

Kapitel VII

Rechnung erstellen mit Formularfeldern

In diesem Kapitel:

Formular zur Bestätigung von Bestellungen

2

Zusammenfassung

10

Das Word-Objektmodell haben Sie bereits in Kapitel 5, ADO in Kapitel 11 und UserForms in Kapitel 15 kennen gelernt. Im vorliegenden Kapitel finden Sie nun eine Lösung, in der die folgenden Themen veranschaulicht werden, wobei das Hauptgewicht auf dem Einsatz von Formularfeldern liegt.

Im Blickpunkt:

ADO-Verbindung zur Access-Datenbank

Formular

Formularfeld Optionen

FormField.Name-Eigenschaft

schützen

Table

Rows.Insert

Rows.Delete

Zelleninhalt mit Range.FormattedText kopieren

UserForm

Instanz wiederverwenden

ListBox-Steuerelement

Tag

Eine logische Verwendung für Formularfelder ist die Erstellung von Offerten, Lieferscheinen, Bestellungsbestätigungen oder Rechnungen. Produkte können aus Dropdown-Feldern gewählt, Preise automatisch eingetragen und Berechnungen automatisch durchgeführt werden. Eine tolle Sache ... wäre nicht eine variable Anzahl von Posten zu berücksichtigen. Da ist eine automatisierte Lösung unabdingbar, um für jeden neuen Posten einen Satz von Formularfeldern einzufügen oder auch überzählige zu entfernen.

Formular zur Bestätigung von Bestellungen

In Abbildung VII.1 ist eine solche Bestellungsbestätigung mit Formularfeldern (hier grau hinterlegt) dargestellt. Bei Eintritt in ein Formularfeld der Spalte »Beschreibung« wird die UserForm mit einer Produktliste eingeblendet. Nach dem Betätigen der Schaltfläche *Übernehmen* wird der ausgewählte Listeneintrag in das aktuelle Feld eingefügt. Zudem werden Produkt-ID sowie Einzelpreis in die entsprechenden Formularfelder der gleichen Tabellenzeile übertragen. Anschließend wird das Formularfeld für die Anzahl der Artikel aktiviert, so dass der Anwender sofort mit der Eingabe fortfahren kann. Den Code für diese Handlungen finden Sie in Listing VII.1.

HINWEIS Die graue Schattierung der Formularfelder kann mit der Schaltfläche *Formularfeld-Schattierung* der *Formular-Symbolleiste* ausgeblendet werden. Die entsprechende Eigenschaft im Objektmodell ist `FieldShading`:

```
doc.ActiveWindow.View.FieldShading = False
```

Die Schaltflächen der Symbolleiste *Rechnung schreiben* dienen dazu, neue Zeilen in die Tabelle einzufügen bzw. die aktuelle Zeile zu löschen. Der zugehörige Code steht in Listing VII.2 bereit.

Abbildg. VII.1 Eine auf Formularfeldern basierende Bestellungsbestätigung mit Hilfe von Makros schreiben

Herrn·Georg·Müller
Hauptstrasse 1
80000·München

Betreff: Ihre Bestellung

Sehr geehrte Herrn Müller,

besten Dank für Ihre Bestellung vom 1. Dezember, 2005.

Prod.-ID	Beschreibung	Anzahl	Stückpreis	Preis
60	Camembert Pierrot	1	17,00	17,00-€
20	Sir Rodney's Marmalade	10	40,50	405,00-€
18	Camarvon Tigers	2	31,25	62,50-€
	Total			484,50-€

Die Bestellung wird bis zum 20. Dezember, 2005, an obige Adresse geliefert.

