本文讨论在.NET中使用进程内COM组件时的公寓模型，以一个示例直观演示STAThread和MTAThread的作用和区别。

# **1. COM中的公寓**

## **1.1 基本规则**

公寓是COM组件的运行环境，日常生活中公寓是用来住人的，COM中的公寓是用来住COM组件的对象的，每个COM对象必须且只能位于一个公寓中：单线程公寓(STA)或多线程公寓(MTA)。

每个进程可以有0或多个STA。

每个进程可以有0或1个MTA。

一个线程只能关联到一个公寓。因此所有关联到MTA的线程都是关联到进程唯一的一个MTA。

本线程访问与本线程关联的STA中的COM对象不需要列集，直接访问。

其他线程对STA中的COM对象的访问需要列集(marshal)，**通过列集，自动实现了多线程访问下的同步**。

所有线程对MTA中的COM对象的访问不需要列集，**直接访问，需要COM组件自身实现多线程下的同步**。

（列集就是将函数调用序列化，实现跨边界调用，在Windows中通常是通过消息机制实现。在COM中RPC就是列集，在WinForm中Control.Invoke就是一种列集，Remoting也是列集，WCF也是列集，最近流行的RESTfull也是。。。）

## **1.2 公寓类型匹配**

一个COM对象所属的公寓，由两个地方的配置确定：**组件公寓模型**和**客户端线程公寓模型**。

1. 组件公寓模型是在组件注册到注册表时设定，通过组件公寓模型，组件声明自己可以住在什么样的公寓里。可选项包括：Apartment,Free和Both。Apartment，我只能住在单线程公寓中；Free，我只能住在多线程公寓中；Both，我随意，单线程公寓或多线程公寓都可以。
2. 客户端线程公寓模型就是线程的公寓模型，表示当前线程提供什么样的公寓。可选项包括：单线程公寓(STA)或多线程公寓(MTA)，也就是本文所讨论的STAThread和MTAThread。

下表列出了组件对象最终会住在什么公寓中的组合表：

|  |  |  |  |
| --- | --- | --- | --- |
| **客户端线程公寓模型 \ 组件公寓模型** | **Apartment** | **Free** | **Both** |
| **STA** | STA | MTA | STA |
| **MTA** | STA | MTA | MTA |

如果组件公寓模型为Apartment，不管客户端线程公寓模型是什么，组件最后都住在STA中，因为组件说了“我只能住在单线程公寓中”。如果当前线程是MTA，COM库会后台创建一个STA来放该组件的对象。

如果组件公寓模型为Free，不管客户端线程公寓模型是什么，组件最后都住在MTA中，因为组件说了“我只能住在多线程公寓中”。如果当前线程是STA，COM库会检查当前进程的MTA有没有创建，没有就创建进程的MTA，然后将组件的对象放在MTA中。

如果组件公寓模型为Both，组件最后都住在与当前线程关联的公寓中，如果当前线程是STA，它就住在STA中；当前线程是MTA，它就住在MTA中。本文中，我们会创建一个并注册一个Both类型的组件，然后分别在STA和MTA中创建该组件的对象。

## **1.3 .NET中设置客户端线程公寓模型**

在.NET中使用COM组件时，需要设置线程的公寓模型。

在.NET中可以通过STAThread和MTAThread属性来设置主线程的公寓类型， 通过Thread.SetApartmentState可以设置工作线程的公寓类型。

对于WinForm或WPF应用程序，主线程的公寓模型必须为STA，因为用户界面对象都不是线程安全的。

对于控制台应用程序，主线程的公寓模型可以随意设置，为了方便，我们用控制台应用程序来演示。（用WinForm也完全可以演示，只是需要在工作线程中创建组件的对象。）

# **2. 一个简单的COM组件**

为了演示单线程公寓和多线程公寓的区别，我们用ATL实现定义一个简单的COM组件SimpleCom，该组件包含一个返回字符串的方法Hello，返回的字符串分三步合成，每步之间通过Consume方法来消耗较长CPU周期，确保Hello不会在操作系统的一个时间片内被执行完成，保证Hello函数被并发执行，以达到演示的效果。代码如下：

[IMG_256](https://www.cnblogs.com/ioexception/p/javascript:void(0);)

1 // CSimpleCom.h 2 class ATL\_NO\_VTABLE CSimpleCom : 3 public CComObjectRootEx<CComSingleThreadModel>, 4 public CComCoClass<CSimpleCom, &CLSID\_SimpleCom>, 5 public IConnectionPointContainerImpl<CSimpleCom>, 6 public CProxy\_ISimpleComEvents<CSimpleCom>, 7 public IDispatchImpl<ISimpleCom, &IID\_ISimpleCom, &LIBID\_ATLTestLib, /\*wMajor =\*/ 1, /\*wMinor =\*/ 0> 8 { 9 public:10 CSimpleCom()11 {12 this->m\_iMember = 0;13 }14 15 DECLARE\_REGISTRY\_RESOURCEID(IDR\_SIMPLECOM)16 17 18 BEGIN\_COM\_MAP(CSimpleCom)19 COM\_INTERFACE\_ENTRY(ISimpleCom)20 COM\_INTERFACE\_ENTRY(IDispatch)21 COM\_INTERFACE\_ENTRY(IConnectionPointContainer)22 END\_COM\_MAP()23 24 BEGIN\_CONNECTION\_POINT\_MAP(CSimpleCom)25 CONNECTION\_POINT\_ENTRY(\_\_uuidof(\_ISimpleComEvents))26 END\_CONNECTION\_POINT\_MAP()27 28 29 DECLARE\_PROTECT\_FINAL\_CONSTRUCT()30 31 HRESULT FinalConstruct()32 {33 return S\_OK;34 }35 36 void FinalRelease()37 {38 }39 40 public:41 STDMETHOD(Hello)(BSTR\* a);42 private:43 int m\_iMember;44 CString m\_str;45 };46 47 OBJECT\_ENTRY\_AUTO(\_\_uuidof(SimpleCom), CSimpleCom)

