Access voor Beginners-Hoofdstuk 20

Handleiding van Helpmij.nl

Auteur: OctaFish

Januari 2014

Access Hoofdstuk 20

Bladwijzers maken in een formulier

Werken met een goed gebouwde database is doorgaans een redelijk prettige aangelegenheid. De bouwer heeft de juiste tabellen gemaakt en aan elkaar gekoppeld, er zijn formulieren gemaakt waarmee gegevens ingevoerd en opgezocht kunnen worden en er zijn rapporten en export procedures gemaakt waarmee de output gegenereerd kan worden. Kortom: er is nog maar weinig te wensen over. Maar toch zijn er altijd wel een aantal zaken te bedenken waarvan je denkt: waarom zit nou net dàt handigheidje er niet in? En over zo'n handigheidje gaat dit hoofdstuk.

Wat is dan zo'n functie waarvan ik denk dat die eigenlijk standaard ingebouwd zou moeten zijn? Om dat uit te leggen, maak ik eerst een zijsprong naar Word. Stel dat je daarin een groot verslag, verhaal, boek etc. aan het schrijven bent. Goede kans dat je dat stuk dan niet in één keer af krijgt, en er een volgende sessie aan zult moeten besteden. Of wel meer. Mij overkomt dat regelmatig. Als ik dan dat stuk de volgende keer open, dan staat dat keurig op het scherm, met de cursor helemaal vooraan uitnodigend te knipperen. Typ vooral verder, lijkt die cursor mij toe te schreeuwen. Probleem is natuurlijk, dat ik helemaal niet vooraan verder wil typen. Ik wil doorgaan waar ik gebleven was. En dat zal meestal niet helemaal vooraan zijn.

Ah, hoor ik de ervaren Word gebruiker al: <Ctrl>+<End> indrukken, en je bent waar je gebleven was. Da's echter niet altijd waar; ben je achter elkaar door aan het schrijven, dan ga je meestal achteraan verder, en dan is deze sneltoets prima. Maar jammer genoeg is mijn schrijfstijl wat erratischer, en zit ik regelmatig midden in het document te schrijven. Tja, ik bedenk me nogal eens over de eerder geschreven stukken, en die moeten dan natuurlijk gelijk worden aangepast. Dus als ik een document sluit, dan staat de cursor maar zelden aan het eind.

De Word expert (jammer genoeg ben ik dat vaak zelf) op kantoor roept nu: <Shift>+<F5>! En inderdaad, voor ik het in de gaten heb staat de cursor op de positie waar ik het laatst getypt heb. Ik helemaal blij natuurlijk! Want daar wil ik toch echt verder werken. Sterker nog: ik gebruik een standaard macro die het document bij openen gelijk naar de laatste positie brengt. Scheelt mij een hoop werk.

"Wat heeft dit nu met Access te maken?", hoor ik al. Heel simpel eigenlijk: Access heeft zo'n functie niet. En net als in Word open je een object (formulier, tabel) en staat de cursor aan het begin van het formulier te trappelen. Maar vaak is dat helemaal niet handig; meestal wil je records toevoegen (is nog wel te regelen) of de laatste records zien (kan ook geregeld worden met sorteren). Maar wat als het je werk is om de gegevens van een geïmporteerd bestand te controleren op hun juistheid? Als dat een heel groot bestand is, zal de kans groot zijn dat je het niet in één keer afmaakt. En de volgende keer dat je het formulier opent, mag je zelf opzoeken waar je ook al weer gebleven was.

In die situatie zou je willen dat je een 'boekenlegger' in je tabel kon leggen zodat je snel naar het punt kunt springen waar je gebleven was. Dit hoofdstuk behandelt twee oplossingen voor het vraagstuk (ik zal het geen probleem noemen) die je iets meer inzicht geven in het RecordsetClone object in combinatie met de Bookmark eigenschap.

