

## Der Webbrowser als Bildanzeige

Die Handhabung von Bildern in Formularen war und ist ein Dauerbrennerthema in vielen Foren. Das genuine Bildsteuerelement von Access weist auch in der zeitweilig modernisierten Version etliche Beschränkungen auf. Das Anlage-Steuerelement ist zwar pfiffig, aber dessen Programmierung fällt reichlich umständlich aus. Doch zu unserem Favoriten haben wir schon früher einmal ein alternatives Steuerelement auserkoren: das Webbrowser Control.

### Beispieldatenbank

Die Beispiele dieses Artikels finden Sie in der Datenbank **1702\_BildWeb.accdb**.

### Das Webbrowser Control als Universalcontainer

In der Ausgabe 04/2016 von Access Basics, **Elemente für die Bildanzeige**, verglichen wir die möglichen Methoden, um Bilder in Formularen und Berichten zur Anzeige zu bringen. Neben dem **Bildsteuerelement**, dem **Anlage-Steuerelement**, einer zweckentfremdeten **Schaltfläche** und dem **MSForms Image Control** kam dort auch ein **Webbrowser-Steuerelement** zum Einsatz. Es stellte sich heraus, dass Letzteres sich sowohl qualitativ, wie auch von der Programmierung her, sehr gut für diesen Zweck eignet.

Tatsächlich ist das **Webbrowser Control** als Host für den **Internet Explorer** eine Universalwaffe! Überall da, wo der Fundus von Access-Steuerelementen versagt, lässt sich über den Webbrowser fast alles realisieren. Schließlich gibt es ja inzwischen sehr komplexe Applikationen, die rein webbasiert arbeiten; und dorthin geht ohnehin der Trend. Liebäugeln Sie also eh schon mit der Portierung Ihrer Datenbank zu einer Weblösung, so ist der integrierte Webbrowser unter Access ein nützlicher Anfang, um sich mit Webprogrammierung vertraut zu machen.

Der Vorteil des **Webbrowser Controls** ist, dass es den gewaltigen Umfang der **HTML-Bibliothek** von Windows offenlegt. Über diese können Webseiten programmatisch nicht nur über HTML gesteuert werden, sondern auch über Objekte und deren Eigenschaften, sowie Methoden. Praktisch jedes **HTML-Tag** (Element) hat ein Pendant als Objektklasse in

der **HTML-Bibliothek** von Microsoft. Über weitere Hilfsbibliotheken, auf die Sie verweisen können, wie die **Microsoft XML Library** oder die **Shell-Bibliothek**, haben Sie schließlich den **Internet Explorer** vollständig im Griff.

### Webbrowser vs. Anlagesteuerelement

Das Anlagesteuerelement scheint alle Lösungen zur Bildanzeige dadurch auszusteichen, dass es die Bilddaten binär aus einer Tabelle mit Anlagefeld hervorholen kann. Mit seinem **Popup-Toolbar** ist es zudem ohne jeglichen Programmcode möglich, Bilddateien zu laden und direkt im gebundenen Feld zu speichern.

Ein Bild im Webbrowser hingegen verlangt nach einer **URL**, die den Pfad zur Datei enthalten muss. Ein Abspeichern der Bilddatei selbst in einem Tabellenfeld scheint damit verwehrt. Bestenfalls könnte man die Datei binär in ein **OLE-Feld** laden, um dieses später wieder als temporäre Datei zu exportieren und deren Pfad als **URL** dem Webbrowser Control zu übergeben.

Tatsächlich jedoch gibt es eine Lösung, die diesen Umweg obsolet macht. Das Formular **frmBilder** der Beispieldatenbank zeigt, wie auf diese temporäre Datei über eine spezielle Kodierung der Bilddaten verzichtet werden kann. Damit steht diese Lösung dem Anlagesteuerelement in nichts nach. Sie erfordert lediglich einige überschaubare VBA-Routinen.

### Bildtabelle

Die Tabelle zur Speicherung sowohl des Bildpfads, wie auch der Binärdaten, nennt sich **tblBilder** und zeigt sich in der Datenblattansicht wie in Bild 1. Das

ID	Datei	BildName	Binaer
1	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Blumenkohl.bmp	Bild	
2	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Blumenkohl.jpg	Bild	
3	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Blumenkohl.png	Bild	
4	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\clock.jpg	Bild	
5	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Garden.jpg	Bild	
6	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\igel.jpg	Bild	
7	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Mond.png	Bild	
8	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\clock.tif	Bild	
(Neu)		Bild	

Bild 1: Die Tabelle **tblBilder** enthält zunächst nur die Pfade zu den Bilddateien im Feld **Datei** und einen Default-Namen

Feld **Datei** nimmt den vollen Pfad der Bilddatei auf. **BildName** dient zur optionalen Angabe einer Bezeichnung des Bilds und ist im Entwurf der Tabelle auf den Text **Bild** voreingestellt. Schließlich sollen die Binärdaten des Bilds im Feld **Binaer** aufgenommen werden, das hier aber nicht, wie eher üblich, als **OLE-Feld** daherkommt, sondern als **Memo-Feld**.

Den Grund erfahren Sie später. Der Entwurf der Tabelle ergibt sich damit gemäß Bild 2. Der Autowert **ID** als Primärschlüssel vervollständigt die Tabelle.

Die angezeigten Datensätze kommen übrigens beim Aufruf des **Intro**-Formulars zustande. Hier ist eine kleine Routine **UpdatePics** hinterlegt, die alle Bilddateien, die sich im Datenbankpfad befinden, enumeriert und in der Tabelle ablegt. Bei Ihnen wird der Inhalt der Tabelle deshalb wohl anders aussehen.

Die Tabelle stellt die Basis für das Formular **frmBilder** dar, wobei dort das Webbrowser-Steurelement an das Feld **Datei** gebunden ist. Damit zeigt es die Bilder auch schon ordnungsgemäß an, falls die Pfadangaben stimmen. Gleichzeitig kodiert eine Prozedur die Bilddaten und speichert sie im Feld **Binaer** (siehe Bild 3). Bei erneutem Aufruf des Datensatzes im For-

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	
Datei	Text	
BildName	Text	
Binaer	Memo	

Feldeigenschaften

Allgemein | Nachschlagen

Feldgröße: 255

Bild 2: Entwurfsansicht der Tabelle **tblBilder**

mular schaut der Code im Ereignis **Beim Anzeigen (Form\_Current)** nach, ob das Feld **Binaer** leer ist. Da das nun nicht mehr der Fall ist, verwendet es die Daten des Memo-Felds zur Bildanzeige, ohne eine Datei lesen zu müssen. Soweit der Überblick! Schauen wir uns das im folgend im Detail an...

