Access voor beginners - Hoofdstuk 23

Handleiding van Helpmij.nl

Auteur: OctaFish

Juni 2014

Inloggen op een database

In de vorige hoofdstukken over *Werken met Recordsets* hebben we gewerkt met formulieren die met behulp van VBA en Recordsets werden gekoppeld aan de gegevens. Een techniek die er voor kan zorgen dat de belasting op de database zo laag mogelijk wordt gehouden. In een eerder hoofdstuk heb ik behandeld hoe je een formulier kunt gebruiken om gebruikers al dan niet geforceerd uit de database te zetten, ook weer met het oogmerk van minimale belasting van de database.

In dit hoofdstuk gaan we ons bezighouden met een onderdeel waar regelmatig naaar gevraagd wordt in het Access forum, en daarom voldoe ik graag aan die vraag. Die luidt: “Hoe kun je de gebruikers toegang geven tot de database met verschillende rechten?” Op deze vraag zijn verschillende antwoorden mogelijk. In dit hoofdstuk gebruik ik de techniek om met behulp van een gebruikerstabel de rechten vast te leggen, en deze rechten uit te lezen als de gebruiker inlogt.

De gebruikerstabel

Laten we eens kijken naar hoe zo’n gebruikerstabel er uit ziet.

Veldnaam	Gegevenstype
UserID	Tekst
Name	Tekst
Access_level	Numeriek
Password	Tekst
Password_date	Datum/tijd

We gebruiken een aantal (redelijk logische) velden zoals [UserID], en [Name] om de gebruiker te identificeren, een veld [Access_level] voor het toegangsniveau en twee velden welke we gebruiken om het wachtwoord te controleren: [Password] en [Password_date]. Dat laatste veld is niet noodzakelijk, maar kun je gebruiken om de gebruiker te dwingen om regelmatig zijn/haar wachtwoord te wijzigen.

We gaan nu niet meer in op het maken van een formulier met keuzelijsten, want die kennis zou je ondertussen moeten hebben. We gaan dus meteen kijken naar de specifieke kenmerken van het formulier, zoals wat er gebeurt als het wachtwoord is verlopen.

Het Opstartformulier

De database moet natuurlijk gelijk openen met het inlogscherm, anders heeft het inlogscherm niet zoveel zin. Dus in de Eigenschappen van de database heb je ingesteld dat het formulier opstart met het formulier frm_Login. Dat ziet er dan zo uit:

Je ziet een keuzelijst met al ingevoerde gebruikers, en een tekstvak voor een wachtwoord. Dat wachtwoord is opgemaakt met de notatie *Password*, zodat je niet ziet wat er wordt ingevoerd. Wel zo veilig! Je kunt overigens afdwingen dat het wachtwoord hoofdlettergevoelig is of niet, een optie die we later nog aangeven.

Als het wachtwoord is ingetypt, klik je op de knop <Login> om de database te openen. Althans: als het wachtwoord correct is, en niet is verlopen. In het eerste geval kom je de DB uiteraard niet in, en in het tweede geval moet je het wachtwoord eerst veranderen voordat je verder kunt.

Laten we de code eens bekijken die het e.e.a. regelt!

```
Private Sub knOK_Click()
    ' ********************************************************
    ' Gebruiker heeft UserID geselecteerd en wachtwoord ingetypt.
    ' ********************************************************
     Call DisplayMenu(Me.Initials)
End Sub
```

Voor het inloggen gebruiken we een functie die we aanroepen, die hier *DisplayMenu* heet. We gebruiken hierbij de keuzelijst *Initials* waarmee we de Gebruikersnaam hebben geselecteerd. Deze UserID wordt dus meegenomen als parameter in de functie-aanroep.

