
PEARSON
Education
jetzt lerne ich
Start
ohne
Vorwissen
Visual Basic 2005
Der einfache Einstieg in Windows Forms, Objekt-
orientierte Programmierung, XML, Internet
PETER MONADJEMI
Markt+Technik
CD
VBasic 2005
Express Edition

Kapitel 3

jetzt lerne ich

Formulare und Steuerelemente

Haben wir nicht schon zu viel Bürokratie im Alltag, muss es daher auch in diesem Buch um Formulare gehen? Keine Sorge, der Begriff Formular ist nicht die ganz optimale Übersetzung des Begriffs »Form«. Nicht ganz optimal deswegen, weil der Begriff, obwohl die Bedeutung stimmt, eine Reihe von Assoziationen weckt, die nichts mit Programmierung zu tun haben. Ein Form oder Formular ist ein anderes Wort für ein Visual Basic-Fenster, auf dem die Programmiererin, die in diesem Zusammenhang besser Designerin heißen müsste, Steuerelemente (auch Controls genannt) anordnet und damit ein Anwendungsfenster oder Dialogfeld kreiert. Formulare und Steuerelemente sind praktisch eine Erfindung von Visual Basic, inzwischen aber auch in vielen anderen Anwendungen zu finden.

In diesem Kapitel lernen Sie das Prinzip der Formulare und Steuerelemente kennen und lernen, wie Steuerelemente auf einem Formular angeordnet, wie Eigenschaften geändert und Ereignisprozeduren angelegt werden. Visual Basic 2005 bietet in diesem Punkt eine Reihe von wirklich netten Verbesserungen wie z.B. Führungslinien, die automatisch beim Verschieben eines Steuerelements auf dem Formular eingeblendet werden, so dass es sehr viel leichter wird, Steuerelemente bündig zueinander anzuordnen.

3.1 Anatomie eines Formulars

Ein Formular ist zwar im Grunde nicht viel mehr als ein leeres Fenster mit einer Innenfläche, auf der sich Steuerelemente anordnen lassen, doch bei näherer Betrachtung steckt ein wenig mehr dahinter, wie auch Abbildung 3.1 offenbart. Ein Formular besitzt neben der Innenfläche zahlreiche Bedienelemente, wie einen Minimal- und Maximalknopf, sowie ein Systemmenü, das über das kleine Symbol in der oberen linken Fensterecke geöffnet wird. Ob diese Bedienelemente angezeigt

werden, lässt sich über die Eigenschaften des Formulars anzeigen. Hier können Sie auch die Position einstellen, an der das Formular nach dem Start angezeigt werden soll.

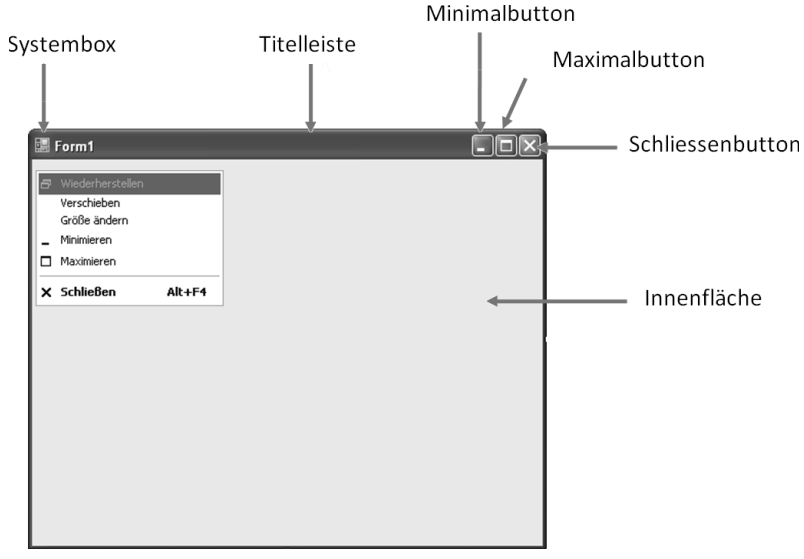

Abb. 3.1: Ein Formular und seine verschiedenen Bedienelemente

3.2 Steuerelemente und die Toolbox

Ein Steuerelement (engl. »control«) ist ein Bedienelement, das auf dem Formular angeordnet wird und das, wenn das Programm später gestartet und das Formular angezeigt wird, an dieser Stelle erscheint. Über Steuerelemente kann der Anwender Eingaben durchführen, Einträge aus einer Liste auswählen, Aktionen ausführen und vieles mehr. Es gibt (in der .NET-Klassenbibliothek) mehrere Dutzend Steuerelemente für fast alle Wünsche. Sollte ein Steuerelement fehlen, findet man es entweder als kostenlosen Download im Internet, im Rahmen eines kommerziellen Tools oder programmiert es einfach selbst, was bei Visual Basic 2005 nicht allzu schwierig ist.

