PH
PTR

计算机技术
译林
精选系列

COM精髓
（第三版）

David S. Platt 著
信达工作室 译

人民邮电出版社
www.pptph.com.cn

计算机技术译林精选系列

COM 精髓
(第三版)

David S.Platt 著

信达工作室 译

人民邮电出版社

图书在版编目(CIP)数据

COM 精髓/(美)帕拉特著；信达工作室译. —3 版. 北京：人民邮电出版社，2001.1
(计算机技术译林精选系列)
ISBN 7-115-09033-5

I. C… II. ①帕… ②信… III.软件接口，COM-程序设计 IV. TP311.52

中国版本图书馆 CIP 数据核字(2000)第 74845 号

计算机技术译林精选系列
COM 精髓（第三版）

◆ 著 David S. Platt
译 信达工作室
责任编辑 俞 彬

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ pptph.com.cn
网址 http://www.pptph.com.cn
北京汉魂图文设计有限公司制作
北京密云春雷印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787 × 1092 1/16
印张：20
字数：494 千字 2001 年 1 月第 1 版
印数：1 - 5 000 册 2001 年 1 月北京第 1 次印刷

著作权合同登记 图字：01 - 1999 - 3336 号
ISBN 7-115-09033-5/TP·2004

定价：38.00 元

内容提要

COM（组件对象模型）将程序员从底层细节中解放出来，同时独立于语言，因此成为编程利器，是开发复杂应用程序的首选工具。

本书介绍了COM的概念、术语、编程技巧和注意事项，从而带领读者步入COM编程的殿堂。本书包括15章，分别介绍了COM对象、对象服务器、自定义接口、自动化、类型库、线程和COM、分布式COM、永久对象、Moniker、异步COM、VB对COM的支持、活动模板库（ATL）、VC对COM的支持、VJ对COM的支持以及包容和积聚等内容。

本书对概念的阐述详细、透彻，同时包含了大量范例。可作为COM初学者的教材或COM程序员的参考资料。

版权声明

前　言

为何撰写本书

COM 以难学著称。一位顾客曾经对我说，世界上只有两个程序员理解 COM，他们都在为微软公司工作，编写对方难以理解的东西。实际上，情况并非如此。COM 的体系结构简单，并具有 Windows API 所没有的内部自身一致性。它只是以不同的方式来看待问题。本书试图澄清事实，并以简单、易懂的方式提供一些材料。

我希望通过本书证明，学习 COM 并非那么难。您也许需要花一些时间来弄懂基本概念，虽然我已尽最大的努力使您容易理解。但克服这些困难后（大约需要两个星期），您便掌握了 COM 的要点。学习 Windows API 主要是记忆一些杂乱无章的函数调用，并无原则可寻，就像在大学里学习有机化学反应式一样。学习 COM 更像是学习一些让整个世界正常运行的基本原则。还记得大学物理吗？最简单的等式（如 $F=ma$）描述了宏观运动的规律，从羽毛下落到天体碰撞。

几年来的进展

本书的第一版是在 1996 年秋季付印的，转眼间已过去了 4 年。对软件行业而言，4 年确实是一段很长的时间。大约 1 年后，即 1997 年的圣诞节，我推出了第 2 版。本版是在 2000 年春季付印的。修改的内容并没有完全反映 COM 的变化，虽然做了大量的修改（DCOM、线程和异步操作，这里只列出其中的几项）。我至今还记得，当我阅读第一版的某些内容时，我简直不敢相信我曾经以那样的方式来考虑问题。当我真正开始在项目中使用 COM 时，我改变了对 COM 中许多东西的看法。

有时，我羡慕哈佛大学艺术系的校友，它们的素材在 200 年中没有任何变化。软件时代以滚雪球的方式发展。COM 已经随处可见，成为所有软件的基础。编写本书是为了跟上时代的发展步伐，并让读者不被时代的列车抛在后面。

精简了内容

阅读过本书第 2 版的读者会发现，本版的厚度只有它的 60%左右。本书的首要目标是简洁：解释一些基本概念，以便读者能够容易地理解它们。自始至终，我只想在荆棘中开辟一条狭窄的通道。第 2 版包含的素材太多。因此，在这一版中，我只介绍那些实用的 COM 内容，并尽力精简其中的内容。我删除了与 MFC 相关的内容，我认为 MFC 已经发展到了尽头，也许 Jeff Prosise 不这样认为；删除了那些不再使用的用户界

