

[articles](#)[Q&A](#)[forums](#)[lounge](#)

Delegates, Multicast delegates and Events in C# - Notes

**Shivprasad koirala**, 1 Dec 2015

CPOL

Rate this:

4.89 (47 votes)

In this article we will try to understand Delegates, Multicast delegates and Events in C#

[Introduction](#)[Answering in short](#)[Delegate is a pointer to a function](#)[So what's the big deal of indirect calling?](#)[No, Do not even think this](#)[Implementing Callbacks with delegates](#)[Multicast delegates](#)[Events: - The encapsulated delegate](#)[Summarizing](#)

Introduction

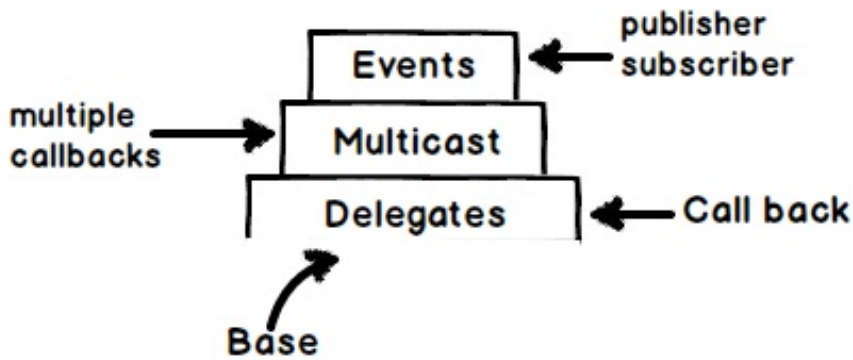
I do understand there are lot of articles on delegates and events and i am sure that more 1000 articles will come in coming times. But still lot of developers are confused about :-

- What is a delegate and where to use it?
- What are multicast delegates?
- When to use Delegate VS Event

So in this article we will try to attempt to answer the above three queries.

Answering in short

Below is a simple diagram which explains how delegates, multicast delegates and events are connected.



- Delegates are pointer to functions and used for call back.
- Multicast delegates help to invoke multiple callbacks.
- Events encapsulate delegate and implement publisher and subscriber model.
- Events and Multicast are types of delegates. So delegate is the base for events and multicast.

In the rest of the article we will try to understand the above statements in more detail.

Delegate is a pointer to a function

"Delegate is a pointer to a function" and you can invoke the pointed function via the delegate.

Creation of delegate is a three step process:-

1. Declare the delegate: - In this step you will use the delegate keyword but you will need to ensure that the signature of this delegate has to be same as the method. For example in the below code "SomeMethod()" is returning void and not taking any input parameters. So delegate "SomeMethodPtr()" is defined accordingly.
2. Create object of delegate: - Once you have defined the delegate you need to create object to use it. This is shown in the code as step 2.
3. Invoke the delegate: - To invoke the delegate you need to call "Invoke" method to call "SomeMethod".

Hide Copy Code

```

delegate void SomeMethodPtr(); // 1. Declare delegate
static void Main(string[] args)
{
    SomeMethodPtr ptrObject = SomeMethod; // 2. Create object of delegate
    ptrObject.Invoke(); // 3 . invoke the delegate
}
static void SomeMethod()
{
    // some code
}
  
```

So what's the big deal of indirect calling?

Ok so now that we have understood that delegate is a pointer to a function, but what is so great about it. How does it benefit to invoke a method indirectly ,when we can invoke it directly and make our life better?.

Just think is not the below code more simple as compared to invoking indirectly via delegate.

Hide Copy Code

```

static void Main(string[] args)
{
    SomeMethod()
}
  
```

```
static void SomeMethod()
{
    // some code
}
```

The first point to understand is importance of delegates is seen when you are invoking code which is in a different library. For example below is a simple file search library which searches files in a directory.

Please note at this moment there is NO code for file search in reality, think that the for loop is a search and the file name is coming in string "str".

