C#中的 this 扩展方法与 javascript 中的 prototype 方法

this 扩展方法是在 Dotnet3.0中加入的一种新特性.

> "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。扩展方法是一种特殊的静态方法，但可以像扩展类型上的实例方法一样进行调用。对于用 C# 和 Visual Basic 编写的客户端代码，调用扩展方法与调用在类型中实际定义的方法之间没有明显的差异。最常见的扩展方法是 LINQ 标准查询运算符，这些运算符在现有 System.Collections..::.IEnumerable 和 System.Collections.Generic..::.IEnumerable<(Of <(T>)>) 类型中添加了查询功能。若要使用这些标准查询运算符，请先使用 using System.Linq 指令将它们纳入范围中。然后，任何实现了 IEnumerable<(Of <(T>)>) 的类型看起来都具有 GroupBy、OrderBy、Average 等实例方法。....."
>
> "*通常，建议您只在不得已的情况下才实现扩展方法，并谨慎地实现。只要有可能，必须扩展现有类型的客户端代码都应该通过创建从现有类型派生的新类型来达到这一目的*"
>
> 摘自 MSDN 介绍:http://msdn.microsoft.com/zh-cn/library/bb383977.aspx

我们暂且先不管 MSDN 的注意事项,我们关注的是这种方法和 javascript 中原型方法的异同.

举例:去除字符串中的所以空格

例子本身特简单,直接上代码

C#版本-传统写法:

首先定义一个 trimStrTrad 方法,代码如下

```
public static string trimStrTrad(string SourceStr)
{
      return Regex.Replace(SourceStr, @"\s", "");
}
```

在每次需要去除字符串空格的时候,调用此方法,如:

```
var strA = " aa bb   ccc ";
trimStrTrad(strA);
```

javascript 版本-传统写法:

过程同上,无需赘述.

下面,我们看一下扩展方法是如何实现的.

在认识上,我对 javascript 的原型要早于 C#的扩展方法,先演示 js 版本的.

javascript 版本:

```
String.prototype.trimStr = function() {
      return this.replace(/\s/g, "");
};
```

使用方法:`var strA=" aa bb   ccc ";strA.trimStr();`

C#版本

```
 public static string trimStr(this string sourceStr)
 {
      return Regex.Replace(sourceStr, @"\s", "");
 }
```

使用方法:其实和 javascript 版本的是一模一样的: `var strA=" aa bb   ccc ";strA.trimStr();`

2 种版本的功能是一样的,就是 string 这个类添加一个扩展方法,使得所有这个类的实例化对象能够直接使用这个方法.对于 javascript 来说,这种方式我们用的习惯了,也就不觉得有什么了,但是 C#的这种实现方式也算是一种改进.这也是一种趋势吧,很多动态语言的崛起,使得传统的静态语言也在追求更便捷的使用方式.

下面我们来试着解决一下实际的一个问题:

实际问题:假如在国外某公司的系统中有一个 person 类,代码如下:

```
public   class Person
{
    private string _name = "Noname";
    private int _age = 1;

    public string name
    {
        get { return this._name; }
        set { this._name = value; }
    }

    public int age
```

```
    {
        get { return this._age; }
        set { this._age=value; }
    }
}
```

如果我们想取得其中的年龄,可以这样做:

```
var p=new Person();
var age=p.age;
```

现在我们要调用这套系统,而中国说年龄的时候,基本上都是说虚岁,我们应该怎么办呢?

原来的处理方式就是单独写一个处理函数,在年龄上加 1 就可以了,但是,这段代码看着就是不舒服.这就正好用到了 this 扩展方法.我们可以在扩展中添加一个虚岁来满足要求.

```
//计算虚岁
public static int nominalage(this Person p)
{
    return p.age + 1;
}
```

然后就可以下面这样使用了:

```
var p=new Person();
var age=p.nominalage(); //此处是扩展方法,并不是属性
```

上面已经说过,c#的这种 this 扩展方法和 javascript 中的 prototype 方法很类似,但就现在来说,this 扩展方法是在学习 prototype 方法,因为在 javascript 中 prototype 的功能要远比这强大的多,也可以说 prototype 威力更大,也希望广大 Dotnet 程序员多学习一下其他语言,只有对比学习,才能收获更多.

写着写着本文也无啥主题了,怨念呀!!!!

ps:娱乐一下

本人今天早上上班时看到的 **"周杰伦　一路发"**