C# 接口（ Interface ） C# 预处理器指令

C# 命名空间（ Namespace ）

命名空间的设计目的是提供一种让一组名称与其他名称分隔开的方式。在一个命名空间中声明的类的名称与另一个命名空间中声明的相同的类的名称不冲突。

定义命名空间

命名空间的定义是以关键字 **namespace** 开始，后跟命名空间的名称，如下所示：

```
namespace namespace_name
{
   // 代码声明
}
```

为了调用支持命名空间版本的函数或变量，会把命名空间的名称置于前面，如下所示：

```
namespace_name.item_name;
```

下面的程序演示了命名空间的用法：

实例

```
using System;
namespace first_space
{
   class namespace_cl
   {
      public void func()
      {
         Console.WriteLine("Inside first_space");
      }
   }
}
namespace second_space
{
   class namespace_cl
   {
      public void func()
      {
         Console.WriteLine("Inside second_space");
      }
   }
}
class TestClass
{
   static void Main(string[] args)
   {
      first_space.namespace_cl fc = new first_space.namespace_cl();
      second_space.namespace_cl sc = new second_space.namespace_cl();
```

```
        fc.func();
        sc.func();
        Console.ReadKey();
    }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Inside first_space
Inside second_space
```

using 关键字

using 关键字表明程序使用的是给定命名空间中的名称。例如，我们在程序中使用 **System** 命名空间，其中定义了类 Console。我们可以只写：

```
Console.WriteLine ("Hello there");
```

我们可以写完全限定名称，如下：

```
System.Console.WriteLine("Hello there");
```

您也可以使用 **using** 命名空间指令，这样在使用的时候就不用在前面加上命名空间名称。该指令告诉编译器随后的代码使用了指定命名空间中的名称。下面的代码演示了命名空间的应用。

让我们使用 using 指定重写上面的实例：

实例

```
using System;
using first_space;
using second_space;

namespace first_space
{
   class abc
   {
      public void func()
      {
         Console.WriteLine("Inside first_space");
      }
   }
}
namespace second_space
{
   class efg
   {
      public void func()
      {
```

```
         Console.WriteLine("Inside second_space");
      }
   }
}
class TestClass
{
   static void Main(string[] args)
   {
      abc fc = new abc();
      efg sc = new efg();
      fc.func();
      sc.func();
      Console.ReadKey();
   }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Inside first_space
Inside second_space
```

嵌套命名空间

命名空间可以被嵌套，即您可以在一个命名空间内定义另一个命名空间，如下所示：

```
namespace namespace_name1
{
   // 代码声明
   namespace namespace_name2
   {
     // 代码声明
   }
}
```

您可以使用点（.）运算符访问嵌套的命名空间的成员，如下所示：

实例

```
using System;
using SomeNameSpace;
using SomeNameSpace.Nested;

namespace SomeNameSpace
{
    public class MyClass
    {
        static void Main()
        {
            Console.WriteLine("In SomeNameSpace");
            Nested.NestedNameSpaceClass.SayHello();
```

```
        }
    }

    // 内嵌命名空间
    namespace Nested
    {
        public class NestedNameSpaceClass
        {
            public static void SayHello()
            {
                Console.WriteLine("In Nested");
            }
        }
    }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
In SomeNameSpace
In Nested
```

C# 接口（ Interface ） C# 预处理器指令

1 篇笔记

写笔记

using的用法：

1. using指令：引入命名空间

这是最常见的用法，例如：

```
using System;
using Namespace1.SubNameSpace;
```

2. using static 指令：指定无需指定类型名称即可访问其静态成员的类型

```
using static System.Math;var = PI; // 直接使用System.Math.PI
```

3. 起别名

```
using Project = PC.MyCompany.Project;
```

4. using语句：将实例与代码绑定

```
using (Font font3 = new Font("Arial", 10.0f),
            font4 = new Font("Arial", 10.0f))
{
    // Use font3 and font4.
}
```

代码段结束时，自动调用font3和font4的Dispose方法，释放实例。

袖里藏云 2年前 (2017-07-24)