# 1.bölüm

**1**

# MAKROLARA GİRİŞ

Bu ilk bölümde makroların ne işe yaradığından ve çalışmaya başlamadan önce nasıl bir organizasyon(dosya ayarları) yapacağımızdan bahsedeceğiz. Sonrasında VBE denen geliştirme ortamını(IDE) tanıyacağız ve Makro kaydetme aracından nasıl faydalanacağımızı göreceğiz. Son olarak da makroları daha iyi anlamak için Excel Nesne Modelini yakından tanıyacağız.

Konuların öyle bir özelliği var ki, birini anlatırken henüz anlatmadığımız temel bir yapıyı o konu içinde kullanmamız gerekebilir. O an için detayını çok anlamanıza gerek yoktur, yeri gelince detaylarıyla zaten öğrenmiş olacaksınız. Mesela koşullu yapılar olan IF blokları ve döngüsel yapılar olan For-Next yapıları sık sık kullanılır. Ancak örneklerimiz daha çok Excel'in temeli olan hücreler ve hücrelerin bulunduğu sayfalar üzerinden olacağı için hücrelerin detayının bulunduğu Range nesnesini ve sayfaların detayının bulunduğu Worksheet nesnesini bu yapılardan daha önce işleyeceğiz. Eğer ki IF ve For-Next konusunu daha öne çekseydim bu sefer Range nesnesinin nasıl işlediğini merak edecektiniz. O yüzden tercihimi bu yönde kullandım. Ancak istediğiniz zaman ilgili sayfalara gidip ara bir bilgi edinme yöntemini de kullanabilirsiniz

Şimdi hepsinden önce “Makro nedir” ile başlayacağız. İkinci bölümde temellere ineceğiz. Üçüncü bölümden itibaren hız kazanacağız.

İlk 2 bölümü sağlam temel atma adına oldukça önemli buluyorum, ancak ilk başlarda anlamayacağınız yerler olabilir, o yüzden kitabın ortalarına gelindiğinde tekrar tekrar bakılması gereken bölümlerdir.

**NOT**: Bu arada **Önsözü** okumadıysanız lütfen okuyun.

## Makro Nedir?

### Tanım

#### Makro vs VBA

Bu kitapta "makro yazmayı öğreneceksiniz" diyebiliriz ancak **Makro** ile eş anlamlı kullanılan bir ifade de VBA'dir(vi bi ey okunur). VBA, Visual Basic(VB) Programlama dilinin Office programları için uyarlanmış bir versiyonudur. Excel makrolarını yazacağımız dil de **VBA(Visual Basic Applications) for Excel** olarak bilinir, yani Excel için Visual Basic uygulamaları.(Access, Outlook ve PowerPoint için de VBA mevcuttur ancak en popüleri Excel için olanıdır)

Makro terimi yalnızca Office programlarına özgü bir ifade değildir aslında, birçok programda Makro özelliği bulunabilir. Özünde makro, adımlardan oluşan bir program bütündür. Excel için söyleyecek olursak, bir hücreyi seçmek, içine bir değer girmek, bir sayfayı silmek gibi.

Makroları yazarken VBA kullanacağımız için de genel olarak Makro ve VBA ifadeleri birbirinin yerine kullanılabilir ama şunu söyleyebilirim ki VBA ifadesini kullanmak sizi biraz daha profesyonel gösterir.

#### Ne işe yarar?

Basitçe Excel’in zaten güçlü olan işlevselliğini daha da ileri götürmek gibi bir temel görevi olduğunu söyleyebileceğimiz Makroların, belli başlı kullanım amaçları sonraki paragrafta verilmiştir, tabiki bunlar çoğaltılabilir.

Bunlara geçmeden önce söylemeden geçemeyeceğim bir husus var. Makroların gücünü tam olarak bilmeyenlerin/görmeyenlerin genelde Excel’i ve VBA'i/Makroları küçümseme gibi bir huyu vardır. Onlara göre bırakın VBA’i, VB bile doğru dürüst bir programlama dili değildir. Evet, VB'den daha gelişmiş dillerin olduğu doğru ancak, elimizdeki tool Excel ise Excel’in işlevselliğini de ileriye götüren en güçlü ve kolay ulaşılabilir tool da VBA'dir. Her ne olursa olsun, VBA ile yapılabileceklerin de bir sınırı var, ki bence bu sınırlar oldukça geniştir, ancak kimsenin VBA ile bir masaüstü program yazma veya Facebook gibi bir site yapma iddiası da yok. Bir diğer uçta ise, öğrenmeye açık olmadığı için makrolarla neler yapılabileceğini hayal bile edemeyen kişiler var, ve bunlar da makroların gücünü önemsemezler. Bu iki uç grubu biz de görmezden gelelim, olumluya odaklanalım ve makrolarla hem kendimizi hem de bu iki grup dışındaki çoğunluğu oluşturan insanları nasıl büyüleyeceğimize bir bakalım:

* Rutin yapılan işler başta olmak üzere birçok işin otomatiğe alınması(Böylece hatasız ve aşırı hızlı işlem yaparsınız)
* Excel’in kendisiyle yapılamayan veya yapılsa bile aylar yıllar sürecek döngüsel işlemler gibi işlemlerin yapılması(Ör: 1000 satırın her birine ayrı ayrı goal seek yapmak gibi)
* Bazı durumlarda yetersiz kalan Excel fonksiyonlarının yerine kendi fonksiyonlarınızı yazabilirsiniz
* Makrolar iş yükünüzü azaltır, siz yokken bile çalışabilir. Kurumunuza verimlilik sağlar, maliyetleri düşürür.
* Bazı durumlarda insanların size Uzaylı gibi bakmasını sağlar 😊
* Beyninizin sürekli çalışmasını, dolayısıyla yeni sinir bağlantılarının oluşmasını sağlar. Bu da daha geç bunayacağınız ve üstelik daha az kilo alacağınız anlamına gelir, evet kilo almak dedim, çünkü beyin, vücudun en çok enerji tüketen organıdır ve ne kadar çok çalışırsa o kadar çok enerji tüketir 😊

Daha özele inecek olursak şunları yapabilirsiniz

* Satış ve performans takip raporları oluşturulabilir
* Çok kanallı/şubeli kurumlarda her alıcıya kendisiyle ilgili dosyaların gönderimi sağlanabilir
* Çeşitli frekanslarda raporların belli gün ve saatlerde kendiliğinden çalışıp refresh olması sağlanabilir
* Uyarı mekanizmaları kurulabilir
* Çeşitli yerlerden toplanan dosyalar birleştirilebilir
* Bütçeler oluşturulabilir
* Dashboardlar oluşturulabilir
* İş ararken sizi diğerlerinin önünde tutar 😊
* Bunamanızı yavaşlatacağı için yaşlandığınızda torunlarınızın adını hatırlayabilirsiniz 😊

Tabiki, Excel'in her yeni versiyonu ile yazdığımız bazı makrolar gereksiz hale gelebilmektedir ve sizin için de bu geçerli olacaktır. Mesela, yıllar önce bir hücrenin formül içerip içermediğini kontrol eden bir UDF(fonksiyon) yazmıştım, ancak artık Excel’in 2013 versiyonunda bu formül dahil edilmiş. O yüzden bu kod artık gereksizdir. Keza yine, Pivot Table yaptıktan sonra aradaki boşlukları otomatik dolduran bir makrom vardı, bu da yine 2010 versiyonu ile birlikte gereksiz hale geldi. Ama karamsar olmayın, Microsoft çalışanları ne yaparsa yapsın kuruma özgü, spesifik ihtiyaçları karşılayan araçlar geliştiremezler, o yüzden VBA bilen birisi olarak yine her zaman el üstünde kalmaya devam eder, iş ararken avantajınızı korursunuz.

