

第14章 COM、COM+和ASP

组件对象模型(Component Object Model, COM)很受编程人员的欢迎,许多使用 COM熟练的开发人员认为COM就像ASP一样能够给编程人员灵感。可以说 COM是微软所创造的最优秀的技术之一。就像 ASP一样, COM是任何严谨的 Windows开发者应该了解和掌握的技术。因此,让我们研究一下 COM到底是什么,为什么编程人员这么喜欢它,并且了解 COM和ASP的内在联系,以及如何开发利用它。

令人惊奇的是大多数开发者和最终客户在一些方法、形式上已经不知不觉地运用了 COM。作为一个ASP开发者,几乎一直都在使用着 COM,所有ASP内置的对象都是COM对象。当用到这些对象时,就调用了 COM对象的方法。IIS在很大程度上要用到 COM。几乎每个微软产品都是基于COM或者使用COM。所以不必担心, COM并不是什么新内容而且也不难理解。

在Windows 2000中,已经引入了COM+,“+”标志着对COM做了一些重要的改动,即引入了略有区别的基本编程模型和许多企业级的服务,例如事务管理和有限资源共享。

本章将讨论以下内容:

- COM的内容。
- COM对开发者的意义。
- COM的工作方式。
- COM + 有哪些改动。
- 用VB编写的一个简单的ASP COM组件。

本章用到COM这个词时,指的是对所有的 COM版本都适用的部分,对 Windows 95、Windows 98、Windows NT和Windows 2000都适用。当用 COM + 时,指的是 Windows 2000中的特殊部分。

14.1 COM的内容

COM就是指客户端与COM对象之间交流的一种二进制规范,换句话说,是指它们如何相互对话。

简单地说,COM是基于对象的,可用合适的编程语言把一些代码变成一个组件,而且允许在编程语言中运用这个组件或功能,即使使用的语言不同,也不会出现问题, COM考虑到了这一点。我们不必了解组件程序是如何编写的(用了什么技术、算法等),用的是什么语言,只须将它作为一个“黑盒子”,知道如何使用,如何通过一个或多个接口(相关功能的集合)来访问其功能即可。

例如,你知道微软用什么语言编写 ASP对象模型或ADO吗?我们只需了解有哪些可用的COM对象、方法和属性,就能使用它们的功能。这些方法和属性称为对象的接口。

如果你认为 ASP的CreateObject函数与 CreateCOMObject一样, Server.CreateObject函数与Server.CreateCOMObject一样,则说明你知道你在使用COM了。

COM能实现客户与其正在使用的 COM对象间的透明定位, 客户和组件可运行在不同的进程中, 甚至在不同的机器上。另外COM充分考虑到了机器的远距离连接的需求, 使用一个组件, 不管距离远近, 都用同样的方式。COM提供了完整的基础结构能使所有的组件工作, 你只需把注意力集中在重要的环节上, 例如编程和运用组件。COM对客户和组件开发者的要求是:

- 组件开发者应使用一种能创建 COM组件的编程语言, 例如 VB、VC++, 所创建的COM组件符合COM规范。
- 客户有一种编程语言或工具 (例如Microsoft Word或ASP), 懂得如何实例化和使用 COM组件, 而且要遵守COM规范。像Microsoft Word和ASP这样的工具实际上使用其他组件来完成这种创建, Word用的是VBA, ASP用的是活动脚本(Active Scripting)。

COM + 并未改变COM的基本模型, 只是在环境 (将在下一章进行更详细的讨论) 方面进行了一些扩展, 并且引入了COM + 服务。

上面已经简单地讲了什么是COM和COM + , 其基本原理就是这么简单。

“COM对象”和“COM组件”这两个词经常混用, 虽然这从技术上讲并不准确, 但一般也不会出错。严格地讲, 它们是不同的, 前者是后者的一个实例。

14.1.1 COM无处不在

COM很复杂, 要想了解COM各方面的细节很费时间, 需要大量的阅读和编写代码, 一般来说, 要达到熟练并按对COM和COM + 的理解编写正确的代码, 粗略地算大概需要六个月。然而, 如果一开始就在应用程序中充分利用它, 并且用VB创建COM组件, 正确地练习, 则仅需几个星期。经过一段时间之后, 就能掌握基本要点, 并能善于使用它。因为理解COM确实很费时间, 所以一开始就要确保经常创建一些可处理的范例。

本书不准备介绍COM的全部内容, 也没必要。本章的目的是让读者对COM的关键部分有一个总体的了解, 重点介绍一些对ASP组件开发者们很有用的细节。你如果需要更详细地了解COM, 请参阅以下两本好书。

- 《Beginning ASP Components》, Wrox 出版社出版 (ISBN 1-861002-88-2)
- 《VB COM》, Wrox 出版社出版 (ISBN 1-861002-13-0)。

下面将详细介绍COM, 不必担心刚开始所遇到的任何令人头疼和迷惑的细节, 只要认真学习一切都会解决。

14.1.2 COM + 的三个方面

COM可以分为三个基本部分

- 二进制规范。
- 运行期或库。
- 服务。

1. 二进制规范

COM规范规定了像VB这样的开发工具在把类模块编译成COM时所需遵守的一些准则。ASP这样的技术用它决定如何创建一个组件 (COM对象)的实例并调用它的方法。对开发者们来说, 这个规范也定义了一个跨语言的一致性编程接口。COM规范没限定的是COM自身如何

实现。

在本书印刷的时候，COM 规范在已放在下面的网站上：<http://WWW.microsoft.com/com/resources/comdocs.asp>

所有COM规范都是用一种语言中立方式制定的，这使编写客户端程序所使用的语言与编写COM对象所使用的语言可以不同，只要支持 COM规范即可。请记住，COM是一种二进制规范。

因为COM定义了这个通用的基于对象的通信中介，它真正简化了传统的一些繁重的过程。过去编写客户端/服务器应用程序，必须为不同的客户类型编写多种 API(例如，VC++和VB各一种)。试想在VB这样的程序语言中使用 C/C++的declare语句的功能，VB程序员经常被不必要的内容搞糊涂，而且不得不弄懂 C/C++的结构。然而也许有人会说 declare语句和其他的语言组接机制并不太差，但是COM更简单，采用的是一种更自然和更一致的方式。

2. 运行期或/库

COM运行期就是COM规范的一种实现，并且随各种版本的 Windows提供。尽管运行期的核心API是在ole32.dll中，但其本身存在于许多 DLL中。在开发ASP时，我们时刻都使用着COM运行期和像ole32.dll这样的DLL提供的所有功能。尽管有时我们并不知道，但为了完成任务确实调用了COM运行期。

COM运行期有时被称为COM库。但作者偏爱前者，因为这暗示环境是“执行期间”；有些人喜欢用后者，因为它也是一个函数库。COM运行期是一个主要的内容，希望大家能熟练地运用它。

(1) ASP页面中的COM客户

在ASP页面中，通过ASP解释器间接地使用COM运行期，而ASP解释器在技术上使用活动脚本，这个技术把VBScript、JavaScript和其他脚本语言转换成COM调用，其过程如图14-1所示。

例如，当在ASP页面中使用CreateObject或Server.CreateObject函数时，可用下面的代码：

```
Dim objConn  
  
Set objConn = CreateObject("ADODB.Connection")
```

ASP页面的代码由ASP解释器的活动脚本引擎转换为对COM运行期的调用。COM运行期实际负责创建ADO Connection对象。

(2) VB 中的COM客户

在VB中，使用NEW语句时，就会调用COM运行期。当使用References对话框向一个项目增加ADO支持时，所做的一切就是告诉VB你所要使用的一些COM组件。所选择的引用决定了这些组件，并且允许像使用任何其他VB固有类型一样使用它们：

```
Dim objConn As ADODB.Connection  
  
Set objConn = New ADODB.Connection
```

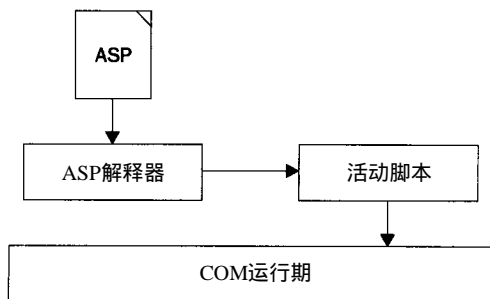


图14-1 ASP页在与COM运行期的关系

作为一位开发者，你发现运行期是 COM 中一个非常有用的部分。COM 规范是 COM 的准则，从细节上规定了所有的方面（我们应该了解它），但我们每天所用的是运行期。

在 VB 中，如果创建的组件不在相同的 DLL 中，new 语句只能转入 COM 运行期，如果是在相同的 DLL 中，则 VB 不需要使用 COM 运行期，这时 VB 可以直创建组件。

