.NET Framework Class Library

**Action(T) Delegate**

Updated: June 2010

Encapsulates a method that has a single parameter and does not return a value.

**Namespace:** [System](http://msdn.microsoft.com/en-us/library/system.aspx)  
**Assembly:** mscorlib (in mscorlib.dll)

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifSyntax

Visual Basic

Public Delegate Sub Action(Of In T) ( \_

obj As T \_

)

C#

public delegate void Action<in T>(

T obj

)

Visual C++

generic<typename T>

public delegate void Action(

T obj

)

F#

type Action =

delegate of

obj:'T -> unit

**Type Parameters**

**in** **In** **in** **in** *T*

The type of the parameter of the method that this delegate encapsulates.

This type parameter is contravariant. That is, you can use either the type you specified or any type that is less derived. For more information about covariance and contravariance, see [Covariance and Contravariance in Generics](http://msdn.microsoft.com/en-us/library/dd799517.aspx).

**Parameters**

*obj*

Type: **T**  
The parameter of the method that this delegate encapsulates.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifRemarks

You can use the **Action<(Of <(T>)>)** delegate to pass a method as a parameter without explicitly declaring a custom delegate. The encapsulated method must correspond to the method signature that is defined by this delegate. This means that the encapsulated method must have one parameter that is passed to it by value, and it must not return a value. (In C#, the method must return **void**. In Visual Basic, it must be defined by the **Sub**…**End Sub** construct. It can also be a method that returns a value that is ignored.) Typically, such a method is used to perform an operation.

|  |
| --- |
| **Description: NoteNote** |
| To reference a method that has one parameter and returns a value, use the generic [Func<(Of <(T, TResult>)>)](http://msdn.microsoft.com/en-us/library/bb549151.aspx) delegate instead. |

When you use the **Action<(Of <(T>)>)** delegate, you do not have to explicitly define a delegate that encapsulates a method with a single parameter. For example, the following code explicitly declares a delegate named DisplayMessage and assigns a reference to either the [WriteLine](http://msdn.microsoft.com/en-us/library/system.console.writeline.aspx) method or the ShowWindowsMessage method to its delegate instance.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl07_code');" \o "Copy Code)

Delegate Sub DisplayMessage(message As String)

Module TestCustomDelegate

Public Sub Main

Dim messageTarget As DisplayMessage

If Environment.GetCommandLineArgs().Length > 1 Then

messageTarget = AddressOf ShowWindowsMessage

Else

messageTarget = AddressOf Console.WriteLine

End If

messageTarget("Hello, World!")

End Sub

Private Sub ShowWindowsMessage(message As String)

MsgBox(message)

End Sub

End Module

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl08_code');" \o "Copy Code)

using System;

using System.Windows.Forms;

delegate void DisplayMessage(string message);

public class TestCustomDelegate

{

public static void Main()

{

DisplayMessage messageTarget;

if (Environment.GetCommandLineArgs().Length > 1)

messageTarget = ShowWindowsMessage;

else

messageTarget = Console.WriteLine;

messageTarget("Hello, World!");

}

private static void ShowWindowsMessage(string message)

{

MessageBox.Show(message);

}

}

Visual C++

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl09_code');" \o "Copy Code)

#using <System.Windows.Forms.dll>

using namespace System;

using namespace System::Windows::Forms;

public delegate void DisplayMessage(String^ message);

public ref class TestCustomDelegate

{

public:

static void ShowWindowsMessage(String^ message)

{

MessageBox::Show(message);

}

};

int main()

{

DisplayMessage^ messageTarget;

if (Environment::GetCommandLineArgs()->Length > 1)

messageTarget = gcnew DisplayMessage(&TestCustomDelegate::ShowWindowsMessage);

else

messageTarget = gcnew DisplayMessage(&Console::WriteLine);

messageTarget(L"Hello World!");

return 0;

}

The following example simplifies this code by instantiating the **Action<(Of <(T>)>)** delegate instead of explicitly defining a new delegate and assigning a named method to it.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl12_code');" \o "Copy Code)

Module TestAction1

Public Sub Main

Dim messageTarget As Action(Of String)

If Environment.GetCommandLineArgs().Length > 1 Then

messageTarget = AddressOf ShowWindowsMessage

Else

messageTarget = AddressOf Console.WriteLine

End If

messageTarget("Hello, World!")

End Sub

Private Sub ShowWindowsMessage(message As String)

MsgBox(message)

End Sub

End Module

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl13_code');" \o "Copy Code)

using System;

using System.Windows.Forms;

public class TestAction1

{

public static void Main()

{

Action<string> messageTarget;

if (Environment.GetCommandLineArgs().Length > 1)

messageTarget = ShowWindowsMessage;

else

messageTarget = Console.WriteLine;

messageTarget("Hello, World!");

}

private static void ShowWindowsMessage(string message)

{

MessageBox.Show(message);

}

}

Visual C++

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl14_code');" \o "Copy Code)

#using <System.Windows.Forms.dll>

using namespace System;

using namespace System::Windows::Forms;

namespace ActionExample

{

public ref class Message

{

public:

static void ShowWindowsMessage(String^ message)

{

MessageBox::Show(message);

}

};

}

int main()

{

Action<String^>^ messageTarget;

if (Environment::GetCommandLineArgs()->Length > 1)

messageTarget = gcnew Action<String^>(&ActionExample::Message::ShowWindowsMessage);

else

messageTarget = gcnew Action<String^>(&Console::WriteLine);

messageTarget("Hello, World!");

return 0;

}

You can also use the **Action<(Of <(T>)>)** delegate with anonymous methods in C#, as the following example illustrates. (For an introduction to anonymous methods, see [Anonymous Methods (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/0yw3tz5k.aspx).)