Unterschrift

Produkt auswählen

- Camembert Pierrot
- Camarvon Tigers
- Chai
- Chang
- Chartreuse verte
- Chef Anton's Cajun Seasoning
- Chef Anton's Gumbo Mix
- Chocolade
- Côte de Blaye

Übernehmen Abbrechen

Rechnung schreiben
Zeile hinzufügen | Aktuelle Zeile löschen

Etwas konkreter ist der Aufbau der Rechnungsdokumentvorlage in Abbildung VII.2 veranschaulicht. Für die Anschrift des Empfängers sind Textformularfelder mit einem beschreibenden Standardtext wie »Anrede« oder »Ort« vorgesehen, der später durch die Benutzereingabe ersetzt wird.

Die Grußformel enthält *If*- sowie *Ref*-Feldfunktionen, die sich auf die Adressfelder »Anrede« und »Nachname« beziehen. Entspricht der Inhalt des Formularfeldes `txtAnrede` der Zeichenfolge »Herr«, wird das Wort »geehrt« um den Buchstaben »r« ergänzt. Die *Ref*-Feldfunktionen geben den Inhalt der Formularfelder `txtAnrede` und `txtNachname` unverändert wieder. Um diese Angaben dynamisch anzuzeigen, wurde für die beiden Formularfelder das Kontrollkästchen *Beim Verlassen berechnen* im Dialogfeld *Formularfeld-Optionen* aktiviert.

HINWEIS Mehr zum Thema Feldfunktionen steht in Kapitel 7 beschrieben.

Abbildg. VII.2 Die Dokumentvorlage für das Rechnungsformular

«Anrede» «Vorname» «Nachname»
«Straße»
«PLZ» «Ort»
Betreff: «Betreff»
Sehr geehrte(r) Herr/Frau/Mr./Mrs.
besten Dank für Ihre Bestellung vom

Prod.-ID	Beschreibung	Anzahl	Stückpreis	Preis
				0,00 €
			Total	0,00 €

«Notizen»
Unterschrift

Dem Textformularfeld für das Bestelldatum, unterhalb der Grußformel, wurde der Typ »Datum« zugewiesen. Somit kann der Anwender nur ein gültiges Datum und keinen anderen Text eingeben.

In den *Formularfeld-Optionen* für die Spalten »Prod.-ID« und »Stückpreis« wurde das Kontrollkästchen *Eingabe zulassen* ausgeschaltet. Für die Inhalte dieser Spalten ist der Automatisierungscode verantwortlich, der Anwender darf sie nicht anwählen.

Die Textformularfelder der Spalte »Preis« sind vom Typ *Berechnung* mit dem Zahlenformat #.##0,00 €; (#.##0,00 €). Der Preis für jede Zeile wird mit der Formel =Product(Left), die Spaltensumme mit der Formel =Sum(Above) berechnet.

HINWEIS

Aggregatfunktionen wie Sum und Left sind die einzige Methode, Berechnungen in Word-Tabellen relativ durchzuführen. Die Inhalte aller Zellen, bis zur ersten ohne numerischen Wert, werden in die Funktion miteinbezogen.

Die Prozedur *ProduktListeEinblenden* (Listing VII.1) wurde aus der Dropdownliste *Ereignis* in den *Formularfeld-Optionen* für die Formularfelder der Spalte »Beschreibung« gewählt. Die UserForm in Abbildung VII.1 wird eingeblendet. Während deren Initialisierung wird eine ADO-Verbindung (Connection) zur *Nordwind*-Datenbank hergestellt (die Prozedur *AccessDatenHolen*), um die Artikel-liste in einen ADO-Datensatz (Recordset) zu lesen. Dieser wird der Prozedur *ListeFuellen* übergeben, wo in einer Schleife der Feldinhalt in die Listenspalten geschrieben wird (wovon nur die erste Spalte angezeigt wird).

TIPP

Um eine Spalte in einem Listenfeld oder in einer Dropdownliste zu unterdrücken, setzen Sie die Spaltenbreite auf 0. Die ColumnWidths-Eigenschaft akzeptiert eine mit dem System-Argument-Trennzeichen getrennte Liste von Werten in Points.