#### Ne bilmek gerekiyor?

Visual Basic bilenler için makro öğrenmek çok daha basit olmakla birlikte ilk defa makro öğrenecek kişilerin gidip de öncelikle VB öğrenmesine gerek bulunmamaktadır. Bu kitap zaten size direkt olarak makro yazımını öğretmeyi hedeflemektedir.

Bu kitaptaki öğrenme sürecinize paralel olarak, Excel’in Makro Kaydetme özelliği ile basit denemeler yapabilirsiniz. Böylece yaptığınız her hareketin sonunda nasıl bir kod ortaya çıktığını takip edebilirsiniz. Zaten ne kadar profesyonel olursanız olun yeri geldiğinde Makro Kaydet aracını kullanmanız gerekecek. Sonra ihtiyaca göre oluşan kodda istediğiniz düzenlemeleri yapabilirsiniz.

### Kodlama öncesi son hazırlıklar

#### Organizasyon

Yazdığınız makrolara sık sık ulaşmak isteyecek, Add-in aktifleştirme/pasifleştirme gibi işlemleri yapacak ve tabiki makro kaydetmek isteyeceksiniz. Bunların ne anlama geldiğini bilmiyor olabilirsiniz ama emin olun iyi şeyler. İşte bunları yapmak için öncelikle Ribbon’a sağ tıklayarak Developer sekmesini etkinleştirelim, ayrıca VBE’yi(VisualBasic düzenleyicisini) QuickAccess Toolbarına almanızda fayda var.(Alt+F11 tuşuyla da ulaşılabilir)

Makrolarınızı derli toplu tutmanın birkaç yolu bulunmaktadır. Ben sizlere, bunlardan Personal.xlsb ve Add-in yaratma yöntemlerinden bahsedeceğim. Aslında ikisinin de kendine özgü amaçları vardır. Kendimin, hangisini ne amaçla kullandığımı söylersem size de ışık tutacaktır diye düşünüyorum.

Çalıştığım kurum içinde diğer kişilerle de paylaşacağım makrolar varsa bunları Add-in olarak hazırlarım, hazırladığım makroların bir menü olarak Ribbon’da gözükmesini istediğim için de bu Add-in içine de menüleri oluşturan bir ‘başlangıç’ makrosu yazarım, en sonunda Add-ini diğer kişilerle paylaşırım. Add-in yöntemiyle diğer kişilere ne yapmaları gerektiğini anlatmak daha kolay olur. Onlara sadece Add-in'i nasıl kurmaları gerektiğini anlatan kısa bir mail atarım, ondan sonra makroları kullanmaları çok kolay olur.

Personal.xlsb dosyası ise daha çok kendinize özel makroları içerir, diğerlerinde bu makroların olmasına gerek yoktur. Benim Personal.xlsb dosyamda scheduling kodları, kısayol kodları, bana özel fonksiyonlar gibi özelleşmiş kodlar bulunur, bunların bir kısmını ilerleyen sayfalarda burada paylaşacağım. Kısayol tuşu atadığınız makroların bu dosyada bulunması çok önemli, çünkü Add-in içine koyarsanız ve başka kişiler başka amaçla bu kısayollara tıklarsa yanlış sonuçlarla karşılaşabilirler ve maalesef makroların Undo'su yoktur. Örneğin ben Copy&PasteSpecial-Value için Ctrl+M kombinasyonunu kullanırım. Bir hücredeki sayılara binlik ayraç uygulamak için Ctrl+L kısayolunu. Halbuki bu kısayolların bazılarında önceden Microsoft tarafından tanımlanmış başka görevleri var olabilir ve kişiler bu kısayol tuşlarını zaten o amaçlar için kullanıyor olabilir.

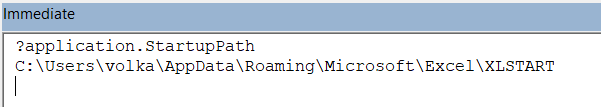
NOT: Excel, 2007 sürümünden sonra standart format içine makro kaydetmeye izin vermiyor, o yüzden "Personal" dosyanızı .xlsb uzantısıyla kaydetmelisiniz.(xlsm de olur ancak genel eğilim xlsb şeklindedir)

#### Personal.xlsb'nin konumu

Personal.xlsb dosyası herhangi bir dosya değildir. Bu dosyaya kaydedilen makroları bütün dosyalarınızda kullanabilirsiniz. Çünkü Excel oturumu boyunca hep açık kalmaktadır. Herhangi bir Excel dosyasındaki makrolar ise sadece o dosya açık olduğu sürece çalışacaktır.

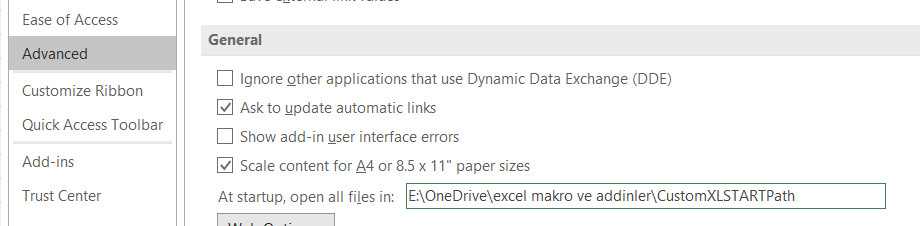
Peki bu dosya nasıl oluşturulur ve nereye kaydedilir? Şu adımları takip edin: *(Adımları sonuna kadar okuyup öyle uygulayın)*

* Boş bir Excel dosyası açın
* View menüsünden dosyayı gizleyin
* Excel’den çıkmaya çalışın
* Excel sizi uyaracak ve az önce gizlediğiniz dosyayı kaydedip kaydetmek istemediğini soracaktır
* Evet deyin ve XLSTART klasörüne kaydedin. Office sürümüne göre bu klasörün yeri değişebilmektedir, bunu tespit etmenin kolay bir yolu var
* Alt+F11 ile VBE’yi açın
* Immediate Window açık değilse Ctrl+G ile bunu açın ve oraya **Application.StartupPath** yazıp Enter'a basın.



Dosyanızı oluşturduktan sonra, her projeniz için ayrı bir modül oluşturmanızı(ilerleyen konularda anlatılacak) ve bu projeyle ilgili tüm prosedürlerinizi bu modül içinde bulundurmanızı tavsiye ederim.

**NOT:** Bazı kurumlarda, BT politikaları gereği bazı klasörlere erişim izni olmamaktadır, XLSTART klasörü de bunlardan biri olabilmektedir. Bu nedenle Personal.xlsb dosyanız için alternatif bir klasör belirlemeniz gerekebilir, bunu da **Excel Options>Advanced>General>At Startup open all files** in kutusuna yazarak belirtebilirsiniz.



**NOT**: Personal.xlsb dosyasını doğrudan Record Macro yaparak da oluşturabilirsiniz. Size makronuzun nereye kaydedileceği sorulur, Personal Macro Workbook(Türkçe Excel'de "Kişisel Makro Çalışma Kitabı") seçeneğini seçerseniz bu dosya otomatikman oluşur, ancak yukarda belirttiğim gibi BT politikanız XLSTART klasörüne erişim izni vermiyorsa sorun oluşabilir.

