**发布符合 .NET Framework 准则的事件（C# 编程指南）**

下面的过程演示了如何将符合标准 .NET Framework 模式的事件添加到您的类和结构中。.NET Framework 类库中的所有事件均基于 [EventHandler](https://msdn.microsoft.com/zh-cn/library/system.eventhandler.aspx) 委托，定义如下：

public delegate void EventHandler(object sender, EventArgs e);

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| .NET Framework 2.0 引入了此委托的一个泛型版本，即 [EventHandler<TEventArgs>](https://msdn.microsoft.com/zh-cn/library/db0etb8x.aspx)。下面的示例显示如何使用这两种版本。 |

虽然您定义的类中的事件可基于任何有效委托类型（甚至是可返回值的委托），但是，通常建议您使用 [EventHandler](https://msdn.microsoft.com/zh-cn/library/system.eventhandler.aspx) 让事件基于 .NET Framework 模式，如下面的示例所示。

**采用 EventHandler 模式发布事件**

1. （如果不需要与事件一起发送自定义数据，请跳过此步骤，进入步骤 3a。）在发行者类和订阅方类均可看见的范围中声明自定义数据的类。然后添加保留您的自定义事件数据所需的成员。在此示例中，会返回一个简单字符串。

public class CustomEventArgs : EventArgs

{

public CustomEventArgs(string s)

{

msg = s;

}

private string msg;

public string Message

{

get { return msg; }

}

}

1. （如果您使用的是 [EventHandler<TEventArgs>](https://msdn.microsoft.com/zh-cn/library/db0etb8x.aspx) 的泛型版本，请跳过此步骤。）在发布类中声明一个委托。为它指定以 EventHandler 结尾的名称。第二个参数指定自定义 EventArgs 类型。
2. public delegate void CustomEventHandler(object sender, CustomEventArgs a);
3. 使用以下任一步骤，在发布类中声明事件。
   1. 如果没有自定义 EventArgs 类，事件类型就是非泛型 EventHandler 委托。无需声明委托，因为它已在创建 C# 项目时包含的 [System](https://msdn.microsoft.com/zh-cn/library/system.aspx) 命名空间中进行了声明。将以下代码添加到发行者类中。
   2. public event EventHandler RaiseCustomEvent;
   3. 如果使用的是 [EventHandler](https://msdn.microsoft.com/zh-cn/library/system.eventhandler.aspx) 的非泛型版本，并且您有一个由 [EventArgs](https://msdn.microsoft.com/zh-cn/library/system.eventargs.aspx) 派生的自定义类，请在发布类中声明您的事件，并且将来自步骤 2 的委托用作类型。
   4. public event CustomEventHandler RaiseCustomEvent;
   5. 如果使用的是泛型版本，则不需要自定义委托。相反，在发行者类中，您应将事件类型指定为 EventHandler<CustomEventArgs>，将尖括号中的内容替换为自己的类的名称。
   6. public event EventHandler<CustomEventArgs> RaiseCustomEvent;

下面的示例通过将自定义 EventArgs 类和 [EventHandler<TEventArgs>](https://msdn.microsoft.com/zh-cn/library/db0etb8x.aspx) 用作事件类型来演示上述步骤。

namespace DotNetEvents

{

using System;

using System.Collections.Generic;

// Define a class to hold custom event info

public class CustomEventArgs : EventArgs

{

public CustomEventArgs(string s)

{

message = s;

}

private string message;

public string Message

{

get { return message; }

set { message = value; }

}

}

// Class that publishes an event

class Publisher

{

// Declare the event using EventHandler<T>

public event EventHandler<CustomEventArgs> RaiseCustomEvent;

public void DoSomething()

{

// Write some code that does something useful here

// then raise the event. You can also raise an event

// before you execute a block of code.

OnRaiseCustomEvent(new CustomEventArgs("Did something"));

}

// Wrap event invocations inside a protected virtual method

// to allow derived classes to override the event invocation behavior

protected virtual void OnRaiseCustomEvent(CustomEventArgs e)

{

// Make a temporary copy of the event to avoid possibility of

// a race condition if the last subscriber unsubscribes

// immediately after the null check and before the event is raised.

EventHandler<CustomEventArgs> handler = RaiseCustomEvent;

// Event will be null if there are no subscribers

if (handler != null)

{

// Format the string to send inside the CustomEventArgs parameter

e.Message += String.Format(" at {0}", DateTime.Now.ToString());

// Use the () operator to raise the event.

handler(this, e);

}

}

}

//Class that subscribes to an event

class Subscriber

{

private string id;

public Subscriber(string ID, Publisher pub)

{

id = ID;

// Subscribe to the event using C# 2.0 syntax

pub.RaiseCustomEvent += HandleCustomEvent;

}

// Define what actions to take when the event is raised.

void HandleCustomEvent(object sender, CustomEventArgs e)

{

Console.WriteLine(id + " received this message: {0}", e.Message);

}

}

class Program

{

static void Main(string[] args)

{

Publisher pub = new Publisher();

Subscriber sub1 = new Subscriber("sub1", pub);

Subscriber sub2 = new Subscriber("sub2", pub);

// Call the method that raises the event.

pub.DoSomething();

// Keep the console window open

Console.WriteLine("Press Enter to close this window.");

Console.ReadLine();

}

}

}