Alle Steuerelemente, die für das Anordnen auf einem Formular zur Verfügung stehen, werden in einem kleinen Fenster der IDE angeboten, das Toolbox heißt. Die Toolbox ist erweiterbar, so dass ihr Umfang mit der Zeit wachsen kann. Wenn Sie Elemente zur Toolbox hinzufügen oder aus der Toolbox entfernen möchten, klicken Sie sie mit der rechten Maustaste an und wählen Sie *Elemente auswählen* – mehr dazu in Kapitel 3.2.4.

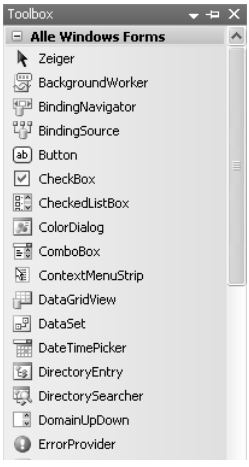

Abb. 3.2: Die Toolbox bietet Steuerelemente und Komponenten für das Anordnen auf einem Formular an.

3.2.1 Steuerelemente auf dem Formular anordnen

Das Anordnen eines Steuerelements auf dem Formular ist sehr einfach – entweder per Doppelklick oder durch Ziehen bei gedrückter linker Maustaste und Loslassen an der gewünschten Position. Ein einmal platziertes Steuerelement kann natürlich jederzeit verschoben werden (übrigens auch während der Programmausführung über seine *Location*-Eigenschaft).

Durch das Anordnen eines Steuerelements auf dem Formular werden in den »Codebehind-Teil« der Formularklasse jene Visual Basic-Befehle eingefügt, die bei der Programmausführung dazu führen, dass das Steuerelement angezeigt wird.

Achten Sie beim Anordnen eines Steuerelements auf die Führungslinien, die immer dann angezeigt werden, wenn sich bereits Steuerelemente auf dem Formular befinden. Damit wird es sehr viel einfacher, Steuerelemente neben- oder untereinander zu positionieren.

3.2.2 Die Rolle der Aufgabenliste

Diese Kleinigkeit kann schnell übersehen werden. Nach dem Platzieren eines Steuerelements auf dem Formular (oder, wenn es nachträglich selektiert wird) erscheint häufig (aber nicht immer) in der rechten oberen Ecke ein kleines Pfeilsymbol. Klicken Sie es an, öffnet sich eine kleine Liste, die Aufgabenliste, mit häufig auszuführenden Aufgaben, die sich auf das Steuerelement beziehen (diese Aufgaben werden zusätzlich auch im unteren Bereich des Eigenschaftenfensters angezeigt). Dies ist sehr praktisch, da Sie nicht im Eigenschaftenfenster nach der passenden Eigenschaft suchen müssen.

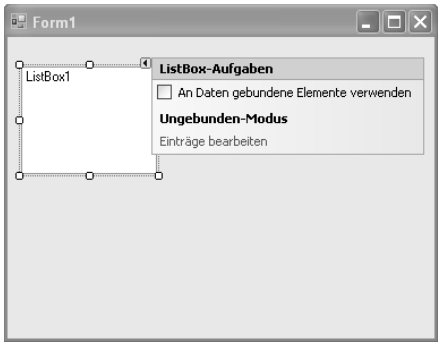

Abb. 3.3: Die Aufgabenliste enthält Aufgaben, die mit dem Steuerelement durchgeführt werden können.

3.2.3 Die Rolle der Komponenten

Neben den Steuerelementen enthält die Toolbox auch Komponenten. Eine Komponente kann zwar ebenfalls auf dem Formular abgelegt werden, sie wird aber nicht auf dem Formular selbst, sondern in einem eigens dafür vorgesehenen Bereich unterhalb des Formulars angeordnet. Komponenten sind während der Programmausführung unsichtbar. Sie stellen eine bestimmte Funktionalität zur Verfügung, die über den Namen der Komponente angesprochen wird. Ein Beispiel für eine Komponente ist `SerialPort`, über die sich die serielle Schnittstelle des Computers sehr einfach ansprechen lässt. Der Umgang mit einer Komponente ist genauso einfach wie der mit einem Steuerelement. Man ordnet die Komponente auf dem Formular an (dass sie dabei in ihrem eigenen Bereich angezeigt wird, spielt keine Rolle), setzt die erforderlichen Werte für einzelne Eigenschaften, fügt eventuell Ereignisprozeduren hinzu und kann die Komponente im Programmcode über ihren Namen ansprechen.

Hinter Komponenten steckt nichts Besonderes. Wie Steuerelemente leiten sie sich von einer Klasse ab. Während sich jedes Steuerelement direkt oder indirekt von der `Control`-Klasse (Namespace `System.Windows.Forms`) ableitet, leitet sich eine Komponente von der `Component`-Klasse (Namespace `System.ComponentModel`) ab. Für die Programmierung hat diese Unterscheidung aber keine direkten Konsequenzen. Genau wie Steuerelemente können auch Komponenten direkt im Quellcode definiert werden. Der (einzige) Vorteil, sie über die Toolbox dem Programm hinzuzufügen, ist, dass sich die Werte ihrer Eigenschaften ein wenig bequemer im Eigenschaftenfenster einstellen lassen (und auf diese Weise überhaupt erst einmal deutlich wird, welche Eigenschaften es gibt).

