注意，在传递给 GetData 的 FORMATETC 结构中使用了多个 TYMED 标志。多个 TYMED 标志可以以这种方式通过“或”运算组合在一起来形成一个数据对象，从而使得调用者可以在多种不同的存储媒体中接收数据。

19.2.5　多种格式和多种存储媒体

数据提供者可以以必要的次数多次调用 CacheData 或 CacheGlobalData，使多种格式的数据对数据使用者都有效。下面的程序提供了两个格式下的项目：一个是用::RegisterClipboardFormat (nFormat)注册的私有格式，另一个是 CF_TEXT 格式：

```
COleDataSource * pods = new COleDataSource;
pods -> CacheGlobalData (nFormat, hPrivateData);
pods -> CacheGlobalData (CF_TEXT, hTextData);
pods -> SetClipboard ();
```

您也可以使不同存储媒体中相同格式的多个数据项目有效。假设您想使 CF_TEXT 数据在全局内存块或文件中都有效，并且已经初始化了 pwszFileName 来指向文件名称(用 Unicode 字符表示)，那么下面就是您要做的工作：

```
FORMATETC fe = {
    CF_TEXT, NULL, DVASPECT_CONTENT, -1, TYMED_FILE
};

STGMEDIUM stgm;
stgm.tymed = TYMED_FILE;
stgm.lpszFileName = pwszFileName;
stgm.pUnkForRelease = NULL;

COleDataSource * pods = new COleDataSource;
pods -> CacheGlobalData (CF_TEXT, hTextData);        // TYMED_HGLOBAL
pods -> CacheData (CF_TEXT, &stgm, &fe);             // TYMED_FILE
pods -> SetClipboard ();
```

多次调用 CacheData 和 CacheGlobalData，然后将数据对象放在剪贴板中，这类似于多次调用::SetClipboardData将两个以上的格式放在传统剪贴板中。但是传统剪贴板不会接受两个相同格式的项目。OLE 剪贴板却可以——只要每个 FORMATETC 结构具有唯一的 tymed 值，也就是说项目保存在了不同类型的存储媒体中。

19.2.6　检查数据有效性

API 函数::IsClipboardFormatAvailable 允许传统剪贴板的用户查找某种格式的数据是否有效。COleDataObject::IsDataAvailable 则允许 OLE 剪贴板的用户这样做。下列程序段检查 CF_TEXT 数据在 HGLOBAL 中是否有效：

```
COleDataObject odo;
odo.AttachClipboard();
if (odo.IsDataAvailable(CF_TEXT)) {
    // CF_TEXT is available in an HGLOBAL.
}
else {
    // CF_TEXT is not available in an HGLOBAL.
}
```

要检查存储媒体类型而不是全局内存,您只要初始化 FORMATETC 结构并将其地址传递给 IsDataAvailable 即可,如下所示:

```
COleDataObject odo;
odo.AttachClipboard();

FORMATETC fe = {
    CF_TEXT, NULL, DVASPECT_CONTENT, -1, TYMED_ISTREAM
};

if (odo.IsDataAvailable(CF_TEXT, &fe)) {
    // CF_TEXT is available in a stream object.
}
else {
    // CF_TEXT is not available in a stream object.
}
```

您也可以通过"或"运算将几个 TYMED 标志组合在传递给 IsDataAvailable 的 FORMATETC 结构的 tymed 字段内。此时只要在被请求的存储媒体中数据是有效的,就会返回非零值。

注意,在 MFC 6.0 中由于出现了错误,因此如果请求的信息在任意存储媒体下都有效,那么 COleDataObject::IsDataAvailable 有时会返回一个非零值。实际上,传递给 IsDataAvailable 的 FORMATETC 结构中的媒体类型信息被忽略了。值得注意的是,该错误只有在附属于 OLE 剪贴板的 COleDataObject 上调用 IsDataAvailable 时才出现,并且它对一些数据类型(特别是 CF_TEXT 数据)的影响要大于对其他类型的影响。在 COleDataObject 用来实现 OLE 拖放目标时,IsDataAvailable 会按所介绍的那样执行。

数据使用者可以使用 COleDataObject 函数 BeginEnumFormats 和 GetNextFormat 来枚举多种有效的格式。下列程序代码枚举了 OLE 剪贴板上所有有效的格式:

```
COleDataObject odo;
odo.AttachClipboard();

FORMATETC fe;
odo.BeginEnumFormats();
```

```
while (odc.GetNextFormat (&fe)) {
    // FORMATETC structure describes the next available format.
}
```

如果一个特定的数据格式在两个以上的存储媒体类型中有效，那么 GetNextFormat 可能会用标识每种存储媒体类型的位标志来初始化 FORMATETC 结构的 tymed 字段，或者是为每个 tymed 返回唯一的 FORMATETC 结构。但是一个有趣（也可能是令人讨厌）的异常情况会发生。如果 OLE 用相同的 cfFormat 不同的 tymed 包含了两个数据项目，GetNextFormat 将只会为其中的一个返回信息。在系统提供的剪贴板数据对象（其 IDataObject 指针是由 ::OleGetClipboard 返回的）中这就是错误。如果需要知道给定的剪贴板格式适用于哪种媒体类型，可以使用 IsDataAvailable 去查询单个剪贴板格式和存储媒体的组合。

19.2.7　用 COleDataSource 进行延时再现

OLE 剪贴板支持延时再现吗？扼要的回答是“是的”，虽然实际上是 MFC 的 COleDataSource 实现而不是 OLE 剪贴板使得延时再现工作的，看一下 COleDataSource 内部就会明白原因了。

COleDataSource 对象的首要任务是作为一个数据高速缓冲区。在内部，它维持着一个描述当前有效数据的 FORMATETC 和 STGMEDIUM 结构的数组。当应用程序调用 CacheData 或 CacheGlobalData 时，带有描述存储媒体类型的 tymed 值的 STGMEDIUM 结构就会被添加到数组中。但是如果应用程序调用 DelayRenderData，一个包含值为 NULL 的 tymed 的 STGMEDIUM 结构就会添加到数组中。当请求获取该数据时，COleDataSource 会看到 tymed 值为 NULL，从而知道数据是要通过延时再现提交的。COleDataSource 通过调用虚函数 OnRenderData 来作出响应。您的工作就是在派生类中覆盖此函数，以便可以提供要求的数据。

下面是一个例子，说明了使用延时再现把一个位图放置在 OLE 剪贴板上的方法。第 1 步要做的是复制位图并将其保存在一个文件中。（您可以把它保存到内存中，但那样的话就违背了使用延时再现的主要目的了。）第 2 步要调用 DelayRenderData：

```
FORMATETC fe = {
    CF_BITMAP, NULL, DVASPECT_CONTENT, -1, TYMED_GDI
};

CMyDataSource* pmds = new CMyDataSource;
pmds->DelayRenderData (CF_BITMAP, &fe);
pmds->SetClipboard ();
```

CMyDataSource 是 COleDataSource 的派生类。当数据源得到提交位图的请求时，OnRenderData 函数就将位图传送到 TYMED_GDI 存储媒体中：

```
BOOL CMyDataSource::OnRenderData (LPFORMATETC lpFormatEtc,
    LPSTGMEDIUM lpStgMedium)
```

```
{
    if (COleDataSource::OnRenderData(lpFormatEtc, lpStgMedium))
        return TRUE;

    if (lpFormatEtc->cfFormat == CF_BITMAP &&
        lpFormatEtc->tymed & TYMED_GDI) {

        // Re-create the bitmap from the file, and store the
        // handle in hBitmap.
            .
            .
            .
        lpFormatEtc->cfFormat = CF_BITMAP;
        lpFormatEtc->ptd = NULL;
        lpFormatEtc->dwAspect = DVASPECT_CONTENT;
        lpFormatEtc->lindex = -1;
        lpFormatEtc->tymed = TYMED_GDI;

        lpStgMedium->tymed = TYMED_GDI;
        lpStgMedium->hBitmap = hBitmap;
        lpStgMedium->pUnkForRelease = NULL;

        CacheData(CF_BITMAP, lpStgMedium, lpFormatEtc);
        return TRUE;
    }
    return FALSE;
}
```

除了必须派生一个类和覆盖 OnRenderData 以外，用 COleDataSource 进行延时再现与立即传送没有多大差别。

其他 COleDataSource 函数有时会简化您编写的延时再现程序。例如，如果您只想把数据传送给 HGLOBAL 存储媒体，就可以覆盖 OnRenderGlobalData 而不是 OnRenderData。您还可以使用一组 COleDataSource 函数，名为 DelayRenderFileData 和 OnRenderFileData，并借助于 CFile 输出函数来延时再现数据。

如果使用 COleDataSource 进行延时再现，要注意的一点细节问题是，如果存储类型为 TYMED_HGLOBAL、TYMED_FILE、TYMED_ISTREAM 或 TYMED_ISTORAGE，那么就应该在调用 OnRenderData 之前分配存储媒体。如果存储媒体已经预分配了，那么 OnRenderData 就必须将数据传送到一个已经存在的存储媒体中，而不是自己创建一个新的存储媒体。传递给 OnRenderData 的 STGMEDIUM 结构中的 tymed 值说明了这一点。如果 lpStgMedium -> tymed 为 TYMED_NULL，那么 OnRenderData 就有责任分配存储媒体。如果 lpStgMedium -> tymed 保存有任何其他值，就说明调用者已经提供了存储媒体并且 lpStgMedium -> tymed 标识了存储类型。下面的示例程序代码说明了对于要求预先分配的媒体类型，应如何恰当地处理 OnRenderData的方法：

```
BOOL CMyDataSource::OnRenderData (LPFORMATETC lpFormatEtc,
    LPSTGMEDIUM lpStgMedium)
{
    if (COleDataSource::OnRenderData (lpFormatEtc, lpStgMedium))
        return TRUE;

    if (lpStgMedium->tymed == TYMED_NULL) { // Medium is not preallocated.
        if (lpFormatEtc->tymed & TYMED_HGLOBAL) {
            // Allocate a global memory block, render the data
            // into it, and then copy the handle to lpStgMedium->hGlobal.
            }
    }
    else { // Medium is preallocated.
        if (lpStgMedium->tymed == TYMED_HGLOBAL) {
            // Render the data into the global memory block whose
            // handle is stored in lpStgMedium->hGlobal.
        }
    }
}
```

本例仅仅说明了存储媒体是 HGLOBAL 时的情形，但是基本原理却很明确。

使用 COleDataSource 的延时再现功能的最常见的原因，是在不需要预先分配存储媒体的情况下，可以在多种不同的存储媒体中提供数据。例如，如果您想在几种不同的存储媒体中提供 CF_TEXT 数据，就可以调用 DelayRenderData 并给它传递一个 FORMATETC 结构，其中的 tymed 字段包含了代表您所支持的每一种媒体类型的位标志。然后您就能够通过检查传递给 OnRenderData 的 FORMATETC 结构中的 tymed 字段来传送数据使用者所请求的任何媒体中的数据了。如果您并不支持数据使用者所请求的数据，只要不执行 OnRenderData 而返回 FALSE 就可以。

19.2.8　COleDataSource 和 COleDataObject 复习

现在您已经知道了如何使用 MFC 的 COleDataSource 和 COleDataObject 类与 OLE 剪贴板进行交互了。为了从总体上把握内容（并巩固所学的知识），表 19-6、表 19-7 中给出了最常用的 COleDataSource 和 COleDataObject 成员函数的简要总结。这些类还有其他函数，而这里列出的是最常用的一些。

表 19-6　主要的 COLEDATASOURCE 成员函数

函数	说　明
SetClipboard	将 COleDataSource 放置在 OLE 剪贴板上
CacheData	给 COleDataSource 提供数据
CacheGlobalData	将保存在全局内存中的数据提供给 COleDataSource
DelayRenderData	为延时再现提供数据

续表

函数	说　明
DelayRenderFileData	为使用 CFile 输出函数的延时再现提供数据
OnRenderData	用来将数据传送给任意存储媒体
OnRenderFileData	用来将数据传送给 CFile
OnRenderGlobalData	用来将数据传送给 HGLOBAL

表 19-7 主要的 COLEDATAOBJECT 成员函数

函数	说　明
AttachClipboard	将 COleDataObject 附加给 OLE 剪贴板
GetData	从 COleDataObject 所属的数据对象中获取数据
GetFileData	使用 CFile 函数获取数据
GetGlobalData	获取 HGLOBAL 中的数据
IsDataAvailable	确定在特定格式下和存储媒体中的数据是否有效
BeginEnumFormats	开始枚举有效数据格式的过程
GetNextFormat	用描述下一个有效数据格式的信息填写 FORMATETC 结构

