Access Cursus hoofdstuk 23

Handleiding van Helpmij.nl

Auteur: OctaFish

Augustus 2014

Verplaatsen met de Muis

Iedereen werkt met een muis: objecten selecteren, programma's starten, vensters verplaatsen, (snel)menu's openen. Wie heeft dat nog nooit gedaan? En ook in een database is de muis heel goed te gebruiken. Je dubbelklikt op een formulier om dat te openen, klikt op een keuzelijst om elementen te selecteren, klikt op een knop om een actie uit te voeren. En veel handelingen kun je ook met de rechter muisknop uitvoeren, als die niet is uitgeschakeld. Voor gebruikers houdt de muishandeling daar meestal op, want typische muishandelingen als slepen worden meestal niet gebruikt. Je kunt in een tabel nog wel slepen om records te selecteren, maar als ontwerper wil je meestal niet dat een gebruiker überhaupt in die tabel kan komen. Je wilt namelijk niet dat een gebruiker met één onverwachte beweging een aantal records tegelijk wist, en dus zul je in de praktijk gebruikers een aantal formulieren en rapporten voorschotelen waarin ze hun activiteiten kunnen uitvoeren.

Als *ontwerper* van een database zul je eerder met muishandelingen werken, want om een aantal objecten tegelijk te selecteren gebruik je vaak een sleeptechniek, en voor het verplaatsen natuurlijk ook. Dat soort selecties kun je echter nooit 'voorkoken' of anderszins afvangen, en dat gaan we dan ook niet doen.

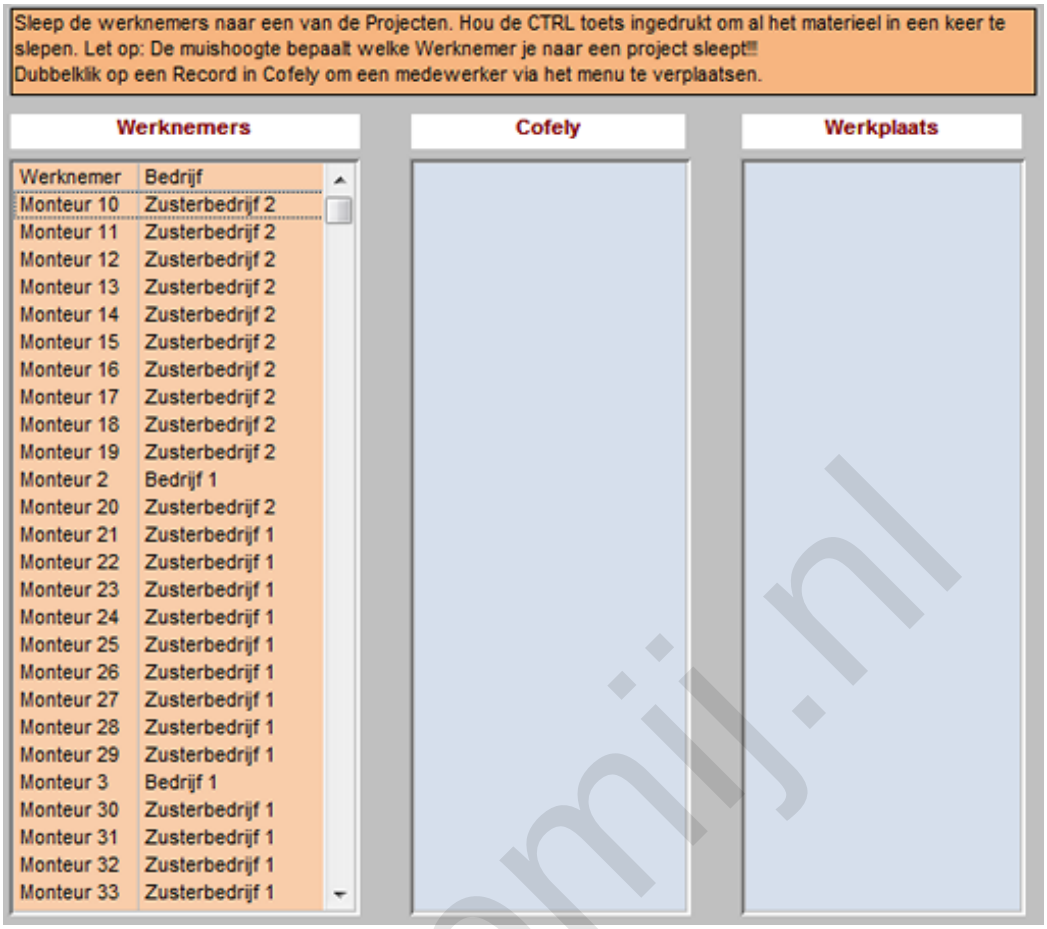
In dit hoofdstuk behandel ik de gebeurtenissen <Bij muis omlaag>, <Bij muis verplaatsen> en <Bij muis omhoog>. Zoals de namen al aangeven, zijn dit gebeurtenissen die reageren op een muishandeling. Daarbij gaan we uit van een 'natuurlijke' volgorde: vóór dat de muis verplaatst wordt, moet de muisknop eerst worden ingedrukt, en nadat je klaar bent met de verplaatsing, laat je de muisknop weer los. De volgorde waarin de procedures worden uitgevoerd is dus:

1. <Bij muis omlaag>
2. <Bij muis verplaatsen>
3. <Bij muis omhoog>

Elk van deze acties kan zelfstandig gebruikt worden; als je bijvoorbeeld een helptekst wilt laten zien in een label, kun je best volstaan met de gebeurtenis <Bij muis omlaag>. Het boeit daarbij niet dat de gebruiker de muisknop ooit weer loslaat en daarbij de gebeurtenis <Bij muis omhoog> triggert; als je daar geen code bij zet dan gebeurt er verder niks. Al zou je een helptekst bij <Bij muis omhoog> natuurlijk weer kunnen laten verdwijnen.

Waarden uit een keuzelijst slepen

Laten we eens naar onderstaande afbeelding kijken:



Op het formulier staat een aantal keuzelijsten (waarvan we er 3 zien) waarvan de keuzelijst *Werknemers* monteurs bevat die we gaan inplannen op een locatie. We doen dat door een waarde uit de eerste keuzelijst te slepen naar de gewenste keuzelijst. En door deze actie worden op de achtergrond de juiste gegevens bijgewerkt.

In een hoop programma's zou zo'n handeling heel simpel zijn, maar in Access is dat nog een hele klus! En daarmee dus automatisch een goed onderwerp voor deze cursus.

Laten we eens bekijken welke handelingen er plaatsvinden:

1. Alles begint bij het selecteren van een werknemer uit de keuzelijst *Werknemers*. Daar vindt dus de gebeurtenis <Bij muis omlaag> plaats.
2. Vervolgens moet de gekozen monteur verplaatst worden naar de juiste keuzelijst. Deze gebeurtenis vangen we af onder de gebeurtenis <Bij muis verplaatsen>. Je zou verwachten dat er bij deze gebeurtenis heel wat plaatsvindt, maar in de praktijk is deze gebeurtenis nauwelijks interessant.
3. Als de muis boven de juiste keuzelijst 'hangt', kan de knop worden losgelaten. Hier vindt dus de gebeurtenis <Bij muis omhoog> plaats. Deze gebeurtenis doet het eigenlijke werk: het verplaatsen van de geselecteerde monteur van de lijst met Werknemers naar de lijst van de gekozen keuzelijst.

Keuzelijst acties

De eerste actie die we uitvoeren is op de keuzelijst Werknemers, die in het voorbeeld List1 heet. De reden hiervoor wordt later nog uitgelegd. Op de keuzelijst zijn 3 gebeurtenissen ingesteld:

Code bij de gebeurtenis <Bij Muis omlaag>

```
Private Sub List1_MouseDown(Button As Integer, _
    Shift As Integer, x As Single, y As Single)
    Me.Label6.Caption = "Drag to other list box."
    DragStart Me
End Sub
```

Code bij de gebeurtenis <Bij Muis verplaatsen>

```
Private Sub List1_MouseMove(Button As Integer, _
    Shift As Integer, x As Single, y As Single)
    DropDetect Me, Me.List1, Button, Shift, x, y
End Sub
```

En Code bij de gebeurtenis <Bij Muis omhoog>

```
Private Sub List1_MouseUp(Button As Integer, _
    Shift As Integer, x As Single, y As Single)
    Me.Label6.Caption = ""
    DragStop
End Sub
```

"Is dat alles?" Zul je je nu wellicht afvragen? Natuurlijk niet; elke gebeurtenis die je hierboven ziet bevat een aanroep naar een aparte functie. En die functies zijn uiteraard het hart van dit hoofdstuk. Toch is er wel wat bijzonders bij de functies, en daarom begin ik toch met de uitleg zonder gelijk naar de functies te kijken.

De gebeurtenis <Bij muis omlaag> kent een paar parameters die belangrijk zijn. Om te beginnen de parameter *Button*: welke knop is er ingedrukt? De volgende parameter is *Shift*, en die kijkt of de Shift toets op het toetsenbord is ingedrukt of niet. Zoals je vermoedelijk wel weet, heeft slepen met de Shift toets ingedrukt in Windows een ander effect, en die eigenschap werkt ook in Access. De volgende twee parameters (x en y) lezen de muispositie op het scherm af op het moment dat de muis wordt ingedrukt.

