.NET und die COM Interop Threading Apartments

Quicktip bei BonnToCode
20. März 2007

Martin Krieger
www.kriegermartin.de

Microsoft CERTIFIED Technology Specialist

Warum dieser Quicktip?

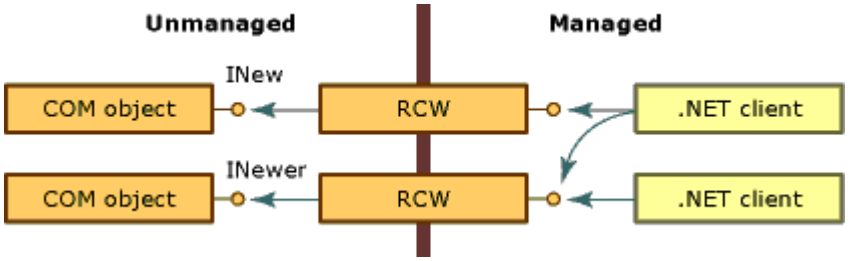
- Es läßt sich unglücklicherweise nicht immer vermeiden, auf vorhandene COM (DCOM, COM+, OLE, ActiveX) Objekte zurückgreifen zu müssen.
- Die Nutzung von COM Objekten in .NET Programmen ist eigentlich ganz einfach. Eigentlich...
- Nichtberücksichtigung der COM Nebenläufigkeitsmodelle kann zu schlechter Performanz, Speicherverlust, Fehlfunktionen, Hängern und Abstürzen führen.
- Die Dokumentation ist eher versteckt in der MSDN und diversen Blogs. Daher soll hier ein Überblick mit Hinweisen auf weiterführende Texte gegeben werden.

Themenüberblick

- COM Interop
  - Early Binding
  - Late Binding
- COM Nebenläufigkeitsmodell
  - Apartments
  - Apartments und Prozesse
  - Objekt Modell
  - Thread Model setzen
  - Nachrichten Pumpen
- Fazit
- Literatur

COM Interop – Early Binding

- Early Binding (statisch - IUnknown)
  - Type Library (.tlb, normalerweise in COM dll eingebunden) erforderlich. Es wird eine Runtime Callable Wrapper Assembly erzeugt, die als .NET Stellvertreter dient.
    - Visual Studio: Verweis hinzufügen, COM.
    - Kommandozeile: TblImp.exe
    - Programm: System.Runtime.InteropServices.TypeLibConverter
    - Händisch: System.Runtime.InteropServices.ComImportAttribute etc.



- Garbage Collector verwaltet auch COM Objekte.
  - Deterministisches Löschen durch System.Runtime.InteropServices.Marshal.FinalReleaseComObject

