**命名空间（C# 编程指南）**

使用 C# 编程时，通过两种方式来大量使用命名空间。首先，.NET Framework 使用命名空间来组织它的众多类，如下所示：

System.Console.WriteLine("Hello World!");

**System** 是一个命名空间，**Console** 是该命名空间中的类。可以使用 **using** 关键字，因此不必使用完整的名称，如以下示例所示：

using System;

Console.WriteLine("Hello");

Console.WriteLine("World!");

有关更多信息，请参见 [using 指令（C# 参考）](https://msdn.microsoft.com/zh-cn/library/sf0df423.aspx)。

其次，在较大的编程项目中，声明自己的命名空间可以帮助控制类名称和方法名称的范围。使用 [namespace](https://msdn.microsoft.com/zh-cn/library/z2kcy19k.aspx) 关键字可声明命名空间，如下例所示：

namespace SampleNamespace

{

class SampleClass

{

public void SampleMethod()

{

System.Console.WriteLine(

"SampleMethod inside SampleNamespace");

}

}

}

**命名空间概述**

命名空间具有以下属性：

* 组织大型代码项目。
* 使用 **.** 运算符将它们分隔。
* **using directive** 不必为每个类指定命名空间的名称。
* **global** 命名空间是“根”命名空间：**global::System** 始终引用 .NET Framework 命名空间 **System**。

# 使用命名空间

## 访问命名空间

大多数 C# 应用程序从一个 **using** 指令节开始。该节列出应用程序将会频繁使用的命名空间，避免程序员在每次使用其中包含的方法时都要指定完全限定的名称。

例如，通过在程序开头包括行：

using System;

程序员可以使用代码：

Console.WriteLine("Hello, World!");

而不是：

System.Console.WriteLine("Hello, World!");

## 命名空间别名

[using 指令（C# 参考）](https://msdn.microsoft.com/zh-cn/library/sf0df423.aspx)还可用于创建[命名空间](https://msdn.microsoft.com/zh-cn/library/z2kcy19k.aspx)的别名。例如，如果使用包含嵌套命名空间的以前编写的命名空间，您可能希望声明一个别名来提供引用特定命名空间的简写方法，如以下示例中所示：

using Co = Company.Proj.Nested; // define an alias to represent a namespace

## 使用命名空间来控制范围

**namespace** 关键字用于声明一个范围。在项目中创建范围的能力有助于组织代码，并可让您创建全局唯一的类型。在下面的示例中，名为 SampleClass 的类在两个命名空间中定义，其中一个命名空间嵌套在另一个之内。 [. 运算符（C# 参考）](https://msdn.microsoft.com/zh-cn/library/6zhxzbds.aspx)用于区分所调用的方法。

namespace SampleNamespace

{

class SampleClass

{

public void SampleMethod()

{

System.Console.WriteLine(

"SampleMethod inside SampleNamespace");

}

}

// Create a nested namespace, and define another class.

namespace NestedNamespace

{

class SampleClass

{

public void SampleMethod()

{

System.Console.WriteLine(

"SampleMethod inside NestedNamespace");

}

}

}

class Program

{

static void Main(string[] args)

{

// Displays "SampleMethod inside SampleNamespace."

SampleClass outer = new SampleClass();

outer.SampleMethod();

// Displays "SampleMethod inside SampleNamespace."

SampleNamespace.SampleClass outer2 = new SampleNamespace.SampleClass();

outer2.SampleMethod();

// Displays "SampleMethod inside NestedNamespace."

