Een Multi-functioneel formulier

Handleiding van Helpmij.nl

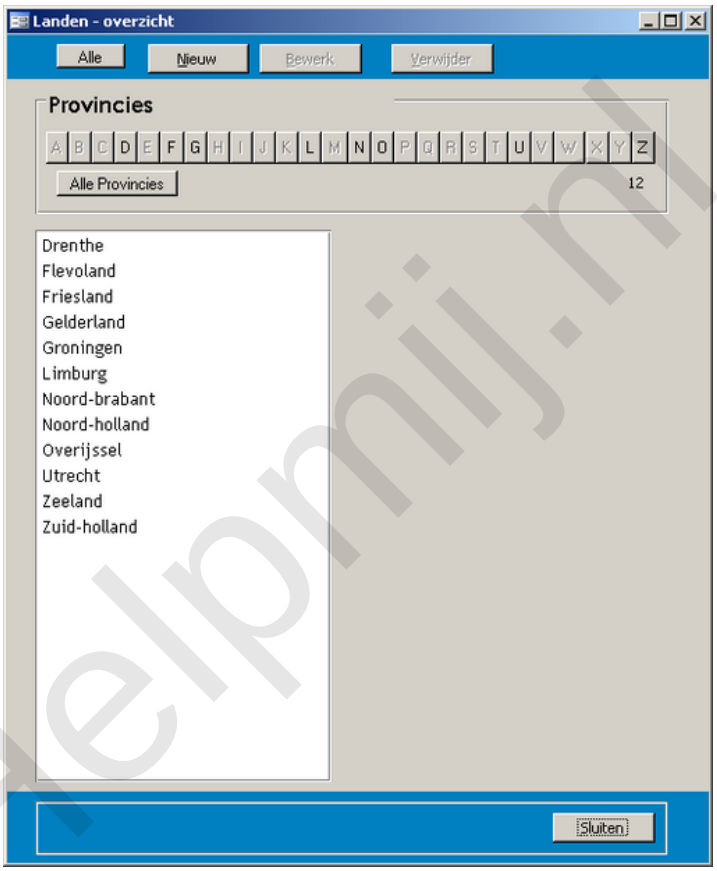
Auteur: OctaHish

December 2012

Een Multi-functioneel formulier

In dit hoofdstuk pakken we de originele draad van het begin van de cursus weer op, en gaan we weer eens kijken naar de Duikclub database. Deze database kan namelijk nog wel wat formulieren etc. gebruiken!

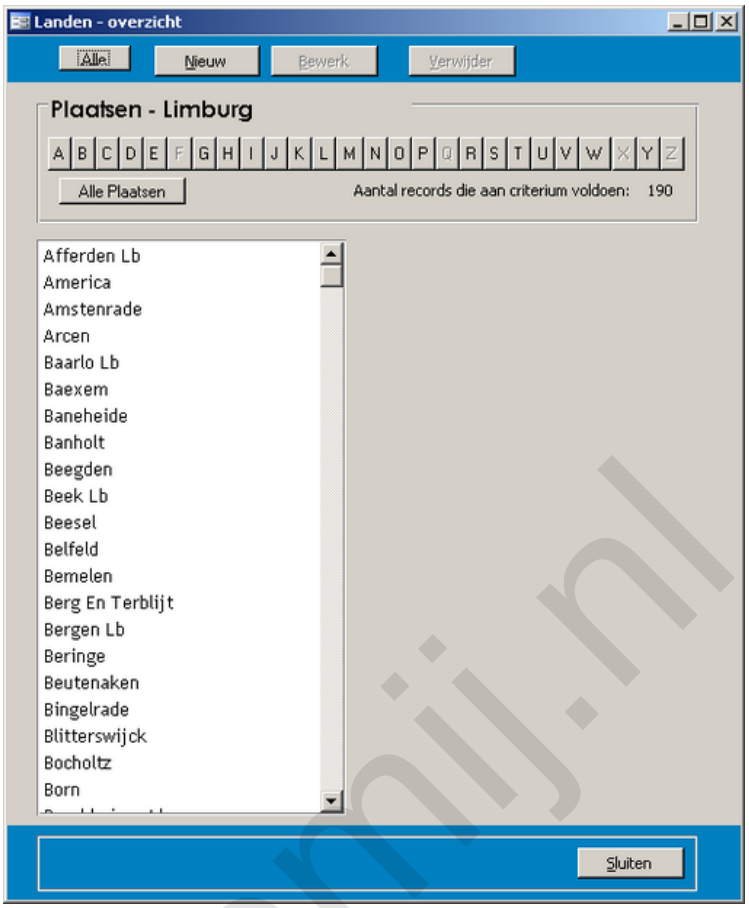
Deze aflevering gaan we een formulier maken dat fungeert als een telefoonklapper. Zo'n formulier gebruikt knoppen om lijsten/subformulieren te filteren, en is dus bij uitstek geschikt om met grote bestanden te werken. Denk bijvoorbeeld aan de Postcodetabel, waarin alle plaatsnamen en straatnamen van Nederland of België zijn opgeslagen.



