ActiveX：

ActiveX[插件](http://baike.baidu.com/view/18979.htm" \t "_blank)以前也叫做[OLE控件](http://baike.baidu.com/view/3525904.htm" \t "_blank)或OCX控件，它是一些软件[组件](http://baike.baidu.com/view/379950.htm" \t "_blank)或[对象](http://baike.baidu.com/view/2387.htm" \t "_blank)，可以将其插入到WEB网页或其它[应用程序](http://baike.baidu.com/view/330120.htm" \t "_blank)中。

使用ActiveX[控件](http://baike.baidu.com/view/185331.htm" \t "_blank)可快速实现小型的组件重用、代码共享。

除此之外，为了满足更多地[需求](http://baike.baidu.com/view/195818.htm" \t "_blank)，也出现集成于各种开发环境的ActiveX控件，由第三方提供，如[ComponentOne Studio](http://baike.baidu.com/view/4665482.htm" \t "_blank)for ActiveX、[ActiveReports](http://baike.baidu.com/view/3105539.htm" \t "_blank)COM、TX Text Control ActiveX等。这些ActiveX[控件](http://baike.baidu.com/view/185331.htm" \t "_blank)有助于开发出商业级的Windows应用程序。

比较流行的[组件模型](http://baike.baidu.com/view/3852154.htm" \t "_blank)有COM（Component Object Model，对象组件模型）/DCOM（Distributed COM，[分布式对象](http://baike.baidu.com/view/4748611.htm" \t "_blank)组件模型）和CORBA（Common Object Request Broker Architecture，[公共对象请求代理体系结构](http://baike.baidu.com/view/3871415.htm" \t "_blank)）。到这里，已经出现了与本文相关的主题COM，而CORBA与本文无关，就不作介绍。之所以从组件与对象的区别说起，是想让大家明确COM和CORBA是处在整个体系结构的最底层，如果暂时对此还不能理解，不妨继续往下看，最后在回过头看一看就自然明白了。

现在开始阐述ActiveX、OLE和COM的关系。首先，让大家有一个总体的概念，从时间的角度讲，OLE是最早出现的，然后是COM和 ActiveX；从体系结构角度讲，OLE和ActiveX是建立在COM之上的，所以COM是基础；单从名称角度讲，OLE、ActiveX是两个商标名称，而COM则是一个纯技术名词，这也是大家更多的听说ActiveX和OLE的原因。既然OLE是最早出现的，那么就从OLE说起，自从 Windows操作系统流行以来，“剪贴板”（Clipboard）首先解决了不同程序间的通信问题（由剪贴板作为[数据交换中心](http://baike.baidu.com/view/929375.htm" \t "_blank)，进行复制、粘贴的操作），但是剪贴板传递的都是“死”数据，应用程序开发者得自行编写、解析数据格式的代码，于是[动态数据交换](http://baike.baidu.com/view/3822141.htm" \t "_blank)（Dynamic Data Exchange，DDE）的通信协定应运而生，它可以让应用程序之间自动获取彼此的最新数据，但是，解决彼此之间的“数据格式”转换仍然是程序员沉重的负担。对象的链接与嵌入（Object Linking and Embedded，OLE）的诞生把原来应用程序的数据交换提高到“[对象交换](http://baike.baidu.com/view/2314467.htm" \t "_blank)”，这样程序间不但获得数据也同样获得彼此的应用程序对象，并且可以直接使用彼此的数据内容，其实OLE是Microsoft的[复合文档](http://baike.baidu.com/view/571675.htm" \t "_blank)技术，它的最初版本只是瞄准复合文档（在文档中嵌入其它软件的功能），但在后续版本OLE2中，导入了COM。由此可见，COM是应OLE的需求而诞生的，所以虽然COM是OLE的基础，但OLE的产生却在COM之前。COM的基本出发点是，让某个软件通过一个通用的机构为另一个软件提供服务。COM的第一个使用者却是OLE2，所以COM与复合文档间并没有多大的关系，实际上，后来COM 就作为与复合文档完全无关的技术，开始被广泛应用。这样一来，Microsoft就开始“染指”通用平台技术。但是COM并不是产品，它需要一个商标名称。而那时Microsoft的市场专家们已经选用了OLE作为商标名称，所以使用COM技术的都开始贴上了OLE的标签。虽然这些技术中的绝大多数与复合文档没有关系。Microsoft的这一做法让人产生这样一个误解OLE是仅指复合文档呢？还是不单单指复合文档？其实OLE是COM的商标名称，自然不仅仅指复合文档。但Microsoft自己恐怕无法解释清楚，这要花费相当的精力和时间。于是，随着Internet的发展，在1996年春，Microsoft改变了主意，选择ActiveX作为新的商标名称。ActiveX是指宽松定义的、基于COM的技术集合，而OLE仍然仅指复合文档。当然，ActiveX最核心的技术还是COM。ActiveX和OLE的最大不同在于，OLE针对的是桌面上应用软件和文件之间的集成，而 ActiveX则以提供进一步的网络应用与用户交互为主。

