‹ Previous Next ›

C# - Delegates

What if we want to pass a function as a parameter? How does C# handles the callback functions or event handler? The answer is - delegate.

The delegate is a reference type data type that defines the method signature. You can define variables of delegate, just like other data type, that can refer to any method with the same signature as the delegate.

There are three steps involved while working with delegates:

1. Declare a delegate
2. Set a target method
3. Invoke a delegate

A delegate can be declared using the `delegate` keyword followed by a function signature, as shown below.

Delegate Syntax

```
[access modifier] delegate [return type] [delegate name]([parameters])
```

The following declares a delegate named `MyDelegate`.

Example: Declare a Delegate

```
public delegate void MyDelegate(string msg);
```

Above, we have declared a delegate `MyDelegate` with a void return type and a string parameter. A delegate can be declared outside of the class or inside the class. Practically, it should be declared out of the class.

After declaring a delegate, we need to set the target method or a lambda expression. We can do it by creating an object of the delegate using the new keyword and passing a method whose signature matches the delegate signature.

Example: Set Delegate Target

```
public delegate void MyDelegate(string msg); // declare a delegate

// set target method
MyDelegate del = new MyDelegate(MethodA);
// or
MyDelegate del = MethodA;
// or set lambda expression
MyDelegate del = (string msg) =>  Console.WriteLine(msg);

// target method
static void MethodA(string message)
{
    Console.WriteLine(message);
}
```

You can set the target method by assigning a method directly without creating an object of delegate e.g., `MyDelegate del = MethodA`.

After setting a target method, a delegate can be invoked using the `Invoke()` method or using the `()` operator.

Example: Invoke a Delegate

```
del.Invoke("Hello World!");
// or
del("Hello World!");
```

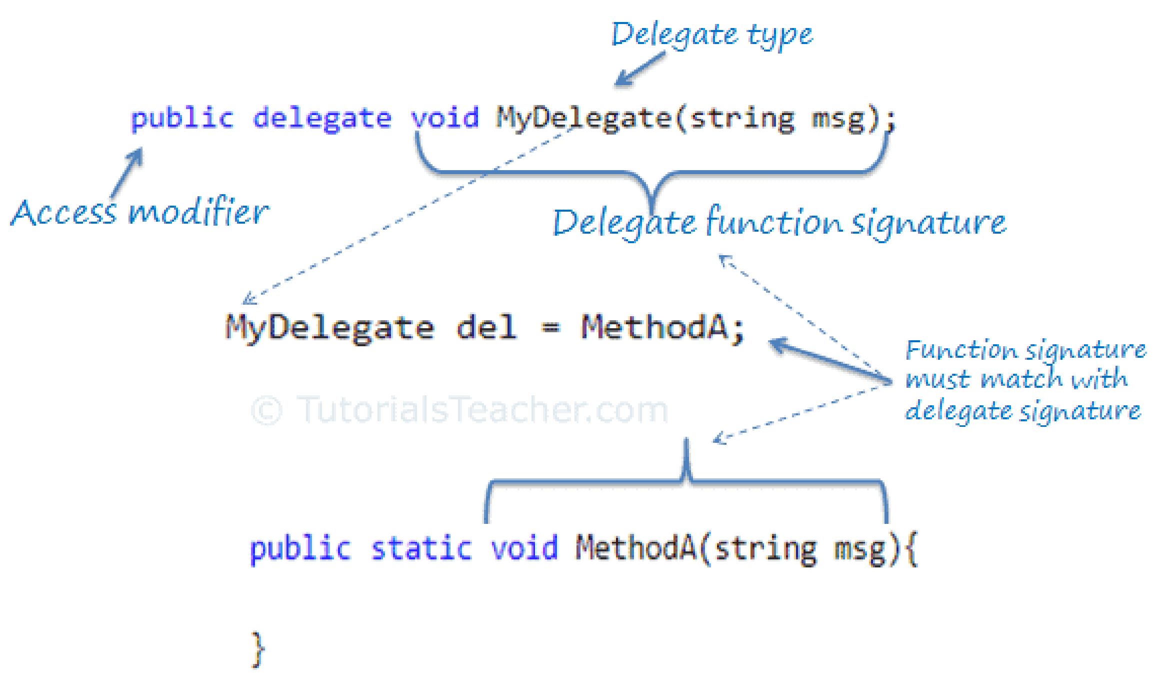
The following is a full example of a delegate.

Example: Delegate

```
public delegate void MyDelegate(string msg); //declaring a delegate

class Program
{
    static void Main(string[] args)
    {
        MyDelegate del = ClassA.MethodA;
        del("Hello World");

        del = ClassB.MethodB;
        del("Hello World");

        del = (string msg) => Console.WriteLine("Called lambda expression: " + msg);
        del("Hello World");
    }
}

class ClassA
{
    static void MethodA(string message)
    {
        Console.WriteLine("Called ClassA.MethodA() with parameter: " + message);
    }
}

class ClassB
{
    static void MethodB(string message)
    {
        Console.WriteLine("Called ClassB.MethodB() with parameter: " + message);
    }
}
```

Try it

The following image illustrates the delegate.


Delegate type
public delegate void MyDelegate(string msg);
Access modifier
Delegate function signature
MyDelegate del = MethodA;
Function signature must match with delegate signature
© TutorialsTeacher.com
public static void MethodA(string msg){
}

C# Delegate

Passing Delegate as a Parameter

A method can have a parameter of the delegate type, as shown below.