COM Interop – Late Binding

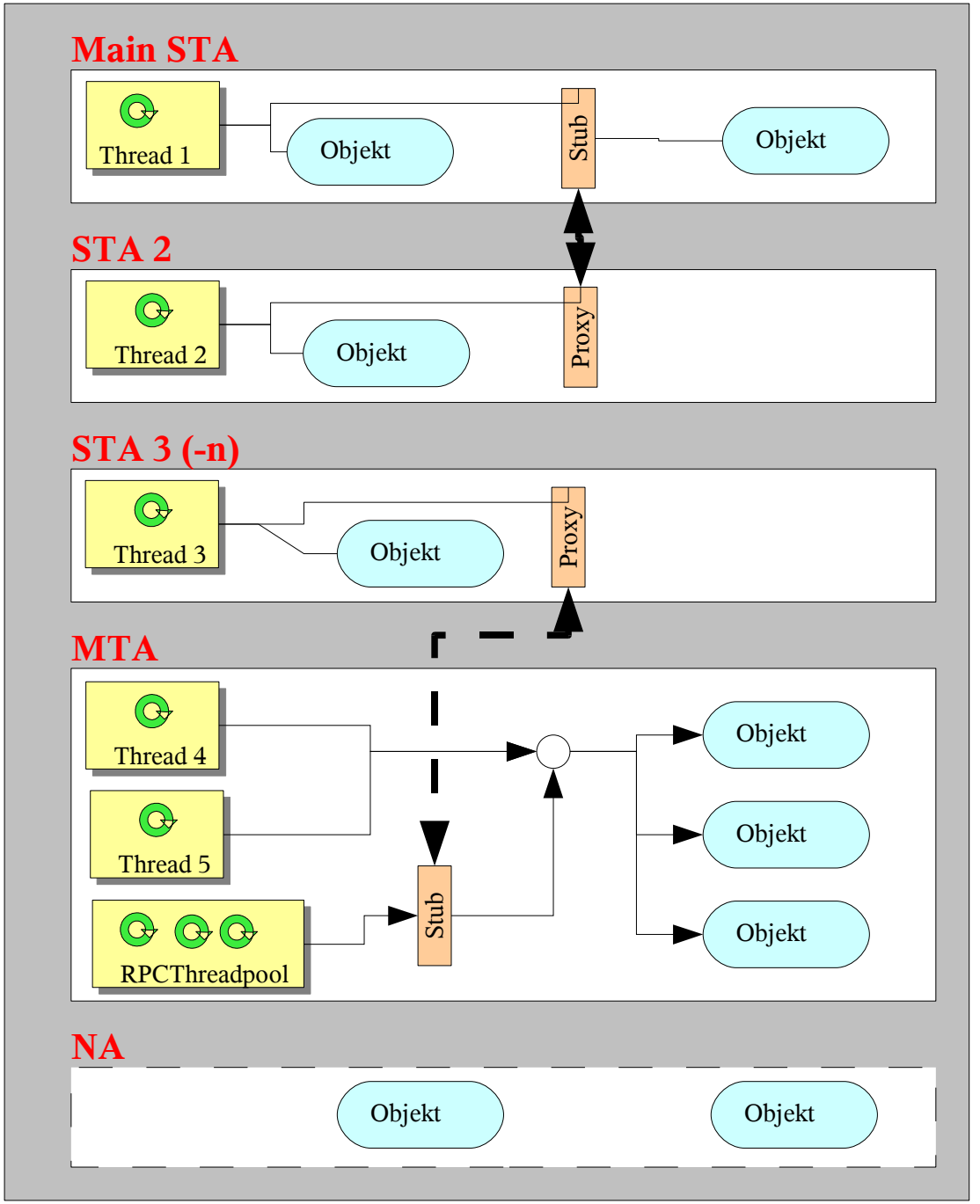
- Late Binding (dynamisch - IDispatch)
  - VB .NET:
    CreateObject(ProgId, ServerName)
    Direkter Methodenaufruf
    Erfordert Option Strict Off
  - Andere Sprachen:
    System.Type.GetTypeFromProgId
    System.Activator.CreateInstance
    System.Type.InvokeMember etc.

Apartments

- Apartments sind logische Gruppen von Threads und COM Objekten.
- Jeder Thread gehört zu genau einem Apartment.
- Jedes COM Objekt gehört zu genau einem Apartment.
- .NET Objekte gehören zu keinem Apartment. Die Apartments der Threads sind für .NET Objekte ohne Belang.
- Threads können auf Objekte ihres Apartments (sowie auf Objekte im Neutral Apartment) frei zugreifen. Der Zugriff auf Objekte in fremden Apartments erfordert einen Threadwechsel (Marshaling).
- Es gibt drei Sorten von Apartments: Single Threaded Apartments (STA) und Multi Threaded Apartments (MTA) sowie seit Windows 2000 Neutral Apartment (NA)