### COM地位

到这里，大家应该对ActiveX、[OLE](http://baike.baidu.com/view/118545.htm" \t "_blank)和[COM](http://baike.baidu.com/subview/25941/5247585.htm" \t "_blank)三者的关系有了一个比较明确的认识，COM才是最根本的核心技术，所以下面的重点 COM。让对象模型完全独立于编程语言，这是一个非常新奇的思想。这一点从C++和Java的对象概念上，我们就能有所了解。但所谓COM对象究竟是什么呢？为了便于理解，可以把COM看作是某种（软件）打包技术，即把它看作是软件的不同部分（完成不同功能），按照一定的面向对象的形式，组合成可以交互的过程和一组支持库。COM对象可以用C++、Java和VB等任意一种语言编写，并可以用DLL或作为不同过程工作的执行文件的形式来实现。使用COM对象的浏览器，无需关心对象是用什么语言写的，也无须关心它是以DLL还是以另外的过程来执行的。从浏览器端看，无任何区别。这样一个通用的处理技巧非常有用。例如，由用户协调运行的两个应用，可以将它们的共同作业部分作为COM对象间的交互来实现（当然，OLE复合文档也能做到）。为在浏览器中执行从Web服务器下载的代码，浏览器可把它看作是COM对象，也就是说，COM技术也是一种打包可下载代码的标准方法（ActiveX控件就是执行这种功能的）。甚至连应用与本机OS进行交互的方法也可以用COM来指定，例如在Windows和Windows NT中用的是新API，多数是作为COM对象来定义的。

COM虽然起源于复合文档，但却可有效地适用于许多软件问题，它毕竟是处在底层的基础技术。用一句话来说，COM是独立于语言的组件体系结构，可以让组件间相互通信。随着计算机网络的发展，COM进一步发展为[分布式组件对象模型](http://baike.baidu.com/view/185351.htm" \t "_blank)，这就是DCOM，它类似于CORBA的ORB，本文对此将不再做进一步的阐述。通过上面的讲述相信大家一定对ActiveX、OLE和COM/DCOM的关系有了一个清楚的了解。

COM+

COM+并不是COM的新版本，我们可以把它理解为COM的新发展，或者为COM更高层次上的应用。COM+的底层结构仍然以COM为基础，它几乎包容了COM的所有内容。有一种说法这样认为，COM+是COM、[DCOM](http://baike.baidu.com/subview/387142/387142.htm" \t "_blank)和[MTS](http://baike.baidu.com/subview/83284/83284.htm" \t "_blank)(Microsoft Transaction Server)的集成，这种说法有一定的道理，因为COM+确实综合了这些技术要素。但更重要的一点是，COM+倡导了一种新的概念，它把[COM组件](http://baike.baidu.com/subview/185316/185316.htm" \t "_blank)软件提升到[应用层](http://baike.baidu.com/view/239619.htm" \t "_blank)而不再是底层的[软件结构](http://baike.baidu.com/view/600142.htm" \t "_blank)，它通过操作系统的各种支持，使[组件对象模型](http://baike.baidu.com/view/185318.htm" \t "_blank)建立在应用层上，把所有组件的底层细节留给操作系统。