(3) 服务

服务是一些编译过的代码（目前只有微软能编写）。它们能提供一些可以增强组件能力的且容易使用的重要功能。如果你用过 MTS，应该用过声明属性和声明安全，通过这两种方法管理员在运行期可以影响组件的行为和使用。

COM + 中的属性是一项描述组件的运行期特性的信息（元数据），可以用组件服务浏览器来设置它们。通过用这种方法设置组件的一种或多种属性，就能控制它的运行期而不需要额外的编程和重新编译，这是重用代码的最优形式。可以充分利用微软编写的基本代码而不是由自己来开发。例如，一个 COM + 应用程序的 Transactions 选项卡，如图 14-2 所示。

在此选项卡中选择 Required 选项，就告诉了 COM + 我们所用的组件要用 COM + 提供的事务服务，即告诉 COM + 每个新的组件实例都应加入到一个事务中。

在第 19 章将要讲到，事务处理能使一组组件做为一个整体单位共同工作。

(4) 没有 COM，ASP 将不再是 ASP

没有 COM，ASP 就不会这么容易扩充，使用也不会如此简单，也许它将不存在。

除非 ASP 是建立在一些相同的规范 / 实现，否则就必须以不同的方式调用不同的组件，而且要根据组件是用什么语言编写的或者是哪个公司提供的。设想一下，为了能引用组件的一些方法，必须弄懂所有方式，这是 C++ 开发者多年以来一直面对的一个问题，到今也没解决。但是 COM 解决了，因为最近许多公司都逐渐适应了 COM。没有 COM，就要花很多时间弄懂组件编写者的意图及不同的 API。应该感谢 COM 提供了一个容易使用和基于对象编程的范型。

微软估计有 300 万开发者利用 COM 开发 COM 应用程序，每天有 2 亿人使用 COM 应用程序，这使 COM 成为世界范围内最成功的对象模型。

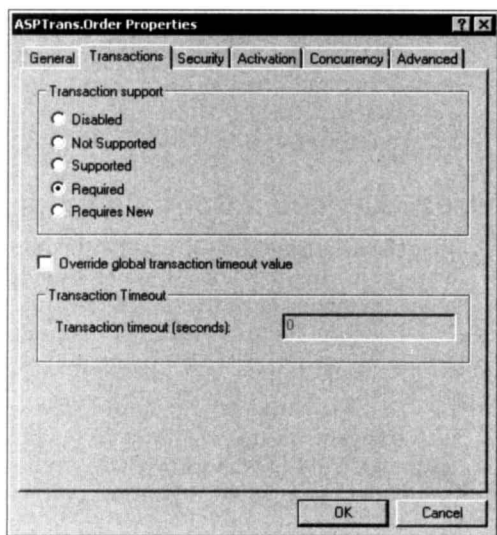


图 14-2 Transactions 选项卡

14.1.3 COM 开发工具

就编写组件的语言而言，微软所提供的主要是 VC++ 和 VB，本书中两者都要讲到。

VC++ 在性能和灵活性的统一上有些缺陷，它使开发者需要深入理解 COM 的内部工作情况。但是，一旦理解了，再使用微软活动模板库 (Active timeplate library, ATL)，编写组件就像用 VB 一样又快又简单。但首先，必须了解相应的内容。

VB是一个很好的折衷方案

如果想不需经过长时间的COM学习就能快速地编写组件，并且对组件的性能要求不高，那么使用VB是一种很好的折衷方案。VB隐含了大部分的COM运行期库，这意味着不需再花时间学习COM，就可以编写COM组件；如果用其他的语言，例如VC++，则必须熟悉COM之后才能创建COM组件。

在VB中，对COM有一点基本的理解就能帮助我们优化组件（通过理解为什么要那样做），这就是我们要介绍更多关于COM的内容的原因。尽管VB隐含了COM的大部分，理解VB如何将其语言结构映射到COM，并且了解VB中COM的限制是很重要的。

VB是ASP的自然发展道路，尤其是对那些习惯于使用VBScript工作的人。本书将用很大的篇幅讲述如何用VB开发组件，同样也会讲述如何使用C++开发组件，这与Java有一些相似之处。我们还要讲述如何用JavaScript开发组件。

本书用VB 6.0作为开发工具，尽管VB 6.0与VB 5.0稍微有一点不同，但对于本书讲述的一些主要内容，用早期的版本一样可以做。

14.2 接口

COM领域的许多人认为COM最重要的和最强大的方面是基于接口的编程。

如果编写代码去完成某项功能的话，其接口就是一系列定义某些东西如何使用的方法。

功能与实现的抽象观念已存在多年。只要有一些面向对象的知识的人都会很快对抽象数据类型(Abstract Data Type, ADT)做一个正确的比较。

基于接口编程就好像控制电视机，遥控器使我们能够通过一个接口来遥控电视，并不需要知道遥控器的内部工作方式，但应知道如何通过遥控器上的各种按钮来控制选台、调节音量和开关电视。

遥控器按钮提供的功能可以由一个接口定义，称为IRemotControl。通过这个接口，各种控制就能实现，遥控器接口具有表14-1所示的方法。

表14-1 遥控器接口具有的方法及说明

方 法	说 明
TurnOnOff	如果电视机已经打开，则关闭；反之亦然
ChangeChannel	转换到指定的频道
IncreaseVolume	增加音量
DecreaseVolume	降低音量
GetChannel	返回目前选定的频道

如果有不同厂家生产的三台电视机，由于使用了接口定义，各个遥控器都能使用上述的方法实现相同的功能，每个遥控器也许内部工作情况不同，但知道如何使用接口（这里的接口就是遥控器上的按钮）就足够了，只要知道面前是哪台电视机，就能通过他的遥控器调换频道，看自己喜欢的节目。

从内部过程看，遥控器上接口的实现是通过另外一个COM接口与电视机交流，这个接口就是ITelevision，这个接口有与IRemoteControl相似的方法。如果每台电视机都实现相同

ITelevision接口，那么一个遥控器就能控制所有的电视机！不管电视机是哪个厂家生产的，只要提供的控制接口相同，遥控器就能控制它。

14.2.1 组件

上面例子中的遥控器就是一个 COM 组件，在 VB 中编译一个包含在一个 ActiveX 项目中的类模块时，就创建了一个组件，如果这个项目含有多个类模块，那么就创建了多个组件。

组件就是通过实现一个或多个接口来提供功能的某种东西。

简单地说，组件就是一个具有唯一名称的功能体，并以某种形式的 DLL 或 EXE 封装或分布。在 VB 中编译一个含有四个类模块的 Activex 对象时，所创建的就是含有四个 COM 组件的 COM 服务器，每个组件对应一个类模块，生成了四个 COM 接口，每个接口对应一个组件，这些接口(类模块)通常提供了能够访问的方法和属性。类模块与 COM 组件的关系如图 14-3 所示。

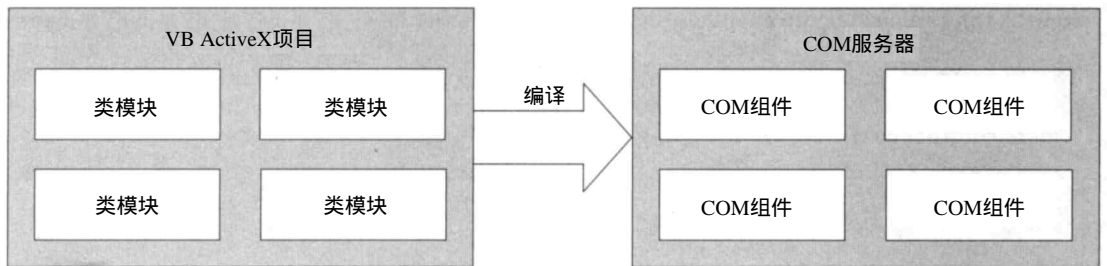


图14-3 类模块与COM组件的关系

棒形图(Lollypop Diagram)

COM 中运用了一种简单的图解方法来表现组件支持的接口，即棒形图。这些棒形图中用一个方框表示组件，方框中的名称就是组件名称，方框左侧伸出的部分表示组件的接口，方框上方伸出的单线表示一个称为 IUnknown 的接口，这是每个组件必须实现的，下面很快就要讲述这个重要的接口。

遥控器的棒形图如图 14-4 所示。

这个图显示了两个组件（电视机），通过 IRemoteControl 接口提供简单的频道变换能力，换句话说就是遥控。这个图并不复杂，但清楚地表示了可以用它来控制电视机。