Das Betätigen einer der Schaltflächen schreibt einen Wert in die Tag-Eigenschaft der UserForm und blendet diese dann aus; sie wird *nicht* aus dem Speicher entfernt. Der Vorteil dieser Vorgangsweise

ist, dass eine Verbindung zur Datenbank nicht bei jedem Aufruf der UserForm erfolgt. Zudem bleibt die Auswahl in der Liste bestehen, so dass der Anwender nicht immer wieder von vorne in der Liste beginnen muss.

In der Prozedur *ProduktListeEinblenden* ist ersichtlich, wie getestet wird, ob die UserForm schon geladen ist und wie diese Instanz weiter verwendet wird. Sonst wird eine neue Instanz aufgerufen.

Am Schluss dieser Prozedur wird der Wert der Tag-Eigenschaft geprüft und – falls »1« – die Prozedur *DatenInFormfelderSchreiben* aufgerufen. Sie überträgt die Artikelangaben aus der Liste in die entsprechenden Formularfelder und springt das Formularfeld in der Spalte »Anzahl« an, so dass der Anwender direkt weiterarbeiten kann.

Listing VII.1 Produktliste einblenden und die Angaben des gewählten Eintrags in die Formularfelder einfügen

```
'Als globale Variable bleibt die UserForm im Speicher.
'Dies vermindert den Zugriff auf die Datenbank, da die Liste
'weniger oft geladen werden muss.

Dim frm As frmProduktListe

Sub ProduktListeEinblenden()
    Dim lFormZaehler As Long
    Dim rw As Word.row

    'Tabellenzeile, wovon die Prozedur aufgerufen wurde, festhalten.
    Set rw = Selection.Rows(1)
    'Schleife nur ausführen, wenn mindestens eine Form vorhanden ist.
    For lFormZaehler = 1 To UserForms.Count
        'Aber der erste Indexwert ist immer 0, also Zähler minus 1.
        If UserForms(lFormZaehler - 1).Name = "frmProduktListe" Then
            'Die Form weiterhin benutzen, wenn sie nur verborgen ist,
            Set frm = UserForms(lFormZaehler - 1)
            Exit For
        End If
    Next
    'sonst, eine neue Instanz aufrufen.
    If frm Is Nothing Then
        Set frm = New frmProduktListe
    End If
    frm.Show
    Select Case frm.Tag
        Case Is = "0"
        Case Is = "1"
            DatenInFormfelderSchreiben frm, rw
        Case Else
    End Select
End Sub

Sub DatenInFormfelderSchreiben(frm As frmProduktListe, rw As Word.row)
    Dim strProduktName As String
    Dim strProduktNr As String
    Dim strProduktPreis As String
    Dim lAuswahl As Long

    lAuswahl = frm.lstProdukte.ListIndex
```