Abb. 3.4: Neben den Steuerelementen enthält die Toolbox auch Komponenten.

3.2.4 Die Toolbox anpassen

Die Toolbox anzupassen bedeutet, Steuerelemente, Komponenten und ActiveX-Steuerelemente hinzufügen oder zu entfernen. Dies geschieht durch Anklicken der Toolbox mit der rechten Maustaste und der Wahl von *Elemente auswählen*. Es erscheint (nach einer kurzen Verzögerung) ein Dialogfeld, in dem jene Elemente ausgewählt werden, die in der Toolbox angezeigt werden, und jene, die nicht angezeigt werden sollen. Ein Element abzuwählen (wofür es im Allgemeinen keinen Grund gibt), bedeutet nicht, dass es vom PC verschwindet, es bedeutet lediglich, dass es nicht mehr in der Toolbox erscheint.

Hinter einem Eintrag in der Toolbox steht immer eine Klasse, die entweder in der .NET-Klassenbibliothek oder einer externen Bibliothek definiert ist. Der genaue »Aufenthaltsort« spielt keine Rolle.

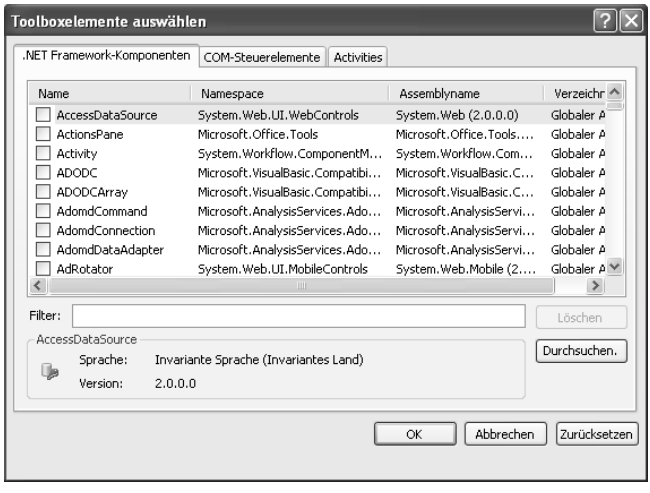

Abb. 3.5: In diesem Dialogfeld wird eingestellt, welche Steuerelemente in der Toolbox angezeigt werden.

3.3 Das Eigenschaftenfenster

Jedes Steuerelement besitzt einen (recht umfangreichen) Satz an Einstellungen, die als *Eigenschaften* (engl. »properties«) bezeichnet werden. Eigenschaften bestimmen unter anderem die Größe und Position des Steuerelements, seine Farbe und vieles mehr. Jede dieser Eigenschaften besitzt von Anfang an einen bestimmten Wert – der Wert der `Location`-Eigenschaft hängt z.B. davon ab, wo das Steuerelement auf dem Formular angeordnet wurde. Die Eigenschaften werden im Eigenschaftenfenster angezeigt und dort eingestellt. Die linke Spalte enthält die Namen der Eigenschaften, die rechte Spalte den aktuellen Wert der Eigenschaften. Die Eigenschaften werden entweder alphabetisch oder nach Gruppen sortiert angezeigt.

Das Eigenschaftenfenster wird z.B. über das *Ansicht*-Menü angezeigt oder durch Drücken der F4-Taste, wenn das Steuerelement zuvor selektiert wurde. Sollten sich keine Eigenschaften einstellen lassen oder das Eintragen eines Wertes zu einer Fehlermeldung führen, liegt es daran, dass das Programm noch läuft. Eigenschaftswerte lassen sich nur ändern, wenn das Programm nicht ausgeführt wird.

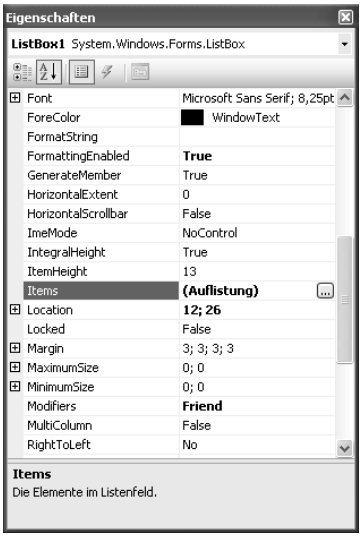

Abb. 3.6: Im Eigenschaftenfenster erhalten Eigenschaften einen neuen Wert.

Einigen Eigenschaften geht im Eigenschaftenfenster ein »+«-Zeichen voraus. In diesem Fall steht die Eigenschaft für eine Klasse, die weitere Eigenschaften besitzt. Ein Beispiel ist die `Font`-Eigenschaft.

3.4 Steuerelemente positionieren

Steuerelemente werden in der Regel mit der Maus verschoben. Enthält das Formular viele Steuerelemente, ist dies unter Umständen ein wenig umständlich. Es gibt daher eine weitere Alternative über das *Format*-Menü (es wird nur angezeigt, wenn das Formular sichtbar ist). Hier besteht die Möglichkeit, alle markierten Steuerele-

mente bezüglich ihrer Größe oder ihrer relativen Position anzupassen. Auch der Abstand zwischen zwei Steuerelementen lässt sich einstellen.

