# 11.bölüm

**11**

# Diğer Uygulamalarile İletişim

Bu bölümde Outlook otomasyonu ile mail gönderme işlerini ve Outlook nesneleri üzerinde manipülasyon yapmayı öğreneceğiz. Ayrıca veritabanlarından data okuma, data yazma, Connection kurma ve ListObject/QueryTable nesneleri hakkında bilgi sahibi olacağız.

## Outlook Programlama

### Giriş

Birçoğunuzun bildiği veya tahmin ettiği üzere Microsoft Office ürünlerinin hepsinin birbirleriyle konuşabilmesi diye birşey söz konusu. Bunu günlük hayatta Excel'deki bir grafiği veya başka bir nesneyi Word'e veya PowerPoint’e kopyalarken görebiliyoruz.

Bu ilişki kod tarafında da kurulabiliyor. Bunun için de hangi uygulamadaysanız, iletişim kurmak istediğiniz diğer uygulamadan bir **Application nesnesi** yaratma yoluyla ilerlersiniz. Bunun için iki yöntem var. Ya ilgili uygulamanın kütüphanesini(library) VBE'de **Tools>Reference** menüsünden eklemek ve bundan bir nesne yaratmak, ki buna Early Binding denir, veya Object türünde bir değişken yaratıp Late Binding yöntemiyle ilgili atamayı yapmak.

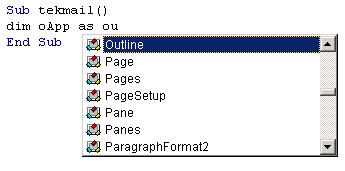
Bu şekilde ilgili uygulamanın nesne modeline erişim sağlamış oluruz.

Bu arada, bu konuyla ilgili örnek dosyayı **githubdan (vbmail.xlsm)** indirebilirsiniz.

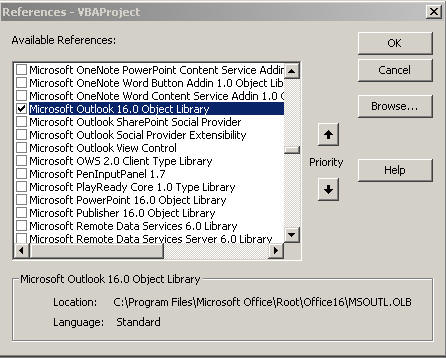
### Kütüphane ekleme

Biz burada ağırlıklı olarak **Early Binding** metodunu göreceğiz, bir iki yerde ise **Late Binding** örneği yapacağız. Tabiki iki tür yöntemin de kendine göre avantajları var, bunların detayını **İleri Seviye Programlama>Nesneler Dünyası konusunda** bulabilirsiniz ama ben intellisense özelliğinden faydalanmak istediğim için genelde Early Bindingi tercih ediyorum.

Kütüphaneyi henüz eklememişken aşağıdaki gibi bir kod yazıp intellisensin bana ne gösterdiğine(daha doğrusu ne göstermediğine) bakalım.

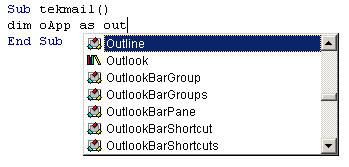


Şimdi referansımızı ekleyelim.



Siz tabi kendinizde hangi Office versiyonu varsa onu eklersiniz. Ben şuan Office 2016 ile çalışıyorum, eğer sizde 2016'dan farklı bir Outlook versiyonu varsa ve bu sayfadaki örnek dosyaya bakıyorsanız sizde MISSING olarak görünür, bunu kaldırıp doğru versiyonu eklediğinizden emin olun. Eğer sıfırdan oluşturuyorsanız zaten başka bir library dosyası göreceksiniz.

Ve intellisense'i tekrar kontrol edelim. Gördüğünüz gibi şimdi Outlook geliyor.



Bu arada Outlook nesne modeliyle ilgili yaygın bir notasyon vardır. Outlook'un kendisini atayacağımız değişken adına genelde **oApp/outApp**, mail değişkenine ise **oMail/outMail** adı verilir. Ben de bu geleneği sürdüreceğim.

### Mail Gönderme

Bu kısımda önce basit bir mail gönderimi yapacağız, sonra da bir liste üzerinden çoklu mail gönderimi yapacağız, aşağıda ayrı bir kısım olarak oluşturduğum **Otomatik Mail Gönderim** formunda ise kompleks bir mail gönderim uygulaması yazacağız.

#### Nesneleri yaratmak

Mail göndermede ilk adım bir Outlook nesnesini bir de mail nesnesini temsil eden değişkenleri yaratmaktır. Bunu aşağıdaki kodla yapıyoruz.

Dim oApp As Outlook.Application 'Outlook'un kendisi

Dim oMail As Outlook.MailItem 'mail nesnesi

Daha sonra bu değişkenlere değerlerini atıyoruz. Akabinde mailin temel özelliklerini (konusu, alıcısı, gövde metni) tanımlıyoruz, ancak maili göndermiyoruz, sadece gösteriyoruz(**Display** metodu). Maili göndermeden sadece gösterme eylemini uygulamak, test yapma konusunda bize büyük kolaylıklar sağlar. Bu metodu sık sık kullanmanızı tavsiye ederim.

Sub mailGöster()

Dim oApp As Outlook.Application

Dim oMail As Outlook.MailItem

Set oApp = New Outlook.Application

Set oMail = oApp.CreateItem(olMailItem)

With oMail

.Subject = "bu bir deneme mailidir"

.To = "volkan.yurtseven@hotmail.com"

.Body = "deneme"

.Display

End With

End Sub

Outlook nesnesini taşıyacak değişkenle değerini atama işini iki ayrı satırda yaptık ancak daha önceki bölümlerden gördüğünüz üzere bunu tek satırda da yapabilirdik: **Dim oApp As New Outlook.Application**

Mail nesnesini yaratma işini ise Application nesnesinin **CreateItem** metodu ile yaptık, parametre olarak da intellisense’den açılan **olMailItem** değerini verdik. Bunların numerik değerleri de var, ancak Early Binding'in nimetlerinden faydalanalım diyorum. Yine de sıklıkla farklı kaynaklarda görebilirsiniz ve belki de kullanmak istersiniz diye sadece mailitem'ın nümerik değerini bilmeniz de faydalı olacaktır; o da **0**'dır.



Late Bindigle gönderim şekli ise şöyle olurdu;

Sub lateBindingMail()

Dim oApp As Object

Dim oMail As Object

Set oApp = CreateObject("Outlook.Application")

Set oMail = oApp.CreateItem(olMailItem)

With oMail

.Subject = "bu bir deneme mailidir"

.To = "volkan.yurtseven@hotmail.com"

.Body = "deneme"

.Display

End With

End Sub

Outlook nesnesini Late Binding ile elde etmenin bir yolu da **GetObject** metodunu kullanmaktır.

O anda çalışan bir Outlook nesnesi varsa, yenisini yaratmamak adına bu bellekteki Outlook nesnesine başvurulur. Eğer yoksa, hata alınır, hata olmaması adına öncesine On Error Resume Next denir, hata kodu kontrol edilir, hata varsa CreateObject’le nesne yaratılır.

On Error Resume Next

Set oApp = GetObject(, "Outlook.Application")

If Err.Number <> 0 Then '0 demek hata yok demek

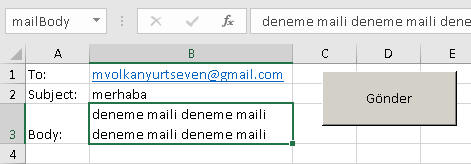
Set oApp = CreateObject("Outlook.Application")

End If

#### Tek mail göndermek

Yukarıda gördüğümüz üzere, mail nesnesini yarattıktan sonra olmazsa olmazımız mailin alıcısını tespit etmektir. Bunu genelde **To** özelliği ile yaparız. Başka bir alternatif daha var, buna sonra değineceğiz.

Aşağıdaki örnekte, bir sayfada çeşitli NamedRange'ler oluşturdum, ve mail gönderme işini de bir butona atadım. Maili gönderme işini **Send** metodu ile yapıyoruz.



Sub Button1\_Click()

Dim oApp As Outlook.Application

Dim oMail As Outlook.MailItem

Set oApp = New Outlook.Application

Set oMail = oApp.CreateItem(olMailItem)

With oMail

.Subject = Range("mailSubj")

.To = Range("mailTo")

.Body = Range("mailBody")

.Send

End With

End Sub

#### Çoklu(Batch) mail göndermek

Öncelikle şunu belirtmek isterim ki, çok sayıda alıcıya mail göndermekle kastımız, alıcıların birbirlerini görmeden mail göndermektir ve bunun birden çok yolu vardır. Bunlara hızlıca bir bakalım.

**Alıcıları bcc'ye koyarak klasik yolla mail göndermek**: Bu yolla parametrik bilgi geçilemez. Mesela kişiye özel hitap edilemez.

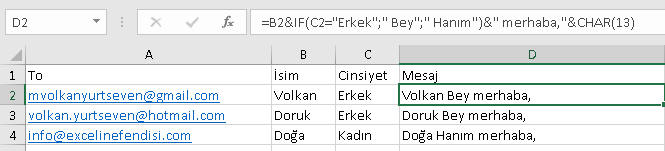
**Word Mailmerge**: Kişiye özel mesajlar vermek için Word'ün MaiMerge özelliğin kullanabilirsiniz. Kişiye ismiyle hatta bay/bayan/doktor gibi çeşitli hitap şekilleriyle hitap edebileceğiniz gibi, onlara çeşitli rakamsal veya metinsel bilgileri kendilerine özel olarak iletebilirsiniz. Bununla ilgili olarak internette bol miktarda makale ve video bulunmakta. Hiç makro bilgisi gerektirmediği için de herkesin kullanabileceği bir özellik olmaktadır.

**VBA**: Mail Merge'le yapılamayan kişiye özel ek, kişiye özel cc/bcc belirleme, koşullu konu başlığı veya gövde metni belirleme gibi daha kompleks işleri VBA ile yapabilirsiniz.

**Diğer Programatik yaklaşımlar**: Tüm diğer programlama dilleriyle bunu yapabilirsiniz, veya bu konuda hazır yazılmış programları veya benim Excelent isimli Add-in'im gibi paketleri kullanabilirsiniz.

Biz doğal olarak burada sadece VBA yöntemine değiniyor olacağız.

Bunun için aşağıdaki gibi bir listemiz olsun.



Mesaj kısmını, gördüğünüz gibi parametrik oluşturdum. Formülün sonundaki CHAR(13) ifadesi Enter işareti anlamına gelir, yani takip eden metin bir alt satırdan devam eder. (Bazı kaynaklarda CHAR(10) da görebilirsiniz, Excel'de ikisi de aynı sonucu üretir)

Sırayla tüm alıcılara mail atılması gerektiği için döngüsel bir kod yazmamız gerekiyor. Şimdi bu kodumuzu oluşturup bir düğmeye atayalım.

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

Next a

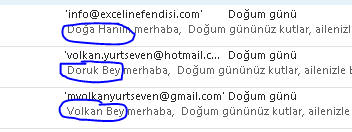
Set oApp = Nothing

End Sub

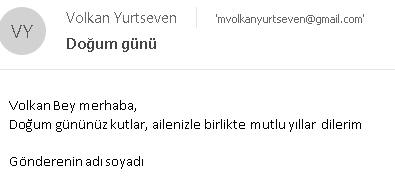
Şimdi fark ettiyseniz, bu sefer mail nesnesini **For Each** döngüsü içine aldık. Zira her gönderim sonunda ilgili nesneyi **Nothing** ataması yaparak yok etmemiz gerekiyor. Yok ettiğimiz nesneyi ise tekrar kullanamayacağımız için tekrar yaratıyoruz.

Aslında yok edilmesi gereken bir nesne daha var, o da **oApp** nesnesi, onu da programın en sonunda yok ediyoruz.

Evet sonuçta Sent Items'a baktığımızda aşağıdaki gibi kişiye özel maillerin gittiğini topluca görebilirsiniz.



Giden maillerden birinin içeriği de aşağıdaki gibidir.



Merhaba satırından sonra bir boşluk daha olsun isterseniz formülün içine iki tane CHAR(13) yazmanız gerekir.

#### Maile canlılık katmak

##### HTMLBody

Gönderdiğiniz mailde çeşitli yerleri renkli, koyu, farklı yazı tipinde v.s göstermek isterseniz mail nesnesinin **Body** özelliğini değil **HTMLBody** özelliğini kullanmanız gerekir.

Bunun için biraz HTML bilmenizde fayda var. Aşağıdaki örnekte renklendirme ve kalınlaştırma örneğini görebilirsiniz.

Sub Button2\_Click()

Dim oApp As Outlook.Application

Dim oMail As Outlook.MailItem

Set oApp = New Outlook.Application

Set oMail = oApp.CreateItem(olMailItem)

imza = "<H3><B><font size=""10"" face=""Arial"" color=""red"">Adım Soyadım</font></B></H3>"

With oMail

.Subject = Range("mailSubj")

.To = Range("mailTo")

.HTMLBody = Range("mailBody")

.HTMLBody = .HTMLBody & imza

.Send

End With

End Sub

Burada, H3, B font gibi ifadelere Tag denir. Size, face, color gibi ifadelere ise özellik denir. Bunlardan <H3>, özel bir başlık türüdür. Ben de mesela bu sayfayı hazırlarken az yukardaki "Maile canlılık katmak" başlığında H3 tag'ini kullanıyorum. <B>, ilgili metni kalın yapar. Bu etiketler </H3>ve </B> etiketleriyle kapatılmak zorundadırlar.

Diğer kullanmak istediğiniz Tag'ler için bir html eğitim sayfasına bakmanızı tavsiye ederim.

Aşağıda giden mailimiz görünmektedir.



##### Resim kullanmak

Mail gövdesine resim dahil etmek için yine **HTMLBody** özelliğini kullanıyoruz. Bunun için <**img**> tag'i kullanılır.

Yukarıdaki kodda **Send** komutundan hemen önceki satıra şunu ekleyelim

.HTMLBody = .HTMLBody & "<img src ='C: \....\images\Excelent.jpg'>"

Giden mail de şöyle görünür:



Bu arada fark ettiyseniz bu sefer img tag'inin özelliklerini kesme işareti arasın aldım. Bir üstteki örnekte ise iki tane çift tırnak içine almıştım. İki yöntem de özdeştir, tek kesme işareti pratik olmakla birlikte iki tane çift tırnak görürseniz de şaşırmayın.

#### Recipients collection’ı ve Recipient nesnesi

Bazen mail göndermeye çalıştığımız mail adresi artık geçerli olmayabilir. Geçerli olmaktan kastım, ilgili adresin AdressBook içinde kayıtlı olmasıdır. Mesela bu bir personel listesiyse, personellerden bazısı kurumdan ayrılmış olabilir. Böyle durumlarda, mail adresinin geçerli olup olmadığını kontrol etmemiz gerekebilir. Bunu da **Recipient** nesnesinin **Resolve** metodu ile yapıyoruz. Bu işlem, aslında Ribbon'dan **Check Names** düğmesine tıklamakla özdeştir.

Tabi ilgili mail adresinin adressbook'ta olması gerekiyor, ki personeller söz konusu olduğunu da bu zaten kesindir, ama başka alıcılara mail göndermeye çalışırken adresbook’ta değillerse resolve yapamazsınız.

Aşağıdaki örnekte, önce Recipients nesnesini yaratıyoruz. Bunu mailitem nesnesine **Add** diyerek yapıyoruz. Parantez içine de bir mail adresi, veya bunu temsil eden bir değişken atıyoruz. Hemen arkasından da **Type** özelliği ile bu alıcının tipini belirtiyoruz.

Sonrasında Resolve diyoruz, Resolve olup olmadığını(Kontrol sonucunda adresin geçerli olup olmadığını) ise bir alt satırdaki **Resolved** property'si ile anlıyoruz, evetse maili gönderiyoruz.

Sub resolveörnek()

Dim oApp As New Outlook.Application

Dim olMailtem As Outlook.MailItem

Dim alıcı As Outlook.Recipient

Set olMailtem = oApp.CreateItem(olMailItem)

Set alıcı = olMailtem.Recipients.Add(geçerlimail\_veya\_outlookkullanıcısı)

alıcı.Type = olTo

alıcı.Resolve 'Check Names tuşuna basmışız gibi

If alıcı.Resolved Then 'Adress Book'ta varsa

olMailtem.Subject = "deneme"

olMailtem.Send

End If

Set olMailtem = Nothing

Set oApp = Nothing

End Sub

Birçok kaynakta Mailitem'ın **To/Bc/Bcc** özelliklerini kullanmak yerine **Recipient**'ın kullanımı önerilir. Genel olarak doğru bir öneri olmakla birlikte, listenizdekilerin kesin geçerli mailler/siciller olduğunu biliyorsanız Recipient çok da elzem değildir. Zira Recepient'ın kullanımı bazen karmaşık olabilmektedir. Mesela diğer üçünde tek seferde birden çok alıcı eklenebilirken, Recepient'ta her alıcı döngüsel olarak tek tek eklenmelidir.

Bu arada bu kontrol işlemini çoklu alıcılı versiyonu **Recipients**'i collection'ı ile ve bunun **ResolveAll** metodu ile yapılmaktadır. ResolveAll da tıpkı Resolve gibi sonuç olarak True/False döndürür. Eğer alıcılardan birisi bile Resolve olmazsa False döner. Hangi alıcılar Resolve olmamış diye bakmak istersek For Next içinde tek tek Recipient'ları dolaşmamız gerekir. Aşağıda bununla ilgili bir örnek bulunmaktadır.

Sub CheckRecipients()

Dim oapp As New Outlook.Application

Dim omail As Outlook.MailItem

Dim myRecipients As Outlook.Recipients

Dim myRecipient As Outlook.Recipient

Set omail = oapp.CreateItem(olMailItem)

Set myRecipients = omail.Recipients

myRecipients.Add ("volkan.yurtseven@hotmail.com")

myRecipients.Add ("Dan Wilson asdas") 'bu mail, Adress book'umda yok

myRecipients.Add ("mvolkanyurtseven@gmail.com")

'tüm mailler için Check Names tuşuna basılıp, hepsi de var mı diye bakıyoruz

If Not myRecipients.ResolveAll Then 'hepsi birden aynı anda mevcut değilse

For Each myRecipient In myRecipients

If Not myRecipient.Resolved Then

myRecipients.Remove (myRecipient.Index) 'Address Bookta bulunmayanları collectiondan çıkarıyoruz

End If

Next

End If

omail.Display 'sadece diğer ikisi görünür

End Sub

### Attachment(Ek) ve Range'i Maille göndermek

#### Attachment(ek) gönderimi

Mail gönderimiyle ilgili en geniş kaynağa sahip sitelerden biri, belki de en popüleri, kendisini birçok kişinin referans olarak verdiği Ron de Bruin ustanın sitesidir (<https://www.rondebruin.nl/win/s1/outlook/mail.htm>). Burda bir dolu mail gönderim şekli var. O anki aktif sayfayı mail olarak göndermek, o an açık olan dosyayı göndermek, hatta bunun da iki versiyonu var, var olan bir dosya ve henüz yeni yaratılmış ancak hiç kaydedilmemiş bir dosyayı göndermek gibi.

Açıkçası şimdiye kadarki otomasyon süreçlerimde bu yöntemlerin birçoğuna başvurmadım. O yüzden kullanılma ihtimali çok düşük olan yöntemleri buraya koyarak vaktinizi almak istemem. İhtiyaç duyanlar, İngilizcesi varsa ustanın sitesinden bakabilir, İngilizceniz olmasa bile bu kitapta öğrendiğiniz teknikler ve genel kodlama bakış açınızla bu işleri çok rahat yapabiliriniz.

Mail otomasyonu sırasında en çok ihtiyaç duyduğum şey, parametrik bilgileri alıp toplu mail gönderimi yapmak oldu. Bu parametrik bilgilerden biri de **kişiye özgü ek** kullanmaktı. Ana kod bloklarını yukarıda gördünüz, ek eklemek için de yapmanız gereken aşağıdaki kodu **Send** metodundan önce bir yere dahil etmek olacaktır.

oMailItem.Attachments.Add "Dosya adı"

Birden fazla ek koyacaksanız, bunları alt alta yazabilirsiniz, veya bir klasördeki tüm dosyaları göndermek istiyorsanız önce I/O teknikleriyle klasördeki dosyaları elde edip bir collectiona atar, sonra da for döngüsü içinde bunları maile eklersiniz.

'Çoklu ek

oMailItem.Attachments.Add "Dosya adı1"

oMailItem.Attachments.Add "Dosya adı2"

'döngüsek ek, öncesinde I/O işlemi yapılıp collectiona alındığı varsayıldı

For Each f in dosyalar

oMailItem.Attachments.Add f

Next f

Zaten biraz aşağıda göreceğiniz gibi mail otomasyonu ile userform bilginizi bir araya getirdiğinizde gayet şık bir **Toplu Mail Gönderim** uygulamasını yazabilirsiniz.

#### Excel Range'i body içinde göndermek

Bazen belli bir hücre kümesini(Range) ek olarak değil de mail bodysi içinde göndermek isteriz. Bunun için yine Ron de Bruin ustanın yazdığı ve hemen hemen tüm blog ve forumlarda referans verilen meşhur RangeToHtml fonksiyonunu kullanabiliriz. Fonksiyona hiç dokunmadan olduğu gibi kullanabilirsiniz. Fonksiyonu çağırırken ona ilgili Range'i parametre olarak vermeniz yeterli. Tabi dönen değeri de mailitem nesnesinin **HTMLBody** özelliğine atarız. Önce fonksiyona bakalım.

Function RangetoHTML(rng As Range)

' Changed by Ron de Bruin 28-Oct-2006

' Working in Office 2000-2016

Dim fso As Object

Dim ts As Object

Dim TempFile As String

Dim TempWB As Workbook

TempFile = Environ$("temp") & "\" & Format(Now, "dd-mm-yy h-mm-ss") & ".htm"

'Copy the range and create a new workbook to past the data in

rng.Copy

Set TempWB = Workbooks.Add(1)

With TempWB.Sheets(1)

.Cells(1).PasteSpecial Paste: =8

.Cells(1).PasteSpecial xlPasteValues, , False, False

.Cells(1).PasteSpecial xlPasteFormats, , False, False

.Cells(1).Select

Application.CutCopyMode = False

On Error Resume Next

.DrawingObjects.Visible = True

.DrawingObjects.Delete

On Error GoTo 0

End With

'Publish the sheet to a htm file

With TempWB.PublishObjects.Add( \_

SourceType: =xlSourceRange, \_

Filename: =TempFile, \_

Sheet: =TempWB.Sheets(1).Name, \_

Source: =TempWB.Sheets(1).UsedRange.Address, \_

HtmlType: =xlHtmlStatic)

.Publish (True)

End With

'Read all data from the htm file into RangetoHTML

Set fso = CreateObject("Scripting.FileSystemObject")

Set ts = fso.GetFile(TempFile).OpenAsTextStream(1, -2)

RangetoHTML = ts.readall

ts.Close

RangetoHTML = Replace(RangetoHTML, "align=center x: publishsource=", \_

"align=left x: publishsource=")

'Close TempWB

TempWB.Close savechanges: =False

'Delete the htm file we used in this function

Kill TempFile

Set ts = Nothing

Set fso = Nothing

Set TempWB = Nothing

End Function

Bu fonksiyonu kullanacağımız makro ise aşağıdaki gibi olabilir. Peki böyle bir makroya ne zaman ihtiyacınız olur? Yine çoklu mail sırasında olabilir. Mesela bir veritabanından şube bazlı bilgi çektiniz, her şubenin birkaç(ek yapmaya değmeyecek kadar az) satır civarında kaydı geliyor olsun. Bir filtreleme mekanizması kurup her şube filtrelenmesi sonucunda filtreye gelen sonuçları ilgili şubeye gönderim yapacak şekilde oturtabilirsiniz. Bunu döngüsel olarak yaptığınızda istediğiniz sonuca ulaşabilirsiniz.

Sub Mail\_Selection\_Range\_Outlook\_Body()

Dim rng As Range

Dim OutApp As Object

Dim OutMail As Object

Range("A1").CurrentRegion.Select

Set rng = Selection.SpecialCells(xlCellTypeVisible)

Set OutApp = CreateObject("Outlook.Application")

Set OutMail = OutApp.CreateItem(0)

With OutMail

.To = "ron@debruin.nl"

.CC = ""

.BCC = ""

.Subject = "This is the Subject line"

.HTMLBody = RangetoHTML(rng) 'fonksiyonu burada çağırıyoruz

.Send 'or use .Display

End With

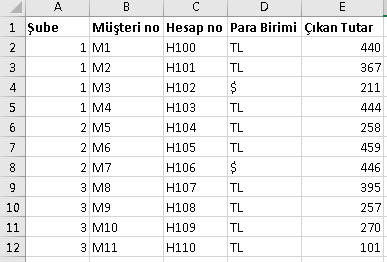
Set OutMail = Nothing

Set OutApp = Nothing

End Sub

Şimdi bir de kendi dünyamızdan gerçek bir örnek yapalım. Tabi bu, hakiki örneğin yine de basitleşmiş hali olacak. Zira hakiki örnekte, rapor schedule edilmiş, kendiliğinde çalışmış, mail gönderimi de kendiliğinden olmaktadır. Biz ise bu örnekte önceki aşamalar zaten olmuş farz edeceğiz ve gönderimi manuel yapacağız. (Full otomasyon detayı için Application.OnTime ve Veritabanı programlama konularına bakınız.)

Diyelim ki çekilen rapor aşağıdaki gibi. Bu 3 şubeye kendi müşterilerine ait listeyi göndereceğiz.



Öncelikle şubeler isimli bir collection’a dinamik bir şekilde listedeki şube kodlarını atarım. Bunun için Collectionlar konusundaki **Coldavarmı** fonksiyonumdan yararlandım.

Sonra da bu collection içinde tek tek hareket edip filtreleme yapıyor, filtre sonucunda tüm alanı seçip visible olanları mail body'sine yapıştırıyorum.

Sub mevduatçıkışları()

Dim rng As Range

Dim OutApp As Object

Dim OutMail As Object

Dim şubeler As New Collection

Set OutApp = CreateObject("Outlook.Application")

For Each r In Range([a2], [a2].End(xlDown))

If ColdaVarmı(şubeler, r.Value) = False Then

şubeler.Add r.Value

End If

Next r

For Each şb In şubeler

'Selection.AutoFilter

ActiveSheet.Range("A1").CurrentRegion.AutoFilter Field: =1, Criteria1: =şb

Range("A1").CurrentRegion.Select

Set rng = Selection.SpecialCells(xlCellTypeVisible)

Set OutMail = OutApp.CreateItem(0)

With OutMail

.To = "volkan.yurtseven@hotmail.com" 'normalde burası parametrik olacak

'.To = "sube." & r.Value & "@kurumadı.local" parametrikken böyle yapılır

.Subject = "Mevduat Çıkışı yaşayan müşterileriniz"

.HTMLBody = RangetoHTML(rng)

'.Send 'ya da .Display

.Display

End With

Set OutMail = Nothing

Next şb

Set OutApp = Nothing

End Sub

---------

Function ColdaVarmı(col As Collection, kontrol As Variant) As Boolean

On Error Resume Next

ColdaVarmı = False

Dim x As Variant

For Each x In col

If x = kontrol Then

ColdaVarmı = True

Exit Function

End If

Next

End Function

Örnek bir mail görüntüsü aşağıdaki gibi olacaktır. Tabi kod içinde comment’li olarak belirttiğim gibi aslında bu mailin 2 nolu şubeye gitmesi gerekiyor. Ayrıca body'ye konacak alanın formatlaması durumunda aynı şekilde gelecektir ve yine ayrıca isterseniz yapıştırılan bu Range'in öncesine veya sonrasına ilave mesaj ekleyebilirsiniz.



#### Excel Range'i ek olarak göndermek

Eğer göndermek istediğiniz kısım body'ye konmayacak kadar büyükse bu Range'i ek olarak göndermek isteyebilirsiniz. Bunun için kodumuz şöyle olabilir.(Bu kod da Ron de Bruin'in sayfasından alınarak sadeleştirilmiştir)

Burda ise geçici bir dosya oluşturup, ilgili range'i bu yeni dosyaya yapıştırıp kaydediyoruz, sonra bunu attachment yapıyoruz, mail gönderildikten sonra da geçici dosyayı siliyoruz.

Sub selection\_ek\_olarak\_mail()

Dim Source As Range

Dim Dest As Workbook

Dim wb As Workbook

Dim TempFilePath As String

Dim TempFileName As String

Dim FileExtStr As String

Dim FileFormatNum As Long

Dim OutApp As Object

Dim OutMail As Object

Dim şubeler As New Collection

Set OutApp = CreateObject("Outlook.Application")

For Each r In Range([a2], [a2].End(xlDown))

If ColdaVarmı(şubeler, r.Value) = False Then

şubeler.Add r.Value

End If

Next r

For Each şb In şubeler

ActiveSheet.Range("A1").CurrentRegion.AutoFilter Field: =1, Criteria1: =şb

Range("A1").CurrentRegion.Select

Set Source = Selection.SpecialCells(xlCellTypeVisible)

Set wb = ActiveWorkbook

Set Dest = Workbooks.Add(xlWBATWorksheet)

Source.Copy

With Dest.Sheets(1)

.Cells(1).PasteSpecial Paste: =8

.Cells(1).PasteSpecial Paste: =xlPasteValues

.Cells(1).PasteSpecial Paste: =xlPasteFormats

.Cells(1).Select

Application.CutCopyMode = False

End With

TempFilePath = Environ$("temp") & "\"

TempFileName = şb & "-dosyadı"

If Val(Application.Version) < 12 Then

'You use Excel 97-2003

FileExtStr = ".xls": FileFormatNum = -4143

Else

'You use Excel 2007-2016

FileExtStr = ".xlsx": FileFormatNum = 51

End If

Set OutApp = CreateObject("Outlook.Application")

Set OutMail = OutApp.CreateItem(0)

With Dest

.SaveAs TempFilePath & TempFileName & FileExtStr, FileFormat: =FileFormatNum

On Error Resume Next

With OutMail

.To = "volkan.yurtseven@hotmail.com"

.Subject = "Mevduat çıkışı yaşayan müşterileriniz"

.Body = "Değerli şube çalışanlarımız ......."

.Attachments.Add Dest.FullName

'.Send

.Display

End With

.Close savechanges: =False

End With

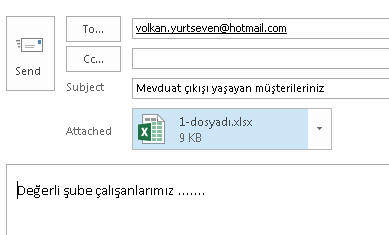
Kill TempFilePath & TempFileName & FileExtStr

Set OutMail = Nothing

Next şb

Set OutApp = Nothing

End Sub



### Folder erişimi ve Mail kontrolü

Outloo’ktaki folder ve maillerle çalışmak için **Namespace** nesnesine ihtiyaç duyarız, bunun için de outlook'un **Application** nesnesnin **GetNamespace** metodunu kullanırız.

Bu GetNameSpace metodu, MAPI adında bir parametre alır. Bu parametre, Outlook'un altta yatan mailleşme protokolüdür.

Kullanımı aşağıdaki gibidir.

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Set ns = oApp.GetNamespace("MAPI")

#### Outlook'un yerleşik(built-in) klasörlerine erişim

Namespace nesnesini elde ettikten sonra artık bunun metodlarını kullanabiliriz.

Yerel klasörlere erişmek için **GetDefaultFolder** metodunu kullanırız.

**"GetDefaultFolder("** yazdıktan sonra intellisense çıkar ve ihtiyacımız olan folder tipini seçeriz. Burada 20 çeşit default folder bulunur.(Inbox, SentItems, DeletedItems v.s )

Sub defaultfolderişleri()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderInbox) 'Inbox seçildi

MsgBox myInbox.Items.Count 'Inboxtaki mail adedi

End Sub

#### Kullanıcı tanımlı klasörlere erişim

Birçoğumuz, Outlook'ta mailleri kategorilendirmek için özel klasörler oluşturur ve bunlar için de rule(kural) tanımlar. İşte bu özel folderlara ulaşmak için de **Folders** metodunu kullanırız. Ancak öncesinde hangi yerleşik klasörde olduğunu seçmemiz gerekiyor.

Mesela aşağıdaki kod ile Inbox'ın altındaki "Önemli Mailler" klasörüne erişiyoruz.

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder, önemliMailler As Outlook.Folder

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderInbox)

Set önemliMailler = myInbox.Folders("Önemli Mailler")

Aradığımız klasör kök(root) dizindeyse o zaman önce root foldera erişir, sonra alt klasöre erişiriz.

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myroot As Outlook.Folder, önemliMailler As Outlook.Folder

Set ns = oApp.GetNamespace("MAPI")

Set myroot = ns.Folders("volkan.yurtseven@hotmail.com")

Set önemliMailler = myroot.Folders("Önemli Mailler")

Aşağıdaki kod ile root dizindeki tüm alt dizinlere erişmiş oluruz.

Sub altklasörler()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myroot As Outlook.Folder

Set ns = oApp.GetNamespace("MAPI")

Set myroot = ns.Folders("volkan.yurtseven@hotmail.com")

For Each mysubfolder In myroot.Folders

Debug.Print mysubfolder.Name & ": " & mysubfolder.Items.Count

Next mysubfolder

End Sub

Bir klasöre erişmenin bir yolu da ona EntryID'si ile doğrudan ulaşmaktır. Bu kodu öğrenmek için Outlookta bir folder üzerinde dururken, Outlook'un VBA editörünü açın(Alt+F11 ile). Sonra Ctrl+G ile immediate windowunu açıp oraya şunu yazın ve entera basın.

?activeexplorer.CurrentFolder.EntryID

Size şuna benzer bir kod verecektir:

 000000003E7EE53625476B448D0FE831F453D7B701005069785A8AC0604C87420FC7F4CA7EE90000000001540000

Bu kodu Excel'de **GetFolderFromID** metoduna parametre olarak verebilirsiniz

Set yeniklasör = ns.GetFolderFromID("000000003E7EE53625476B448D0FE831F453D7B701005069785A8AC0604C87420FC7F4CA7EE90000000001540000")

#### Bir klasördeki mailler içinde dolaşma

Bir kez hedef klasörü belirledikten sonra döngüsel bir yapı ile mailler içinde dolaşabiliriz

Sub folderdakimaillerdegezinme()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myroot As Outlook.Folder, myfolder As Outlook.Folder

Dim msg As Outlook.MailItem

Set ns = oApp.GetNamespace("MAPI")

Set myroot = ns.Folders("volkan.yurtseven@hotmail.com")

Set myfolder = myroot.Folders("önemli mailler")

For Each msg In myfolder.Items

Debug.Print msg.Subject

Next msg

End Sub

Arzu eden basit For veya Do While döngüsü ile de gezinti yapabilir.

#### Beklediğiniz mail gelmiş mi kontrolü

Diyelim ki, günlük schedule raporlarınızın çalışmaya başlaması için kurumunuzun Veri ambarındaki(DWH) tabloların yüklenmesi gerekiyor. Ve bu bilgi de size günlük olarak maille geliyor. Bu maillerin subjecti de "DWH yüklemesi tamamlanmıştır" olsun. Ortalama olarak sabah 7-8 civarında yüklemenin tamamlandığını varsayalım.

Böyle bir durumda ana schedule işlemini 06: 00'ye kurduğunuzu düşünelim. Ana prosedürdeki işlerden biri de tabiki "DWH yüklemesi tamam mı" kontrolü olacaktır. Bu tamam değilse, 10 dakika sonraya kendini tekrar schedule etsin ve kalan kodları çalıştırmasın istiyoruz.

Aşağıdaki örnekte ana schedule kodu yok, böyle bir kod örneğine Applicaiton konusundaki OnTime kısmında ulaşabilirsiniz. Biz sadece DWH kontrolü yapan koda bakıyoruz, ki bu kodun içinde 10 dk sonra kendisini tekrar çalıştıran recursive bir kod da eklenmiş durumda.

Bu örnekte ilgili maili bulmak için **Find** metodunu kullanıyoruz. Eğer aranan maili bulamazsa ilgili değişkenin değeri Nothing olacaktır ve bu yüzden hata almaması için **Nothing mi?** kontrolü yapıyoruz, bulamadıysa 10 dk sonraya tekrar schedule ediyoruz. Bulursa ama bulduğu bugünden eski bir mailse, yine 10 dk sonraya schedule ediyoruz, ta ki bugün gelen maili bulana kadar.

Sub dwh\_kontrol()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder

Dim eItems As Outlook.Items

Dim eItem As Outlook.MailItem

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderInbox)

Set eItems = myInbox.Items

Set eItem = eItems.Find("[Subject]=""DWH Yüklemesi tamamlanmıştır""")

If eItem Is Nothing Then

Application.OnTime Now + TimeSerial(0, 10, 0), procedure: ="dwh\_kontrol"

ElseIf DateValue(eItem.ReceivedTime) < Date Then 'var ama eski mailse

Application.OnTime Now + TimeSerial(0, 10, 0), procedure: ="dwh\_kontrol"

Else 'bugün geldiyse

'buraya yapılacak işler konur

End If

End Sub

**Find** yerine doğrudan tüm mailleri tarama yolunu da seçebilirdiniz. Ancak bu durumda tüm mailleri taraması gerekebildiği için kod performansı kötü etkilenebilir. Böyle olmaması adına bir rule tanımlayıp DWH yükleme maillerini belli bir klasöre alabilirsiniz, böylece göreceli daha küçük bir klasörde tarama yapmış olursunuz. Ama çok özeli bir sebebiniz yoksa bence bunlara hiç gerek yok, Find metodu oldukça kullanışlıdır.

Olur da bir nedenle aynı konuyla ilgili başka mail var mı diye bakmak isterseniz, döngü içinde **FindNext** metodunu kullanabilirsiniz. Bu durumda klasördeki tüm maillerde dolaşmanız gerekebilir.

Sub dwh\_kontrol()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder

Dim eItems As Outlook.Items

Dim eItem As Outlook.MailItem

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderInbox)

Set eItems = myInbox.Items

Set eItem = eItems.Find("[Subject]=""DWH yüklemesi tamamlanmıştır""")

'çeşitli işler

For i = 1 To eItems.Count

If TypeName(eItem) = "Nothing" Then

MsgBox "Başka mail yok!"

Exit Sub

End If

'Diğer işlemler

Set eItem = eItems.FindNext

Next i

End Sub

##### Restrict metodu

Find ve FindNext'e bir alternatif **Restrict** metodudur. Find ve FindNext ile ilgili kritere uyan maillerde teker teker ilerlenirken, Restrict ile bunlar filtrelenir ve yeni bir collection elde edilmiş olur.

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder

Dim eItems As Outlook.Items

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderInbox)

Set eItems = myInbox.Items

Kriter ="[Subject]=""DWH yüklemesi tamamlanmıştır"""

Set Kriterliler = eItems.Restrict(Kriter)

If Kriterliler.Count < 1 Then

MsgBox "Hiçbir DWH yükleme maili bulunamadı"

Exit Sub

End If

For Each eItem In Kriterliler

'yapılacak işler

Next

Birden fazla kriter söz konusu ise ilgili metodu ardışık bir şekilde uygulayabilirsiniz.

Sub restricli\_çokkriter()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder

Dim eItems As Outlook.Items

Dim kriterliler As Outlook.Items

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderInbox)

Set eItems = myInbox.Items

Kriter = "[Subject]=""DWH yüklemesi tamamlanmıştır"""

Kriter2 = "[ReceivedTime]>'" & Format("01.06.2018", "dd.mm.yyyy") & "'"

Set kriterliler = eItems.Restrict(Kriter).Restrict(Kriter2)

If kriterliler.Count < 1 Then

MsgBox "Aranan kritere uygun mail bulunamadı"

Exit Sub

End If

MsgBox kriterliler.Count

End Sub

**İpucu**: Arama yapılacak klasörde çok mail varsa Restrict'i, görece daha az mail varsa Find ve FindNext'i kullanın.(Eğer tabi aranan kritere uygun tüm maillerle ilgileniyorsanız. Yoksa yukardaki DWH yükleme örneğindeki gibi sadece ilk maille ilgileniyorsanız Find yeterlidir.)

#### Bir maildeki ekleri bir klasöre kaydetme

Kurum olarak çok sağlam bir veri ambarımız var ama bazen dataya erken ulaşmak önem arz edebiliyor. O yüzden iştirakler gibi dışardan gelen datayı veri ambarına eklenmesini beklemeden işlemek isteyebiliyorsunuz. Bunun için de az yukarda bahsettiğim "beklediğiniz mail gelmiş mi" kontrolünü yaptıktan sonra ekleri istediğim klasöre alıyorum, akabinde raporumu oluşturuyorum.

Bu koda Ron de bruin ustanın sayfasından (<https://www.rondebruin.nl/win/s1/outlook/saveatt.htm>) ulaşabilirsiniz.

Örnek kodlar şöyle:

'Bu kod, MyFolder altındaki tüm mailleri "C: \mailek" klasörüne taşır

SaveEmailAttachmentsToFolder "MyFolder", "", "C: \mailek"

'Bu kod, MyFolder altındaki tüm xlsx uzantılı mailleri "C: \mailek" klasörüne taşır

SaveEmailAttachmentsToFolder "MyFolder", "xlsx", "C: \mailek"

**SaveEmailAttachmentsToFolder**  kodunun kendisi ise şöyle:

Sub SaveEmailAttachmentsToFolder(OutlookFolderInInbox As String, \_

ExtString As String, DestFolder As String)

Dim ns As Namespace

Dim Inbox As MAPIFolder

Dim SubFolder As MAPIFolder

Dim Item As Object

Dim Atmt As Attachment

Dim FileName As String

Dim MyDocPath As String

Dim I As Integer

Dim wsh As Object

Dim fs As Object

On Error GoTo ThisMacro\_err

Set ns = GetNamespace("MAPI")

Set Inbox = ns.GetDefaultFolder(olFolderInbox)

Set SubFolder = Inbox.Folders(OutlookFolderInInbox)

I = 0

' Check subfolder for messages and exit of none found

If SubFolder.Items.Count = 0 Then

MsgBox "There are no messages in this folder : " & OutlookFolderInInbox, \_

vbInformation, "Nothing Found"

Set SubFolder = Nothing

Set Inbox = Nothing

Set ns = Nothing

Exit Sub

End If

'Create DestFolder if DestFolder = ""

If DestFolder = "" Then

Set wsh = CreateObject("WScript.Shell")

Set fs = CreateObject("Scripting.FileSystemObject")

MyDocPath = wsh.SpecialFolders.Item("mydocuments")

DestFolder = MyDocPath & "\" & Format(Now, "dd-mmm-yyyy hh-mm-ss")

If Not fs.FolderExists(DestFolder) Then

fs.CreateFolder DestFolder

End If

End If

If Right(DestFolder, 1) <> "\" Then

DestFolder = DestFolder & "\"

End If

' Check each message for attachments and extensions

For Each Item In SubFolder.Items

For Each Atmt In Item.Attachments

If LCase(Right(Atmt.FileName, Len(ExtString))) = LCase(ExtString) Then

FileName = DestFolder & Item.SenderName & " " & Atmt.FileName

Atmt.SaveAsFile FileName

I = I + 1

End If

Next Atmt

Next Item

' Show this message when Finished

If I > 0 Then

MsgBox "You can find the files here : " \_

& DestFolder, vbInformation, "Finished!"

Else

MsgBox "No attached files in your mail.", vbInformation, "Finished!"

End If

' Clear memory

ThisMacro\_exit:

Set SubFolder = Nothing

Set Inbox = Nothing

Set ns = Nothing

Set fs = Nothing

Set wsh = Nothing

Exit Sub

' Error information

ThisMacro\_err:

MsgBox "An unexpected error has occurred." \_

& vbCrLf & "Please note and report the following information." \_

& vbCrLf & "Macro Name: SaveEmailAttachmentsToFolder" \_

& vbCrLf & "Error Number: " & Err.Number \_

& vbCrLf & "Error Description: " & Err.Description \_

, vbCritical, "Error!"

Resume ThisMacro\_exit

End Sub

#### Toplu mail silme, arşive kaldırma

Otomasyon süreci sonunda eğer ki bilgisayarınızdan çok sayıda mail gidiyorsa(Bende bir ara farklı konu başlıklarında olmak üzere günde 5 bine yakın mail gidiyordu), bir süre sona mail hesabınız şişmeye başlar, hatta admin'lerin belirlediği kapasiteye göre her gün bu kapasiteyi aşabilirsiniz. Bunun için yine otomatize edilmiş bir temizlik makrosuna ihtiyacınız olacaktır.

Şimdi öncelikle basit bir döngüsel mail silme nasıl oluyor ona bakalım. Silme işlemi **Delete** metodu ile yapılıyor.

Sub mailsilme()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myroot As Outlook.Folder, delFolder As Outlook.Folder

Dim silinecekler As Outlook.Items

Set ns = oApp.GetNamespace("MAPI")

Set myroot = ns.Folders("volkan.yurtseven@hotmail.com")

Set delFolder = myroot.Folders("klasörler")

Set silinecekler = delFolder.Items

For n = silinecekler.Count To 1 Step -1

silinecekler(n).Delete

Next n

End Sub

DİKKAT: For döngülerini anlatırken yaptığım bir uyarı vardı:

Akılda bulundurulması gereken önemli bir husus, ForEach kullanıldığında read-only bir özellik gösterir. Yani bu yöntemle dizi elemanlarını değiştiremezsiniz. Elamanları değiştirmek istiyorsanız basit For döngüsü kullanmanız lazım.

O yüzden silme işlemlerinde asla ForEach kullanmayın, normal For kullanın.

##### Otomasyon süreçlerinde toplu mail silme

Konu başında belirttiğim gibi, eğer sizin hesabınızdan günde binlerce mail gidiyorsa SentItems'ınızı arada bir temizlemek yararlı olacaktır.

Bunun için ilk önerim şudur, ki ben de böyle yapardım, otomasyon sürecinizdeki maillerin subject kısmında son karakteri "\*" veya benzer bir özel karakter koyun. Ör: "Mevduat çıkışlarınız hakkında\*". Böylece silinecek mailleri özel bir işaretle etiketlemiş olursunuz ve böylece bunları tek bir kontrolle silebilirsiniz. Aksi halde her gönderim subjecti için ayrı ayrı kontrol yapmak zorunda kalırsınız, bu da hem kod yazımını uzatır hem de silme işleminin uzun sürmesine neden olabilir.

Bu makroyu da yine ana schedule kodunuz içinde gece çalışacak şekilde kurabilirsiniz, zira silinecek mail çoksa silme işlemi gün içinde sizi bloke edebilir.

Şimdi SentItems içindeki mailler arasında subject'inde "\*" geçenleri bulmalıyız. Restrict metodu ile bunu yaparız ancak normalde yukarda restrict metodunda tam eşleşmeli bir kriter vermiştik. Şimdiki kriterimiz ise tam eşleşmeli değil, içinde "\*" geçenleri arıyoruz. Bunun için aşağıdaki özel yöntemi kullanıyoruz. Bu özel yöntemdeki 0x0037001f ifadesi sadece Subject alanına filtre uygulamış olur, diğer alanlar için MSDN’ye([https://msdn.microsoft.com/en-us/VBA/outlook-VBA/articles/referencing-properties-by-namespace](https://msdn.microsoft.com/en-us/vba/outlook-vba/articles/referencing-properties-by-namespace)) gerekmektedir.

Sub mailsilme\_kriterli()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim delFolder As Outlook.Folder

Dim silinecekler As Outlook.Items

Set ns = oApp.GetNamespace("MAPI")

Set delFolder = ns.GetDefaultFolder(olFolderSentMail)

kriter = "@SQL=""http: //schemas.microsoft.com/mapi/proptag/0x0037001f"" like '%" & "\*" & "%'"

Set silinecekler = delFolder.Items.Restrict(kriter)

MsgBox silinecekler.Count

For n = silinecekler.Count To 1 Step -1

silinecekler(n).Delete

Next n

End Sub

Bir nedenle maillerinizin ortasında \* işareti kullanıyorsanız, bu yöntem onların da silinmesine neden olabilir. O yüzden başka bir işaret kullanabilirsiniz. Veya SentItems içindeki tüm mailler içinde dolaşıp **If Right(subject,1) = "\*"** diye de kontrol edebilirsiniz.

##### Arşive kaldırma

Mailleri silmek yerine bir süre elinizde kanıt/kayıt olması babında arşiv klasörlerine kaldırmak isteyebilirsiniz. Zira bir gün birisi size gelip de şu alıcılara şöyle şöyle mail atmışsın(aslı astarı olmadığı halde) diyebilir. Arşivlemek için **Move** metodunu kullanabilirsiniz. Aslında arşivlemekten kastımız arşiv olarak belirlediğiniz bir klasöre taşımaktır, yoksa arşive özel bir durum yoktur.

bunun için kendinize bir hedef klasör belirlersiniz, sonra ilgili maili bu klasöre taşırsınız.

'Önceki tanımlamalar

Set hedefklasör = myInbox.Folders("Gönderilen Arşivi")

oMailitem.Move hedefklasör

Bu yöntemi sadece arşive kaldırma için değil herhangi bir klasöre taşımak için de kullanabilirsiniz. Aşağıdaki örnekte Sentitems'da subjecti "\*" karakterini içeren tüm mailleri arşiv klasörüne alıyoruz. Bu sefer Restrict metodu ile değil Find-FindNext metodu ile ilerledik.

Sub arşive\_kladırma()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myInbox As Outlook.Folder

Dim hedefklasör As Outlook.Folder

Dim myItems As Outlook.Items

Dim myItem As Object

Set ns = oApp.GetNamespace("MAPI")

Set myInbox = ns.GetDefaultFolder(olFolderSentMail)

Set myItems = myInbox.Items

Set hedefklasör = ns.Folders("volkan.yurtseven@hotmail.com").Folders("Gönderilen arşivi")

kriter = "@SQL=""http: //schemas.microsoft.com/mapi/proptag/0x0037001f"" like '%" & "\*" & "%'"

Set myItem = myItems.Find(kriter)

Do While TypeName(myItem) <> "Nothing"

myItem.Move hedefklasör

Set myItem = myItems.FindNext

Loop

End Sub

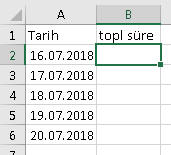
### Diğer İşlemler

Calendar'a Contact'lara ve Task'lere erişme de VBA ile pek tabi yapılabilir ama bu kitabın kapsamı göz önüne bulundurulduğuna çok fazla alakalı olduğunu söyleyemeyeceğim, ve şahsen ben de ufak tefek kodlar yazma dışında pek haşır neşir olmadım.

Mesela [http://www.globaliconnect.com/Excel/index.php?option=com\_content&view=article&id=166:export-contacts-from-outlook-to-Excel-automate-in-VBA&catid=79&Itemid=475](http://www.globaliconnect.com/excel/index.php?option=com_content&view=article&id=166:export-contacts-from-outlook-to-excel-automate-in-vba&catid=79&Itemid=475) sayfasında Outlook contact listenizdeki herkesin bilgilerini Excel'e alabileceğiniz bir örnek var.

Bunun dışında küçük işleriniz için bile outlook programlamadan faydalanabilirsiniz. Mesela daha bu hafta işyerinde, iş yükümü planlayan bir kod yazdım. Yeni işim gereği çok fazla toplantım oluyor, o yüzden yapmayı planladığım işlerin tahmini iş yüküne ek olarak toplantı saatlerini de toplam iş yükü listesine almam gerekiyor, ki kendime aşırı fazla yükleme yapmayayım.

Bunun için de **Recipients** nesnesinin **Freebusy** metoduna erişerek o haftanın toplantı saatlerini çekip onu bir dönüştürme işlemine tabi tutuyor, sonra da Excel’e diğer işlerin yanına çekiyorum.



Aşağıda topldurumu isimli değişkene yarım saatlik periyotlardan oluşacak şekilde 09: 00-18: 00 arasındaki toplantı durumlarını arıyorum. Bu değişkenin içeriği 000200222002002020 gibi birşey oluyor. 0 olanlar müsait, 2 olanlar toplantıda olduğunuzu gösterir(Bir de 1 var, Kararlaştırılmamış demektir). Sonra bütün bu 0 ve 2'yi tek tek karakterler olacak şekilde birbirinden ayırıp bir diziye(karakterler) aktarıyorum. Son olarak da bunları sayıya çevirip **Sgn** fonksiyonu işle işaret değerini yazdırıyorum. Bu fonksiyonla 0'lar için 0, 2'ler için 1 üretiyorum. Sonra bütün bunları toplayarak yarım saatlik periyotların kaçında toplantım olduğunu buluyorum, ve en son bu toplamı da 2'ye bölerek saat birimi cinsinden toplantı süremi buluyorum.

Sub toplantı\_süresi\_bul()

Dim oApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim rcp As Outlook.Recipient

Dim dt As Date

Dim karakterler() As String

Set ns = oApp.GetNamespace("MAPI")

Set rcp = ns.CreateRecipient(12345) 'buraya outlookta sizi tanımlayan sicil numaranız v.s gelecek

For Each dts In Range("tarihler") 'tarihler diye bir NamedRange var

dt = DateValue(dts)

topldurumu = Mid(rcp.FreeBusy(dt, 30, True), 19, 37)

karakterler = Split(StrConv(topldurumu, vbUnicode), Chr$(0))

ReDim Preserve karakterler(UBound(karakterler) - 1)

For Each krk In karakterler

toplam = toplam + Sgn(Val(krk))

Next krk

süre = toplam / 2

dts.Offset(0, 1) = süre 'ilgili tarih hücreinin 1 sağındaki hücreye o günkü toplam toplantı süresini yazdırıyorum

toplam = 0 'resetliyorum

Next dts

Set oApp = Nothing

End Sub

Siz de kendinize şöyle bir ödev verebilirsiniz: Contact List'teki tüm alıcılara ismiyle hitap ettiğiniz bir yılbaşı kutlamam otomasyonu.

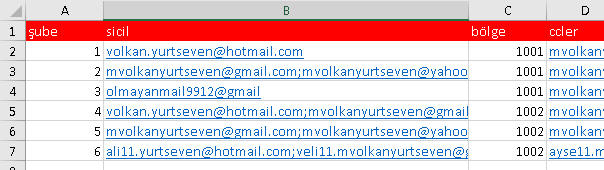
### Otomatik Toplu Mail Gönderim uygulaması

Çalıştığım kurumda genelde bölme makrosuyla(**Userformlar>Form Kontrolleri:Çeşitli Örnekler:Bölme Formu**) kullanılan ve en çok kullanılan kodlarımdan biri de mail gönderim formudur. Bu form ek olmadan da kullanılabilmekle birlikte, ek de göndermeyi düşünüyorsanız ekleri oluşturmak için bu bölme makrosuna ihtiyaç duyulmaktadır. Tabi isteyen öyle bir kod yazar ki, ayrı bir bölme koduna ihtiyaç olmaz; aynı kod içinde önce bölüp hemen arkasından da mail gönderir.

Ben ise farklı bir yöntem belirledim, önce bölme makrosu ile tüm dosyaları oluşturuyorum, sonra da mail atma işlemini yapıyorum. Yani özetle iki ana işi birbirinden ayırdım. Zira böldüğümüz şube dosyalarını zipleyip bölgeye de topluca atma durumu olabilir, o yüzden benim yöntemi benimsemenizi tavsiye ederim.

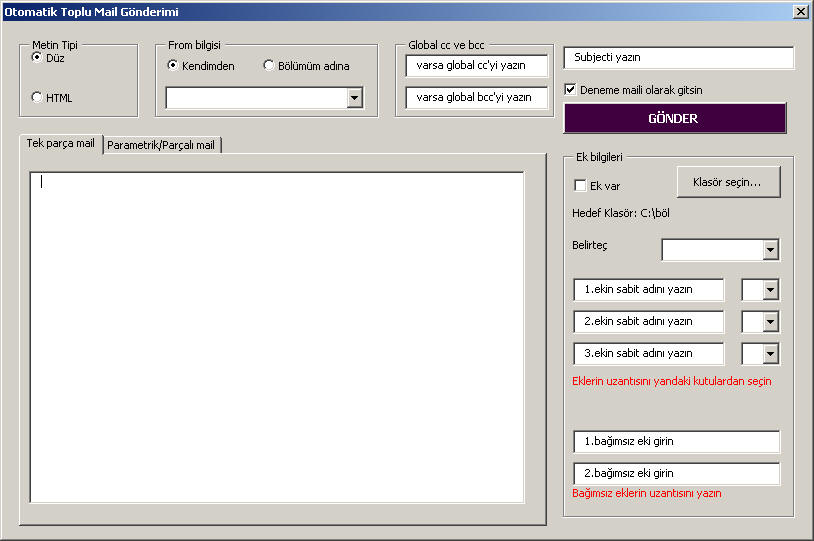
Daha yukarıda belirttiğim gibi, bunun daha farklı ve gelişmiş bir versiyonu Excelent içinde yer almaktadır. İstediğiniz, kod yazmadan böyle bir çözüme ulaşmaksa Excelent'ı kullanabilirsiniz. Şimdi kod yazmak isteyenlerle devam edelim.

**Amacımız** aşağıdaki gibi bir listedeki alıcılara tek tek kendilerine özel olacak şekilde mail göndermek.



Öncelikle aşağıdaki gibi bir form oluşturacağız.(Buraya koyacağım form, işyerinde kullandığıma göre biraz daha basit olacak, siz bunu daha işlevsel hale getirebilirsiniz.)

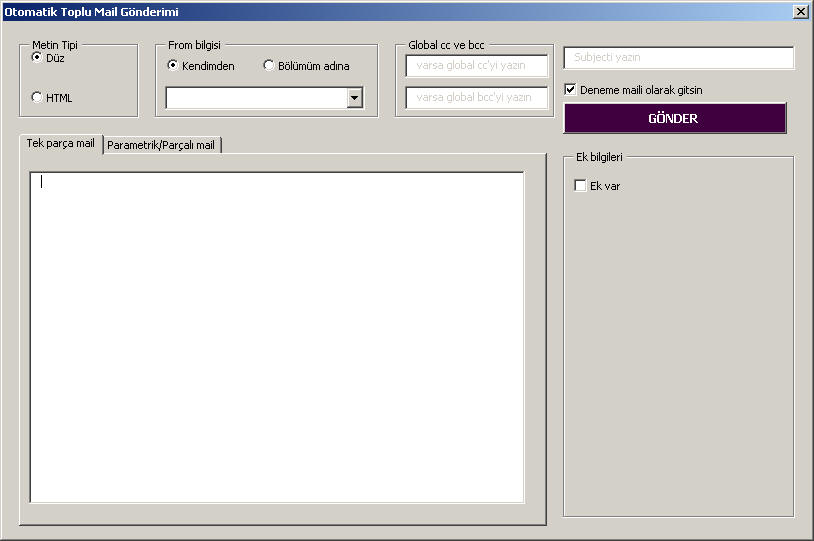
Gerekli açıklamaları hem kod içinde hem de aşağıdaki çeşitli yerlerde bulacaksınız.



Bu formu açmak için sayfaya bir buton koydum ancak tabi ideal dünyada bunun bir Add-in'den açılması gerekir. Bunu yapmayı da **İleri Seviye Konular>Addinler yaratmak** konusunda göreceğiz.

#### Formu oluşturma

Bu formun başlangıç ayarlarını öyle bir yapacağız ki, aşağıdaki gibi görünecek.(Siyah yazılar grileşti ve bazı kontroller gizlendi)



Başlangıç için Properties ayarları ve Formun Initiliaze eventi aşağıdaki gibidir.

Bu arada belirtmem gerekir ki gerekli tüm kontrollere anlaşılır bir isim verdim. Mesajı dinamik değişmeyen label'lara veya sadece bir kez kullanılan Multipage gibi kontrollere özel isim vermedim.

##### Properties ayarları

txtBody'nin ve parametrik mail sayfasındaki 4 kutunun **EnterKeyBehaviour** özelliğini True yapın. Böylece Enter tuşuna bastıkça bir alt satıra geçersiniz, aksi halde textbox içinden çıkmış olursunuz.

Yine aynı kontrollerin **Multiline** özelliğine de True atayın.

* optMetinTip'i seçili halde başlatmak için Value=True yapın.
* optFromKendimden'in de Value değerini True yapın.
* chkDeneme'nin de Value değerini True yapın.

##### Initialize ayarları

Burada tek seferde yapılan görünmezlik, renk ayarı ve combobox değer atamalarını yapıyoruz.

Private Sub UserForm\_Initialize()

Dim ctrl As Control

MultiPage1.Value = 0 'her zaman ilk sayfa açık gelsin

'ek'teki tüm kontrolleri görünmez yapıyoruz, checkbox hariç

For Each ctrl In Me.frmEk.Controls

If Not TypeOf ctrl Is MSForms.CheckBox Then

ctrl.Visible = False

End If

Next ctrl

'şimdi tüm textboxların metnini gri yapalım

For Each cf In Me.Controls

If TypeName(cf) = "Frame" Then

For Each ctrl In cf.Controls

If TypeName(ctrl) = "TextBox" Then

ctrl.ForeColor = &H808080

End If

Next ctrl

End If

Next cf

Me.txtSubject.ForeColor = &H808080 'bu frame içinde değil, ayrıca yapıyoruz

'şimdi de comboboxların değerlerini yüklüyoruz

Me.cbFromBölümler.List = Array("satış@kurum.com", "pazarlama@kurum.com", "ik@kurum.com")

Me.cbEkUzantı1.List = Array("xlsx", "xlsb", "xls", "pdf", "docx", "doc", "zip", "rar")

Me.cbEkUzantı2.List = Array("xlsx", "xlsb", "xls", "pdf", "docx", "doc", "zip", "rar")

Me.cbEkUzantı3.List = Array("xlsx", "xlsb", "xls", "pdf", "docx", "doc", "zip", "rar")

Me.cbEkUzantı1.Value = "xlsx"

Me.cbEkUzantı2.Value = "xlsx"

Me.cbEkUzantı3.Value = "xlsx"

Me.cbBelirteç.List = Array("Sicil", "Şube", "Bölge")

End Sub

Şimdi de font rengini grileştirdiğimiz textbox'lara tıklandığında hem talimat metnini silen hem de metin rengini siyaha dönüştüren kod parçasını yazalım. Şimdilik bu kontrollere tek tek kod yazıyoruz, ancak yazılan kodu fonksiyonel yapmak için ana bir fonksiyon yazdım, textboxların Enter eventinde bu kodu çağırıyorum. (Bununla birlikte bizi her bir kontrol için tek tek aynı kodu yazmaktan kurtaran bir yöntem var. Bunu Class Modül içine bir custom eventi olan değişken tanımlayarak yapabiliyoruz. Bununla ilgili bilgiyi Form kontrolleri konusunda bulabilirsiniz)

'-----EVENT BÖLGE BAŞLANGICI: initializede grileştirilen tüm textboxlar için event ayarlıyoruz

Sub txtEnter(txt As MSForms.TextBox)

If txt.ForeColor = &H808080 Then

txt.Text = ""

txt.ForeColor = vbBlack

End If

End Sub

Private Sub txtGlobalcc\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtGlobalcc

Call txtEnter(txtctrl)

End Sub

Private Sub txtGlobalbcc\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtGlobalbcc

Call txtEnter(txtctrl)

End Sub

Private Sub txtBağımsızEk1\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtBağımsızEk1

Call txtEnter(txtctrl)

End Sub

Private Sub txtBağımsızEk2\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtBağımsızEk2

Call txtEnter(txtctrl)

End Sub

Private Sub txtEk1\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtEk1

Call txtEnter(txtctrl)

End Sub

Private Sub txtEk2\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtEk2

Call txtEnter(txtctrl)

End Sub

Private Sub txtEk3\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtEk3

Call txtEnter(txtctrl)

End Sub

Private Sub txtSubject\_Enter()

Dim txtctrl As MSForms.TextBox

Set txtctrl = Me.txtSubject

Call txtEnter(txtctrl)

End Sub

'-----EVENT BÖLGE BİTİŞ

##### Framelerin içi

Bölüm/departman adına mail göndermek isterseniz, bu combobox’a tıkladığınızda bununla ilgili seçeneği aktive edelim.

Private Sub cbFromBölümler\_Enter()

Me.optFromBölüm.Value = True

End Sub

Kimden gideceği ile ilgili olarak optionbuttonlar’da seçim yapıldığında sadece birini kontrol etmek yeterli, diğeri zaten otomatikman if kontrolünün Else bölümüne düşer.

Private Sub optFromMe\_Change()

If Me.optFromBölüm.Value = True Then

Me.cbFromBölümler.Value = "varsayılanbirim@kurum.com"

Me.cbFromBölümler.Enabled = True

Else 'kendi adresimden gidecekse

Me.cbFromBölümler.Value = ""

Me.cbFromBölümler.Enabled = False

End If

End Sub

Şimdi de eklerle ilgili framede, ek checkbox’ına tıklandığında neler olduğuna bakalım. Burada **After\_update** eventine bakıyoruz. Chekbox’ın kendi dışındakileri, hep bir önceki(zıttı) değerine atıyoruz. Görünürlerse görünmez, görünmezlerken görünür olsunlar.

Private Sub chkEk\_AfterUpdate()

Dim ctrl As Control

For Each ctrl In Me.frmEk.Controls

If Not TypeOf ctrl Is MSForms.CheckBox Then

ctrl.Visible = Not ctrl.Visible

End If

Next ctrl

End Sub

Son olarak, ek çerçevesindeki Klasör seçme düğmesindeki koda bakalım. Burada varsayılan olarak [C: \böl](file:///C:\böl) klasörü dikkate alınırken(zira bölme makrosu da varsayılan olarak dosyaları bu klasöre atar) bu düğme ile istersek dosyaların bulunduğu klasörü değiştirebiliriz.

Private Sub btnKlasör\_Click()

Dim fd As FileDialog

Set fd = Application.FileDialog(msoFileDialogFolderPicker)

With fd

.Title = "Klasör seçin"

If .Show = True Then

Me.lblKlasör.Caption = .SelectedItems(1)

End If

End With

End Sub

##### Ana mail gönderim kodu

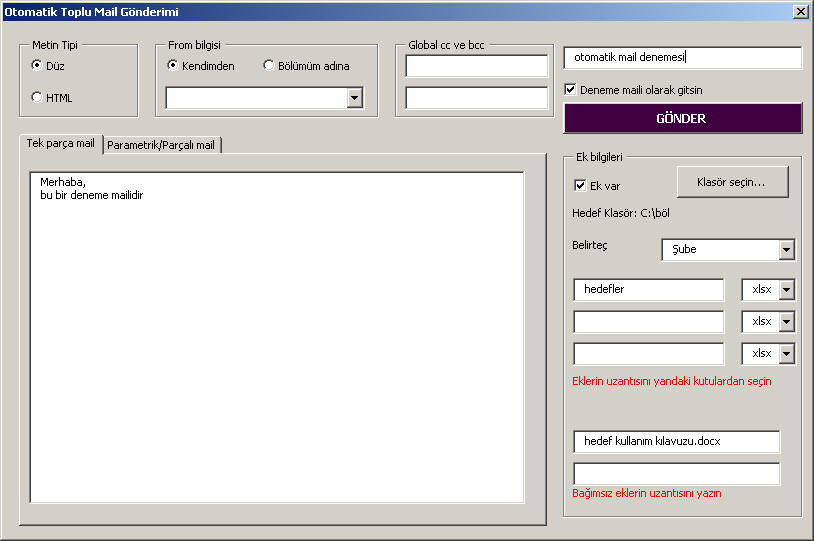
Şimdi geldik esas baba kodun olduğu kısma. Burada şimdiye kadar gördüğümüz birçok tekniği kullanacağız. Hata yönetiminden, DoEvents kullanımına, dizilerden döngülere kadar birçok içerik var.

Öncelikle performans açısından önem arz eden bir kodlama olacağı için kendimizi değişken tanımlamaya zorlayalım, en tepeye **Option Explicit** koyalım.

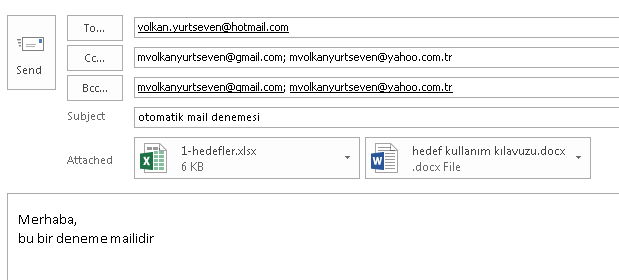
Bu örnekte Outlook nesnesini Late Binding ile yaratacağım. Çünkü bu uygulamayı sadece kendim kullanmayı düşünmüyorum, başka arkadaşlara da göndermeyi düşünüyorum, o yüzden olur da onların Office versiyonu benden farklı olabilir diye böyle yapıyorum. Bunu yaparken önce Early Bindingle başladım, böylece intellisense’ten faydalandım, sonra kodları Late Binding'e çevirdim.

Kodumuz biraz uzun olduğu için **githubdan(btnGönder\_Click)** görebilirsiniz.

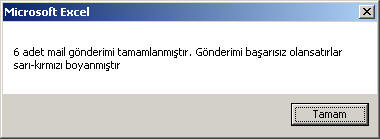
Form'u açıp aşağıdaki gibi doldurup, deneme maili gönderiyoruz. Burada ek olarak bir tane parametrik ek, bir tane de sabit bağımsız ek koyuyoruz. Gönderim yaptığımız kişilere **şube koduna göre bölünmüş** dosyaları atacağımız için **belirteç olarak Şube** seçiyoruz.



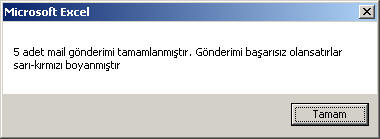
Örnek deneme maili aşağıdaki gibi oluyor. Kod içindeki açıklamalardan göreceğiniz üzere, deneme mailini **Send** metodu ile değil **Display** metodu ile oluşturuyoruz.

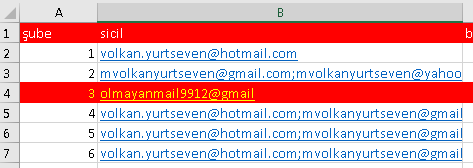


"Deneme" chekcbox'ınbdan işareti kaldırıp gönderirsek hepsi gider ve aşağıdaki mesajla karşılaşırız.



Mail adreslerinden birini bilerek hatalı hale getirirsem, kodumuz bu satırı atlar ve onu işaretler. Hatalı mailden kastımız, ribbondan Check Names dediğimizde bize sorun çıkaran mail adresleridir.



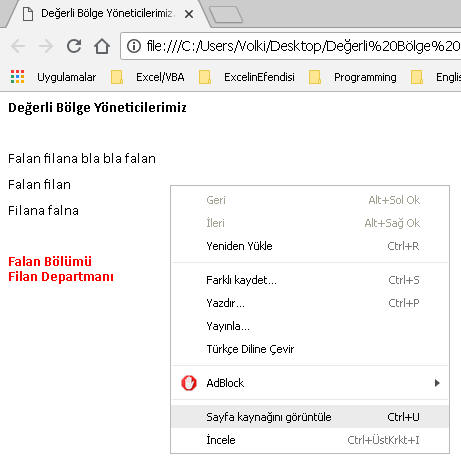


Tüm mailler giderken statusbar’da da ne aşamada olduğumuzu görürüz. 6 mail olduğu için sırayla(az olduğu için çok hızlı) %16,33,49,66,83,100 olarak görürüz.



##### HTMLBody ile formatlı maili gönderimi

Göndermek istediğimiz mail özel formatlandırılmış olacaksa bu maili önce Word’de hazırlarız, sonra bunu **web page** olarak(**html** uzantılı) kaydederiz. Sonra bu dosyayı bir tarayıcıda açarız, ve sayfada herhangi bir yere sağ tıkayıp aşağıdaki gibi **sayfa kaynağını görüntüle** deriz.



Word, bir sürü ıvır zıvır kodu kendiliğinden üretir. Bunların çoğu bize lazım değildir, aşağıdaki gibi Body tag'inden sonraki kısmı alırız ve bunu formumuzdaki kutuya yapıştırırız.



Sonra bunu <html> </html> tagleri içine alırız.

<html>

<body lang=TR style='tab-interval: 35.4pt'>

<div class=WordSection1>

<p class=MsoNormal><b style='mso-bidi-font-weight: normal'>Değerli Bölge

Yöneticilerimiz<o: p></o: p></b></p>

<p class=MsoNormal><o: p>&nbsp;</o: p></p>

<p class=MsoNormal>Falan filana bla bla falan</p>

<p class=MsoNormal>Falan filan</p>

<p class=MsoNormal>Filana falna</p>

<p class=MsoNormal><o: p>&nbsp;</o: p></p>

<p class=MsoNormal><b style='mso-bidi-font-weight: normal'><span

style='color: red'>Falan Bölümü<br>

Filan Departmanı<o: p></o: p></span></b></p>

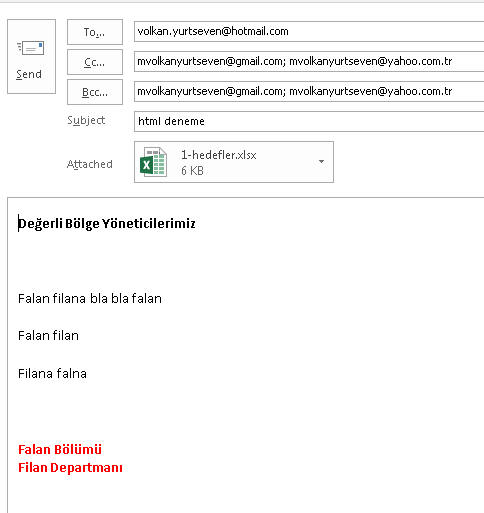
</div>

</body>

</html>

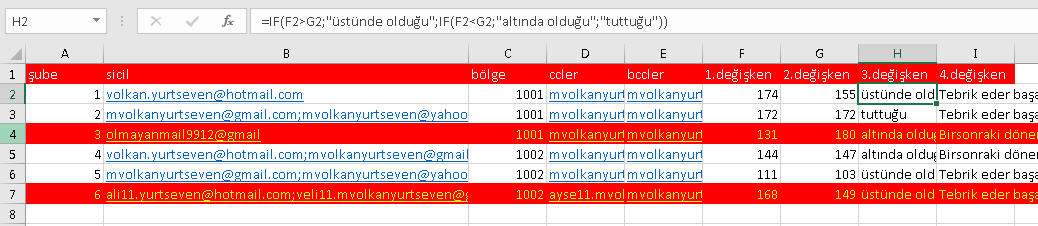
Kafanız çok karışacak gibi olursa hepsini de alabilirsiniz, sıkıntı olmaz.

Şimdi bir de ek ekleyelim ve Gönder diyelim: İşlem tamam.

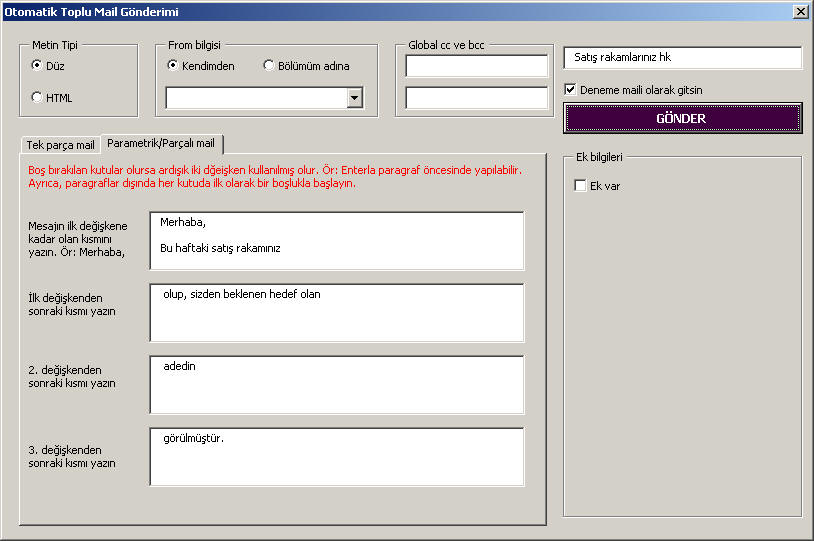


##### Parametrik bilgili mail gönderimi

Listemiz aşağıdaki gibi. İlk değişkende personelin yaptığı satış adetleri, onun yanında hedefleri, onun yanında hedefine göre nerede olduğu ve nihai bir mesaj var. Buraya ilaveten kişinin adı, hitap şekli için cinsiyeti v.s de olabilirdi, biz basit olsun diye değişken miktarını 4'te tuttuk.



Formumuzu açalım ve aşağıdaki gibi dolduralım:



2. kutudan itibaren tüm kutulara bir boşlukla başlamak gerekiyor, yoksa kendinden önce gelen değişkene yapışık çıkar. Sonuçta giden mailimiz aşağıdaki gibi görünecek.



Maalesef bu kod ile **html** ve **parametrik’i** bir arada çalıştıramayız, bunun için ilave kod yazılması gerekir. Ben açıkçası bununla uğraşmadım ancak isterseniz siz yapabilirsiniz, sizin için de güzel pratik olur. Bu arada unutmayın, eğer cc ve ek gönderimi olmayacaksa parametrik ve formatlı mail gönderimini Word'ün mail merge aracı ile de yapabilirsiniz.

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

## Veritabanı Programlama Terminolojisi

### Giriş

Veritabanı/Database(VT/DB) işlemlerine geçmeden önce genel terminolojiye hakim olmak, konuları anlamak açısından oldukça faydalı olmaktadır. O yüzden ilk olarak terminoloji ile başlayacağız.   
Kanımca, burdaki terimleri içselleştirmeden ilerlemek sizin VT bağlantılarını tam anlamıyla anlamayacağınızı, ezbere ve copy-paste(internetten veya makro recorderdan) kodları çalışmak zorunda kalacağınızı garantiler. Bu da kodunuzun çirkin ve uzun görünmesine neden olacağı gibi, zamanla “ODBC neydi, neden bunda OLE DB olmuş da ODBC olmamış, bunda neden CommandText de şunda CommandType yazıyor” gibi soruları sormaya başladığınızda sorularınızın cevapsız kalmasına neden olacaktır.

Terminoloji konusu çok kapsamlı bir konu olmakla birlikte ben buraya özellikle VBA ile doğrudan veya dolaylı alakası olan konuları koymaya çalıştım. İnternette de bol miktarda makale var, ben de bir nevi kendi tecrübelerim ile bu makalelerin derlemesini yaptım diyebilirim, yoksa kendim bir VT uzmanı değilim.

### Genel Terimler

İlk etapta bu bölümde yazılanlardan genel bir fikir edinmeye çalışın, her şeyi anlamaya çalışmayın, zaten bu konulara aşina değilseniz biraz karmaşık gelebilir. Sonrasında hemen diğer Veritabanı sayfalarına geçin. Ancak veritabanı bağlantıları konusunda deneyim sahibi oldukça bu bölüme gelip bilgilerinizi tazeleyin.

**COM(Component object model):** Programlama dilleri, hatta platformlar arası uyumsuzlukları ortadan kaldırma amacıyla ortaya çıkan bir standarttır. Microsoft(MS)’un teknolojisi değildir.

**ActiveX**: Nesnelerin yaratıldıkları dilden bağımsız olarak birbiriyle etkileşmesini sağlayan MS’a özgü COM teknolojisidir.

**API(Application Programming Interface)**: Uygulamaların birbiriyle anlaşmasını sağlayan program paketleri.

**OLE**: MS’un bir uygulamasında bir başka MS uygulamasının içeriğini gösterebilme teknolojisi. Ör: Excel’deki bir grafiği Word içine kopyalayabilirsiniz. Ve bu Word dosyasını birisi açtığında bilgisayarında Excel yüklü olmasa bile sorunsuz açılır. Bu da bir COM türüdür.

**MDAC**: Microsfotun MDAC(Microsoft Data Access Components) adı verilen teknolojileriyle programlama dillerinden VT bağlantıları yapabilmekteyiz. Bunlar 3 önemli API’yi içerir.   
   
**ADO,OLE DB,ODBC**  
Bu konuda o kadar çok araştırma yaptım ki, bunları spesifik olarak tanımlayan bir ifade bulamadım. Bunlar için farklı yerlerde API, Model, Standart, Teknoloji gibi isimler kullanılmaktadır. Her ne kadar MSDN sitesi bu 3’ünü aynı kategori içinde ele aldıysa ben araştırmalardan edindiğim izlenime göre bunları Data Sağlayıcıları ve Erişim Arayüzleri olarak iki sınıfta ele alacağım, zira bu 3 terime yapacağım eklemeler de olacak.

### Veri sağlayıcılar

Bir veritabanı üzerinde data okuma, yazma veya güncelleme gibi işlemlerin yapılabilmesi için bu kaynaklara ulaşılması gerekmektedir. Veritabanı yönetim sistemlerine erişim yapabilmek için iki temel yöntem bulunur: ODBC ve OLE DB.

#### ODBC(Open Database Connectivity)

MS’un veritabanlarına programlama dillerinden erişmek için ortaya attığı standarttır ancak sonunda bir endüstri standardı olmuştur. Neredeyse tüm VT şirketleri kendi ürünlerine ait ODBC driverı(sürücüsü) geliştirmektedirler.

ODBC, uygulamaların veritabanlarına erişimini sağlayan bir arayüz oluşturur. Driverlar aracılığı ile uygulama ve veritabanı arasında köprü oluştururlar. Driverlar SQL komutlarını veritabanına iletir ve sonuç kümesini döndürür.

ODBC ile sadece ilişkisel veritabanı(verinin satır ve sütunlar şeklinde tutulduğu) erişim sağlanır. Oracle, SQL Server, DB2, MySql, Access gibi. Bu yüzden iletişim için yalnızca SQL metni kullanır.

Bununla birlikte ODBC’yi doğrudan VBA ile temasa sokamıyoruz, çünkü low level(alt seviye) bileşenleri var. Onun yerine onu sarıp sarmalayıp dokunabilir hale getiren arayüzlerle(DAO gibi) dokunabiliyoruz, ki buna birazdan geleceğiz.

ODBC dünyasında akış şöyledir: **Uygulama(Ör: VBA)--> SQL-->DAO-->ODBC-->Veritabanı**

#### OLEDB(Object Linking and Embedding Database)

OLE DB, ODBC’den sonra gelmiştir. MS’un ODBC’ye şekil verip onu ActiveX modeline soktuğu standarttır. COM tabanlıdır. Her tür veritabanına erişim için tasarlanmıştır. Yani hem relational(ilişkisel) hem de non-relational(ilişkisel olmayan) veritabanlarından veri çekebilmektedir. Non-relational’a örnek olarak Email sistemlerini, Text dosyalarını, Excel dosyalarını verebiliriz.

ODBC’de driver ne ise OLE DB’de provider odur. Yani veriye providerlar aracılığı ile ulaşılır. OLE DB ayrıca ODBC’ye bir köprü de atarak, ODBC driverlarının da kullanımını sağlar. Provider listesinde “OLE DB provider for ODBC Driver” olarak görünen şey budur. Amacı, ODBC ile ADO’yu konuşturmaktır. Çünkü az önce söylediğimiz gibi ODBC normalde sadece DAO(söz konusu uygulama VBA ise) ile konuşuyor. Neden ADO ile konuşmak istediğine az sonra geleceğiz.

OLEDB, iletişim için SQL dışında da teknikler kullanır(XML gibi). Outlooktaki belli kişilerden gelen mailleri filtreleme işlemi de OLEDB ile yapılıyor.

ODBC’den farklı olarak, OLE DB’yi doğrudan kullanabilmekle birlikte genelde VBA ile doğrudan temasa sokamıyoruz, çünkü bunun da ODBC gibi low level bileşenleri var. Onun yerine onu sarıp sarmalayıp dokunabilir hale getiren arayüzlerle(ADO) VBA’ye iletişme sokuyoruz, ki buna birazdan geleceğiz.

OLEDB dünyasında akış şöyledir: **Uygulama(Ör: VBA)-->SQL/XML/v.s -->ADO-->OLE DB-->Veritabanı(RDBMS veya nonRDBMS)**

**NOT**: Bir ara MS’un ODBC’yi desteklemeyi durduracağı, tüm geliştirmeyi OLE DB üzerinde yapacağı ve ODBC’nin OLE DB tarafından replace edileceği söyleniyordu ama son zamanlarda(2012’den beri) rüzgar tersine esmeye başladı, ODBC tekrar gözde oldu. En büyük sebep de performans olarak gösteriliyor. Gelecekte ne olacağını öngörmek zor.

### Erişim arayüzleri(API’ler)

Bu kısımda da yukardakilerden çok da ayrılamayan ama mantık olarak birbiriyle daha ilişkili olan diğer terimlere, API’lere, bakacağız. Bunların hepsi birbiriyle ilişkili, ancak daha çok “ODBC vs OLE DB” ile “ADO vs DAO” karşılaştırmaları yapıldığı için ben de böyle bir gruplama yaptım. Yoksa hem yukarda kısmen gördüğünüz hem de birazdan göreceğiniz gibi ADO ile OLE DB de oldukça örtüşen kavramlar.

VBA’de veri erişim arayüzü olarak 3 teknoloji bizlere sunulmuş durumda. Veri erişim arayüzü ne demek? Veriye erişim için bir nesne modelinin hazırlanmış olması demek. VBA de nesneye dayalı programlamayı desteklediği için bu yöntemlerle veriye erişiriz. Bu 3 teknoloji şunlardır:

**DAO (Data Access Objects):** Yerel veritabanları için(Access gibi) geliştirildi. Hatta Access'e özgüdür bile diyebiliriz. ODBC ile bağlanır. İlk bu çıkmıştır.

**RDO (Remote Data Objects):** Oracle gibi daha büyük çaplı veri tabanları için geliştirildi, DAO’nun hafıza ve yüksek kaynak tüketimi sorunları nedeniyle ortaya çıktı. Biz bunu hiç kullanmayacağız.

**ADO (ActiveX Data Objects)**: Sonradan geldi ve her tür VT için kullanılmaya başlandı. MS’un OLE DB teknolojisiyle de etkileşim içinde çalışır. ADO’da DAO’ya göre daha az nesne ama daha çok nesne üyesi(özellik, metod, olay) bulunur.

Neden 3 ayrı yöntem tane var diye soracak olursanız teknolojinin sürekli gelişmesini ve müşteri beklentilerindeki değişikliği söyleyebiliriz. Mesela DAO’yu web server üzerinde kullanamıyoruz ancak eski kullanıcıları da üzmemek için eskilere destek sürüyor. Bunların en sonuncusu ve güçlüsü  ADO’dur diyebiliriz. O yüzden yeni kodlarımızda mümkünse ADO kullanmalıyız, ancak eski kodları da anlamak adına ve sadece Access içinde kalacaksak DAO’yu da öğrenmekte fayda var.

Sonraki sayfalarda bunlara detaylı bakacağız, ancak şimdi kısaca bahsetmek istiyorum. Bu arada önemli bir hatırlatma yapmak isterim. Gerek DAO gerek ADO olsun, bunlarda veritabanına linkli bir bağlantı kurmuyoruz, yani datayı refreshlenebilir(sağ tıklayıp Refresh düğmesiyle güncellenebilir) bir şekilde almıyoruz. Bunun yerine veritabanına bağlanıp datayı okuyoruz(Tabi istersek aynı zamanda datayı değiştirebiliyoruz/silebiliyoruz da). Refreshlenebilir data bağlantısı(ki sadece bağlantı kurulur, editleme/silme yapılamaz) için başka teknikler var, onları da sonraki sayfalarda göreceğiz.

#### DAO

Microsoft, DAO’yu, Acces'in kullandığı Microsoft JET Database’ine erişim sağlamak için geliştirmiştir. DAO, COM-tabanlı olup ODBC bağlantıları için kullanılır.

Accessle çalışırken DAO kullanmak daha iyidir diyebiliriz. DAO’nun ADO’ya göre daha çok nesnesi vardır ve Access'e özgüdür demiştik. Bu arada “Accessle çalışırken” ifadesinden kastım Access VBA değil, Excel VBA içinden Access’e bağlanmayı kastediyorum. Aynı şekilde aynı kodlar tabiki Access VBA içinde de kullanılabilir ancak Access VBA için Access nesne modelini bilmek gerekir.

DAO, akılda Jet DB motoru bulundurularak tasarlandığı için ADO seçenek bile olmamalı diyenler de var. MS tarafından ADO’nun DAO’nun tüm özelliklerini taşıyacak şekilde geliştirileceği söylenmiş ancak bu henüz olmamıştır. Ben şahsen Access'le sadece DAO kullanıyorum, size de bunu öneriyorum. Diğerlerinde ADO kullanıyorum.

Bununla beraber Access 2007’den itibaren DAO yenilenmiş ve ACEDAO adını almıştır, Jet(Joint Engine Technology) motoru da ACE(Access Connectivity Engine) motoru tarafından replace edilmiştir.

#### ADO

ADO, MS’un sonradan ortaya çıkardığı, veriye erişim için kullanılan COM nesneleri kümesidir. Programlama dili ve OLE DB arasında katman sunar. Bizim kapsamımızda programlama dili VBA oluyor. VBA kodlamacısı olarak bize, veritabanının nasıl ele alındığını bilmeden dataya erişen programlar yazma imkanı verir. SQL bilmeye bile gerek yoktur, bununla beraber SQL komutları da çalıştırılabilir. 4 Collectionu ve 12 nesnesi vardır. Kullanımı basittir. Bu detaylara bilahare değineceğiz.

Yukarıda belirttiğimiz gibi OLE DB ayrıca ODBC’ye bir köprü de atarak, ODBC driverlarının da kullanımını sağlar. Amacı, ODBC ile ADO’yu konuşturmaktır.

### Kullanım ve Farklar

* DAO ODBC ile, ADO hem OLE-DB hem wrapper sayesinde ODBC ile çalışır.
* DAO’da daha çok nesne daha az metod, ADO’da az nesne çok metod bulunur.
* DAO daha hızlıdır.
* DAO Access’e özgü, ADO tüm veritabanlarına uygun(Aynı şey ODBC ve OLEDB için geçerli değil. Tamam; VBA ODBC’ye ulaşırken ara katman olarak DAO’yu kullanır ancak bu sadece VBA ve Access ikilisi içindir, başka programlama dillerinden başka DB’lere bağlanılabilir, fakat onlar için ara katmanın ne olduğunu bilmiyorum)
* DAO web uygulamalarında kullanılmaz, ADO kullanılır.
* Bundan sonraki sayfada DAO ve ADO’yu daha detaylı ele alacağız. Yalnız, bu iki konunun tüm detaylarını ele almayacağız, zira oldukça geniş konular. İşimize yarayan konulara ağırlık verilecek, MIS ağırlıklı bir kişi olarak kullanma ihtimali olan bazı detaylara da girilecek ama kullanma ihtimaliniz olmayan veya çok düşük olan detaylar atlanacaktır. Daha detaylı bir öğrenim için lütfen MSDN veya diğer kaynaklara bakabilirsiniz.

**NOT**: "Ben iki ayrı teknolojiyi bilmekle uğraşmayayım sadece tek şeyi bilsem olmaz mı?" diyorsanız, o zaman size önerim direkt ADO’ya bakın.

## Veritabanı İşlemleri - DAO ve ADO

Bu bölümde gösterilen örneklere ait dosyaları **githubdan(VBAdata.rar)** indirebilirsiniz.

### DAO DETAY

#### Giriş

Önceki sayfayı okuduysanız DAO’nun MS Accese’e özgü bir API olduğunu anlamış olmalısınız. Daha önce belirttiğimiz gibi MS Access dışında bir veritabanına bağlanmak için yeni kodların ADO ile yazılması tavsiye ediliyor ancak mevcut kodlarınızı ADO’ya çevirmeniz pek bir anlamı yok. Accesse gelince, kullanımı daha pratik olduğu için ben hala DAO'yu kullanıyorum. Seçim size kalmış, siz hangisinde kendinizi rahat hissederseniz onu kullanabilirsiniz.

##### Motor/Engine

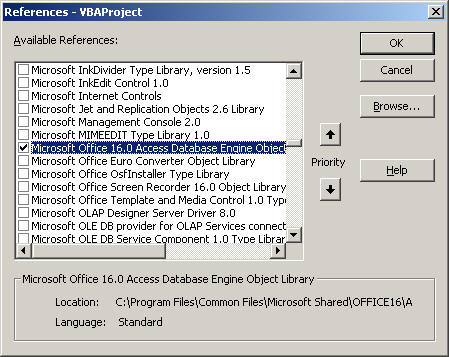
DAO, dataya erişim için JET DB motorunu kullanırdı, ancak 2007 versiyonu ile birlikte ACE geldi. ACE ile eski mdb dosyalarına da yeni accdb dosyalarına da erişebileceğiz. Ancak araştırmalarınız sırasında JET’e de rastlarsanız şaşırmayın.

Şimdi gelin bunu kullanabilmek için neler yapılmalı bir bakalım.

##### Referans ekleme

Öncelikle DAO kütüphanesini VBA’e eklemek gerekiyor. Personal.xlsb üzerinde çalıştığımız düşünerek bu dosyadayken aşağıdaki işlemleri yapalım.

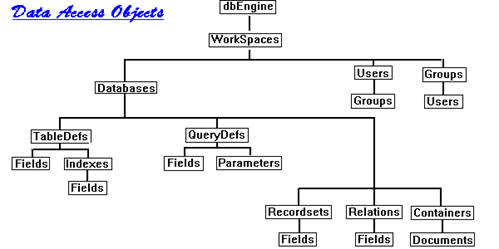
Birçok yerde DAO 3.6 ekleyin denir, ama bunların çoğu eski siteler. Artık Access 2003 kullanan kalmadığını düşünerek bunun yerine **Microsoft Office x.x Access Database Engine Object Library** şeklindeki referansı eklemeniz gerekecektir. Bendeki aşağıdaki gibi Access 2016 olduğu için 16 versiyonunu ekledim, sizde bu biraz daha farklı olabilecektir. (Not: "Microsoft Access 16.0 Object Library" olan kütüphane Access nesne modeline erişmemizi sağlar, buna dikkat edin, isim benzerliği karışıklık yaratabilir, biz onu eklemiyoruz, "Microsoft Office 16.0 Access Database Engine Object Library"'sini ekliyoruz)



#### Nesneler

DAO’daki nesneler veritabanına bağlanmak, dataya erişmek ve veritabanının yapısını değiştirmek için kullanılır. En tepede **DBEngine** nesnesi olan hiyerarşik yapının genel görünümü aşağıdaki gibidir.

##### DAO Nesne Modeli



Biz bunlardan DBEngine ve Workspace’i hiç kullanmayacağız. Bu yüzden yaptığımız tüm database işlemleri default workspace üzerinde olmuş olacak. Farklı oturumlarda farklı workspace açma ihtiyacınız olursa MSDN’den(<https://msdn.microsoft.com/en-us/library/office/ff822782.aspx>) detay bilgi edinebilirsiniz.

DAO ile çalışırken genel süreç şöyledir:

* Database ve recordset nesnesi tanımlanır,
* DB ataması yapılır,
* DB üzerinden recordset yaratılır,
* Sonrasında dataya erişilir,
* Tüm işlemler bitince recordset ve DB Nothing atanıp kapatılır

##### Tanımlamalar

DAO ve ADO’nun bazı ortak nesneleri var. O yüzden özellikle iki referansı da birden kullanıyorsanız mutlaka referans(library) ismini nesnelerin önünde kullanmanız gerekir, yoksa karışıklık çıkar ve hata alırsınız. Ancak aynı ismi kullanmayan nesneler için referans belirtmeye gerek yok.

Dim db As DAO.Database '(Bunda DAO’ya gerek yok çünkü ADO’da Database nesnesi yok, karışma olmaz)

ama

Dim rs As DAO.Recordset 'ya da Dim rs As ADODB.Recordset

"DAO"’yu yazmak intellisense’in çıkması adına bir kolaylık sağladığı için ben size bunu sürekli kullanmanızı(kafa karışıklığı olmayan durumlarda bile) tavsiye ederim.

##### Database nesnesi

Öncelikle **Dim db As DAO.Database** diye tanımladık. New ifadesi olmadan tanımlama yapılır. Zira yaratımını bir fonksiyon ile yapacağız.

Bazı kaynaklarda bunun arkasından Set db = DBEngine(0)(0) diye bir kod geldiğini görebilirsiniz. Bu “Workspaces(0).Databases(0)” yazmanın kısa yoludur ama yukarıda belirttiğim gibi biz ikisini de kullanmayacağız, zaten hep default Workspace(yani 0 indeksli) üzerinde çalışıyor olacağız.

Bundan sonra gelen kod ise şöyle bir şey olacaktır.

Dim db As DAO.Database

Set db = DAO.OpenDatabase(dbİsmi)

Buradaki metodun **OpenDatabase** olması sizi yanıltmasın, gerçekte bir Access penceresi açılmamaktadır. İlgili database’in bir nevi hafızada açıldığını düşünebilirsiniz.

Bundan sonrasında bu nesneyle ilgili olarak başka bir işimiz olmayacak. Aslında Database nesnesinin **CreateTableDef**, **CreateQueryDef** gibi tablo ve sorgu yaratmaya yarayan metodları var ama ben bunların Excel VBA içinden kullanılması gerektiğini düşünmüyorum. Bizim işimiz daha çok Accesten data okumak ve gerekirse tablolarda güncelleme yapmak, kayıt eklemek, silmek olacaktır. İhtiyacımız olan tablo ve sorguları zaten Access üzerinde yaparız diye düşünüyorum. O yüzden bu tür metotlara değinmeyeceğim. Ender de olarak ihtiyacınız olursa bunlarla ilgili makaleleri Google’da kolaylıkla bulabilirsiniz. (Belki Access VBA ile ilgili bir sayfada bu konuda örnekler yapmayı düşünebilirim. Access VBA’i, Excel VBA kadar sık kullanmasak da zaman zaman oldukça faydasını görmekteyim.)

Veritabanıyla işimiz bitince **Close** metodunu kullanarak bağlantıyı kapatırız ve son olarak Nothing ataması ile belleği boşaltırız. Özetle;

Dim db As DAO.Database

Set db = DAO.OpenDatabase("………..accdb")

'diğer kodlar

db.Close

Set db=Nothing

##### TableDef

DAO’da tablolarla ilgili iki nesne bulunur; **TableDef(s)** ve **Recordset**.  
TableDef(s) tablolar hakkında metadata sunar. Alanlar, indexler, tablonun adı v.s gibi işlemler için kullanılır. Tabloların yaratılması ve silinmesi de bununla yapılır. İçindeki dataya erişim ise RecordSet ile yapılır. Ancak yukarda belirttiğim gibi biz tabloyla ilgili genel işlere burada girmeyeceğiz. Bu konuda bilgi lazım olursa yine bir google search yapabilirsiniz.

Keza, Sorgu yaratma gibi sorgu işlemlerinin nesnesi olan QueryDef(s) ile de çok bir işimiz olmayacak. Bununla beraber bir sorgunun içini okumak istersek yine RecordSet nesnesini kullanırız.

#### Recordset nesnesi

DAO’da en çok bu nesneyle haşır neşir olacağız. Çünkü data okuma ve manipülasyonu bu nesne ile yapılır. O yüzden bunu "Nesneler" başlığı altında incelemek yerine ayrı bir başlık altında incelemenin daha doğru olduğunu düşündüm.

Recordset nesnesi bize adından anlaşılacağı üzere belirli bir kayıt seti verir. Bu tüm bir tablo olabileceği gibi çeşitli filtreler uygulanmış bir sorgu sonucu da olabilir.

##### Tanımlama ve Yaratma

RecordSet'i tanımlama klasik değişken tanımıyla yapılır ancak ataması yapılırken New kelimesi kullanılmaz, zira bunu başka bir objenin(genelde DB objesinin) bir metodundan dönen değerle elde edeceğiz.

4 çeşit yaratma şekli vardır: DB’den, table'dan, query'den ve başka bir recordsetten. Önce genel syntax'a sonra parametrelere bakalım.

**Syntax: Object.OpenRecordset(Name, [Type], [Options], [LockEdit]).**

**Name** olarak tablo adı sorgu adı girilebileceği gibi SQL de girilebilir.

Dim rs As DAO.Recordset

Set rs = dbobj.OpenRecordset(Type, Options)

Set rs = TableDefObject.OpenRecordset(Type, Options)

Set rs = QueryDef.OpenRecordset

Set rs = RecordsetObject.OpenRecordset(Type, Options) 'varolan rs’de ilave filtre için

Type'a az sonra detaylı bakacağız. Son iki parametreyi ise neredeyse hiç kullanmayacağız. Bunların bir kısmının geriye dönük uyumluluk içeren parametreler olup bir kısmı ise küçük uygulamalarda çok kullanılmayan özelliklerdir, daha büyük uygulamalar için zaten Access yerine diğer Veritabanı uygulamaları kullanılmalıdır. Bir şekilde kullanım ihtiyacı olursa(Ör: aynı anda iki kişinin güncelleme yapması durumundaki davranışı belirlemek isterseniz) google'da araştırabilirsiniz. Ben hiç ihtiyaç duymadığım için araştırıp öğrenme zahmetine de girmedim açıkçası, o yüzden size de anlatamıyorum. Şimdi gelelim Type'a.

###### Type parametresi: DAO Recordset Tipleri

5 tür tip vardır.

**Table-type recordset(dbOpenTable)**: Bunlar, düz tablolara dayanır, yani bu tiple sorgular ve linkli tablolar okunmaz. Lokal tablolar için varsayılan tip budur. Sadece düz kayıt okuma veya kayıt ekleme/güncelleme yapacaksanız bunu kullanabilirsiniz ancak diğer işlemlerde bazı metod ve propertyler(AbsolutePosition, FindFirst v.s) çalışmadığı için bu işlemlere ihtiyacınız olduğunda bunu kullanamazsınız. Bu tiple yaratım yapıldığında, kayıt bulmak için **Seek** metodu kullanılabilir ama **Find** ve türevleri kullanılamaz. Seek kullanımı için indexlerden yararlanılır, bu yüzden Find metodundan daha hızlıdır.

**Dynaset-Type(dbOpenDynaset)**: Tablolara ek olarak sorgularda ve linkli tablolarda da kullanılır. Linkli tablolar için varsayılan tip budur. Kayıt bulmak için **Find** metodu kullanılırken **Seek** metodu bunda kullanılamaz.

**Snapshot-type(dbOpenSnapshot)**: Bir recordset elde edilmiş ve resmi çekilmiştir, bunun üzerinden kayıt okumak için bu tip kullanılır. Statik bir veri setine sahip olduğumuz için kayıtlarda güncelleme yapılamaz, yani read-only bir yöntemdir. **Find** metodunu destekler.

**Forward-only-type(dbOpenForwardOnly)**: Snapshota çok benzer, sadece ileri doğru okuma yapar.

**Dynamic-type(dbOpenDynamic):** DynaSet’e çok benzer. Farkı şu: O sırada başka kullanıcılar da recordsetiniz için temel aldığınız tabloda bir güncelleme/ekleme/silme işlemi yaptıysa bunlar da sizin recordsetinize anında yansır.

Görüldüğü üzere Seek metodunu sadece dbOpenTable tipinde açılmış recordsetlerde kullanabilir. Bu bağlamda örnek bir veritabanı erişim koduna bakalım.

Sub DAOOrnek()

Dim db As DAO.Database

Dim rs1 As DAO.Recordset, rs2 As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs1 = db.OpenRecordset("Data", dbOpenTable)

Set rs2 = rs1.OpenRecordset

End Sub

Tip belirtilmezse, default tipler baz alınır: "Name" olarak verilen kaynak, bir tablo ise dbOpenTable, linkli tablo ise dbOpenDynaset.

###### Hangi Tip ne zaman kullanılır?

Düz okuma, kayıt ekleme, güncelleme yapılacaksa ve/veya Seek ile hızlı arama yapma ihtiyacı varsa(indeks bulunmalı): **dbOpenTable**

Query ve linkli tablolarda yeni kayıt ekleme, güncelleme, silme + ayrıca Find türevlerini kullanma ihtiyacı varsa: **dbOpenDynaset veya dbOpenDynamic(farkları yukarıda var)**

Küçük veri kümelerinde sadece okuma(ileri geri fark etmez) yapacaksanız ve hızlı kayıt arama ihtiyacı yoksa(yani seek kullanmayacaksanız): **dbOpenSnapshot**

Küçük veri kümelerinde sadece ileri okuma yapacaksanız ve hızlı kayıt arama ihtiyacı yoksa(yani seek kullanmayacaksanız) **dbOpenForwardOnly**

##### Field nesnesi(Alanlar)

(Field nesnesi recordsetten bağımsız bir nesne olmakta birlikte hep onunla kullanıldığı için ayrı bir kısım açmak yerine Recordset kısmı altında ele almak istedim.) Önceki kısımlarda belirttiğim gibi, Field yaratma, bunlarda index belirleme gibi konulara girmiyoruz. Bu tür işlemleri VBA içinden yapmak yerine doğrudan Access'te yaparız, zira bunlar genelde dinamik olarak değiştirilecek şeyler değildir. Olur da ihtiyaç duyarsanız MSDN veya Google’da bunlara ulaşmak oldukça kolay. Bizim işimiz daha çok bu alanlara erişmek olacak. Erişimin de 3 yolu bulunmaktadır.

* Alan adı ile: rs.Fields("İsim")
* Alan adı kısayolu ile: rs![İsim]
* Alan item no ile: rs.Fields(1)

rs.Fields(0).**Name**: İlk kolonun adını yani kolon başlığını getirir.  
rs.Fields(0).**Value**: Bu ise ilk kolondaki geçerli kaydın(satırın) içeriğini döndürür. Bu arada Value özelliği default özellik olup yazılmasa da olur, ama biz iyi bir programlamacı olup yazıyoruz.  
rs.Fields.**Count**: İlgili kayıt setindeki alan sayısını verir.

##### Yeni kayıt, mevcut kaydı düzenleme ve kaydetme

**AddNew** metodu yeni boş bir satır ekler. Sonra Field nesnesi ele alınarak ilgili alan atamaları yapılır. Normalde Access’te manuel kayıt ekledikten veya değiştirdikten sonra onu kaydetme(save etme) diye birşey yoktur ancak VBA’de ismi “Save” olmasa bile bir kaydetme işlemi var, onu da **Update** metodu ile yapıyoruz

Bir recordsete yeni kayıt eklendiğinde geçerli(aktif) kayıt otomatikman yeni eklenen kayıt olmaz. Bunun için yeni kayda çapa atarak erişmemiz gerekir. Bunla ilgili detayları az aşağıda göreceğiz.

rs.AddNew

'ilgili alan atamaları yapılır

rs.Update 'Kayıt işlemi gerçekleşir

rs.Bookmark = rs.LastModified 'çapayı attık, şimdi yeni kayıt üzerinde çalışabiliriz

**Edit** metodu kaydı değiştirir(Access’teki Update sorgusunun muadilidir). Az önce belirttiğim gibi Update metodu yapılan değişikliklerin yansımasını sağlar, Update Query ile karıştırılmaması lazım. Yani isim benzerliği kafanızı karıştırmasın. Özetle; Access’teki Update Sorgu işlemi DAO'nun Edit metodu ile yapılırken, Access’te otomatik gerçekleşen Save işlemi DAO'nun Update metodu ile yapılır.

'yeni kayıt

rs.AddNew

rs.Fields(0) = "3333"

rs.Fields(1) = "Aksaray"

rs!Durum = "Açık" 'bu !'li yazım "."lı yazıma alternatif yöntemdir

rs.Fields("Bölge kodu") = 7030

rs.Update 'bunu demeden kayıt eklenmez

'Editleme

rs.Edit

rs.Fields("Bölge kodu") = 5555

rs.Update 'bunu demeden update etmez

##### Silme

Kayıtlarda dolaşırken cursor'ın bulunduğu kaydı silmek için **Delete** metodunu kullanıyoruz. Silme işlemini yapabilmek için recordseti Table veya Dynaset tipinde açmış olmak gerekiyor, aksi halde hata alınır.

Silme işlemi sonucunda sonraki kayıt otomatikman geçerli kayıt olmaz, o yüzden ilgili işlemden sonra MoveNext yapmanız gerekir.

Aşağıdaki örnekte Durum kodu 0 olan kayıtlar siliniyor.

'ön tanımlar

If Not (rs.EOF And rs.BOF) Then

Do While Not rs.EOF

Durum = rs.Fields(3).Value

If Durum = 0 Then

rs.Delete

End If

rs.MoveNext

Loop

End If

Silme işlemini, belli kriterleri sağlayan kayıtlar için yapacaksanız ben SQL metni çalıştırmanızı(SQL biliyorsanız tabi) tavsiye ederim. (Bunun detaylarını ileride göreceğiz.)

Mesela yukardaki örnekte kayıtları silmek için şu kodu çalıştırmak bana daha pratik geliyor.

'ön tanımlar

db.Execute "Delete from tabloadı where Durum=0"

##### Kayıtlarda dolaşma

###### Move metodu ile belli satırlardaki kayıtlara konumlanma

**Move** metodu, belirli bir satır numarasıyla kullanılabileceği gibi Move’un türevleri şeklinde de kullanılabilir.

Rs.Move 10 '10 kayıt aşağı konumlanır, 10.kayda değil(negatif olursa geriye doğru hareket)

Rs.Move 0 'olduğu yerde kalır

Rs.MoveFirst 'ilk kayda konumlanır

Rs.MoveLast 'son kayda konumlanır

Rs.MoveNext 'bir sonraki kayda konumlanır. Özellikle döngülerde satır satır ilerlerken kullanılır.

Rs.MovePrevious 'bir önceki kayda konumlanır

Bunları kullanırken BOF ve EOF ile birlikte kullanımı tavsiye edilir.

###### BOF & EOF

**BOF**, ilk kayıttan önceki bir pozisyonda olup olmadığınızı, **EOF** da son kayıttan sonraki bir pozisyonda olup olmadığınızı gösterir. Kullanım amacı da tabloda hareket ederken tablonun sınırları içinde kalıp kalmadığınızı görmektir.

İkisi de True olursa geçerli kayıt yok demektir. O yüzden bir recordset içinde kayıt olup olmadığını her ikisinin de False olması veya Not True olması şeklinde aşağıdaki gibi test ederiz.

If Not (rs.EOF And rs.BOF) Then

'kodlar

End If

If crst.EOF=False And rst.BOF=False Then

'kodlar

End If

###### RecordCount

BOF ve EOF'un amacı, RecordSet içinde kayıt olup olmadığını anlamaktan ziyade(dolaylı olarak bu amaca da hizmet edebilirler), hareket sonrasında tablonun dışına çıkıp çıkmadığımızdır. Recordsette kayıt olup olmadığını görmek için **RecordCount** özelliğini kullanıyoruz.

Dikkat: Recordsetimizi TableType tipinde açtıysak, RecordCount sorunsuz çalışır. Ancak DynasetType veya diğer tiplerde açtıysak RecordCount property’si o ana kadar erişilen kayıtların sayısını getirir; ve bu tipte açılan recordsetlerde ilk kayda gelindiğinde VBA kodu okunmaya devam eder. Bu yüzden recordseti açtıktan hemen sonra kayıt sayısını elde etmeye çalışırsak **sonuç hep 1 döner**. Bunun için **Dynaset** tipinde(ve diğerlerinde) açıldığında **önce** **MoveLast** ile son kayda konumlanmalı ondan sonra RecordCount'ı elde etmeye çalışmalıyız. Bununla birlikte büyük tablolarda bu yöntem çok vakit alan bir iş olabilir, o yüzden dikkatli kullanılmalıdır.

Bu noktada ilk önerim, eğer başka nedenlerle gerekli değilse tabloyu DynasetType tipinde(veya diğer tiplerde) değil TableType tipinde açın. Diyelim ki DynasetType tipinde açtık, o zaman ikinci önerim de şudur: Eğer amacınız gerçekten kayıt sayısını elde etmekse MoveLast ise son kayda gidin(büyük tablolarda performans sorunu yaşatabilir) ama amacınız “içerde kayıt varmı yok mu” diye bakmaksa sonucun 1 dönmesi yeterlidir, MoveLast’a gerek yoktur, bu yüzden sade bir “RecordCount >0 mı?” kontrolü yaparsınız, o kadar.

###### BookMark, LastModified

Daha sonra dönmek üzere geride bıraktığınız bir kayda bookmark aracılığı ile ulaşabilirsiniz. Hem okunur hem yazılır bir özelliktir.  
Geçerli kaydı bir Bookmark olarak atamak için, önceden kaydedilmiş bir bookmark kullanılabileceği gibi **LastModified** özelliği ile en son değiştirilmiş/eklenmiş kayıt da atanabilir. Aşağıdaki örnekte **AbsolutePosition** özelliği de kullanılmış olup kaydın o anki satır numarasını verir. AbsolutePosition, dbOpenTable tipinde çalışmaz, dbOpenDynaset olmalı.

Sub dao\_bookmark()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase("….\daodeneme.accdb")

Set rs = db.OpenRecordset("şubeler", dbOpenDynaset)

Debug.Print rs.AbsolutePosition

rs.MoveNext

Debug.Print rs.AbsolutePosition

rs.MoveNext

Debug.Print rs.AbsolutePosition

rs.MoveNext

Debug.Print rs.AbsolutePosition

x = rs.Bookmark 'çapa atıyoruz

Debug.Print rs.AbsolutePosition

rs.Move 500 'çeşitli işlemler sonucunda şuan 500 kayıt aşağı geldik diyelim

Debug.Print rs.AbsolutePosition

rs.Bookmark = x 'tekrar çapamıza dönüyoruz

Debug.Print rs.AbsolutePosition

'son değişen kayıt

rs.Move 80

rs.Edit

rs.Fields("Bölge kodu") = 3333

rs.Update

Debug.Print rs.AbsolutePosition

rs.Move 200

Debug.Print rs.AbsolutePosition

rs.Bookmark = rs.LastModified

Debug.Print rs.AbsolutePosition

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

###### Döngüsel Örnek

Gerçek dünyada kayıtlarda döngüsel olarak dolaşmak pek daha olasıdır. O yüzden şimdi bir de döngüsel örnek yapalım.

Sub dao\_tablolardadolas()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Dim tdf As DAO.TableDef

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data")

'başlangıçta kaç kayıt var bakalım

Debug.Print rs.RecordCount

'önce geçici kayıt ekleyelim

For i = 1 To 3

rs.AddNew

rs.Fields(0) = "deneme bölge" & i

rs.Fields(1) = "deneme şube" & i

rs.Fields(2) = "deneme ürün" & i

rs.Fields(3) = 3 - i

rs.Update

Next i

Debug.Print rs.RecordCount

If Not (rs.EOF And rs.BOF) Then

Do While Not rs.EOF

Durum = rs.Fields(3).Value

If Durum = 0 Then

rs.Delete 'Durum=0 olan kayıtlar silinir

End If

rs.MoveNext

Loop

End If

Debug.Print rs.RecordCount

End Sub

###### Seek ve Findek ile kayıt arama

DAO, kayıtları bulmanın iki yolunu bize sunuyor. **Seek** ve **Find** türevleri. Bunların kullanımı, Recordset oluşturulurken kullanılan tipe göre değişmektedir. Öncelikle şu ayrımı iyi yapmak gerekiyor. Ne zaman Recordset, ne zaman SQL, ne zaman diğer metodlar?

* Eğer yapabiliyorsak recordsetimizi direkt aradığımız kayıt üzerine oluşturmalıyız. Yani bir SQL ile tek satır döndüren bir recordset tanımlayabiliriz.
* Eğer daha geniş kümeli bir recordsetimiz olacak ve bunu çeşitli aşamalarda farklı şekillerde filtrelemeye/araştırmaya tabi tutacaksak
  + Recordsetin recordsetini yapabileceğimiz gibi
  + Aşağıdaki diğer arama yöntemlerini kullanabiliriz
    - Seek
    - Find türevleri
    - SQL

###### Seek Metodu

Recordset tipi olarak **sadece TableType** seçildiyse kullanılabilir. Çünkü tablodaki indekslere ihtiyaç duyar ve doğal olarak da kolonlardan en az birinde indeks olması gerekir. O yüzden Seek metodu uygulanmadan önce Indeks property’si belirtilir. İndeksli arama da en hızlı yöntem olduğu için Find’a göre daha hızlı bir yöntemdir. Tabiki Access’te ilgili tabloda ilgili kolonda indeks olduğundan emin olmalısınız. Bu arada PrimaryKey dışındaki kolonlarda indeks adı genelde kolonadı ile aynı olurken, PrimaryKey olan bir kolonda indeks adı "PrimaryKey" olur, o yüzden indeks olarak da bu şekilde belirtmelisiniz.

Parametre olarak "=","<",">" gibi karşılaştırma işaretleri ve aranan değer girilir.

Aradığımız değere konumlanma girişiminin başarılı olup olmadığını **NoMatch** özelliği ile test ederiz. Eğer True dönerse aranan kriterlere göre uygun kayıt bulunamamıştır demektir.

Sub dao\_seek()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("Data", dbOpenTable)

rs.Index = "Şube Adı" 'index belirtiyoruz

rs.Seek "=", "Şube115"

If rs.NoMatch Then 'konumlanma başaraılı mı diye kontrol ediyoruz

Debug.Print "Kayıt bulunamadı"

Else

Debug.Print rs.Fields("Aylık Gerç")

End If

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

###### Find türevleri

Move’un aksine Find’ın solo halde bir versiyonu yoktur. 4 çeşidi vardır. **FindFirst**, **FindPrevious**, **FindNext**, and **FindLast**. Find kullanımı için Recordsetimizin Tabletype dışındaki bir tiple tanımlanması gerekir. Genellikle Dynaset yeterlidir.

Indeks kullanmak zorunda değildir, bu yüzden indekssiz bir kolonda arama yaptığınızda Seek metoduna göre çok daha yavaş çalışır.  
Find metodlarıyla "?" ve "\*" gibi joker karakterleri kullanabiliyoruz.

Sub dao\_find()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenDynaset)

rs.FindFirst "[Şube Adı] LIKE '\*deneme\*'"

If rs.NoMatch Then

Debug.Print "Kayıt bulunamadı"

Else

Debug.Print rs.Fields("Şube Adı")

End If

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

Ben bu metodların bir karşılaştırmasını yaptım. Buna göre 5 milyon kayıtlık bir tabloda;

* indekssiz bir kolonda FindFirst araması yapmak 127 sn,
* indeksli kolonda FindFirst yapmak 1,5 sn,
* (İndeksli kolonda) Seek ile arama yapmak ise 0,04 sn sürüyor.

Gördüğünüz gibi indeksli olması her halükarda hızı inanılmaz arttırıyor, ancak Seek’in Find’a göre üstünlüğü ise aşikar. Mutlak değer olarak fazla bir fark olmasa da oransal fark çok büyük.

###### SQL

Aranan değeri bulmada bir diğer yöntem, Recordseti çekerken tek sonuç döndürecek bir SQL çalıştırmaktır. Veya çoklu sonuç dönecekse de MoveFirst diyerek ilk kaydın sonucunu almak olacaktır. Veya zaten çok sonuç arıyorsak da çoklu sonuç dönen bir SQL hazırlanır.

#### Filter ile Recordseti filtreleme

Her ne kadar filtreleme ve sıralama işlemlerini SQL metni içinde yapmamızda fayda olsa da bazen recordset üzerinden de bunları yapmamız gerekebilir. Biz burada sadece Filtreleme işlemine bakacağız.

Öncelikle belirtmek isterim ki **Filter** işlemini dbOpenTable tipinde açılmış bir recodsette yapamıyoruz. Diğer tiplerde açılmış olması gerekir.

Bu işlem için tahmin edileceği üzere Filter property'si kullanılır. Bunun içine SQL metninde yazacağımız gibi bir kriter yazarız. Eğer ki kolon adımız 1'den çok kelimeden oluşuyorsa bunları [] içine yazarız.

Aşağıda ADO kısmında göreceksiniz, orda da Filter işlemi yapılıyor ancak DAO'da ADO'dan farklı olarak uygulanış şekli biraz farklıdır. DAO'da iki farklı recordsetimizin olması gerekir. İlk recordsetin Filter property'sine kriterler girilir ve ikinci recordset bu ilk recordsetten filtrelenmiş şekilde elde edilir. Hemen örneğimize bakalım.

Sub dao\_filter()

Dim db As dao.Database

Dim rs As dao.Recordset

Dim rsFilter As dao.Recordset

Set db = dao.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenDynaset)

rs.Filter = "[Ürün Adı]='Ürün1'"

Set rsFilter = rs.OpenRecordset

rs.MoveLast

Debug.Print rs.RecordCount '1868

rsFilter.MoveLast

Debug.Print rsFilter.RecordCount '466

End Sub

NOT: Tarihsel alanları mutlaka Amerikan formatında(ay-gün-yıl) girilmesi gerekiyor.

#### Execute ile Sorgu/SQL çalıştırma

**Execute** metodu ile doğrudan basit bir SQL veya var olan bir eylem sorgusu(Append,Delete,Update) çalıştırılabilir. Hem Database nesnesi hem de QueryDef nesnesi için kullanılabilen bir metoddur.

Sub Dao\_Execute()

Dim db As DAO.Database

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

db.Execute "Query1" 'veya açık bir şekilde SQL metni

' veya db.QueryDefs(0).Execute

db.Close

Set db = Nothing

End Sub

Execute’la birlikte kullanılan parametreler var. Biz bunlara burada girmeyeceğiz, detay bilgi edinmek istiyorsanız MSDN’ye(<https://msdn.microsoft.com/en-us/library/office/ff197654.aspx>) bakabilirsiniz.

Ayrıca yukarıda belirttiğimiz gibi, Filter işlemlerini mümkün olduğunca SQL içinde çalıştırmak daha hızlı sonuç almamızı sağlar, özellikle büyük veri kümelerinde. Eğer ki bu elde ettiğimiz veri setinde, farklı case'lere göre dinamik filtrelemeler yapmak gerekirse o zaman Filter'ı devreye sokabiliriz.

#### Datayı Excel’e almak(Import işlemi)

DAO kullanımında en sık yapacağımız işlem, eriştiğimiz datayı Excel içine almak olacaktır. Bunun için de birkaç yöntem bulunuyor.

##### 1.yöntem: CopyFromRecordset metodu

Bu yöntem en hızlı yöntemdir. Çoğu durumda bu yeterli olmaktadır.

**Syntax:** Range.CopyFromRecordset(Data, [MaxRows], [MaxColumns])

Burda önemli olan husus, başlıkların gelmiyor oluşudur. Başlık için döngüsel bir kod yazılır. MaxRows ve MaxColumns ile çekilen kayıt sayısı sınırlandırılabilir.

Sub import1()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenTable)

If Not (rs.EOF And rs.BOF) Then

'başlık yazma kısmı

For i = 0 To rs.Fields.Count - 1

ActiveCell.Offset(0, i).Value = rs.Fields(i).Name

Next i

'şimdi de data yazılır

ActiveCell.Offset(1, 0).Select

ActiveCell.CopyFromRecordset rs

End If

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

##### 2.yöntem: Diziye atayıp diziyi yazdırmak

Bu yöntem 2. en hızlı yöntemdir. Eğer diziye atadıktan sonra diziyi başka yerde de kullanacaksanız veya dizi elemanları üzerinde işlem yaptıktan sonra Excel'e aktaracaksanız bu yöntemi kullanabilirsiniz.

Sub import2()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Dim hucreler() As Variant

Dim alan As Range

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenTable)

ReDim hucreler(rs.RecordCount - 1, rs.Fields.Count - 1)

'başlık yazma kısmı

For i = 0 To rs.Fields.Count - 1

ActiveCell.Offset(0, i).Value = rs.Fields(i).Name

Next i

'diziyi dolduralım

If Not (rs.EOF And rs.BOF) Then

Do While Not rs.EOF

For i = 0 To rs.Fields.Count - 1

hucreler(j, i) = rs.Fields(i).Value

Next i

j = j + 1

rs.MoveNext

Loop

End If

'Excel’e yazalım

Set alan = Range("A2").Resize(UBound(hucreler, 1), UBound(hucreler, 2) + 1)

alan.Value = hucreler

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

##### 3.yöntem: GetRows metodu ile dizi elde ederek

**GetRows** ile iki boyutlu bir dizi elde ederiz. Boyutlardan ilki kolonu, ikincisi satır numarasını ifade eder. Tek kolonluk bir veri çekseniz bile 2 boyutlu bir diziniz olur.

Rs.GetRows tüm recordseti döndürürken Rs.GetRows(x) ilk x kaydı diziye aktarır. Mesela ilgili veri kümesinden sadece örnek bir küme almak istiyorsanız 100 satırlık data çekebilirsiniz. GetRows, Move metodu gibi davranır, yani parametre olarak 100 derseniz 100. kayda gelir.

Sub import3()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Dim dizi As Variant

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenTable)

If Not (rs.EOF And rs.BOF) Then

dizi = rs.GetRows(10) 'ilk 10 kayıt

Debug.Print dizi(0, 0) 'ilk kolon ilk satır

Debug.Print dizi(rs.Fields.Count - 1, UBound(dizi, 2)) 'son kolon son satır

End If

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

Yukarıda belirttiğim gibi, tek kolon çeken bir SQL'iniz bile olsa iki boyutlu bir dizi elde edersiniz. Diyelim ki böyle bir veri çektiniz ve ihtiyacınız da, bu veri setini aralarında ";" işareti olacak şekilde birleştirmek. Bunu döngüsel olarak dolaşıp yapabileceğiniz gibi WorksheetFunction.Index fonksiyonundan da yararlanabilirsiniz.

Örneğin, diyelim ki çektiğiniz veri seti bazı müşteri numaraları olsun. Bunları aralarında ";" olacak şekilde birleştirmek için şöyle bir kod yazabiliriz:

'önceki kodlar

rs.Open strSQL,con,adOpenStatic,adLockOptimistic

müşteriler=rs.GetRows

müşteriStr=Join(WorksheetFunction.Index(müşteriler,0),";")

'sonraki kodlar

##### 4.yöntem: Range’e döngüsel şekilde yazdırma

En yavaş yöntemdir. Dizilerle çalışmayı bilmiyorsanız veya hücreler üzerinde başka işlemler de yapacaksanız bu yöntemi kullanabilirsiniz ancak büyük data kümelerinde tavsiye edilmez.

Sub import4()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenTable)

'başlık yaz

For i = 0 To rs.Fields.Count - 1

ActiveCell.Offset(0, i).Value = rs.Fields(i).Name

Next i

If Not (rs.EOF And rs.BOF) Then

ActiveCell.Offset(1, 0).Select

k = ActiveCell.Row

Do While Not rs.EOF

For i = 0 To rs.Fields.Count - 1

Cells(k, i + 1).Value = rs.Fields(i).Value

Next i

k = k + 1

rs.MoveNext

Loop

End If

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

#### Excel’deki datayı Access’e atma(Export işlemi)

Bunda iki yöntem uygulanabilir.

##### 1.Yöntem: Döngü içinde Addnew+update

Sub export1()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("exporttable", dbOpenTable)

For k = 2 To [a1].End(xlDown).Row

rs.AddNew

For i = 1 To [a1].End(xlToRight).Column

rs.Fields(i - 1) = Cells(k, i).Value

Next i

rs.Update

Next k

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

End Sub

##### 2)Access sorgusunu çalıştırmak.

Bu yöntem DAO'ya ait bir örnek değildir aslında. Burada Access nesne modeline girmiş oluyoruz. Çok basit bir mantığı var. Aşağıdaki kodda yorumlara bakın lütfen.

Bu arada bu yöntemde mevcut Excel dosyasını Access'e linklemiş olmak gerekir.

Sub export2()

Set accessApp = GetObject(adres+"VBAdb.accdb", "Access.Application") 'İlgili access dosyasını bir değişkene atıyoruz, ama bunu uygulama olarak atıyoruz

With accessApp

.Application.Visible = False 'Arka planda çalışsın istiyoruz

.DoCmd.Openquery "AppendQuery1" 'DoCmd metodu ile kayıtlı bir sorguyu çalıştırıyoruz

.Run "Modül1" 'Run metodu ile Access VBA ile yazımış bir kodu çalıştırıyoruz

End With

End Sub

Access sorgusunu çalıştırmanın bir  yöntemi de aslında yukarıda gördüğümüz Execute metodudur.

Sub export3()

Dim db As dao.Database

Dim rs As dao.Recordset

Set db = dao.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("data", dbOpenDynaset) 'açılış tipi dynaset

db.Execute "srg\_Excel’dekiniburayaappend"

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

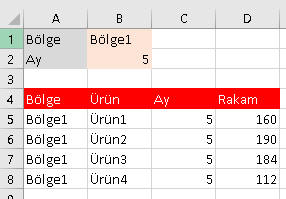
End Sub

#### Örnek Çalışma - İnteraktif Veri Çekme Formatı

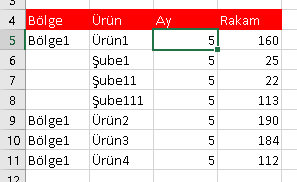
Şimdi DAO ile interaktif bir şekilde, yani parametreleri dinamik şekilde değiştirerek nasıl veri çekilir, buna ait bir örnek yapmak istiyorum.

Bu örnekte, bir tablodan bir bölgenin belli bir aydaki çeşitli ürünlerine ait rakamlarını çekiyoruz. Ürünlerden birine çift tıkladığımızda da şube detayı gösteriliyor. Ürüne tekrar çift tıklandığında şubeler kaybolup tekrar ilk haline dönüyor.

Bir seçim sonunda görünen tablo aşağıdaki gibidir.



B5 hücresindeki Ürün1'e çift tıklanınca tablo aşağıdaki şekle dönüşüyor,



Kod bloğumuz aşağıda duruyor. Buna göre;

* Önce global değişkenlerimizi(biri sabit) yaratıyoruz
* Sonra Bölge veya Ay bilgileri değiştiğinde(Bunlar B1 ve B2 hücrelerinde bulunuyor) Change event'ini tetikliyoruz.
* Sonra da çift tıklama eventini handle ediyoruz.

Her iki event prosedürünü de başına breakpoint koyup F8 ile ilerleyerek kodu incelemenizi tavsiye ederim.

Dim db As dao.Database

Dim rs As dao.Recordset

Const adres As String = "C: \inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar"

'--------Bölge ve Ay bilgileri değiştiğinde tetiklenecek prosedür

Private Sub Worksheet\_Change(ByVal Target As Range)

'yanlışlıkla başka bir hücreye çift tıklarsa onun içine girmiş olur ve burası tetiklenir

If Target.Row = 4 Then Exit Sub 'başlığa çift tıklanırsa

If IsEmpty(Target) Then Exit Sub 'bir de herhangi boş bir hücreye çift tıklanırsa

Application.EnableEvents = False

Range("a4").CurrentRegion.Offset(1).Clear 'önce temizlik

If Not Intersect(Target, Range("B1: B2")) Is Nothing Then

[a5].Select

Set db = dao.OpenDatabase(adres + "\BölgeŞubeRakamları.accdb")

mySql = "select \* from bölgerakam where Bölge='" & Range("bölge") & "' and Ay=" & Range("ayno")

Set rs = db.OpenRecordset(mySql)

ActiveCell.CopyFromRecordset rs

End If

Application.EnableEvents = True

End Sub

'----Ürün bilgisine çift tıklandığında tetiklenip şube detayını gösterecek olan prosedür

Private Sub Worksheet\_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)

If IsEmpty(Target.Offset(1, -1)) And Not IsEmpty(Target.Offset(1, 0)) And Not IsEmpty(Target.Offset(0, -1)) Then

Application.EnableEvents = False

Do

ActiveCell.Offset(1, 0).EntireRow.Delete

Loop Until Not IsEmpty(ActiveCell.Offset(1, -1))

Application.EnableEvents = True

Target.Offset(0, 1).Select

Exit Sub

End If

If Not Intersect(Target, Range([b5], [b5].End(xlDown))) Is Nothing And Not IsEmpty(Target.Offset(0, -1)) Then

Application.EnableEvents = False

Set db = dao.OpenDatabase(adres + "\BölgeŞubeRakamları.accdb")

mySql = "select şube,ay,rakam from şuberakam where Bölge='" & Range("bölge") & "' and Ay=" & Range("ayno") & " and ürün = '" & Target.Value2 & "'"

Set rs = db.OpenRecordset(mySql)

'Debug.Print rs.Type

rs.MoveLast 'bir üst satırdaki ' işaretini kaldırıp F8 ile ilerlersek

'görürüz ki recordsetin tipi dynaset, o yüzden recordcoutn ele etmek için en sona konumlanmalıyız

şubeadet = rs.RecordCount

For i = 1 To şubeadet

Target.Offset(1, 0).EntireRow.Insert Shift: =xlDown, CopyOrigin: =xlFormatFromLeftOrAbove

Next i

rs.MoveFirst

Target.Offset(1, 0).CopyFromRecordset rs

Target.Offset(0, 1).Select

Application.EnableEvents = True

End If

End Sub

### ADO DETAY

#### Giriş

Yukarıda belirttiğim gibi Access dışındaki yeni çalışmalarınızda ADO’yu kullanmanızı öneriyorum(hatta isterseniz Access’te bile ADO’yu kullanabilirsiniz, ancak DAO Access’e özgü olduğu için daha hızlıdır ve esnektir. Ben iki şeyi bilmekle uğraşmayayım sadece tek şeyi bileyim diyorsanız ADO size yeter, sadece ADO’yu öğrenin)

##### Referans ekleme

ADO’yu çalışmalarınızda kullanabilmek için buna ait Library’nin reference olarak eklenmesi gerekir. İşletim sisteminin versiyonuna göre uygun library seçimi yapılır, genelde en yüksek versiyon seçilir. Windows 7 ve sonrası için 6.1 gibi. Ancak yapacağınız çalışmayı başka kişiler de kullanacaksa ve onlar sizin işletim sisteminden daha aşağı seviyelerde bir işletim sistemi kullanıyorlarsa siz de ya daha düşük versiyonu seçmeli veya Late Binding yöntemini kullanmalısınız.

NOT: DAO’daki TableDef ve QueryDef nesneleriyle yapılan DB seviyesindeki işlemleri ADO ile yapamıyoruz. Bunun yerine ADOX library’si eklenmelidir. Ama bu tür işlemler zaten konumuzun dışında olduğu için burada buna hiç girilmeyecektir.

##### Yakından bakış

Genel mantık DAO’ya benzer. Bundaki süreç ise şöyledir:

* Connection yaratılır
* Recordset yaratılır
* Kayıtlara erişilip işlem yapılır
* Recordset ve Connection kapatılır

Gördüğünüz gibi burada **Database** nesnesi yok, onun yerine **Connection** nesnesini kullanıyoruz.

En başta belirttiğim gibi ADO, MS’un en güncel data erişim teknolojisidir ancak ADO bunu tek başına yapmaz, OLE DB Provider denen bir aracı teknoloji ile yapar. Genelde her data kaynağı için ayrı bir OLE DB sağlayıcısı vardır, ancak MS’taki abiler ODBC bağlantı türü(Genel amaçlı provider) için de OLE DB sağlayıcısı yapmışlar, böylece ADO ile her tür veri kaynağına bağlanılabilmektedir.

Datayı manipüle etmede kullanılan ve arka plandaki esas yazılıma **DB Engine(VT motoru)** deniyor. Access, DB Engine olarak **Jet** (Joint Engine Technology) kullanır. 2003 öncesi versiyonlarda bu, Jet 4.0 OLE DB provider iken 2007 sonrasında (.accdb database), "Microsoft.ACE.OLEDB.12.0" oldu, **ACE** (Access Connectivity Engine).

Bu arada sadece Access’e değil, daha önce söylediğimiz gibi bir metin dosyasına hatta bir Excel dosyasına bile ADO ile bağlanabiliriz.

Daha detaylı bilgiye MSDN’den(<https://msdn.microsoft.com/en-us/library/office/jj249129.aspx>) ulaşabilirsiniz.

#### Nesneler

##### ADO Nesne modeli

ADO’daki nesne sayısının daha az olduğunu ama metod sayısının çok olduğunu söylemiştik. Temel nesnemiz **Connection**’dır. Bunun Open ve Close metodları vardır.

**Recordset** nesnesi ile DAO'da olduğu gibi dataya erişiriz ve gerektiğinde onu işleriz (Update,Delete..)

**Record** nesnesi, Recordsetteki bir satır kaydı gösterir.

**Fields** Collection’ı tablodaki tüm kolonları gösterirken, **Field** nesnesi, bu kolonlardan herhangi birini ifade eder.

DAO konusunu anlatırken de bahsetmiştik. Bu objeleri nesneleri tanımlarken başlarına library’sini(DAO için DAO, ADO için ADODB) koymakta fayda var, özellikle ilgili VBA projesi içinde hem DAO hem ADO refere edildiyse. Bazıları için nesne isimleri ortak olmamakla ve bir karışıklığa neden olmamakla birlikte bu alışkanlık iyi bir alışkanlıktır.(Sadece ADO’cu olmaya karar verdiyseniz gerek yok tabi)

DAO ile ADO nesneleri arasındaki farkları ve benzerlikleri <http://www.databasejournal.com/features/mssql/article.php/1490571/From-DAO-to-ADO.htm> sayfasında bulabilirsiniz.

Bu arada önemli bir husus da şu: Nesnelerin isimleri aynı olabilir ama metod ve propertyler farklıdır. Yani aynılar diye kullanım şekilleri de aynı olmak zorunda değil.

Şimdi bu nesnelere yakından bakalım.

##### Connection nesnesi

###### Tanımlama ve Yaratım

DAO’da Database nesnesini yaratırken New keywordunu kullanmıyorduk. Çünkü DAO’da nesnelere Set atamasını yaparken bir fonksiyon(OpenDataBase) ile yaratıyorduk. Ancak ADO’da ise yeni(New) Connection nesnesini yaratıp, bu nesnenin kendi Open metodu ile bağlantıyı açıyoruz.

'Earlybinding

Dim con As New ADODB.connection 'tek satır

'Veya  iki satırda

Dim con As ADODB.connection

Set con = New ADODB.connection

'Late binding

Dim con As Object

Set con = CreateObject("ADODB.Connection")

Connection nesnesini **Open** metodu ile başlatırız(açarız). DAO’dan farklı olarak sadece Access’e değil başka veritabanlarına da ulaşabildiğimizi söylemiştik. Bunun için **Connection String** denen bir ifadeye ihtiyaç duyarız. Syntaxı aşağıdaki gibidir.

**ConnectionObject.Open ConnectionString, UserID, Password, Options.**

Genelde ilk parametre yani ConnectionString yeterlidir. Bunun da Provider ve Datasource değerlerini(ODBC için Driver ve DBQ değerlerini) girmek yeterlidir.

Şimdi çeşitli connection yaratma alternatiflerine bakalım. Bunlarda Access veritabanına OLEDB ile bağlanacağız.

Sub adoconyarat()

Dim con As ADODB.connection

Dim strDB As String

Set con = New ADODB.connection

strDB = adres + "VBAdb.accdb"

' ConnectionString propertysini belirterek

With con

.ConnectionString = "Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB 'oledb için

' odbc için:

'.ConnectionString = "Driver={Microsoft Access Driver (\*.mdb, \*.accdb)};Dbq=" & strDB

.Mode = adModeRead

.Open

End With

'....

con.Close

End Sub

veya connection string yerine provider ve datasource ayrı ayrı belirtilerek;

With con

.Provider = "Microsoft.ACE.OLEDB.12.0"

.Properties("Data Source ") = strDB

.Mode = adModeRead

.Open

End With

Aşağıda Access dışındaki diğer data kaynaklarına bağlanırken kullanılan ConnectionString’leri görebilirsiniz.  
   
Diğer bağlantı türleri için [http: //www.connectionstrings.com](http://www.connectionstrings.com) sitesine bakabilirsiniz, ancak biz aşağıda zaten birkaç türünü göreceğiz. Şimdi bunlara bakalım.

###### Excel dosyalarına ulaşmak

ADO ile başka Excel dosyalara da ulaşabilip veri alabilmekteyiz. Tabi isterseniz bu işlemi ilgili dosyayı yine makro ile açıp sonra copy paste yaptırıp kapatma veya bir sonraki sayfada göreceğimiz gibi refreshlenebilir bir Connection kurma yoluyla da yapabilirsiniz. ADO'nun farkı, ilgili sayfayı olduğu gibi almak yerine satırların/sütunların sayısını almak veya sadece belli bir hücreyi/satırı/kolonu almak için rahatlıkla kullanılabilmesindedir.

Bağlantıyı aşağıdaki gibi kuruyoruz. Bağlantı için OLEDB sağlayıcısını kullanırız. Connection string olarak dosya adresi yazıldıktan sonra Extended Properties özelliğine Excel’in versiyonu yazılır ve Xml ifadesi eklenir.

Recordseti açarken de veriyi hangi sayfadan alacağımızı SQL metni olarak yazarız. Kolon başlığına Where ifadesi ile filtre de koyabiliriz Ör: Where Bölge='Akdeniz'.

Sub ExcelADO()

Dim cn As New ADODB.Connection

Dim rs As New ADODB.Recordset

dbAdres = "C: \inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\pivotdata.xlsx"

constr = "Provider = Microsoft.ACE.OLEDB.12.0; Data source=" + dbAdres + ";Extended Properties='Excel 12.0';"

cn.ConnectionString = constr

cn.Open

'veya aşağıdaki gibi

'With cn

' .Provider = "Microsoft.ACE.OLEDB.12.0"

' .Properties("Data Source") = dbAdres

' .Properties("Extended Properties") = "Excel 12.0" 'burda ayrıca ' içine yazmaya gerek yok

' .Open

'End With

rs.Open "Select \* FROM [Sheet1$]", cn, adOpenStatic, adLockReadOnly, adCmdText

ActiveCell.CopyFromRecordset rs

rs.Close

cn.Close

End Sub

Kaynak alanımız belirli bir adres ise onu adresiyle birlikte yazarak "SELECT \* FROM [Sheet1$A1: C100]" şeklinde veya kaynağımız bir Table ise "SELECT \* FROM [Table1]" şeklinde belirtebiliriz.

###### Text dosyasına ulaşmak

Text dosyalarına bağlanırken hem OLEDB hem ODBC kullanabiliriz. Text dosyasının kendisini connection string içinde geçirmeyiz, bunun yerine ilgili dosyanın bulunduğu klasörü yazarız. Dosya adını recordset içinde belirtiriz.

**OLEDB**Provider=Microsoft.ACE.OLEDB.12.0;Data Source=c: \txtFilesFolder\;  
Extended Properties="text;HDR=Yes;FMT=Delimited";  
  
**ODBC**  
Driver={Microsoft Text Driver (\*.txt; \*.csv)};Dbq=c: \txtFilesFolder\;  
Extensions=asc,csv,tab,txt;

Önemli Not: ODBC bağlantılarında ilgili veri kümesinin ilk satırı her zaman başlık varsayılır, ve ikinci satırdan itibaren data okunur.

Şimdi text örneklerine bakalım:

Sub ado\_txt1\_odbc()

'odbc örneği

Dim conn As ADODB.Connection

Dim rs As ADODB.Recordset

Set conn = New ADODB.Connection

conn.Open "DRIVER={Microsoft Text Driver (\*.txt; \*.csv)};" & \_

"DBQ=" & adres & ";" & "Extended Properties=""text;FMT=Delimited""" 'ODBC'de her zaman ilk satır başlık kabul edildiği için HDR parametresi belirtmedik

Set rs = New ADODB.Recordset

rs.Open "select \* from [hatalog.txt]", conn, adOpenStatic, adLockReadOnly, adCmdText

Range("a1").CopyFromRecordset rs

rs.Close

conn.Close

Set rs = Nothing

Set conn = Nothing

End Sub

Şimdi de OLEDB ile

Sub ado\_txt2\_oledb()

'oledb örneği

Dim conn As ADODB.Connection

Dim rs As ADODB.Recordset

Set conn = New ADODB.Connection

conn.Provider = "Microsoft.ACE.OLEDB.12.0"

conn.ConnectionString = "Data Source=" & adres & ";" & "Extended Properties=""text;HDR=no;FMT=Delimited;"""

conn.Open

Set rs = New ADODB.Recordset

rs.Open "select \* from [hatalog.txt]", conn, adOpenStatic, adLockReadOnly, adCmdText

Range("a1").CopyFromRecordset rs

rs.Close

conn.Close

Set rs = Nothing

Set conn = Nothing

End Sub

Çok kullanılmayan FixedWidth tipli dosyalar ve Delimiter'ların farklı türlerinin nasıl kullanılacağı ile MSDN’den(<https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/scripting-articles/ms974559(v=msdn.10)>) bilgi edinebilirsiniz.

Bu arada, şunu belirtmeden de geçmeyelim: Text dosyalarını okumanın, tüm içeriğini elde etmenin ve içeriğini değiştirmenin başka yolları da var, bunun için de **Dosya okuma yazma işlemleri** adlı konuya bakabilirsiniz. Ancak oradaki yöntemler daha çok küçük datalar için anlamlı. Text dosyasını veritabanı amaçlı kullanmak için(yüzbinlerce satırdan bahsediyorum) yine ADO’yu tercih edin. Yukarıdaki linkteki yöntemleri ise daha ziyade kısa metin içeren dosyalarda kullanın.

##### Mode propertysi

Yukarıdaki örneklerde fark ettiyseniz Mode özelliği Connection henüz açılmamışken atandı, çünkü bu özellik sadece kapalı bağlantılarda atanabilir. Bunun alacağı değerler ve açıklamaları aşağıda verilmiştir.

**adModeUnknown**: Default budur. İzinler henüz set edilmemiştir ve tam karar verilemez. ADO, provider’a kendisi karar verecektir.

**adModeShareDenyNone:** Başkalarının da her türlü yetkiyle açmasına izin verir.

**adModeReadWrite**: Read/Write olarak açar. Yani hem okuma hem yazma yapılabilir.

**adModeShareDenyRead**: Sizde açıkken başkaları buradan okuma yapamaz.

**adModeRead**: Sadece okuma yapabilirsiniz, yazma yapamazsınız.

**adModeShareDenyWrite**: Sizde açıkken başkaları buraya yazma yapamaz.

**adModeWrite** : Sadece yazma yapabilirsiniz, okuma yapamazsınız.

**adModeShareExclusive**: Bağlantı sizde açıkken başkaları o an bağlanamaz.

Mesela sadece okuma amaçlı açılan aşağıdaki kodda AddNew satırında hata alırsınız.

Sub adomode()

Dim con As ADODB.Connection

Dim rs As ADODB.Recordset

Set con = New ADODB.Connection

With con

.ConnectionString = "Provider = Microsoft.ACE.OLEDB.12.0; data source=" & adres & "VBAdb.accdb"

.Mode = adModeRead 'Write modu açılırsa sorun yok

.Open

End With

Set rs = New ADODB.Recordset

rs.Open "data", con, adOpenDynamic, adLockOptimistic

rs.AddNew 'burada hata

rs.Fields(0)="x bölgesi"

rs.Fields(1)="y şubesi"

rs.Fields(2)="z ürünü"

rs.Fields(3)=100

rs.Update

End Sub

#### Recordset nesnesi

DAO’da olduğu gibi ADO’da da veritabanına bir kez eriştikten sonra artık kayıtlarda istediğimiz işlemleri yapabiliyoruz.

##### Tanımlama ve Yaratma

DAO’da Recordset nesnesini New keywordu olmadan yaratıyorduk. Set atamasını yaparken de başka bir nesne olan Database nesnesinin OpenRecordSet metodunu kullanıyorduk. ADO’da ise New keywordu kullanılarak(Dim satırında veya Set satırında) yeni bir RecordSet nesnesi yaratılır ve bu nesnenin kendi metodu olan Open Metodu kullanılır(İstisna: Connection veya command nesnelerinin Execute metodu ile recordset elde edeceksek New ifadesini kullanmayız). Aşağıda genel syntaxı bulunmaktadır.

**Recordset.Open Source, ActiveConnection, CursorType, LockType, Options**

Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

rs.Open "Select \* from Subeler", con

**Source,** bir SQL metni olabileceği gibi, tablo/sorgu adı da olabilir, bir Command nesnesi de. Genel olarak çekilecek kayıt miktarını minimize etmek iyi bir alışkanlıktır. O yüzden mümkünse tam bir tablo yerine bir SQL metni veya amaca hizmet eden bir sorgu(query) seçilmelidir. Hatta sadece test datası görmek istiyorsanız çekilen kayıt sayısını sınırlayabilirsiniz.

Sub adorecordset()

Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

strDB = adres & "VBAdb.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

strSQL = "SELECT \* from [Tarihsel Data] where Ürün='Ürün1' and Ay=#1/31/2016#"

rs.Open Source: =strSQL, ActiveConnection: =con, CursorType: =adOpenDynamic, LockType: =adLockOptimistic

ActiveCell.CopyFromRecordset rs, 10 'çekilen kayıt sayısını 10 ile sınırladık

End Sub

**CursorType**, arama yönü ve görüntüleme tipini ifade eder. Alabileceği değerler şunlardır:

**adOpenForwardOnly**: Default budur. Sadece ileri hareket eder. Aksi gerekmedikçe bunu kullanırsanız daha hızlı erişim sağlarsınız.

**adOpenStatic**: Tüm yönlere izin vardır ve başkaları tarafından yapılan değişiklikler o an size görünmez. Yani bir nevi siz eriştiğiniz anda ilgili kayıt setinin resmi çekilir ve siz hep onu görürsünüz.

**adOpenDynamic**: Bu da tüm yönlere izin verir ama bu sefer başkaları tarafından yapılan değişiklikler size anında görünür.

**adOpenKeyset**: adOpenDynamic’e benzer, ama silinen veya eklenenler size o an görünmez, sadece değişiklikleri görebilirsiniz.

**LockType**, kayıtlar güncellenirken ne tür kilit konacağını ifade eder. Çok kullanıcının eriştiği bir dosyada aynı anda birden çok kullanıcı dataya erişmeye veya değiştirmeye çalışırsa nasıl davranılması gerektiğini belirler.

**adLockReadOnly** : Default budur. Kayıtlar herkeste readonly açılır ve kimse editleyemez.

**adLockOptimistic**: Update sırasında(Update metodu çağrıldığında) kilitler. Başkaları da o sırada görebilir ve editleyebilir.

**adLockPessimistic**: Editlemeye başladığınız anda kilitler. Başkaları o sırada bu kaydı okuyamaz ve editleyemez.

**adLockBatchOptimistic**: adLockOptimistic’in aynısı, sadece toplu güncelleme yapıldığında kullanılır.

Şimdi, yukarda Mode konusunda verdiğimiz örneğe bakalım. Oradaki Mode’u Read yerine Write yapalım, ki veritabanına yazma izni vermiş olalım. Ancak bu sefer de lock tipini adLockReadOnly yapalım. Böyle bir durumda yine yazma izni verilmemiş olur. Mesela çok kullanıcılı bir dosyada, sadece belli kullanıcıların yazma izni olsun istiyorsanız Mode’u Write yaparsınız, ama kullanıcı grubuna göre kimini ReadOnly lock tipinde kimini Diğer lock tiplerinde açarsınız.

Sub adomode2()

Dim con As ADODB.Connection

Dim rs As ADODB.Recordset

Set con = New ADODB.Connection

With con

.ConnectionString = "Provider = Microsoft.ACE.OLEDB.12.0; data source=" & adres & "VBAdb.accdb"

.Mode = adModeWrite

.Open

End With

Set rs = New ADODB.Recordset

rs.Open "data", con, adOpenDynamic, adLockReadOnly

rs.AddNew 'burada hata

rs.Fields(0)="x1 bölgesi"

rs.Fields(1)="y1 şubesi"

rs.Fields(2)="z1 ürünü"

rs.Fields(3)=100

rs.Update

End Sub

**Options** parametresini genelde boş bırakıyoruz, böyle olunca ADO, bunun ne olduğuna kendi karar vermeye çalışıyor. Ben şimdiye kadar belli durumlar haricinde bunu kullanmadım. Sadece aşağıda belirttiğim gibi Seek metodu kullanılırken bunun özel bir değer olarak girilmesi lazım, onun dışında tespiti ADO’ya bırakıyorum.

##### Field nesnesi(Alanlar)

DAO’da söylediğim gibi, bizim Field’larla işimiz daha çok bunlara erişmek şeklinde olacak. Erişimin de 3 yolu bulunmaktadır.

Alan adı ile: rs.Fields("İsim")  
Alan adı kısayolu ile: rs![İsim]  
Alan item no ile: rs.Fields(1)

**Recordset.Fields(0).Name**: İlk kolonun adını yani kolon başlığını getirir.  
**Recordset.Fields(0).Value**: Bu ise ilk kolondaki geçerli kaydın(satırın) içeriğini döndürür. Bu arada Value özelliği default özellik olup yazılmasa da olur, ama biz iyi bir programlamacı olup yazıyoruz.  
**rs.Fields.Count**: İlgili kayıt setindeki alan(kolon) sayısını verir.

##### Yeni kayıt, mevcut düzenleme ve kaydetme

Burada bir çok şey DAO’ya benzer, o yüzden benzer olanları sadece belirteceğim, bunların detaylarına girmektense farklılık gösterenlere değinmeyi tercih edeceğim.

**AddNew** ve **Update** metodları aynen DAO’daki gibi geçerlidir.  
DAO’dan farklı olarak ADO'da, Update işleminden sonra aktif kayıt yeni kayıt olur, DAO’da ise yeni kayda çapa atmak gerekiyordu.

ADO kayıt seti her zaman edit modunda olduğu için ayrıca bir Edit metoduna ihtiyaç duyulmamaktadır.

Update metodu var olmakla birlikte MoveNext gibi cursorın konumunu değiştiren işlemlerde otomatik Update kolaylığı gelmiştir. Bununla birlikte değişikliklerden sonra cursor konumunu değiştiren bir metodu uygulanmayacaksa Update yine de yapılmalıdır.

##### Silme

DAO konusunda bahsettiğim gibi, silme işlemi için ben SQL metni çalıştırmayı tercih ediyorum. Üstelik ADO'da silme işlemi biraz daha detaylı olabiliyor. O yüzden ileri bir vadede bu kısımda güncelleme yapana kadar SQL çalıştırmak ile devam edebilirsiniz.

##### Kayıtlarda dolaşma

DAO’daki Move ve türevleri aynen var. Oraya bakabilirsiniz.

##### RecordCount ile kayıt sayısı

Recordsetteki kayıt sayısı için DAO’daki gibi **RecordCount** özelliği kullanılır. Ancak DAO’da olduğu gibi burda da dikkat edilmesi gereken bazı hususlar vardır. Cursor tipi ve provider türüne göre sonuçlar farklılık gösterir.

* Cursor tipi adOpenForwardOnly ise -1 döner.(Connection nesnesnin Execute metodu recordseti bu tipte açar)
* Cursor tipi adOpenStatic or adOpenKeyset ise sorunsuz çalışır
* Cursor tipi adOpenDynamic ise data kaynağına göre sonuç değişir
  + Destekliyorsa sorunsuz
  + Desteklemiyorsa -1

Kayıt sayısını elde etmenin bir yolu da **GetRows** ve Ubound birleşimi olacaktır ve bu yol her zaman garantidir.

Sub adokayıtsayısı()

Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

strDB = adres & "VBAdb.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

strSQL = "SELECT \* from [Tarihsel Data] where Ürün='Ürün1' and Ay=#1/31/2016#"

rs.Open Source: =strSQL, ActiveConnection: =con, CursorType: =adOpenDynamic, LockType: =adLockOptimistic

dizi = rs.GetRows

Debug.Print rs.RecordCount 'cursortype durumuna göre değişkenlik gösterir

Debug.Print UBound(dizi, 2) + 1 'her zaman garantidir

End Sub

Bu yukardaki kodun DAO karşılığı aşağıdaki gibi olacaktır.

Sub dao2()

Dim db As DAO.Database

Dim rs As DAO.Recordset

Set db = DAO.OpenDatabase(adres + "VBAdb.accdb")

Set rs = db.OpenRecordset("SELECT \* from [Tarihsel Data] where Ürün='Ürün1' and Ay=#1/31/2016#", dbOpenDynaset)

rs.MoveLast

x = rs.RecordCount

Debug.Print x

rs.MoveFirst 'Getrows demek için tekrar başa geliyoruz

dizi = rs.GetRows(x)

Debug.Print UBound(dizi, 2) + 1

End Sub

##### Seek,Find, Filter ve SQL ile kayıt arama

DAO’da olduğu gibi ADO’da da aradığımız bilgiyi bulmanın birkaç yolu bulunmaktadır. Okumadıysanız DAO kısmındaki giriş notlarını okumanızı tavsiye ederim.

###### Seek

DAO’da bahsettiğimiz Find ve Seek arasındaki farklar ve bunların özellikleri büyük ölçüde geçerlidir. Mesela en hızlısı yine Seek metodudur. Ancak, Seek metodunun kullanılabilmesi için Recordsetin Options parametresi adCmdTableDirect olarak belirtilmelidir.

Ayrıca açılan recordsetin index özelliğini ve seek metodunu destekleyip desteklenmediğini **Supports** metodu ile kontrol etmemiz gerekir.

Sub adoseek()

Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

strDB = adres & "VBAdb.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

rs.Open "data", con, adOpenKeyset, adLockOptimistic, adCmdTableDirect 'Seek'in kullanılması için adCmdTableDirect olmalı

If rs.Supports(adIndex) And rs.Supports(adSeek) Then

rs.Index = "Şube Adı"

rs.Seek ("Şube1")

End If

'nomatch yok. filter ve recordcount yapılabilir. veya BOF/EOF kontrolü

If (rs.BOF = True) Or (rs.EOF = True) Then

Debug.Print "Data Not Found"

Else

Debug.Print rs.Fields("Bölge").Value

End If

End Sub

###### Find

Kayıt aramadaki diğer alternatiflerimiz arasında Find var. Özellikle küçük bir recordset üzerinde çalışırken kullanılabilir. Büyük recordsetlerde çok hantal olacaktır.

DAO’da Find yerine Find’ın türevleri bulunmakta idi, ADO’da ise sadece Find bulunmaktadır. Syntax’ı şöyledir: **Find (Criteria, SkipRows, SearchDirection, Start)**

MSDN'de Find'ı kullanmadan önce ilk kayda konumlanılması öneriliyor ancak ben bunu yapmadığımda da kod çalışıyor, yine de tavsiyeye uyuyorum.

Eğer arama yapacağımız küme büyük bir veri kümesiyse öncesinde **Sort** metodu ile recordseti sıralamakta fayda var. Bunun için recordseti açamadan önce **CursorLocation** özelliğine **adUseClient**  değerini atamak gerekiyor.

Önemli bir detay; Metinsel alanlara filtre uygulanırken kriter tek tırnak arasına alınır. Aşağıdaki kodda bir örneği var(Şube112 filtresi).

Sub adofind()

Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

strDB = adres & "VBAdb.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

'rs.CursorLocation = adUseClient 'sort yapılacaksa burasını açın

rs.Open "data", con, adOpenKeyset, adLockOptimistic

rs.MoveFirst 'Find kullanmadan önce ilk kayda konumlanılmalıdır

'rs.Sort = " [Şube Adı]" 'Find’ı kullanmadan önce sıralama yaparak arama hızı artırılabilir.

rs.Find "[Şube Adı] = 'Şube112'"

'nomatch yok. filter ve recordcount yapılabilir. veya BOF/EOF kontrolü

If (rs.BOF = True) Or (rs.EOF = True) Then

Debug.Print "Data Not Found"

Else

Debug.Print rs.Fields("Bölge").Value

End If

End Sub

Diğer hususlar

* NULL yazarken “is null” değil sadece “Null” yazarız.
* DAO’da Find türevleri içinde “AND” kullanılabilirken ADO’da kullanılamıyor. Yani sadece tek bir kolon için arama yapılabilirsiniz.
* Joker eleman desteklenir ve Like kelimesi ile kullanılır.  
  rs.Find = "ŞubeAdı like '\*AKSARAY\*' "
* <https://www.techrepublic.com/article/why-ados-find-method-is-the-devil/> sitesinde Find metodunun kullanılmaması gerektiği ile ilgili tavsiyeler var, kararı siz verin.

###### Filter

Bir diğer yöntem Filter’dır. Filter’ın SQL’deki Where kısmına benzeyen bir kullanım şekli vardır. Hani demiştik ya, ADO kullanırken SQL bilmeye gerek yok, işte bu yöntem ile bir nevi SQL ile filtre uygulamış oluyoruz. **Syntax’ı şöyledir: rs.Filter = Kriterler**

Özellikle belli kriterlere göre **birden çok kayıt getirmek istediğinizde** bunu kullanırız. Tek kayıt ararken daha çok Seek(destekleniyorsa) veya Find kullanılmalı.

Önemli bir nokta var, o da **Recordseti açmadan önce** filtreyi uygulamamız gerektiği. Filtreyi kaldırmak istediğimizde ya "" atarız veya adFilterNone uygularız. Bir diğer nokta da, DAO'da olduğundan farklı olarak burda ayrı bir recordset tanımlamamıza gerek yok, aynı recordset üzerinde filtre uygulanmaktadır.

Sub adofilter()

Dim con As New ADODB.Connection

Dim rs As New ADODB.Recordset

strDB = adres & "VBAdb.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

rs.Filter = "[Bölge] = 'Başkent 1' and [Ürün Adı]='Ürün1' and [Aylık Gerç] > 40000"

rs.Open "data", con, adOpenKeyset, adLockOptimistic

'nomatch yok. filter ve recordcount yapılabilir. veya BOF/EOF kontrolü

If (rs.BOF = True) Or (rs.EOF = True) Then

Debug.Print "Data Not Found"

Else

ActiveCell.CopyFromRecordset rs

End If

End Sub

###### Diğer hususlar

* Filter, DB2’da desteklenmiyor.
* AND/OR keywordunu Find içinde kullanamıyoruz, Filter’da ise kullanabiliyoruz.
* Joker eleman desteklenir ve Like ifadesi ile kullanılır.  
  rs.Filter =”ŞubeAdı like ‘\*AKSARAY\*’ ”

###### SQL

Aranan değeri bulmada son yöntem ise SQL kodu çalıştırmaktır. Aradığımız değer tek ise tek sonuç döndürecek bir SQL yazmamız gerekir. Eğer yazdığımız SQL’in çoklu sonuç döndürme ihtimali varsa MoveFirst diyerek ilk kaydın sonucunu alırız, tabi SQL’de uygun OrderBy işlemini yapmış olmamız kaydıyla.

Aradığımız şey birden çok değer ise çoklu sonuç döndüren bir SQL hazırlarız. Aşağıda SQL çalıştırma(Hem Select hem de eylem sorguları) detayları bulunmaktadır.

#### Execute ile SQL çalıştırma

DAO’da hem Database hem de QueryDef nesnesi için Execute metodu vardı, ADO’da da hem Connection için hem de Command için var. İkisi de recordset döndürüyor. Syntaxları aşağıdaki gibidir:

Set rs1 = conn.Execute (CommandText, RecordsAffected, Options)

Set rs2 = cmd.Execute( RecordsAffected, Parameters, Options )

Tabi çalıştırılacak SQL, Update/Insert gibi değişiklik yapan bir SQL ise bir recordset nesnesine atanmadan doğrudan kullanılabilir.

Genel olarak DAO’daki mantıkla aynıdır. Oraya bakabilirsiniz. Aşağıda bir örneğimiz de var zaten.

Sub adosql()

Dim con As New ADODB.Connection

Dim rs As ADODB.Recordset 'New ifadesini kullanmıyoruz

strDB = adres & "VBAdb.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

cmdstr = "Select \* from data where [Bölge] = 'Başkent 1' and [Ürün Adı]='Ürün1' and [Aylık Gerç] > 40000"

Set rs = con.Execute(cmdstr)

'nomatch yok. filter ve recordcount yapılabilir. veya BOF/EOF kontrolü

If (rs.BOF = True) Or (rs.EOF = True) Then

Debug.Print "Data Not Found"

Else

ActiveCell.CopyFromRecordset rs

End If

End Sub

##### Command Nesnesi

Yukarda gördüğümüz üzere, command nesnesini SQL metinleri çalıştırmak için kullanabiliyoruz. Aşağıda daha detaylı bir örnek var:

Sub adocommand()

Dim Conn As New ADODB.connection

Dim Cmd As New ADODB.Command

Dim Rs As New ADODB.Recordset

Cmd.CommandText = "SELECT \* from data"

Cmd.CommandType = adCmdText

strDB = adres & "VBAdb.accdb"

Conn.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

Cmd.ActiveConnection = Conn

Set Rs = Cmd.Execute

ActiveCell.CopyFromRecordset Rs

End Sub

Her ne kadar connection nesnesi ile de SQL çalıştırabilsek de, command nesnesinin bazı avantajları vardır. Bu avantajları sayesinde bazen tercih edilebilirler.

* Command nesnesi daha hızlıdır
* Connection'ın aksine parametre kabul edebilir. Ama biz buna burada girmeyeceğiz.
* Conection'da olmayan bazı özellikler Command’da bulunur, bunların da detayına girmeyeceğiz.

Bütün bu avantajları nedeniyle command nesnesi bazı durumlarda tercih sebebi olabilmektedir. Bu nesnenin detaylı araştırmasını size bırakıyorum.

NOT: SQL metni çalıştırmanın bir diğer yolu da, recordsetin içinde bunu belirtmektir, ki bunu daha yukarıda zaten görmüştük. Bu yöntemde tabiki sadece Select sorguları çalıştırılır.

#### Datayı Excel’e almak(Import işlemi)

DAO’daki yöntemlerle aynı olduğu için tekrar yapmak istemiyorum. Oraya bakabilirsiniz.

#### Excel’deki datayı Access’e atma

Burda kullanılacak yöntemler de DAO’daki gibidir. Oraya bakabilirsiniz.

#### Şifre güvenliği

Bu maddeyi sadece ADO’ya koydum. Zira DAO ile sadece Access’e bağlanacağımız için ve onda da çoğunlukla şifresiz dosyalara bağlanacağımız için böyle bir sürece gerek bulunmamaktadır. Ancak olur da şifreli bir Access dosyanız varsa, burdaki yöntemleri DAO’da uygulayabilirsiniz.

Neden bahsediyorum, tabiki connection string içine yazdığınız bağlantı şifresinden. Bunu kodlarınız içine ulu orta yazarsanız bir güvenlik sorunu yaratmış olabilirsiniz. Bu güvenliği artırmak bizim elimizde. Aşağıda benim uyguladığım çeşitli yöntemler bulunuyor.

* **VBA project’e protection konması:** İlgili dosyanın VBA koduna birden çok kişinin erişmesi gerekiyorsa(tek geliştirici değilseniz) bu yöntem kullanılmamalıdır. Detaylara **İleri Seviye İşlemler>Add-inler yaratmak** konusunda görüyoruz.
* **XlVeryHidden modundaki bir sayfadan şifrenin okunması:** Detaylara **Dört Temel Nesne>Worksheet** konusunda ulaşabilirsiniz.
* **Alakasız bir klasöre koyacağınız bir text dosyadan okunması**. (Ör: “C: \yemek tarifleri\sebzeliler\brokoli ziyafeti.txt”) Üstelik bu dosyada gerçek bir yemek tarifi olmasında fayda var. Sadece aralarda bir yerlerde şifrenizi gizleyebilirsiniz. FSO/FS gibi nesnelerle de ilgili karakterleri okutabilirsiniz.

Bu 3 maddeyi bir arada bile kullanarak ekstra güvenlik sağlayabilirsiniz.

Bunların dışında bir yöntem daha var ki bu en güvenlisidir: Şifrenin ve hatta User’ın kullanıcıya (her defasında) sordurulması. Bunun dezavantajı ise Schedule edilmiş kodlarda kullanılamaması. Manuel çalıştırılan kodlarda kullanımı uygundur.

#### Örnek Çalışma - İnteraktif Veri Çekme Formatı

DAO'da yaptığımız örneğin ADO versiyonu da aşağıdaki gibidir.

Dim rs As New adodb.Recordset

Dim con As New adodb.Connection

'--------Bölge ve Ay bilgileri değiştiğinde tetiklenecek prosedür

Private Sub Worksheet\_Change(ByVal Target As Range)

'yanlışlıkla başka bir hücreye çift tıklarsa onun içine girmiş olur ve burası tetiklenir

If Target.Row = 4 Then Exit Sub 'başlığa çift tıklanırsa

If IsEmpty(Target) Then Exit Sub 'bir de herhangi boş bir hücreye çift tıklanırsa

On Error GoTo hata

Application.EnableEvents = False

Range("a4").CurrentRegion.Offset(1).Clear 'önce temizlik

If Not Intersect(Target, Range("B1: B2")) Is Nothing Then

[a5].Select

strDB = adres + "\BölgeŞubeRakamları.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

mysql = "select \* from bölgerakam where Bölge='" & Range("bölge") & "' and Ay=" & Range("ayno")

rs.Open mysql, con, adOpenForwardOnly, adLockOptimistic

ActiveCell.CopyFromRecordset rs

rs.Close

con.Close

End If

hata:

Application.EnableEvents = True

End Sub

'----Ürün bilgisine çift tıklandığında tetiklenip şube detayını gösterecek olan prosedür

Private Sub Worksheet\_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)

If Target.Column <> 2 Then 'B kolonu dışında çift tıklarnırsa çıksın

Cancel = True

Exit Sub

End If

If IsEmpty(Target.Offset(1, -1)) And Not IsEmpty(Target.Offset(1, 0)) And Not IsEmpty(Target.Offset(0, -1)) Then

Application.EnableEvents = False

Do

ActiveCell.Offset(1, 0).EntireRow.Delete

Loop Until Not IsEmpty(ActiveCell.Offset(1, -1))

Application.EnableEvents = True

Target.Offset(0, 1).Select

Exit Sub

End If

If Not Intersect(Target, Range([b5], [b5].End(xlDown))) Is Nothing And Not IsEmpty(Target.Offset(0, -1)) Then

Application.EnableEvents = False

strDB = adres + "\BölgeŞubeRakamları.accdb"

con.Open ConnectionString: ="Provider = Microsoft.ACE.OLEDB.12.0; data source=" & strDB

mysql = "select şube,ay,rakam from şuberakam where Bölge='" & Range("bölge") & "' and Ay=" & Range("ayno") & " and ürün = '" & Target.Value2 & "'"

rs.Open mysql, con, adOpenStatic, adLockOptimistic

şubeadet = rs.RecordCount 'adOpenStatic olduğu içn sorunsuz çalışır

For i = 1 To şubeadet

Target.Offset(1, 0).EntireRow.Insert Shift: =xlDown, CopyOrigin: =xlFormatFromLeftOrAbove

Next i

rs.MoveFirst

Target.Offset(1, 0).CopyFromRecordset rs

Target.Offset(0, 1).Select

rs.Close

con.Close

Application.EnableEvents = True

End If

End Sub

## Connection, ListObject ve QueryTable nesneleri

Bu bölümde, DAO ve ADO'dan yöntem olarak biraz farklılaşan, refreshlenebilir bağlantı tekniklerine değineceğiz. Bu sayfada anlatılanları anlamak için Excel ön yüzde dış veri kaynaklarıyla çalışma tecrübenizin olması gerekiyor. Eğer daha önce böyle bir çalışma yapmadıysanız öncelikle bu tecrübeyi edinin, sonra bu konuyu inceleyin.

Örneklerdeki kodların olduğu dosya ve veritabanı önceki konuda **githubdan** indirdiğimiz VBAdata.rar içindedir. İndirmediyseniz şimdi indirebilirsiniz.

### Giriş

Önceki bölümde DAO ve ADO ile bir veritabanına bağlanıp nasıl veri çekileceğini gördük. Bunların ortak noktası datayı değişkenler/diziler içinde depolayarak veya metodlar aracılığı ile Excel’e yapıştırmak şeklindeydi. Bu yöntemlerle elde edilen veri, Data menüsünden veya sağ tıklanarak refreshlenemez. Yani DAO/ADO refresh yapılamayan ve statik data getiren yöntemlerdir.  Ancak ilgili kod tekrar çalıştırılarak(ilgili kod bir butona atanabilir), güncel data getirilebilir, yani dolaylı yoldan bir nevi refresh uygulanabilir. Refresh yapılabilirliğin VBA açısından önemi **Refresh** ve **RefreshAll** metotlarının kullanım kolaylığındandır.

Excel’den veritabanlarına ulaşmanın bir yolu da MS Query veya Data Connection Wizard aracılığı ile refreshelenebilir bir Tablo(VBA karşılığı ListObject) yaratmaktır. Bu kısımda bu yöntemlere bakıyor olacağız, tabi sadece VBA ile ilgili kısmına.

Refreshlenebilir bir dosyayı ilk kez oluşturma işini genelde Excel içinden manuel yaparız. Ancak bazı durumlarda VBA ile de yapmamız gerekebilmektedir. Yine de ilk Connection String bilgilerini v.s ilk başta Excel’den yapmak daha kolaydır, hatta macro recorder ile oluşan kodu alıp, aradaki gereksiz kodu temizledikten sonra VBA kodunu hazırlayabilirsiniz.

Bunun refresh işlemini ise VBA’e bırakabiliriz, tabiki manuel olarak refresh edilebilir ancak bu sayfada VBA ile bunun yapılış yollarını öğreneceğiz. Zira bu bilgiler schedule işlemlerinde kritik öneme sahiptir.

#### Önemli Classlar

DAO ve ADO’dan farklı olarak bu sefer herhangi bir library’yi referans olarak eklememize gerek bulunmamaktadır. Bunun yerine Excel'in açılmasıyla mevcut bulunan 3 sınıf/nesne üzerinde çalışıyor olacağız.

* **Connection(s) sınıfı:** Varolan bağlantılarla ilgili işlem yapılmasını veya bilgi edinilmesini sağlar.
* **ListObject(s) sınıfı:** Yeni connection kurmamızı sağlar. Nihai liste Table şeklindedir.
* **QueryTable(s) sınıfı:** Yeni connection kurmamızı sağlar. Nihai liste Table şeklinde değildir.

Şimdi bunlara detaylıca bakalım.

### Connections sınıfı

Connections’ın sonundaki s harfinden bunun bir collection olduğunu anlıyoruz. Normalde hemen her collectiona ait bir de sonunda “s” olmayan nesnesi vardır, Workbooks/Workbook gibi. Ancak Connections için bu tipte bir nesne yok. Bununla ilgili hiçbir kaynak bulamadım ancak benim tahminim tek bir tür connection olmaması. Bizim burada ilgileneceğimiz nesne **WorkbookConnection**’dır. WorkbookConnection nesnesi tüm dış bağlantıları yöneten nesnedir. Bunun da aşağıdaki gibi kendi içinde alt türleri vardır:

(Yandaki rakamlar Type propertysinin değeridir.)  
2, xlConnectionTypeODBC  
1, xlConnectionTypeOLEDB  
4, xlConnectionTypeTEXT  
5, xlConnectionTypeWEB  
Diğer bikaç tip daha var, ama onlar bizi şu aşamada hiç ilgilendirmiyor.

Connection’ların güzelliği bir kere yaratıldıklarında Excel içinde depolanabilir olmalarıdır. Connectionlar, ayrı bir connection dosyasında da tutulabilir. Bunlar genelde ODC uzantılı dosyalardır. Ama biz bu şekilde çalışmayacağız, Excel içinde depolayacağız. Daha önceden bahsettiğim gibi bazen de direkt kodun içinde connection yaratımı olabilir, dolayısıyla dosya ilk açıldığında connection yoktur, onu yaratan kod tetiklenince oluşur. Her ne kadar çok sık ihtiyacımız olmayacaksa da bu yöntemi de göreceğiz.

#### Yeni yaratım

Her bir QueryTable, ListObject ve harici dataya başvuran bir PivotCache yaratıldığında yeni bir WorkbookConnection nesnesi yaratılır. Bu şekilde dolayı olarak yaratılan connectionlara biraz aşağıda bakacağız.

Sonradan Existing Connections içinden kullanılmak üzere standalone(tek başına duran) WorkbookConnection nesnesi de yaratılabilir, ama buna çok gerek olacağını sanmıyorum, o yüzden yeni yaratım yerine var olan connectionlar üzerinde duracağız.

#### ConnectionString

Bir connection’ın en önemli öğesi Connection String’dir. DAO ve ADO’da gördüğümüz gibi bu öğe, dataya nasıl bağlanılacağını belirten yönergeyi içerir. Bunun içinde genellikle bağlantı türünün ODBC/OLEDB mi olduğu, Driver/Provider’in ne olduğu, varsa user ve şifreler ve diğer parametreler  bulunur.

Çeşitli veritabanlarına bağlanma yöntemlerini [www.connectionstrings.com](http://www.connectionstrings.com) adresinden bulabilirsiniz.

#### Connection'a Erişim ve Varolan Connection'ı Editleme

Bir kez daha belirtmek gerekirse biz daha çok Connectionlar’ın yeni yaratımlarından ziyade mevcut Connectionlar’ın özelliklerini okuma/değiştirme işlemleri yapacağız.

Connection’ın kendisine Connections koleksiyonuna index veya isim vererek ulaşabiliyoruz.

Activeworkbook.Connections(1)

Activeworkbook.Connections("Query from PDWH\_USR")

Connection’ın kendisini elde ettikten sonra bunla ilgili işlemleri yapmak için genelde Connection’ın tipine göre ilgili property’yi seçip daha çok bunun üzerinden işlem yapıyoruz, Connection’ın kendisiyle pek bir işimiz olmuyor. Yani bağlantı tipimiz ODBC ise Activeworkbook.Connections(1) yerine **Activeworkbook.Connections(1).ODBCConnection** üzerinde işlem yapıyoruz.

Connection’ı kendisiyle ilgili olarak tek ihtiyaç duyacağımız bilgi connection’ın tipidir. Yani ODBC ise farklı işler olsun OLEDB ise farklı şeyler olsun demek için tip bilgisine ihtiyaç duyarız. Bunları da yukarıda görmüştük; ODBC için tip xlConnectionTypeODBC olup bunun nümerik değeri 2'dir. Buna property **Activeworkbook.Connections(1).Type** şeklinde ulaşabiliriz

Bu tip bilgisine ulaştıktan sonra connection tipine göre ilgili property seçilir: **Activeworkbook.Connections(1).ODBCConnection** gibi. Bunun da sadece birkaç propertysine ihtiyaç duyacağız. En önemlisi, bize **ConnectionStringi** veren **Connection** propertysi’dir. Gördüğünüz gibi ConnectionString diye bir property yok onun yerine Connection var: **Activeworkbook.Connections(1).ODBCConnection.Connection**

Mesela connection tipi ODBC ise ConnectionString ... olsun demek için aşağıdaki kodu yazarız.

If Activeworkbook.Connections(i).Type = 2 Then

Activeworkbook.Connections(i).ODBCConnection.Connection="....."

End If

Bu ConnectionString’i elde etmenin bir yolu daha var, bunu ilgili kısımda ayrıca göreceğiz ancak ön bilgi adına aşağıda veriyorum,  QueryTable'lar üzerinden:

ActiveSheet.ListObjects(1).QueryTable.Connection 'access, oracle v.s

'Veya text/web connection ise Doğrudan QT üzerinden(web ve txt con için)

ActiveSheet.QueryTables(1).Connection

#### Refresh işlemleri

Workbook nesnesinin **RefreshAll** adında bir metodu vardır, ilgili dosyadaki tüm bağlantıları (ve pivot tabloları) refreshler. Bu bazı durumlarda işimizi gören pratik bir yöntemdir. Özellikle bu metod, kodumuzun son satırı yer alıyorsa. Mesela uzunca bir işten sonra ActiveWorkbook.RefreshAll derseniz, dosyadaki tüm bağlantılar sorunsuzca refreshlenir.

Ancak(aslında büyük bir ancaaaak), bu metod **asenkron(asynchronous)** bir metoddur. Yani, refresh işlemleri henüz bitmeden sonraki satırlar okunmaya devam edilir. Bu da her zaman olmasa da bazen sorunlara neden olur. Zira arkasından bir kaydetme işlem gibi bir işlem varsa  hatalara ve kitlenmelere neden olabilir.

O yüzden kodun son satırı olmadığı sürece refresh işlemlerini connection nesnesi üzerinden(bir döngü içinde tüm bağlantıları ele alacak şekilde) yapmayı tercih etmeliyiz, workbook nesnesi üzerinden değil. Böylece tüm bağlantılar refresh olana kadar beklemiş oluruz. Zira Connection nesnesinin Refresh metodu, **BackgroundQuery** özelliğine **False** atandığında **synchronous** çalışır, yani bir sonraki satıra geçmeden önce mevcut satır icrası beklenir.

Sub AlternatifRefreshAll()

'Önceki kodlar

For Each cn In ActiveWorkbook.Connections

cn.ODBCConnection.BackgroundQuery = False 'kodun synchronous işlemesini sağlar

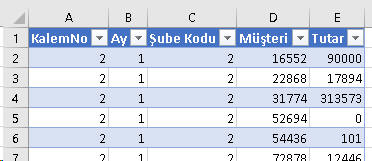
cn.Refresh

Next

'Sonraki kodlar

End Sub

Örneğin aşağıdaki tabloda mevcutta bulunan kayıt sayısını(listobjectin row sayısı) yazdırmak istiyoruz diyelim. Önce mevcudu yazdıracağız, hemen arkasından DAO ile bir kayıt ekleyeceğiz ve hemen arkasından tekrardan kayıt sayısını okuyacağız.



Bunun için kodumuzu şu şekilde hazırlarsak

Sub async\_refresh()

Dim ws As Worksheet

Set ws = ActiveSheet

Debug.Print "Yeni kayıt öncesi: " & ws.ListObjects(1).ListRows.Count, Time

Call KayıtEkle 'Kayıt ekleme işini yapan prosedür

ActiveWorkbook.RefreshAll 'yaklaşık 15-20 sn çalışır

Debug.Print "Yeni kayıt sonrası: " & ws.ListObjects(1).ListRows.Count, Time

End Sub

Bunun çıktısı şöyledir:   
  Yeni kayıt öncesi: 334 21: 46: 17   
  Yeni kayıt sonrası: 334 21: 46: 18

Gördüğünüz gibi, uzun süren refresh işini beklememiş ve önceki kayıt sayısı ile sonrakini aynı vermiş ve bunu 1 sn içinde yapmıştır.

Ancak güvenli yöntem ile çalıştırırsak,

Sub sync\_refresh()

Dim con As WorkbookConnection

Dim ws As Worksheet

Set ws = ActiveSheet

Debug.Print "Yeni kayıt öncesi: " & ws.ListObjects(1).ListRows.Count, Time

Call KayıtEkle

For Each con In ThisWorkbook.Connections

con.ODBCConnection.BackgroundQuery = False

con.Refresh

Next con

Debug.Print "Yeni kayıt sonrası: " & ws.ListObjects(1).ListRows.Count, Time

End Sub

Sonuç şöyle çıkacaktır:   
  Yeni kayıt öncesi: 336 21: 47: 57   
  Yeni kayıt sonrası: 337 21: 48: 06

Görüldüğü üzere 9 sn'lik refresh işlemi beklenmiş ve kayıt sayısının 1 arttığı gösterilmiştir.

"KayıtEkle" prosedürümüz de şöyledir.

Sub KayıtEkle()

Dim db As Database

Dim rs As Recordset

Set db = DAO.OpenDatabase("buraya dosya adresini yazın")

Set rs = db.OpenRecordset("Anadata", dbOpenTable)

rs.AddNew

rs.Fields(0) = 2

rs.Fields(1) = 1

rs.Fields(2) = 2

rs.Fields(3) = WorksheetFunction.RandBetween(1000, 2000)

rs.Fields(4) = WorksheetFunction.RandBetween(1, 1000000)

rs.Update

rs.Close

db.Close

End Sub

ÖNEMLİ: RefreshAll yapılması durumunda ortaya çıkan sorunlar için birçok sitede önerilen DoEvents, Wait, Sleep yöntemlerinin hiçbiri işe yaramaz. Bu işin tek çözümü budur. İlgilenenler [https://stackoverflow.com/questions/22083668/wait-until-activeworkbook-refreshall-finishes-VBA](https://stackoverflow.com/questions/22083668/wait-until-activeworkbook-refreshall-finishes-vba) sayfasında da detay bakabilirler.

### ListObject ve QueryTable

#### Data erişim yolları

Excel’den dış dataya çeşitli yollardan ulaşabiliyoruz. Bunların türlerini web sitemdeki Excel ana menüsünden ulaşabilirsiniz. Bizim için önemli olan hangi yöntemle yaratıldığında (2013 ve 2016’da yeni gelen Get&Transform, New Query yöntemleri dahil) nereler değişiyor, ne tipte bir bağlantı oluşuyor detayı. Çünkü oluşan bağlantı tipinin ODBCConnection mı yoksa OLEDBConnection mı olduğuna bağlı olarak kodlarımızda ayarlama yapmamız gerekecek.

Genel olarak bakıldığında;

* **MS Query** her zaman ODBC yaratır
* **Data\_Conn\_Wizard**: (2016'dan itibaren Legacy)
  + ODBC DSN-->ODBC
  + Other/Advanced-->OLEDB
* **Data\_Import\_Access**: OLEDB
* **Txt import**: TEXT
* **SQLServer**: OLEDB
* **PowerQuery**: OLEDB(Mashup)

NOT: ODBC’de CommandType her zaman SQL’dir, yani **xlCmdSql**. OLEDB’de ise SQL olabileceği gibi doğrudan bir tablo/query de olabilir.

#### ListObject ve QueryTable türleri

**ListObject** dediğimiz nesne aslında bir **Table** alanıdır.  Bu nesnenin Name özelliğine bakınca Table menüsündeki ismin aynısını görürüz.(ListObject’in eski adı zaten TableList idi, sonradan ListObject olmuş.)

İki tür listobject olabilir:

**Data bağlantılı**: Böyleyse bunun QueryTable propertysi vardır, ve refreshlenebilir.

**Data bağlantısız**: FormatAsTable ile Table yapıp uygulanmıştır. Bunun bir data bağlantısı yoktur. Querytable özelliği de yoktur, dolayısıyla refreshlenemez.

Biz doğal olarak burada ilk tür Listobject ile ilgileneceğiz.

QueryTable’a gelince, o her halükarda refreshable data içerir. ListObject’deki gibi QueryTable’ın da iki türünden bahsedebiliriz.

**ListObject ile ilişkili**: Access, Oracle gibi DB’lerden kurulan bağlantılar.

**ListObject ile ilişkisiz**: Text ve Web bağlantıları tek başına bulunan(standalone) QueryTable’lar üretir.

Excel’de bir bağlantı kurduğumuzda bu bir text veya web bağlantısı değilse ListObject ve QueryTable ikisi de otomatikman oluşur, ister VBA içinden ister Excel menülerden oluşturmuş olalım fark etmez. Yani VBA’de standalone ListObject yaratsak bile ona eşlik eden QueryTable de yaratılmış olur.

ListObject’e eşlik eden QueryTable’lar Worksheet.QueryTables koleksiyonun bir parçası değildir. Bunlara sadece ListObject.QueryTable özelliği ile erişilebilir.

Access,Oracle veya SQL Server gibi DB bağlantılarında Listobject'in bir adı varken, QueryTable'ın adı olmaz.(QueryTable'ın Name propertysi sadece text ve web queryler için var)

Yani bir text bağlantısı(Standalone Querytable) ve bir Oracle bağlantısı içeren bir dosyada şu sonuçları elde ederiz.

Debug.Print ActiveWorkbook.Connections.Count '2 (ikisi de)

Debug.Print ActiveSheet.ListObjects.Count '1 (Oracle)

Debug.Print ActiveSheet.QueryTables.Count '1 (Text)

Ama dosyamızda aslında iki tane QueryTable var, fakat bunlardan biri Listobject'n query'si. Bunlarda ayrı ayrı ilgili Listobject'in QueryTable özelliği ile ulaşılabilir ama tek seferde hepsini saydıracak bir Count özelliği yok, zaten buna  gerek de yok, Listobject sayısı kadar Listobjecte bağlı QueryTable sayısı vardır.

Yine mesela aşağıdaki kod bir Oracle bağlantısı olan dosyada hiçbir şey yazmazken text bağlantısı olan bir dosyada QueryTable’nin adını yazar.

Sub LOQT()

Dim ws As Worksheet

Dim q As QueryTable

Set ws = ActiveSheet

For Each q In ws.QueryTables 'bu bize standalonelarınkini verir

Debug.Print q.Name

Next q

End Sub

#### Yeni yaratımlar

Yukarıda kısaca değindiğimiz data erişim yollarından biri kullanılarak yeni bir bağlantı yaratabiliyoruz. Bu yeni yaratımlara çeşitli örnekleri aşağıda bulabileceğiniz gibi Makro Kaydetme aracı ile kendiniz de görebilirsiniz. Tüm yapmanız gereken Record Macro tuşuna bastıktan sonra bağlantınızı kurmak ve sonrasında kodu incelemek. Tabi makro kaydedici gereksiz uzunlukta bir kod üretmektedir, zira tüm parametrelere değer atamak gibi bir özelliği var bu aracın. Siz burada gereksiz parametre atamalarını çıkarın, diğerlerine otomatikman default değerleri atanacaktır. Array şeklindeki CommandText'i de tek bir string olarak yazın.

Örneğin **Access**'ten **MS Query** ile bir bağlantı kurduğumuzda aşağıdaki gibi bir kod oluşmakta

Application.CutCopyMode = False

With ActiveSheet.ListObjects.Add(SourceType: =0, Source: =Array(Array( \_

"ODBC;DSN=MS Access Database;DBQ=C: \inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar\VBAdb.accdb;DefaultDir=C: \inetp" \_

), Array( \_

"ub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar;DriverId=25;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;" \_

)), Destination: =Range("$A$1")).QueryTable

.CommandType = 0

.CommandText = Array( \_

"SELECT Data.Bölge, Data.`Şube Adı`, Data.`Ürün Adı`, Data.`Aylık Gerç`" & Chr(13) & "" & Chr(10) & "FROM Data Data" \_

)

.RowNumbers = False

.FillAdjacentFormulas = False

.PreserveFormatting = True

.RefreshOnFileOpen = False

.BackgroundQuery = True

.RefreshStyle = xlInsertDeleteCells

.SavePassword = False

.SaveData = True

.AdjustColumnWidth = True

.RefreshPeriod = 0

.PreserveColumnInfo = True

.ListObject.DisplayName = "Table\_Query\_from\_MS\_Access\_Database"

.Refresh BackgroundQuery: =False

End With

Biz bunu manuel olarak yazmak istediğimizde aşağıdaki gibi çok daha sade yazarız.

Sub yeni\_access\_LO\_odbc()

Dim constr As String

Dim qt As QueryTable

constr = "ODBC;DSN=MS Access Database;DBQ=C: \inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar\VBAdb.accdb;DefaultDir=C: \inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar;DriverId=25;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;"

' OLEDB versiyonu--> constr = "OLEDB;Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C: \......."

Set qt = ActiveSheet.ListObjects.Add(SourceType: =0, Source: =constr, Destination: =ActiveCell).QueryTable

With qt

.CommandType = xlCmdSql 'recordmacro sırasında 0 yazar, aynen brakabilecğeiniz gibi manuel yazarken intellisense çıkar ve bunu seçersiniz

.CommandText = "select \* from data" 'Record macroda otomatik oluşan Array yerine tek bir string içine yazarsınız

.ListObject.Name = "Access bağlantı"

'Diğer öğeleri pas geçebiliriz, böylece default değerleri atanır

.Refresh BackgroundQuery: =False

End With

End Sub

Bu arada eğer dataya erişim yöntemi olarak **Data>Get&Transform>From Database** deyip Access seçerseniz, üretilen kod biraz daha farklı olmakta, zira burada **Power Query** devreye sokulmaktadır.

ActiveWorkbook.Queries.Add Name: ="Data (2)", Formula: = \_

"let" & Chr(13) & "" & Chr(10) & " Source = Access.Database(File.Contents(""C: \inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar\VBAdb.accdb""), [CreateNavigationProperties=true])," & Chr(13) & "" & Chr(10) & " \_Data = Source{[Schema="""",Item=""Data""]}[Data]" & Chr(13) & "" & Chr(10) & "in" & Chr(13) & "" & Chr(10) & " \_Data"

ActiveWorkbook.Worksheets.Add

With ActiveSheet.ListObjects.Add(SourceType: =0, Source: = \_

"OLEDB;Provider=Microsoft.Mashup.OleDb.1;Data Source=$Workbook$;Location=""Data (2)"";Extended Properties=""""" \_

, Destination: =Range("$A$1")).QueryTable

.CommandType = xlCmdSql

.CommandText = Array("SELECT \* FROM [Data (2)]")

.RowNumbers = False

.FillAdjacentFormulas = False

.PreserveFormatting = True

.RefreshOnFileOpen = False

.BackgroundQuery = True

.RefreshStyle = xlInsertDeleteCells

.SavePassword = False

.SaveData = True

.AdjustColumnWidth = True

.RefreshPeriod = 0

.PreserveColumnInfo = True

.ListObject.DisplayName = "Data\_\_2"

.Refresh BackgroundQuery: =False

End With

Biz şimdilik MS Query yöntemiyle ilerleyeceğiz. Zira birçok kullanıcıda şuan için PowerQuery entegrasyonunun olmadığını varsayabiliriz. Gerçi olsa da fark etmez, kodlar büyük ölçüde aynı, sadece connection strginlerinde küçük farklar söz konusu.

##### Standalone bağlantı

Öncelikle standalone(kendi başına) bağlantılara bakalım.

Bu örneğe bakarken hali hazırda üzerinde bağlantı olan bir dosya ile çalışmanızı tavsiye ederim. Bu dosya sizin kendinize ait bir dosya olabileceği gibi en üstte benim linkini verdiğim örnek dosya da olabilir.

Sub standalone\_LO\_QT()

'F8 ile tek tek çalıştırın

Dim ws As Worksheet

Dim qts As QueryTables

Dim qt1 As QueryTable, qt2 As QueryTable, yeniqt As QueryTable

Dim los As ListObjects

Dim lo1 As ListObject, lo2 As ListObject, yenilo As ListObject

Set ws = ActiveSheet

Set dest = ActiveCell

'1)\*\*\*\*\*\*\*QueryTable işlemleri\*\*\*\*\*\*\*\*\*\*\*\*Bunlar standalone QueryTable'ler içindir.(Text dosyası gibi)

'1a)Mevcutlara erişim: Mevcutta standalone Querytable bulunan bir sayfada çalışın

Set qts = ws.QueryTables

Set qt1 = qts.Item(1) 'veya item kullanılmadan doğrudan qts(1) yazılabilir

Set qt2 = qts("ExternalData\_4") 'ismiyle de ulaşabiliriz

'1b)Yeni yaratım

Set yeniqt = qts.Add(connection: ="TEXT;" & adres & "hatalog.txt", Destination: =dest)

yeniqt.Refresh

'!!!!!!!!!!!!!!!!!!!!!!----şimdi başka bir boş hücre seçin-----------!!!!!!!!!!!!!!!!

Set dest = ActiveCell

'2)\*\*\*\*\*\*\*standalone ListObject işlemleri\*\*\*\*\*\*\*\*\*(Access, Sqlserver, oracle v.s)

'2a)Mevcutlara erişim: Mevcutta listobject bulunan bir sayfada çalışın(standalone olup olmadığı fark etmez)

Set los = ws.ListObjects

Set lo1 = los.Item(1) 'veya item kullanılmadan doğrudan los(1) yazılabilir

Set lo2 = los("Table\_Query\_from\_MS\_Access\_Database") 'ismiyle de ulaşabiliriz

'2b)yeni eklemek için querytable lazım zira refreshlenen şey aslında querytabledır. o yüzden bu madde standalone listobject içermez

strSource = "ODBC;DSN=MS Access Database;DBQ=" & adres & "VBAdb.accdb;DefaultDir=" & \_

adres & ";DriverId=25;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;"

Set yenilo = los.Add( \_

SourceType: =xlSrcExternal, \_

Source: =strSource, \_

LinkSource: =True, \_

XlListObjectHasHeaders: =xlGuess, \_

Destination: =dest)

With yenilo.QueryTable

.CommandText = "SELECT Data.Bölge, Data.`Şube Adı`, Data.`Ürün Adı`, Data.`Aylık Gerç`" & Chr(13) & "" & Chr(10) & "FROM Data"

.CommandType = xlCmdSql

.Refresh BackgroundQuery: =False

End With

'Ancak yaratım olduktan sonra standalone listobject refreshlenebilir

yenilo.Refresh

End Sub

##### ListObject ile ilişkili bağlantılar

Aşağıdaki örnekte ListObject ilişkili bir QueryTable'la ilgili çeşitli özelliklere bakıyoruz.

Sub qt\_ilişkili\_lo()

Dim lo As ListObject

Dim loqt As QueryTable

Set lo = ActiveSheet.ListObjects.Item(1)

Set loqt = lo.QueryTable

Debug.Print loqt.Connection 'connection string verir

Debug.Print loqt.CommandText 'SQLi verir

'Debug.Print loqt.FieldNames 'bişey dğeil, true/false, isimler gürnümsünmü

Debug.Print loqt.ListObject.Name 'Table\_Query\_from\_MS\_Access\_Database7

'Debug.Print loqt.Name 'hata verir

Debug.Print loqt.QueryType '1 yazar. 1: xlODBCQuery 2: xlDAORecordset 7: xlADORecordset 5: xlOLEDBQuery 6: xlTextImport

Debug.Print loqt.WorkbookConnection.Name 'Query from PDWH\_USR11

Debug.Print lo.DisplayName 'Query from MS Access Database1

Debug.Print lo.ListColumns.Count

Debug.Print lo.ListColumns(5) 'son eklediğim manuel alanı da saydı

Debug.Print lo.Name 'Table\_Query\_from\_MS\_Access\_Database7

'Debug.Print lo.QueryTable.Name 'hata verir

Debug.Print lo.Range.Address 'son yeni eklenen kolon dahil verir

Debug.Print lo.SourceType '3 3: xlSrcQuery, 0: xlSrcExternal 1: xlSrcRange

loqt.Refresh 'lo.refresh de olur

End Sub

##### Text dosyası bağlantısı

Sub txt\_yeni\_con()

'sadece QueryTable, ListObject olmaz  
    kaynak = "TEXT;Q: \.......\kredilog.txt"  
    Set qt = ActiveSheet.QueryTables.Add(Connection: =kaynak, Destination: =ActiveCell)  
    qt.Name = "kredi sorgusu"  
    qt.Refresh  
End Sub

#### Çeşitli üyeler

**Unlink**: Listobject'in veri kaynağıyla bağını koparır ama Table özelliği devam eder.

**Unlist**: Listobject'in hem veri kaynağıyla bağı koparır hem de Table özelliği bozulur, yani Table formatından normalde Range’e döndürür. Gerçi zebra desen olarak yine kalır ama fonksiyonalite olarak artık Table gibi davranamaz. Yani herhangi bir yere yeni satır eklenince zebra deseni de bozmuş olursunuz. (Table seçiliyken Design menüsündeki "Convert to Range" ile aynı işi yapar)

Ben daha çok Unlink’i tercih etmenizi tavsiye ederim. Ne de olsa Table candır.

**ListColumns**: Bu özellik de yine ListObject'e ait olup, çoğu durumda çektiğimiz veri kaynağındaki kolon sayısını elde etmede kullanılır. Ancak bazı durumlarda biz Excel üzerinde de yeni kolonlar ekleyebilir ve bu kolonda çeşitli hesaplamalar yapabiliriz. Örneğin ListObjectin altında yatan QueryTable’ımıza baz teşkil eden SQL metni (veya tablo/query) 5 kolondan oluşuyor diyelim. Sonuç döndükten sonra biz bir de formül yazacağımız 6.kolon açarsak, ListObject’in kolon sayısı(**ListColumns.Count**) 5 değil 6 olacaktır.

**ListRows**: İlgili ListObject'te kayıt sayısıyla ilgili işlemler yapmayı sağlayan collection’dır. Bunun da Count property'si ve index alma özelliği kullanılabilir. Toplam kayıt sayıt sayısını veren B ve son kaydı silen **lo.ListRows(lo.ListRows.Count).Delete** gibi.

Diğer üyelerini de Macro Recorder ile Table menüsündeki ayarlarla oynayarak kendinizi görebilirsiniz ama en önemlileri bunlardı diyebilirim.

#### Refresh

Bir Table'ı refreshleme 2 türlü olabilir.

##### 1)ListObject veya QueryTable üzerinden

Refresh işlemleri için QueryTable’nin de ListObject’in de **Refresh** metodu kullanılabilir. Bu bağlamda **Activesheet.ListObjects(1).QueryTable.Refresh** ile  **Activesheet.ListObjects(1).Refresh** tamamen özdeştir.

'i)Text ve Web dışı kaynaklarda

ActiveSheet.ListObjects(1).QueryTable.Refresh BackgroundQuery: =False

'veya

ActiveSheet.ListObjects(1).Refresh BackgroundQuery: =False

'ii)Text veya web kaynaklarda

ActiveSheet.QueryTables(1).Refresh BackgroundQuery: =False

##### 2)Doğrudan connection üzerinden gidilebilir

Bu yöntemi yukarıda connections konusunda görmüştük.

ActiveWorkbook.Connections(1).ODBCConnection.BackgroundQuery = False

ActiveWorkbook.Connections(1).Refresh

NOT: Her iki grupta da senkron bir refresh işlemi için BackgroundQuery özelliklerine False değeri atanmıştır.

#### Şifre Güvenliği

ListObject ve QueryTable kullanmanın bir sakıncası veritabanı bağlantı şifrelerinizin Excel içinde gömülü olarak bulunması olacaktır. Bunun ele geçmemesi için alınabilecek birkaç önlem olmakla birlikte bilgili (ve kötü niyetli) bir kullanıcı tarafından ele geçirilebilir. Bu önlemlere aşağıda yer verilmiştir.

* Bir sayfa Activesheet.Visible= xlVeryHidden yapılıp şifre buradan okunabilir.
* Ribbon ilgili dosya aktifken geçici olarak kapatılabilir. Böylece Data veya Table menüsünden Properties butonuna tıklayamazlar.

Private Sub Workbook\_Activate()

If Environ("UserName") = 12345 Then Exit Sub

Application.ExecuteExcel4Macro "SHOW.TOOLBAR(""Ribbon"",False)"

Application.DisplayFormulaBar = False

Application.DisplayStatusBar = Not Application.DisplayStatusBar

ActiveWindow.DisplayWorkbookTabs = False

End Sub

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

Private Sub Workbook\_Deactivate()

If Environ("UserName") = 12345 Then Exit Sub

Application.ExecuteExcel4Macro "SHOW.TOOLBAR(""Ribbon"",True)"

Application.DisplayFormulaBar = True

Application.DisplayStatusBar = True

ActiveWindow.DisplayWorkbookTabs = True

End Sub

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

Private Sub Workbook\_Open()

If Environ("UserName") = 12345 Then Exit Sub

MsgBox "Ribbon bu dosya aktif olduğu sürece devre dışı olacaktır. Başka bir Excel dosyasına geçerseniz tekrar aktif olacaktır."

End Sub

* Unlist veya Unlink kullanımı: Söz konusu dosya günlük oluşan bir rapor ise şifreniz Format dosyanızda durur, dosya hergün kaydolurken ilgili günün tarihi ile kaydolur ve bu dosyada Unlist/Unlink yaparak şifrenin görünmesini engellemiş olursunuz. Ancak Format dosyasında şifreniz hala kalacaktır.
* Kullanıcıların şifreyi içeren Format dosyasını açmasını engellemek için de ilgili dosyanın Workbook\_Open markosuna sizden başkasının açmasını engelleyecek bir kod koyabilirsiniz. Üstelik bu tür giriş denemelerini Logger gibi bir fonksiyonla da kayıt altına alabilirsiniz. (Tabi kötü niyetli kullanıcı makroları disable yaparak bu engeli de aşabilir.)

If readonlykontrol = True Then

If Environ("UserName") <> 12345 Then 'bu sizin siciliniz olsun

If InStr(ActiveWorkbook.Name, "Format") > 0 Then

Logger WorksheetFunction.Rept(" ", 50 - Len(raporad)) & raporad, "İzinsizGiriş", 0, Environ("UserName") + " nolu sicil Q'dan giriş yapmaya çalıştı"

MsgBox "Hey dostum, ortalığı karıştırmayı bırak. Format dosyama sakın bir daha dokunayım deme"

ActiveWorkbook.Close savechanges: =False

Exit Sub

End If

End If

End If

* ListObject ve QueryTable’yi VBA ile oluşturmak: Son çözüm, Workbook\_Open içinde ilgili bağlantıların yaratıldığı kodları yazmaktır. VBA projesine de Protection koydunuz mu işlem tamamdır. Bu da kırılamaz bir şifre değildir tabi ancak bu hinlikleri bilen biri tarafından kırılabilir.
* Çözüm değil ancak alternatif olarak yukarıdaki ADO ve DAO yöntemlerini kullanmak da işinize yarayabilir. Ancak SQL çok çok çoooook uzunsa bunu direkt VBA ortamında hazırlamak sıkıntılı olabilir, onun yerine SQL kısmını bir txt/sql uzantılı dosyasından okutmakta fayda var.

### Hayalet Protokol: Application.OnTime ile Scheduling işlemleri

Aşağıdaki kodlar benim günde yaklaşık 40 raporun kendiliğinden çalışmasını, refresh olmasını ve yeni günün tarihiyle kaydedilmesini, gerekiyorsa ilgili alıcılara maille iletimini sağlar. (NOT: Ben yerimde yokken departmana benden mailler geldiğinde arkadaşların "Volkan sen yokken senin hayaletin ortalıkta dolaşıp mailler atıyordu" demesinden kaynaklı olarak bu sistemin adını Hayalet Protokol olarak belirledim.)

Gerek konu gerek kodlar ilk başta karışık gelebilir. Anlaşılırlığı artırmak adına kodlardan hemen önce bir akış diyagramı da ekledim. Bununla beraber Udemy'deki eğitimime kaydolursanız(bu kitabı alanlara ücretsiz) orada görsel olarak da akışı takip edebilirsiniz. **Düzenli rapor üreten biriyseniz bu konuyu mutlaka anlamaya çalışın. Size ve bölümünüze sunacağı faydayı tahmin bile edemezsiniz.** En az 3-4 adamlık tasarruftan bahsediyorum. Şahsen tüm kitapta benim en önem verdiğim konu burasıdır.

**Ön bilgiler:**

* İki bilgisayarla çalışıyorum. Biri, sadece birkaç makronun çalıştığı PC'm, biri de schedule edilmiş işlerin ağırlıklı olarak çalıştığı laptop.
* Kodların bazısı her iki makinada da çalışıyor, bazısı sadece PC'de, bazısı da sadece laptopta
* PC'de çalışanlar genelde mesai saati dışında çalışıp bitiyor. Gün içinde çalışanlar da hep laptopta çalışıyor.
* Diyelim ki sicilim 12345. PC ismim B12345, laptop ismim L12345
* Raporların bir kısmı Veri Ambarı(kısaca DWH) tablolarına SQL atıyor, bazısı DWH'tan türeyen datamartlara, bazıları da online sistemlere. Online olanlar için herhangi bir "tablo dolmuş mu" kontrolü yapmıyorken, DWH ve ondan türeyen datamartlar için dünün datası dolmuş mu diye kontrol yapıyorum
* İlk kod bloğunda kapatOnMakroPC diye bir modül var. Bunun hikayesi şu. Her akşam laptopta(makroPC) Excel’i otomatik kapattırıyorum, ki güncel Personal.xlsb ve Schedule.xlsb dosyaları laptopa yedeklenebilsin. Böyelce kendi PC'mde shcedule ayarlarında bir değişiklik yaptığımda veya Personal.xlsb dosyama yeni bir prosedür eklediğimde laptopa da aynen geçmesini sağlıyorum ki işlerde aksama olmasın. Bunun için PC'de bu iki dosyayı ne zaman kaydetsem, bunların "Workbook\_AfterSave" eventi devreye giriyor ve dosyaların güncel halini ortak alana da kaydediyor. Laptopta da Windows Task Scheduler ile her gece yarısı önce yedeklemeyi yaptırıyorum, sonra Excel'i tekrar açtırıyorum
* Hem PC'de hem laptop'ta Personal.xlsb ve Schedule.xlsb dosyam XLSTART klasöründe yer alıyor, böylece Excel açılır açılmaz onlar da gizli olarak açılıyorlar
* Keza, her iki bilgisayarda da bilgisayar açılır açılmaz Excel'in de açılmasını sağlayan bir kurgu var. Bunu Windows ayarlarından siz de yapabilirsiniz.

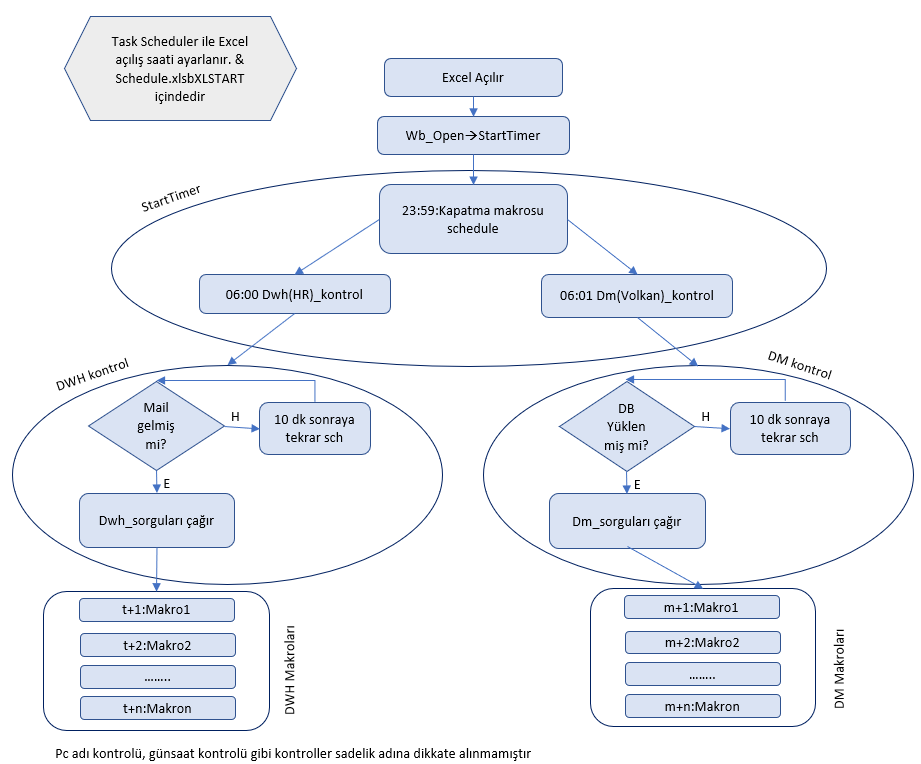
Bundan sonra süreç şu şekilde devam ediyor.

* Schedule dosyasının Workbook\_Open eventi devreye giriyor, ve bu event de StartTimer prosedürünü çalıştırıyor.
* Bu prosedür, Dwh(Veri ambarı) dolmuş mu yani güncel data yüklenmiş mi kontrolünü yapıyor.
* Dwh dolmadıysa 10 dk sonrasına kendisini tekrar schedule ediyor, dwh dolduysa "dwh raporları" isimli prosedürü çalıştırıyor
* “dwh raporları” içindeki tüm prosedürler sırayla çalıştırılıyor.
* Varsa diğer datamart kontrolleri de aynı süreçte yapılmaktadır.

**Akış şeması**

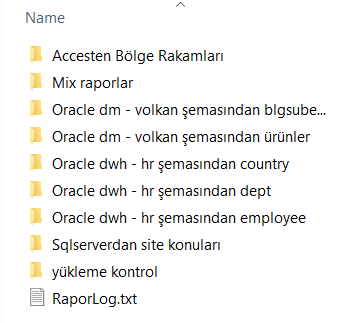
Bu şemayı Udemy kursum için hazırlamıştım. Videoda animasyonlu olarak teker teker uygun sırada açıldıkları için oradan bakmanızı da tavsiye ederim.

Aşağıdaki DWH(HR) diye geçen şey, bilgisayarıma kurduğum Oracle veritabanındaki HR şeması. Temsili olarak DWH, yani veri ambarı tabloları bu şemada olacak, Volkan şemasında da DM yani datamart olacak. Bunlardaki tablolar dolmuş mu diye kontrol edeceğiz.



Schedule dosyasını **githubdan(schedule.xlsb)** indirebilirsiniz. Açılır açılma çalışmasını engellemek için Workbook\_Open'ın başlangıcına geçici olarak Exit Sub satırı ekledim.

Şimdi bir de örnek klasörlere bakalım.



Aşağıdaki vereceğim tüm prosedürler **Schedule.xlsb** dosyamda bulunmaktadır. Şimdi, ilk olarak Workbook\_Open prosedürüne bakalım.

Private Sub Workbook\_Open()

ilksaat = Date + TimeSerial(23, 30, 0)

'Debug.Assert False

'PC'yi mesai başlangıç saatinden sonra açmışsak kontrol raporları hemen başlasın, diğerlerine gerek yok

If TimeValue(Now) > ilksaat Then

Application.OnTime Now + TimeSerial(0, 0, 10), Procedure:="dwh\_kontrol"

Application.OnTime Now + TimeSerial(0, 0, 20), Procedure:="dm\_kontrol"

Else 'gece 00:00'da programlandığı saatte açıldıysa

Call StartTimer

End If

End Sub

Şimdi de StartTimer prosedürünü inceleyelim. Bu, schedule işlemlerini ele aldığım ana prosedürdür.

Sub StartTimer()

'anapc veya makropc dışında bir bilgisayarda çalışırsa çıksın

If Environ("computername") <> volkanpc And Environ("computername") <> makropc Then Exit Sub

'hem anapc hem makropcde şunlar çalışsın

Application.OnTime Now + TimeSerial(24, 0, 0), Procedure:="StartTimer"

If Environ("computername") = volkanpc Then

Application.OnTime TimeValue("00:05:00"), Procedure:="eskilerisil"

Application.OnTime ilksaat, Procedure:="dwh\_kontrol"

Application.OnTime ilksaat + TimeSerial(0, 1, 0), Procedure:="dm\_kontrol"

Application.OnTime TimeValue("11:00:00"), Procedure:="AccessRaporu"

Application.OnTime TimeValue("11:30:00"), Procedure:="SQLServerRaporu"

Else 'makropc

'.....

Application.OnTime TimeValue("23:55:00"), Procedure:="kapatOnMakroPC" 'gece diğer pclere yedekleme yapılsın diye dosyayı kapatmamız gerekiyor

End If

End Sub

Şimdi de kontroller modülündeki prosedürlere bakalım. Burada da kalabalık olmaması adına tüm kontrol prosedürlerini koymadım, onun yerine iki farklı yöntemi göstermesi adına iki prosedürü verdim. İlkinde , BT ekiplerinden gelen "Tablo doldu" tarzındaki mailin varlığını kontrol ediyorum. İkincisinde ise farklı dinamiğe sahip olduğundan "Tablo doldu" maili gelmeyen bir datamart için bir tabloya SQL atıp kayıt dönüp dönmediğine bakıyorum.

'1.yöntem

Sub dwh\_kontrol()

Const raporad As String = "dwh\_kontrol"

Const adresKontrol As String = "yükleme kontrol\dwh\dwhyüklemekontrol - "

Dim outApp As New Outlook.Application

Dim ns As Outlook.Namespace

Dim myroot As Outlook.Folder, klasör As Outlook.Folder

Dim eItems As Outlook.Items

Dim eItem As Outlook.MailItem

Dim fso As New FileSystemObject

'Debug.Assert False

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "Girildi"

On Error GoTo hata

If Mevcutmu(adres & adresKontrol & Date - 1 & " Sonuçları.txt") Then

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "ZatenOK"

Call dwh\_raporları

Exit Sub

End If

AlertUpdatingEvent False, False 'normalde commentsiz olmalı

Set ns = outApp.GetNamespace("MAPI")

'hem benim hem serkanın pcsinde aynı klasör olduğunu varsayıyorum

Set myroot = ns.Folders("volkan.yurtseven@hotmail.com")

Set klasör = myroot.Folders("Kontrol Mailleri")

Set eItems = klasör.Items

Kriter1 = "[Subject]=""DWH yüklemesi tamamlanmıştır"""

Kriter2 = "[ReceivedTime]>'" & Date & "'" 'Tarihin saat kısmından dolayı > olmalı

Set kriterliler = eItems.Restrict(Kriter1).Restrict(Kriter2)

If kriterliler.Count < 1 Then 'maili bulamadıysak 10 dk sonra yeniden schedule ederiz

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "Reschedule"

Application.OnTime Now + TimeSerial(0, 10, 0), Procedure:="dwh\_kontrol"

Else 'mail varsa sürece devam

fso.CreateTextFile adres & adresKontrol & Date - 1 & " Sonuçları.txt"

Call MailBilgi("DWH Kontrolü", anaAlicilar)

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "OK"

Call dwh\_raporları

GoTo çıkış

End If

çıkış:

Set myroot = Nothing: Set klasör = Nothing

Set ns = Nothing: Set outApp = Nothing

Set eItems = Nothing: Set eItem = Nothing: Set fso = Nothing

AlertUpdatingEvent True, True

Exit Sub

hata:

AlertUpdatingEvent True, True

Set myroot = Nothing: Set klasör = Nothing

Set ns = Nothing: Set outApp = Nothing

Set eItems = Nothing: Set eItem = Nothing: Set fso = Nothing

Logger Now, Environ("computername"), raporad, "Kontrol", Err.Number, Err.Description

End Sub

Şimdi de datamartların kontrol prosedürü:

'2.yöntem

Sub dm\_kontrol()

Const raporad As String = "dm\_kontrol"

Const adresKontrol As String = "yükleme kontrol\dm\dmyüklemekontrol - "

'Debug.Assert False

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "Girildi"

AlertUpdatingEvent False, False 'normalde commentsiz olmalı

On Error GoTo hata

If Mevcutmu(adres & adresKontrol & Date - 1 & " Sonuçları.xlsb") Then

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "ZatenOK"

Call dm\_raporları

Exit Sub

End If

Workbooks.Open adres & adresKontrol & "Format.xlsb" 'workbook\_open kodu çalışacak

'veritabanı yüklenmediyse 10 dk sonraya reschedule

If Not Mevcutmu(adres & adresKontrol & Date - 1 & " Sonuçları.xlsb") Then

ActiveWorkbook.Close savechanges:=False

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "Reschedule"

Application.OnTime Now + TimeSerial(0, 10, 0), Procedure:="dm\_kontrol"

Exit Sub

Else 'başarılıysa devam edelim

ActiveWorkbook.Close

Kill "C:\geçici\geçici2.xlsb"

Call MailBilgi("DM Kontrolü", anaAlicilar)

Logger Now, Environ("computername"), raporad, "Kontrol", 0, "OK"

Call dm\_raporları

GoTo çıkış

End If

çıkış:

AlertUpdatingEvent True, True

Exit Sub

hata:

AlertUpdatingEvent True, True

End Sub

Şimdi de dwh\_raporları ve dm\_raporları prosedürlerine bakalım. Burada raporlar çalışma sırasına sokuluyor.

Sub dwh\_raporları()

'Debug.Assert False

Application.OnTime Now + TimeSerial(0, 0, 10), Procedure:="dwhrapor1"

Application.OnTime Now + TimeSerial(0, 0, 20), Procedure:="dwhrapor2"

Application.OnTime Now + TimeSerial(0, 0, 30), Procedure:="dwhrapor3"

End Sub

Sub dm\_raporları()

'Debug.Assert False

Application.OnTime Now + TimeValue("00:00:10"), Procedure:="dmrapor1"

Application.OnTime Now + TimeSerial(0, 0, 20), Procedure:="dmrapor2"

Application.OnTime Now + TimeSerial(0, 0, 30), Procedure:="mixrapor1"

End Sub

Şimdi de online olan, ve dolayısıyla "dolmuş mu?" kontrolüne gerek olmayan prosedürlere bakalım.

Sub AccessRaporu()

rprLog = "AccessBlg"

rprMail = "Accesten Bölge Rakamları"

yol = "Accesten Bölge Rakamları\Accesten Bölge Rakamları - "

kontrol1 = vbNullString

alıcı = diğerAlıcılar

RutinKod rprLog, rprMail, yol, kontrol1, alıcı

End Sub

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

Sub SQLServerRaporu()

rprLog = "dwh1"

rprMail = "dwh 1 raporu"

yol = "Sqlserverdan site konuları\Sqlserverdan site konuları - "

kontrol1 = vbNullString

alıcı = diğerAlıcılar

RutinKod rprLog, rprMail, yol, kontrol1, alıcı

End Sub

Şimdi dwh raporlarından birine ait prosedüre bakalım; şuan hala schedule.xlsb içindeyiz.

Sub dwhrapor1()

'Debug.Assert False

rprLog = "dwh1"

rprMail = "dwh 1 raporu"

yol = "Oracle dwh - hr şemasından country\Oracle dwh - hr şemasından country - "

kontrol1 = "yükleme kontrol\dwh\dwhyüklemekontrol - " & Date - 1 & " Sonuçları.txt"

alıcı = dwhRapor1Alıcıları

RutinKod rprLog, rprMail, yol, kontrol1, alıcı

End Sub

Bazı raporlar ise hem dwh hem dm tablolarının yüklenmesini gerekebilir.

Sub mixrapor1()

rprLog = "mix"

rprMail = "Mix raporu"

yol = "Mix raporlar\Mix rapor1 - "

kontrol1 = "yükleme kontrol\dm\dmyüklemekontrol - " & Date - 1 & " Sonuçları.xlsb"

kontrol2 = "yükleme kontrol\dwh\dwhyüklemekontrol - " & Date - 1 & " Sonuçları.txt"

alıcı = diğerAlıcılar

RutinKod rprLog, rprMail, yol, kontrol1, alıcı, kontrol2

End Sub

Şimdi de bu son iki kodun sonundaki RutinKod prosedürüne bakalım. Gördüğünüz üzere bu prosedür, oldukça çok parametre alıyor. Bunları anlayabileceğinizi düşünüyorum. Son parametre opsiyonel olup, mixrapor1'deki gibi, iki rapor kaynağından(dwh ve dm) da beslenen raporlar için kullanılmaktadır.

Sub RutinKod(ByVal raporLogger As String, ByVal raporMail As String, ByVal folderfile As String, \_

ByVal kontrol1 As String, ByVal alıcılar As String, \_

Optional ByVal kontrol2 As Variant)

'Debug.Assert False

'normalde kontrol dosyalarını kontrole gerek yok. ihtiyaden kontrol yapıyoruz

If Not Mevcutmu(adres + kontrol1) Then Exit Sub

If Not IsMissing(kontrol2) Then

If Not Mevcutmu(adres + kontrol2) Then Exit Sub

End If

'Daha önce oluştuysa çıkalım

If Mevcutmu(adres & folderfile & Date - 1 & " Sonuçları.xlsb") Then Exit Sub

On Error GoTo hata

AlertUpdatingEvent False, False

Workbooks.Open adres & folderfile & "Format.xlsb" 'çalışmaya başlayacak

ActiveWorkbook.Close

Kill "C:\geçici\geçici2.xlsb"

Call MailBilgi(raporMail, alıcılar)

Logger Now, Environ("computername"), raporLogger, "OK", 0, "rapor başarıyla çalıştı"

AlertUpdatingEvent True, True

Exit Sub

hata:

AlertUpdatingEvent True, True

Logger Now, Environ("computername"), raporLogger, "Hata", 0, Err.Description

End Sub

Şimdi schedule.xlsb'den çıkıp, veritabanı sorgulaması yapılacak rapor dosyasındaki koda bakalım.

Private Sub Workbook\_Open()

Const raporad = "dwh-country"

Application.Run "schedule.xlsm!wbopen", raporad

End Sub

Gördüğünüz üzere kod çok kısa ve bizi tekrar schedule.xlsb'ye yönlendiriyor. Bunu böyle yaptık, çünkü yaklaşık 40 rapor içinde her birine aynı kodu yazmanın anlamı yok. Onun yerine bi kere yazıyoruz, ve her defasında bu kodu, uygun parametreyle(rapor adı) çağırıyoruz. Hadi şimdi bir de bu son koda bakalım.

Sub wbopen(ByVal raporad As String, Optional ptrefresh As Boolean = False)

Dim lo As ListObject

Dim con As WorkbookConnection

Dim ws As Worksheet

Dim pt As PivotTable

On Error GoTo hata

'Debug.Assert False

'yetkili kişiler dışında direkt açan kişilere uyarı çıksın

If Environ("username") <> "Volkan" And Environ("username") <> "Serkan" Then

If ActiveWorkbook.ReadOnly = False Then

'Date - 1'de bazen manuel değişiklikler yapmak gerekebiliyor

If InStr(ActiveWorkbook.Name, "Format") > 0 Or InStr(ActiveWorkbook.Name, Date - 1) > 0 Then

Logger Now, Environ("computername"), raporad, "YetkisizGiriş", 0, ""

MsgBox "Dosyaya doğrudan değil, Kokpit formundan ulaşınız"

ActiveWorkbook.Close savechanges:=False

Exit Sub

Else 'date - 2 ve öncesini açarsa sorun yok, ama burda koddan çıksın

Exit Sub

End If

Else 'readonly açılmazsa yani kokpitten açılmışsa sorun yok, ama burda koddan çıksın

Exit Sub

End If

End If

'volkan ve serkansa devam ediyoruz. İçinde refreshlenebilir bir Table yoksa çıksın

'Excel versiyonunuza göre Queries soruglaması desteklenmeyebilir, o zaman bunu kaldırırsınız

If ActiveWorkbook.Queries.Count = 0 And ActiveWorkbook.Connections.Count = 0 Then Exit Sub

Logger Now, Environ("computername"), raporad, "WBOpen", 0, "WB giriş yapıldı"

isim = Left(ActiveWorkbook.Name, Len(ActiveWorkbook.Name) - 14) 'Neden 14, "dd.mm.yyy - Sonuçları"nın uzunluğu

'tüm connectionlar refresh

For Each con In ActiveWorkbook.Connections

If con.Type = xlConnectionTypeODBC Then

con.ODBCConnection.BackgroundQuery = False

Else

con.OLEDBConnection.BackgroundQuery = False

End If

con.Refresh

Next con

For Each ws In ActiveWorkbook.Sheets

ws.Activate

If ws.ListObjects.Count > 0 Then 'refresh olacak sayfalarda

'refresh sonucu data gelmiyorsa çıkalım

If Range("a1").End(xlDown).Row = 2 Then GoTo iptal '??????dosya açık kalmıyor mu????

'refresh sonucu data geliyorsa unlink yapalım

For Each lo In ws.ListObjects

lo.Unlink

Next lo

End If

'pivot refresh

If ptrefresh = True Then

For Each pt In ws.PivotTables

pt.RefreshTable

Next pt

End If

Next ws

hedef = ActiveWorkbook.Path & "\" & isim & " - " & Date - 1 & " Sonuçları.xlsb"

saveas\_işlemi\_inworkbook isim, hedef

Logger Now, Environ("computername"), raporad, "WBOpen", 0, "OK"

iptal:

'???wb close??

Exit Sub

hata:

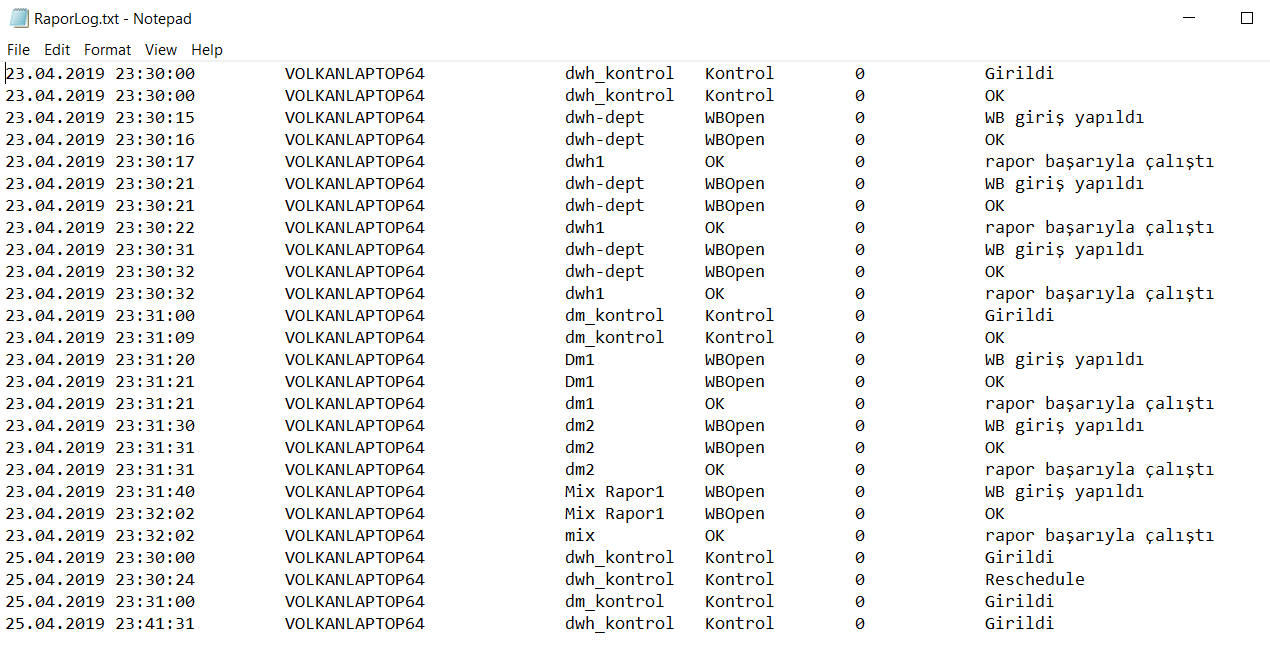
Logger Now, Environ("computername"), raporad, "WBOpen", Err.Number, Err.Description

ActiveWorkbook.Close savechanges:=False

End Sub

Bu kod da çalışınca sıra, schedule kuyruğundaki diğer kodlara geçer ve işlemler aynen devam eder.

Rapor log dosyamızın içine de bakalım.



Schedule.xlsb dosyamızdaki ThisWorkbook modülünde bir kodumuz daha var, ona da bakalım. Açıklamasını en başta yapmıştık.

Private Sub Workbook\_AfterSave(ByVal Success As Boolean)

yol = "C:\inetpub\wwwroot\aspnettest\Excelefendiana\Ornek\_dosyalar\Makrolar\makroyedek\"

FileSystem.SetAttr yol & Me.Name, vbNormal

fso.CopyFile Me.FullName, yol

FileSystem.SetAttr yol & Me.Name, vbReadOnly

End Sub

Son olarak Schedule.xlsb dosyamdaki diğer kodlar ise şöyle;

Sub AlertUpdatingEvent(a As Boolean, u As Boolean, Optional e As Boolean = True)

With Application

.DisplayAlerts = a

.ScreenUpdating = u

.EnableEvents = e

End With

End Sub

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

Function Mevcutmu(adres As String) As Boolean

On Error GoTo hata

If GetAttr(adres) >= 0 Then Mevcutmu = True

Exit Function

hata:

Mevcutmu = False

End Function

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

Sub kapatOnMakroPC()

Application.Quit

End Sub

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

Sub Logger(ByVal tarih As String, ByVal pc As String, ByVal rapor As String, ByVal logtipi As String, ByVal hatano As Integer, \_

ByVal açıklama As String)

Dim dosya As String

Dim dosyano As Integer

dosyano = FreeFile()

dosya = adres & "RaporLog.txt"

Open dosya For Append As #dosyano

Print #dosyano, tarih, pc, rapor, logtipi, hatano, açıklama

Close #dosyano

End Sub

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

Sub MailBilgi(ByVal rapor As String, ByVal alıcılar As String)

Dim outApp As New Outlook.Application

Dim outMail As MailItem

Set outMail = outApp.CreateItem(olMailItem)

With outMail

.Subject = rapor & " raporu yüklemesi"

.Body = "Güncel " & rapor & " raporu oluşmuştur"

.To = alıcılar

.Send

End With

End Sub

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

Sub saveas\_işlemi\_inworkbook(ByVal dosyaismi As String, ByVal hedef As String)

ActiveWorkbook.SaveAs "C:\geçici\geçici.xlsb" 'dosya tipine göre kontrol de yapılabilir

kaynak = ActiveWorkbook.FullName

ActiveWorkbook.SaveAs "C:\geçici\geçici2.xlsb" 'dosya tipine göre kontrol de yapılabilir

kaynak2 = ActiveWorkbook.FullName

FileCopy kaynak, hedef

Kill kaynak

'kaynak 2 nerde kullanım?

End Sub

#### Özet

Tüm bu süreç kendini besleyen otomatik bir süreçtir. Başarılı bir şekilde kurulması durumunda müthiş bir verimlilik sağlanacaktır.

Yukarıda belirttiğim üzere, konu biraz karmaşık olup, daha verimli bir şekilde algılanabilmesi için Udemy eğitimimden yararlanmanızı tavsiye ederim.

#### Scheduled program akışını kesintiye uğratan unsurlar

Bazen schedule edilmiş dosyalarınızı gün içinde de açmanız ve kullanmanız gerekebilir. O yüzden belli saatlerde açıldığında faklı bir davranış göstermesini isteyebilirsiniz. Keza dosyayı açan sizseniz başka, diğer kişilerse başka şekilde davranmasını isteyebilirsiniz.

Bu davranış şekillerinden biri de kullanıcıya bir mesaj göstermek olabilir. Eğer programın bir yerinde mesaj verirseniz, bu mesaj kutusundaki bir düğmeye basılana kadar ekranda kalır, bu da takip eden schedule kodlarınızın çalışmasını engellemiş olur. O yüzden MsgBox'lı uyarılarınızı Schedule zamanı içinde **size kesinlikle çıkarmamalı**, sadece diğer kişilere çıkarmalıdır. Gerekiyorsa kendinize bir maille bilgilendirme yaptırabilir veya Statusbara mesaj yazdırabilir veya **Dosya Okuma Yazma işlemleri** konusunda gördüğümüz **Logger** prosedürüyle kayıt altına alabilirsiniz.

Benim kullandığım yöntem şudur. Kullanıcılara UserForm kullanarak Kokpit diye bir arayüz oluşturdum ve bunu onlara verdiğim Add-in'deki menüye yerleştirdim. Kullanıcılar ilgili dosyayı kokpitten açtıklarında, dosyanın readonly açılmasını sağlıyorum. Normalde herkes hep dünün raporunu açmak ister , o yüzden kokpitten de sadece dünün raporlarını açabiliyorlar. Daha eski raporları açmak için ortak alana gidip klasörden manuel açmaları gerekiyor. Şimdi öyle bir kod yazmam gerekiyor ki, dünkü raporu ortak alandan açamasınlar, zira bir rapor gün içinde iki defa farklı zamanlarda refresh ediliyor; aynı zamanda ana format dosyamın da açılmaması lazım, zira onu açarlarsa schedule makrom takılmaktadır. O yüzden dünün dosyasını veya Format dosyasını klasörden manuel açmaya çalışırlarsa uyarı verdiriyorum. Şimdi aşağıdaki kodu inceleyelim.

Private Sub Workbook\_Open()

On Error GoTo hata

If Environ("UserName") <> 12345 Then 'Dosya benim dışında biri tarafından açıldıysa

If Right(Me.Name, 11) = "Format.xlsm" Or (Right(Me.Name, 25) = Date - 1 & " Sonuçları.xlsm" And Not Me.ReadOnly) Then

MsgBox "Dosyayı ortak alandan değil, Kokpitten açın."

Me.Close savechanges: =False

End If

End If

If Environ("computername") <> "L12345" Then Exit Sub 'dosyayı bazen PC'mde açıp SQL'inde veya genel sayfa düzeninde güncelleme yapmam gerekiyor. Açılır açılmaz da refresh olmasın diye laptop dışında açılırsa yani PC'mde açılırsa kalan kodlar çalışmasın istiyorum

'Diğer kodlar

End Sub

Bir diğer önemli husus da, programınızda son derece ince ayarlanmış bir hata yönetimi mekanizması olması gerekir. Detaylar için hata yakalama bölümünü iyice özümsemiş olmanızı tavsiye ederim. Schedule edilmiş programlarda hataları hata mesajı olarak vermek yerine bir Logger prosedürüyle bir dosyaya yazdırmayı tercih edebilirsiniz.

### Yöntemlerin Karşılaştırılması

* QueryTable’lar oldukça hızlıdır ve özellikle VBA bilmeyenler için bile dataya ulaşmada kolaylık sağlar. ADO için mutlaka VBA bilmek gerekir. Ancak ADO’nun da QueryTable’ya göre avantajları vardır.
* ADO ile aynı anda birden çok DB(Ör: SQL Server ve Access) sorgulanabilirken QueryTable ile tek seferde sadece tek DB sorgulanabilir.
* ADO ve DAO ile gelen data formatsız bir data olacaktır. Bunu formatlı hale getirmek için ek kod yazmak gerekir. Halbuki bir WorkbookConnection kurulduğunda(text/web değilse) Table formatı otomatik gelmektedir.
* Ayrıca manuel refreshlemek gereken durumlarda ListObject yine bir kolaylık sağlamakta iken ADO/DAO için kod sayfasına girip kodu manuel çalıştırmak gerekmektedir. Belki bunun için sayfaya bir buton konularak kodun refresh mantığıyla çalıştırılması sağlanabilir.

### Çeşitli Örnekler

#### Bir grup dosyadaki şifreleri (ve/veya userları) toplu olarak değiştirme

Örnek kodumuz uzun olup **githubdan(toplu\_sifre\_degisme)** indirebilirsiniz.

Bu kod özellikle benim gibi günde 30-40 civarı dosyayı schedule etmiş kişiler için çok faydalı olacaktır. Oracle/DB2 gibi veritabanlarına bağlanıp veri çekiyorsanız çok büyük ihtimalle ayda bir şifrenizi değiştirmeniz gerekmektedir. Excel içine gömdüğünüz şifre bilgilerinin de aynı şekilde değişmesi gerekir. Sadece şifre değil, bazen schedule işlemini başka birine devretmeniz de gerekebilir, bu durumda user bilgilerinin de değişmesi gerekir. İşte böyle bir durumda bu 30-40 dosyayı tek tek açıp, kimisinde 5-6 connection da bulunabilir, tek tek user/şifre bilgisi değiştirmek tüm gününüzü alabilir, zira tüm sorguların çalışması gerekecektir. İşte bu aşağıdaki kod, sadece birkaç dakika içinde tüm dosyalarda gerekli değişiklikleri yapacaktır.(Dosyaları kaydedip kapatma işlemini de prosedürün sonuna siz ekleyin)

Burda toplamda üç ayrı prosedür bulunuyor. Bunlardan biri, diğer iki prosedür tarafından çağrılan bir Function. Zaten iki kez kullanılması gerektiği için bunu Function yaptım. (Bu Function'da çoklu değer döndürmek için Collection kullandım). Yaptığımız değişiklikler olmuş mu diye görmek için baktığımız üçüncü prosedüre gerek duymazsanız, Function içindeki kodu ana prosedüre yedirebilirsiniz(isterseniz tabi).

Kodun içinden de gördüğünüz üzere, oracle bağlantısı olanlar için ayrı, DB2 bağlantıları için ayrı olmak üzere ana prosedürü toplamda 2 kez çalıştırıyorum. Siz, kaç farklı veritabanına bağlanıyorsanız o kadar çalıştırmanız gerekecek. (Database'lerin illa farklı makinalarda olması gerekmiyor, Oracle üzerinde bulunan 3 farklı database'e de bağlanmanız durumunda yine 3 kez çalıştırmanız gerekir.)

(NOT: Her ne kadar ADO konusu altındaki [şifre güvenliği](file:///E:\OneDrive\Uygulama%20Geliştirme\web%20sitelerim\Publishlerim\EE\precompile\Konular\VBAMakro\DigerUygulamalarlailetisim_VeriTabani-ConnectionListObjectveQueryTable.aspx#passsecurity) kısmında şifrenizi connection string içinde yazmamanızı önerdiysem de, olur da böyle birşey yaptıysanız diye bu örneği koydum. Aslında şifreyi tek bir yerden okumanız durumunda sadece orada değiştirmeniz yeterli olacaktır ve yukarıdaki zahmete de girmemiş olursunuz. Dediğim gibi bir nedenden bu yolu benimsediyseniz diye bu örneği koydum)

#### Açık olan dosyalardaki Connectionların Şifre/user ve SQL bilgilerini değiştirme

Diyelim ki içinde bir veya daha fazla connection olan bir veya daha fazla dosya açık durumda. Bir önceki örnekte dosyalar kapalıydı, bu örnekte bu sefer açık durumdalar. Bir diğer fark da bu sefer SQL değişikliği de yapmak isteyebilirsiniz. Mesela açık olan tüm dosyalarda 1 gün öncesinin datasını çeken SQL’ler(“sysdate-1” satırını içeren) olduğunu düşünün. Bunların hepsini 2 gün öncesinin datasını gösteren SQL’e(sysdate-2) çevirmek isteyebilirsiniz. Bunu manuel yaparsanız yine şifre değişimde olduğu gibi SQL değişiminde de tüm bağlantıların çalışmasını beklemek zorundasınızdır. İşte bu aşağıdaki kod ile beklemeden tek seferde hızlıca gerekli kod değişikliklerini yapabilirsiniz.

Kodun yeterince açık olduğunu düşündüğüm için ilave bir açıklama yazmıyorum. Anlaşılmayan birşey olması durumunda aşağıdaki Yorum bölümüne sorunuzu yazabilirsiniz.

Sub ConString\_veya\_SQL\_degistir()

'açık olan bir dosyada SQL veya connectionstring(genellikle şifre ve user için) değiştirir

On Error GoTo çıkış

Dim wB As Workbook

Dim cn As WorkbookConnection

Dim eski As String

Dim yeni As String

Dim contype As Byte

Dim neyi As Byte

Do

neyi = InputBox("SQL için 1, connection string için 2 girin")

Loop Until neyi < 3

eski = InputBox("Eski değeri girin")

If eski = "" Then Exit Sub

yeni = InputBox("Yeni değeri girin")

If yeni = "" Then Exit Sub

N = 0

dosyalar = ""

For Each wB In Application.Workbooks

If Windows(wB.Name).Visible = True Then 'Personal.xlsb gibi gizli dosyalarda çalışmasına gerek yok

If wB.Connections.Count > 0 Then 'içinde bağlantı olan dosyalarda yapsın

For Each cn In ActiveWorkbook.Connections

'sadece ODBC ve OLEDB'lerde yapsın, diğerlerinde bir işlem yapmasın

If cn.Type = xlConnectionTypeODBC Or cn.Type = xlConnectionTypeOLEDB Then

If neyi = 1 Then 'SQL-comnandtext

If cn.Type = xlConnectionTypeODBC Then

cn.ODBCConnection.CommandText = Replace(cn.ODBCConnection.CommandText, eski, yeni)

Else

cn.OLEDBConnection.CommandText = Replace(cn.OLEDBConnection.CommandText, eski, yeni)

End If

Else 'con string

If cn.Type = xlConnectionTypeODBC Then

cn.ODBCConnection.connection = Replace(cn.ODBCConnection.connection, eski, yeni)

Else

cn.OLEDBConnection.connection = Replace(cn.OLEDBConnection.connection, eski, yeni)

End If

End If

End If

Next cn

N = N + 1

dosyalar = wB.Name & ";" & dosyalar

End If

End If

Next wB

MsgBox N & " adet dosyada şifre değişti. "

Workbooks.Add

Range("a1") = dosyalar

Exit Sub

çıkış:

MsgBox "Bir hata oluştu"

End Sub

#### Tüm workbookların refresh olmasını sağlama

Diyelim ki, o anda açık olan tüm dosyaların refresh olmasını istiyorsunuz. Normalde Data menüsündeki **RefreshAll** sadece açık olan dosyadaki bağlantıları refreshler. Biz ise açık olan tüm dosyalarda bunu yapmak istiyoruz. Tabi gizli olan Personal.xlsb tarzı makro dosyalarımız hariç. Bunun için gerekli kod aşağıdaki gibidir.

Sub tum\_wblarda\_tum\_connectionlar\_refresh()

On Error GoTo çık

For Each wB In Application.Workbooks

If Windows(wB.Name).Visible = True Then

If wB.Connections.Count > 0 Then

Debug.Print wB.Name

For Each cn In ActiveWorkbook.Connections

Debug.Print cn.Name

bBackground = cn.ODBCConnection.BackgroundQuery

cn.ODBCConnection.BackgroundQuery = False

cn.Refresh

cn.ODBCConnection.BackgroundQuery = bBackground

Next cn

End If

End If

Next wB

Exit Sub

çık:

MsgBox "hata"

End Sub

## Webden veri çekme

Web ortamından iki ana kategoride veri çekilebilir. Bunlardan birisi bir sayfanın belirli HTML elemanlarına bakıp ihtiyacımız olan kısmını almak(bu işleme parse etmek denir), bir diğeri de bir XML dokümanını(web serviceler dahil) almak şeklindedir. Bu sayfada sadece ilk yönteme bakacağız, ikinci yöntemi araştırmayı size bırakıyorum.

**Githubdan(webdenveri.xlsm)** örnek uygulamayı indirebilirsiniz.

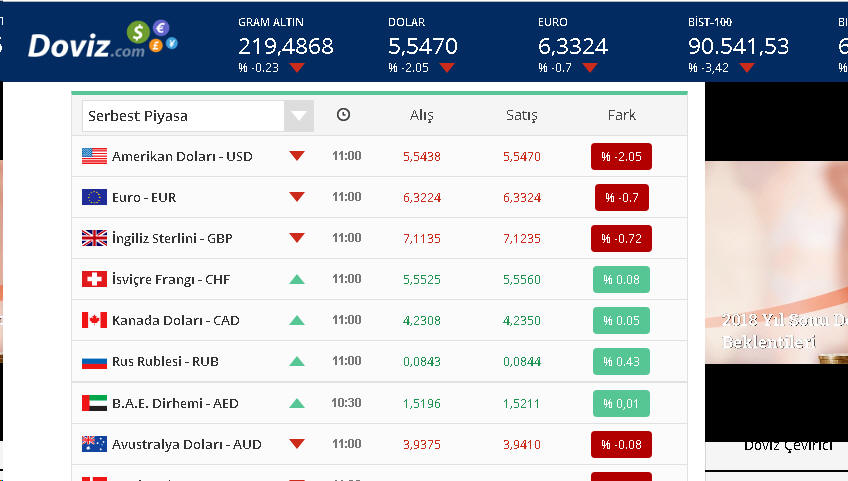
**ÖNEMLİ NOT:** Bu örnek için verilen kod zaman içinde geçerliliğini yitirmiş olabilir. Zira ilgili sayfanın tasarımcısı HTML kodlarını değiştirmiş olabilir. Ana mantığı anladıktan sonra gerekli değişiklikleri yapabileceğinizi düşünüyorum.

### HTML elemanlarını parse etmek

Hemen belirtmek isterim ki, bu yöntemi uygulamak için biraz da olsa Html ve javascript(bazen de css) bilgisi gereklidir. Bunlara aşina değilseniz uygun bir tutorial sitesinden(w3school olabilir) temelleri almanızda fayda var.

Web'den tüm bir sayfa içeriğini VBA ile alabilmeniz mümkün olsa da biz bu kitapta daha çok verilerle ilgilendiğimiz için veri okumayla ilgili kısma yoğunlaşacağız. Verilerin bulunduğu HTML elemanları da büyük çoğunlukla **table** elemanları içinde bulunmaktadır.

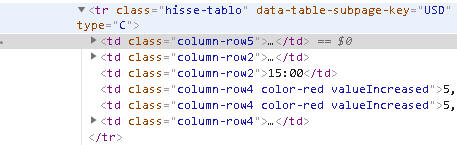
Lafı uzatmadan hemen örneğimize geçelim. Diyelim ki aşağıdaki sitede([https: //kur.doviz.com/](https://kur.doviz.com/)) bulunan döviz tablosunu Excel içine almak istiyoruz.



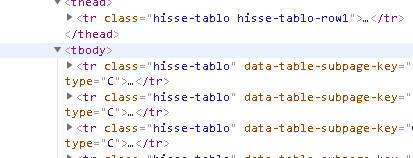
Şimdi bu noktada ilk yapmamız gereken tabloda bir yere gelip sağ tıklamak ve "**İncele**" demek olacaktır.



Bunu yaptığımızda browser penceremizde bir bölme açılır ve seçtiğimiz kısma konumlanarak onun elemanlarını bize gösterir.

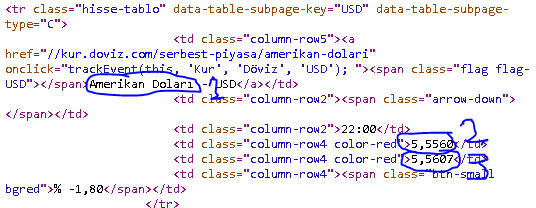


Bu kısmı biraz kurcalarsak **thead** ve **tbody** kısımlarını görürüz. thead kısmında ilgili listenin başlıkları yazmakta. tbody kısmında ise veriler bulunmakta.(Daha geniş bir görüntüyü, sağ tıkladıktan sonra İncele yerine **Sayfa kaynağını görüntüle** seçeneği ile görebilirsiniz)



Gördüğünüz üzere, bizim hedef kitlemiz "**hisse-tablo**" class'ına sahip olan "tr" tag'leri. Yanız thead içinde de hisse-tablo var bir tane(her ne kadar tam class adı "hisse-tablo hisse-tablo-row1" görünse de). Dolayısıyla biz ilk "hisse-tablo" class'lı elemanı değil sonrasındakilere bakmalıyız.

Şimdi üstteki resimlerden belli olmadığı için tüm sayfa kaynağından bakacak olursak, bu **tr** tag'inin açılmış halini daha açıkça görebiliriz. Bunların içinde aşağıdaki gibi **td** taglerini görüyoruz. Bizim ihtiyacımız olan bunlardan döviz adı, alış ve satış fiyatları; yani sırayla 1, 4, ve 5. elemanlar. Javascriptte indexler 0'dan başladığı için  0, 3 ve 4. elemanlar. Bu td elemanları tr elemanlarının bir alt seviyesi olduğu için bunlara tr'nin **çocukları** denir, yani **children** özelliği ile bunlara ulaşılabilir.



Şimdi kod yazımı için hazırlığımızı yapalım.

Kurgumuz aşağıdaki gibi olacak. B1 hücresinden döviz kurlarının kaynağını seçeceğiz.(Serbest piyasa mı yoksa başka bir kurumun fiyat bilgileri mi diye). Buna göre dönen sonuç A5-C5 range'inden aşağıya doğru gelecek. Yani bir worksheet\_change event'i söz konusu.



İlk olarak bir dictionary'ye ihtiyacımız olacak. Bunu B1'den seçilen değerin karşılığı olarak Url'ye bir ek yapmak için kullanacağız. Örneğin B1'de kaynak olarak "Serbest Piyasa" seçilirse url eki Serbest-Piyasa oluyor. Bunun için global geçerli olması gereken bir dict değişkenini Module1 içinde tanımladım. Buna hem workbook\_open eventinden hem de worksheet\_change eventi içinde erişeceğiz. Workbook\_open içinden dictionary'yi dolduruyoruz ve B1 hücresine data validation yapıyoruz.

'Module1 içeriği

Public dict As New Scripting.Dictionary

'ThisWorkbook içeriği

Private Sub Workbook\_Open()

dict.Add "Serbest Piyasa", "Serbest-Piyasa"

dict.Add "Akbank", "Akbank"

dict.Add "Denizbank", "Denizbank"

dict.Add "Merkez Bankası", "Merkez-Bankasi"

For Each k In dict.Keys

liste = liste & "," & k

Next k

With Range("b1").Validation

.Delete

.Add Type: =xlValidateList, AlertStyle: =xlValidAlertStop, Operator: = \_

xlBetween, Formula1: =Mid(liste, 2, Len(liste) - 1)

.IgnoreBlank = True

.InCellDropdown = True

.InputTitle = ""

.ErrorTitle = ""

.InputMessage = ""

.ErrorMessage = ""

.ShowInput = True

.ShowError = True

End With

End Sub

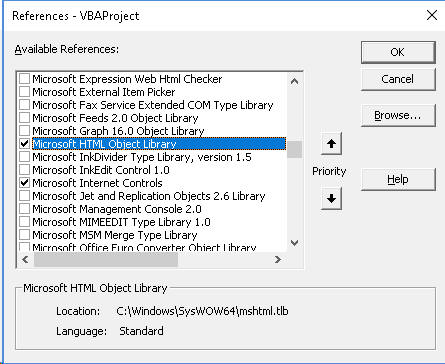
Kodun bundan sonraki kısmı iki ayrı yöntemle yapılabilir. Biz, Internet Explorer yöntemini kullanacağız, HTTP yöntemini araştırmayı size bırakıyorum.

IE ile kod yazımı

Bunun için Internet Explorer'ın(IE) bir örneğini yaratacağız, bunu gerçekten açmıyoruz tabi, sadece bellekte açıyoruz. O yüzden IE'nin nesne modeline ihtiyacımız var. (Bu yöntem için PC'nizde IE olması gerektiği aşikar.)

Ayrıca HTML elemanları ile çalışacağımız için HTML nesne modeline de ihtiyacımız bulunuyor.

Bunları eklemek için **Tools>Reference** üzerinden aşağıdaki libraryler eklenir:



Worksheet\_change eventinin kodu da aşağıdaki gibi olacaktır. Kod içinde gerekli tüm açıklamalar bulunmakta.

Private Sub Worksheet\_Change(ByVal Target As Range)

Dim IE As InternetExplorer

Dim elements As IHTMLElementCollection 'birçok tr tagini tutacak olan html collectionımız

Dim url As String

Dim url\_ek As String

Dim r As Integer

On Error GoTo hata

If Not Target.Address = "$B$1" Then Exit Sub 'sadece B1'e tıklanırsa tetiklensin

Application.EnableEvents = False 'data çekilirken recursive şekilde tetiklenme olmasın diye eventleri geçici olarak pasifleştiriyoruz

Application.StatusBar = "Lütfen bekleyiniz..."

url\_ek = dict([B1].Value)

[b2].ClearContents

başlangıç = Timer 'süre ölçmek istersek diye koydum

'önce eski veriyi temizleyelim

Range("Table1").Select

Selection.ClearContents

Set IE = New InternetExplorer

IE.Visible = False

url = "https: //kur.doviz.com/" & url\_ek

IE.navigate url 'ilgili sayfayı bellekte açıyoruz

'Bu kısım önemli, IE'yi bellekte açar açmaz sayfa yüklemesi hemen olmaz

Do While IE.Busy Or IE.READYSTATE <> 4 'sayfa yüklemesi tamamlanana kadar bekliyoruz

DoEvents

Loop

Application.StatusBar = "Veri Çekiliyor..." 'veri çekilmeye başlandığında statusbarı güncelliyoruz

Set elements = IE.document.getElementsByClassName("hisse-tablo") 'classı "hisse-tablo" olan tüm elemanları elements değişkenine atıyoruz

'Application.Wait (Now + TimeValue("00: 00: 05")) 'eğer eksik veri geliyorsa tam olarak belleğe alamıyordur, bunun için gereken süre kadar bekleriz, burayı duruma göre sizin ayarlamanız gerekecek

r = 5 'başlangıç satırı

For i = 1 To elements.Length - 1 '0'dan değil 1den başlıyoruz, çünkü başlığın olduğu kısmı atlıyoruz

If elements(i).ID = "linkUnit" Then GoTo atla 'RUS rublesindeki sonraki çizgiyi atlıyoruz

Cells(r, 1).Value = elements(i).Children(0).innerText 'döviz adı, 0 indeksli yani 1. eleman

'Küsuratlarda problem çıkmaması için virgülleri nokta ile replace ediyoruz

Cells(r, 2).Value = Replace(elements(i).Children(3).innerText, ",", ".") 'alış fiyatı 3 indeksi yani 4.eleman

Cells(r, 3).Value = Replace(elements(i).Children(4).innerText, ",", ".") 'Satış fiyatı 4 indeksi yani 5.eleman

r = r + 1

atla:

Next i

bitiş = Timer

Debug.Print bitiş - başlangıç 'geçen süreyi yazdırıyoruz

IE.Quit 'ilgili sayfada reklam v.s varsa arka planda müzik çalmaya devam eder, bu satırla arka planda açık olan Internet Exlporer'dan çıkarız ve reklam müziği sona erer

Set IE = Nothing: Set htmldoc = Nothing: Set elements = Nothing 'ilgili objeleri bellekten atıyoruz

Range("Table1[[Alış]: [Satış]]").Select

Selection.NumberFormat = "0.00"

[lastruntime].Value = Now 'güncelleme zamanını yazıyoruz

[lastruntime].Select

Application.StatusBar = "İşlem tamam"

Application.EnableEvents = True 'Enableevent özelliğini tekrar aktive ediyoruz

Exit Sub

hata:

Set IE = Nothing: Set htmldoc = Nothing: Set elements = Nothing

Application.StatusBar = ""

MsgBox "Bi hata oluştu" + Err.Description

Application.EnableEvents = True

End Sub

For Next döngüsünü aşağıdaki gibi de yapabilirdik, ancak bu sefer ilk If kontrolünü tüm döngü boyunca yapmak durumunda kalırdık, ki bu da performansı yoran bir işlem olurdu, o yüzden yukarıdaki yöntem daha hızlıdır. Ama bu tür kontrollerin gerekmediği durumlarda For Each döngüleri daha pratik olmaktadır.

'En başta da şu tanım yapılmalıdır

Dim element As IHTMLElement

For Each element In elements

If element.className = "hisse-tablo hisse-tablo-row1" Then GoTo atla 'başlığın olduğu kısmı atlıyoruz

If element.ID = "linkUnit" Then GoTo atla 'RUS rublesindeki sonraki çizgiyi atlıyoruz

Cells(r, 1).Value = element.Children(0).innerText 'döviz adı, 0 indeksli yani 1. eleman

Cells(r, 2).Value = Replace(element.Children(3).innerText, ",", ".") 'alış fiyatı 3 indeksi yani 4.eleman

Cells(r, 3).Value = Replace(element.Children(4).innerText, ",", ".") 'Satış fiyatı 4 indeksi yani 5.eleman

r = r + 1

atla:

Next element

Gördüğünüz üzere, html elemanlarını parse ederken çeşitli metodlar ve özellikler kullanırız. Bunlar özetle aşağıdaki gibi olup özellikle bunları araştırmanız ve öğrenmenizi tavsiye ederim.

**getElementsByTagName**: Çok elemandan oluşan bir  IHTMLElementCollection collection'ı döndürür

**getElementsByClassName**: Çok elemandan oluşan bir  IHTMLElementCollection collection'ı döndürür

**getElementById**: Tek bir IHTMLElement elemanı döndürür

**innerText**: İlgili elemanın içindeki metini verir.

**innerHTML**: İlgili elemanın tüm HTML metnini verir.

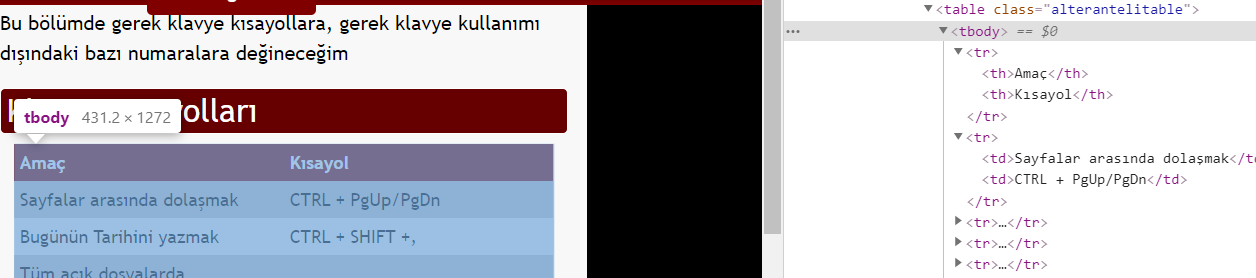
**textContent**: İlgili elemanın içindeki <span> v.s elemanları kapsayacak şekilde metnini verir.

### 2.örnek

Şimdi ise, konuyu pekiştirmek adına ve değişme ihtimali çok düşük bir sayfadan veri çekeceğiz. Değişme ihtimalinin çok düşük olması sayfanın benim web sitemdeki bir sayfa([https://www.Excelinefendisi.com/Konular/Excel/Giris\_PratikKisayollar.aspx](https://www.excelinefendisi.com/Konular/Excel/Giris_PratikKisayollar.aspx)) olmasından. Olur da bir değişklik yaparsam bu çok büyük ihtimalle “class” isminde olacaktır.

Hemen örneğimize geçelim.

Veri çekeceğimiz tablo ve html görüntüsü aşağıdaki gibidir.



Buna göre kodlarımız şöyle olacaktır. Tüm açıklamaları kod içinde bulabilirsiniz.

Sub webdenveri()

Dim IE As InternetExplorer

Dim tablolar As IHTMLElementCollection 'tablolarımızın ID'si yok, classı var, class olması demek 1den çok tablo olabilir demek,

'bu sayfamızda 1 tablo var gerçi, ama biz yine de bu şekilde ilerlemek durumundayız

Dim tbody As IHTMLElement 'tablodaki tbody elementini tutacak

Dim r As Integer

On Error GoTo hata

Set IE = New InternetExplorer

IE.Visible = False

url = "https://www.Excelinefendisi.com/Konular/Excel/Giris\_PratikKisayollar.aspx"

IE.navigate url 'ilgili sayfayı bellekte açıyoruz

'Bu kısım önemli, IE'yi bellekte açar açmaz sayfa yüklemesi hemen olmaz

Do While IE.Busy Or IE.READYSTATE <> 4 'sayfa yüklemesi tamamlanana kadar bekliyoruz

DoEvents

Loop

Set tablolar = IE.document.getElementsByClassName("alterantelitable") 'classı "alterantelitable" olan tüm tabloları tablolar değişkenine atıyoruz, 1 tane var zaten

'Application.Wait (Now + TimeValue("00:00:05")) 'eğer eksik veri geliyorsa tam olarak belleğe alamıyordur, bunun için gereken süre kadar bekleriz, burayı duruma göre sizin ayarlamanız gerekecek

r = 1 'başlangıç satırı

Set tbody = tablolar(0).Children(0) 'ilk tablonun(her ne kadar 1 tane olsa da) ilk elementi, yani tbody

For i = 0 To tbody.Children.Length - 1 'tbody altındaki tüm child elementlar kadar, yani tr tagleri kadar, özetle tüm satırlarda döneceğiz

Cells(r, 1).Value = tbody.Children(i).Children(0).innerText 'tbodynin ilk satırının ilk child elemanı, yani ilk td'si, yani ilk kolonu(ilk satır için td değil, th ama bizim için değişen birşey yok)

Cells(r, 2).Value = tbody.Children(i).Children(1).innerText 'tbodynin ilk satırının ikinci child elemanı, yani ikinci td'si, yani ikinci kolonu

r = r + 1

Next i

IE.Quit 'ilgili sayfada reklam v.s varsa arka planada müzik çalmaya devam eder, bu satırla arka planda açık olan Internet Exlporer'dan çıkarız ve reklam müziği sona erer

Set IE = Nothing: Set tablolar = Nothing: Set tbody = Nothing 'ilgili objeleri bellekten atıyoruz

Exit Sub

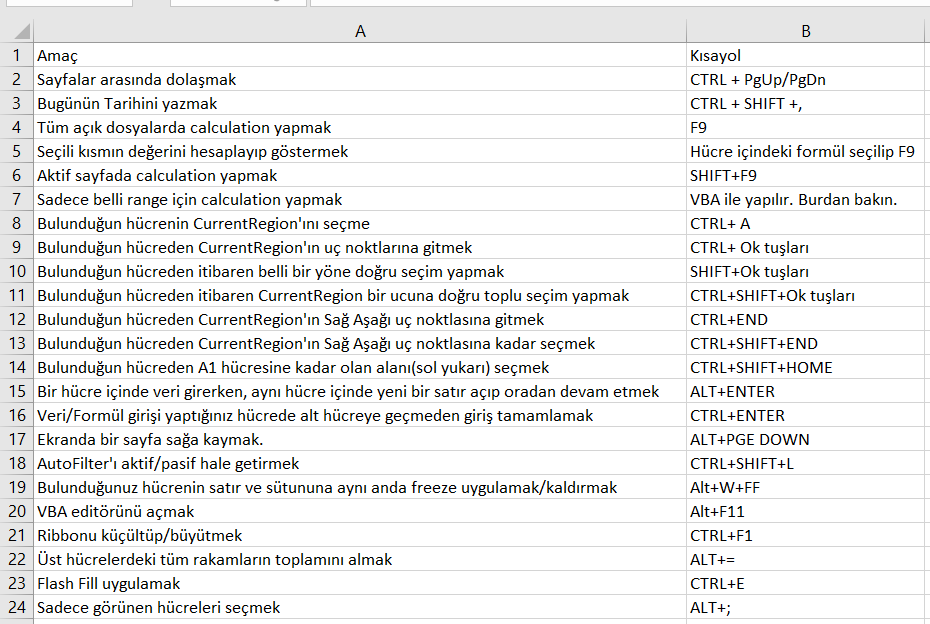
hata:

Set IE = Nothing: Set tablolar = Nothing: Set tbody = Nothing

MsgBox "Bi hata oluştu, " + Err.Description

End Sub

Sonuç aşağıdaki gibi olacaktır.



Gerekli formatlama işlerini size bırakıyorum.

## Neler Öğrendik

Outlook obje modeline girerek Outlook otomasyonu yapmayı; bu şekilde otomatik toplu mailler göndermeyi, Outlook klasörlerinde dolaşmayı öğrendik.

Veri tabanı terminolojisini öğrenerek altyapımızı sağlamlaştırdık, sonrasında bu altyapı üzerine DAO(Access’e özgü) ve ADO(Genel amaçlı) bağlantıları kurmayı öğrendik. Bu şekilde, veritabanlarından okuma ve onlara yazma işlemlerini yapmayı öğrendik.

Ayrıca, Excel içinden ListObjectler ve QueryTablelar ile refreshlenebilir bağlantılar kurmayı gördük.