#### Güvenlik Ayarları

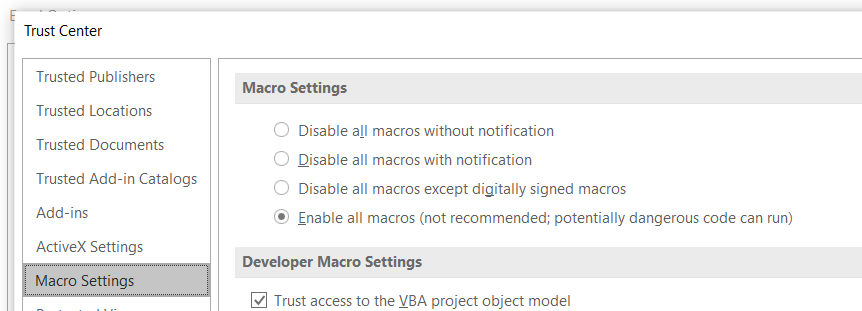
Makrolar, kötü amaçlar için kullanılabilir, tabiri caizse makrolar içine virüs yazılabilir. Mesela isterseniz(ki aslında istememelisiniz) bir add-in yazıp, belirli bir tarih geldiğinde kullanıcıların bilgisayarındaki önemli dosyaların silinmesini sağlayabilirsiniz.

Şunu düşünebilirsiniz; Ya makroyu bir düğmeye, menüye atamıyor muyuz, veya VBE üzerinde o makroya gelip de F5 yaparak çalıştırmıyor muyuz, biz bir şeye basmadan nasıl zararlı olabilir ki? Cevap basit: İleride göreceğiz ki Workbook eventlerinden biri de Workbook\_Open() eventidir(bir de Auto\_open() var) ve dosya açılır açılmaz aktive olur. İşte o sırada olanlar olur 😊

İşte bu nedenle makroların kullanılabilmesi için çeşitli güvenlik ayarları ve seçenekleri mevcuttur.

**File>Options>Trust Center>Trust Center Settings** düğmesine  tıklayarak bu ayarların olduğu yere geliriz.

Burada ilk göz atacağımız yer Makro Settings menüsüdür.



İlk seçenekle hiçbir makroya güvenmediğinizi belirtmiş olursunuz, makro içeren tüm dosyaların makrosu pasifleştirilir.

İkinci seçenek varsayılan seçenektir ve makro içeren bir dosya karşısında sizi uyarır. Siz de enable veya disable diyerek ilerlersiniz.

Üçüncü seçenek benim tercih ettiğim seçenektir. Eğer bir makro dijital olarak imzalandıysa ve Güvenilir Yayıncılar bölümüne eklendiyse uyarı çıkmadan makroyu etkin kılar, aksi halde size uyarı çıkarır, yani ikinci seçeneğin biraz gelişmiş şeklidir. Yalnız hiç imza yoksa direkt pasifleştirir.

Son seçenek ise pek güvenli değildir, zira bütün makroları aktif kılar, bu da sizi hackerlara açık hale getirir, o yüzden bilmediğiniz kaynaktan gelen ve güvenmediğiniz dosyalarla çalışırken makro ayarlarınızı kesinlikle buna getirin.

Tabi bu arada biraz yukarıda belirttiğim gibi Excel'in 2007 versiyonundan sonra standart dosya tipi(.xlsx) içine makro kaydedilememektedir, o yüzden standart Excel dosyaları güvenlidir diyebiliriz.

Biz şimdilik öğrenim sürecinde olduğumuz için son seçeneği aktif yapalım, ancak unutmayın, öğrenim ve test süreciniz bitince 3.seçeneği işaretleyelim.

#### İzinler

Evet, makro ayalarını yaptınız ama bir süre sonra Excel’in çıkardığı uyarılar canınızı sıkmaya başlar. İkide bir çıkan bu uyarılardan güvenlik sınırları içinde kurtulmanın yolları da var elbette. Şimdi bunları inceleyelim.

##### Dijital İmza

Dijital imzayı ticari bir kurumdan alabileceğiniz gibi kendinizin imzalayacağı sertifikalar(self-signed certificates(SSC)) da olabilmektedir.

Kendi kullanımınız için veya ekip arkadaşlarınız makrolarınızı kullanacaksa SSC kullanabilirsiniz. Bu, makronuzu kullanacak kişilere "bana güven; ben, kim olduğumu söylediğim kişiyim" demek oluyor.

Ticari kurumlardan alınacak imzalar ise daha güvenlidir ve makronuzu kullanacak kişiye "Bana güven, X firması benim kim olduğumu biliyor ve benim o kişi olduğumu teyit ediyor" demek oluyor.

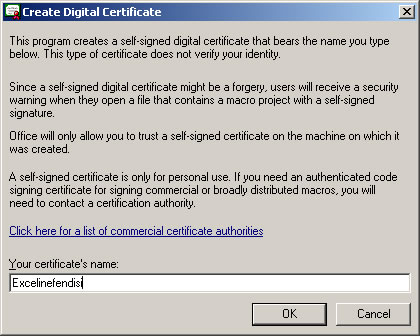
İmzalar, sadece makroyu yazan kişinin kim olduğunu belirtmekle kalmaz, bu kişinin elinden çıktıktan sonra hiçbir şekilde değişime uğramadan geldiğini de gösterir, bir nevi mühür görevi görür diyebiliriz.

Bu konu hakkında internette bol bilgi bulunuyor, bunun detayın burada daha fazla girmeden bir SSC nasıl oluşturulur ona bakalım.

**DİKKAT**: Dijital sertifika, makroyu kimin yazdığını gösteriyor olmakla birlikte, içindeki kodun güvenli olduğunu garanti etmez. Dosyaya güvenmek, tamamen kullanıcıya kalmıştır, bu da onun makroyu yazan kişiye güvenip güvenmemesiyle ilgilidir.

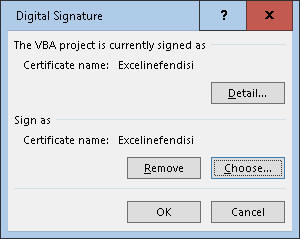
Peki, bu SSC nasıl alınır. SelfCert.Exe dosyası ile. Windows ve Office sürümlerindeki farklılıklar nedeniyle bu dosyanın konumu değiştiği için dosyayı doğrudan Windows Explorer içinde aramanızı öneririm.

Dosyayı çalıştırdığınızda şu görüntü çıkacak, isminizi yazın ve OK diyin.



Bu işlemi sadece bir kez yapmış olacaksınız. Ondan sonra her dosyanız için aşağıdaki işlemi yapmanız gerekecektir.

VBE'ye geçin, sol panelden imza atayacağınız dosyayı seçin, **Tools>Digital Signature** yolunu takip edip Choose düğmesi aracılığıyla dosyanızı imzalayın.



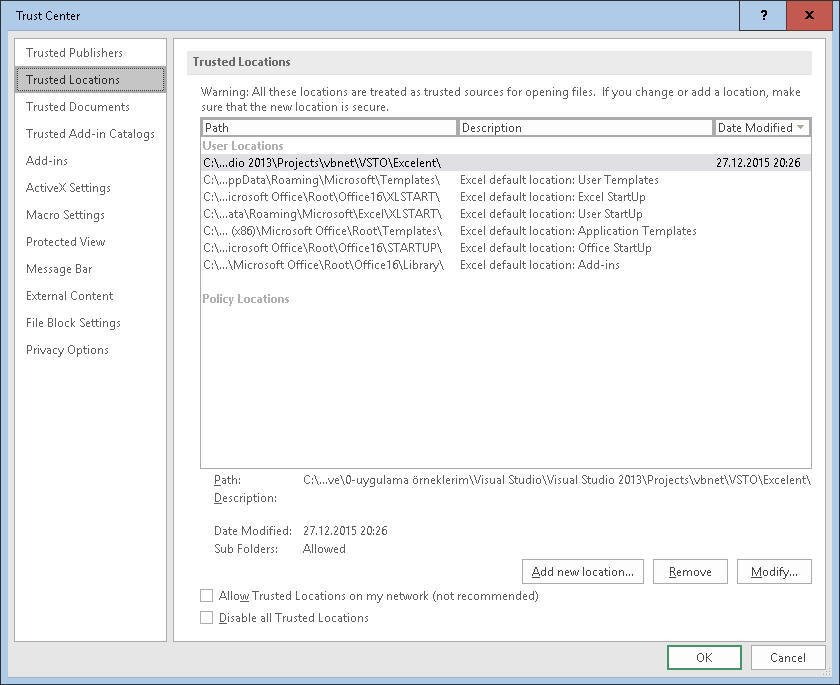
##### Güvenilen Yayıncı listesine birini eklemek

Arkadaşınızın imzaladığı makrolu bir dosyayı açtınız ve güvenlik uyarısı çıktı, şimdi güvendiğiniz kullanıcıya ait imzalı bir makroyu listeye ekleyeceksiniz.