Eén van de eerste handelingen die we doen, is het vullen van de Globale variabele sUserID. Deze wordt in de algemene sectie van de module gedeclareerd. De reden om dit zo te doen, is dat we de waarde van de UserId ook op andere momenten nodig hebben.

```
Public sUserID_ori As String
Public iPW_Count As Integer

Function DisplayMenu(UserId As String)
Dim PortSyd As String, CheckPassword As String
Dim strSQL As String, sFilter As String
Dim strMsg As String
Dim PasswordPeriod As Date
Dim CheckUser As Integer
Dim ValidUser As Integer
Dim AccessLevel As Integer

' ********************************************************************
' validate user_id
' ********************************************************************
strSQL = "SELECT [UserID],[password],[access_level],[password_date] " _
    & "FROM tbl_users " _
    & "WHERE UserID='" & UserId & "'"
With CurrentDb.OpenRecordset(strSQL)
    If .RecordCount = 1 Then
        If TempVars.Count > 0 Then TempVars.RemoveAll
        TempVars.Add "sUserID", .Fields(0).Value
        TempVars.Add "sPassword", .Fields(1).Value
        TempVars.Add "AccessLevel", .Fields(2).Value
        TempVars.Add "ExpireDate", .Fields(3).Value
        sUserID_ori = sUserID
        sUserID = .Fields(0).Value
        ValidUser = 2
        If .Fields("password") = Forms!Frm_Login!Password Then
            AccessLevel = .Fields("Access_Level").Value
            PasswordPeriod = CDate(.Fields("password_date").Value)
        Else
            If sUserID = sUserID_ori Then
                iPW_Count = iPW_Count + 1
            Else
                iPW_Count = 1
            End If
            ValidUser = 1
            AccessLevel = 3
        End If
    Else
        ValidUser = 0
    End If
    .Close
End With
```


We beginnen de functie uiteraard met het declareren van de noodzakelijke variabelen. Daarna maken we een SQL string die in de tabel tbl_Users kijkt of de ingevoerde gebruiker voorkomt. Dat is natuurlijk zo, want de keuzelijst waarmee we een gebruiker selecteren is gebaseerd op dezelfde tabel. Maar het gaat ons om de controle van het wachtwoord, en om de controle van de verloopdatum van dat wachtwoord.

Voor het geval er toch iets is fout gegaan met het invoeren (of meenemen met de functieaanroep) van de UserID, doen we een check op het vinden van het User record. Bij het vinden ervan, wordt de UserID eerst in een back-up variabele (sUserID_ori) gezet. Daarna geven we de variabele ValidUser de waarde 2 omdat we met een geldige gebruiker te maken hebben. Het veld [Password] wordt nu gecontroleerd a.d.h.v. wachtwoord dat op het formulier is ingevuld en als dat overeenkomt, wordt het toegangsniveau toegewezen aan de variabele AccessLevel. Tevens wordt de verloopdatum van het wachtwoord in de variabele PasswordPeriod gezet.

Als de gebruiker het wachtwoord niet correct heeft ingevoerd, wordt een teller opgehoogd. Deze teller wordt in het volgende deel gecontroleerd, want de gebruiker krijgt maar 3 pogingen om een wachtwoord correct in te voeren. Tevens wordt, bij verkeerd wachtwoord, de variabele ValidUser met de waarde 1 gevuld, en het Accesslevel ingesteld op het laagste niveau. Dat laatste is eigenlijk niet

