Access voor Beginners Hoofdstuk 6

Handleiding van Helpmij.nl

Auteur: Octafish

Juli 2011

In dit hoofdstuk maken we een begin met programmeren, en gaan we de database uitbreiden met functies om er een soepel werkend geheel van te maken.

Deze cursus is overigens geen cursus programmeren; daar zijn vele (al dan niet) vuistdikke uitstekende boeken over geschreven. Bovendien is dat onderwerp dermate uitgebreid, dat de cursus tot ver in het volgende decennium zou moeten doorlopen. Wat we wèl gaan doen, is aan de hand van een aantal praktische voorbeelden uitleggen hoe VBA werkt, en hoe we zelf op relatief eenvoudige manier van VBA gebruik kunnen maken van programmeren om het programma te laten doen wat we voor ogen hebben.

Programmeren vs Macro's

In Microsoft Access kun je, net als in Microsoft Word en Microsoft Excel macro's maken, maar er is een groot (en wezenlijk) verschil: in de twee andere programma's kun je handelingen letterlijk *opnemen*. Je start derhalve in Excel een nieuwe macro, neemt een aantal handelingen op, zoals het selecteren van een cel, tekst of een formule typen, de cel opmaken en de volgende cel bewerken. Vervolgens sluit je de macro af, en kun je hem letterlijk opnieuw afspelen. De macro volgt dus een chronologische lijn. Hetzelfde principe werkt in Word. Daarna kun je met VBA de gemaakte macro's in het VBA scherm bewerken, door er bijvoorbeeld een herhalingsblok van te maken.

In Access daarentegen moet je een macro maken door zelf alle handelingen uit een lijst te selecteren, en de daarbij behorende opties in te stellen bij de Eigenschappen van die handeling. Er is geen enkele controle op de volgorde van de acties, dus je ziet het resultaat niet van de gekozen handelingen, zodat je niet weet of ze überhaupt wel werken. Ook is het geen enkel probleem als je acties selecteert die helemaal niet bij elkaar kunnen.

In Access 2007/2010 heeft Microsoft de macro tool aanzienlijk verbeterd, na jarenlang niets aan de ontwikkeling van de macro tool te hebben gedaan. Hierdoor kan het interessant zijn om dit onderdeel in deze versies te bekijken. Omdat ik van mening ben dat je desondanks de verbeteringen veel meer kunt doen met zelf programmeren, zal ik in de cursus verder geen aandacht besteden aan het maken van macro's.

Programmeren met de Wizard

Het zal veel mensen niet opvallen, maar als je in Access iets maakt met een wizard, dan is het resultaat vaak een stukje VBA code. Oftewel: je hebt feitelijk al iets geprogrammeerd als je een wizard gebruikt! En deze code kun je uiteraard bekijken, en als je dat wilt, aanpassen.

Als voorbeeld gaan we op een formulier knoppen maken voor de navigatie. Weliswaar kun je op een formulier een Navigatiebalk laten zien (een optie die standaard overigens aan staat), maar de knoppen zijn relatief klein, en derhalve niet overdreven gebruiksvriendelijk. Bovendien schakelt de knop <Vorig record> zich uit als je bij het eerste record bent (op zichzelf wel verklaarbaar), en ga je met de knop <Volgend record> naar een nieuw record als je bij het laatste record bent aangekomen. En soms wil je voorkomen dat een gebruiker per ongeluk een nieuw record aanmaakt. Door zelf knoppen te maken, kun je de knop <Vorig record> na het bereiken van het eerste record door laten lopen naar het laatste record, en de knop <Volgend record> de optie <Nieuw record> over laten slaan, en naar het eerste record laten springen. Zodat de gebruiker eindeloos door de records kan bladeren zonder tegen de

grenzen van de tabel aan te lopen. We gaan dus eerst de noodzakelijke knoppen maken met de wizard.

Knop in Access 2003

1. Open het formulier in de *Ontwerp modus*
2. Zet in het formulier de Voettekst weergave aan met **Beeld,Formulierkoptekst/Voettekst**
   Het is vaak het mooist om knoppen in de voettekst van een formulier te zetten.
3. Klik op de knop **Werkset** (), en klik vervolgens op de knop **Opdrachtknop** ()
   De cursor verandert in een kruiscursor + een vak ()
4. Klik ergens in de voettekst van het formulier

   De wizard *Opdrachtknop* wordt gestart.

