VBA voor Doe het Zelvers deel 3

Handleiding van Helpmij.nl

Auteur: leofact

Februari 2014

VBA voor Doe het Zelvers is een reeks artikelen, bedoelt voor mensen die met VBA in Excel aan de slag willen om taken te automatiseren, of om deze toegankelijk te maken voor gebruikers met weinig Excel kennis. VBA is een volwaardige programmeertaal, met een woordenschat en een zinsopbouw (syntaxis). Om deze goed te leren wordt het aangeraden om hierover boeken te lezen. Bijvoorbeeld uit deze willekeurige selectie.

Vorige aflevering

In de vorige aflevering is uitgelegd wat een procedure en een Function inhouden en waarin ze verschillen. Verder werd de opbouw van VBA als Object Model uitgelegd en werden Methodes en Eigenschappen uitgelegd. De invoer van code m.b.v. van IntelliSense werd besproken en als laatste werden de verschillende manieren om een Macro te starten besproken.

In deze aflevering

Als eerste maken we pas op de plaats en noem ik sites met achtergrondinformatie over VBA. Ook wordt MZ-Tools geïntroduceerd. Met MZ-Tools wordt het makkelijker om VBA code in te voeren. Verder wordt in deze aflevering uitgelegd wat variabelen zijn, wat je er mee kunt en hoe je ze gebruikt. Als laatste wordt het werken met cellen en bereiken besproken. Er is een voorbeeld bestand als bijlage toegevoegd, waarin het werken met bereiken zichtbaar wordt gemaakt.

Informatie op internet

Er is op internet veel te vinden over VBA. Een kleine selectie van sites die voor mij handig zijn gebleken, vind je hieronder:

http://www.snb-vba.eu SNB is een actieve helper op Helpmij. Hij houdt een uitgebreide site bij waar veel uitleg is te vinden van VBA. Bibliotheken, Objecten, Eigenschappen en Methoden komen uitgebreid aan de orde. (Nederlands en Engels)

http://msdn.microsoft.com/library/office De ontwikkelaars-site van Microsoft zelf (Engels).

www.rondebruin.nl Bevat veel code voorbeelden (Engels).

www.ozgrid.com/VBA/ Nog meer voorbeelden (Engels).

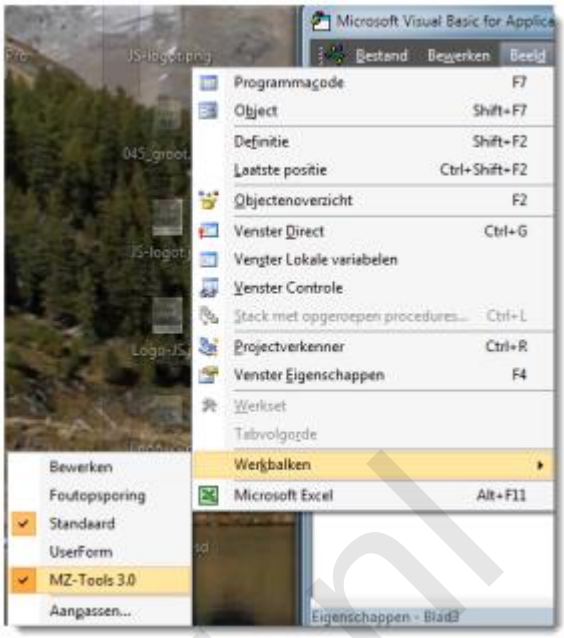
wikipedia.org/wiki/Lijst_van_Microsoft_Excel-functies Alle Excelfuncties en de Engelse vertaling daarvan.

Dit overzicht is verre van compleet. Google daarom vooral zelf naar de sites die je aanspreken.

MZ-Tools

Dit is een handige en gratis uitbreiding van de Macro Editor. MZ-Tools biedt een scala aan handige gereedschappen en assistenten die het op allerlei manieren makkelijker maken om met code om te gaan.

1. Download MZ-Tools: ** klik hier **.
   (kies de “for VBA” versie)
2. Voer de installer uit, door op het gedownloade bestand te klikken. Hierbij dient Excel gesloten te zijn.
3. Open Excel en vervolgens de VBA Editor (Alt + F11)
   Mogelijk is MZ-Tools werkbalk nog niet zichtbaar. Ga dan naar Beeld > Werkbalken > MZ – Tools 3.0 en selecteer deze.



