C# .NET中自带的委托Actoin、Func详解 - 曾元军

时间 2014-06-12 02:06:00  [博客园-所有随笔区](http://www.tuicool.com/sites/QbYvau)

原文  [http://www.cnblogs.com/yzeng/p/3783160.html](http://www.cnblogs.com/yzeng/p/3783160.html?utm_source=tuicool&utm_medium=referral)

主题 [C#](http://www.tuicool.com/topics/11050001)

委托是.NET中强大的功能，而.NET为我们提供了2个非常实用的泛型委托，分别是Action、Func

**【Action】**

Action是.NET内部提供的一个无返回值、泛型的委托，有16重载，分别是0个参数~15个参数；

它们的定义如下：

**delegate** **void** **Action**();

**delegate** **void** Action<**in** T>(T obj);

**delegate** **void** Action<**in** T,**in** T1>(T obj,T1 obj1);

**delegate** **void** Action<**in** T,**in** T1,**in** T2>(T obj,T1 obj1,T2 obj2);

……

可以看到这个委托的方法签名都是无返回值、参数由0~15个，因此可用它来匹配几乎任何无返回值的方法。

例如绑定Button的Click事件：

**this**.button1.Click += **new** Action<object, EventArgs>((sender, e) => {

MessageBox.Show("hellow");

});

事件的方法签名为：void (object EventArgs) 因此以上方法完全没问题。然后里面用了一个lambda表达式传递方法体。

再如，使用线程：

Thread t = **new** Thread(**new** ThreadStart(**new** Action(() => {

//线程代码

})));

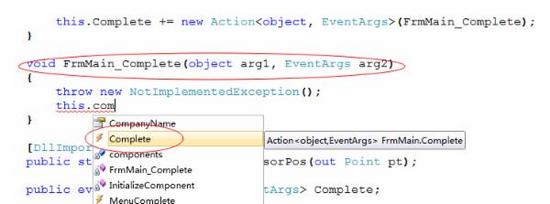
t.Start();

线程ThreadStart需要的方法签名为：void()，因此以上方也完全没问题。

再例如我们要在自己的类中定义事件：

**public** **event** Action<**object**, EventArgs> Complete;

这样就定义一个标准的事件Complete。使用时跟.net中其它控件事件一样：



可以看到完全一样，只是生成的事件方法里面的参数名字不同，这个可以手动修改的。

**【Func】**

Func其实跟Action很像，只是它是带有返回值的委托。也是有15个重载：

**delegate** TR **Func**(**out** TR);

**delegate** TR Func<**in** T,**out** TR>(T obj);

**delegate** TR Func<**in** T,**in** T1,**out** TR>(T obj,T1 obj1);

**delegate** TR Func<**in** T,**in** T1,**in** T2,**out** TR>(T obj,T1 obj1,T2 obj2);

……

可以看到，每个重载的最后一个泛型参数均为返回值类型，可以用它来匹配任何有返回值的方法。

例如获取百度首页的HTML内容然后MessageBox.Show出来：

MessageBox.Show(**new** Func<string>(() => {

System.Net.WebClient wc = **new** System.Net.WebClient();

**return** wc.DownloadString("http://www.baidu.com");

})());

一句话就搞定了，是不是很霸气。

以上代码实例化了一个返回值为string无参数的委托，然后用lambda写方法体再最后加了一个()进行调用，如果是有参数的需要在这里传递参数。

怎么样，.net提供的这两个委托还是很有用的吧，在很多情况下节省了我们自定义委托的时间。如果你对委托不了解，建议你先学学委托~~~