VBA voor Doe het Zelvers deel 15

Handleiding van Helpmij.nl

Auteur: leofact

Februari 2015

Vorige aflevering

In deel 14 werd de Microsoft Office beveiligingsupdate van 9 december 2014 besproken. Deze update maakte in een aantal gevallen alle ActiveX-objecten onbruikbaar. Er werden verschillende methodes beschreven om toch weer met ActiveX-objecten te kunnen werken. Daarnaast werd het automatiseren van Word vanuit Excel uitgelegd met behulp van Early en Late Binding. Dit aan de hand van een voorbeeldprocedure waarin een tabel met factuurgegevens werd aangemaakt.

In deze aflevering

Dit keer wordt het zoeken naar gegevens in een werkboek besproken. In VBA is daar de Methode *Find* voor beschikbaar. Deze opdracht kan worden herhaald met *FindNext* (volgende) en *FindPrevious* (vorige). Met de methode *Replace* kunnen de zoekresultaten naar wens worden vervangen. Er kan ook met behulp van één van de werkbladfuncties worden gezocht. De belangrijkste worden genoemd. Daarvan wordt *Match* meer uitgebreid besproken. Daarbij wordt er ook aandacht geschonken aan de effecten op de performance. Om deze te kunnen vergelijken wordt een eenvoudige methode besproken om in milliseconden te kunnen meten. De voorbeelden en testen zijn terug te vinden in de bijlage, die hier is te downloaden.

Zoeken

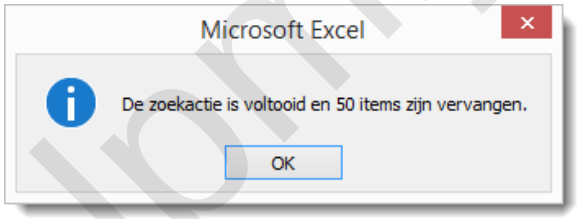
Excel kent een ingebouwde tool waarmee prima gezocht kan worden in de gegevens binnen een werkboek. Veel lezers zullen met deze zoek-tool bekend zijn en daarom wordt deze hier slechts kort besproken. De tool is op te roepen met *Ctrl + F*, of via menu **Tab Start** > **Zoeken en Selecteren** > **Zoeken**. Hierop verschijnt het Zoeken en Vervangen venster. Na een klik op **Opties** [1] wordt de uitgebreide modus getoond:

Bij [2] wordt de zoekopdracht ingevoerd. Daarbij kan worden opgeven waarbinnen wordt gezocht [3], of er op rij of op kolom wordt gezocht [4] en in wat voor soort gegevens wordt gezocht [5]. Hiernaast kan worden aangegeven of er hoofdlettergevoelig gezocht moet worden [6] en of de celinhoud identiek moet zijn aan de zoekopdracht [7]. Als laatste kan zelfs de celopmaak in de zoekopdracht worden verwerkt [8]. Met **Volgende Zoeken** [9] kan er eenvoudig stap voor stap langs de cellen worden gelopen waarvan de inhoud overeenkomt met de zoekopdracht. Door voor **Alles Zoeken** [9] te kiezen worden alle overeenkomstige cellen in één overzichtsvenster getoond.

Vervangen

Dit gaat op een vergelijkbare manier in hetzelfde venster. Hiervoor wordt echter de tab **Vervangen** geselecteerd:

Naast de zoekopdracht wordt er een vervangtekst [2] ingegeven. Ook de opmaak kan worden vervangen [2]. Door **Vervangen** [3] te kiezen worden één voor één de zoekresultaten vervangen door de vervangopdracht. Wanneer er voor **Alles Vervangen** [4]wordt gekozen, worden alle resultaten in één keer vervangen. Vrijwel onmiddellijk verschijnt de volgende melding om aan te geven dat de actie voltooid is:



Zoeken in VBA

In de VB-editor is een vergelijkbare functie ingebouwd om binnen het project te zoeken in de code. In Run Time is deze niet beschikbaar en dient er door middel van een methode in de gegevens van een werkblad te worden gezocht. Om dit te bereiken zou je mogelijk op het idee kunnen komen om met een lus een bepaalde range af te lopen om vervolgens de inhoud van de cellen te vergelijken met de zoekopdracht. Dat zou als volgt uitgewerkt kunnen worden:

Dit zal prima werken. Het kan echter eenvoudiger en, zeker wanneer het een omvangrijke range betreft, een stuk sneller.