Listing VII.1 Produktliste einblenden und die Angaben des gewählten Eintrags in die Formularfelder einfügen (*Fortsetzung*)

```

'-1: kein Eintrag wurde ausgewählt; 0 = erster Eintrag der Liste
If lAuswahl >= 0 Then
    strProduktName = frm.lstProdukte.List(lAuswahl, 0)
    strProduktNr = frm.lstProdukte.List(lAuswahl, 1)
    strProduktPreis = frm.lstProdukte.List(lAuswahl, 2)

    rw.Cells(1).Range.FormFields(1).Result = strProduktNr
    rw.Cells(2).Range.FormFields(1).Result = strProduktName
    rw.Cells(4).Range.FormFields(1).Result = Format(strProduktPreis, "0.00")
    rw.Cells(3).Range.FormFields(1).Select
End If
End Sub

Sub AccessDatenHolen(ByRef RS As ADODB.Recordset, strSQL As String)
    Dim conn As ADODB.Connection

    Set conn = New ADODB.Connection
    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=C:\WordBuch\CD-ROM\Datenbank\nordwind.mdb;"
    Set RS.ActiveConnection = conn
    RS.CursorLocation = adUseClient
    RS.CursorType = adOpenStatic
    RS.LockType = adLockOptimistic
    RS.Open Source:=strSQL
    'Da Daten nur gelesen und nicht geschrieben werden, können wir
    'die Verbindung kappen und Ressourcen sparen.
    Set RS.ActiveConnection = Nothing

    conn.Close
    Set conn = Nothing
    'Debug.Print RS.Fields.Count, RS.RecordCount
End Sub

Sub ListeFuellen(RS As ADODB.Recordset, lst As msforms.ListBox)
    Dim lDS_Counter As Long

    lDS_Counter = 0
    'Artikel aus der Datenbank in das Listfeld füllen.
    Do While Not RS.EOF
        lst.AddItem RS.Fields("Artikelname").Value
        lst.List(lDS_Counter, 1) = RS.Fields("Artikel-Nr").Value
        lst.List(lDS_Counter, 2) = RS.Fields("Einzelpreis").Value
        lDS_Counter = lDS_Counter + 1
        RS.MoveNext
    Loop
End Sub

'Code hinter der UserForm
Private Sub btnAbbrechen_Click()
    Me.Hide
    Me.Tag = "0"
End Sub

Private Sub btnUebernehmen_Click()

```

Listing VII.1 Produktliste einblenden und die Angaben des gewählten Eintrags in die Formularfelder einfügen (Fortsetzung)

```

Me.Hide
Me.Tag = "1"
End Sub

Private Sub UserForm_Initialize()
    Dim RS As ADODB.Recordset
    Set RS = New ADODB.Recordset
    AccessDatenHolen RS, "SELECT Artikelname, [Artikel-Nr], Einzelpreis FROM Artikel" & _
        " ORDER BY Artikelname"
    ListeFuellen RS, 1stProdukte
    RS.Close
    Set RS = Nothing
End Sub

```

Die Rechnungsposten werden mit der Symbolleiste *Rechnung schreiben* verwaltet. Hinter den Symbolschaltflächen stehen die Prozeduren *TabellenZeileLöschen* sowie *TabellenZeileEinfuegen*. Die erstere ist relativ einfach. Zunächst wird sichergestellt, dass sich die Markierung in der Tabelle befindet. Dann wird die aktuelle Tabellenzeile sowie der Indexwert der letzten Tabellenzeile festgehalten. Handelt es sich bei der aktuellen Zeile weder um die erste noch die letzte, wird der Dokumentschutz aufgehoben, die aktuelle Zeile gelöscht, der Dokumentschutz wieder hergestellt und das Formularfeld mit dem Gesamtbetrag aktualisiert.

Das Hinzufügen einer neuen Zeile ist ein etwas komplexer Vorgang. Nachdem sichergestellt ist, dass sich die Einfügemarke in der Tabelle befindet, wird ermittelt, in welcher Zeile sie steht. Befindet sie sich in der ersten Zeile, wird die neue direkt darunter, an zweiter Stelle, eingefügt. Befindet sie sich in der letzten Zeile, wird die neue unmittelbar davor eingefügt. Sonst wird die neue Zeile immer unter der aktuellen eingefügt.

HINWEIS

Diese Nachprüfung wurde der Vollständigkeit halber zwar im Beispiel eingebaut, aber in dieser Vorlage wird die Einfügemarke nie in der ersten oder letzten Zeile stehen, weil sich darin keine ungesperrten Formularfelder befinden.

Der Dokumentschutz wird aufgehoben, dann wird das Einfügen der neuen Zeile von der Prozedur *NeueZeileEinfuegen* vorgenommen. Sie sorgt auch dafür, dass der Inhalt der noch aktuellen Zeile in die neue übernommen wird. Dazu wird die *Range.FormattedText*-Eigenschaft auf jede Zelle der Zeile angewendet. Bitte beachten Sie, wie der Bereich zuerst um ein Zeichen verkleinert wird, so dass die Ende-der-Zelle-Marke nicht mit einbezogen wird.

