C#'ta foreach Döngü Değişkeni Neden Read-Only'dir?

8-Eylül-2009

C#'taki *foreach* döngülerinde kullanılan döngü değişkenlerinin neden *read-only* olduğu bana çok soruluyor. Soranların çoğu döngü değişkeninin *read-write* olması durumunda işlevselliğin artacağını iddia ediyorlar. Örneğin onlara göre eğer *foreach* döngü değişkeni *read-only* olmasaydı bir dizinin tüm elemanları belirli bir değerle şöyle doldurulabilirdi:

```
foreach (int x in a)
    x = val;
```

foreach döngü değişkeninin *read-only* olmasının basit bir nedeni var. *foreach* deyimi çokbiçimli olarak *IEnumerable* arayüzü yoluyla dolaşım uygular.[1] *IEnumerable* arayüzünün *GetEnumerator* metodu *IEnumerator* arayüzü türündendir. *IEnumerator* arayüzünün elemana erişmekte kullanılan *Current* property'si de *read-only*'dir. Aşağıdaki *foreach* döngüsüne bakın:

```
foreach (V v in x)
    <deyim>
```

C# standartlarına göre (burada bazı ayrıntıları göz ardı edeceğim) bu deyimin eşdeğeri şöyledir:[2]

```
{
    IEnumerator e = x.GetEnumerator();

    E e = x.GetEnumerator();
    try
    {
        V v;
        while (e.MoveNext())
        {
            v = (V) e.Current;
            <deyim>
        }
    }
    finally
    {
        // e.Dispose(); --> E, IDisposable arayüzünü
destekliyorsa
    }
}
```

Gördüğünüz gibi bu tanımlamaya göre *foreach* döngü değişkeni *read-write* olamaz. Ancak bu noktada *IEnumerable.Current* property'sinin neden *read-only* olduğu da sorulabilir. Eğer *Current* property'si *read-only* olmasaydı bu arayüzü destekleyen *collection* sınıfların *read-only* olma olanakları kaldırılmış olurdu. Tabi yine de *C++*'taki gibi *const iterator* sistemi uygulanabilirdi. Örneğin kütüphaneye *IEnumerable*

arayüzünün yanı sıra bir de *IConstEnumerable* arayüzü eklenebilirdi. *Microsoft* tasarımcıları böyle bir tasarımın sadeliği bozacağını düşünmüş olmalılar.

Burada C# için söylediklerim *VB.NET* için de aynı nedenlerden dolayı geçerli. *Java*'daki *foreach* deyimi de *C#*'takine çok benziyor*..NET*'teki *IEnumerable* arayüzünün *Java*'daki karşılığı *Iterable* arayüzü*.* Ancak *Java*'da döngü değişkeni *read-only* değil. Fakat döngü değişkeni yerel bir değişken gibi ele alındığından ona atama yapılması *collection* nesnesinin ilgili elemanına atama yapıldığı anlamına da gelmiyor. Peki *C/C++*'ta durum nedir? Bildiğiniz gibi *C* ve *C++*'ta zaten *foreach* deyimi yok. Ancak bu döngünün *"Range Based Loop"* ismiyle *C++*'a eklenmesine çoktan karar verildi. *C++0x*'te belirtilen bu döngü işlevsel olarak *C#* ve *Java*'dakinden farklı. Burada döngü değişkeni referans olabildiği için diziler üzerinde güncelleme işlemi de yapılabiliyor.

---

[1] *Generic* konusunun *C#*'a eklenmesiyle *IEnumerable* arayüzünün yanı sıra *IEnumerable<T>* arayüzü de *foreach* ile ilişkilendirilmiştir. Burada ben ayrıntıya girmek istemedim.

[2] C# standartlarına göre *bir collection* sınıfın *foreach* ile kullanılması için *IEnumerable* ya da *IEnumerable<T>* arayüzünü desteklemesi zorunlu değildir. Eğer *collection* sınıf *GetEnumerator* metodunu doğrudan bulunduruyorsa yine *foreach* deyimi ile kullanılabilir. Yine ben bu ayrıntılara girmek istemedim.