



Whitepaper Dynamische kolombepaling in MS Excel

Auteur: Emiel Nijhuis

Gepubliceerd: 15 maart 2010

Inleiding

Programmeren in Excel is en blijft lastig. Een belangrijke reden daarvan is Excel zelf: in rekenbladen is geen duidelijke scheiding tussen data, logica en presentatie aan te brengen. Het gevolg hiervan is dat er vanuit VBA veel mis kan gaan bij het aanroepen van de verschillende objecten.

Vooraf het benaderen van ranges vanuit code vormt een risico. Een grote bron van bugs ontstaat doordat er hard verwezen wordt naar een cel(len)adres:

```
Dim rColumn As Range
Set rColumn = Blad1.Columns("C:C")
rColumn.ClearContents
```

Wanneer een gebruiker een kolom invoegt of verwijderd vóór de derde kolom verwijst de code naar de verkeerde range: in het voorbeeld wordt nog steeds de derde kolom geleegd.

Het gebruiken van constanten biedt in dit geval ook geen oplossing. Het maakt de code wel makkelijker aanpasbaar maar lost het probleem niet op wanneer er een kolom wordt ingevoegd of verwijderd.

```
Const COLNR_CITY = 3
Dim rColumn As Range
Set rColumn = Blad1.Columns(COLNR_CITY)
rColumn.ClearContents
```

Een oplossing kan uiteraard zijn om het werkblad te beveiligen tegen het invoegen van kolommen. Deze beperking is echter niet altijd wenselijk. Er bestaan situaties waarin het invoegen/verwijderen van kolommen door gebruikers moet worden toegestaan.

Probleemstelling

De uitdaging is dus de volgende:

Op welke wijze kunnen vanuit VBA kolommen correct worden blijven benaderd m.b.t. een werkblad waarin kolommen handmatig kunnen worden ingevoegd/verwijderd?

Oplossing

Het antwoord moet worden gevonden in het gebruik van gedefinieerde namen ('named ranges').

In een werkblad kan een naam aan een cel(len)adres worden gegeven. Deze naam kan vervolgens in VBA worden gebruikt. Bijvoorbeeld: Geef de eerste cel van de kolom die plaatsnamen bevat de naam "Col_City". In ons voorbeeld is dat dus cel Blad1!C1.

Daarna is het eenvoudig om het kolomnummer van dit adres dynamisch te bepalen:

```
Dim rColumn As Range
Dim iColCity As Integer

iColCity = Blad1.Range("Col_City").Column
Set rColumn = Blad1.Columns(iColCity)
rColumn.ClearContents
```

Nb.: Gebruik i.p.v. `Blad1.Range("Col_City")` nooit de verkorte notatie `[Col_City]`. Dit leidt namelijk tot onvoorspelbaar gedrag bij het benaderen van het object.

Op deze wijze kunnen voor alle te benaderen kolommen de kolomnummers worden bepaald. Voor de onderhoudbaarheid van de code worden hierbij uiteraard constanten gebruikt. In het voorbeeld hierna worden de kolomnummers als membervariabelen geïnitieerd:



```
Public Const COL_NAME = "Col_Name"  
Public Const COL_ADDRESS = "Col_Address"  
Public Const COL_CITY = "Col_City"  
  
Private miColName As Integer  
Private miColAddress As Integer  
Private miColCity As Integer  
  
Public Sub InitializeColumnNumbers()  
    miColName = Blad1.Range(COL_NAME).Column  
    miColAddress = Blad1.Range(COL_ADDRESS).Column  
    miColCity = Blad1.Range(COL_CITY).Column  
End Sub
```

Binnen VBA kunnen vervolgens kolommen als volgt worden benaderd:

```
Private Sub ClearCities()  
    InitializeColumnNumbers  
    Blad1.Columns(miColCity).ClearContents  
End Sub
```

Helaas zijn we er dan nog niet. Omdat ons werkblad onbeveiligd is, kan een gebruiker een cel met een gedefinieerde naam verwijderen. Dit dient daarom te worden afgevangen. We zouden ervoor kunnen kiezen om bij sluiting van het bestand te controleren of alle door de code gebruikte gedefinieerde namen nog geldig zijn. Er is dan echter niet meer te bepalen wanneer een dergelijke verwijdering heeft plaatsgevonden. Beter is het daarom om de check uit te voeren tijdens het Change-event van het werkblad zelf.

Om de controle uit te voeren wordt van de routine `InitializeColumnNumbers` gebruik gemaakt. Deze routine wordt aangepast om hem ook aan te roepen vanuit het Change-event van `Blad1` én om indien nodig een zinvolle foutmelding te tonen.

```
Public Sub InitializeColumnNumbers(bEvents As Boolean)  
  
    Dim sNamedRange As String  
  
    On Error GoTo ErrH  
  
    'Sla gedefinieerde naam op i.v.m. evt. foutmelding  
    sNamedRange = COL_NAME  
    miColName = Blad1.Range(sNamedRange).Column  
  
    sNamedRange = COL_ADDRESS  
    miColAddress = Blad1.Range(sNamedRange).Column  
  
    sNamedRange = COL_CITY  
    miColCity = Blad1.Range(sNamedRange).Column  
  
    Exit Sub  
ErrH:  
    Dim sMessage As String  
  
    If bEvents Then  
        'Fout ontstaan tijdens Change-event  
        sMessage = "Deze actie wordt daarom ongedaan gemaakt."  
    Else  
        'Fout is al eerder ontstaan.  
        sMessage = "Sluit dit bestand zonder de wijzigingen op te slaan" & vbCrLf & _  
            "of open een versie waarin de gedefinieerde naam geldig is."  
    End If  
    Err.Raise 666, , "Er is een gedefinieerde naam verwijderd die voor de" & vbCrLf & _  
        "correcte werking van deze applicatie van belang is." & vbCrLf & _  
        "Verwijderde gedefinieerde naam: " & sNamedRange & String(2, vbCrLf) & _  
        sMessage  
End Sub
```