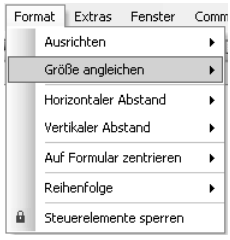

Abb. 3.7: Über das Format-Menü lassen sich Steuerelemente in Gruppen positionieren.

3.4.1 Ein Blick in die Designer-Datei eines Formulars

In diesem Abschnitt wird es etwas fortgeschrittener. Was auf den ersten Blick nicht ersichtlich wird, ist der Umstand, dass jede Formulardatei von einer zweiten Datei begleitet wird, die im Projektmappen-Explorer zunächst nicht angezeigt wird. Diese Datei trägt den Anhang *.designer.vb*. Heißt die Formulardatei z.B. »Form1.vb«, lautet der Name dieser Datei »Form1.designer.vb«. In diese Datei trägt der Formulardesigner die Befehle ein, die beim Platzieren von Steuerelementen auf dem Formular und dem Einstellen von Eigenschaften erzeugt werden. Da sich diese Datei praktisch hinter dem Formular befindet, wird sie auch als *Codebehind-Datei* bezeichnet. Ihr Vorteil ist, dass der Designer die Befehle nicht in die eigentliche Formulardatei einträgt und diese daher übersichtlich bleibt. Möchten Sie die Codebehind-Dateien sehen, müssen Sie die unsichtbaren Dateien im Projektmappen-Explorer sichtbar machen, indem Sie auf das Symbol *Alle Dateien anzeigen* klicken.

Auch die Codebehind-Datei ist eine reguläre Datei. Sie kann daher auch editiert werden, auch wenn dies im Allgemeinen nicht empfohlen wird.

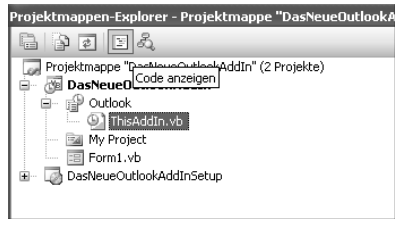

Abb. 3.8: Codebehind-Dateien müssen im Projektmappen-Explorer sichtbar gemacht werden.

Die Codebehind-Datei erweitert einfach die Klasse, die durch die Formulardatei definiert wird. Möglich wird dies über die partiellen Klassen, die neu sind bei Visual Basic 2005. Geht einer Klasse das *Partial*-Schlüsselwort voraus, kann sie auf mehrere Dateien verteilt werden.

3.5 Ereignisse und Ereignisprozeduren

Auch wenn sich ein Steuerelement zur Laufzeit, also nachdem das Programm gestartet wurde, »bedienen« lässt, werden dadurch noch keine Befehle ausgeführt. Soll daher nach dem Anklicken eines Buttons, der Eingabe eines Zeichens in eine TextBox, beim Bewegen des Mauszeigers oder dem Drücken der ↵-Taste etwas passieren, geschieht dies über Ereignisse (engl. »events«) und Ereignisprozeduren, auch Eventhandler genannt. Jedes Steuerelement besitzt einen Satz an Ereignissen, wie z.B. *Click*, *KeyDown* oder *MouseMove*. Soll etwas passieren, wenn das Ereignis während der Programmausführung eintritt, muss das Ereignis mit einer Ereignisprozedur verknüpft werden. Dies kann auf zwei Weisen geschehen:

1. Durch Doppelklick auf das Ereignis in der Kategorie *Ereignisse* im Eigenschaftenfenster .
2. Im Programmcodefenster durch Auswahl des Steuerelements in der linken Auswahlliste und Auswahl des entsprechenden Ereignisses in der rechten Auswahlliste.

In beiden Fällen wird die Ereignisprozedur eingeblendet, die durch das `Handles`-Schlüsselwort am Ende mit dem Event verknüpft wird.

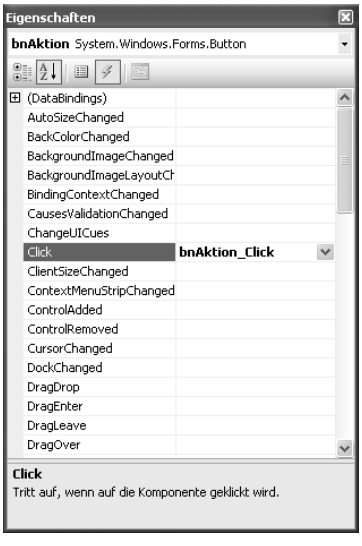

Abb. 3.9: Auch Ereignisse werden im Eigenschaftenfenster angeboten.

Und was soll man in eine Ereignisprozedur hineinschreiben? Nun, ganz einfach jene Visual Basic-Befehle, die immer dann ausgeführt werden sollen, wenn das Ereignis während der Programmausführung eintritt. Sie werden im Laufe dieses Buches zahlreiche Beispiele kennenlernen, so dass sich diese Frage fast von alleine beantwortet.

Möchten Sie in der Ereignisprozedur ein Steuerelement ansprechen, geschieht dies über den Namen des Steuerelements, den dieses über seine `Name`-Eigenschaft erhält.