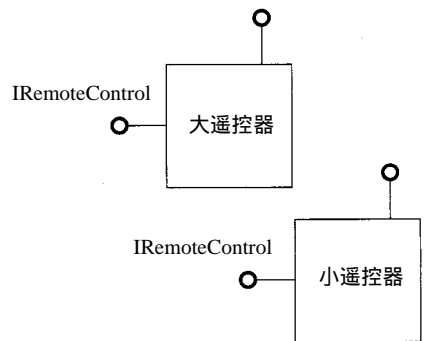


图14-4 遥控器棒形图

IUnknown 接口是唯一从方框上面出去的接口，所以在图上未标注它。

14.2.2 缺省接口

创建 COM 组件时，这个组件可以包含很多接口，COM 允许把其中一个接口设置为客户使用的缺省接口，不能自己选择指定接口的客户将使用缺省接口，其理由下面讲述。

在 VB 中创建一个类模块，并编译成一个组件时，并不能控制哪个接口成为缺省接口，一个缺省接口通常不仅包括所定义类模块，还包括公共的方法和属性。可以通过使用

Implements关键字使得一个类模块支持附加的接口，但缺省时 ASP不能调用这些接口的方法。

在ActiveX项目中创建一个公共类模块时，VB所创建的接口的名称就是类模块的名称前加一个短下划线，例如，MyTV类模块的缺省接口就是_MyTV。

14.2.3 GUID——实体的确定名称

编译一个类模块并创建一个COM组件时，VB就为这件组件赋予了一个全局唯一的标识符，称为GUID，GUID是一个根据系统时钟和网卡的MAC地址生成的一个128位的数字，保证是唯一的。

如果没有网卡，GUID仍然是唯一的，但只能保证在本地计算机中是唯一的。

GUID是一些标识符的专用术语，这些标识符跨时间和空间唯一准确地表示一个实体，由于GUID采用128位数，按每秒增加一千万个GUID的速度，可以使用到公元5770年。

当GUID用来标识一个类模块时，被称为类标识符(CLSID)。在COM中，可以使用GUID来标识很多东西，所以，在不同的环境中GUID有不同的名称，这些环境包括接口标识符(IID)和应用程序标识符(APPID)。

GUID的用途

为什么需要用一个128位数字来标识一个组件？难道是我们给类模块取的名称不够用吗？答案当然否定的。全球所有的分析家和开发人员每时每刻都在给他们的类模块命名（逻辑名称），所以两个人同时使用同一种逻辑名称的情况会经常出现，尤其是当人们为同一个范围的问题设计应用程序时。解决这个问题的办法就是在名字中增加一部分标识符，这并不困难。

128位数字是毫无含义的，也不好使用。人们在注册和其他必须处理GUID的地方用一个字符串来代替GUID。以下的CLSID用来标识(物理名称)用于微软ADO Connection组件的一个组件：

```
{00000293-0000-0010-8000-00AA006D2EA4}
```

这不是一个程序化的标识符(ProgID)，后面将讨论ProgID。

14.2.4 接口的详细内容

正如前面提到的，COM是一种二进制规范，这个二进制规范包括描述一个接口在存储器中的形式以及如何在运行期中访问。

1. 虚拟方法表

定义一个接口时，接口方法的次序、每一种方法的参数和各种其他的属性，例如接口的GUID等，都要通过接口特征来定义。当编译一个包含COM组件的DLL或EXE文件时，接口特征(二进制形式)就被转换进创建的文件，这个信息用来建立虚拟方法表(vtable)，它决定一个客户在运行期如何调用接口的各种方法。为了便于理解，可以把一个vtable看作一个含有各种函数的N维数组，这里的N表示一个接口所含方法的个数，如图14-5所示。

这个数组实际上并不含有代码，但含有能定位代码地址的指针。因此，通过使用vtable，可知下标0的项目指向TurnOnOff方法，下标1的项目指向ChangeChannel方法，等等。

当编译使用一个接口的客户代码时，这些数组下标(例如0, 1, ...)就放置在所创建的DLL或EXE中，这就是早期绑定。客户知道如何通过按给定的偏移量访问某个函数，从而调用接

口中的一个方法。客户不必在运行期查询任何附加信息，只要有接口指针就能进行调用。接口指针是一种指向可以调用的函数的数组的指针。

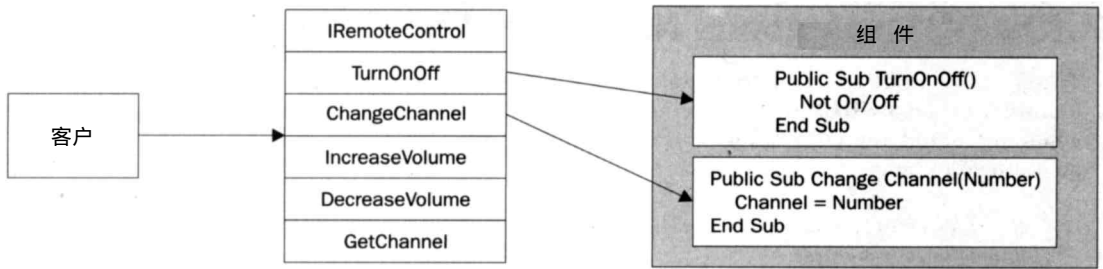


图14-5 虚拟方法表

属性就是函数

提供读写对象的数据(或状态)的能力的接口方法叫做属性。下面讲述的方法与 C++中的方法类似，但与 VB中给出的例子不同。从语义的角度看它们是相同的，VB也是一种很好的工具；但在实现时却不是，VB引入的封装层可能导致人们的误解。

- 只读属性等同于单个接口方法，该方法允许读取一个值。
- 只写属性等同于单个接口方法，该方法允许更新一个值。
- 读/写属性等同于两个接口方法，两个方法分别允许读取和更新一个值。

因此，如果有四个读/写属性，VB就会创建8个方法来读取和更新这四个值。

2. 接口的要素

一般来说，接口通常至少有一个方法，最多 1024个(COM和跨场所调度的限制导致的限制)，一个接口有多少方法是一个设计问题，这个问题是由程序员决定的，可以有一个或者多个，但不是必须有，一个接口可以没有方法。通常一个设计得很好的接口的方法不超过 10到15个。

没有方法的接口是不常见的，但也是有用的。它常用于提供组件和顾客间的一种秘密交流或信号，就像你约好了一没见过面的人，约定他穿着一件特别的衣服，因而当你在人多的场合遇见他时，能很快识别他。客户通过接口能检查确保组件是存在的。

这里的要素化指的是逻辑上把相关的方法一起放到一个接口，因此，如果我说小心接口要素化，意思是你应当特别注意那些放在接口的方法。

3. 接口的原则

从许多方面来看，接口设计与用户界面设计相似。对于用户界面设计问题，需要考虑用户想通过界面做什么，并且使用户非常简单地知道如何做他们想做的事，并且能通过界面去做，而最后一步(做)是最重要的。

不同之处在于，在进行用户界面设计时我们处理的是控件，像文本框和单选按钮，及它们在一个或多个窗体上的布局。对于组件设计，我们处理的是属性和方法，以及它们对一个或多个接口的影响，影响的接口越多，对客户就越有用。

就像用户界面设计，COM接口设计从某种意义上讲是一个基于经验的过程。也许为某一个项目采用一种方式设计，因为它适合这些客户；也许因为有特殊限制或技术上的可能而采用另一种方式设计；不管采用哪种方式，其目的就是让客户满意。

本书不可能对所遇到的每个问题都给出一套实际的接口设计大纲，但设计得好的接口有一些共同特征：

- 确保接口对使用者有用。COM有定义接口的能力，这些接口在语言上或者应用程序上是不友好的，也就是说它们不能运行。当把COM设计成一种二进制标准时，一些语言就比另一些语言功能更强，就像一些浏览器比其他浏览器功能更多一样。如果COM限制这些语言，就不能得到广泛地使用，因此，必须认真考虑接口的性能。例如，一个主要用于像ASP这样的脚本语言环境的组件，应当使用Variant作为参数，并且参数能够输入、修改和输出。这是脚本引擎的一个特征。因此，必须确保接口遵守这些原则并能在这样的环境中正常工作。
- 接口名称(或类模块)和方法名称采用描述性的名称。调用一个叫IDoSomething接口和使用一个叫DoIt的方法没什么意义，这些名称应该有意义，应尽可能说明它在客户的问题中所完成的功能。
- 方法应当很好地要素化，逻辑相关，并不要有太多的方法。如果你已经有一个IChannelSelector的接口，它只应当含有与频道转换有关的方法。例如，如果需要一个FineTuneChannel的方法，它应该包含在IChannelTuner接口中。
- 属性相同。接口的属性应当与客户感兴趣的信息类型一致。
- 接口应当是强类型的。像VB和C++这些语言都是强类型的，也就是说我们所定义的变量属于某一种类型，它只会有这种类型的数据，例如一个数(Long型或Integer型)，如果有人想分配给该变量一个String型数据，这时，编译器就会产生错误。ASP只支持像VBScript这样的弱类型脚本语言，在VBScript中，所有的变量都被定义为Variant型，这种类型的变量可以含有任何类型的数据。把接口定义为强类型的好处是使用时简单明了，可以看到所使用的参数的类型，而不必猜测一个方法能处理什么数据类型。
- 喜欢使用自己的接口。这是一个全球所有成功的公司所遵循的金科玉律：使用自己的接口，或者至少应该确信所使用的接口是个好接口。