对COM有使用经验的读者一定可以感觉到，虽然COM已经改变了Windows[程序员](http://baike.baidu.com/subview/39175/9475487.htm" \t "_blank)的应用开发[模式](http://baike.baidu.com/view/37878.htm" \t "_blank)，把组件的概念融入到Windows应用中，但是由于种种原因，DCOM和MTS的许多优越性还没有为广大的Windows程序员所认识。MTS针对[企业应用](http://baike.baidu.com/subview/9530552/9589930.htm" \t "_blank)和Web应用的特点，在COM/DCOM的基础上又添加了许多功能和特性，包括[事务](http://baike.baidu.com/view/121511.htm" \t "_blank)特性、安全模型、管理和[配置](http://baike.baidu.com/view/132733.htm" \t "_blank)等，MTS使COM成为一个完整的组件[体系结构](http://baike.baidu.com/view/1188494.htm" \t "_blank)。由于历史的原因，COM、DCOM和MTS相互之间并不很融洽，难以形成统一的整体，不过，这种状况很快就要结束，因为COM+将把这三者有效地统一起来，形成一个全新的、功能强大的组件体系结构，并且把[DCOM](http://baike.baidu.com/subview/387142/387142.htm" \t "_blank)和[MTS](http://baike.baidu.com/subview/83284/83284.htm" \t "_blank)的各种优势以更为简捷的方式带给Windows2000[程序员](http://baike.baidu.com/subview/39175/9475487.htm" \t "_blank)和用户。

COM+不再局限于COM的组件技术，它更加注重于分布式网络应用的设计和实现，已经成为Microsoft系统平台策略和软件发展策略的一部分。COM+继承了COM几乎全部的优势，同时又避免了COM实现方面的一些不足。COM+紧紧地与操作系统结合起来，通过[系统服务](http://baike.baidu.com/view/685551.htm" \t "J:/_blank)为[应用程序](http://baike.baidu.com/view/330120.htm" \t "J:/_blank)提供全面的服务，这一部分介绍COM+的基本结构。

与COM一样，COM+基于[二进制](http://baike.baidu.com/subview/18536/18536.htm" \t "_blank)组件和基于接口的编程。通过使用透明[RPC](http://baike.baidu.com/subview/32726/32726.htm" \t "_blank)层，可以跨越进程和计算机边界进行[远程方法调用](http://baike.baidu.com/subview/14042770/14617985.htm" \t "_blank)。正如[COM组件](http://baike.baidu.com/subview/185316/185316.htm" \t "_blank)那样，COM+组件可以在成品中升级和扩展，而不会对使用它们的客户端[应用程序](http://baike.baidu.com/view/330120.htm" \t "_blank)造成负面影响。

与[MTS](http://baike.baidu.com/subview/83284/83284.htm" \t "_blank)一样，COM+支持[分布式事务](http://baike.baidu.com/view/262223.htm" \t "_blank)和基于角色的安全性。它提供内置[线程池](http://baike.baidu.com/view/262150.htm" \t "_blank)方案，该方案与MTS的线程池方案一样透明。COM+编程模型同样利用侦听通过声明性属性向开发人员公开平台服务。但是，COM+比MTS更进一步地利用了基于属性的编程。除了[事务](http://baike.baidu.com/view/121511.htm" \t "_blank)性服务和集成安全性外，COM+还公开自定义对象构造、同步、[对象池](http://baike.baidu.com/view/262255.htm" \t "_blank)等服务。COM+的其他新功能（如排队组件和COM+事件）也通过可[配置](http://baike.baidu.com/view/132733.htm" \t "_blank)的属性公开.

OCX:

OCX 是对象类别扩充组件（Object Linking and Embedding (OLE) Control eXtension）；是不可执行的文件； 是 ocx 控件的扩展名，与 .exe、.dll 同属于[PE文件](http://baike.baidu.com/view/1087038.htm" \t "J:/_blank)。

控件的最大好处是可以重复使用，甚至可以在不同的[编程语言](http://baike.baidu.com/view/552871.htm" \t "J:/_blank)之间使用，例如，你可以在 VB 中嵌入用 VC 开发的控件。

控件的本质是微软公司的对象链接和嵌入(OLE)标准。由于它充分利用了面向对象的优点，使得程序效率得到了很大的提高，从而得到了广泛的应用。国外有很多公司就是专门制作各种各样控件的。控件的最早形式是以.VBX的格式出现的，后来变成了.OCX。由于Internet的广泛流行，微软公司推出了ActiveX技术，就是从OLE发展起来的，加入了WWW上的功能。所以目前最流行的是ActiveX控件。