### Bild im Webbrowser im Formular

Der Entwurf des Formulars **frmBilder** (siehe Bild 4) ist so schlicht, wie die zugrundeliegende Tabelle. Die Textfelder oben im Formularkopf sind an die entsprechenden Tabellenfelder **BildName** für die Bezeichnung und **Datei** für den Pfad zur Bilddatei gebunden. Über den blauen Button **cmdLoad** kann ein Dateiauswahldialog geöffnet werden, um ein Bild aus dem Dateisystem laden zu können. Das rote Label mit der Beschriftung **Binär** ist auf Unsichtbar gestellt und

ID	Datei	BildName	Binaer
56	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Blumenkohl.bmp	Bild	data:image/bitmap;base64,Qk0sZgMAAAAAADYAAAAoAAACwEABUBAAAB/
57	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Blumenkohl.jpg	Bild	data:image/jpeg;base64,9j/4Rq3RXhpZgAATU0AKgAAAQADAAEAAAAMAAAABA
58	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Blumenkohl.png	Bild	data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAVoAAAFdCAIAAADN
59	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\clock.jpg	Bild	data:image/jpeg;base64,9j/4S4IRXhpZgAATU0AKgAAAQADAAEAAAAMAAAABA
60	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Garden.jpg	Bild	data:image/jpeg;base64,9j/4S5nRXhpZgAATU0AKgAAAQADAAEAAAAMAAAABA
61	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\igel.jpg	Bild	data:image/jpeg;base64,9j/4SVeRXhpZgAASUkqAAgAAAAQAAABAwBAAAAE
62	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\Mond.png	Bild	data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAUAAAAFACIAAABCI
63	E:\ABASICS\AccessBasics_2017_02\BildanzeigWeb\Demo\clock.tif	Bild	data:image/tiff;base64,SukqAAgAAAAOAAABAAABAAAgEAAAEBAABAAABAA

Bild 3: Das Memo-Feld **Binaer** füllt sich später beim Aufruf eines Bilds im Webbrowser des Formulars **frmBilder**

wird erst per Code zum Vorschein gebracht.

Im Detailbereich des Formulars befindet sich in voller Ausdehnung nur das Webbrowser-Steuerelement **ctlWeb**. Seine Verankerung ist auf **Quer und nach unten dehnen** eingestellt, damit es alle Größenänderungen des veränderbaren Formularrahmens mitmacht. Links oben steht in ihm auch schon der Steuerelementinhalt, der jedoch nicht den Namen des Tabellenfelds anzeigt, sondern den Ausdruck **= "about:blank"**.

Tatsächlich ist das Control nicht wirklich an die Tabelle gebunden, sondern nur indirekt über den VBA-Code des Formulars. Denn das Webbrowser-Steuerelement von Access **muss** nicht zwingend an ein Feld gebunden sein. Sie können auch eine feste **URL** im Eigenschaftenblatt unter **Steuerelementinhalt** eintragen, die dann standardmäßig angezeigt wird. Mit dem vorliegenden Eintrag erhalten Sie immer eine leere Seite.

Im Fußbereich des Formulars gibt es einige zusätzliche Elemente, die Einfluss auf die Anzeige des Bilds haben. Ist das Kontrollkästchen **Originalgröße** aktiviert, so wird das Bild in seiner vollen Ausdehnung gerendert. Andernfalls passt es sich, je nach Breite oder Höhe, genau in die Fläche des Webbrowser Controls ein.

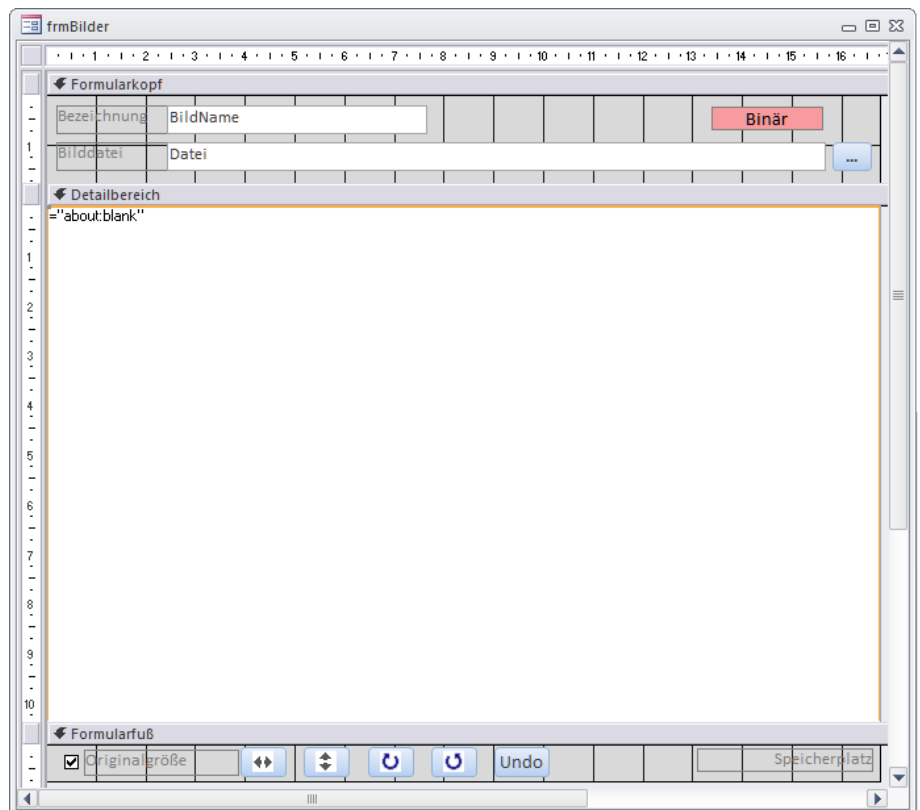


Bild 4: Das Webbrowser-Control in der Mitte des Formularentwurf von **frmBilder**