4. 接口的不变性

一旦设计好一个COM接口，并且通过某些形式向客户发布了，这个接口就应当被认为是不可变的，不能对其做任何可能影响其二进制表现的改动。

看一下IVolumeControl接口的描述，其中有IncreaseVolume方法和DecreaseVolume方法。如果发布了这个组件，而且人们把这个遥控组件用在他们的应用程序中，并使用了我们的接口及其所有方法。如果我们再决定去掉DecreaseVolume方法会怎么样呢？显然，那些应用程序将被严重破坏，或者将会有一些非常烦人的相关问题，如果改变一个方法参数的数量或者类型，也可能产生同样的问题。

因此，有一些原则需要遵守：

- 在对一个接口满意前，不要发表他。
- 如果实在需要改变一个接口，不应修改已存在的这个接口，而是为所需的扩展功能再创建一个接口。这当然要求客户类型支持多重接口，但ASP目前不支持多重接口。
- 一个接口一旦发布，就不能改变其中的任何一部分，这包括方法的顺序、参数类型等。如果改变了一个已经存在的接口(我们推荐不要这样)，应重新编译所有使用该接口的客户应用程序。就像我们已经讲过的，使用早期绑定的客户将会把接口的设计硬编码为

EXE或DLL，因此，它们必须升级。

- 记住VB隐藏了许多接口信息。因此，当修改或增加一个类模块的公用函数、子程序或属性时，实际上就是在修改缺省接口，VB会自动为一个ActiveX项目中的每个公用类模块创建和管理COM接口。

这些都是好的COM原则，但应用于ASP中的组件更具有灵活性，因为它们使用后期绑定。

14.2.5 IUnknown接口

前面说过每个COM组件都要实现一个叫 IUnknown的接口，这个接口在COM中起着极为重要的作用，有两个作用：

- 引用计数。
- 通过询问支持的接口，动态地揭示接口功能。

1. QueryInterface

QueryInterface是一个方法，通过该方法，可以在运行期动态显示和查询组件的功能。这种方法接受一个接口标识符(IID，另一种类型的GUID)，并且如果此接口被支持的话，就返回被请求的接口，供发出请求的代码使用。

在VBScript中，使用Set关键字调用该方法，这就是查询组件是否支持这个 IID代表的接口。

```
Set RemoteControl = CreateObject("TV.RemoteControl")
```

2. 对象的生存期和引用计数

COM允许传递一个接口指针到应用程序中，并且，当一个接口在使用时，不能破坏提供指针的COM对象。这可以比作一个四方同时通话的电话会议：在会议结束前，电信局应当使电话线保持连接。

为了跟踪接口指针的使用和一个COM对象的生存期，我们使用引用计数。

当一个组件第一次由COM运行期创建时，它的“生命”就开始了，而且 IUnknown接口的 AddRef方法已经被组件自身隐式调用了。AddRef方法的功能很简单，仅将组件引用计数器的值增加1。引用计数器的初值为0，创建组件后，其引用计数器的值为1。

每当QueryInterface函数给顾客提供一个接口指针，就调用 AddRef方法将引用计数器的值加1；相反地，当顾客使用接口完毕，就调用 Release方法，将引用计数器的值减1。如果引用计数器的值达到0，那么对象就知道不再有顾客使用它了，这时，它就自我取消，结束其生存期。

COM对象必须能够可靠地管理自己的生存期。

作为一个非常简单的概念，引用计数是一个强大的功能。然而，在像 C++这样的语言中有点麻烦，因为程序员必须记住手工调用 AddRef和Release，如果调用不配对，对象就永远不会被取消。在VB和ASP中，从不直接调用 AddRef和Release，因此不存在调用不配对问题。

14.2.6 使用IDispatch——后期绑定

IDispatch是一个接口，COM使用它允许客户应用程序在运行期动态地发现和调用组件缺省接口的方法。这一种调用组件功能的方式叫作后期绑定，因为在调用方法之前，组件必须

查询在运行期是否支持它。

在VB中只要声明一个Object类型的变量，就使用IDispatch接口：

```
Dim objRemote as Object

Set objRemote = CreateObject("TV.RemoteControl")

objRemote.TurnOn
```

在ASP中，除了不用Object变量类型而用Server对象去创建实例外，代码几乎是完全相同的：

```
Dim objRemote

Set objRemote = Server.CreateObject("TV.RemoteControl")

objRemote.TurnOn
```

因为ASP中变量是无类型的，也就是说每个定义的变量都是Variant类型的，不能请求一个指定接口。例如，在VB中可以编写以下代码去访问遥控器对象的音量控制接口，并且通过调用IncreaseVolume函数来增大音量：

```
Dim objVolumeControl As IVolumeControl
Dim objRemote As Object

Set objRemote = CreateObject("TV.RemoteControl")
Set objVolumeControl = objRemote

objVolumeControl.IncreaseVolume
```

在ASP中，不能这样做，因为不能请求一个指定的接口。所能做的就是访问组件的缺省接口，因此，为了达到同样的效果，必须用不同的方式去访问这个功能，确实需把组件设计得稍有不同。

不是设计一个有多重接口的组件，而是设计多个组件，每个组件支持一个接口。因此，若最初的设计是使一个组件有四个接口，现在设计四个组件，每个组件支持一个接口，这些组件能一起工作，并能以属性方式相互暴露，以便允许在接口间导航。

例如，有一个叫作VolumeControl的遥控器对象属性，返回一个实现IVolumeControl接口(作为它的缺省接口)的COM对象，一个ASP页面就能使用返回的属性访问接口的方法。程序代码如下：

```
Dim objRemote

Set objRemote = Server.CreateObject("TV.RemoteControl")

objRemote.VolumeControl.IncreaseVolume
```

如果你使用过Microsoft Office对象模型，就会对这种类型代码十分熟悉。这不像使用多重接口技术的VB那么精巧，还需要我们做许多额外的工作，但其有相同的效果。因为随着时间的推移，前面的开发者已经编程创建了多种对象，我们将会发现那些为后期绑定客户设计的组件的每个接口会有更多的方法。

使用多重对象的一种替代方法是作为一个属性返回一个组件的非缺省接口，从而避免对多重对象的需要。这种做法在COM+应用程序中不保险而且违反了COM的

一条基本规则：一个对象只应返回一个接口的一个实现。

后期绑定在有些环境下更灵活，但也有一些缺陷：

- 运行比早期绑定慢，这是因为在调用组件的一个方法之前必须先查询组件是否支持这一个方法，而且IDispatch接口效率也不高。
- 由于是在运行期查询，可能会由于组件不支持某一方法而产生错误。例如，方法的名称拼写错误。

总的来说，在可能的情况下，应该用早期绑定，这在 VB中需要有一个类型库（后面将讨论）。

在ASP中，没有别的选择，只有用后期绑定。

14.2.7 组件信息的中央存储库

前面已经讲了接口和组件，现在再来讨论一下怎样使组件在 VB环境内外都可以使用。

第一个问题是：在 VB中References和Components对话框是如何工作的，从哪里获得组件的信息，如何知道列表框应列出的内容。答案是 Windows的注册表。

在COM世界里，注册表目前起着领导作用，它是一些分层存储的系统数据，用来保存与COM有关的信息和其他内容。在 COM+中的作用稍有不同，组件信息的中央存储库的一般概念相同，但除了遗留组件外，注册表不再是中央存储库了。

应用程序一般不能直接访问注册表，COM运行期提供了API封装组件的使用。

为了进一步观察注册表，启动注册表编辑程序 regedit.exe，运行后可以看到许多键，如图14-6所示。

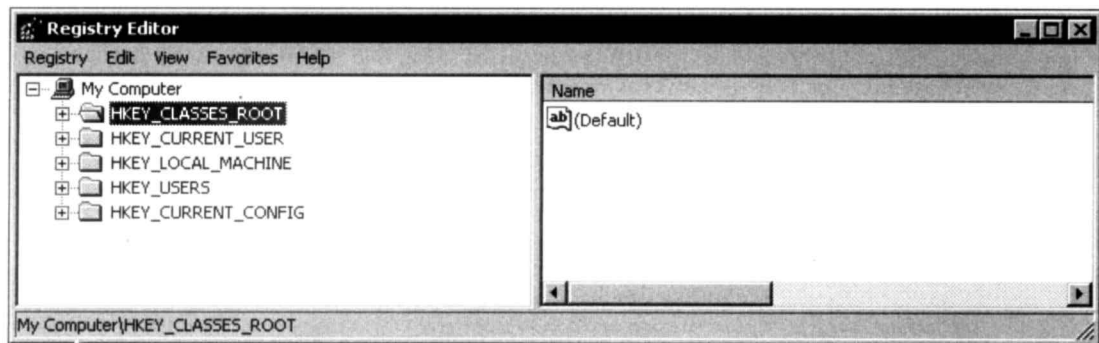


图14-6 编辑注册表的界面

COM的信息存放在HKEY_CLASSES_ROOT(HKCR)键下，这里并不准备详细讨论怎样使用注册表编辑器以及各种各样的键，如果有时间钻研这个问题是十分值得的。但记住，如果使用这种方式，在不能确认所做的事情对系统的影响时，不要删除任何东西。

