Access voor beginners hoofdstuk 22 (Werken met Recordsets deel 2)

Handleiding van Helpmij.nl

Auteur: OctaFish

Mei 2014

In het vorige hoofdstuk over *Werken met Recordsets* heb ik de verschillen uitgelegd tussen een ADO connectie en een DAO connectie. In dit hoofdstuk gaan we een formulier in zijn geheel aansturen met Recordsets. Daarbij zijn twee varianten te onderscheiden: een formulier dat gekoppeld is aan de recordbron, en een formulier dat alleen een connectie maakt met de tabel als er gegevens opgehaald of geschreven worden, of worden bijgewerkt. De laatste variant heeft de minste belasting voor de database, want er is alleen verbinding op het moment dat het echt nodig is. Met deze variant kun je dus met een relatief grote groep gebruikers in dezelfde database werken.

Formulier koppelen aan Recordset

In het eerste voorbeeld gaan we een formulier vullen met een ADO recordset. Dat betekent dat we een formulier maken, en dat naderhand vullen met de juiste gegevens. We gebruiken hiervoor een tabel, die ofwel in de database aanwezig is, ofwel via een koppeling in de database zit. In het laatste geval spreken we over een Frontend-Backend database, waarbij de tabel in de backend zit. De Frontend kun je dan bij een aantal gebruikers uitzetten, die allemaal in hun eigen versie van de database werken, maar wel met dezelfde datatabellen. Deze techniek is al eerder behandeld, dus die zal ik niet verder uitwerken in dit hoofdstuk.

In deze oefening gaan we elk veld in het formulier met VBA vullen, en het spreekt voor zich dat je op het formulier dus in ieder geval dezelfde velden nodig hebt als je straks vanuit de Recordset gaat vullen. Je kunt daarbij kiezen voor een simpele aanpak, waarbij je een leeg formulier maakt en daar tekstvelden op zet die je oplopend nummert (bijvoorbeeld txtVeld1, txtVeld2, txtVeld3 ... txtVeld12) of je koppelt (tijdelijk) de tabel die je straks gaat gebruiken aan het formulier, zodat je alle velden uit de Lijst met velden kunt halen.

Neutraal formulier

In de eerste variant kijk je in je tabel hoeveel velden je nodig hebt, en plaats je een even groot aantal tekstvelden op het formulier. Gebruik de lay-out van de tabel om te bepalen hoe je de tekstvelden gaat verdelen op het formulier en gebruik de eigenschap <Naam> om de tekstvelden netjes oplopend te nummeren. Standaard gebruikt Access weliswaar een logische opvolging in de naam, maar de tekstvaknummering loopt op met een factor 2, en dat is niet handig. Pas de naam dus aan volgens een eigen systeem, en zorg er voor dat de naamgeving consequent is, zoals hierboven.

De labels krijgen bij het aanmaken van het tekstvak een standaard tekst die overeenkomt met de oorspronkelijke naam van het veld, maar daar hoef je weinig tot niets aan te doen, want de Caption eigenschap kun je vullen vanuit de tabel, bijvoorbeeld door de Veldnaam als bijschrift te gebruiken.

Voordeel van deze werkwijze is, dat je het formulier simpel kunt kopiëren als je een extra formulier wilt maken. Je hoeft alleen maar de hoeveelheid velden aan te passen aan het nieuwe gebruiksdoel, en het formulier werkt gelijk.

Formulier op basis van bestaande tabel

In de tweede variant weet je precies welke velden je hebt, en waar je ze wilt plaatsen. Het formulier is hier gekoppeld aan de tabel, en je kunt de lay-out dus precies maken zoals je wilt. Als je klaar bent met de lay-out verwijder je de Recordbron van het formulier, en het formulier is in beginsel klaar voor gebruik. Nadeel van deze methode is wel dat je het formulier maar voor één bron kunt maken, en dat je wat werk hebt aan het handmatig loskoppelen van de Recordbron en de Besturingselementbron van de verschillende tekstvakken. Want alle objecten op het formulier moeten uiteindelijk niet-afhankelijk zijn.

In het voorbeeld zie je de lay-out van een formulier dat is gebaseerd op de tabel Contactpersonen. Daarbij zijn de velden uit de tabel gehaald, die dus eerst is gekoppeld aan het formulier. Daarna zijn alle objecten weer losgekoppeld.

