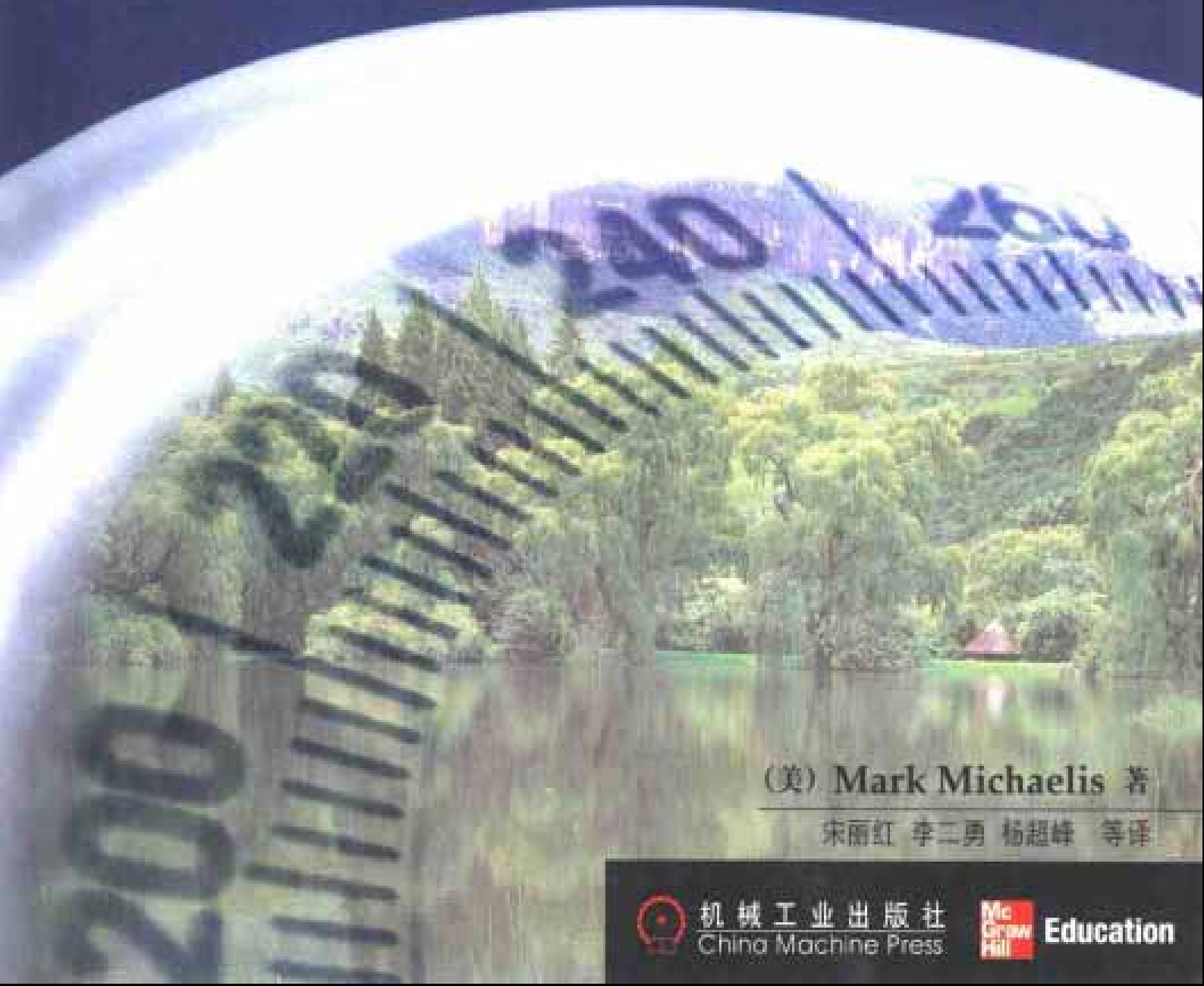
COM+ Programming from the Ground Up

软件开发技术丛书

COM+ 编程指南



(美) Mark Michaelis 著

宋丽红 李二勇 杨超峰 等译

机械工业出版社 China Machine Press

Mc Graw Hill Education

软件开发技术丛书

COM+编程指南

（美） Mark Michaelis 著

宋丽红 李二勇 杨超峰 等译

前导工作室 审校

机械工业出版社
China Machine Press

本书是一本极具价值的学习 COM+ 的编程参考书。它循序渐进地带领读者一步步深入到 COM+ 的核心技术，以具体生动的实例把复杂的 COM+ 结构展示在读者面前。书中不仅注重 COM 基本知识的分析，而且向读者介绍了许多在 COM+ 服务及客户程序编写过程中常用的编程技巧和经验。

本书适合那些对 COM 及 COM+ 有兴趣的初、中级读者阅读，也适合各大专院校的师生学习使用。相信无论是初学者，还是有一定编程经验的开发者都会从本书中受益。

本书版权登记号：图字 01-2001-3955

图书在版编目(CIP)数据

COM+ 编程指南/(美)麦克利斯(Michaelis, M.)著；宋丽红等译. —北京：机械工业出版社，2002.1
(软件开发技术丛书)
书名原文：COM+ Programming from the Ground Up
ISBN 7-111-09536-7

Ⅰ.C… Ⅱ.①麦… ②宋… Ⅲ.软件工具-程序设计-指南 Ⅳ.TP311.56-62

中国版本图书馆 CIP 数据核字(2001)第 078834 号

机械工业出版社(北京市西城区百万庄大街 22 号 邮政编码 100037)
责任编辑：李云静
北京昌平奔腾印刷厂印刷·新华书店北京发行所发行
2002 年 1 月第 1 版第 1 次印刷