Um sicherzustellen, dass jedes Formularfeld einen eindeutigen Namen hat, werden mit der Prozedur *NameHolen* alle am Ende der aktuellen Namen stehenden Ziffern entfernt. Diesem »Ur-Namen« wird der Inhalt einer eigens für diesen Zweck erstellten Dokumentvariablen hinzugefügt – eine laufende Zahl, die zu Beginn der Prozedur um eins erhöht wurde.

Um dem Formularfeld den Namen zuweisen zu können, muss es markiert werden. Es ist nicht möglich, einem Formularfeld über die *Name*-Eigenschaft einen Namen zu geben. Dies muss über das Dialogfeld *Formularfeld-Optionen* geschehen, das sich nur auf die gegenwärtige Markierung auswirkt.

Zum Schluss wird der Dokumentschutz wieder aktiviert, das neue Formularfeld in der zweiten Spalte (»Beschreibung«) angesprungen und die UserForm eingeblendet, so dass der Anwender sofort weiterarbeiten kann.

Listing VII.2 Tabellenzeilen für die Rechnungsposten löschen und hinzufügen

```

Sub TabellenZeileLöschen()
    Dim doc As Word.Document
    Dim rng As Word.Range
    Dim tbl As Word.Table
    Dim rw As Word.row
    Dim rwLetzte As Word.row

    Set doc = ActiveDocument
    Set rng = Selection.Range
    Set tbl = AktuelleTabelle(rng)
    If Not tbl Is Nothing Then
        Set rw = rng.Rows(1)
        Set rwLetzte = tbl.Rows.Last
        'Die zu löschende Zeile darf weder die erste noch die letzte sein
        If (rw.Index <> 1) And (rw.Index <> rwLetzte.Index) Then
            DokschutzAufheben doc
            rw.Delete
            Dokschuetzen doc
            doc.Bookmarks("txtTotal").Range.Fields.Update
        End If
    End If
End Sub

Sub TabellenZeileEinfuegen()
    Dim doc As Word.Document
    Dim rng As Word.Range
    Dim tbl As Word.Table
    Dim rw As Word.row
    Dim rwNeu As Word.row
    Dim lLetzteZeileIndex As Long

    Set doc = ActiveDocument
    Set rng = Selection.Range
    Set tbl = AktuelleTabelle(rng)
    If Not tbl Is Nothing Then
        DokschutzAufheben doc
        lLetzteZeileIndex = tbl.Rows.Last.Index
        Set rw = rng.Rows(1)
        Select Case rw.Index
            Case 1
                'Falls die aktuelle Zeile die erste ist, wird
                'die neue Zeile nach dieser eingefügt.
                'Die Formularfelder kommen auch aus dieser Zeile.
                Set rwNeu = NeueZeileEinfuegen(tbl.Rows(2), tbl.Rows(2))
            Case lLetzteZeileIndex
                'Falls die aktuelle Zeile die letzte ist, wird
                'die neue Zeile vor dieser eingefügt.
                'Die Formularfelder kommen aus der vorherigen Zeile.
                Set rwNeu = NeueZeileEinfuegen(tbl.Rows(lLetzteZeileIndex - 1), tbl.Rows.Last)
            Case Else
                'Sonst wird die neue Zeile nach der aktuellen eingefügt
                Set rwNeu = NeueZeileEinfuegen(rw, tbl.Rows(rw.Index + 1))
        End Select
        Dokschuetzen doc
        rwNeu.Range.FormFields(2).Select
    End If
End Sub