Iets vergelijkbaars zien we bij de gebeurtenis <Bij Muis verplaatsen> en bij de gebeurtenis <Bij Muis omhoog>. De parameters worden vervolgens gebruikt bij de functies *DragStart*, *DropDetect* en *Dragstop*.

De functie DragStart begint altijd op een actief object, meestal een keuzelijst of tekstvak op een formulier. De functie gebruikt het object *Me* dan ook als parameter, en dat is dus het actieve formulier. Ga je de muis verplaatsen, dan treedt de functie DropDetect in werking, die continue afleest waar de muis zich bevindt. En de functie DragStop wordt gestart zodra de muisknop wordt losgelaten.

Laten we de functies eens bekijken! Deze staan in een eigen module, zodat we ze op elk willekeurig formulier kunnen gebruiken.

De variabelen

Voordat we naar de functies gaan kijken, nemen we eerst de variabelen onder de loep, want we hebben een aantal instellingen nodig voordat we de functies kunnen gebruiken.

```
Dim DragFrm As Form
Dim DragCtrl As Control
Dim DropTime As Variant
Dim CurrentMode As Integer

Const MAX_DROP_TIME = 0.1
Const NO_MODE = 0
Const DROP_MODE = 1
Const DRAG_MODE = 2
```

We beginnen met het vastleggen van variabelen voor het formulier (DragFrm), het object (DragCtrl) en tijd (DropTime). Ook leggen we de variabelen CurrentMode en een aantal constanten vast. De constante MAC_DROP_TIME gebruiken we om te controleren of de muisknop nog is ingedrukt of niet.

1e Functie: Dragstart

De functie Dragstart doet eigenlijk ook nog steeds niet zoveel: hij kijkt naar het formulier waar vandaan de sleep actie wordt gestart, en van welk object. Tevens wordt een systeemvariabele ingesteld: de modus CurrentMode. Oftewel: we beginnen een sleepactie met de muis. Het startformulier wordt meegegeven in de functieaanroep (DragStart Me), het object dat als startpunt dient bepalen we middels Screen.ActiveControl.

```
Sub DragStart(SourceFrm As Form)
    Set DragFrm = SourceFrm
    Set DragCtrl = Screen.ActiveControl
    CurrentMode = DRAG_MODE
End Sub
```

Eigenlijk zou je ook CurrentMode =2 kunnen gebruiken, maar door de waarde in een constante te zetten, heb je visuele controle op de betekenis van de waarde. Het slepen is nu dus gestart!

2e Functie: DragStop

De tweede functie waar we naar kijken is niet de gebeurtenis <Bij muis verplaatsen>, zoals je misschien zou verwachten, maar de gebeurtenis <Bij muis omhoog>. De sleepfunctie wordt immers bepaald tussen het moment dat de muis wordt ingedrukt, en het moment dat de muisknop wordt losgelaten. Die twee momenten verschillen in tijd (je kunt niet in één moment de muisknop indrukken en loslaten), en op basis daarvan wordt de sleep bepaald.

We kijken dus eerst naar wat er gebeurt als we de muisknop loslaten. Die code ziet er zo uit:

```
Sub DragStop()
    CurrentMode = DROP_MODE
    DropTime = Timer
End Sub
```

We zien dat de variabele CurrentMode de waarde van DROP_MODE krijgt, en dat de variabele DropTime met de tijd (middels de standaard functie Timer) wordt gevuld. Nu we weten wanneer het verplaatsen is gestopt, kunnen we bepalen waar de muis zich nu bevindt.

3e Functie: DropDetect

Nu we in de sleepmodus zitten, kunnen we de muis verplaatsen. We moeten nu de juiste locatie van de muis registreren. De functie die we gebruiken ziet er nu zo uit:

```
Sub DropDetect(DropFrm As Form, DropCtrl As Control, _
Button As Integer, Shift As Integer, x As Single, y As Single)
If CurrentMode <> DROP_MODE Then Exit Sub
CurrentMode = NO_MODE

If Timer - DropTime > MAX_DROP_TIME Then Exit Sub
On Error Resume Next
If (DragCtrl.Name <> DropCtrl.Name) _
    Or (DragFrm.Hwnd <> DropFrm.Hwnd) Then
    DragDrop DragFrm, DragCtrl, DropFrm, DropCtrl, _
        Button, Shift, x, y
End If
End Sub
```

We zien in de eerste regel code dat het slepen afhankelijk is van de gebeurtenis <Bij muis omhoog>. Deze heeft de status van CurrentMode op DROP_MODE gezet, en die status hebben we nodig om te bepalen wát er nu eigenlijk geselecteerd is. Dus zolang de status DROP_MODE niet actief is, blijven we slepen.

