Ekstra hastighed med array's i Excel-VBA

Brugen af arrays til indlæsning af og skrivning til celler i Excel kan give betragtelige hastighedsforøgelser af VBA-koder med en relativ lille indsats.

Skrevet den **03. Feb 2009** af **bak** I kategorien **Regneark / Excel** | ★★★☆☆

Ekstra hastighed med array's i Excel-VBA

Mange koder i excel består af et loop gennem er mængde celler. Dette bruges ofte til at søge gennem cellerne eller lave diverse ændringer. Dette virker også fint normalt på mindre områder. Når området bliver større begynder koden at blive langsom pga. de mange læsninger og skrivninger. Selve det, man laver i loopet, er sjældent langsom. Den store tidsrøver i den forbindelse er at skrive til regnearket. Indlæsning er normalt relativ hurtigt.

At løbe et array igennem går derimod meget hurtigt. Ideen går derfor ud på at læse alle cellerne ind i et array på én gang, gennemløbe loopet og skrive hele arrayet tilbage til regnearket i et hug..

Der er i princippet flere slags array's der kan bruges, men de fælles for dem er at det kun er én- og todimensionelle, der umiddelbart kan bruges.

Én-dimensionelle arrays
Kunne være et eget defineret array som fx:
MyArray = Array("Jan", "Feb", "Mar", "Apr", "Maj", "Jun")

For at skrive dette array til et regneark kan man bruge standardmetoden

```
MyArray = Array("Jan", "Feb", "Mar", "Apr", "Maj", "Jun")
For x = 0 to 5
    Cells(1,x+1) = MyArray(x)
Next
```

Dette giver 6 skrivninger til arket
Et alternativ er at skrive alle på én gang.

```
Range("A1:F1") = MyArray
```

Skal de skrives I en kolonne istedet skal vi lige have dem vendt engang. Dette kunne gøres i loopet men lidt enklere således:

```
Range("A1:A6") = Application.Transpose(MyArray)
```

To-dimensionelle arrays:

For at hente data fra et regneark er det nemmest at bruge et to-dimensionelt variantarray.
Et variantarray behøver ikke dimensiones først men kan bruges direkte på arket
I dette eksempel hentes data fra alle celler området A1:C5000, altså 15.000 celler

```
Dim MyArray as Variant
MyArray = Range("A1:C5000")
```

Tidsforbruget er så lavt at det næsten ikke kan måles
Hvis man nu undersøger MyArray nærmere i Local's vinduet vil man se at arrayet er en nøjagtig kopi af de data der er i A1:C5000, men UDEN formler. Der kommer altså kun rå data over.
MyArray vil være automatisk dimensioneret således MyArray(5000, 3), hvilket betyder at rækkerne ligger i 1. dimension og kolonnerne i 2. dimension. Dvs. samme system som cellerne og værdier kan nu kaldes enkeltvis efter samme system

MyArray(10, 2) vil være data fra række 10, kolonne 2 (B10)
MyArray(100, 3) vil være data fra række 100, kolonne 3 (C100)

For at gennemløbe alle elementer i arrayet findes laveste elementnummer og højeste elementnummer for henholdsvis 1. dimension og 2. dimension

```
For x = Lbound(MyArray, 1) to Ubound(MyArray, 1)
     For y = Lbound(MyArray, 2) to Ubound(MyArray, 2)
           MyArray(x, y) = MyArray(x, y) * 10
     Next
Next
```

Lbound(MyArray, 1) giver laveste elementnummer i 1. dimension, Ubound højeste.
Lbound(MyArray, 2) giver laveste elementnummer i 2. dimension.

Et sådant array er meget hurtigt at gennemløbe, da alt befinder sig i maskinens hukommelse og ikke først skal læses.

At skrive tilbage til et regneark, er lige så enkelt, som at hente data.

```
Range("A1:C5000") = MyArray
```

Dette skriver alle 15.000 celler I et hug og tidsforbruget er uhyre lille.
Man behøver selvfølgelig ikke at skrive sammested som man hentede data.

'Hent:
MyArray = Worksheets(1).Range("A1:C5000")

'Indsæt:
Worksheets(2).Range("D1:F5000") = MyArray

Hvis man ønsker at placere arrayet et andet sted og kun kender startcellen kan Resize-kommandoen bruges. Her placeres arrayet i F20, som øverste, venstre celle.