Code voor het gebonden formulier

De tweede en belangrijkste stap is natuurlijk het koppelen van het formulier aan de gegevensbron. Daarvoor is de formuliereigenschap <Bij laden> prima geschikt want die wordt uitgevoerd als het formulier start. En de koppeling hoeft maar één keer uitgevoerd te worden, want als de koppeling is gelegd, dan werkt het formulier zoals je gewend bent. We gaan dus kijken naar de formuliereigenschap <Bij laden>.

Zoals gebruikelijk, breek ik de code weer op in bij behapbare brokken. De code begint met het declareren van een Connection object, en een Recordset. Daarna wordt de Recordset ingesteld.

```
Private Sub Form_Load()
Dim cnt As New ADODB.Connection
Dim rst As New ADODB.Recordset
Dim strSQL As String
    Set cnt = CurrentProject.Connection
    Set rst = New ADODB.Recordset
    strSQL = "SELECT * FROM tblContacts"
    With rst
        Set .ActiveConnection = cnt
        .CursorType = adOpenKeyset
        .LockType = adLockOptimistic
        .Open "tblContacts"
    End With
    If rst.BOF And rst.EOF Then
        MsgBox "Geen records te vinden...", vbOKOnly
    Else
        Set Me.Recordset = rst
    End If
```

Met Set cnt wordt de Connection ingesteld op het huidige project, en dat is dus de database met tabellen zoals die zijn vastgelegd in de tabellenstructuur van de geopende Acces database.

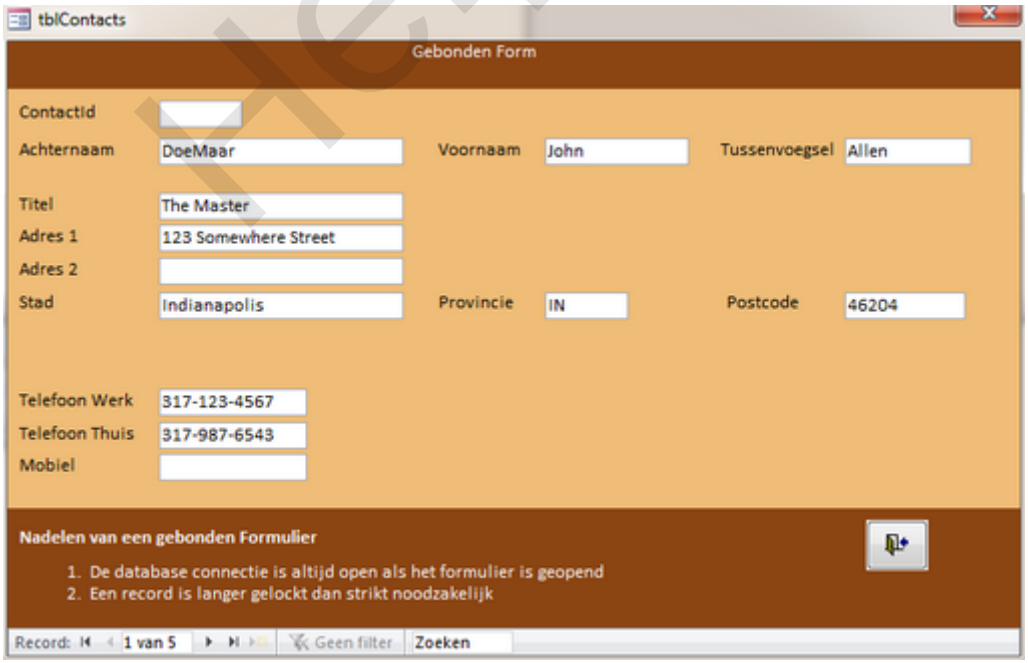
Vervolgens wordt het object rst gekoppeld aan de Connection, en de eigenschappen CursorType en LockType ingesteld. En uiteraard wordt de tabel geopend.

Om te voorkomen dat we een lege recordset aan het formulier hangen, checken we of er records in zitten. Dat doe je met de eigenschappen BOF en EOF, wat resp. staat voor *BeginOfFile* en *EndOfFile*. Als het begin tevens het eind is, dan zit er niks in de tabel/query, en kun je dus actie ondernemen, zoals in dit geval een Message Box. Je kunt het formulier dan ook bijvoorbeeld gelijk sluiten als volgende actie. Als er wél records in de Recordset aanwezig zijn, dan wordt de Recordset als Recordbron toegewezen aan het formulier met Set Me.Recordset = rst.

