



## STRING DÖNÜŞÜMLERİ

08.08.2006

Bilgisayar ile insan iletişimi genelde yazı ile oluyor. Bilgisayarın çıktıları genellikle yazı olarak vermesini isteriz. Eğer sayıları göstermesi gerekiyorsa ekranda, göstereceği sayıyı yazıya dönüştürerek yapar.

Meselâ, 45 sayısını 32-bit tamsayı olarak saklar. Bellekteki bitler aşağıdaki gibidir.

```
0000-0000-0000-0000-0000-0000-0010-1101
```

Buna sayının ikili gösterimi denir. Bir sayının ekranda gösterilebilmesi için onun Unicode karaktere dönüşmüş olması gerekir. Unicode karakterin gerçek değeri sayının genliğine eşittir. Meselâ 45 sayısı 4 ve 5 karakterlerinden oluşur. Yani iki tane 16-bit koddan oluşur.

```
0000-0000-0011-0100 ve 0000-0000-0011-0101
```

Bildiğiniz gibi, Console.Write ve Console.WriteLine yöntemleri ekranda gösterebilmek için otomatik olarak tamsayıları string haline dönüştürürler. Bazen string veriyi tamsayıya, tamsayıyı stringe dönüştürmek istersiniz. Bir string ifadeyi tamsayıya dönüştürmek için string ifade sadece rakamlardan oluşmalıdır, eksi işareti de içerebilir.

Şimdi bir tane tamsayı bir tane de string değişken tanımlayarak işe başlayalım.

```
int I = 45;  
string S;
```

Basit bir şekilde bir tamsayıyı string ifadeye dönüştüremezsiniz.

```
S = I; // Bu şekilde olmaz.
```

Eğer yukarıdaki ifadeyi denerseniz aşağıdaki gibi bir hata mesajı alırsınız.

```
Cannot implicitly convert type 'int' to 'string'
```

(int türünde bir sayıyı string türüne kapalı dönüşümle dönüştüremezsiniz.)

Aşağıdaki gibi casting yapmak da mümkün değildir.

```
S = (string)I; // Bu şekilde de olmaz.
```

Eğer yukarıdaki ifadeyi denerseniz aşağıdaki gibi bir hata mesajı alırsınız.

```
Cannot convert type 'int' to 'string'
```

(int türünü string türüne dönüştüremedi.)

C# tip güvenli bir dildir yani farklı türlerin kapalı dönüşüm ya da açık dönüşüm ile dönüştürülmesinde bir takım kurallar ve kısıtlamalar söz konusudur.

Ama buna rağmen bazı durumlarda tamsayı bir türün string bir türe dönüştürülmesi sahne arkasında meydana gelir, meselâ Console.WriteLine yöntemi böyledir.

Bir tamsayıyı string ifadeye dönüştürmenin bir yolu, onu boş bir string ile birleştirmektir.

```
S = "" + I;
```

ya da

```
S = I + "";
```

Her iki durumda da C# tamsayıları string bir ifade ile birleştirebilmek için dönüştürme işlemini yapar. Tamsayı ile birleştirilmek istenen string boş olduğuna göre sonucu etkilemez, neticede tamsayı bir ifade string bir ifadeye dönüşmüş olur. Mesela I değişkeni 45 sayısına eşit olsa dönüştürme işleminden sonra '4' ve '5' karakterlerinden meydana gelen bir yazı olur.

Hala dönüştürme işlemi sahne arkasında yapıldı. Biraz da hile yaptık. Ama açık bir şekilde bir tamsayıyı string ifadeye dönüştürmek için ToString isminde bir yöntem kullanabiliriz. Herhangi bir değişkeni string ifadeye dönüştürür.

```
S = I.ToString(); // Dönüştürme işlemi başarılı.
```

Bu işlem sonucunda S string ifadesi "45" olan iki karakterli bir yazı oldu.

ToString bir yöntemdir. her zaman yöntemleri isminin sonunda parantez aç ve parantez kapa işaretleri ile birlikte kullanırız. Yöntemi tanımlarken de, kullanırken de bu böyledir. Yukarıdaki ifadede geçen ToString yöntemi hiç argümana sahip değildir ama dönen değeri bir string ifadedir.