Trust Settings içinde '**Trust all documents from this publisher**' düğmesine tıklayarak bu kullanıcıyı güvenilir yayıncılar listesine eklemiş olursunuz.

##### Güvenilir Lokasyon

Diyelim ki bir makro içeren dosyayı açacaksınız, imzası yok ama biliyorsunuz ki güvenli. İkide bir uyarı çıkmasın istiyorsunuz, ama makro ayarlarını da daha düşük bir seviyeye çekmek istemiyorsunuz. O zaman onu güvenli alan olarak adlandırılan yere alırsınız ki Excel güvenlik merkezi bir daha sizi rahatsız etmesin.



**DİKKAT**: Belgelerim(Documents/My Documents) klasörünü 'güvenilir yer' olarak işaretlemek yerine bu klasör altında başka bir klasör açın, onu işaretleyin. Aksi taktirde tüm Belgelerim klasörünüzü hackerlara açık hale getirmiş olabilirsiniz.

Ek bilgi: Otomatik olarak güvenilir yer olarak gelen bazı klasörler de bulunmaktadır.

### Önemli Uyarı!!!

Makrolar gerçekten çok faydalı araçlardır, ancak tıpkı faydalı ama yanlış kullanıldığında felakete neden olabilen diğer her şeye benzerler. Tüp gazlar faydalıdır ama yanlış kullanım sonucu patlar. Bıçaklar faydalıdır ama yanlış kullanım yaralanmalara neden olur. O yüzden kullanımları dikkat ister.

Makrolarla ilgili de dikkat edilmesi gereken iki konu var.

Makroyu çalıştırdığınızda **Undo yapamazsınız**. O yüzden makroyu çalıştırmadan önce üzerinde çalıştığınız dosyanın son halinin bir yedeğini almanızda fayda var. İsterseniz her makronuzun önüne, MsgBox ile bu soruyu sordurabilirsiniz.

cevap = MsgBox("Dosyanın yedeğini aldın mı?", vbYesNo)

If cevap = 6 Then 'bu yes demek oluyor

GoTo ilerle

Else

MsgBox "O zaman yedeği al sonra tekrar çalıştır. "

Exit Sub

End If

ilerle:

'Diğer Kod bloğu

İkinci dikkat edilecek konu da, makronuzu her yönüyle test etmeden çalıştırırsanız istisnai bir durumda farklı bir sonuçla karşılaşabilirsiniz. Bu yüzden olası her durum için çalıştırmanızda fayda var. Makronuzu özellikle başkaları kullanacaksa, onlara nasıl kullanacaklarını hiç anlatmadan direkt çalıştırmalarını söyleyin, olacak her hata için de gerekli kontrolleri, yönlendirme sorularını ve hata denetim elemanlarını koyun. Mesela, makronuzun doğru çalışması için ele alınacak sütunun A sütunu olması gerekebilir, bu nedenle kullanıcıya "İşlem yapılacak bilgi A sütununda değil mi?" diye bir soru yöneltebilirsiniz. Bunların hepsini yeri geldiğinde öğreneceğiz.

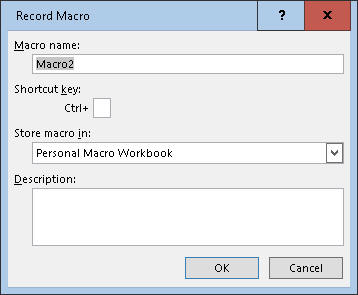
## Makro Kaydetme ve VB Editörü

### Makro Kaydetme

İlk sayfalarda, bir makro yazmaya başlamadan önce bir makro kaydedip oluşan kodu inceleyerek de makro öğrenmeye başlayabileceğinizden bahsetmiştim. Şimdi gelin birlikte, bir makro nasıl kaydedilir ve yazılan kod üzerinden nasıl oynama yapılır ona bakalım.

İlk olarak Developer menüsünde Record Macro diyoruz.

Makrolarımızı genelde **Personal.xlsb** dosyasına kaydedeceğimiz için depolama yeri olarak burayı seçiyoruz.



Şimdi basit bir dizi işlem yapalım.

* A sütununu seçelim
* Tüm sütunu **kalın** ve rengini de kırmızı yapalım
* A1'e "Merhaba" yazalım
* Son olarak, Excel’in versiyonuna göre pencerenin çeşitli yerlerinde bulunabilecek olan Stop tuşuna basalım.( Benimki pencerenin sol alt köşesinde)

https://www.excelinefendisi.com/images/Vbarecorderstop.jpg

Şimdi **Alt+F11** tuşlarına basarak veya Developer menüsünden(veya QuickAccessToolbardan) VB editörünü açalım.

Burda son yazdığımız makro hep en büyük numaralı ModuleX içine gider. Ör: Module1, Module2,,,,Module5 varsa biz Module 5 içine gidip bakalım ve kodumuzu görelim.

Sub Macro2()

'

' Macro2 Macro

'

'

Columns("A: A").Select

Selection.Font.Bold = True

With Selection.Font

.Color = -16776961

.TintAndShade = 0

End With

Range("A1").Select

Selection.FormulaR1C1 = "Merhaba"

End Sub

Gördüğünüz gibi makromuz **'Sub'** ifadesi ile başladı, sonra makromuzun adı, sonra da () işaretleri geliyor. Arada kod parçaları var, en son da **'End Sub'** ifadesi gelir.

' işareti açıklama cümleleri içindir, kendinize makronun o bölgesiyle ilgili hatırlatmalarda bulunmak isteyebilirsiniz. Bu açıklamalar oldukça faydalı olabilmektedir, özellikle bir makroyu seyrek kullanıyorsanız, neyi niçin yaptığınızı hatırlamak adına faydalı olmaktadır. Burada da, Excel'in kendisi otomatik bir açıklama eklemiş olduğunu görüyorsunuz. Yorumlar yeşil renkli görünürler. Kaydedilen makrolardaki bu açıklamaları silebilirsiniz.

İsterseniz bundan sonra kendiniz çeşitli satırlara müdahale edebilir, bazılarını çıkarabilir, yeni satırlar ekleyebilirsiniz. Mesela bu makro her çalıştığında A sütununu seçecektir, bunun yerine hep o anda seçili olan sütunla ilgili işlem yapan bir makro yazmak isteyebilirsiniz, ki bu daha işlevseldir. Bunun için, yukardaki gibi herhangi bir sütunu seçen bir makro kaydedin, sonra kod kısmına giderek, Columns("A: A") yazan satırı çıkarın, böylece direkt seçili olan şeyle (burada sütun seçilidir) işlem yapılır. Sütun seçili değilse, ama bulunduğunuz hücrenin bağlı olduğu sütunun seçilmesini isterseniz sütunlu ifadeyi şu şekilde değiştirmeniz yeterli olacaktır.