In het volgende blok worden de tekstvelden op het formulier gevuld. Dat wil zeggen: ze krijgen de *velden* uit de recordset als ControlSource, want we willen geen *waarden* invullen in de tekstvakken, maar het formulier kunnen gebruiken om door de records te bladeren. Daarom volstaat het om de velden te koppelen. En dat doen we met de eigenschap Name van de velden.

```
    With Me
        .txtLastName.ControlSource = rst.Fields(1).Name
        .txtFirstName.ControlSource = rst.Fields(2).Name
        .txtMiddleName.ControlSource = rst.Fields(3).Name
        .txtTitle.ControlSource = rst.Fields(4).Name
        .txtAddress1.ControlSource = rst.Fields(6).Name
        .txtAddress2.ControlSource = rst.Fields(7).Name
        .txtCity.ControlSource = rst.Fields(8).Name
        .txtState.ControlSource = rst.Fields(9).Name
        .txtZip.ControlSource = rst.Fields(10).Name
        .txtWorkPhone.ControlSource = rst.Fields(11).Name
        .txtHomePhone.ControlSource = rst.Fields(12).Name
        .txtCellPhone.ControlSource = rst.Fields(13).Name
    End With
End Sub
```

Als alle tekstvelden gematched zijn, kun je het formulier al bekijken, want het is nu klaar voor gebruik. Het formulier laat nu alle records zien, en het eerste record is gevuld in het formulier.



In het voorbeeld zitten er 5 records in de tabel, zoals je kunt zien in de Recordcounter onderin het navigatiepaneel.

Code voorbeeld 2

In de tekst hierboven gaf ik ook een andere techniek aan, waarbij je een aantal standaard tekstvakken op het formulier zet, en daarvan zowel bijschrift als tekstveld koppelt en vult. De code daarvan is zelfs nog wat simpeler qua structuur, omdat je de veldnamen en het aantal velden niet meer hoeft te weten. Wil je het echt mooi doen, dan lees je uit de tabel ook de bijschriften uit, zodat je die op het formulier kunt zetten.

De tabel is bijvoorbeeld zo ontworpen:

Veldnaam	Gegevenstype	Beschrijving
intContactId	AutoNummering	Contactnummer
txtLastName	Tekst	Achternaam
txtFirstName	Tekst	Voornaam
txtMiddleName	Tekst	Tussenvoegsel
txtTitle	Tekst	Titel
GebDat	Datum/tijd	Geboortedatum
txtAddress1	Tekst	Adres 1
txtAddress2	Tekst	Adres 2
txtCity	Tekst	Stad
txtState	Tekst	Provincie
txtZip	Tekst	Postcode
txtWorkPhone	Tekst	Telefoon Werk
txtHomePhone	Tekst	Telefoon Thuis
txtCellPhone	Tekst	Mobiel nummer

De veldnamen zijn op zich wel verklarend genoeg, maar op een formulier ziet dat er toch wat minder fraai uit. Het is mooier als je de Beschrijving of het Bijschrift kon gebruiken. Dat kan gelukkig, als we die eigenschap uitlezen met DAO. ADO is helaas niet in staat om die veldeigenschappen uit te lezen. Maar zoals ik al eerder heb uitgelegd, er zit niet zo heel veel verschil in het gebruik tussen ADO en DAO, en er is dus ook geen noemenswaardig bezwaar om beide technieken samen te gebruiken. Als we de bijschriften uitlezen, dan ziet het formulier er straks zo uit:

We beginnen de procedure weer op een vergelijkbare manier als in voorbeeld 1, want alle instellingen

worden weer uitgevoerd bij het laden van het formulier. Zoals ik al aangaf, kan de code een stuk korter omdat we niet meer afhankelijk zijn van de veldnamen en objectnamen.

```
Private Sub Form_Load()
Dim cnt As New ADODB.Connection
Dim rst As New ADODB.Recordset
Dim sTabel As String, sVelden As String
Dim arr As Variant
```

Er zijn wat variabelen bijgekomen, om de labels te kunnen vullen.

```
    sTabel = "tblContacts"
    Set cnt = CurrentProject.Connection
    Set rst = New ADODB.Recordset
    strSQL = "SELECT * FROM " & sTabel
    With rst
        Set .ActiveConnection = cnt
        .CursorType = adOpenKeyset
        .LockType = adLockOptimistic
        .Open "tblContacts"
    End With
```