面，只留下剪贴板和拖放范例，因为它对于说明 COM 的重要原则很有用。因此，内容少了大约 40%。

在线范例代码和每季一期的免费 COM/COM+开发简报

我决定将本书的范例代码放在我的网站（www.rollthunder.com）上，而不是放在光盘中。因为这样容易更新和添加，而读者不用记住光盘放到哪里去了。任何人都可以免费下载这些内容，即使他们没有购买本书。

既然登录到我的网站了，为何不订阅免费提供的、每季一期的 E-mail 简报呢？其标题为 ThunderClap。每一期都包含一篇技术文章，介绍最新的 COM 或 COM+开发问题。最新的一期讨论了 COM+中对象自动同步、COM+补偿资源管理器和 COM+事件系统探查程序。这也是免费的，请下载吧。

如何使用本书

如果您是 COM 新手，请首先阅读第 1 章和第 2 章！在熟悉这些内容之前，不要阅读其他章节。这两章介绍了新的编程模型和术语，不了解这些东西，将无法理解后续章节中的内容。我讨厌“模式转移”的提法，这个词被滥用了，已变得毫无意义，但 COM 确实会改变您的做法。

只有自己编写某些软件后，才能真正理解软件知识。因此，每章的最后都包含了自学的编程练习，并提供了详细的指示。这些练习与相应章节的内容紧密相关，因此并不难。练习中的选做部分是为开阔您的思维而设计的。

当您真正地编写程序时，请将本书放在手边。我的学生告诉我，这是他们首先参考的资料，因为从中容易找到所需的内容，而且容易理解。只有在本书中找不到需要的内容时，才到更厚的书中去查找。有的学生甚至有两本，一本放在家里，一本放在办公室。

提供反馈意见

我希望听到读者的反应，以便知道书中的哪些内容最有用。另外，虽然我和审阅人员尽了最大的努力，但错误在所难免，感谢读者批评指正，以便在下一版中改正。我的 E-mail 地址为 dplatt@rollthunder.com。我的网站（www.rollthunder.com）中包含了本书的更新内容和勘误表。

致谢

这里要感谢一些人员，如果没有他们，本书将无法完成。Mehedad Givehchi 审阅了本书，如果不做编程工作，他将是一名非常有前途的技术编辑。Pat Duggan 找出了前一版中的许多排印错误，他将免费获得本书。Kraig Brockschmidt 与我一起讨论了 OLE 的宗旨。我需要感谢帮助我理解 COM 内容的所有人员。《微软系统杂志》(Microsoft Systems Journal)的全体工作人员以及微软开发人员关系小组(Microsoft's Developer Relations Group)对我研究 COM 线程提供了极大的帮助。感谢 Dave Edson、Sara Williams、Charlie Kindel、Nat Brown、Dave D'Sousa 和 Eric Maffei。我在参加波士顿大学的 WindDev 会议时，从其他教员的讲解中学到了许多知识。感谢 Don Box、Jeffery Richter、Jeff Prosise、Ken Ramirez、Joe Najjar 和 Richard Hale Shaw。这里还要感谢 Mike Meehan、Jane Bonnell 和我的妻子 Linda。最后，感谢购买和推荐本书前两版的每一个人。希望您同样喜欢这一版。

·作者·

目　录

第 1 章　绪论：使用对象

1.1　概念和定义

COM（Component Object Model，组件对象模型）是微软公司的最高级的、包罗万象的二进制通信规范，用于软件组件间跨越多个进程、机器、硬件（微软公司希望有一天能够）和操作系统进行互操作。OLE 是一个高级功能（主要与用户界面相关）集合，它使用 COM 在组件间进行通信。COM 和 OLE 之间的界限并非总是非常清晰的，有关这个主题的争论一直很多。

术语 OLE/COM/ActiveX 非常混乱，如果您在微软开发人员网络光盘(Microsoft Developer Network CD-ROM)中搜索，您将发现，许多术语被赋予大量不同的含义，而且随时间的变化非常大，偶尔还会回到原来的意思。这些术语没有单一的含义，而且从来也没有真正有过。下面介绍如今文章中使用的这些术语的起源。