Example: Delegate

```
public delegate void MyDelegate(string msg); //declaring a delegate

class Program
{
    static void Main(string[] args)
    {
        MyDelegate del = ClassA.MethodA;
        InvokeDelegate(del);

        del = ClassB.MethodB;
        InvokeDelegate(del);

        del = (string msg) => Console.WriteLine("Called lambda expression: " + msg);
        InvokeDelegate(del);
    }

    static void InvokeDelegate(MyDelegate del) // MyDelegate type parameter
    {
        del("Hello World");
    }
}

class ClassA
{
    static void MethodA(string message)
    {
        Console.WriteLine("Called ClassA.MethodA() with parameter: " + message);
    }
}

class ClassB
{
    static void MethodB(string message)
    {
        Console.WriteLine("Called ClassB.MethodB() with parameter: " + message);
    }
}
```

Try it

Note:

In .NET, Func and Action types are built-in generic delegates that should be used for most common delegates instead of creating new custom delegates.

ADVERTISEMENT

Multicast Delegate

The delegate can point to multiple methods. A delegate that points multiple methods is called a multicast delegate. The "+" or "+=" operator adds a function to the invocation list, and the "-" and "-=" operator removes it.

Example: Multicast Delegate

```
public delegate void MyDelegate(string msg); //declaring a delegate

class Program
{
    static void Main(string[] args)
    {
        MyDelegate del1 = ClassA.MethodA;
        MyDelegate del2 = ClassB.MethodB;

        MyDelegate del = del1 + del2; // combines del1 + del2
        del("Hello World");

        MyDelegate del3 = (string msg) => Console.WriteLine("Called lambda expression: " + msg);
        del += del3; // combines del1 + del2 + del3
        del("Hello World");

        del = del - del2; // removes del2
        del("Hello World");

        del -= del1 // removes del1
        del("Hello World");
    }
}

class ClassA
{
    static void MethodA(string message)
    {
        Console.WriteLine("Called ClassA.MethodA() with parameter: " + message);
    }
}

class ClassB
{
    static void MethodB(string message)
    {
        Console.WriteLine("Called ClassB.MethodB() with parameter: " + message);
    }
}
```

Try it

The addition and subtraction operators always work as part of the assignment: `del1 += del2;` is exactly equivalent to `del1 = del1+del2;` and likewise for subtraction.

If a delegate returns a value, then the last assigned target method's value will be return when a multicast delegate called.

Example: Multicast Delegate Returning a Value

```
public delegate int MyDelegate(); //declaring a delegate

class Program
{
    static void Main(string[] args)
    {
        MyDelegate del1 = ClassA.MethodA;
        MyDelegate del2 = ClassB.MethodB;

        MyDelegate del = del1 + del2;
        Console.WriteLine(del());// returns 200
    }
}

class ClassA
{
    static int MethodA()
    {
        return 100;
    }
}

class ClassB
{
    static int MethodB()
    {
        return 200;
    }
}
```

Try it

Generic Delegate

A generic delegate can be defined the same way as a delegate but using generic type parameters or return type. The generic type must be specified when you set a target method.

For example, consider the following generic delegate that is used for int and string parameters.

Example: Generic Delegate

```
public delegate T add<T>(T param1, T param2); // generic delegate

class Program
{
    static void Main(string[] args)
    {
        add<int> sum = Sum;
        Console.WriteLine(sum(10, 20));

        add<string> con = Concat;
        Console.WriteLine(conct("Hello ","World!!"));
    }
    public static int Sum(int val1, int val2)
    {
        return val1 + val2;
    }

    public static string Concat(string str1, string str2)
    {
        return str1 + str2;
    }
}
```

Try it

Delegate is also used to declare an Event and an Anonymous Method.

Learn more about delegate here.

Points to Remember :

1) Delegate is the reference type data type that defines the signature.
2) Delegate type variable can refer to any method with the same signature as the delegate.
3) Syntax: *[access modifier] delegate [return type] [delegate name]([parameters])*
4) A target method's signature must match with delegate signature.
5) Delegates can be invoke like a normal function or Invoke() method.
6) Multiple methods can be assigned to the delegate using "+" or "+=" operator and removed using "-" or "-=" operator. It is called multicast delegate.
7) If a multicast delegate returns a value then it returns the value from the last assigned target method.
8) Delegate is used to declare an event and anonymous methods in C#.

Learn C# using coding questions with answers and explanations. C# Questions & Answers	Want to check how much you know C#? Start C# Skill Test

Related Articles

- Difference between delegates and events in C#
- Difference between Array and ArrayList
- Difference between Hashtable and Dictionary
- How to write file using StreamWriter in C#?
- How to sort the generic SortedList in the descending order?
- How to read file using StreamReader in C#?
- How to calculate the code execution time in C#?
- Design Principle vs Design Pattern
- How to convert string to int in C#?
- Boxing and Unboxing in C#
- More C# articles

Share Tweet Share Whatsapp

Previous Next

TUTORIALSTEACHER.COM

TutorialsTeacher.com is optimized for learning web technologies step by step. Examples might be simplified to improve reading and basic understanding. While using this site, you agree to have read and accepted our terms of use and privacy policy.

feedback@tutorialsteacher.com

TUTORIALS

- ASP.NET Core
- ASP.NET MVC
- IoC
- Web API
- C#
- LINQ
- Entity Framework
- AngularJS 1
- Node.js
- D3.js
- JavaScript
- jQuery
- Sass
- Https

E-MAIL LIST

Subscribe to TutorialsTeacher email list and get latest updates, tips & tricks on C#, .Net, JavaScript, jQuery, AngularJS, Node.js to your inbox.

Email address GO

We respect your privacy.

HOME PRIVACY POLICY TERMS OF USE ADVERTISE WITH US