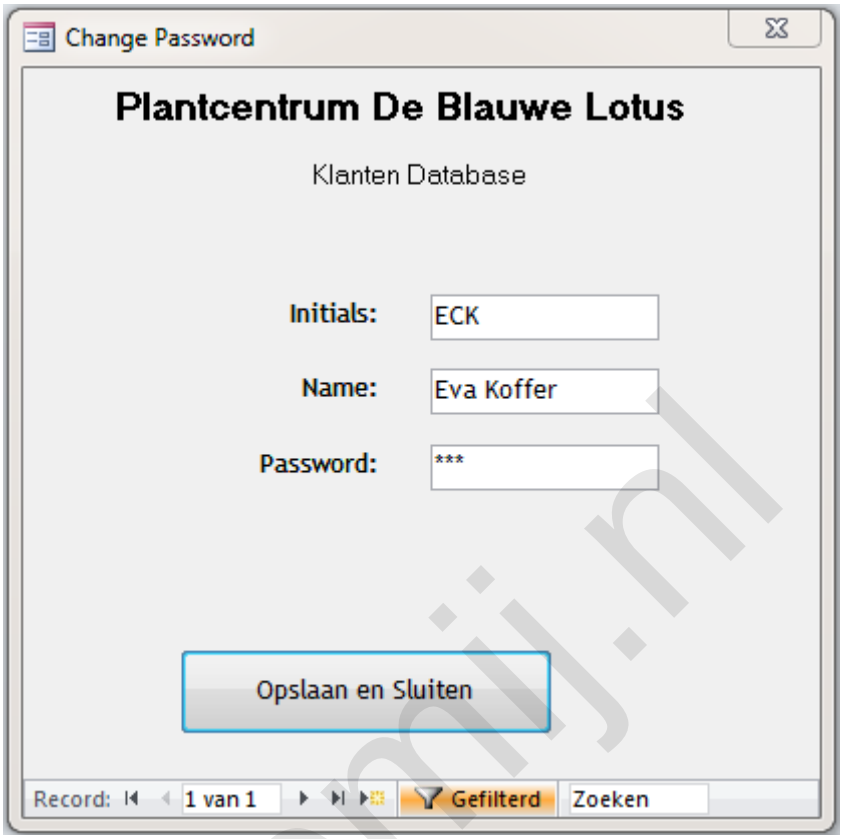
nodig, want de gebruiker komt toch niet veel verder dan het inlogscherm.

```
' ********************************************************************
' validate access_level
' ********************************************************************
Select Case ValidUser
    Case 0, 1
        If iPW_Count < 3 Then
            strMsg = "Geen toegang tot Database; " & vbCrLf _
                & "Je hebt een verkeerd wachtwoord ingetypt." & vbCrLf _
                & "Je hebt nog " & 3 - iPW_Count & " poging(en)."
            MsgBox strMsg, vbInformation, "ONGELDIGE  INLOGPOGING"
            Exit Function
        Else
            DoCmd.Quit
        End If
    Case 2
        ' validate password expiry
        If PasswordPeriod < Date - 30 Then
            strMsg = "Je wachtwoord is verlopen." _
                & "Je moet het wachtwoord eerst veranderen."
            MsgBox strMsg, vbInformation, "Verlopen wachtwoord"
            sFilter = "UserID='" & sUserID & "'"
            DoCmd.OpenForm "Frm_change_password", acNormal, , sFilter
        Else
            DoCmd.Close acForm, "Frm_Login"
            DoCmd.OpenForm "Frm_switchboard"
        End If
    Case Else
        strMsg = "Geen toegang." & vbCrLf _
            & "Neem contact op met de Administrator."
        MsgBox strMsg, vbInformation, "Onbekende Gebruiker of Wachtwoord"
End Select

Exit_Display_Menu:
    Exit Function

Err_Display_Menu:
    MsgBox Err.Description
    Resume Exit_Display_Menu

End Function
```

Dat blijkt uit het vervolg van de code, waarin de variabele *ValidUser* wordt gecontroleerd. Bij de waarden 0 en 1 (geen geldige user ingevoerd, of geen geldig wachtwoord) krijg je een Msgbox te zien die met een teller op het scherm bijhoudt hoeveel pogingen je nog hebt om het wachtwoord correct in te voeren.

Als het wachtwoord wél correct is, wordt de datum gecontroleerd. In het voorbeeld mag het wachtwoord maximaal 30 dagen verlopen zijn. Je kunt daar natuurlijk elke waarde voor gebruiken die

voor je eigen situatie van toepassing is.



We controleren natuurlijk of de gebruiker wel een nieuw wachtwoord invoert. Doet hij/zij dat niet, dan verschijnt deze melding:

En dat blijft het systeem doen tot er een correct nieuw wachtwoord is ingevoerd. In hoeverre het nieuwe wachtwoord moet afwijken van het oude, kun je uiteraard ook regelen. Maar daar is dit hoofdstuk niet voor bedoeld, dus dat laat ik nog even rusten. Vooralsnog is het voldoende als het nieuwe wachtwoord *niet* hetzelfde is als het oude. Eén van de manieren om dit af te dwingen is om het *oude* wachtwoord op te slaan, en deze te vergelijken met het *nieuwe* wachtwoord.

Laten we eens kijken naar de code die we gebruiken in het formulier.

```
Option Compare Binary
Option Explicit
Const Max_Chars = 10

Private Sub Form_Load()
    DoCmd.Close acForm, "Frm_Login"
    Me.Password_ori = Me!Password
    Me.password_date = Date
End Sub

Private Sub Password_Change()
On Error GoTo Err_Password_Change
Dim strMsg As String

With Me.Password
    If Len(.Text) > Max_Chars Then
        strMsg = "Wachtwoord is langer dan " _
            & Max_Chars & Space(20) & vbCrLf _
            & " Graag aanpassen en opnieuw invoeren."
        MsgBox strMsg, vbCritical, "Password length"
        .Text = Left(.Text, Max_Chars)
        .SelStart = Max_Chars
    End If
End With
Exit Sub

Err_Password_Change:
    MsgBox Err.Description
End Sub
```