OLE 首次出现是在 1991 年，是 Windows 3.1 自带的，它使用动态数据交换（DDE）作为底层通信协议。OLE 最初的含义是对象链接和嵌入，因为这是它完成的功能，而且仅此而已。1993 年，微软推出了 OLE 2.0，它极大地提高了原来的对象链接和嵌入功能，增加了诸如在位激活和链接等真正管用的优秀功能。OLE 2.0 用 COM 代替 DDE 作为底层通信协议，这也是 COM 的第一项重要用途。它还增加了 OLE 自动化编程功能——一项到目前为止还没有任何人真正知道如何使用的功能。那时，微软宣称 OLE 不再是对象链接和嵌入的首字母缩写，而只是一个微软公司可能将其作为商标的随机音节组合——指的是建立在 COM 上的一整套技术。这种状况持续了多年。Kraig Brockschmidt 编写的《OLE 内幕》（*Inside OLE*,1995 年出版）介绍了基于 COM 的整套技术，并将它们称为 OLE。本书的第一版（1995 年初编写）的书名为《使用 ActiveX 的 OLE 技术基础》(*The ESSence of OLE with ActiveX*)。

大约在 1996 年，大多数开发人员开始编写 32 位的 Windows 95 应用程序，他们发现，OLE 使用 COM 的方式是一种非常好的设计软件的方法。开发人员开始使用类似的方法编写自己的对象和界面。另外，操作系统也开始要求使用 COM 技术编程，如编写 Windows 95 用户界面。这既不是对象链接和嵌入，也不是自动化。那么它到底是什么呢？此时，术语 COM 越来越频繁地被单独使用，也许最初就应该如此。OLE 不再像曾经的那样作为一个

包罗万象的术语使用。一些作者试图将 OLE 恢复到原来的含义——对象链接和嵌入，但这将使与 COM 相关的界面技术（如拖放和剪贴板处理）没有对应的术语。针对这种混乱情况，本人曾经杜撰了一个新词：MINFU——表示 Microsoft Nomenclature Foul-Up（微软公司使术语混乱）。

术语 ActiveX 是另一个赢家。在 Byte.com 的专栏中，我详细地介绍了该术语的起源以及随后被淡忘的情况。这里做一简短的总结，术语 ActiveX 没有任何技术上的含义，虽然在不同的时期，它确实有几种不同的含义。而现在，它只是一个标记性前缀，就像装有金色拱门的餐馆内所有食品的前面都有字母“Mc”一样。它指的是“微软热心制造混乱（Microsoft warm-fuzzy-good）”，仅此而已。

启动使用 OLE 或 COM 的应用程序时，必须首先在应用程序中调用 API 函数 OleInitialize( )或 CoInitialize( )来初始化它。前一个函数在内部调用后一个函数，并设置对 OLE 用户界面部分的支持。如果不需要使用用户界面支持，则只需要调用后一个函数。所有 OLE 和 COM 函数的包含文件都是通过头文件“windows.h”和引入库“ole32.lib”提供的。

如果仅是该调用成功，应用程序在关闭时才应该通过函数 OleUninitialize( )或 CoUniniltialize( )卸载这些库。OleUninitialize( )以及几乎所有其他与 COM 相关的函数的返回类型都是 HRESULT，它被定义为一个长整型。如果调用成功，该长整型值的高位被清空，否则设置它的值。宏 SUCCEEDED( )检测该位的值，如果调用成功，则返回 TRUE。下面是一个例子：

```
#include    <windows.h>  //include all OLE and COM files
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow)
{
   MSG msg;

/*
Initialize OLE Libraries
*/

    HRESULT hr = OleInitialize (NULL) ;
/*
Run the app.
*/

<all the other WinMain stuff>

/*
If they were initialized successfully,shut down the OLE libraries.
```

```
S_OK is a constant defined in the system header files.
*/
    if (SUCCEEDED(hr))
    {
        OleUninitialize ( ) ;
    }
     return (msg.wParam);

}
```

1.2　组件对象模型

在 COM 中，应用程序不是通过诸如 ShowWindow( )的 API 函数进行操纵。程序是由对象组成的，对象向外提供一个或多个接口。接口是一组相关的函数，函数操作它们所属的对象。不能直接访问对象中的数据，而只能通过对象的接口函数访问。从概念上说，这种模型与所有成员变量都是私有的 C++类有些类似。一些接口是标准的，而其他的则是程序员定义的。