在以前我说过,使用 OLE 剪贴板的主要原因是可以获得使用存储媒体而不仅仅是全局内存的能力。这是实话,但是这里还有个原因。因为有了 COleDataSource 和 COleDataObject 提供的抽象,一旦您编写了使用 OLE 剪贴板的程序,就可以做少量的工作实现一种更便于使用的数据传送形式:OLE 拖放。OLE 拖放允许用户通过鼠标抓取和释放来传送数据。如果没有类库的帮助,编写 OLE 拖放程序可能就不是件好玩的事情,但是 MFC 却使得整个实现过程无可挑剔。

19.3 OLE 拖放

如果您从来没有见过实际的 OLE 拖放,那么可以使用 Visual C++ 中的源程序编辑器做一个简单的演示。首先打开源程序代码文件并加亮显示一行文本。用鼠标左键抓取加亮的文本,保持鼠标键的按下状态,向下拖移几行。然后释放鼠标键。这时文本就会从原来的地方消失而出现在现在放开鼠标键的地方,正像执行了剪切/粘贴操作那样。在按下 Ctrl 的状态下重复此操作过程,文本就会被复制而不是移动。这就是 OLE 拖放。使用它不仅可以把文本从文档的某处传送到另一处,还可以传送到别的不同的文档中,甚至其他应用程序中。并且正像使用 OLE 剪贴板那样,可以使用 OLE 拖放传送任何数据类型,不仅是文本。

从编程的角度看,OLE 拖放与 OLE 剪贴板非常相似。数据提供者,或称为"拖放来源",创建封装数据的数据对象并使得 IDataObject 指针有效。数据使用者,或称为"拖放目标",得到 IDataObject 指针并使用它从数据对象中提取数据。

OLE 拖放和 OLE 剪贴板之间的一个不同点在于 IDataObject 指针是如何传送的。OLE 剪贴板使用::OleSetClipboard 和::OleGetClipboard 从发送者向接收者传送指针。在 OLE 拖放

中，拖放来源通过给::DoDragDrop 传递一个 IDataObject 指针来开始拖放操作。在另一端，任何想成为拖放目标的窗口都要通过调用 API 函数::RegisterDragDrop 向系统注册。如果拖放是在以这种方式注册的窗口间进行，就会把传递给::DoDragDrop 的 IDataObject 的指针递交给拖放目标。

如果这些就是全部工作，那么 OLE 拖放就根本不会那么困难了。实际上，更复杂的问题是 OLE 拖放要求 3 个 COM 对象而不仅仅是一个：

- 一个实现 IDataObject 的数据对象
- 一个实现 IDropSource 的拖放来源对象
- 一个实现 IDropTarget 的拖放目标对象

数据对象与用于 OLE 剪贴板中的相同。拖放来源对象和拖放目标对象是两个新对象。图 19-2 给出了一个拖放数据传送中的参与者的示意图。在事务处理的发送端是一个实现了两个 COM 对象的应用程序：数据对象和拖放来源对象。（并没有什么防止用一个对象支持两个接口的理由，但是实际中这两个对象总是分开实现的。）在接收端是实现了拖放目标对象的应用程序。拖放来源和拖放目标都不会接收到彼此引用的 IDropSource 或 IDropTarget 指针，而是由系统作为中间人在恰当的时候调用两个接口中的方法。

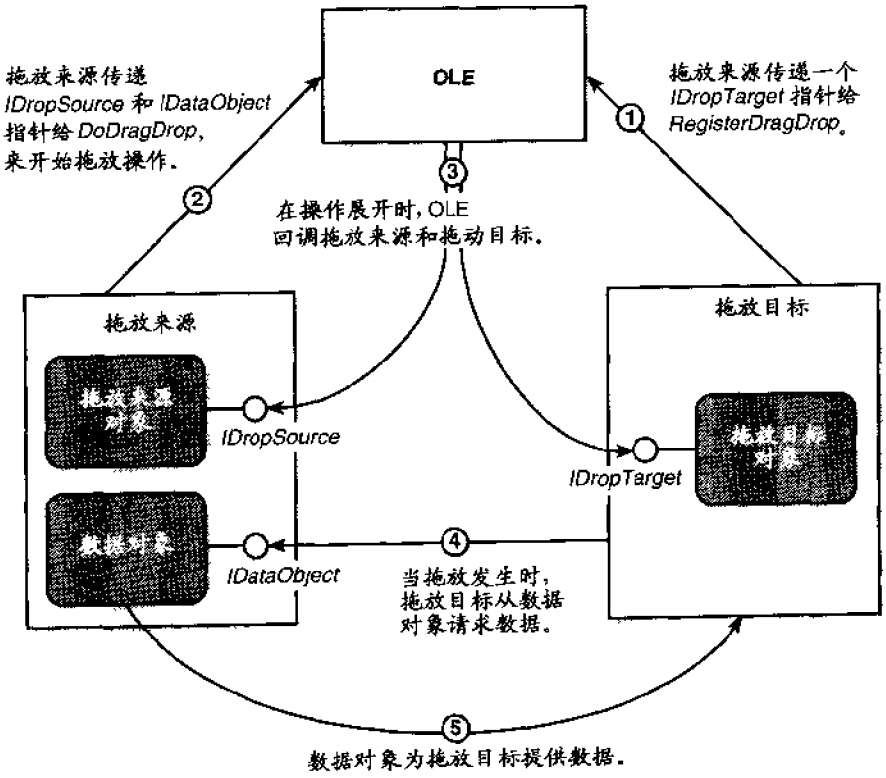


图 19-2　OLE 拖放操作中的参与者

19.3.1 拖放来源剖析

在应用程序调用::DoDragDrop 时 OLE 拖放操作就开始了,它向函数中传递以下 4 个关键信息:

- IDataObject 指针
- IDropSource 指针
- 包含一个以上 DROPEFFECT 代码的输入值,该代码标识了对数据允许进行的操作类型(例如移动、复制或移动并复制)
- 指向 DWORD 的指针,接收标识另一端所发生操作的种类的 DROPEFFECT 代码(例如,是否发生了拖放?如果是,那么数据被移动还是被复制了?)

当满足下列任何一个条件时,::DoDragDrop 返回:

- 释放动作发生了。
- 操作被取消。

取消拖放操作的动作因应用程序的不同而不同,但最终都是由拖放来源决定的。在大多数情况下,按下 Esc 键可以激发取消动作。如果操作被取消或是拖放目标拒绝拖放,那么::DoDragDrop 就会将 DROPEFFECT_NONE 值复制到第 4 个参数的地址中。如果拖放成功,::DoDragDrop 则会将第 3 个参数中其中一个 DROPEFFECT 代码复制到第 4 个参数的地址中,以便拖放来源可以确切地知道发生了什么操作。

假设 pdo 和 pds 分别保存有 IDataObject 和 IDropSource 指针。下列语句将启动一个拖放操作,数据对象中封装的数据可以被移动也可以被复制:

```
DWORD dwEffect;
HRESULT hr = ::DoDragDrop(pdo, pds,
    DROPEFFECT_MOVE | DROPEFFECT_COPY, &dwEffect);
```

在::DoDragDrop 返回后,dwEffect 告诉拖放来源另一端发生了什么事情。如果 dwEffect 等于 DROPEFFECT_NONE 或 DROPEFFECT_COPY,拖放来源就不需要做任何事情。但是,如果 dwEffect 等于 DROPEFFECT_MOVE,拖放来源就必须从源文档中删除数据:

```
if (SUCCEEDED(hr) && dwEffect == DROPEFFECT_MOVE) {
    // Delete the original data from the document.
}
```

这里并没有给出删除数据的程序,很显然它应该由具体的应用程序实现。

调用::DoDragDrop 是一种同步操作;就是说在操作完成或被取消之前,::DoDragDrop 不会返回。但是当执行拖放操作时,系统要与拖放来源通过提供给::DoDragDrop 的 IDropSource 指针进行通信。IDropSource 是一个简单的接口,除了对所有 COM 接口都公用的 IUnknown 方法以外,它仅包含两个方法,如表 19-8 所示。

表 19-8　IDropSource 方法

方法	说　明
GiveFeedback	每次光标移动或键状态改变时被调用，允许拖放来源更新光标
QueryContinueDrag	当键状态或鼠标键状态改变时被调用，允许拖放来源指定以后的具体操作：继续、取消或执行释放操作

每当拖放来源所关心的键或鼠标键的状态改变时，就会调用拖放来源对象的 QueryContinueDrag 方法。QueryContinueDrag 接收两个参数：一个 BOOL 值说明是否按下了 Esc 键；一个 DWORD 包含反映当前鼠标键以及 Ctrl、Alt 和 Shift 键状态的标志。利用此信息，QueryContinueDrag 必须告诉系统接下去要进行的操作，返回 3 个返回值中的一个，如表 19-9 所示。

表 19-9　QueryContinueDrag 的返回值

返回值	说　明
S_OK	继续拖放操作
DRAGDROP_S_DROP	通过执行释放操作结束拖放
DRAGDROP_S_CANCEL	取消拖放操作

如果 Esc 键被按下，典型的响应就是取消操作；如果是鼠标左键被释放则执行释放操作；或者如果不满足以上两种条件则允许继续操作。下面的 QueryContinueDrag 实现就采用了此逻辑条件：

```
HRESULT __stdcall CDropSource::QueryContinueDrag(BOOL fEscape,
    DWORD grfKeyState)
{
    if(fEscape)
        return DRAGDROP_S_CANCEL; // Esc key was pressed.
    if(!(grfKeyState & MK_LBUTTON))
        return DRAGDROP_S_DROP; // Left mouse button was released.
    return S_OK; // Let the operation continue.
}
```

这段程序假定拖放操作是在鼠标左键被按下时开始的。如果您用了鼠标右键，则可以检查鼠标右键(MK_RBUTTON)来决定是否执行拖放。如果您喜欢用 Esc 以外的其他键来取消操作，则可以调用::GetAsyncKeyState 读取键的状态并使用该值而不是 fEscape 来决定是否返回 DRAGDROP_S_CANCEL。

在拖放操作开始后，拖放来源会迅速地接收到对 IDropSource::GiveFeedback 方法的调用。GiveFeedback 接收一个函数参数：DROPEFFECT 代码，它告诉拖放来源，如果即刻开始释放操作的话会发生什么情况（在下一小节您会看到此信息是从拖放目标中得到的，因为最终还是拖放目标在控制另一端发生的操作。）GiveFeedback 的工作是检查此参数并为用户

更新光标,以便提供可视的反馈。当在拖放数据传送过程中,光标从一个窗口移动到另一个窗口时看到其形状改变了,或是在按下 Ctrl 时看到有一个小加号出现,实际上看到的正是拖放来源对 IDropSource::GiveFeedback 的响应。

您也可以创建自己的光标并在每次调用 GiveFeedback 时显示它们;系统已经为此提供了几个预定义光标。要使用它们,只要从 GiveFeedback 实现中返回 DRAGDROP_S_USEDEFAULTCURSORS 即可。请不要这样做:

```
HRESULT __stdcall CDropSource::GiveFeedback (DWORD dwEffect)
{
    HCURSOR hCursor;
    switch (dwEffect) {
        // Inspect dwEffect, and load a cursor handle in hCursor.
    }
    ::SetCursor (hCursor);
    return S_OK;
}
```

而应该如此处理:

```
HRESULT __stdcall CDropSource::GiveFeedback (DWORD dwEffect)
{
    return DRAGDROP_S_USEDEFAULTCURSORS;
}
```

这就是在大多数 IDropSource::GiveFeedback 实现中所做的工作。当然,您还可以做得更多,但除非您有足够的理由那样做,否则最好还是使用默认光标。

19.3.2 拖放目标剖析

当应用程序调用::RegisterDragDrop 之后,在其中传递了窗口的句柄和指向 IDropTarget 接口的指针,该窗口就成为 OLE 拖放目标了:

```
::RegisterDragDrop (hWnd, pdt);
```

而通过调用::RevokeDragDrop 可以取消对拖放目标的注册。如果在拖放目标窗口被销毁之前您没有调用此函数,那么系统也会清除它,但是作为一种好的风格,最好还是要自己调用它。

在拖放操作过程中,当光标进入、离开或在拖放目标窗口上移动时,系统将通知拖放目标这些事件,通知是靠通过提供给::RegisterDragDrop 的 IDropTarget 指针调用 IDropTarget 方法来实现的。IDropTarget 具有表 19-10 中列出的 4 个方法。

表 19-10　IDROPTARGET 方法

方法	说　明
DragEnter	当光标进入拖放目标窗口时调用
DragOver	当光标在拖放目标窗口上移动时调用
DragLeave	当光标离开拖放目标窗口，或是在拖放目标窗口上而操作被取消时调用
Drop	在释放操作发生时调用

