7 Andre Excel objekter

7.1 Introduktion

Dette kapitel skal egentlig betragtes som en udvidelse af både kapitel 6 og kapitel 7. Vi skal her arbejde med yderligere tre meget anvendte objekter. For at kunne arbejde i Excel skal man først åbne en ny eller eksisterende **Workbook**. I denne **Workbook** findes de regneark og **Charts,** som ens applikation skal (vil) bestå af. Det er netop disse tre objekter, vi vil se nærmere på. Der findes mange andre objekter i Excel, men hvis du kender disse tre samt **Range** objektet, kan du lave meget. Som sædvanligt er der under online hjælpen en mere udtømmende dokumentation af de tre objekters metoder og egenskaber, end vi når at se på her.

7.2 Collections og specielle medlemmer af Collections

I forrige kapitler så vi, hvorledes vi sondrede imellem Collections og medlemmer af Collections ved at føje flertals s'et til Collections. I dette afsnit vil vi se på, hvordan vi specificerer et medlem af en Collection, og hvordan vi specificerer hierarkiet i objekt modellen. Lad os se på **Workbook** objektet. **Workbooks** Collectionen er samlingen af alle åbne workbooks, og ethvert medlem af denne Collection kan specificeres ved dets navn, som f.eks.

```
Workbooks("MedarbUndersøgelse")
```

som refererer til "MedarbUndersøgelse.xls" filen. Et bestemt regneark i denne fil, f.eks. Data, kan refereres som `Worksheets("Data")`, og en graf over resultaterne kan refereres som `Charts("DisplayData")`. Som du ser, er ideen, at du skriver objektet med flertals s'et, og at du i en parentes med anførelsestegn skriver navnet på medlemmet. Du kan selvfølgelig også angive et tal i stedet, men igen for at få klarhed over tingene, vil jeg anbefale at du skriver navnet.

Hierarkiet i ovenstående er, som vi har set lidt på, at **Workbooks** Collectionen består af de individuelle medlemmer af workbooks, og enhver af disse indeholder en **Worksheets** Colllection og en **Charts** Collection. Hvis et bestemt worksheet, f.eks. Data, tilhører den p.t. aktive **Workbook**, kan du referere til regnearket som `Worksheets("Data")`, eller mere pædagogisk `ActiveWorkbook.Worksheets("Data")`, så du indikerer, at det er arket i den p.t. aktive workbook. Andre gange, f.eks. når du skal referere til andre workbooks eller blot eksplicit vil angive et helt bestemt ark, skriver du det helt ud, som:

```
Workbooks("MedarbUndersøgelse").Worksheets("Data")
```

Samme historie kan gentages ved **Charts** Collectionen, hvor `Charts("DisplayData")` eller `ActiveWorkbook.Charts("DisplayData")` kan anvendes i den p.t. aktive workbook, og hvor `Workbooks("MedarbUndersøgelse").Charts("DisplayData")`, anvendes, når vi ønsker at angive tingene eksplicit.

Worksheets Collectionen er et step ned i hierarkiet i forhold til **Workbooks** collectionen, og **Range** objekter er endnu et step ned i hierarkiet. Forstil dig, at du vil angive ranget B2:B19. Hvis ranget eksisterer i det p.t. aktive *regneark*, kan du referere til det som `Range("B2:B1")` eller som `ActiveSheet.Range("B2:B19")`. Hvis du på den anden side vil angive ranget eksplicit, skal du i stedet skrive `Worksheets("Data").Range("B2:B19")`, hvor Data arket selvfølgelig tilhører den p.t. aktive workbook.

Hvis du eksplicit vil angive at ranget findes i et regneark i Medarbundersøgelse.xls filen, skal du også angive dette som:

```
Workbooks("MedarbUndersøgelse").Worksheets("Data").Range("B2:B19").
```

Når du først ved, hvordan du refererer til disse objekter, er det meget nemt at referere til deres egenskaber ved at tilføje et punktum og derefter egenskaben eller metoden efter referencen, eksempelvis:

```
Workbooks("MedarbUndersøgelse").Worksheets("Data").Range("B2:B19").Font.Bold = True
```

Som jeg kort nævnte i et andet kapitel, er **Workbooks**, **Worksheets**, og **Charts** Collectionerne også objekter, hvilket betyder, at disse også besidder deres egne metoder og egenskaber. En af de mest anvendte ved samtlige objekter er nok **.Count** egenskaben. F.eks. returnerer egenskaben `ActiveWorkbook.Worksheets.Count` antallet af regneark i den aktive workbook. Den vel nok mest anvendte metode i de tre Collections er **.Add** metoden, som adderer et nyt medlem, som bliver det nuværende aktive medlem, til Collectionen. Et eksempel:

```
ActiveWorkbook.Worksheets.Add
ActiveSheet.Name = "NyeData"
```

Den første linie definerer et nyt medlem af **Worksheets** Collectionen, og den anden navngiver dette nye medlem med navnet "NyeData"

7.3 Eksempler på at håndtere Workbooks i VBA

7.3.1 Åben og lukke en Workbook

Vi vil i dette afsnit se eksempler på metoder til at åbne og lukke workbooks fra VBA og eksempler på de egenskaber, som er forbundet med **Workbooks** objektet, som også kan håndteres fra VBA. Betragt følgende kode:

```
Sub OpenWorkbook()
       Workbooks.Open Filename := ”C:\myfiles\vba\Test.xls”
       MsgBox “Der er “ & ActiveWorkbook.Worksheets.Count & “ regneark i “ & _
              ActiveWorkbook.Name & “ filen.”
       Workbooks(“Test.xls”).Close
End Sub
```

I koden anvender jeg to egenskaber, **Count** og **Name**, som er forbundet med hhv. **Worksheets** og **Workbooks** Collectionerne. **Count** har vi lige set og **Name** egenskaben returnerer her navnet på workbooken. Der findes selvfølgelig også en **Name** egenskab som lægger sig til **Worksheets** Collectionen. For at åbne en workbook, kan du se, at **.Open** metoden kaldes. **Open** metoden kræver et filnavn, som skal overføres som argument til metoden. Argumentet skal specificere filnavnet samt stien til filen, før det går godt. Efter beskeden er vist, (MsgBoksen du ved!), kalder jeg **.Close** metoden. Jeg har her ikke angivet nogen argumenter, for at anvende default måden metoden. Du vil ved selvsyn se, hvilke argumenter der kan angives både til **Open** og **Close** metoderne ved at konsultere online hjælpen i VBE.