Er wordt nu een werkbalk zichtbaar met knoppen voor de diverse gereedschappen. Zie hieronder voor een gedeeltelijke selectie daarvan:

a. Kopieert en plakt objecten met de bijbehorende code.
b. Activeert de zoek-functie welke handige extra’s bevat.
c. Opent een nieuwe procedure.
d. Plaatst een Module of Procedure Header. Ook een Error Handler (fout afhandeling procedure) kan hiermee worden ingevoegd.
e. Zet regel nummers aan of uit.
f. Breekt regels af op leesbare breedte, of geeft ze weer in de volle breedte.
g. Opent een menu met extra Tools, o.a. om een Message Box op te bouwen.
h. Opent het Option Menu, waarin alle functies naar eigen wens kunnen worden aangepast.

Variabelen

Variabelen maken het mogelijk eenzelfde berekening te maken met verschillende getallen of gegevens. Ze zijn bekend vanuit de wis- en meetkunde. Een bekend voorbeeld daarvan is de stelling van Pythagoras (berekent rechthoekige driehoeken): $A^2 + B^2 = C^2$. In getallen $3^2(=9) + 4^2(=16) = 25$. C is dus de wortel van 25 = 5. Alle getallen A en B geven uitkomst C.
In VBA zijn er verschillende soorten variabelen:

- *Boolean*: omvat alleen de waarden; waar of onwaar.
- *Byte*: een geheel getal van 0 tot 255 (één byte groot)
- *Integer*: voor gehele getallen van -32768 tot +32767
- *Long*: voor getallen van -2.147.483.648 tot 2.147.483.647
- *Single*: voor getallen van -3.402823E38 tot -1.401298E-45 en positieve getallen van 1.401298E-45 tot 3.402823E38. (getallen met bewegende komma)
- *Double*: voor negatieve getallen van -1,79769313486232E308 tot -4,94065645841247E-324 en positieve getallen van 4,94065645841247E-324 tot 1,79769313486232E308 (zeer grote getallen met bewegende komma)

- *Currency:* speciaal voor getallen met decimaal punten (bedragen) tussen -9223372036854770,5808 en 9223372036854770,5808.
- *String*: voor tekst.
- *Date*: voor datum en tijd waarden.
- *Object*: bevat een object. Bijvoorbeeld een werkblad
- *Variant*: dit datatype kan alle overige datatypes bevatten.

De verschillende types zijn van belang, omdat zij aangeven wat voor data zij bevatten. Bovendien is het geheugengebruik van de verschillende types verschillend. Zo is er voor Byte slecht één byte nodig, voor een Single 4 bytes en voor een Double 8 bytes. Om het gegevenstype van een variabele in te stellen dient deze gedeclareerd of gedimensioneerd te worden:

```
Dim lTeller as Long;
```

Dit zorgt ervoor dat lTeller alleen Long getallen kan bevatten. Variabelen kunnen stuk voor stuk onder elkaar of achter elkaar in één regel worden gedeclareerd:

```
Dim lTeller as Long, bSwitch as Boolean, dDatum as Date
```

Er wordt ook wel als volgt gedeclareerd:

```
Dim lTeller, lHoogte, lBreedte as Long
```

Excel slaat dan geen alarm. De variabelen worden echter dan als Variant ingesteld en dat kan, buiten onnodig geheugen gebruik, onbedoelde effecten opleveren; een Long variabele blijkt dan bijvoorbeeld onbedoeld een Variant. Er volgt dan ook geen foutmelding als daar abusievelijk tekst aan wordt toegewezen.

Het is een goede gewoonte om variabelen een naam te geven die op de inhoud slaat. Geef in de naam ook het gegevenstype op. Voor naamgeving zijn er eenduidige afspraken gemaakt. Hierover kun je meer lezen in de volgende (Engelse) link: http://goo.gl/tqlqC1
In deze reeks gebruik ik alleen de eerste letter van het gegevenstype.

Over het algemeen zal een procedure ook werken als de variabelen niet gedeclareerd zijn. Het is dan niet duidelijk welk gegevenstype de variabele bevat en het geheugengebruik zal onnodig groot zijn. Option Explicit zorgt ervoor dat Excel een foutmelding (Variabele is niet gedefinieerd) genereert wanneer je vergeet om de variabele te declareren.

Tip: de Integer was in het verleden nuttig omdat hij weinig geheugen nodig had. In de praktijk wordt de Integer tegenwoordig door de compiler in een Long omgezet en kan er dus beter de Long worden gebruikt. (informatie)

Om een variabele te gebruiken dient deze te worden geïnitialiseerd. D.w.z. er moet een waarde worden toegewezen. Dat gaat als volgt:

```
LTeller =10
```

En bij een Object:

```
Set oBlad = ActiveSheet
```

De variabele is op de volgende manier weer vrij te geven:

```
Set oBlad = Nothing
```

Dit geeft de gebruikte geheugenplaats weer vrij.