NestedNamespace.SampleClass inner = new NestedNamespace.SampleClass();

inner.SampleMethod();

}

}

}

## 完全限定名

命名空间和类型的名称必须唯一，由指示逻辑层次结构的完全限定名描述。例如，语句 A.B 表示 A 是命名空间或类型的名称，而 B 则嵌套在其中。

下面的示例中有嵌套的类和命名空间。在每个实体的后面，需要完全限定名作为注释。

namespace N1 // N1

{

class C1 // N1.C1

{

class C2 // N1.C1.C2

{

}

}

namespace N2 // N1.N2

{

class C2 // N1.N2.C2

{

}

}

}

在以上代码段中：

* 命名空间 N1 是全局命名空间的成员。它的完全限定名是 N1。
* 命名空间 N2 是命名空间 N1 的成员。它的完全限定名是 N1.N2。
* 类 C1 是 N1 的成员。它的完全限定名是 N1.C1。
* 在此代码中使用了两次 C2 类名。但是，完全限定名是唯一的。 C2 的第一个实例是在 C1 中声明的；因此，其完全限定名为：N1.C1.C2。 C2 的第二个实例是在命名空间 N2 中声明的；因此，其完全限定名为：N1.N2.C2。

使用以上代码段，可以用以下方法将新的类成员 C3 添加到命名空间 N1.N2 内：

namespace N1.N2

{

class C3 // N1.N2.C3

{

}

}

一般情况下，应使用 **::** 来引用命名空间别名或使用 **global::** 来引用全局命名空间，并使用 **.** 来限定类型或成员。

与引用类型而不是命名空间的别名一起使用 **::** 是错误的。例如：

using Alias = System.Console;

class TestClass

{

static void Main()

{

// Error

//Alias::WriteLine("Hi");

// OK

Alias.WriteLine("Hi");

}

}

记住单词 **global** 不是预定义的别名，因此 global.X 没有任何特殊的含义。仅当与 **::** 一起使用时，它才获得特殊的含义。

定义名为 global 的别名会生成编译器警告 CS0440，因为 **global::** 始终引用全局命名空间而不是别名。例如，下面的行将产生警告：

using global = System.Collections; // Warning

最好将 **::** 与别名一起使用，这样可以避免意外引入其他类型。以下面的代码为例：

using Alias = System;

namespace Library

{

public class C : Alias.Exception { }

}

这样做可行，但是如果接着引入一个名为 Alias 的类型，则 Alias. 将改为绑定到该类型。使用 Alias::Exception 可以确保 Alias 被当作命名空间别名，而不会被误认为类型。

有关 **global** 别名的更多信息，请参见主题[如何：使用全局命名空间别名（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/c3ay4x3d.aspx)。

# 使用全局命名空间别名（C# 编程指南）

当成员可能被同名的其他实体隐藏时，能够访问全局[命名空间](https://msdn.microsoft.com/zh-cn/library/z2kcy19k.aspx)中的成员非常有用。

例如，在下面的代码中，Console 在 [System](https://msdn.microsoft.com/zh-cn/library/system.aspx) 命名空间中解析为 TestApp.Console 而不是 **Console** 类型。

using System;

class TestApp

{

// Define a new class called 'System' to cause problems.

public class System { }

// Define a constant called 'Console' to cause more problems.

const int Console = 7;

const int number = 66;

static void Main()

{

// The following line causes an error. It accesses TestApp.Console,

// which is a constant.

//Console.WriteLine(number);

}

}

由于类 TestApp.System 隐藏了 **System** 命名空间，因此使用 **System.Console** 仍然会导致错误：

// The following line causes an error. It accesses TestApp.System,

// which does not have a Console.WriteLine method.

System.Console.WriteLine(number);

但是，可以通过使用 global::System.Console 避免这一错误，如下所示：

// OK

global::System.Console.WriteLine(number);

当左侧的标识符为 **global** 时，对右侧标识符的搜索将从全局命名空间开始。例如，下面的声明将 TestApp 作为全局空间的一个成员进行引用。

class TestClass : global::TestApp

显然，并不推荐创建自己的名为 System 的命名空间，您不可能遇到出现此情况的任何代码。但是，在较大的项目中，很有可能在一个窗体或其他窗体中出现命名空间重复。在这种情况下，全局命名空间限定符可保证您可以指定根命名空间。

