Why is the double colon(::) operator required to resolve a namespace conflict? [duplicate]

Asked 6 years, 8 months ago Active 6 years, 8 months ago Viewed 10k times

4

2

This question already has an answer here:
What is the purpose of :: in C#? *3 answers*

Please see my below sample program. I have two namespaces containing the same `struct` . To avoid conflict while using in `Main()` , I have given the namespaces aliases. While invoking the `struct` from `Main()` , I am able to invoke directly through namespace alias, like `test.MyStruct` . I have another option also using `::` operator, like `test::MyStruct` .

Why is the `::` operator required, and where should I use it instead of an alias?

```
using System;
using test=counter;
using duplicatecounter;

namespace counter
{
    struct MyStruct
    {

    }
}

namespace duplicatecounter
{
    struct MyStruct
    {

    }
}

class Program
{
    public static void Main()
    {
        test.MyStruct a = new test.MyStruct();
        test::MyStruct a1 = new test::MyStruct();
```

```
    }
}
```

c# .net namespaces

edited Mar 4 '13 at 16:59

Brian
4,749 7 30 43

asked Mar 4 '13 at 16:27

Deepak Raj
346 5 20

marked as duplicate by Bobson, IAbstract, Peter Ritchie, Peter O., mattytommo Mar 5 '13 at 9:05

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

Also, link to documentation. – Bobson Mar 4 '13 at 16:31

well, here the question is more of why to use `namespace::type` operator over `namespace.type` form. I am trying to get the differences here. – Deepak Raj Mar 5 '13 at 2:15

Deepack - It's a reasonable question, but the way it was asked here didn't make that very clear. If you're still wondering, I'd suggest asking a new question which explicitly says either "What are the differences" or "When to use one over the other". I wouldn't suggest bringing up namespace aliases, since they just confuse the matter. – Bobson Mar 5 '13 at 15:20

2 Answers

5

It is mainly needed when someone wrote code without consideration of code being used. I.e. duplicate classes in namespaces that are expected to be used together or hiding namespaces.

MSDN sample shows one case in Use the Global Namespace Alias :

```
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);

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

    }
}
```

answered Mar 4 '13 at 16:36

Alexei Levenkov
88.3k 9 98 144

thanks Alexei for your detailed explanation. – Deepak Raj Mar 5 '13 at 5:32

It is actually not needed for any aliased name space except the global. If you use any other alias, you can use the regular c# dot syntax (i.e. SomeAliasedAssembly.Namespace.Class.etc...) – David Refaeli May 30 '18 at 20:03

3

the `::` operator doing the same like `namespace.` ,but the `::` operator is used to look up identifiers. It is always positioned between two identifiers

example :

```
global::System.Console.WriteLine("Hello World");
```

a good example explained here : http://msdn.microsoft.com/en-us/library/c3ay4x3d.aspx

answered Mar 4 '13 at 16:34

Eslam Hamouda
1,005 9 12

2 great. now I understand `global::` form does not have the alias version like `global.` . Good answer Eslam. Anyway, what does identifiers mean in your example? I am confused with the reasoning. – Deepak Raj Mar 5 '13 at 2:21

3 well, I was able to do this `using global = System; class TestApp { static void Main() { global.Console.WriteLine("test"); } }` but fine, i

got the answer.. – Deepak Raj Mar 5 '13 at 5:30