De oplossing die we in variant 1 geven werkt prima, maar je moet een extra veld maken in je tabel. Dat zal niet altijd kunnen, denk maar aan een Frontend-Backend database waarbij je als gebruiker niet aan een tabel kunt knutselen. En als je de bookmark functie op meerdere formulieren wilt gebruiken, dan moet je meerdere tabellen aanpassen. Een ander nadeel is, dat je maar voor één gebruiker (namelijk: alle gebruikers) een bookmark kunt maken. Wil je optimale vrijheid, dan is variant 2 te overwegen. Daarbij slaan we voor alle gebruikers een eigen bookmark op, en lezen we dat bij het openen van het formulier weer uit.

Variant 1: het selectieveld

De eerste variant is eigenlijk een simpele: voeg een extra selectieveldje toe aan je tabel, en zorg ervoor dat het record waarin je aan het werken bent automatisch geselecteerd wordt. Ga je naar een volgend record, dan wordt het selectieveld weer leeg gemaakt, en het volgende veld gemarkeerd. Sluit je vervolgens het formulier af, dan hoef je bij het openen van het formulier alleen maar een procedure aan te roepen die het record met het selectievakje opzoekt, en actief maakt in het formulier.

Je bent dan gelijk op het punt waar je verder kan werken.

In het voorbeeld maak ik gebruik van een tabel met boekbestanden. Die tabel is gekoppeld aan een tabel Auteurs, en die tabel vormt de basis voor het hoofdformulier <fBoeken>. De tabel met boeken is dan de bron voor een subformulier, dat op basis van AuteurID is gekoppeld aan het hoofdformulier. De tabel [eBoeken] krijgt in dit voorbeeld het extra Ja/Nee veld, en bij het bladeren wordt het eerste boek van een auteur gemarkeerd.

Omdat we de cursorpositie continue moeten monitoren en kunnen opvragen in elke procedure, worden die variabelen in het algemene deel van het formulier gedeclareerd. We declareren ook een formulier object, omdat we regelmatig gegevens ophalen uit het subformulier. Door daarvoor een object te gebruiken, wordt de code overzichtelijker.

```
Option Compare Database
Dim PrevRec As Long, CurRec As Long
Dim subFrm As Form
Dim rst As Recordset
```

We beginnen met de code waarmee we het formulier initialiseren. Die staat in de procedure < Form_Load >. Deze code zoekt het eerste record op waarvan het selectievakje is aangevinkt. Daarin wordt eerst de objectvariabele subFrm ingesteld op het subformulier. Vervolgens wordt een SQL string opgebouwd op basis van de boekentabel met als criterium < Selected = TRUE>. Dit levert, als het goed is, één record op.

Opmerking: Het nadeel van een extra veld is natuurlijk dat je in de tabel zelf meerdere records kunt aanvinken, en op dat moment werkt deze query niet goed, want de procedure kijkt alleen naar het eerste record met een vinkje.

Vervolgens wordt de recordset geopend en de naam van de gezochte auteur in de variabele sAuteur gezet.

```
Private Sub Form_Load()
Dim sAuteur As String
    Set subFrm = Me!fBoekenDoorlopend.Form
     strSQL = "SELECT AuteurID, Selected " _
        & "FROM eBoeken WHERE Selected = TRUE"
         Set rst = CurrentDb.OpenRecordset(strSQL)
          With rst
           sAuteur = !AuteurID
          .Close
  End With
```

Nu wordt de recordset opnieuw ingesteld, en wel op een kopie van de tabel. We gebruiken hiervoor de formuliereigenschap RecordsetClone. Deze kopie is niet zichtbaar, en je gebruikt hem ook niet om gegevens te muteren want de recordset is alleen lezen, maar hij is prima geschikt om records in op te zoeken. De gevonden record kun je dan weergeven in je formulier, zoals we gaan doen. Zoeken doen we met <FindFirst> want we hoeven de naam maar één keer te vinden. Het zoeken levert uiteraard een treffer op, of niet. Dat laatste zou niet voor mogen komen, omdat we immers al gezocht hebben op AuteurID. Omdat een procedure altijd wel een keer fout kan gaan is het wel zo netjes om fouten af te vangen met een msgbox.
De belangrijkste actie wordt uitgevoerd als er wél een record wordt gevonden: <Me.Bookmark = rst.Bookmark>. hiermee wordt in het formulier zelf de Bookmark eigenschap ingesteld op het gevonden record in de recordsetclone.