5. Kies uit *Categorieën* **Recordnavigatie**, en uit *Acties* de optie **Naar eerste record gaan** en klik op de knop **Volgende**
6. Kies een afbeelding voor de knop, bijvoorbeeld *Naar eerste 2*, of typ een tekst die op de knop zal worden geplaatst door een omschrijving in het vak *Tekst* te typen en klik op **Volgende**
7. Typ een logische naam voor de knop, bijvoorbeeld: **cmdEersteRecord** en klik op **Voltooien**

Knop in Access 2007/2010

In Access 2007/2010 moet eerst een eigenschap van Access worden aangepast, anders maakt de wizard knoppen op basis van ingebouwde macro's. Hoewel deze knoppen prima werken, kunnen ze niet worden bewerkt, en dat is nu net de bedoeling van deze oefening.

1. Kies in het menu *Opties* de groep **Ontwerpfuncties voor Objecten**

2. Kies in de groep *Ontwerpweergave van Formulier/Rapport* de optie **Altijd gebeurtenisprocedures gebruiken**
3. Kies vervolgens de groep **Vertrouwenscentrum** en klik op de knop **Instellingen voor het vertrouwenscentrum**
4. Klik nu op **Instellingen voor macro's** en kies daar **Alle macro's inschakelen**
5. Klik op **OK**, gevolgd door **OK**

Opdracht

Maak met behulp van de wizard knoppen voor de functies *Eerste record*, *Volgend Record*, *Laatste record* uit de categorie *Recordnavigatie*, en *Nieuw record* en *Record verwijderen* uit de categorie *Recordbewerkingen*.

De knoppen uittesten

Nu de knoppen gemaakt zijn, kun je kijken of ze ook doen waarvoor we ze gemaakt hebben: navigeren door de records van het formulier.

1. Kies in het menu *Beeld* de optie **Formulierweergave**
2. Klik op de knoppen, en controleer de werking

Als je op het eerste record staat, en je klikt op de knop **Vorig record**, verschijnt, als het goed is, een melding:

Deze melding gaan we straks vervangen door een andere opdracht, waarmee de cursor wordt verplaatst naar het laatste record.

Knoppen aanpassen, en opdrachten toevoegen

Nu de knoppen gemaakt zijn, kunnen we de code bekijken, en uitbreiden.

1. Klik met de rechtermuis knop op een van de knoppen. Kies vervolgens uit het snelmenu de optie *Gebeurtenis opbouwen*
   Je komt in het VBA scherm terecht, waar je de code kunt bekijken.

cmdEersteRecord

```
Option Compare Database
Option Explicit

Private Sub cmdEersteRecord_Click()
On Error GoTo Err_cmdEersteRecord_Click


    DoCmd.GoToRecord , , acFirst

Exit_cmdEersteRecord_Click:
    Exit Sub

Err_cmdEersteRecord_Click:
    MsgBox Err.Description
    Resume Exit_cmdEersteRecord_Click

End Sub

Private Sub cmdVorigRecord_Click()
On Error GoTo Err_cmdVorigRecord_Click


    DoCmd.GoToRecord , , acPrevious

Exit_cmdVorigRecord_Click:
    Exit Sub

Err_cmdVorigRecord_Click:
    MsgBox Err.Description
    Resume Exit_cmdVorigRecord_Click

End Sub
```


In de afbeelding zie je de code voor de eerste twee knoppen die gemaakt zijn. Hierbij vallen een aantal zaken op, die we dan ook zullen uitleggen:

- Helemaal bovenaan in het voorbeeld staan twee regels. Deze regels zijn niet verplicht; de tweede regel (*Option Explicit*) bijvoorbeeld is afhankelijk van een instelling in Access. Onder de tweede regel zie je een grijze lijn. Deze wordt door Access zelf gemaakt, en hoeft dus niet te worden getypt.
  Regels die helemaal bovenaan staan, noemen we *Algemene Declaraties*. Wat deze regels betekenen, wordt later uitgelegd.
- De eerste regel onder de grijze lijn begint met de tekst *Private sub*. De laatste regel vóór de volgende grijze lijn begint met de tekst *End Sub*. Alles tussen deze twee regels is één *Routine*.
- Achter de woorden *Private Sub* zie je de naam van de knop waar de code bijhoort; de naam van de routine is dus de naam die je hebt ingetypt in de *Wizard Opdrachtknop*, gevolgd door de tekst *_Click()*. Het woord Click geeft in dit geval de Actie aan die wordt aangeroepen om de routine uit te voeren: het klikken op een knop dus.