We beginnen met het declareren van de *Compare* parameter. Standaard staat die in Access op *Database*, wat prima is voor normaal gebruik. Bij het controleren van wachtwoorden wil je echter onderscheid kunnen maken tussen hoofdletters en kleine letters, en dat kan alleen als je gebruik maakt van de optie *Binary.* Hiermee wordt de ingevoerde tekst op ASCII waarde vergeleken volgens de reeks A < Z < a < z < À < Ê < Ø < à < ê < ø. Daarmee is het wachtwoord DitIsEenTest dus niet hetzelfde als Ditiseentest. En dat wil je natuurlijk ook als je met wachtwoorden werkt. Want daarmee verdubbel je het aantal tekens dat een gebruiker kan invoeren.
Je ziet ook een constante Max_Chars gedefinieerd. Deze wordt gebruikt om te controleren of het wachtwoord niet langer is dan 10 tekens.

Bij het laden van het formulier (procedure Form_Load) wordt het Login formulier gesloten (wat ook op dat formulier geregeld kan worden trouwens) en wordt het oude wachtwoord in een tekstvak op het formulier gezet. Daar zijn ook andere mogelijkheden voor (Variabele of TempVar bijvoorbeeld) maar om de code uit te proberen kun je het beste een tekstvak gebruiken, omdat je dan ziet wat er gebeurt. Als alles werkt, maak je het tekstvak uiteraard onzichtbaar. Al zijn er redenen te bedenken om het oude wachtwoord wél te laten zien. Zoals: de gebruiker heeft zijn/haar wachtwoord al correct moeten invoeren om überhaupt op het formulier te komen, dus die check is al gedaan: het ingevoerde wachtwoord was dus correct. En we vullen de datum met de huidige datum.

Checken van nieuw wachtwoord

Het nieuwe wachtwoord wordt ‘live’ tijdens het intypen gecontroleerd. We controleren dus op de *lengte van het wachtwoord*. Als er teveel tekens worden getypt, worden de laatste tekens verwijderd en

blijven alleen de 10 toegestane tekens over.

Als de gebruiker klaar is met het invoeren van het wachtwoord, en op de Tab toets drukt, wordt de knop OK geactiveerd. De gebruiker kan natuurlijk ook een muishandeling gebruiken om een ander object te selecteren voordat hij/zij klaar is; in dat geval wordt de gebeurtenis LostFocus getriggerd die het invoerproces afbreekt (CancelEvent) en het wachtwoord object opnieuw selecteert, zodat de gebruiker alsnog verder kan gaan met het invoeren van het wachtwoord.

```
Private Sub Password_LostFocus()
On Error GoTo Err_Password_LostFocus
Dim strMsg As String

    With DoCmd
        .CancelEvent
        .GoToControl "Password"
    End With
''End If

Exit_Password_LostFocus:
    Exit Sub

Err_Password_LostFocus:
    MsgBox Err.Description
    Resume Exit_Password_LostFocus
End Sub
```

```
Private Sub knSaveExit_Click()
Dim strMsg As String
Dim access_level As Integer

If Me.Password_ori = Me.Password Then
    strMsg = "Wachtwoord hetzelfde als het oude wachtwoord." _
        & vbCrLf & " Graag aanpassen en opnieuw invoeren."
    MsgBox strMsg, vbCritical, "Incorrect Password"
Else
    If [password_date] <= Date Then
        DoCmd.OpenForm "Frm_switchboard"
        DoCmd.Close acForm, "Frm_Change_Password"
    Else
        MsgBox ("Your Password must be changed NOW ")
    End If
End If

End Sub
```


Als de gebruiker op de knop <Opslaan en Sluiten> heeft geklikt, volgt de check of het nieuwe wachtwoord voldoet aan de regels (in het voorbeeld: het mag niet gelijk zijn aan het oorspronkelijke wachtwoord) en daarna wordt het formulier gesloten, en het formulier frm_Switchboard geopend. Op dit formulier worden de ingestelde rechten gecontroleerd.

Formulier opstarten op basis van rechten

Op de formulieren die je opent controleer je eerst welke rechten een gebruiker heeft. En op basis daarvan bepaal je wat die gebruiker mag doen, en mag zien. Je kunt bijvoorbeeld knoppen zichtbaar of onzichtbaar maken, en/of bewerkingsrechten toewijzen. In het voorbeeld word op het startmenu een aantal knoppen getoond op basis van de waarde in de gebruikerstabel.

