عنوان: **کمی درباره دستورات using**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۰۹/۲۲ ۰:۵۵
آدرس: www.dotnettips.info
گروه‌ها: C#, Visual Studio

شخصی سازی using directives موقعی که یک کلاس جدید را در VS.NET باز میکنید، فضاهای نامی مشخص و تکراری، همیشه به صورت پیش فرض صدا زده شده‌اند و این فضاهای نام را مایکروسافت بر اساس بیشترین کاربرد و استفاده توسط برنامه نویسان قرار داده است؛ ولی در خیلی از اوقات این فضاهای نام پیش فرض، چنان هم برای خیلی از برنامه نویسان کاربردی نداشته یا با توجه به برنامه هایی که می‌نویسند همیشه متفاوت هست و هربار مجبورند فضاهای نام خاصی را صدا بزنند.
برای مثال فضای نام System.ComponentModel.DataAnnotations را در نظر بگیرید که برنامه نویس میخواهد برای مدل‌های نوشته شده خود از تگ‌های متا استفاده کند و باید در هرکلاس ساخته شده، یکبار مورد بالا را صدا بزند که بیشتر باعث کند شدن کار برنامه نویس می‌شود. پس باید کاری کنیم که پیش فرض‌های فضای نام به آنچه خودمان میخواهیم تغییر پیدا کند.
برای این منظور، به محل نصب ویژوال استودیو رفته و مسیر زیر را دنبال کنید (به مسیر دقت کنید ،در اینجا زبان سی شارپ انتخاب شده است):

```
X:\...\IDE\ItemTemplates\CSharp\Code\1033
```

در اینجا تعدادی دایرکتوری با اسامی آشنا می‌بینید که داخل هر کدام از آن‌های یک فایل به اسم class.cs هست و اگر آن را باز کنید یک نمونه یا قالب برای usingها قابل مشاهده است. برای مثال ما وارد دایرکتوری class می‌شویم و فایل class.cs را باز می‌کنیم:

```
using System;
using System.Collections.Generic;
$if$ ($targetframeworkversion$ >= 3.5)using System.Linq;
$endif$using System.Text;
$if$ ($targetframeworkversion$ >= 4.5)using System.Threading.Tasks;
$endif$
namespace $rootnamespace$
{
    class $safeitemrootname$
    {
    }
}
```

الان باید با یک نگاه به الگو، مشخص باشد که چکار باید بکنید.
یک سری از فضاهای نام که در تمامی فریمورک‌ها استفاده میشوند به همان شکل عادی نوشته شده‌اند. ولی آنهایی که از نسخه‌ی خاصی از یک فریم ورک اضافه شده‌اند باید توسط شرط مورد نظر اضافه شده و اعلام شود که این فضای نام از چه نسخه‌ی فریم ورکی به بعد باید اضافه گردد:

```
$if$ ($targetframeworkversion$ >= 3.5)using System.Linq;//فضای نام مورد نظر
$endif$
```

حالا تغییرات را ذخیره کنید و در VS.NET یک کلاس جدید را ایجاد کنید. همانطور که خواهید دید، تغییرات شما اعمال شده‌است.
برای اعمال تغییرات نیازی به بستن و باز کردن مجدد VS.NET نمی‌باشد. در لحظه ایجاد کلاس الگو خوانده می‌شود.
حال در همان دایرکتوری سی شارپ دقت کنید، می‌بینید که برای موارد دیگری هم فایل هایی وجود دارند. برای مثال برای اینترفیس‌ها یا silverlight و ... که هر کدام را می‌توانید جداگانه تغییر دهید. *نکته:احتمال دارد در نسخه‌های متفاوت به خصوص پایین‌تر مثل نسخه 8 ویژوال استودیو ، فایل class.cs به صورت zip باشد که بعد از تغییرات باید دوباره به حالت zip بازگردانده شود.*

حذف فضای نام‌های اضافی

هر موقع که کلاس جدیدی میسازیم، namespaceها به صورت پیش فرض که در بالا اشاره کردیم وجود دارند و شاید اصلا در آن کلاس از آن‌ها استفاده نمی‌کنیم یا حتی خودمان در حین نوشتن کدها چند namespace خاص را اضافه می‌کنیم که شاید در طول برنامه نویسی چندتایی را بلا استفاده بگذاریم. برای همین همیشه فضای نام هایی صدا زده شده‌اند که اصلا در آن کلاس استفاده