787mm×1092mm 1/16·28.75 印张
印数：0 001-4000 册
定价：48.00 元

译 者 序

COM+是微软 COM 技术的最新版本，它是微软事务处理服务和几个新服务的并集。COM 代表组件对象模型，它定义了标准构建软件组件的方法。COM+这项技术推动了软件组件的革命，它重新制定了编写程序的方法。软件组件提供了一个灵活、有弹性、便于管理大规模软件系统的方法。软件组件革命重新定义了程序真正的本质和我们思考程序的方式。

COM 和 COM+已成为 Windows 程序设计的标准。基于 Windows 平台的许多技术都利用了 COM 技术，如活动目录服务、数据远程访问、微软的 DirectX 技术等等。COM 技术已经是开发基于 Windows 平台应用程序的程序员必备的一项关键技术。

但许多程序员在使用组件技术搭建应用程序的时候，面对 COM+技术庞大的体系结构和编程规范望而却步；本书可使初中级开发人员不再彷徨，并能全面学习 COM+技术，掌握 COM+编程规范，解决实际问题。

本书假定读者以前没有 COM 方面的知识和经验，并从头开始教授 COM+编程知识。它从介绍隐含在 COM 背后的基本概念开始，如接口、虚函数表和 IUnknown，然后再一步步地推进，遍及所有的 COM 及 COM+基础内容，包括接口定义语言、事件、智能指针、连接点和活动模板库的使用等。本书以简洁的语言、完整的代码实例、详实的分析使读者能够真正理解 COM 每一个要点的意义和用法，并在很短的时间内对 COM+有一个彻底的了解。

全书由宋丽红、李二勇、杨超峰、杨晓鹏、朱明峰、雷臻、韩隽、张睿、王大峰、于蒙、喻湘宁、刘斌、李冬、冷涛、刘延文、郝建平、顾红玲、黄迪颖、陈松、卢毅然等进行翻译，最后由梁静统稿。前导工作室全体工作人员为本书的出版付出了辛勤的劳动。由于时间仓促，且译者的水平有限，在翻译过程中难免会出现一些错误，恳请读者批评指正。