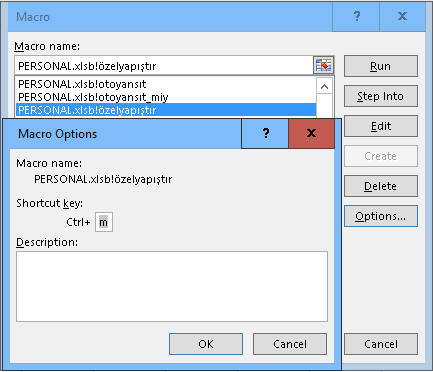
Columns("A: A").Select=>ActiveCell.EntireColumn.Select

Bu konularda detaylı bilgi ilerleyen sayfalarda verileceği için şuanda daha fazla ayrıntıya girilmeyecektir.

Oluşturduğunuz makrolara kısayol tuşu da atayabilir, bunları sonradan amacınıza uygun olarak değiştirebilir veya iptal edebilirsiniz. Şimdi bir de hızlıca kısayol işlemleri nasıl yapılır ona bakalım.

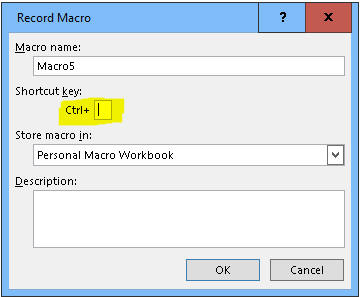
#### Kısayol işlemleri

Bir makroya kısayol tuşunu makro kaydederken atamak oldukça kolaydır: Aşağıdaki kutuya istediğiniz harfi koyabilirsiniz.(Harf dışında bir karakteri kabul etmez.)



Peki makro kaydediciyle oluşturmadığımız bir koda nasıl kısayol tuşu atarız? Basit: Developer menüsünden Macros butonuna basalım, istediğimiz makroyu seçelim ve Edit diyelim. Eğer Personal.xlsb gibi gizli bir dosyadaki makroyu editleyeceksek Excel buna izin vermez. Dosyayı önce Unhide etmeli, arkasından makro ayarını yapmalıyız, o da bittikten sonra tekrar gizlemeliyiz.

Aşağıda çıkan kutuda aynı yere istediğimiz harfi yazalım. Bu arada fark ettiyseniz bu ekranda sadece Ctrl tuşuyla çalıştırılan kısayollar oluşturabiliyoruz. Farklı tuş kombinasyonlarına kısayol tuşu atamayı öğrenmek için biraz daha(nesnelerin efendisi olan Application nesnesini tanıttığımız sayfaya gelmeyi) beklemelisiniz.

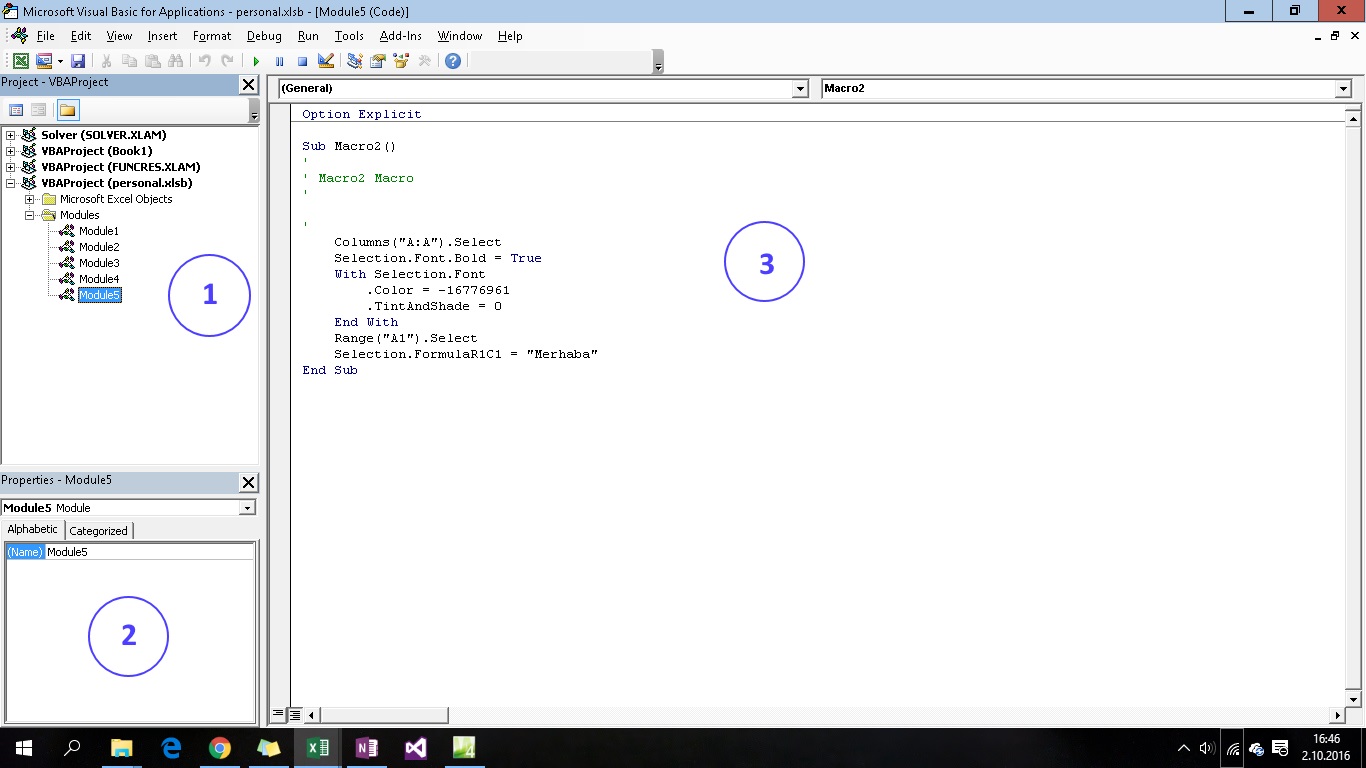


Son olarak, kısayol atadığımız bir makrodan bu kısayol tuşunu nasıl kadırırız? Basit: Yine Macros düğmesine basarız, yukardaki çıkan diyalog kutusunda çıkan harfi sileriz.

### VBE(Visual Basic Editörü)

#### Üç pencere

Henüz başlangıç aşamasındayken bir VB penceresi genel olarak aşağıdaki gibi görünür. 1 numaralı bölmede makrolara ismen ulaşacağınız project penceresi bulunur, onun altında 2 numaralı bölmede Workbook/Worksheet veya modüllerin özelliklerini gösteren properties penceresi bulunur, burayı genel olarak modüllerin adını değiştirmek için kullanacağız, onun dışında kapalı tutabilirsiniz. Bizi ilgilendirecek kısım daha çok 3 nolu geniş alan olacaktır. Kodlarımızı burada yazacağız, bütün düzeltme ve çalıştırma işlemlerini de bu alan üzerinden yapacağız.



Projects penceresiyle ilgili olarak söyleyebileceklerim şimdilik şu kadar olacaktır. 'Bir makro nereye yazılır?'ın ilk cevabını burada belirledikten sonra 3 nolu panele geçip yazmaya başlıyoruz. O yüzden kodumuzun ilk etapta neyin içine yazılacak sorusunun cevabıdır. Seçenekler şöyle:

* Excel Objects içinde bir sayfa
* Excel Objects içinde ThisWorkbook
* (Genel) Modül
* Class Modül
* UserForm

Microsoft Excel Objects altında Sheets ve ThisWorkbook nesneleri bulunur. Sheets'lerden birinde yazılan makrolar genellikle Worksheet\_SelectionChange and Worksheet\_Calculate gibi sayfa olaylarını yönetmek için kullanılırken, ThisWorkbook içine yazılan makrolar dosya seviyesindeki olayları ele almak için kullanılır, Open/Close gibi. Bu nesneler hakkında daha detaylı bilgi ilerleyen bölümlerde verilecektir.