Er zijn uiteraard meer zaken die je opgevallen zouden kunnen zijn, maar ook deze komen later aan bod. Eerst beginnen bij het begin!

Werken met Routines

Als je gaat programmeren, dan maak je in beginsel één van de twee volgende objecten: een *Routine*, of een *Functie*. Om je niet gelijk in het diepe te gooien, beperken we ons voorlopig even tot de *Routines*.

Je zou een Routine kunnen definiëren als een stukje VBA code waarmee je een handeling kunt uitvoeren. Deze handeling moet worden uitgevoerd nadat een gebruiker een handeling verricht. Dat kan een bewuste handeling zijn, zoals het klikken op een knop, of een muisverplaatsing, waarbij een routine wordt getriggerd als de gebruiker met de muis over een object beweegt. De gebruiker zal in het laatste geval niet altijd direct in de gaten hebben dat er een actie wordt uitgevoerd, dus met triggers op muisbewegingen moet je altijd een beetje oppassen!

Als je een routine hebt gemaakt, zoals met de wizard een aantal keren is gedaan, dan bevat die routine dus minstens twee regels:

Private Sub

End Sub

Daar tussenin staan de opdrachten die uitgevoerd worden. Laten we de code van de tweede knop eens van dichtbij bekijken:

```
Private Sub cmdVorigRecord_Click()
On Error GoTo Err_cmdVorigRecord_Click
     DoCmd.GoToRecord , , acPrevious

 Exit_cmdVorigRecord_Click:
    Exit Sub

 Err_cmdVorigRecord_Click:
    MsgBox Err.Description
    Resume Exit_cmdVorigRecord_Click
End Sub
```

In regel 1 en regel 12 zien we dus de begin- en eindcode staan van de procedure. Regel 2 begint met de tekst **On Error GoTo**. Deze regel bevat een opdracht die bepaalt wat er moet gebeuren als er een fout ontstaat tijdens de uitvoering van de code. Iets wat in dit voorbeeld optreedt als de gebruiker op het eerste record staat, en nogmaals op de knop klikt. De opdracht van de knop was immers: ga naar het vorige record. En dat wordt meestal wel gevonden, maar niet als je niet verder terug kunt. Dan is er een probleem: de code kan niet worden uitgevoerd, en er is dus sprake van een foutsituatie. In dat geval wordt regel 2 dus uitgevoerd. Ik kom daar straks nog op terug.

De belangrijkste regel, want de feitelijke opdracht van de knop, vind je op regel 4:
DoCmd.GoToRecord , , acPrevious

Deze regel bestaat uit 3 elementen: een **Commando** of **Object** (*DoCmd*), een **Methode** (*GoToRecord*) en een **Argument** (*acPrevious*). Gelukkig hoeven we de meeste begrippen en

termen niet uit het hoofd te leren, want Access heeft een Auto-Aanvul functie, die ons helpt bij het maken van opdrachten. Deze hulp bestaat uit twee verschillende methoden: ten eerste wordt er als we een commando typen een lijst met methodes getoond, en bij het selecteren van de argumenten verschijnt een balk met de juiste syntax, en/of een keuzelijst met argumenten.

We hebben dus gezien dat in regel 4 een opdracht staat, namelijk: voer het commando GoToRecord uit, en wel de variant: ga naar het vorige record.

We hebben ook bedacht dat deze opdracht meestal wel kan worden uitgevoerd, behalve als we op het eerste record staan. Een beetje vergelijkbaar met een rij stoeptegels die eindigt bij een ravijn: wat gebeurt er als je op de laatste tegel staat, en iemand je vraagt om nog één tegel op te schuiven. Het zou een goed moment kunnen zijn om een wedervraag te stellen, namelijk: kunnen we niet één tegel de andere kant op? In Access termen: kunnen we niet acNext doen i.p.v. acPrevious?

Om dit probleem op te vangen, beschikt de routine dus over een stukje code dat het probleem moet oplossen. We noemen dat in programmeertermen: een *Subroutine*. Dit is dus eigenlijk een routine *binnen* de hoofdroutine. En je zorgt er voor dat de code naar de subroutine springt door hem aan te roepen met het commando **On Error GoTo**.