如果你在阅读中碰到了什么问题，请同前导工作室联系：qiandao@263.net。我们会尽力解决你的问题。

2001 年 11 月

前　言

COM+是微软COM技术的最新版本，它是微软事务处理服务和几个新服务的并集。正如大多数读者所了解的，COM代表组件对象模型，它定义了标准构建软件组件的方法。"+"号部分增强和优化了这个过程。虽然前面对COM+的描述是准确的，但并没有说明COM+在计算领域中所产生的重大影响。

COM+很重要，因为它促进了软件组件的革命，它重新制定了编写程序的方法。程序再也不受传统的、整体模块模型的束缚，即那种在一个文件中包含所有的功能，在编译时被确定的模式。代替它的是，基于组件的体系结构使用那些可以在需要时使用、在必要时改变、在有可能的情况下增强性能的自给自足的组件，而所有这些都不用改变核心应用程序的代码。软件组件提供了一个灵活、有弹性、对组织大规模软件系统也便于管理的方法。软件组件革命重新定义了程序真正的本质和我们思考程序的方式，这样说一点也不过分。

本书假定读者以前没有COM方面的知识和经验，并从头开始教授COM+编程知识。它从介绍隐含在COM背后的基本概念开始，如接口、虚函数表和IUnknown。然后再一步步地推进，遍及所有的COM基础内容，包括接口定义语言、事件、智能指针、连接点和活动模板库的使用等。

全书以分析COM+新增加的COM特征，如组件服务、事务处理和排队组件结束。当你看完本书的时候，就会对COM有一个清晰、完整的理解。

你必须具有的编程技能

从任何角度来看，COM+都是一个具有挑战性的专题。你必须已经是一个有经验的C++程序员，具备对如虚函数、模板和抽象类等的正确认识。你也必须知道如何编写Windows程序，能够处理它重要的概念，如消息循环、回调函数和线程。坦率地说，完全掌握COM+需要使用你所有的编程技能。虽然这具有很大的挑战性，但一个专业程序员完全可以掌握COM+这项重要的规范。

你需要的软件

为了验证本书中的所有例子，你需要Windows 2000系统和基于Windows的C++编译器。在书中前些章节的大多数例子，也可以在Windows 98和Windows NT 4系统中工作。但专门讲COM+的章节(第14章和第15章)要求Windows 2000系统。用做测试书中例子的编译器是装有Service Pack 3的补丁包的Visual C++ 6.0。

不要忘记:在网上的代码

记住,本书中所有程序的源代码可以在 http://www.osborne.com 网站上免费获得。下载这些代码可以在测试这些例子时,省去麻烦的输入工作。

目　录

第1章 COM+介绍

COM+是微软COM系统的最新版本。COM代表组件对象模型(Component Object Model),它定义了一套为构建一个组件所必须遵循的规则。"+"表示对一些特征和服务进行了改进,在某些方面,简化了某些与COM相关任务的新属性和服务。在不远的将来,成为一个专业的程序员将意味着必须具有编写遵循COM规范的软件组件的能力。没有能力处理COM的程序员将会落伍。坦率地说,COM几乎将会被所有程序员使用。

本书是最早和最重要的一本COM+编程的实践指南。因此,它不会用大量的无节制的篇幅写COM的理论部分,除非这些理论直接与编程相关。我们的目标是提供一个可以使你能举一反三的尽可能利用COM+功能的方法。也就是说,为了高效地使用COM+去创建软件组件,要求你理解驱动创建组件的力量、组件的基本设计原则和一些基本概念。你也需要熟悉一些新的术语和元素。