Als je in de Stap modus door de code loopt, zul je zien dat het formulier na deze regel uit de <Form_Load> procedure springt, en in de <Form_Current> verder gaat. Dit is logisch, want deze procedure wordt altijd uitgevoerd zodra er een ander record actief wordt op het formulier. En daar is sprake van, want dat hebben we net met het instellen van de Bookmark gedaan. In de <Form_Current> procedure moeten we hier ook rekening mee houden, zoals je straks zal zien.

```
Set rst = Me.RecordsetClone
 With rst
  .FindFirst "AuteurID = " & sAuteur
    If rst.NoMatch Then
```

```
    MsgBox "Auteur niet gevonden."
Else
 Me.Bookmark = rst.Bookmark
End If
 .Close
End With
 PrevRec = subFrm!boekID
End Sub
```

Nadat de procedure weer terug is in de <Form_Load>, wordt de rest van de procedure uitgevoerd. Hierbij wordt de recordsetclone gesloten, en de variabele PrevRec gevuld met het eerste BoekID uit het subformulier.

De code die het echte werk doet, zit in de gebeurtenis <Bij aanwijzen>, zoals je wellicht al had vermoed. De procedure is bij het laden dus al een keer doorlopen, en daarom moeten we controleren of we wel op een correct record staan. Dat doen we door te controleren of de variabele PrevRec groter is dan nul. Zoals we hierboven hebben gezien, worden de variabelen algemeen gedeclareerd. Ze krijgen hierbij standaard de waarde nul. Bij het laden van het formulier krijgt de variabele PrevRec als laatste actie de waarde uit het BoekID uit het subformulier. De eerste keer dat de procedure < Form_Current> wordt doorlopen, is de variabele nog niet gevuld, en heeft dus de waarde nul. Dat is een mooi uitgangspunt voor de eerste check die we gaan gebruiken als we gaan bladeren door de records op het hoofdformulier.

Eerst nog even samenvatten wat er allemaal moet gebeuren als we gaan bladeren: we willen in het formulier het eerste record op het subformulier markeren d.m.v. een selectievakje. Dat houdt ook in, dat het selectievakje *in het vorige record* moet worden gedeselecteerd, want we willen maar één selectievakje gebruiken in de hele tabel.

```
Private Sub Form_Current()
Const strSQL As String = "SELECT BoekID, Selected " _
  & "FROM eBoeken WHERE BoekID = "
    Set subFrm = Me!fBoekenDoorlopend.Form
     CurRec = subFrm!boekID
      If PrevRec <> CurRec And PrevRec > 0 Then
```

We beginnen met het instellen van het formulierobject. Vervolgens wordt de waarde van het huidige eerste record in het subformulier in de variabele CurRec gezet. Daarna vergelijken we de waarde van CurRec met PrevRec. Zijn die verschillend, en dat zou moeten als we naar een andere auteur bladeren, *en* de waarde in PrevRec is groter dan nul (zoals hierboven uitgelegd, treedt die situatie pas op nadat het formulier volledig is geladen) dan weten we dus dat we een nieuw record moeten markeren, en het oude record moeten deselecteren. Beide acties doen we met een recordset, al zou je daar ook een Bijwerkquery voor kunnen gebruiken. Die keuze mag je uiteraard zelf maken.

Opmerking: Als je een query gebruikt, zorg er dan wel voor dat je de standaardmeldingen in de database uitzet. Dat kun je doen in de Opties van de database, maar dan weet je niet of de code ook goed werkt op een andere computer, of afvangen met DoCmd.SetWarnings. De laatste oplossing garandeert in ieder geval een juiste werking.