```


Listing VII.2 Tabellenzeilen für die Rechnungsposten löschen und hinzufügen (Fortsetzung)

```

    ProduktListeEinblenden
End If
End Sub

Function NeueZeileEinfuegen(rwOrig As Word.row, rwPos As Word.row) As Word.row
    Dim tbl As Word.Table
    Dim rwNeu As Word.row
    Dim rngOrig As Word.Range
    Dim rngNeu As Word.Range
    Dim ffldOrig As Word.FormField
    Dim ffldNeu As Word.FormField
    Dim lZellenZaehler As Long
    Dim docVar As Word.Variable

    Set tbl = rwOrig.Range.Tables(1)
    Set docVar = tbl.Parent.Variables("FormularfeldSatz")
    'Den Zeilenzaehler um eins erhöhen.
    docVar.Value = CStr(Trim(Val(tbl.Parent.Variables("FormularfeldSatz")) + 1))
    'Die neue Zeile hinzufügen.
    Set rwNeu = tbl.Rows.Add(rwPos)
    'Durch die Zellen beider Zeilen schleifen.
    For lZellenZaehler = 1 To rwOrig.Cells.Count
        Set rngNeu = rwNeu.Cells(lZellenZaehler).Range
        Set rngOrig = rwOrig.Cells(lZellenZaehler).Range
        'Die Bereiche dürfen die Ende-der-Zelle-Marke nicht enthalten
        rngNeu.MoveEnd Unit:=wdCharacter, Count:=-1
        rngOrig.MoveEnd Unit:=wdCharacter, Count:=-1

        'Den Inhalt in die neue Zeile "kopieren"
        rngNeu.FormattedText = rngOrig.FormattedText

        Set ffldOrig = rngOrig.FormFields(1)
        Set ffldNeu = rngNeu.FormFields(1)
        'Bei der Benennung muss das Formularfeld markiert sein ...
        ffldNeu.Select
        '... weil der Name nur über das Dialogfeld zugewiesen werden kann.
        With Dialogs(wdDialogFormFieldOptions)
            .Name = NameHolen(ffldOrig.Name) & docVar.Value
            .Execute
        End With
    Next
    Set NeueZeileEinfuegen = rwNeu
End Function

'Alle Ziffern vom Ende der Zeichenkette entfernen und das Resultat zurückgeben.
Function NameHolen(ByVal strName As String) As String
    Dim lZaehler As Long

    For lZaehler = Len(strName) To 1 Step -1
        If Not IsNumeric(Mid(strName, lZaehler, 1)) Then
            strName = Mid(strName, 1, lZaehler)
            Exit For
        End If
    Next
    NameHolen = strName

```

Listing VII.2 Tabellenzeilen für die Rechnungsposten löschen und hinzufügen (*Fortsetzung*)

```
End Function

'Gibt ein Table-Objekt zurück, wenn die Markierung in einer Tabelle ist.
Function AktuelleTabelle(rng As Word.Range) As Word.Table
    Dim tbl As Word.Table

    If rng.Information(wdWithInTable) Then
        Set tbl = rng.Tables(1)
    End If
    Set AktuelleTabelle = tbl
End Function

Sub DokschutzAufheben(doc As Word.Document)
    If doc.ProtectionType <> wdNoProtection Then
        doc.Unprotect
    End If
End Sub

Sub Dokschuetzen(doc As Word.Document)
    doc.Protect Type:=wdAllowOnlyFormFields, NoReset:=True, Password:="", _
        UseIrm:=False, EnforceStyleLock:=False
End Sub
```



Die Beispieldatei *BspVII_01.dot* finden Sie auf der CD-ROM zum Buch im Ordner *\Beispiele\KapVII*.

Zusammenfassung

In diesem Lösungsvorschlag für eine Bestellungsbestätigungsvorlage wurden aus dem Word-Objektmodell folgende Elemente besonders hervorgehoben:

- Formularfelder und ihre programmtechnische Anpassung
- Tabellen

Zudem wurde Folgendes vorgestellt:

- Eine ADO-Verbindung zu einer Access-Datenbank, um Informationen zu holen
- Eine UserForm mit Listen-Steuerelement