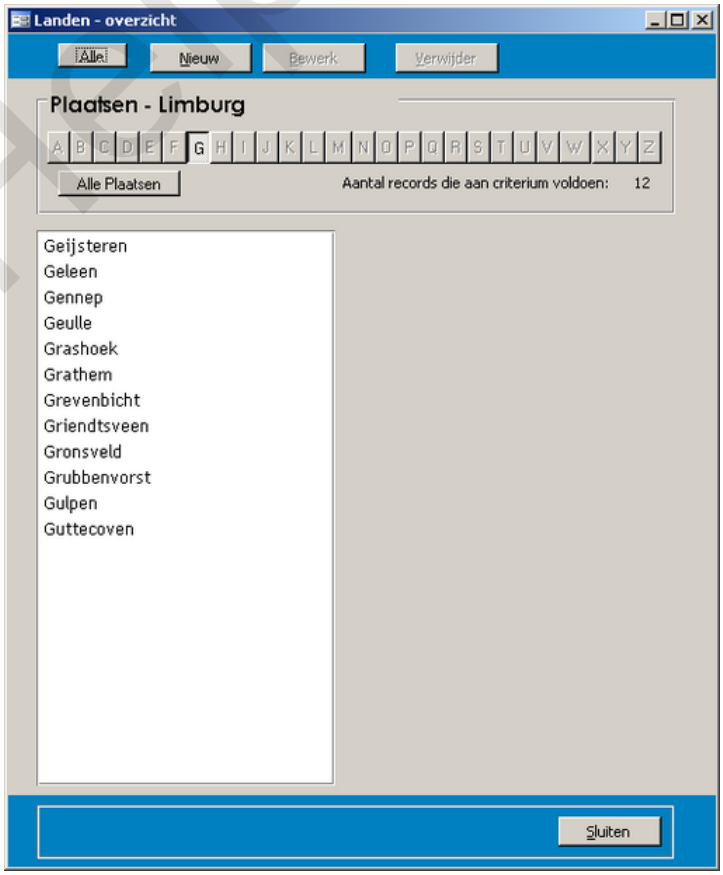
In dit plaatje zie je een overzicht van provincies in Nederland, met daarboven knoppen waarmee je de lijst kunt filteren. In dit geval overigens niet echt nodig. Klik je op een provincienaam, dan verandert het formulier, en zie je de steden van die provincie.

In de rij met knoppen zijn alleen de knoppen actief waarvan de tekst op de knop overeenkomt met de beginletter van een provincie. Je kunt dus alleen filteren op bestaande lettercombinaties.

Klik je op (bijvoorbeeld) Limburg, dan zie je de plaatsen in Limburg. Omdat de lijst nu een stuk langer is, zijn er meer letterknoppen actief. Uiteraard kun je scrollen door de lijst, maar je kunt ook filteren door op een letterknop te klikken.

