عنوان: **ویژگی Static Using Statements در سی شارپ 6**
نویسنده: مانی مهدوی
تاریخ: ۱۳۹۴/۰۷/۲۱ ۲۰:۵
آدرس: www.dotnettips.info
گروه‌ها: C#, C# 6.0

مروری بر کاربردهای مختلف دستور Using تا پیش از ارائه‌ی سی شارپ 6

-1 اضافه کردن فضاهای نام مختلف، برای سهولت دسترسی به اعضای آن:

```
using System.Collections.Generic;
```

-2 تعریف نام مستعار (alias name) برای نوع داده‌ها و فضای نام‌ها

```
using BLL = DotNetTipsBLLLayer;//نام مستعار برای فضای نام
using EmployeeDomain = DotNetTipsBLLLayer.Employee;//نام مستعار برای یک نوع داده
```

-3 تعریف یک بازه و مشخص کردن زمان تخریب یک شیء و آزاد سازی حافظه‌ی تخصیص داده شده:

```
using (var sqlConnection = new SqlConnection())
            {
                //کد
            }
```

در سی شارپ 6 ، **Static** Using Statements برای بهبود کدنویسی و تمیزتر نوشتن کدها ارائه شده‌است.
در ابتدا نحوه‌ی عملکرد اعضای static را مرور می‌کنیم. متغیرها و متدهایی که با کلمه‌ی کلیدی static معرفی می‌شوند، اعلام می‌کنند که برای استفاده‌ی از آنها به نمونه سازی کلاس آن‌ها احتیاجی نیست و برای استفاده‌ی از آنها کافی است نام کلاس را تایپ کرده (بدون نوشتن new) و متد و یا خصوصیت مورد نظر را فراخوانی کنیم.
با معرفی ویژگی جدید Static Using Statement نوشتن نام کلاس برای فراخوانی اعضای استاتیک نیز حذف می‌شود.
اتفاق خوبی است اگر بتوان اعضای استاتیک را همچون Data Typeهای موجود در سی شارپ استفاده کرد. مثلا بتوان به جای Console.WrriteLine() نوشت WriteLine() **نحوه استفاده از این ویژگی:** در ابتدای فایل و بخش معرفی کتابخانه‌ها بدین شکل عمل می‌کنیم using **static** namespace. **className** .
در بخش className، نام کلاس استاتیک مورد نظر خود را می‌نویسیم .
مثال :

```
using static System.Console;
using static System.Math;

namespace dotnettipsUsingStatic
{
    class Program
    {
        static void Main(string[] args)
        {

            Write(" *** Cal Area *** ");
            int r = int.Parse(ReadLine());
            var result = Pow(r, 2) * PI;
            Write($"Area  is : {result}");
            ReadKey();
       }
    }
}
```

همان طور که در کدهای فوق می‌بینید، کلاس‌های Console و Math، در ابتدای فایل با استفاده از ویژگی جدید سی شارپ 6 معرفی شده‌اند و در بدنه برنامه تنها با فراخوانی نام متدها و خصوصیت‌ها از آنها استفاده کرده ایم.

استفاده از ویژگی using static و Enum:

فرض کنید می‌خواهیم یک نوع داده‌ی شمارشی را برای نمایش جنسیت تعریف کنیم:

```
enum Gender
    {
        Male,
        Female
    }
```

تا قبل از سی شارپ 6 برای استفاده‌ی از نوع داده شمارشی بدین شکل عمل می‌کردیم:

```
var gender = Gender.Male;
```

و اکنون بازنویسی استفاده‌ی از Enum به کمک ویژگی جدید static using statement :

در قسمت معرفی فضاهای نام بدین شکل عمل می‌کنیم:

```
using static dotnettipsUsingStatic.Gender;
```

و در برنامه کافیست مستقیما نام اعضای Enum را ذکر کنیم .

```
var gender = Male;//تخصیص نوع داده شمارشی
WriteLine($"Employee Gender is : {Male}");//استفاده مستقیم از نوع داده شمارشی
```

استفاده از ویژگی using static و متدهای الحاقی :

تا قبل از ارائه سی شارپ 6 اگر نیاز به استفاده‌ی از یک متد الحاقی خاص همچون where در فضای نام System.Linq.Enumeable داشتیم می‌بایستی فضای نام System.Linq را به طور کامل اضافه می‌کردیم و راهی برای اضافه کردن یک فضای نام خاص درون فضای نام بزرگتر وجود نداشت.

اما با قابلیت جدید اضافه شده می‌توانیم بخشی از یک فضای نام را اضافه کنیم:

```
using static System.Linq.Enumerable;
```

متدهای استاتیک و متدهای الحاقی در زمان استفاده از ویژگی using static:

فرض کنید کلاس static ای بنام MyStaticClass داشته باشیم که متد Print1 و Print2 در آن تعریف شده باشند:

```
public static class MyStaticClass
    {
        public static void Print1(string parameter)
        {
            WriteLine(parameter);
        }
        public static void  Print2(this string parameter)
        {
            WriteLine(parameter);
        }

    }
```

برای استفاده از متدهای تعریف شده به شکل زیر عمل می‌کنیم :

```
فراخوانی تابع استاتیک//
Print1("Print 1");//روش اول
MyStaticClass.Print1("Prtint 1");//روش دوم
فراخوانی متد الحاقی استاتیک//
MyStaticClass.Print2("Print 2");
"print 2".Print2();
```

ویژگی‌های جدید ارائه شده در سی شارپ 6 برای افزایش خوانایی برنامه‌ها و تمیزتر شدن کدها اضافه شده‌اند. در مورد ویژگی‌های ارائه شده در مقاله‌ی جاری این نکته مهم است که گاهی قید کردن نام کلاس‌ها خود سبب افزایش خوانایی کدها می‌شود .