We maken eerst een query die het vorige record selecteert. Omdat we in de procedure een paar keer dezelfde query nodig hebben, is de basis als Constante gedefinieerd. Bij het openen van de recordset hoeven we alleen het recordID toe te voegen: OpenRecordset(strSQL & PrevRec). Daarna wordt de recordset in de Edit modus gezet zodat we het veld [Selected] de waarde FALSE kunnen geven.

```
Set rst = CurrentDb.OpenRecordset(strSQL & PrevRec)
  With rst
   .Edit
    !Selected = False
      .Update
        .Close
End With
```

In de volgende stap openen we de recordset met het huidige record. Hiervan staat het selectievakje uit, en dat moet nu aangezet worden. Als laatste in dit deel van de IF vergelijking wijzen we de huidige waarde van CurRec toe aan PrevRec zodat we na het bladeren de procedure weer kunnen uitvoeren

met de dan geldende records.

```
Set rst = CurrentDb.OpenRecordset(strSQL & CurRec)
  With rst
    .Edit
      !Selected = True
       .Update
         .Close
End With
  PrevRec = CurR
End If
```

Als laatste onderdeel wordt het veld Auteur geselecteerd op het hoofdformulier, en daarbij wordt de cursor aan het eind van het tekstveld gezet. Dit om de focus van het subformulier af te halen. Je kunt ook bijvoorbeeld een aantal bladerknoppen op het formulier zetten en één van die knoppen de focus geven.

```
With Me.Auteur
  .SetFocus
   .SelStart = .SelLength
End With
End Sub
```

Variant 2: De Historietabel

Zoals dat vaker gaat in het leven, zijn er meerdere wegen die naar (pak 'm beet) Rotterdam leiden. De tweede oplossing gebruikt daarom een aparte tabel waarin je de huidige gegevens opslaat en uiteraard weer uitleest. Hierdoor zijn we in staat om van elk formulier de locatie op te slaan voor elke gebruiker.

In deze werkwijze gebruiken we twee functies om de tabel [tWieWasWaar] te laden en uit te lezen. De tabel hoeft niet te bestaan, want die wordt bij het eerste gebruik aangemaakt. De tabel komt in de Frontend database te staan als een lokale tabel, dus zelfs bij het distribueren van de frontend naar verschillende eindgebruikers zitten die elkaar niet in de weg.

We bouwen de routine op dezelfde brontabel ([eBoeken] en hetzelfde formulier (<fBoeken>) maar nu hoef je het extra veld er niet in te zetten, want alle noodzakelijke data slaan we op in een aparte tabel. De methode moet universeel inzetbaar zijn, dus om per gebruiker per formulier bij te kunnen houden waar hij/zij is gebleven, moeten we een structuur vinden die altijd wordt uitgevoerd, ongeacht wat de gebruiker doet. Omdat we de gebruiker niet willen lastigvallen met dialoogvensters, plaatsen we twee procedures bij het openen en sluiten van het formulier.

De procedure die we gebruiken bij het sluiten kunnen we overigens zo maken dat we de gebruiker middels een dialoogvenster vragen of de bladwijzer moet worden opgeslagen of niet. In het laatste geval kan het record worden verwijderd. Wil je meer flexibiliteit voor de gebruiker, dan kun je zelfs nog een vraag inbouwen of de gebruiker de bladwijzervraag wil zien of niet. Daar heb je dan wel een extra veld nodig. Als de gebruiker de vraag niet wil, dan zet je bijvoorbeeld in de personentabel een vinkje aan die je checkt bij het opslaan. Er zijn uiteraard ook nog andere opties mogelijk. In dit voorbeeld nemen we alleen de optie mee om de bladwijzer te verwijderen.

