عنوان: **C# 6 - Expression-Bodied Members**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۷/۱۸ ۱۶:۵۵
آدرس: www.dotnettips.info
گروه‌ها: C#, C# 6.0

در ادامه مطالب منتشر شده در رابطه با قابلیت‌های جدید سی‌شارپ 6، در این مطلب به بررسی یکی دیگر از این قابلیت‌ها، با نام Expression-Bodied Members خواهیم پرداخت. در واقع در سی‌شارپ 6، هدف، ساده‌سازی سینتکس و افزایش بهره‌وری برنامه‌نویس می‌باشد. در نسخه‌های قبلی سی‌شارپ برای یکسری از اعمال روتین می‌بایستی روالی‌هایی را مدام تکرار می‌کردیم؛ به عنوان مثال در تعریف پراپرتی‌های یک کلاس در حالت get-only باید هر بار توسط return مقداری را برگردانیم:

```
public class Person
{
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public string FullName
   {
       get
       {
                return FirstName + " " + LastName;
       }
   }
}
```

نوشتن پراپرتی‌هایی همانند FullName منجر به نوشتن خطوط کد اضافه‌تری خواهد شد، هرچند می‌توان این حالت را با برداشتن خطوط اضافی بهبود بخشید:

```
public string FullName
{
       get { return FirstName + " " + LastName; }
}
```

اما در سی‌شارپ 6 میتوان آن را توسط expression body به یک خط کاهش داد!

استفاده از expression body برای پراپرتی‌های get-only (فقط خواندنی):

اگر در کلاس‌هایتان پراپرتی‌های get-only دارید، به راحتی می‌توانید بدنه‌ی پراپرتی را با استفاده از expression syntax خلاصه‌نویسی کنید. در واقع شما با استفاده از سینتکس lambda expression اقدام به نوشتن بدنه پراپرتی‌های موردنظرتان می‌کنید. یعنی به جای نوشتن کدی مانند:

```
{ get { return your expression; } }
```

به راحتی می‌توانید از سینتکس زیر استفاده نمائید:

```
=> your expression;
```

به عنوان مثال، میتوان پراپرتی FullName را در کلاس Person با کمک قابلیت expression body به صورت زیر بازنویسی کنیم:

```
public class Person
{
        public string FirstName { get; set; }
        public string LastName { get; set; }

        public string FullName => FirstName + " " + LastName;
}
```

با کد فوق به راحتی توانستیم قسمت‌های اضافه‌ای را حذف کنیم. اکنون ممکن است بپرسید آیا این تغییر در performance برنامه

تاثیری دارد؟ خیر؛ زیرا سینتکس فوق دقیقاً همان کد ILای را تولید خواهد کرد که در حالت عادی تولید می‌شود. همچنین delegateی را تولید نخواهد کرد؛ بلکه تنها از سینتکس lambda expression برای خلاصه‌نویسی بدنه پراپرتی استفاده می‌کند. در حال حاضر برای حالت setter سینتکسی ارائه نشده است.

استفاده از expression body برای Indexerها:

همچنین از این قابلیت برای Indexerها نیز میتوان استفاده کرد، مثلاً به جای نوشتن کد زیر:

```
public string this[int number]
{
            get
            {
                if (number >= 0 && number < _values.Length)
                {
                    return _values[number];
                }
                return "Error";
            }
}
```

می‌توانیم کد فوق را به این صورت خلاصه‌نویسی کنیم:

```
public string this[int number] => (number >= 0 && number < _values.Length) ? _values[number] : "Error";
```

نکته: توجه داشته باشید که در هر دو حالت فوق تنها می‌توانیم برای get از expression body استفاده کنیم، هنوز سینتکسی برای حالت set ارائه نشده است.