We hebben eerder gezien dat de Timer ook geactiveerd is bij die gebeurtenis. Nu kijken we of de huidige tijd (die immers doorloopt) de ingestelde waarde van de variabele MAX_DROP_TIME overschrijdt. We trekken daarvoor de DropTime, die is vastgelegd bij het loslaten van de muisknop, af van de huidige Timer waarde. Is de waarde te groot, dan gaat de procedure er vanuit dat er weliswaar een knopactie is geweest, maar dat er in nog steeds gesleept wordt. De DropDetect procedure wordt dan alsnog stopgezet.

We zijn nu een stukje verder, want in de code zijn we op het punt aanbeland dat aan alle sleepvoorwaarden is voldaan. We gaan nu kijken *waar* de muis is losgelaten. Daarbij geldt de voorwaarde dat we er weinig mee opschieten als de muis boven de keuzelijst wordt losgelaten waar we zijn gestart, want dan zouden we een waarde verslepen naar zichzelf. Dat gebeurt met If (DragCtrl.Name <> DropCtrl.Name). En dat is natuurlijk meestal een zinloze handeling.

Ook controleren we welk formulier we gebruiken. Daarvoor gebruiken we een formuliereigenschap die gebruikt wordt in Windows API calls: de eigenschap hWnd. Deze eigenschap heet in het Engels een *handle* en in het Nederlands is dat vertaald als *ingang.* De eigenschap retourneert een getal dat gebruikt wordt om een formulier te identificeren. Als je van het ene naar het andere formulier sleept, dan zijn de hWnd waarden van de formulieren dus niet gelijk. En dan is precies wat we willen controleren bij het slepen: Or (DragFrm.Hwnd <> DropFrm.Hwnd).

We hebben de voorwaarden gecontroleerd waaraan de sleepactie moet voldoen, en nu kunnen we echt aan de slag; er wordt namelijk een nieuwe functie gestart met alle variabelen als argument. Deze functie heet *DragDrop.*

De functie DragDrop

Ook deze functie is nog niet het eindresultaat, want de functies die we tot hier toe hebben gebruikt zijn universeel, en kunnen dus overal waar gesleept kan worden aan een object worden gehangen. Alleen zal er niet altijd dezelfde actie hoeven te worden uitgevoerd, en daarom gebruiken we een speciale functie die kijkt wat op welk moment moet worden gedaan. In het voorbeeld zie je één specifieke actie, maar dat kunnen er dus veel meer zijn.

```
Function DragDrop(DragFrm As Form, DragCtrl As Control, _
    DropFrm As Form, DropCtrl As Control, _
    Button As Integer, Shift As Integer, _
    x As Single, y As Single)
    Select Case DropFrm.Name
        Case "frmDragDropListBoxes"
            ListBoxMove DragFrm, DragCtrl, DropFrm, DropCtrl, _
                Button, Shift, x, y
        Case Else
            On Error Resume Next
            DropCtrl = DragCtrl
            If Err Then MsgBox Error$
    End Select
End Function
```

De functie heeft weer de inmiddels bekende parameters, zoals DragFrm en DragCtrl, de waarden voor de extra toetsen en de muispositie. De actie wordt gestuurd door de naam van het doelformulier, en daar wordt de Select Case dan ook op gebaseerd. We bekijken in deze code de laatste gebeurtenis, die het echte werk doet: de functie ListBoxMove.