Array

Een Array is een variabele met één of meer dimensies, die te zien is als een reeks variabelen, die logisch bij elkaar horen. Bijvoorbeeld de elftallen van een voetbalclub en de spelers daarvan. Stel de club heeft 8 elftallen met 11 spelers elk, dan zou het Array als volgt opgebouwd kunnen worden.

```
Dim sSpelers(8, 11) As String
```

Een Array lijkt op een tabel. Eén van de voordelen van het gebruik van een Array is dat bewerkingen vele malen sneller worden uitgevoerd.

Constants

Constants (constanten) onderscheiden zich van gewone variabelen, doordat zij vaste waardes bevatten. Een Constant wordt als volgt gedeclareerd:

```
Const sClub As String = "AZ"
```

De inhoud van een Constant kan niet meer worden gewijzigd na de declaratie. VBA gebruikt ook Constants om waarden te vervangen. Deze worden vooraf gegaan door vb. vbCr staat bijvoorbeeld voor een hard Return Chr (13)

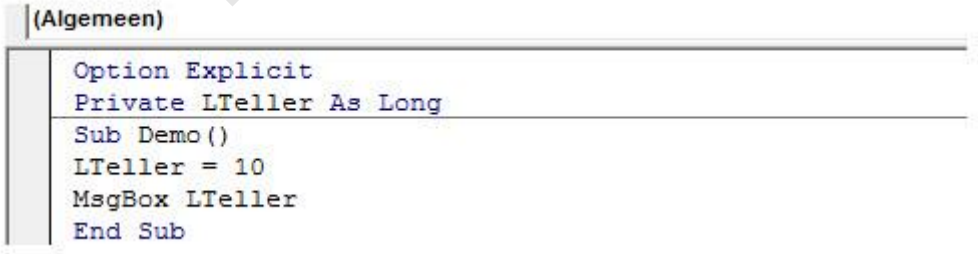
```
MsgBox "Dit is een zin." & Chr(13) & "met een tweede regel"

MsgBox "Dit is een zin." & vbCr & "met een tweede regel"
```

Deze twee regels geven dan ook exact hetzelfde resultaat weer. In de help functie zijn alle te gebruiken Constants te vinden.

Levensduur

Variabelen hebben een levensduur of *Scope*. Wanneer een variabele in een Procedure of Function wordt gedimensioneerd met Dim, blijft deze alleen “bestaan” in de betreffende procedure. Daarbuiten wordt deze niet meer herkend en heeft geen waarde meer. Soms wil je juist dat de variabele ook buiten de procedure blijft bestaan. Dat kan op Module niveau, door de variabele boven de bovenste Procedure te declareren. Gebruik hiervoor Private. Zo wordt de Module variabele duidelijk gescheiden van de Procedure variabele, waarvoor Dim wordt gebruikt.



De waarde van deze variabele kan nu worden doorgegeven naar andere Procedures of Functions in dezelfde Module.

Een Variabele kan ook openbaar (Global) worden gemaakt. De variabele blijft bestaan in alle Modules van het actieve werkboek. Deze kan boven elke willekeurige module worden gedeclareerd d.m.v. Public:

```
(Algemeen)
Option Explicit
Public LTeller As Long
Sub Demo()
LTeller = 10
MsgBox LTeller
End Sub
```

Een andere methode is om variabelen door te geven bij aanroepen van de Procedure of Function:

```
(Algemeen)
Option Explicit
Sub Demo()
Dim lTeller As Long
lTeller = 10
Ontvanger (lTeller)
End Sub
'Dit kan een andere module zijn
Sub Ontvanger(ByVal lTeller As Long)
MsgBox lTeller
End Sub
```

Procedures of Functions aanroepen vanuit een procedure wordt in een latere aflevering verder behandeld.