استفاده از expression body برای متدها:

برای متدها نیز می‌توانیم از قابلیت عنوان شده استفاده نمائیم، به عنوان مثال اگر داخل کلاس Person متد زیر را داشته باشیم:

```
public override string ToString()
{
      return FirstName;
}
```

می‌توانیم آن را به صورت زیر بنویسیم:

```
public override string ToString() => FirstName;
```

همانطور که مشاهده می‌کنید به جای نوشتن curly braces یا {} از lambda arrow یا => استفاده کرده‌ایم. در اینجا عبارت سمت راست lambda arrow نمایانگر بدنه‌ی متد است. همچنین برای متدهای دارای پارامتر نیز به این صورت عمل می‌کنیم:

```
public int DoubleTheValue(int someValue) => someValue * 2;
```

یک عضو از کلاس که به صورت expression body نوشته شده باشد، expression bodied member نامیده می‌شود. این عضو از کلاس در ظاهر شبیه به عبارات لامبدای ناشناس (anonymous lambda expression) است. اما یک expression bodied member باید دارای نام، مقدار بازگشتی و بدنه متد باشد.
تقریباً تمامی access modifierها در این حالت قابلیت استفاده را دارند. تنها متدهای abstract نمی‌توانند استفاده شوند.

محدودیت‌های Expression Bodied Members

یکی از محدودیت‌های استفاده از expression body داشتن چندین خط دستور برای بدنه متدهایمان است. در اینحالت باید از روش سابق (statement body) استفاده نمائید.
یکی دیگر از محدودیت‌ها عدم امکان استفاده از if, else, switch است. به عنوان مثال نمی‌توان کد زیر را با داشتن if و else به صورت expression body نوشت:

```
public override string ToString()
{
	if (MiddleName != null)
	{
		return FirstName + " " + MiddleName + " " + LastName;
	}
	else
	{
		return FirstName + " " + LastName;
	}
}
```

برای حالت فوق به عنوان یک روش جایگزین می‌توان از conditional operator استفاده کرد:

```
public override string ToString() =>
                    (MiddleName != null)
                    ? FirstName + " " + MiddleName + " " + LastName
                    : FirstName + " " + LastName;
```

همچنین نمی‌توان از for, foreach, while, do در expression body استفاده کرد، به جای آن می‌توان از عبارت‌های LINQ برای بدنه تابع استفاده کرد. به عنوان مثال متد زیر:

```
public IEnumerable<int> SmallNumbers()
{
    for (int i = 0; i < 10; i++)
        yield return i;
}
```

را می‌توان در حالت expression body به این صورت نوشت:

```
public IEnumerable<int> SmallNumbers() => from n in Enumerable.Range(0, 10)
                                                                    select n;
```

و یا به این صورت:

```
public IEnumerable<int> SmallNumbers() => Enumerable.Range(0, 10).Select(n => n);
```

همانطور که عنوان شد، استفاده از expression body در قسمت پراپرتی‌ها تنها محدود به پراپرتی‌های get-only (فقط خواندنی) میباشد.

استفاده از این قابلیت برای متدهای سازنده
استفاده در رخدادها
استفاده در finalizers

نکته: اگر می‌خواهید expression bodied member شما هم initializer داشته باشد و همچنین یک read only auto property باشد، باید مقداری سینتکس آن را تغییر دهید. همانطور که می‌دانید auto propertyها نیازی به backing field ندارند؛ بلکه در زمان کامپایل به صورت خودکار تولید خواهند شد. در نتیجه برای مقداردهی اولیه به backing fieldها می‌توانیم درون سازنده کلاس آنها را initialize کنیم:

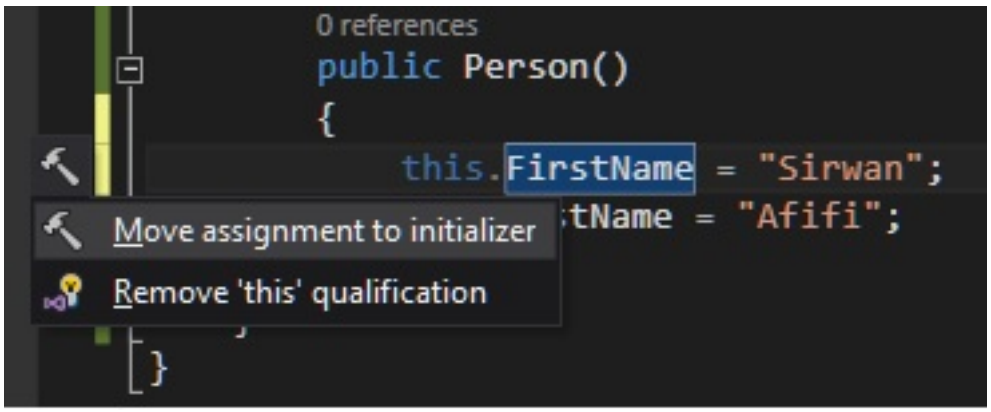
```
public class Person
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }

        public Person()
        {
            this.FirstName = "Sirwan";
            this.LastName = "Afifi";
        }
    }
```