De functie ListBoxMove

Het uiteindelijke doel van de sleepactie is het ‘verplaatsen’ van een waarde van de ene keuzelijst naar de andere keuzelijst. Ik heb tot nu toe de indruk gewekt dat dit mogelijk is, maar de waarheid gebiedt mij te zeggen dat dat niet klopt. In Access kun je niets verslepen. Wat er feitelijk gebeurt is niets anders dan dat de geselecteerde waarde(n) wordt/worden gekopieerd naar de andere keuzelijst. Hoe dat gebeurt, hangt een beetje af van de inrichting van de database. In dit voorbeeld gebruiken we selectievakjes in een tabel om aan te geven welke werkplaats actief is. Dat betekent dus een routine die de startwaarde deselecteert, en de gekozen waarde activeert. En dat gebeurt in de procedure ListBoxMove. De code ziet er daarom zo uit:

```
Sub ListBoxMove(DragFrm As Form, DragCtrl As Control, _
    DropFrm As Form, DropCtrl As Control, _
    Button As Integer, Shift As Integer, _
    x As Single, y As Single)
Dim db As Database
Dim SQL As String

Set db = CurrentDb()
If Not DropCtrl.Tag = "" Then
    SQL = "UPDATE tblDragListBox SET "
    If Not DragCtrl.Tag = "" Then
        SQL = SQL & "[" & DragCtrl.Tag & "] = Not [" & DragCtrl.Tag & "]," _
            & "[" & DropCtrl.Tag & "] = Not [" & DropCtrl.Tag & "]"
    Else
        SQL = SQL & "[" & DropCtrl.Tag & "]= Not [" & DropCtrl.Tag & "]"
    End If
Else
    SQL = "UPDATE tblDragListBox " _
        & "SET [" & DragCtrl.Tag & "]= Not [" & DragCtrl.Tag & "]"
End If

If (Shift And CTRL_MASK) = 0 Then
    SQL = SQL & " WHERE [ID Mur]='" & DragCtrl & "'"
End If
db.Execute SQL
DragCtrl.Requery
DropCtrl.Requery

End Sub
```

Als we de controleslag even laten voor wat hij is, dan zien we de opbouw van een SQL string die

uiteindelijk een Update query samenstelt. Dat gebeurt op basis van de TAG eigenschap van de keuzelijsten. Deze techniek heb ik vaker gebruikt, dus ik hou de uitleg hier kort: de eigenschap <Extra Info> (of Tag in het Engels) wordt hier gebruikt om de *veldnaam* in op te slaan van het tabelveld waarop de keuzelijst is gebaseerd. En die eigenschap wordt nu uitgelezen en in de SQL string gezet. Omdat we te maken hebben met selectievakjes die voor ons doel maar 2 instellingen hebben (aangevinkt of niet aangevinkt) kunnen we de onderliggende velden heel simpel instellen door de velden bij te werken met de omgekeerde waarde. De query zou dus kunnen luiden:

UPDATE tblDragListBox SET [Project 1] = Not [Project 1]

Het vinkje wordt dus aangezet of uitgezet, afhankelijk van de huidige waarde. Dit doen we in de code voor zowel het bronobject als het doelobject dat logischerwijze de omgekeerde waarde heeft van het bronobject. En met deze query verdwijnt de waarde dus uit de ene keuzelijst, en verschijnt hij in de andere. Het enige dat nog nodig is om dat visueel te maken is het verversen van de keuzelijsten op het formulier, en dat gebeurt door beide keuzelijsten een Requery te geven.

Samenvatting

Access kent drie soorten muisgebeurtenissen die we kunnen gebruiken om de sleeptechniek te simuleren. Daarbij kunnen we bijvoorbeeld waarden uit een tekstvak of een keuzelijst slepen naar een ander tekstvak of keuzelijst. Hierbij worden verschillende acties gebruikt om te controleren welke toetsen er zijn ingedrukt bij het slepen, en welke objecten zijn betrokken bij het slepen. Uiteindelijk wordt met een Update procedure de onderliggende tabel bijgewerkt, zodat de gesleepte waarden in de juiste velden in de tabel worden opgeslagen.

Wat ik in de uitleg nog niet heb verteld, is dat de techniek één groot nadeel heeft. En dat is, dat de sleepactie actief blijft op het bronobject tot de muis wordt losgelaten. In de praktijk merk je dat snel genoeg, want je ziet de geselecteerde waarde in de bron keuzelijst veranderen als je de muis naar boven of naar beneden verplaatst. Je begint dus te slepen op de derde waarde in de keuzelijst, maar tegen de tijd dat je met de muis boven de doel keuzelijst hangt, kan de selectie best op de vijfde waarde staan. En dát is dan het veld dat je aanpast! Het luistert dus wel enigszins nauwkeurig bij het slepen, want anders verplaats je de verkeerde waarde.

Ik zoek nog steeds naar een manier om de ‘sleep’ op de bron keuzelijst te bevriezen zodra de muis de rand ervan passeert, en daarmee dus de waarde(n) van die keuzelijst vast te leggen. Dan is in het vervolgtraject van de sleepbeweging alleen nog de naam van het doel object nodig om de Update te kunnen uitvoeren. Maar die techniek heb ik nog niet gevonden. Zodra ik daar een werkende techniek voor heb, zal ik die uiteraard ook publiceren in de Nieuwsbrief!

Omdat er nogal wat code aan de hele actie vast zit, zal ik die niet in dit artikel zetten, maar een werkende database posten in het bij deze cursus horende topic in het Access forum.