DragEnter 和 DragOver 在它们的参数列表中首先要接收一个指向 DWORD 的指针。在它们中的一个被调用时，拖放目标必须通过将 DROPEFFECT 的值复制到 DWORD 中来通知拖放来源如果释放操作发生的话会出现什么情况。复制到 DWORD 中的值是传递给拖放来源的 GiveFeedback 方法的值。DragEnter 和 DragOver 还接收一组光标坐标值（万一释放的输出结果依赖于当前光标位置）以及指定了 Ctrl、Alt 和 Shift 键和其他鼠标键的状态的标志。另外，DragEnter 接收一个 IDataObject 指针用来查询数据对象。下列 DragEnter 和 DragOver 的实现根据数据对象中的文本是否有效以及 Ctrl 键是抬起（移动）还是按下（复制）来给数据源返回 DROPEFFECT_NONE、DROPEFFECT_MOVE 或 DROPEFFECT_COPY：

```
HRESULT __stdcall CDropTarget::DragEnter (IDataObject * pDataObject,
    DWORD grfKeyState, POINTL pt, DWORD * pdwEffect)
{
    FORMATETC fe = {
        CF_TEXT, NULL, DVASPECT_CONTENT, -1, TYMED_HGLOBAL
    };
    if pDataObject -> QueryGetData (&fe) == S_OK) {
        m_bCanAcceptData = TRUE;
        *pdwEffect = (grfKeyState & MK_CONTROL) ?
            DROPEFFECT_COPY : DROPEFFECT_MOVE;
    }
    else {
        m_bCanAcceptData = FALSE;
        pdwEffect = DROPEFFECT_NONE;
    }
    return S_OK;
}

HRESULT __stdcall CDropTarget::DragOver (DWORD grfKeyState,
    POINTL pt, DWORD pdwEffect)
{
    if (m_bCanAcceptData)
        *pdwEffect = (grfKeyState & MK_CONTROL) ?
            DROPEFFECT_COPY : DROPEFFECT_MOVE;
    else
        *pdwEffect = DROPEFFECT_NONE;
```

```
    return S_OK;
}
```

m_bCanAcceptData 是一个 BOOL 成员变量,保存着一个记录信息,说明由拖放来源提供的数据是否采用了拖放目标接受的格式。在调用 DragOver 时,拖放目标将使用这个值来确定是否指示希望接受释放操作。

如果没有执行释放操作光标就离开了拖放目标窗口,或是当光标在拖放目标窗口上时取消了拖放操作,就要调用拖放目标的 DragLeave 方法。调用 DragLeave 给了拖放目标清除自身的机会,如果预计的释放操作没有发生,则将释放所有在 DragEnter 或 DragOver 中分配的资源。

最后一个 IDropTarget 方法 Drop 当且仅当释放操作发生时才被调用。Drop 在它的参数列表中接收所有与处理释放操作有关的信息,包括一个 IDataObject 指针;一个 DWORD,指定了 Ctrl、Alt 和 Shift 键以及鼠标键状态;以及光标坐标。它还接收一个 DWORD 指针并且必须给该指针复制 DROPEFFECT 值来通知数据源释放操作后的结果。下面 Drop 的实现在文本字符串有效时从数据对象那里获取到文本字符串:

```
HRESULT _stdcall CDropTarget::Drop (IDataObject * pDataObject,
    DWORD grfKeyState, POINTL pt, DWORD * pdwEffect)
{
    if (m_bCanAcceptData) {
        FORMATETC fe = {
            CF_TEXT, NULL, DVASPECT_CONTENT, -1, TYMED_HGLOBAL
        };

        STGMEDIUM stgm;

        if (SUCCEEDED (pDataObject -> GetData (&fe, &stgm)) &&
            stgm.hGlobal != NULL) {
            // Copy the string from the global memory block.
                .
                .
                .
            ::ReleaseStgMedium (&stgm);
            *pdwEffect = (grfKeyState & MK_CONTROL) ?
                DROPEFFECT_COPY : DROPEFFECT_MOVE;
            return S_OK;
        }
    }
    // If we make it to here, the drop did not succeed.
    *pdwEffect = DROPEFFECT_NONE;
    return S_OK;
}
```

Drop 调用后并不是跟着 DragLeave 调用，因此如果在释放操作完成后要执行清理工作的话，就应该在 Drop 方法中进行。

19.3.3　MFC 对 OLE 拖放的支持

编写 OLE 拖放程序的大部分工作都是用来实现 COM 对象的。幸运的是 MFC 会为您实现这些工作。为 OLE 剪贴板提供数据对象的同一个 COleDataSource 类对于 OLE 拖放也适用。COleDropSource 提供了便于使用的拖放来源对象，COleDropTarget 提供了拖放目标对象。通常您甚至不必亲自实例化 COleDropSource，因为 COleDataSource 会为您做此工作。但是您必须实例化 COleDropTarget，为此通常只要给应用程序的视图类添加 COleDropTarget 成员变量就可以了。

假设您希望在 MFC 应用程序中使用 OLE 拖放传送一个文本字符串。下面给出了将全局内存块作为存储媒体而实现您的目的的方法：

```
char szText[] = "Hello, world";
HANDLE hData = ::GlobalAlloc (GMEM_MOVEABLE, ::lstrlen (szText) + 1)
LPSTR pData = (LPSTR) ::GlobalLock (hData);
::lstrcpy (pData, szText);
::GlobalUnlock (hData);

COleDataSource ods;
ods.CacheGlobalData (CF_TEXT, hData);

DROPEFFECT de =
    ods.DoDragDrop (DROPEFFECT_MOVE | DROPEFFECT_COPY)

if (de == DROPEFFECT_MOVE) {
    // Delete the string from the document.
```

这段程序与本章前面提供的使用 COleDataSource 将文本字符串放在 OLE 剪贴板上的程序非常相似。除了 COleDataSource 对象是在堆栈上而不是在堆上创建的这点差别外（在本例中对象不需要在创建它的函数以外存在，因此在堆栈上创建它是对的），唯一真正的区别就是调用了 COleDataSource::DoDragDrop 而不是 COleDataSource::SetClipboard。COleDataSource::DoDragDrop封装了同名的 API 函数。除了为您调用了::DoDragDrop，它还创建了 COleDropSource 对象，其 IDropSource 接口指针传递给了::DoDragDrop。

如果您愿意创建自己的 COleDropSource 对象，那也完全可以把它的地址在函数的可选的第二个参数中传递给 COleDataSource::DoDragDrop 即可。您亲自创建这个对象的唯一原因是想从 COleDropSource 派生一个类以便使用，而不是使用 COleDropSource。程序员偶尔会从 COleDropSource 中派生类并覆盖它的 GiveFeedback 和 QueryContinueDrag 成员函数，以便为相同名称的 IDropSource 方法提供自定义响应。

MFC 使得实现 OLE 拖动数据传送中的目标也相当容易。首先要做的就是给应用程序的视图类添加 COleDropTarget 数据成员:

```
// In CMyView's class declaration
COleDropTarget m_oleDropTarget;
```

然后,在视图的 OnCreate 函数中,调用 COleDropTarget::Register 并给它传递一个指向视图对象的指针:

```
m_oleDropTarget.Register(this);
```

最后,覆盖视图的 OnDragEnter、OnDragOver、OnDragLeave 和 OnDrop 函数或它们的一些组合。这些 CView 函数与名称相似的 IDropTarget 方法配对使用。例如,当调用拖放目标对象的 IDropTarget::Drop 方法时,COleDropTarget::OnDrop 就会调用视图的 OnDrop 函数。要响应对 IDropTarget::Drop 的调用,只要覆盖 CView::OnDrop 即可。

下例说明了如何在 CScrollView 派生类中覆盖 OnDragEnter、OnDragOver 和 OnDrop 来使视图成为文本的拖放目标。在本例中没有覆盖 OnDragLeave,因为在调用它时没有特别需要的工作要做。注意在每个函数的参数列表中都提供了预先分配的 COleDataObject。这个 COleDataObject 封装了传递给拖放目标的 IDropTarget 方法的 IDataObject 指针:

```
DROPEFFECT CMyView::OnDragEnter(COleDataObject* pDataObject,
    DWORD dwKeyState, CPoint point)
{
    CScrollView::OnDragEnter(pDataObject, dwKeyState, point);
    if(!pDataObject->IsDataAvailable(CF_TEXT))
        return DROPEFFECT_NONE;
    return (dwKeyState & MK_CONTROL) ?
        DROPEFFECT_COPY : DROPEFFECT_MOVE;
}

DROPEFFECT CMyView::OnDragOver(COleDataObject* pDataObject,
    DWORD dwKeyState, CPoint point)
{
    CScrollView::OnDragOver(pDataObject, dwKeyState, point);
    if(!pDataObject->IsDataAvailable(CF_TEXT))
        return DROPEFFECT_NONE;
    return (dwKeyState & MK_CONTROL) ?
        DROPEFFECT_COPY : DROPEFFECT_MOVE;
}

BOOL CMyView::OnDrop(COleDataObject* pDataObject, DROPEFFECT dropEffect,
    CPoint point)
{
    CScrollView::OnDrop(pDataObject, dropEffect, point);
```

```
    HANDLE hData = pDataObject -> GetGlobalData (CF_TEXT);
    if (hData != NULL) {
        // Copy the string from the global memory block.
            .
            .
            .
        ::GlobalFree (hData);
        return TRUE; // Drop succeeded.
    }
    return FALSE; // Drop failed.
}
```

这段程序看上去像是上一小节提供的非 MFC 版本的程序。OnDragEnter 和 OnDragOver 通过在它们的参数列表中提供的指针调用 COleDataObject::IsDataAvailable 来确定文本是否有效。如果回答是否定的，两个函数都将返回 DROPEFFECT_NONE，说明它们都不接受释放操作。接下来，拖放来源可能会显示一个“放不下”光标。如果文本有效，根据 Ctrl 键是否被按下，OnDragEnter 和 OnDragOver 将返回 DROPEFFECT_MOVE 或 DROPEFFECT_COPY。在释放操作发生时，OnDrop 使用 COleDataObject::GetGlobalData 来获取数据。

19.3.4　拖放目标滚动

上一小节中的例子假定拖放目标是基于视图的应用程序。您也可以使用 COleDropTarget 在没有视图的应用程序中实现拖放目标，此时从 COleDropTarget 中派生自己的类并覆盖 OnDragEnter、OnDragOver、OnDragLeave 和 OnDrop 即可。但是，如果视图具有滚动条的话，使用视图作为拖放目标具有一个非常吸引人的好处：可以不付任何代价获得拖放目标滚动功能，这是 MFC 的恩惠。

什么是拖放目标滚动呢？假设已经开始了拖放操作，用户希望将数据放在 CScrollView 中当前视野以外的地方。如果光标在视图边界几个像素范围内暂停一会儿，只要光标还在邻近范围内，CScrollView 就会自动滚动。这样用户就可以将光标移动到窗口的边上等待释放位置出现。如果 MFC 不为您实现此功能的话，这就是您必须处理的另一个细节问题了。

19.4　综合应用：WIDGET 应用程序

图 19-3 所示的应用程序说明了在现实中应用本章讲述的概念、原理和程序段的一种方法。Widget 在响应 Insert 菜单中的命令时将创建各种颜色的三角形“饰件”。使用 Edit 菜单中的命令可以将饰件传送到 OLE 剪贴板或从 OLE 剪贴板中获取饰件。在能够使用 Cut 和 Copy 命令之前，必须通过鼠标单击先选中饰件。饰件变为绿色时说明正处于被选中状态。还可以使用 OLE 拖放来移动和复制饰件。在执行释放操作时，如果按下 Ctrl 键，饰件就会被复制；否则被移动。为了形象地说明 OLE 拖放操作，可以并排运行两个 Widget 实例，在它

们之间来回拖放饰件。

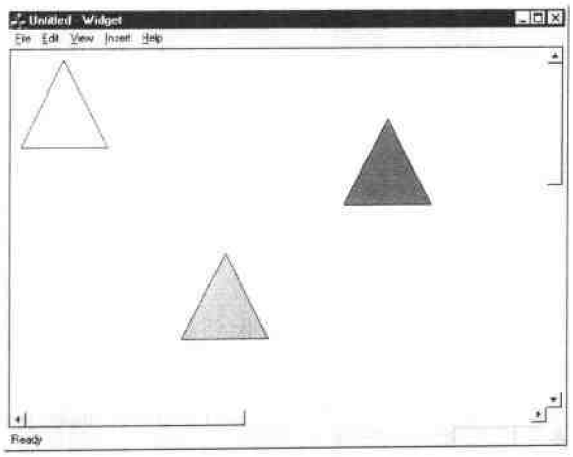


图 19-3 Widget 窗口

图 19-4 给出了相关的 Widget 源程序代码。WidgetView.cpp 包含了大部分精彩内容,包括 Cut、Copy 和 Paste 命令处理程序。它还包含了 OnDragEnter、OnDragOver、OnDragLeave 和 OnDrop 的覆盖版本,以及当单击鼠标左键时启动拖放数据传送的程序。(参见 OnLButtonDown。) Widgets 是通过全局内存传送数据的。Widget 为饰件注册了私有剪贴板格式,并在调用 COleDataSource::CacheGlobalData 和 COleDataObject::GetGlobalData 中使用了它。相应的 ID 保存在应用程序对象中,并且可以通过使用 CWidgetApp::GetClipboardFormat 得到。