(Genel) Modüller, herhangi bir sayfa üzerinde çalışabilecek genel prosedürleri depolamak için kullanılır. Vereceğimiz örneklerin çoğu bu Modüller içine yazılacak makrolar şeklinde olacaktır.

UserForm ve Class modüllere de ilerleyen bölümlerde değinilecektir.

**Önemli Not:** İzleme Pencereleri bölümünde detaylı göreceğiz ancak, başlangıçta öğrenmeniz gereken bir pencere daha var. Ctrl+G tuşlarına basarsanız en aşağıda **Immediate Window** çıkacaktır. Kodlarımız içinde sık sık **Debug**.**Print** ifadesini kullanacağız. Bu, kendisinden sonra gelen ifadenin değerini Immediate Window'a yazan bir metoddur. O yüzden bu pencereniz de sürekli açık kalsın derim.

#### Menüler

Excel'in kendi içindeyken Ribbon’daki menülere sık sık başvururuz ancak açık söylemek gerekirse VBE penceresindeki menülerle çok fazla işiniz olmayacak. Digital İmza ayarlamasını yapmak ve projemize referans(dll) eklemek için Tools menüsünü arada bir kullanacağız.

Bir diğer kullanacağımız menü de View menüsü olacak. Burada Immediate, Local gibi yardımcı pencereleri açacağız. Bunlara da sonrada değinilecektir.

Bir de, özel Add-inler vardır ki, eğer bunlardan birini satın aldıysanız veya ileri seviyelere geldiğinizde kendiniz yaptıysanız(bunun için VBA’in dışında şeyler bilmek gerekir) buna ait menüleri de kullanabilirsiniz. Eğer maddi durumunuz uygunsa [**http://codeVBA.com**](http://codevba.com) sayfasındaki gibi bir Add-in kurmanızı tavsiye ederim. Mesela bu adresteki Add-in hem muhteşem bir pratiklik sağlıyor hem de hatasız kod yazılmasını sağlıyor. Bence bu linki bir yere not edin ve kendinizi biraz geliştirdikten sonra ilk etapta deneme sürümünü bir indirin, kendiniz görün neler yapılabildiğini.

## Excel Nesne Modeli

### Giriş

Visual Basic(VB) gibi nesne yönelimli(İngilizce tabiriyle object oriented(OO)) dillerde **nesneler** ve bu nesnelerin özellikleri(properties), eylemleri(methods) ve olayları(event) bulunur. Gerçi VB tam anlamıyla bir OO dil değildir; bu konsepti destekler ama tam bir OO dil olabilmesi için tüm gereken kriterleri(Encapsulation, Abstraction, Inheritance ve Polymorphism) karşılamaz. Dolayısıyla VB’den türeyen VBA de tam bir OO dil değildir. Ancak biz burada OO konseptinin detaylarıyla ilgilenmekten ziyade genel anlamda nesne kavramı ve nesnelerin üyeleri üzerinde duracağız.

Gündelik hayattan örnek verecek olursak pencere bir nesnedir. Pencerenin kulpu bir alt nesnedir, kulpun yapıldığı madde ve rengi ise bir özelliktir, açılmak ise onun bir eylemidir.(İngilizcesi method olduğu ve programcılıkta metod kelimesi daha çok kullanıldığı için bundan sonra eylem yerine metod terimini kullanacağım.)

Şimdi bu gündelik hayattaki pencere kulpuyla ilgili bir örnek yapalım.

“Eğer Pencerenin Kulpu alüminyumsa pencere kapansın, ahşapsa açılsın, başka bir maddeyse ne yapılacağına kullanıcı karar versin”.

Sub Pencere()

If pencere.kulp.malzeme=alüminyum then

Pencere.kapat

ElseIf pencere.kulp.malzeme=ahşap

pencere.açıl

Else

Msgbox("kararı kullanıcı versin")

End If

End Sub

Pencerenin açılma olayına da bir kod ekleyebilirsiniz, şöyle ki:

“Eğer pencere açılırsa radyatörün peteklerini kapat”, ki bunun da kodlaması şuna benzer birşey olacaktır.

Sub pencere\_afteropen()

radyator.statu=off

End Sub

Gördüğünüz gibi nesnelerin üyeleri(özellikleri ve metodları) nokta(.) ile nesnelerinden ayrılmaktadır.

Bu arada kod bloğumuz içine bir döngü de yerleştirebiliriz, mesela evdeki bütün pencereler için bu yukardaki sorgulamaları yapmak istediğimizi düşünün, her pencere için tek tek kod yazmak çok zahmetli olacaktı, bunun yerine çeşitli döngü yapılarını kullanabiliriz. Burada VBA kodlarından ziyade sadece genel mantığı vereceğim, ilerleyen sayfalarda zaten kodlamasının nasıl yapıldığını göreceksiniz.

* Birinci pencereden başla
* pencerenin kulpu alüminyumsa açılmasına izin verme,
* Ahşapsa izin ver
* Diğer pencereye geç
* Eğer son penceredeysen programdan çık
* Başa dön

### Detaylar

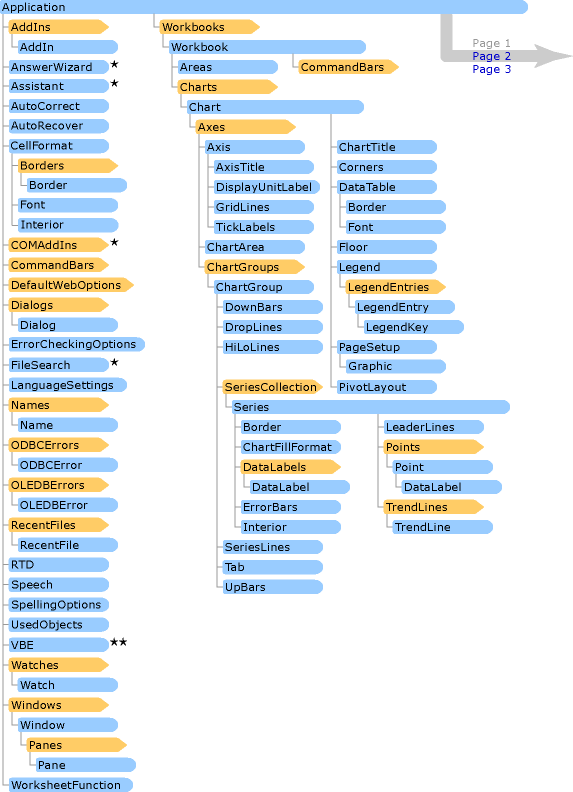
#### Üyelerin(Property ve Metodlar) dönen değeri

Excel’de hemen herşey bir nesnedir ve bunlar bir hiyerarşi içinde bulunurlar. Hiyerarşinin en tepesinde **Application** nesnesi vardır, yani Excel’in kendisi. Onun altında workbook ve başka nesneler vardır. En sık kullanılacak nesneler Dört Temel Nesne bölümünde detaylıca ele alınacak olup bunların hepsi hiyerarşinin bir seviyesini gösterir.

Alt nesneden kastımız aslında, bir propertydir, yani terminolojik olarak **nesnenin nesnesi** diye bir kavram yok, ancak nesnenin propertysinin **dönüş değeri** bir nesne tipinde(object type) olduğu için bundan nesnenin alt nesnesi gibi bahsederiz. Ör: Worksheet nesnesinin Range propertysinin dönüş değeri Range nesnesi döndürür ve biz de bunu sanki nesnenin alt nesnesi gibi yorumlarız.

#### Nesne Modeli Grafik Gösterimi

Excel Nesne Modelinin grafiksel gösterimi aşağı yukarı şöyledir. Eski Excel versiyonlarında buna program içinden ulaşabiliyorduk ancak şuan yok.