由于COM+是COM的扩展,因此,除非我们明确地提到COM+所特有的特征,否则在整本书中仅使用COM这个术语。

注意 本书假定你熟悉C++,已经写过Windows程序。如果你在这两个方面的知识有些欠缺,那么在进一步学习COM+编程之前,你需要弥补这方面的不足。

1.1 组件的革命

我们,也就是世界上的程序员,正处在改变编写程序方式的组件革命中。这是一个强有力的声明,但在软件专业中,总的说来只是不算少的软件革命中的一次革命,因为我们已经经受过许多的"革命"。例如,C语言代替了FORTRAN,C++促进了面向对象编程的发展,Java激发了Web技术。但即将到来的这次革命意义是非常重大的,因为它从根本上重新制定了编写程序的结构和体系结构的方式。而且,这些变化影响了一个广泛的应用程序领域,并正在改变着计算技术的本身前景。我们所指的革命是向组件软件(component software)发展。可以肯定地说,组件软件革命将产生深刻和持久的影响。在我们开始分析COM之前,先理解为什么软件组件是这么重要和它们怎样改变了程序的体系结构是很有帮助的。正如你将看到的,组件不仅仅是另一种编程方法,而且它们是一种更好的编程方法。

1.1.1 两个目标

组件革命用于设法达到编程中的两个最基本的目标:

- 管理日益增加的复杂的应用程序。
- 重用代码。

让我们一起分析每一个目标。

1. 程序复杂性

复杂性是程序员面对的一个最大挑战。初学者在他们的编程生涯中，很早就知道，程序越长，调试的时间就越长。随着程序大小的增长，通常程序的复杂性也在增加，而我们人类所能控制的复杂性是有限的。从纯组合的观点来看，单行代码越多，出现副作用和不必要的交互的机会就会越多。正如大多数程序员现在所了解的，程序发展得非常复杂了。

软件组件通过允许我们"分开和克服"帮助我们控制程序的复杂性。通过应用组件技术，我们可以把一个应用程序归纳成几个组成部分。每一部分作为独立的单元，可以分别地进行编码和维护。也可以使用第三方提供的现成的组件。通过把功能单元划分成独立的资源，程序员可以减少一个大程序的复杂性。

2. 重用代码

从数据处理一开始，程序员就开始探求能重用其代码这个能力。由于开发和调试是一个付出很大代价的过程，因此重用代码是大家梦寐以求的。在早些时候，重用代码是从一个程序源代码中剪切，到另一个程序的源代码中粘贴完成的，这种方法到现在仍然使用。可重用函数库的形式向前发展了一步，如那些由C提供的可重用函数库。很快接下来的就是标准的类库，如那些用C++表示的类库。软件组件使组件重用向前跨越了巨大的一步，因为它们允许由独立的开发者创建自给自足的二进制模块。但组件不像类或函数库，它们是编译时的模块，组件则是在应用程序运行时即插即用的。而且，任何需要这个组件的应用程序都可以使用它。这样，一旦写好了一个组件，它可以被无数的应用程序使用。

软件组件以另一种方式扩展了重用。组件可以用任何一门支持创建组件的计算机语言编写。它可以被任何可以允许使用组件的应用程序所使用，无论使用哪种语言创建的应用程序。这样，在一种语言中创建的代码可以被用另一种语言编写的应用程序所使用。这使应用程序可以包括多个独立的模块，每一个模块可以使用最适合的工具去创建。

1.1.2 单一的大模块与组件模型的比较

基于组件应用程序的体系结构从根本上与传统的单一的大模块应用程序的体系结构不同。为了理解它们之间的区别，我们比较一下用每一种方法实现的同一个应用程序。

1. 单一的大模块模型

假设你为一家软件公司工作，担任负责为制造业开发软件的产品经理。软件在生产车间的计算机上运行，跟踪许多种产品的生产，从飞机到瓶子盖儿。应用程序还必须跟踪描述如何在制造零件中执行一个特定步骤的设计图表。软件应该存储关于产品资源的信息，以使在制造过程中的改进可以和有需要生产的部件订单的商务系统相连接。另外，你的系统将与生产车间的机器相互作用，开始和停止该机器的工作，并要对各种事件做出反应如机器故障或损坏。

写这个制造软件的传统的方法是创建一个大的单一模块的应用程序。应用程序将包含几千行的代码，需要一个大开发团队的内部程序员来维护它。这个程序要用一种语言开发，如C++。即使应用程序的某些部分在其他语言中编写会更方便，但由于有将两种或更多种的语言相协调