Access is niet in staat om echt flexibele namen te maken voor routines, en subroutines. De naam van de routine is afgeleid van de naam van de knop, en de namen van de Subroutines zijn dat ook. In de opdracht zien we twee subroutines: een routine met de naam Exit_cmdVorigRecord_Click, en een routine met de naam Err_cmdVorigRecord_Click. De namen zijn, zoals je kunt zien, bijna identiek.

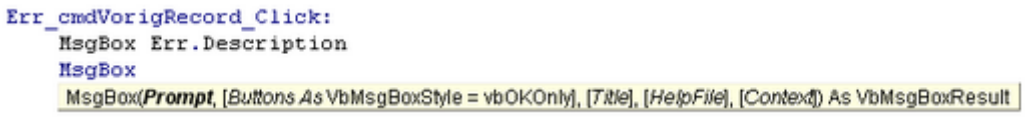
De opdracht GoTo verplaatst de uitvoering naar regel 9, de subroutine die Err_cmdVorigRecord_Click heet. De code wordt vanaf het begin uitgevoerd, hij begint uiteraard bij regel 1, en gaat daarna naar regel 2. Er wordt vervolgens een fout geconstateerd, en dus springt de routine naar regel 9, en voert vervolgens de opdracht uit in regel 10. Deze opdracht laat een dialoogvenster zien, met de foutmelding die is geconstateerd. Dat gebeurt met het commando **MsgBox**. Messageboxen zijn bijzonder geschikt om informatie door te geven aan de gebruiker. Op het moment dat er een messagebox verschijnt, kan de gebruiker namelijk niks anders doen dan de tekst op het scherm lezen, en op de knop **OK** klikken. De code gaat pas verder, als het dialoogvenster is afgesloten.

In het voorbeeld laat de Msgbox een melding zien die is gebaseerd op de methode **Err**. En dat staat niet helemaal onlogisch voor: Error. Deze methode heeft o.a. dus het argument *Description*, wat in de opdracht wordt gebruikt om de omschrijving van het probleem te geven. Iets als: u staat met de rug tegen de muur, en kunt niet verder! (maar dan anders).

Om te zien hoe Access nu werkt met objecten en commando's, maken we een nieuwe regel.

- Verwijder de regel *MsgBox Err.Description*
- Typ aan het begin van de regel: **msgbox**, en druk op [**spatie**]

  Je ziet dit:



Er verschijnt een regel met help aangaande het object MsgBox. Zo zie je dat de msgbox een *Prompt* vraagt. Hiermee wordt bedoeld: welke tekst wil je zien? Dat kan eigen tekst zijn, of een tekst die uit een Access commando wordt gehaald, wat we nu dus gaan doen.

- Typ nu: **Err** gevolgd door een punt (**.**)
  Er verschijnt een keuzelijst.

Deze keuzelijst bevat de methodes die horen bij de opdracht *Err*. Door het commando te typen, gevolgd door een punt, wordt de lijst automatisch geopend. Je kunt dus rustig de juiste methode uitzoeken die je wilt gebruiken. In het lijstje zitten twee keuzes die interessant zijn: *Description* en *Number*. Description hebben we al in actie gezien, want die staat in de regel er boven. Aangezien er honderden verschillende foutmeldingen zijn, kan het wel eens nuttig zijn om ook het fout*nummer* te weten; soms kun je op het internet a.d.h.v. dit nummer namelijk oplossingen vinden voor bepaalde problemen.

- Kies **Number** uit de lijst en druk op [**spatie**]

We willen behalve het nummer uiteraard ook de melding zelf zien. Daarvoor moeten we de twee commando's combineren. Eerst zorgen we voor een scheidingsteken die straks in de Msgbox zal verschijnen.

- Typ de volgende tekst letterlijk over: **& " – " &**

Met het Ampersand teken (&) combineer je verschillende opdrachten. In dit geval willen we een koppelteken (- streepje) hebben tussen het nummer en de omschrijving. Het tweede ampersand teken is uiteraard nodig omdat we nog een Err. opdracht nodig hebben.

