Hedefler

Bu üniteyi çalıştıktan sonra;

- Visual Basic.NET'te Hata Ayıklamayı öğrenmek,

İçindekiler

- Hatalar,
- Yazım Hataları,
- Çalışma Zamanı Hataları,
- Mantık Hataları.

Visual Basic.NET' te Hata Ayıklama

Hatalar

Program yazmak, belli kurallara bağlı olduğu için programın yazımı sırasında bazı hatalar yapılabilir. Karşılaşılabilecek hataları şöyle sınıflandırabiliriz :

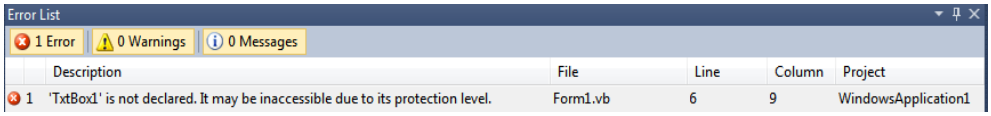
- Yazım (Syntax) hataları
- Çalışma zamanı hataları (Runtime Errors)
- Mantık hataları

Yazım hataları, programı yazarken yapılan hatalardır. Örneğin ***For*** deyimi olmadan kullanılan ***Next*** deyimi, bir tırnak ya da doğru konmamış bir virgül gibi. Program yazımında, dilin bileşenlerinin uygun olarak kullanılmaması, diğer bir deyişle yazım hataları da derleme sırasında ortaya çıkar. Aslında bu hatalar program yazarken kullanıcıya anında bildirilir. Örneğin kod yazımı sırasında ***TextBox*** yazacağımız yerde, ortadaki ***"e"*** harfini unuttuğunuzda, bu kelimenin altı dalgalı çizgi ile çizilecektir. Bunun anlamı burada bir yazım hatası olduğunu belirtmektir.

```
TxtBox1.Text = "Sakarya"
```

Eğer kullanıcı bu hatayı fark etmeden programı çalıştırmaya kalkarsa, ekranda aşağıdaki hata mesajı çıkacaktır.

Burada ***No*** butonuna tıklayarak, programın çalışmasını durdururuz. Aşağıdaki ***Error List*** bölgesinde hatalar nedenleri ve satır numaraları ile açıklanır.



Çalışma zamanı hataları ise program çalıştığı sırada ortaya çıkar. Örneğin olmayan bir dosyayı açmaya kalmak, sıfıra bölmek yada sabit diskte yer kalmamış gibi. Bu hatalar programın geliştirilmesi sırasında anlaşılmazlar. Ancak programın test edilmesi aşamasında yada programın kullanıcılar tarafından kullanılması aşamasında beklenmedik bir olaydan dolayı oluşurlar.

Çalışma zamanı hatalarını önlemek için Visual Basic 6.0' da da kullanılan komutları kullanarak programda hatalara karşı önlem alabiliriz.

On Error GoTo : Bu komut yazıldıktan sonra, herhangi bir hata oluştuğunda program belirtilen etikete gider ve bu etiketten işlemeye devam eder.

On Error Resume Next : Bu komutun kullanıldığı satırdan itibaren tüm hatalar göz ardı edilir. Hiçbir şekilde kullanıcıya hata bildirimi olmaz ve programın çalışması durmaz.

On Error GoTo 0 : Hatalar göz ardı edilmişse, bu komutla yeniden hata yakalamaya başlanır.

Err : Herhangi bir hata oluşması durumunda, oluşan hatanın numarası ve açıklanması ***Err*** nesnesinin alt özelliklerine aktarılır. Bu nesnedeki hata bilgileri kontrol edilerek hata öğrenilir.

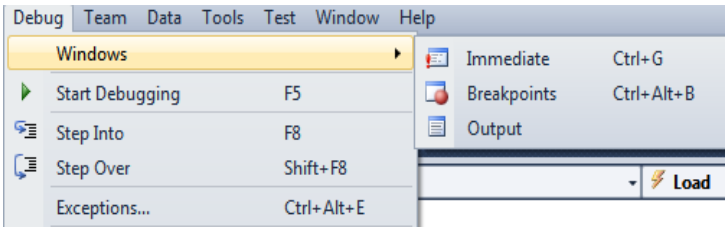
Örnek :

```
If Err.Number = 53 Then
        MessageBox.Show("Dosya Bulunamadı!!!")
End If
```

Sık karşılaşılan hata numaraları ve açıklamaları aşağıdaki tabloda verilmiştir.