Widget.h

```
// Widget.h : main header file for the WIDGET application
//

#if !defined(AFX_WIDGET_H__02909A45_3F5C_11D2_AC89_006008A8274D__INCLUDED_)
#define AFX_WIDGET_H__02909A45_3F5C_11D2_AC89_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
```

```
// CWidgetApp:
// See Widget.cpp for the implementation of this class
//

class CWidgetApp : public CWinApp
{
public:
    UINT GetClipboardFormat ();
    CWidgetApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWidgetApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CWidgetApp)
    afx_msg void OnAppAbout();
       // NOTE - the ClassWizard will add and remove member functions here.
       //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
protected:
    UINT m_nFormat;
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(AFX_WIDGET_H__02999A45_3F5C_11D0_AC89_006008A827ID__INCLUDED_)
```

Widget.cpp

```
// Widget.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Widget.h"

#include "MainFrm.h"
#include "WidgetDoc.h"
#include "WidgetView.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CWidgetApp

BEGIN_MESSAGE_MAP(CWidgetApp, CWinApp)
    //{{AFX_MSG_MAP(CWidgetApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CWidgetApp construction

CWidgetApp::CWidgetApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CWidgetApp object

CWidgetApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CWidgetApp initialization

BOOL CWidgetApp::InitInstance()
{
    if (!AfxOleInit ()) {
        AfxMessageBox (_T ("AfxOleInit failed"));
        return FALSE;
    }

    SetRegistryKey(_T("Local AppWizard-Generated Applications"));
    LoadStdProfileSettings();  // Load standard INI file
                               // options (including MRU)

    // Register the application's document templates.  Document templates
    //  serve as the connection between documents, frame windows and views.
```

```
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CWidgetDoc),
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CWidgetView));
    AddDocTemplate(pDocTemplate);

    // Enable DDE Execute open
    EnableShellOpen();
    RegisterShellFileTypes(TRUE);

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so show and update it.
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();

    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();

    //
    // Register a private clipboard format for widgets.
    //
    m_nFormat = ::RegisterClipboardFormat(_T("Widget"));
    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
```

```
    virtual void DoDataExchange(CDataExchange * pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange * pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CWidgetApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CWidgetApp message handlers

UINT CWidgetApp::GetClipboardFormat()
{
    return m_nFormat;
}
```

WidgetDoc.h

```
// WidgetDoc.h : interface of the CWidgetDoc class
//
```

```
////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_WIDGETDOC_H__02909A4B_3F5C_11D2_AC89_006008A8274D__INCLUDED_)
#define AFX_WIDGETDOC_H__02909A4B_3F5C_11D2_AC89_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "WidgetObj.h"

typedef CTypedPtrArray< CObArray, CWidget* > CWidgetArray;

class CWidgetDoc : public CDocument
{
protected: // create from serialization only
    CWidgetDoc();
    DECLARE_DYNCREATE(CWidgetDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWidgetDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    virtual void DeleteContents();
    //}}AFX_VIRTUAL

// Implementation
public:
    BOOL RemoveWidget (int nIndex);
    int AddWidget (int x, int y, COLORREF color);
    CWidget* GetWidget (int nIndex);
    int GetWidgetCount ();
    virtual ~CWidgetDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
    CWidgetArray m_arrWidgets;
```

```
// Generated message map functions
protected:
    //{{AFX_MSG(CWidgetDoc)
    afx_msg void OnInsertRedWidget();
    afx_msg void OnInsertBlueWidget();
    afx_msg void OnInsertYellowWidget();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{ AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_WIDGETDOC_H__02909A4B_3F5C_11D2_AC89_006008A8274D__INCLUDED_)
```

WidgetDoc.cpp

```
CWidgetDoc::~CWidgetDoc()
{
}

BOOL CWidgetDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CWidgetDoc serialization

void CWidgetDoc::Serialize(CArchive& ar)
{
    m_arrWidgets.Serialize(ar);
}

/////////////////////////////////////////////////////////////////////////////
// CWidgetDoc diagnostics

#ifdef _DEBUG
void CWidgetDoc::AssertValid() const
{
```

```
    CDocument::AssertValid();
}

void CWidgetDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CWidgetDoc commands

void CWidgetDoc::DeleteContents()
{
    int i = m_arrWidgets.GetSize();
    while (i)
        delete m_arrWidgets[--i];
    m_arrWidgets.RemoveAll();
    CDocument::DeleteContents();
}

int CWidgetDoc::GetWidgetCount()
{
    return m_arrWidgets.GetSize();
}

CWidget* CWidgetDoc::GetWidget(int nIndex)
{
    if (nIndex >= m_arrWidgets.GetSize())
        return NULL;
    return (CWidget*)m_arrWidgets[nIndex];
}

int CWidgetDoc::AddWidget(int x, int y, COLORREF color)
{
    int nIndex = -1;
    CWidget* pWidget = NULL;

    try {
        pWidget = new CWidget(x, y, color);
        nIndex = m_arrWidgets.Add(pWidget);
        SetModifiedFlag();
    }
    catch (CMemoryException* e) {
        AfxMessageBox(_T("Out of memory"));
        if (pWidget != NULL)
            delete pWidget;
        e->Delete();
        return -1;
```

```
    :
    return nIndex;
}

BOOL CWidgetDoc::RemoveWidget(int nIndex)
{
    if (nIndex >= m_arrWidgets.GetSize())
        return FALSE;

    delete m_arrWidgets[nIndex];
    m_arrWidgets.RemoveAt(nIndex);
    return TRUE;
}

void CWidgetDoc::OnInsertBlueWidget()
{
    AddWidget(10, 10, RGB(0, 0, 255));
    UpdateAllViews(NULL);
}

void CWidgetDoc::OnInsertRedWidget()
{
    AddWidget(10, 10, RGB(255, 0, 0));
    UpdateAllViews(NULL);
}

void CWidgetDoc::OnInsertYellowWidget()
{
    AddWidget(10, 10, RGB(255, 255, 0));
    UpdateAllViews(NULL);
}
```

WidgetView.h

```
// WidgetView.h : interface of the CWidgetView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_WIDGETVIEW_H__02909A4D_3F5C_11D2_AC89_006008A8274D__INCLUDED_)
#define AFX_WIDGETVIEW_H__02909A4D_3F5C_11D2_AC89_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

typedef struct tagWIDGETINFO {
    int x;                  // x coordinate of widget's upper left corner
    int y;                  // y coordinate of widget's upper left corner
```

```
    int cx;                 // Horizontal drag offset
    int cy;                 // Vertical drag offset
    COLORREF color;         // The widget's color
} WIDGETINFO;

class CWidgetView : public CScrollView
{
protected: // create from serialization only
    CWidgetView();
    DECLARE_DYNCREATE(CWidgetView)

// Attributes
public:
    CWidgetDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWidgetView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual DROPEFFECT OnDragEnter(COleDataObject* pDataObject,
        DWORD dwKeyState, CPoint point);
    virtual DROPEFFECT OnDragOver(COleDataObject* pDataObject,
        DWORD dwKeyState, CPoint point);
    virtual void OnDragLeave();
    virtual BOOL OnDrop(COleDataObject* pDataObject,
        DROPEFFECT dropEffect, CPoint point);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CWidgetView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
    CWidget* m_pTempWidget;
    CSize m_offset;
    CPoint m_pointLastImage;
    CPoint m_pointLastMsg;
```

```
    int m_nSel;
    COleDropTarget m_oleDropTarget;

// Generated message map functions
protected:
    //{{AFX_MSG(CWidgetView)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnEditCut();
    afx_msg void OnEditCopy();
    afx_msg void OnEditPaste();
    afx_msg void OnEditDelete();
    afx_msg void OnUpdateEditCut(CCmdUI * pCmdUI);
    afx_msg void OnUpdateEditCopy(CCmdUI * pCmdUI);
    afx_msg void OnUpdateEditPaste(CCmdUI * pCmdUI);
    afx_msg void OnUpdateEditDelete(CCmdUI * pCmdUI);
    afx_msg void OnSetFocus(CWnd * pOldWnd);
    afx_msg void OnKillFocus(CWnd * pNewWnd);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in WidgetView.cpp
inline CWidgetDoc * CWidgetView::GetDocument()
   { return (CWidgetDoc * )m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_WIDGETVIEW_H__02909A4D_3F5C_11D2_AC89_006008A8274D__INCLUDED_)
```