的相关问题,你也不愿意这样做。

单一的大模块解决方法提供了固定的功能。换句话说,程序几乎不可能为(或者由)每一个客户端程序进行定制。如果客户端程序不喜欢系统一个特定的部分,它们只能请求改变,希望这个建议能在下一个版本中被实现。每一个不同的制造过程需要一个定制的实现。每一个新客户端程序将产生一个新的开发周期。在这个领域中升级是非常困难的,因为应用程序包含一个单一的大执行文件。单一的大模块化的方法也妨碍了(或严格地限制了)第三方的附加或增强模块。因为所有的代码包含在一个大模块中,很难向编译过的代码中增加新的功能。

2. 组件模型

现在考虑用软件组件实现一个同样的制造系统。基于组件的解决方法围绕着一组独立的、提供所需要功能的组件设计。例如,控制生产车间一些设备(像包装机)的这部分代码可以成为一个独立的组件。主应用程序以定义好的、不可改变的方式与这个组件进行相互通信。基于组件的方法有几个优点:首先,整个应用程序的复杂性降低了。在单一模块内的代码量,复杂性趋向于以指数级而不是以直线级地增长。使用单一的大模块方法开发的开发者必须不仅熟悉他们自己的代码,也必须熟悉他们希望与之互相作用的所有代码。在单一的大模块应用程序中,在一部分代码中一个很小的变化也会对其他部分有相当大的影响。当使用基于组件的方法时,每一个模块都是独立的。这样,每一个程序员仅仅需要控制他或她正在编写的组件的复杂性。由于每个组件是独立的,一个组件的内部对另一个组件的内部没有影响。通过"分开和克服"技术,应用程序的复杂性被降低了。

因为每一个组件是独立的,所以可以用任何适合的语言编写组件。只要组件保持二进制级的兼容,就是与语言无关的。这使每个开发者可以使用最适合这项工作的开发工具。

基于组件的方法提供了一个方便的定制应用程序或给应用程序增加功能的方法。例如,如果用户用另一种机器代替了加了标签的机器,用新机器的组件替代旧的处理标签机器的组件就是很容易的一件事。只要新组件是与旧组件"插入兼容"的,新组件就可以提供所需要的附加功能。此外,处理新的标签机器的新组件不必由应用程序的开发者提供。例如,它可以由生产新标签机器的公司或其他的第三方提供。这样,应用程序的功能可以以开发者没料到的方式得到改变或扩充。

1.2 COM 的产生

在 20 世纪 80 年代晚期和 90 年代早期,微软公司遇到了用单个大模块模式开发受到限制的困难。对于他们来说,这个系统是微软的 Office。他们想使一种类型的文档,如 Microsoft Excel 的电子表格,嵌入到第二种类型的文档中,如 Microsoft Word 文档。但是,他们不满足于仅在字处理文档中显示电子表格。他们还想从字处理文档内部提供电子表格的所有功能,好像电子表格是被创建到字处理器中的一样。针对这个问题,在 1991 年,微软开发并公开发布了一种叫做对象链接和嵌入(OLE)的技术。对象链接和嵌入这项技术的创建使一种类型的文档可以被链接或嵌入到另一种类型的文档中。OLE 1.0 使用的基本技术是动态数据交换(DDE),虽然它能够工作,但它非常复杂。面对这些复杂性,当微软创建 OLE 2.0 时,有一个重大飞跃。OLE 2.0 的一个最重要的新特点是组件对象模型的定义。微软使用 COM 代替了对象链接和嵌入的动态数

据交换的体系结构。

到1995年,非常明显,COM而不是OLE,是关键的技术。OLE仅是可以应用COM的一个小子集。结果是把对OLE技术的关注,即允许两个不同文档类型互相通信的技术,转移到COM上,即COM技术使任何的COM组件可以和任何其他的COM组件相互通信。这成为了解决微软Office的和那些打算写包含许多二进制模块复杂系统的其他系统开发者的解决方法。

