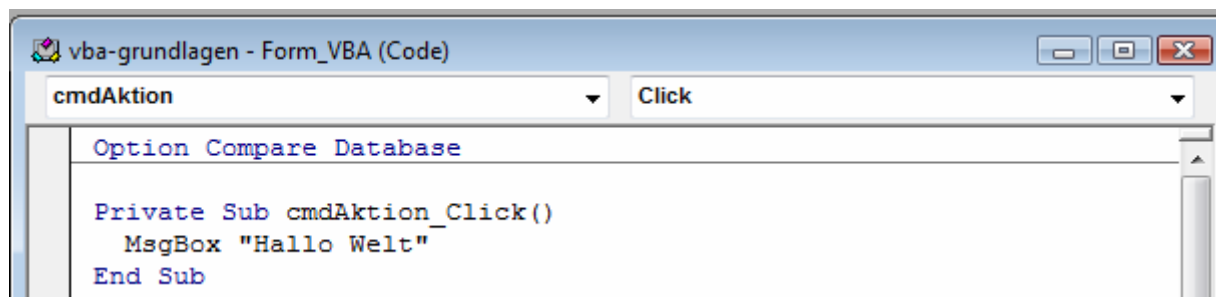
Alle Visual Basic-Programme bestehen aus in sich abgeschlossenen Programmteilen = Unterprogrammen. Man unterscheidet **Prozeduren** (Sub) und **Funktionen** (Function). Jede Prozedur (auch jede Funktion) besteht aus Zeilen, in jeder Zeile steht eine Visual Basic-Anweisung.

**Ereignisgesteuerte Programmierung:** Jede Prozedur beginnt mit dem Wort `Sub` und endet mit `End Sub`. Nach dem Wort `Sub` steht der Name der Prozedur. Dieser Name ist bei **Ereignisprozeduren** nicht frei wählbar, sondern besteht aus dem Namen des Objekts und dem Namen der Aktion (des **Ereignisses**), bei der das Unterprogramm ausgelöst werden soll. Will man zum Beispiel, dass die zu programmierende Prozedur ausgelöst wird, wenn der Benutzer auf die Befehlsschaltfläche mit dem Namen `cmdAktion` klickt, so lautet der Name der Prozedur `cmdAktion_Click()`.

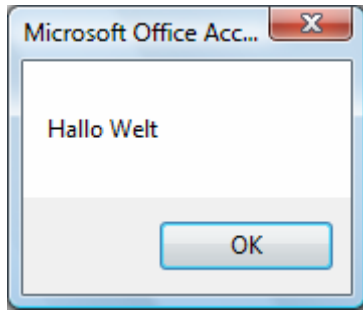
Als traditionell erstes Programm wollen wir ein „Hallo Welt“-Programm schreiben, welches nichts anderes tut, als uns zu begrüßen, wenn auf die Schaltfläche geklickt wird.

Dazu benötigen wir eine Ereignisprozedur mit dem Namen `cmdAktion_Click()`. Genau diese Ereignisprozedur wird von VBA bereits vorgeschlagen.

Beachten Sie: Sie können das betroffene Objekt und das Ereignis ändern, indem Sie die jeweilige Auswahl treffen (oberste Zeile im Editor)!



Testen Sie die Prozedur:



### 2.3 Ausgabe mit MsgBox

In der Datenverarbeitung gibt es seit langer Zeit das Grundprinzip **Eingabe – Verarbeitung – Ausgabe**. Auch in der Zeit der Windows-Programmierung sind diese Elemente aus Programmen nicht wegzudenken. Daher ist es nötig, zunächst diese Grundbegriffe durcharbeiten.

Die einfachste Möglichkeit, in VBA zu einer Ausgabe auf dem Bildschirm zu kommen, ist die Verwendung der Anweisung MsgBox.

Die Anweisung MsgBox verlangt mehrere **Parameter**. Darunter versteht man Werte, die übergeben werden müssen, damit die Anweisung korrekt ausgeführt werden kann.

```
Private Sub cmdAktion_Click()
    MsgBox "Hallo Welt",
End Sub
```

En MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult

- vbApplicationModal
- vbCritical
- vbDefaultButton1
- vbDefaultButton2
- vbDefaultButton3
- vbDefaultButton4

Deren Inhalt muss in einer Variablen `Msg` abgespeichert sein. Messageboxen dürfen max. 1024 Zeichen enthalten, davon max. 256 in ununterbrochener Reihenfolge.

Messageboxen werden folgendermaßen aufgerufen:

```
MsgBox Prompt, Buttons, Title
```

`Prompt` ist der Inhalt der Messagebox, also der Text, der in der Box erscheinen soll.

`Buttons` ist eine Zahl, die durch Addition folgender verschiedener Werte entsteht:





#### 1. Gruppe:

Mit diesem Wert wird bestimmt, welche Buttons in der Messagebox angezeigt werden.

<code>vbOkOnly</code>	0	Nur "OK" anzeigen.
<code>vbOkCancel</code>	1	"OK" und "Abbrechen" anzeigen.
<code>vbAbortRetryIgnore</code>	2	"Abbrechen", "Wiederholen" und "Ignorieren" anzeigen.
<code>vbYesNoCancel</code>	3	"Ja", "Nein" und "Abbrechen" anzeigen.
<code>vbYesNo</code>	4	"Ja", und "Nein" anzeigen.
<code>vbRetryCancel</code>	5	"Wiederholen" und "Abbrechen" anzeigen.

#### 2. Gruppe:

In dem Messagebox-Fenster kann ein Symbol angezeigt werden, mit dem man auf die Art der Meldung aufmerksam machen kann:

vbCritical	16	 -Symbol anzeigen
vbQuestion	32	 -Symbol anzeigen
vbExclamation	48	 -Symbol anzeigen
vbInformation	64	 -Symbol anzeigen.

### 3. Gruppe:

Man kann selbst festlegen, welcher Button beim Drücken der Enter-Taste reagiert, ohne dass ein anderer Button zuerst ausgewählt wird.

vbDefaultButton1	0	Erste Schaltfläche ist Voreinstellung.
vbDefaultButton2	256	Zweite Schaltfläche ist Voreinstellung.
vbDefaultButton3	512	Dritte Schaltfläche ist Voreinstellung.
vbDefaultButton4	768	Vierte Schaltfläche ist Voreinstellung.

### 4. Gruppe:

vbApplicationModal	0	An die Anwendung gebunden. Der Benutzer muss das Meldungsfeld beantworten, bevor er seine Arbeit an der aktuellen Anwendung wieder aufnehmen kann.
vbSystemModal	4096	An das System gebunden. Alle Anwendungen werden angehalten, bis der Benutzer das Meldungsfeld beantwortet.

Beim Addieren von Zahlen zur Bildung eines Endwerts für das Argument Typ können Sie nur eine Zahl aus jeder Gruppe verwenden. Falls nicht angegeben, ist der Vorgabewert für Typ 0.

### Rückgabewerte:

Konstante	Wert	Beschreibung
vbOK	1	OK
vbCancel	2	Abbrechen
vbAbort	3	Abbruch
vbRetry	4	Wiederholen
vbIgnore	5	Ignorieren
vbYes	6	Ja
vbNo	7	Nein

Einer dieser Werte wird von der MessageBox zurückgegeben, je nachdem, welcher Button gedrückt wird.

Beispiel:

`MsgBox "Schwerer Fehler ist aufgetreten", 2+16+512+0, "Fehler"`  
liefert folgende MessageBox: