# 8.bölüm

**8**

# Dosya ve klasör işlemleri

Bu bölümde klasörlerde gezinme, klasörler ve dosyalar hakkında bilgi alma, metin dosyalarının içini okuma, değiştirme ve sıfırdan dosya oluşturma gibi işlemlere bakacağız.

## Dosya ve Klasör erişimi

### Giriş

Makro yazarken zaman zaman dosyaların/klasörlerin adını değiştirmek, taşımak, kopyalamak, silmek v.b gibi manipüle edici; bazen de bunların adını, bulunduğu klasörü, adresini, dosya uzantısını öğrenmek v.b gibi bilgi edinici işlemler yapmak isteriz.

Bu tür işlemler için iki temel modül var. **FileSystemObject**(**FSO** olarak anılacak)Class Modülü ve **FileSystem(FS)** Modülü.

FS normal bir modül olduğu için bunlardaki fonksiyonları doğrudan kullanabiliyoruz. O yüzden FSO ile yapacak başka işlemimiz yoksa her zaman FS fonksiyonlarını tercih etmeliyiz, tabi FS ile yapılamayan işlemleri de mecburen FSO ile yapmamız gerekecektir.

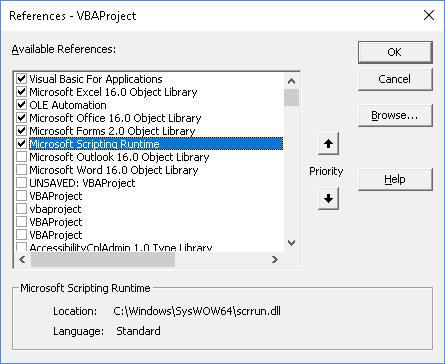
Şimdi bunlara detaylıca bakalım. (Her zaman olduğu gibi sadece işimize yarayan veya yarama ihtimali olan üyelere bakıyor olacağız. Ayrıca başka sayfanın konusu olan üyelere o sayfalarda yer vereceğiz.)

**NOT:** Klasör/Dosya açma/seçme v.s gibi işlemleri bu bölümde değil **Dört Temel Nesne>Application** konusunda ele alınmıştır.

### FileSystemObject(FSO)

#### Giriş

FSO nesnesi, web server üzerinde VBScript diliyle kullanılmak için yaratılmıştır(ancak tabiki VBA dünyasından da kullanıma girmiştir). Bu yüzden ayrı bir dll(library) içinde bulunur: **Scripting Runtime**. Bu nesneyi kullanabilmek için **Tools>Reference** menüsünden aşağıdaki gibi eklemek gerekir.(Tabi ki bu iş early binding için geçerli, late binding için böyle bir işleme gerek yoktur.)



Late binding için ise aşağıdaki gibi bir kod yeterli olacaktır. Ancak biz kodlarımızda genellikle early binding kullanacağımız için devam etmeden önce bu library'yi eklemenizi tavsiye ederim.

Set FSO = CreateObject("Scripting.FileSystemObject") 'Late binding

#### Nesne Hiyerarşisi

Dosya işlemleriyle ilgili olarak tepede FSO objesi bulunmaktadır. FSO kendi altında sırasıyla şu nesneleri&collectionları bulundurur.

* Drive(s)
* Folder(s)
* File(s)
* TextStream

FSO'yu doğrudan kullanmak yerine ya bu alt nesneleri üretmek için veya bu nesneleri temsil eden string ifadeler üzerinden işlem yapmak için kullanırız.

Mesela bir klasörü silmek için ya fso'nun DeleteFolder metodunu kullanıp parametre olarak da ilgili klasörün adresini yazarız, ya da GetFolder metodu ile bir Folder nesnesi yaratıp, sonra bu Folder nesnesinin Delete metodunu kullanırız.

Yalnız Folder nesnesini yaratırken dikkatli olmak lazım. Eğer ki, üzerinde çalıştığınız dosyada aynı zamanda Outlook library'si için de referans tanımladıysanız, onun da bir Folder class'ı vardır. İki class'ın karışmaması ve hata almamanız için Folder class'ının önüne library adını yazmamız gerekir: **Scripting.Folder** şeklinde.

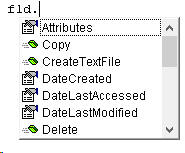
Zaten şu şekilde ikisi arasındaki ayrımı görmek kolaydır:

Dim fld As Scripting.Folder

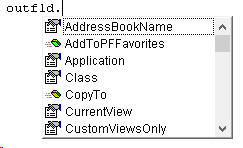
Dim outfld As Folder 'outlook folder, bunu kullanmayacağız

İki değişkenin de intellisense'ine baktığımızda çıkan üyelerin farklı olduğunu görüyoruz.

Dosya Klasörü olan Folder'ın üyeleri böyle iken,



Outlook Klasörü olan Folder'ın üyeleri böyledir.



Madem ki bu nesneyi doğrudan kullanmayacağız, ikide bir bu nesneden yaratmamak için bunu global seviyede public olarak tanımlamak akıllıca olacaktır. Global tanımlamayacaksak her prosedürün sonunda buna Nothing değerini atamak bellek yönetimi açısından iyi olacaktır.

Şimdi bu silme işlemine ait örneğe bakalım.

Public fso As New FileSystemObject 'global tanımlama

Sub foldersil()

Dim fld As Scripting.Folder

Dim outfld As Folder 'outlook folder, bunu kullanmayacağız

fso.DeleteFolder "C: \Users\Volkan\Desktop\sil1"

Set fld = fso.GetFolder("C: \Users\Volkan\Desktop\sil2")

fld.Delete

End Sub

#### Metodlar

FSO'nun en sık kullanacağımız 2 temel metodu şöyledir:

**GetFolder**: Folder(Klasör) nesnesi döndürür.

**GetFile**: File(Dosya) nesnesi döndürür.

Bu ikisinden başka metodlar da var tabi ama bu ikisinin özelliği, diğer 2 temel nesneyi yaratıyor olmaları.

Bu nesnenin metotlarına daha genel bir bakış ise şöyle olacaktır.

|  |  |
| --- | --- |
| Metodlar | Görevleri |
| GetDrive, GetFolder, GetFile | Yukarda bahsettik. |
| CreateFolder, CreateTextFile | Yeni klasör ve dosya yaratır. |
| DeleteFile, DeleteFolder | Klasör ve Dosya siler |
| CopyFile, CopyFolder | Klasör ve Dosya kopyalar |
| MoveFile, MoveFolder | Klasör ve Dosya taşır |
| DriveExists, FolderExists, FileExists | İlgili birim mevcut mu kontrolü yapar |

**NOT**: Fso işlemlerinde ilgili dosya/klasör adresi verilirken son karakterin "/" olup olmaması önem arz e**tm**emektedir. Yani "C: \deneme" ile "C: \deneme\" özdeştir.(Ancak daha aşağıda göreceğimiz gibi "Dir" ile kullanırken durum farklıdır.)

#### Folder ve File işlemleri

Folder/File işlemlerinde akılda tutulması gereken en önemli şey, öncelikle dosyanın var olup olmadığını öğrenmeye çalışmaktır. Dosya özellikle **Application.FileDialog** ile kullanıcıya seçtirilmemişse bu kontrol işlemini mutlaka yapın derim. (FileDilaog ile yapılan seçimlerde bu kontrole gerek yoktur.)

##### Var mı? kontrolü

Bunun içn **FileExists** ve **FolderExists** metotlarını kullanırız.

Sub kontrol\_fso()

If fso.FileExists("C: \Users\Volkan\Desktop\denemeler\deneme.xlsx") Then

Debug.Print "Dosya var"

End If

If fso.FolderExists("C: \Users\Volkan\Desktop\denemeler\") Then 'sonda \ olup olmaması fark etmez

Debug.Print "Klasör var"

End If

End Sub

##### Klasör içindeki klasörleri elde etme

Bu işlem için **SubFolders** property'si kullanılır, bu özellik bize Folders collection'ı döndürür. Gerisi ForEach yapmaktan ibarettir.

Sub KlasördekiKlasörler()

Dim anaklasorStr As String

Dim fol As Folder, alt As Folder

anaklasorStr = "C: \Users\Volkan\OneDrive\Dökümanlar"

Set fol = fso.GetFolder(anaklasorStr)

For Each alt In fol.SubFolders

Debug.Print alt.Name

'diğer fso işlemleri

Next

End Sub

**NOT**: Bu işlemi FS modülündeki Dir ile de yapabiliyoruz. Başka FSO işlemi yapacaksak(FolderExists kullanmak gibi) bu yöntemi kullanalım, yoksa Dir yöntemini.

##### Klasör içindeki dosyaları elde etme

Bu işlem için **Files** property'si kullanılır. Folder'da olduğu gibi bir Collection elde eder ve Foreach uygularız.

Sub KlasördekiDosyalar()

Dim anaklasorStr As String

Dim fol As Folder

Dim f As File

anaklasorStr = "C: \Users\Volkan\OneDrive\Dökümanlar"

Set fol = fso.GetFolder(anaklasorStr)

For Each f In fol.Files

Debug.Print f.Name

'diğer fso işlemleri

Next

End Sub

**NOT**: Bu işlemi FS modülündeki Dir ile de yapabiliyoruz. Başka FSO işlemi yapacaksak(FolderExists kullanmak gibi) bu yöntemi kullanalım, yoksa Dir yöntemini.

##### Klasör içindeki (tüm alt klasörlerin içindekiler dahil) dosyaları elde etme

Bu örnek üsttekilere göre biraz daha karmaşıklık içerir ancak mantığı açısından güzel bir örnektir.

Alt klasörleri işleme dahil etmek için **Collection** tipinde bir yığın oluşturuyoruz ve her defasında bu yığının ilk üyesi üzerinde işlem yapıyoruz. İşlem yapmadan önce bu ilk elemanı yığından dışarı atıyoruz ki bir daha işleme girmesin. Sonra yığındaki elemanların(klasörlerin) her biri için işlemi yineliyoruz. Bu işlemi anlamanın en iyi yolu, Local penceresi açıkken F8 ile ilerlemek olacaktır.

NOT: Bunun daha hızlı ve basit you aşağıda Dir bölümünde ele alınacaktır. Ancak fso nesnesiyle ilgili başka kontroller veya işlemler yapılması gerekirse bu yöntemin kullanılması tercih edilmelidir. Bu arada internette araştırırsanız başka yöntemler olduğunu da görebilirsiniz. Kullanım tercihi size kalmış.

Bu örnekte bir klasördeki dosyaların adını, bulunduğu klasörü ve dosya boyutunu yazdırıyoruz. Kendi diskinizdeki bir klasör ile yer değiştirerek deneyebilirsiniz.(Örnek klasörü **github’dan(FSO Örnek.rar**) indirebilirsiniz)

Sub KlasördekiAltKlasörDahilDosyalar()

Dim kls As Scripting.Folder, altkls As Scripting.Folder

Dim dosya As File

Dim klasörYığını As New Collection

Dim i As Integer

Dim YığındaNeVar As String, sabtiStr As String

Dim adım As Integer

klasörYığını.Add fso.GetFolder("C: \Users\Volkan\Videos\Movavi Screen Capture Studio\Udemy Kurslar\2-ileri VBA-makro\FSO Örnek")

sabitstr = "C: \Users\Volkan\Videos\Movavi Screen Capture Studio\Udemy Kurslar\2-ileri VBA-makro\"

i = 1

adım = 1

'yığındaki eleman sayısı 0 olana kadar yani tüm alt klasörler bitene kadar devam edeceğiz

Do While klasörYığını.Count > 0

Set kls = klasörYığını(1)

klasörYığını.Remove 1 'ilk klasörü yığından çıkarıyoruz

'alt klasörleri yığına ekliyoruz

For Each altkls In kls.SubFolders

klasörYığını.Add altkls

Next altkls

'bu if bloğu informativedir, silinebilir

If klasörYığını.Count > 0 Then

For Each k In klasörYığını

YığındaNeVar = Replace(k, sabitstr, "") & ";" & Replace(YığındaNeVar, sabitstr, "")

Next k

ActiveCell(i, 1).Value = Mid(YığındaNeVar, 1, Len(YığındaNeVar) - 1)

End If

For Each dosya In kls.files

ActiveCell(i, 2).Value = adım & ". adımdaki klasör: " & kls.Name 'informativedir, silinebilir

ActiveCell(i, 2).Offset(0, 1).Value = Replace(dosya.ParentFolder, sabitstr, "")

ActiveCell(i, 2).Offset(0, 2).Value = dosya.Name

ActiveCell(i, 2).Offset(0, 3).Value = dosya.Size

i = i + 1

Next dosya

YığındaNeVar = vbNullString

adım = adım + 1

Loop

End Sub

Bu işlemi yapmanın bir diğer yolu da işlemi recursive bir şekilde ele almaktır.

'ana prosedür

Sub recursive\_fulldosya()

Dim anaklasorStr As String

anaklasorStr = "C: \inetpub\wwwroot\aspnettest\Excelefendi2\"

Recursiveİlerle fso.GetFolder(anaklasorStr)

End Sub

'recursive prosedür

Sub Recursiveİlerle(kls As Variant) 'variant çünkü ilk girerken Folder sonra Folders olacak

Dim altKlasorler As Variant

Dim dosya As file

Static i As Integer ' her defasında bir önceki değerini koruması için

On Error Resume Next 'erişim izni olmayan yerlerde hata almasın diye

For Each altKlasorler In kls.SubFolders

Debug.Print altKlasorler 'bilgi amaçlıdır

Recursiveİlerle altKlasorler

Next

For Each dosya In kls.Files

ActiveCell(i, 1).Value = dosya.ParentFolder

ActiveCell(i, 1).Offset(0, 1).Value = dosya.Name

ActiveCell(i, 1).Offset(0, 2).Value = dosya.Size

i = i + 1

Next

End Sub

#### Diğer FSO ve File/Folder işlemleri

##### Dosyaları ReadOnly yapmak

Bu işlemi de FS modülü ile yapabiliyoruz. Neden böyle bir işlemi yapmak isteyeceğimi orada açıklıyorum. Burada sadece kısaca bu işlemin nasıl yapıldığına bakalım. Önceki örneklerde olduğu gibi eğer File nesnesi ile FileSystem ile yapılamayacak başka işlemler yapacaksanız bu yöntemi kullanın, yoksa en hızlısı FileSystem olduğu için onu kullanın.

dosyaStr = "C: \deneme.xlsx"

Set dosya = fso.GetFile(dosyaStr)

dosya.Attributes = 1 'bu özellik hem okunur hem yazılırdır

##### Silme işlemi

Yine FS ile de yapılabilir ve öncekilerde olduğu gibi FS'nin Kill fonksiyonunu kullanmak FSO'nun metotlarından daha efektiftir, özellikle büyük çaplı işlemlerde.

Bazı eylemler için FSO'nun metotlarını da File/Folder'ın metotlarını da kullanabiliyoruz,  FS'nin fonksiyonlarını da. Tıpkı silme işleminde olduğu gibi. Aşağıdaki örnekte bu 3 yönteme de bakalım.

Sub silmeler()

Dim f As file

'1.yöntem: Filesystem modülündeki Kill fonk ile

a = "C: \Users\Volkan\Desktop\a.txt"

FileSystem.Kill a 'FileSystem yazmaya gerek yoktur

'2.yöntem: fso nesnesi ile. Fso'yu başka amaçla da kullanacaksak

b = "C: \Users\Volkan\Desktop\b.txt"

If fso.FileExists(b) Then

fso.DeleteFile b

End If

'3.yöntem: file nesnesi ile. File bilgisi lazımsa

c = "C: \Users\Volkan\Desktop\c.txt"

Set f = fso.GetFile(c)

If f.Size > 1024 Then

f.Delete

End If

End Sub

##### Kopyalama

###### Tek dosya kopyalamak

Kaynak olarak her zaman dosya adı belirtilir. Hedef olarak klasör adı veya dosya adı belirtilebilir.(fso.CopyFile kaynakdosya, hedefklasör)

'hedef: klasör

fso.CopyFile "C: \Users\Volkan\Desktop\denemeler\Şubeliste.xlsx", "C: \Users\Volkan\Desktop\ıvır zıvır\" 'sonda \ olmalı

'hedef: dosya adı, dosyanın adı dğeiştirilebilir

fso.CopyFile "C: \Users\Volkan\Desktop\denemeler\Şubeliste.xlsx", "C: \Users\Volkan\Desktop\ıvır zıvır\ŞubelisteforBölgeler.xlsx"

###### Aynı tipteki çoklu dosya kopyalama

Aşağıdaki kod ile kaynak klasördeki tüm xlsx, xlsm, xlsb,xls uzantılı dosyaları hedef klasöre kopyalamış oluyoruz.

kaynak = "C: \Users\Volkan\Desktop\denemeler"

hedef = "C: \Users\Volkan\Desktop\ıvır zıvır\"

fso.CopyFile Kaynak & "\\*.xl\*", Hedef

##### Yeniden adlandırmak ve taşımak

**MoveFile** kullanılabileceği gibi MSDN'de dokümante edilmemiş bir **Name** fonksiyonu var bu da kullanılabilir. MoveFile kullanımı CopyFile'a benzer.

Sub rename()

'kaynak dosya ve hedef klasör belirterek

fso.MoveFile "C: \Users\Volkan\Desktop\denemeler\Şubeliste.xlsx", "C: \Users\Volkan\Desktop\ıvır zıvır\" 'sonda "\" olmalı

'veya kaynak dosya, hedef dosya adı(farklı isim olabilir)

fso.MoveFile "C: \Users\Volkan\Desktop\denemeler\Şubeliste.xlsx", "C: \Users\Volkan\Desktop\ıvır zıvır\ŞubelisteforBölgeler.xlsx"

'veya

Name "C: \Users\Volkan\Desktop\denemeler\Şubeliste.xlsx" As "C: \Users\Volkan\Desktop\ıvır zıvır\ŞubelisteforBölgeler.xlsx"

End Sub

##### Dosya son değişim tarihi hakkında bilgi almak

Bir dosyanın gün içinde birkaç kez açılma durumu varsa ve dosyada bir kaydetme işlemi uygulanıyorsa, dosyanın daha önce kaydedilip edilmediği bilgisine bakarak sonraki açılışlarda kodun çalışmamasını sağlayabilirsiniz(veya tamamen başka sebeplerle).

Set f = fso.GetFile(gunlukyol & adres)

If DateValue(f.DateLastModified) = Date Then

Exit Sub

Else

'diğer kodlar

End If

##### Dosya isim, uzantı ve adresleri

Dosya isim, uzantı ve adreslerine sıklıkla ihtiyaç duyuyor olacağız. Bunların açıklamasını doğrudan kod içinde vermek daha kolay olacaktır.

Sub cesitli\_fsofilefolder()

Dim f As file

Dim k As Folder

dosya = "C: \Users\Volkan\Desktop\denemeler\deneme.xlsx"

Debug.Print fso.GetAbsolutePathName(dosya) 'C: \Users\Volkan\Desktop\denemeler\deneme.xlsx

Debug.Print fso.GetBaseName(dosya) 'deneme

Debug.Print fso.GetDriveName(dosya) 'C:

Debug.Print fso.GetExtensionName(dosya) 'xlsx

Debug.Print fso.GetFileName(dosya) 'deneme.xlsx

Debug.Print fso.GetParentFolderName(dosya) 'C: \Users\Volkan\Desktop\denemeler

Set f = fso.GetFile(dosya)

Debug.Print f.Name 'deneme.xlsx

Debug.Print f.ParentFolder 'denemeler

Debug.Print f.Path 'C: \Users\Volkan\Desktop\denemeler\deneme.xlsx

Debug.Print f.ShortName 'DENEME~1.XLS

Debug.Print f.ShortPath 'C: \Users\Volkan\Desktop\DENEME~1\DENEME~1.XLS

Set k = f.ParentFolder

Debug.Print k.Name 'denemeler

Debug.Print k.ParentFolder 'C: \Users\Volkan\Desktop

Debug.Print k.Path 'C: \Users\Volkan\Desktop\denemeler

Debug.Print k.ShortName 'DENEME~1

Debug.Print k.ShortPath 'C: \Users\Volkan\Desktop\DENEME~1

End Sub

### FileSystem(FS)

Yukarıda belirttiğimiz gibi FS modülü, class modül olmayıp bunun içindeki fonksiyonları kullanmak için bir FS nesnesi yaratmaya gerek bulunmamaktadır, daha da önemlisi FSO gibi başka bir library'yi reference olarak göstermeye gerek yoktur.

Yine yukarıda belirttiğimiz gibi FSO ve FS'nin ortak üyeleri bulunmaktadır. Amacımız sadece bu metodu kullanmak ise tercih her zaman FS'den yana olmalı, ancak FSO'nun diğer üyeleriyle de işlem yapılacaksa işte o zaman FSO kullanılmalıdır.

Şimdi FS'nin metotlarına bakalım.

#### ****Dir****

**Dir** heralde FS'nin en önemli fonksiyonudur. Parametre olarak dosya/klasör adı alır. Bunu kullanırken bir klasörle işlem yapacaksak **en sonda hep "\" olmasına dikkat edilmelidir**. Zira bu fonksiyonu aldığı parametrenin dosya mı klasör mü olduğunu sondaki "\" ile anlıyor. O yüzden Dir ile ilgili işlem yapmadan önce sağ tarafa "\" olup olmadığına göre aşağıdaki gibi bir düzeltme işlemi yapılmasında fayda var(tabi bir klasör ile uğraştığımızı düşünüyorsak).

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

klasor = klasor & "\"

End If

Bu fonksiyonun dönüş değeri String'tir. Peki ne döndürür?

* Parametre olarak Klasör adresi alıyorsa, klasördeki ilk dosyanın adını
* Parametre olarak Dosya adresi alıyorsa, dosyanın kendisini
* Parametre olarak ne alırsa alsın, eğer böyle bir dosya/klasör yoksa ZLS, yani sıfır uzunluklu bir metin döndürür

Sub fs\_dir()

kls1 = "C: \Users\Volkan\Desktop\denemeler" 'sonunda \ yok

kls2 = "C: \Users\Volkan\Desktop\denemeler\"

dsy = "C: \Users\Volkan\Desktop\denemeler\algo.xlsx"

Debug.Print Dir(kls1) 'hiçbişrey döndürmez

Debug.Print Dir(kls2) 'klasördeki ilk dosyayı döndürür

Debug.Print Dir(dsy) 'dosyanın kendisini döndürür

Debug.Print Len(Dir("var\_olmayan\_dosya\_veya\_klasör"))

End Sub

Bu fonksiyon FSO'nun FileExists/FolderExists özellikleri yerine de kullanılabilir ancak bunlardan farklı olarak True/False döndürmez, yukarıda belirttiğimiz gibi bir string döndürür, ilgili parametreyi bulamazsa ZLS döndürür demiştik. O yüzden sonucun vbNullString olup olmadığına bakmalıyız.

Var mı? kontrolü

Sub kontrol2()

If Dir("C: \Users\Volkan\Desktop\denemeler\deneme.xlsx") <> vbNullString Then

Debug.Print "Dosya var"

End If

If Dir("C: \Users\Volkan\Desktop\denemeler\") <> vbNullString Then 'sondaki \ işaretine dikkat. Bu klasör mevcut olsa bile \ işareti olmazsa Null döndürür

Debug.Print "Klasör var"

End If

End Sub

Aşağıda Excel gurularından Ken Puls tarafından bir fonksiyon haline dönüştürülmüş(benim de bir zamanlar sıklıkla kullandığım) versiyonunu görüyorsunuz.

Public Function FileFolderExists(strFullPath As String) As Boolean

'Author : Ken Puls (www.Excelguru.ca)

'Macro Purpose: Check if a file or folder exists

If strFullPath = vbNullString Then Exit Function

On Error GoTo EarlyExit

If Not Dir(strFullPath, vbDirectory) = vbNullString Then FileFolderExists = True

EarlyExit:

On Error GoTo 0

End Function

'Kullanımı

Sub varmı\_test()

'adres olarak dosya adresi de klasör adresi de verilebilir

Const adres As String = "var\_olmayan\_dosya\_veya\_klasör"

If FileFolderExists(adres) Then

Debug.Print "var"

Else

Debug.Print "yok"

End If

End Sub

Yalnız bu tür şeylerin üzerine kafa yorunca siz daha basit çözümler bulabiliyorsunuz. Mesela ben aşağıdaki çözümü buldum. Yaptığım şey basit: Parametre olarak verilen şeyin hem file hem folder için işleyen **GetAttr** metodu ile attribute bilgisini almaya çalışıyorum; eğer bilgi alabilirsem demek ki dosya veya klasör mevcuttur; bilgi alamıyorsam, ki öyleyse hata üretir, o zaman hata yönetim bloğu ile de fonksiyona False değerini atarım.

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

'kullanım

Sub mevcutmu\_test()

Debug.Print Mevcutmu("C: \Users\Volkan\") 'True

Debug.Print Mevcutmu("C: \Users\Volki\") 'False

Debug.Print Mevcutmu("C: \Users\Volkan\Desktop\denemeler\deneme.xlsx") 'True

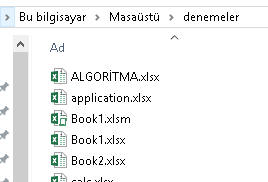
Debug.Print Mevcutmu("C: \Users\Volkan\Desktop\denemeler\deneme5.xlsx") 'false

End Sub

##### İkinci parametre

Dir'in ikinci parametresi çeşitli constantları alır, bunların en önemlisi vbDirectory'dir. İkinci parametre girilmezse(veya vbNormal olarak girilirse, ki default olanı budur) ilk dosyayı elde ederiz demiştik, işte bu ikinci parametre vbDirectory olarak girilirse klasörün kendisini elde ederiz, ve bunu "." olarak görürüz.

Aşağıdaki klasör göz önüne alındığında;



Sub dir\_ikinciparametre()

Debug.Print Dir("C: \Users\Volkan\Desktop\denemeler\") 'klasördeki ilk dosyayı verir, ALGORİTMA.xlsx

Debug.Print Dir("C: \Users\Volkan\Desktop\denemeler\ALGORİTMA.xlsx") 'dosyanın kendisini verir

Debug.Print Dir("C: \Users\Volkan\Desktop\denemeler\", vbDirectory) 'Klasörün kendisi yani '.'

Debug.Print Dir("C: \Users\Volkan\Desktop\denemeler\ALGORİTMA.xlsx", vbDirectory) 'dosyanın kendisini verir

End Sub

##### Çoklu kullanım

Dir'in iki parametresi de opsiyonel olmakla birlikte, ilk kullanımı mutlaka parametreli olmalıdır. Ondan sonra hiç parametre almadan da kullanılabilir. Parametresiz kullanım, **"sonraki dosya/klasöre geç"** demektir.

Dir'i bu şekilde arka arkaya kullandığımızda bir yerde artık sona ulaşılır ve yeni bir dosya/klasör elde edilemez; işte bu noktada sonuç ZLS'dir. O yüzden bu noktaya ulaşıldığında ya bu tekrarlı çağırma işini sonlandırmalı, ki bu genelde bir döngü içinde yapıldığından döngüden çıkmak anlamına gelir, ya da yeniden parametreli bir kullanıma geçmelisiniz.

Sub çok\_kullanım()

Debug.Print Dir("C: \Users\Volkan\Desktop\denemeler\")

Debug.Print Dir 'parantezsiz de kullanılabilir

Debug.Print Dir() 'parantezli de

Debug.Print Dir("") 'klasördeki ilk dosyaya döner

Debug.Print Dir("C: \Users\Volkan\Desktop\denemeler\ALGORİTMA.xlsx")

Debug.Print Dir 'Dosyalarda bu ZLS döndürür

'Debug.Print Dir() 'Bu hata verir

Debug.Print Dir("") 'klasördeki ilk dosyaya döner

End Sub

##### Klasör içindeki klasörleri elde etme

Bu işlemi FSO altında da incelemiştik. Ancak eğer FSO ile ilgili başka bir işlem yapılmayacaksa Dir ile çok daha hızlı yapılır. (Koddaki GetAttr'le ilgili satıra çok takılmayın, bunu şimdilik ezbere kullanın, zira konu biraz karışık, merak edenler için NSDN’de(<https://msdn.microsoft.com/en-us/library/ee200232.aspx>) detaylı açıklama var.)

Sub LoopinKlasör()

Dim i As Integer

i = 1

anaklasor = "C: \inetpub\wwwroot\aspnettest\Excelefendi2\"

klasor = Dir(anaklasor, vbDirectory) 'kendisini alarak başlıyoru

Do Until klasor = "" 'Dir'den klasor dönmeyene kadar ilerliyoruz

If (GetAttr(anaklasor & klasor) And vbDirectory) = vbDirectory Then

ActiveSheet.Cells(i, 1).Value = anaklasor & klasor

i = i + 1

End If

klasor = Dir() 'Parametresiz dir ile sonraki klasöre ilerliyoruz(dosyalarla çalışsaydık sonraki dosya)

Loop

End Sub

NOT: Bu döngü sırasında görünen "..", parent klasörü temsil eder.

##### Klasör içindeki dosyaları elde etme

Yine bu işlem de FSO ile yapılabilmekteydi ancak FSO'nun başka kullanımı olmayacaksa Dir ile yapmak daha hızlıdır.

İlk parametreli Dir'den sonraki parametresiz Dir ile aynı klasörde ilerleriz. Ayrıca Dir, joker eleman da aldığı için tüm dosyalarda değil belli dosyalar üzerinde de dolaşabilirsiniz.

Aşağıdaki örneklerde, bir klasörde "VBA" ifadesi içeren dosya isimlerini yazdırıyoruz. 350 dosya içeren bu klasörde Dir'in çalışma hızı 0,17 sn iken FSO 0,53 sn olmuştur. Daha kalabalık klasörlerde fark daha da açılmaktadır.

'Dir ile

Sub dosyalarda\_gezin\_dir()

Dim bas As Single, bts As Single

bas = Timer

f = Dir("C: \Users\Volkan\Desktop\denemeler\\*al\*") 'içinde al geçen ilk dosyayı bulur

Debug.Print f

Do While Len(f) > 0

f = Dir 'sonraki dosyaya geçiyoruz. Sonunda () gerekli değil, olsa da olur olmasa da

Debug.Print f

Loop

bts = Timer

Debug.Print bts - bas

End Sub

'FSO ile daha uzun

Sub dosyalarda\_gezin\_fsofilefolder()

Dim bas As Single, bts As Single

Dim fso As New FileSystemObject

Dim kaynak As Scripting.Folder

Dim dosya As File

bas = Timer

Set kaynak = fso.GetFolder("C: \Users\Volkan\Desktop\denemeler\")

For Each dosya In kaynak.files

If InStr(dosya.Name, "al") > 0 Then

Debug.Print dosya.Name

End If

Next dosya

bts = Timer

Debug.Print bts - bas

End Sub

##### Klasör içindeki (tüm alt klasörler dahil) dosyaları elde etme

Yine aynı açıklamamız geçerli. FSO ile de yapabiliriz, ancak Dir daha hızlıdır. Eğer ki FSO'yu başka amaçla da kullanacaksak FSO tercih edilmeli, yoksa Dir.

Aşağıdaki örneği <http://www.ammara.com/> sitesinden aldım. Kendi ihtiyacınıza göre değiştirebilirsiniz. F8 ve Local Windows aracılığı ile değişiklikleri izleyerek incelemenizi tavsiye ederim.

Sub recursive\_Dir()

Dim colFiles As New Collection

RecursiveDir colFiles, "C: \deneme", "\*.\*", True

Dim vFile As Variant

For Each vFile In colFiles

Debug.Print vFile

Next vFile

End Sub

'recursive Fonksiyonumuz

Public Function RecursiveDir(colFiles As Collection, \_

strFolder As String, \_

strFileSpec As String, \_

bIncludeSubfolders As Boolean)

Dim strTemp As String

Dim colFolders As New Collection

Dim vFolderName As Variant

'Add files in strFolder matching strFileSpec to colFiles

strFolder = TrailingSlash(strFolder)

strTemp = Dir(strFolder & strFileSpec)

Do While strTemp <> vbNullString

colFiles.Add strFolder & strTemp

strTemp = Dir

Loop

If bIncludeSubfolders Then

'Fill colFolders with list of subdirectories of strFolder

strTemp = Dir(strFolder, vbDirectory)

Do While strTemp <> vbNullString

If (strTemp <> ".") And (strTemp <> "..") Then

If (GetAttr(strFolder & strTemp) And vbDirectory) <> 0 Then

colFolders.Add strTemp

End If

End If

strTemp = Dir

Loop

'Call RecursiveDir for each subfolder in colFolders

For Each vFolderName In colFolders

Call RecursiveDir(colFiles, strFolder & vFolderName, strFileSpec, True)

Next vFolderName

End If

End Function

'Dir kullanırken sondaki \ işareti önemli olduğu için bunu ele alma fonksiyonu

Public Function TrailingSlash(strFolder As String) As String

If Len(strFolder) > 0 Then

If Right(strFolder, 1) = "\" Then

TrailingSlash = strFolder

Else

TrailingSlash = strFolder & "\"

End If

End If

End Function

#### Diğer FileSystem üyeleri

##### ReadOnly yapmak

Bir dosyayı neden Readonly yapmak isteyesiniz ki? Kitabın önsözünde belirttiğim gibi her şeyi öğrenmeye çalışın, en azından böyle bir şeyin var olduğunu bilin, bir gün lazım olabilir. Bendeki ihtiyaç şöyle doğdu:

İşyerinde 2 pc ile çalışıyorum. Ana pc'mdeki Personal.xlsb dosyamda yaptığım değişikliklerin dosyanın Workbook\_AfterSave eventi ile network üzerindeki ortak alana kaydolmasını sağlıyorum. Diğer pc ve benimle çalışan arkadaşımın da bendeki personal dosyasının en güncel versiyonuna sahip olmasını istiyorum. Bunun için de onların pcler’indeki Task Scheduler ile her gece güncel dosyayı kendi pcler’ine kopyalamasını sağlayan bir Task yarattım(Tabi önce Application.OnTime ile akşam belli bir saatte Excel’in kapatılmasını sağlıyorum). Süreç özetle şöyle:

* Personal.xlsb dosyamda bir değişklik yaparım
* Dosyanın güncel hali anında ortak alana kaydolur
* Akşam 21: 00'de arkadaşımın pc’sinde ve benim diğer pc'mde Excel kapanır
* Gece 00: 00da arkadaşımın ve benim diğer pc'mde Task Scheduler önce güncel dosyayı bunların XLSTART klasörüne kopyalar
* Hemen arkasından Excel otomatik açılır

Şimdi burda şöyle bir tehlike var. Eğer ki arkadaşım kendi pc'sindeki Personal.xlsb dosyasında bir şeyler denemek isteyip sonrasında bunu da kaydetmek isterse ortak alana da bu değişikliklerle kaydedilmiş olacak. Belki önemli bazı şeyleri bozmuş olacak. O yüzden arkadaşımın bu dosya üzerinde Save etme işlemini engellemem gerekir. Bunun için dosyanın ortak alana kaydolurken **Readonly** olarak kaydolmasını sağlayabilirim. Böylece bu dosya, arkadaşımda ve benim diğer pc'de readonly açılacak. PC'de sadece otomatik işler çalıştığı için orda zaten kaydetme endişesi yok, ancak arkadaşım kaydetmeye çalışınca ona uyarı çıkacak, ya başka isimle kaydetmek zorunda kalacak ya da kaydetme işleminden vazgeçecek. İşte bütün bu olay için kodumuz şöyle:

Private Sub Workbook\_AfterSave(ByVal Success As Boolean)

FileSystem.SetAttr Path & Me.Name, vbNormal 'Varolanın üzerine yazması için önce Readonly özelliğini kaldırıyorum

fso.CopyFile Me.FullName, Path 'Dosya o an açıkken FilseSystem'in FileCopy metodu hata veriri. O yüzden fso nesnesini kullandık

FileSystem.SetAttr Path & Me.Name, vbReadOnly 'en son readonly yapıyorum

End Sub

##### Diğer metodlar

**Kill ve RmDir:** Sırayla dosya ve klasör silerler. Kill'in iki muadili var, FSO altında karşılaştırması var. RmDir  FSO'daki DelefeFolder muadilidir.

**MkDir:** Klasör yaratır. FSO'daki CreateFolder muadili

**FileCopy:** Fso'daki CopyFile'ın muadilidir.

**FileLen:** File'ın Size muadili.

**FileDateTime:** Yine File'ın DateLastModified muadili

**GetAttr ve SetAttr:** File'ın Attributes muadili.

## Dosya Okuma ve Yazma işlemleri

### Giriş

İçine veri yazılan birçok dosya tipi bulunmaktadır, ancak biz VBA konsepti içinde genellikle text(csv dahil) dosyalarıyla çalışacağız. SQL kodlarını depoladığınız ".sql" uzantılı dosyaları da bu kapsamda düşünebilirsiniz. Excel dosyalarla çalışmayı zaten Workbook ve Application bölümlerinde görmüştük.

Text dosyaları bilgi depolayıp okumanın kolay bir yolunu sunarlar. Özellikle settings(ayar) bilgileri veya aşamalı bir sürecin durum bilgilerini(log) okumakta/yazmakta oldukça kullanışlıdırlar.

VBA'de iki tür okuma/yazma yöntemi bulunuyor. Öncelikle biz VB6'dan miras gelen klasik okuma yazma yöntemine bakacağız.

**UYARI**: Buradan itibaren aşağıda göreceğiniz tüm dosya okuma işlemlerinde, dosya okuma hep sağa ve aşağı yönlüdür. Döngüsel işlemlerde "bir sağ kolona/karaktere veya bir alt satıra geç" tarzında ilave bir kod ifadesi yoktur. Bu işlem otomatik olmaktadır.

### Klasik Okuma/Yazma işlemleri

#### Dosya açma ve Kapama

##### Dosya Açma

Okuma işlemi için de de Yazma işlemi için de öncelikle dosyanın açılması gerekir. Bunun için **Open** fonksiyonu kullanılır. Aşağıdaki gibi bir syntax'a sahiptir.

**Open dosyayolu For mod [Erişim tipi] [lock] As Dosyano**

**Dosyayolu**: Dosyanın bulunduğu tam adres. Ör: C: \deneme\deneme.txt

**Mod**: **Input** ise okuma, **Output** ise yazma, **Append** ise dosya sonuna ekleme yapılır. 2 tane daha var ama bize bu üçü yeter. Output seçildiğinde mevcut dosya varsa ezilip içeriği yeniden oluşturulur, olmayan bir dosya girildiyse bu dosya yaratılır.

**Erişim tipi ve Lock tipi**: Opsiyoneldirler. Dosya açıkken, başkalarının ne yapabileceğini gösterir. Biz bunları kullanmayacağız, o yüzden default değerleri devreye girecek.

##### Freefile

Dosyalar açıldığında onlara bir sıra numarası verilir. Bu numara manuel belirtilebileceği gibi, çok sayıda okuma yazma yapılan bir prosedür içinde o andaki müsait sıra numarasını veren **Freefile** deyimi de kullanılabilir. Manuel giriş için sıra numarası # ile kullanılır. #1 gibi.

##### Dosyayı Kapama

Dosyayı Close ifadesi ile kaparız, ancak parametre olarak dosya adresi değil, numarasını alır.

Dosyayı kapatmazsak, tekrar aynı dosyayı açmaya çalıştığımızda "Dosya zaten açık" hatası alırız.

##### Kod

Bu durumda, örnek bir dosya açma kodu aşağıdaki gibi olacaktır.

adres = "C: \Users\Volkan\Desktop\denemeler\dosya1.txt"

Open adres For Input As #1 'veya 1 veya FreeFile

'çeşili işlemler

Close #1

#### Dosyadan veri okuma

##### Kolon kolon bilgi okumak

Aşağıdaki bilgileri içeren bir text dosyamız olsun. Kişinin adı, yaşı ve baba adı bilgileri var.  
  
volkan ,38 ,ismail  
ayşe ,40 ,murat  
serkan ,35 ,osman

Buradan ilk kaydın yaş bilgisini almak istiyoruz diyelim. Bunun için **Input** deyimini kullanıp, istediğimiz kolon sayısı kadar değişken belirleyip kolon bilgilerini bu değişkenlere atıyoruz.

Sub teksatırdan\_tekkolon\_okuma()

adres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt

Open adres For Input As 1

Input #1, adı, yaş 'kolon sayısı kadar parametre alır. # zorunlu

Debug.Print yaş '38 verir

'ikinci bir Debug.Print yaş yazsak bile yine 38 yazar. sadece tek satır okuması var.

Close 1

End Sub

Bu kod tabiki sadece ilk satır için bilgi döndürür. Satırda ilerleme yapamıyoruz. Özellikle her defasında üzerine yazma yapılan tek satırlık bilgi içeren dosyalarda kullanışlıdır. Mesela ikinci kolonunda, bilginin yazdırıldığı tarihi veya kişiyi gösteren bir dosyadan bu tarihi veya kişiyi elde etmek isteyebiliriz. Böylece bu dosyaya en son ne zaman bilgi yazıldığını veya kimin tarafından yazıldığını elde edebiliriz.

##### Tek satırdan kısmi bilgi okuma

Input ifadesini **Input(karakteradedi, dosyano)** şeklinde kullandığımızda belirli adette karakter okumuş oluruz.

Sub teksatır\_kısmen\_okuma()

adres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open adres For Input As 1

x = Input(6, 1) '1 nolu dosyadan 6 karakter oku

Debug.Print x 'volkan

End Sub

##### İlk satırın tamamını okuma

Yine yukarıdaki dosyamı elimizde bulunsun. Bu sefer ilk satırın tamamını elde edeceğiz. Bunu **Line Input** deyimi ile yapıyor ve içeriği ikinci parametredeki değişkene atıyoruz .

Sub teksatır\_tamamını\_okuma()

adres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open adres For Input As 1

Line Input #1, metin 'ilk satırı oku ve metin değişkeninde depola

Debug.Print metin 'volkan,38,ismail yazar

Close 1

End Sub

##### x adet satırı tek tek okuma

Bu işlem Line Input'un bir For Next döngüsü ile kullanımı ile yapılabilir. Belli sayıda satır bilgisinin yeterli olduğu durumlarda kullanılır.

Sub x\_adet\_satır\_oku()

adres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open adres For Input As 1

For i=1 to 2

Line Input #1, metin

Debug.Print metin

Next i

Close 1

End Sub

##### Dosyadaki tüm metni okuma

###### 1.Yöntem: Dosyadaki karakter sayısı kadar okumak

Bunu **LOF** ifadesi ile yapıyoruz. Bu, Length Of File'ın kısaltılmışıdır, yani dosyadaki karakter sayısını verir. Input ile birleştirerek de dosyadaki tüm karakter sayısını oku demiş oluruz. "volkan naber" şeklinde 12 karakterli bir metni içeren bir dosyada;

LOF(1): 12 döndürür  
Input(LOF(1),1): 1 nolu dosyayı tamamen okur: "volkan naber"

Örnek bir kodumuz ise şöyledir. Bu kodda ayrıca **Seek** ifadesini de kullandık. Bununla dosyada belirli bir sıradaki karaktere konumlanıyoruz, ki bunu genelde belirli bir sırayla ilerledikten sonra tekrar ilk karaktere dönmek için kullanırız. Syntax: **Seek dosyano, konum**

Sub DosyaOkuTümü1()

adres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open adres For Input As 1

Debug.Print LOF(1) 'Length Of File Of File

içerik = Input(LOF(1), 1) '1 nolu dosyanın hepsini oku

Debug.Print içerik

Seek 1, 1 '1 nolu dosyanın 1.karakterine yan en başa git

içerik = Input(LOF(1) - 10, 1) '1 nolu dosyanın son 10 karakteri hariç oku

Debug.Print içSeek 1, 5 '1 nolu dosyanın 5.karakterine git

içerik = Input(10, 1) '1 nolu dosyanın 5.karakterinden sonraki ilk 10 karakterini oku

Debug.Print içerik

Close 1

End Sub

Aynı mantıkla bir şekilde dosyanın ilk x karakterini okumak için içerik değişkenine **Input(x,1);** son x karakteri hariç okuma yapmak isterseniz içerik değişkenine **Input(LOF(1)-x,1)** şeklinde atama yaparsınız.

Bu yöntemde dosya içindeki metnin kaç satırda yer aldığı önemli değildir. Tüm metin tek bir değişkende depolanır.

###### 2.Yöntem: Dosya sonuna kadar satır satır okumak

Yukarıda gördüğümüz belli sayıdaki satırları tek tek okumadan farklı olarak tüm satırları tek tek okuyoruz. Satırların bittiğini **EOF**(End Of File'ın kısaltması )özelliği ile anlıyoruz.

Sub DosyaOkuTümü2()

tamadres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open tamadres For Input As 1

Line Input #1, satırmetni

içerik = satırmetni

Do Until EOF(1)

Line Input #1, satırmetni

içerik = içerik & vbNewLine && satırmetni

Loop

Debug.Print içerik

Close 1

End Sub

NOT: İçerik değişkenini oluştururken kayıtlar arasına **vbNewLine** koyarak satırbaşı yapıyoruz. Ancak ilk başta değişkenin içi boş olacağı için fazladan bir boş satır oluşmaması için en başta bir kezliğine döngüye girmeden ilk satırın atamasını yapıyorum sonrasında döngü içinde **vbNewLine** ekliyorum.

Bu arada istenirse ilgili metinler **vbNewLine** denmeden satır satır değil de ardışık bir şekilde de bir araya getirilebilir.

##### Tüm içeriği bir diziye aktarmak

Bunu da kendi içinde iki ayrı yöntemle yapabiliriz. İlk yöntemde satır satır okur ve her satırı bir collection’a atarız. Özellikle her satırın başına/sonuna başka bir metin eklemek gereken durumlarda bunu kullanabiliriz. İkinci yöntemde ise tüm metni okuyup Enterları(vbCrLf veya vbNewLine) Split ederek diziye atayabilirsiniz.

'1)Collectiona atama

Do Until EOF(1)

Line Input #1, metin

coll.Add metin

Loop

'2)Diziye atama

içerik=Input(LOF(1), 1)

dizi = Split(içerik, vbCrLf)

Bu iki yöntemle de yukardaki örneği yapalım.

Sub dosyaoku3()

adres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open adres For Input As 1

Debug.Print "------önce dizi yöntemi-----"

içerik = Input(LOF(1), 1)

dizi = Split(içerik, vbCrLf)

For Each satır In dizi

Debug.Print "Prefix: " + satır + " -Suffix"

Next satır

'veya

Debug.Print "------collection yöntemi-----"

Seek 1, 1

Dim coll As New Collection

Do Until EOF(1)

Line Input #1, metin

coll.Add metin

Loop

For Each satır In coll

Debug.Print "Prefix: " + satır + " -Suffix"

Next satır

Close 1

End Sub

##### Virgülle ayrılmış metinleri hücrelere yazdırmak

Virgülle ayrılmış tüm değerleri farklı kolonda olacak şekilde satır satır Excel’e yazdırmak isteyebilirsiniz. Bunun için satır satır okuma yapmamız gerekir ve her satırı Split ile bir diziye atayabilir, sonra da ilgili hücrelere bu dizi elemanlarını döngüsel şekilde yazdırabiliriz.

Sub DosyaOkuTümü4()

tamadres = "C: \Users\volkan\Desktop\denemeler\dosya1.txt"

Open tamadres For Input As 1

i = i + 1

Do Until EOF(1)

Line Input #1, x

dizi = Split(x, ",")

For j = 0 To UBound(dizi)

Cells(i, j + 1) = dizi(j)

Next j

i = i + 1

Loop

Close 1

End Sub

#### Dosyaya veri yazma

##### Sıfırdan yazma

Dosyaya veri yazdırmak için dosyayı **Output** modunda açmamız gerekir. Yazdırma eylemi için iki fonksiyonumuz var. **Write** ve **Print**. Write, yazdırılan ifadeyi " " içine alarak yazarken Print böyle bir işlem yapmaz.

Her iki deyim de ardışık kullanımlarında satır satır yazdırır. Yani birbirini takip eden metinler şeklinde yazılmaz. Örneğin;

ad="Volkan"

soyad="Yurtseven"

Print ad

Print soyad

'bu kodun çıktısı aşağıdaki gibidir

'Volkan

'Yurtseven

'VolkanYurtseven değil

Eğer yazdırılan metinlerin ardışık yazdırılması isteniyorsa aşağıda anlattığım **TextStream** nesnesini kullanmanız gerekir.

Şimdi veri yazdırmaya ait küçük bir kod yazalım.

Sub DosyaYazTekSatır()

tamadres = "C: \Users\volkan\Desktop\denemeler\dosya2.txt

ff = FreeFile ' o an uygun olan dosya numarası verilir.

Open tamadres For Output As ff

Print #ff, "bu ilk satır"

Print #ff, "bu da ikinci satır"

Close ff 'kapatırken kaydeder, ayrı bir save işlemi yoktur

End Sub

**NOT**: Eğer dosya mevcut değilse otomatikman oluşturulur, varsa ezilir ve üzerine yazılır.

##### Varolan dosyaya ekleme yapmak

Varolan bir dosyanın sonuna ekleme yapmak istiyorsak bu işi **Append** deyimi ile yaparız. Eğer dosya mevcut değilse Outputta olduğu gibi otomatikman oluşturulur.

Sub DosyaAppend()

tamadres = "C: \Users\volkan\Desktop\denemeler\dosya2.txt

ff = FreeFile ' o an uygun olan dosya numarası verilir.

Open tamadres For Append As ff

Print #ff, "bu üçüncü satır"

Close ff 'kapatırken kaydeder, ayrı bir save işlemi yoktur

End Sub

##### Excel’den okuyup dosyaya yazma

İçiçie iki For Next ile satır ve sütunlarda dolaşırız, araya "," veya istediğimiz başka bir ayraç ekleriz.

Aşağıdaki örnekte 10 satır 3 kolondan oluşan bir listeyi metin dosyasına yazdırıyoruz.

Sub Excel’denyaz()

tamadres = "C: \Users\volkan\Desktop\denemeler\dosya3.txt

ff = FreeFile

Open tamadres For Output As ff

For i = 1 To 10

For j = 1 To 3

If j = 3 Then

satırmetin = satırmetin + Trim(Cells(i, j).Value)

Else

satırmetin = satırmetin + Trim(Cells(i, j).Value) + ","

End If

Next j

Print #ff, satırmetin

satırmetin = ""

Next i

Close ff

End Sub

##### Metin değişikliği yapmak

Bazen tek bir dosyada bazense birçok metin dosyasında aynı anda bir metin değişikliği yapmak isteriz. Birçok dosyayı elde etmeyi önceki konuda görmüştük. Bir döngü ile bu klasörleri/dosyaları elde ettikten sonra yapmanız gereken iş 4 aşamadan oluşur:

* Önce dosyayı açmak
* Metni okumak
* Aradığımız metni bulup değiştirmek
* Dosyaya tekrar yazdırmak

Sub MetinDeğiştir()

Dim adres As String

Dim içerik As String

adres = "C: \deneme\deneme.txt"

Open adres For Input As 1

içerik = Input(LOF(1), 1)

Close 1

içerik = Replace(içerik, "abc123", "abc345")

Open adres For Output As 1

Print #1, içerik

Close 1

End Sub

Yapılan değişikliği illa dosyaya kaydetmek zorunda değilsiniz. Mesela benim bazı SQL'leri tuttuğum metin dosyalarım var, içinde değişkenlerin olduğu bölümler var. Bu dosyaları bir nevi şablon olarak tutuyorum, onların üzerinde değişiklik yapmıyorum, onun yerine dosyayı okuyup bir değişkene atıyorum ve onun üzerinde replace işlemi yapıp, SQL metni olarak işleme sokuyorum. En aşağıdaki örnekler bölümünde bu işlemi görebilirsiniz.

UYARI/ÖNERİ: Yapılandırılmış tipteki(belli format, uzunluk ve kolonlardan oluşan) büyük metin dosyalarında büyük çaplı değişiklikler yapılacaksa bu dosyaya ADO ile bağlanıp Update işlemi yapılması daha hızlı sonuç verecektir. I/O yöntemi ile replace işlemi küçük dosyalarda tercih edilmelidir.

### TextStream nesnesi

Dosyalara yazma ve okumanın bir diğer yolu da **TextStream** nesnesi yoluyladır. Niye böyle bir yöntem daha var? Bu sınıf, aslında web sayfalarında **VBscript** diliyle yazılmak üzere tasarlanmış bir sınıftı ama sonradan VBA içinde de kullanıma alındı. O yüzden **Scripting** Runtime librarysi içindedir ve  references menüsünden eklenmesi gerekir.

Ben şahsen hem okunurluk hem de kullanım kolaylığı açısından TextStream nesnesini kullanmayı tercih ediyorum, ancak her zaman olduğu gibi başkalarının yazdığı kodları okumanız/kullanmanız gerekebileceği için her yöntemi bilmekte fayda var. Aşağıda göreceğiniz üzere TextStream'in bazı ek özellik ve metodları da onu ayrıcalıklı kılmaktadır.

#### Erişim & Yaratım

Bir TextStream nesnesine erişmek için **FSO'**nun **CreateTextFile** veya **OpenTextFile** metotlarını kullanabileceğimiz gibi **File** nesnesinin **OpenAsTextStream** nesnesini de kullanabiliriz.

##### CreateTextFile

**Syntax:** fso.CreateTextFile(dosyadı,Owerrite?,Unicode desteği?)

Aşağıdaki kod ile var olan bir dosyayı, eğer mevcutsa üzerine yazdırarak(yani içini boşaltarak) Türkçe karakterleri de destekleyecek şekilde açıyoruz.

'global fso nesnesnin var olduğunu düşünerek ilerliyoruz

Dim ts As TextStream

Set ts = fso.CreateTextFile("c: \deneme\deneme.txt", True, True)

Eğer ikinci parametreyi False olarak kullanmak yani dosya mevcutsa onu ezmeyelim istiyorsak, aşağıdaki gibi dosyanın varlığını kontrol ederek açmalıyız yoksa "dosya zaten mevcut" hatası alırız.

If Not fso.FileExists("C: \Users\Volkan\Desktop\denemeler\deneme1.txt") Then

Set ts = fso.CreateTextFile("C: \Users\Volkan\Desktop\denemeler\deneme1.txt", False, True)

ts.Write ("merhaba")

End If

##### OpenTextFile

**Syntax:** fso.OpenTextFile(dosyadı,I/O tipi,MevcutdeğilseYaratılsınmı?,Format)

Set ts = fso.OpenTextFile("C: \deneme\deneme.txt",ForWriting,True,TristateFalse)

I/O tipi, **ForWriting** açıldığında içerik ezilir. Bu CreateTextFile'ın ikinci parametresinin True olarak açılmasıyla aynı etkidedir.

**ForReading** ile okuma yaparsınız, yazmaya izin verilmez.

**ForAppending** ile en sona konumlanır ve oraya yazarsınız, böylece mevcut içerik silinmemiş olur.

Üçüncü parametreyi, dosya mevcut değilse yaratmak istediğinizde True olarak kullanırız. Eğer burası False ise ve aradığınız dosya yoksa hata alırsınız. O yüzden ya burayı True yapmalısınız ya da dosyanın mevcut olup olmadığını kontrol etmelisiniz. Mesela aşağıdaki kod ile, dosya mevcut ise sonuna ekleme yapmak istiyoruz, mevcut değilse yaratarak açıyoruz.

If Not fso.FileExists("C: \Users\Volkan\Desktop\denemeler\deneme2.txt") Then

Set ts = fso.OpenTextFile("C: \Users\Volkan\Desktop\denemeler\deneme2.txt", ForWriting, True, TristateFalse)

Else

Set ts = fso.OpenTextFile("C: \Users\Volkan\Desktop\denemeler\deneme2.txt", ForAppending, False, TristateFalse)

End If

Son parametre Unicode desteği ile olup olmayacağını verir.

##### OpenAsTextStream

Elinizde bir File nesnesi varsa bunun **OpenAsTextStream** metodunu kullanarak da metin dosyalarını açabilirsiniz. Gerçi File nesnesi için de yine bir FSO nesnesi gerekiyor. O yüzden her ikisini de yaratmak gerekecek. Eğer File nesnesini başka birşey için kullanmayacaksanız boşuna bu zahmete gerek yok, direkt FSO ve onun metodları yeterli olacaktır.

**Syntax:** File.OpenAsTextStream(I/O modu,Format)

İki parametre de opsiyonel olup default değerleri sırasıyla ForReading ve TristateFalse'tur. Aşağıda bir örnek bulunmakta.

Dim f As File, ts1 As TextStream, ts2 As TextStream

Set f = fso.GetFile("C: \Users\Volkan\Desktop\denemeler\deneme2.txt")

Set ts1 = f.OpenAsTextStream 'default değerlerle açıldı

x = ts1.ReadAll

ts1.Close

Set ts2 = f.OpenAsTextStream(ForAppending, TristateMixed)

y = ts2.Write("yeni")

ts2.Close

#### TextStream Üyeleri

##### Metin okuma şekilleri

ts.Read(5) 'Bulunulan yerden itibaren 5 karakter okur

ts.ReadLine 'Bulunulan satırı okur

ts.ReadAll 'Tüm dosya içeriğini okur

Üç yöntemde de bir değişkene atama işlemi yapılmalıdır.

Ör: içerik=ts.ReadLine

##### Yazma şekilleri

ts.Write(metin): Dosyaya metni yazar

ts.WriteLine(metin): Dosyaya metni yazar ve bir alt satıra geçer

ts.WriteBlankLines(5): Dosyaya 5 adet boş satır ekler

**Line** ile cursor'ın o anki satır numarasını elde ederiz.

**Close** metodu ile TextStream nesnesini kapatırız.

Dosyada okuma yaparken, belirli koşullar durumunda o satırı **SkipLine** ile atlayarak bir sonraki satıra geçebiliriz. Aşağıdaki kod ile, sayısal bir ifadeyle başlayan her şeyi bir collectiona atayıp en son da bunları yazdırıyoruz. Read ile bir karakter okuduktan sonra kalanını ReadLine yaparken başına ilk okuduğumuz kısmı eklediğimize dikkatinizi çekmek isterim. Örnek dosyamız aşağıdaki gibi olsun

1-birinci satır

2-ikinci satır

falanfilan

3-üçüncü satır

falanfilan

4-dördüncü satır

Sub Satıratla()

Dim ts As TextStream

Dim col As New Collection

Set ts = fso.OpenTextFile("C: \Users\Volkan\Desktop\denemeler\deneme1.txt", ForReading, False, TristateMixed)

Do

kelime = ts.Read(1)

If IsNumeric(kelime) Then

col.Add kelime + ts.ReadLine

Else

ts.SkipLine

End If

Loop Until ts.AtEndOfStream

For Each Item In col

Debug.Print Item

Next Item

End Sub

Çıktı ise şöyle olacaktır:

1-birinci satır

2-ikinci satır

3-üçüncü satır

4-dördüncü satır

Hepsi bir arada bir örneğimi aşağıdaki gibi olabilir:

Sub çeşitli\_üyeler()

Dim ts As TextStream

Const dosya As String = "C: \Users\Volkan\Desktop\denemeler\ts\_üyeler.txt"

Set ts = fso.CreateTextFile(dosya, True, True)

ts.WriteLine ts.Line & "-" & Now

ts.Write ts.Line & "-": ts.WriteBlankLines (1) 'Dosyaya 1 adet boş satır ekler

ts.WriteLine ts.Line & "-" & Environ("username")

ts.WriteLine ts.Line & "-" & "selam"

ts.WriteLine ts.Line & "-" & "naber"

Debug.Print ts.Line

ts.Close

Set ts = fso.OpenTextFile(dosya, ForReading, False, TristateMixed)

x = ts.ReadLine ' Bulunulan satırı okur

ts.skipline 'ilgili satırı atlar

y = ts.Read(5) 'Bulunulan yerden itibaren 5 karakter okur, artık 3. satırdayız: 3-Vol

z = ts.ReadAll 'Cursordan itibaren tüm dosya içeriğini okur, baştan itibaren değil : kan4-selam5-naber

Debug.Print z

End Sub

### Olay/Hata Logu tutan bir uygulama(Logger Prosedürü)

Her gün belli saatlere schedule edilmiş(Application.Ontime aracılığı ile) makrolarınızın olduğunu düşünün. Bunlar içinde çeşitli aşamaları gün/saat başta olmak üzere diğer önemli bilgilerle birlikte kayıt altına almak, nerde hata alınmış, bunları görmek isteyebilirsiniz, hatta iyi bir programcı olarak görmek istemelisiniz.

Keza, bölümünüz için raporlara ulaşım amacıyla hazırladığınız bir arayüz olması durumunda, kim ne zaman hangi rapora girmiş, en çok hangi rapor kullanılıyor, Kokpiti en çok kim kullanıyor gibi soruların cevabını elde etmek için bir log kaydı da tutmak isteyebilirsiniz.

İşte bu amaçlarla dosya yazma/okuma işlemlerini kullanabiliriz.

#### Otomasyon süreçlerinde Logger kullanımı

Diyelim ki aşağıdaki prosedür günün belirli saatlerinde çalışıyor. Çalışmanın belirli aşamalarını(kritik önemde veya ana işlerin öncesinde/sonrasında) kayıt altına alıyoruz. Ayrıca bir hata oluşursa yine bunu da kayıt altına alalım.

Otomasyon süreçlerinde Log tutmanın bir alternatifi kendinize veya ilgili kişilere mail attırmak olacaktır. Ancak çalışan çok fazla iş varsa mail kalabalığında boğulursunuz. O yüzden log sistemi daha güzel bir seçenektir.

Şimdi aşağıdaki örnekte Log dosyamızda Tarih/Saat, Kullanıcı, bilgisayar adı, rapor adı, log tipi, varsa hata kodu, açıklama kolonları olmak üzere 7 kolon bilgi bulunmaktadır. Bunun ilk 3'ü Logger fonksiyonu içinde dinamik olarak ele alınmakta, son 4 parametre ise Logger fonksiyonuna KrediRaporu modülünden argüman olarak gönderilmektedir. Diğer hususlar şöyledir.

Rapor ismi toplamda 50 hane olacak şekilde ayarlanır. 50den kısa olan rapor isimleri için başına 50ye tamamlanacak kadar boşluk eklenir. Bunun amacı datayı Excel’e aktardığınızda aynı hizada görünmeleri içindir. Bu sizin dünyanızda daha yüksek bir sayıya ayarlanabilir.

Bilgisayar ismi de yine aynı şekilde 10 haneye tamamlanmaktadır. Bu da sizin dünyanızda daha yüksek ayarlanabilir.

Hata yoksa hata kodu olarak 0 gönderilmektedir.

'\*\*\*\*\*Logger'ı çağıran prosedür\*\*\*\*\*

Sub KrediRaporu()

On Error GoTo hata

raporLoggerAd="KrediRaporu"

'çeşitli işler

Logger WorksheetFunction.Rept(" ", 50 - Len(raporLoggerAd)) & raporLoggerAd, "OK", 0, "Bölme işlemi başlayacak"

'çeşitli işler

Logger WorksheetFunction.Rept(" ", 50 - Len(raporLoggerAd)) & raporLoggerAd, "OK", 0, "Bölme işlemi bitti"

'çeşitli işler

Logger WorksheetFunction.Rept(" ", 50 - Len(raporLoggerAd)) & raporLoggerAd, "OK", 0, "Rapor başarıyla çalıştı"

Exit Sub

hata:

Logger WorksheetFunction.Rept(" ", 50 - Len(raporLoggerAd)) & raporLoggerAd, "Hata", Err.Number, Replace(Err.Description, vbNewLine, vbNullString)

End Sub

'\*\*\*\*\*Logger prosedürümüz\*\*\*\*\*

Sub Logger(rpr As String, logtip As String, hatano As Integer, açıklama As String)

On Error GoTo hata

Dim dosya As String

Dim dosyano As Variant

dosyano = FreeFile

dosya = gunlukklasor + "\GünlükRaporlarLog.txt"

Open dosya For Append As #dosyano

Print #dosyano, CStr(Now), Environ("UserName"), WorksheetFunction.Rept(" ", 10 - Len(Environ("computername"))) & Environ("computername"), rpr, logtip, hatano, açıklama

Close #dosyano

Exit Sub

hata:

Call mail\_logger\_hata(rpr, alicilar) 'Log prosedüründe bir şekilde hata önceden berlirlenmiş alınırsa alıcılara özel formatta mail atılır

End Sub

#### Kokpit uygulamalarında Logger kullanımı

Userform konusunda göreceğimiz Kokpit uygulamalarında, uygulamayı kullanan kişilerin aktivitelerini aşağıdakine benzer bir kod ile kayıt altına alabiliriz.

'Form üzerindeki bir butona tıklanınca

Sub Btn\_KrediRaporAc()

On Error Goto hata

'rapor açma kodları

Rapor="KrediRapor"

frekans="Günlük"

Call detayraporlogu(Rapor, frekans)

Exit Sub

hata:

On Error Goto -1

On Error Goto hata2

'burada hata kaydını tutan bir log kaydı(aşağıdaki log prosedürünü gölgede bırakmasın diye detayına girmedim)

Exit Sub

hata2:

'Diskte yer olmaması, veya kullanıcının ilgili diske yazma yetkisinin olmaması gibi bir sebeple hata olması durumunda

Call LogHata 'bu sefer maille size bilgilendirme yapılır

End Sub

'\*\*\*\*\*Logger prosedürümüz\*\*\*\*\*

Sub detayraporlogu(ByVal Rapor As String, ByVal frekans As String)

If Environ("UserName") = sizinuserınız Then Exit Sub 'kendimizi loglamıyoruz

i = FreeFile

Open adres & "Kokpitlog\_detayrapor.txt" For Append As i

Print #1, Environ("UserName"), Date, Time, frekans, Rapor

Close #1

End Sub

Daha sonra bu text dosyasını bir Excel dosya içine aktarır veya her açıldığında refresh olan bir bağlantı kurarak gelen data üzerinde pivot tablolarınızı oluşturabilirsiniz. Metin dosyalarından bağlantı kurmak için Veritabanı Programlama konusuna bakabilirsiniz.

##### Logger içinde hata

Bir sebeple logger fonksiyonu içinde de hata olursa bunu da başka bir hata bloğuyla ele alabilirsiniz. Veya ana gönderici modülde On Error GoTo -1 deyip 2. bir hata bloğu açabilirsiniz. Yukarıdaki ilk örnekte Logger fonksiyonu içinde hata bloğu ile yakaladık. İkinci örnekte ise ana prosedürde On Error GoTo -1 yöntemini kullandık.

### Çeşitli Örnekler

#### Settings işlemleri

Bir dosyadan bir database'in kullanıcı adı ve şifresini okuma, veya bir dosyanın path'ini okuma gibi işlemler de bu sayfada öğrendiklerimizle yapılabilir.

Diyelim ki jenerik bir Add-in yaptınız. Bu Add-indeki makrolardan bir tanesi bir klasördeki bir Excel dosyasını açacak. İşte bu Excel dosyanın yerinin sabit olmasının mümkün olmadığı, bunun hangi klasörde olacağını kullanıcıya bırakmanız gereken durumlar olabilecektir. SettingforAddin1.txt gibi bir dosya içine bu klasörün tam path'i yazılabilir. Hatta buna birden fazla dosya için birden fazla klasör de eklenebilir. İstenirse ";" ile ayırılır, istenirse satır satır yazılır, hiç fark etmez. Yukarıdaki yöntemlerden biriyle ilgili adresi elde etmek oldukça kolaydır.

#### SQL metinlerini değiştirme

Diyelim ki raporlama araçlarınız çok hantal ve katı. Siz de gerek kendiniz gerek departmanınız için Excel içinden çalışan hızlı ve esnek bir raporlama platformu oluşturdunuz. İlgili raporların SQL'ini bir metin dosyası içine koydunuz. Kullanıcıya tarih ve müşteri listesi gibi sorular sordurarak dosyadaki parametrik kısımlarla kullanıcının verdiği cevapları replace ettirerek nihai SQL'inizi elde edersiniz. Böylece uzun bir SQL'i VBA içine satır satır yazmaktan kurtulmuş olursunuz. VBA içine de SQL kodu yazılabilir ama bu hem kodun uzun ve çirkin görünmesine neden olur hem de çok zahmetli bir iştir, özellikle SQL onlarca hatta yüzlerce satırdan oluşuyorsa.

Kodumuz şöyle olabilir:

Sub SQLDeğiştir()

tarih=InputBox("tarihi girin")

If tarih=vbNullString Then Exit Sub

adres="C: \SQLller\kredi.txt"

Open adres For Input As #1

içerik=Input(LOF(1),1)

Close #1

strSQL=Replace(içerik,"trh",tarih) 'SQLi elde ettik

'bundan sonra SQL'i çalıştıracak kodlar devreye girer

End Sub

## Neler Öğrendik

Dosya ve klasörlere erişim nasıl yapılır, bunlarla ilgili kopyalama, silme gibi işlemler nasıl yapılır, bunları öğrendik.

Text dosyaları nasıl okunur, nasıl oluşturulur, metin eklenir, içleri nasıl değiştirilir, bunları gördük.