

Re: What is 'managed code'?

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2005-09/msg00375.html>

- *From:* "Michael D. Ober" <obermd.@.alum.mit.edu.nospam>
 - *Date:* Sat, 03 Sep 2005 19:02:44 GMT
-

Very simply, managed code is code that the runtime system handles all aspects of memory and resource management. The managed code model is that of infinite resources. But since we know that resources on any given system are finite, the runtime must do a lot of work to present this model. To do this requires that the runtime system present a garbage collector that can sweep through the application memory and find all unreferenced resources, be they memory, IO Ports, files, etc., and properly close and release them back to the OS. In the case of memory, the garbage collector must also compact available memory to consolidate free space for further allocations.

VB 5&6 almost do this for memory only, and then only if you don't use direct Windows API calls. When using ActiveX components, which are all COM Objects, the Component Object Module runtime DLLs perform a simple reference counting. When an ActiveX component's references go to zero, the COM runtime releases the memory of the COM Object. Note that the COM runtimes don't close files or other resources – they belong to the process and the OS will close them when the process exits. In VB 5&6, memory references are also reference counted, which means that it is possible to create two objects that reference each other, yet are impossible to reference from the application. The VB 5&6 Garbage Collector will never free these objects. Nor can the VB 5&6 Garbage Collector consolidate free memory space. Neither of these weaknesses of a reference counted memory scheme are usually a problem.

Now to the kicker – VB 5&6 are a very sophisticated implementation of a COM object, to the extent that each form is a COM Object. This is why you need to be careful when referencing controls on a form during your Form_Unload event procedures. Doing so creates a new reference to the form, which prevents the form from being reclaimed by the COM runtime.

The major weakness of fully managed code is that it is significantly slower than unmanaged code. The TAPI API is a near real-time API. Real-time and Managed cannot be used in the same sentence. Other common Windows APIs that fall into this category include the Multimedia API and DirectX. VB 5&6 are partially managed and as such, tend to lie between fully managed code and unmanaged code in performance. Runtime resource management of any type incurs a performance penalty, regardless of language.

Re: What is 'managed code'?

To those of you who think this is a .NET question – it's not. This is a question of language design and implementation. In the Microsoft world, .NET is fully managed, FoxPro and VB 5&6 are partially managed, and C/C++ (including the ATL and MFC extensions) is unmanaged. JAVA, APL, Smalltalk, and Lisp are also fully managed. You can tell unmanaged languages by the presence of explicit pointers to memory. Managed code generally doesn't support explicit pointers except where required for call backs and event handlers.

Mike Ober.

"Jack" <replyto@newsgroup> wrote in message
news:uLY4xBEsFHA.3720@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

> From MSN:

> This article discusses why Telephony Application Programming Interface
> (TAPI) is not supported from managed code.

>

> What is that managed code?

>

>

>

.

• *Follow-Ups:*

- ◆ [*Re: What is 'managed code'?*](#)

 - ◇ From: Jack

- ◆ [*Re: What is 'managed code'?*](#)

 - ◇ From: David

• *References:*

- ◆ [*What is 'managed code'?*](#)

 - ◇ From: Jack

- Prev by Date: [*Re: Why dll Form disappears immediately?*](#)

- Next by Date: [*Re: Why dll Form disappears immediately?*](#)

- Previous by thread: [*Re: What is 'managed code'?*](#)

- Next by thread: [*Re: What is 'managed code'?*](#)

- Index(es):

 - ◆ [*Date*](#)

 - ◆ [*Thread*](#)