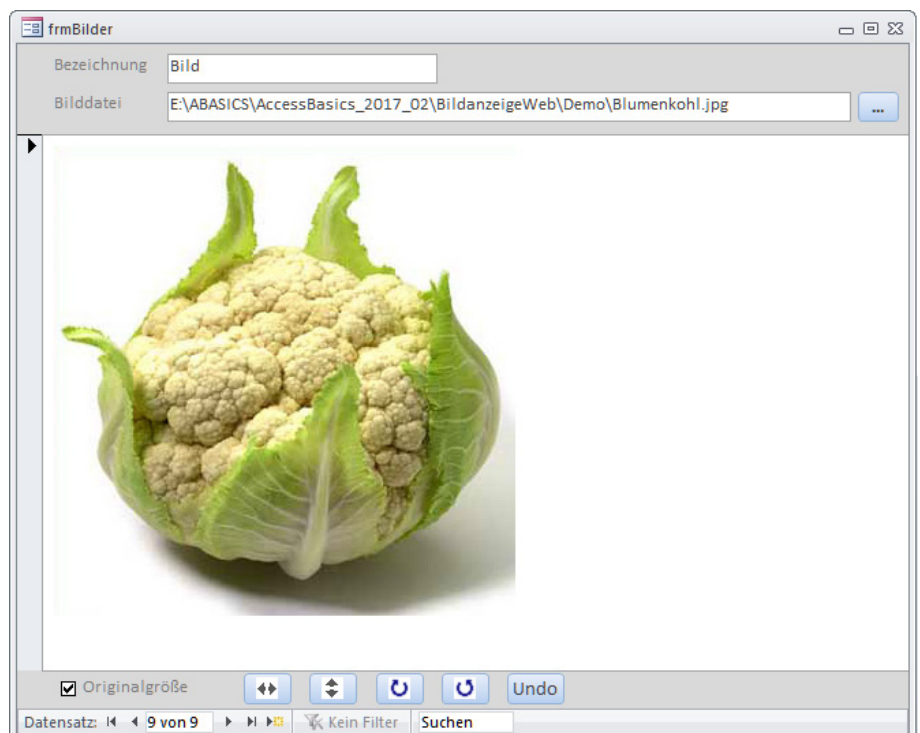


Bild 5: So präsentiert sich das Formular **frmBilder** dann zur Laufzeit

Die fünf Schaltflächen daneben demonstrieren, dass sich über das **HTML-Objektmodell** auch auf sehr einfache Weise Bildeffekte erzielen lassen. Hier können Sie das Bild horizontal oder vertikal spiegeln, sowie im oder gegen den Uhrzeigersinn um 90° drehen.

Der **Undo**-Button macht solche Änderungen rückgängig. In Bild 5 sehen Sie ein geladenes Beispielbild. Hier hat das Control das Bild aus dem Dateipfad eingelesen. Im Hintergrund aber wurden dessen Daten, wie erwähnt, in die Tabelle geschrieben.

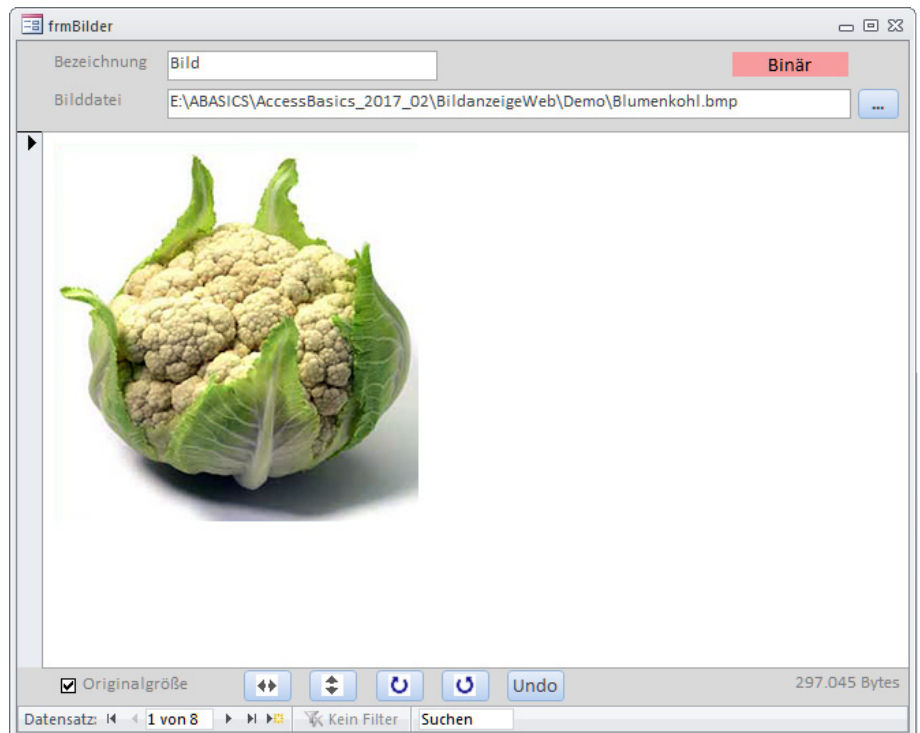


Bild 6: Ist ein Bild bereits eingelesen, so zeigt sich das Label **Binär** oben rechts

So kommt es bei erneutem Aufruf des Datensatzes zur Darstellung in Bild 6: Das **Binär**-Label oben wird eingeblendet und zusätzlich gibt rechts unten ein weiteres Label Auskunft über den Speicherbedarf des Bilds im Memo-Feld der Tabelle. Da es sich im Beispiel um ein Bitmap-Format handelt, im Textfeld **Bilddatei** zu erkennen, ist dieser ziemlich hoch.

Kommt das Bild hingegen binär aus der Tabelle, so zeigt die Meldung, wie in Bild 8, deren Memo-Text ausschnittsweise an. Dafür kommt keineswegs anderer Code zum Einsatz. Tatsächlich handelt es sich bei

Bei Linksklick auf das Bild erscheint eine Meldung, die direkt die aktuell im Webbrowser Control hinterlegte **URL** ausgibt. Bei aus dem Dateisystem geladenem Bild ergibt sich dabei etwa die Meldung in Bild 7. Hier hat der **Internet Explorer** den Pfad in das **file**-Protokoll umgewandelt und die **Backslashes** automatisch in **Slashes** umgewandelt.

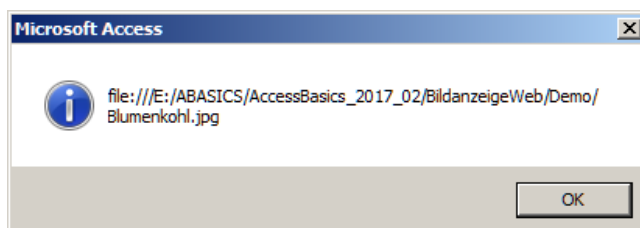


Bild 7: Info-Meldung beim Klick auf das Bild

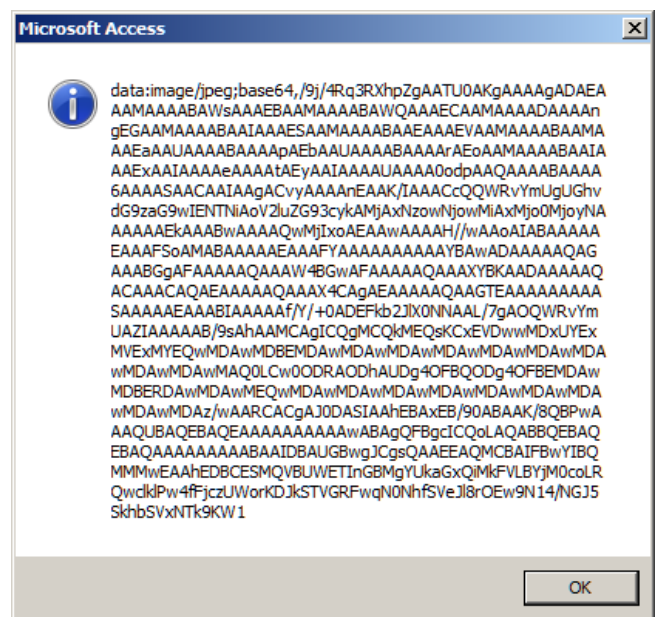


Bild 8: Info-Meldung bei binärem Inhalt des Bilds

dem langen String unmittelbar um die **Source-URL** des Browsers!

