

1 System.Func<+TResult> Delegate

```
2 [ILAsm]  
3 .class public sealed System.Func`1<+TResult> extends  
4 System.MulticastDelegate  
  
5 [C#]  
6 public delegate TResult Func<out TResult>();
```

7 Assembly Info:

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

13 Summary

14 Encapsulates a method that has no parameters and returns a value of the type specified
15 by the *TResult* parameter.

16 Inherits From: System.MulticastDelegate

17

18 **Library:** BCL

19

20 Returns

21

22 The return value of the method that this delegate encapsulates.

23

24 Description

25 You can use this delegate to represent a method that can be passed as a parameter
26 without explicitly declaring a custom delegate. The encapsulated method must
27 correspond to the method signature that is defined by this delegate. This means that the
28 encapsulated method must have no parameters and must return a value.

29

30 [*Note:* To reference a method that has no parameters and returns `void`, use the
31 `System.Action` delegate instead.

32

33]

34

35 When you use the `System.Func`1<TResult>` delegate, you do not have to explicitly
36 define a delegate that encapsulates a parameterless method.

37

38 You can use the `System.Func`1<TResult>` delegate with anonymous methods in C#.
39 (For an introduction to anonymous methods, see the C# standard.)

40

41 If you have an expensive computation that you want to execute only if the result is
42 actually needed, you can assign the expensive function to a `System.Func`1<T>`

1 delegate. The execution of the function can then be delayed until a property that
2 accesses the value is used in an expression.

3