We gebruiken in beginsel dezelfde werkwijze als op het hoofdformulier, met dien verstande dat we nu gebruik maken van de Public variabele sUserID, die we op het startformulier hebben gevuld met de procedure *DisplayMenu*.

```
Private Sub Form_Load()
ValidUser = 2
strSQL = "SELECT [access_level],[UserID] FROM tbl_users " _
    & "WHERE [UserID]='" & TempVars!sUserID & "'"

With CurrentDb.OpenRecordset(strSQL)
    If .RecordCount = 1 Then
        ValidUser = 2
        AccessLevel = .Fields("Access_Level").Value
    Else
        ValidUser = 1
        AccessLevel = 3
    End If
    .Close
End With

Select Case ValidUser
    Case 0, 1
        strMsg = " Toegang geweigerd" & vbCrLf _
            & "Neem contact op met de beheerder " _
            * "als het probleem zich herhaalt."
        MsgBox strMsg, vbInformation, _
            "Niet-geautoriseerde gebruiker"
        DoCmd.Close acForm, Me.Form.Name
    Case 2
        Select Case AccessLevel
            Case 1 ' lv1 menu
                Me.cmdOverzicht.Visible = True
                Me.cmdMagazijn.Visible = True
                Me.cmdRapporten.Visible = True
                Me.cmdDatabaseWindow.Visible = True
                Me.AllowAdditions = True
                Me.AllowEdits = True
            Case 2 ' lv2 menu
                Me.cmdOverzicht.Visible = True
                Me.cmdMagazijn.Visible = True
                Me.cmdRapporten.Visible = True
                Me.cmdDatabaseWindow.Visible = False
            Case 3 ' lv3 menu
                Me.cmdOverzicht.Visible = True
                Me.cmdMagazijn.Visible = False
                Me.cmdRapporten.Visible = False
                Me.cmdDatabaseWindow.Visible = False
            Case Else
                strMsg = " Toegang geweigerd" & vbCrLf _
            & "Neem contact op met de beheerder " _
            * "als het probleem zich herhaalt."
                MsgBox strMsg, vbInformation, _
                    "Niet-geautoriseerde gebruiker"
        End Select
    Case Else
End Select

End Sub
```

Vervolgens zie je dat we op basis van het uitgelezen AccessLevel bepalen welke elementen op het formulier zichtbaar mogen zijn, en welke niet.

Dit is een relatief simpel voorbeeld; je kunt dat simpel uitbreiden met opdrachten als Me.AllowAdditions = True/False en Me.AllowEdits = True/False.

De Tempvars collectie

In het voorbeeld wordt gebruik gemaakt van een Globale variabele die in een aparte module is gedefinieerd. Door deze aanpak heeft de variabele de grootste scope binnen de database, maar daardoor ook de grootste kwetsbaarheid. De waarde ervan blijft namelijk intact zolang de DB goed blijft werken. Als een formulier echter verkeerd wordt afgesloten, of er op een andere manier een fout in de DB optreedt, bestaat de kans dat de variabele zijn inhoud verliest. En dan werkt de procedure die de waarde nodig heeft natuurlijk niet goed meer.

Om dat gevaar te ondervangen, kun je de waarde op andere manieren vastleggen. Een optie is bijvoorbeeld om daar een aparte tabel voor te gebruiken die je steeds opnieuw vult. Of je gebruikt, als je met Access vanaf versie 2007 werkt, de TempVars collectie. Die behoudt namelijk wél zijn waarde als de DB crasht. Dus als je de DB of een formulier na een crash weer opent, heeft de Tempvar zijn waarde gewoon behouden, en is hij nog steeds bruikbaar.

Om de Tempvars collectie te gebruiken, passen we de opstart procedure dus enigszins aan.

```
strSQL = "SELECT [UserID],[password],[access_level],[password_date] " _
    & "FROM tbl_users " _
    & "WHERE UserID='" & UserId & "'"
With CurrentDb.OpenRecordset(strSQL)
    If .RecordCount = 1 Then
        If TempVars.Count > 0 Then TempVars.RemoveAll
        TempVars.Add "sUserID", .Fields(0).Value
        TempVars.Add "sPassword", .Fields(1).Value
        TempVars.Add "AccessLevel", .Fields(2).Value
        TempVars.Add "ExpireDate", .Fields(3).Value
        sUserID_ori = TempVars!sUserID
        ValidUser = 2
```

We hebben eerst uiteraard weer de tabel tbl_Users geopend om de gebruiker gegevens op te halen. Vervolgens controleren we weer of de gebruiker bestaat, en daarna gaan we met de TempVars aan de slag. Eerst wordt de collectie verwijderd met RemoveAll. Deze techniek kan riskant zijn als je er niet zeker van bent dat er nog andere TempVars bestaan die moeten blijven. In dat geval kun je de TempVars beter verwijderen met TempVars.Remove ("sUserID") en dit dan ook voor de overige TempVars die je opnieuw definieert.