No	Hata	Açıklama
5	Invalid Procedure Call	Geçersiz prosedür çağırımı
6	Overflow	Limit aşımı
7	Out of memory	Bellek yetersiz
9	Subscript of range	Dizi belirtilen aralıklar dışında
10	This Array Fixed or Temporarily locked	Bu dizi sabitlendi yada geçici olarak kilitlendi
11	Division by zero	Sıfıra bölüm hatası
13	Type Mismatch	Tip uyuşmazlığı
16	Out of String Space	Metin uzunluğu sınırlar dışında
17	Can't perform request operation	İstenen işlem gerçekleşemez
18	User interrupt occured	Kullanıcı tarafında işlem iptal edildi
20	Resume without error	Hatasız sona erme
28	Out of stack space	Stack boyutu aşıldı
35	Sub, function, or property not defined	Sub, fonksiyon veya özellik tanımsız
47	Too many DLL application clients	Çok fazla DLL istemcisi
48	Error in loading DLL	Yüklenen DLL' de hata
49	Bad DLL calling convention	DLL çağırmada yanlış kullanım
51	Internal error	İçsel hata
52	Bad file name or number	Yanlış dosya ismi veya numarası
53	File not found	Dosya bulunamadı
54	Bad file mode	Yanlış dosya modu
55	File already open	Dosya zaten açık
57	Device I/O error	Sürücü girişi
58	File already open	Dosya zaten açık
59	Bad record length	Yanlış kayıt uzunluğu
61	Disk full	Disk dolu
62	Input past end of file	Dosya bittiği halde okumaya çalışılıyor
63	Bad record number	Yanlış kayıt numarası
67	Too many files	Çok fazla dosya
68	Device unavailable	Sürücü kullanılamıyor
70	Permission denied	Giriş yetkisi yok
71	Disk not ready	Disk hazır değil
74	Can't rename with different drive	Farklı sürücü ile yeniden isimlendirilemez
75	Bad/File Access error	Dosya erişiminde hata
76	Path not found	Yol bulunamadı
91	Object variable or with block variable not set	Nesne değişkeni yada bloklu değişken tanımlanmamış
92	For loop not initialized	For döngüsü doğru şekilde başlatılamadı
93	Invalid pattern string	Geçersiz metin
94	Invalid use of null	Geçersiz Null kullanım

Mantık hataları, program düzgün çalışırken sonuçların yanlış elde edilmesiyle meydana gelir.



Program hatasız çalışıyor gibi görünür ancak beklenen sonucu vermiyordur. Bu hataları düzeltmek, diğer hataları düzeltmeye göre çok daha zordur. Bu durumlar için programcılıkta kullanılan tabir ***BUG'*** dır ve bu sorunu gidermeye de ***Debugging*** (hata ayıklama) denir. Visual Basic.NET ortamında mantık hatalarının bulunabilmesi için ***Debug*** menüsünde bazı yardımcı araçlar mevcuttur.

Step Into (F8) : ***Step Into*** düğmesi ile program kesildikten sonra, programın satırlarını adım adım çalıştırmayı sağlar. ***Step Into*** programı kaldığı yerden devam ettirir.

Spep Over (Shift + F8) : ***Step Over*** düğmesi procedure' ı bir birim olarak işletir. ***Step Over*** düğmesine tıklatıldığında bulunan procedure içindeki bir sonraki deyim işletilir.

Immediate (Ctrl + G) : Program kesildikten sonra, procedure içinde bir değişkenin yada bir ifadenin seçilerek değerinin ne olduğuna bakılmak istenirse o zaman anlık izleme penceresi kullanılır.

Uygulamayı ***debug*** etmek, uygulamamızı adım adım ilerletmek olarak açıklanabilir. Adım adım ilerletme işini yapabilmek için, programı ***breakpoint*** adı verilen bir noktada ilerletebilmek için beklemesini sağlamamız gerekir. ***Breakpoint'*** i kod sayfasında, sol tarafta bulunan gri alana tıklayarak koyabiliriz.

Uygulamamıza, ***breakpoint*** koyduktan sonra, ilerleme işini iki şekilde yapabiliriz.

Klavyeden ***F10*** tuşuna basarak ilerleyerek, sadece o kod bloğundaki adımları takip edebilirsiniz. ***F10*** ile ilerleyerek dönen değeri görürsünüz, ancak işlemlerin hangi sıra ile gerçekleştiğini göremezsiniz.

F11 tuşunu kullanarak ilerlediğiniz takdirde, kod bloğu içerisinde bir metot çağrılıyorsa, o metodun içerisine girer ve metot içerisindeki işlemleri de görebilirsiniz.

Ayrıca uygulamanızı satır satır çalıştırmak için ***F8*** tuşunu da kullanabilirsiniz.

Try … Catch … Finally

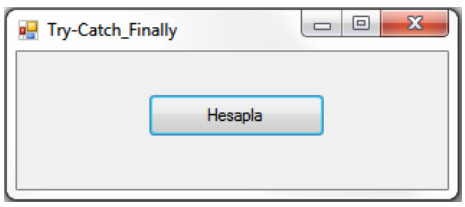
Hataları engellemek için, hata çıkacak satırın başına ***"on error resume next", "on error goto 0", "on error goto etiket"*** gibi deyimler çok kullanılınca büyük bir karmaşa çıkar ve ***"Err"*** nesnesinde dönen değerler de hatanın neden kaynaklandığını tam olarak anlamamızda yardımcı olmadığı gibi bir de ***err*** kodlarını ezberlememiz gerekir.