De Connection code is identiek, want de procedure is hetzelfde gebleven.

```
    sVelden = TableInfo(sTabel)
    arr = Split(sVelden, "|")
```

Hier zit het verschil met de eerste procedure: we gebruiken een DAO functie die de informatie uit de tabel leest. Die functie heet hier TableInfo, en heeft (logisch, want het gaat om velden uit een tabel) de naam van de tabel nodig. De functie wordt zo behandeld; voor het moment mag je er van uitgaan dat alle veldbeschrijvingen in één string zijn gezet met het scheidingsteken '|', en die string wordt met de vaker gebruikte functie Split in een matrix gezet. Deze matrix bevat exact dezelfde indeling als de recordset, zodat we de velden straks kunnen koppelen aan hun beschrijving.

```
    If rst.BOF And rst.EOF Then
        MsgBox "Geen records te vinden...", vbOKOnly
    Else
        Set Me.Recordset = rst
    End If
```

Ook weer identiek aan de eerdere code. De echte verandering komt nu. In plaats van alle velden apart te vullen, gebruiken we een FOR .. NEXT om door alle velden uit de recordset te lopen, en die aan de matchende objecten te koppelen. Een Recordset collectie begint bij 0 te nummeren, en de tekstobjecten bij 1, en daarom gebruiken we i-1, en niet i. Maar zelfs dat kun je natuurlijk nog gelijktrekken als je dat wilt.

```
    For i = 1 To rst.Fields.Count - 1
        Me("lbl" & i).Caption = arr(i - 1)
        Me("txtVeld" & i).ControlSource = rst.Fields(i - 1).Name
    Next i
End Sub
```

De lus gebruikt voor de *objecten* dus de velden uit de tabel, en voor de *labels* de matrix. En dat levert dus een perfect formulier op.

De functie TableInfo

In de code heb je al gezien dat er een functie wordt aangeroepen die de veldbeschrijvingen uitleest uit de velden. Deze functie is gebaseerd op een DAO Recordset, omdat je met ADO deze eigenschappen niet kunt uitlezen.

Uiteraard beginnen we weer met het vastleggen van de (DAO) Recordset variabelen.

```
Function TableInfo(strTableName As String) As String
On Error GoTo TableInfoErr
Dim db As DAO.Database
Dim tdf As DAO.TableDef
Dim fld As DAO.Field
Dim tmpDesc As String
    Set db = CurrentDb()
    Set tdf = db.TableDefs(strTableName)
```

We hebben de tabel toegewezen aan de variabele *tdf*, en nu kunnen we de veldeigenschappen uitlezen in een lus. We maken hierbij één string aan, waarbij de velden worden gescheiden door een herkenbaar scheidingsteken. Dit teken gebruiken we vervolgens om de string weer te splitsen, zoals we al gezien hebben.

```
    For Each fld In tdf.Fields
        If Not tmpDesc = vbNullString Then tmpDesc = tmpDesc & "|"
        tmpDesc = tmpDesc & fld.Properties("Caption")
    Next
    TableInfo = tmpDesc
TableInfoExit:
   Set db = Nothing
   Exit Function
TableInfoErr:
   Resume TableInfoExit
End Function
```

Het uitlezen gebeurt met de eigenschap *Properties* van het veld. We kunnen hier zowel Caption als Description gebruiken, afhankelijk van wat er is ingevuld. Oplettende lezers zullen hebben gezien dat ik in de tabelafbeelding de labelteksten heb ingevuld bij de veldeigenschap *Beschrijving* (*Description)*, maar in de code gebruik ik de eigenschap *Notatie* (*Caption)*. Dat kan dus niet, maar in de voorbeeld database heb ik beide ingevuld (met dezelfde teksten) zodat ik zowel Description als Caption kan gebruiken.

Samenvatting

In dit hoofdstuk hebben we de eerste formulieren gemaakt die pas bij het openen aan een tabel worden gekoppeld. Hoewel deze methode al een lagere belasting vormt voor de backend (alleen koppelen als de tabel gebruikt wordt) is het nog niet de meest optimale situatie. Die maakt alleen een koppeling op het moment dat er informatie opgehaald en weggeschreven wordt. Die techniek vraagt veel meer van de programmeur, omdat je in die situatie elke ‘beweging’ in de tabel moet programmeren. Dat houdt dus in dat je code moet maken voor het bladeren, openen, toevoegen etc.

Daarom wordt die techniek in een volgend hoofdstuk behandeld.