ActiveX控件.ActiveX控件是可重用的[软件组件](http://baike.baidu.com/view/551014.htm" \t "_blank)。使用它可以很快地在网络应用程序、桌面应用程序、以及开发工具中加入特殊的功能。ActiveX控件的使用者可以无需知道这些组件是如何开发的，便可以完成网页或应用程序的设计。

对象链接和嵌入[用户控件](http://baike.baidu.com/view/160070.htm" \t "_blank)（OCX）是一种可以由在[微软](http://baike.baidu.com/view/2353.htm" \t "_blank)的Windows系统中运行的[应用软件](http://baike.baidu.com/view/7886.htm" \t "_blank)创建使用的特殊用途的程序。OCX提供操作滚动条移动和视窗恢复尺寸的功能。如果你有Windows系统，你会发现在你的Windows地址录里有大量OCX[文件名后缀](http://baike.baidu.com/view/1196023.htm" \t "_blank)的文件。 对象链接和嵌入（OLE）被设计来支持混合文档（包含多种资料类型，[比如](http://baike.baidu.com/view/6814120.htm" \t "_blank)[文本](http://baike.baidu.com/view/300107.htm" \t "_blank)、绘画图像、声音、动画）。Windows[桌面](http://baike.baidu.com/view/79807.htm" \t "_blank)就是一个混合文档的范例，[微软](http://baike.baidu.com/view/2353.htm" \t "_blank)使用OLE来建立的。OLE和[组件对象模型](http://baike.baidu.com/view/185318.htm" \t "_blank)（COM继OLE后的更常用的概念）支持“[即插即用](http://baike.baidu.com/view/33701.htm" \t "_blank)”程序的发展，“即插即用”程序在系统中可以用任何语言写入并可以由任何应用程序动态地使用。这些程序被认为是他们在其上运行的container的组建和应用程序。这种基于组件的近似应用程ˉ序的发展缩减了[程序调试](http://baike.baidu.com/view/182316.htm" \t "_blank)时间并改善了程序的性能和品质。Windows应用发展程序（[比如](http://baike.baidu.com/view/6814120.htm" \t "_blank)[powerbuilder](http://baike.baidu.com/view/9581.htm" \t "_blank)和Microsoft Access）具有OCX的优势。

ocx控件的安装方式有很多种，这里介绍最简单的一种。

步骤：

1.进入开始，点击运行。

2.在出现的框中键入regsvr32 C:\xxxx.ocx 。（XXXX为控件名， C:\为目录））（**regsvr32E:\控件的制作\myOcxTest\Debug\myOcxTest.ocx）**

3.点击确认后等待出现提醒注册成功即可。

**oca 文件**

Oca文件不是控件，其中保存Ocx控件的属性

**ocx的注销**

1.进入开始，点击运行。

2.在出现的框中键入regsvr32 /u C:\xxxx.ocx 。（XXXX为控件名， C:\为目录）（**regsvr32 /u E:\控件的制作\myOcxTest\Debug\myOcxTest.ocx**）

3.点击确认后等待出现提醒注销成功即可。

可能问题：

同一个OCX的多个控件在同一窗体上会出现问题的问题，主要原因是不能有全局变量（或共享数据） ，应该尽量定义成类的成员变量，全局变量在同一个进程中要被共享的。

|  |
| --- |
| ActiveX OCX是一个自定义控件,ActiveX DLL是一个动态库. OCX中有usercontrol,也可以包含class,而 DLL只能包括Class. |

**关于DLL的介绍**   
   
     DLL，动态链接库，Dynamic Link Library的缩写，是一个包含函数和数据的模块集合，可以被其它应用程序共享的程序模块。DLL作为共享函数库的可执行文件，封装了一个或多个已被编译、链接的函数。多个进程可以同时使用一个 DLL，在内存中共享该 DLL 的一个副本。DLL 还有助于共享数据和资源。   它和可执行文件(.EXE文件)非常类似，他们的区别在于 DLL 中虽然包含了可执行代码却不能单独执行，只能由需要使用它的应用程序来直接或间接调。

    通俗的说，在Windows操作系统中，许多应用程序并不是一个完整的可执行文件，它们的正确执行需要调用一些相对独立的动态链接库，即DLL文件。一个应用程序可以调用多个DLL文件，一个DLL文件也可能被几个应用程序所共用，这样的DLL文件被称为共享 DLL 文件。DLL 文件一般被存在C:\Windows\System 目录下，也可能放在应用程序所在的目录或是子目录中。