Cellen en bereiken

VBA kent verschillen manieren om bereiken te definiëren. Een bereik kan variëren van één enkele cel tot een complex van verschillende samengestelde bereiken, die al of niet kunnen overlappen. Het gedefinieerde bereik vormt vervolgens een object waar bewerkingen op uitgevoerd kunnen worden. Een greep uit de mogelijkheden om een bereik te definiëren:

- *Cells (rij, kolom)*
  Deze opdracht verwijst naar één enkele cel met de gespecificeerde rij en kolom nummer.
- *ActiveCell*
  Dit geeft de geselecteerde cel als resultaat.
- *Range (eerste cel, laatste cel)*
  Geeft een bereik van een blok aaneengesloten cellen als resultaat. Er kunnen meerdere blokken worden gedefinieerd.
- *Offset (Rij stappen, Kolom stappen)*
  Heeft als resultaat een relatieve verwijzing. Dit wordt opgegeven in aantal cellen horizontaal en aantal cellen verticaal. Dit kunnen negatieve getallen zijn.
- *Union (bereik 1, bereik 2, bereik 3...)*
  Dit geeft een samenvoeging van het gespecificeerde bereiken.
- *Intersect (bereik 1, bereik 2, bereik 3...)*
  Hiermee wordt het overlappende deel in de vorm van een blok cellen als resultaat gegeven.
- *UsedRange*
  Dit geeft als resultaat het in gebruik zijnde bereik.
- *Application.Goto (bereik, scroll)*
  Hiermee wordt het genoemde bereik geactiveerd en geselecteerd (ook als het werkblad niet actief is).

Standaard is het resultaat een bereik op het actieve werkblad. Dat is lang niet altijd gewenst. In dat

geval moet het gewenste werkblad (object) worden opgegeven.

Worksheets("Demo1").Cells(1,1).Value, of kort: *Demo1.Cells(1,1)*

Tip: Het is niet nodig om een cel te selecteren of te activeren, om daar een bewerking op uit te laten voeren door VBA. De procedure zal veel sneller lopen, wanneer dit achterwege wordt gelaten.

Range

Er zijn verschillende manieren en notatiewijzen om een bereik met Range te definiëren. Een korte greep:

- de standaard manier:
  ActiveSheet.Range("A1:D4").Select
- meerdere bereiken:
  ActiveSheet.Range("A1:D4" , "F2:G6" , "K2:P9").Select
- met een variabele:
  ActiveSheet.Range("A1:D" & lRij).Select
- met gebruik van Cell:
  ActiveSheet.Range("A1", Cells(12,12)).Select
- met een benoemd bereik:
  ActiveSheet.Range("naam").Select

Er is ook een verkorte schrijfwijze mogelijk. Dit geeft kortere code en dus minder type werk en (theoretisch) minder compileertijd. Voor deze reeks gebruik ik zelf de uitgebreide schrijfwijze van Range. Voorbeelden van de verkorte notatie:

[A2]
[A5:H6]
[A1:D4, F4:K8, L9:Q12]
[naam]

Bijzondere cellen.

VBA kent een set speciale celtypes, die handig kunnen zijn bij het opgeven van een bereik. Dit zijn de SpecialCells. Een paar voorbeelden:

- *xlCellTypeBlanks*: alle lege cellen
- *xlCellTypeFormulas*: alle cellen die formules bevatten
- *xlCellTypeLastCell*: de laatste cel in het gebruikte bereik
- *xlCellTypeSameFormatConditions*: cellen met dezelfde opmaak
- xlCellTypeVisible: alle zichtbare cellen

Met *xlCellTypeLastCell* kan bijvoorbeeld de laatste gebruikte cel worden gevonden. Excel houdt echter niet altijd correct bij wat de laatst cel is, van het in gebruik zijnde bereik.

Op de volgende wijze is het rijnummer van de laatste cel meer betrouwbaar te achterhalen:

Range("A" & Rows.Count).End(xlUp).Row

Bij het onderdeel over Range hoort een voorbeeldwerkboek, dat hier is te downloaden.

Het werkboek is een .xls, zodat het ook in Excel 2000 of later werkt. In de eerste aflevering van VBA voor Doe het Zelvers kun je terug vinden hoe een VBA werkboek in gebruik moet worden genomen.

Dit was het voor deze keer.

In deze aflevering werden de variabelen besproken, welke soorten er zijn, welke levensduur zij hebben en hoe ze gedeclareerd dienen te worden. Verder is de Array besproken en werd uitgelegd wat Constants zijn. Daarnaast werd het werken met bereiken besproken met de diverse mogelijkheden om naar bereiken te verwijzen. Het gebruik van Range werd daar uitgelicht en met diverse notatiewijzen besproken. Daarbij zijn tips gegeven die procedures kunnen versnellen en die bepaalde selecties kunnen vereenvoudigen.

Volgende keer wordt het werken met teksten besproken, aan de hand van de bouw van een dialoogvenster welke tot doel heeft de invoer van tekst makkelijker te maken.