Bei Rechtsklick auf das Bild erscheint ein Kontextmenü, wie sonst im **Internet Explorer** auch (siehe Bild 9). Leider hat man auf dessen Einträge ohne größere Umstände keinen Einfluss. Man kann es allerdings mit wenig Code auch deaktivieren, wie wir noch sehen werden. Wir haben dies zunächst nicht getan, weil damit deutlich wird, welche Möglichkeiten das Webbrowser Control zur Verfügung stellt. So etwa lässt sich das Bild, egal, ob als Datei oder binär geladen, als Datei abspeichern, über den E-Mail-Client versenden oder ausdrucken.

Beim Speichern hat man die Wahl zwischen mehreren Dateiformaten, wie Bild 10 zeigt. Dabei scheint es keine Rolle zu spielen, welches Ursprungsformat das Bild hat. Auch bei **JPG** werden beim Abspeichern nur **PNG** und **BMP** angeboten.

Auch ein Kopieren des Bilds in die Zwischenablage ist möglich, so dass es danach etwa in **Word** wieder eingefügt werden kann. Allerdings kopiert das Control gleich mehrere Formate in die Zwischenablage, wie etwa neben dem Bitmap auch den Quelltext des Browsers. Beim Einfügen in Word etwa müssen Sie die Funktion **Einfügen | Inhalte einfügen...** im **Start**-Tab des Ribbon bemühen, was den Dialog in Bild 11 auf den Plan ruft. Hier ist **HTML** voreingestellt. Sie müssen stattdessen den Eintrag **Geräteunabhängige Bitmap** auswählen, um das Bild in ein Dokument zu bekommen.

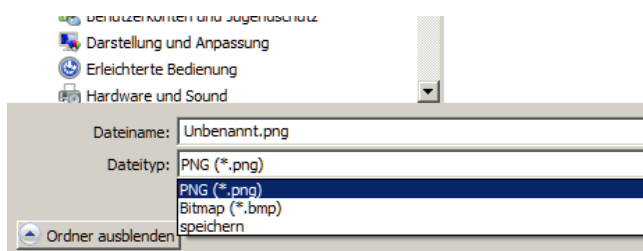


Bild 10: Beim Export des Bilds in eine Datei stellt der Webbrowser diese zwei Dateiformate zur Verfügung

Am Störendsten sind im Kontextmenü sicherlich die Einträge weiter unten, die einen Anwender, der gar nicht weiß, dass er einen Browser für die Bildanzeige vor sich hat, verwirren. Deshalb wäre ein Abschalten des Kontextmenüs ratsam. Denn alle darin enthaltenen Anweisungen lassen sich gegebenenfalls auch recht einfach per Code über das Objektmodell des Browsers auslösen. Mit einigen Klimmzügen

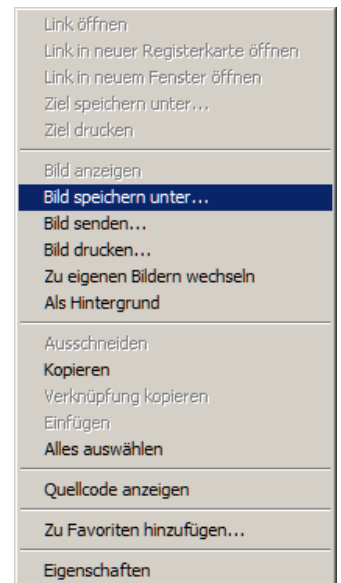


Bild 9: Das Kontextmenü des Webbrowser-Bilds

ließe sich das Kontextmenü sogar durch ein eigenes in der Datenbank angelegtes ersetzen. Da Microsoft leider seit mehreren Versionen keinen Editor für solche Menüs mehr in Access einbaut und deren Anlage deshalb rein programmtechnisch erfolgen muss, lassen wir diese Möglichkeit hier außen vor.

### Programmierung der Bildanzeige

Gehen wir Schritt für Schritt vor. Schalten Sie im Formular auf einen neuen Datensatz. Die Bildfläche bleibt weiß. Über den Button für den Dateiauswahl-dialog (**cmdLoad**) öffnen Sie eine Bilddatei. Der dem **Click**-Ereignis hinterlegte Code steht in Listing 1 und

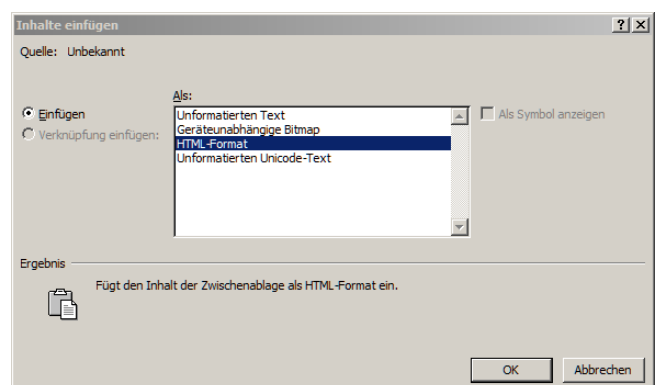


Bild 11: Einfügen des kopierten Bilds in MS Word



verwendet das **FileDialog**-Objekt von Access oder, genauer, der Office-Bibliothek.

Hier wird zunächst die Objektvariable **ODlg** auf das von Access zur Verfügung gestellte **FileDialog**-Objekt gesetzt, wobei gleich der Dateiauswahlmodus über die Konstante **msoFileDialogPicker** angefordert wird. Im folgenden **With**-Block werden nun einige Einstellungen für den Dialog vorgenommen. Mehrfachauswahl etwa ist untersagt (**AllowMultiSelect**), die Ansicht wird auf Details gesetzt (**InitialView**), der Baum navigiert zum Datenbankverzeichnis (**InitialFileName**) und die erlaubten Dateierweiterungen sind auf einige Bildformate beschränkt, mit denen der Internet Explorer klarkommt (**Filters**). Die Methode **Show** zeigt dann den Dialog erst an. Da der Dialog modal ist, hält der Code an dieser Stelle an. Hat der Anwender eine Datei ausgewählt, so quittiert die Funktion dies mit der Rückgabe **True**, wonach die Variable **sFile** mit dem ersten und einzigen Element der **SelectedItems**-Auflistung bestückt wird. Andernfalls ist die Länge des Strings in **sFile** gleich 0 und die Prozedur wird verlassen.

Weiter wird nun der Dateipfad in das Textfeld **txtDatei** des Formulars geschrieben. Die Dateierweiterung wird über einen Ausdruck mithilfe der **Mid**- und **InstrRev**-Funktionen ermittelt und in die Variable **sExt** gespeichert. Soweit die Vorarbeiten. Für das Verständnis der restlichen Zeilen der Routine, die im Listing ausgelassen wurden, müssen erst die Vorgänge beim Laden und Anzeigen des Formulars näher beleuchtet werden.

