عنوان: **C# 6 - String Interpolation**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۷/۱۱ ۱۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: C#, C# 6.0

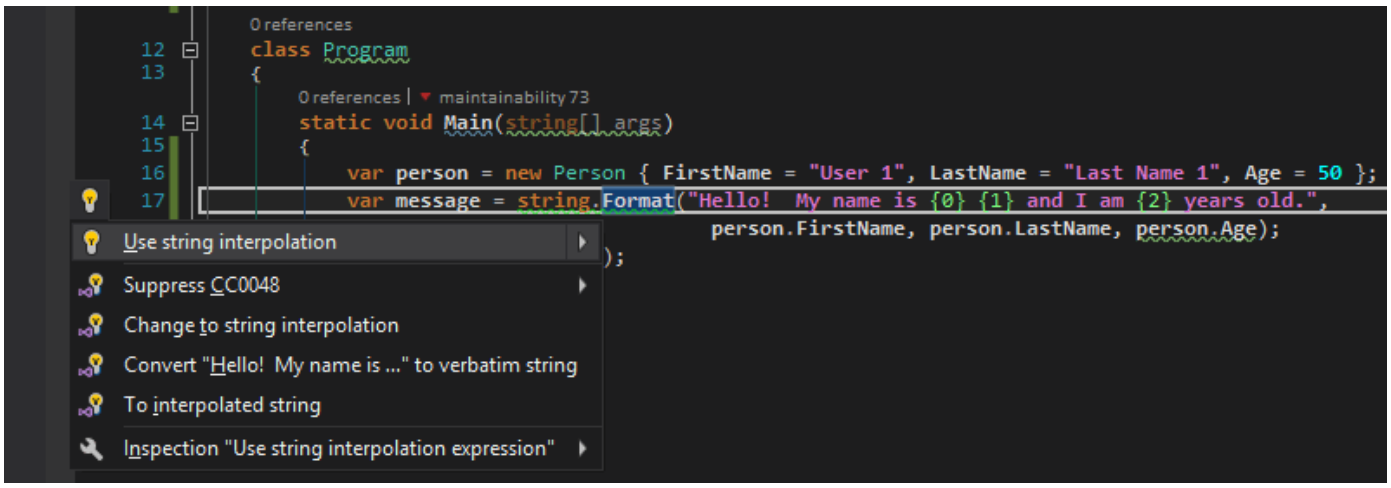
تا پیش از C# 6 یکی از روش‌های توصیه شده‌ی جهت اتصال رشته‌ها به هم، استفاده از متدهایی مانند string.Format و StringBuilder.AppendFormat بود:

```
using System;

namespace CS6NewFeatures
{
    class Person
    {
        public string FirstName { set; get; }
        public string LastName { set; get; }
        public int Age { set; get; }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var person = new Person { FirstName = "User 1", LastName = "Last Name 1", Age = 50 };
            var message = string.Format("Hello!  My name is {0} {1} and I am {2} years old.",
                                        person.FirstName, person.LastName, person.Age);
            Console.Write(message);
        }
    }
}
```

مشکل این روش، کاهش خوانایی آن با بالا رفتن تعداد پارامترهای متد Format است و همچنین گاهی از اوقات فراموش کردن مقدار دهی بعضی از آن‌ها و یا حتی ذکر ایندکس‌هایی غیر معتبر که در زمان اجرا، برنامه را با یک خطا متوقف می‌کنند.
در C# 6 جهت رفع این مشکلات، راه حلی به نام String interpolation ارائه شده‌است و اگر افزونه‌ی ReSharper یا یکی از افزونه‌های Roslyn را نصب کرده باشید، به سادگی امکان تبدیل کدهای قدیمی را به فرمت جدید آن خواهید یافت:



در این حالت کد قدیمی فوق، به کد ذیل تبدیل خواهد شد:

```
static void Main(string[] args)
{
    var person = new Person { FirstName = "User 1", LastName = "Last Name 1", Age = 50 };
    var message = $"Hello!  My name is {person.FirstName} {person.LastName} and I am {person.Age} years
old.";
```

```
    Console.Write(message);
}
```

در اینجا ابتدا یک $ در ابتدای رشته قرار گرفته و سپس هر متغیر به داخل {} انتقال یافته‌است. همچنین دیگر نیازی هم به ذکر string.Format نیست.
عملیاتی که در اینجا توسط کامپایلر صورت خواهد گرفت، تبدیل این کدهای جدید مبتنی بر String interpolation به همان string.Format قدیمی در پشت صحنه‌است. بنابراین این قابلیت جدید C# 6 را به کدهای قدیمی خود نیز می‌توانید اعمال کنید. فقط کافی است VS 2015 را نصب کرده باشید و دیگر شماره‌ی دات نت فریم ورک مورد استفاده مهم نیست.

امکان انجام محاسبات با String interpolation

زمانیکه $ در ابتدای رشته قرار گرفت، عبارات داخل {}ها توسط کامپایلر محاسبه و جایگزین می‌شوند. بنابراین می‌توان چنین محاسباتی را نیز انجام داد:

```
var message2 = $"{Environment.NewLine}Test {DateTime.Now}, {3*2}";
Console.Write(message2);
```

بدیهی اگر $ ابتدای رشته فراموش شود، اتفاق خاصی رخ نخواهد داد.

تغییر فرمت عبارات نمایش داده شده توسط String interpolation

همانطور که با string.Format می‌توان نمایش سه رقم جدا کننده‌ی هزارها را فعال کرد و یا تاریخی را به نحوی خاص نمایش داد، در اینجا نیز همان قابلیت‌ها برقرار هستند و باید پس از ذکر یک : عنوان شوند:

```
var message3 = $"{Environment.NewLine}{1000000:n0} {DateTime.Now:dd-MM-yyyy}";
Console.Write(message3);
```

حالت کلی و استاندارد آن در متد string.Format به صورت {index[,alignment][:formatString]} است.