Console ve WriteLine arasında noktaya ihtiyaç duyduğumuz gibi, tamsayı değişkenimiz ve ToString arasında da noktaya ihtiyaç duyduk. Hatta ToString yöntemini bir tamsayı ifadenin önünde kullanabilirsiniz.

```
S = 279.ToString();
```

Fark ettiğiniz gibi ToString yöntemi çok sık ihtiyaç duyulan bir yöntemdir. Bu yöntemi her nesne ve tür ile kullanırsınız ve nesnelerin yazıya dönüşmüş hallerini elde edersiniz.

Şimdi string bir ifadeyi tamsayıya dönüştürelim. Önce string bir değişken tanımlayalım ve rakamlardan oluşan bir değer atayalım.

```
S = "57";
```

Şimdi bu string ifadeyi tamsayıya dönüştürmek istiyoruz. Şansımıza, string ifademiz de sadece rakamlardan oluşuyor. Burada da söylemek gerekirse atama işlemini yine kullanamayız.

```
int I = S; // İşe yaramaz.
```

Hemen derleme hatasını görürsünüz.

```
Cannot implicitly convert type 'string' to 'int'
```

(string ifadeyi tamsayı bir ifadeye kapalı dönüştürme ile dönüştüremezsiniz.)

Şimdi casting yapalım.

```
int I = (int)S; // Hala işe yaramadı.
```

Aşağıdaki derleme hatası oluştu.

```
Cannot convert type 'string' to 'int'
```

(string türünü int türüne dönüştüremezsiniz.)

Hemen aklımıza bir yöntem olabileceği geliyor. ToString bi bir yöntem olsa da dönüştürme işlemi yapsak.

```
I = S.ToInt(); // Ama böyle bir yöntem yok.
```

Yok ama doğru yoldasınız. int Türü System.Int32 yapısının bir aliasıdır. Bu yapının Parse diye bir dönüşüm işi yapan bir yöntemi var.

```
I = Int32.Parse(S);
```

Int32.Parse yönteminin dönen değeri bir tamsayıdır. Parse yöntemine geçeceğimiz string rakamlardan oluşmalıdır ama eksi işaretine sahip olabilir. Sağında solunda boşluk olabilir. Mesela aşağıda doğru yazılmış bir ifade var.

```
I = Int32.Parse(" -572 ");
```

Eğer string, sayısal olmayan karakterler içeriyorsa ya da eksi işareti ile sayı arasında boşluk varsa Parse yöntemi bir istisna ortaya çıkaracaktır. Sayının int sınırlarının dışında olması da Parse yönteminin istisna ortaya çıkarmasına neden olur.

Şimdi her iki dönüşümü yapan yöntemleri bir karşılaştıralım, benzer ya da farklı yanlarına bakalım.

```
S = I.ToString();  
I = Int32.Parse(S);
```

Her iki yöntem de System.Int32 yapısının elemanıdır. Aralarında temel bir fark var.

ToString yönteminin sol tarafına bakalım. Bir tamsayı değişken var. ToString yöntemi belirli bir tamsayı var ve ona uygulanıyor. Fakat Parse yönteminin sol tarafında bir değişken yok. Int32 yapısının ismi var. Parse yöntemi belirli bir tamsayıya uygulanmıyor. Bir tamsayı oluşturuyor.

Aralarındaki temel fark ise ToString bir instance (örnek) yöntemdir, Parse ise static (durağan) bir yöntemdir.

ToString bir örnek yöntemdir, dolayısıyla belirli bir tamsayıya uygulanır, diğer bir deyişle Int32 yapısının bir örneğine uygulanır. Bir tamsayıya ya bir tamsayı değişkene ya da Int32 yapısının bir örneğini yani tamsayı döndüren bir yöntemle sahip olmanız ki ToString kullanabilirsiniz.

Int32.Parse durağan bir yöntemdir. Yöntemin soluna Int32 yapısının adını yazdıktan sonra kullanırsınız. Int32.Parse yöntemini çağırarak belirli bir tamsayıya sahip olmanıza gerek yok. Yöntemin kendisi bir tamsayı oluşturur.

Başka durağan yöntemler de biliyoruz. Console.Write ve Console.WriteLine yöntemleri, Main yöntemi de durağan yöntemlerdir. MinValue ve MaxValue (field) alanları ise Int32 yapısının durağan alanlarıdır. Bu yöntemlerin önüne yapıların adını yazmak gerekiyor. String sınıfının Length yöntemi bir örnek alandır. Belirli bir string ifadeye uygulayabilirsiniz.

