

System.Func<+TResult> Delegate

```
[ILAsm]
.class public sealed System.Func`1<+TResult> extends
System.MulticastDelegate

[C#]
public delegate TResult Func<out TResult>();
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 4.0.0.0
- *Attributes:*
 - CLSCompliantAttribute(true)

Summary

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

Inherits From: System.MulticastDelegate

Library: BCL

Returns

The return value of the method that this delegate encapsulates.

Description

You can use this delegate to represent a method that can be passed 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 no parameters and must return a value.

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

]

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

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

If you have an expensive computation that you want to execute only if the result is 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