

扩展方法

北京理工大学计算机学院
金旭亮



复习“静态方法”语法特性

给方法添加一个**static**关键字，此方法成为静态方法……

```
class MyClass
{
    public static void MyStaticFunc(string info)
    {
        Console.WriteLine(info);
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        MyClass.MyStaticFunc( "Hello" );
        Console.ReadKey();
    }
}
```

静态方法是归属于“类”的，因此，在使用静态方法时无需创建对象，直接通过“**类名.静态方法名(参数列表)**”就可以调用它了。

C# 引入了一种神奇的“静态方法”——

扩展方法 (Extension Method)

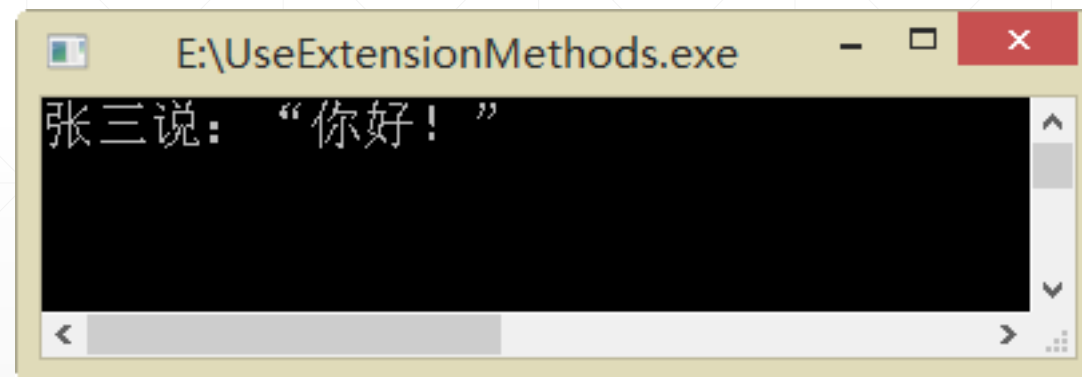
```
class Person
{
    2 references
    public string Name { get; set; }
    1 reference
    public int Age { get; set; }
}
```

定义一个简单的类

请看以下“神奇”的代码：


```
static void Main(string[] args)
{
    var person = new Person()
    {
        Name = "张三",
        Age = 30
    };
    person.SayHello();
    Console.ReadKey();
}
```

SayHello这个方法从哪冒出来的？
Person类在定义时并没有这个方法啊！

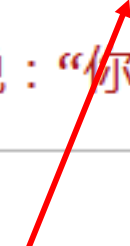


示例： UseExtensionMethods

扩展方法通常定义在“**静态**”类中



```
static class PersonExtensions
{
    1 reference
    public static void SayHello(this Person person)
    {
        Console.WriteLine("{0}说：‘你好！’", person.Name);
    }
}
```



扩展方法中第一个参数前有一个**this**，专用于指明此扩展方法所“适用”的类型。
所以，扩展方法的第一个参数既起到普通参数的作用，同时也是“扩展方法”的标识。

扩展方法的特点

扩展方法是一种特殊的静态方法，但其调用方式却与调用实例方法一样。

```
person.SayHello();
```

“扩展方法”使你能向现有的类添加新的方法，却不需要创建新的派生类或直接修改这个类的源代码。

扩展方法可以随意定义参数和返回值

扩充的参数

```
public static void SayHelloTo(this Person p1, Person p2)
{
    Console.WriteLine("{0}对{1}说：“您吃了吗？”", p1.Name, p2.Name);
}
```

扩展方法可以有返回值

```
public static bool IsOlderThan(this Person p1, Person p2)
{
    return p1.Age > p2.Age;
}
```