在此示例中，命名空间 **System** 用于包括类 TestClass，因此必须使用 global::System.Console 来引用 **System.Console** 类，该类被 **System** 命名空间隐藏。而且，别名 colAlias 用于引用命名空间 **System.Collections**；因此，将使用此别名而不是命名空间来创建 [System.Collections.Hashtable](https://msdn.microsoft.com/zh-cn/library/system.collections.hashtable.aspx) 的实例。

using colAlias = System.Collections;

namespace System

{

class TestClass

{

static void Main()

{

// Searching the alias:

colAlias::Hashtable test = new colAlias::Hashtable();

// Add items to the table.

test.Add("A", "1");

test.Add("B", "2");

test.Add("C", "3");

foreach (string name in test.Keys)

{

// Searching the global namespace:

global::System.Console.WriteLine(name + " " + test[name]);

}

}

}

}

A 1

B 2

C 3

# 使用 My 命名空间（C# 编程指南）

[Microsoft.VisualBasic.MyServices](https://msdn.microsoft.com/zh-cn/library/microsoft.visualbasic.myservices.aspx) 命名空间（Visual Basic 中的 **My**）提供对许多 .NET Framework 类的简单直观的访问，使您能够编写可与计算机、应用程序、设置、资源等交互的代码。虽然 **MyServices** 命名空间最初是为使用 Visual Basic 而设计的，但它也可以在 C# 应用程序中使用。

有关在 Visual Basic 中使用 **MyServices** 命名空间的更多信息，请参见 [使用 My 开发 (Visual Basic)](https://msdn.microsoft.com/zh-cn/library/5btzf5yk.aspx)。

## 添加引用

在解决方案中使用 **MyServices** 类之前，必须添加一个对 Visual Basic 库的引用。

### 添加对 Visual Basic 库的引用

1. 在**“解决方案资源管理器”**中右击**“引用”**节点，再选择**“添加引用”**。
2. 出现**“引用”**对话框后，向下滚动列表，选择“Microsoft.VisualBasic.dll”。

您可能还需要在程序开头的 **using** 节中包括以下行。

using Microsoft.VisualBasic.Devices;

此示例调用 **MyServices** 命名空间中包含的各种静态方法。要编译此代码，必须在项目中添加一个对 Microsoft.VisualBasic.DLL 的引用。

using System;

using Microsoft.VisualBasic.Devices;

class TestMyServices

{

static void Main()

{

// Play a sound with the Audio class:

Audio myAudio = new Audio();

Console.WriteLine("Playing sound...");

myAudio.Play(@"c:\WINDOWS\Media\chimes.wav");

// Display time information with the Clock class:

Clock myClock = new Clock();

Console.Write("Current day of the week: ");

Console.WriteLine(myClock.LocalTime.DayOfWeek);

Console.Write("Current date and time: ");

Console.WriteLine(myClock.LocalTime);

// Display machine information with the Computer class:

Computer myComputer = new Computer();

Console.WriteLine("Computer name: " + myComputer.Name);

if (myComputer.Network.IsAvailable)

{

Console.WriteLine("Computer is connected to network.");

}

else

{

Console.WriteLine("Computer is not connected to network.");

}

}

}

并不是 **MyServices** 命名空间中的所有的类都可以从 C# 应用程序调用：例如 [FileSystemProxy](https://msdn.microsoft.com/zh-cn/library/microsoft.visualbasic.myservices.filesystemproxy.aspx) 类就不兼容。在这种特定情况下，可以改用作为 [FileSystem](https://msdn.microsoft.com/zh-cn/library/microsoft.visualbasic.fileio.filesystem.aspx)（它也包含在 VisualBasic.dll 中）的一部分的静态方法。例如，下面介绍了如何使用这样的方法来复制目录：

// Duplicate a directory

Microsoft.VisualBasic.FileIO.FileSystem.CopyDirectory(

@"C:\original\_directory",

@"C:\copy\_of\_original\_directory");