Functies op het formulier

We gebruiken dus een extra tabel waarin we het log opslaan. De tabel [tWieWasWaar] ziet er zo uit:

tWieWasWaar

Veldnaam	Gegevenstype	Beschrijving
HistorieID	AutoNummering	
User	Tekst	Naam van de gebruiker
Bron	Tekst	Naam van het formulier
ObjectNaaam	Tekst	Naam van het opzoekveld
Bladwijzer	Numeriek	Waarde van het opzoekveld
DatumGewijzigd	Datum/tijd	Datum waarop veld is gewijzigd

Zoals ik hierboven al aangaf, wordt de tabel bij het eerste gebruik aangemaakt. Het aanmaken behandelen we verder bij de procedure [WieIsWaar]. De code die op het formulier komt te staan, is relatief simpel, omdat we aparte functies gebruiken. De enige actie die op het formulier zelf moet worden bijgehouden, is het vullen van de variabelen waarin het actieve record wordt opgeslagen. Dat doen we bij de gebeurtenis <Bij aanwijzen>. Bij het sluiten van het formulier zijn de recordgegevens namelijk al weg. De formuliercode ziet er derhalve zo uit:

```
Option Compare Database
Dim CurRec As Long, curAut As Long
Dim subFrm As Form
Dim bOpslaan As Variant
Const strSQL As String = "SELECT BoekID, Selected FROM eBoeken WHERE BoekID = "
```

We hebben uiteraard weer een aantal variabelen nodig die public werken binnen het formulier. Die zijn hierboven gedeclareerd.

```
Private Sub Form_Load()
   Call WieWasWaar(Me)
    CurRec = Me!fBoekenDoorlopend.Form!boekID
End Sub
```

Bij het laden van het formulier wordt de procedure <WieWasWaar> uitgevoerd. Deze procedure gebruikt één parameter: het actieve formulier. De functie wordt verderop uitgelegd. Onderstaande gebeurtenis wordt vervolgens uitgevoerd bij het bladeren door de records.

```
Private Sub Form_Current()
   Set subFrm = Me!fBoekenDoorlopend.Form
   CurRec = subFrm!boekID
   curAut = Me.AuteurID
   With Me.Auteur
      .SetFocus
      .SelStart = .SelLength
   End With
End Sub
```

We halen het gewenste boek weer uit een subformulier, dus voor het gemak maken we daar weer een objectvariabele voor. Het recordnummer halen we daarna op met de opdracht CurRec = subFrm!boekID. Het auteurID staat uiteraard op het hoofdformulier, dus dat halen we op met Me.AuteurID. De volgende regels zijn niet nodig als je op je eigen formulier een eigen object hebt dat de focus krijgt. Die mag je dus naar believen aanpassen of weglaten.

```
Private Sub Form_Close()
   bOpslaan = MsgBox("Wil je de bladwijzer opslaan?", vbYesNo, "Bladwijzer")
    Call WieIsWaar(Me, Me.AuteurID.Name, curAut, bOpslaan)
End Sub
```

Bij het sluiten van het formulier zetten we eerst het antwoord op de vraag "Wil je de bladwijzer opslaan?" in de variabele bOpslaan, die we vervolgens, samen met de Auteur en het actieve record, meegeven met de functie <WieIsWaar>.

De Functieprocedures

Zoals gezegd, zijn er twee procedures nodig. Eén procedure die de gegevens al dan niet opslaat of verwijdert, en één die de gegevens inleest. Dat laatste kan uiteraard alleen als er een record is. Omdat we steeds dezelfde tabel gebruiken, zetten we die in een constante.

```
Option Compare Database
Const sHistorieTabel As String = "tWieWasWaar"
```

De eerste functie is die waarmee we een record opslaan. Deze wordt aangeroepen vanuit het te registreren formulier, en moet een aantal andere gegevens weten, zoals : ‘welk veld moet worden opgeslagen?’ En ‘welke waarde heeft dit veld?’ Deze gegevens kunnen voor elk formulier anders zijn en worden, zoals we hierboven hebben gezien, vanuit het formulier meegegeven. In de functie worden de parameters vervolgens weer uitgelezen.

```
Function WieIsWaar(frm As Form, obj As String, RecID As Long, Opslaan As Variant)
Dim sAuteur As String
 Dim bCheck As Boolean
  Dim rst As DAO.Recordset
   Dim aObj As AccessObject, dbs As Object
    'Stap 1: Controleren of de historietabel bestaat.
     Set dbs = Application.CurrentData
      For Each aObj In dbs.AllTables
      If aObj.Name = sHistorieTabel Then
          bCheck = True
          Exit For
 End If
    Next aObj
     ' ... Zo niet, aanmaken
    If bCheck = False And Opslaan = vbYes Then
```