### Initialisierung des Webbrowser Controls beim Laden des Formulars

In **Form\_Load** steht nur eine Zeile:

```
Set oBrowser = Me!ctlWeb.Object
```

Das Webbrowser-Steuerelement hat im Formular den Namen **ctlWeb**. Seine **Object**-Eigenschaft gibt den eigentlichen Webbrowser zurück, die der Objektvariablen zugewiesen wird. Deren Deklaration befindet sich im Modulkopf des Formulars:

```
Private Sub cmdLoad_Click()  
    Dim ODlg As Office.FileDialog  
    Dim sFile As String  
    Dim sExt As String  
    Dim bin() As Byte  
  
    Set ODlg = Application.FileDialog(msoFileDialogFilePicker)  
    With ODlg  
        .AllowMultiSelect = False  
        .InitialView = msoFileDialogViewDetails  
        .InitialFileName = CurrentProject.Path  
        .Filters.Add "Bild-Dateien", "*.jpg;*.jpeg;*.png;*.bmp;*.tif"  
        If .Show Then  
            sFile = .SelectedItem(1)  
        End If  
    End With  
    If Len(sFile) = 0 Then Exit Sub  
  
    Me!txtDatei = sFile  
    sExt = Mid(sFile, InstrRev(sFile, ".") + 1)  
    ...  
    ... (Fortsetzung folgt) ...  
End Sub
```

**Listing 1:** Beim Laden einer Bilddatei über den Button **cmdLoad** spielt sich diese Prozedur ab (erster Teil)

```
Private WithEvents oBrowser As WebBrowser
```

Die Variable ist also vom Typ **Webbrowser** und das Statement **WithEvents** gibt an, dass dieses Objekt Ereignisse auslösen kann. Nun werden Sie eine Klasse **Webbrowser** im Objektkatalog vergeblich suchen. Die Access-Bibliothek enthält deren Definition nicht. Der Typ ist nämlich in einer Bibliothek zu finden, die Sie erst einmal in die Verweise des VBA-Projekts laden müssen: die **Microsoft Internet Controls (shdocvw)**. Diese Bibliothek ist das Kernstück des **Internet Explorers**. Markieren Sie nun im Objektkatalog die Klasse **Webbrowser**, so stellen Sie fest, dass diese fast die gleichen Methoden und Eigenschaften aufweist, wie die Klasse **WebbrowserControl** von Access, nur dass jene viele zusätzliche Eigenschaften zeigt, die sich auf die Gestalt des Steuerelements im Formular beziehen. Microsoft hat im **Webbrowser**-

**Control** die Eigenschaften von Access-Steuerelementen mit denen des Internet Explorers vermengt.

Nach dem Laden des Formulars navigiert das Steuerelement nun zur **URL about:blank**, wie das in seiner Eigenschaft **Steuerelementinhalt** festgelegt wurde. Also wird eine leere Seite in ihm angezeigt. Das Rendern dieser Seite benötigt dennoch etwas Zeit, weshalb im Ereignis **Beim Anzeigen** des Formulars (**Form\_Current**) über die Funktion **WebDocReady** erst die Beendigung des Vorgangs abgewartet wird (siehe Listing 2).

Der Routine wird das Webbrowser-Objekt **oBrowser** als Parameter übergeben. Eine Schleife, die lediglich die Anweisung **DoEvents** enthält, wird solange wiederholt, bis der Ladezustand (**ReadyState**) des Browsers mindestens den Modus **LOADED** erreicht hat. Dann nämlich erst können die Objekteigenschaften des geladenen Dokuments angesprochen werden. Diese Schleife ist in eine externe Routine ausgelagert, weil sie unter Umständen öfters im VBA-Projekt der Datenbank angesprochen wird.

Listing 3 demonstriert den Code im Ereignis **Beim Anzeigen (Form\_Current)** des Formulars. Das nun geladene Dokument wird einer Objektvariablen **oDoc** zugewiesen, die ebenfalls im Kopf des Moduls deklariert ist:

```
Private WithEvents oDoc As HTMLDocument
```

Auch hier kommt die Anweisung **WithEvents** ins Spiel, denn nicht nur der Browser kann Ereignisse auslösen, sondern auch das Dokument selbst! Wählen Sie aus der linken oberen Combo des Code-Editors den Eintrag **oDoc** aus und wählen Sie aus der rechten ein beliebiges Ereignis aus. Sofort legt VBA eine passende Ereignisprozedur an. So etwa auch für das Ereignis **oncontextmenu**, die ausgelöst wird, sobald im Dokument die rechte Maustaste bedient wird:

```
Private Function oDoc_oncontextmenu() As Boolean
    oDoc_oncontextmenu = True
End Sub
```

```
Sub WebDocReady(oBrowser As WebBrowser)
    Do
        DoEvents
    Loop Until oBrowser.ReadyState > READYSTATE_LOADED
End Sub
```

**Listing 2:** Hilfsroutine zum Warten auf den Ladezustand eines Dokuments im Webbrowser Control

```
Private Sub Form_Current()
    WebDocReady oBrowser
    Set oDoc = oBrowser.Document

    oDoc.body.innerHTML = "<img id='bild' alt='fehlt!'" & _
        "label='Ein Bild' text-align: left></img>"
    DoEvents
    Set oPic = oDoc.getElementById("bild")

    If Me.NewRecord Then Exit Sub

    If Not IsNull(Me!Binaer) Then
        oPic.src = Me!Binaer.Value
        Me!LblBinaer.Visible = True
        Me!LblInfo.Visible = True
        Me!LblInfo.Caption = Format(Len(Me!Binaer.Value), _
            "###.###.###" & " Bytes")
    Else
        If Not IsNull(Me!Datei) Then
            If Len(Dir(Me!Datei.Value)) > 0 Then
                oPic.src = Me!Datei.Value
                LblBinaer.Visible = False
                Me!LblInfo.Visible = False
            End If
        End If
    End If
    Do
        DoEvents
    Loop Until oPic.ReadyState = "complete"
    wPic = oPic.Width: hPic = oPic.Height
    chkScale_AfterUpdate
End Sub
```

**Listing 3:** Vorgänge im Ereignis **Beim Anzeigen** des Formularmoduls

Stellen Sie den Rückgabewert der Ereignisfunktion auf **True** ein, so zeigt das Dokument das Kontextme-

nü des Internet Explorers an. Setzen Sie stattdessen **False** ein, so unterbleibt dies!

Sie können den Wert auch in Abhängigkeit eigener Berechnungen setzen. Das Kontextmenü soll in unserem Fall nur dann angezeigt werden, wenn die rechte Maustaste über dem Bild betätigt wird, nicht aber im restlichen Bereich des Dokuments. Dazu ist diese Ereignisfunktion etwas auszubauen, worauf wir später noch genauer eingehen.

Die Objektvariable **oDoc** ist als Typ **HTMLDocument** deklariert. Auch für diese Typklasse benötigen Sie einen zusätzlichen Verweis in der Datenbank, nämlich die **Microsoft HTML Object Library (MSHTML)**.