关于注册表的详细描述和各种与 COM有关的键，请参考 Richard Grimes 的《COM and Registry》一文，该文在 WWW.comdeveloper.com 网站上。

在 VB中编写使用组件和接口的代码时，代码中所使用的名称称为逻辑名称，例如 SomeClass或IRemoteControl。组件和接口的物理名称分别是 CLSID和IID，COM运行期使用

这些标识符(显示在图的上方)创建组件和查询接口,因此最终编译到组件和应用程序中的是这些标识符。

需要重点指出,如果由于某些原因这些标识符发生变化,则客户应用程序不再工作。为了理解在VB中怎样建立这些标识符,需要关注项目设置,尤其是版本相容性。

1. 版本相容性

如果在VB中设置一个ActiveX项目的Project Properties,并且选择Component选项卡,显示的界面如图14-7所示。

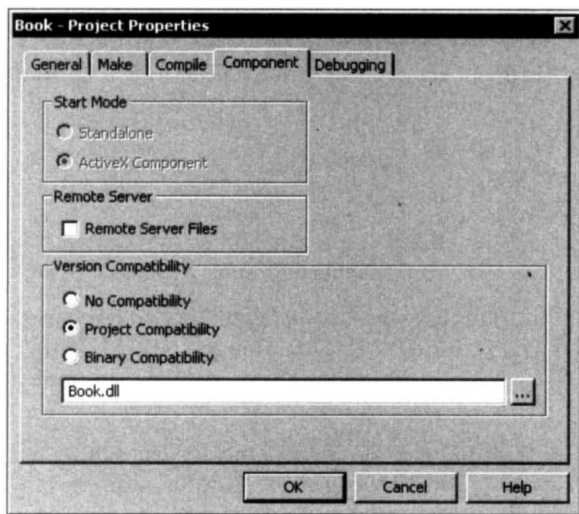


图14-7 Component选项卡

Version Compatibility选项告诉VB如何管理唯一的GUID,这些GUID分配给组件的CLSID和接口的IID。有三个选项,下面分别介绍。

(1) No Compatibility

No Compatibility选项告诉VB在任何时候重新编译应用程序时,都将为组件的公共类、接口和类型库生成一套新的GUID,即使不修改组件本身的接口,也要这样做。这就意味着所获得的组件与使用早期绑定编译的客户应用程序不兼容。

一个类型库是一个描述COM服务器中所有的组件和接口的文件,VB自动建立这一文件并且将它作为COM服务器中的一个绑定资源。在Reference对话框中选择引用时,VB用这个类型库使得各种组件和接口能够在IDE中使用。

(2) Project Compatibility

Project Compatibility选项告诉VB为组件的公共类和接口生成一套新的GUID,但是不修改组件类型库的GUID。

为了使这一选项可以正常工作,必须提供编译过的EXE或DLL的路径,当再次编译这个项目时,VB重新使用存在于此文件中的类型库的GUID。如果不改变这一编译过的类模块的公共接口,VB也将保留定义在一个已编译引用文件中的CLSID和IID。

(3) Binary Compatibility

Binary Compatibility选项告诉VB总是尝试和组件早期版本相容,如果有一点变化就发出警

告。对于每一次重新编译，VB试着再次使用在指定类型库中发现的 CLSID和IID。如果组件的接口发生变化，VB将提示以下三个选项：

- 取消编译，以便返回到代码编辑器中，修改不兼容的语句。
- 打破兼容性，让VB为所有组件、组件暴露的接口和类型库创建一套新的 GUID。
- 保护组件早期版本的兼容性，在这种情况下，VB重新使用旧组件中的所有 GUID，甚至忽略已存在的客户端程序用新的版本不能工作这种情况。这一选项有时不可用，例如删除一个类模块后。

如果打破兼容性，系统将询问改变项目的名称和编译的 EXE或DLL的名字，项目名称将成为类型库的名称，这一过程允许新旧版本的组件共存于同一台机器上，这样使用旧版本组件的客户应用程序将可以像以前那样继续工作。

然而，如果拒绝改变项目的名称和 EXE文件的名字，VB将产生一个新组件，除了含有不兼容的定义的接口外，新组件从旧组件那儿继承所有的 CLSID，这样可以帮助维护那些不愿意修改接口的客户的兼容性，但是一般应该遵照 VB的建议，改变项目和可执行文件的名字。

2. 程序设计标识符

如果你用过CreateObject方法，应该熟悉程序设计标识符(ProgID)，它为创建一个COM组件提供了一个名称。例如：

```
Dim objConnection  
  
Set objConnection = CreateObject("ADODB.Connection")
```

当在VB的一个ActiveX项目中创建一个公用类模块时，产生一个 ProgID，如下：

```
[ProjectName].[ClassModuleName]
```

ProgID是HKEY_CLASSES_ROOT键下的一个子键，提供了一种在运行期从一个字符串标识符查询CLSID的方式。使用ADO Connection组件作为一个例子，可以看到在 ProgID键值下有一个称为CLSID的子键，那里有一个缺省的字符串值，这就是字符串格式的 GUID，如图14-8所示。

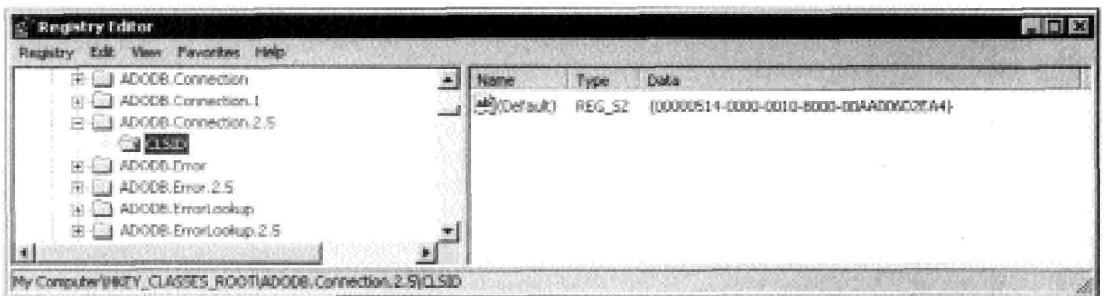


图14-8 注册表编辑器的界面

14.3 COM+运行期的变化

由于提供了大量的能用于增加应用程序功能的新服务，COM+也在COM运行期中包含了一些变化。

- 现在有两种类型的组件：配置的和非配置的。
- 注册表不再是组件元数据的主要信息存储库，而 COM+类别将是主要的信息存储库。

另外，还有许多低层的变化，将在下一章讨论。

14.3.1 配置的和非配置的组件

在COM+中的非配置的组件是指在 Windows 9x或Windows NT环境下创建的组件。它们不使用COM+类型的服务，并且有关它们的信息仅放在系统的注册表中。

COM+类别(注册数据库)里描述了配置的组件，这些组件使用 COM+的服务，例如事务处理支持。

配置的组件在 Windows 2000环境中向后兼容，它们只可用于进程内的服务，并作为一个组放置在一个COM+Application中。如果已经使用了MTS，一个COM+Application与一个软件包基本相同，其作用是作为服务器的替代进程。如果没有使用 MTS，可以认为一个 COM+ Application是进程内 COM服务器的一个集合，进程内是指程序运行在一个可执行文件(dllhost.exe)的地址空间内。这些将在下一章里进一步解释。

14.3.2 COM+类别

在COM+中需要花费很多时间通过 Component Services的图形用户界面(GUI)来管理配置的组件，如图14-9所示。

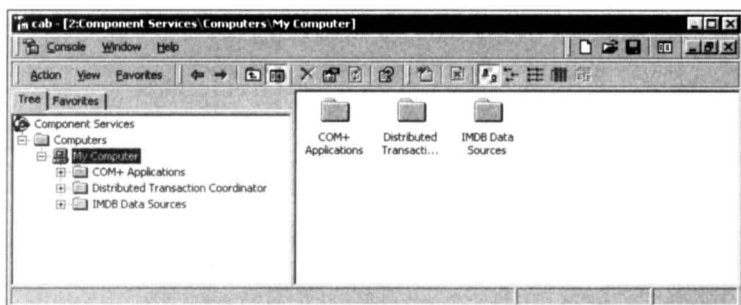


图14-9 Component Services的图形用户界面

使用这个GUI维护的所有与配置有关的组件，都存储在 COM+类别(catalog)中，而不是像传统的COM组件那样存储注册表中。COM+类别提供到注册数据库(RegDB)的一个接口，注册数据库是COM+用来激活COM+组件的优化数据库，与COM在Windows 9x和NT中使用的注册表相似。