7.3.2 Gem en Workbook

Når nu du har lavet nogle ændringer i en workbook, kan det være en fordel også at kunne gemme disse, uden at du først skal stoppe programmet og gøre det fra Excel. Til **Workbooks** Collectionen er der tilknyttet to metoder til at udføre netop denne opgave. Den ene er **Save** metoden, og den anden er **SaveAs** metoden. Begge aggerer på samme måde som de tilsvarende metoder under Excel. Save kræver ingen argumenter, hvorimod **SaveAs** kræver argumenter, som fortæller noget om, hvordan metoden skal udføre denne saving. Betragt følgende eksempel:

```
Sub SaveWorkbook()
       With ActiveWorkbook
       ‘Gemt under samme navn – ingen spørgsmål tak!
              .Save
       ‘SaveAs kræver argumenter som dem der skal indtastes hvis metoden_
       ’blev anvendt under Excel
              .SaveAs Filename := ”C:\myfiles\vba\NyTest.xls”, _
                     FileFormat := xlWorkbooknormal
              MsgBox “Filen hedder nu “ & .Name
       End With
End Sub
```

Læg her mærke til **Filename** og **FileFormat** argumenterne, som er de to mest anvendte argumenter til **.SaveAs** metoden. Check argumenterne i online hjælpen.

7.3.3 Find stien til en Workbook

Når du i Excel skal åbne en fil, skal du typisk angive stien ned til den mappe, hvor filen ligger, inden den kan åbnes. I VBA kan du anvende **ThisWorkbook** objektet, som altid refererer til den workbook som indeholder VBA koden. Hvis du vil have returneret stien ned til den mappe, hvor filen du p.t. arbejder på ligger, kan du kalde **.Path** metoden. Eksempelvis, hvis koden,

som du udvikler, ligger i samme folder som workbooken Test.xls, kan du åbne Test.xls ved at udføre:

```
Workbooks.Open ThisWorkbook.Path & "\Test.xls"
```

Hvor du skal bemærke det foranstående \ - tegn, der er nødvendig, idet `ThisWorkbook.Path` ikke inkluderer det bagerste \ - tegn.

Hvis du på den anden side har lyst til at få en menu op på skærmen, hvor du kan klikke dig rundt til den rigtige fil, kan du anvende **.GetOpenFilename** metoden:

```
Sub OpenWorkbook2()
    Dim Filnavn As String
    Filnavn = Application _
                .GetOpenFilename("(*.xls),*.xls, (*.xla),*.xla, (*.*),*.*", , "Vælg
fil")
    Workbooks.Open Filename:=Filnavn
    MsgBox "Stien til filen er " & ActiveWorkbook.Path
    ActiveWorkbook.Close
End Sub
```

I eksemplet kan du kan se, at jeg først finder et filnavn med **.GetOpenFilename** metoden, som ligger under **Application** objektet. Derefter prøver jeg at åbne filen. Prøv selv i online hjælpen at få mere hjælp omkring denne metode. Læg også mærke til `ActiveWorkbook.Path` kaldet, som skal anvendes her, for at få stien vist. I dette tilfælde virker `ThisWorkbook.Path` nemlig *ikke*. Den ovenstående metode er kendetegnet ved at være "quick and dirty", fordi jeg ikke har testet, om jeg egentlig finder en fil (f.eks. kunne jeg jo have trykket på Annuller). Det kan ikke siges ofte nok – test nu om der er fundet et filnavn, *inden* du kalder **Open** metoden, ellers sker der uforudsigelige ting. En metode til at håndtere situationer som denne vil være at indføre et par passende linier kode som vist nedenfor:

```
Sub OpenWorkbook3()
    Dim Filnavn As String
    Filnavn = Application _
           .GetOpenFilename("(*.xls),*.xls,  (*.xla),*.xla,  (*.*),*.*",  ,  "Vælg
fil")
    If Filnavn <> False Then
           Workbooks.Open Filename:=Filnavn
           MsgBox "Stien til filen er " & ActiveWorkbook.Path
           ActiveWorkbook.Close
    End If
End Sub
```

hvor **If** konstruktionen tager hånd om netop det problem, idet `Filnavn` vil have værdien **False** hvis der trykkes på Annuller.

7.4 Eksempler på at håndtere Worksheets i VBA

I dette afsnit ser vi på, hvordan **Worksheets** håndteres fra VBA. De egenskaber og metoder, vi så i relation til **Workbook** objektet, gælder stort set også for **Worksheet** objektet. F.eks. er måden at oprette, navngive og slette et **Worksheet** på fuldstændig analog til den måde som anvendes ved **Workbook**.

```
ActiveWorkbook.Worksheets.Add      'Oprette et nyt ark under aktive Workbook
ActiveSheet.Name = "Nyt Navn"      'Navngive det p.t. aktive regneark
ActiveSheet.Delete                 'Slette[12] det p.t. aktive regneark
```

Informationer om Amter

For at kunne illustrere operationer med **Worksheet** objektet har jeg lavet en workbook, som indeholder nogle oplysninger om en del af de danske amter. Oplysningerne for hvert amt ligger i hvert sit worksheet. Det første worksheet, AlleAmter, indeholder navnene på de amter, jeg har medtaget, og ser således ud:

	A	B	C	D	E
1	**Oplysninger om Amter i Danmark**				
2	Fredensborg Amt				
3	Københavns Amt				
4	Nordjyllands Amt				
5	Ringkøbing Amt				
6	Roskilde Amt				
7	Sønderjyllands Amt				
8	Vejle Amt				
9	Vestsjællands Amt				
10	Århus Amt				
11					

De efterfølgende worksheets indeholder hver oplysninger om Amtsgårdens placering, antal indbyggere i amtet og amtets areal i km^2, og ser for f.eks. Frederiksborg Amt således ud:

	A	B
1	Amtsgård	Hillerød
2	Antal indb.	368116
3	Areal, km^2	1347
4		
5		
6		

[12] `Husk evt. DisplayAlert = False`

Alle amts tallene har jeg "sakset" fra web'en og de er her gengivet i tabelform.