- Typ nogmaals **Err** gevolgd door een punt, en kies nu uit de lijst: **Description**
  Het resultaat ziet er nu zo uit: MsgBox Err.Number & " - " & Err.Description
- Klik op de knop **Beeld Access** () om terug te keren naar Access, en toon het formulier in de Normale weergave, en probeer hem uit, door eerst naar het eerste record te gaan, en daarna op de knop **Vorige record** te klikken
  De Msgbox verschijnt met de melding *2015 – Kan niet naar de opgegeven record gaan*.

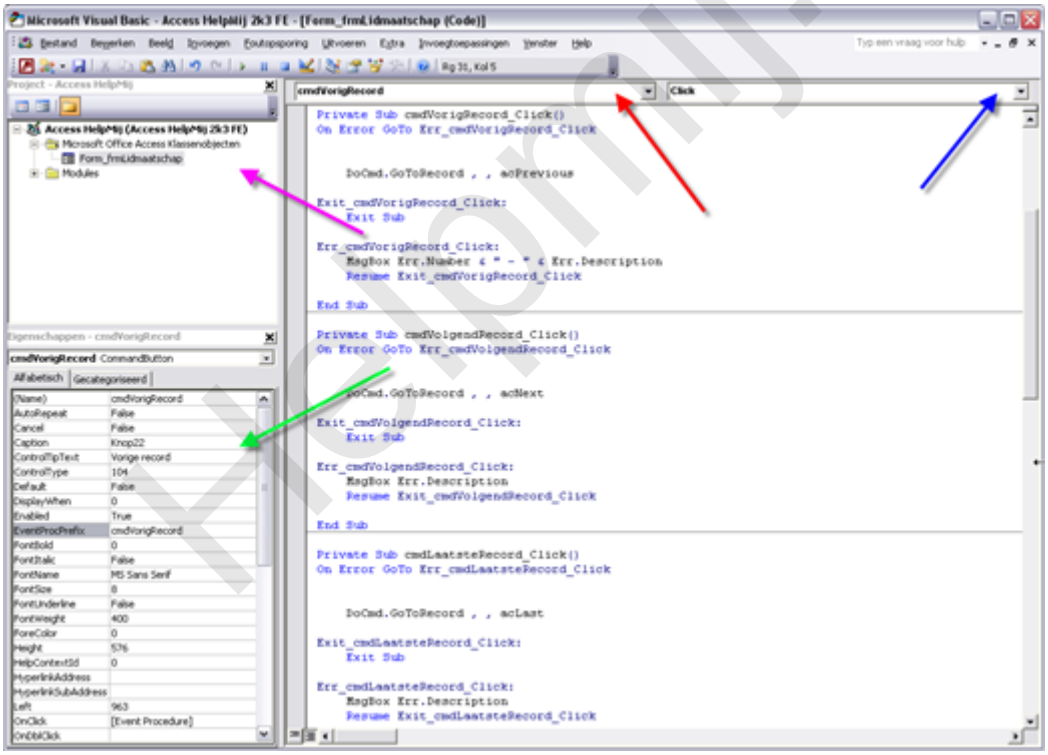
Zoals we gezien hebben, werkt de foutmelding zoals we hem bedoelen. De knop kan echter ook een andere functie krijgen als hij bij het eerste record wordt gebruikt: verplaats de cursor naar het laatste record. Zodat de gebruiker zonder gestoord te worden door een melding door de records kan blijven bladeren. Om dat te bewerkstelligen, gebruik je geen Msgbox, maar een andere opdracht. Welke dat is, hoef je niet meer zelf te ontdekken, want die opdracht hebben we bij een andere knop gebruikt. Je kunt die dus probleemloos kopiëren, al mag je hem uiteraard ook zelf maken op de manier zoals we ook de Msgbox hebben gemaakt.

Opdracht

Verander de twee knoppen Vorige Record en Volgend Record zo, dat ze bij een foutsituatie naar respectievelijk het laatste record gaan, en naar het eerste record.

Het VBA scherm verklaard

We hebben nu een paar VBA opdrachten gemaakt, maar nog weinig verteld over de omgeving waarin we met VBA werken. En die is toch wel belangrijk, omdat we hier veel mee (kunnen) doen. Dus eerst maar een schermafdruk van de werkomgeving:



Met een gekleurde pijlen heb ik een aantal belangrijke schermonderdelen aangegeven.