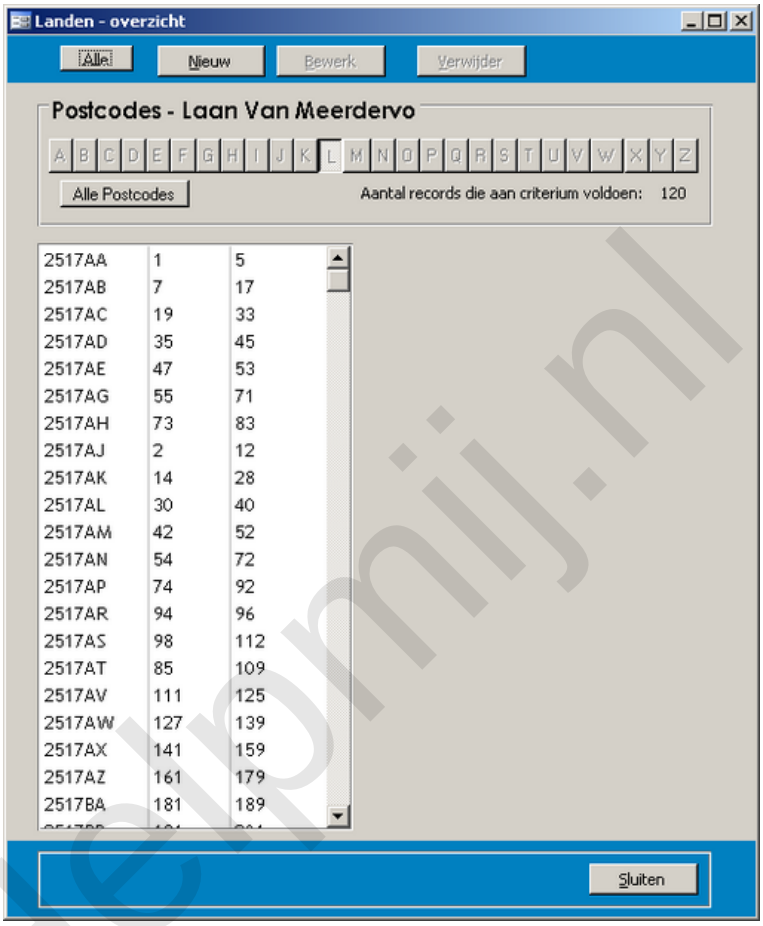


Klik je bijvoorbeeld op <G>, dan zie je de volgende plaatsnamen:



De lijst is nu overzichtelijk. Wil je terug naar het overzicht van alle plaatsnamen in Limburg, dan klik je op de knop <Alle Plaatsen>. Je krijgt dan weer het overzicht van de vorige afbeelding. Met een volgende keuze krijg je van de gekozen plaats de Straatnamen te zien, met weer dezelfde keuzemogelijkheden.

De laatste laag bevat de postcodes van de gekozen straat.



Het lijkt er op alsof het formulier op één bron werkt, en dat je alleen maar filtert binnen die bron, maar dat is niet zo. Bij elke keuze die je maakt, wordt een andere tabel als bron voor het formulier gebruikt. Al is dat eigenlijk ook niet helemaal waar, want het formulier gebruikt wel degelijk één vaste tabel als bron. Maar deze tabel wordt steeds opnieuw aangemaakt, en de veldnamen en gegevens in die (tijdelijke) tabel worden vastgesteld in de procedure die binnen het formulier gebruikt wordt.
Hoe de velden in die bronnen heten, doet daarbij dus niet eens zoveel ter zake, omdat de tijdelijke tabel in de procedure gevuld wordt met vaste veldnamen.

In dit hoofdstuk zitten ook functies die gebruikt worden om ook te kunnen filteren met diacrieten. Dit zijn (mede)klinkers met accenten,zoals Ä, Ñ en Ç. Zou je alleen op beginletter filteren, dan vallen er namelijk steden, straatnamen etc. af die met zo'n diacriet beginnen. Deze functies komen uiteraard ook aan bod. Laten we de procedure eens globaal doornemen!

De elementen

Om te beginnen de knoppen: dat zijn Wisselknoppen, die zijn gemaakt met de wizard <Groepsvak>. Met de wizard kun je (in Access 2003) maximaal 20 knoppen maken, maar de overige 7 kun je probleemloos kopiëren. Het groepsvak krijgt de naam [fraSelect].

Het volgende stuk is daarna wel wat werk, want elke knop moet een logische naam krijgen, en de standaardnamen die Access aan knoppen geeft is minder geschikt. Een naam voor een knop kan het bijschrift ervan zijn (de Beginkapitaal dus), maar je mag de knoppen ook een nummer geven dat

overeenkomt met de ASCII waarde van de bijbehorende letter. In het voorbeeld heten de knoppen A, B, C ... X, Y, Z en met ASCII waarden zou dat 65, 66, 67 .., 88, 89, 90 zijn. De naam stel je in via de <Eigenschappen> van de knop.

