# 12.bölüm

**12**

# İleri Seviye Konular

Bu bölümde Nesne kavramına daha yakında bakacağız. Shape ve OleObjects nesnelerini detaylı göreceğiz. Pivot tablo, Slicer ve Chart gibi yönetimi zor olan konuları bir arada göreceğiz. Ve son olarak yazdığım kodların bir menüden nasıl servis edileceğini göreceğiz.

## Nesneler Dünyası

VBA konularının başında nesne kavramına biraz girmiş ve Excel'in nesne modelinden bahsetmiştik. O bölümü okumayanların önce orayı okumasını tavsiye ederim.

Bu bölümde ise nesnelere biraz daha yakından bakacağız. Burada Türkçe'nin avantajını kullanarak bu kavrama Türkçe adıyla hitap edeceğim, zira bir de değişken tipi olarak **Object** var elimizde. İkisi birbirine karışmasın diye genel nesne kavramını **Nesne** olarak, tip olanı ise **Object** olarak belirteceğim.

### Giriş

#### Nedir?

Nesnelerin ne olduğuna bakmadan önce nesnelerin ne olmadıklarına bakalım. Basit değişkenlerle nesnelerin birbirinden çok temel bir farkı vardır. Basit değişkenlerin tek bir amacı vardır: Bir değer depolamak.

Dim i As Integer

Dim ad As String

i=10

ad="Volkan"

Nesneler ise, bir değer depolamaktan daha fazlasını yaparlar. Nesneler, hem çoklu veri tutarlar hem de bir eylem icra ederler.

Dim ws As Worksheet

Set ws = Activesheet

ws.Name="Kredi" 'veri

Debug.Print ws.Index 'veri

ws.Add 'eylem

#### Nesnelerin bileşenleri

Artık bildiğiniz üzere, Excel'de herşey bir nesnedir, hatta nesneler topluluğu olan collectionlar(sonu "s" ile biten) da nesnedir(Ör: Workbook da nesne, Workbook**s** collection'ı da).

Bu nesnelerin bazıları sadece veri tutarlar, bu verilere özellik(property) denir, bazı nesneler belirli eylemleri(metod) de icra ederler. Bazıları ise ayrıca kendileriyle ilgili bir eylem olduğunda bir olay(event) meydana getirirler.

Nesne kavramı olmasaydı her şeyi değişkenlerle yönetmemiz gerekirdi ki bu inanılmaz karmaşık bir dünyaya neden olurdu. Düşünsenize, 30 propertysi olan bir obje içi 30 ayrı değişken tanımlamanız gerekirdi, ki bu sadece değişken tanımlamayla ilgili endişemiz, diğer dezavantajlarını saymıyorum bile.

**NOT**: VBA, tam anlamıyla bir Nesne Yönelimli Programlama(OOP) dili değildir, ama bu kavramı destekler. Excelent eklentisini yazdığım dil olan VB.NET ise tam bir OOP dilidir.

##### Property'ler

Bazı propertyler Readonly'dir(salt okunur), yani bunlara değer atayamazsınız, bazıları ise hem okunabilir hem yazılabilirdir.

MsgBox Activecell.Address 'bu readonlydir

ActiveCell.Value = 1 'bu hem okunur hem yazılabilirdir.

MsgBox ActiveCell.Value

Property'lere değer atamak, eğer dönüş değerleri basit data tiplerindeyse, aynı bunlar gibi atanır. i=1 ile ActiveCell.Value = 1 örneğindeki gibi

Ancak dönüş değeri nesne olan propertylere nesne atamalarındaki **Set** ifadesi ile atama yaparız.

Set ilkkolon = ActiveCell.CurrentRegion.Columns(1)

##### Metodlar

Metodlar, nesnelerin eylem icra eden üyeleridir. Sub olarak da Function olarak da tanımlanmış olabilirler. Mesela Workbook nesnesinin Add metodu bir Function'dır, zira geriye bir Workbook nesnesi döndürür. Ancak Save metodu bir Sub'dır, zira geriye birşey döndürmez, sadece bir eylem icra eder.

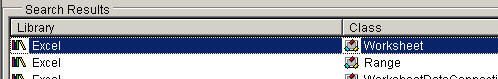
##### Eventler

Belli nesnelerin de belirli eylemler olduğunda meydana gelen olayları vardır. Bu konuyu olaylar bölümünde genişçe ele almıştık.

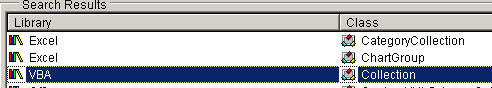
#### Nesne türleri

Ben nesneleri 4 gruba ayırıyorum(bu sınıflama tamamen bana aittir, resmi bir gruplama değildir)

**İlk** grupta Excel’in nesne modelindeki nesneler var. Range, Cell, Worksheet gibi. Bunlar Excel librarysinde yer alırlar. Tümüne MSDN’den([https://msdn.microsoft.com/en-us/VBA/Excel-VBA/articles/object-model-Excel-VBA-reference](https://msdn.microsoft.com/en-us/VBA/Excel-VBA/articles/object-model-excel-vba-reference)) ulaşabilirsiniz



**İkinci** grupta, default gelen librarylerdeki nesneler bulunuyor. Yani ilave herhangi bir library'yi reference olarak eklemeden yaratacağımız nesneler. Collectiongibi; bu nesne VBA librarysinde bulunur.



**Üçüncü** grupta, bir library ekleyerek yaratılan nesneler var. Dictionary, FileSystemObject gibi. Bunlar da Scripting Runtime library'sinde bulunurlar.



Excel'in nesne modeli dışında kendi nesnelerimizi de yaratabiliriz, bunlar da **dördüncü** grup oluyor. Tabi bunun için önce bu nesnenin taslağını oluşturan Class yaratmamız gerekiyor. Classlardan bir sonraki konuda kısaca bahsedeceğiz.

#### Nesne üyelerine erişmek

##### Klasik yöntem

Bir nesne üyesine erişmek için en bilinen yol, nesne adını yazıp sonra "." koymaktır. Ör: Workook.Name

##### With - End With

Bir diğer yöntem de daha önce gördüğümüz **With** kalıbı. Bu kalıbı **Excel Nesne Modeli** konusunda anlatmıştık, sadece kısa bir örnek verelim.

Dim fd As FileDialog

Set fd = Application.FileDialog(msoFileDialogFolderPicker)

With fd

.Title = "Klasör seçin"

If .Show = True Then

lblKlasör.Caption = .SelectedItems(1)

End If

End With

##### Me

Bir class modülündeyken(workbook, worksheet modülleri dahil, veya kendinize ait bir class) kendisine Me ifadesi ile erişebiliyorsunuz. Burada unutulmaması gereken şudur; Me, her zaman o an içinde bulunulan classa başvurur. Ör: Thisworkbook modülündeyken: Me.Save

### Nesne tanımlama ve yaratma

#### Genel tanımlama ve yaratım teknikleri

##### New ifadesi

Excel nesne modelinde bulunan nesneleri tanımlarken **New** ifadesini kullanmayız. Zira Excel açıldığında bunlar otomatikman yaratılmış olurlar, o yüzden sadece değişken tanımlamak yeterlidir.

Bunlara atama yapmak için ise **Set** ifadesini kullanıyoruz.

Dim rng As Range

Dim ws As Worksheet

Set rng = Activecell

Set ws = Activesheet

Excel nesne modeli dışındaki nesneleri yaratmak için ise **New** ifadesini kullanmak zorundayız. Bu şekilde nesne yaratmanın da iki yolu vardır.

###### Yöntem 1: Tek satırda

İlk yöntemde Dim ve New ifadelerini aynı satırda kullanırız, yani **tanımlama** ve **yaratım** aynı anda olur. Sonra nesnenin üyelerini hemen kullanmaya başlayabiliriz. Aslında yaratım aynı anda olmamaktadır, bu nesne ilk nerede görülürse işte o sırada yaratım olmaktadır.

Dim coll As New Collection

coll.Add "elma"

###### Yöntem 2: Ayrı satırlarda(Set'li yöntem)

Bu yöntemde ise tanımlama ile yaratım&atama ayrı satırlarda gerçekleşir. Tanımlama Dim ile, yaratım&atama Set ve New ile yapılır.

Dim coll As Collection 'Tanımlama

Set coll = New Collection 'Set ile atama, New ile yaratım

###### İstisna

Eğer ki, elde edeceğimiz nesneyi bir fonksiyon veya metod ile elde edeceksek o zaman New ifadesi kullanılmaz.

Dim oApp As New Outlook.Application 'bunda New gerekli

Dim oMail As Outlook.Mailitem 'bunda New kullanılmaz

Set oMail = oApp.CreateItem(0) 'başka bir nesnenin metoduyla elde ettik

Başka bir örnek de veritabanı işlemlerinden olsun. Aşağıdaki iki nesne için de New gerekmedi.

Dim db As Database

Dim rs As Recordset

Set db = OpenDatabase(....)

Set rs = db.OpenRecordset(.....)

###### Hangi yöntem ne zaman kullanılır?

İkinci yöntem performans yönetimi açısından tercih edilir. Aslında günümüz bilgisayarları açısından bakıldığında buradaki performans etkisi artık ihmal edilebilir düzeydedir. O yüzden iki yöntem de kullanılabilir. Ancak belli özel durumlarda ikinci yöntemin kullanılması tavsiye edilir.

Eğer, tanımladığımız değişkeni **belli bir duruma/şarta** göre yaratmamız söz konusu ise tek satırda değil, iki satırda yani Set'li yöntemle tanımlarız.

Mesela mail gönderimi yapılacak bir durum düşünelim. Eğer B2 hücresinde bir değer varsa o zaman mail gönderilsin, yoksa gönderilmesin. O yüzden ilk başta tanımlamayı yapalım, ama henüz nesneyi yaratmamıza gerek yok.

Dim oApp As Outlook.Application

If Not IsEmpty(Range("F2")) Then

Set oApp = New Outlook.Application

End If

Burada, ilgili nesne bir değişkene atanana kadar onu yaratmaz. Diğer yöntemle farkını görmek için aşağıdaki iki kodu çalıştırıp kendiniz görün, gerçi ben yazılacak değerleri yanlarında belirttim, ama kendinizin de görmesinde fayda var.(Outlook librarysini eklemeyi unutmayın)

Sub setliyöntem()

Dim oApp As Outlook.Application

Dim coll As Collection

Debug.Print TypeName(coll) 'Nothing

Debug.Print coll.Count 'hata

Debug.Print TypeName(oApp) 'Nothing

Debug.Print oApp 'hata alınır

If Not IsEmpty(Range("F2")) Then

Set oApp = New Outlook.Application

End If

End Sub

'-------------------

Sub teksatıryöntemi()

Dim oApp As New Outlook.Application

Dim coll As New Collection

Debug.Print TypeName(coll) 'collection

Debug.Print coll.Count '0

Debug.Print TypeName(oApp) 'Application

Debug.Print oApp 'Outlook

End Sub

Tek satırda kullanım yönteminin avantajı ise, ilgili değişkeni Nothing ile yok etseniz bile tekrar kullanabiliyor olmanızdır. Mesela şu kod problemsiz çalışır.

Sub coll1()

Dim coll As New Collection

coll.Add "Apple"

Set coll = Nothing

coll.Add "Pear" 'yeni bir collection yaratılır

End Sub

Ancak aynı kodu Setli yönteme çalıştırırsak hata alırız.

Sub coll2()

Dim coll As Collection

Set coll = New Collection

coll.Add "Apple"

Set coll = Nothing

coll.Add "Pear" 'hata

End Sub

#### Early ve Late Binding

Bir diğer nesne yaratım yöntemi ise **CreateObject** ile yaratımdır, ki buna **LateBinding** yöntemi ile yaratım denir.

Bu yönteme daha çok, VBA'nin default libraryleri olan VBA, Excel ve Office libraryleri dışındaki librarylerde bulunan classlardan nesne yaratmak istediğimizde başvururuz. En sık kullanılan classlar şunlardır:

* Scripting.Runtime librarysi içindeki **FileSystemObject**
* Yine Scripting.Runtime librarysi içindeki **Dictionary**
* Outlook, Word gibi diğer **Ofis uygulamaları**

Syntax'ı şu şekildedir: **CreateObject("library.class")**

Değişken tanımlamayı **Object** tipli yapıp sonra da Set ile atama ve yaratmayı yaparız.

Dim obj As Object 'tanımlama

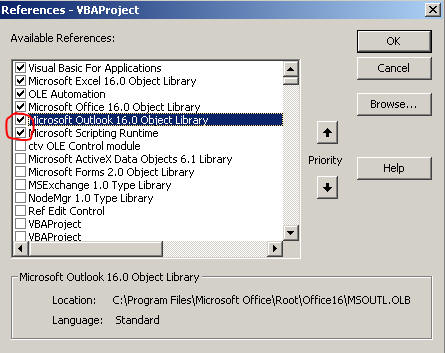
Set obj = CreateObject("Outlook.Application") 'yaratma ve atama

Set obj = CreateObject("Scripting.Dictionary")

Set obj = CreateObject("Scripting.FileSystemObject")

NOT: Nasıl olsa LateBinding yapıyorum, o yüzden değişken tanımlamaya gerek yok diye düşünmeyin. Zira object tipli değişkenler hafızda 4 byte yer işgal ederken, değişken tanımlamadığınız durumda bunlar otomatikman Variant algılanacakları için hafızada 16 byte işgal ederler.

**Early Binding**'te ise tanımlamak istediğimiz nesnenin bulunduğu library'yi **Tools>References** menüsünden eklememiz gerekir.



Library'yi ekledikten sonra artık klasik yoldan değişken tanımlayabiliriz, nesneyi doğru yazma konusunda Intellisense bize yardımcı olur.



veya bunu 2 satırlık versiyonla da yapabiliriz.

Dim dict As Dictionary

Set dict = New Dictionary

Bu örnekte olduğu gibi aynı classtan başka bir library içinde yoksa library adını yazmamıza gerek yoktur, aksi halde karışıklık olur. Böyle bir durumda library adını da belirtmeliyiz.(Veritabanı işlemlerinde bu durumu görüyoruz)



##### Early Binding vs Late Binding

* Late Binding’in avantajı, yazdığınız kodu bir başkasına gönderdiğinizde(veya network üzerinden çalıştırdıklarında) onda da kesinlikle sorunsuz çalışacağını biliyor olmanızdır. Zira herhangi bir kütüphane eklenmesi durumu olmadığı için versiyon farkları problem yaratmayacaktır. Early Binding’te ise siz Office 2016 ile çalışıyorken, diğer kişi Office 2013 ile çalışıyorsa sizin eklediğiniz Outlook 16.0 reference'ini bulamayacağı için hata alacaktır. O yüzden eğer yaptığınız çalışmayı başka birinin bilgisayarında çalıştırma durumu varsa Late Binding kullanmanız faydalı olacaktır, aksi halde Early Binding kullanın.
* Early Binding’le çalışmanın avantajı ise intellisense’ten faydalanmaktır. Bu hem daha hızlı hem de hatasız kod yazmanızı sağlayacaktır. Late Binding'de yazdığınız kod Intellisense ile teyit edilmemiş olacağı için tam çalışmadan önce birkaç kez derleme hatası almanız olasıdır.
* Early Binding’le çalışmanın bir diğer faydası da kodun çalışma hızıdır. VBA, ilgili nesnenin ne olduğunu direkt bileceği için arka planda bir dönüştürme işlemi yapmasına gerek kalmayacak ve de kod hızlı çalışacaktır. Zaten Early denmesinin sebebi de budur, **öncende/erkenden** hangi classla çalışacağımıza karar vermiş oluyoruz. Late Bindingte ise **sonradan/gecikmeli** bir tespit işlemi de olacağı için kod daha yavaş çalışacaktır. Ortalama hız farkı 2 kattır.
* Bazı nesneler Late Bindingle ile yaratılamaz. Mesela CreateObject("DAO.Database") gibi bir kullanım söz konusu değildir. Bu yöntemin kullanımı için ilgili nesnelerle ilgili bazı teknik ön ayarların yapılmış olması gereklidir. Şu sayfalardan teknik detaylara ulaşabilirsiniz:
* [https://docs.microsoft.com/en-us/office/VBA/language/reference/user-interface-help/activex-component-can-t-create-object-or-return-reference-to-this-object-error-4](https://docs.microsoft.com/en-us/office/vba/language/reference/user-interface-help/activex-component-can-t-create-object-or-return-reference-to-this-object-error-4)
* <https://support.microsoft.com/en-us/help/828550/you-receive-run-time-error-429-when-you-automate-office-applications>
* <https://support.microsoft.com/en-us/help/245115/using-early-binding-and-late-binding-in-automation>

Benim nihai önerim şudur: Intellisense ve performans sebepleriyle her zaman Early Bindingle başlayın, başkasının bu dosyayı çalıştırması da söz konusu olacaksa, Early binding olan yerleri late bindinge çevirin.

**NOT**: Bir de **GetObject** diye bir fonksiyon vardır. "Mevcutta açık olan bir uygulama varken **CreateObject** ile o nesneyi tekrar yaratmanın anlamı yok, ona GetObject ile ulaşabilirsiniz" amacıyla vardır. Günümüz bilgisayarlarındaki bellek kapasitesi düşünüldüğünde çok gereği olmayan bir fonksiyondur. Yine de görürseniz şaşırmayın diye bahsetmek istedim.

### Hafızada neler oluyor

#### Hafıza adresleri

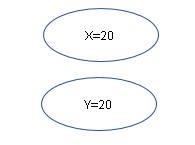
Basit veri tipleri söz konusu olduğunda bunlar için hafızada ayrı yerler açılır. Mesela aşağıdaki örnekte X ve Y için hafızada iki ayrı alan açılır.

Dim X As Integer, Y As Integer

X = 20

Y = 20

Bellek gösterimini ise aşağıdaki gibi yapabiliriz. İki değişken de aynı değere sahip olduğu halde iki farklı alan işgal edilir: 2şer byte'tan toplam 4 byte.



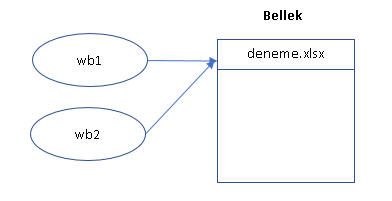
Nesnelerde ise durum biraz farklıdır. Nesneler söz konusu oluğunda değişkenlerde nesnenin kendisi değil, nesnenin işaret ettiği adres depolanır. Buna programlamada Pointer denir.

Aşağıdaki örnekte bellekte sadece bir alan işgal edilir, o da Object tipli değişkenlerin değeri 4 byte olduğu için toplam 4 byte'tır, 8 byte değil. Zira iki değişken de hafızadaki aynı yere işaret ediyorlar.

Dim wb1 As Workbook, wb2 As Workbook

Set wb1 = Workbooks("deneme.xlsx")

Set wb2 = wb1



Bu da şu demek oluyor; **nesne değişkeni ile nesnenin kendisi farklı şeylerdir**. Aşağıdaki örnekte **VarPtr** değişkenlerin bellekteki adresini verirken, **ObjPtr** nesnelerin kendisinin adresini verir. Adresten kastımız, uzunca bir sayıdır, bu sayının ne olduğu önemli değildir, önemli olan içeriğidir, commentlere bakınız.

Sub hafıza()

Dim wb1 As Workbook

Dim wb2 As Workbook

Set wb1 = ActiveWorkbook

Set wb2 = wb1

Debug.Print wb1.Sheets.Count '1

Debug.Print wb2.Sheets.Count '1

wb1.Sheets.Add

Debug.Print wb1.Sheets.Count '2

Debug.Print wb2.Sheets.Count '2

'wb1 ve wb2 "değişkenlerinin" adresi

Debug.Print "wb1 nesne değişkeninin adresi: " & VarPtr(wb1) 'aşağıdaki ile farklı

Debug.Print "wb2 nesne değişkeninin adresi: " & VarPtr(wb2)

'wb1 ve wb2'nin işaret ettiği yerin adresi

Debug.Print "wb1'in adresi: " & ObjPtr(wb1) 'aşağıdaki ile aynı

Debug.Print "wb2'nin adresi: " & ObjPtr(wb2)

End Sub

#### Hafızayı temizleme

##### Otomatik ama gecikmeli temizlik

Bir değişkene bir nesne atadıktan sonra tekrar başka bir nesne atarsak, artık ilk nesne var olmaz ve bir süre sonra bellekten silinir.

Mesela aşağıdaki örnekte, 1'den 10'a kadar sayıları tutan collection son satırdan itibaren yok olur ve ona erişmenin hiçbir yolu kalmaz.(Bu işlem hemen değil biraz gecikmeli olur)

Dim coll As Collection

Set coll = New Collection

For i = 1 to 10

coll.Add i

Next i

Set coll = New Collection 'ilk nesne yok olur

Burda aslında biz bellekte iki tane collection için yer açtık ama son satıra geldiğimizde artık ilkine hiçbir şey atanmış olmadığı için **Garbage Collector** denen sistem bir süre sonra bunu bellekten atar, yani özetle bir nesneye hiçbir değişken işaret etmiyorsa bu nesne bellekten silinir.

##### Manuel ama anında temizlik

Bu yöntemi şimdiye kadar birçok örnekte gördük aslında. Bir değişkene **Nothing** değerini atayınca o değişkenle onun başvurduğu nesne arasındaki ilişkiyi kopartırız.

Aslında çoğu durumda bu işlem gerekli değildir, zira yukarda gördüğümüz gibi bir nesneye başvuran bir değişken kalmadığında bu nesne bellkten gecikmeli de olsa otomatikman silinir.

Ancak bazı durumlarda, özellikle döngüsel işlemlerde Nothing ataması gerekebilir. Çünkü Garbage Collector'ın(GC) ne zaman devreye gireceği belli değildir, ve biz belleği **hemen** **boşaltmak** istiyorsak işte o zaman Nothing ataması yaparız.

Mesela toplu mail gönderiminde kullandığımız aşağıdaki koda bakalım. oMail değişkenine Nothing atamak faydalıdır, zira bunu yapmazsak ilk mail nesnesi hala bir süre daha bellekte kalmaya devam edecek, ta ki GC gelip onu yok edene kadar. Eğer tek seferde yüzlerce mail atacaksanız bu işlemi yapmanızı şiddetle öneririm, aksi halde bellekte yüzlerce mail nesnesi birikebilir ve işlem bellek yetersizliğinden yarıda kesilebilir.

Bununla beraber son satırdaki oApp değişkenine Nothing ataması çok da kritik değildir. Ben yine de alışkanlıkla bunu yapmayı tercih ediyorum. yorum.

Sub çoklumail\_Button1\_Click()

Dim oApp As Outlook.Application

Dim oMail As Outlook.MailItem

Dim alıcılar As Range, a As Range

Set oApp = New Outlook.Application

Set alıcılar = Range(Range("A2"), Range("A2").End(xlDown))

For Each a In alıcılar

Set oMail = oApp.CreateItem(olMailItem)

With oMail

.Subject = "Doğum günü"

.To = a.Value

.Body = a.Offset(0, 3).Value & "Doğum gününüz kutlar, ailenizle birlikte mutlu yıllar dilerim"

.Body = .Body & vbCrLf & "Gönderenin adı soyadı"

.Send

End With

Set oMail = Nothing 'zorunlu değil ama faydalı

Next a

Set oApp = Nothing 'zorunla da değil kritik faydası da yok

End Sub

## Class ve Class Modüller

Bu sayfada kendim birşey yazmak yerine sizi çeşitli sitelere yönlendirmek istedim.

Açıkçası 20 küsur yıllık VBA geçmişimde çok fazla class yaratma gereği duymadım(aşağıdaki istisna hariç). Class yaratmayı gerektirecek bir durumdaysanız size tavsiyem bu işi VBA'de değil, daha gelişmiş bir programlama dilinde yapmanız. Ayrıca çözümünüz aslında bir veritabanı erişimi veya dizimsilerden birinin kullanımı veya başka bir şekilde çözülebilecek birşey de olabilir.

Benim açıkçası class yarattığım tek durum, **custom event** yaratmak için olmuştur. Bunlara Olaylar bölümünde bakabilirsiniz.

Classlarla ilgili web kaynağı önerilerim aşağıdaki gibidir:

* [https://Excelmacromastery.com/VBA-class-modules/](https://excelmacromastery.com/vba-class-modules/) sayfasında çok detaylı bir anlatım sunuluyor.
* Bir diğer detay anlatım olan sayfaya [http://www.globaliconnect.com/Excel/index.php?option=com\_content&view=article&id=160:Excel-VBA-custom-classes-a-objects-class-modules-custom-events&catid=79&Itemid=475](http://www.globaliconnect.com/excel/index.php?option=com_content&view=article&id=160:excel-vba-custom-classes-a-objects-class-modules-custom-events&catid=79&Itemid=475) asdresinden ulaşabilirsiniz.
* <http://www.cpearson.com/Excel/Classes.aspx>
* [http://www.ExcelVBA.net/viewtopic.php?f=130&t=4522](http://www.excelvba.net/viewtopic.php?f=130&t=4522) adresli Türkçe sayfada ise bir soruna güzel bir çözüm sunulmuş.

## Shape(s)'ler ve OleObject(s)'ler

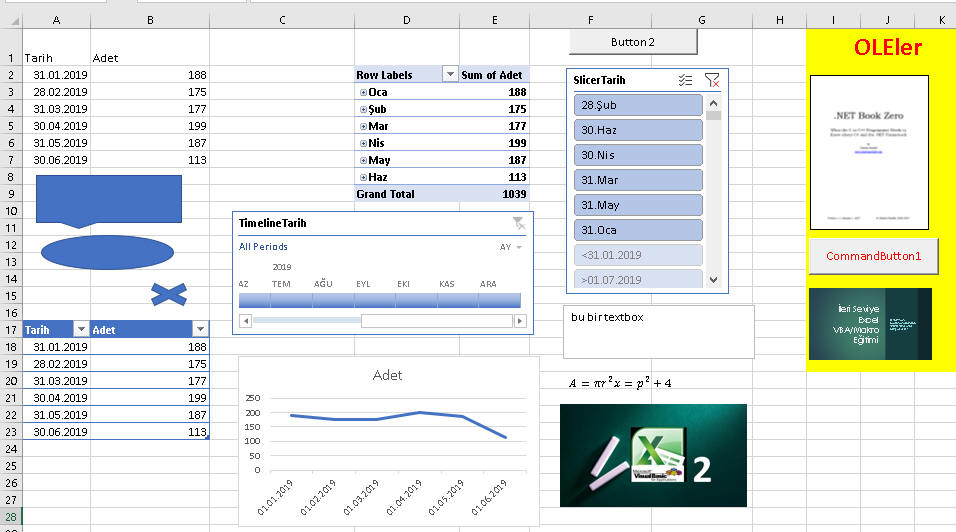
Geldik, en kafa karıştıran konulardan birine. Bu bölümde, **Shape**(sayfa içi nesneler) ve onların bir alt grubu olan **OleObject** kavramlarına değineceğiz.

Özellikle, **dashboard** tasarlayanlar, çok fazla **ActiveX** veya form kontrolü kullananlar, Slicer/Chart gibi nesnelerin görünürlük ve konum bilgilerini dinamik olarak yönetmek isteyenler için önemli olmakla birlikte çalışmalarına görsellik katmak isteyen herkesin bilmesinde fayda olan bir konudur.

### Giriş

Burada bu nesnelerin sayfa düzeni, sitil özellikleri, görünürlük gibi "şekle özgü" özelliklerine bakacağız. Özellikle kafanızın karışmaması gerek nokta şudur: AutoShape dışındaki şekillerin nesne modelindeki özelliklerine burada girmeyeceğiz. Örneğin bir Slicer'la ilgili seçim yapma, filtreleri kaldırma gibi özelliklerden ziyade Slicer'ın sayfanın neresinde konumlandırılacağı, gösterilip gösterilmeyeceği gibi özellikler kapsamımızda olacaktır. Bu nesnelerin, nesne modeline ait konuları kendilerine ait sayfalarda bulunacaktır.

Şimdi, elimizde aşağıdaki şekilleri içeren bir sayfa olduğunu düşünün. İsterseniz bunları içeren dosyayı **githubdan(shapeler.xlsm)** indirebilirsiniz.



Burada hemen her türden nesne(Grafik, Slicer, TextBox, Konuşma balonu, Resim, gömülü pdf dosyası, Form Buton, ActiveX button) var. PivotTable ve Table dışında buradaki herşey bir **Shape** nesnesidir ve bunlar doğal olarak **Shapes** collection'ının bir üyesidir.

Aslında bir range'e yayılmamış olan herşey bir Shape'tir diye düşünebilirsiniz. Bu bağlamda, bir range bölgesi olan Table ve PivotTable'lar Shape olmamakta. Bunlar sırasıyla **ListObjects** ve **PivotTables** collectionlarının üyeleridir.

**OleObject**'ler de Shape'lerin bir alt türüdür, **yani her OleObject aynı zamanda bir Shape'tir.** Bu yukarıdaki örnekte; ActiveX commandbutonu,  gömülü pdf dosyası ve linkli powerpoint sunumu. Zaten bunlara tıkladığımızda fonksiyon çubuğunda ya **EMBED** ile başlayan bir ifade görürüz: =EMBED("Forms.CommandButton.1";"") ve =EMBED("Acrobat Document";"") ya da Link adresi, "=PowerPoint.Slide.12|'C: \Users\Volkan\Videos\Movavi Screen Capture Studio\Udemy Kurslar\2-ileri VBA-makro\dosyalar\İLERİ EXCEL VBA(MAKRO) EĞİTİMİ - Giriş.pptx'!'!265'"

Şimdi bunlara yakından bakalım.

### Shapes Collection'ı ve Shape Nesnesi

Yukarıda belirttiğimiz gibi, bir Range'ten ziyade sayfa üzerinde ayrı bir nesne gibi duran ve mouse ile seçildiğinde köşelerinde ve kenar ortalarında küçük yuvarlaklar çıkaran herşey Shape'tir.

Shape'lerle ilgili önemli birkaç özellik/metod aşağıdaki gibidir.

Activesheet.Shapes.SelectAll 'sayfadaki tüm shapeleri seçer

Activesheet.Shapes.Count 'sayfadaki shapelerin sayısını verir

Activesheet.Shapes.Addxxx 'xxx yerine şekil tipi gelir, Ör: AddOLEObject

#### Shape'lere erişim

Tüm diğer collection(workbooks, worksheets v.s) tiplerinde olduğu gibi shape'lerde de indeks(indeks no veya isim) ile tekil shape'lere ulaşabiliyor ve sonra bunlara ait özellikler veya metodları kullanabiliyoruz.

Activesheet.Shapes(i).Delete 'i. şekili siler

ActiveSheet.Shapes("Button 1").Visible = msoFalse 'Button 1 isimli shape grünmez yapar

Tüm shapelerde dolaşmak için aşağıdaki gibi bir döngü kullanabiliriz.

Sub shapelerde\_dolaş()

Dim şekil As Shape

For Each şekil In ActiveSheet.Shapes

Debug.Print şekil.Name

Next şekil

End Sub

**Type** property'si ile de ilgili şeklin tipinin enumeration değeri döner. Bunlara ait değerleri MSDN’den(<https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2007/aa432678(v=office.12)>) görebilirsiniz. Aşağıda en sık kullanılanlara ait bir tabloyu bulabilirsiniz.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Value | Description | Örnek |
| msoShapeTypeMixed | -2 | Mixed shape type | TimeLine |
| msoAutoShape | 1 | AutoShape | Diktörtgen |
| msoChart | 3 | Chart | Grafik |
| msoEmbeddedOLEObject | 7 | Embedded OLE obj. | Gömü PDF |
| msoLinkedOLEObject | 10 | Linked OLE object | Linkli PowerPoint |
| msoOLEControlObject | 12 | OLE control object | ActiveX CommandButotn |
| msoPicture | 13 | Picture | Resim |
| msoTextBox | 17 | Text box |  |
| msoSlicer | 25 | Slicer | Slicer |

Bu Type özelliği, özellikle If ile kontrol yapıp sadece belirli şekillerle işlem yapmak istediğimizde kullanışlıdır. Mesela aşağıdaki kod ile sayfadaki tüm grafikleri silebiliriz.

Sub shapleerde\_dolaş()

Dim şekil As Shape

For Each şekil In ActiveSheet.Shapes

If şekil.Type = 3 Then 'veya msoChart

şekil.Delete

End If

Next şekil  
End Sub

Aşağıdaki kod ile de grafik ve comment dışındaki tüm şekilleri siliyoruz.

If şekil.Type <> msoChart And şekil.Type <> msoComment Then şekil.Delete

### OleObjects Collection'ı ve OleObject Nesnesi

Insert menüsünden "Object" olarak eklenenler ve Developer menüsünden eklenen **ActiveX** objeleri **OleObject** olarak görünür. ActiveX dışı OleObjeler sayfaya gömülü veya linkli olurlar. **OLEType** özelliği ile ilgili nesnenin linkli mi yoksa gömülü mü olduğu tespit edilebilir. Alacağı değerler şöyledir:

* Linkli: 0(xlOLELink),
* Gömülü: 1(xlOLEEmbed) ve
* ActiveX Kontrol: 2(xlOleControl).

#### Nesnelere Erişim

Bunlarda da tekil objelere yine indeks ile ulaşırız.

Activesheet.OLEObjects("ListBox1").Delete 'veya indexno

Yine döngüyle tüm oleobjeleri dolaşalım:

Sub oleobjelerdedolaş()

Dim oleo As OLEObject

For Each oleo In ActiveSheet.OLEObjects

Debug.Print oleo.Name

Next oleo

End Sub

Collectionlarla toplu işlemler de gerçekleştirebiliriz.

Activesheet.OLEObjects.Visible = False 'hepsini gizler

Sayfaya dinamik olarak OleObje eklemek de mümkündür.

Worksheets(1).OLEObjects.Add FileName: ="arcade.gif" 'gömülü gif dosyası

Worksheets(1).OLEObjects.Add ClassType: ="Forms.ListBox.1" 'ActiveX kontrolü

#### Nesnelerin Özelliklerine erişim

##### Gömülü/Linkli nesneler

Gömülü/Linkli öğelerde daha çok Visible ve Top/Left gibi konum özellikleriyle ilgileneceğiz.

Gömülü bir pdf dosyasındaki özelliklere erişime bir bakalım.

ActiveSheet.OLEObjects("Ole1\_embed\_PDF").Visible = True

Bu özelliklerin bir kısmına Shapes collection'ı üzerinden de ulaşabiliriz. Ne de olsa tüm OleObjectler aynı zamanda bir Shape'tir. Enabled gibi bazı özellikler Shape class'ında bulunmadığı için bunları kullanamayız, dolayısıyla mecburen OleObject nesnesini kullanırız.

ActiveSheet.Shapes("Ole1\_embed\_PDF").Visible = msoFalse 'False yerine msoFalse

'ama

ActiveSheet.OLEObjects("Ole1\_embed\_PDF").Enabled = True

Özellikler dışında bir de Activate ve Verb gibi metodlarla da ilgilenebiliriz. Mesela bir pdf dokümanına tıklandığında onu açtıracak kodu aşağıdaki gibi yazabiliriz.

Sub Ole1\_embed\_PDF\_Click()

ActiveSheet.OLEObjects("Ole1\_embed\_PDF").Verb xlOpen

End Sub

##### ActiveX kontrolleri

ActiveX kontrollerinde ise durum biraz karışıktır. Bu case'de OleObjectler, içinde bu kontrolü barındıran bir sarmalayıcı(wrapper) objeden oluşur. Bazı durumlarda sarmalayıcının özelliklerini kullanmak bazen içteki nesneyi bazen de ikisini kullanmak gerekebilir. Yine bunların bir kısmına Shapes collection'ı üzerinden de ulaşabiliriz.

Bu iç kısımdaki esas objeye ulaşmak için OLEObject nesnesinin **Object** propertysi kullanılır, veya buna Shape nesnesi üzerinden ulaşıyorsak önce OleObject'yi elde etmek için OLEFormat'ı, sonra da bunun Object property'si kullanılır. Özetle aşağıdaki ifadeler özdeştir:

Shapes("Ole1\_embed\_PDF").OLEFormat.Object ile OLEObjects("Ole1\_embed\_PDF") 'sarmalayıcı oleobject nesnesi

Shapes("Ole1\_embed\_PDF").OLEFormat.Object.Object ile OLEObjects("Ole1\_embed\_PDF").Object 'iç kısımdaki kontrol

### Shape ve OleObject bir arada

Tüm OleObject'lerin aynı zamanda bir Shape de olduğunu söylemiştik. Peki bir Shape döngüsü içindeyken veya bir şekilde elimizde bir shape nesnesi varken bunların OleObect özelliklerine erişmek istersek ne yaparız? Öncelikle **OLEFormat** özelliğini kullanırız. Bu bize bir OleFormat nesnesi döndürür, bu nesnenin de **Object** özelliğini kullanarak OleObject nesnesine erişiriz. Bu nesne bir ActiveX nesnesi ise ilk Object property'siyle sarmalayıcı nesneye erişmiş oluruz. OleObjectin sarmaladığı içteki esas kontrole ulaşmak için ise bir **Object** propertysi daha kullanırız. Evet çok karışık oldu, farkındayım, şimdi hemen kodlara bakalım, sonra açıklamayı tekrar okuyalım.

Mesela yukarıdaki pdf dokümanının Click eventine aşağıdaki kodu da yazabilirdik.

Sub Ole1\_embed\_PDF\_Click()

ActiveSheet.Shapes("Ole1\_embed\_PDF").OLEFormat.Object.Verb xlOpen

End Sub

Veya tüm Shapelerde dolaşırken OleObject olanların isim bilgisini öğrenmek için aşağıdaki kodu yazabiliriz.

For Each şekil In ActiveSheet.Shapes

If TypeName(şekil.OLEFormat.Object) = "OLEObject" Then

Debug.Print şekil.OLEFormat.Object.Name

End If

Next şekil

İç kısımdaki objeye ulaştıktan sonra bu sefer onun özelliklerine erişebilirsiniz. Maalesef adından anlaşılacağı üzere bu bir Object olduğu için intellisense çıkmamaktadır. Instellisense’ten yaralanmak isterseniz ilgili tipte bir nesne tanımlamanız gerekir. Aşağıdaki örnekte sayfada duran iki adet CommandButtonun Caption özelliklerine ulaşıyorum, tabi bunların Caption özelliğine sahip olduklarını bildiğimiz için intellisense çıkmamış olsa bile ezberden yazabiliyoruz.

Sub oledetay()

Dim şekil As Shape

Dim oo As OLEObject

For Each şekil In ActiveSheet.Shapes

Set oo = şekil.OLEFormat.Object

Debug.Print oo.Name, oo.OLEType, TypeName(oo), TypeName(oo.Object), oo.Object.Caption, oo.Left

Next şekil

End Sub

Sonuç:

CommandButton1 2 OLEObject CommandButton Düğme1 66

CommandButton2 2 OLEObject CommandButton Düğme2 150,6

Dikkat ettiyseniz sarmalayıcının Type'ı OLEObject iken iç nesnelerin CommandButton çıkıyor.

#### Özelliklere erişim yöntemleri

Şimdi buraya kadar öğrendiklerimizi iyice pekiştirmek adına 3 tür erişim şekline bakalım.

##### Gömülü nesne(PDF dokümanı, Word veya PowerPoint dokümanı v.s) için Visible özelliği

###### Shape üzerinden

ActiveSheet.Shapes("Ole1\_embed\_PDF").Visible = msoFalse

###### OleObject üzerinden

Buna da istersek OleObject nesnesi üzerinden doğrudan veya Shape üzerinden dolaylı olarak erişebiliriz.

ActiveSheet.OLEObjects("Ole1\_embed\_PDF").Visible = True 'Doğrudan

'veya

ActiveSheet.Shapes("Ole1\_embed\_PDF").OLEFormat.Object.Visible = True 'Dolaylı

##### ActiveX kontrolü ile Visibile özelliğine erişim

###### Shape üzerinden

ActiveSheet.Shapes("Ole2\_embed\_ActiveXcmdbuton").Visible = msoFalse

###### OleObject üzerinden

Buna da istersek OleObject nesnesi üzerinden doğrudan veya Shape üzerinden dolaylı olarak erişebiliriz.

ActiveSheet.OLEObjects("Ole2\_embed\_ActiveXcmdbuton").Visible = True 'Doğrudan

'veya

ActiveSheet.Shapes("Ole2\_embed\_ActiveXcmdbuton").OLEFormat.Object.Visible = True 'Dolaylı

NOT: **Visible** özelliği **sadece sarmalayıcı obje** için söz konusudur, dolayısıyla içteki Obje'ye ulaşarak bu nesnenin Visible özelliğine değer atayamayız.

##### ActiveX kontrolü ile Enabled özelliğine erişim

###### Shape üzerinden

Shape'in bu özelliği bulunmamaktadır.

###### OleObject üzerinden

Buna da istersek OleObject nesnesi üzerinden doğrudan veya Shape üzerinden dolaylı olarak erişebiliriz.

ActiveSheet.OLEObjects("Ole2\_embed\_ActiveXcmdbuton").Enabled = False 'Sarmalayıcı nesneye doğrudan

'veya

ActiveSheet.Shapes("Ole2\_embed\_ActiveXcmdbuton").OLEFormat.Object.Enabled = False 'Sarmalayıcı nesneye dolaylı

Ayrıca Enabled özelliği, içteki esas kontrol için de bulunduğu için içteki nesne üzerinden de ulaşabiliriz.

ActiveSheet.OLEObjects("Ole2\_embed\_ActiveXcmdbuton").Object.Enabled = True 'İç nesneye doğrudan

'veya

ActiveSheet.Shapes("Ole2\_embed\_ActiveXcmdbuton").OLEFormat.Object.Object.Enabled = True 'İç nesneye dolaylı

##### ActiveX kontrolü ile FontSize özelliğine erişim

###### Shape üzerinden

Shape'in bu özelliği bulunmamaktadır.

###### OleObject üzerinden

Yukarıdaki Visible özelliğinde karşılaştığımız durumun tersine Font'un Size bilgisi gibi bilgilere sadece **iç nesne** aracılığı ile ulaşılmaktadır. Dış nesneden bu özelliğe erişim yoktur.

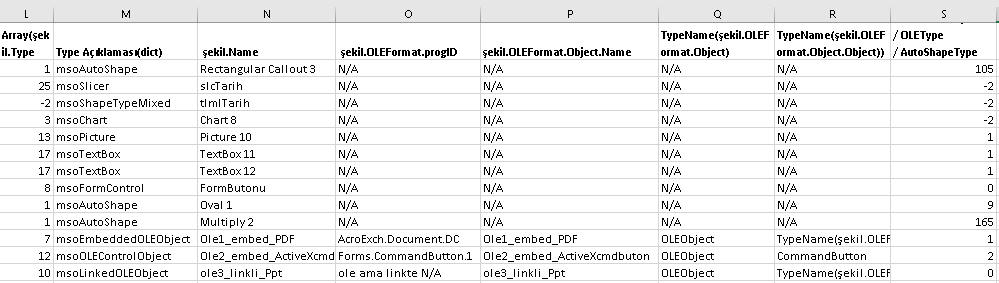
ActiveSheet.OLEObjects("Ole2\_embed\_ActiveXcmdbuton").Object.Font.Size = 11 'Doğrudan

'veya

ActiveSheet.Shapes("Ole2\_embed\_ActiveXcmdbuton").OLEFormat.Object.Object.Font.Size = 14 'Dolaylı

#### Özelliklere erişimle ilgili örnek

Örnek dosyamız üzerindeki tüm shape'lerin çeşitli bilgilerini aşağıdaki gibi yazdırmak için bir kod yazalım.



Başlıkları manuel yazdığımızı düşünecek olursak, L2 hücresine konumlanıp aşağıdaki kodu çalıştırınca bu çıktıyı elde ederiz.

Sub shapleerde\_dolaş()

Dim şekil As Shape

Dim dict As New Dictionary

'OLEType,FormControlType ve AutoShapeType'lar da dictionary yapılarak isimleri yazdırılabilir

'OleType basit: (0,1,2: Linkli, Embedded, Control)

'https: //docs.microsoft.com/en-us/office/VBA/api/Excel.xlformcontrol,

'https: //docs.microsoft.com/en-us/office/VBA/api/office.msoautoshapetype

dict.Add 1, "msoAutoShape"

dict.Add 2, "msoCallout"

dict.Add 20, "msoCanvas"

dict.Add 3, "msoChart"

dict.Add 4, "msoComment"

dict.Add 27, "msoContentApp"

dict.Add 21, "msoDiagram"

dict.Add 7, "msoEmbeddedOLEObject"

dict.Add 8, "msoFormControl"

dict.Add 5, "msoFreeform"

dict.Add 28, "msoGraphic"

dict.Add 6, "msoGroup"

dict.Add 24, "msoIgxGraphic"

dict.Add 22, "msoInk"

dict.Add 23, "msoInkComment"

dict.Add 9, "msoLine"

dict.Add 29, "msoLinkedGraphic"

dict.Add 10, "msoLinkedOLEObject"

dict.Add 11, "msoLinkedPicture"

dict.Add 16, "msoMedia"

dict.Add 12, "msoOLEControlObject"

dict.Add 13, "msoPicture"

dict.Add 14, "msoPlaceholder"

dict.Add 18, "msoScriptAnchor"

dict.Add -2, "msoShapeTypeMixed"

dict.Add 19, "msoTable"

dict.Add 17, "msoTextBox"

dict.Add 15, "msoTextEffect"

dict.Add 26, "msoWebVideo"

dict.Add 25, "msoSlicer"

For Each şekil In ActiveSheet.Shapes

'ilk 3ü ole

If şekil.Type = msoEmbeddedOLEObject Then

Dizi = Array(şekil.Type, dict(şekil.Type), şekil.Name, şekil.OLEFormat.progID, şekil.OLEFormat.Object.Name, TypeName(şekil.OLEFormat.Object), "TypeName(şekil.OLEFormat.Object.Object)", şekil.OLEFormat.Object.OLEType)

ElseIf şekil.Type = msoLinkedOLEObject Then

Dizi = Array(şekil.Type, dict(şekil.Type), şekil.Name, "ole ama linkte N/A", şekil.OLEFormat.Object.Name, TypeName(şekil.OLEFormat.Object), "TypeName(şekil.OLEFormat.Object.Object)", şekil.OLEFormat.Object.OLEType)

ElseIf şekil.Type = msoOLEControlObject Then

Dizi = Array(şekil.Type, dict(şekil.Type), şekil.Name, şekil.OLEFormat.progID, şekil.OLEFormat.Object.Name, TypeName(şekil.OLEFormat.Object), TypeName(şekil.OLEFormat.Object.Object), şekil.OLEFormat.Object.OLEType)

'form butonu

ElseIf şekil.Type = msoFormControl Then

Dizi = Array(şekil.Type, dict(şekil.Type), şekil.Name, "N/A", "N/A", "N/A", "N/A", şekil.FormControlType)

'diğer hepsi

Else

Dizi = Array(şekil.Type, dict(şekil.Type), şekil.Name, "N/A", "N/A", "N/A", "N/A", şekil.AutoShapeType)

End If

Range(ActiveCell, ActiveCell.Offset(0, 7)).Value = Dizi

ActiveCell.Offset(1, 0).Select

Next şekil

End Sub

|  |  |  |
| --- | --- | --- |
|  |  |  |

## PivotTable, Slicer ve Chart nesneleriyle çalışmak

### Pivot Tablolar

#### Giriş

Pivot(Özet) tablolarla çalışmaya başlamadan önce Excel tarafında konuya hakim olmak önemlidir. Buraya kadar geldiğinize göre konuyu bildiğinizi varsayıyorum.

Baştan belirtmekte fayda var; Pivot tablo(bundan sonra PT diye anılacak) konusu VBA tarafında biraz alengirli bir hal alır, zira Excel önyüzde görmediğimiz bir nesneyle çalışmamız gerekmektedir. Bunun adı PivotCachedir. Onun dışında aslında recorder ile üreteceğiniz birkaç kodu elden geçirmekle kolaylıkla öğrenebileceğiniz bir konudur. O yüzden ben bu sayfada gereksiz yere recorderla elde edebileceğiniz detayları koymamaya karar verdim. Onun yerine refresh işlemleri, birkaç önemli işlem, birkaç event ve sonrasında da slicerlarla olan ilişkisini anlatmak istiyorum. Tabiki otomasyon projelerinizde data üretimi sonrasında kodun bir PT da üretmesi gerekiyorsa PT yaratmayı bilmeniz lazım ama dediğim gibi bunu recorderla çok rahat görebilirsiniz, gereksiz fazlalıkları atmak size kalıyor.

Size tavsiyem öncelikle birkaç işlemi(pivot oluşturma, filtre koyma, filtre kaldırma, repeat all items yapma v.s) recorder ile yapın ve kodu inceleyin. Yine de "ben recorderla uğraşamam diyorsanız MSDN'den detay bilgi edinebilirsiniz. İngilizceniz yoksa ve pivot detaylarını yine de yazmamı isterseniz bu sayfanın altına yorum bırakın, uygun bir zamanda bunları da eklerim, ama dediğim gibi gerçekten burada anlatacaklarım dışında nerdeyse her şeyi recorderla kolaylıkla elde edebilirsiniz ve bunlarla ilgili olarak dikkat çekecek bir husus bulunmamaktadır.

#### Veri Seti

Üzerinde çalışacağımız data seti, Excel konularındaki örnek data setidir, **githubdan(pivotdata.xlsb)** indirebilirsiniz.

Mesela aşağıda, bu dosyanın "calculatedlar" sayfasındaki dataya uyguladığımız PT'ın recorderla elde edilmiş kodu bulunuyor. Tabi bu kodu şuan çalıştırsanız sizde çalışmaz zira sayfa isimleri değişmiş olacak. Onun yerine kendiniz yeni bir recorder çalıştırıp görün derim.

Sub Macro1()

'

' Macro1 Macro

'

'

Sheets.Add

ActiveWorkbook.PivotCaches.Create(SourceType: =xlDatabase, SourceData: = \_

"calculatedlar!R1C1: R175C6", Version: =6).CreatePivotTable TableDestination \_

: ="Sheet1!R3C1", TableName: ="PivotTable1", DefaultVersion: =6

Sheets("Sheet1").Select

Cells(3, 1).Select

With ActiveSheet.PivotTables("PivotTable1")

.ColumnGrand = True

.HasAutoFormat = True

.DisplayErrorString = False

.DisplayNullString = True

.EnableDrilldown = True

.ErrorString = ""

.MergeLabels = False

.NullString = ""

.PageFieldOrder = 2

.PageFieldWrapCount = 0

.PreserveFormatting = True

.RowGrand = True

.SaveData = True

.PrintTitles = False

.RepeatItemsOnEachPrintedPage = True

.TotalsAnnotation = False

.CompactRowIndent = 1

.InGridDropZones = False

.DisplayFieldCaptions = True

.DisplayMemberPropertyTooltips = False

.DisplayContextTooltips = True

.ShowDrillIndicators = True

.PrintDrillIndicators = False

.AllowMultipleFilters = False

.SortUsingCustomLists = True

.FieldListSortAscending = False

.ShowValuesRow = False

.CalculatedMembersInFilters = False

.RowAxisLayout xlCompactRow

End With

With ActiveSheet.PivotTables("PivotTable1").PivotCache

.RefreshOnFileOpen = False

.MissingItemsLimit = xlMissingItemsDefault

End With

ActiveSheet.PivotTables("PivotTable1").RepeatAllLabels xlRepeatLabels

With ActiveSheet.PivotTables("PivotTable1").PivotFields("KANAL")

.Orientation = xlRowField

.Position = 1

End With

With ActiveSheet.PivotTables("PivotTable1").PivotFields("URUN")

.Orientation = xlColumnField

.Position = 1

End With

ActiveSheet.PivotTables("PivotTable1").AddDataField ActiveSheet.PivotTables( \_

"PivotTable1").PivotFields("TUTAR"), "Sum of TUTAR", xlSum

End Sub

Yukarıda da bahsettiğim gibi recorder kodunda gereksiz bir sürü kısım var, zira artık bildiğiniz üzere recorder bir nesneye ait tüm propertylere değer atıyor.

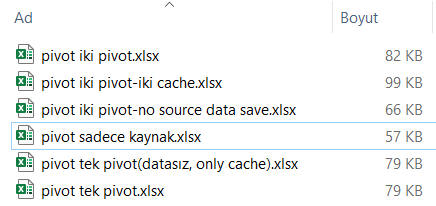
#### PivotCache nesnesi

Yukarıdaki koddan göreceğiniz üzere bir pivot table yaratılmadan önce PivotCache(Bundan sorna PC olarak anılacak) nesnesi yaratılıyor. **PC, veri kaynağının bellekte özetlenmiş halidir**. Her PT'ın arka planda dayandığı bir PC vardır. Neden böyle bir şeye ihtiyaç var diye sorabilirsiniz. PT'lar bence ve birçok kişi tarafından Excel'in en güçlü aracı olarak kabul edilir. Bunların gücü, birçok gruplama işini çok hızlı yapmasında yatıyor. İşte bu hız, bellekteki PC tarafından sağlanır, aksi halde her defasında veri kaynağına gitmek gerekecekti ki bu da hızın düşüşü anlamına gelecekti. İşte bu yüzden öncelikle PC yaratılır.

Excel 2007'den önce bir veri kaynağından üretilen her PT'ın ayrı bir PC'si olurdu, bu da bellek tüketimi için çok iyi olmazdı ve dosya boyutunu da gereksiz yere şişirirdi. Excel 2007 ile birlikte aynı kaynaktan üretilen PT'ların PC'si ortaklaştırıldı. Bununla beraber bazı durumlarda bunların PC'sini ayrıştırmak gerekebilir, bunu nasıl yapacağımızı aşağıda göreceğiz.

Rakamlarla örnek vermek gerekirse; 2007 öncesinde kaynak datanız 1 MB büyüklüğündeyse ve 3 PT yarattıysanız dosya boyutu 3 MB daha artıyordu. 2007 ve sonrasında sadece 1 MB artıyor. Bu 1 MB'dan da tasarruf etmek için "save source data" yapabilirsiniz.

Aşağıda benin üzerinde çalıştığım birkaç dosyanın boyutunu görebilirsiniz(Edit: Dosya adlarındaki baştaki "pivot"u görmezden gelin):



Daha detaylı analiz ettiğimizde;

* Boş bir dosya 9 KB
* Sadece kaynak datayı içerdiğinde 57 KB. Demek ki datanın getirdiği net yük: 57-9=48 KB
* Bu dosyada tek bir PT yaratınca boyut 79'a çıkıyor. Demek ki, cache + pivot sayfasındaki özet data=79-48=31 KB(Gerçi özet alanında hiç veri koymasam da boyut yine 79, demek özet veri ihmal edilebilir düzeyde)
* PT sayısını 2ye çıkarınca boyut 3 KB daha artıp 82 oluyor. İki PT da aynı cacheyi kullandığına göre, bu 3 KB'lık artış sadece ikinci PT'ın özet verisinden geliyor
* İki pivotun cachesini birbirinden ayırırsak boyut 99'a çıkıyor. Demek ki ikinci cache katkısı 99-82=17 KB
* Cacheleri tekrar ortaklaştırıp ancak kaynak datayı dosya ile birlikte kaydetmezsem boyut 66 KB'a düşüyor.

##### Cacheyi tek mi tutmalı çok mu?

Paylaşımlı PC'ler tabiki hafıza kullanımı açısından ve çoklu PT'ların aynı davranışı göstermesi gereken durumlarda sunduğu verimlilik açısından faydalıdır. Ancak bunun arzu edilmediği durumlar olacaktır:

* Mesela bir PT refreshlendiğinde diğerleri refreshlenmesin istiyorsunuzdur
* Bir alan üzerinde gruplama yaptıysanız, bu gruplama tüm PT'larda uygulanacaktır. Halbuki siz bunun sadece birinde olmasını istiyorsunuzdur.
* Keza, bir PT'a calculated item/field koyduğuzuzda tümünde görünecektir. Bunu istemiyorsunuzdur.
* İşte bu sebeplerden dolayı cacheleri ayırmayı bilmek önemlidir. Bunu az sonra göreceğiz. Önce PT ve PC'lere biraz daha yakından bakalım.

##### PC ve PT'larla çalışmak

PC'ler Workbook nesnesinin bir collection'ı iken, PT'lar Worksheet'indir. Yani,  toplam PC sayısını Activeworkbook.PivotCaches.Count diye alabilirken toplam PT sayısını bu şekilde alamayız, her bir sayfada dolaşıp toplam aldırmamız gerekiyor.

Sub pivot\_adetler1()

T = 0

Debug.Print "Toplam pivot cache sayısı: " & ActiveWorkbook.PivotCaches.Count

For Each ws In ActiveWorkbook.Worksheets

T = T + ws.PivotTables.Count

Next ws

Debug.Print "toplam pivottable sayısı: " & T

End Sub

Tabi istersek belli kriterleri sağlayan PT adedini de saydırabiliriz. Mesela adında "Piv" ifadesi geçenleri saydıralım.

Sub pivot\_adetler2()

T = 0

For Each ws In ActiveWorkbook.Worksheets

For Each pt In ws.PivotTables

If InStr(1, pt.Name, "Piv") > 0 Then

T = T + 1

End If

Next pt

Next ws

Debug.Print "toplam pivottable sayısı: " & T

End Sub

PC'ler otomatik üretilen nesneler olduğu için bunlarda isim(name propertysi) olmaz, sadece indeks olur. Ancak PT'ların ismi olabilir.   
PT'lara doğrudan isim veya indeksiyle ulaşılabileceği gibi pivotun içindeki bir hücreyi referans gösterip onun PivotTable propertysi ile de ulaşabilirsiniz.

Tüm PT'lara hangi PC'den beslendikleri ve hangi sayfada olduklarıyla beraber ulaşmak için şu kodu yazabiliriz.

Sub pivot\_ve\_cacheleri\_gör()

For Each ws In ActiveWorkbook.Worksheets

For Each pt In ws.PivotTables

Debug.Print pt.Name, pt.CacheIndex, pt.Parent.Name

Next pt

Next ws

End Sub

Tabi böyle bir kodu çalıştırmanın bir esprisi yok. Bu haliyle olsa olsa test/kontrol yapmakta kullanılır. Onun yerine ihtiyacınıza göre modifiye edip öyle çalıştırırsınız. Ör: Adında "Mevduat" geçen PT'ları refreshlemek gibi.

Bazen de özellikle test/kontrol amacıyla üzerinde çalıştığınız pivottableların indekslerini öğrenmek istersiniz. Malesef Pivottablelar için bir Index/Id gibi property yok, o yüzden aşağıdaki çözümle ilerleyebiliriz.

Sub pivotlarınİndeksleriniGör()

Dim ws As Worksheet

Set ws = ActiveSheet

For i = 1 To ws.PivotTables.Count

Debug.Print i, ws.PivotTables(i).Name

Next i

End Sub

NOT: Bu çözüm sadece bir oturuma ait bir çözüm olabilir. Zira dosyayı sonraki açışlarınızda bu indekslerin değiştiğini görebilirsiniz. O yüzden PT'lara indeks numaralarıyla ulaşmak yerine isimleriyle ulaşmak daha güvenlidir.

#### Refreshleme

Şunu akılda bulundurmak lazım ki, hem PC'ler hem PT'lar refreshlenbilir ama PC'leri refreshlemek her zaman daha hızlıdır, zira bunlar kaynağa doğrudan bağlıyken PT'lar PC'ler üzerinden ikinci seviyeden bağlıdır. Üstelik bir PC'ye bağlı 3 PT varsa döngüsel olarak tümünü refreshlemek anlamsızdır, zira biri refreshlenince zaten hepsi refreshlenir, ancak döngü içinde dönerken mecburen hepsi de refreshlenmek durumunda kalır, ki bu da kodun gereksiz uzun çalışmasına neden olur.

Refresh yöntemleri aşağıdaki gibidir:

##### 1) wb.Refreshall

Workbook\_Open içinde yazıyorsak asenkron çalışma konusuna dikkat edilmelidir. Bu metod ile sadece PT'lar değil tüm connectionlar da refreshlenir. O yüzden bu her zaman tercihiniz olmayabilir.

##### 2) PC.Refresh veya birden çok PC'de döngüsel refresh

ActiveWorkbook.PivotCaches(1).Refresh

'veya

Sub cahceleri\_refreshle()

For Each pc In ActiveWorkbook.PivotCaches

pc.Refresh

Next pc

End Sub

##### 3) PT.RefreshTable veya birden çok PT'da döngüsel refresh

PT'larda döngüsel dolaşılarak tek tek refresh edilir.

Sub ptleri\_refresh()

For Each ws In ActiveWorkbook.Worksheets

For Each pt In ws.PivotTables

pt.RefreshTable

Next pt

Next ws

End Sub

'veya tek bir PT için: ActiveWorkbook.Worksheets.PivotTable(1).RefreshTable

##### 4)Tüm cacheler loop ile RefreshOnFileOpen = True veya bunları oluştururken manuel işaretleme.

#### Paylaşımlı(Ortaklaştırılmış) PC'leri ayırmak:

Biraz yukarıda bahsettiğimiz gerekçelerden ötürü bazen PT'ların PC'lerini ayrıştırmak gerekebilir. Bu, manuel yapılabileceği gibi VBA ile de yapılabilir.

Genelde bu ihtiyaç, rutin değil de tek seferlik bir ihtiyaçtan doğacağı için Excel ön yüzden yapılması yeterlidir ancak otomasyona bağlanmış bir işle her gün PT oluşturuluyor ve bunların cahcelerinin ayrışması gerekiyorsa bu kod faydalı olacaktır.

Sub CacheAyrıştır()

Set pt = ActiveCell.PivotTable

Set ws = Worksheets.Add

ActiveWorkbook.PivotCaches.Create( \_

SourceType: =xlDatabase, \_

SourceData: =pt.SourceData).CreatePivotTable \_

TableDestination: =ws.Range("a3"), \_

TableName: ="GeçiciPT"

pt.CacheIndex = ws.PivotTables(1).CacheIndex

Application.DisplayAlerts = False 'sayfa silerken uyarı çıkmasın diye

ws.Delete

Application.DisplayAlerts = True

End Sub

Bir şekilde ayrıştırılmış bu cacheleri tekrar birleştirmek için de aşağıdaki kod yazılabilir. Mantık şu: Tüm PT'larda dolaşıyoruz, Temel PT'nın cachesiyle aynı kaynağa sahip olanların cachesini Temel PT'nın cachesine eşitliyoruz.

Sub CacheBirleştir()

For Each ws In ActiveWorkbook.Worksheets

For Each pt In ws.PivotTables

If pt.SourceData = Worksheets(1).PivotTables("TemelPivot").SourceData Then

pt.CacheIndex = Worksheets(1).PivotTables("TemelPivot").CacheIndex

End If

Next pt

Next ws

End Sub

#### Nesneler hiyerarşisi

PT ve PC dışında elimizde **PivotField**(s) ve **PivotItem**(s) collectionları/nesneleri var. Bunlardan PivotField PT'ın alt nesnesi, PivotItem da PivotField'ın alt nesnesi oluyor.

Yani itemlara ait bir bilgiyi şu sırayla elde etmeliyiz. PT.PF.PI. Mesela aşağıdaki kod ile aktif sayfadaki 1 numaralı PT'nin Bölge isimli fieldındaki tüm seçili değerleri alırız.

ActiveSheet.PivotTables(1).PivotFields("Bölge").VisibleItems

PivotField olarak belirttiğimiz alanlar RowField olabileceği ColumnField veya DataField da olabilir. Ancak biz spesifik olarak bunlardan birini almak istiyorsak o zaman bunlara ait propertyleri kullanmamız gerekir.(ColumnFields, RowFields vs.)

ActiveSheet.PivotTables(1).ColumnFields(1).Name

Bir de **PivotFilter** nesnesi var, adından anlaşılacağı üzere, belirli fieldlara filtre uygulamaya yarıyor. Bununla ilgili örnekleri de aşağıda göreceğiz.

#### Çeşitli işlemler

Peki biz bu nesnelerle neler yapabiliyoruz, kısaca bir onlara bakalım. Öncelikle şunu belirtmekte fayda var. Recorder ile baktığınızda tıpkı yeni bir PT yaratmada olduğu gibi VBA her bir değer için Seçili/SeçiliDeğil şeklinde bir değer üretir. Biz bunu döngülerle daha pratik yapabileceğiz.

##### PT'daki bir Field'da filtreli elemanları elde etme

Bir field'da seçili olan elemanları elde etmek VisibleItems özelliği ile oldukça kolaydır. Aşağıda, parametre olarak sayfa, PT adı ve field adını alan bir fonksiyonumuz ve bu fonksiyonu çağıran bir test prosedürümüz bulunuyor. Bu arada fonksiyonumuzun dönüş tipi PivotItems olup döngüsel olarak içinde dönerek elemanları yazdırıyoruz.(Dönüş tipini collection olarak da belirleyebilirdik, ancak bu sefer elemanları tek tek geçici bir Collection nesnesine atmak gerekirdi. Bu örneği Slicerdaki seçili elemenları getirme örneğinde yapacağız.)

Function Pivotun1FieldındakileriGetir(ws As Worksheet, ptName As String, ptfield As String) As PivotItems

'tek bir field için basit örnek

Dim pt As PivotTable

Dim pf As PivotField

Set pt = ws.PivotTables(ptName)

Set pf = pt.PivotFields(ptfield)

Set Pivotun1FieldındakileriGetir = pf.VisibleItems

End Function

Sub test\_pivot\_seçili\_elemanlar()

'Örnek olarak, Tarihsl Datanın Pivot ve Slcr sayfasındayken çalıştırabilirsiniz. Pivotta bikaç bölge filtreli olsun

Dim kimlerseçili As PivotItems

Set kimlerseçili = Pivotun1FieldındakileriGetir(ActiveSheet, "ptBolgekodUrun", "Bölge Kodu")

For Each vi In kimlerseçili

Debug.Print vi.Name

Next vi

End Sub

##### PT'daki birden fazla fieldda seçili olan elemanları elde etme

Bu sefer, hangi fieldda hangi değerlere filtre konmuş, bunları elde edeceğiz. Tabi şimdi işimiz biraz daha zor, o yüzden farklı bir bakış açısıyla yaklaşmak lazım. Aslında bu paragrafın ilk cümlesi bize bir dictionary'nin gerekliliğini gösteriyor: Hangi fieldda ne seçili, yani bildiğimiz Key ve Value ikilisi. O yüzden fonksiyonumuzun dönüş tipini bu sefer Dictionary olarak tanımlayacağız.

Tekil field örneğinden farklı olarak bu sefer Field isimlerini bir collection'a atayıp gönderiyoruz. Fonksiyonumuzdan gelen bilgileri yazdırırken de "birleştir" isimli  yardımcı bir fonksiyon ile her fielddaki itemları yanyana birleştirip öyle yazdırıyoruz.

Function PivotunÇokluFieldındakileriGetir(ws As Worksheet, ptName As String, ptfields As Collection) As Dictionary

Dim pt As PivotTable

Dim pf As PivotField

Dim dict As New Dictionary

Set pt = ws.PivotTables(ptName)

For Each p In ptfields

Set pf = pt.PivotFields(p)

dict.Add p, pf.VisibleItems

Next p

Set PivotunÇokluFieldındakileriGetir = dict

End Function

Sub test\_pivot\_seçili\_elemanlar\_çoklufield()

'Örnek olarak, Tarihsl Datanın Pivot ve Slcr sayfasındayken çalıştırabilirsiniz. Pivotta bikaç bölge ve ürün filtreli olsun

Dim mydict As Dictionary

Dim alanlar As New Collection

alanlar.Add "Bölge Kodu"

alanlar.Add "Ürün"

Set mydict = PivotunÇokluFieldındakileriGetir(ActiveSheet, "ptBolgekodUrun", alanlar)

For Each k In mydict.Keys

Debug.Print k, birleştir(mydict(k))

Next k

End Sub

Function birleştir(pis As PivotItems)

For Each pi In pis

geçici = geçici & "-" & pi

Next pi

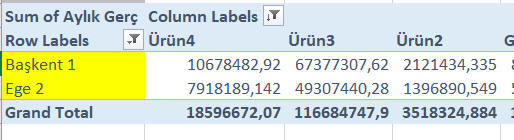
birleştir = Mid(geçici, 2)

End Function

Aşağıdaki örnek üzerinden çalıştırdığımızda dönen sonuç da şöyle olmaktadır.

Bölge Kodu Başkent 1-Ege 2

Ürün ÜÜrün Ürün4-Ürün3-Ürün2



##### Bir fieldda filtre var mı kontrolü

Burda da **AllItemsVisible** şeklinde boolean tipli bir property ile sorgulamamızı yapabilyoruz.

Function pivot\_field\_filtrelimi(ws As Worksheet, ptName As String, ptfield As String) As Boolean

pivot\_field\_filtrelimi = Not ws.PivotTables(ptName).PivotFields(ptfield).AllItemsVisible

End Function

Sub test\_pivot\_field\_filtrelimi()

'data sayfasında çalışıtırın

Debug.Print pivot\_field\_filtrelimi(ActiveSheet, "PivotTable1", "Bölge")

Debug.Print pivot\_field\_filtrelimi(ActiveSheet, "PivotTable1", "Şube Adı")

End Sub

##### Bir fieldda 1 adet filtre kriteri uygulamak

Tek bir field üzerinde 1 adet filtre uygulamanın en pratik yolu PivotFilters nesnesini kullanmaktır. Syntaxı aşağıdaki örnekten görüldüğü üzere oldukça pratiktir. Type argümanının alacağı değerlerin intellisense ile çıkmasını isterseniz PivotFilters nesnesini en başta Dim ile tanımlayıp öyle ilerleyebilirsiniz. Aşağıdaki örnekte spesifik bir tarih ile iki tarih arası değer uygulama örneği bulunmaktadır. Bunların alabileceği değerlerin bir listesini MSDN’den([https://docs.microsoft.com/en-us/office/VBA/api/Excel.xlpivotfiltertype](https://docs.microsoft.com/en-us/office/vba/api/excel.xlpivotfiltertype)) görebileceğiniz gibi, makro recorder ile denemeler sırasında da elde edebilirsiniz.

Sub pivotta\_filtrekoy()

'Tek fieldfda tek değer filtreleme

Sheets("Tarihsl Datanın Pivot ve Slcr").Select

Application.EnableEvents = False 'ilgili sayfadaki eventler tetiklenmesin

ActiveSheet.PivotTables("ptAy").ClearAllFilters

ActiveSheet.PivotTables("ptAy").PivotFields("Ay").PivotFilters.Add2 \_

Type: =xlSpecificDate, Value1: ="31.01.2016"

ActiveSheet.PivotTables("ptAy").ClearAllFilters

ActiveSheet.PivotTables("ptAy").PivotFields("Ay").PivotFilters.Add2 \_

Type: =xlDateBetween, Value1: ="31.01.2016", Value2: ="31.05.2016"

Application.EnableEvents = True

End Sub

Diğer PivotFilter örnekleri aşağıdaki gibi olabilir.

ActiveCell.PivotField.PivotFilters.Add FilterType : = xlThisWeek

ActiveCell.PivotField.PivotFilters.Add FilterType : = xlTopCount

DataField : = MyPivotField2 Value1 : = 10

ActiveCell.PivotField.PivotFilters.Add FilterType : =

xlCaptionIsNotBetween Value1 : = "A" Value2 : = "G"

ActiveCell.PivotField.PivotFilters.Add FilterType : =

xlValueIsGreaterThanOrEqualTo DataField : = MyPivotField2 Value1 : = 10000

##### 1 fieldda 1 filtre uygulamanın alternatifi

1 fielda sadece 1 adet kriter gireceksek PivotFilter gayet pratik ve yeterlidir. Bununla birlikte bunun başka bir yolu daha var, bu tekniği öğrenerek daha kompleks filtreleri uygularken rahat ederiz.

Bu yöntemde, PT'ın ilgili fieldındaki tüm elemanlarda dolaşıp istediğimiz eleman dışındakilerin hepsinin **Visible** özelliğine False atıyorum. Bunu yaparken eleman formatını da opsiyonel olarak gönderdiğime dikkat edin. Zira tarihsel bir alan Amerikan formatı olan ay/gün/yıl(m/dd/yyyy) formatında olduğu için biz tarihsel alanlara yerel formatımız olana gün.ay.yıl(dd.mm.yyyy) gönderiyoruz. Metinsel ve sayısal alanlar için bu parametreyi göndermemize gerek bulunmuyor.

Sub Pivotun1Fieldında1ElemanFiltrele(ws As Worksheet, ptName As String, ptfield As String, eleman As Variant, Optional elemanFormat)

Dim item As PivotItem

Application.EnableEvents = False 'ilgili sayfadaki eventler tetiklenmesin

ws.PivotTables(ptName).PivotFields(ptfield).ClearAllFilters 'Önce tüm filtreleri kaldıralım ki, hiç elamanın seçili olmaadığı mantıksız bir durum oluşmasın

For Each pi In ws.PivotTables(ptName).PivotFields(ptfield).PivotItems

deger = IIf(IsMissing(elemanFormat), pi, Format(pi, elemanFormat))

If deger <> eleman Then

pi.Visible = False

End If

Next pi

Application.EnableEvents = True

End Sub

Sub test\_pivotta\_tek\_eleman\_seç()

'Tarihsl Datanın Pivot ve Slicerı sayfasındayken çalıştırın

Pivotun1Fieldında1ElemanFiltrele ActiveSheet, "ptBolgekodUrun", "Bölge Kodu", "Ege 1"

Pivotun1Fieldında1ElemanFiltrele ActiveSheet, "ptAy", "Ay", "31.03.2016", "dd.mm.yyyy"

End Sub

##### Birden çok fieldda filtre uygulama

Bir önceki örneğimizde bir fielda bir adet filtre uygulamanın PivotFilter nesnesi kullanımına bir alternatifini görmüş ve bunun daha kompleks uygulamalara baz teşkil edeceğini söylemiştik.

İşte bu örneğimizde benzer yaklaşımı kullanarak birden çok kolonda bir eleman veya bir kolonda çok eleman veya çok kolonda çok eleman seçimlerine bakacağız.

Tabi önceki örneğe ana yaklaşım olarak benzemekle birlikte parametrelerimizde değişiklikler olacaktır. Mesela elemanları fieldıyla birlikte tutabilmek için Dictionary kullandım.

Test prosedüründe de her bir alandaki değerleri bir Collection'da tutmak yerine yine Dictionary kullandım. Dictionary'leri çok sevdiğimden mi bu kadar çok dictionary kullanıyorum? Sevdiğim doğrudur, ama neden collection kullanmadığımı siz bir düşünün, hatta deneyin ve neden olmadığını kendiniz görün. Cevabı bulamazsanız aşağıdaki kod bloğunun bir altına bakın.

Test prosedüründe kriterler dictionary’sini sıfırladığımız satıra dikkat. Burada bir önceki elemanlarla karışmasın diye içeriği boşaltmak lazım ama bunu RemoveAll diyerek yapmamalıyız, zira dictionary'yi komple baştan yaratmamız gerekiyor. İsterseniz bu satırda Removeall yazıp ne olduğunu kendiniz kontrol edin.

NOT: Bu örnekte eleman formatı kontrol etmedim. Bunu ödev olarak size bırakıyorum. (Çözemezseniz, Slicer'lara bakabilirsiniz, orda çok eleman seçme işinde kullandığım yöntem kullanılabilir)

Sub PivotunÇokFieldındaÇokElemanFiltrele(ws As Worksheet, ptName As String, pFieldveItems As Dictionary)

Dim pi As PivotItem

Application.EnableEvents = False

ws.PivotTables(ptName).ClearAllFilters

For Each pf In pFieldveItems

For Each pi In ws.PivotTables(ptName).PivotFields(pf).PivotItems

If Not pFieldveItems(pf).Exists(pi.Name) Then

pi.Visible = False

End If

Next pi

Next pf

Application.EnableEvents = True

End Sub

Sub test\_PivotunÇokFieldındaÇokElemanFiltrele()

Dim kriterler As New Scripting.Dictionary

Dim dict As New Scripting.Dictionary

kriterler.Add "Başkent 1", 100

kriterler.Add "Ege 2", 200

dict.Add "Bölge Kodu", kriterler

Set kriterler = New Dictionary 'sıfırlıyoruz ki ürünlerin içine bölgeler karışmasın. ama dikkat removeall değil, new dict diyoruz

kriterler.Add "Ürün4", 300

kriterler.Add "Ürün2", 400

kriterler.Add "Ürün3", 500

dict.Add "Ürün", kriterler

PivotunÇokFieldındaÇokElemanFiltrele ActiveSheet, "ptBolgekodUrun", dict

End Sub

Neden Dictionary: Collection kullanıp elemanlarda tek tek dolaşınca az önce işaretlemediğiniz bir elemanı sonraki loopta işaretleyebiliyorsunuz, bu da işimizi bozuyor. Halbuki bize tek tek dolaşarak kontrol yerine tek seferde kontrol etme özelliği lazım, bunu da Dictionary'nin Exists metodu ile yapabiliyoruz.(NOT: Tabiki Collection da kullanıp collection için custom yazdığımız ColdaVarmı fonksiyonunu da kullanabiliriz ama kodumuzu neden uzatalım, hazır bir metod varken)

#### Önemli Eventler

PT'larla ilgili olarak iki önemli event var. Event'ler PT'ların, ancak bu eventler devreye girince daha çok diğer nesnelerle iş yapılıyor diye düşünüyorum. Ben şahsen bunları daha çok Slicer'larla birlikte kullanıyorum. Mesela Slicer'daki seçim değişikliğini yakalamak için kullanacağız, zira Slicer eventi diye bir event yok, onun yerine PT eventlerinden faydalanacağız.

Bu iki eventimiz şunlardır:

* Worksheet\_PivotTableChangeSync
* Worksheet\_PivotTableUpdate

Bir PT refresh olursa veya uygulanan filtre değişirse tetiklenirler. Refresh olduğunda önce ilki sonra ikincisi tetiklenirken filtre değişiminde ise tam tersi sıra izlenir. Bu sıranın önemli olmadığı caselerde(ki çoğunlukla önemli olmayacak) ilk eventi kullanabilirsiniz. Herkesin önerdiği de budur.

Sıraları görmek için aşağıdaki kodları kullanarak bir PT üzerinde önce bir filtre uygulayın ve sıraya bakın, sonra da refreshteki sıraya bakın.

Private Sub Worksheet\_PivotTableChangeSync(ByVal Target As PivotTable)

Debug.Print "changesync oldu"

End Sub

Private Sub Worksheet\_PivotTableUpdate(ByVal Target As PivotTable)

Debug.Print "update oldu"

End Sub

Bu eventlerin örnek bir casede uygulanışına aşağıda Slicerlar bölümünde değineceğiz.

#### Diğer hususlar

##### PivotChart

PT yaratırkenki yöntem şöyleydi:

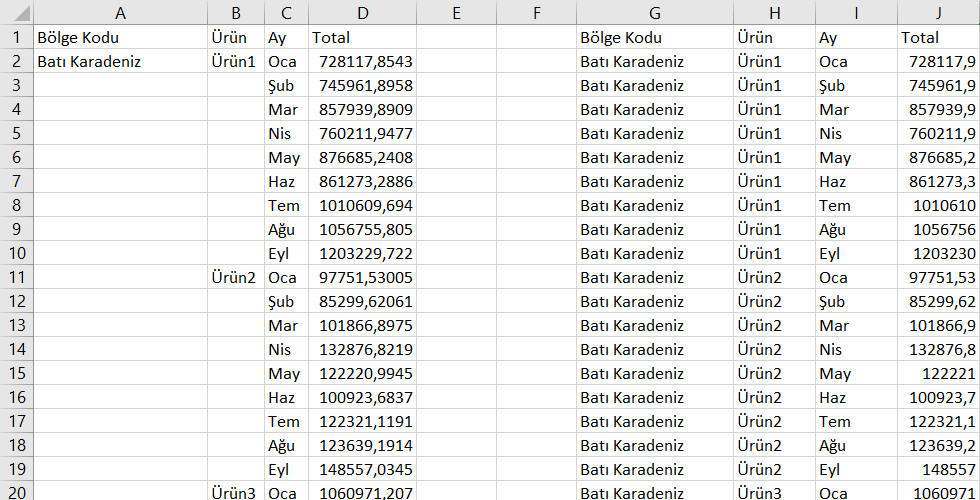
Dim pvt As PivotTable  
Dim pvtcch As PivotCache  
Set pvt = pvtcch.CreatePivotTable(TableDestination: = )

PivotChart da buna çok benzer,

Dim shp As Shape  
Dim pvtcch As PivotCache  
Set shp = pvtcch.CreatePivotChart(ChartDestination: = )

##### Repeat Label yapılmamış bir Pivot Table

Diyelim ki elinizde bir PT'ın copy-paste yapılmış hali var ve maalesef repeat label işlemi yapılmamış. Bizim bunu repeat label yapılmış hale getirmemiz lazım. Yani aşağıdaki görüntüde soldaki gibi olan listeyi sağdaki gibi yapmamız gerekiyor.



Bunun için yıllar önce bir kod hazırlamıştım, tabi Excel 2010la birlikte Repeat Label özelliiği gelince bu çöpe gitti gibi oldu. Bununla birlikte size dışarıdan copy-paste halde yani pivot fonksiyonalitesinden kurtulmuş şekilde gelen listeleri dönüştürmek için hala kullanılabilir.

Kodda, işlem yapılacak kolonlar soruluyor. Bu örnek için repeat label yapılacak olan kolonlar A ve B olduğu için A: B seçimini yaparız.

Sub pivot\_convert()

Dim kolonlar As Range

Dim baslangic As Range

Set kolonlar = Application.InputBox("İşlem yapılacak kolonları saeçin", Type: =8)

Set baslangic = Range("a2")

enalt = baslangic.CurrentRegion.Rows.Count

baslangic.Select

k = kolonlar.Columns.Count - 1

Do

baslangic.Offset(0, k).Select

Do

ActiveCell.Offset(1, 0).Select

If IsEmpty(ActiveCell) Then

ActiveCell.Value2 = ActiveCell.Offset(-1, 0).Value2

End If

Loop Until ActiveCell.Row = enalt

k = k - 1

Loop Until k < 0

End Sub

### Slicerlar

#### Giriş

Slicerlar’da da yine PT'larda olduğu gibi bir Slicercache(SC) ve bu cacheden beslenen Slicerın kendisi vardır. Burada da bir recorder çalıştırıp bakalım. Biz tabi PT'da olduğu gibi Slicer eklemeyi de genelde Excel ön yüzde manuel olarak yapmış olacağız. VBA ile daha ziyade Slicerlar’ın elemanlarına erişim, bunların seçimini değiştirme, refreshleme v.s gibi işlemler yapacağız ama yine de nasıl bir kod oluşuyor, ona bir bakalım.

Şimdi ilk olarak Data sayfasındaki Table'a(ListObject) Slicer uygulandığında nasıl bir kod oluştuğuna bakalım. Pivot üzerine uygulanan Slicer ile yine pivot üzerine uygulanan Timeline tipli Slicer kodları örnek dosya içinde bulunmaktadır.

Sub Macro2()'

' Macro2 Table Slicerı

'

'

ActiveWorkbook.SlicerCaches.Add2(ActiveSheet.ListObjects("Table7"), "Bölge"). \_

Slicers.Add ActiveSheet, , "Bölge", "Bölge", 40.5, 414, 144, 198.75

ActiveWorkbook.SlicerCaches.Add2(ActiveSheet.ListObjects("Table7"), "Ürün Adı") \_

.Slicers.Add ActiveSheet, , "Ürün Adı", "Ürün Adı", 78, 451.5, 144, 198.75

ActiveSheet.Shapes.Range(Array("Ürün Adı")).Select

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge")

.SlicerItems("Başkent 1").Selected = True

.SlicerItems("Başkent 2").Selected = False

.SlicerItems("Batı Karadeniz").Selected = False

.SlicerItems("Doğru Karadeniz").Selected = False

.SlicerItems("Ege 1").Selected = False

.SlicerItems("Ege 2").Selected = False

.SlicerItems("Güney Akdeniz").Selected = False

.SlicerItems("Güneydoğu Anadolu").Selected = False

.SlicerItems("İç Anadolu").Selected = False

.SlicerItems("İstanbul 1").Selected = False

.SlicerItems("İstanbul 2").Selected = False

.SlicerItems("İstanbul 3").Selected = False

.SlicerItems("Marmara").Selected = False

End With

ActiveWorkbook.SlicerCaches("Slicer\_Bölge").ClearManualFilter

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge")

.SlicerItems("Başkent 2").Selected = True

.SlicerItems("Başkent 1").Selected = False

.SlicerItems("Batı Karadeniz").Selected = False

.SlicerItems("Doğru Karadeniz").Selected = False

.SlicerItems("Ege 1").Selected = False

.SlicerItems("Ege 2").Selected = False

.SlicerItems("Güney Akdeniz").Selected = False

.SlicerItems("Güneydoğu Anadolu").Selected = False

.SlicerItems("İç Anadolu").Selected = False

.SlicerItems("İstanbul 1").Selected = False

.SlicerItems("İstanbul 2").Selected = False

.SlicerItems("İstanbul 3").Selected = False

.SlicerItems("Marmara").Selected = False

End With

With ActiveWorkbook.SlicerCaches("Slicer\_Ürün\_Adı")

.SlicerItems("Ürün1").Selected = True

.SlicerItems("Ürün2").Selected = False

.SlicerItems("Ürün3").Selected = False

.SlicerItems("Ürün4").Selected = False

End With

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge")

.SlicerItems("Doğru Karadeniz").Selected = True

.SlicerItems("Başkent 1").Selected = False

.SlicerItems("Başkent 2").Selected = False

.SlicerItems("Batı Karadeniz").Selected = False

.SlicerItems("Ege 1").Selected = False

.SlicerItems("Ege 2").Selected = False

.SlicerItems("Güney Akdeniz").Selected = False

.SlicerItems("Güneydoğu Anadolu").Selected = False

.SlicerItems("İç Anadolu").Selected = False

.SlicerItems("İstanbul 1").Selected = False

.SlicerItems("İstanbul 2").Selected = False

.SlicerItems("İstanbul 3").Selected = False

.SlicerItems("Marmara").Selected = False

End With

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge")

.SlicerItems("Başkent 2").Selected = True

.SlicerItems("Doğru Karadeniz").Selected = True

.SlicerItems("Başkent 1").Selected = False

.SlicerItems("Batı Karadeniz").Selected = False

.SlicerItems("Ege 1").Selected = False

.SlicerItems("Ege 2").Selected = False

.SlicerItems("Güney Akdeniz").Selected = False

.SlicerItems("Güneydoğu Anadolu").Selected = False

.SlicerItems("İç Anadolu").Selected = False

.SlicerItems("İstanbul 1").Selected = False

.SlicerItems("İstanbul 2").Selected = False

.SlicerItems("İstanbul 3").Selected = False

.SlicerItems("Marmara").Selected = False

End With

ActiveSheet.Shapes.Range(Array("Bölge")).Select

ActiveWorkbook.SlicerCaches("Slicer\_Bölge").Slicers("Bölge").Name = "BölgeSlc"

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge").Slicers("BölgeSlc")

.Caption = "Bölge"

.DisplayHeader = True

.SlicerCache.CrossFilterType = xlSlicerCrossFilterShowItemsWithDataAtTop

.SlicerCache.SortItems = xlSlicerSortAscending

.SlicerCache.SortUsingCustomLists = True

End With

End Sub

Görüldüğü üzere sadece bir iki adet işaretleme yapsam bile Slicer’daki diğer seçenekler için de otomatikman False değerini üretti. Manuel yazarken biz sadece istediklerimizi True yazarız, diğerleri mevcut değerleriyle kalır.

Mantıken, bir Slicer’da belirli bir anda en az bir eleman seçili olmalıdır. Bu eleman seçiliyken başka bir elamanı da seçmek istersek, sadece onu seçen kodu yazmak yeterli. Ör: Slicer’da A,B,C,D,E değerleri var olsun. Şuan sadece A seçiliyse, diğerlerinin Selected değeri False'tur. Biz ilaveten B'yi de seçmek istersek, **B.Selected=True** yazmak yeterli, ama sadece B seçili olsun istersek, A'ya da false atamak gerekir. Kendi bölge örneğimiz üzerinden gidecek olursak:

Sub ilave\_secim()

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge")

'cachelere ya indeks ile ya da "formula name" ile ulaşılır, name ile dğeil. Name Slicerın adıdır, cahcenin değil.

.SlicerItems("Ege 1").Selected = True

End With

End Sub

Sub tek\_secim()

With ActiveWorkbook.SlicerCaches("Slicer\_Bölge")

.SlicerItems("Ege 1").Selected = True

.SlicerItems("Doğru Karadeniz").Selected = False

End With

End Sub

Bu arada fark ettiyseniz SCs collection'ına bir indeks verince elde ettiğim tekil bir SC üzerinden doğrudan Sliceritemlar’a ulaşabiliyoruz, yani arada Slicer’ı seçmeye gerek yok. Zira bunlarda, PT'lardan farklı olarak bir SC ile bir Slicer eşleşiyor gibi bir durum var, zaten aksi garip olurdu.

Yukarıda recorder kodu olarak bir Table Slicer'ı vermiştik. Pivot Slicer’ı oluşturunca da pek farklı bir durum yok, sadece parametrelerde ufak bir iki değişiklik var. Bundan başka Table'larda olmayıp sadece PT'larda olan Timeline da eklenebiliyor, ki bunlar da bir tür Slicer olarak kabul görüyor.

#### Çeşitli propertyler

Şimdi küçük bir kod üzerinde birkaç özelliğe bakalım.

NOT: OLAP yani PowerPivot gibi bir kaynaktan besleneme durumunda kullanılabilen özellikler(level, visibleitemlist v.s) bu bölümlerin kapsamı dışındadır.

Az önce Timelineların da bir tür Slicer olduğunu söylemiştik, bunların property olarak sadece SlicerCacheTypeları farklı oluyor; Microsoft dokümantasyonunda bu type değeri için sadece 1 ve 2 değerleri var; 1 Slicer, 2 Timeline oluyor ama ben aşağıdaki sorguyu çalıştırdığımda 0 değerini de görüyorum. Zannımca bu 2013te Slicer özelliği gelen Table'lardan üretilen SC'leri ifade ediyor. Microsoftun bunu hala(bu kitabın hazırlandığı tarih olan 2019'da) dokümante etmemiş olması ilginç.

'Global değişkenler

Dim sl As Slicer

Dim slc As SlicerCache

Dim sli As SlicerItem

Dim ws As Worksheet

Sub tüm\_slicerlarda\_dolaşma()

For Each slc In ActiveWorkbook.SlicerCaches

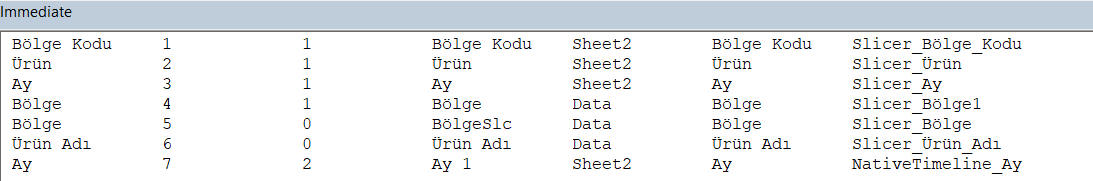
For Each sl In slc.Slicers

Debug.Print slc.SourceName, slc.Index, slc.SlicerCacheType, sl.Name, sl.Parent.Name, sl.Caption, slc.Name

Next sl

Next slc

End Sub



#### Çeşitli İşlemler

##### Seçili olan tek bir elemanı elde etme

Bu örnekte aktif olarak seçili olan elemanı elde edeceğiz.

Sub seçili\_eleman()

'focuslu ise çalışır, aksi halde hata verir

Debug.Print ActiveWorkbook.SlicerCaches("Slicer\_Bölge\_Kodu").Slicers(1).ActiveItem.Name

End Sub

##### Seçili olan birçok elemanı elde etme

Şimdi de birden çok eleman varsa onları yazdıralım. Tabi burada örnekleri basit tutmak adına elemanları sadece yazdırıyoruz, ihtiyaca göre elemanlar bir collectiona, formdaki listboxa v.s atanabilir.

Sub seçili\_elemanlar()

For Each sI In ActiveWorkbook.SlicerCaches("Slicer\_Bölge\_Kodu").VisibleSlicerItems

Debug.Print sI.Name

Next sI

End Sub

Bu işlem için birçok forumda aşağıdaki koda benzer bir kod sunuluyor. Biz PT kısmında seçili elemanları elde ederken farklı bir yöntem izlemiştik, orda dönüş tipini PivotItems olarak belirlemiştik ve demiştik ki Slicer’da Collection yöntemini kullanacağız. Pek tabiki burada da dönüş tipini SlicerItems yapabiliriz. Ancak biz geleneğe uyalım ve Collection yöntemini kullanalım, hemen altında da SlicerItems yöntemini deneriz.

Public Function SlicerdaSeçilileriGetir(SlicerName As String) As Collection

Dim geçici As New Collection

Set slc = ActiveWorkbook.SlicerCaches(SlicerName)

For Each sli In slc.SlicerItems

If sli.Selected Then

geçici.Add sli.Name

End If

Next

Set SlicerdaSeçilileriGetir = geçici

End Function

Sub test\_SlicerdaSeçilileriGetir()

Dim seçililer As Collection

Set seçililer = SlicerdaSeçilileriGetir("Slicer\_Bölge\_Kodu")

For Each s In seçililer

Debug.Print s

Next s

End Sub

Bu örnekte tekil ve çoklu eleman döndürme işlemini tek bir prosedürle yapmış olduk. Çoklu eleman seçiminde döngüsel olarak dolaşmayı yukarıdaki kodla görmüş olduk. Tek eleman seçiminde ise collection'ın ilk elemanına ulaşmak yeterlidir.

SlicerdaSeçilileriGetir("Slicer\_Bölge\_Kodu")(1)

SlicerItems dönüş tipli yönteme göre ise kodumuz daha kısa. Neden bu değil de Collection'lı kullanılıyor gerçekten bilmiyorum, Timer ile performans kontrolü de yaptım, ikisi de aynı sürede çalışıyor gibi görünüyor. Her neyse siz ikisini de bilin.

Public Function SlicerdaSeçilileriGetirSlicerItemslıYöntem(SlicerName As String) As SlicerItems

Set SlicerdaSeçilileriGetirSlicerItemslıYöntem = ActiveWorkbook.SlicerCaches(SlicerName).VisibleSlicerItems

End Function

Sub test\_SlicerdaSeçilileriGetirSlicerItemslıYöntem()

Dim kimlerseçili As SlicerItems

Set kimlerseçili = SlicerdaSeçilileriGetirSlicerItemslıYöntem("Slicer\_Bölge\_Kodu")

For Each vi In kimlerseçili

Debug.Print vi.Name

Next vi

End Sub

##### Tek bir elemanı seçme

Bu yöntemde PT'larda yaptığımız gibi elemanlar üzerinde tek tek dolaşarak ilerleyeceğiz. Parametrelerden son parametreye dikkat, orada elemanların formatı farklı olabilir diye bunu da belirtme seçeceği veriyorum. Eğer bu parametre prosedüre geçirilmezse eleman aynen alınırken, geçirilme durumunda format fonksiyonu ile şekillendirilip öyle seçilir.

Sub slicerda\_tek\_eleman\_seç(cacheName As String, eleman As Variant, Optional elemanFormat)

Dim item As SlicerItem

Application.EnableEvents = False 'ilgili sayfadaki eventler tetiklenmesin

ActiveWorkbook.SlicerCaches(cacheName).ClearManualFilter 'Önce tüm filtreleri kaldıralım ki, hiç elamanın seçili olmaadığı mantıksız bir durum oluşmasın

For Each item In ActiveWorkbook.SlicerCaches(cacheName).SlicerItems

'pivot örneğinde pivotitem'ı formatlıyorduk, burada filtre değerini

If item.Name <> IIf(IsMissing(elemanFormat), eleman, Format(eleman, elemanFormat)) Then

item.Selected = False

End If

Next item

Application.EnableEvents = True

End Sub

Sub test\_slicerda\_tek\_eleman\_seç()

slicerda\_tek\_eleman\_seç "Slicer\_Ürün", "Ürün3"

slicerda\_tek\_eleman\_seç "Slicer\_Ay", "29.02.2016", "Long Date"

End Sub

##### Çok eleman seçme

Bu örnekte ise birden çok elemanı seçiyoruz. Birden çok değerin varlığını sorgulamak durumunda olduğum için Dictionary kullanıyorum.

Sub slicerda\_çok\_eleman\_seç(cacheName As String, elemanlar As Dictionary, Optional elemanFormat)

Dim item As SlicerItem

Application.EnableEvents = False

If Not IsMissing(elemanFormat) Then

For Each e In elemanlar.Keys

elemanlar.Key(e) = Format(e, elemanFormat)

Next e

End If

ActiveWorkbook.SlicerCaches(cacheName).ClearManualFilter 'Önce tüm filtreleri kaldıralım ki, hiç elamanın seçili olmadığı mantıksız bir durum oluşmasın

For Each item In ActiveWorkbook.SlicerCaches(cacheName).SlicerItems

If Not elemanlar.Exists(item.Value) Then

item.Selected = False

End If

Next item

Application.EnableEvents = True

End Sub

Sub test\_slicerda\_çok\_eleman\_seç()

Dim dict As New Scripting.Dictionary

dict.Add "Ürün2", 1

dict.Add "Ürün3", 2

slicerda\_çok\_eleman\_seç "Slicer\_Ürün", dict

'olmayan veri de gönderelim

dict.RemoveAll

dict.Add "29.02.2016", 1

dict.Add "25.02.2016", 2

dict.Add "31.03.2016", 3

slicerda\_çok\_eleman\_seç "Slicer\_Ay", dict, "Long Date"

End Sub

##### PT eventlerini kullanarak Slicer değişimlerini yakalamak

Yukarıda PT'lar kısmında PT eventlerini görmüştük. Şimdi bu eventleri Slicerlar’la birlikte nasıl kullanırız, ona bakacağız.

###### PT değişimiyle seçili elemanı elde etmek

İster PT üzerinden manuel, ister kod ile, ister Slicer'ın kendisi aracılığı ile PT üzerinde bir seçim yapmış olalım. Bu seçilen elemanı bir hücrede, bir grafik başlığında veya kod içinde her hangi bir yerde değişken olarak kullanmak isteyebiliriz.

"Tarihsel Datanın Pivot ve Slicerı" sayfasındaki PT'lardan ptBolgekodUrun isimli olan PT'da bir değişklik olduğunda F1 hücresine ilgili PT'da seçilen elemanı yazdırıyoruz. Birden çok eleman seçilirse ilki yazılır. Bizim kodumuz, özellikle bir eleman seçiminin beklendiği durumlarda anlamlıdır. Zira muhtemelen bunu küçük bir veri bloğunun başlığı halinde kullanacağızdır, "**Ege 1** Bölgesi Kredi Rakamları" gibi.

Private Sub Worksheet\_PivotTableChangeSync(ByVal Target As PivotTable)

Debug.Print "changesync oldu"

If Target.Name = "ptBolgekodUrun" Then

[F1].Value = SlicerdaSeçilileriGetir("Slicer\_Bölge\_Kodu")(1) & " Bölgesi"

End If

End Sub

###### Dashboard Yönetimi

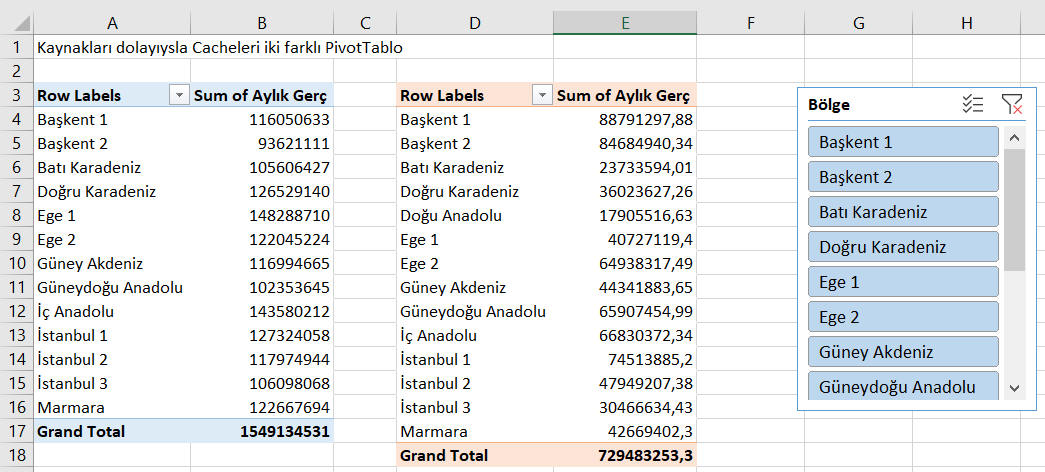
PT kısmında birden fazla PT aynı PC'yi kullanıyorsa bunlardan birinde yapılan seçimi diğerlerini de otomatikman etkiler demiştik.

Peki ya elimizde PC'leri farklı olan PT'lar varsa ve biz birinde yapılan seçim  hepsinde geçerli olsun istiyorsak?

Mesela aşağıdaki farklı renkle gösterilen bloklardan oluşan bir dashboardda farklı kaynaklardan gelen PT'lar bulunuyor olsun. Bunların her biri için Slicer yapmaya gerek yok. Biz biri için bir Slicer yaparız, bunda seçilen elemanı bir değişkende depolarız(yukarıda gördüğümüz yöntemlerle) ve diğer PT'larda da bu elemanı seçtiririz(yine yukarıdaki yöntemlerle). İkinci kısmı yaptırabilmek için bir haber almamız, yani bir eventin tetiklenmesi lazım, acaba hangi event; tabiki **Worksheet\_PivotTableChangeSync** eventi.



Biz örneğimizi basit tutmak adına sadece iki PT kullanacağız. Soldakinin adı ptSol, sağdakinin adı ptSağ. Slicer'ımız soldakine aittir.



Buna göre kodumuz aşağıdaki gibidir.

Private Sub Worksheet\_PivotTableChangeSync(ByVal Target As PivotTable)

If Target.Name = "ptLeft" Then

If Not pivot\_field\_filtrelimi(ActiveSheet, "ptLeft", "Bölge") Then

secilen = SlicerdaSeçilileriGetir("Slicer\_Bölge2")(1)

Pivotun1Fieldında1ElemanFiltrele ActiveSheet, "ptRight", "Bölge Kodu", secilen

Else

ActiveSheet.PivotTables("ptRight").ClearAllFilters

End If

End If

End Sub

Dikkat ettiyseniz ilk olarak eventin soldaki PT için tetiklenip tetiklenmediğine bakıyoruz. Sağdaki için manuel bir filtre yapılırsa kod ilk If bloğundan sonra çıkar. Bir diğer kontrolümüz de değişklik sonucunda PT'da filtre var mı yok mu kontrolü, varsa esas kodlarımızı işletiyoruz, filtre yoksa sağdaki için de ClearAllFilters yapıyoruz.

Aşağıda seçim sonrası tabloların genel durumu görünüyor.



##### Table eventlerini kullanmak

Sadece PT eventi değil bir Table eventini kullanarak da Slicerlarla işlem yapabiliriz. Mesela aşağıdaki kod ile, Tarihsel Data sayfasında varsayalım ki bir veritabanı bağlantısı var ve bu tablo belli zamanlarda(dosya ilk açıldığında, sayfa aktive olduğunda v.s) refresh oluyor olsun. Refresh sonucunda mevcut tarihten daha büyük bir tarih gelirse, Slicer da buna göre güncellensin, yani slicerda en büyük tarih seçili olsun.

Private Sub Worksheet\_TableUpdate(ByVal Target As TableObject)

Dim maxTarih As String

maxTarih = Format(WorksheetFunction.Max(Worksheets("Tarihsel Data").Columns(3)), "dd.mm.yyyy")

Application.EnableEvents = False

slicerda\_tek\_eleman\_seç "Slicer\_Ay", maxTarih, "Long Date"

Application.EnableEvents = True

End Sub

##### Kullanıcılara PT Slicerını Table Slicer'ı gibi kullandırtmak

Bazen olur ki, Table'a ait Slicer üzerinde bir değişklik yaptığınızda bunun bir eventi tetiklemesini istersiniz. Ancak Table'ların filtrelenmesine yönelik böyle bir event yoktur. Sadece TableUpdate eventi var, o da tablo güncellendiğinde devreye girer.

Böyle bir durumda Excel'i kandırma yoluna gidebiliriz. İzleyeceğimiz yollardan biri şudur:

* Bu tabledan bir PT yaratmak
* Bu PT üzerine bir Slicer koymak
* PT' bölgesini gizlemek veya aşağı bir bölgeye almak, yeter ki görünmesin
* Slicer seçildiğinde PT'ın ChangeSync eventinin tetiklenmesini sağlamak
* PT eventi tetiklenince seçilen değeri alıp bu değeri Table üzerinde filtre olarak uygulamak(Table üzerinde doğrudan veya ona ait Slicerda dolaylı. Slicer kullanacaksak bunu genelde gizlemeyi tercih ederiz, iki slicer karışmasın diye, ama ben bu sayfadakini gizlemeden bıraktım)

ve diğer işlerimizi yaptırmak

Buna ait basit bir örneği aşağıda bulabilirsiniz. Kod Data sayfasına aittir.

Private Sub Worksheet\_PivotTableChangeSync(ByVal Target As PivotTable)

'Exit Sub 'Event örneklerini yaptığımızda bu satırı commentleyin, event dışındaki örneklerde açık kalsın

Application.EnableEvents = False

If ActiveWorkbook.SlicerCaches("Slicer\_Bölge1").VisibleSlicerItems.Count = ActiveWorkbook.SlicerCaches("Slicer\_Bölge1").SlicerItems.Count Then 'pivotun slicerı

ActiveWorkbook.SlicerCaches("Slicer\_Bölge").ClearManualFilter 'Tableınki, görünmezdir, veya aşağılarda görünmez olsa da olur

GoTo çıkış

Else 'sadece tek seçim yapıldığı varsayılmıştır, çoklu seçimlerde işe yaramaz

secilen = SlicerdaSeçilileriGetir("Slicer\_Bölge1")(1)

ActiveWorkbook.SlicerCaches("Slicer\_Bölge").ClearManualFilter 'önce clear

slicerda\_tek\_eleman\_seç "Slicer\_Bölge", secilen

End If

çıkış:

Application.EnableEvents = True

End Sub

İzleyeceğimiz bir başka yol da **Worksheet\_Calculate** eventinden faydalanmaktır. Her iki alternatifi de göreceğimiz daha anlamlı bir örneği Chart nesnesinden sonraki Örnek Uygulama bölümünde bulabilirsiniz.

### Chart(Grafik) Nesnesi

Dashboard yönetimi kısmında makroların gücünü döngüler ve koşullu yapıları kullanarak çok güzel görmüştük. Grafiklerde de benzer kolaylıklardan yararlanacağız. Tabiki PT'larda olduğu gibi grafik yaratmayla pek uğraşmayacağız. Otomasyon projelerinde kodunuzun bir grafik yaratması gerekse bile bunu recorder ile ürettiğiniz kodu elden geçirerek yapabilirsiniz.

Mesela yukarıda Slicerlar bölümünde 6 farklı renkli gösterilmiş dashboardu düşünün, orada o bloklar PT'lardan oluşuyordu, şimdi varsayalım ki bunlar birer grafik. Bu sefer her birinin tipini, boyutunu, pozisyonunu, lejandını, renkleri v.s tek seferde değiştirmek istiyoruz, tabiki bunu döngülerle yapacağız.

Veya grafiğe konu olan datasetinde seçilen bir değere göre grafiği dinamik olarak güncellemek gerekecek. Bu gibi işleri yine makro ile başaracağız.

#### Nesneler hiyerarşisi

Şimdi öncelikle elimizdeki nesneler bir bakalım

**Chart** nesnesi: Bunlar bağımsız grafik sayfaları olabileceği gibi normal bir sayfa(worksheet) içindeki gömülü grafikler de olabilir. Ancak bunlardan sadece bağımsız sayfa olarak bulunanlar Charts koleksiyonunun bir üyesidir, gömülü olanlar değil.

**ChartObject** nesnesi: Worksheet içinde bulunan grafiklerin(Worksheet Chartlarının) konteynerıdır. Esasen bir grafik nesnesi değil, onun kabuğudur. Propertylerine falan bakıldığında grafikten ziyade Shape nesnesinin özelliklerini görürsünüz. ChartObjects koleksiyonunun bir üyesidir. Aynı zamanda Shapes koleksiyonunun da bir üyesidir. Hatta macro recorder ile grafik oluşturduğunuzda Shapes üzerinden oluşturur.

Yani aslında 3 tür nesnemiz var diyebilir.

* **Worksheetlerdeki gömülü Chart**: Chart. Parent nesnesi ChartObjecttir.
* **Bağımsız Chart**: Bu da Chart'tır. Parent nesnesi Workbook'tur.
* **Worksheet Chartı için konteyner**: ChartObject. Buna ulaşmadan içindeki Charta ulaşılamaz. Parent nesnesi Worksheet'tir.

Örnek birkaç erişim şekli şöyle olabilir:

'Worksheet içindeki bir Chart'a zincirleme olarak

Sheets("Sheet1").ChartObjects("Chart 1").Chart.ChartTitle.Text = "Yeni başlık"

'veya önce konteynerı seçince aktif olan ActiveChart ile

Sheets("Sheet1").ChartObjects("Chart 1").Activate

ActiveChart.ChartTitle.Text = "Yeni başlık"

'bağımsız chart

Sheets("Chart 1").ChartTitle.Text = "Yeni başlık" 'sayfa adından

ActiveWorkbook.Sheets("Chart1").ChartTitle.Text = "Yeni başlık" 'nesne adından

Bu bilgiler ışığında chartlarda dolaşma şu şekilde olur.

Sub chartlarda\_dolaşma()

Dim chto As ChartObject

Dim cht As Chart

'sayfa olan chartlarda dolaşalım

For Each cht In ActiveWorkbook.charts

Debug.Print cht.Name, TypeName(cht), TypeName(cht.Parent)

Next cht

'gömülü chartlarda dolaşalım

For Each ws In ActiveWorkbook.Worksheets

For Each chto In ws.ChartObjects

Debug.Print chto.Name, chto.Index, chto.Parent.Name 'Chart'ın adı biraz anlamsız, ChartObject'in adını kullanalım

Next chto

Next ws

End Sub

#### Yaratım ve değişiklik

Yaratım ve özelliklerdeki değişiklikler için hep recorder kullanmanızı tavsiye ediyorum arkadaşlar. O yüzden buna ait kodları buraya koymuyorum.

#### Diğer hususlar

##### Grafik datasının olduğu bölgeyi grafikle kaplamak

Özellikle dashboard sayfalarında grafik datasının ortalıkta görünmesini istemeyiz. Gizli satır/kolon veya sayfanın aşağılarında bir yerlere koymak bir çözüm olabileceği gibi, data alanı çok küçük/büyük değilse bu bölgeyi grafiğin kendisiyle kapatmak bir çözüm olabilir.

Aşağıdaki kod ile bunu yapabiliriz. Kodda bir yardımcı fonksiyon kullanıyorum. Bu fonksiyonda F8 ile ilerleyerek "dizi" değişkenini incelerseniz adres1 ve adres2de ne yaptığımı anlayacaksınız.

Dashboard sayfanızdaki tüm grafiklerde döngüsel olarak çalışacak şekilde bu kodu modifiye edebilirsiniz.

Sub RangeiGrafikleKapla()

Dim cht As ChartObject

Dim grafikAlan As Range

Set cht = ActiveChart.Parent

Set grafikAlan = GrafikRange(cht.Chart)

cht.Left = grafikAlan.Left

cht.Width = grafikAlan.Width

cht.Top = grafikAlan.Top

cht.Height = grafikAlan.Height

End Sub

'Yardımcı fonksiyonumuz

Function GrafikRange(cht As Chart) As Range

dizi = cht.SeriesCollection(1).Formula

adres1 = Split(Replace(Split(dizi, ",")(1), cht.Parent.Parent.Name & "!", ""), ": ")(0)

adres2 = Split(Replace(Split(dizi, ",")(2), cht.Parent.Parent.Name & "!", ""), ": ")(1)

Set GrafikRange = Range(adres1, adres2)

End Function

##### Grafikleri silme ve gizleme

Silme ve gizleme işlemlerini Shapeler aracılığı ile yapabileceğimiz gibi ChartObject'in kendisiyle de yapabiliriz. Burada önemli olan Chart nesnesini değil ChartObject nesnesini silebildiğimizdir.

ActiveChart.Delete 'çalışmaz

ActiveSheet.ChartObjects(1).Delete 'bu 1 nolu chartı siler

ActiveSheet.ChartObjects.Delete 'bu sayfadaki tüm grafikleri siler

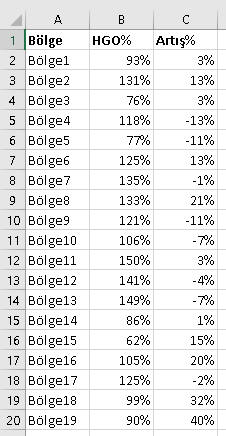
Chart sayfalarını ise normal sayfa siler gibi siliyoruz.

### Örnek Uygulama

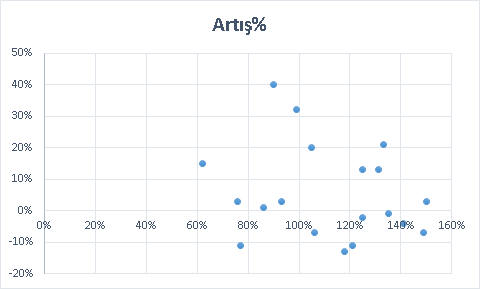
Bu örnekte öncelikle bir scatter grafik üzerinde çalışacağız. Öncelikle Excel önyüzde grafiğimizi oluşturalım, sonra örneğimizi biraz değiştirerek VBA’ya nerde ihtiyacımız olacağına bakalım.

#### Öncü örnek:

Aşağıdaki gibi bir data kümeniz var ve siz hem Hedef Gerçekleştirme Oranı(HGO%) hem de geçen yıla göre artış oranı küçük bölgeleri yakın izlemeye almak istiyorsunuz diyelim. Bu arada örnek dosyayı **github’dan(scatter.xlsx)** indirebilirsiniz. Bununla beraber diğer 3 kombinasyonu da görmek istiyorsunuz, kim nerede diye. Bunu basit bir IF formülü ile de yapabilirsiniz, ancak bu yöntemde 2 soru karşımıza çıkar. 1- Hangi spesifik değer altını/üstünü hangi kategoriye koyacaksınız, bunu seçmek zor olabilir 2-Seçtiniz diyelim, bunları görsel olarak görmenin rahatlığını elde edebilecek misiniz? Çok büyük ihtimalle hayır.

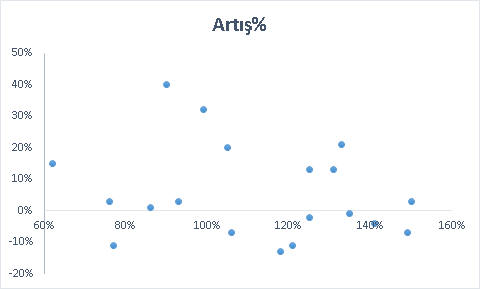


İşte böyle durumlarda XY(Scatter) Grafikleri en uygun çözümü sunar. Recommended Charts içinden seçelim kendisini.



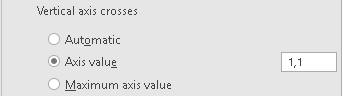
Bu haliyle çok yavan, yine de üç aşağı beş yukarı yoğunlaşma bölgelerini görebiliyorsunuz, ama daha işimiz var, şimdi biraz bu grafiğe çeki düzen verelim.

Öncelikle her iki eksendeki min ve max noktalarını iyi belirleyelim ki, boş alanlar gereksiz yer kaplamasın. Bunun için X eksenine sağ tıklayıp Axis Options'a girip Bounds>Minimum değerini 0,6 yapıyorum. Bu arada arka plandaki grid çizgilerini kaldırıyorum.

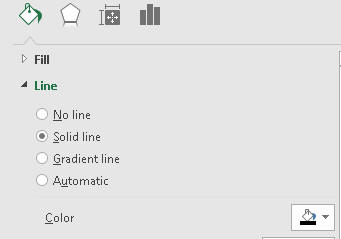


Şimdi Y ekseninin X eksenini tam ortasından kesmesini sağlayacağım, aynı zamanda X ekseni de 0 çizgisinden değil, Y ekseninin min ve max değerlerinin ortasından geçsin istiyorum.(İlla orta noktalardan geçmek zorunda değil, isterseniz belirlediğiniz referans değerlerden de geçebilir)

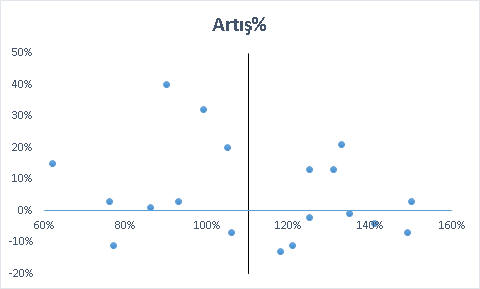
Önce Y'yi halledelim: 0,6 ile 1,60'ın ortası 1,10. Şimdi burası biraz karışık gelebilir. Y eksenini ayarlıyoruz dedik ama beklediğinizin aksine Y eksenini değil X eksenini seçip Axis Options deriz ve Vertical(yani Y) eksenin geçtiği noktayı Automaticten Axis Value=1,1'e değiştiririz.



Hemen arkasından Y eksenini seçip **Label Position**olarak **Low**deriz, ve Fill&Line alt sekmesinden de Line formatını **Solid Line**seçip rengini de siyah olarak belirleyelim.

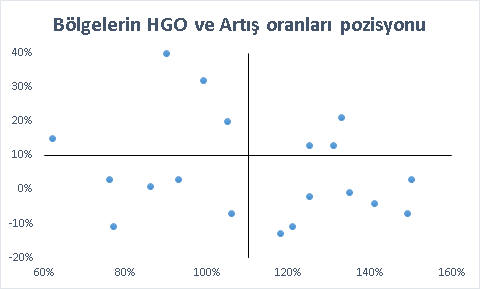


ve bu haliyle grafiğimiz biraz daha istediğimiz şekle bürünmüş oldu.



Şimdi de X eksenini biraz yükseltelim. -20 ile +40'ın ortası +10dur. Bu sefer Y eksenini seçip, Eksenin geçtiği nokta olarak 0,1 değeri girilerek, X ekseni için Label Position'ı Low ve Solid Black line seçimleri yapılır. Bu arada Y ekseni için max değeri %50den %40a da çekelim, onu unutmuşuz, yukarda %10luk alan boş yere işgal olmasın.

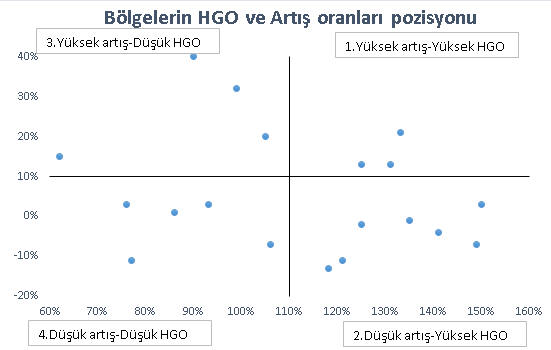
Son olarak otomatik gelen başlığı da daha anlamlı hale getirelim.



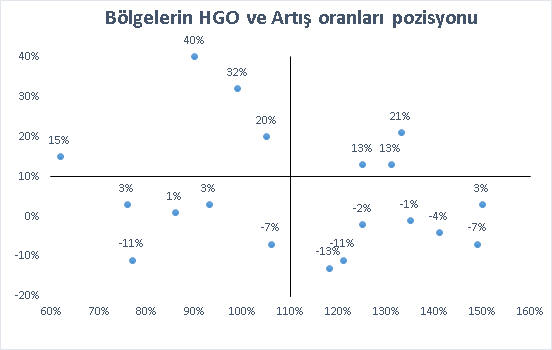
Vee işlem tamamdır.

Şimdi aklınıza şöyle bir soru gelebilir. Bu eksenlerin ikisi de yüzdesel, bunlardan hangisi hangisi? Grafiğe Axis Title ekleyerek bu sorunu çözebileceğiniz gibi grafik alanınızı daraltmamak adına akıl yürütme yoluna da başvurabilirsiniz. Siz bilirsiniz ki HG% oranı genelde %100 etrafındadır ve asla negatif olamaz; artış oranları ise hem negatif olabilir hem de göreceli daha düşük değerlerdir. Bu bilgiyle Y ekseninin artış ekseni, X ekseninin de HGO ekseni olduğunu anlarsınız. Tabi bazı durumlarda bir eksen mutlak değer bir eksen yüzdesel olur, ki böyle durumlarda neyin ne olduğu zaten bellidir.

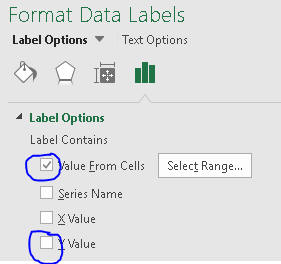
Buraya istenirse 4 adet textbox konarak bölmelerin açıklaması ve önem sırasına göre numaralandırılması yapılabilir.



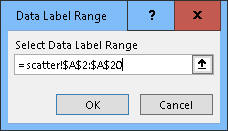
Peki hangi yuvarlak nokta hangi Bölgeyi ifade ediyor. İşte bu sorunun cevabı Excel 2013e kadar verilemiyordu. 2013 ile birlikte bu sorun da çözüldü. Bunun için öncelikle grafiğimize Data Labels eklememiz lazım.



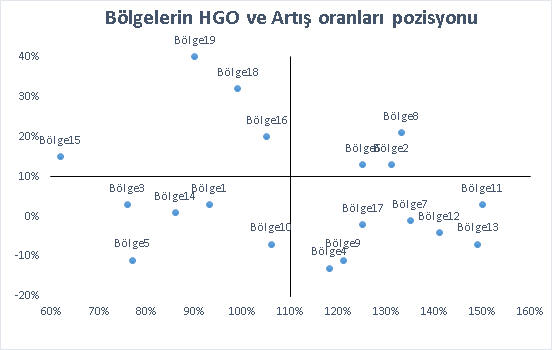
Sonrasında bu Data Labellara tıklayarak, Value From Cells seçimi yapılır, istenirse Y değerlerinin gösterilmemesi de sağlanır, zira ne kadar çok veri o kadar karmaşa demektir.



Alan olarak bölge isimlerinin olduğu yer seçilir.



Nihai grafiğimiz aşağıdaki gibi olacaktır.



##### Dinamik Seçimler

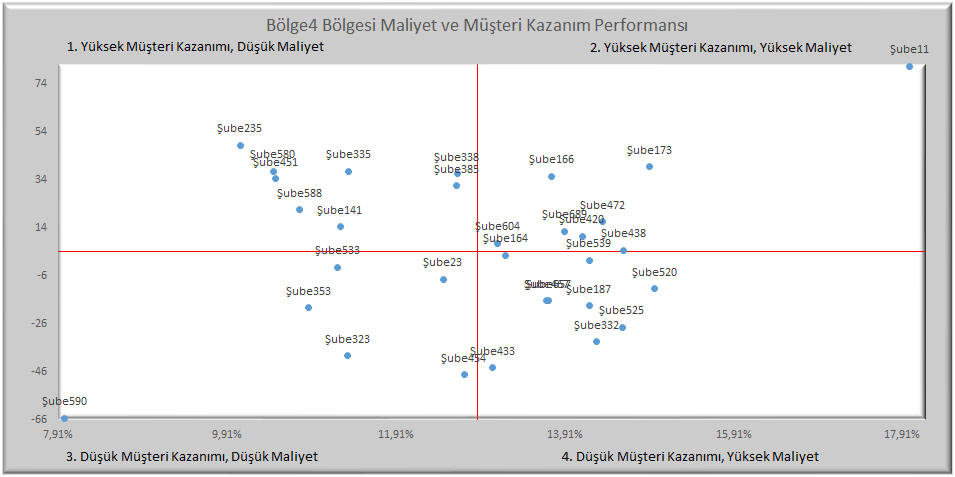
Her bölgenin şubeleri için böyle bir çalışma yapmanız gerekti diyelim. İşler o zaman biraz karmaşıklaşır. Bir kere her bölgenin min/max değerleri farklı olacağı için her bölge seçiminde ona göre min max ve orta noktaların ayarlanması için VBA kullanarak makro yazımı gerekir. Bu arada esas soru seçimin ve seçilen bölgeye göre şubelerin gelme işinin nasıl yapılacağıdır. Bu işlem, Data Validation ile bölge seçimi ve Dinamik Named Range ile şubelerin listelenmesi şeklinde olabileceği gibi, daha basit olarak Table üzerinde Slicer kullanımı ile de sağlanabilir, ancak bunun için en az 2013 versiyonu gereklidir.

İşte şimdi VBA’li örneğe geçebiliriz.

#### VBA’li örnek

(NOT:Bu örnek için iki çözüm metodu olacak, o yüzden indirdiğiniz dosyanın**(pivotdata.xlsb)** en sonunda iki sayfa göreceksiniz)

Bu seferki örneğimizde bir bankanın bölge müdürlüklerini Müşteri artışı ve Vadeli Mevduat faizini(maliyetini) bir grafikte göstereceğiz. Bunu yaparken de her seçilen bölge için hem x hem y ekseninde bir ortalama çizgimiz olsun istiyoruz. Böylece grafiği 4 parçaya ayırmış oluyoruz. Sol üst taraf en iyi bölge olurken sağ alt taraf en kötü bölgeleri göstermektedir.



Her seçim sonrasında seçilen bölgeyi bir yere yazdırmam lazım ki, o bölgeye ait değerleri bir yerde hesaplayabileyim. Bunun için A1 hücresine seçilen bölgeyi yazdırıyorum. T1-T6 arasında da hesaplanan bilgiler bulunuyor. Bunlar;

* Bölge Ort Maliyet : Bölgedeki şubelerin ortalama maliyetidir. AVERAGEIF ile hesaplanır.
* Bölge Ort Müş Artış : Bölgedeki şubelerin ortalama müşteri artışıdır. AVERAGEIF ile hesaplanır.

min maliyet(x) : Bölgede en düşük maliyete sahip şubenin maliyetidir. SMALL-IF mantığındaki bir dizi formülü ile hesaplanır. Dizi formüllerini bilmiyorsanız web sitemdeki [https://www.Excelinefendisi.com/Konular/Excel/FormulasMenusuFonksiyonlar\_DiziFormulleriveSumproduct.aspx](https://www.excelinefendisi.com/Konular/Excel/FormulasMenusuFonksiyonlar_DiziFormulleriveSumproduct.aspx) sayfasına bakabilirsiniz. MIN yerine SMALL-IF kullanma sebebim şudur: Eğer şubelerden birinin maliyeti aşırı uç noktalardaysa, bunu grafikte göstermeyelim, gerekirse grafiğin print edilmiş/sunuma alınmış haline elle müdahale ederiz şeklindeki bir görüşten dolayıdır.(Evet bu örnek bir gerçek dünya örneğidir bu arada ve benden istenen de aynen buydu) Formüllerin yanındaki 1 rakamları varsayılan değerdir, istenirse bunlar değiştirilebilir. Tabi bu değişikliği Worksheet\_Change eventi ile de yakalayacağız. Bölge4, Bölge12 gibi bölgeleri seçerek uç değerlerin grafiğini ne kadar bozduğunu kendini de görebilirsiniz.

* min müş(y) : Yukarıdakinin müşteri adedi artışı için olanı
* max maliyet(x) : Yukarıdakilerin LARGE-IF mantıklı hali
* max müş(y) : Yukarıdakilerin LARGE-IF mantıklı hali

##### Data, Grafik ve Ön Hazırlık

Veri setimiz oldukça basit: Bölge, Şube, Maliyet ve Müşteri Kazanımı bilgilerinden oluşuyor. Biz buraya bir formülle bir kolon daha ekleyip, grafikte label'ın nasıl görüneceğini belirledik. Formül karışık gelebilir, ancak dikkatli analiz edilirse oldukça basit bir formül olduğu görülecektir. Formülde tek atraksiyonlu kısım yukarı ve aşağı ok işaretlerini UNICHAR fonksiyonu ile bulmak oldu. (Görüntü çok kalabalık olduğu için sonra sadece şube isimlerini göstermeye karar verdim)

Sonrasındaki adımlar şöyleydi:

* Veri setini aşağılara aldım ki görünmesin
* Grafiği manuel oluşturdum,
* grafikayar prosedüründeki ayarları macro recordar ile ayarladım
* gridleri manuel sildim
* labelları "low" pozsiyona aldım
* Sırasıyla iki eksen için de seçiliyken font wieght(ağırlık)ını 1 yapıp rengini kırmızı yaptım
* Chart objectin arka planı gri ve bevel>round efekti uygulamadım
* Plot area için de arkaplanı beyaz ve bevel>round yaptım
* Add data label>more options>label options> Value from cells diyerek karışık olan formüllü kolondaki datayı seçtim, X ve Y işaretliyse işareti kaldırdım
* T kolonundaki formülleri oluşturdum.
* Named Range'leri oluşturdum: T3, T4, T5, T6 ve U3: U6 (Değerlerine siz bakarsınız)

Bütün bunlardan sonra ana kodumuz nasıl ona bir bakalım ve girişte bahsettiğim iki çözüm yöntemini ayrı ayrı görelim.

Sub grafikayar()

Dim cht As ChartObject

Set cht = ActiveSheet.ChartObjects("Chart 4") 'Diğer sayfada Chart 2

cht.Activate

With cht.Chart

.Axes(xlValue).MinimumScale = [min\_y] \* IIf([min\_y] < 0, 1.01, 0.99)

.Axes(xlValue).MaximumScale = [max\_y] \* IIf([max\_y] > 0, 1.01, 0.99)

.Axes(xlCategory).MinimumScale = [min\_x] \* IIf([min\_x] < 0, 1.01, 0.99)

.Axes(xlCategory).MaximumScale = [max\_x] \* IIf([max\_x] > 0, 1.01, 0.99)

.Axes(xlCategory).CrossesAt = [blgmaliyet]

.Axes(xlValue).CrossesAt = [blgmusart]

.ChartTitle.Text = [a1].Value & " Bölgesi Maliyet ve Müşteri Kazanım Performansı"

End With

End Sub

Kodun mantığı oldukça basit. Grafiğin en solundaki değeri tüm bankadaki şubelere göre seçme de, bu bölgenin en düşük maliyetli şubesinden birazcık solda seç. "Birazcık" için de IIf([min\_y] < 0, 1.01, 0.99) formülünü kullandım. En sağındaki değeri de o bölgenin en yüksek maliyetli şubesinden biraz büyük seç. Y ekseni yani müşteri artışı için de benzer mantık var.

X ve Y eksenlerinin kesişimi olarak da bölgenin ortalama değerlerini veriyorum. (CrossesAt satırları)

Son olarak da başlığı da seçilen bölgeye göre yazdırıyorum.

Evet şimdi kodun geri kalan kısımları için iki farklı çözüm yöntemine bakabiliriz.

##### 1.Yöntem: PT eventleri ile

İlk yöntem Worksheet\_Calculate aklıma gelmediği için düşündüğüm uzun yol yöntemi. Burada PT eventlerinden faydalanacağız. Aslında 2.yöntem olan Calculate yöntemini gördükten sonra "1.yönteme ne gerek var, niye koydun ki?" diye sorabilirsiniz. Şundan; burada naçizane yaratıcı bir çözüm var. Microsoft tarafından dokumante edilmemiş farklı bir yöntem uygulaması var. Calculate eventi diye bir event olmasaydı nasıl bir çözüm üreteceğimize dair programatik bir bakış açısı var. İşte bu bakış açısı başka bir yerde yine lazım olabilir. Nasıl esnek düşünmeniz gerektiğine dair güzel bir örnek bence.

Burada yaptığımız şey şu:

* Verimizin bulunduğu Table'ın hemen altına Bölge isimlerinden oluşan bir Pivot tablo yarattım.
* Bu PT'ı kontrol eden de bir Slicer oluşturdum ve bunu en yukarıya koydum. Kullanıcı buna bastığında sanki Table'ın Slicerına bastığını sanacak
* Table'a ait bir Slicer da oluşturdum(Zorunda değildim ama yukarıda öğrendiklerimizi uygulamak adına iyi olur dedim, yoksa doğrudan Table üzerinde de filtreleme yapabilirdik). Bu slicerı aşağıda bıraktım, Shapes collection'ından yaralanıp gizli hale de getirebilirdim.
* Burada kullanıcı, PT Slicerından ne zaman eleman seçse Worksheet\_PivotTableChangeSync eventi tetikleniyor ve seçilen değeri bulup bunu hem A1'e yazıyor ki böylece grafiğin değerleri yeniden belirlensin, hem de Table'ın Slicerında bu seçimi yaptırıyor ki grafik yeni bölgeye göre güncellensin.
* Sayfada yapılan değişiklikler Change eventini tetiklemesin diye eventler baskılanıyor
* Seçim sonucunda tüm filtre kaldırılıyorsa Table Slicerında da aynısını yaptırıyorum, filtre değişiyorsa bunu buldurup esas işlemleri yaptırıyorum.

Şimdi kodumuza bakalım.

Private Sub Worksheet\_PivotTableChangeSync(ByVal Target As PivotTable)

Dim spi As SlicerItem, sti As SlicerItem

On Error GoTo hata

Application.EnableEvents = False: Application.ScreenUpdating = False

[minmax].Value = 1 'Bir önceki bölgede manuel oalrak farklı bişey yaılzdıysa baştan tekrar hepsin 1 yapalım

If ActiveWorkbook.SlicerCaches("Slicer\_Bölge4").VisibleSlicerItems.Count = ActiveWorkbook.SlicerCaches("Slicer\_Bölge4").SlicerItems.Count Then 'pivotun slicerı

ActiveWorkbook.SlicerCaches("Slicer\_Bölge5").ClearManualFilter 'Tableınki, görünmezdir, veya aşağılarda görünmez olsa da olur

GoTo çıkış

End If

secilen = SlicerdaSeçilileriGetir("Slicer\_Bölge4")(1)

slicerda\_tek\_eleman\_seç "Slicer\_Bölge5", secilen

Application.EnableEvents = False 'yukardakilerden döndüğünde True oluyor, tekrar false yapalım

çıkış:

[a1].Value = secilen

Call grafikayar

Application.EnableEvents = True: Application.ScreenUpdating = True

Exit Sub

hata:

Application.EnableEvents = True: Application.ScreenUpdating = True

MsgBox Err.Description

End Sub

Bir de minmax range'inde SMALLIF ve LARGEIF değerlerini etkileyecek bir değişiklik yapıldığında bunları yakalıyoruz demiştik, onun kodu da aşağıda.

Private Sub Worksheet\_Change(ByVal Target As Range)

On Error GoTo hata

Application.ScreenUpdating = False

If Not Intersect(Target, [minmax2]) Is Nothing Then 'bölge seçildiğinde de değişiklik oluyor, bu tetiklemesin

Call grafikayar

End If

Application.ScreenUpdating = True

Exit Sub

hata:

Application.ScreenUpdating = True

MsgBox Err.Description

End Sub

##### 2.Yöntem: Calculate eventi ile

Bu yöntem daha sadedir. Table'da filtre konup konmadığını anlamanın yine doğrudan bir yolu yok ancak bunun için Table'ın altına bir SUBTOTAL formülü koyunca table her filtrelendiğinde bu formül de yeniden hesaplanmış olacak, ki bu da Calculate eventini tetikler, işte bu değişikliği yakalıyoruz ve akabinde seçilen değeri bulup grafikayar prosedürünü çağırıyoruz.

Dikkat: Calculate eventi bazen beklenmedik şekilde çalışabilir. Yani siz o sırada başka sayfada işlem bile yapsanız Calculate eventinin bulunduğu sayfa yine de tetiklenebiliyor. Bunun sebebi ya o sayfada diğer sayfalardan etkilenen formüller vardır ya da volatil bir fonksiyon kullanmışsınızdır. Gerçi bizim sayfamızda ikisi de yok, çok araştırdım ama sebebini bulamadım. Ben de aşağıdaki gibi sayfa adını kontrol etmekle çözdüm bu sorunu. Bunu da yapamasaydım 1.yöntemle ilerlerdim.

Private Sub Worksheet\_Calculate()

If ActiveSheet.Name <> "Örnek UygulamaTableEventli" Then Exit Sub

Application.EnableEvents = False

secilen = SlicerdaSeçilileriGetir("Slicer\_Bölge3")(1)

[a1].Value = secilen

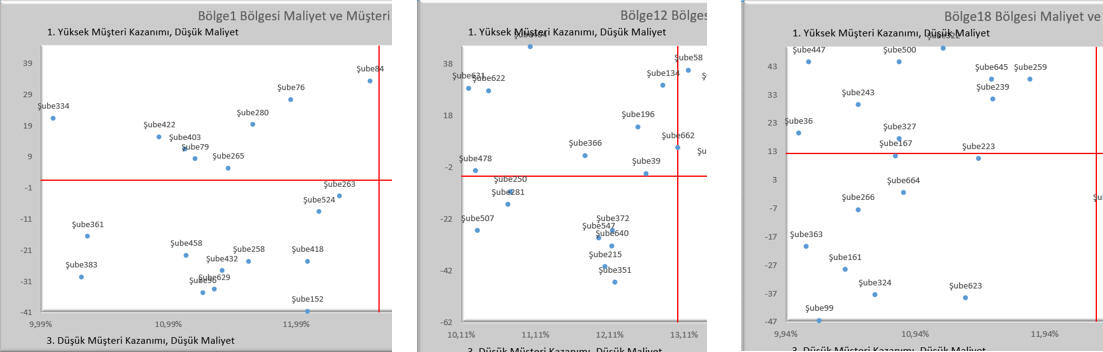
Call grafikayar

Application.EnableEvents = True

End Sub

#### Sonuç

Aşağıdaki resimlerden, bölgeler değiştikçe ortalama çizgilerinin (hem x hem y) nasıl değiştiğini görebilirsiniz.



|  |  |  |
| --- | --- | --- |
|  |  |  |

## Add-inler yaratmak ve Menülerle çalışmak

### Giriş

#### Add-in nedir?

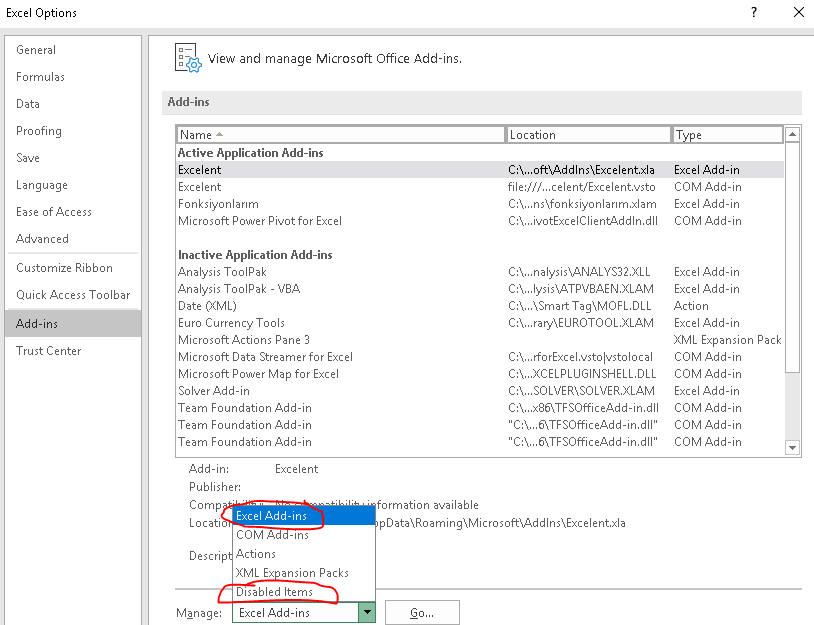
Add-inler, Excel'in fonksiyonalitesini artıran araçlardır. 3 çeşidi bulunur:

* **Excel Add-in**: Bunlar kendi içinde ikiye ayrılır. XLA/XLAM uzantılı olanlar VBA ile geliştirilirler. Bu sayfada bunlar incelenecektir. Bir de XLL uzantılı olanlar vardır, bunlar C/C++ ile yazılırlar ve genelde UDF yazmak için kullanılırlar. VBA ile yazılan kodlara göre çok daha hızlıdırlar. Ancak biz bu kitapta bu tür add-inlere hiç değinmeyeceğiz.
* **COM Add'in**: VSTO teknolojisi ile geliştirilirler. VBA tecrübesi olan birinin VB.Net'i(ve/veya C#'ı) hızlıca öğrenmesi akabinde geliştirilebilir. Benim de kendime ait bir COM add-in'im([https://www.Excelinefendisi.com/Excelent.aspx](https://www.excelinefendisi.com/Excelent.aspx)) bulunmaktadır.
* **Office/Web Add-ins**: Javascript ile web tabanlı geliştirilirler. En son çıkan yöntemdir. Microsoft bundan sonra bu konuya ağırlık verecek gibi görünüyor, ancak hiç javascript bilmeyen birinin harcı değildir. Genelde ticari amaçlı olarak oluşturulurlar.

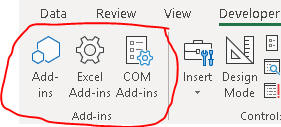
Bunların dışında bir de VBE add-inleri vardır, ki bunların esas amacı Excel ön yüzde değil, VBE'de kodlama yaparken hayatımızı kolaylaştırmaktır. [http: //codeVBA.com/](http://codevba.com/) sayfasında böyle bir add-in bulunmaktadır. Özellikle VBA dünyasında yeniyseniz ve kodları hızlı bir şekilde yazamıyorsanız bu add-in'in size sunduğu hazır kodlar oldukça işinize yarayacaktır. Normalde ücretli ancak 30 günlük deneme süresince kullanıp karar verebilirsiniz.

#### Erişim

Biz bu sayfada yeni bir Excel Add-in yapmayı göreceğiz ancak hali hazırda Microsoft tarafından eklenmiş mevcut Excel Add-inlere(Solver, AnalysisToolpak gibi) **File>Options>Add-ins** mensünden ulaşabiliyoruz.



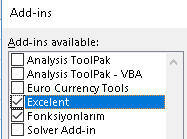
Bir diğer ve hızlı alternatif de, Developer menüsünden ulaşmaktır.



Ancak o anda hiç açık sayfa yoksa bu kısım inaktif gelir.



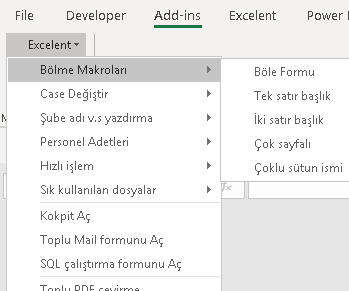
Excel kurulduğunda bunlardan bazısı hazır olarak gelir ancak hiçbiri seçili değildir. Bunlardan Analysis ToolPak ve Solver çok sağlam araçlardır. Bunları Excel ana menüsünde ele alacağız.



#### Kullanım amacı

Üst seviyede baktığımızda, kendimiz için olsun başkalarının kullanımı için olsun fark etmez, kodlarımızı bir workbook içine koyup insanlara dağıtmanın daha iyi bir alternatifidir Add-inler. Zira, o workbook içindeki kodlarımız sadece o workbookta kullanılır. Personal.xlsb haline getirip göndersek bile kullanımı zor olur, zira bir arayüz sağlamamış oluruz. Belki bir userform yapıp, workbook\_open içine de bu formu açan bir makro yazabiliriz ama bu formun kullanımı add-inlerin menülü kullanımı kadar pratik olmayacaktır. Ayrıca UDF'ler için hala dosya adını uzun uzun yazmanız gerekecektir: **Personel.xlsb!myfunction** gibi. İşte add-inler bizi bu zahmetlerden kurtarır.

Amacımız, Solver veya kendi oluşturduğumuz Toplu Mail Gönderim Formu gibi bir tool'un çalışmasını sağlamak olabileceği gibi aşağıdakine benzer bir menü oluşturup çeşitli makrolara kolay erişim arayüzü sağlamak da olabilir.



Özetleyecek olursak, add-inler;

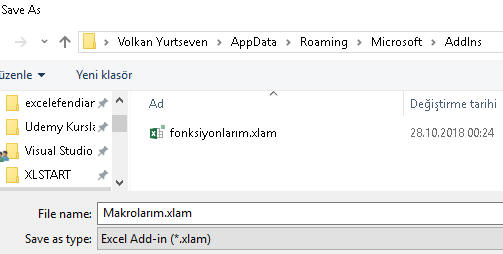
* UDF'leri depolamak için en uygun yerdir
* Hazırladığınız tool'lara ulaşmak için menüler sağlar
* Bir görev yerine getiren makrolarınızı çalıştırmak için menüler sağlar

#### Yaratım

Makrolardan oluşan dosyamızı **Save As** dedikten sonra **xlam** uzantılı olarak kaydederiz.(2007 öncesi versiyonlarda yaratılan add-inler xla uzantılıdır)

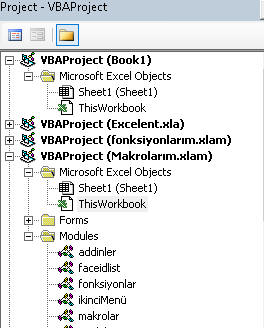
Bu add-ini istediğimiz yere kaydedebiliriz ancak default add-in adresine yerleştirmek ona ulaşmak adına da kolaylık sağlayacaktır. Default adres, Immediate Window’a **Application.UserLibraryPath** yazarak elde edilebilir, bende "C: \Users\Volkan\AppData\Roaming\Microsoft\AddIns\" gibi geliyor.(Application.LibraryPath komutu ile Solver gibi built-in add-inlerin adresine ulaşırız)

**NOT**: Bir şekilde Excel, add-ininizi kaydetmezse, **File>Open** diyerek ilgili adresten add-ini açmaya çalışın. Bu durum, add-inlerle ilgili bilinen bir problemdir.



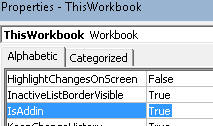
Bu arada örnek dosyaya **githubdan(makrolarım.xlam)** ulaşabilirsiniz.

Ok der demez dosya Excel önyüzden kaybolur. Onları ancak VBE ortamında görürüz.(XLA/XLAM olanlardan bahsediyorum, zira XLL uzantılı Excel add-inleri VBE'de dahi görünmezler)



##### IsAddin özelliği

VBE'ye geçip ThisWorkbook'tayken Properties'e bakınca **IsAddin** özelliğinin True olduğunu görürüz. Bunu False yaparsak dosyamız Excel önyüzde de görünür, bu bilgiyi niye verdim, çünkü bir dosyayı add-in yaptığımızda bu bilgisine True atanır. Bunu gösterme sebebi biz değiştirebilelim diye değil, nesne tabanlı bir dil olduğu için mecburen gösterir. Siz siz olun buna hiç dokunmayın.



Şimdi, sırada, bu add-ini içine birkaç fonksiyon yerleştirme var.

#### Fonksiyonları ekleme

Yeni bir modül açıp buna "fonksiyonlar" adını verelim, sonra buraya bir iki fonksiyon ekleyelim.

Function hucresatırno(hucre As Range)

hucresatırno = hucre.Row

End Function

Function hucreadres(hucre As Range)

hucreadres = hucre.Address

End Function

#### Makrolar ekleme

Şimdi de geçici olarak VBE içinden çalıştırılabilecek makro kodları yazalım, bunlara bir alt kısımda menülerden ulaşmayı da göreceğiz.

Aşağıdaki iki örnek kod, daha önce başka sayfalarda gördüğümüz toplu rakam güncelleme makrolarına aittir.

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*rakam güncelleme bölümü\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sub cBin()

Dim rakamlar As Variant 'Dizi tanımı

rakamlar = Selection.Value 'burada hücreden diziye okuma yaptık

For i = LBound(rakamlar) To UBound(rakamlar) 'i'ler satır boyutudur

For j = LBound(rakamlar, 2) To UBound(rakamlar, 2) 'j'ler satır boyutudur

rakamlar(i, j) = rakamlar(i, j) \* 1000

Next j

Next i

Selection.Value = rakamlar

End Sub

Sub bBin()

Dim rakamlar As Variant 'Dizi tanımı

rakamlar = Selection.Value 'burada hücreden diziye okuma yaptık

For i = LBound(rakamlar) To UBound(rakamlar) 'i'ler satır boyutudur

For j = LBound(rakamlar, 2) To UBound(rakamlar, 2) 'j'ler satır boyutudur

rakamlar(i, j) = rakamlar(i, j) / 1000

Next j

Next i

Selection.Value = rakamlar

End Sub

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*dosya açma bölümü\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sub abc\_ac()

'gerçek dosya açmak yerine geçici olarak bir mesajbox çıkaralım

MsgBox "abc dosyası"

End Sub

Sub xyz\_ac()

'gerçek dosya açmak yerine geçici olarak bir mesajbox çıkaralım

MsgBox "xyz dosyası"

End Sub

Şimdi bir de boş bir Userform oluşturalım, ve bu formu açacak küçük bir prosedür daha yazalım, ve bunu makrolarım modülü içine koyalım.

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*form açma bölümü\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sub formac()

UserForm1.Show vbModeless

End Sub

Add-inimiz bu haliyle çok kullanışlı değildir ancak bu haliyle bile dağıtıma uygundur. Dosyayı alan kullanıcılar, içindeki fonksiyonları ve makroları tüm dosyalarında kullanabilirler.

### Add-inimiz için menü oluşturma

#### Menü dizaynı

Dedik ki, add-inlerin normalde IsAddin özelliği True atandıkları için görünmezdirler. O zaman bunlarla nasıl etkileşime geçeceğiz? Hadi biz işi biliyoruz, VBE ortamına geçip oradan makrolarımızı çalıştırabiliriz ancak bu add-ini kullanacak son kullanıcılar için bu yöntem pek uygulanabilir bir yöntem değildir, üstelik şık da değildir. Bizim için bile çok pratik değildir, her defasında gidip VBE'den kod çalıştıracak değiliz. İşte add-inlerdeki makroların kolay kullanılabilmesi için menüler burada devreye girer.(Fonksiyonlar zaten her halükarda Excel ön yüzde kullanılabilir durumdadır)

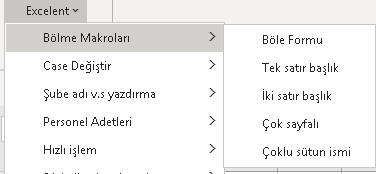
Şimdi, menüler oluşturup bunlar elimizdeki prosedürlerimizi yerleştirmeye çalışalım. Bu arada tabiki içiçe alt menüler oluşturulabilir ancak bunlar pek sevimli olmaz, siz 1 taneyle yetinmeye çalışın. Şöyle yapmak yerine,

* AnaMenü
  + alt menü1
    - alt1.1
      * buton1
      * buton2
      * ...
    - alt1.2
      * buton1
      * ...
  + alt menü2
    - * ....

Şöyle tercih edilmelidir. Aksi halde kullanıcılar menülerde kaybolur.

* AnaMenü
  + alt 1.1
    - buton1
    - buton2
  + alt 1.2
    - buton1
    - buton2

Aşağıda örnek uygun bir örnektir.



##### Menü nesneleri

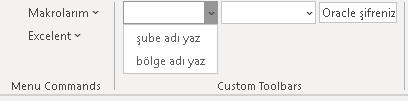
Öncelikle kavram karmaşasını netleştirmek adına elimizdeki kavramları bir ele alalım.

**Menü**: Konuşma dilinde, tıklandığında altında alt menüler ve butonlar çıkan nesnelerdir. VBA'de böyle resmi bir nesne **bulunmamaktadır**. Biz bu ifade ile hem CommandBarları hem de PopupControlleri kastedeceğiz. Kullandığımız bağlama göre anlamı değişebilir.

**CommandBar**: Biz çok fazla CommandBar oluşturmayacağız, zira Excel 2007 ile gelen Ribbon arayüzünden sonra commandbarların bir esprisi kalmamıştır. Zaten eklenen tüm commandbarlar olsun popupcontroller olsun hep ribbonda **Addins** menüsü altına giderler. Biz yeni Commandbar yaratmak yerine var olan CommandBarlar üzerine PopupControl yerleştireceğiz. Tabi örnek olmasında adına küçük bir CommandBar da yaratacağız.

**PopupControl**: Bunları hem **Ana Menüleri** oluşturmak için kullanacağız. hem de **Alt menüleri** oluşturmak için kullanacağız. Ben CommandBara yerleştirilen ilk popupcontrole Ana menü, onun altına yerleştirilen popupcontrollere ise Alt menü diyorum. Yukarıdaki resimde "Excelent" bir Ana menüdür, "bölme makroları" ise bir alt menü. Ana menüler aşağı doğru açılır ve menü oku aşağı yöndedir. Alt menüler yana doğru açılır ve menü okunun yönü de sağa doğrudur. Ama sonuçta ikisi de bir PopupControldür.

Aşağıdaki resimde Makrolarım ve Excelent menüleri, mevcut CommandBarlardan 1 numaralı commandbar olan "Worksheet Menü Bar"'a birinci seviyeden eklenen Ana Menülerdir. Yan taraftaki Custom Toolbars kısmında eklenen menü(adı "Benim Menüm") ise bir Commandbar olup, seviye olarak diğer commandbarlarla aynı seviyededir.



Şimdi diyeceksiniz ki, diğer commandbarlar nerede? Ribbon arayüzü geldiği için artık bunlar yok. Excel 2003 kullananlar bunları hatırlayacaktır. Gerçi bunlar bir şekilde gizli olarak duruyor gibidirler. Zira şu kodu çalıştırırsanız tüm commandbarları görebilirsiniz.(Ben çok şişmesin diye ilk 10unu yazdırdım)

Sub ribbonöncesimenülerilisteleme()

For k = 1 To 10 ' Application.CommandBars.Count

Debug.Print k & "-" & Application.CommandBars(k).Name

Debug.Print "---------------------------"

For i = 1 To Application.CommandBars(k).Controls.Count

Debug.Print Application.CommandBars(k).Controls(i).Caption

Next i

Debug.Print vbNewLine

Next k

End Sub

Bunun sonucu aşağıdaki gibi olacaktır. Fark ettiyseniz Worksheet Menu Bar'ın en sonunda benim eklediğim "Makrolarım" isimli popupcontrol duruyor.



Aşağıdaki kodu çalıştırdığınızda ise visible olan commandbarları görebilirsiniz.

Sub barları\_listele()

Dim bar As CommandBar

For Each bar In Application.CommandBars

If bar.Visible = True Then

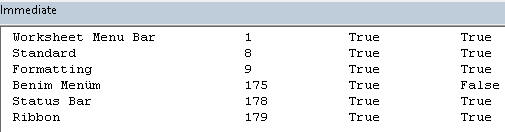
Debug.Print bar.Name & WorksheetFunction.Rept(" ", 20 - Len(bar.Name)), bar.Index, bar.Visible, bar.BuiltIn

End If

Next

End Sub

Bu kodu ben çalıştırdığımda aldığım sonuç aşağıdaki gibi.



Gördüğünüz gibi bir tek "Benim Menüm" Built-In değeri False çıktı.

##### Diğer nesneler

Popupcontrollere ya bir başka popupcontrol(alt menü olarak) ya da bir buton ekleyebiliyoruz. Ve günün sonunda bu butonlara da çeşitli makrolar atayacağız.

Commandbarlarda ise durum biraz farklı. Popupcontroller altına Combobox, Editbox(Textbox gibi bir control) veya Dropdown koyamıyorken Commandbarlara koyabiliyoruz.

##### Nesneler özeti

Elimizdeki nesnelere özetle tekrar bir bakalım:

**CommandBar**: Hiyerarşinin en üstünde bulunur. Ribbondan sonra anlamsızlaşmıştır ancak yine de kullanılabilmektedir.

**CommandBarPopup**: CommandBar altında eklenen AnaMenü/AltMenü kontrolleri için kullanılırlar.

**CommandBarButton**: Menülerdeki makroları çalıştırmak için kullanılırlar.

**CommandBarControl**: CommandBar nesnesine eklenen EditBox(Textbox) ve DropDown nesnelerini temsil etmek için kullanılırlar. MsoControlType parametresine atanan değerle ne olacağı belirlenir. Bunlara da makro atanabilmektedir.

**CommandBarComboBox:** Nedendir bilmem, bunun için ayrı bir nesne yaratılmış. Açıklamaya gerek yok, adı gayet açıklayıcı diye düşünüyorum. Bunlara da makro atanabilmektedir.

##### Kod Yazımı

###### Yöntem tespiti

Kodları yazarken farklı yöntemler benimsenebilir. Şöyle ki, Excel açılırken Workbook\_Open içine menüleri kalıcı olarak ekleyip, Excel’den çıkılırken Before\_Close içine bunları kaldıran bir kod yazmak bir çözüm olabilir. Araştırma yaparken bu tür kodları çok görebilirsiniz. Ben ise, Workbook\_Open içine menüleri geçici olarak ekletiyorum, Before\_Close içine ise hiçbir şey yazmıyorum, böylece daha az kod yazmış oluyorum.

Bir de yükleme ve kaldırma event prosedürlerine yazılacak kodlar var, onları biraz daha aşağıda ele alacağız.

###### Kodlar

Şimdi, ilk olarak Workbook\_Open eventini yazalım. Zira add-inimizi bir kez yükledikten sonra Excel her açılışta bu add-in de arka planda açılacaktır. Yukarıda benimsediğimiz yönteme göre menüleri her açılışta  baştan oluşturmamız gerekecek.

Bu kısımda söylemek istediğim en önemli husus, kontroller eklenirken **Temporary** parametresine True değerini atayarak ilgili kontrolü geçici olarak eklemiş olduğumuzdur. Yukarıda yöntem kısmında belirttiğim işi böyle yapıyorum işte. Diğer gerekli açıklamaları kod içine comment olarak yazdım.

Private Sub Workbook\_Open()

Call anaMenuEkle 'ilk olarak standart bir popup ana menü ekliyoruz

Call DropDownMenü 'Şimdi de yeni bir commandbar ekleyeceğiz

End Sub

İlk olarak ana menü kodumuza bakalım

Sub anaMenuEkle()

Dim bar As CommandBar 'Mevcut bir commandbara erişmek için

Dim Menu As CommandBarPopup 'Anamenü için

Dim altMenu As CommandBarPopup 'alt menü yaratmak için. Bu değişkeni birkaç kez kullanaceğız

Dim Dugme As CommandBarButton 'klasik buton. Bu değişkeni birkaç kez kullanaceğız

'Menülerin ekleneceği commandbarı seçelim

Set bar = Application.CommandBars(1) '1: "Worksheet Menu Bar", veya 9: "Formatting" de yazılabilir

'Öncelikle Menü çubuğuna "Makrolarım" isminde Ana menümüzü ekleyelim.

Set Menu = bar.Controls.Add(msoControlPopup, , , , True) '1 nolu 'Worksheet Menu Bar' menüsü altına bir Control(popupcontrol) olarak ekleniyor

'id: Burası hep 1. 1den sonrakiler built-in butonlar ekler, Ör: 3 Save butonu ekler

'parameter: boş geçilir

'before: boş geçilir

'temporary: True parametresi ile ilgili kontrolü geçici yaratmış oluyoruz

With Menu

.Caption = "Makrolarım"

.Tag = "etiket"

.BeginGroup = True

End With

'1)şimdi ilk altmenuyu ekleyelim

Set altMenu = Menu.Controls.Add(msoControlPopup, 1, , , True)

With altMenu

.Caption = "Rakam güncelleme"

End With

'şimdi de makroları çalıştıracak butonları ekleyelim

Set Dugme = altMenu.Controls.Add(msoControlButton, 1, , , True)

With Dugme

.Caption = "\*1000"

.OnAction = "cBin"

.TooltipText = "1000le çarpar"

.Style = msoButtonIconAndCaption

.FaceId = 71 'FaceId ekleyebilmek için bir üstteki gibi style = msoButtonIconAndCaption olmalı

End With

Set Dugme = altMenu.Controls.Add(msoControlButton, 1, , , True)

With Dugme

.Caption = "/1000"

.OnAction = "bBin"

.TooltipText = "1000e böler"

.Style = msoButtonIconAndCaptionBelow

'.FaceId = 72

.Picture = stdole.StdFunctions.LoadPicture("C: \inetpub\wwwroot\aspnettest\Excelefendiana\images\VBAuserform4.jpg")

End With

'+ ve - için örnekler çoğaltılabilr

'2)şimdi ikinci altmenuyu ekleyelim

Set altMenu = Menu.Controls.Add(msoControlPopup, 1, , , True)

With altMenu

.Caption = "Dosya açma"

End With

'şimdi de makroları çalıştıracak butonları ekleyelim

Set Dugme = altMenu.Controls.Add(msoControlButton, 1, , , True)

With Dugme

.Caption = "abc dosyasını aç"

.OnAction = "abc\_ac"

.Style = msoButtonIconAndWrapCaption

.FaceId = 73

End With

Set Dugme = altMenu.Controls.Add(msoControlButton, 1, , , True)

With Dugme

.Caption = "xyz dosyasını aç"

.OnAction = "xyz\_ac"

.Style = msoButtonIconAndWrapCaptionBelow

.FaceId = 74

End With

'başka dosyalar da ekleneblir

Set Dugme = Menu.Controls.Add(msoControlButton, 1, , , True)

With Dugme

.Caption = "Form aç"

.OnAction = "formac"

.Style = msoButtonWrapCaption

.FaceId = 75

.BeginGroup = True

End With

'başka düğmeler de eklenebilir

End Sub

Şimdi de bir popupmenü değil yeni bir commandbar yaratmış olacağız.

Sub DropDownMenü()

Dim bar As CommandBar

Dim combobox As CommandBarComboBox 'combobox

Dim textordrop As CommandBarControl 'Textbox(EditBox) ve DropDown için

Set bar = Application.CommandBars.Add("Benim Menüm", MsoBarPosition.msoBarFloating, False, True)

'position: Ne yazdığımız çok fark etmiyor, ben boş bırakmayı veya msoBarFloating yazmayı tercih ediyorum

'menubar: Aktif menübarı bununla yer değiştirmek istiyorsanız True. Ben hep False bırakırım, default'u da False'tur.

'temporary: Commandbar geçici kılmak için True atarız, böylece Excel kapanınca yok edilirler. Default değeri False'tur. Ben hep True bırakım

bar.Visible = True

Set combobox = bar.Controls.Add(MsoControlType.msoControlComboBox, 1, , , True)

With combobox

.AddItem "şube adı yaz"

.AddItem "bölge adı yaz"

.Caption = "ben comboyum"

.OnAction = "comboMakro"

'.ID = 1 'Read-only. A control's ID determines the built-in action for that control. The value of the Id property for all custom controls is 1.

.Tag = "mycombo" 'Gets or sets information about the CommandBarControl, such as data that can be used as an argument in procedures, or information that identifies the control. Read/write.

.TooltipText = "otomatik şube/bölge isim lookupları için seçim yapın"

End With

Set textordrop = bar.Controls.Add(MsoControlType.msoControlDropdown, 1, , , True)

With textordrop

.AddItem "bireysel temsilci adedi yaz" 'normalde intellisende bu çıkmaz ama manuel yazınca yine de çalışır

.AddItem "ticari temsilci adedi yaz"

.Caption = "ben dropdownım"

.OnAction = "dropMakro"

.Tag = "mydrop"

.TooltipText = "otomatik personel adetleri için seçim yapın"

End With

Set textordrop = bar.Controls.Add(MsoControlType.msoControlEdit, 1, , , True)

With textordrop

.Text = "Oracle şifrenizi girin"

'.OnAction = "" textbox için bir makromuz olmayacak

End With

End Sub

Yukarıda kullanılan bazı özelliklerin açıklaması şöyle:

**BeginGroup**: Yatay bir kesme işaretiyle butonlar gruplandırılmak istendiğinde bu özelliğe True atanır.

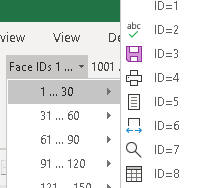
**OnAction**: Butona basıldığında hangi makro çalışsın.

**Tag**: Butona verdiğimiz bir etiket adıdır. Kendisine bu etiket adı kullanılarak da ulaşılabilir.

**ToolTipText**: Formlarda da gördüğümüz bir özellikti. Fare ile buton üzerine geldiğinizde o butonun kısaca ne yaptığını anlatan açıklamalar eklemeye yarar.

**Style**: Butonda sadece metin mi, yoksa metin+resim mi olacağını belirtiriz.

**FaceId**: Butonlarımızda belirli sembolleri kullanmak istediğimizde kullanırız. Bunların listelerine [http://supportingtech.blogspot.com/2011/03/microsoft-faceid-numbers-for-VBA.html](http://supportingtech.blogspot.com/2011/03/microsoft-faceid-numbers-for-vba.html) sayfasından ulaşabilirsiniz. Veya örnek dosyadaki faceid modülündeki kodu çalıştırabilirsiniz. Kod çalıştığında aşağıdaki gibi bir commandbar Addins menüsüne eklenecektir.



FaceID kodu aşağıda da bulunmaktadır.(Kod, [https://stackoverflow.com/questions/7382624/could-anyone-show-list-of-button-face-id-in-Excel-2010/40134814](https://stackoverflow.com/questions/7382624/could-anyone-show-list-of-button-face-id-in-excel-2010/40134814) sayfasından alınmıştır)

Const APP\_NAME = "FaceIDs (Browser)"

' The number of icons to be displayed in a set.

Const ICON\_SET = 30

Sub BarOpen()

Dim xBar As CommandBar

Dim xBarPop As CommandBarPopup

Dim bCreatedNew As Boolean

Dim n As Integer, m As Integer

Dim k As Integer

On Error Resume Next

' Try to get a reference to the 'FaceID Browser' toolbar if it exists and delete it:

Set xBar = CommandBars(APP\_NAME)

On Error GoTo 0

If Not xBar Is Nothing Then

xBar.Delete

Set xBar = Nothing

End If

Set xBar = CommandBars.Add(Name: =APP\_NAME, Temporary: =True) ', Position: =msoBarLeft

With xBar

.Visible = True

'.Width = 80

For k = 0 To 4 ' 5 dropdowns, each for about 1000 FaceIDs

Set xBarPop = .Controls.Add(Type: =msoControlPopup) ', Before: =1

With xBarPop

.BeginGroup = True

If k = 0 Then

.Caption = "Face IDs " & 1 + 1000 \* k & " ... "

Else

.Caption = 1 + 1000 \* k & " ... "

End If

n = 1

Do

With .Controls.Add(Type: =msoControlPopup) '34 items \* 30 items = 1020 faceIDs

.Caption = 1000 \* k + n & " ... " & 1000 \* k + n + ICON\_SET - 1

For m = 0 To ICON\_SET - 1

With .Controls.Add(Type: =msoControlButton) '

.Caption = "ID=" & 1000 \* k + n + m

.FaceId = 1000 \* k + n + m

End With

Next m

End With

n = n + ICON\_SET

Loop While n < 1000 ' or 1020, some overlapp

End With

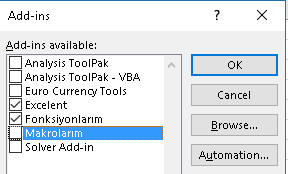
Next k

End With 'xBar

End Sub

#### Add-in’leri kurma ve kaldırma

Add-in'leri kurmak oldukça basittir. Öncelikle aşağıdaki pencerede Browse deyip ilgili add-ini listeye eklemelisiniz. Ancak eklemek yeterli değil, onun işaretlenmesi de gerekmektedir.



İlgili add-ini işaretlediğiniz anda kurulmuş olur. Ancak menüleriniz hemen aktif olmaz. Excel'den çıkıp tekrar girmeniz gerekir, ki bu anlamsız bir süreç olur. Onun yerine menü aktivasyonunun yüklemeyle paralel gerçekleşmesini sağlayabilecek bir yöntem var: Workbook\_AddInInstall eventinden yararlanmak. Bu prosedür, kurulum yapılır yapılmaz devreye girer. Bu event prosedüründe ise doğrudan yukarıda gördüğümüz Workbook\_Open prosedürünü çağırırız.

Private Sub Workbook\_AddinInstall()

Call Workbook\_Open

End Sub

Aslında normalde bu prosedüre gerek yoktur, zira add-in yüklendiğinde zaten Workbook\_Open da çalışıyor, o yüzden Workbook\_Open içindeki kodların iki kez çalışması söz konusu olabilir. İşyerindeki bilgisayarımda Workbook\_AddinInstall prosedürüne gerek duymazken evdeki bilgisayarda sebebini bilmediğim bir nedenden ötürü gerek duyuluyor. Burda size bir kontrol görevi düşüyor: Eğer menününüz iki kere eklenmiş olursa bu kısmı iptal edebilirsiniz.

Yukarıdaki add-in penceresinden seçimi kaldırdığımızda yüklemeyi de iptal etmiş oluruz. Bu yüzden ilgili menülerinde yok olmasını bekleriz. Ancak hemen kaybolmazlar. hemen kaybolmasını istiyorsak aynı şekilde aşağıdaki event prosedürünü yazarız.

Private Sub Workbook\_AddinUninstall()

On Error Resume Next 'öngörülemeyen durumlar için gerekebiliyor

Application.CommandBars.FindControl(, , "etiket", False).Delete 'tag ile silme

Application.CommandBars("Worksheet Menu Bar").Controls("Makrolarım").Delete 'popupcontrolü, control adı ile silme

Application.CommandBars("Benim Menüm").Delete 'commandbarı, commandbar adı ile silme

End Sub

##### İdeal Süreç

Yukarıda biraz kafanızı karıştırmış olabilirim. Şimdi size, bir add-in'i kurmanın her türlü senaryosuna karşı dayanıklı bir kod dizisi vereceğim. Buna göre ister Excel o sırada kapalı ister açık olsun, ister Add-ins penceresinde yer alıyor ister yer almıyor olsun, ister bu pencerede seçim yapılmış(kurulmuş) olsun ister olmasın fark etmez, her şekilde sağlıklı çalışacak bir kurgu.

'Bu şekilde kullanıcıya options menüsüne girdirmeye gerek kalmıyor, direkt kendisi istediği yere kaydedip 1 kere açması yeterli.

Private Sub Workbook\_AddinInstall()

'manuel eklenirse menüyü de oluşturalım

Call anamenuekle

End Sub

Private Sub Workbook\_AddinUninstall()

On Error Resume Next 'öngörülemeyen durumlar için gerekebiliyor

Application.CommandBars.FindControl(, , "Excelent", False).Delete 'tag ile silme

ThisWorkbook.Close False 'bu satırı sonra ekledim

End Sub

Function AddinIsLoaded(AddinName As String) As Boolean

On Error Resume Next

AddinIsLoaded = Len(Workbooks(AddIns(AddinName).Name).Name) > 0

End Function

Private Sub Workbook\_Open()

'Excel direkt bu add-ine tıklanarak açılırsa boş sayfa olmayacağı için add-in pencere komutu çalışmaz,

'bi tane geçici dosya açalım, ama gizli dosyalar açık olabilir diye activewb var mı diye bakıyoruz

If ActiveWorkbook Is Nothing Then

Workbooks.Add

End If

If Not AddinIsLoaded("Excelent") Then

'doğrudan dosyayı açarsa, ilk başta yüklü olmayacağı için önce bi yüklesin, sorna install eventi devreye girecek

Set myAddIn = Application.AddIns.Add(ThisWorkbook.FullName)

myAddIn.Installed = True

Call anamenuekle

Else

'demekki daha önce manuel yüklenmiş, sadece menüyü eklesin

Set myAddIn = Application.AddIns.Add(ThisWorkbook.FullName)

'Workbook\_AddinInstall eventi devreye girmesin diye geçici olarak eventleri kapatıyoruz

Application.EnableEvents = False

myAddIn.Installed = True

Application.EnableEvents = True

Call anamenuekle

End If

End Sub

Sub anamenuekle()

'burada menüler oluşturulur

End Sub

Private Sub kaldir(customcontroltag As String)

On Error Resume Next

Do

Application.CommandBars.FindControl(, , customcontroltag, False).Delete

Loop Until Application.CommandBars.FindControl(, , customcontroltag, False) Is Nothing

On Error GoTo 0

End Sub

Sub manuelkaldır()

kaldir "Excelent"

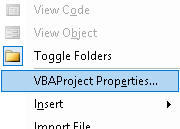
End Sub

### Diğer Konular

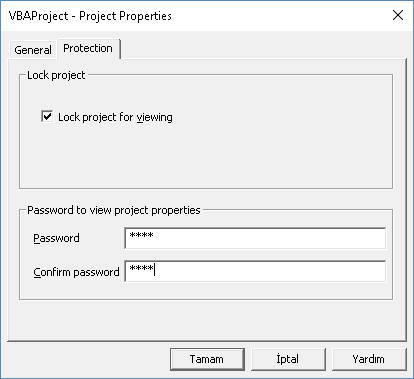
#### Password Koruma

Tahminimce yazdığınız kodlar içinde en çok emek verdikleriniz add-in'leriniz olacaktır. Bu add-inleri başkalarına da kullanmaları için gönderiyorsanız, kodlarınızın kolaylıkla kopyalanıp değişikliğe uğramasını istemeyebilirsiniz. Bu yüzden kodlarınızı şifreyle korumaya almak isteyebilirsiniz.

Bunun için yapmanız gereken VBE'de ilgili workbookta dosya adına veya herhangi bir modüle sağ tıklayıp VBAProject Properties'i açın.



Sonra ikinci sekmedeki Lock seçimini yapıp şifrenizi girin.



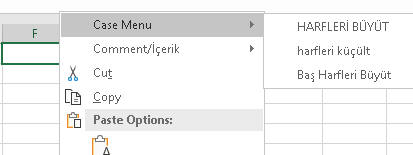
Son olarak dosyayı kaydedip, kapatın ve tekrar açın, artık şifre korumalı olarak gelecektir. Bu yöntem sadece add-inler için değil tüm makro dosyaları için geçerlidir.

Bununla birlikte, bu koruma yöntemi oldukça düşük bir koruma sağlar. Zira bu kodları kırmaya yarayan yollar internette kolayca bulunabilir. Bu yöntemin amacı korumadan daha çok tecrübesiz kişilerce kodun yanlışlıkla bozulmasını engellemek olmalıdır, veya inlik cinlik bilmeyen kişilere karşı da bir kod hırsızlığına karşı koruma sağladığı söylenebilir.

Daha sağlam bir koruma yöntemi istiyorsanız, bunun için XLL veya COM add-ini yazmanız gerekecektir.

#### Context menü/Sağ tık menüsü

Bazen bir hücre/satır/kolon üzerinde işlem yapmak isteriz. Bunun için herhangi bir hücre/satır/kolona sağ tıklayıp satır/sütun silme, hücreyi biçimlendirme gibi işlemler yaparız. İşte bu tür işlemlerin yeterli olmadığı durumlar için Add-ins menüsü altına menüler eklemek yerine hücrelere sağ tıkladığında açılmasını istediğimiz menüler eklemek daha kullanışlı olabilir. Aşağıda, az sonra oluşturacağımız iki menünün görüntüsü bulabilirsiniz.



Bunlarla seçili hücrelerin Case durumunu değiştirebilecek veya hücreden commente, commentten hücreye bilgi geçişi sağlayabileceğiz.

Hemen kodlarımıza bakalım.(Sadece bu menüleri oluşturan kodlara bakacağız, çalıştırdığı makro kodlarını örnek dosya içinde bulabilirsiniz)

Sub contextMenu()

Dim contextMenu As CommandBar

Dim altMenu As CommandBarPopup

'Cell commandbarını elde edelim

Set contextMenu = Application.CommandBars("Cell")

Set altMenu = contextMenu.Controls.Add(msoControlPopup, , , 1, True)

With altMenu

.Caption = "Case Menu"

.Tag = "casetag"

With .Controls.Add(Type: =msoControlButton)

.OnAction = "Buyukyap"

.Caption = "HARFLERİ BÜYÜT"

End With

With .Controls.Add(Type: =msoControlButton)

.OnAction = "Kucukyap"

.Caption = "harfleri küçült"

End With

With .Controls.Add(Type: =msoControlButton)

.OnAction = "ProperYap"

.Caption = "Baş Harfleri Büyüt"

End With

End With

Set altMenu = contextMenu.Controls.Add(msoControlPopup, , , 2, True)

With altMenu

.Caption = "Comment/İçerik"

.Tag = "commenticerik"

With .Controls.Add(Type: =msoControlButton)

.OnAction = "IcerikiCommentYap"

.Caption = "İçeriği comment yap"

End With

With .Controls.Add(Type: =msoControlButton)

.OnAction = "CommentiIcerikYap"

.Caption = "Commenti içerik yap"

End With

End With

End Sub

Bu prosedürü Workbook\_Open içinden çağırmanız gerektiği aşikardır. Keza add-in'in kaldırılması durumunda yok olmaları için Workbook\_AddinUninstall prosedürüne de ilgili kaldırma kodlarını yazarız.

Bu prosedürde önemli olan, menüyü ekleyeceğimiz commandbarın seçimidir. Bu, **Application.CommandBars("Cell")** isimli commandbardır. Ondan sonrası yukarıda yaptığımız işlemlerin nerdeyse aynısıdır. Bunları yine Temporary parametresine True atayarak geçici olarak ekliyoruz. Yukarıdan farklı olarak, context menünün en başına eklenmeleri için Before parametresine ilki için 1, ikincisi için 2 değerlerini atıyoruz.

#### Hangi butona basıldığını tespit etme

Bazen, oluşturduğunuz menüde hangi butona basıldığını bilmek ve bu göre bir işlem yaptırmak isteyebilirsiniz. Bunu CommandBarın **ActionControl** özelliği ile yapıyoruz. OnAction ile tabiki belirli bir prosedürü çalıştırıyoruz ancak biz belki sadece tıklanan butonun metadatasını kullanmak istiyoruzdur. Hatta düğmelerin hepsi için OnAction özelliğine aynı prosedür atanmış da olabilir. Siz koşullu bir yapı ile tıklanan butona göre işlem yaptırabilirsiniz. Bütün bunlar için Tag propertysini de kullanıyor olacağız.

Aşağıda Microsoft'un dokümantasyonundaki örneği görebilirsiniz.

Sub addinyap()

'çeşitli kodlar

Set myBar = CommandBars \_

.Add(Name: ="Custom", Position: =msoBarTop, \_

Temporary: =True)

Set buttonOne = myBar.Controls.Add(Type: =msoControlButton)

With buttonOne

.FaceId = 133

.Tag = "RightArrow"

.OnAction = "whichButton"

End With

Set buttonTwo = myBar.Controls.Add(Type: =msoControlButton)

With buttonTwo

.FaceId = 134

.Tag = "UpArrow"

.OnAction = "whichButton"

End With

Set buttonThree = myBar.Controls.Add(Type: =msoControlButton)

With buttonThree

.FaceId = 135

.Tag = "DownArrow"

.OnAction = "whichButton"

End With

myBar.Visible = True

End Sub

'bu kısımda da ActionControl ile hangi butona tıklandığını görebiliyoruz

Sub whichButton()

Select Case CommandBars.ActionControl.Tag

Case "RightArrow"

MsgBox ("Right Arrow button clicked.")

Case "UpArrow"

MsgBox ("Up Arrow button clicked.")

Case "DownArrow"

MsgBox ("Down Arrow button clicked.")

End Select

End Sub

#### Değişiklikleri kaydetmeme problemi

Add-inlerle ilgili bilinen bir bug'dan sayfa başında bahsetmiştim. Bir diğer bug ise bazen yaptığınız değişikliklerin Save butonuna bastığınız halde kaydolmamasıdır. Bununla ilgili olarak ne kadar araştırma yaptıysam da şuana kadar sebebini henüz öğrenemedim ancak çözümünü öğrendim.

Aşağıdaki tek satırlık bir prosedür ile yaptığınız değişiklikleri böyle kaydedin, VBE içindeki Save düğmesi ile değil.

Sub kaydet()

'olur da bir şekilde add-ine yaptığınız değişiklikler kaydolmazsa

ThisWorkbook.Save

End Sub

## Neler Öğrendik

Nesnelerle çalışırken arka planda neler olduğunu, belleğin nasıl işlediğini, etkin bir bellek yönetimi için neler yapılması gerektiğini gördük.

Shapeler, OleObject’ler, Pivot tablolar, Slicerlar ve grafikler ile çalışmayı, burdaki sık kullanılan operasyonları öğrendik.

Add-in’leri kullanarak menüler yapmayı, makrolarımızı bu menülere yerleştirmeyi öğrendik. Add-inler sayesinde kodlarımızın başkaları tarafından da kullanılabilmesini sağladık.