Dieses gewaltige Objektmodell enthält alles, was der Umgang mit einem Web-Dokument oder seinen Elementen erfordern kann. Sie ist deshalb auch die größte Bibliothek, die Sie überhaupt unter Windows finden werden. Zum Glück benötigen wir nur einen winzigen Teil aus ihr.

Die dritte Zeile in **Listing 3** zeigt gleich, wie auf das Objektmodell Bezug genommen wird. Das Dokument hat einen **Body**, der aus der Eigenschaft **body** von **oDoc** hervorgeht. Das **Body**-Objekt wiederum kennt zahllose Eigenschaften, von denen **innerHTML** eine ist. Sie bezieht sich auf den HTML-Code, der innerhalb der **body**-Tags steht. Und der kann hier ganz einfach modifiziert werden, was der Browser auch sofort in eine Änderung der Anzeige umsetzt, was sicherheitshalber noch mit einem eingeschobenen **DoEvents** vervollständigt wird, da dieser Vorgang ebenfalls etwas Zeit benötigt.

Zum besseren Verständnis: Das Navigieren zu **about:blank** hat diesen Dokument-Code zur Folge:

```
<HTML><BODY></BODY></HTML>
```

Und zwischen diese **Body**-Tags wird nun der Code für ein Bild eingefügt:

```
<img id='bild' alt='fehlt!' label='Ein Bild'></img>
```

Das Bild hat die Element-ID **bild** und die Beschriftung ist auf **Ein Bild** festgelegt. Über die **ID** kann später auf einfache Weise Bezug auf das Bild genommen werden:

```
Set oPic = oDoc.getElementById("bild")
```

Die Objektvariable **oPic** ist wieder einmal im Kopf des Moduls deklariert und spiegelt das Bild-Objekt wieder:

```
Private WithEvents oPic As HTMLImg
```

Die Typklasse **HTMLImg** finden Sie abermals in der Bibliothek **MSHTML**. Über die Variable **oPic** kann in der Folge auf sämtliche Eigenschaften des Bilds Einfluss genommen werden!

Bevor die Routine das tut, wird erst über **NewRecord** ermittelt, ob es sich um einen neuen Datensatz im Formular handelt. Dann nämlich ist weiter nichts zu tun und die Routine wird verlassen. Andernfalls blickt der Code auf das Datenfeld **Binear**. Ist es bereits mit Bilddaten gefüllt, so verzweigt die Routine entsprechend. Die Sichtbarkeit einiger Labels wird eingestellt und die Datengröße aus **Len(Me!Binear)** erhalten. Die Beschriftung des Labels zur Anzeige dieser Größe wird über die **Format**-Funktion verschönert. Sind noch keine Bilddaten angelegt, so setzt der Code den Inhalt des HTML-Bildobjekts **oPic** auf den Dateipfad:

```
oPic.src = Me!Datei.Value
```

Falls doch, so wird der gleichen Eigenschaft der komplette Inhalt des **Memo**-Felds zugewiesen:

```
oPic.src = Me!Binaer.Value
```

**src** entspricht in etwa der **URL** des Bildelements (**href**). Das Zuweisen dieses Eigenschaftswerts führt wieder zum Neuendern des Dokuments im Browser. Das kann asynchron durchaus einige Zeit beanspruchen, weshalb auch hier eine ähnliche Schleife eingebaut ist, die den Zustand des Bilds abfragt.



Gibt die Eigenschaft **ReadyState** des Bildobjekts den String **complete** zurück, so ist das Bild fertig dargestellt. Abschließend werden noch die Abmessungen des Bilds für die weitere Verwendung in den Variablen **wPic** und **hPic** (Breite, Höhe) zwischengespeichert und die Routine **chkScale\_AfterUpdate** aufgerufen, die auch beim Aktivieren oder Deaktivieren der Checkbox **Originalgröße (chkScale)** im Formular ausgelöst wird. Mehr braucht es für die Anzeige des Bilds im Webbrowser-Steuerelement zunächst nicht!

## Kontextmenü steuern

Der erweiterte Code für das Ereignis **oncontextmenu** der Dokumentvariablen **oDoc** steht in Listing 4. Das Ereignis wird grundsätzlich bei Rechtsklick auf das Dokument ausgelöst. Der Code ermittelt nun, auf welchen Koordinaten dieser Klick geschah. Dazu befragt es das Elternobjekt des Dokuments vom Typ **HTMLWindow**. Dieses Objekt kennt eine Objekt-Eigenschaft **event**, die alle Ereignisse des Browser-Fensters vereint. In **x** und **y** stehen darin die Koordinaten des Klicks. Die Funktion **elementFromPoint** gibt zurück, welches Element sich an diesen Koordinaten befindet. Handelt es sich um unser Bild, das ja die **ID bild** besitzt, so aktiviert sich das Kontextmenü, indem der Rückgabewert der Prozedur auf **True** gesetzt wird.

## Ereignisse des Bildobjekts

Die Variable **oPic** vom Typ **HTMLImg** wurde ja ebenfalls mit der Auszeichnung **WithEvents** deklariert. Auch das HTML-Bild kann Ereignisse auslösen! Wir verwenden zur Demonstration das **onclick**-Ereignis, das beim Anklicken des Bilds hervorgerufen wird:

```
Private Function oPic_onclick() As Boolean
    MsgBox oPic.src, vbInformation
End Function
```

Dabei muss hier nicht ermittelt werden, welche Maustaste bedient wurde, denn bei der rechten zeigt sich ohnehin das Kontextmenü. Die **Msgbox** gibt die Eigenschaft **src** des Bildobjekts aus, die zuvor in der Ereignisprozedur **Beim Anzeigen** des Formulars zugewiesen wurde. Neben diesem Ereignis kann das

```
Private Function oDoc_oncontextmenu() As Boolean
    Dim x As Long, y As Long
    x = oDoc.parentWindow.event.x
    y = oDoc.parentWindow.event.y
    If oDoc.elementFromPoint(x, y).ID = "bild" Then
        oDoc_oncontextmenu = True
    End If
End Function
```

**Listing 4:** Aktivieren des Kontextmenüs in Abhängigkeit vom angeklickten Element des Webdokuments

Bildobjekt noch etwa 50 weitere auslösen, wie ein Blick in den Objektkatalog verrät!