Om te beginnen de paarse pijl. Deze wijst naar het *Objectenoverzicht*. Standaard maakt Access voor elk formulier en elk rapport dat je maakt een eigen *Klasseobject* aan. Hierop vind je de VBA routines terug die je op dat formulier maakt. Code op een Klasseobject van het type Formulier of Rapport kun je alleen op dat object gebruiken. Daarom kent Access nog een paar andere objecten: *Modules* en *Klassemodules*. Deze staan, als je ze hebt aangemaakt, apart vermeld in het overzicht. In de cursus maken we voorlopig alleen gebruik van Formulieren en Modules.

De groene pijl wijst naar het venster *Eigenschappen* van het geselecteerde object op het formulier. In het plaatje staat de cursor in de routine cmdVorigRecord, dus de eigenschappen van die knop zijn zichtbaar in het Eigenschappenvenster.

De rode pijl wijst naar een keuzelijst waarmee je snel naar de verschillende objecten van het formulier kunt springen, zoals de eerder aangemaakte tekstvakken, opdrachtknoppen, of de formuliereigenschappen zelf. Zodra je een optie selecteert, verandert de ernaast gelegen keuzelijst in een lijst die alle gebeurtenissen van het geselecteerde object laat zien. Als je bijvoorbeeld een knop hebt geselecteerd, zijn de volgende gebeurtenissen beschikbaar:

Je ziet dat de optie Click vet is gedrukt; daaraan kun je zien dat deze optie in gebruik is. Interessante opties in de lijst zijn: *GotFocus*, *MouseDown* en *MouseUp*. Met GotFocus kun je een actie uitvoeren op het moment dat de knop geselecteerd wordt, wat we de Focus noemen. Met MouseDown en MouseUp kun je acties uit laten voeren als de muisknop resp. wordt ingedrukt, of weer wordt losgelaten.

Alles bij elkaar dus een heel scala aan gebeurtenissen, die je allemaal op eigen momenten kunt laten afvuren.

Ook interessant is het object *Form*. Hierin vind je allerlei acties die uitgevoerd kunnen worden op formulierniveau. Het plaatje laat overigens maar een klein deel van de lijst zien.

De Help functie

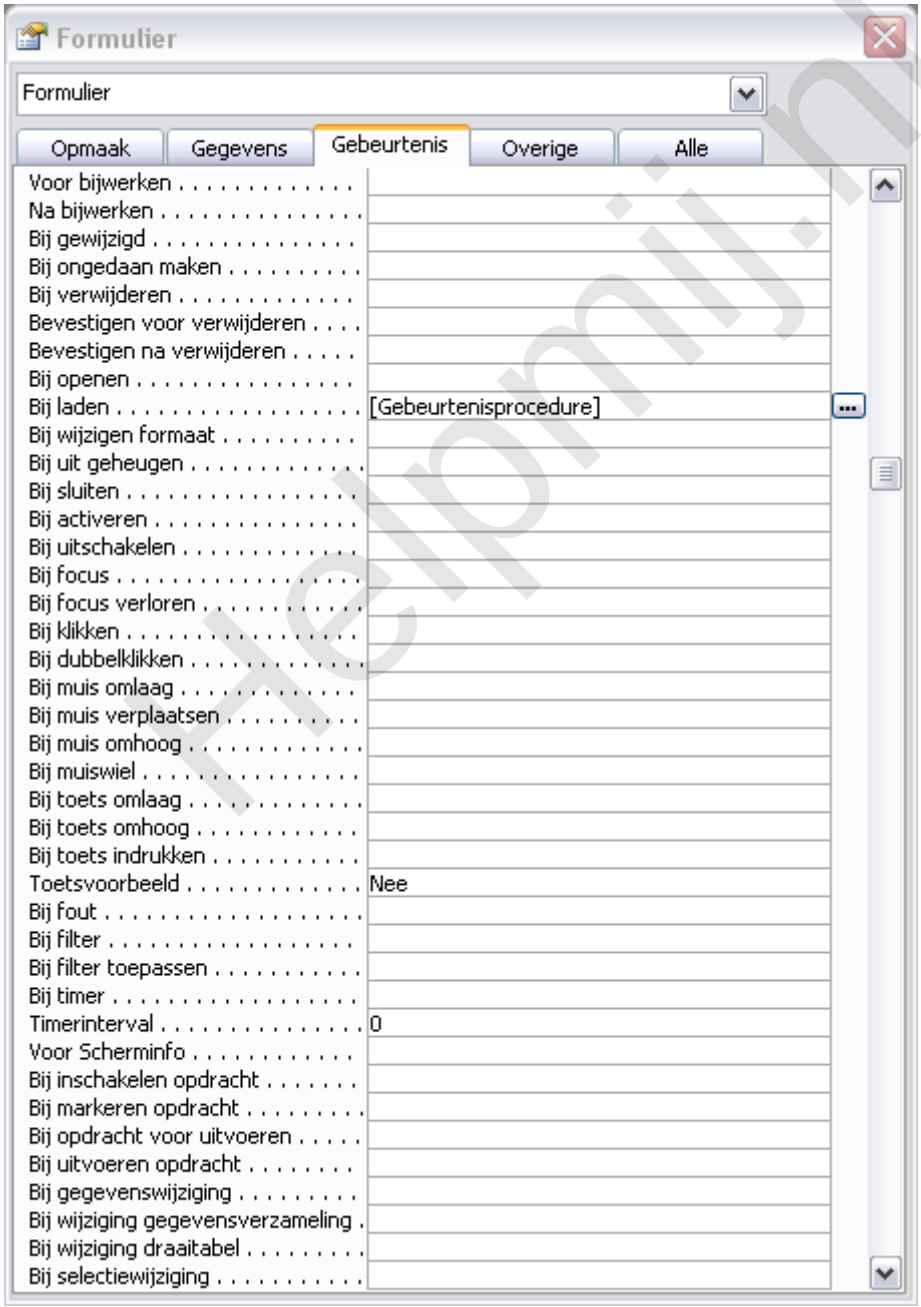
Access heeft een uitgebreide Help voor het programmeren. Als je bijvoorbeeld iets wilt weten over een bepaalde opdracht, dan zet je de tekstcursor in die opdracht, en druk je op **F1**. Je krijgt dan gelijk een helpscherm over die specifieke opdracht.