Voordeel van de laatste methode is, dat je straks geen conversie meer hoeft te doen van Letter naar Letterwaarde. Het loont dus om vooruit te denken :)

De knoppen staan in een groepsvak, zodat er maar één letter tegelijk ingedrukt kan zijn. Je mag uiteraard ook kiezen voor radiorondjes, of selectievakjes, waarmee je dan de selectie op basis van meerdere letters kunt laten uitvoeren, maar die laatste techniek wordt in dit hoofdstuk niet behandeld.

Verder staat er op het formulier een Keuzelijst (Listbox) met de naam < lstSelection> die niet gebonden is aan een rijbron, want er is steeds een andere rijbron nodig voor de keuzelijst. Het formulier is overigens ook een niet-gebonden formulier. Je muteert dus geen gegevens in de tabellen.

In de koptekst van het formulier staan nog twee verborgen tekstvakken met de namen [txtNiveau] en [txtLijst]. Deze tekstvakken zijn nodig om de statussen van het formulier bij te houden (welke tabel moet worden gebruikt, en welk veld is aangeklikt in de keuzelijst).
En dat zijn zo'n beetje alle noodzakelijke elementen om het formulier te kunnen maken!

De opbouw

Een formulier als dit drijft natuurlijk op VBA. Dat begint al gelijk bij het openen van het formulier, met deze code:

```
Private Sub Form_Open(Cancel As Integer)
    Me.txtNiveau = 1
    Call FormulierAanpassen(1)
    With Me.lblResult
        .Visible = False
        .ValidationRule = False
    End With
End Sub
```

Hier wordt de waarde 1 toegekend aan het tekstvak [txtNiveau]. Op zich logisch: als je het formulier opent, begin je op het hoogste niveau. Je zult zien dat deze waarde bij elk lager niveau wordt opgehoogd, en dat op basis van de nieuwe waarde andere bronnen worden geselecteerd.

In de volgende regel wordt de functie <FormulierAanpassen> aangeroepen met een parameter. En hier gebeurt het echte werk! We gaan de functie dus weer zoals gewoonlijk blokje voor blokje bekijken.

```
Function FormulierAanpassen(Niveau As Integer, Optional Selectie As
Integer)
Dim qTmp As DAO.QueryDef
Dim fld As DAO.Field
    Select Case Me.txtNiveau
        Case 1
            strSQL = "SELECT DISTINCT StrConv(Provincie,3) As V1, " _
                 & "ProvincieID As V2, " _
                 & "EersteLetter(Provincie) AS Beginletter " _
                 & "FROM Provincie "
            If Selectie > 0 Then strSQL = strSQL & "WHERE Provincie " _
                 & "Like " & KnoppenFilter(Me.fraSelect)
            strSQL = strSQL & "ORDER BY EersteLetter(Provincie);"
            Me.lstSelection.ColumnWidths = "6cm;0cm;0cm;0cm"
        Case 2
            strSQL = "SELECT DISTINCT StrConv(Plaats,3) As V1, " _
                 & "PlaatsID  As V2,  " _
                & "EersteLetter(Plaats) AS Beginletter " _
```

```
            & "FROM Plaats " _
            & "WHERE ProvincieID=" & Me.txtLijst & " "
        If Selectie > 0 Then strSQL = strSQL & "AND Plaats " _
             & "Like " & KnoppenFilter(Me.fraSelect)
        strSQL = strSQL & "ORDER BY EersteLetter(Plaats);"
    Case 3
        strSQL = "SELECT DISTINCT StrConv(Straat,3) As V1, " _
             & "StraatID As V2, " _
            & "EersteLetter(Straat) AS Beginletter " _
            & "FROM Straat " _
            & "WHERE PlaatsID = " & Me.txtLijst & " "
        If Selectie > 0 Then strSQL = strSQL & "AND Straat " _
             & "Like " & KnoppenFilter(Me.fraSelect)
        strSQL = strSQL & "ORDER BY EersteLetter(Straat);"
    Case 4
        strSQL = "SELECT Postcode As V1, Van As V2, " _
             & "Tem As V3, Soort, " _
            & "EersteLetter(Postcode) AS Beginletter " _
            & "FROM Postcodes " _
            & "WHERE StraatID = " & Me.txtLijst & " "
        strSQL = strSQL & "ORDER BY Postcode, Soort, Van"
        Me.lstSelection.ColumnWidths = "2cm;1,5cm;1,5cm;0cm"
    Case Else
End Select
```

In de naam van de functie staan de parameters die hij accepteert: (Niveau As Integer, Optional Selectie As Integer). Er is één verplichte parameter (Niveau) en een optionele: Selectie. Deze laatste wordt gebruikt om te filteren op de keuzelijst, die niet altijd een selectie zal hebben of actief zal zijn. Op het moment dat een keuzelijst de focus niet meer heeft, kun je de waarde ervan niet meer uitlezen. Daarom wordt de waarde bij het klikken weggeschreven naar een apart tekstvak op het formulier.