[Hide](#) [Copy Code](#)

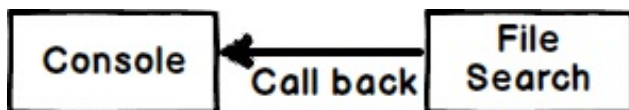
```
public class SearchFile
{
    public void Search()
    {
        // File search is happening
        for(int i=0;i<100;i++)
        {
            string str = "File " + i;
        }
    }
}
```

Now the above component is consumed in a console application as shown below. Now we want that as soon as the file is searched we want the UI to be notified immediately with the file name.

[Hide](#) [Copy Code](#)

```
static void Main(string[] args)
{
    SearchFile f1 = new SearchFile();
    f1.Search();
}
```

In other words we need a CALLBACK from the filesearch component back to the console application. And that's what exactly delegate is meant for CALLBACK , CALLBACK and CALLBACK.



No, Do not even think this

One of the things as a developer which you would think is why cannot we just write "Console.Write" in the search method itself as shown below. But by doing this your "SearchFile" is tightly coupled with the UI technology. What if we need to use the below code in a "Winform" or "Wpf" application ?. The below code will crash.

[Hide](#) [Copy Code](#)

```
public class SearchFile
{
    public void Search()
    {
        // File search is happening
        for(int i=0;i<100;i++)
        {
            string str = "File " + i;
            console.writeline(str);
        }
    }
}
```

```
    }
}
```

Implementing Callbacks with delegates

Now in order to implement delegate the first thing is the file search component should provide the necessary infrastructure by exposing a delegate. Below is the "SearchFile" class through which we have exposed a delegate "wheretocall". Look for step 1 and step 2 in the below code. This delegate "wheretocall" is empty at this moment and its saying that any client who is expecting callbacks from me please pass your method pointer here.

Inside the for loop the delegate is passing data of the current file name through the delegate.

[Hide](#) [Copy Code](#)

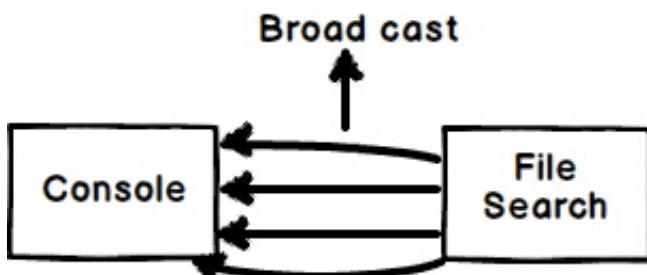
```
public class SearchFile
{
    public delegate void Wheretocall(string status); // Step 1
    public Wheretocall wheretocall = null; // Step 2
    public void Search()
    {
        // File search is happening
        for (int i = 0; i < 100; i++)
        {
            string str = "File " + i;
            wheretocall(str); // Step 3
        }
    }
}
```

Now the client who is consuming the above class needs to pass the method reference to the delegate pointer. See Step 1 in the below code. In this place we are passing our method where we expect "SearchFile" class to call back and send information.

[Hide](#) [Copy Code](#)

```
static void Main(string[] args)
{
    SearchFile f1 = new SearchFile();
    f1.wheretocall = CallHere; // Step 1
    f1.Search();
}
static void CallHere(string message)
{
    Console.WriteLine(message);
}
```

Multicast delegates



Now think about a situation where we want the "SearchFile" class to send notifications(broadcast) to three methods in one go.. We can also term this as publisher subscriber architecture. Where the "SearchFile" is publisher and all the other methods are

subscribers attached to the publisher.

[Hide](#) [Copy Code](#)

```
static void CallHereToWriteToFile(string message)
{
    System.IO.File.WriteAllText(@"c:\sometext.txt", message);
}
static void CallHereForConsole(string message)
{
    Console.WriteLine(message);
}
static void CallHereToWriteInternally(string message)
{
    messages.Add(message);
}
```

In order to achieve the same we do not have to change a single line of code in the "SearchFile" class. In the client side we just need to use "+" sign to assign the methods to the "wheretocall" delegate.

If you want to unsubscribe you can use "-" sign.