Range("F20").Resize(Ubound(MyArray, 1), Ubound(MyArray, 2)) = MyArray

Resize kommandoen udvider rangen F20 til samme størrelse som arrayet.

Her er hele koden :

```
Dim MyArray As Variant
MyArray = Range("A1:C5000")
For x = LBound(MyArray, 1) To UBound(MyArray, 1)
     For y = LBound(MyArray, 2) To UBound(MyArray, 2)
           MyArray(x, y) = MyArray(x, y) & "xxx"
     Next
Next
Range("F20").Resize(UBound(MyArray, 1), UBound(MyArray, 2)) = MyArray
Set MyArray = Nothing
```

Bemærk en sidste linie er til for at fjerne arrayet fra hukommelsen igen. Dette vil ske automatisk ved kodens afsluttelse (End Sub) da MyArray er lokal - defineret, men man kan jo få brug for ekstra hukommelse før og det er en god regel at "rydde op" efter en kode..
Koden kører på under 0,4 sek. på en ældre 300 Mhz labtop.

Almindeligt Array
Man kan ikke tildele et range til et almindelig array.
Men man kan godt gøre det omvendte, altså indsætte et array i et range. Man bruger samme metode som med variant-arrayet.

```
Dim MyArray(5000, 3)
Dim x As Long, y As Long
For x = 1 To 5000
    For y = 1 To 3
        MyArray(x, y) = x * y
    Next
Next

Range("F1:H5000") = MyArray
```

Ligesom ved den en-dimensionelle array en det muligt at vende et to-dimensionelt array med Application.WorksheetFunction.transpose

```
Dim MyArray as Variant
MyArray = Range("A1:C30")
Range("A1:AD3") = Application.WorksheetFunction.Transpose(MyArray)
```

Afslutning.

Brugen af array's giver muligvis et par kodelinier ekstra, men den hastighed koden afvikles med kan være op til en faktor 100 større end med direkte læsning og skrivning til celler i et regneark.
Med lidt øvelse er det faktisk nemt at bruge dem, men husk at man kun kan overføre rå data og ikke formler, formater, valideringer mv.

Kommentar af hnteknik d. 27. Jan 2006 | 1

Fin artikel

Kommentar af x-lars d. 22. Feb 2005 | 2

Kommentar af jpvj d. 24. Nov 2004 | 3

Hej Bak,

Rigtig flot artikel. Du beskriver grundigt og samtidigt enkelt hvordan man overkommer et muligt performance problem i Excel.

Alt i alt en lille begrænset problemstilling med en god forklaring og gode eksempler!

/JP

Kommentar af jesperfjoelner d. 14. Sep 2005 | 4

Denne artikel har løst et specifikt problem, jeg havde med data eksport fra Access til Excel. Herligt.

Kommentar af skarvenneverdies d. 05. Dec 2004 | 5

Lækkert stykke arbejde...
Det er artikler af denne kvalitet der burde ligge på E....så behøver jeg vidst ikke sige mere.. :-)

Kommentar af webgon d. 29. Nov 2004 | 6

Vil nu ikke blot kalde denne artikel et 'fif' men en god, uddybende artikel ;-) 5/5 points!

Kommentar af miko67 d. 23. Nov 2004 | 7

kompetent og overskuelig artikel der holder sig fint til et begrænset emne.

Kommentar af mrjh d. 28. Sep 2006 | 8

Meget forståeligt skrevet for en nybegynder i VBA, og en artikel jeg jævnligt vender tilbage til, for at genopfriske min viden om brug af arrays.

Kommentar af mikeot d. 26. Nov 2004 | 9

Godt fif! - havde ikke tænkt på denne måde at gøre det på.

Kommentar af lorentsnv d. 15. Jan 2005 | 10

Meget nyttig artikkel!!

Kommentar af nils_ d. 07. Mar 2007 | 11

interessante tidsbetragtninger

Kommentar af Sitestory d. 09. Nov 2013 | 12

Glimrende guide til et godt fif, som jeg bruger meget. Afslutningsbemærkningen, om at man ikke kan overføre formler, er dog ikke korrekt. Hvis man skriver:

MyArray() = Range(etellerandet).Formula

kopieres formlerne i stedet for værdierne, og det gælder også den anden vej, når man kopierer sit array til et range:

Range(etellerandet).Formula = MyArray()