Vervolgens wordt op basis van het tekstvak <txtNiveau> gekeken welke query er gemaakt moet worden. Zoals je kunt zien, worden er verschillende tabellen gebruikt: [Provincie], [Plaats], [Straat] en [Postcode]. Dit hoeft overigens niet, want je kunt het systeem ook gebruiken om bijvoorbeeld categorieën en subcategorieën te tonen die in dezelfde tabel staan. In de extra oefening van dit hoofdstuk ga je een vergelijkbaar formulier in de Duikclub database maken we dat is gebaseerd op de tabel st_Duiklocaties. Gebruik je één tabel, dan zul je vaak in die tabel de verschillende niveaus al vastleggen, en dat is exact wat er gebeurt in die tabel.

Wat misschien nog opvalt, is dat bij zowel Case 1 als bij Case 4 een extra opdracht wordt gegeven, resp.
Me.lstSelection.ColumnWidths = "6cm;0cm;0cm;0cm"
en
Me.lstSelection.ColumnWidths = "2cm;1,5cm;1,5cm;0cm".
Deze regels zijn nodig omdat de layout van de tabel Postcodes niet overeenkomt met die van de overige tabellen: er zijn meer velden, en je wilt die velden ook in de keuzelijst zien. Bij Case 1 t/m Case 3 is de layout gelijk, dus je hoeft alleen bij Case 1 de layout aan te passen, en uiteraard dus bij Case 4.

Elke query gebruikt twee aparte functies, en die vind je hier:

```
"Plaats Like " & KnoppenFilter(Me.fraSelect)
"ORDER BY EersteLetter(Plaats)"
```

Deze functies worden straks behandeld. Met deze functies wordt het filter gemaakt (Knoppenfilter) en worden alle letters met diacrieten verzameld (EersteLetter).

```
    On Error GoTo MaakQuery
    Set qTmp = CurrentDb.QueryDefs("qTemp")
    qTmp.SQL = strSQL
    On Error GoTo 0
```

Er wordt een vaste query gebruikt voor het formulier, maar deze krijgt steeds een nieuwe SQL bron toegewezen, wat hierboven gebeurt. De eerste keer dat je het formulier opent, kan het voorkomen dat de query [qTemp] nog niet bestaat, en deze moet dan worden aangemaakt. Dat gebeurt in de subprocedure < MaakQuery>. Deze wordt uitgevoerd op het moment dat de procedure probeert om de SQL van het object qTemp te wijzigen.

Bestaat deze query niet, dan kan er uiteraard ook geen SQL aan worden toegewezen. De procedure levert dan een fout op. Deze fout wordt afgevangen door de regel "On Error GoTo MaakQuery" die er voor zorgt dat de procedure verder gaat in de subprocedure MaakQuery. Die ziet er zo uit:

```
MaakQuery:
    Set qTmp = CurrentDb.CreateQueryDef("qTemp", strSQL)
    Resume
```

De variabele qTmp (van het type QueryDef) wordt aangemaakt met de opdracht "CurrentDb.CreateQueryDef("qTemp", strSQL)". Deze opdracht kan alleen worden uitgevoerd op objecten die nog niet bestaan, maar dat was ook precies de situatie! Om weer verder te gaan met de oorspronkelijke procedure, eindigt dit deel met "Resume".