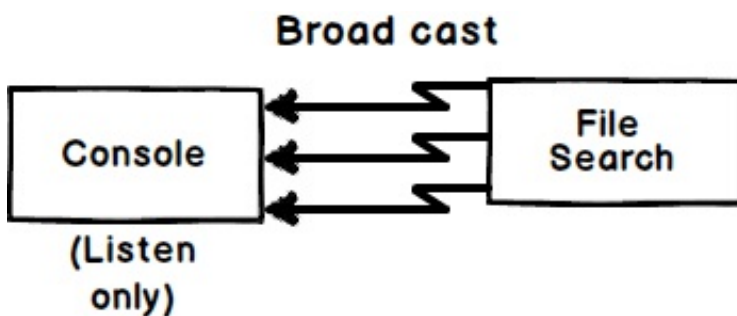
[Hide](#) [Copy Code](#)

```
static void Main(string[] args)
{
    SearchFile f1 = new SearchFile();
    f1.wheretocall += CallHereForConsole;
    f1.wheretocall += CallHereToWriteInternally;
    f1.wheretocall += CallHereToWriteToFile;
    f1.Search();
}
```

Now when the "SearchFile" class invokes the delegate all the three methods attached with the delegate gets invoked.

Events: - The encapsulated delegate

The publisher subscriber architecture created by multicast delegate has one serious problem that subscribers CAN MODIFY THE DELEGATE. Now in a true publisher subscriber architecture or broadcasting architecture the subscribers can only subscribe and unsubscribe. In other words they can only listen but not modify the delegate.



For example in the below code you can see the subscriber code is making the publisher delegate reference NULL and also he can invoke the delegate. In other word the client can modify the delegate. If you visualize in real world a television works on publisher subscriber architecture. As an end user we can subscribe or unsubscribe from a TV channel but we do not have authority to slap a TV anchor if we do not like any show.

[Hide](#) [Copy Code](#)

```
SearchFile f1 = new SearchFile();
f1.wheretocall += CallHereForConsole;
f1.wheretocall.invoke(); // The client can invoke.
```

```
f1.wheretocall = null; // The delegate can be modified.
f1.wheretocall += CallHereToWriteInternally;
f1.Search();
```

That's where events come in to picture. Events encapsulate delegates. On the publisher side we need to create the instance of delegate using the "event" keyword.

[Hide](#) [Copy Code](#)

```
public class SearchFile
{
    public delegate void WhereToCall(string status);
    public event WhereToCall wheretocall = null;
    // Remaining code removed for clarity.
}
```

If we do that the client can no more modify the delegate. You can see the below error highlighted on the subscriber side that you can use only "+" or "-" but not "=".

If we translate the below error to simple English it says that you can only subscribe (+) or unsubscribe (-) from a delegate but not modify them. So by using event we have achieved a proper publisher subscriber model.

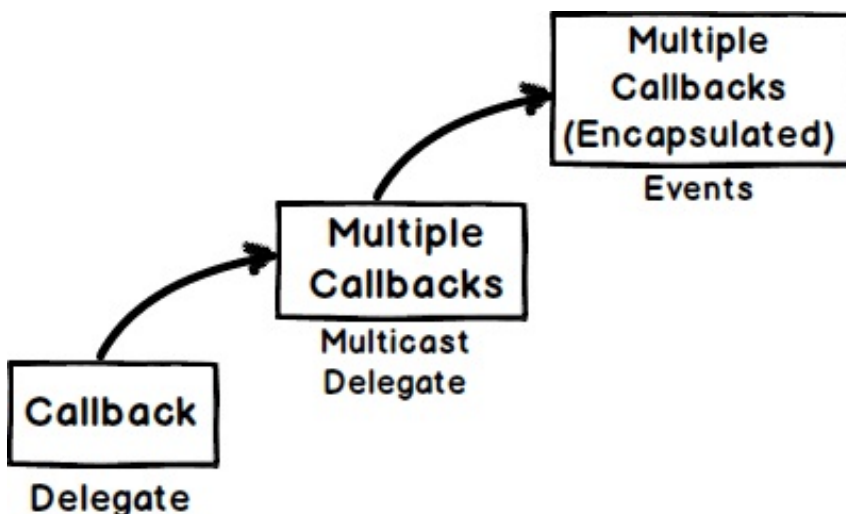
```
SearchFile f1 = new SearchFile();
f1.wheretocall += CallHereForConsole;
f1.wheretocall = null; // The delegate
f1.
f1.
```

SearchFile.WhereToCall SearchFile.wheretocall

Error:
The event 'FileSearch.SearchFile.wheretocall' can only appear on the left hand side of += or -=

Summarizing

So if we summarize delegate is a call back , multicast delegate is for multiple call backs while events do multicast callback but client has no control on the delegate. In case of event clients can only subscribe to the delegate.