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl18_code');" \o "Copy Code)

using System;

using System.Windows.Forms;

public class TestAnonMethod

{

public static void Main()

{

Action<string> messageTarget;

if (Environment.GetCommandLineArgs().Length > 1)

messageTarget = delegate(string s) { ShowWindowsMessage(s); };

else

messageTarget = delegate(string s) { Console.WriteLine(s); };

messageTarget("Hello, World!");

}

private static void ShowWindowsMessage(string message)

{

MessageBox.Show(message);

}

}

You can also assign a lambda expression to an **Action<(Of <(T>)>)** delegate instance, as the following example illustrates. (For an introduction to lambda expressions, see [Lambda Expressions (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/bb397687.aspx).)

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl22_code');" \o "Copy Code)

Imports System.Windows.Forms

Public Module TestLambdaExpression

Public Sub Main()

Dim messageTarget As Action(Of String)

If Environment.GetCommandLineArgs().Length > 1 Then

messageTarget = Sub(s) ShowWindowsMessage(s)

Else

messageTarget = Sub(s) ShowConsoleMessage(s)

End If

messageTarget("Hello, World!")

End Sub

Private Function ShowWindowsMessage(message As String) As Integer

Return MessageBox.Show(message)

End Function

Private Function ShowConsoleMessage(message As String) As Integer

Console.WriteLine(message)

Return 0

End Function

End Module

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl81_ctl00_ctl23_code');" \o "Copy Code)

using System;

using System.Windows.Forms;

public class TestLambdaExpression

{

public static void Main()

{

Action<string> messageTarget;

if (Environment.GetCommandLineArgs().Length > 1)

messageTarget = s => ShowWindowsMessage(s);

else

messageTarget = s => Console.WriteLine(s);

messageTarget("Hello, World!");

}

private static void ShowWindowsMessage(string message)

{

MessageBox.Show(message);

}

}

The [ForEach](http://msdn.microsoft.com/en-us/library/bwabdf9z.aspx) and [ForEach<(Of <(T>)>)](http://msdn.microsoft.com/en-us/library/zecdkyw2.aspx) methods each take an **Action<(Of <(T>)>)** delegate as a parameter. The method encapsulated by the delegate allows you to perform an action on each element in the array or list. The example uses the [ForEach](http://msdn.microsoft.com/en-us/library/bwabdf9z.aspx) method to provide an illustration.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExamples

The following example demonstrates the use of the **Action<(Of <(T>)>)** delegate to print the contents of a [List<(Of <(T>)>)](http://msdn.microsoft.com/en-us/library/6sh2ey19.aspx) object. In this example, the Print method is used to display the contents of the list to the console. In addition, the C# example also demonstrates the use of anonymous methods to display the contents to the console. Note that the example does not explicitly declare an **Action<(Of <(T>)>)** variable. Instead, it passes a reference to a method that takes a single parameter and that does not return a value to the [List<(Of <(T>)>)..::.ForEach](http://msdn.microsoft.com/en-us/library/bwabdf9z.aspx) method, whose single parameter is an **Action<(Of <(T>)>)** delegate. Similarly, in the C# example, an **Action<(Of <(T>)>)** delegate is not explicitly instantiated because the signature of the anonymous method matches the signature of the **Action<(Of <(T>)>)** delegate that is expected by the [List<(Of <(T>)>)..::.ForEach](http://msdn.microsoft.com/en-us/library/bwabdf9z.aspx) method.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl82_ctl00_ctl13_code');" \o "Copy Code)

Imports System

Imports System.Collections.Generic

Class Program

Shared Sub Main()

Dim names As New List(Of String)

names.Add("Bruce")

names.Add("Alfred")

names.Add("Tim")

names.Add("Richard")

' Display the contents of the list using the Print method.

names.ForEach(AddressOf Print)

End Sub

Shared Sub Print(ByVal s As String)

Console.WriteLine(s)

End Sub

End Class

' This code will produce output similar to the following:

' Bruce

' Alfred

' Tim

' Richard

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl82_ctl00_ctl14_code');" \o "Copy Code)

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

List<String> names = new List<String>();

names.Add("Bruce");

names.Add("Alfred");

names.Add("Tim");

names.Add("Richard");

// Display the contents of the list using the Print method.

names.ForEach(Print);

// The following demonstrates the anonymous method feature of C#

// to display the contents of the list to the console.

names.ForEach(delegate(String name)

{

Console.WriteLine(name);

});

}

private static void Print(string s)

{

Console.WriteLine(s);

}

}

/\* This code will produce output similar to the following:

\* Bruce

\* Alfred

\* Tim

\* Richard

\* Bruce

\* Alfred

\* Tim

\* Richard

\*/

Description: http://i.msdn.microsoft.com/Global/Images/clear.gif