برای نوشتن پراپرتی‌های فوق به صورت expression body می‌توانیم به این صورت عمل کنیم:

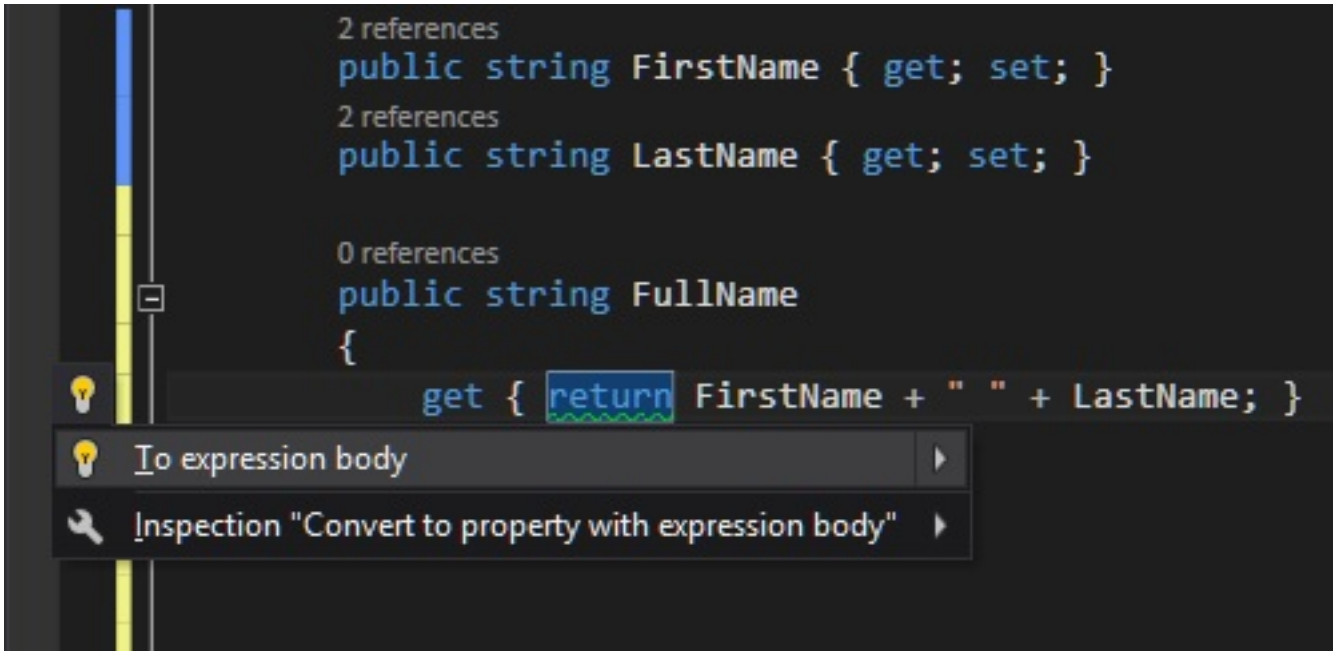
```
public string FirstName { get; set; } = "Sirwan";
public string LastName { get; set; } = "Afifi";
```

اگر ReSharper را نصب کرده باشید، به شما پیشنهاد می‌دهد که از expression body استفاده نمائید: :

برای حالت فوق:



برای پراپرتی‌ها:


2 references
public string FirstName { get; set; }
2 references
public string LastName { get; set; }
0 references
public string FullName
{
get { return FirstName + " " + LastName; }
To expression body
Inspection "Convert to property with expression body"