Below is a simple you tube video which shows the importance of delegates and callbacks in a practical manner , worth a watch.



License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share



About the Author



Shivprasad koirala

Architect <http://www.questpond.com>

India 

Do not forget to watch my Learn step by step video series.

[Learn MVC in 16 hours](#)

[Learn design pattern in 8 hours](#)

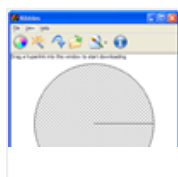
[Learn C# and .NET in 60 days](#)

[Learn MSBI in 32 hours](#)

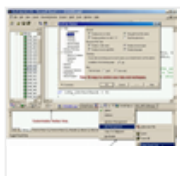
You may also be interested in...



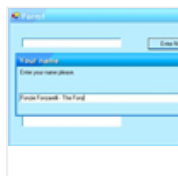
SAPrefs - Netscape-like Preferences Dialog



WTL for MFC Programmers, Part IX - GDI Classes, Common Dialogs, and Utility Classes



Window Tabs (WndTabs) Add-In for DevStudio



Multi-line InputBox control - MC++



WPF: If Carlsberg did MVVM Frameworks: Part 3 of n



OLE DB - First steps

Comments and Discussions

You must **Sign In** to use this message board.

Search Comments

Go

First Prev Next

My vote of 5 📌

Raul Iloc 13-Jan-16 1:57

very nice 📌

BillW33 12-Jan-16 10:51

My vote of 5 📌

maq_rohit 11-Jan-16 10:21

My vote of 5 📌

Santhakumar M 7-Jan-16 21:41

Top notch! 📌

jediYL 3-Jan-16 17:24

No, a delegate instance is not a pointer to a function 📌

Sergey Alexandrovich Kryukov 4-Dec-15 16:26

Re: No, a delegate instance is not a pointer to a function 📌

Shivprasad koirala 4-Dec-15 17:28

In my article... 

Sergey Alexandrovich Kryukov 4-Dec-15 19:48

Re: No, a delegate instance is not a pointer to a function 

Vijay Gill 7-Dec-15 3:25

Visual Studio C# compiler 

jrobb229 4-Dec-15 12:21

Re: Visual Studio C# compiler 

Shivprasad koirala 4-Dec-15 15:40

You really shouldn't post articles if you don't speak the language. 

exyyle 4-Dec-15 10:37

Re: You really should post articles if you don't speak the language. 

Vijay Gill 4-Dec-15 11:00

Re: You really should post articles if you don't speak the language. 

jrobb229 4-Dec-15 12:04

Re: You really should post articles if you don't speak the language. 

Shivprasad koirala 4-Dec-15 15:45

Re: You really shouldn't post articles if you don't speak the language. 

Shivprasad koirala 4-Dec-15 15:39

You are on codeproject and not english teaching site 

Member 12191515 7-Dec-15 5:37

Re: You are on codeproject and not english teaching site 

exyyle 10-Dec-15 7:41

My first post 

Member 12133353 3-Dec-15 22:28

Great explanation 

Marc Murdoch 3-Dec-15 9:12

static delegate 

Member 11451717 2-Dec-15 21:40

Re: static delegate 

Shivprasad koirala 2-Dec-15 21:52











Re: static delegate 

Sergey Alexandrovich Kryukov 4-Dec-15 16:28

Good Article Once Again 


aarif moh shaikh 2-Dec-15 20:14

My vote of 5 **Mahsa Hassankashi 2-Dec-15 3:01**[Refresh](#)**1**

 General  News  Suggestion  Question  Bug  Answer  Joke  Praise  Rant  Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | Mobile
Web01 | 2.8.160115.1 | Last Updated 2 Dec 2015

 Select Language | ▼
Layout: [fixed](#) | [fluid](#)

Article Copyright 2015 by **Shivprasad koirala**
Everything else Copyright © [CodeProject](#), 1999-2016