# COM中自动化对象的方法调用探析

## Study on Method Calling of Automation Object in COM

摘　要：本文主要从内部实现机理上对COM中自动化对象的方法调用进行了详细的探析，并结合一个相应的应用实例，具体说明了在访问自动化对象方法时几种不同的方法。

关键词：COM 自动化 自动化对象

唐国维 肖勇军 程丹 （大庆石油学院计算机科学与工程系 163318）

### 1 引言

组件对象模型（Component Object Model，即COM）是由Microsoft提出的组件标准，它不仅定义了组件程序之间的交互标准，而且也提供了组件程序运行所需的环境。在COM中，客户与组件之间的通信一般都是通过COM接口直接来完成的，而自动化又为我们提供了一种COM客户控制组件的方法，它使得用解释性语言和宏语言访问COM组件变得更加容易。本文主要是从自动化的角度出发，根据笔者长时间钻研COM所得出的一些心得，详细说明了COM中自动化对象的方法调用问题，其主要目的是给读者提供一些帮助。

### 2 自动化与自动化对象

自动化（Automation）是应用程序或动态链接库（DLL）显露可编程对象给其他应用程序的手段，也就是从一种应用程序的内部自动控制另一个应用程序的方法。显露可编程对象的应用程序或动态链接库（DLL）称为自动化服务器，访问和控制自动化服务器的应用程序称为自动化控制器，也就是自动化客户。自动化服务器按照与其客户所处位置的不同，大致可分为两类：进程内自动化服务器和进程外自动化服务器。

自动化不是独立于COM，而是建立在COM基础上的。一个自动化服务器实际上就是一个实现了IDispatch接口的COM组件，而一个自动化控制器则是一个通过IDispatch接口同自动化服务器进行通信的COM客户。同样，自动化对象（Automation Object）在本质上是一种实现了IDispatch接口的COM对象，它有两个基本特性：属性和方法。属性是指自动化对象的数据特征，而方法则是自动化对象所能提供的功能服务。当要访问自动化对象的方法时，自动化控制器不会直接调用自动化服务器实现的各种方法，而是通过IDispatch接口中的成员函数实现服务器中各种方法的间接调用。下面，笔者就着重谈一下COM中自动化对象的方法调用问题。

### 3 COM中自动化对象的方法调用

自动化为客户和组件提供了一种新的通信方式，而IDispatch是实现自动化的标准COM接口。因此，有了IDispatch接口之后，COM组件就可以通过一个标准的接口提供它所支持的服务，而无需提供多个特定于服务的接口。在Delphi提供的System.pas文件中，IDispatch接口是这样定义的：

```
IDispatch = interface ( IUnknown )
    ['{0020400-0000-0000-C000-000000000046}']
    function GetTypeInfoCount ( out Count: Integer):HResult;stdcall;
    function GetTypeInfo ( Index,LocaleID: Integer;out TypeInfo):HResult;stdcall;
    function GetIDsOfNames ( const IID: TGUID;Names:Pointer;NameCount,LocaleID:
        Integer;DispIDs:Pointer):HResult;stdcall;
    function Invoke ( DispID:Integer;const IID: TGUID;LocaleID:Integer;Flags:Word;
        var Params;VarResult,ExcepInfo,ArgErr: Pointer):HResult;stdcall;
    end;
```

#### 3.1 通过COM接口调用方法

COM接口是COM对象与外界交互不可缺少的部分，它是一个指针，通过指向一个虚方法表（Virtual Method Table，即VMT）而间接指向COM对象，该虚方法表包含了COM对象的部分或全部方法。因此，当通过COM接口来调用自动化对象的方法时，实际上就是直接访问COM对象的VMT，而并不需要通过调用IDispatch接口中的成员函数来实现自动化对象中各种方法的间接调用。另外，当通过COM接口来控制自动化服务器时，它采用的是“前期捆绑”（early binding），也就是说对接口方法的所有调用在编译时检查参数是否正确，而不是等到运行时才进行参数校验。这样在运行时，因为对象的入口点已经找到，数据类型和语法已经得到验证，所以它的执行速度很快。因此，在大多数情况下，笔者更倾向于这种方法。

### 3.2 通过Variant调用方法

采用Variant来调用自动化对象的方法，实际上是通过调用IDispatch接口中的成员函数来间接完成的。而所谓的“后期捆绑”（late binding），就是指通过IDispatch.Invoke调用的自动化方法，之所以称为“后期捆绑”，是因为方法调用并捕获错误都是在运行期确定的，在编译期，只是将方法的有关参数传递给IDispatch.Invoke，而不用编译语法和进行类型校验。因此，通过Variant调用自动化对象的方法采用的是“后期捆绑”。