如果你以前使用过MTS，COM+类别与MTS类别基本上是同类的东西。

14.4 创建一个ASP COM组件

对于COM和COM+，我们做了相当多的阐述，下面看一下它们的作用，通过快速而简单地建立一个DNA组件来说明为什么VB是开发组件的一个重要工具。

在这个DNA组件中，将是创建一个以数据为中心的中间层组件，能够从 SQL Server 7.0自带的pubs数据库检索一系列书名。这个组件让 ASP开发者不考虑数据库结构，使他们能够获取、增加和删除书名。通过后面几章的讨论，可知这一组件在以用户为中心的组件中是最适合的，但是现在我们将在ASP中直接使用它。

在ASP主页中，应该包含一个数据库内书名的视图，包括下列字段：

- ID：由创建者赋予的唯一的标识符。
- Title：书名。
- Price：书的价格。
- Notes：关于书的简单描述。

页面主界面如图 14-10所示。

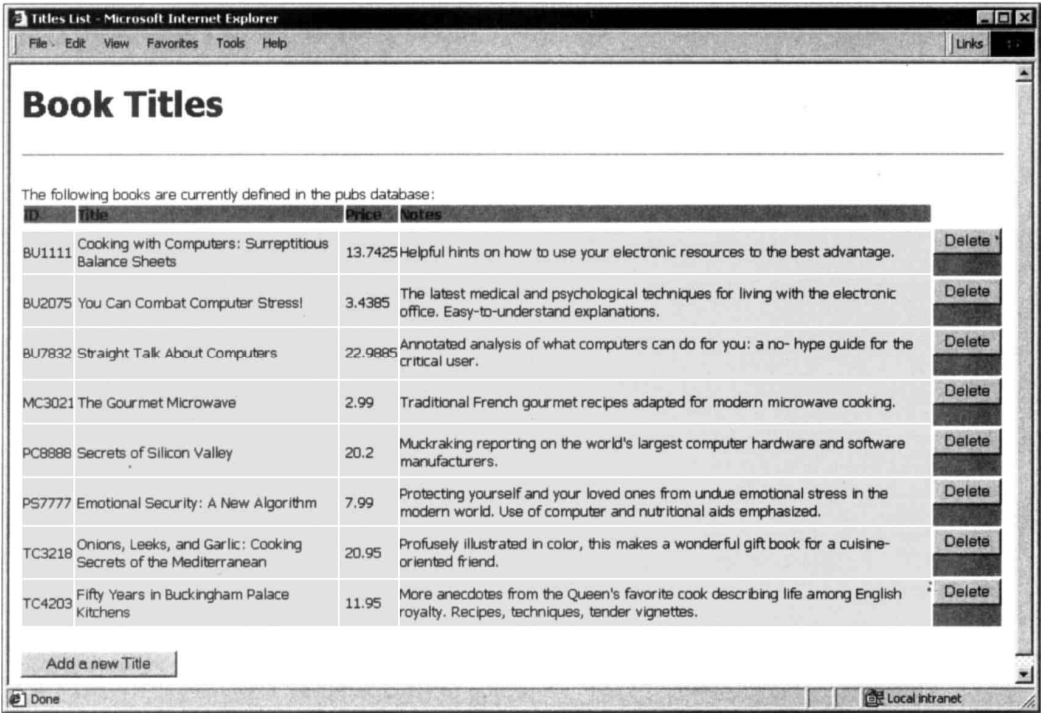


图14-10 页面主界面

单击位于任何一个项目右边的 Delete按钮，将删除这个书名。按了 Delete按钮后，向用户呈现出一个简单的确认删除的界面。如果单击了 Add a new Title按钮，将运行 Add.asp页面，允许用户加入一个新书名，如图 14-11所示。

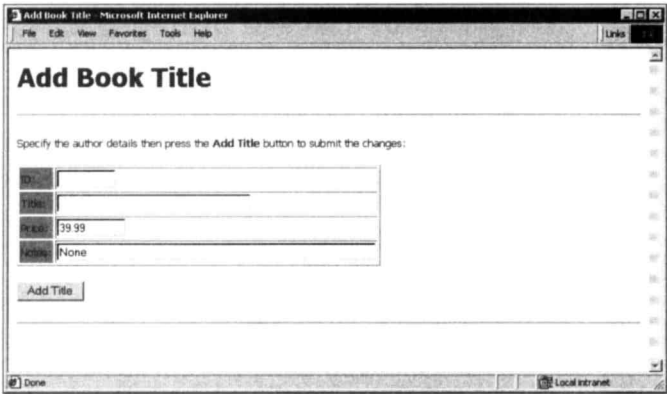


图14-11 Add.asp页面

点击Add Title按钮，将调用Add_UpdateDB.asp页面，使用组件实际完成插入工作。

14.4.1 组件的接口

我们将使用一个十分简单的接口，提供如表 14-2所示的方法。

表14-2 组件接口具有的方法及说明

方 法	说 明
AddTitle	向数据库增加一个新书名
DeleteTitle	根据所指定的书名ID在数据库中删除一个书名
OpenTitles	打开数据连接，对要读书名的数据集初始化
NextTitle	从列表中返回下一个书名
CloseTitles	释放由组件使用的所有资源
IsEOF	指示书名列表是否结束

这些方法遵守前面讲过的接口准则，所有名称都是描述性的，实现这个接口的类模块名称为BookTitles，在VB中将创建一个名为 _BookTitles的接口。

我们要实现的这个组件具有状态，这就意味着任何一个方法的调用返回后都不会使对象失效或破坏。因此由一个方法 (例如OpenTitles)调用设置的成员变量也可以被另外的方法 (例如NextTitle)调用使用。在本例中，组件的状态就是数据库连接和记录集。

组件的实现只是简单地包装 ADO数据库连接和用于访问 pubs数据库的Recordset对象。例如当调用NextTitle方法列举书名时，所做的仅仅是调用 Recordset对象。

也可以采取其他途径。例如，在调用 OpenTitles时列举所有的书名，并将书名的细节存储在内存中。这样做的优点是：较早释放了数据库连接，其他程序可以继续使用这个数据库连接，增加了系统的可扩展性。缺点是随着存储在内存中的信息的增加，增加了管理这些信息的难度，因而也增加了系统内存量的需求、组件的开发时间和复杂性。

14.4.2 创建组件

启动VB，创建一个新的ActiveX DDL项目，把缺省类模块名改为BookTitles，然后再把项目的名称改为Book。由于我们使用ADO，因此需要增加一个对Microsoft ActiveX Data Object 2.5 Library的引用，如图14-12所示。

1. 组件的方法

下面检查需要增加到BookTitles类模块的函数。

(1) OpenTitles方法

OpenTitles方法使用SQLOLEDB OLE DB提供者建立一个与SQL Server 7.0的连接，一旦建立这一连接，就创建一个包含所有书名列表的记录集，这个缺省的记录集使用 Open参数，这意味着不能更新此记录集。代码如下：

```
Option Explicit

Private objConn As ADODB.Connection
```

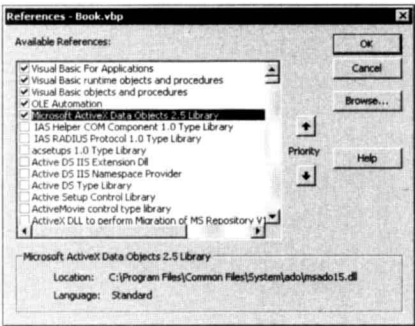


图14-12 增加引用的界面

```
Private mrsBooks As ADODB.Recordset

' Open the connection and the recordset
Public Sub OpenTitles()

    Set mobjConn = New ADODB.Connection
    Set mrsBooks = New ADODB.Recordset

    mobjConn.Open "Data Source=<SERVER_NAME>;" + _
        "Provider=SQLOLEDB;" + _
        "Initial Catalog=pubs;" + _
        "User ID=sa"

    mrsBooks.Open "SELECT * FROM Titles", mobjConn

End Sub
```

(2) CloseTitles方法

CloseTitles方法释放在OpenTitles中打开的Recordset和Connection对象。显示了书名列表后，为了尽可能早地断开与数据库的连接，在书名列表显示完成后调用此方法。尽可能早地释放像数据库连接这样的有限资源总是一个好习惯，list.asp中的程序如下：

```
' Close the list
Public Sub CloseTitles()

' Close the record set
If Not (mrsBooks Is Nothing) Then
    If mrsBooks.State = adStateOpen Then
        mrsBooks.Close
    End If
    Set mrsBooks= Nothing
End If

' Close the connection
If Not (mobjConn Is Nothing) Then
    mobjConn.Close
    Set mobjConn= Nothing
End If

End Sub
```