نشده‌اند. پس برای همین بهتر هست که این رفرنس‌های بلا استفاده را پیدا کرده و آن‌ها را حذف کنیم.
شاید این سوال برای بعضی‌های پدید بیاد که چرا باید این‌ها را حذف کنیم، چون کاری هم با ما ندارند و ما هم کاری با آن‌ها نداریم؟
این کار چند علت میتواند داشته باشد:
تمیزکاری کد و خلوت شدن فضای کدنویسی
ممکن هست بعدها گیج کننده شود که من چرا از این‌ها استفاده کردم؟ در آینده با نگاه به یک کد تمیزتر متوجه میشوید یک کد از چه چیزهایی برای انجام کارش بهره‌مند شده و هم اینکه در کارهای گروهی و تیمی هم این مورد به شدت تاثیرگذار هست.
باعث کند شدن تحلیل‌های ایستا میشه (اینجا و اینجا)
کمپایل شدن کد کندتر میشه
موقع تست برنامه، اجرای اولیه کندتر خواهد بود چون CLR باید این نوع موارد را شناسایی و حذف کند
همه موارد بالا در مورد رفرنس‌های موجود یا همان dllهای موجود در شاخه‌ی Bin و References هم صدق می‌کند.
برای حذف فضاهای نام اضافی در یک صفحه می‌توانید از طریق این مسیر انجام بدید:

```
Edit>
```

IntelliSense >Organize Usings>Remove Unused using برای مرتب سازی هم گزینه Sort Usings و انجام هر دو کار Remove and Sort موجود هست. البته اگه روی صفحه هم راست کلیک کنید گزینه Organize Usings هم وجود دارد. می توانید از ابزارهایی چون Power tools Extensions هم استفاده کنید (در صورتی که ویژوال استودیوی شما گزینه‌های مورد نظر را ندارد، این ابزار را نصب نمایید)
در صورتی که از ابزارهایی چون telerik یا devexpress استفاده می‌کنید یا از هر ابزار اضافی که بر روی IDE نصب می‌شود، عموما چنین گزینه هایی حتی با امکانات وسیعتر وجود دارند. مثلا whole tomato هم یکی از این ابزارهاست.
این نکته را هم خاطر نشان کنم در صورتیکه فضاهای نامی بین پیش پردازنده ها که در قبل توضیح دادیم محصور شده باشند، حذف نخواهند شد و همانطور باقی خواهند ماند.
در مورد کامنت‌های بین usingها به قطعه کد زیر نگاه کنید:

```
using System;
/* Comment before remains */
using /* Comment between removed */ System.Linq;
// Comment after remains
namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("My Example");
}
}
}
```

و حالا بعد از حذف فضای نام‌های اضافی

```
using System;
/* Comment before remains */
// Comment after remains
namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("My Example");
}
}
}
```

برای اینکه این عمل را بتونید در کل صفحات اعمال کنید می‌توانید از *cleanup selected code* هم استفاده کنید؛ به جز اینکه فضاهای نام اضافی را هم پاک می‌کند، کلیه کدهای شما را در قالبی شکیل‌تر و خواناتر قرار خواهد داد.

با کلیدهای *Ctrl+k+d* سند انتخابی و با کلیدهای ترکیبی *Ctrl+k+f* هم محدوده انتخاب شده قالب بندی می‌شود. یکی دیگر از ابزارهایی که می‌توان با آن‌ها به کد سر و سامان بهتری داد، افزونه‌ی codemaid هست.

ویژگی سی شارپ 6 در مورد Using فرض کنید ما یک کلاس ایستا به نام utilities ایجاد کردیم که یک متد به اسم addints دارد. حالا و این کلاس در namespace به نام SomeNamespace قرار دارد. مطمئنا در این حالت ما ابتدا فضای نام را using میکنیم و سپس در کد کلاس، متد را به شکل زیر صدا میزنیم:

```
using System;
using SomeNamespace;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int sum = Utility.AddInts(5, 2);

            Console.ReadLine();
        }
    }
}
```

ولی در سی شارپ 6 میتوانید بعد از فضای نام، یک . گذاشته و سپس اسم کلاس ایستا static را بیاورید و در کد مستقیما متد دلخواه خود را صدا بزنید.

به شکل زیر دقت کنید:

```
using System;
using SomeNamespace.Utility;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int sum = AddInts(5, 2);

            Console.ReadLine();
        }
    }
}
```

نکته پایانی:در visual studio 2014 فضاهای نام اضافی به رنگ خاکستری نمایش داده می‌شوند.

منابع: http://blogs.msdn.com/b/steve/archive/2007/04/10/changing-the-default-using-directives-in-visual-studio.aspx

http://stackoverflow.com/questions/629667/why-remove-unused-using-directives-in-c

http://stackoverflow.com/questions/5755942/how-do-you-auto-format-code-in-visual-studio

http://csharp.2000things.com/

نظرات خوانندگان

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۰۲ ۲۰:۲۸

یک نکته اضافه در مورد فایل class.cs در ابتدای مقاله.
مدتی هست که من هر موقع کلاسی به خصوص برای بخش مدل‌ها ایجاد میکنم مرتب هر کلاسی را باید یک public را بنویسم چون به طور پیش فرض کلاس هایی که دات نت ایجاد میکند private هستند.
برای حل این مشکل در فایل class عبارت public را قبل از کلمه class به آن اضافه کنید:

```
class $safeitemrootname$
    {
    }

========= To

public class $safeitemrootname$
    {
    }
```

احتمالا این معضل خیلی‌ها هست چون نوشتن تعداد کلاس‌های عمومی بیشتر از خصوصی است