笔者现就说明一下采用此方法调用的过程中，具体发生了哪些事情。如图1所示，首先把要调用的方法名传递到IDispatch.GetIDsOfNames，然后通过该过程返回代表此方法名的调度标识符（即DispID），最后使用返回的调度标识符来调用IDispatch.Invoke方法，通过IDispatch.Invoke方法的实现完成了调用要访问的COM对象的方法。在图1中，Add、Reduce、Devide三种方法指的是要调用的自动化对象的方法，它们是为了说明问题而任意定义的。通过Variant来访问自动化对象的每种方法都存在上述步骤，执行这三步所需的时间比直接通过一个COM接口访问同一个对象的时间要长许多。因此，IDispatch接口比其他接口访问速度要慢。但是，IDispatch却是一个在自动化中具有举足轻重作用的重要接口。它对于Visual Basic是大有用处的，这是因为VB实际上并不是一种完全面向对象的编程语言，因此它并不具备COM的所有优点，但正是IDispatch将VB带进了COM中，它可以允许程序员们不使用对象也可以访问COM接口。同样，许多不支持接口的应用程序（如Word和Excel等），也可以通过IDispatch来调用COM接口。

因此，虽然IDispatch接口相对于其他接口来说访问速度要慢，但IDispatch在以下两个方面还是非常有用的：

（1）当想要找到一种迅速、简单的方法来访问一个自动化对象，但又不想费力去输入一个类型库时，这时可以采用IDispatch接口。

（2）当想要确保所创建的COM对象可被第三方开发者使用时，具体来说，就是想要自己的代码从VB或其他不支持真实对象的应用程序里也可以调用时，可以采用IDispatch接口。

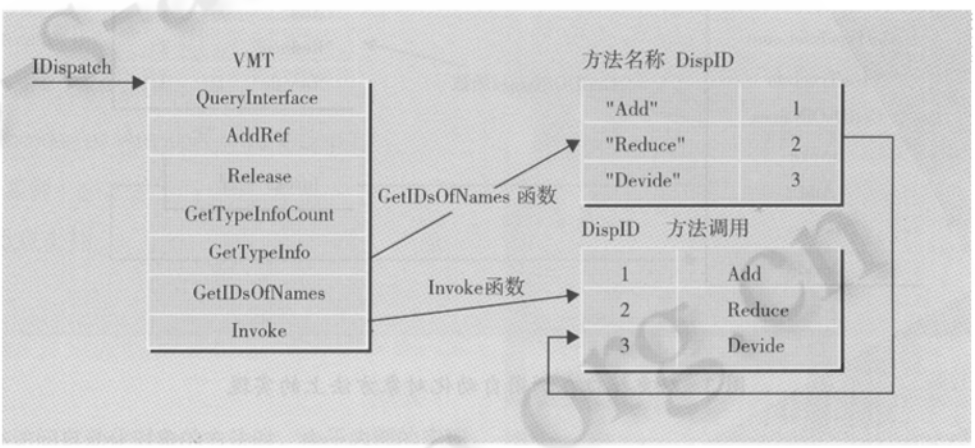


图1 IDispatch调用自动化对象方法的一种实现

### 3.3 通过派遣接口（DispInterface）调用方法

DispInterface支持一组可以通过IDispatch访问的方法和属性，任何能通过IDispatch访问其方法和属性的对象都提供DispInterface。笔者上面已经提到，通过IDispatch接口访问一个方法时必须对对象做两个调用，第一个调用得到要使用的调度标识符DispID，第二个调用用来实际访问方法。而DispInterface是具有通过IDispatch来实现的对象DispID的全部知识的接口。换句话说，DispInterface已经知道所需要的DispID，因此它在调用IDispatch.Invoke之前不需要再进行查询，即不用调用IDispatch.GetIDsOfNames过程，这样就加速了使用IDispatch接口的进程，但这一进程仍不如通过虚方法表（VMT）直接调用方法那样迅速，图2就为读者形象地描述了通过DispInterface调用自动化对象方法的具体实现。另外，虽然通过DispInterface来调用方法是在编译期得到方法的DispID，但是方法的调用还是要在运行期确定，只是这样避免了在运行期为了调用一个方法而去调用IDispatch.GetIDsOfNames过程。因此，通过DispInterface调用方法还是属于“后期捆绑”，只不过它是“后期捆绑”的一种优化措施，这种优化措施常被称为“ID捆绑”。既然它是“后期捆绑”的一种，那么在调用方法的速度上就要比“前期捆绑”要慢一些。因此，在调用方法的执行速度上，采用COM接口最快，派遣接口次之，而采用Variant则速度最慢。

