WIKIPEDIA

Delegate (CLI)

A **delegate** is a form of type-safe function pointer used by the Common Language Infrastructure (CLI). Delegates specify a method to call and optionally an object to call the method on. Delegates are used, among other things, to implement callbacks and event listeners. A delegate object encapsulates a reference to a method. The delegate object can then be passed to code that can call the **referenced** method, without having to know at compile time which method will be invoked.

A **multicast delegate** is a delegate that points to several methods.[1][2] Multicast delegation is a mechanism that provides functionality to execute more than one method. There is a list of delegates maintained internally, and when the multicast delegate is invoked, the list of delegates is executed.

In C#, delegates are often used to implement callbacks in event driven programming. For example, a delegate may be used to indicate which method should be called when the user clicks on some button. Delegates allow the programmer to notify several methods that an event has occurred.[3]

Contents

C# code example

Code to declare a `delegate` type, named `SendMessageDelegate`, which takes a `Message` as a parameter and returns `void`:

```
delegate void SendMessageDelegate(Message message);
```

Code to define a method that takes an instantiated delegate as its argument:

```
void SendMessage(SendMessageDelegate sendMessageDelegateReference)
{
    // Call the delegate and any other chained delegates synchronously.
    sendMessageDelegateReference(new Message("hello this is a sample message"));
}
```

The implemented method that runs when the delegate is called:

```
void HandleSendMessage(Message message)
{
    // The implementation for the Sender and Message classes are not relevant to this example.
    Sender.Send(message);
}
```

Code to call the SendMessage method, passing an instantiated delegate as an argument:

```
SendMessage(new SendMessageDelegate(HandleSendMessage));
```

Delegates (C#)

```
delegate void Notifier(string sender);  // Normal method signature with the keyword delegate

Notifier greetMe;                       // Delegate variable

void HowAreYou(string sender) {
    Console.WriteLine("How are you, " + sender + '?');
}

greetMe = new Notifier(HowAreYou);
```

A delegate variable calls the associated method and is called as follows:

```
greetMe("Anton");                       // Calls HowAreYou("Anton") and prints "How are you, Anton?"
```

Delegate variables are first-class objects of the form `new DelegateType(obj.Method)` and can be assigned to any matching method, or to the value `null`. They store a method and its receiver without any parameters:[4]

```
new DelegateType(funnyObj.HowAreYou);
```

The object `funnyObj` can be `this` and omitted. If the method is `static`, it should not be the object (also called an instance in other languages), but the class itself. It should not be `abstract`, but could be `new`, `override` or `virtual`.

To call a method with a delegate successfully, the method signature has to match the `DelegateType` with the same number of parameters of the same kind (`ref`, `out`, `value`) with the same type (including return type).

Multicast delegates (C#)

A delegate variable can hold multiple values at the same time:

```
void HowAreYou(string sender) {
    Console.WriteLine("How are you, " + sender + '?');
}

void HowAreYouToday(string sender) {
    Console.WriteLine("How are you today, " + sender + '?');
}

Notifier greetMe;

greetMe = HowAreYou;
greetMe += HowAreYouToday;

greetMe("Leonardo");                    // "How are you, Leonardo?"
                                        // "How are you today, Leonardo?"

greetMe -= HowAreYou;

greetMe("Pereira");                     // "How are you today, Pereira?"
```

If the multicast delegate is a function or has no `out` parameter, the parameter of the last call is returned.[5]

Technical implementation details

Although internal implementations may vary, delegate instances can be thought of as a tuple of an object and a method pointer and a reference (possibly null) to another delegate. Hence a reference to one delegate is possibly a reference to multiple delegates. When the first delegate has finished, if its chain reference is not null, the next will be invoked, and so on until the list is complete. This pattern allows an event to have overhead scaling easily from that of a single reference up to dispatch to a list of delegates, and is widely used in the CLI.

Performance

Performance of delegates used to be much slower than a virtual or interface method call (6 to 8 times slower in Microsoft's 2003 benchmarks),[6] but, since the .NET 2.0 CLR in 2005, it is about the same as interface calls.[7] This means there is a small added overhead compared to direct method invocations.

There are very stringent rules on the construction of delegate classes. These rules permit optimizing compilers a great deal of leeway when optimizing delegates while ensuring type safety.

See also

- Continuation
- Delegation pattern
- Delegation (programming)
- Hooking

References

1. Microsoft Developer Network (MSDN) Article (http://msdn.microsoft.com/en-us/library/ms173175(VS.80).aspx), *How to: Combine Delegates (Multicast Delegates)(C# Programming Guide)*, Accessed 5/20/2008
2. "About Microsoft's "Delegates"" (https://web.archive.org/web/19990210053040/http://www.java.sun.com/docs/white/delegates.html). *Sun Developer Network*. Sun Microsystems. Archived from the original (http://java.sun.com/docs/white/delegates.html#Multicast) on 10 February 1999.
3. Wikibooks:C Sharp Programming/Delegates and Events
4. Mössenböck, Hanspeter (2002-03-25). "Advanced C#: Variable Number of Parameters" (http://ssw.jku.at/Teaching/Lectures/CSharp/Tutorial/Part2.pdf) (PDF). http://ssw.jku.at/Teaching/Lectures/CSharp/Tutorial/: Institut für Systemsoftware, Johannes Kepler Universität Linz, Fachbereich Informatik. pp. 23–24. Retrieved 2011-08-04.
5. Mössenböck, Hanspeter (2002-03-25). "Advanced C#: Variable Number of Parameters" (http://ssw.jku.at/Teaching/Lectures/CSharp/Tutorial/). Institut für Systemsoftware, Johannes Kepler Universität Linz, Fachbereich Informatik. p. 25. Retrieved 2011-08-04.
6. Gray, Jan (June 2003). "Writing Faster Managed Code: Know What Things Cost" (http://msdn2.microsoft.com/en-us/library/ms973852). Microsoft. Retrieved 2007-09-09.
7. Sturm, Oliver (2005-09-01). "Delegate calls vastly sped up in .NET 2" (http://www.sturmnet.org/blog/2005/09/01/delegate-interface-speed). Retrieved 2007-09-09.

External links

- MSDN documentation for Delegates (http://msdn2.microsoft.com/en-us/library/system.delegate.aspx)
- Sun's White Paper on Delegates (https://web.archive.org/web/20120627043929/http://java.sun.com/docs/white/delegates.html)
- Microsoft answer to Sun (http://msdn.microsoft.com/en-us/vjsharp/bb188664.aspx)