## Bild beeinflussen

Sie möchten das Bild im Browser verändern? Das geht verblüffend einfach über die **style**-Eigenschaft des Bildobjekts **oPic**. Sie können es etwa um 90 Grad nach rechts drehen, falls die Aufnahme dies erfordert. Dann setzen Sie lediglich dieses Statement ab:

```
oPic.Style.Transform = "rotate(90deg)"
```

Linksseitig geht es damit:

```
oPic.Style.Transform = "rotate(-90deg)"
```

Oder Sie spiegeln es horizontal:

```
oPic.Style.Transform = "scaleX(-1)"
```

Das vertikale Pendant:

```
oPic.Style.Transform = "scaleY(-1)"
```

Die Buttons im Fußbereich des Formulars lösen diese Anweisungen aus. Dabei wird allerdings zuvor jeweils diese Zeile abgesetzt:

```
oPic.Style.Transform = "initial"
```

Das bewirkt, dass das Bild sich wieder in den Ausgangszustand zurückbegibt. Dasselbe erreichen Sie auch mit folgendem Befehl:

```
oPic.Style.Transform = ""
```

Hier wird deutlich, dass sich diese Beeinflussung nur auf die Anzeige im Browser bezieht, nicht aber auf das Bild selbst. Speichern etwa Sie das rotierte Bild ab, so hat sich der Transform nicht auf die Datei ausgewirkt. Eine Bildbearbeitung ist auf diese Weise nicht so einfach zu erreichen, obwohl die style- und transform-Eigenschaften unerschöpfliche Möglichkeiten anbieten. Wichtiger, als die eben besprochenen Effekte ist sicher das Skalieren des Bilds, falls es größer ist, als das Webbrowser Control im Formular, wie in Bild 12.

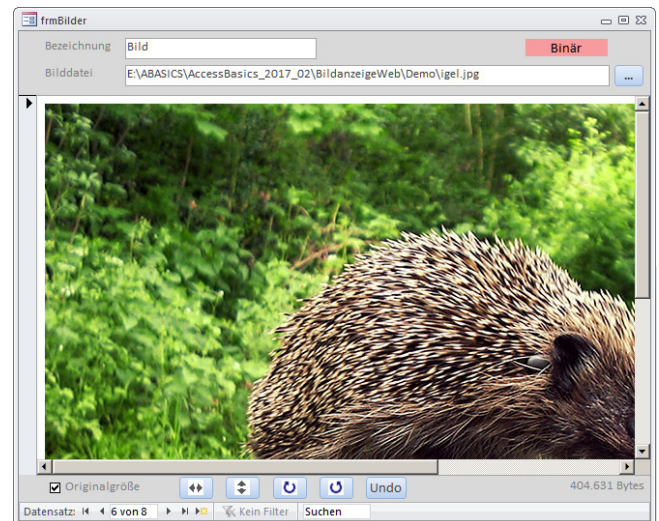
Deaktivieren Sie das Kontrollkästchen **Originalgröße** im Formularfuß, so passt sich das Bild, je nach Breite und Höhe, genau in den sichtbaren Bereich des Dokuments ein (siehe Bild 13).

Die Ereignisroutine **chkScale\_AfterUpdate** (siehe Listing 5) übernimmt diese Anpassung. In den modulweit deklarierten Variablen **wPic** und **hPic** speicherten wir bereits beim Laden des Bilds dessen Abmessungen. Hier werden sie nun verwendet, um aus dem Seitenverhältnis den Skalierungsfaktor zu errechnen. Die Breite und Höhe des Browser-Fensters (in **w** und

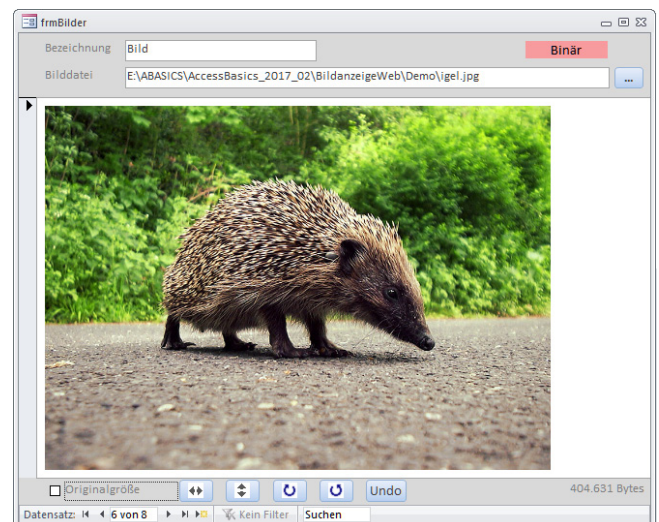
```
Private Sub chkScale_AfterUpdate()
    Dim w As Long, h As Long
    Dim factor As Double

    If chkScale = True Then
        oPic.Width = wPic: oPic.Height = hPic
    Else
        w = oBrowser.Width: h = oBrowser.Height
        If (w / h) > (wPic / hPic) Then
            factor = h / hPic
        Else
            factor = w / wPic
        End If
        oPic.Width = wPic * factor - 20
        oPic.Height = hPic * factor - 16
    End If
End Sub
```

**Listing 5:** Routine zum optionalen Skalieren des Bilds



**Bild 12:** Ist das geladene Bild größer, als das Webbrowser-Control, so entstehen ohne weiteres Zutun für das Dokument automatisch Scroll-Balken



**Bild 13:** Nach Deaktivieren des Kontrollkästchens skaliert sich das Bild passend im Web-Dokument

**h)** erhält man aus der Objektvariablen **oBrowser** und deren Eigenschaften **Width** und **Height**. Das ist übrigens ein Trick! Denn das Webbrowser-Steuerelement **ctlWeb** wurde mit dem **Verankern**-Feature versehen, weshalb es seine Größe dem Formular anpasst. Leider geben alle Steuerelemente dann in **Width** und **Height** nicht etwa die angepassten Abmessungen wieder, sondern die des Entwurfs. Sie können die **verankerte** Größe eines Controls in VBA nicht ohne API-Funktionen ermitteln! ActiveX-Steuerelemente, wie der Webbrowser, werden von Access automa-

tisch in die tatsächliche Größe des zugrundeliegenden Controls eingepasst. Also lesen wir einfach dessen Größe aus, was hier über die Objektvariable **oBrowser** geschieht. Zwar könnte man etwa auch über die tatsächliche Breite des Formulars (**Inside-Width**) abzüglich der Breite des Controls im Entwurf und einem Offset ebenfalls seine Breite bestimmen, aber unser Weg ist einfacher. Wie auch immer, die geänderten Abmessungen des Bilds im Browser können nun über die Eigenschaften **Width** und **Height** auf die Bildvariable **oPic** neu gesetzt werden.

**Fazit:** Sie benötigen weder ausführliche **HTML**-Kenntnisse, noch **Javascript**, um Elemente im Webbrowser-Control zu steuern. Praktisch alles lässt sich über die Objektklassen der **MSHTML**-Bibliothek erledigen. Übrigens reflektieren sich solche Aktionen auf die Elementobjekte dennoch im zugrundeliegenden **HTML**-Code. Rotieren Sie das Bild einfach einmal mit dem entsprechenden Button und lassen Sie sich danach den Quelltext des Dokuments ausgeben. Setzen Sie dazu etwa diese Zeile im VBA-Direktfenster ab:

```
? Forms!frmBilder!ctlWeb.Object.Document.all(0).outerHTML
```

#### Das Ergebnis:

```
<html><head></head><body><img width="267" height="277"
id="bild" style="transform: rotate(90deg);"
alt="fehlt!" src="data:
...</img></body></html>
```

#### Kodieren der Bilddaten

Bisher haben wir nur den Umgang mit einem existierenden Bild beschrieben, das entweder als Datei oder binär in den Daten des Memo-Felds vorliegt. Wie aber bekommen Sie diese Daten aus einer Datei in das Memo-Feld? Schauen Sie sich dazu den zweiten Teil der Ereignisprozedur beim Klick auf den Laden-Button an (siehe Listing 6).