Excel’in Nesne modeli hakkında daha detaylı bilgi edinmek istiyorsanız [**MSDN'yi(https://msdn.microsoft.com/en-us/library/wss56bz7.aspx)**](file:///C:\Users\volka\AppData\Roaming\Microsoft\Word\MSDN'yi(https:\msdn.microsoft.com\en-us\library\wss56bz7.aspx)) ziyaret etmenizi tavsiye ederim.

#### Collection kavramı

O anda açık olan tüm nesne(obje) grubuna collection denir.  Bir nesnenin çoğul hali olarak ifade edilir. Ör: Workbook nesnesi, Workbooks collectionının bir üyesidir. Mesela o anda sadece birinci Workbooku kapayacaksanız Workbooks(1).Close derken, tüm Workbookları kapatmak için Workbooks.Close dersiniz.

Collectionları döngüler içinde çok kullanacağız. Mesela aktif dosyanın tüm sayfalarında işlem yapmak için aşağıdaki gibi bir kod yazacağız.

Sub collectionlar()

'Tanımlamalar

For each ws in ActiveWorkbook.Sheets

'kodlar buraya

Next ws

End Sub

Collection konusunu burada bitirelim, döngülerde ve nesnelerde karşımıza tekrar çıkacak, orada detaylarına değineceğiz.

**NOT**: VBA da bize Collection sınıfını sunar, böylece biz de kendi collectionlarımızı yaratabiliyoruz. Bu konuya da yine ilerleyen sayfalarda değineceğiz.

#### Class kavramı

Felsefeyle ilgilendiyseniz platonun idealar dünyasını duymuşsunuzdur. Ona göre dünyada gördüğümüz herşey idealar dünyasındaki bir ideanın dünyada somutlaşmış halidir. Tıpkı bunun gibi VBA’deki her nesne de bir classın somutlaşmış halidir. VBA ile gelen bir dolu class olmakla birlikte İleri Seviyeler bölümünde göreceğiniz üzere kendi class ve dolayısıyla nesnelerinizi de yaratabilirsiniz. Biz mevcut classlar üzerinden bir örnek verip konuyu burada bitirelim, çünkü gerçekten bu kadar detaya boğulmanıza şu aşamada hiç gerek yok, sadece kavramları genel olarak bilin diye bu konuya değiniyorum.

Mesela Workbooks koleksiyonunun bir üyesi olan Workbook nesnesi aslında bir Workbook classının Workbook tipinde bir nesnesidir. Arka planda bu class için tanımlanmış özellik ve metodları vardır. Nasıl idealar dünyasındaki bir atın kulakları, uzun kuyruğu, 4 bacağı gibi özellikleri ve kişnemesi, koşması v.s gibi eylemleri(metodları) varsa workbook classı için tanımlanmış name, Path gibi özellikler ve Open, Close, Add gibi metodları vardır ve bunlar bütün workbook nesneleri için geçerlidir.

#### Library(Kütüphane) kavramı

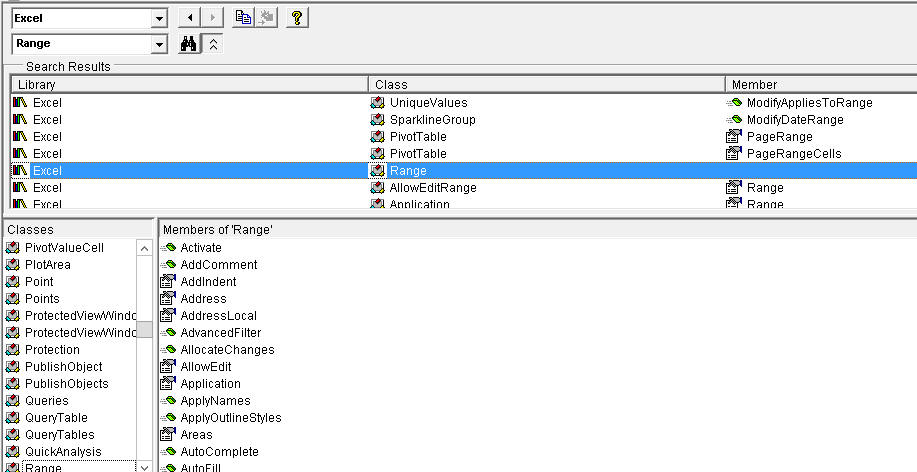
Bir veya daha çok classtan oluşan kümelere Library denir. (Teknik not: Bunlar aslında bir dll dosyasından başka birşey değildir). Bunların bir kısmı default olarak VBA projelerimize dahildir, bir kısmını ise ihtiyaca göre biz ekleriz, bir kısmını ise hiç kullanmıyor olacağız. Default olarak gelen ve en sık kullanacağımız kütüphaneler şunlardır.

* Excel
* VBA
* Office

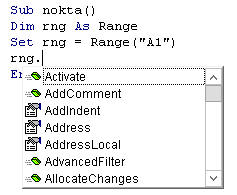
Bunun dışında Access ve Outlookla birlikte çalışmak için bunlara ait kütüphaneleri de VBE içindeki **Tools>References** menüsünden ekleriz. Bir diğer önemli kütüphane de **Scripting.Runtime**'dır. Her ikisini de yeri geldiğinde detaylıca göreceğiz.

#### Object Browser ve nokta notasyonu

Tüm Excel Nesne Modeline ve fazlasına ulaşabileceğiniz yer VBE içinden ulaşabileceğiniz **Object Browser**'dır. Burda sol üstte önce bir library seçip sonra bu library içindeki classları ve classların hemen yanında da yani sağ altta da bu classlara ait üyelere(metod, özellik ve olaylara) ulaşabilir, en alt blokta da bunlar hakkında kısa bir bilgi alabilirsiniz.

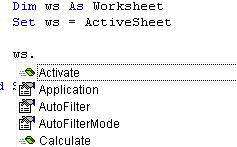


Nesnelerin üyeleri hakkında bilgiye ulaşmanın bir yolu da **intellisense** teknolojisidir. Object tipli bir dönüş değeri olmayan tüm nesnelerde nesne adını yazıp nokta koyduktan sonra tüm üyelerin gösterilmesine intellisense teknolojisi denir. Bu şekildeki yazım tekniğine de nokta notasyonu denir. Mesela aşağıda bir Range tipli nesnenin intellisense çıktısı görünmektedir.



Tabiki Object Browser toplu bir araştırma ve nesneler yazmak yerine seçme imkanı sunduğu için daha makbuldür, intellisense ise araştırma yapmaktan ziyade daha çok şu işe yaramaktadır. Eğer nesne adından sonra ortaya çıkmıyorsa nesne ismini hatalı yazmışız demektir(Object tipli değil spesifik bir dönüş tipli bir nesne olduğunu varsayıyorum). Ayrıca üye ismini uzun uzun yazmak yerine bir iki harfi yazdıktan sonra Tab tuşuna basarak üye adı otomatik tamamlanmakta ve bu da bize hız kazandırmaktadır.