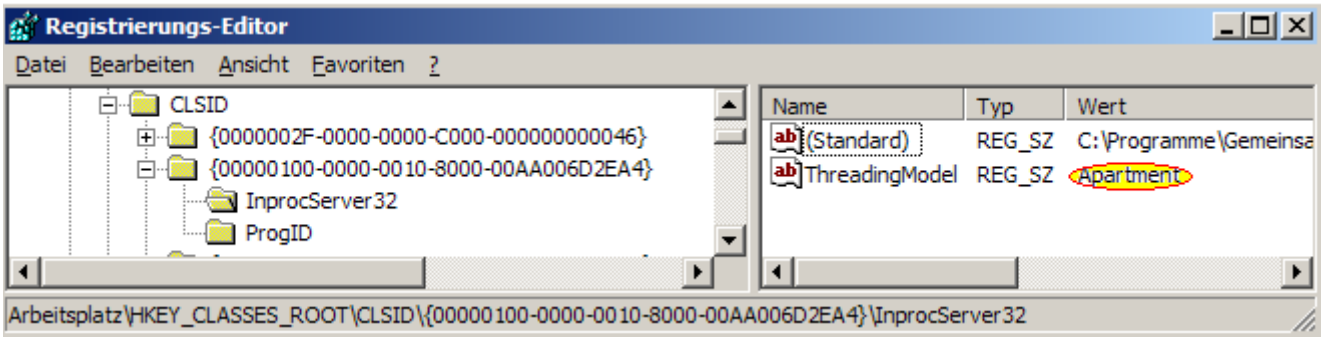
Apartments und Prozesse

- Prozesse haben beliebig viele STA aber höchstens ein MTA
- Zugriffe auf andere Apartments erforder Serialisierung - Deseria-lisierung (Marshaling) und einen Threadwechsel – ebenso wie prozessübergreifende Zugriffe
- STA haben genau einen Thread
  - Objekte in STA können somit nicht parallel von mehreren Threads ausgeführt werden
  - Aufrufe aus anderen Apartments werden in eine Warteschlange eingereiht
- MTA haben mehrere Threads
  - Objekte sind daher selbst für die Synchronisierung verantwortlich
  - Aufrufe aus anderen Apartments werden über einen Threadpool ausgeführt
- NA Objekte können aus jedem Apartment ohne Threadwechsel aufgerufen werden



COM Objekt Apartment Model

- Der Apartmenttyp eines COM Objektes wird durch den Registryeintrag `ThreadingModel` unter `HKEY_CLASSES_ROOT\CLSID\{guid}\InprocServer32` festgelegt



- Regedit, OleView, Tool auf meiner Homepage

	(kein)	Apartment	Free	Both	Neutral
STA	Main STA	Aktuelles STA	MTA	Aktuelles STA	NA
MTA	Main STA	STA, ggf. neu	MTA	MTA	NA

Thread Apartment Model setzen

	STA	MTA

Startthread eines Programms	`[STAThread]`
	`static void Main()`
		`[MTAThread]`
		`static void Main()`
		(default)

Threadpool Thread	*nicht möglich* *(außer durch eigene Threadpool Klasse)*	*(default)*
Eigener Thread	`Thread.SetApartmentState(ApartmentState.STA)`	`Thread.SetApartmentState(ApartmentState.MTA)` *(default)*

.aspx	`<%@ Page AspCompat="true" %>`	*(default)*
.asmx	*eigener http handler (siehe MSDN Mag 10/06)*	*(default)*

Hör' nicht auf zu Pumpen...

- Threadwechsel in STA erfolgen über die Nachrichtenwarteschlange eines dem Apartment zugeordneten, unsichtbaren, Fensters.
  - Die Nachrichten müssen gepumpt werden
  - .NET pumpt u.a. bei
    - Anfordern eines COM-Interfaces
    - Monitor.Enter, WaitHandle.WaitOne, WaitHandle.WaitAny
    - Thread.Join(int)
    - GC.WaitForPendingFinalizers
  - Der Finalizer ist ein MTA Thread. Fehlendes Pumpen kann das Entfernen der COM Objekte blockieren (und somit zu Speicherverlust führen)
- Beim Pumpen werden in der Nachrichtenwarteschlange befindlichen Aufrufe ausgeführt – was u.U. zu Reentranzproblemen führen kann
- Debugging
  - ContextSwitchDeadlock MDA
  - sos.dll: FinalizeQueue -detail

Fazit

- Beim Umgang mit COM Objekten in .NET:
  - Geeignetes Threading Modell wählen.
  - Zwischen deterministischem und undeterministischem (GC) Löschen der Objekte wählen.
  - Nachrichten pumpen.

Literatur

- Prosise, Jeff: Understanding COM Apartments
  www.codeguru.com/cpp/com-tech/activex/apts/article.php/c5529/ (Teil 1)
  www.codeguru.com/cpp/com-tech/activex/apts/article.php/c5533/ (Teil 2)
- Prosise, Jeff: Running ASMX Web Services on STA Threads
  msdn.microsoft.com/msdnmag/issues/06/10/WickedCode/default.aspx
- Brumme, Chris: Apartments and Pumping in the CLR
  blogs.msdn.com/cbrumme/archive/2004/02/02/66219.aspx
- Duffy, Joe: Pump me baby one more time (and break my invariants)
  www.bluebytesoftware.com/blog/PermaLink,guid,1ca19b0e-ef5b-4efc-b614-48d8c913efb9.aspx
- MSDN Library

- Fragen?
- Fragen!