WidgetView.cpp

```
// WidgetView.cpp : implementation of the CWidgetView class
//

#include "stdafx.h"
#include "Widget.h"

#include "WidgetDoc.h"
#include "WidgetView.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CWidgetView

IMPLEMENT_DYNCREATE(CWidgetView, CScrollView)

BEGIN_MESSAGE_MAP(CWidgetView, CScrollView)
    //{{AFX_MSG_MAP(CWidgetView)
    ON_WM_CREATE()
    ON_WM_LBUTTONDOWN()
    ON_COMMAND(ID_EDIT_CUT, OnEditCut)
    ON_COMMAND(ID_EDIT_COPY, OnEditCopy)
    ON_COMMAND(ID_EDIT_PASTE, OnEditPaste)
    ON_COMMAND(ID_EDIT_DELETE, OnEditDelete)
    ON_UPDATE_COMMAND_UI(ID_EDIT_CUT, OnUpdateEditCut)
    ON_UPDATE_COMMAND_UI(ID_EDIT_COPY, OnUpdateEditCopy)
    ON_UPDATE_COMMAND_UI(ID_EDIT_PASTE, OnUpdateEditPaste)
    ON_UPDATE_COMMAND_UI(ID_EDIT_DELETE, OnUpdateEditDelete)
    ON_WM_SETFOCUS()
    ON_WM_KILLFOCUS()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CWidgetView construction/destruction

CWidgetView::CWidgetView()
{
}

CWidgetView::~CWidgetView()
{
}

BOOL CWidgetView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CScrollView::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CWidgetView drawing

void CWidgetView::OnDraw(CDC * pDC)
```

```
    CWidgetDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    int nCount = pDoc->GetWidgetCount();
    if (nCount) {
        //
        // Draw all widgets.
        //
        for (int i = 0; i < nCount; i++)
            pDoc->GetWidget(i)->Draw(pDC);

        //
        // Draw the selected widget if this view has the input focus.
        //
        if (m_nSel != -1 && CWnd::GetFocus() == this)
            pDoc->GetWidget(m_nSel)->DrawSelected(pDC);
    }
}

void CWidgetView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();
    SetScrollSizes(MM_TEXT, CSize(1280, 1024));
    m_pTempWidget = NULL;
    m_nSel = -1;
}

////////////////////////////////////////////////////////////////////////////
// CWidgetView diagnostics

#ifdef _DEBUG
void CWidgetView::AssertValid() const
{
    CScrollView::AssertValid();
}

void CWidgetView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}

CWidgetDoc * CWidgetView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CWidgetDoc)));
    return (CWidgetDoc * )m_pDocument;
}
#endif // _DEBUG

////////////////////////////////////////////////////////////////////////////
```

```
// CWidgetView message handlers

DROPEFFECT CWidgetView::OnDragEnter(COleDataObject * pDataObject,
    DWORD dwKeyState, CPoint point)
{
    CScrollView::OnDragEnter(pDataObject, dwKeyState, point);

    //
    // If a widget is available from the drop source, create a temporary
    // widget for drag imaging.
    //
    UINT nFormat = ((CWidgetApp * ) AfxGetApp ()) -> GetClipboardFormat ();
    HGLOBAL hData = pDataObject -> GetGlobalData (nFormat);

    if (hData != NULL) {
        WIDGETINFO * pWidgetInfo = (WIDGETINFO * ) ::GlobalLock (hData);
        int x = point.x - pWidgetInfo -> cx;
        int y = point.y - pWidgetInfo -> cy;
        m_offset.cx = pWidgetInfo -> cx;
        m_offset.cy = pWidgetInfo -> cy;
        COLORREF color = pWidgetInfo -> color;
        ::GlobalUnlock (hData);
        ::GlobalFree (hData);

        m_pTempWidget = new CWidget (x, y, color);
        m_pointLastImage.x = m_pointLastImage.y = -32000;
        m_pointLastMsg = point;

        //
        // Return DROPEFFECT_COPY if the Ctrl key is down, or
        // DROPEFFECT_MOVE if it is not.
        //
        return (dwKeyState & MK_CONTROL) ?
            DROPEFFECT_COPY : DROPEFFECT_MOVE;
    }
    //
    // The cursor isn't carrying a widget. Indicate that the drop target
    // will not accept a drop.
    //
    m_pTempWidget = NULL;
    return DROPEFFECT_NONE;
}

DROPEFFECT CWidgetView::OnDragOver(COleDataObject * pDataObject,
    DWORD dwKeyState, CPoint point)
{
    CScrollView::OnDragOver(pDataObject, dwKeyState, point);

    //
```

```
    // Return now if the object being dragged is not a widget.
    //
    if (m_pTempWidget == NULL)
        return DROPEFFECT_NONE;

    //
    // Convert the drag point to logical coordinates.
    //
    CClientDC dc (this);
    OnPrepareDC (&dc);
    dc.DPtoLP (&point);

    //
    // If the cursor has moved, erase the old drag image and
    // draw a new one.
    //
    if (point != m_pointLastMsg) {
        CPoint pt (point.x - m_offset.cx, point.y - m_offset.cy);
        m_pTempWidget -> DrawDragImage (&dc, m_pointLastImage);
        m_pTempWidget -> DrawDragImage (&dc, pt);
        m_pointLastImage = pt;
        m_pointLastMsg = point;
    }

    //
    // Return DROPEFFECT_COPY if the Ctrl key is down, or DROPEFFECT_MOVE
    // if it is not.
    //
    return (dwKeyState & MK_CONTROL) ?
        DROPEFFECT_COPY : DROPEFFECT_MOVE;
}

void CWidgetView::OnDragLeave()
{
    CScrollView::OnDragLeave();

    //
    // Erase the last drag image and delete the temporary widget.
    //
    if (m_pTempWidget != NULL) {
        CClientDC dc (this);
        OnPrepareDC (&dc);
        m_pTempWidget -> DrawDragImage (&dc, m_pointLastImage);
        delete m_pTempWidget;
        m_pTempWidget = NULL;
    }
}
```

```
BOOL CWidgetView::OnDrop(COleDataObject * pDataObject,
    DROPEFFECT dropEffect, CPoint point)
{
    CScrollView::OnDrop(pDataObject, dropEffect, point);

    //
    // Convert the drop point to logical coordinates.
    //
    CClientDC dc(this);
    OnPrepareDC(&dc);
    dc.DPtoLP(&point);

    //
    // Erase the last drag image and delete the temporary widget.
    //
    if (m_pTempWidget != NULL) {
        m_pTempWidget->DrawDragImage(&dc, m_pointLastImage);
        delete m_pTempWidget;
        m_pTempWidget = NULL;
    }

    //
    // Retrieve the HGLOBAL from the data object and create a widget.
    //
    UINT nFormat = ((CWidgetApp *) AfxGetApp())->GetClipboardFormat();
    HGLOBAL hData = pDataObject->GetGlobalData(nFormat);

    if (hData != NULL) {
        WIDGETINFO * pWidgetInfo = (WIDGETINFO *) ::GlobalLock(hData);
        int x = point.x - pWidgetInfo->cx;
        int y = point.y - pWidgetInfo->cy;
        COLORREF color = pWidgetInfo->color;
        ::GlobalUnlock(hData);
        ::GlobalFree(hData);

        CWidgetDoc * pDoc = GetDocument();
        m_nSel = pDoc->AddWidget(x, y, color);
        pDoc->UpdateAllViews(NULL);
        return TRUE;
    }
    return FALSE;
}

int CWidgetView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CScrollView::OnCreate(lpCreateStruct) == -1)
        return -1;
```

```
    m_oleDropTarget.Register (this);
    return 0;
}

void CWidgetView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CScrollView::OnLButtonDown(nFlags, point);

    CWidgetDoc * pDoc = GetDocument ();
    int nCount = pDoc -> GetWidgetCount ();

    if (nCount) {
        //
        // Convert the click point to logical coordinates.
        //
        CClientDC dc (this);
        OnPrepareDC (&dc);
        dc.DPtoLP (&point);

        //
        // Find out whether a widget was clicked.
        //
        int i;
        BOOL bHit = FALSE;
        for (i = nCount - 1; i >= 0 && !bHit; i--) {
            CWidget * pWidget = pDoc -> GetWidget (i);
            if (pWidget -> PtInWidget (point)) {
                bHit = TRUE;
            }
        }

        //
        // If no widget was clicked, change the selection to NULL and exit.
        //
        if (!bHit) {
            m_nSel = -1;
            InvalidateRect (NULL, FALSE);
            return;
        }
        //
        // Select the widget that was clicked.
        //
        int nWidgetIndex = i + 1;

        if (m_nSel != nWidgetIndex) {
            m_nSel = nWidgetIndex;
            InvalidateRect (NULL, FALSE);
            UpdateWindow ();
```

```
        }

        //
        // Begin a drag-and-drop operation involving the selected widget.
        //
        HANDLE hData = ::GlobalAlloc (GMEM_MOVEABLE, sizeof (WIDGETINFO));

        WIDGETINFO * pWidgetInfo = (WIDGETINFO *) ::GlobalLock (hData);
        CWidget * pWidget = pDoc -> GetWidget (nWidgetIndex);
        ASSERT (pWidget != NULL);
        CRect rect = pWidget -> GetRect ();
        pWidgetInfo -> cx = point.x - rect.left;
        pWidgetInfo -> cy = point.y - rect.top;
        pWidgetInfo -> color = pWidget -> GetColor ();
        ::GlobalUnlock (hData);

        COleDataSource ods;
        UINT nFormat = ((CWidgetApp *) AfxGetApp ()) -> GetClipboardFormat ();
        ods.CacheGlobalData (nFormat, hData);

        int nOldSel = m_nSel;
        DROPEFFECT de = ods.DoDragDrop (DROPEFFECT_COPY | DROPEFFECT_MOVE);

        if (de == DROPEFFECT_MOVE) {
            pDoc -> RemoveWidget (nWidgetIndex);
            int nCount = pDoc -> GetWidgetCount ();
            if (nOldSel == m_nSel || nCount == 0)
                m_nSel = -1;
            else if (m_nSel >= nCount)
                m_nSel = nCount - 1;
            pDoc -> UpdateAllViews (NULL);
        }
    }
}

void CWidgetView::OnEditCut()
{
    if (m_nSel != -1) {
        OnEditCopy ();
        OnEditDelete ();
    }
}

void CWidgetView::OnEditCopy()
{
    if (m_nSel != -1) {
        //
        // Copy data describing the currently selected widget to a
        // global memory block.
```

```
        //
        HANDLE hData = ::GlobalAlloc (GMEM_MOVEABLE, sizeof (WIDGETINFO));

        WIDGETINFO * pWidgetInfo = (WIDGETINFO * ) ::GlobalLock (hData);
        CWidgetDoc * pDoc = GetDocument ();
        CWidget * pWidget = pDoc -> GetWidget (m_nSel);
        ASSERT (pWidget != NULL);
        CRect rect = pWidget -> GetRect ();
        pWidgetInfo -> x = rect.left;
        pWidgetInfo -> y = rect.top;
        pWidgetInfo -> color = pWidget -> GetColor ();
        ::GlobalUnlock (hData);

        //
        // Place the widget on the clipboard.
        //
        COleDataSource * pods = new COleDataSource;
        UINT nFormat = ((CWidgetApp * ) AfxGetApp ()) -> GetClipboardFormat ();
        pods -> CacheGlobalData (nFormat, hData);
        pods -> SetClipboard ();
    }
}

void CWidgetView::OnEditPaste()
{
    //
    // Create a COleDataObject and attach it to the clipboard.
    //
    COleDataObject odo;
    odo.AttachClipboard ();

    //
    // Retrieve the HGLOBAL from the clipboard and create a widget.
    //
    UINT nFormat = ((CWidgetApp * ) AfxGetApp ()) -> GetClipboardFormat ();
    HGLOBAL hData = odo.GetGlobalData (nFormat);


if (hData != NULL) {
        WIDGETINFO * pWidgetInfo = (WIDGETINFO * ) ::GlobalLock (hData);
        int x = pWidgetInfo -> x;
        int y = pWidgetInfo -> y;
        COLORREF color = pWidgetInfo -> color;
        ::GlobalUnlock (hData);
        ::GlobalFree (hData);

        CWidgetDoc * pDoc = GetDocument ();
        m_nSel = pDoc -> AddWidget (x, y, color);
```

```
        pDoc -> UpdateAllViews (NULL);
    }
}

void CWidgetView::OnEditDelete()
{
    if (m_nSel != -1) {
        CWidgetDoc * pDoc = GetDocument ();
        pDoc -> RemoveWidget (m_nSel);
        m_nSel = -1;
        pDoc -> UpdateAllViews (NULL);
    }
}

void CWidgetView::OnUpdateEditCut(CCmdUI * pCmdUI)
{
    pCmdUI -> Enable (m_nSel != -1);
}

void CWidgetView::OnUpdateEditCopy(CCmdUI * pCmdUI)
{
    pCmdUI -> Enable (m_nSel != -1);
}

void CWidgetView::OnUpdateEditPaste(CCmdUI * pCmdUI)
{
    UINT nFormat = ((CWidgetApp * ) AfxGetApp ()) -> GetClipboardFormat ();
    pCmdUI -> Enable (::IsClipboardFormatAvailable (nFormat));
}

void CWidgetView::OnUpdateEditDelete(CCmdUI * pCmdUI)
{
    pCmdUI -> Enable (m_nSel != -1);
}

void CWidgetView::OnKillFocus(CWnd * pNewWnd)
{
    CScrollView::OnKillFocus(pNewWnd);
    InvalidateRect (NULL, FALSE);
}

void CWidgetView::OnSetFocus(CWnd * pOldWnd)
{
    CScrollView::OnSetFocus(pOldWnd);
    InvalidateRect (NULL, FALSE);
}
```

WidgetObj.h

```
#if !defined(
    AFX_WIDGETOBJ_H__02909A57_3F5C_11D2_AC89_006008A8274D__INCLUDED_)
#define AFX_WIDGETOBJ_H__02909A57_3F5C_11D2_AC89_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

// WidgetObj.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CWidget command target

class CWidget : public CObject
{
    DECLARE_SERIAL(CWidget)

// Attributes
public:

// Operations
public:
    CWidget();
    CWidget(int x, int y, COLORREF color);
    virtual ~CWidget();
    void DrawSelected(CDC* pDC);
    BOOL PtInWidget(POINT point);
    virtual void DrawDragImage(CDC* pDC, POINT point);
    virtual void Draw(CDC* pDC);
    COLORREF GetColor();
    CRect GetRect();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWidget)
    //}}AFX_VIRTUAL
    virtual void Serialize(CArchive& ar);

// Implementation
protected:
    COLORREF m_color;
    CRect m_rect;
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_WIDGETOBJ_H__02909A57_3F5C_11D2_AC89_0C6008A8274D__INCLUDED_)
```

WidgetObj.cpp

```
// WidgetObj.cpp : implementation file
//

#include "stdafx.h"
#include "Widget.h"
#include "WidgetObj.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CWidget

IMPLEMENT_SERIAL(CWidget, CObject, 1)

CWidget::CWidget()
{
    m_rect = CRect(0, 0, 90, 90);
    m_color = RGB(255, 0, 0);
}
CWidget::CWidget(int x, int y, COLORREF color)
{
    m_rect = CRect(x, y, x + 90, y + 90);
    m_color = color;
}

CWidget::~CWidget()
{
}

/////////////////////////////////////////////////////////////////////////////
// CWidget message handlers

CRect CWidget::GetRect()
{
    return m_rect;
}
```

```
COLORREF CWidget::GetColor()
{
    return m_color;
}

void CWidget::Serialize(CArchive& ar)
{
    CObject::Serialize(ar);

    if (ar.IsStoring())
        ar << m_rect << m_color;
    else
        ar >> m_rect >> m_color;
}

void CWidget::Draw(CDC pDC)
{
    CBrush brush(m_color);
    CBrush pOldBrush = pDC->SelectObject(&brush);

    CPoint points[3];
    points[0].x = m_rect.left;
    points[0].y = m_rect.bottom;
    points[1].x = m_rect.left + (m_rect.Width() / 2);
    points[1].y = m_rect.top;
    points[2].x = m_rect.right;
    points[2].y = m_rect.bottom;
    pDC->Polygon(points, 3);

    pDC->SelectObject(pOldBrush);
}

void CWidget::DrawSelected(CDC pDC)
{
    CBrush brush(RGB(0, 255, 0));
    CBrush pOldBrush = pDC->SelectObject(&brush);

    CPoint points[3];
    points[0].x = m_rect.left;
    points[0].y = m_rect.bottom;
    points[1].x = m_rect.left - (m_rect.Width() / 2);
    points[1].y = m_rect.top;
    points[2].x = m_rect.right;
    points[2].y = m_rect.bottom;
    pDC->Polygon(points, 3);

    pDC->SelectObject(pOldBrush);
}
```

```
void CWidget::DrawDragImage(CDC pDC, POINT point)
{
    int nOldMode = pDC->SetROP2 (R2_NOT);
    CBrush pOldBrush = (CBrush*) pDC->SelectStockObject (NULL_BRUSH);

    CPoint points[3];
    points[0].x = point.x;
    points[0].y = point.y + m_rect.Height ();
    points[1].x = point.x + (m_rect.Width () / 2);
    points[1].y = point.y;
    points[2].x = point.x + m_rect.Width ();
    points[2].y = point.y + m_rect.Height ();
    pDC->Polygon (points, 3);

    pDC->SelectObject (pOldBrush);
    pDC->SetROP2 (nOldMode);
}

BOOL CWidget::PtInWidget(POINT point)
{
    if (! m_rect.PtInRect (point))
        return FALSE;

    int cx = min (point.x - m_rect.left, m_rect.right - point.x);
    return ((m_rect.bottom - point.y) <= (2 * cx));
}
```