Als je de tekstcursor bijvoorbeeld in het woord **GoToRecord** zet, en je drukt op **F1**, dan is dit scherm het gevolg:

Je ziet een uitgebreide uitleg van de (in dit geval) methode, met een voorbeeld ter demonstratie. De voorbeelden in de Help maken meestal gebruik van de Amerikaanse versie van de voorbeeld database Noordenwind, die in het Engels *Northwind* heet. Het is absoluut handig als je die download van de Microsoft website, want dan kun je de voorbeelden bijna letterlijk overnemen als je ze eens wilt uitproberen.

Een gebeurtenis maken vanuit het Ontwerpscherm

Je kunt een gebeurtenis ook maken vanuit het formulier zelf, je hoeft dus niet eerst naar het VBA scherm te schakelen. Als je een formulier opent in de Ontwerpweergave, en je klikt op de knop **Eigenschappen**, zie je op het tabblad Gebeurtenissen dezelfde mogelijkheden als in de keuzelijst, alleen dan in de eigen taal, wat uiteraard wel zo prettig werkt.



In deze lijst zie je bij gebruikte opties de aanduiding [Gebeurtenisprocedure]. Daarachter zie je een knop met drie puntjes (). Als je deze aanklikt, kom je ook in het VBA scherm. Deze knop verschijnt overigens bij elke regel die je aanklikt, dus je kunt voor elke regel een gebeurtenis maken.

Samenvatting

We hebben een tipje van de sluier van het programmeren opgelicht. We hebben een paar knoppen gemaakt, en bekeken hoe de VBA code er uit ziet. We hebben gezien hoe Access ons helpt bij het maken van opdrachten door helpregels te laten zien bij opdrachten, of lijsten met beschikbare objecten. Ook hebben we bekeken hoe het VBA scherm er uit ziet, en waar we de onderdelen van het formulier kunnen terugzien en instellen.

In de Help functie van het VBA venster kun je uitgebreide informatie vinden over de verschillende opdrachten, methodes etc. Maak daar gebruik van!

Opdracht

Maak voor een aantal formulieren met behulp van de wizard meer knoppen, zoals een knop die een rapport afdrukt, en een knop die een ander formulier opent. Bekijk in het VBA venster vervolgens welke code door Access gemaakt is.

Volgende Aflevering

In de volgende aflevering gaan we dieper in op de werking van formulieren. We gaan een startscherm ontwerpen, waarin we verschillende formulieren met verschillende taken kunnen openen. Ook maken we een begin met Queries, die we als basis voor een formulier gaan gebruiken.