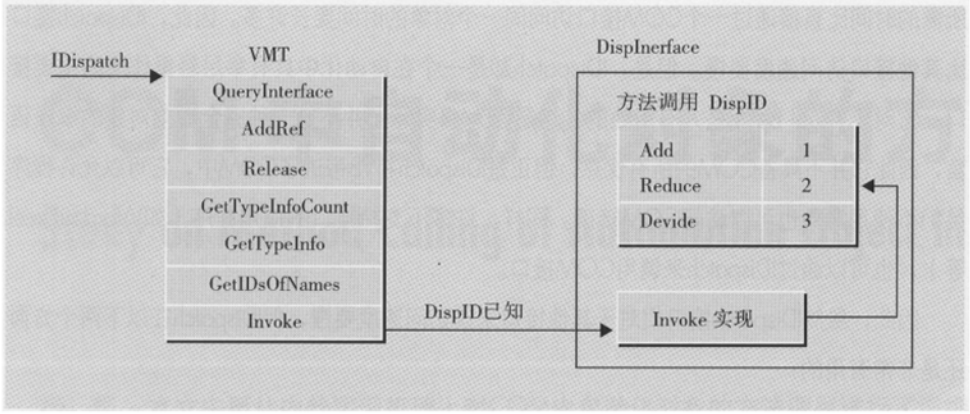


图 2 DispInterface 在调用自动化对象方法上的实现

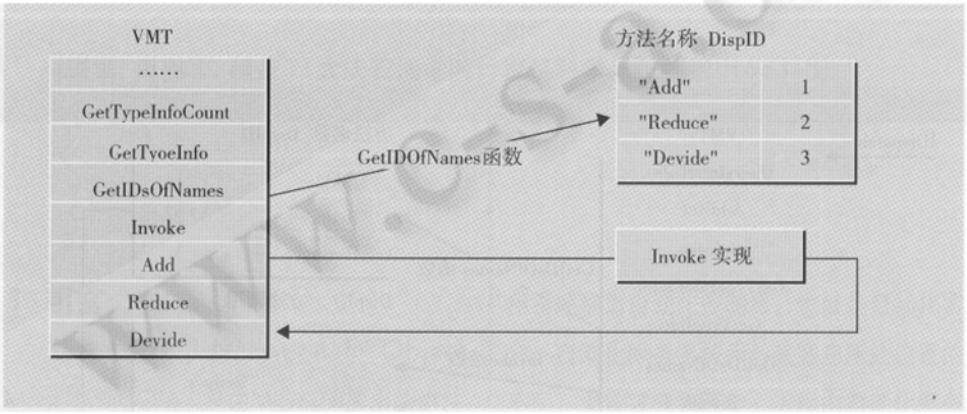


图 3 双重接口在调用自动化对象方法上的实现

### 3.4 双重接口（Dual Interface）

综上所述，由于采用COM接口在访问自动化对象的方法时，其执行速度很快，因此Object Pascal和C++的程序员仍主要依赖于COM接口并通过虚方法表（VMT）来调用对象的方法，但IDispatch和DispInterface对于诸如VB、Word、Excel等的开发者们是极有用处的。显然，二者的需要在利益上出现了冲突。基于此，COM采用了双重接口来解决这一问题。双重接口可以使一个单独的接口同时支持IDispatch接口和用户自定义的接口，它可以让那些不支持真实对象的语言（如VB、Microsoft Word、Excel等）通过IDispatch来访问COM对象，同时也可以让Object Pascal和C++的程序员通过虚方法表（VMT）来调用对象的方法。因此，这就意味着读者编写的任何自动化服务器几乎可以被目前市场上的任何软件所访问。

双重接口是一种从IDispatch继承而来的COM接口，因此，在它的虚方法表（VMT）中不仅包含IDispatch的所有方法，而且还包含你想要调用的自动化对象的方法。如图3所示，当想要调用自动化对象的方法时，你可以直接访问对象的虚方法表（VMT）来调用其方法。另外，你也可以调用IDispatch接口中的成员函数来实现自动化对象中各种方法的间接调用，只不过IDispatch.Invoke的实现方法是根据DispID的值来调用用户自定义接口上的相应方法。