سفارشی سازی String interpolation

اگر متغیر رشته‌ای معرفی شده‌ی توسط $ را با یک var مشخص کنیم، نوع آن به صورت پیش فرض، از نوع string خواهد بود. برای نمونه در مثال‌های فوق، message و message2 از نوع string تعریف می‌شوند. اما این رشته‌های ویژه را می‌توان از نوع IFormattable و یا FormattableString نیز تعریف کرد.
در حقیقت رشته‌های آغاز شده‌ی با $ از نوع IFormattable هستند و اگر نوع متغیر آن‌ها ذکر نشود، به صورت خودکار به نوع FormattableString که اینترفیس IFormattable را پیاده سازی می‌کند، تبدیل می‌شوند. بنابراین پیاده سازی این اینترفیس، امکان سفارشی سازی خروجی string interpolation را میسر می‌کند. برای نمونه می‌خواهیم در مثال message2، نحوه‌ی نمایش تاریخ را سفارشی سازی کنیم.

```
class MyDateFormatProvider : IFormatProvider
{
    readonly MyDateFormatter _formatter = new MyDateFormatter();

    public object GetFormat(Type formatType)
    {
        return formatType == typeof(ICustomFormatter) ? _formatter : null;
    }

    class MyDateFormatter : ICustomFormatter
    {
        public string Format(string format, object arg, IFormatProvider formatProvider)
        {
            if (arg is DateTime)
                return ((DateTime)arg).ToString("MM/dd/yyyy");
```

```
            return arg.ToString();
        }
    }
}
```

در اینجا ابتدا کار با پیاده سازی اینترفیس IFormatProvider شروع می‌شود. متد GetFormat آن همیشه به همین شکل خواهد بود و هر زمانیکه نوع ارسالی به آن ICustomFormatter بود، یعنی یکی از اجزای {} دار در حال آنالیز است و خروجی مدنظر آن همیشه از نوع ICustomFormatter است که نمونه‌ای از پیاده سازی آن‌را جهت سفارشی سازی DateTime ملاحظه می‌کنید.
پس از پیاده سازی این سفارشی کننده‌ی تاریخ، نحوه‌ی استفاده‌ی از آن به صورت ذیل است:

```
static string formatMyDate(FormattableString formattable)
{
     return formattable.ToString(new MyDateFormatProvider());
}
```

ابتدا یک متد static را تعریف کنید که ورودی آن از نوع FormattableString باشد؛ از این جهت که رشته‌های شروع شده‌ی با $ نیز از همین نوع هستند. سپس سفارشی سازی پردازش {}ها در قسمت ToString آن انجام می‌شود و در اینجا می‌توان یک IFormatProvider جدید را معرفی کرد.
در ادامه برای اعمال این سفارشی سازی، فقط کافی است متد formatMyDate را به رشته‌ی مدنظر اعمال کنیم:

```
var message2 = formatMyDate($"{Environment.NewLine}Test {DateTime.Now}, {3*2}");
Console.Write(message2);
```

و اگر تنها می‌خواهید فرهنگ جاری را عوض کنید، از روش ساده‌ی زیر استفاده نمائید:

```
public static string faIr(IFormattable formattable)
{
    return formattable.ToString(null, new CultureInfo("fa-Ir"));
}
```

در اینجا با اعمال متد faIr به عبارت شروع شده‌ی با $، فرهنگ ایران به رشته‌ی جاری اعمال خواهد شد.
نمونه‌ی کاربردی‌تر آن اعمال InvariantCulture به String interpolation است:

```
static string invariant(FormattableString formattable)
{
    return formattable.ToString(CultureInfo.InvariantCulture);
}
```

یک نکته: همانطور که عنوان شد این قابلیت جدید با نگارش‌های قبلی دات نت نیز سازگار است؛ اما این کلاس‌های جدید را در این نگارش‌ها نخواهید یافت. برای رفع این مشکل تنها کافی است این کلاس‌های یاد شده را به صورت دستی در فضای نام اصلی آن‌ها تعریف و اضافه کنید. یک مثال

غیرفعال سازی String interpolation

اگر می‌خواهید در رشته‌ای که با $ شروع شده، بجای محاسبه‌ی عبارتی، دقیقا خود آن‌را نمایش دهید (و { را escape کنید)، از {{}} استفاده کنید:

```
var message0 = $"Hello! My name is {person.FirstName} {{person.FirstName}}";
```

در این مثال اولین {} محاسبه خواهد شد و دومی خیر.

پردازش عبارات شرطی توسط String interpolation

همانطور که عنوان شد، امکان ذکر یک عبارت کامل هم در بین {} وجود دارد (محاسبات، ذکر یک عبارت LINQ، ذکر یک متد و امثال آن). اما در این میان اگر یک عبارت شرطی مدنظر بود، باید بین () قرار گیرد:

```
Console.Write($"{(person.Age>50 ? "old": "young")}");
```

علت اینجا است که کامپایلر سی‌شارپ، : بین {} را به format specifier تفسیر می‌کند. نمونه‌ی آن‌را پیشتر با مثال «تغییر فرمت عبارات نمایش داده شده» ملاحظه کردید. ذکر : در اینجا به معنای شروع مشخص سازی فرمتی است که قرار است به این حاصل اعمال شود. برای تغییر این رفتار پیش فرض، کافی است عبارت مدنظر را بین () ذکر کنیم تا تمام آن به صورت یک عبارت سی‌شارپ تفسیر شود.