**匿名函数（C# 编程指南）**

匿名函数是一个“内联”语句或表达式，可在需要委托类型的任何地方使用。可以使用匿名函数来初始化命名委托，或传递命名委托（而不是命名委托类型）作为方法参数。

共有两种匿名函数，以下主题中分别讨论了这些函数：

* [Lambda 表达式（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/bb397687.aspx).
* [匿名方法（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/0yw3tz5k.aspx)

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| Lambda 表达式可以绑定到表达式树，也可以绑定到委托。 |

**C# 中委托的发展**

在 C# 1.0 中，您通过使用在代码中其他位置定义的方法显式初始化委托来创建委托的实例。C# 2.0 引入了匿名方法的概念，作为一种编写可在委托调用中执行的未命名内联语句块的方式。C# 3.0 引入了 Lambda 表达式，这种表达式与匿名方法的概念类似，但更具表现力并且更简练。这两个功能统称为“匿名函数”。通常，针对 .NET Framework 版本 3.5 及更高版本的应用程序应使用 Lambda 表达式。

## 实例1

下面的示例演示了从 C# 1.0 到 C# 3.0 委托创建过程的发展：

class Test

{

delegate void TestDelegate(string s);

static void M(string s)

{

Console.WriteLine(s);

}

static void Main(string[] args)

{

// Original delegate syntax required

// initialization with a named method.

TestDelegate testDelA = new TestDelegate(M);

// C# 2.0: A delegate can be initialized with

// inline code, called an "anonymous method." This

// method takes a string as an input parameter.

TestDelegate testDelB = delegate(string s) { Console.WriteLine(s); };

// C# 3.0\. A delegate can be initialized with

// a lambda expression. The lambda also takes a string

// as an input parameter (x). The type of x is inferred by the compiler.

TestDelegate testDelC = (x) => { Console.WriteLine(x); };

// Invoke the delegates.

testDelA("Hello. My name is M and I write lines.");

testDelB("That's nothing. I'm anonymous and ");

testDelC("I'm a famous author.");

// Keep console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

/\* Output:

Hello. My name is M and I write lines.

That's nothing. I'm anonymous and

I'm a famous author.

Press any key to exit.

\*/

## 实例2

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace csharpbasic\_anonymous\_func  {  internal class Program  {  public delegate void MyDele(string s);  static void PrintStr(string s)  {  Console.WriteLine(s);  }  static void TestDelegate1() //最古老的使用delegate的方法  {  MyDele fun = new MyDele(PrintStr);  fun("Hello,Old Fashion Delegate!!!");  }    static void TestDelegate2() ////笔记老的使用delegate的方法  {  MyDele dele = delegate (string s) { Console.WriteLine(s); };  dele("Newer Method of delegate");  }    static void TestDelegate3()//最新使用delegate的方法,直接传递一个lambda表达式  {  MyDele dele = (string s) => { Console.WriteLine(s); };  dele("Hello ,Lambda delegate");  }  static void Main(string[] args)  {  //TestDelegate1();  //TestDelegate2();  TestDelegate3();  }  }  } |