Find

Hiervoor is de methode *Find* beschikbaar. Deze doorzoekt de hele aangegeven range in één keer op de zoekopdracht en geeft de eerst gevonden cel als resultaat. Dit kan als volgt:

Find is heel simpel in het gebruik. Het wordt iets ingewikkelder wanneer je meerdere kenmerken wil meegeven aan je zoekopdracht. Bij *Find* zijn de volgende eigenschappen in te stellen:

expression.Find (What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase, MatchByte, SearchFormat)

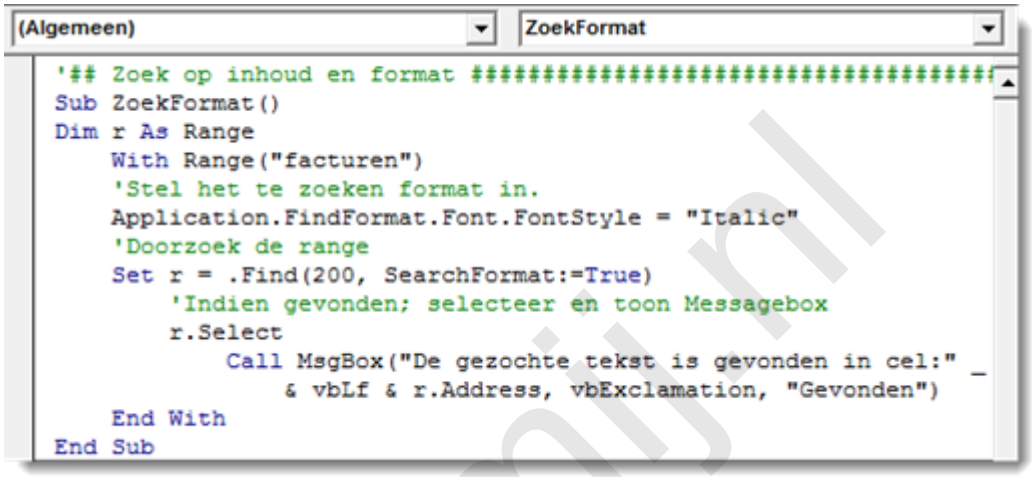
De eigenschappen kunnen uit Variant bestaan (variabel met ider mogelijk data type, of een Boolean (Variabele met twee mogelijke waartdes; waar of onwaar). Zie deel drie voor meer informatie over variabelen.

- *expression* (verplicht);
  Dit kan ieder geldig rangeobject zijn.
- *What* (verplichte Variant);
  De zoekopdracht zelf. De opdracht kan iedere waarde bevatten.
- *After* (optionele Variant);
  Hier kan een cel (uitsluitend één enkele cel, geen range) binnen de zoekrange worden gespecificeerd. De zoekopdracht zal starten in de cel volgend op de gespecificeerde cel.
- *Lookin* (optionele Variant);
  Hiermee kan worden gespecificeerd op welk type data moet worden gezocht. Mogelijkheden zijn: *xlFormulas* (formules) en *xlValues* (waarden)
- *Lookat* (optionele Variant);
  Hiermee kan worden aangegeven of er naar de complete cel-inhoud moet worden gezocht (*xlWhole*), of naar een gedeelte daarvan (*xlPart*).
- *SearchOrder* (optionele Variant);
  Dit stelt de zoekvolgorde in: per rij (*xlByRow*), of per kolom (*xlByColumns*).
- *SearchDirection* (optionele Variant);
  Hier wordt de zoekrichting gespecificeerd. Standaard is dit richting de volgende cel (*xlNext*). Zoeken richting de voorafgaande cel kan ook (*xlPrevious*).
- *MatchCase* (optionele Boolean);
  Deze waarde specificeert of er hoofdlettergevoelig (*True*) wordt gezocht, of niet (*False*).
- *MatchByte* (optionele Boolean);
  Deze waarde is alleen van belang als er dubbelbyte taalondersteuning is geïnstalleerd (*True*). Standaard is dit niet het geval (*False*).

- *SearchFormat* (optionele Boolean);
  Hiermee kan worden gespecificeerd dat er naar de opmaak van de gevonden cel moet worden gekeken (*True*), of niet (*False*).
  Standaard wordt er niet naar de celopmaak gekeken. Wanneer dat wel wordt ingesteld, is het noodzakelijk dat de opmaak eerst wordt gespecificeerd met *FindFormat*. De volgende syntax dient bijvoorbeeld voor de zoekopdracht gegeven te worden om op Italic tekst te zoeken:

  Application.FindFormat.Font.FontStyle = "Italic"

  Zie deze voorbeeld routine:



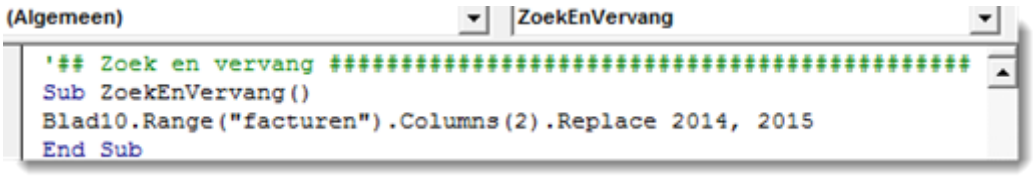
Replace

Net als bij zoeken zou je ook het vervangen van tekst kunnen uitvoeren met behulp van een loop. Bijvoorbeeld op de volgende manier:

Hier wordt *Replace* als functie ingezet. Dit kan echter ook als range methode:

expression.Replace (What, Replacement, LookAt, SearchOrder, MatchCase, MatchByte, SearchFormat, ReplaceFormat)

Dit zijn de zelfde eigenschappen als bij de methode *Find*, alleen is *After* vervangen door *Replacement*. Bij deze verplichte waarde wordt de vervangende tekst gegeven. Uit onderstaand voorbeeld blijkt wel dat dit heel eenvoudig is te programmeren:



Werkblad functies

Find is een zeer flexibele methode en kan heel goed de beschikbare werkbladfuncties vervangen welke ingebouwd zijn in Excel. Namelijk; *Vergelijken* (*Match*), Vert.Zoeken (*Vlookup*), Horiz.Zoeken (*HLookup*) en Index.

Toch zijn er redenen te bedenken om deze werkbladfuncties wel te gebruiken. Ze zijn soms eenvoudiger in te zetten. Dit omdat ze al bekend zijn als werkbladfunctie en in VBA op een vergelijkbare manier werken. Bovendien kunnen de procedures die hier gebruik van maken soms sneller zijn. Dat kan bijvoorbeeld bij gebruik van *Match* het geval zijn. Hiermee loopt de procedure veel sneller dan wanneer *Find* wordt ingezet. Nu is het niet zo, dat *Find* daarom niet meer gebruikt moet worden. Pas bij omvangrijke zoekopdrachten in grote hoeveelheden data zijn er verschillen merkbaar.

Performance meten

De ingebouwde timer in Excel die met VBA kan worden gebruikt is niet nauwkeurig genoeg om de verschillen in performance goed te kunnen meten. In VBA is echter aan zowat alles een mouw te passen. Op MSDN staat een interessant artikel (Engels) over het optimaliseren van de berekentijd van werkboeken. Dat is hier te vinden. In het werkboek wat als bijlage is bijgevoegd wordt de doorlooptijd vergelijken van de methode *Find* met die van de functie *Match*. De tijd meten we met een simpele vorm van de timer die in bovenstaand artikel wordt voorgesteld. Hierbij wordt gebruik gemaakt van de veel nauwkeuriger Windows timer. In de bijlage wordt daarvoor het volgende bovenin de module gedeclareerd.

Private Declare Function GetTickCount Lib "kernel32" () As Long

Hiermee wordt het object *GetTickCount* uit de Windows “kernel32” bibliotheek gebruikt. De waarde is nu in te stellen én uit te lezen met dit object.

In het werkboek zijn 63014 regels data aangemaakt waarin gezocht wordt. De procedure zoekt alle te late facturen en markeert de tekst rood. Bij herhaling van de procedure kan de executietijd soms verschillen. Dat is onder meer afhankelijk van wat er op de achtergrond in het systeem meedraait (virus scan, updates installeren etc.). Bij mij neemt de procedure ongeveer 1000 milliseconden in beslag. Om steeds het volgende resultaat te kunnen vinden wordt de zoekrange dynamisch aangepast (verkleind) met behulp van *Resize*. Hierdoor valt de cel van het laatste resultaat steeds net buiten de range, waardoor het volgende item gevonden kan worden. Zie hiervoor onderstaand voorbeeld:

(Algemeen) FacturenFind

```
'## Zoek te laten facturen en kleur de tekst ####################
Sub FacturenTestMatch()
Dim lTimer As Long
Dim r As Range
Dim lTeller As Long
'Begin, reset eerdere tests
With Blad2
        .Activate
        .Range("testdata").Font.ColorIndex = 1
        Set r = .Range("testdata")
End With
Application.ScreenUpdating = False
'Start de timer
lTimer = GetTickCount
'Geen match meer? Dan naar einde
On Error GoTo einde
Do
    lTeller = Application.WorksheetFunction _
        .Match("te laat", r.Columns(5), 0)
    'Indien gevonden; kleur tekst
    r.Range("B" & lTeller, "F" & lTeller).Font.ColorIndex = 3
    'Stel in op de volgende rij van resultaat
    'Zoek het volgende resultaat
    Set r = r.Resize(r.Rows.Count - lTeller, 1).Offset(lTeller, 0)
Loop
einde:
'Lees de timer uit
lTimer = (GetTickCount - lTimer)
'Selecteer het reslutaat
With Blad2
    .Activate
    .Cells(16, 2).Select
End With
'Geef de tijd
MsgBox lTimer
End Sub
```