图 19-4　Widget 应用程序

Widgets 是由 CWidget 类的对象代表的，在 WidgetObj.h 和 WidgetObj.cpp 中可以找到其源程序。为了派生 CWidget，我使用 ClassWizard 从 CCmdTarget 中派生类，然后手工编辑源程序对基类 CObject 进行修改。我还修改了由 ClassWizard 插入 SERIAL 宏中的 DYNCREATE 宏，并覆盖了 CObject::Serialize 使 CWidget 成为可串行化的类。这些工作把文档中的 Serialize 函数减小到简单的一行：

```
m_arrWidgets.Serialize (ar);
```

m_arrWidgets 是 CWidgetDoc 成员变量，用来保存 CWidget 指针。在下列情况下会创建 CWidget 对象：从 Insert 菜单中选中命令时；从剪贴板粘贴饰件时；饰件被拖放到 Widget 窗口上时。

CWidget 类具有一对成员函数 Draw 和 DrawSelected，用来给输出设备绘制饰件。Draw 绘制未选中状态下的饰件；DrawSelected 绘制选中状态下的饰件。视图的 OnDraw 程序是一个简单的循环，它从文档中一个接一个地检索 CWidget 指针，并请求每个饰件绘制自身。如果

视图具有输入焦点并且选中了当前饰件(即,如果 CWidgetView::m_nSel != -1),选中的饰件就会在所有其他饰件绘制之后再绘制一次:

```
for (int i = 0; i < nCount; i ++ )
    pDoc -> GetWidget (i) -> Draw (pDC);
if (m_nSel != -1 && CWnd::GetFocus () == this)
    pDoc -> GetWidget (m_nSel) -> DrawSelected (pDC);
```

最后绘制选中的饰件确保了它总是可见的,处于其他饰件之上。

另一个 CWidget 绘图函数是 DrawDragImage,用来绘制拖动的图像。在您在屏幕上拖动饰件时,注意观察跟随光标移动的三角形轮廓。这就是拖动图像。当文件系统对象被拖动时,操作系统命令解释器也使用拖动图像实现了类似的效果。因为在拖放过程中是拖放来源负责显示光标,所以程序员常常认为是拖放来源使拖动图像成为光标的一部分而显示它的。其实一般情况下不是这样。实际上是拖放目标(不是拖放来源)在 OnDragOver 中绘制拖动图像的。为完成此工作,拖放目标必须了解光标所承载的负担种类,以便在屏幕上绘出轮廓。

Widget 处理拖动图像的过程如下:在 OnDragEnter 中创建一个临时饰件对象,在 CWidgetView::m_pTempWidget 中保存指针,并在每次调用 OnDragOver 时都调用对象的 DrawDragImage 函数。实际上,OnDragOver 调用 DrawDragImage 两次:一次删除旧的拖动图像,一次绘制新的拖动图像。DrawDragImage 在 R2_NOT 绘图方式下进行绘制,因此在原来的拖动图像上面绘一个新的可以有效地删除原来的拖动图像。上一个拖动图像的位置被保存在了 CWidgetView 的 m_pointLastImage 数据成员中。当调用 OnDragLeave 或 OnDrop 时,就会删除临时饰件。本例也说明了覆盖 OnDragLeave 有时是很有用的。在本例中,即使释放操作并不发生也必须释放由 OnDragEnter 所分配的资源。

拖动饰件并将其放在视图下边或右边几个像素范围内,您就会看到拖放目标滚动开始动作了。停留短暂的时间以后,视图就会开始滚动,直到释放操作发生或光标从边界上移开时才会停止。利用拖放目标滚动,您不必将手指从鼠标上移开去单击滚动条就可以将饰件放在视图的任何部位。这再次体现出,当拖放目标是 CScrollView 时,用户可以绝对免费地得到拖放目标滚动功能。

19.4.1 AfxOleInit 函数

当我使用 AppWizard 创建 Widget 项目时,在 Step 3 中没有选择任何 OLE 选项。AppWizard 在这种方式下运行时,生成的源程序并不包含对最重要的 AfxOleInit 函数的调用,该函数初始化 OLE 库。此函数必须在 MFC 应用程序以任何方式接触到 COM 或 OLE 之前得到调用。因此,我将对 AfxOleInit 的调用添加到了应用程序类的 InitInstance 函数的开始处。这样我就必须给 Stdafx.h 添加语句

```
#include <afxole.h>
```

否则，对 AfxOleInit 的调用不会得到编译。

我提到这一点主要是为了说明在您编写使用 COM 或 OLE 的应用程序时，如果在 AppWizard中没有选择任何 OLE 选项，那么就必须手工添加 AfxOleInit 调用以及语句 #include <Afxole.h>。如果不这样，您的应用程序可能会得到顺利的编译，但是对函数如 COleDataSource::DoDragDrop 的调用就会失败。我曾经浪费了半个工作日的时间，不理解为什么看上去完全正确的剪贴板程序却不能执行？然后我意识到自己忘了在源程序中添加这些关键的语句了。如果您在编写应用程序时发现调用 DoDragDrop 或其他 OLE 函数时奇怪地失败了，就应该确定一下在应用程序启动时是否调用了 AfxOleInit。这样可以给自己省去许多麻烦。

第 20 章　Automation

想象一下这样的世界，您可以使用复杂而又易于使用的脚本语言来编写任何 Microsoft Windows 应用程序。再进一步，所有的脚本语言都是相同的，只要学会了为一个应用程序编写脚本，也就能够为其他应用程序编写脚本了。当实现这一切的时候，请设想一下如果用 Microsoft Visual C++ 、Microsoft Visual Basic 和其他编程语言编写的程序也可以访问这种假想语言的特性那该多好。

听上去太好了，不可相信是吧？不是。我们要感谢基于 COM 的技术 Automation，因为有了它，任何 MFC 应用程序都可以转换为可用脚本编写的应用程序。Automation 是标准化了的方法，可以将应用程序的特性提供给用 Visual Basic，Visual Basic for Applications（VBA），Visual BasicScripting Edition（VBScript），以及其他语言编写的客户程序。它解决了这些语言不能使用常规的 COM 接口与 COM 对象对话而引起的问题，但是它并没有局限于 Visual Basic 体系的语言中；用 C++ 和其他语言编写的客户也可以使用 Automation。

Microsoft 积极地鼓励应用程序开发者在他们的程序中建立对 Automation 的支持，而且还带头在许多重要的应用程序中创建了 Automation 功能。例如：Microsoft Excel 是一个功能强大的 Automation 服务器。Microsoft Word、Microsoft PowerPoint 以及其他 Microsoft Office 应用程序也是。通过 Automation 公开应用程序的特性，可以使应用程序在想学习脚本编程语言的用户手里成为强大的工具。您同时也给开发者提供了一种用您的应用程序作为平台开发他们自己的应用程序的途径。

Automation 是一种智能型的万能工具，它远远超出了脚本编程的应用范围。它可以用来为 Active Server Pages 创建软件组件，也是 ActiveX 控件中采用的主要技术之一。本章中要学习的内容有：什么是 Automation、它的工作原理以及使用 MFC 编写能够应用 Automation 的应用程序的方法。Visual C++ 程序员可以创建一个简单的 Automation 服务器并立即运行它。如果掌握了一些方法，您就可以创建能提供复杂的层次结构化对象模型的 Automation 服务器。Visual C++ 和 MFC 也简化了创建 Automation 客户的过程，客户指的是利用 Automation 服务器所提供的服务的程序。

20.1　Automation 基础

与通常公开接口和方法的 COM 对象不同，Automation 对象公开了方法和属性。“方法”是可以被客户调用的函数。“属性”是指对象的属性，例如：颜色或文件名。

理解 Automation 的语言如 Visual Basic 对程序员隐藏了引用计数、接口指针以及其他 COM 的习惯用语。它们还允许您就像访问本地子例程和变量一样很容易地访问 Automation 的方法和属性。下列 Visual Basic 语句初始化了一个 Automation 对象并调用了一个名为 Add 的方法执行 2 加 2 操作：

```
Dim Math as Object
Set Math = CreateObject ("Math.Object")
Sum = Math.Add (2, 2)
Set Math = Nothing
```

在本例中，Math 是类型为 Object 的变量。“Math.Object”是 Automation 对象的 ProgID。（回忆第 18 章中的内容，ProgID 是 COM CLSID 的对等字符串。）最后一条语句释放了对象，执行的 Visual Basic 语句与通过接口指针来调用的 Release 等价。

在 Visual Basic 中，访问 Automation 属性与调用 Automation 方法在语法上相似。下例中创建了一个银行账目对象，通过给属性 Balance 赋值来结算账目：

```
Dim Account as Object
Set Account = CreateObject ("BankAccount.Object")
Account.Balance = 100
```

检查账目中的收支差额就相当于简单的读取属性值：

```
Amount = Account.Balance
```

读出和写入 Automation 属性类似于访问 C++ 类中的公用成员变量。事实上，COM 对象和 C++ 对象同不能公开私有数据成员一样不能公开成员变量。这种 Automation 对象可公开其值和方法的错觉正是 Automation 不可思议的一个地方。

用 VBScript 编写的 Automation 客户看上去和用 Visual Basic 编写的 Automation 客户很相似。下列脚本语言使用 VBScript 内置的 FileSystemObject 对象（实际上是一个 Automation 对象）来创建包含字符串“Hello，world”的文本文件：

```
Set fso = CreateObject ("Scripting.FileSystemObject")
Set TextFile = fso.CreateTextFile ("C:\Hello.txt", True)
TextFile.WriteLine ("Hello, world")
TextFile.Close
```

您自己可以试一下这个脚本程序，使用 Notepad 或程序编辑器在文本文件中输入这几行语句并以扩展名 .vbs 将其保存。然后在操作系统的命令解释器中双击文件或在命令提示窗口输入 START filename.vbs。在 Windows 98 和 Windows 2000 系统中，会调用内置的 Windows Scripting Host，它将打开文件并执行在其中找到的语句。

您也可以用 C++ 编写 Automation 客户。下一节中将介绍具体做法，但是要注意 C++ 客户并不好用，它并不具有 Visual Basic 运行程序或脚本编程引擎来作为它和 Automation 服务

器之间的中间人。好消息是 Visual C++ 可以帮助您创建 Automation 客户,基于 MFC 的 COleDispatchDriver 类生成易于使用的封装类。在本章稍后您就会明白我的意思了。

20.1.1 IDispatch:所有 Automation 的基础

Automation 从外边看上去相当简单,对于 Visual Basic 程序员而言,Automation 确实也简单。但事实上是它涉及到了 COM,这就意味着在其内部情况就大为不同了。

理解 Automation 工作的关键是理解 COM 接口 IDispatch。Automation 对象是一个实现了 IDispatch 的 COM 对象。IDispatch 包含 4 个方法(在表 20-1 中列出),不算 3 个对所有 COM 接口都公有的 IUnknown 方法。在 4 个方法中,Invoke 和 GetIDsOfNames 是最重要的。客户调用 Invoke 来调用 Automation 方法来读取或写入 Automation 属性值。Invoke 不接受像"Add"或"Balance"这样的方法或属性名称。相反,它接受整型"调度 ID",或称为 dispid,它用来标识属性或方法。GetIDsOfNames 将属性名字或方法名字转换为传递给 Invoke 的调度 ID。从 dispinterface 中通过 IDispatch 接口同时公开了方法和属性。

表 20-1 IDISPATCH 接口

方法	说 明
Invoke	调用 Automation 方法或访问 Automation 属性
GetIDsOfNames	返回属性或方法的调度 ID
GetTypeInfo	获取 ITypeInfo 指针(如果有效)来访问 Automation 对象的类型信息
GetTypeInfoCount	如果 Automation 对象没有提供类型信息则返回 0;如果提供了则返回 1