Hier werden zwei Hilfsfunktionen angesprochen. Die eine nennt sich **FileContent** (siehe Listing 7) und speichert eine als String-Parameter übergebene Datei

```
Private Sub cmdLoad_Click()
    Dim sExt As String, sPrefix As String
    Dim bin() As Byte

    ...
    bin = FileContent(Me!txtDatei.Value)
    Select Case sExt
        Case "jpg", "jpeg": sPrefix = "data:image/jpeg;base64,"
        Case "png": sPrefix = "data:image/png;base64,"
        Case "bmp": sPrefix = "data:image/bitmap;base64,"
        Case "tif": sPrefix = "data:image/tiff;base64,"
    End Select
    oPic.src = sPrefix & EncodeBase64(bin)
    Me!Binaer.Value = oPic.src
End Sub
```

**Listing 6:** Beim Laden einer Bilddatei über den Button **cmdLoad** spielt sich diese Prozedur ab (zweiter Teil)

```
Function FileContent(sFile As String) As Byte()
    Dim F As Integer
    Dim bin() As Byte

    F = FreeFile
    Open sFile For Binary Access Read As F
    ReDim bin(LOF(F) - 1)
    Get F, , bin
    Close F
    FileContent = bin
End Function
```

**Listing 7:** Überführen eines Dateiinhalts in ein Byte-Array

in ein Byte-Array (**bin**). Sie verwendet dazu die **Open**-Methode von VBA. Danach wird über die Endung der Datei, welche in **sExt** abgespeichert wurde, in einem **Select-Case**-Block der **HTML-MIME**-Typ festgelegt und der Variablen **sPrefix** zugewiesen. Kommt ein Bild nicht aus einer Datei, sondern aus binären Daten, so hat der Inhalt der **src**-Eigenschaft nicht mit dem Präfix **file:** zu beginnen, sondern mit **data:**, gefolgt vom **MIME**-Typ. Dabei wird nur eine **base64**-Kodierung unterstützt, wie Sie das eventuell auch aus dem Quelltext von E-Mails mit Bildern kennen.

Aufgabe ist es nun, den Inhalt des Byte-Arrays **bin** in einen solchen **base64**-String zu verwandeln.

Hier kommt die zweite Hilfsroutine mit dem Namen **EncodeBase64** ins Spiel (siehe Listing 8). Die Konvertierung von Daten nach **base64** ist keine triviale Aufgabe und verlangt üblicherweise nach recht komplexem seitenlangen VBA-Code. Tatsächlich ist die Angelegenheit verblüffend einfach, wenn man sich Objekte der **XML**-Bibliothek von Microsoft zunutze macht. Dazu setzen Sie abermals einen Verweis auf die Bibliothek **Microsoft XML, v 6.0** (im Objektkatalog dann **MSXML2**), die Bestandteil von Windows ist.

Die Routine erzeugt zunächst ein neues XML-Dokument in der Objektvariablen **oXML** und legt dann ein nicht typisiertes XML-Element **oElem** mit dem Namen **tmp** in ihm an. Zwei Eigenschaften dieses Objektelements werden nun gesetzt: Der Datentyp (**bin.base64**) und der Inhalt über **NodeTypedValue**. Das war's auch schon! Über die Eigenschaft **Text** können Sie nun die Textinterpretation der Binärdaten als String auslesen. Dieser String wird in Listing 6 mit dem Präfix **sPrefix** zusammengesetzt und abschließend dem Bildobjekt als **Source (src)** zugewiesen. Der Browser rendert daraufhin sofort das Bild. Außerdem speichert die letzte Zeile noch den **src**-String im Formularfeld **Binaer**. Die Bilddaten befinden sich damit in der Tabelle und können beim nächsten Mal anstatt der Datei für die Bildanzeige benutzt werden.

Übrigens benötigen die Bilddaten in der **base64**-Kodierung nicht viel mehr Speicherplatz, als die Ursprungsdateien. Aus einem **JPG** mit 100 Kilobyte wird ungefähr ein String von 130 Kilobyte. Das ist akzeptabel und kann mit dem Speicherbedarf von Anlagefeldern mithalten. Im Modul **mdlBase64** der Beispieldatenbank ist noch eine Funktion **DecodeBase64** enthalten, die einen **base64**-String in ein Byte-Array zurückverwandeln kann. Sie verwendet ebenfalls **XML**-Objekte. In der Datenbank findet sie allerdings keine Verwendung, weshalb auf ein Listing hier verzichtet wird. Falls Sie dennoch Interesse an ihr haben, dann ist die Information wichtig, dass als Parameter nicht der komplette Inhalt des Binaer-Felds zu übergeben werden muss, sondern nur der Teil nach dem **MIME**-Präfix, also nach dem ersten Komma.

```
Function EncodeBase64(bin() As Byte) As String
    Dim oXML As New MSXML2.DOMDocument
    Dim oElem As MSXML2.IXMLDOMElement

    Set oXML = New MSXML2.DOMDocument
    Set oElem = oXML.createElement("tmp")
    With oElem
        .DataType = "bin.base64"
        .NodeTypedValue = bin
        EncodeBase64 = .Text
    End With
End Function
```

**Listing 8:** Hilfsfunktion zum Kodieren in base64 über XML

### Modifizierung des Webbrowser Controls

Damit die Bildanzeige im Webbrowser-Steuerelement ordnungsgemäß funktioniert, sollte es die Version **11** des Internet Explorers offenbaren. Dazu reicht leider nicht, dass dieser auch installiert ist. Microsoft versetzt das Webbrowser-Steuerelement unmodifiziert immer in den Modus der Version 7, egal, ob etwa auch **Edge** installiert ist. Um das aufzuheben, muss beim Start der Datenbank die Prozedur **PrepareBrowser** im Modul **mdlWebControl** aufgerufen werden, die einige Einstellungen für den gehosteten Internet Explorer vornimmt. In der Beispieldatenbank geschieht dies beim Öffnen des Intro-Formulars **frmIntro**.

Auf eine Beschreibung des Moduls können wir an dieser Stelle verzichten. Es kam bereits im Beitrag **Google- und Bing-Maps** im Formular (Ausgabe **03/2016**) zur Anwendung. Dort sind seine Funktionsweise und die Gründe für den Einsatz eingehend erläutert worden. Für den vorliegenden Zweck erweiterten wir es nur geringfügig.

### Zusammenfassung

Mit dem Webbrowser-Steuerelement haben Sie ein interessantes Instrument in der Hand, um Bilder in Formularen ohne Dateien direkt aus Tabellen anzuzeigen. Der dafür benötigte Code hält sich in Grenzen. Der Clou der Lösung besteht vor allem darin, dass Sie diese Web-Bilder über die **style**-Methoden auf einfache Weise in ihrer Gestalt steuern können.