De variabele wordt dus ofwel aangemaakt, met de SQL uit de variabele strSQL, ofwel wordt de SQL van de Query qTemp veranderd met de regel qTmp.SQL = strSQL.

```
DoCmd.DeleteObject acTable, "Temp"
DoCmd.OpenQuery "qTemp_Maken"
With Me.lstSelection
    .RowSource = strSQL
    .Requery
End With
On Error Resume Next
```

In het volgend deel wordt de tijdelijke table [Temp] verwijderd, en wordt een vaste query (qTemp_Maken) uitgevoerd. Deze query gebruikt de query qTemp als basis. Dat is dus een query die steeds andere gegevens bevat, want daarvan hebben we net de SQL aangepast. Als laatste in dit deel wordt de rijbron van de Keuzelijst <lstSelection> ingesteld. Deze krijgt de query strSQL als Rowsource, en hij laat dus steeds een andere bron zien.

```
For iLoop = 65 To 90
    Me(Chr(iLoop)).Enabled = False
Next iLoop
```

In bovenstaande lus worden eerst alle tekstknoppen uitgeschakeld. Verderop in de procedure worden ze weer geactiveerd als er een overeenkomende beginletter in de bron zit. Omdat de knoppen een Letter als naam hebben, moet de lus gebruik maken van de functie CHR om de luswaarde te vertalen naar een letter (Me(Chr(iLoop))). Als je cijfers hebt gebruikt, hoeft dat niet: Me(iLoop).Enabled is dan voldoende.

```
With CurrentDb.OpenRecordset("Temp")
    For iVeld = 0 To .Fields.Count - 1
        If .Fields(iVeld).Name = "Beginletter" Then
            strSQL = "SELECT DISTINCT Beginletter FROM Temp"
            With CurrentDb.OpenRecordset(strSQL)
                If .RecordCount > 0 Then
                    .MoveFirst
                    Do While Not .EOF
                        On Error Resume Next
                        Me(.Fields(0)).Enabled = True
                        .MoveNext
                    Loop
                End If
```

```
                End With
                Exit For
            End If
        Next iVeld
    End With
```

Deze lus controleert eerst of het veld [Beginletter] wel in de bron zit. Dat is namelijk niet het geval als de keuzelijst de Postcodes laat zien. Door daar een check op te maken, voorkom je dat de procedure een fout produceert. Vervolgens wordt de tabel [Temp] geopend, en wordt elke knop die in de tabel zit geactiveerd.

In Access 2007/2010 kan de code wat korter, omdat je rechtstreeks naar de Fields collectie van qTmp kunt verwijzen. De code ziet er dan zo uit:

```
    For Each fld In qTmp.Fields
        If fld.Name = "Beginletter" Then
            strSQL = "SELECT DISTINCT Beginletter FROM qTemp"
            With CurrentDb.OpenRecordset(strSQL)
                If .RecordCount > 0 Then
                    .MoveFirst
                    Do While Not .EOF
                        On Error Resume Next
                        Me(.Fields(0)).Enabled = True
                        .MoveNext
                    Loop
                End If
            End With
            Exit For
        End If
    Next fld
```

Als laatste (niet echt noodzakelijke) onderdeel wordt in het tekstvak <txbResult> het aantal gevonden records gezet. Dit gebeurt op basis van de eigenschap ListCount, die kijkt hoeveel elementen de Keuzelijst bevat.

```
    If Me.lstSelection.ListCount <> 0 Then
        lngResult = Me.lstSelection.ListCount - 0
    Else
        lngResult = Me.lstSelection.ListCount
    End If
    Me.lblResult.Visible = True
    Me.txbResult.ValidationRule = True
    Me.fraSelect.Requery
    'Tonen resultaat in textbox
    Me.txbResult = lngResult
    Exit Function
End Function
```

De Keuzelijst

De keuzelijst triggert de de actie die de volgende tabel aan de keuzelijst hangt; de filterknoppen filteren dus alleen de waarden die je in de keuzelijst ziet. De actie vindt plaats op de gebeurtenis <Bij klikken> van de keuzelijst.

```
Private Sub lstSelection_Click()
    If Me.fraSelect.Value = 27 Then Me.txtNiveau.Value = 0
    If iNIV + 1 > 4 Then iNIV = 1 Else: iNIV = Nz(Me.txtNiveau, 0) + 1
    Me.txtNiveau.Value = iNIV
```