We willen echter niet dat bij elk Change-event in `Blad1` de controle wordt uitgevoerd. Het kan de performance negatief beïnvloeden en maakt debugging omslachtig. Vandaar dat wanneer dit event wordt afgevuurd eerst een check wordt gedaan; er wordt gecontroleerd of de Target-range van het Change-event betrekking heeft op de checken gedefinieerde namen.



```
Private Sub Worksheet_Change(ByVal Target As Range)

    Dim rCheck As Range
    Dim oName As Name

    On Error GoTo ErrH

    'Check of de eerste gedefinieerde naam bestaat
    If Not ValidNamedRange(COL_NAME) Then
        Call InitializeColumnNumbers(True) 'Provoceer foutmelding
    End If
    'Check of de laatste gedefinieerde naam bestaat
    If Not ValidNamedRange(COL_CITY) Then
        Call InitializeColumnNumbers(True) 'Provoceer foutmelding
    End If

    'Beide gedefinieerde namen bestaan. Stel het check-gebied vast
    Set rCheck = Me.Range(Me.Range(COL_NAME).Cells(1), _
        Me.Range(COL_CITY).Cells(1))

    'Check of er in de bewuste range is gewijzigd door gebruiker
    If Not Intersect(rCheck, Target) Is Nothing Then
        'Controleer of alle gedefinieerde namen die worden gebruikt
        'in de applicatie of ze niet verwijderd zijn.
        Call InitializeColumnNumbers(True)
    End If

    Exit Sub
ErrH:
    'Toon de foutmelding
    MsgBox Err.Description, vbExclamation
    'Maak de actie ongedaan
    Application.EnableEvents = False
    Application.Undo
    Application.EnableEvents = True
End Sub
```

Wanneer een range uit een werkblad wordt verwijderd waarin zich een gedefinieerde naam bevindt wordt die zelf niet verwijderd. Het adres van de gedefinieerde naam wordt echter veranderd in #VERW! of #REF!. Vandaar dat het Change-event de volgende routines gebruikt om vast te stellen of een gedefinieerde naam nog valide is:

```
'Check of de gedefinieerde naam (nog) valide is
Private Function ValidNamedRange(sName As String) As Boolean
    ValidNamedRange = NamedRangeExists(sName) And ValidNamedRangeRef(sName)
End Function

'Check of een gedefinieerde naam bestaat
Private Function NamedRangeExists(sName As String) As Boolean

    Dim oName As Name

    For Each oName In ThisWorkbook.Names
        If oName.Name = sName Then
            NamedRangeExists = True
            Exit For
        End If
    Next

End Function

'Check of de gedefinieerde naam een geldige verwijzing bevat
Private Function ValidNamedRangeRef(sName As String) As Boolean

    Dim s As String

    On Error GoTo ErrH
    s = Range(sName).Address
    ValidNamedRangeRef = True
    Exit Function
ErrH:
End Function
```



Het verdient aanbeveling om bij het uitvoeren van macro's die wijzigingen veroorzaken in Blad1 het afvuren van events te onderdrukken. Dit wordt hieronder geïllustreerd:

```
Private Sub ClearCities()  
  
    On Error GoTo ErrH  
  
    Application.EnableEvents = False  
  
    Call InitializeColumnNumbers(False)  
    Blad1.Columns(miColCity).ClearContents  
  
CleanUp:  
    'Opruimactie(s)  
    Application.EnableEvents = True  
    Exit Sub  
ErrH:  
    'Toon foutmelding  
    MsgBox Err.Description, vbExclamation  
    Resume CleanUp  
End Sub
```

Conclusie

Het is lastig maar niet ondoenlijk om op een sluitende wijze het dynamisch bepalen van specifieke kolomnummers te implementeren.

De routine InitializeColumnNumbers wordt gebruikt voor zowel de initialisatie van de membervariabelen als het checken van de gebruikte gedefinieerde namen. Hierdoor kunnen centraal en eenduidig de gedefinieerde namen worden onderhouden die in de code worden gebruikt. Uitzondering hierop vormen de aanduiding van de begin- en eindcel in het Change-event. Hier kunnen echter eenvoudig eigen constanten/variabelen voor worden gedefinieerd.

Opmerkingen

- De gebruiker dient uiteraard wel de macro's in te schakelen alvorens het bestand te openen.
- Het Excel-bestand Dynamische_Kolombepaling.xls bevat het voorbeeld zoals in dit whitepaper uiteen is gezet.

Emiel Nijhuis
Delegate
<http://www.delegate.nl>