

# 1 System.Func<-T1,-T2,-T3,+TResult> 2 Delegate

```
3 [ILAsm]  
4 .class public sealed System.Func`4<-T1,-T2,-T3,+TResult> extends  
5 System.MulticastDelegate  
  
6 [C#]  
7 public delegate TResult Func<in T1,in T2,in T3,out TResult>(T1 arg1, T2  
8 arg2, T3 arg3);
```

## 9 Assembly Info:

- 10 • *Name:* mscorlib
- 11 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 12 • *Version:* 4.0.0.0
- 13 • *Attributes:*
  - 14 ○ CLSCompliantAttribute(true)

## 15 Summary

16 Encapsulates a method that has three parameters and returns a value of the type  
17 specified by the *TResult* parameter.

## 18 Parameters

Parameter	Description
<i>arg1</i>	The first parameter of the method that this delegate encapsulates.
<i>arg2</i>	The second parameter of the method that this delegate encapsulates.
<i>arg3</i>	The third parameter of the method that this delegate encapsulates.

## 19 20 Inherits From: System.MulticastDelegate

21  
22 **Library:** BCL

## 23 24 Returns

25  
26 The return value of the method that this delegate encapsulates.

## 27 28 Description

1 You can use this delegate to represent a method that can be passed as a parameter  
2 without explicitly declaring a custom delegate. The encapsulated method must  
3 correspond to the method signature that is defined by this delegate. This means that the  
4 encapsulated method must have three parameters, each of which is passed to it by  
5 value, and that it must return a value.

6  
7 *[Note:* To reference a method that has three parameters and returns `void`, use the  
8 generic `System.Action`3<T1, T2, T3>` delegate instead.

9  
10 ]

11  
12 When you use the `System.Func`4<T1, T2, T3, TResult>` delegate, you do not have to  
13 explicitly define a delegate that encapsulates a method with three parameters.

14  
15 You can use the `System.Func`4<T1, T2, T3, TResult>` delegate with anonymous  
16 methods in C#. (For an introduction to anonymous methods, see the C# standard.)

17