Dass es, wenn ein Formular einen Button mit dem Namen »Button1« besitzt, im Programmcode auch eine Variable *Button1* gibt, ist bei Visual Basic 2005 keinesfalls selbstverständlich. Es hängt vom Wert der Eigenschaft `GenerateMember` ab. Besitzt diese den Wert `True`, was im Allgemeinen der Fall ist, wird für jedes Steuerelement automatisch eine sogenannte *Membervariable* definiert. Normalerweise gibt es keinen Grund, diese Eigenschaft für ein Steuerelement auf `False` zu setzen.

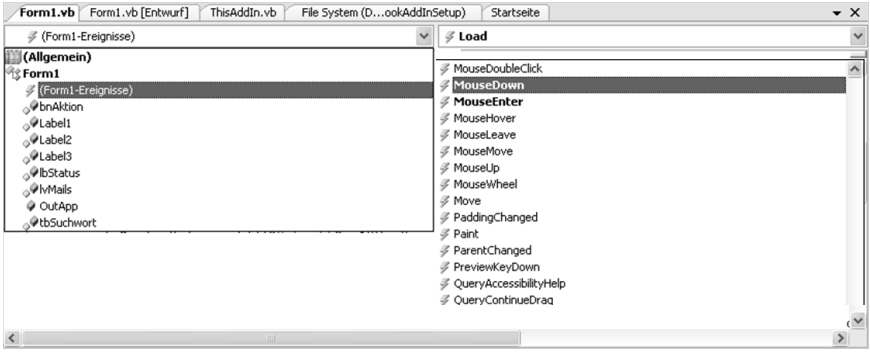

Abb. 3.10: Im Programmcodefenster werden zu jedem in der linken Auswahlliste selektierten Control in der rechten Auswahlliste alle dazugehörigen Events angezeigt.

Lassen Sie sich von der Fülle der Events nicht abschrecken, die es pro Steuerelement gibt. Viele Events sind nicht wirklich wichtig. Der Umgang mit Events ist sehr einfach. Zu den wirklich wichtigen Events gehören in erster Linie `Click`, `Validating`, `KeyUp`, `KeyDown` und `MouseDown`.

3.6 Tipps zu Formularen und Steuerelementen

Zum Schluss ein paar allgemeine Tipps zum Umgang mit Steuerelementen, die Ihnen mit Sicherheit nützliche Dienste leisten.

Wie lässt sich das Formular in der Mitte des Bildschirms anzeigen?

Dafür gibt es die `StartPosition`-Eigenschaft des Formulars.

Wie lässt sich vor dem Laden des Formulars ein Steuerelement selektieren?

Dafür gibt es die `ActiveControl`-Eigenschaft des Formulars. Ihr muss beim Laden des Formulars (also im `Form_Load`-Event) das Steuerelement zugewiesen werden, das aktiv sein soll:

```
Me.ActiveControl = Button2
```

Wie lässt sich die Reihenfolge festlegen, in der die Steuerelemente eines Formulars mit der ⇆-Taste angesteuert werden?

Die Reihenfolge wird durch die *TabIndex*-Eigenschaft eines Steuerelements bestimmt. Am einfachsten lässt sich diese Einstellung für alle Steuerelemente eines Formulars über *Ansicht|Aktivierreihenfolge* einstellen. Mehr dazu in Kapitel 7, wenn es auch um die Rolle des Eingabefokus geht.

3.7 Die wichtigsten Steuerelemente (für den Anfang)

Zum Schluss dieses Kapitels lernen Sie ein paar jener Steuerelemente kennen, mit denen Sie von Anfang an sehr viel zu tun haben werden. Lassen Sie sich von der bunten Vielfalt, vor allem aber von den scheinbar unzähligen Eigenschaften, nicht entmutigen – der Umgang mit den Steuerelementen ist wirklich sehr einfach. Und was die scheinbar endlose Zahl an Eigenschaften angeht – alle Steuerelemente leiten sich von der allgemeinen `Control`-Klasse ab und übernehmen damit deren Eigenschaften, Methoden und Ereignisse. Mit anderen Worten. Die etwa 100+ verschiedenen Mitglieder der `Control`-Klasse finden Sie daher auch bei jedem Steuerelement. Die einzelnen Steuerelemente unterscheiden sich daher nur in relativ wenigen Eigenschaften, Methoden und Ereignissen, die es nur bei einem Steuerelement gibt. Daher gilt das Motto: Kennen Sie eines, kennen Sie praktisch auch alle anderen.

3.7.1 Das Label

Das Label ist dazu da, kleinere Texte auf einem Formular zu platzieren, die vom Anwender nicht geändert werden können. Im Allgemeinen wird das Label als Beschriftung für andere Steuerelemente benutzt (daher auch sein Name). Eine interessante Eigenschaft ist `AutoEllipsis`. Ist sie `True`, werden zu lange Texte mit den üblichen drei Punkten am Ende abgekürzt. Bewegt der Anwender den Mauszeiger über die drei Punkte, wird der komplette Text angezeigt.