Areal og indbyggertal

Amt	Antal indb. 1. januar 2001	Areal, km²	Indb. pr. km² 1. januar 2001
Københavns Amt	615.115	526	1.170
Frederiksborg Amt	368.116	1.347	273
Roskilde Amt	233.212	891	262
Vestsjællands Amt	296.875	2.984	99
Storstrøms Amt	259.691	3.398	76
Bornholms Amt	44.126	589	75
Fyns Amt	472.064	3.486	135
Sønderjyllands Amt	253.249	3.939	64
Ribe Amt	224.446	3.132	72
Vejle Amt	349.186	2.997	117
Ringkjøbing Amt	273.517	4.854	56
Århus Amt	640.637	4.561	140
Viborg Amt	233.921	4.122	57
Nordjyllands Amt	494.833	6.173	80
Amter i alt	**4.758.988**	**42.999**	**111**
Københavns Kommune	499.148	88	5.656
Frederiksberg Kommune	91.076	9	10.385
Hele landet	**5.349.212**	**43.096**	**124**

Kilde: Danmarks Statistik.

hvor de amter, jeg endnu ikke har indføjet, er amterne[13]:

Amt	Amtsgård placering
Bornholms Amt	Rønne
Fyns Amt	Odense
Ribe Amt	Ribe
Storstrøms Amt	Nykøbing F.
Viborg Amt	Viborg

Visning af Amter og placeringen af Amtsgården

Vi ønsker nu at lave en sub, som gennemløber alle ark, dog ikke det første, og udskriver oplysningerne om det enkelte Amt til en **MsgBox**. Et amts ark refererer vi til ved hjælp af en variabel, `ws`, af typen **Worksheet**, så kan vi nemlig anvende en **For Each** løkke.

```
Sub Worksheets1()
    Dim ws As Worksheet
    'Går nu igennem hver amt og udskriver info.
    For Each ws In ActiveWorkbook.Worksheets
        With ws
            If .Name <> "AlleAmter" Then
```

[13] Tallene finder du i ovenstående tabel

```
                MsgBox "Amtgården i " & .Name & " ligger i byen " & _
                    .Range("b1") & vbCrLf _
                    & "Antal indb. er " & .Range("b2") & _
                    " Hvilket over et area på " & .Range("b3") & _
                    " km^2 betyder at der er " & vbCrLf & _
                     Format(.Range("b2") / .Range("b3"), "#.00") _
                     & " indb. pr. km^2"
            End If
        End With
    Next
End Sub
```

If sætningen sikre her, at vi ikke udskriver oplysningerne for det første ark, fordi kun i de tilfælde hvor `ws.Name <> "AlleAmter"`, bliver **If** sætningen evalueret til at være sand.

Vi kan også lave en sub, som skriver Amtsnavnene og Amtsgårds placeringerne i én **MsgBox**. For at kunne gøre det, skal vi både kunne finde navnet på selve regnearket og finde den celle, som indeholder bynavnet for alle indførte amter. Det gør vi så:

```
Sub Worksheets2()
' Denne sub skriver alle amter fra AllAmter arket
' Læg også her mærke til den indbyggede konstant vbCrLf, som er kort for
' Visual Basic Carriage Return Line Feed
    Dim ws As Worksheet, Msg As String
    Msg = "Amter og amtsgårdsplacering:"
    For Each ws In ActiveWorkbook.Worksheets
        If ws.Name <> "AlleAmter" Then _
            Msg = Msg & vbCrLf & ws.Name & ": " & ws.Range("B1")
    Next
    MsgBox Msg, vbInformation, "Amt info"
End Sub
```

I rutinen kan du se, at jeg opdaterer en `Msg` variabel for hvert ark, jeg undersøger. For hvert ark tildeler jeg variablen dels navnet på arket og dels oplysningen i celle B1. Når alle ark er gennemløbet, stopper **For Each** løkken og jeg kalder derefter **MsgBox** funktionen med argumenterne `Msg`, **VbInformation**, "Amt info" som argumenter.

Resultatet af kørslen medfører følgende beskedboks:

Tilføjelse af et nyt Amt

Den næste rutine, vi skal se på, er en rutine, som kan tilføje et nyt amt til den eksisterende workbook. Informationerne har du ovenfor i den viste tabel. Rutinen skal laves på den måde, at den bliver ved med at spørge om et nyt amt indtil brugeren faktisk indtaster et regulært nyt amt, dvs. der skal undersøges om amtet eksisterer i forvejen.

Når brugeren har indtastet et nyt amt, skal opbygningen af arket for dette nye amt kopieres fra et eksisterende ark og amtets oplysninger skal herefter indskrives i dette nye ark. Endelig, skal amtets navn tilføjes listen af eksisterende amter i arket AlleAmter. Lad os se på det:

```
Sub Worksheets3()
' Denne sub spørger brugeren efter et nyt amt og dets informationer, og laver
dette
' ved at lave et nyt ark
    Dim IsNew As Boolean, NytAmt As String, AG As String, Indb As Long, _
        Areal As Double, ws As Worksheet
' Bliv ved med at spørge efter et nyt amt indtil der faktisk er tale om et nyt amt
    Do
        NytAmt = InputBox("Indtast et nyt Amt (kun navnet ikke Amt). ", "Nyt Amt")
        IsNew = True
        For Each ws In ActiveWorkbook.Worksheets
            If NytAmt & " Amt" = ws.Name Then
                MsgBox "Dette Amt findes allerede - indtast et andet.", _
                    vbExclamation, "Duplikeret Amt"
                IsNew = False
                Exit For
            End If
        Next
    Loop Until IsNew
' Kommet hertil ved vi, at der er tale om et nyt Amt –
' Faa de nødvendige oplysninger.
    AG = InputBox("Indtast byen hvor amtsgården findes " & NytAmt, "Amtsgården")
    Indb = InputBox("Indtast indbygger antal for " & NytAmt, _
        " Antal indb.")
    Areal = InputBox("Indtast arealet for " & NytAmt, " Arealet")
' Tilføj nu amtet til AlleAmt arket.
    Worksheets("AlleAmter").Range("A1").End(xlDown).Offset(1, 0) = NytAmt & " Amt"
' Kopier nu et eksisterende ark over til et nyt ark,
' som nu bliver det nuværende aktive ark.
' Opdater så navnet og oplysningerne
```

```
    Worksheets("Frederiksborg Amt").Copy after:=Worksheets(Worksheets.Count)
    With ActiveSheet
        .Name = NytAmt & " Amt"
        .Range("B1") = AG
        .Range("B2") = Indb
        .Range("B3") = Areal
    End With
End Sub
```

Der er flere ting i denne rutine, som fortjener en kommentar. I **Do** løkken kan du se, at jeg kræver at `IsNew` variablen er sand, førend jeg kan slutte løkken. Hvis der bliver indtastet et amt, der allerede eksisterer – det er det samme som at sige, at **If** betingelsen bliver opfyldt - opdateres `IsNew` til at være falsk, og **For** løkken termineres. Kun i det tilfælde hvor vi ikke får evalueret **If** betingelsen til at være sand, bryder vi **Do** løkken, altså hvor `IsNew` vedbliver med at være sand, kan vi gå videre. Når vi går videre, sørger et par **InputBox** kald for, at brugeren bliver spurgt om de relevante oplysninger.

Umiddelbart herefter indskriver vi det nye amt i AlleAmter arket ved at anvende følgende kode:

```
Worksheets("AlleAmter").Range("A1").End(xlDown).Offset(1, 0) = NytAmt & " Amt"
```

Hvor vi med udgangspunkt i celle A1 anvender `.End(xlDown)` til at gå til den sidste ikke tomme celle i kolonnen A, og endelig anvender `.Offset(1,0)` til at gå endnu en celle ned. Denne celle er nemlig kendetegnet ved at være den første ubrugte celle i kolonne A.

Det næste, vi udfører, er kopieringen af et eksisterende ark til et nyt ark. Det gør vi med linien:

```
Worksheets("Frederiksborg Amt").Copy after:=Worksheets(Worksheets.Count)
```

hvor arket tilhørende Frederiksborg amt agerer skabelon for det nye ark. Det nye ark bliver placeret som det sidste ark i hele samlingen af alle ark med kommandoen

```
after:=Worksheets(Worksheets.Count)
```

som netop tæller alle nuværende ark = 9 og placerer det nye ark efter nummer 9. Det resulterende 10. ark bliver efter kopieringen det p.t. aktive ark, hvilket betyder, at vi med **ActiveSheet** kan opdatere oplysningerne med det indtastede.

Det skal til sidst bemærkes, at der ikke undersøges for tomme indtastninger. Muligheden for tomme indtastninger anvendes typisk til at give en bruger chancen for at kunne stoppe programmet. Dette gøres ved at undersøge, om brugeren bare har trykket på <Enter> i en passende **If** sætning. Hvis det sker, skal programmet afsluttes.

Sortering af Worksheets

Når vi har indtastet flere amter, ligger de ikke i alfabetisk orden, men det kan vi gøre bod på ved at skrive en rutine, som netop kan sortere ark. I dette eksempel gør vi det ved først at sørge for, at amterne i AlleAmter arket bliver sorteret. Baseret på denne sorterede liste, sorterer vi nu ved at anvende **Move** metoden sammen med **After** argumentet derefter de enkelte amts specifikke ark. Lad os se på det:

```
Sub Worksheets4()
' Denne sub sorterer alle amter i neg. alfabetisk rækkefølge
' Først sorteres alle amter i AlleAmter arket, derefter anvendes denne rækkefølge.
    Dim Sht1 As String, Sht2 As String, cell As Range
    With Worksheets("AlleAmter")
        .Range("A1").Sort Key1:=.Range("A1"), Order1:=xlDescending, Header:=xlYes
        With .Range("A1")
            Range(.Offset(1, 0), .End(xlDown)).Name = "Amter"
        End With
    End With
    Sht1 = "AlleAmter"
    For Each cell In Range("Amter")
        Sht2 = cell.Value
        Worksheets(Sht2).Move after:=Worksheets(Sht1)
        Sht1 = Sht2
    Next
    MsgBox "Nu er alle amter sorteret."
End Sub
```

Læg her mærke til **For Each** løkken. Initialt er `Sht1` AlleAmter arket og efterfølgende peger `Sht1` altid på det nuværende ark i den alfabetiske orden. `Sht2` er navnet på det næste ark i den alfabetiske rækkefølge, som jo skal flyttes hen efter (umiddelbart til højre for) `Sht1` arket. Efter flytningen sættes `Sht1` til at pege på det netop flyttede ark, og proceduren gentages. Du kan se, hvad der sker, hvis du definerer nogle Watches for `Sht1` og `Sht2` variablene. For at gøre det skal du stille dig i sub'en og vælge Debug/Add Watch menuen og indskrive `Sht1.` Derefter skal du gentage det hele og skrive `Sht2`. Når du så eksekverer sub'en med F8, kan du se, hvorledes `Sht1` og `Sht2` løbende skifter indhold, og især at `Sht1` får `Sht2`'s indhold.

7.5 Eksempler på Chart objektet i VBA

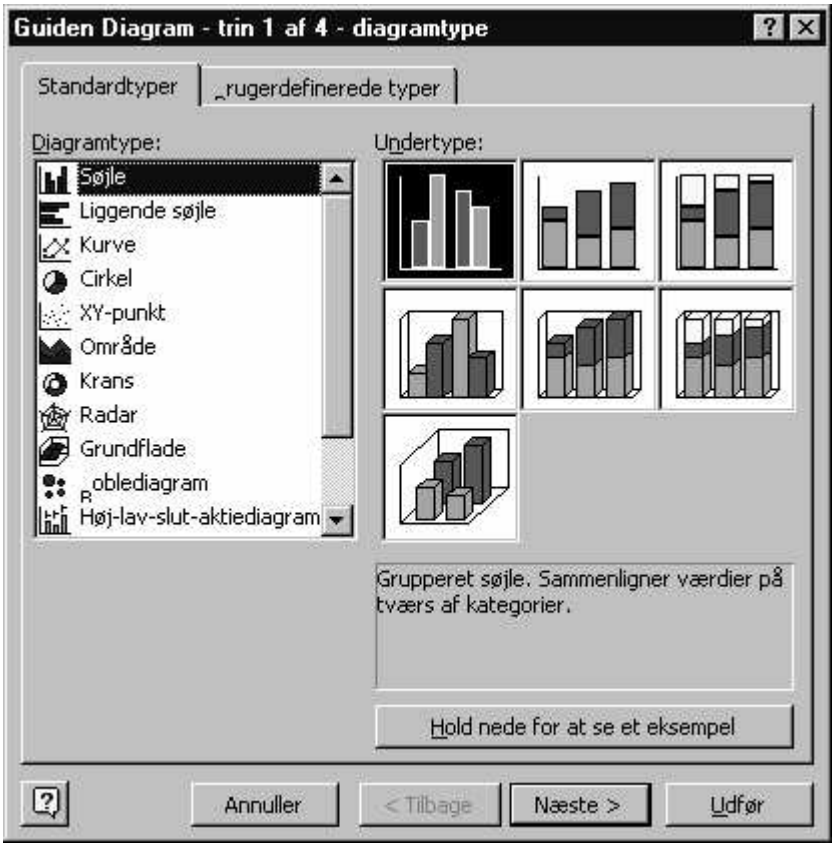
Chart objektet er helt klart det mest besværlige Excel objekt, der kan kontrolleres fra VBA. Grunden er de mange objekter, der er knyttet til **Chart** objektet, og hver af disse tilknyttede objekter har igen mange egenskaber og mange metoder. En rimelig god fremgangsmåde er derfor, som vi så, at optage det meste af grafgenereringen og derefter indarbejde den resulterende makro i din egen VBA kode. De følgende subrutiner giver dig et vink om nogle af mulighederne. Andre muligheder finder du selvfølgelig i den meget omfattende online hjælp,

som følger med Excel. I følgende eksempel anvender vi nedenstående ark, kaldet Salgsark1, som beskriver det månedlige salg, i stk., for 12 forskellige produkter.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Måned\Produkt	1	2	3	4	5	6	7	8	9	10	11	12
2	jan-00	635	430	660	539	697	666	511	629	381	464	349	551
3	feb-00	593	639	472	565	413	580	403	696	466	344	467	408
4	mar-00	579	655	581	397	462	488	339	642	385	470	381	400
5	apr-00	414	596	623	443	483	436	686	668	549	553	599	655
6	maj-00	687	543	323	482	623	400	683	321	471	338	684	472
7	jun-00	681	495	625	359	404	699	588	470	514	386	641	648
8	jul-00	400	346	571	497	361	475	323	326	584	452	679	410
9	aug-00	571	440	663	395	647	410	608	303	682	592	446	483
10	sep-00	449	423	330	312	463	527	695	660	671	331	562	607
11	okt-00	311	348	665	626	306	376	392	423	414	542	316	317
12	nov-00	515	338	349	631	350	675	640	398	598	613	468	678
13	dec-00	301	324	636	580	505	510	426	389	422	421	442	316
14	jan-01	580	623	526	680	466	509	553	358	392	387	532	457
15	feb-01	419	409	500	477	362	494	310	538	414	322	626	516
16	mar-01	674	445	435	403	340	441	595	578	563	531	380	440
17	apr-01	327	466	541	369	629	452	527	686	542	680	483	682
18	maj-01	667	679	439	443	615	553	474	348	634	596	624	365
19	jun-01	412	559	671	516	698	467	341	686	499	415	391	581
20	jul-01	532	648	621	520	372	585	340	639	500	375	479	574
21	aug-01	348	469	369	686	501	321	481	340	360	572	678	647
22	sep-01	308	435	673	666	389	348	428	519	555	402	506	450
23	okt-01	630	698	690	628	663	620	420	615	364	677	565	512
24	nov-01	486	659	550	691	531	680	508	389	618	648	422	450
25	dec-01	566	360	473	307	492	531	540	340	303	601	447	384
26	jan-02	399	636	532	315	593	587	573	396	572	333	694	574
27	feb-02	439	514	670	658	605	678	579	471	693	599	553	406
28	mar-02	331	603	694	648	644	402	566	391	685	389	520	692
29	apr-02	613	648	486	454	378	509	307	465	686	537	644	634
30	maj-02	442	383	412	460	383	623	680	315	505	470	397	410

Som omtalt, er det meget nemmere at lave grafer vha. **Chart Wizard** hjælpeprogrammet end at anvende VBA til at generere grafer, og derfor vil vi anvende **Chart Wizard** til at generere en graf over disse tal. Den hurtige måde at generere en graf på er :

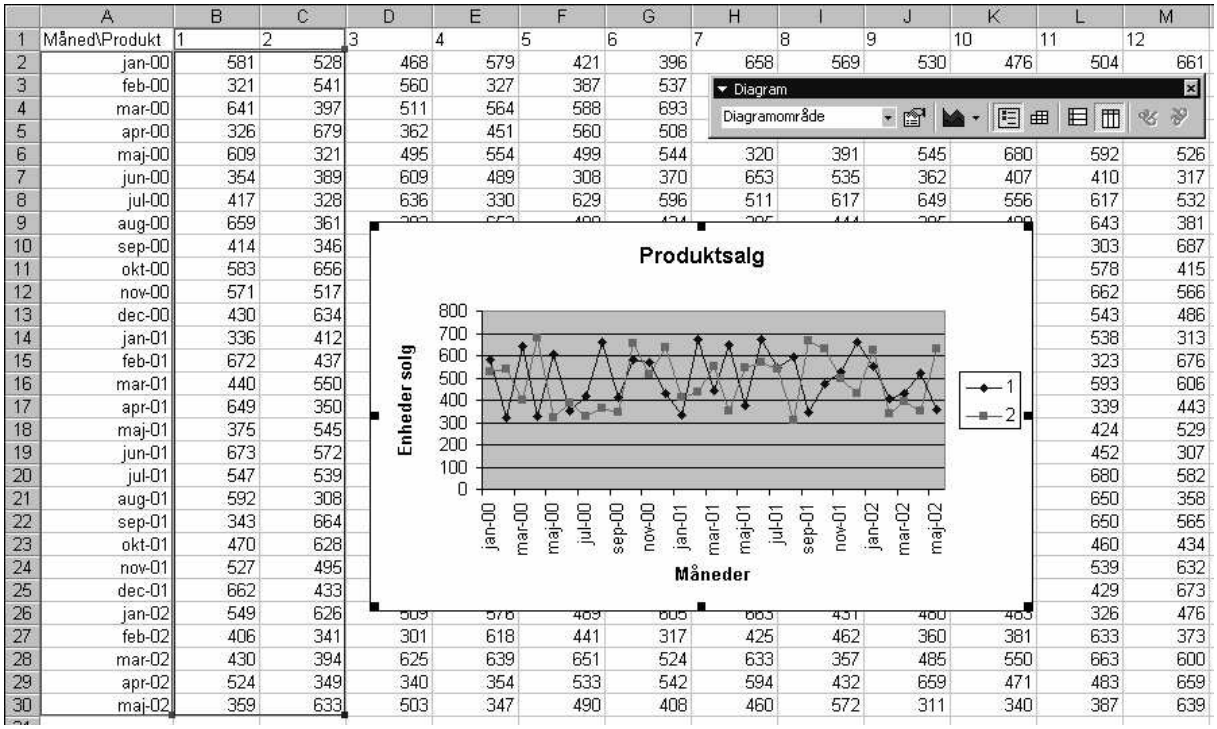
- Vælg området (her kan du trykke Ctrl-A for at markere det hele).
- Tryk på **Chart Wizard** knappen.  hvorefter du får følgende menu:



vælg for eksempel Kurve med datamærker, og tryk næste.

- Vælg Dataområdet. Da du har trykket Ctrl – A vil der stå =Ark1!$A$1:$M$30. Jeg har for overskuelighedens skyld valgt kun at få en graf over de to først produkter =Ark1!$A$1:$C$30. Når du trykker næste, får du mulighed for at indskrive bla. titlen, x- og y -akse labeler og sætte en masse andre ting.

- Når du har trykket næste, skal du til sidst angive hvor grafen skal placeres. Den kan enten placeres i sit eget **ChartSheet** - et selvstændigt ark eller som i nedenstående figur indhyllet i et eksisterende ark.



Måned\Produkt	1	2	3	4	5	6	7	8	9	10	11	12
jan-00	581	528	468	579	421	396	658	569	530	476	504	661
feb-00	321	541	560	327	387	537						
mar-00	641	397	511	564	588	693						
apr-00	326	679	362	451	560	508						
maj-00	609	321	495	554	499	544	320	391	545	680	592	526
jun-00	354	389	609	489	308	370	653	535	362	407	410	317
jul-00	417	328	636	330	629	596	511	617	649	556	617	532
aug-00	659	361									643	381
sep-00	414	346									303	687
okt-00	583	656									578	415
nov-00	571	517									662	566
dec-00	430	634									543	486
jan-01	336	412									538	313
feb-01	672	437									323	676
mar-01	440	550									593	606
apr-01	649	350									339	443
maj-01	375	545									424	529
jun-01	673	572									452	307
jul-01	547	539									680	582
aug-01	592	308									650	358
sep-01	343	664									650	565
okt-01	470	628									460	434
nov-01	527	495									539	632
dec-01	662	433									429	673
jan-02	549	626									326	476
feb-02	406	341	301	618	441	317	425	462	360	381	633	373
mar-02	430	394	625	639	651	524	633	357	485	550	663	600
apr-02	524	349	340	354	533	542	594	432	659	471	483	659
maj-02	359	633	503	347	490	408	460	572	311	340	387	639

Hvis du vælger at oprette et selvstændigt ark, og du har kaldt arket SalgsGraf, kan du i VBA referere til dette som `Charts("Salgsgraf")`, hvor **Charts**, som nu bekendt, er den Collection der holder alle graf arkene.

Hvis du derimod vælger at anvende den anden metode, skal du referere til arket via det objekt, der holder grafen. Hvis du har valgt at placere grafen på Salgsark1[14], skal du derfor referere til grafen med følgende: `Worksheet("Salgsark1").ChartObjects(1)`. Nu er **Chart** objektet et skridt nede i hierarkiet – lige under **ChartObjects**. **ChartsObjects** objekterne "flyder" med andre ord ovenpå et regnearks celler og har kun til opgave at holde **Chart** objekter. Fordelen ved **ChartsObject** objekterne er, at du kan ændre egenskaberne på den underliggende graf, såsom ændring af akserne, titlen osv. Du skal bare huske, at **ChartsObject** objektet kun har relevans for grafer, som bliver placeret på et regneark.

Visning af grafegenskaberne

Den efterfølgende sub arbejder på vores graf fra før. Grafen har vi jo genereret via **Chart Wizard**'en og det eneste, VBA koden nedenfor udfører, er at vise os egenskaberne for denne.

```
Sub Graf1()
```

[14] antager her at grafen er den eneste graf på arket Salgsark1

```
' Denne sub illustrerer nogle af egenskaberne for grafen. Grafen eksisterer
allerede
    With Worksheets("Salgsark1").ChartObjects(1)
        MsgBox "De næste fire beskeder viser positionerne for grafen."
        MsgBox "Venstre : " & .Left
        MsgBox "Top : " & .Top
        MsgBox "Højde : " & .Height
        MsgBox "Bredde : " & .Width
        MsgBox "De næste beskeder viser nogle andre egenskaber ved grafen."
        With .Chart
            MsgBox "Grafens navn: " & .Name
            MsgBox "Graftype: " & .ChartType
            MsgBox "HasLegend egenskaben: " & .HasLegend
            MsgBox "HasTitle egenskaben: " & .HasTitle
            MsgBox "Titel: " & .ChartTitle.Text
            MsgBox "Antal serier som er vist: " & .SeriesCollection.Count
            MsgBox "Nogle egenskaber for den horizontale akse :"
            With .Axes(xlCategory)
                MsgBox "Formatet på ""tick labels"": " & .TickLabels.NumberFormat
                MsgBox "Titel: " & .AxisTitle.Caption
                MsgBox "Font størrelse på titlen: " & .AxisTitle.Font.Size
            End With
            MsgBox "Nogle egenskaber for den vertikale akse:"
            With .Axes(xlValue)
                MsgBox "Titel: " & .AxisTitle.Caption
                MsgBox "Font størrelse på titlen: " & .AxisTitle.Font.Size
                MsgBox "Minimum skala værdi: " & .MinimumScale
                MsgBox "Maksimum skala værdi: " & .MaximumScale
            End With
        End With
    End With
End Sub
```

Det første, rutinen gør, er at anvende **Left**, **Top**, **Height**, og **Width** egenskaberne for **ChartsObject** til at vise os, hvor grafen er placeret, samt hvor stor grafen er. **Top** er afstanden (i punkter = 1/72 tomme) fra toppen af **ChartsObject** objektet til toppen af første række. **Left** er afstanden fra venstre kant af objektet til venstre side af kolonne A. og **Height** og **Width** er højden og bredden af objektet.

Rutinen anvender herefter **With** konstruktionen (`With .Chart`) til at referere til et antal af **Chart** objektets egenskaber, som jo er indeholdt i **ChartObject**. Du ser sikkert et underligt tal, når du viser **ChartType** egenskaben (65?), som selvfølgelig er et internt indeks for den valgte graftype.

Ændring af grafegenskaber

Vi har set, hvordan vi kan få vist egenskaberne ved en eller anden graf, og i dette eksempel vil vi se på, hvorledes disse egenskaber kan ændres. Vi vil lave en rutine, som kan give brugeren mulighed for at vælge, hvilke to af de 12 serier der skal vises. For at kunne gøre det, skal vi anvende **SeriesCollection** objektet, som er et skridt nede i hierarkiet under **Chart** objektet.

Generelt viser en graf et antal **Series** objekter, benævnt SeriesCollection(1), SeriesCollection(2) osv. Egenskaberne ved hver serie kan ændres vilkårligt, specielt **Value** egenskaben som holder dataranget for serien, **XValues** egenskaben, som holder ranget for værdierne på den horisontale akse, og **Name** egenskaben som er et beskrivende navn for serien som vises i forklaringen. Det første, vi skal have afklaret, er range navnene for de enkelte produkter. Vi vil kalde de enkelte range det de er data for, dvs. Produkt1 for dataene i produkt1 kolonnen, Produkt2 for tallene i produkt2 kolonnen osv. Fordelen er, at vi så kan referere direkte til de enkelte range ved at anvende linien:

```
.Name = Range(”Produkt” & prod1).Cells(1).Offset(-1,0)
```

som anvender `.Cells(1)` til at gå til det første salgstal i det givende produkt range og dernæst anvender `.Offset(-1,0)` til at gå til rækken ovenfor for at finde produktets navn.

```
Sub Graf2()
' Denne sub tillader dig at ændre produkt kolonnerne (to af dem).
    Dim Prod1 As Integer, Prod2 As Integer
    Dim NProducts As Integer, NMonth As Integer
    Application.ScreenUpdating = False
    With Range("a1")
      Range(.Offset(1, 0), .Offset(1, 0).End(xlDown)).Name = "Måned"
      NProducts = Range(.Offset(0, 1), .Offset(0, 1).End(xlToRight)).Columns.Count
      NMonth = Range(.Offset(1, 0), .Offset(1, 0).End(xlDown)).Rows.Count
      For i = 1 To NProducts
         Range(.Offset(1, i), .Offset(NMonth, i)).Name = "Produkt" & .Offset(0, i)
      Next
    End With
    Application.ScreenUpdating = True
    MsgBox "Du kan vælge to fra enhver af de 12 kolonner."
    Prod1 = InputBox("Indtast det første indeks (1 til 12)")
    Prod2 = InputBox("Indtast det andet indeks (1 til 12, ikke " _
        & Prod1 & ")")
' Nu har dataene i kolonnerne allerede et range navn som  Produkt1, Produkt2, osv.
    With Worksheets("Salgsark1").ChartObjects(1).Chart
        With .SeriesCollection(1)
' Values egenskaben angiver data ranget som vises. XValues
' egenskaben x værdierne (i situation, måneder). Name egenskaben
' er navnet på serien (som vises i forklaring). Dette navn findes
' i række 1, lige over den første celle i det pågældende range.
            .Values = Range("Produkt" & Prod1)
            .XValues = Range("Måned")
            .Name = Range("Produkt" & Prod1).Cells(1).Offset(-1, 0)
        End With
        With .SeriesCollection(2)
            .Values = Range("Produkt" & Prod2)
            .Name = Range("Produkt" & Prod2).Cells(1).Offset(-1, 0)
        End With
    End With
End Sub
```

Chart objektet er svært. Der er alt for mange ting, man skal huske. Vejen frem er en kombination imellem optageren og online hjælpen og ikke mindst en god portion tålmodighed.

7.6 Opsummering

Dette kapitel har bygget videre på din viden omkring **Range** objektet. Det er i mange situationer vigtigt at kunne manipulere med **Workbooks**, **Worksheets** og **Charts** med VBA, og dette kapitel har illustreret nogle af de mest anvendelige aspekter til netop dette. Det er ikke særlig vigtigt, at du på nuværende tidspunkt kan det hele på rygraden. Det er mere vigtigt, at du på nuværende tidspunkt har en føling med, hvad der egentlig kan lade sig gøre i VBA, resten skal nok komme hen ad vejen.