Bildiğiniz gibi int veri türü System.Int32 yapısının bir aliasıdır. Bu durumda System.Int32 yerine int yazabilirsiniz hatta using direktifi ile System isim uzayını deklare etmişseniz sadece Int32 de yazabilirsiniz.

```
I = Int32.Parse(S);
```

Ya da aşağıdaki gibi olabilir.

```
I = int.Parse(S);
```

Her iki kullanım da aynı işi yapar ama gerçek sınıfın ya da yapının adını kullanmayı tercih edebilirsiniz.

Bütün tamsayı türleri ToString ve Parse yöntemine sahiptirler. Meselâ:

```
ushort US = UInt16.Parse(S);
```

ya da

```
ushort US = ushort.Parse(S);
```

Parse yöntemini işaretli tamsayılarla kullanırken eksi işareti kullanmak bir istisna ortaya çıkaracaktır.

Yöntem çağrılarının işlem önceliği vardır. Peki, aynı ifade içerisinde 2 tane yöntemi yan yana çağırırsak ne olacak?

```
string S2 = Int32.Parse(S).ToString();
```

Aynı önceliğe sahip 2 tane yöntem var. Bu tür durumlarda öncelik sırası soldan sağa doğrudur. Öncelikle Int32.Parse yöntemi çalışır. Bu durağan yöntemin çalışması sonucunda bir tamsayı oluşur. O zaman ifade aşağıdaki şekle dönüşür.

```
string S2 = 45.ToString();
```

Artık kolay. ToString yöntemi örnek yöntemdir. 45 sayısını '4' ve '5' şeklinde 2 tane karakterden oluşan S2 string ifadesine dönüştürür.

Şimdi tersini yapalım. Önce bir tamsayıyı string ifadeye dönüştürelim, sonra bu string ifadeyi tekrar tamsayıya dönüştürelim ve tamsayı bir değışkende tutalım.

```
int I2 = Int32.Parse(I.ToString());
```

Bu iç içe yöntem çağırmasıdır. Bir yöntem diğer bir yöntemle argüman olarak yazılmış. Bu durumda görünüme bakılırsa Parse yöntemi önce çalışacak gibi ama değil. Öncelikle argüman gerekiyor. Bunun için de ToString yönteminin çalışması gerekiyor. Çalışınca ifade aşağıdaki şekle dönüşür.

```
int I2 = Int32.Parse("45");
```

Daha sonrası zaten kolay. Parse yöntemi çalışır ve bir tamsayı döndürür.

System isim uzayı Covert diye de bir sınıf barındırır. Çok güzel bir sınıftır. Bütün temel türleri diğerlerine dönüştürebilecek durağan yöntemlere sahiptir. Şimdi aşağıda gördüğümüz örnek bir string ifadeyi tamsayıya dönüştürmek için alternatif yol sunar.

```
I = Convert.ToInt32(S);
```

Covert sınıfının dokümanları incelendiğinde Int32.Parse kullandığı görülür. Onun için olan bütün kurallar ve kısıtlamalar Convert için de geçerlidir.

Aşağıdaki örnek de ToString örnek yöntemine durağan bir alternatiftir.

```
S = Convert.ToString(I);
```

Convert sınıfı aynı zamanda bu yöntemlerin aşırı yüklenmiş şekillerine de sahiptir. Sayı tabanları ile çalışmaya da izin verir. Bu aşırı yüklenmiş yöntemler virgülle ayrılmış 2 tane argümana ihtiyaç duyarlar. Meselâ:

```
I = Convert.ToInt32(S, 16);
```

Bu örnekte S değişkenindeki string ifadenin hexadesimal rakamlar içerdiğini varsayar. Yöntemin ikinci argümanı sadece 16 değil 2, 8, 10, 16 olabilir. Her durumda ilgili string ifade o tabanda geçerli olacak rakamlara sahip olmalıdır. Meselâ eğer ikinci argüman 2 ise string ifade sadece 0 ve 1 rakamlarını içermelidir.

Benzer şekilde aşağıdaki ifade de tamsayıyı hexadesimale çevirir. Aşağıdaki gibi bir ifadede de yine ikinci argüman 2, 8, 10, 16 olabilir.

```
S = Convert.ToString(I, 16);
```

Parse yönteminin de aşırı yüklenmiş sürümleri var. Yöntemlerin aşırı yüklenmesi meselesini de başka bir yazıda ele alacağız.