Yukarda bahsettiğimiz nesnelerin **dönüş değeri** konusunu **Intellisense** ile bağdaştırmamızda fayda var. Şöyle ki; "Nesne", bildiğiniz gibi Objenin Türkçesidir ama bir ayrım var. Nesne derken nesnenin kendisinden bahsediyoruz, Object derken dönüş tipinden. Bu bağlamda Activesheet de nesnedir ActiveCell de. Ancak ilkinin dönüş tipi Object iken ikincisininki Range'tir. Aktif sayfanın dönüş tipinin WorkSheet olmasını istiyorsak bunu bir değişkene atamalı ve bu değişkeni WorkSheet olarak tanımlamalıyız. Bu arada Activesheet'in dönüş değeri neden Object? diye düşünebilirsiniz. Bunun sebebi, bu nesnenin birden fazla şekle sahip olabilmesidir: Worksheet, Chart gibi. **İşte Activesheet'te olduğu gibi, birden fazla anlama gelebilecek nesnelerin dönüş değeri hep Object olmaktadır.**



#### Global sınıfı

Object Browserda Classes bölümünde ilk başta duran mavi renkli bir <**globals**> classı vardır. Bu class içinde bulunan üyeler global tanımlanmışlardır ve **bağlı oldukları nesnenin kullanımına ihtiyaç duymazlar**. Örneğin, **Math** classında bulunan ve mutlak değer almaya yarayan **Abs** metodunu kullanmak için bir Math nesnesine ihtiyaç duymayız, bu metodu doğrudan kullanırız.

İşte bu globals içinde bulunan tüm üyeler, farklı farklı classların global üyelerini gösterirler. (Bunlar C# gibi dillerde static tanımlanan üyelere benzemektedirler)

#### With ... End With yapısı

Makrolarınızı kaydederken sıklıkla göreceğiniz bir yapı olacak. With.. End With yapısı. Bir nesnenin üyelerine arka arkaya sıklıkla başvurmanız gerektiği durumlarda bu yapıyı kullanırız. Zorunlu değil tabiki ancak, hem daha az kod yazmamızı sağlar hem de okunurluğu iyileştirir.

Bu yapıda, **With... End With** arasında bulunan ilgili nesnenin üyeleri, önlerinde nesnenin adı yazılmadan sadece . işaretini takip edecek şekilde yazılırlar.

Şimdi yine yukarıdaki pencere örneği üzerinden giderek bir örnek yapalım.

With Pencere

.kulp.malzeme="alüminyum"

.kulp.kalınlık=10

.kulp.çevir

.aç

evi\_havalandır 'burada başka bir fonksiyon çağırıyoruz, o yüzden başında . yok

If ev.hava="iyi" then

.kapat

End If

End With

## Yerleşim ve Erişim

Yazdığım kodlar nerede gözükecek, bunları nasıl ve nerede çalıştıracağız gibi sorularınızın olması muhtemeldir. Bu konuyu, bu sorularınıza cevap için hazırladım.

### Kodların yazılacağı yerler

İlk örneklerde, kodlarımızı hep Personal.xlsb içindeki Modüller içine yazdık. Peki başka yere kod yazamaz mıyız? Tabiki yazarız.

* Mesela, Dosyaların kendisine ait bir kod bölümü var, VBE editöründe ThisWorkbook içine gider. Bu konu Workbook Olayları konusunda ele alınacaktır
* Sayfaların da kendine ait kodları olabilir. Bu konu Worksheet Olayları konusunda ele alınacaktır
* Bir butona tıklandığında bir kod çalışmasın sağlayabiliriz ancak bu da Modül seviyesinde ele alınır.
* Bir UserForm oluşturulabilir(bu konu ayrıca Userformlar bölümünde ele alınacak)

### Kodların çalıştırma yöntemleri

Yazdığınız kodların çalışmasını sağlamanın da birkaç yolu var. Yukardaki maddelerle bağlantılı olarak;

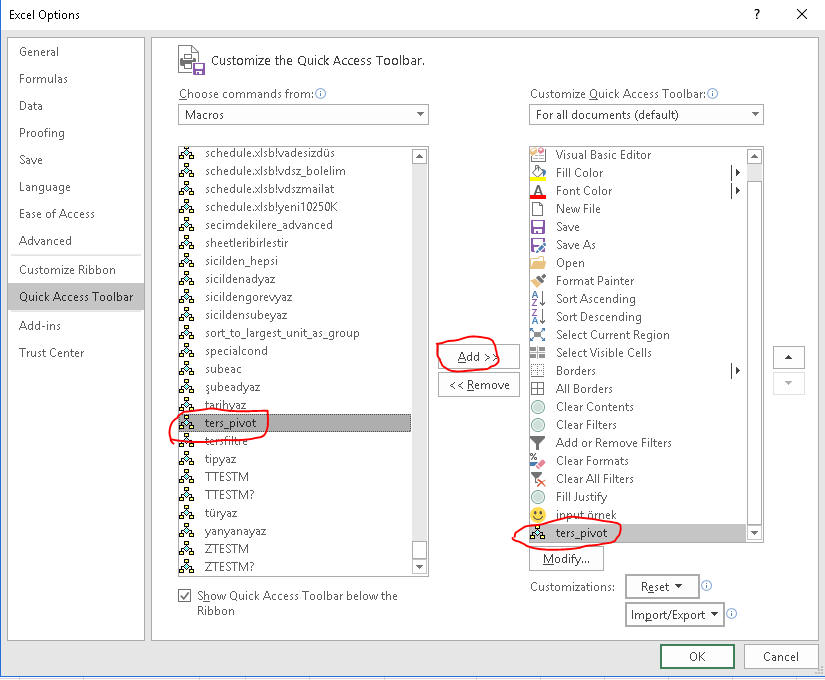
* VBE açıkken ve bir prosedürün içindeyken F5 ile
* Workbook veya Worksheet ile ilgili bir olay gerçekleştiğinde kendiliğinden devreye girecek event bazlı kodlar
* Ribbona veya QuickAccesToolbara(QAT) atadığınız butonlara tıkladığınızda çalışacak kodlar
* Add-in olarak hazırladığınız kodlar
* Sayfa üzerinde bir butona bastığınızda çalışacak kodlar
* "Macros" dialog kutusu (Alt+F8)
* Kısayol(Shortcut) atadığınız kodlar

Bunların hepsini yeri geldikçe göreceğiz. Şimdilik sadece Ribbon'a ve QAT'ye düğme nasıl eklenir ona bakacağız.

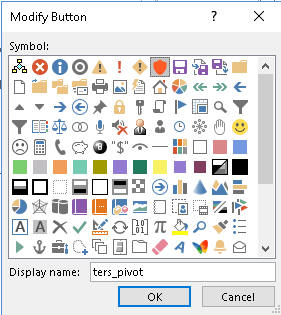
### Ribbon veya QAT'ye makro düğmesi atama

İşlemler her ikisi için de aynı olacağı için ben sadece QAT üzerinden anlatacağım.

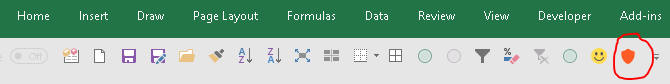
QAT'a sağ tıklayarak özelleştir diyelim. Sonra menüden "Macros"u seçip, aşağıdan da istediğimiz makroyu Add düğmesine tıklayarak QAT'de istediğimiz yere alalım.



Düğmemiz eklendikten sonra Modify tuşuna basarak ikonu ve makronun görünen ismini istediğimiz gibi değiştirebiliriz.



OK dedikten sonra düğmemizin QAT'ye eklendiğini görürüz. Personal.xlsb üzerindeki bir makroyu eklemişsek, ki genelde öyle yaparız, bu makro tüm dosyalarda çalışır halde olacaktır.



## Neler Öğrendik

Bu bölümde genel olarak makroların ne işe yaradığını öğrendik. Makroları yazacağımız editörü tanıdık. İlk adımları atarken bize yardımcı olacak Makro Kaydediciden bahsettik. İlk makroları içinde barındıracağımız Personal.xlsb dosyasını oluşturmayı, güvenlik ayarlarını yapmayı öğrendik.

Excel nesne modelini tanıdık. Bu konu oldukça önemli, üzerinde çok fazla okuma yapmanızı tavsiye ederim.

Son olarak da kodlarımızı nasıl ve nerelerden çalıştırabileceğimizi gördük.

Şimdi bir sonraki bölüme geçebiliriz.