7.7 Opgaver

1) Åben en ny projektmappe og insæt et nyt modul. Skriv en subrutine, som udfører følgende:
   a) Åbner en projektmappe som, du ved, eksisterer på dit drev.
   b) Viser antallet af ark i denne projektmappe.
   c) Viser en beskedboks med indholdet "Navnet på denne projektmappe er _, og den blev konstrueret af _", hvor _ er ting, som skal udfyldes ved at kalde bestemte egenskaber for ActiveWorkbook og/eller Application objekterne.
   d) Lukker projektmappen igen
   e) Prøver at åbne en projektmappe, som du ved ikke findes
      i) Hvad sker der egentlig, når du prøver at åbne en ikke eksisterende projektmappe?
2) I filen Byer_Kap7.xls filen findes et ark, AlleByer, som indeholder de byer, hvor et firma p.t. opererer. Skriv en subrutine som udfører følgende:
   a) Sorterer alle byerne, så de står i alfabetisk orden
   b) For hver by, undersøger om der eksisterer et selvstændigt ark med bynavnet. Hvis der ikke gør, skal et sådant oprettes.
   c) Slet ethvert ark, som har et navn, der ikke findes i listen af bynavne, men som findes i arket AlleByer.
   d) Programmet skal skrives, så hver eneste ændring i AlleByer listen, enten tilføjelse af en by eller en sletning af en by, kan håndteres.

3) I filen ProduktInfo.xls finder du arket AlleProdukter. Dette ark indeholder enhedspriser for forskellige software pakker, et e-firma sælger. Hvert produkt tilhører en katagori, som er listet i kolonne E. Du skal nu skrive en subrutine som:

   a) opretter et selvstændigt ark for hver katagori, som er repræsenteret i listen af programmer. Dette ark skal du give navnet på katagorien f.eks. "Java Compilers".

   b) kopierer informationerne, der står i kolonnerne: varenummer, beskrivelse og pris i AlleProdukter arket over til de respektive katagoriark, begyndende i række 4 kolonne A.

   c) skriver overskrifter i hvert katagoriark på formen: "Programmer i Java Compilers katagorien" i celle A1

   d) skriver overskrifterne: "Varenummer", "Beskrivelse", og "Pris" i cellerne A3, B3 og C3 og formaterer kolonnebredderne, som specificeret i AlleProdukter arket.

   (Bemærk, at der p.t. er et endeligt antal katagorier. Du skal dog skrive programmet, så det kan håndtere, at der tilføjes flere katagorier. Bemærk også den vedlagte subrutine Shuffle() som bytter om på rækkerne tilfældigt. Selv om dette program bliver kørt, skal dit program stadig kunne håndtere ovenstående)

4) Alleprodukter arket i filen Produktkøb.xls indeholder informationer om forskellige software produkter, som et mail-ordre firma sælger. Filen indeholder også et Faktura ark. Når fimaet modtager en ordre fra en kunde, skal medarbejderen indtaste navnet på kunden, købsdatoen og hvilket antal af hvert produkt kunden ønsker. Disse mængder indtastes i kolonne C i AlleProdukter arket. Disse informationer anvendes til at lave en faktura for kunden i arket faktura. Det nuværende ark er egentlig en skabelon for en generel faktura. Du skal nu skrive to subrutiner, som udfører følgende:

   a) SletData subrutinen skal slette alle mængder der måtte stå i kolonne C i AllProdukter arket. Rutinen skal også slette alle data fra og med række 5 og ned i Faktura arket.

   b) GenFak subrutinen skal køres umiddelbart efter, at medarbejderen har fået kundens navn, indtastet datoen og skrevet mængderne i kolonne C i AlleProdukter arket. Det første rutinen, skal spørge efter, er kundens navn og datoen (denne må gerne autogenereres). Disse informationer skal skrives i cellerne B1 og B2 i Faktura arket. Herefter skal rutinen transferere alle relevante data skrevet i kolonne C (altså hvor

der er positive mængder) til Faktura arket. I Faktura arket skal enhedpriser, mængder, pris ekskl. moms og inkl. moms skrives i de rette kolonner. Endelig, skal alle køb summeres, og det total momsbeløb skrives så det ligner følgende:

	A	B	C	D	E
1	**Kundenavn:**	Tor Beltov			
2	**Dato:**		22-03-2002		
3					
4	**Produkt**	**Enhedspris**	**Mængde**	**Pris**	**Incl. Moms**
5	Adobe PhotoShop 5.5 for Windows 95/NT	kr 4.199,65	2	kr 8.399,30	kr 10.499,13
6	Adobe PhotoShop 5.5 Upgrade for Windows 95/NT	kr 1.259,65	3	kr 3.778,95	kr 4.723,69
7	Corel WordPerfect Office 2000 Professional	kr 2.309,65	4	kr 9.238,60	kr 11.548,25
8	Corel WordPerfect Office 2000 Professional Upgrade/Trade-Up	kr 1.259,65	5	kr 6.298,25	kr 7.872,81
9					
10			Totalt	kr 27.715,10	kr 34.643,88
11		hvoraf moms udgør			kr 6.928,78
12					

c) Som det sidste kan du jo prøve at tilføje udprintningsmuligheder ved at skrive nogle ekstra linier til GenFak rutinen. (Jeg ved, at jeg ikke har skrevet noget om det her i dette kapitel, men du skulle gerne være i stand til at finde ud af hvordan – enten ved at anvende optageren eller online hjælpen)

5) I filen SalgsGrafFin.xls er der nogle månedlige salgstal og fire trykknapper. Range navnene for produktdataene er hhv. Product1 og Product2. For at vide hvad det egentlig er, du skal lave i denne opgave, skal du åbne filen og trykke på de forskellige trykknapper. Problemet er, at koden er kodeordsbeskyttet og derfor ikke kan redigeres. Jeg har lavet graferne osv. i filen SalgsGraf – vil du lave koden, så knapperne virker som de skal? (Koden, som skal laves, er svær! Du skal sikkert bruge onlinehjælpen et par gange ☺)