最初,许多程序员接受COM的过程很慢,因为它是一项全新的、看起来很复杂的技术。当然,这一点,要改变,因为COM的功能太强大了,不能被忽视。就像你读完本书所能领会到的,COM用起来并不困难,现代的编程工具使它非常容易使用。

1.3 什么是COM

正如本章开始部分所提到的,组件对象模型定义了一套在构建一个组件时所必须遵循的规则。COM不是一个特殊类型的应用程序,而是一个可以用来为任何类型的应用程序构建组件的普遍的模型。例如,ActiveX和OLE都使用COM,但两者都不是COM。

> COM定义了标准的构建组件的方法。
> COM规定了一个二进制的标准。

COM是重要的,因为它定义了标准的构建组件的方法。理论上,任何人都可以定义一个组件的体系结构。但是一个人的方法可能与另一个人的方法不同,他们的组件会互不兼容。仅当我们达成一致使用同一个规范时,软件组件才是有价值的。这样做可以使一个程序员写的组件可以被第二个程序员使用。只要两个程序员都遵循同一个标准,他们的组件就可以一起工作。

可能COM最重要的一个特点就是COM规定了一个二进制的标准。那就是说,它定义了一个组件在它被编译后的形式是怎样的。COM不规定该怎样编写组件的源代码。它仅指定二进制对象的格式。这使COM成为了一个强大而灵活的组件体系结构。

COM规定了以面向对象的方法去创建组件。因为COM代表组件对象模型,这一点并不奇怪。尽管COM对象与你所熟悉的C++类对象有所不同,但COM对象遵守面向对象程序设计(OOP)的主要原则。

COM属性

下列属性表现了组件对象模型的特性:

- 面向对象的编程。
- 松耦合(松驰耦合)。
- 稳定的版本转变。
- 位置透明性。
- 语言无关性。

我们将依次对每一个属性进行分析。

1. 面向对象编程

COM支持面向对象的三个原则:封装、多态和继承。

封装对于面向对象编程是重要的,与它对于 COM 组件是重要的有同一个原因:封装提供了一种将代码和数据绑定到一起的方法,保护两者的安全,防止外部接口的访问或误用。

通过封装,组件的用户仅仅需要知道怎样和组件进行交互,而不用知道组件是怎样工作的。组件的用户不必知道组件内部的数据格式、设计和它的算法。这个模块是用什么语言编写的对于组件用户也是不重要的。用户仅需要了解专门用于与组件交互的方法。

多态性,它的特点是"一个接口,多个方法",是使一个接口可以控制对一个一般类的访问动作。当多态性应用于 COM 时,多态性是指使程序员可以描述某个组件的一般的本质,同时使每个程序员可以按照他或她认为合适的方法自由地实现这个组件的属性。这样,你可以指定一个组件要完成哪些工作而让其他人决定组件该如何完成这些工作。

继承是使一个组件可以继承另一个组件性能的属性。正如你所了解的,COM 支持继承与 C++有一点不同。这需要从开发者这方面做一点模式上的变换,但尽管如此,继承在 COM 工作的方式上起到了关键的作用。

2. 松耦合(松驰耦合)

每个组件都是松耦合(松驰耦合)。这使组件可以随意地更换。例如,如果一个来自第三方的压缩组件证明功能不足,那么你会想灵活地用另一个第三方提供的组件替代,并仅对应用程序有最小的影响。只要新的压缩组件与原来的组件可以以同样的方式访问(说得更精确些,就是如果新的压缩组件继续支持同样的二进制标准),那么它可以代替原来的组件。

3. 稳定的版本转变

随着组件从一个版本升级到另一个版本,与新组件相互作用的以前存在的模块不应该破坏。这使每一个组件的升级不破坏使用这个组件的应用程序。这一点特别重要,考虑一下,那些使用你软件的用户在系统上安装其他版本的应用程序,这些应用程序可能提供比你的应用程序所使用组件版本更新的组件。使用 COM,版本转变是稳定的,使一个发布版本到下一个发布版本可以平稳地过渡。

