.NET'te Explicit ve Implicit Dönüşüm Operatörleri

Explicit ve Implicit operatörleri dönüşümlerde kullanılır. C# 12 ve .Net 8 ile gelen bu özellik kendi Mapper özelliklerimizi üretmek için kullanacağımız yenilikçi çözümdür.

Implicit bulunduğu class'ı üretmek için kullanılan operatördür.

Explicit operatörü bulunduğu class'tan başka bir obje üretmek için kullanılır.

Dönüşümleri kontrol etmemizi sağlayan bu operatörler sayesinde, kodlarımızda tip güvenli ve esnek dönüşümler ile tek bir yerden objelerin dönüşümü kontrol edilebilir.

```
public class TurkLirasi
{
    5 references
    public decimal Deger { get; set; }
    public static explicit operator AmerikanDolari
        (TurkLirasi turkLirasi)
    {
        return new AmerikanDolari() { Deger = turkLirasi.Deger / 41.1m };
    }
    public static implicit operator TurkLirasi(decimal para)
    {
        return new TurkLirasi() { Deger = para };
    }
}
7 references
public class AmerikanDolari
{
    5 references
    public decimal Deger { get; set; }
    public static explicit operator TurkLirasi
        (AmerikanDolari amerikanDolari)
    {
        return new TurkLirasi() { Deger = amerikanDolari.Deger * 41.1m };
    }
    public static implicit operator AmerikanDolari(decimal para)
    {
        return new AmerikanDolari() { Deger = para };
    }
}
```

Şimdi elimizde iki adet class olsun. Bunlar para birimi class'ları olsun. Güncel kur değerimiz 41.1 olsun. Yani 1 dolar 41.1 türk lirasına eşit olsun. Buradaki kodlar sayesinde biz diğer sayfadaki gibi kolayca sınıfı üretebileceğiz. Aynı zamanda dönüşümde kolayca sınıfları birbirine dönüştürebileceğiz.

Eğer burada kur değeri online olarak çekilirse. Rahatlıkla o anki değere göre otomatik kur çevrimi olacaktır. bu sayede her yerde kuru dönüştürmek yerine tek yerde bunu kontrol edebilirsiniz. Tabi sadece 2 para birimi class'ı varken kolay çok, Fazla para biriminde farklı çözümler gerekebilir.

```
TurkLirasi turkLirasi = 1000.0m;
AmerikanDolari amerikanDolari = 1000.0m;

/*1*/Console.WriteLine(amerikanDolari.Deger);
/*2*/Console.WriteLine(((TurkLirasi)amerikanDolari).Deger);

/*3*/Console.WriteLine(turkLirasi.Deger);
/*4*/Console.WriteLine(((AmerikanDolari)turkLirasi).Deger);

// Çıktı

/*
    1-) 1000,0
    2-) 41100,00
    3-) 1000,0
    4-) 24,3309002433090024330900243 31
 */
```

Önceki sayfada kurulumu yapılan class'ları buradaki gibi kullanabiliriz artık. Burada her seferinde new TurkLirasi(){Deger = 1000.0m}; fibi bir kullanım yerine doğrudan = 1000 diyerek kullanabiliriz. Bu sayede bizden sonraki veya bizimle geliştiren yazılımcılara kolaylık sağlamış oluruz. Dönüşümlerde ise yine örnekteki gibi kolay bir dönüşüm sağlarız.

Generic yapılarda bazı durumlarda hayat kurtaran bu yapılar, yeni gelen method olduğu için fazla bilinmemektedir. Çeşitli kullanım senaryoları olan bu yapıları kullanarak projelerde farkınızı gösterebilirisiniz.

Ayrıca dönüşümleri farklı yerde tutup bu methodların içinde de çağırabilirisiniz.

Olabildiğince temel bir anlatım yapmaya çalıştım. Bu methodlar kullanılarak basit gözüken ama projelerin temel yapılarından olan tür dönüşümlerini kontrol altına alarak projelerde kolay gelişimler sağlayabilirisiniz.