(3) NextTitle方法

NextTitle方法用于返回当前书名的信息，并通过调用 MoveNext方法前移记录集的指针。代码如下：

```
' Returns the next title
Public Sub NextTitle(Id As Variant, Title As Variant, Price As Variant, _
    Notes As Variant)

' Ignore errors and more specifically null values in this demo
On Error Resume Next

If IsEOF = True Then
    Err.Raise vbObject + 1, "BookTitles Component", "End of cursor"
    Exit Sub
End If

' Copy the fields back to the caller
Id = mrsBooks.Fields("Title_id")
Title = mrsBooks.Fields("Title")
```



```
Price = mrsBooks.Fields("Price")
Notes = mrsBooks.Fields("Notes")
```

```
' Move to the next title
mrsBooks.MoveNext
```

```
End Sub
```

注意，所有的参数都定义为 Variant 型，因为这个函数接收的所有参数都定义在 List.asp 页面中，其代码如下：

```
<%
Dim strID
Dim strTitle
Dim curPrice
Dim strNotes

' Initialize the list
objTitles.OpenTitles

' Process each author
While objTitles.IsEOF = False
    objTitles.NextTitle strID, strTitle, curPrice, strNotes
%>
```

ASP 解释器(在本例中，更确切的说是 VBScript 引擎)把这些参数(strID 等)定义为 Variant。因为 NextTitle 参数是 ByRef 参数，是一个指向含有传递到函数的 Variant 型参数的内存指针，因此，通过函数可以直接更新其值。

如果企图使用 String 型和 Currency 型来定义 NextTitle 函数，就会得到如图 14-13 所示的错误。

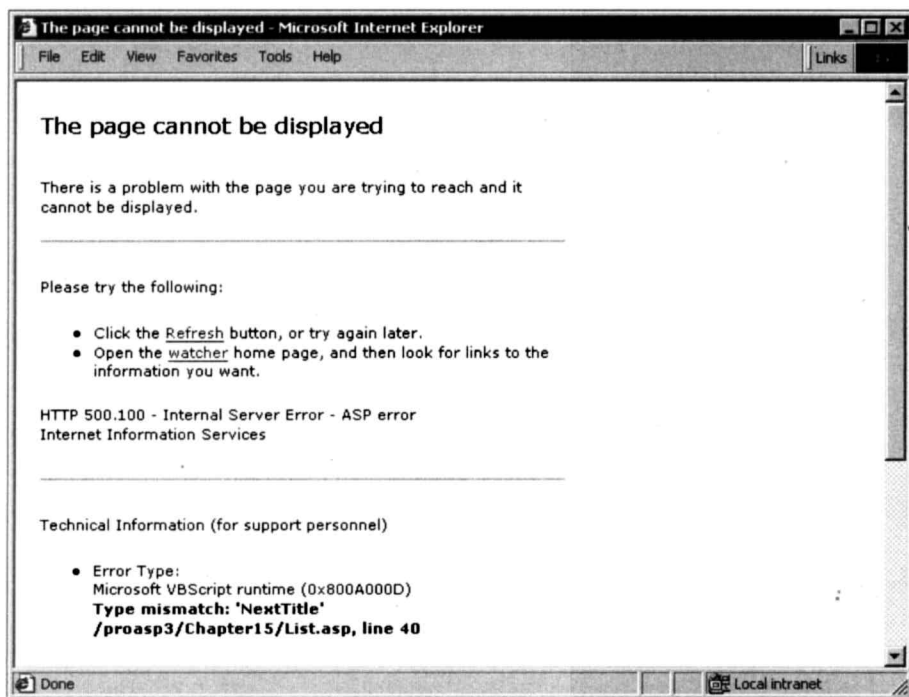


图14-13 显示错误信息的界面

产生这个错误的原因很简单，只是因为一个 ByRef 参数必须接收一个正确类型的指针。如果把参数定义为 ByVal 型，那么参数就是由值传递，活动脚本引擎试着转换值的类型。当然，如果不可能转换，还是会出现同样的错误。

许多商业组件在编写时没考虑到 ASP 有这种类型的问题，所以经常需要为这些组件编写一些小的包装程序。

(4) IsEOF 方法

IsEOF 函数指示是否还有更多的书名，直接反映记录集的 EOF 属性，当枚举所有书名时，用它来确定是否还有书名。代码如下：

```
Public Function IsEOF() As Boolean

    IsEOF = mrsBooks.EOF

End Function
```

(5) AddTitle 方法

AddTitle 方法与一个数据库建立连接，并且打开一个动态的记录集，此记录集可以更新。代码如下：

```
' Adds a new title to the database
Public Sub AddTitle(Id As String, Title As String, Price As Currency, _
    Notes As String)

    Dim objConn As New ADODB.Connection
    Dim rsNewBook As New ADODB.Recordset

    objConn.Open "Data Source=<SERVER_NAME>;" + _
        "Provider=SQLOLEDB;" + _
        "Initial Catalog=pubs;" + _
        "User ID=sa"

    rsNewBook.Open "SELECT * FROM Titles", objConn, _
        adOpenDynamic, adLockOptimistic

    rsNewBook.AddNew
    rsNewBook.Fields("Title_id") = Id
    rsNewBook.Fields("Title") = Title
    rsNewBook.Fields("Price") = Price
    rsNewBook.Fields("Notes") = Notes
    rsNewBook.Update

End Sub
```

在这个程序中，没有重新利用可能已经由 OpenTitles 打开的连接，这样做是为了使程序简单。即不必检查数据库是否已经打开，这里总是将其打开，当此子程序退出时会自动释放数据库连接和记录集，因此，不必将每个对象设置为 Nothing。

读者会注意到，在这个程序中，没有使用 Variant 型参数，这样做是为了证明在原型中不一定非使用 Variant 型，只要不把在 ASP 页面变量表中定义的变量传送给函数。下面的例子说明了在 Add_UpdateDB.asp ASP 页面中如何不使用页面中定义的变量而调用函数：

```
objTitles.AddTitle Request.Form("Id"), Request.Form("Title"), _
    Request.Form("Price"), Request.Form("Notes")
```

因为参数没有被定义为变量，ASP 解释器就会友好地执行必要的转换，为什么它不对页

面定义的变量也做这种自动转换呢？这确实很奇怪，只有寄希望于将来的脚本引擎版本会支持这种转换。

(6) DeleteTitle方法

这个方法用来删除书名和数据库中 Roysched、Sales和TitleAuthor表中有关的行，这很必要，因为pubs数据库结构由SQL Server提供。当建立一个书名时，我们没有在所有表中增加行，但是有可能删除不是通过这个ASPUI创建的书名，为了安全起见，必须这样做。代码如下：

```
' Delete a title from the database
Public Sub DeleteTitle(Id As String)

    Dim objConn As New ADODB.Connection
    Dim rsDelTitle As New ADODB.Recordset

    objConn.Open "Data Source=<SERVER_NAME>;" + _
        "Provider=SQLOLEDB;" + _
        "Initial Catalog=pubs;" + _
        "User ID=sa"

    ' Delete from the various tables
    objConn.Execute "DELETE FROM Roysched WHERE Title_id = '" & Id & "'", , _
        adCmdText + adExecuteNoRecords
    objConn.Execute "DELETE FROM Sales WHERE Title_id = '" & Id & "'", , _
        adCmdText + adExecuteNoRecords
    objConn.Execute "DELETE From TitleAuthor WHERE Title_id = '" & Id & "'", , _
        adCmdText + adExecuteNoRecords
    objConn.Execute "DELETE FROM Titles WHERE Title_id = '" & Id & "'", , _
        adCmdText + adExecuteNoRecords

End Sub
```

一旦把这些函数复制到了VB类模块，编译这个类模块就能创建一个含有此组件的COM服务器，这样做以后，继续我们的工作，编写使用这个组件的ASP页面。这与以前编写其他页面没什么区别，因此不再赘述。

2. 示例的ASP页面

总共有4个ASP页面，它们是：

- List.asp：使用组件显示书名的完整列表。
- Add.asp：提供一个增加新书名的详细信息的简单界面。
- Add_UpdateDB.asp：使用组件和Add.asp传送过来的信息增加一个新的书名到数据库。
- Delete.asp：使用组件在数据库中删除一个书名。

(1) List.asp页面

这个列表页面是一个非常典型的ASP页面，它检索相应的值并显示为一个列表，唯一值得注意的是这个页面并不使用ADD，而是用组件的一个实例：

```
<%
    Dim objTitles          ' Holds the book titles component
    Set objTitles = Server.CreateObject("Book.BookTitles")
%>
```