当 Visual Basic 遇到下列语句时

```
Sum = Math.Add(2, 2)
```

它将调用 GetIDsOfNames 把"Add"转换为调度 ID。然后调用 Invoke,传递给它从 GetIDsOfNames 得到的调度 ID。在调用 Invoke 之前,Visual Basic 将用方法参数的值初始化一个结构数组,在本例中值为 2 和 2。它将此数组的地址和一个空结构的地址一起传递给 Invoke,该空结构将接收方法的返回值(两个输入参数的和)。Automation 对象会检查调度 ID,如果它与 Add 方法相应,则拆开输入值,把它们相加并将其复制在调用者提供的结构中。

说明此过程的一个好方法是看一下 C++ 程序员调用 Automation 方法的过程。下列代码是上一节第一个例子的 C++ 等价程序:

```
// Convert the ProgID into a CLSID.
CLSID clsid;
::CLSIDFromProgID(OLESTR("Math.Object"), &clsid);

// Create the object, and get a pointer to its IDispatch interface.
IDispatch* pDispatch;
```

```
::CoCreateInstance (clsid, NULL, CLSCTX_SERVER, IID_IDispatch,
    (void * * ) &pDispatch);

// Get the Add method's dispatch ID.
DISPID dispid;
OLECHAR * szName = OLESTR ("Add");
pDispatch -> GetIDsOfNames (IID_NULL, &szName, 1,
    ::GetUserDefaultLCID (), &dispid);

// Prepare an argument list for the Add method.
VARIANTARG args[2];
DISPPARAMS params = { args, NULL, 2, 0 };
for (int i = 0; i < 2; i ++ ) {
    ::VariantInit (&args[i]); // Initialize the VARIANT.
    args[i].vt = VT_I4; // Data type = 32-bit long
    V_I4 (&args[i]) = 2; // Value = 2
}

// Call Add to add 2 and 2.
VARIANT result;
::VariantInit (&result);
pDispatch -> Invoke (dispid, IID_NULL, ::GetUserDefaultLCID (),
    DISPATCH_METHOD, &params, &result, NULL, NULL);

// Extract the result.
long lResult = V_I4 (&result);

// Clear the VARIANTs.
::VariantClear (&args[0]);
::VariantClear (&args[1]);
::VariantClear (&result);

// Release the Automation object.
pDispatch -> Release ();
```

可以明白地看到对 IDispatch::GetIDsOfNames 和 IDispatch::Invoke 的调用，以及创建 Automation对象的::CoCreateInstance 语句。还可以看到输入和输出参数被封装在结构 VARIANTARG中了，关于此内容我们在下一节讨论。

我们使用 IDispatch::Invoke 访问 Automation 属性。第 4 个参数在上例中等于 DISPATCH_METHOD，您可以把它设置为 DISPATCH_PROPERTYPUT 或 DISPATCH_PROPERTYGET 来说明参数 1 中调度 ID 命名的属性值正在被设置或检索。另外，IDispatch::Invoke 可以在调用者提供的 EXCEPINFO 结构中返回出错信息。在 Invoke 的第 7 个参数中传递此结构的地址，NULL 指针意味着调用者并不关心这样的信息。Invoke 还支持带有可选的和已命名参数的Automation方法和属性，这一点对于 C++ 客户关系不大，但却可以简化用 Visual Basic 编写的客户程序。

从这些例子中我们可以明白地看出 Automation 需要 C++ 程序员做更多的工作。对于 C++ 客户,调用常规的 COM 方法要比调用 Automation 方法更迅速更高效。在 Visual Basic 中 Automation 显得很简单,原因是 Visual Basic 为了使它显得简单而做了许多工作。但是把表象剥去,Automation 看上去就完全不同了。

20.1.2 Automation 数据类型

IDispatch::Invoke 的一个更有趣的内容是它处理输入输出参数的方法。在 Automation 中,所有的参数都是在称为 VARIANT 的数据结构中传递的。(技术上讲,输入参数是在 VARIANTARG 中传递的,而输出参数是在 VARIANT 中传递的,但是由于这些结构相同,程序开发者通常使用术语 VARIANT 来描述它们。)从本质上说,VARIANT 是自我描述数据类型。在 VARIANT 内部是一个数据类型联合用来保存 VARIANT 的数据,还包括一个分开的字段用来定义数据类型。下面给出在 Oaidl.idl 中结构的定义:

```
struct tagVARIANT {
    union {
        struct __tagVARIANT {
            VARTYPE vt;
            WORD    wReserved1;
            WORD    wReserved2;
            WORD    wReserved3;
            union {
                LONG            lVal;          /* VT_I4              */
                BYTE            bVal;          /* VT_UI1             */
                SHORT           iVal;          /* VT_I2              */
                FLOAT           fltVal;        /* VT_R4              */
                DOUBLE          dblVal;        /* VT_R8              */
                VARIANT_BOOL    boolVal;       /* VT_BOOL            */
                _VARIANT_BOOL   bool;          /* (obsolete)         */
                SCODE           scode;         /* VT_ERROR           */
                CY              cyVal;         /* VT_CY              */
                DATE            date;          /* VT_DATE            */
                BSTR            bstrVal;       /* VT_BSTR            */
                IUnknown *      punkVal;       /* VT_UNKNOWN         */
                IDispatch *     pdispVal;      /* VT_DISPATCH        */
                SAFEARRAY *     parray;        /* VT_ARRAY           */
                BYTE *          pbVal;         /* VT_BYREF T_UI1     */
                SHORT *         piVal;         /* VT_ BYREF T_I2     */
                LONG *          plVal;         /* VT_BYREF T_I4      */
                FLOAT *         pfltVal;       /* VT_BYRE T_R4       */
```

```
            DOUBLE *          pdblVal;          /* VT_BYREF T_R8          */
            VARIANT_BOOL *    pboolVal;         /* VT_BYREF T_BOOL        */
             VARIANT_BOOL *   pbool;            /* (obsolete)             */
            SCODE *           pscode;           /* VT_BYREF T_ERROR       */
            CY *              pcyVal;           /* VT_BYREF T_CY          */
            DATE *            pdate;            /* VT_BYREF T_DATE        */
            BSTR *            pbstrVal;         /* VT_BYREF T_BSTR        */
            IUnknown * *      ppunkVal;         /* VT_BYREF T_UNKNOWN     */
            IDispatch * *     ppdispVal;        /* VT_BYREF T_DISPATCH    */
            SAFEARRAY * *     pparray;          /* VT_BYREF T_ARRAY       */
            VARIANT *         pvarVal;          /* VT_BYREF T_VARIANT     */
            PVOID             byref;            /* Generic ByRef          */
            CHAR              cVal;             /* VT_I1                  */
            USHORT            uiVal;            /* VT_UI2                 */
            ULONG             ulVal;            /* VT_UI4                 */
            INT               intVal;           /* VT_INT                 */
            UINT              uintVal;          /* VT_UINT                */
            DECIMAL *         pdecVal;          /* VT_BYREF T_DECIMAL     */
            CHAR *            pcVal;            /* VT_BYREF T_I1          */
            USHORT *          puiVal;           /* VT_BYREF T_UI2         */
            ULONG *           pulVal;           /* VT_BYREF T_UI4         */
            INT *             pintVal;          /* VT_BYREF T_INT         */
            UINT *            puintVal;         /* VT_BYREF T_UINT        */
            struct __tagBRECORD {
                PVOID pvRecord;
                IRecordInfo * pRecInfo;
            }   VARIANT_NAME_4;    /* VT_RECORD                         */
        } __VARIANT_NAME_3;
    } __VARIANT_NAME_2;

    DECIMAL decVal;
} __VARIANT_NAME_1;
};
```

vt 域保存了标识数据类型的一个以上的 VT_标志。另一个域保存了实际的值。例如：代表 32 位长整型、值为 128 的 VARIANT，它的 vt 等于 VT_I4 而 lVal 等于 128。头文件 Oleauto.h 中定义了宏用来读取和写入封装在 VARIANT 中的数据。另外，系统文件 Oleaut32.dll 中包含了 API 函数，如::VariantInit 和::VariantClear，可以用来管理和操纵 VARIANT，并且 MFC 的 COleVariant 类为 VARIANT 数据结构和在其上执行的操作进行了友好的封装。

当您编写 Automation 对象时，对于所有对象的属性和方法必须使用与 Automation 兼容的数据类型，也就是可以用 VARIANT 代表的数据类型。表 20-2 中总结了有效的数据类型。

表 20-2 与 VARIANT 兼容的数据类型

数据类型	说 明
BSTR	Automation 字符串
BSTR *	指向 Automation 字符串的指针
BYTE	8 位字节
BYTE *	指向 8 位字节的指针
CHAR	8 位字符
CHAR *	指向 8 位字符的指针
CY	64 位货币值
CY *	指向 64 位货币值的指针
DATE	64 位日期和时间值
DATE *	指向 64 位日期和时间值的指针
DECIMAL *	指向 DECIMAL 数据结构的指针
DOUBLE	双精度浮点值
DOUBLE *	指向双精度浮点值的指针
FLOAT	单精度浮点值
FLOAT *	指向单精度浮点值的指针
IDispatch *	IDispatch 接口指针
IDispatch * *	指向 IDispatch 接口指针的指针
INT	(在 Win32 平台上是 32 位)
INT *	指向带符号整型的指针
IUnknown *	COM 接口指针
IUnknown * *	指向 COM 接口指针的指针
LONG	32 位带符号整型
LONG *	指向 32 位带符号整型的指针
PVOID	无类型指针
SAFEARRAY *	SAFEARRAY 指针
SAFEARRAY * *	指向 SAFEARRAY 指针的指针
SCODE	COM HRESULT
SCODE *	指向 COM HRESULT 的指针
SHORT	16 位带符号整型
SHORT *	指向 16 位带符号整型的指针
UINT	无符号整型
UINT *	指向无符号整型的指针
ULONG	32 位无符号整型
ULONG *	指向 32 位无符号整型的指针

续表

数据类型	说 明
USHORT	16 位无符号整型
USHORT *	指向 16 位无符号整型的指针
VARIANT *	指向 VARIANT 数据结构的指针
VARIANT_BOOL	Automation Boolean
VARIANT_BOOL *	指向 Automation Boolean 的指针

一般来说，Automation 对于 VARIANT 可兼容数据类型的依赖是一种限制，它约束了习惯于创建“纯粹”COM 对象的程序开发者，所谓“纯粹”COM 对象，是指使用常规接口而不是调度接口的对象，因此在所能使用的数据类型方面它们的约束也较小。但是，只使用与 Automation兼容的数据类型有一个好处：COM 知道如何去调度 VARIANT，因此 Automation 对象并不要求有自定义的名称解析代理/桩基模块 DLL。而付出的代价则是在方法的参数列表中不能使用结构体(或指向结构体的指针)，并且使用数组也要求特殊的处理，因为它们必须封装在结构 SAFEARRAY 中。

BSTR 数据类型

在上一节中提供的大多数数据类型都是自解释的。但是，其中还有两个值得进一步解释。BSTR 是 Automation 的字符串数据类型。与 C++ 字符串不同，C++ 字符串是一个带有 0 定界符的字符数组，而 BSTR 是一个计数字符串。前四个字节保存了字符串中的字节数(不是字符数)；以后的字节保存字符本身。在 BSTR 中的所有字符都是 16 位 Unicode 字符。BSTR 的值实际上是指向字符串中第一个字符的指针。(参见图 20-1。)事实上，字符串是 0 定界的，这意味着通过将其强制转换为 LPCWSTR 就可将 BSTR 转变为 C++ 字符串指针。

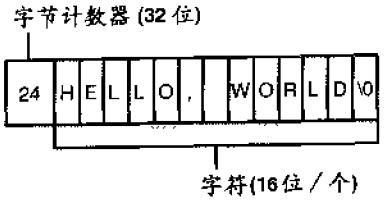


图 20-1 BSTR 数据类型

在 MFC 中，处理 BSTR 通常意味着将 CString 转换为 BSTR 以及将 BSTR 转换为 CString。CString::AllocSysString 从 CString 生成一个 BSTR：

```
CString string = _T("Hello, world");
BSTR bstr = string.AllocSysString();
```

如果没有定义预处理器符号 _UNICODE,则表明这是一个 ANSI 程序结构,那么 AllocSysString 会自动将 ANSI 字符转换为 Unicode 字符。CString 还包含一个成员函数 SetSysString,可以用来修改已存在的 BSTR。

将 BSTR 转换成 CString 也同样容易。CString 的 LPCWSTR 运算符根据 BSTR 初始化 CString,并且如果 CString 具有 ANSI 类型,就可以很方便地将字符转换为 8 位 ANSI 字符:

```
CString string = (LPCWSTR) bstr;
```