Als je op de knop <Alles> klikt, moet de lijst geschoond worden. Een klik op de knop levert als waarde 27 op (waarde van het groepsvak). De waarde txtNiveau wordt dan op 0 gezet. In andere gevallen

wordt de waarde txtNiveau steeds met 1 verhoogd, tot de waarde 4 is; dan wordt de waarde ingesteld op 1. Op die manier kun je door de verschillende opties lussen, en steeds opnieuw bij het hoofdniveau beginnen. Niet noodzakelijk, maar wel handig, is het om de labels op de knoppen te laten zien wat het formulier nu doet.

```
    Select Case iNIV
        Case 1
            Me.lblSoort.Caption = "Provincies"
            Me.cmdAlles.Caption = "Alle Provincies"
            Me.txtVan.Visible = False
            Me.txtVan = ""
            Me.txtTot.Visible = False
            Me.txtTot = ""
        Case 2
            Me.lblSoort.Caption = "Plaatsen - " & Me.lstSelection.Column(0)
            Me.cmdAlles.Caption = "Alle Plaatsen"
        Case 3
            Me.lblSoort.Caption = "Straten - " & Me.lstSelection.Column(0)
            Me.cmdAlles.Caption = "Alle Straten"
        Case 4
            Me.lblSoort.Caption = "Postcodes - " &
Me.lstSelection.Column(0)
            Me.cmdAlles.Caption = "Alle Postcodes"
    End Select
```

De Select Case wordt gebruikt om het bijschrift van het groepsvak aan te passen; dat krijgt een tekst die overeenkomt met de geselecteerde waarde. Die waarde wordt weer uit de eerste kolom van de listbox gehaald.

```
    Me.txtLijst = Me.lstSelection.Column(1)
    Call FormulierAanpassen(iNIV)
    Exit Sub
End Sub
```

Als laatste wordt het formulier weer ingesteld; er moet immers een nieuwe bron worden ingelezen, en de knoppen moeten uit- en ingeschakeld worden.

De extra functies

Er worden twee functies gebruikt om resp. een filter te maken (de functie KnoppenFilter) of om een de letterknoppen in te stellen (EersteLetter).

Als eerste de functie <KnoppenFilter>. Deze gebruikt de waarde die uit het Groepsvak <fraSelect> komt; elke letter daarin krijgt van Access een unieke waarde, en die wordt meegenomen naar de functie. Kijken we naar knop <A>, dan heeft die de waarde 1. Als we de functie aanroepen, wordt dus Case 1 uitgevoerd. Bij alle letters met diacrieten wordt een tekststring gemaakt met de voorkomende letters.

```
Function KnoppenFilter(KnopWaarde As Integer) As String
    On Error GoTo err_Sorteren
    Select Case KnopWaarde
        Case 1
            KnoppenFilter = "'[AÀÁÂÃÄÅÆ" & LCase("AÀÁÂÃÄÅÆ") & "]*'"
        Case 3
            KnoppenFilter = "'[CÇ" & LCase("CÇ") & "]*'"
        Case 5
            KnoppenFilter = "'[EÈÉÊË" & LCase("EÈÉÊË") & "]*'"
        Case 9
            KnoppenFilter = "'[IÌÍÎÏ" & LCase("IÌÍÎÏ") & "]*'"
        Case 14
```

```
            KnoppenFilter = "'[NÑ" & LCase("NÑ") & "]*'"
        Case 15
            KnoppenFilter = "'[OÒÓÔÕÖØ" & LCase("OÒÓÔÕÖØ") & "]*'"
        Case 19
            KnoppenFilter = "'[SŠ" & LCase("SŠ") & "]*'"
        Case 21
            KnoppenFilter = "'[UÙÚÛÜ" & LCase("UÙÚÛÜ") & "]*'"
        Case 25
            KnoppenFilter = "'[" & UCase("YÝÿ") & LCase("YÝÿ") & "]*'"
        Case 27
            KnoppenFilter = "'*'"
        Case 99
            KnoppenFilter = "'*'"
```

De functie levert dus een tekststring op die er zo uit kan zien: '[AÀÁÂÃÄÅÆaàáâãäåæ]*'
Deze string wordt dan weer in de query gebruikt met de parameter LIKE.
Alle overige knoppen worden op hun eigen letter gefilterd, en dat filter wordt in de Case Else opgebouwd. Hierbij weet de functie uiteraard niet op welke letter is geklikt. Vandaar dat er een letter wordt samengesteld op basis van de Letterknop waarde (van 1 tot en met 26) waarbij de getalwaarde 64 wordt opgeteld, gecombineerd met een waarde + 96. Hiermee worden zowel een hoofdletter als een kleine letter samengesteld.

```
        Case Else 'Alle records, dus geen beperking
            KnoppenFilter = "'[" & Chr(64 + KnopWaarde)  & Chr(96 +
KnopWaarde) & "]*'"
    End Select
    Exit Function
err_Sorteren:
    MsgBox Err.Description
    Exit Function
End Function
```