这里不使用ADO的直接好处是：开发者使用组件，不必担心连接的细节、表的名称和建立SQL查询等问题。

一旦打开列表，程序的主要部分只是简单地枚举每个书名，代码如下：

```

<%
    Dim strID
    Dim strTitle
    Dim curPrice
    Dim strNotes

    ' Initialize the list
    objTitles.OpenTitles

    ' Process each author
    While objTitles.IsEOF = False
        objTitles.NextTitle strID, strTitle, curPrice, strNotes
    %>

```

返回的数据用来创建一个有多行记录的表：

```

<TR>
    <TD bgcolor="#FFFF6C"><%=strID%></TD>
    <TD bgcolor="#FFFF6C"><%=strTitle%></TD>
    <TD bgcolor="#FFFF6C"><%=curPrice%></TD>
    <TD bgcolor="#FFFF6C"><%=strNotes%></TD>
    <TD bgcolor="#3AC2EF">
        <FORM NAME="A" METHOD="POST" ACTION="Delete.asp?id=<%=strID%>">
            <INPUT TYPE="SUBMIT" VALUE="Delete" NAME="B1">
        </FORM>
    </TD>
</TR>

```

注意，这里用一种简单的窗体来为每行提供一个 Delete 按钮，在下面将看到，按下这个按钮将调用 Delete.asp 页面，书名代码用它的 ID 参数传送。

为完整起见，下面给出整个页面的代码：

```

<%
    Dim objTitles          ' Holds the book titles component
    Set objTitles = Server.CreateObject("Book.BookTitles")
%>

<HTML>
<HEAD>
<TITLE>Titles List</TITLE>
<STYLE TYPE="text/css">
    BODY {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}
    TD {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}
</STYLE>
</HEAD>
<BODY BGCOLOR=WHITE>
<H1>Book Titles</H1>
<HR>
<P>The following books are currently defined in the pubs database:

<TABLE cellpadding="2" cellspacing="0">
<TR>
    <TD bgcolor="#3AC2EF"><STRONG>ID</STRONG></TD>
    <TD bgcolor="#3AC2EF"><STRONG>Title</STRONG></TD>
    <TD bgcolor="#3AC2EF"><STRONG>Price</STRONG></TD>
    <TD bgcolor="#3AC2EF"><STRONG>Notes</STRONG></TD>
</TR>
<TR>
</TR>
</TR>

<%
    Dim strID

```

```

Dim strTitle
Dim curPrice
Dim strNotes

' Initialize the list
objTitles.OpenTitles

' Process each author
While objTitles.IsEOF = False
    objTitles.NextTitle strID, strTitle, curPrice, strNotes
%>

<TR>
    <TD bgcolor="#FFFF6C"><%=strID%></TD>
    <TD bgcolor="#FFFF6C"><%=strTitle%></TD>
    <TD bgcolor="#FFFF6C"><%=curPrice%></TD>
    <TD bgcolor="#FFFF6C"><%=strNotes%></TD>
    <TD bgcolor="#3AC2EF">
        <FORM NAME="A" METHOD="POST" ACTION="Delete.asp?id=<%=strID%>">
            <INPUT TYPE="SUBMIT" VALUE="Delete" NAME="B1">
        </FORM>
    </TD>
</TR>

<%
Wend
%>

</TABLE>
<FORM NAME="A" METHOD="POST" ACTION="Add.asp">
    <INPUT TYPE="SUBMIT" VALUE="Add a new Title" NAME="B1">
</FORM>

</BODY>
</HTML>

```

(2) Delete.asp页面

单击Delete按钮去删除一个书名时，调用这个页面，通过下面这几行命令删除书名。

```

<%
Dim objTitles          ' Holds the book titles component
Set objTitles = Server.CreateObject("Book.BookTitles")
objTitles.DeleteTitle Request.QueryString("id")
%>

```

这里显示了整个页面，除了提供一个返回主列表的按钮，没有其他控件。返回主列表后，可能需要刷新画面以查看该书名已从列表中删除了。

整个ASP页面代码如下：

```

<HTML>
<HEAD>
<TITLE>Delete Book Title</TITLE>
<STYLE TYPE="text/css">
    BODY {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}
    TD {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}
</STYLE>
</HEAD>

<BODY BGCOLOR=WHITE>

<H1>Delete Book Title</H1>
<HR>

```



```

<%
    Dim objTitles          ' Holds the book titles component
    Set objTitles = Server.CreateObject("Book.BookTitles")
    objTitles.DeleteTitle Request.QueryString("id")
%>

<P>
<P>Book title deleted succesfully.</P>

<FORM NAME="A" METHOD="POST" ACTION="list.asp">
    <INPUT TYPE="SUBMIT" VALUE="Back to Book Title List" NAME="B1">
</FORM>

</FORM>
</BODY>
</HTML>

```

(3) Add.asp页面

Add.asp页面是一个典型的ASP窗体捕获页面，它让用户输入书名ID、名称、注释和价格，然后把这些细节传送给 Add_UpdateDB.asp 页面。代码如下：

```

<HTML>
<HEAD>
<TITLE>Add Book Title</TITLE>

<STYLE TYPE="text/css">
    BODY {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}
    TD {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}
</STYLE>
</HEAD>
<BODY BGCOLOR=WHITE>
<H1>Add Book Title</H1>
<HR>

<P>Specify the author details then press the <strong>Add Title</strong> button to
submit the changes:</P>
<FORM NAME="CUSTINFO" ACTION="Add_UpdateDB.asp" METHOD="POST" BORDER="1">

<TABLE border=1>
<TR>
    <TD bgcolor="#3AC2EF">ID:</TD>
    <TD> <INPUT TYPE="TEXT" NAME="id" SIZE="8" VALUE=""> </TD> </TR>
<TR>
    <TD bgcolor="#3AC2EF">Title:</TD>
    <TD> <INPUT TYPE="TEXT" NAME="title" SIZE="35" VALUE=""> </TD> </TR>
<TR>
    <TD bgcolor="#3AC2EF">Price:</TD>
    <TD><INPUT TYPE="TEXT" NAME="price" SIZE="10" VALUE="39.99"> </TD> </TR>
<TR>
    <TD bgcolor="#3AC2EF">Notes:</TD>
    <TD><INPUT TYPE="TEXT" NAME="notes" SIZE="60" VALUE="None"> </TD> </TR>
</TABLE>

<P>
<INPUT TYPE="SUBMIT" VALUE="Add Title">
<P>
<HR>
</FORM>
</BODY>
</HTML>

```

(4) Add_UpdateDB.asp页面

根据Add.asp传送来的参数，这个页面用下面两行程序增加了一个新书名：

```
<%  
    Dim objTitles          ' Holds the book titles component  
    Set objTitles = Server.CreateObject("Book.BookTitles")  
    objTitles.AddTitle Request.Form("Id"), Request.Form("Title"),  
                        Request.Form("Price") , Request.Form("Notes")  
%>
```

再次看到，通过使用组件，开发者的工作变得比较简单。整个页面代码如下：

```
<HTML>  
<HEAD>  
<TITLE>Add Book Title</TITLE>  
<STYLE TYPE="text/css">  
    BODY {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}  
    TD {font-family:Verdana,Tahoma,Arial,sans-serif; font-size:10pt}  
</STYLE>  
</HEAD>  
<BODY BGCOLOR=WHITE>  
  
<H1>Add Book Title</H1>  
<HR>  
  
<%  
Dim objTitles          ' Holds the book titles component  
Set objTitles = Server.CreateObject("Book.BookTitles")  
objTitles.AddTitle Request.Form("Id"), Request.Form("Title"),  
Request.Form("Price") , Request.Form("Notes")  
%>  
  
<P>Book title added successfully.</P>  
  
<FORM NAME="A" METHOD="POST" ACTION="List.asp">  
    <input type="SUBMIT" value="Back to Book Title List" name="B1">  
</FORM>  
  
</FORM>  
</BODY>  
</HTML>
```

14.5 小结

COM+是Windows 2000中提供一种功能非常强大的技术，除非你喜欢违背潮流，否则就应该在产品中使用该技术，这样使客户能够通过你所创建的组件和接口来访问你开发的功能。

正如你所见，在VB中创建组件很简单，真正所需要注意的是编写代码实现我们想让客户使用的功能。这些客户可以是ASP页面、任何其他能使用COM的工具或编程语言。

COM能满足所有语言的需要，由于有些语言的功能远比另一些语言强大，因此必须了解那些可能使用你的组件的客户的限制，例如对于 ASP客户，只有当一个函数的参数定义为Variant时，在ASP页面中定义的变量才能传送给这个函数，我们也看到 ASP实际上只能使用组件的缺省接口。

在后面几章，将更深入研究COM+对运行期的进一步改进，并且讨论如何使用一些COM+服务。