Naast de gebruikelijke variabelen gebruiken we een nieuw type: het AccessObject. We gebruiken dit om te checken of de tabel [tWieWasWaar] al bestaat. De eerste keer dat een formulier wordt opgeslagen hoeft de tabel nog niet te bestaan, en in dat geval moet hij worden aangemaakt. De check voeren we uit op de verzameling CurrentData van de huidige database, die we in de variabele dbs hebben gezet. Met een lus lopen we door de collectie AllTables om de namen van de huidige tabellen uit te lezen. Deze lus stopt als we de historietabel hebben gevonden, en daarbij wordt de variabele bCheck op Waar gezet. Als de tabel dus niet gevonden wordt, dan blijft de waarde op Onwaar staan. Deze variabele gebruiken we om te controleren of de tabel moet worden aangemaakt. Dat is verder ook afhankelijk van de variabele die we bij het aanroepen van de functie hebben meegegeven. De tabel wordt alleen aangemaakt als bCheck Onwaar is (tabel bestaat niet) en Opslaan de waarde vbYes bevat (historie opslaan).

```
strSQL = "CREATE TABLE " & sHistorieTabel & " " _
  & "([HistorieID] COUNTER CONSTRAINT [HistorieID] PRIMARY KEY, " _
   & "[User] TEXT(100), [Bron] TEXT(100), [Object] TEXT(100), " _
     & "[Bladwijzer] LONG, [DatumGewijzigd] DATETIME )"
      DoCmd.RunSQL (strSQL)
       DoCmd.SetWarnings True
        On Error GoTo 0
 End If
```

We maken een tabel aan met een aantal tekstvelden, een datum/tijd veld en een numeriek veld ([Bladwijzer]) om het actuele recordnummer in op te slaan. Gebruik je ook tekstvelden als sleutel(identificatie)veld dan kun je daar beter een tekstveld van maken. Het is wel nuttig om een sleutelveld aan te maken, en dat doen we in twee stappen. Eerst wordt het veld [HistorieID] als COUNTER aangemaakt, daarna wordt het met de opdracht CONSTRAINT [HistorieID] PRIMARY KEY als primaire sleutel vastgelegd. Dit hoeft overigens niet, want de tabel wordt niet aan andere tabellen gekoppeld.

```
'Stap 2: Controleren of er al een bladwijzer voor het formulier is.
 strSQL = "SELECT * FROM " & sHistorieTabel & " " _
  & "WHERE [User] = '" & Environ("UserName") & "' " _
   & "AND [Bron] = '" & frm.Name & "'"
     Set rst = CurrentDb.OpenRecordset(strSQL)
```

```
  With rst
```

De historietabel wordt nu geopend, en het formulierrecord van de betreffende gebruiker wordt opgezocht. Dit levert een record op, of niet. Dat is afhankelijk van of de gebruiker al eerder op dat formulier is geweest, en een bladwijzer heeft opgeslagen. De eerste controle kijkt of er een record niet bestaat, en of de gebruiker een record wil opslaan. In dat geval moet het record worden opgeslagen.

```
If .RecordCount = 0 And Opslaan = vbYes Then
  ' ... Zo niet, nieuw record aanmaken als Opslaan True is ....
       .AddNew
         !User = Environ("UserName")
          !Bron = frm.Name
           !Object = obj
            !Bladwijzer = RecID
             !DatumGewijzigd = Now()
         .Update
```

Als de gebruiker ervoor heeft gekozen om de bladwijzer niet op te slaan, moet het record worden verwijderd. Bij het opnieuw openen van het formulier moet het formulier dan weer op de standaard manier geopend worden. Uiteraard alleen als het record bestaat, dus we checken met .RecordCount het aantal records. Dat zou overigens 1 moeten opleveren, dus .RecordCount -= 1 zou ook moeten werken. Een record verwijderen doe je met de eigenschap .Delete.

```
 ElseIf .RecordCount > 0 And Opslaan = vbNo Then
   ' ... Het bestaande record verwijderen als Opslaan False is....
       .Delete
```

In alle overige gevallen moet het bestaande record worden bijgewerkt met de nieuwe bladwijzer.

```
 Else
   ' ... En anders het bestaande record bijwerken met nieuwe bladwijzer.
       .Edit
         !Bladwijzer = RecID
          !DatumGewijzigd = Now()
           .Update
End If
  .Close
End With
End Function
```

De recordset wordt in Edit modus gezet, en bladwijzer en datum worden bijgewerkt.

