扩展方法

扩展方法使您能够向现有类型“添加”方法，而无需创建新的派生类型、重新编译或以其他方式修改原始类型。

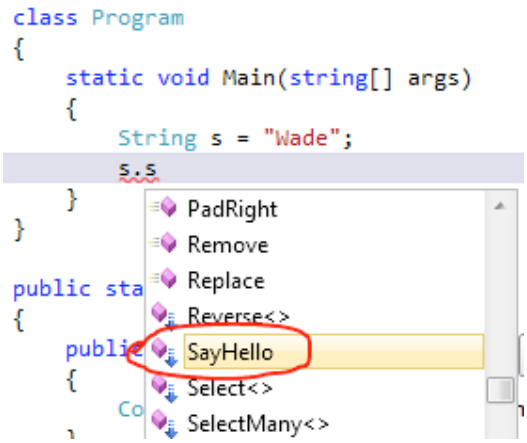
我们来为String类型定义一个简单的扩展方法：

```
public static class Extensions
{
    public static void SayHello(this String name)
    {
        Console.WriteLine("Hello {0}",name);
    }
}
```

之后，我们可以在main函数中这样使用扩展方法：

```
static void Main(string[] args)
{
    String s = "Wade";
    s.SayHello();
}
```

这样看起来就像是String类型中直接定义好的方法一样，而且还能被智能感知到



我们注意这个方法名前有个向下的箭头，表示这是一个扩展方法的意思。
但是我们需要认识到这个扩展方法并非真的是String的实例方法，它只是一个静态方法而已，所以我们将上面代码中的s设置为null时调用SayHello()方法，不会报空引用的错误。

注意事项

扩展方法是静态类中的静态方法；

扩展方法的第一个参数的类型之前必需使用this修饰符，this指代被扩展的类；

扩展方法的优先级比常规方法要低，如果某个类拥有一个同名方法的话，那么调用的将是实例方法；

扩展方法隐式使用ExtensionAttribute；

扩展方法不仅可以“扩展”基本类型和自定义复合类型，还可以扩展接口和委托。

```
interface IMyInterface
{
    void MethodA();
}
static class Extensions
```

```
{
    public static void MethodB(this IMyInterface myInterface)
    {
        Console.WriteLine("this is Extension MethodB");
    }
}

class classA :IMyInterface
{
    public void MethodA()
    {
       Console.WriteLine("This is classA MethodA");
    }
}
class classB : IMyInterface
{
     public void MethodA()
     {
       Console.WriteLine("This is classB MethodA");
     }
}
class Program
{
     static void Main(string[] args)
     {
        classA A = new classA();
        classB B = new classB();
        A.MethodA();
        A.MethodB();
        B.MethodA();
        B.MethodB();
     }
      /*Output:
       * This is classA MethodA
       * this is Extension MethodB
       * This is classB MethodA
       * this is Extension MethodB
       */
}
```

泛型扩展方法

当扩展方法与泛型结合在一起时，就变得非常强大了。下面我们来看下例子：

```
static class Extensions
{
     //首先我们为实现了IList接口的集合泛型类型添加了一个扩展方法
     public static IList<TSource> MyFind<TSource>(this IList<TSource> list, Func<TSource,
bool> fun)
    {
        List<TSource> resultList = new List<TSource>();
        foreach (TSource item in list)
        {
            if (fun(item))
            {
                resultList.Add(item);
            }
```

```
            }
            return resultList;
        }
}
class Program
{
        static void Main(string[] args)
        {
            IList<int> numList = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8 };

            //然后调用自己定义的扩展方法，是不是有点Linq方法的感觉
            IList<int> resultList = numList.MyFind(n => n % 2 == 0);

            foreach (var item in resultList)
            {
                Console.WriteLine(item);
            }
            /*Output
             * 2
             * 4
             * 6
             * 8
             */
        }
}
```

我们可以利用Reflector这样的工具，查看下System.Linq命名空间，就像我们看见的那样，Linq其实就是一大堆扩展方法（还有泛型的）组成。