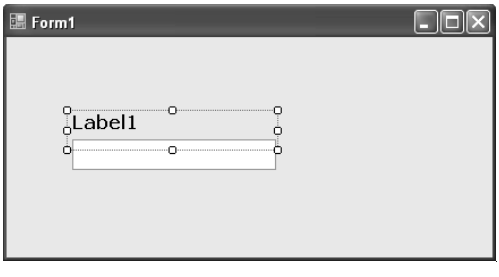

Abb. 3.11: Ein Label beschriftet andere Steuerelemente, wie z.B. eine TextBox.

3.7.2 Der Button

Der Button ist der Knopf, über den im Allgemeinen Aktionen ausgelöst werden. Wird der Button angeklickt, löst dies ein `Click`-Event aus und die `Click`-Ereignisprozedur des Buttons (sofern vorhanden) wird aufgerufen und die dort enthaltenen Befehle werden ausgeführt. Interessant ist, dass sich neben Text auch Bitmaps anzeigen lassen. Weitere Möglichkeiten, die Optik des Buttons zu gestalten, gibt es nicht.

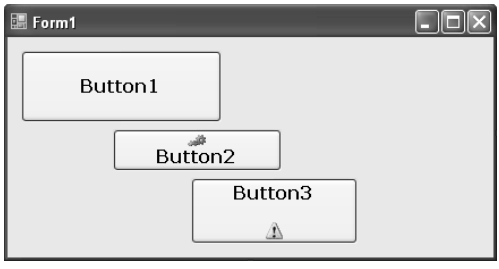

Abb. 3.12: Buttons in verschiedenen Größen – ihr gemeinsames Merkmal ist, dass sie dazu da sind, Aktionen auszulösen.

3.7.3 Die TextBox

Die TextBox ist ein universelles Eingabefeld für Text und Zahlen. Ihr Inhalt wird durch die *Text*-Eigenschaft repräsentiert. Soll die Eingabe mehrzeiliger Texte möglich sein, muss die `MultiLine`-Eigenschaft auf `True` gesetzt werden. In diesem Fall empfiehlt es sich, über die `Scrollbars`-Eigenschaft zusätzlich eine Bildlaufleiste hinzuzufügen.

Der Text der TextBox kann jeweils nur in einer Farbe und einer Schriftart angezeigt werden. Möchte man jeden Buchstaben einzeln formatieren, muss stattdessen die RTFTextBox zum Einsatz kommen. Sie wird in Kapitel 9 vorgestellt.

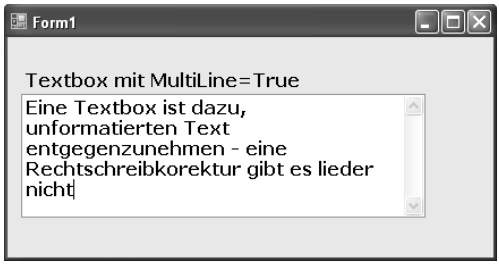

Abb. 3.13: Die TextBox ist für die Eingabe von Text da.

Die TextBox erlaubt die Eingabe beliebiger Texte. Soll der Anwender die Eingabe nur in einem bestimmten Format durchführen können, etwa dem einer Telefonnummer oder Kreditkartenummer, ist die *MaskedEditTextBox* unter Umständen besser geeignet.

3.7.4 Der Radiobutton

Der Radiobutton ist ein kleiner Knopf, der entweder eingedrückt ist oder nicht. Da er sich wie die Knöpfe an den uralten Radios verhält, kann in einer Gruppe von Radiobuttons jeweils nur ein Button eingedrückt sein – wird ein anderer Button eingedrückt, springt der bis dahin eingedrückte Button wieder in seine Ausgangsposition (über das GroupBox-Steuerelement werden Radiobuttons zu Gruppen zusammengefasst). Ob ein Radiobutton eingedrückt ist oder nicht gibt seine `Checked`-Eigenschaft an.

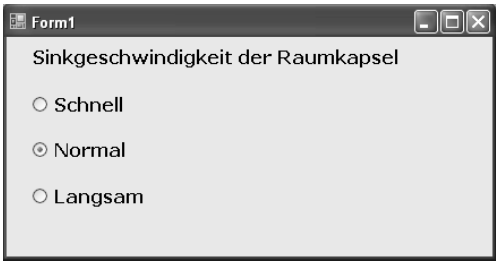

Abb. 3.14: Bei einer Gruppe von Radiobuttons kann immer nur ein Button eingedrückt sein.

3.7.5 Die CheckBox

Die CheckBox (auch Kontrollkästchen genannt) zeigt neben einem Namen ein kleines Kästchen an, das angekreuzt werden kann, um eine Option zu wählen. Anders als beim Radiobutton können beliebig viele Kontrollkästchen angekreuzt sein. Wie beim Radiobutton wird der aktuelle Zustand (angekreuzt oder nicht) über die `Checked`-Eigenschaft gesetzt oder abgefragt. Anders als beim Radiobutton, der nur gedrückt oder nicht gedrückt sein kann, gibt es bei der CheckBox noch einen dritten Zustand, der als unbestimmt bezeichnet wird. In diesem Fall ist die CheckBox leicht abgegraut dargestellt. Ein solcher Zustand liegt zum Beispiel vor, wenn noch keine Auswahl getroffen wurde und keine Vorauswahl existiert. Neben der `Checked`-Eigenschaft gibt es daher noch die `CheckState`-Eigenschaft, mit der sich der Zustand des Kästchen insgesamt (und nicht nur des Auswahlhäkchens) abfragen lässt.