De volgende functie haalt het record weer op als het formulier wordt geopend. De functie heeft alleen de naam van het formulier nodig als input; de naam van de gebruiker wordt al uit de Environ variabele gehaald. En de overige gegevens staan in de historietabel.

```
Function WieWasWaar(frm As Form)
Dim rst As DAO.Recordset
Dim varObj As Variant
Dim sBookmark As String, sVeld As String
Dim i As Integer
    On Error Resume Next
    strSQL = "SELECT ObjectNaaam, Bladwijzer FROM " & sHistorieTabel & " " _
    & "WHERE (User='" & Environ("UserName") & "' AND Bron='" & frm.Name & "');"
    Set rst = CurrentDb.OpenRecordset(strSQL)
    With rst
```

Nadat de query is gemaakt en geopend, doen we weer een Recordcount om de vervolgacties te bepalen.

```
 i = .RecordCount
  If i > 0 Then
    sVeld = .Fields(0).Value
      sBookmark = .Fields(1).Value
```

```
 End If
  .Close
End With
   If i > 0 Then
    Set rst = frm.RecordsetClone
     With rst
```

Op basis van de variabelen sVeld en sBookmark die we vullen vanuit de recordset maken we nu een kopie van de Recordset van het formulier met frm.RecordsetClone. In de kopie zoeken we met .FindFirst het eerste record op dat voldoet aan de zoekcriteria. Er is maar één record dat voldoet overigens, maar FindFirst werkt prima in dit geval. Als het record gevonden wordt (en theoretisch moet dat dus altijd het geval zijn) zetten we dat record over naar het formulier met frm.Bookmark = rst.Bookmark. Deze opdracht triggert de gebeurtenis <Bij aanwijzen> van het formulier, want er wordt nu een ander record op het formulier gezet. Vervolgens wordt alles afgesloten, en keert de functie weer terug naar het formulier.

```
 .FindFirst sVeld & " = " & sBookmark
   If rst.NoMatch Then
    MsgBox "Record not found"
  Else
    frm.Bookmark = rst.Bookmark
End If
  .Close
End With
End If
End Function
```

Samenvatting

In dit hoofdstuk hebben we twee manieren gezien waarop je een bladwijzer kunt maken in een formulier. Met deze bladwijzerfunctie kun je de locatie opslaan van het record waarin je als laatste hebt gewerkt. Je kunt daarbij kiezen uit een simpele methode die wel een extra veld in de Recordbron van het formulier nodig heeft, en die alleen op dat formulier werkt voor één record, of je gebruikt een methode die alle bladwijzers opslaat in een aparte tabel. De laatste oplossing is multi-inzetbaar, en werkt voor meerdere gebruikers.

De tweede methode kun je ook prima gebruiken om algemene mutaties in op te slaan. Stel dat je in een klantentabel mutaties van adressen of contactpersonen wilt bijhouden. Dan kun je daar prima een aparte tabel voor maken die je koppelt aan de adressentabel. Maar de oude gegevens zullen niet vaak gebruikt gaan worden; als een contactpersoon bij een klant weg is, kan je die niet meer bellen. Dus eigenlijk ben je alleen geïnteresseerd in de actuele gegevens. Ander voorbeeld: prijsmutaties voor je productentabel. Je werkt in je verkooptabel met de actuele prijs, maar het kan best nuttig zijn om de vorige prijzen in een historietabel te hebben, zodat je daar statistieken en analyses op los kunt laten. In dat soort gevallen heb je dus nut van een historietabel.