**泛型委托（C# 编程指南）**

[委托](https://msdn.microsoft.com/zh-cn/library/900fyy8e.aspx) 可以定义自己的类型参数。引用泛型委托的代码可以指定类型参数以创建已关闭的构造类型，就像实例化泛型类或调用泛型方法一样，如下例所示：

public delegate void Del<T>(T item);

public static void Notify(int i) { }

Del<int> m1 = new Del<int>(Notify);

C# 2.0 版具有称为方法组转换的新功能，此功能适用于具体委托类型和泛型委托类型，并使您可以使用如下简化的语法写入上一行：

Del<int> m2 = Notify;

在泛型类内部定义的委托使用泛型类类型参数的方式可以与类方法所使用的方式相同。

class Stack<T>

{

T[] items;

int index;

public delegate void StackDelegate(T[] items);

}

引用委托的代码必须指定包含类的类型变量，如下所示：

private static void DoWork(float[] items) { }

public static void TestStack()

{

Stack<float> s = new Stack<float>();

Stack<float>.StackDelegate d = DoWork;

}

根据典型设计模式定义事件时，泛型委托尤其有用，因为发送方参数可以为强类型，不再需要强制转换成 [Object](https://msdn.microsoft.com/zh-cn/library/system.object.aspx)，或反向强制转换。

delegate void StackEventHandler<T, U>(T sender, U eventArgs);

class Stack<T>

{

public class StackEventArgs : System.EventArgs { }

public event StackEventHandler<Stack<T>, StackEventArgs> stackEvent;

protected virtual void OnStackChanged(StackEventArgs a)

{

stackEvent(this, a);

}

}

class SampleClass

{

public void HandleStackChange<T>(Stack<T> stack, Stack<T>.StackEventArgs args) { }

}

public static void Test()

{

Stack<double> s = new Stack<double>();

SampleClass o = new SampleClass();

s.stackEvent += o.HandleStackChange;

}

**请参阅**

[System.Collections.Generic](https://msdn.microsoft.com/zh-cn/library/system.collections.generic.aspx)

[C# 编程指南](https://msdn.microsoft.com/zh-cn/library/67ef8sbd.aspx)

[泛型介绍（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/0x6a29h6.aspx)

[泛型方法（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/twcad0zb.aspx)

[泛型类（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/sz6zd40f.aspx)

[泛型接口（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/kwtft8ak.aspx)

[委托（C# 编程指南）](https://msdn.microsoft.com/zh-cn/library/ms173171.aspx)

[.NET Framework 中的泛型](https://msdn.microsoft.com/zh-cn/library/ms172192.aspx)