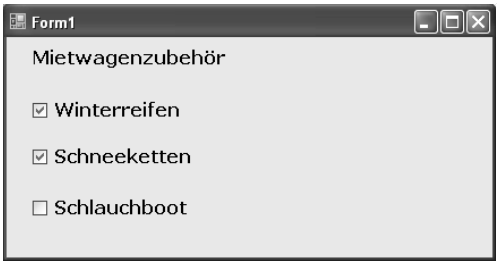

Abb. 3.15: CheckBoxen zeigen mit einem Häkchen an, ob eine Option gewählt wurde oder nicht, oder ob noch keine Auswahl getroffen wurde.

3.7.6 Die ListBox

Die ListBox ist dazu da, eine Liste von Namen anzuzeigen, von denen der Anwender einen oder mehrere Namen selektieren kann. Die in der ListBox angezeigten Namen werden über die `Items`-Eigenschaft repräsentiert. `Items` steht für ein Auflistungsobjekt (mehr dazu in Kapitel 7). Über dessen `Add`-Methode wird der Liste ein Objekt hinzugefügt:

```
Listbox1.Items.Add ("Happy, Monday")
```

Auch wenn es so erscheinen mag, als enthielte die ListBox nur Texte, sind es in Wirklichkeit Objekte. Es ist daher problemlos möglich, pro Eintrag ein komplettes Objekt zur `Items`-Auflistung hinzuzufügen, wobei in diesem Fall unter Umständen zusätzlich über die `DisplayMember`-Eigenschaft festgelegt werden muss, welche Eigenschaft des Objekts in der ListBox angezeigt werden soll. Wird ein Eintrag in der ListBox selektiert, löst dies zuerst ein `Click`- und anschließend ein `SelectedIndexChanged`-Ereignis aus. Die Nummer des aktuell selektierten Eintrags liefert die `ListIndex`-Eigenschaft, wobei die Nummerierung bei 0 beginnt. `-1` bedeutet, dass kein Eintrag selektiert wurde.

Soll das Selektieren mehrerer Einträge möglich sein, muss die `SelectionMode`-Eigenschaft auf den Wert `SelectionMode.MultiSimple` gesetzt werden. Hinter diesem Namen steht lediglich eine Zahl (2). In diesem Fall stehen die Nummern der selektierten Einträge über die `SelectedItems`-Eigenschaft zur Verfügung. Eine enge Verwandte der ListBox ist die `CheckedListBox`, die für jeden Eintrag eine CheckBox anzeigt.

Es ist bei der ListBox leider nicht möglich, jeden Eintrag in einer unterschiedlichen Farbe oder Schriftart anzuzeigen (dazu müsste eine neue Klasse definiert werden, die sich von der `ListBox`-Klasse ableitet, und bei der die `DrawMode`-Eigenschaft auf den entsprechenden Wert gesetzt ist). Die normale ListBox kann auch keine Bitmaps anzeigen. Aber auch das ließe sich über eine »selbst gezeichnete« ListBox mit relativ wenig Aufwand realisieren.

Es ist ebenfalls nicht möglich, mehrere unabhängige Spalten anzuzeigen. Es ist lediglich möglich, die Liste auf mehrere Spalten zu verteilen, so dass sie auch in der Horizontalen gescrollt werden kann. Wer das möchte, muss auf das ListView-Steuerelement ausweichen.

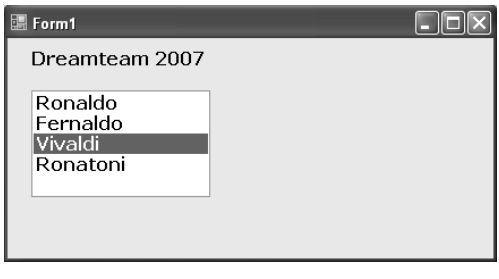

Abb. 3.16: Eine ListBox zeigt eine Reihe von Einträgen an, die über die Items-Eigenschaft zur Verfügung gestellt werden.

3.7.7 Die ComboBox

Die ComboBox verdankt ihren vielleicht beim ersten Lesen ein wenig ungewöhnlich klingenden Namen einem simplen Umstand: Sie ist eine Kombination aus TextBox und ListBox. Es lassen sich sowohl Namen eingeben als auch aus einer Liste auswählen. Dies ist z.B. bei einer Suchabfrage sehr praktisch, bei der der Suchbegriff entweder eingegeben oder aus einer Liste bereits eingegebener Begriffe ausgewählt wird (wie Sie eine List- oder ComboBox mit solchen Begriffen zur Autovervollständigung füllen, erfahren Sie in Kapitel 7).

Über die `DropDownStyle`-Eigenschaft kann eingestellt werden, ob eine Eingabe in die TextBox möglich ist, und ob die Liste aufklappbar sein oder immer angezeigt werden soll.