## 4 应用实例

下面笔者将结合上面所述的方法给出一个具体的应用实例。该实例是用Delphi实现的一个小程序，它的主要功能是实现各种面积单位的换算，面积单位主要有平方米、平方厘米、平方千米、平方英尺、平方英里和英亩。笔者主要是借这个例子来说明一下COM接口、Variant和DispInterface在调用自动化对象的方法时的具体实现，并进一步验证采用COM接口在访问自动化对象方法时执行速度是最快的。

### 4.1 创建自动化服务器

在此实例中，笔者选择的是进程内自动化服务器模式。在该自动化服务器中，定义了一个自动化对象AreaunitConverter，同时该自动化对象又包含一个方法Convert，此方法的主要作用就是实现在各种面积单位中进行换算，该方法的具体实现如下所示：

```
function TAreaunitConverter.Convert
(Quantity: Double; Inunit,Outunit: SYSINT):
Double;
const areafactor:array[auSquareMeters..
auAcres] of double = (10000.0,1.0,
10000000000.0,
929.0304,6.4516,25899881103.4,
40468726.0987);
begin
result:=Quantity*areafactor[Inunit]/
areafactor[Outunit];
end;
```

在注册完自动化服务器后，下一步就是要建立客户程序来访问自动化服务器。

### 4.2 建立客户程序并访问自动化服务器

由图4可以看到，此客户应用程序主要由六个按钮组成，左列按钮主要是通过COM接口、Variant以及DispInterface来访问自动化服务器，右列按钮则用来测试这三种方法在调用一次自动化对象方法时的运行时间。如图4所示，这三种方法的执行结果是完全一致的，只不过在运行时间上，Variant方法的调用时间最长，它几乎是COM接口调用的27倍，而通过DispInterface调用的运行时间居于二者之间。因此，COM接口是到目前为止在

一个自动化服务器上调用一种方法的最快的途径，Variant慢一些，而派遣接口介于二者之间。

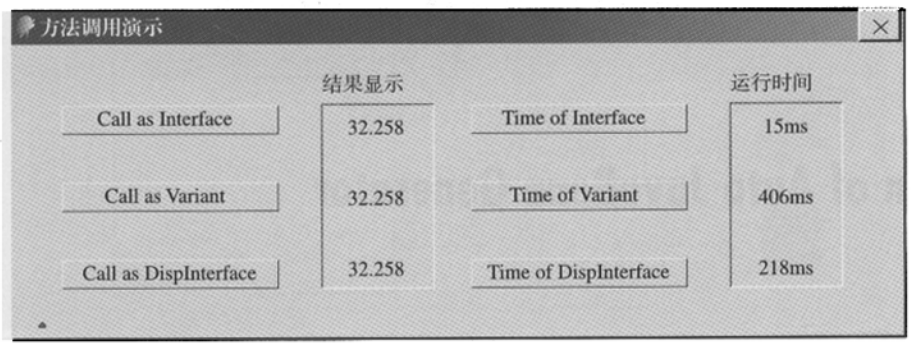


**图 4　客户应用程序的运行界面**

以下是在调用自动化对象的方法时上述三种方法的部分实现代码，以供读者参考。

```
myintf:=CoAreaunitConverter.Create; 或
myintf:= CreateComObject(Class_AreaunitConverter) as IAreaunitConverter;
myvar:=CreateOleObject('UnitSrv.AreaunitConverter');
mydisp:=CoAreaunitConverter.Create as IAreaunitConverterDisp; 或
mydisp:=CreateComObject(Class_AreaunitConverter) as IAreaunitConverterDisp;
（通过以上三种方法创建自动化对象的一个实例）
callintfLabel.Caption:=floattostr(myintf.convert(5.0,4,1));
（自动化对象的方法调用）
```

## 5 结论

综上所述，COM接口、Variant和派遣接口在访问自动化对象的方法时，由于内部的实现机理不同，因此它们都有最适合自己的应用环境。Object Pascal和C++的程序员仍主要依赖于COM接口来快速的调用对象的方法，而对于VB、Word等的开发者们，Variant和DispInterface却是非常有用的。因此，当读者在访问自动化对象的方法时，可以根据自己的实际需要选择相应的方法，这样就可以避免走弯路。


参 考 文 献

1 陈旭等译，Delphi COM 深入编程，机械工业出版社，2000.10。

2 杨秀章译.COM 技术内幕，清华大学出版社，1998.12。

3 潇湘工作室译，Delphi4 编程技术内幕，机械工业出版社，1999.6。