为什么我们能“扩展”一个类的方法？

结论：

在编译使用了扩展方法的代码时，编译器会生成相应的IL汇编指令，将这些调用代码转换为对相应静态方法的调用。

编译器是“幕后英雄”！

开发实践中扩展方法用在何处？

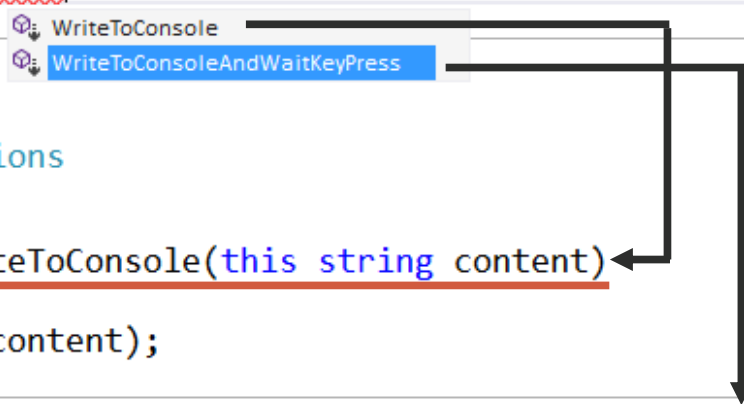
当你需要给某个类扩充新功能，但又不想定义一个新的子类，或者是想给一堆的类扩充一样的功能，这时可以考虑使用扩展方法。

应用实例

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        "my String".WriteToConsole();
        "Press any Key...".wri;
    }
}

0 references
public static class Extensions
{
    1 reference
    public static void WriteToConsole(this string content)
    {
        Console.WriteLine(content);
    }
}

0 references
public static void WriteToConsoleAndWaitKeyPress(this string content)
{
    Console.Write(content);
    Console.ReadKey();
}
}
```



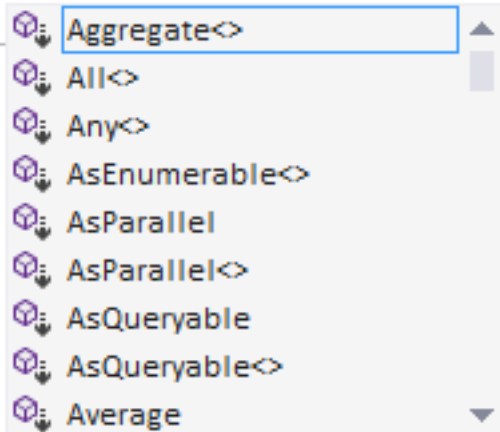
给String类添加专用于控制台应用程序的扩展方法，可以简化编码。

Visual Studio能正确地识别出扩展方法并给出智能提示。

示例：WriteToConsole

扩展方法的一个真实应用是.NET基类库所提供的一组集合查询方法。

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        var nums = new int[] { 1, 2, 3, 4 };
        nums.
    }
}
```



它们向现有的IEnumerable和IEnumerable<T>类型添加了各种常用的数据查询功能。

IEnumerable和IEnumerable<T>是绝大多数集合类型都实现的接口，从而大大地减少了反复编写类似功能代码的需求。

动手动脑

如果给MyClass类自己有一个Process()方法，又为其定义了一个同名的Process()**扩展方法**，两者的方法声明一模一样。那么，问题来了：

当针对MyClass对象调用Process()方法时，到底是调用哪个？

请自行编写示例代码进行测试一下。

参考示例：TheSameName

问题的答案

如果扩展方法与原始类型的方法相同，那么到底调用哪个呢？

与原始类型的方法具有相同名称和签名的扩展方法
永远不会被调用！

举个例子来说：

如果某个类型具有一个名为`Process(int i)`的方法，而又有一个具有相同名字和参数的扩展方法，则编译器总是生成调用实例方法的指令（而不是调用对应于扩展方法的静态方法的指令）。

“扩展方法” 语法特性小结

- 必须定义在一个非泛型的静态类中（需要重用的扩展方法推荐放到独立的类库项目中，编译为程序集）
- 扩展方法不能存取被扩展类型的私有成员
- 无法隐藏或重写被扩展类型的方法
- 必须手动在使用扩展方法的源代码文件中引入扩展方法的命名空间（namespace），并给项目添加对于扩展方法所在程序集的引用，否则，源码无法编译。

知识巩固练习

为字符串类型编写以下扩展方法：

(1) 能把一个字符串自动地转换为数值类型。

(2) 能验证某一字符串是否是一个有效的邮编或身份证号。