平时我们如果要用到委托一般都是先声明一个委托类型，比如：

private delegate string Say();

string说明适用于这个委托的方法的返回类型是string类型，委托名**Say**后面没有参数，说明对应的方法也就没有传入参数。

写一个适用于该委托的方法：

public static string SayHello()

{

return "Hello";

}

最后调用:

static void Main(string[] args)

{

Say say = SayHello;

Console.WriteLine(say());

}

这里我们先声明委托，然后再将方法传给该委托。有没有办法可以不定义委托变量呢？

答案是肯定的，我们可以用**Func**.

**Func**是.NET里面的内置委托，它有很多重载。

**Func<TResult>**：没有传入参数，返回类型为**TResult**的委托。就像我们上面的Say委托，就可以用**Func<string>**来替代，调用如下：

static void Main(string[] args)

{

Func<string> say = SayHello;

//Say say = SayHello;

Console.WriteLine(say());

}

怎么样，有了**Func**很简单吧。看一下**Func**别的重载。

**Func<T, TResult>** 委托：有一个传入参数**T**，返回类型为**TResult**的委托。如：

//委托 传入参数类型为string,方法返回类型为int

Func<string, int> a = Count;

//对应方法

public int Count(string num)

{

return Convert.ToInt32(num);

}

**Func<T1, T2, TResult>** 委托：有两个传入参数：**T1**与**T2**，返回类型为**TResult**。

类似的还有**Func(T1, T2, T3, TResult)** 委托、**Func(T1, T2, T3, T4, TResult)** 委托等。用法差不多，都是前面为方法的传入参数，最后一个为方法的返回类型。

**Func**也可以与匿名方法一起使用如：

[复制代码](javascript:void(0);)

public static void Main()

{

Func<string, int, string[]> extractMeth = delegate(string s, int i)

{

char[] delimiters = new char[] { ' ' };

return i > 0 ? s.Split(delimiters, i) : s.Split(delimiters);

};

string title = "The Scarlet Letter";

// Use Func instance to call ExtractWords method and display result

foreach (string word in extractMeth(title, 5))

Console.WriteLine(word);

}

[复制代码](javascript:void(0);)

同样它也可以接*lambda* 表达式

[复制代码](javascript:void(0);)

public static void Main()

{

char[] separators = new char[] {' '};

Func<string, int, string[]> extract = (s, i) =>

i > 0 ? s.Split(separators, i) : s.Split(separators) ;

string title = "The Scarlet Letter";

// Use Func instance to call ExtractWords method and display result

foreach (string word in extract(title, 5))

Console.WriteLine(word);

}

[复制代码](javascript:void(0);)

**Func**都是有返回类型的，如果我们的方法没有返回类型该怎么办呢？铛铛铛，这时**Action**就要粉墨登场了。

**Action** 委托：没有传入参数，也没有返回类型，即Void。如：

[复制代码](javascript:void(0);)

static void Main(string[] args)

{

Action say = SayHello;  
 say();

}

public static void SayHello( )

{

Console.WriteLine("Say Hello");

}

[复制代码](javascript:void(0);)

**Action<T>**委托：传入参数为**T**，没有返回类型。如：

[复制代码](javascript:void(0);)

static void Main(string[] args)

{

Action<string> say = SayHello;

say("Hello");

}

public static void SayHello(string word )

{

Console.WriteLine(word);

}

[复制代码](javascript:void(0);)

**Action<T1, T2>** 委托:两个传入参数，分别为**T1**与**T2**，没有返回类型。

**Action**同样的还有许多其它重载，每个重载用法一样，只是方法的传入参数数量不一样。

其实**Action**与**Func**的用法差不多，差别只是一个有返回类型，一个没有返回类型，当然**Action**也可以接匿名方法和*Lambda*表达式。

匿名方法：

[复制代码](javascript:void(0);)

static void Main(string[] args)

{

Action<string> say = delegate(string word)

{

Console.WriteLine(word);

};

say("Hello Word");

}

[复制代码](javascript:void(0);)

*Lambda*表达式：

static void Main(string[] args)

{

Action<string> say = s => Console.WriteLine(s);

say("Hello Word");

}

如果我的文章对你有帮助，就点一下推荐吧.(\*^\_\_^\*)