您将经常听到术语“指向对象的指针”。虽然该术语经常作为一种方便的简写而使用，但这确实是用词不当。在 COM 中，根本没有指向对象的指针这种东西，有的只是指向对象接口的指针。实际上，是指向另一个指针的指针。第二个指针指向一个指针表，表中的指针指向接口成员函数。该指针表称为 VTBL（V 形表）。使用 C++语言实现对象的主要优点之一，是编译器将自动为您建立 VTBL。当您将指针指向其他人编写的对象的接口时，您不能对该对象的内部结构做任何假设。该对象可能是使用 COBOL 编写的。在图 1.1 中，VBTL 指针和对象的私有数据都是不确定的，以强调以下事实：不能对对象的内部结构做任何假设，将指针指向对象的接口时，可以调用接口中的方法，仅此而已。不能直接操纵内部引用计数 m_RefCount，而只能通过调用 Iunknown 接口方法 AddRef( )和 Release( )来引用它。

将指针指向对象（记住，这是将指针指向对象接口的简称）后，就可以通过调用接口中的成员函数与该对象通信。但如何将指针指向第一个对象呢？方法有多种。可以调用一个返回指向对象的指针的 COM/OLE API 函数，例如：

```
CoCreateInstace( ) //look in registry, launch COM server and get object from it
OleGetClipboard( ) //return an IDataObject interface pointer to the clipboard
's contents
```

一些接口中包含创建新对象并返回指向该对象的指针的方法。如果对象中包含这种方法，则可以调用相应的方法，例如：

```
IDataObject::EnumFormatEtc( ) //return an object that enumerates formats the data object
has
```

IClassFactory::CreateInstance( )//create a new object of the class manufacture by the factory.

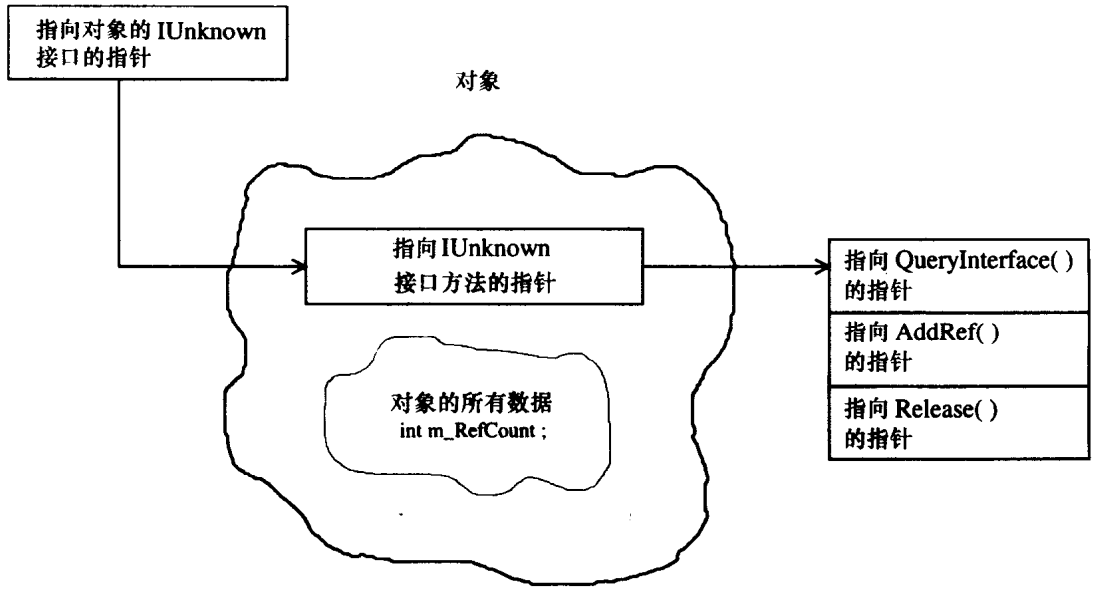

图 1.1

您可能注册一个这样的回调对象，即 OS 将调用其方法，并给您传递一个指向创建好的对象的指针，例如：

RegisterDragDrop( )　//为拖放操作注册一个回调函数

或者，您可能实例化一个其 VTBL 符合 COM 接口要求的 C++对象，例如：

new CSomeClass( )　//实例化一个实现 COM 接口的对象

1.3 IUnknown接口

每个 COM 对象都提供一个名叫 IUnknown 的接口，该接口包含方法 AddRef( )、Release( )和 QueryInterface( )。每个接口都是从 IUnknown 接口派生而来的，在其 VTBL 中开始的三个条目是指向上述三个函数的指针。不论您拥有哪个接口指针，也不论是如何得到的，您总是可以调用这三个方法，而且会经常调用它们。