VB.NET ile birlikte gelen çalışma zamanı hatalarını önlemede önemli bir özellik olan ***Try-Catch-Finally*** komutunun kullanımı aşağıdaki şekildedir :

```
Try
        Hatanın oluşabileceği kod bloğu
Catch
        Hata meydana geldiğinde uygulanacak kod bloğu
Finally
        Her durumda çalışan kod bloğu
End Try
```

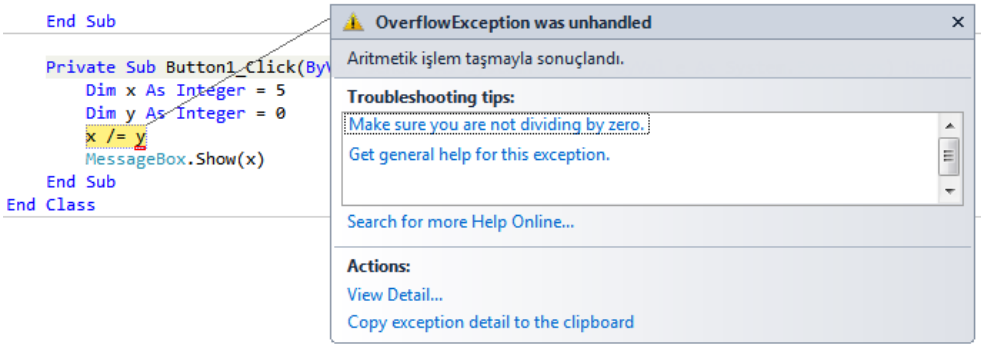
Uygulama çalışırken herhangi bir sorun oluştuğunda programın akışı olduğu yerde kesilir ve ***Catch*** bloğuna düşer. Buradaki kodlar işletildikten sonra ***End Try'*** dan devam edilir. ***Finally*** bloğu hata oluşsa da oluşmasa da çalışacaktır. ***Try*** ve ***End Try***'ı kullanmak zorunlu, ***Catch*** ve ***Finally*** isteğe bağlı olarak kullanılabilir.

Basit bir uygulamayla ***Try-Catch-Finally*** deyimini gösterelim : Yeni bir projede ***Form1*** üzerine ***Hesapla*** Text' li buton ekleyelim ve butonun kodları arasına aşağıdaki kodu yazalım ;



```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
            Dim x As Integer = 5
            Dim y As Integer = 0
            x /= y
            MessageBox.Show(x)
End Sub
```

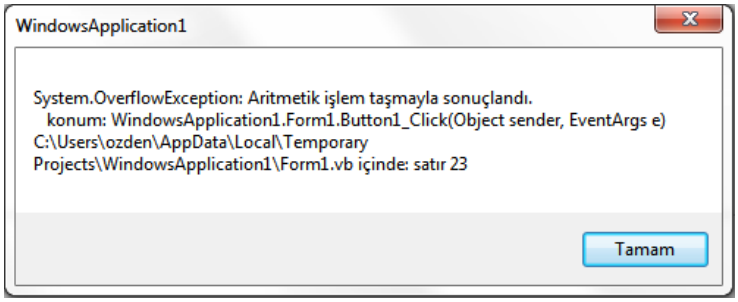
Programı çalıştırıp, ***Hesapla*** butonuna tıklarsak, aşağıdaki hata mesajı (sıfıra bölmeden dolayı) çıkacaktır. ***Break*** tuşu ile programı kesmemiz gerekir.



Şimdi aynı örneği ***Try-Catch-Finally*** ekleyerek yapalım. ***Hesapla*** butonunun kodları arasına aşağıdaki kodları yazalım ;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
            Dim x As Integer = 5
            Dim y As Integer = 0
            Try
                x /= y
            Catch ex As Exception When y = 0
                MsgBox(ex.ToString)
            Finally
                Beep()
             End Try
End Sub
```

Kodu çalıştırdığınızda programın bir önceki örnekte karşılaştığınız hatayı vermediğini göreceksiniz. Verilen hata mesajını (ex.ToString) okuyup, OK butonuna tıklarsak, hiçbir şey olmamış gibi program devam edecektir.



Hata çıkması muhtemel olan kod bloğunu ***Try-End Try*** arasına alıyoruz. Bu aralıkta bulunan kod bloğunda herhangi bir hata durumu oluştuğunda program hata vermeden ve programı sonlandırmadan doğrudan ***Catch'*** e gidiyor ve bu blok arasında yer alan kodu uygulamaya geçiyor.