   提到动态链接，先说明一下静态连接。什么是静态连接呢?在程序链接的过程中，需要将编译后的二进制代码链接成目标代码，链接器从静态链接库中获得所有被引用的函数，并将这些被引用的函数同代码一起放到可执行文件中。那么关于 DLL 的静态连接则是指链接器将被引用的库函数的代码复制到调用 DLL 的可执行模块(.dll 文件或 .exe 文件)中。

  什么是动态链接呢？动态链接是系统允许可执行模块(.dll 文件或 .exe 文件)在运行程中，只需要包含在定位 DLL 函数的可执行代码所需的信息。换句话说，可执行模块(.dll 文件或 .exe 文件)在运行时加载这些模块(亦即所需的模块映射到调用进程的地址空间)。

   那么动态链接和静态连接 相比，优点有哪些呢？

    (1)节省内存，减少交换操作。使用动态链接，多个进程可以同时使用一个 DLL，在内存中共享该 DLL 的一个副本。使用静态链接，每个应用程序都包含被引用的库函数的代码，那么Windows 必须在内存中为每个应用程序加载引用的库函数的代码的一个副本。   
    (2)节省磁盘空间。使用动态链接，在磁盘上仅需要 DLL 的一个副本。使用静态链接，每个应用程序都包含被引用的库函数的代码。   
    (3)更易于升级。使用动态链接，DLL 中的函数发生变化时，只要函数的参数和返回值没有更改，就不需重新编译或重新链接使用它们的应用程序。使用静态链接，在函数发生变化时，需要重新链接来生成应用程序。  
    (4)支持多语言程序，只要程序遵循函数的调用约定，用不同编程语言编写的程序就可以调用相同的DLL 函数。  
    (5)提供扩展 MFC 库类的机制。可以从现有 MFC 类派生类，并将它们放到 MFC 扩展 DLL 中供 MFC应用程序使用。   
    (6)支持多语言程序，并使国际版本的创建轻松完成。通过将资源放到 DLL 中，创建应用程序的国际版本变得容易得多。可将用于应用程序的每个语言版本的字符串放到单独的 DLL 资源文件中，并使不同的语言版本加载合适的资源。[4]

  DLL 中包含下面两类函数的定义：   
  导出函数：这些函数由可执行模块(.dll 文件或 .exe 文件)调用。  
  内部函数：这些函数仅从定义它们的 DLL 中调用。DLL 还导出数据。不过，这些数据由相应的函数使用。

   可以通过下列方式调用 DLL 中的函数：    
    加载时动态链接：可执行模块执行显式调用以导出 DLL 函数。为 DLL 创建导入库，然后将 DLL 链接到应用程序。在加载应用程序时，导入库提供加载 DLL 和查找导出的 DLL 函数所需的信息。  
    运行时动态链接：在运行时加载 DLL 时，可执行模块使用 LoadLibrary 函数或 LoadLibraryEx 函数。可执行模块调用 GetProcAddress 函数以获取导出的 DLL 函数的地址。在链接时，Windows 搜索预安装的一组 DLL，例如性能库 (Kernel32.dll) 和安全库 (User32.dll)。然后，Windows 按以下顺序搜索DLL：  
1.当前进程的可执行程序所在的目录。  
2.当前目录。  
3.Windows 系统目录。（GetSystemDirectory 函数获取 Windows 系统目录的路径。）  
4.Windows 目录。（GetWindowsDirectory 函数获取 Windows 目录的路径。）  
5.PATH 环境变量中列出的目录。注意：LIBPATH 环境变量不用于搜索。[3]

    DLL 有一个特殊的入口点（DllMain 函数），它在附加和分离进程和线程时运行。此行为允许根据需要创建和销毁数据结构。文件扩展名为 .ocx、.cpl 和 .drv 的文件类型也是 DLL，尽管文件扩展名已改变。  
    您可以通过创建 DLL 实现以下目的：  
    (1)将程序划分为可按需加载的单独模块。   
    (2)存储特定于语言或特定于区域的资源。   
    (3)使您自己的应用程序能够使用核心代码库。   
    (4)生成进程内 COM 对象或 ActiveX 控件 (OCX)。   
    (5)将 OLE 对象用作进程内 DLL。这一用法可改进 OLE 链接的性能。   
    (6)使用控制面板扩展或使用某些类型的驱动程序。 [3]