前两个方法操纵一个控制对象的使用期限的内部引用计数。当对象第一次被创建时，创建者必须调用该对象的 AddRef( )，将计数加 1。每当其他的用户将一个指针指向该对象时，必须再次调用该对象的 AddRef( )方法。当用户不再使用对象时，他调用对象的 Release( )方法，将引用计数减 1。当最后一个用户调用对象的 Release( )方法后，计数的

值变为 0，导致对象释放自己。下面是 IUnknown::AddRef( )和 IUnknown::Release( )方法的简单实现：

```
/*
Increment member varible containing reference count.
*/

ULONG IUnknown::AddRef(void)
{
   m_RefCount ++ ;
   return m_RefCount ;
}

/*
Decrement member variable containing reference count. If 0, destroy object.
*/

ULONG IUnknown::Release(void)
{
  m_RefCount-- ;
  if (m_RefCount == 0)
   {
    delete this;
    return 0;
   }
   return m_RefCount ;
}
```

那么由什么负责调用 AddRef( )和 Release( )方法呢？有两个简单的规则：

- 每当您调用任何返回指向对象的指针的函数或方法（如 OleGetClipboard( )）或者对象的 QueryINterface( )方法时，提供指针的代码将会为您调用对象的 AddRef( )方法。在您不再使用该对象时，即使您没有对该对象执行任何操作，也必须调用其 Release( )方法。在您调用该方法之前，对象负责确保自己始终存在。
- 当其他人调用您的函数，并向一个回调函数传递一个对象指针参数（例如，在拖放操作中）时，则不会为您调用该对象的 AddRef( )方法。如果要存储一个指向该对象的指针，则必须调用它的 AddRef( )方法；而且只有在调用了 AddRef( )方法时，才需要在不再使用该对象时调用其 Release ( )方法。

当您获得一个指向对象的指针时，您真正获得的是一个指向其接口之一的指针。至于指向哪个接口，则取决于您是如何获得该指针。对象可能（或可能不）支持您感兴趣的其他接口，这无法预先知道。但由于每一个对象都支持 IUnknown 接口，因此可以通过

IUnknow::QueryInterface( )来询问对象是否支持您感兴趣的其他接口。

接口通过接口 ID（IID）——一个 16 个字节的常量（将在下一节介绍）来标识。标准接口的 IID 在系统头文件中定义，例如 IUnknown 接口的 IIID_IUnknown。当您调用 QueryInterface( )时，您需要传递欲查询的接口的 IID 以及一个指向一个输出参数的指针。如果对象支持请求的接口，则返回成功码 S_OK（被定义为 0），并使用指向被请求的接口的指针填充提供的输出参数。如果对象不支持该请求接口，则返回一个非零的错误码，并将输出参数设置为 NULL。下面是 IUnknown::QueryInterface( )的简单实现：

```
HRESULT IUnknown::QueryInterface(REFIID riid, LPVOID FAR *ppv)
{

/*Check the IID to see if we support the requested interface. if we do,
increment the reference count, fill the supplied output variable with a
pointer to the interface, and return the success code.
*/

   if (riid == IID_IUnknown || riid == IID_IDropTarget)
   {
      *ppv = (LPVOID)this;
      AddRef();
      return S_OK ;  //success code is 0, repeat zero
   }
/*
We don't support the interface. Set the supplied output variable to NULL
and return an informative error code.
*/
   else
   {
    *ppv = NULL;
    return  E_NOINTERFACE ; //Failure codes are nonzero
   }
}
```

1.4 GUID和UUID

用于唯一地区分 COM 中的条目的标识符是一个被称为 GUID（全局唯一标识符）或 UUID（通用唯一标识符）的常量。在前一节的 QueryInterface 范例中，使用一个接口 ID（IID）代表调用者查询的接口，该 IID 是一个用于标识接口的 GUID。在第 2 章，将介