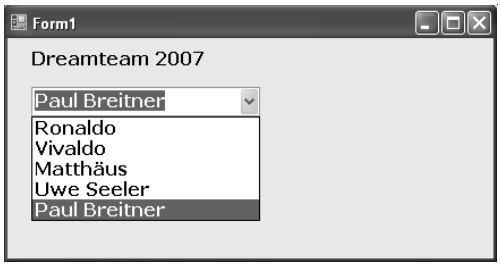

Abb. 3.17: Eine ComboBox ist eine Kombination aus TextBox und ListBox.

3.7.8 Die PictureBox

Die PictureBox ist dazu da, Bitmaps anzuzeigen, wobei alle wichtigen Formate erlaubt sind. Auch animierte GIFs, die nicht nur auf Webseiten für ein wenig Bewegung sorgen. Über die `SizeMode`-Eigenschaft wird eingestellt, wie die Bitmap positioniert werden soll.

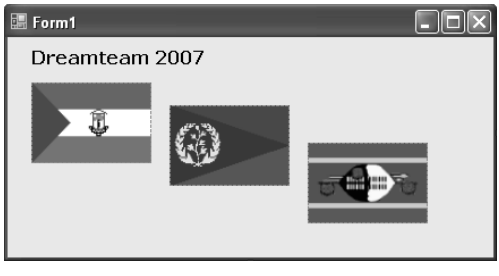

Abb. 3.18: Das PictureBox-Steuerelement zeigt Bitmaps an.

Eine PictureBox erhält über ihre `Image`-Eigenschaft eine Bitmap zugewiesen. Das Dialogfeld ist beim ersten Mal ein wenig irritierend, da es scheinbar keine Möglichkeit gibt, eine Datei direkt auszuwählen. Das liegt daran, dass bei Visual Basic 2005 Bitmaps in erster Linie aus der Ressourcendatei stammen sollen, aber nicht müssen. Wer eine Bitmap-Datei direkt auswählen möchte, wählt zuerst die Einstellung *Lokale Ressource* und klickt dann auf *Importieren*.

Abb. 3.19: Eine Bitmap stammt entweder aus einer Ressourcendatei oder direkt aus einer Datei.

Auch wenn es möglich wäre, bietet das PictureBox-Steuerelement keinerlei »Special Effects«, wie ein Rotieren, Invertieren oder Ein-/Ausblenden der angezeigten Bitmap. Diese müssen mithilfe der Klasse im Namespace `System.Drawing` nachträglich eingebaut werden.

3.7.9 Der Rauf- und Runterzähler (NumericUpDown)

Das UpDown-Steuerelement (zu Deutsch »Rauf- und Runterzähler«) ist dazu da, eine Zahl über zwei kleine Pfeiltasten in der Größe verändern zu können. Das ist oft für den Anwender praktischer, als die Zahl in eine TextBox eingeben zu müssen. Der erlaubte Bereich wird über die Eigenschaften `MinValue` und `MaxValue` eingestellt. Der aktuelle Wert steht über die `Value`-Eigenschaft zur Verfügung.

Abb. 3.20: Das UpDown-Steuerelement erlaubt das Rauf- und Runterzählen einer Zahl.

3.8 Zusammenfassung

Bei der Umsetzung von Windows-Programmen mit Visual Basic dreht sich am Anfang fast alles um Formulare und Steuerelemente. Ein Formular ist ein anderer Name für ein Fenster, ein Steuerelement (das ebenfalls eine »Art« Fenster darstellt) ist ein typisches Bedienelement, wie ein Button zum Auslösen einer Aktion, eine TextBox für die Eingabe von Text oder ein Label als Beschriftung für andere Steuerelemente. Damit beim Anklicken des Buttons oder dem Bewegen des Mauszeigers im Programm etwas passiert, gibt es die Ereignisprozeduren. Es sind Visual Basic-Prozeduren, die mit einem Event verknüpft sind und daher mit Eintreten des Events aufgerufen werden. Auch wenn die Toolbox, in der alle Steuerelemente und Komponenten versammelt sind, viele Einträge anbietet, für die ersten Schritte mit Visual Basic kommen Sie mit relativ wenigen Steuerelementen aus.

3.9 Fragen

1. **Wie unterscheidet sich ein (Visual Basic-)Formular von einem Windows-Fenster?**
2. **Wie lässt sich ein Steuerelement, wie eine TextBox, am einfachsten mehrere Dutzend Mal auf einem Formular platzieren?**
3. **Was ist der schnellste Weg, um zu erreichen, dass vier nebeneinander platzierte TextBoxen die gleiche Größe und den gleichen Abstand zueinander besitzen?**
4. **Welche Rolle spielt die Ereignisprozedur bei Visual Basic?**
5. **Was ist der einfachste Weg, den Prozedurrahmen für eine Ereignisprozedur in das Programmcodefenster einzufügen?**
6. **Aus der Toolbox sind auf einmal wichtige Steuerelemente und Komponenten verschwunden (das kann passieren). Was ist der einfachste Weg, um sie wiederherzustellen?**

Die Lösungen zu den Fragen finden Sie in Anhang C.