**二、关于以.ocx为后缀名的ActiveX控件**

    刚才提到过，文件扩展名为 .ocx、.cpl 和 .drv 的文件类型也是 DLL。

    现在的ActiveX 控件等价与以前的OLE控件或OCX，一个典型的控件包括设计时和运行时的用户界面，唯一的IDispatch接口定义了控件的属性和方法，唯一的 IConnectionPoint接口定义控件可引发的事件。一个控件可以在容器中运行，所以从运行的角度看它类似与一个DLL。

    尽管 ActiveX 和 OLE 都基于组建对象模型(Component Object Model，COM)，它们为程序员提供的却是截然不同的服务。COM提供的是低级的对象捆绑机制，该机制支持对象之间的交互通讯。OLE使用COM来提供低级的应用服务，例如采用连接和嵌入机制，支持用户创建复合文档。与之不同，ActiveX提供更精细的结构，用以支持在网络站点上嵌入控件，以及对事件的交互反应。优化ActiveX，目的是为了提高时间和空间效率，而优化OLE，是为了便于终端用户的使用和集成台式系统的应用程序。ActiveX还为Internet技术带来了一些技术上的变革，例如，ActiveX大大减小了代码量(代码量减少了百分之五十到七十)，支持更多的提交和异步连接。 [6]

   ActiveX的基础是OLE和COM，但是通过MS的各种开发工具可以屏蔽掉COM模型中许多另人费解的技术细节。ActiveX组件技术包括以下一些方面：(1)自动化服务器 (2)自动化控制器 (3)控件 (4)COM对象 (5)文档 (6)容器 。

   以.ocx为后缀名的ActiveX控件主要应用在WEB上和Window Forms程序开发上。应用程序使用ActiveX/COM组件来扩展自身的业务逻辑、事务处理和应用服务的范围。

   顺便提一下以.ocx为后缀名的ActiveX控件的注册和卸载方法，在“开始”菜单的“运行”输入以下代码完成任务：  
    regsvr32 path & "\xxx.ocx" '注册  
    regsvr32 /u path & "\xxx.ocx" '卸载

    其中path代表该xxx.ocx所以在的目录的路径。

**三、OCX和DLL的区别**

   以.ocx为后缀名的ActiveX控件是一种比较特殊的DLL，它的基础是OLE和COM，是有交互界面的可视化控件,定义了控件的属性和方法,定义控件可引发的事件的响应。我们通常说的.DLL为后缀名的文件是一个包含函数和数据的模块集合，可以被其它应用程序共享的程序模块。

一般把Control翻译成控件，把Component翻译成组件。   
控件就是具有用户界面的组件。要说的具体一点，就得回顾早期 Windows 的历史根源，当时控件指任何子窗口——按钮、列表框、编辑框或者某个对话框中的静态文本。从概念上讲，这些窗口——控件——类似用来操作收音机或小电器的旋钮和按钮。随着控件数量的增加（组合框、日期时间控件等等），控件逐渐成为子窗口的代名词，无论是用在对话框中还是用在其它种类的主窗口中。没过多久 BASIC 程序员开始编写他们自己专用的控件，自然而然地人们便想到共享这些控件。共享代码的方法之一是通过磁盘拷贝，但那样显然效率低下。必须要有一种机制使开发者建立的控件能够在其它程序员的应用中轻而易举地插入，这便是VBA控件，OLE控件，OCX和最后ActiveX 控件的动机。   
这就是控件和组件之间产生混淆之所在。因为为了解决控件的可复用问题，所有这些技术必须首先解决更为一般的组件重用问题。（COM，如果你还记得它的话，意思是组件对象模型）。在软件行话中，组件这个术语指任何可复用的对象或任何可与其它对象交互的代码体。子程序的发明，曾经一度成为程序员趋之若鹜的软件工程圣杯：一种统一的编程理论，它使程序员从基本构建块——也就是用所选语言编写的各种组件建立大型系统。从子程序演变到OOP，到DLLs，再到COM，再到.NET框架的每一种新的编程范例都代表了一种不同的提供可重用性的方案。VBX使用DLLs的固化名称。COM使用接口和IUnknown。.NET框架使用微软的中间语言（MSIL）层和公共语言运行时（CLR）来提供统一的粘合。