[IMG_257](https://www.cnblogs.com/ioexception/p/javascript:void(0);)

[IMG_258](https://www.cnblogs.com/ioexception/p/javascript:void(0);)

1 // CSimpleCom.cpp 2 double Cosume() 3 { 4 double d = 0; 5 for (int i = 0; i < 1000\*1000\*300; i++) 6 { 7 d += i; 8 } 9 return d;10 }11 12 STDMETHODIMP CSimpleCom::Hello(BSTR\* a)13 {14 m\_str = L"0>你好！ ";15 Cosume();16 CString str;17 str.Format(L"1>m\_iMember = %d; " , this->m\_iMember++);18 m\_str += str;19 Cosume();20 m\_str += L"2>再见~";21 \*a = m\_str.AllocSysString();22 return S\_OK;23 }

[IMG_259](https://www.cnblogs.com/ioexception/p/javascript:void(0);)

将组件的ThreadingModel设置为Both，生成项目，组件会自动注册。

接下来创建C#客户端，使用该组件。

# **3. C#客户端**

新建一个C#控制台应用程序，添加对SimpleCom组件的引用，在主线程中创建SimpleCom组件的对象，在两个工作线程中同时调用该对象。

通过修改主线程的公寓类型，演示进程内COM组件对象在不同类型的公寓中的行为差异。

## **3.1 多线程公寓**

在多线程公寓中创建SimpleCom组件的对象的代码如下：

[IMG_260](https://www.cnblogs.com/ioexception/p/javascript:void(0);)

1 namespace ConsoleApplication1 2 { 3 class Program 4 { 5 [MTAThread()] 6 static void Main(string[] args) 7 { 8 var v = new ATLTestLib.SimpleCom(); 9 Thread t = new Thread(x =>10 {11 Run((ATLTestLib.ISimpleCom)x);12 });13 t.SetApartmentState(ApartmentState.STA);14 t.Start(v);15 Thread.Sleep(300);16 Thread t2 = new Thread(x =>17 {18 Run((ATLTestLib.ISimpleCom)x);19 });20 t2.SetApartmentState(ApartmentState.STA);21 t2.Start(v);22 }23 24 static public void Run(ATLTestLib.ISimpleCom sc)25 {26 try27 {28 for (var i = 0; i < 5; i++)29 {30 Console.WriteLine(string.Format("[{0}] {1}",31 Thread.CurrentThread.ManagedThreadId,32 sc.Hello()));33 }34 }35 catch (Exception ex)36 {37 Console.WriteLine(ex);38 }39 }40 }41 }

[IMG_261](https://www.cnblogs.com/ioexception/p/javascript:void(0);)

### **运行结果：**

[3] 0>你好！ 1>m\_iMember = 0; 1>m\_iMember = 1; 2>再见~  
[5] 0>你好！ 2>再见~  
[3] 0>你好！ 1>m\_iMember = 2; 1>m\_iMember = 3; 2>再见~  
[5] 0>你好！ 2>再见~  
[3] 0>你好！ 1>m\_iMember = 4; 1>m\_iMember = 5; 2>再见~  
[5] 0>你好！ 2>再见~  
[3] 0>你好！ 1>m\_iMember = 6; 1>m\_iMember = 7; 2>再见~  
[5] 0>你好！ 2>再见~  
[3] 0>你好！ 1>m\_iMember = 8; 1>m\_iMember = 9; 2>再见~  
[5] 0>你好！ 1>m\_iMember = 8; 1>m\_iMember = 9; 2>再见~2>再见~  
请按任意键继续. . .

### **原理说明：**

由于两个线程的代码能够同时调用组件对象v的方法，组件中m\_str的值被两个线程同时修改，Hello方法返回的值出现了混乱，典型的缺乏的同步的结果。

## **3.2 单线程公寓**

单线程公寓只需要将上面代码中的MTAThread改为STAThread即可。

### **运行结果：**

[3] 0>你好！ 1>m\_iMember = 0; 2>再见~  
[4] 0>你好！ 1>m\_iMember = 1; 2>再见~  
[3] 0>你好！ 1>m\_iMember = 2; 2>再见~  
[4] 0>你好！ 1>m\_iMember = 3; 2>再见~  
[3] 0>你好！ 1>m\_iMember = 4; 2>再见~  
[4] 0>你好！ 1>m\_iMember = 5; 2>再见~  
[3] 0>你好！ 1>m\_iMember = 6; 2>再见~  
[4] 0>你好！ 1>m\_iMember = 7; 2>再见~  
[3] 0>你好！ 1>m\_iMember = 8; 2>再见~  
[4] 0>你好！ 1>m\_iMember = 9; 2>再见~  
请按任意键继续. . .

### ****原理说明：****

由于对STA中对象的调用都被COM运行时列集，自动对多线程调用实现了同步。

https://blog.csdn.net/dyllove98/article/details/9735955