De functie <EersteLetter> doet zo ongeveer het omgekeerde. Omdat de letter Â een andere ASCII waarde heeft als A, maar uiteraard wel in de lijst onder de knop A moet zitten, worden alle diacrieten verzameld onder de bronletter. Deze letter wordt in de query gegenereerd op basis van de naam; dat kan een plaatsnaam zijn, een straatnaam etc. De beginletter zelf wordt uit de eerste letter van de naam gehaald, maar het kan zijn dat er een voorloopteken bij de naam zit, zoals bij 't Harde. In dat geval moet de functie een H genereren, en niet een enkel aanhalingsteken ('). Er zit dus eerst een check op dat voorloopteken.

```
Public Function EersteLetter(Waarde As String) As String
Dim Letter As String
    Letter = Left(Waarde, 1)
    If Asc(Letter) = 39 Then
        Letter = Mid(Waarde, InStr(Waarde, " ") + 1, 1)
    End If
    If Asc(Letter) = 39 Then
        Letter = Mid(Waarde, InStr(Waarde, "-") + 1, 1)
    End If
```

De ASCII waarde van een enkel aanhalingsteken is 39, dus daar is prima op te controleren. Ook al omdat een enkel aanhalingsteken in Access 2003 en Access 2007/2010 staat voor een commentaar regel, en je wilt natuurlijk niet dat de procedure daar op struikelt. Er zijn twee controles: op een spatie, en op een koppelteken. Daarmee kun je de meeste gevallen wel afvangen. Heb je er meer nodig, dan kun je dat uiteraard wel inbouwen.

```
    Select Case Letter
        Case "À", "Á", "Â", "Ã", "Ä", "Å", "Æ"
            EersteLetter = "A"
        Case "Ç"
            EersteLetter = "C"
```

```
        Case "È", "É", "Ê", "Ë"
            EersteLetter = "E"
        Case "Ì", "Í", "Î", "Ï"
            EersteLetter = "I"
        Case "Ñ"
            EersteLetter = "N"
        Case "Ò", "Ó", "Ô", "Õ", "Ö", "Ø"
            EersteLetter = "O"
        Case "Š"
            EersteLetter = "S"
        Case "Ù", "Ú", "Û", "Ü"
            EersteLetter = "U"
        Case "Ý", "ÿ"
            EersteLetter = "Y"
        Case Else
            EersteLetter = Letter
    End Select
End Function
```

De Select Case kijkt hier naar de letter in de variabele <Letter>, en bepaalt op basis daarvan welke letter er wordt geretourneerd in het veld BeginLetter in de query (zie code hierboven). En op basis van dat veld wordt dus uiteindelijk de knop geactiveerd of niet.

Als je alle code hebt gemaakt, kun je het formulier testen!

Samenvatting

In dit hoofdstuk heb je een formulier gemaakt waarmee je selecties kunt maken middels een keuzelijst. De lijst kun je met Alfabet knoppen filteren op basis van een beginletter. Voor het bepalen van de juiste beginletters zijn functies gemaakt, die in een query gebruikt kunnen worden. Door deze functies Public te maken, kun je ze ook rechtstreeks in een query gebruiken; ze zijn dus niet alleen bruikbaar binnen een formulier.

Het formulier is gekoppeld aan een vaste tijdelijke tabel, die steeds opnieuw gemaakt wordt met records uit andere tabellen. Door een vaste structuur te gebruiken voor die tijdelijke tabel, kun je andere queries en keuzelijsten baseren op die tijdelijke tabel.

Opdracht

Maak een multifunctioneel formulier op basis van de tabel st_Duiklocaties, waarin de lijst wordt gebaseerd op de cyclus <Duiklanden> à <Duiklocaties> à <Duikplaatsen>.

Zorg er ook voor dat de knoppen correct filteren op de beginletter.

Bestanden

Dit hoofdstuk maakt gebruik van de tabellen uit de Postcodetabel die je hier kunt downloaden.
In de vaste topic van het Access forum vind je de database met de Duikclub bestanden en een leeg formulier, en de uitwerking van dit hoofdstuk.