Component Object Model（COM——组件对象模型）

组件对象模型，一种[面向对象](http://baike.baidu.com/view/125370.htm" \t "J:/_blank)的编程模式。它定义了对象在单个应用程序内部或多个应用程序之间的行为方式。

COM是微软自1993年便提出的组件式软件平台，用来做[进程间通信](http://baike.baidu.com/view/1492468.htm" \t "J:/_blank)（Inter-process communication, IPC）以及当作组件式软件开发的平台。COM提供跟编程语言无关的方法实现一个软件对象，因此可以在其他环境中运行。COM要求软件组件必须遵照一个共同的[接口](http://baike.baidu.com/view/159864.htm" \t "J:/_blank)，该接口与实现无关，因此可以隐藏实现属性，并且被其他对象在不知道其内部实现的情形下正确的使用。

COM并被实现于多个平台之上，并不限于[Windows](http://baike.baidu.com/view/4821.htm" \t "J:/_blank)操作系统之上。但还是只有Windows最常使用COM，且某些功能已被目前的[.NET](http://baike.baidu.com/view/4294.htm" \t "J:/_blank)平台取代。

[Windows](http://baike.baidu.com/view/4821.htm" \t "J:/_blank) 操作系统提供了三种应用程式间的通讯机制：[剪贴板](http://baike.baidu.com/subview/50291/50291.htm" \t "J:/_blank)（clipboard）、[DDE](http://baike.baidu.com/view/812487.htm" \t "J:/_blank)与[OLE](http://baike.baidu.com/view/118545.htm" \t "J:/_blank)。OLE原名是物件连结与嵌入（Object Linking and Embedding），OLE可说是DDE的改良版，OLE 1.0版提供[复合文件](http://baike.baidu.com/view/3940098.htm" \t "J:/_blank)（compound document）处理。但过於复杂，Brockschmidt, Kraig「Inside OLE」一书中提到，必须经过六个月的心灵混沌期，才能了解OLE是什么。因此OLE 2.0后，微软提出COM架构。所有OLE元件皆是继承COM而来，这些技术包含OLE Document和OLE Controls、Drag and Drop等。

COM组件类型：

COM是基于组件对象方式概念来设计的，在基础中，至少要让每个组件都可以支持二个功能：

查询组件中有哪些界面

让组件做自我生命管理，此概念的实践即为[引用计数](http://baike.baidu.com/subview/4163962/4163962.htm" \t "J:/_blank)（Reference Counting）

这二个功能即为COM的根：[IUnknown接口](http://baike.baidu.com/subview/2009591/2009591.htm" \t "J:/_blank)所提供的IUnknown::[QueryInterface](http://baike.baidu.com/subview/5395542/5430176.htm" \t "J:/_blank)()，IUnknown::AddRef()及IUnknown::[Release](http://baike.baidu.com/subview/946743/946743.htm" \t "J:/_blank)()三个方法的由来。所有的[COM组件](http://baike.baidu.com/subview/185316/185316.htm" \t "J:/_blank)都要实现IUnknown，表示每个COM组件都有相同的能力。

只由COM派生实现出来的组件，称为**纯COM组件**。

但在[Windows](http://baike.baidu.com/view/4821.htm" \t "J:/_blank)持续发展时，[Visual Basic](http://baike.baidu.com/view/14260.htm" \t "J:/_blank) 4.0开始支持[OCX](http://baike.baidu.com/view/393671.htm" \t "J:/_blank)，也就是OLE Custom Control，这让微软开始思考要如何让COM组件可以跨语言支持，在这样的要求下，必须要提供一个一致的接口，以及提供一组可以调用接口内方法的能力，由于纯[COM组件](http://baike.baidu.com/subview/185316/185316.htm" \t "J:/_blank)只能够支持[C/C++](http://baike.baidu.com/view/1107274.htm" \t "J:/_blank)的直接访问，为了要达到跨语言的能力，在COM中必须要支持在外部调用内部方法的机能，这个机能造就了Invoke()方法，另外为了跨语言的支持，COM应该要提供简单的组件访问识别方式，这也就是会有GetIDsOfNames()的原因，将这些方法组合起来，定义出的必要接口，称为[IDispatch](http://baike.baidu.com/view/10222704.htm" \t "J:/_blank)接口，所有实现此接口的，都可以支持跨语言的支持。

微软将实现此接口的组件都称为**自动化（Automation）**组件。