Vervolgens worden de TempVars aangemaakt met Add, en gevuld met een waarde. In het voorbeeld komen die dus uit de Recordset, maar je kunt een TempVar overal mee vullen, net als gewone variabelen. Je hoeft overigens geen variabele type aan te geven; TempVars zijn altijd van het type Variant.

Nu de TempVars zijn gedefinieerd en gevuld, kun je ze gebruiken wanneer je ze nodig hebt. Het aardige is, dat je ze, behalve in je formulieren met VBA, ook in queries kunt gebruiken. Je kunt er dus mee werken alsof het een gewoon veld is. Die toepassing hebben we hier niet nodig, we gebruiken ze als normale variabelen.

Bij het laden van de formulieren kunnen we de TempVar variabele nu dus gebruiken in onze SELECT statement. Dat ziet er dan zo uit:

```
Private Sub Form_Load()
ValidUser = 2
strSQL = "SELECT [access_level],[UserID] FROM tbl_users " _
    & "WHERE [UserID]='" & TempVars!sUserID & "'"
```

Hierbij wordt de variabele sUserID rechtstreeks uit de collectie TempVars gehaald. Je kunt een

variabele op nog 2 andere manieren aanroepen, namelijk op Indexnummer en op Indexnaam. Dat ziet er dan zo uit: TempVars.Item("sUserID") en TempVars.Item(0). Alle varianten leveren uiteraard hetzelfde resultaat op.

Extra mogelijkheden

Wil je het écht uitgebreid hebben, dan kun je een aparte tabel maken voor de rechten en instellingen, en die uitlezen bij het openen van een formulier. Je kunt dan denken aan Kleurschema's die de gebruiker zelf kan instellen, en vertalingen van het formulier. Zo kan in België een werknemer dan kiezen uit Franstalige formulieren of Nederlandstalig. Uiteraard geldt hier: hoe meer mogelijkheden je aanbiedt, hoe meer je moet inrichten. De techniek echter blijft steeds hetzelfde: je leest de rechten van de gebruiker uit, en bepaalt a.d.h. daarvan wat er moet gebeuren met het formulier.

Wil je van deze mogelijkheid gebruik maken, dan is het zinvol om eerst na te denken over de inrichting van de tabel waarin je de aanpasbare instellingen wilt opslaan. Voor kleurschema's en vaste lettertypes kun je denken aan een aparte tabel die je koppelt aan de Gebruikerstabel, want je maakt voor elke aanpassing een apart record aan. En je wilt natuurlijk dat die is gekoppeld aan de juiste gebruiker. De tabel bevat dan alle eigenschappen die generiek zijn voor alle objecten in Access die op dezelfde manier aangepast worden.

Wil je aanpassingen opslaan op Formulierniveau, zoals vertalingen van labels etc., dan moet je ook de formulierobjecten apart opslaan. Je kunt daar een aparte tabel voor maken, waarin je dan weer de UserID opneemt, de formuliernaam en de objecten die je wilt kunnen aanpassen. Als een label in 4 talen moet kunnen worden getoond, dan maak je dus 4 velden voor Frans, Nederlands, Engels en Duits, een veld UserID, een veld FormulierNaam en een veld ObjectNaam. In de tabel tbl_Users staat dan welke taal ze willen zien. Bij het inlezen van het formulier lees je de daarbij behorende taalelementen uit, en koppel je ze aan de formulierobjecten met dezelfde naam.

Als je de gebruiker zélf elementen wilt laten aanpassen, zoals kleurschema's, dan moet je de gebruiker uiteraard de optie geven om de wijzigingen op te slaan.

Samenvatting

In dit hoofdstuk hebben we gezien hoe we met behulp van een tabel Gebruikersgegevens kunnen opslaan, en deze gegevens gebruiken om op formulieren rechten in te stellen. De mogelijkheden die deze aanpak biedt zijn heel uitgebreid; je kunt de gebruiker een eigen taal aanbieden voor je formulieren, eigen kleurenschema's laten samenstellen etc. Wél is het belangrijk dat je de in te stellen opties goed kenbaar maakt aan de gebruiker, en goed vastlegt.