4. 位置透明性

位置透明性是 COM 组件的另一个关键属性。把组件从一台计算机转移到另一台计算机上仅仅会涉及到重新配置的问题,而不会涉及到一个大的开发项目。这一点提供了在网络环境中分布式装载的灵活性。位置透明性也适用于组件运行的上下文。这使组件可以与你的应用程序在同一个进程中或在一个独立的进程中运行。

深入分析

为什么传统的 DLL 不足以成为一个组件模型

你可能想到,COM 的这几个方面的属性也是传统的动态链接库(DLL)的属性。例如,DLL 也是一个二进制标准,它也提供了有限的封装形式。这些相似点自然会引起下列问题:为什么 DLL 不能是软件组件的模型?因为 DLL 的规范没有提出某些关键的问题。例如,当使用第三方提供的压缩模块时,在哪里负责分配内存?如果指向压缩模块的指针被传递给另一个模块,谁负责卸载这个模块,怎样才能让他或她知道什么时候卸载它才是安全的?而且,哪一个机制可以保证由一个压缩模块提供的接口与另一个模块提供的接口是相匹配的?一个

传统的DLL不能回答这些问题。这需要一个严格地确定成为一个组件意味着什么和那些组件之间如何相互作用的标准。当然,这就是COM的目的。

最后一点:虽然传统的DLL自身不足以完全地指定一个组件标准,但它们支持COM。例如,通常COM组件被存储在DLL中。因此,DLL对输送COM组件是很方便的,因为它们提供了绑定和装载机制。有了COM相对于DLL附加的部分,普通的DLL机制的缺点都被去除了。

5. 语言无关性

因为COM定义了一个二进制标准,所以COM是与语言无关的。换句话说,用于写COM组件的计算机语言是不重要的。例如,一个由第三方提供的压缩组件是用C++开发的,另一个组件是用汇编语言开发的,这个事实并不阻止你用一个第三方的组件替代第二个组件,反过来也可以。另外,在你自己的应用程序内部,你可以对不同的组件使用不同的语言。如果在Visual Basic中开发用户界面和在C++中开发机器间的通信模块比较容易,那么就可以这样做。因为它与COM相关联,语言不是个问题。

1.4 组件对象

组件对象是任何一个遵循COM标准的二进制对象。

直到现在,我们仍使用"组件"这个词很不精确地指某种类型的"二进制大对象"。现在是精确定义它的时候了。在COM的上下文中,这些二进制对象叫做COM组件对象或简单地叫做组件对象。组件对象是任何支持COM标准和提供COM服务的二进制对象。通过定义,一个COM对象必须支持可以使自己被其他COM对象引用和当所有的引用都删除后,自己销毁自己的方法。在本书的剩余部分中,除非另外说明,术语组件(component)的意思是一个组件对象。

1.5 服务器程序和客户端程序

服务器程序是提供服务的组件。
客户端程序是使用服务器程序的对象。

COM综合体有两个侧面:COM服务器程序和COM客户端程序。服务器程序是给客户端程序提供服务的组件。这样,客户端程序可以使用服务器程序执行一些任务。在图1-1中,服务器程序由FCAComp组件实现,被Email.exe文档使用。

想像一下,你创建了一个FCA(Fantabulous Compression Algorithm)压缩算法。这个算法可以将文件压缩到文件的1/200,即使其他的算法已经压缩过它们!为了让其他人可以利用自己的算法,你可以把它编译到一个COM服务器程序中。这就可以把算法的功能暴露给其他人,使他们可以利用FCA了。你的客户所写的应用程序是COM客户端程序。例如,如图1-1中所展示的,一个电子邮件程序为了在发送文件到因特网之前进行压缩文件调用了你的组件。

虽然,图1-1显示的客户端程序是一个可执行文件,服务器程序是一个DLL,但它们两个都