عنوان: **C# 6 - Index Initializers**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۷/۲۱ ۱۰:۳۰
آدرس: www.dotnettips.info
گروه‌ها: C#, C# 6.0

زمان زیادی از ارائه‌ی امکان Collection Initializer برای ایجاد یک متغیر از نوع Collection می‌گذرد؛ برای نمونه به مثال زیر توجه کنید:

```
enum USState {...}
var AreaCodeUSState = new Dictionary<string, USState>
    {
        {"408", USState.California},
        {"701", USState.NorthDakota},
        ...
    };
```

در پشت صحنه، کامپایلر، Collection Initializer را می‌گیرد، با استفاده از یک *Dictionary<TKey, TValue>* و با فراخوانی متد Add آن بر روی لیست Collection Initializer شروع به درج آن در دیکشنری ساخته شده می‌کند. Collection Initializer فقط بر روی کلاس هایی که در آن‌ها *IEnumerable* پیاده سازی شده باشد امکان پذیر است چرا که کامپایلر کار اضافه کردن مقادیر اولیه را به IEnumerable.Add() می‌سپارد.

اکنون در C# 6.0 ما می‌توانیم از Index Initializer استفاده کنیم:

```
enum USState {...}
var AreaCodeUSState = new Dictionary<string, USState>
    {
        ["408"] = USState.California,
        ["701"] = USState.NorthDakota,
        ...
    };
```

اولین تفاوتی که این دو روش با هم دارند این است که در حالت استفاده‌ی از Index Initializer پس از کامپایل، IEnumerable.Add() فراخوانی نمی‌شود. این تفاوت بسیار مهم است و کار اضافه کردن مقادیر اولیه را با استفاده از کلید (Key) ویژه انجام می‌دهد.
شبه کد مثال بالا به صورت زیر می‌شود:

Collection Initializer

```
create a Dictionary<string, USState>
add to new Dictionary the following items:
     "408", USState.California
     "701", USState.NorthDakota
```

Index Initializer

```
create a Dictionary<string, USState> then
using AreaCodeUSState's default Indexed property
    set the Value of Key "408" to USState.California
    set the Value of Key "701" to USState.NorthDakota
```

حال به مثال زیر توجه کنید:

Collection Initializer

```
enum USState {...}
var AreaCodeUSState = new Dictionary<string, USState>
```

```
        {
            { "408", USState.Confusion},
            { "701", USState.NorthDakota },
            { "408", USState.California},
            ...
        };
Console.WriteLine( AreaCodeUSState.Where(x => x.Key == "408").FirstOrDefault().Value );
```

Index Initializer

```
enum USState {...}
var AreaCodeUSState = new Dictionary<string, USState>
    {
        ["408"] = USState.Confusion,
        ["701"] = USState.NorthDakota,
        ["408"] = USState.California,
        ...
    };
Console.WriteLine( AreaCodeUSState2.Where(x => x.Key == "408").FirstOrDefault().Value );  // output = California
```

هر دو کد بالا با موفقیت کامپایل و اجرا می‌شود، اما در زمان اجرای Collection Initializer هنگامیکه می‌خواهد مقدار دوم "408" را اضافه کند با استثناء *ArgumentException* متوقف می‌شود چرا که کلید "408" از قبل وجود دارد.
اما در زمان اجرا، Index Initializer به صورت کامل و بدون خطا این کار را انجام می‌دهد و در کلید "408" مقدار USState.Confusion قرار می‌گیرد. سپس "701" مقدار USState.NorthDakota و بعد از استفاده‌ی مجدد از کلید "408" مقدار USState.California جایگزین مقدار قبلی می‌شود.

```
var fibonaccis = new List<int>
    {
        [0] = 1,
        [1] = 2,
        [3] = 5,
        [5] = 13
    }
```

این کد هم معتبر است و هم کامپایل می‌شود. البته معتبر است، ولی صحیح نیست. List<T> اجازه‌ی تخصیص اندیسی فراتر از اندازه‌ی فعلی را نمی‌دهد.
تلاش برای تخصیص مقدار 1 با کلید 0 به List<int>، سبب بروز استثناء *ArguementOutOfRangeException* می شود. وقتی List<T>.Add(item) فراخوانی می‌شود اندازه‌ی لیست یک واحد افزایش می‌یابد. بنابراین باید دقت داشت که Index Initializer از ()Add. استفاده نمی‌کند؛ در عوض با استفاده از خصوصیت اندیس پیش فرض، مقداری را برای یک کلید تعیین می‌کند.
برای چنین حالتی بهتر است از همان روش قدیمی Collection Initializer استفاده کنیم:

```
var fibonaccis = new List<int>()
    {
        1,
        3,
        5,
        13
    };
```