Bij gebruik van de functie *Match* kan het van belang zijn dat voor het criteriumgetal (laatste argument) de waarde 0, oftewel *exacte overeenkomst* wordt meegegeven. De lijst gegevens hoeft dan niet gesorteerd te zijn. Dit moet bij de andere waardes wel het geval zijn. Bij een ongesorteerde lijst volgt er anders een foutmelding. Hetzelfde geldt voor *vLookup* en *hLookup*, alleen wordt dit criterium met *True* of *False* aangegeven (*False*) voor *exacte overeenkomst*. Om meerdere resultaten te vinden met de methode *Find* dient de zoekopdracht herhaald te worden met *FindNext*:

expression.FindNext (After)

Hiermee wordt de zoekopdracht herhaald vanaf de cel ná de cel die bij *After* wordt gespecificeerd. Dit mag alleen een enkele cel zijn. Door hier de cel van het laatste resultaat te geven wordt er automatisch steeds verder gezocht. De zoekopdracht kan in een lus worden herhaald. Het aantal keer dat de lus moet worden doorlopen kan worden bepaald met *CountIf* (Aantal.Als).

De complete procedure, inclusief timer, kan er dan bijvoorbeeld zo uitzien:

(Algemeen) FacturenTestFind

```
'## Zoek te laten facturen en kleur de tekst ######################
Sub FacturenTestFind()
Dim lTimer As Long
Dim r As Range
Dim lTeller As Long
Dim lRij As Long
'Begin, reset eerdere tests
With Blad2
        .Activate
        .Range("testdata").Font.ColorIndex = 1
        Set r = .Range("testdata")
End With
Application.ScreenUpdating = False
'Start de timer
lTimer = GetTickCount
'Start in de juiste rij van Resultaat
lRij = 16
'Zoek de eerste te late factuur
lTeller = lTeller + 1
With Blad2.Range("testdata")
Set r = .Find("te laat", SearchOrder:=xlByRows)
'Zoek de volgende te late facturen
For lTeller = 1 To Application.WorksheetFunction. _
    CountIf(.Columns(5), "te laat")
    'Indien gevonden; kleur de tekst
    Range("B" & r.Row, "F" & r.Row).Font.ColorIndex = 3
    'stel in op de volgende rij van resultaat
    lRij = lRij + 1
    'Zoek het volgende resultaat.
    Set r = .Columns(5).FindNext(After:=r)
Next lTeller
End With
einde:
'lees de timer uit
lTimer = GetTickCount - lTimer
'Selecteer het reslutaat
With Blad2
    .Activate
    .Cells(16, 2).Select
End With
'Geef de tijd
MsgBox lTimer & " milliseconden"
End Sub
```


Bij mij blijkt deze ongeveer twee keer zo lang te duren als met gebruik van de functie *Match*. Ik werd nieuwsgierig of dit ook beïnvloed werd door het verschil tussen een For Next-lus en een Do Loop-lus. Ik heb daarom deze procedure ook als Do Loop in het werkboek opgenomen. Deze blijkt (op mijn systeem) inderdaad een kleine 10 procent langzamer te lopen. Daar de verschillen klein zijn, is dit te weinig om daar een conclusie aan te verbinden.

Samenvatting:

In deze aflevering werd het zoeken naar gegevens in een werkboek besproken aan de hand van de methode *Find* en het zusje daarvan: *Findnext* (volgende). Daarnaast werd de methode *Replace* besproken waarmee gegevens kunnen worden vervangen. De beschikbare werkbladfuncties waarmee ook gezocht kan worden werden genoemd. Daarvan werd *Match* wat uitgebreider besproken. Hierbij werd ook aandacht geschonken aan de effecten op de performance. Om deze te kunnen vergelijken werd een makkelijke methode aangeboden om deze in milliseconden te kunnen meten. De

voorbeelden en testen zijn ook dit keer weer terug te vinden in een bijlage, welke hier is te downloaden.

Volgende aflevering

De volgende keer wordt besproken hoe een werkboek gedeeld kan worden zodat er meerdere gebruikers tegelijkertijd in kunnen werken. Hierbij worden de consequenties besproken die het delen voor het gebruik van macro's kan hebben en hoe hier mee kan worden omgegaan.