要小心,如果 BSTR 包含嵌入的 0 值时(非常可能,因为 BSTR 是计数字符串),把它按此方法转换为 CString 实际上就会截断字符串。

SAFEARRAY 数据类型

SAFEARRAY 是 Automation 的数组数据类型。它被称为"安全"数组是因为除了包含有组成数组元素的数据以外,它还保存着有关数组中的维数、每一维上的尺寸边界等等信息。

SAFEARRAY 实际上是一个结构体。在 Oaidl.h 中它的定义如下:

```
typedef struct tagSAFEARRAY
    {
    USHORT cDims;
    USHORT fFeatures;
    ULONG cbElements;
    ULONG cLocks;
    PVOID pvData;
    SAFEARRAYBOUND rgsabound[ 1 ];
    } SAFEARRAY;
```

SAFEARRAYBOUND 也是一个结构,定义如下:

```
typedef struct tagSAFEARRAYBOUND
    {
    ULONG cElements;
    LONG lLbound;
    } SAFEARRAYBOUND;
```

cDims 字段保存着 SAFEARRAY 中的维数。rgsabound 是嵌入数组,它为每一维都包含一个元素。每个元素都定义了每一维的边界(存储单元的数量)以及每维中下边界的下标。与 C++ 数组不同(它从 0 到 n 给元素编号),SAFEARRAY 的元素可以用任意连续整数来编号,例如从 -5 到 n-5。fFeatures 保存着一个标志,指定了 SAFEARRAY 保存的数据种类和 SAFEARRAY 的分配方式。cbElements 以字节为单位保存着每个元素的大小。最后,pvData 指向元素自身。

Windows API 中包含了许多可以用来创建和使用 SAFEARRAY 的函数;所有函数都以名字 SafeArray 打头。MFC 有自己的处理 SAFEARRAY 的方式,以 COleSafeArray 类的形式实现。下列

程序创建一个 COleSafeArray 对象，代表着包含整数 1 到 10 的一维 SAFEARRAY：

```
COleSafeArray sa;
LONG lValues[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
sa.CreateOneDim(VT_I4, 10, lValues);
```

SAFEARRAY 所在的 VARIANT 数据结构的地址可以用 COleSafeArray 的 LPVARIANT 或 LPCVARIANT 运算符获得：

```
VARIANT* pVariant = (LPVARIANT) sa;
```

使用 COleSafeArray 创建一维数组相对比较容易，要是创建多维数组就要花更大力气了，可以肯定地说，即使是 COleSafeArray 也不能使 SAFEARRAY 像普通数组那样符合 C++ 程序员的胃口。

20.1.3 滞后绑定与超前绑定的对比

一个 C++ 程序员在第一次看到 Automation 时可能会感到奇怪，为什么会存在调度接口呢？其中原因对他们很模糊。如果说 Automation 对象本质上就很复杂并且使用效率也比不上常规 COM 对象，那么为什么不使用自定义 COM 接口而非得使用 IDispatch 接口呢？那样的话可以给程序开发者节省许多时间，减少许多麻烦。

创建调度接口是为了在 Visual Basic 不能完全支持常规 COM 接口时，能允许 Visual Basic 程序员去使用 COM 对象。在它的早期阶段，Visual Basic 并不能通过普通接口指针调用 COM 方法。当前 Visual Basic 版本已经部份消除这种限制了，但是许多脚本编程语言，包括 VBScript，也只能通过 IDispatch 接口与 COM 对象对话。

那么 IDispatch 有什么特殊之处呢？在内部，它避免了编译器（或注释器，看情况而定）必须去理解虚函数表。COM 接口指针实际上是指向对象内部某处的指针，该对象保存着一个函数指针表格（C++ 的习惯用语，指的是虚函数表，或称为 vtable）的地址。如果 pMath 保存着一个 IMath 接口指针，当 C++ 编译器遇到如下语句

```
pMath->Add(2, 2, &sum);
```

它将解析调用，执行一段程序来从接口的 vtable 中提取 Add 方法的地址，因为接口的定义包含在了头文件中，所以它了解 vtable 的布局。但是这里就有问题了。脚本编程语言并不理解 C++ 接口定义。这些语言不能解析如下语句，除非它们可以把方法名称传递给对象并请求对象自身解析调用：

```
Sum = Math.Add(2, 2)
```

脚本编程语言虽然不理解 vtables，但是它们却对 IDispatch 非常熟悉。给定一个指向 IDispatch 接口的指针，它们就知道在 vtable 中何处可以找到 GetIDsOfNames 和 Invoke 的地址，

所以,对它们而言调用 IDispatch::Invoke 并在运行时"联编"一个方法非常简单。

这就是 IDispatch 的关键:将解析方法调用的责任从客户移交给了对象。程序员称之为"滞后绑定",因为实际上联编是在运行时进行的。相反,我们称 C++ 客户采用了"超前绑定",因为解析方法调用所要求的大量工作是在编译时完成的。

20.1.4 双接口

滞后绑定的缺点在于它要求在运行时进行查找操作,而超前绑定不必如此。这样会对运行效率产生负面影响。并且由于 IDispatch 依赖于调度 ID,每次属性或方法的调用名义上都要求到服务器上的两次往返过程:一次是调用 GetIDsOfNames,接下来再调用 Invoke。一个智能型的 Automation 客户可以通过高速存储调度 ID 来减少往返次数,但是本质上讲滞后绑定要比超前绑定效率低。

在 IDispatch 接口和常规 COM 接口之间的选择其实就是对灵活性和速度的选择。通过 IDispatch 接口公开其特性的对象可以对多种多样的客户进行服务,而使用普通 COM 接口的对象对滞后绑定的客户(特别是 C++ 客户)而言服务效率更高。

双接口是一种 COM 接口,它从 IDispatch 派生而来。它的 vtable 为 IDispatch 方法(GetIDsOfNames、Invoke 等)以及自定义方法包含了输入项。图 20-2 说明了双接口中 vtable 的布局,它允许间接地通过 IDispatch::Invoke 来访问方法 Add 和 Subtract,也允许直接地通过 vtable 进行访问。那些只依赖于 IDispatch 的客户可以通过 IDispatch::Invoke 调用 Add 和 Subtract;它们甚至不会意识到有自定义的 vtable 部分存在。另一方面,C++ 客户实际上会忽视 vtable 中的 IDispatch 部分并采用超前绑定来调用 Add 和 Subtract。这样,同一个对象就可以同时支持超前和滞后绑定了。注意在 vtable 中自定义部分定义的方法必须使用与 Automation 兼容的数据类型,正如同通过 IDispatch::Invoke 公开的方法一样。

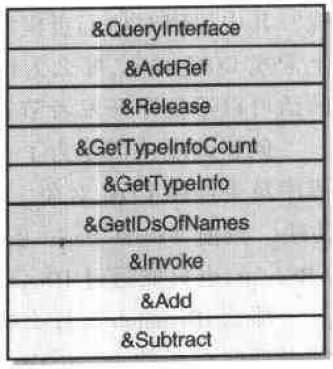


图 20-2 双接口的虚函数表

对于 MFC 程序员,使用双接口的最大阻碍在于要花大量的精力去实现它。MFC Technical Note 65 描述了给 MFC Automation 服务器添加双接口的方法,但是它讲的整个过程对于心脏不好的人来说很不适合。当今最好的办法应该是放弃 MFC 而使用 Active Template Library (ATL),它真正实现了可以毫不费力地创建双接口。

20.1.5 类型库

大多数 Automation 服务器都伴有类型库。在《Inside COM》(1997, Microsoft Press)一书中,Dale Rogerson 将类型库描述为"有关接口和组件信息的口袋"。给定一个类型库,客户就可

以找到对 COM 对象感兴趣的所有事情，包括它实现的接口、在接口中存在的方法，以及每个方法的参数列表的详细情况。可以在一个独立的文件中提供类型库(常用扩展名.tlb，有时也使用.olb)或是把它作为资源嵌入对象的可执行文件中。无论用什么方法封装它，类型库都要在注册表中注册以便客户可以轻易地找到它。

可以以不同的方式使用类型库。例如：ActiveX 控件使用类型信息(在类型库中找到的一种数据)告诉控件容器它们可以激发何种事件。类型库还可以用来实现 IDispatch 接口并给对象浏览器提供信息。但是类型库对于 Automation 程序员之所以重要的一个原因，是它们可以使 Visual Basic 客户使用双接口的自定义部分访问服务器的 Automation 方法和属性。给定类型信息，当今的 Visual Basic 程序甚至可以使用常规的 COM 对象了——那些通过自定义 COM 接口而不是 IDispatch 接口公开其功能的对象。但是类型库也不仅仅用于 Visual Basic 用户，C++ 程序员也可以使用它们。很快您就会看到如何使用 ClassWizard 生成封装类来简化 MFC Automation 客户的编写工作。值得注意的是只有类型库有效时 ClassWizard 才可以发挥它的功能。

类型库是怎样创建的呢？您可以使用 COM API 函数和方法通过编程来创建它们，但是大多数类型库是从 IDL 文件中创建的。MIDL 会阅读 IDL 文件并根据其中语句生成类型库。您还可以在 ODL 文件中定义对象和它们的接口，并通过调用特殊的工具 MkTypeLib 编译该文件从而得到类型库。IDL 文件是首选的方法，但是对于 MFC Automation 服务器，Visual C++ 仍然使用的是 ODL 文件。下列 ODL 语句定义了一个类型库，其中描述了一个 Automation 组件 Math 以及调度接口 IAutoMath：

```
[uuid(B617CC83-3C57-11D2-8E53-006008A82731), version(1.0)]
library AutoMath
{
    importlib("stdole32.tlb");
    [uuid(B617CC84-3C57-11D2-8E53-006008A82731)]
    dispinterface IAutoMath
    {
        properties:
            [id(1)] double Pi;
        methods:
            [id(2)] long Add(long a, long b);
            [id(3)] long Subtract(long a, long b);
    };
    [uuid(B617CC82-3C57-11D2-8E53-006008A82731)]
    coclass Math
    {
        [default] dispinterface IAutoMath;
    };
};
```

ODL 中的 importlib 语句类似于 C++ 中的 # include。uuid 将 GUID 赋给了对象或接口，dispinterface 定义了一个调度接口。在 dispinterface 块中的语句声明了 Automation 方法和属性以及它们的调度 ID。本例中的对象具有一个名为 Pi 的属性和方法 Add 及 Subtract。它们的调度 ID 分别为 1、2 和 3。

在您编写 MFC Automation 服务器时，AppWizard 将创建一个 ODL 文件并将其添加到项目中。每次添加方法或属性时，ClassWizard 都会修改 ODL 文件，以便下次编译时会更新类型库。只要您是使用 MFC 向导制作 MFC Automation 服务器，类型库自然就会成为创建过程产生结果的一部分，完全不需要额外的工作。

20.2 MFC Automation 服务器

可以使用 MFC 编写独立的 Automation 组件，但是更常见的情况是使用它对 Automation 的支持将应用程序的特性公开提供给 Automation 客户。通过此方式公开特性可以使应用程序成为可用脚本编程的程序并得到最佳效果。

为了编写 MFC Automation 服务器，您不必是 IDispatch 接口和 VARIANT 方面的专家，因为 MFC 已经把其方法和属性封装得与普通类成员函数的相应部分相似。实际上，编写 MFC Automation 服务器是如此简单，使得 Visual C++ 程序员通常在一些使普通 COM 对象为难的地方使用 Automation 组件。

编写 MFC Automation 服务器简单的原因是向导 AppWizard 添加了将应用程序变换为 Automation 服务器所需的全部基础内容。ClassWizard 将添加方法和属性的工作简化到了只要单击几个按钮即可。由这些向导生成的程序很大程度上依赖于 MFC 中已经提供的对 Automation的支持。让我们在讨论创建 Automation 服务器所需要的步骤之前，先来看一下 MFC内部操作，看看它是如何使 Automation 成为可能的。

20.2.1 MFC、IDispatch 和调度映射

MFC 对 Automation 服务器支持的基础是内置的 IDispatch 实现。此实现来自于类 COleDispatchImpl，通过 CCmdTarget::EnableAutomation 函数该类被实例化并合并到了 CCmdTarget 对象中。这就意味着支持 Automation 的 MFC 类必须直接或间接地从 CCmdTarget 进行派生。通常在类的构造函数中调用 EnableAutomation。

在调用 MFC 的 IDispatch::Invoke 实现时，MFC 必须一定程度上将方法调用和属性访问转换为对类成员函数的调用。与此相似，在调用 IDispatch::GetIDsOfNames 时，MFC 必须将伴随的属性和方法名称转换为调度 ID。这两个任务都是使用“调度映射”(dispatchmap)来完成的。

调度映射是以 BEGIN_DISPATCH_MAP 开头而以 END_DISPATCH_MAP 结束的表。它们之间的语句定义了对象的方法和属性。通过调度映射，MFC 对 IDispatch::Invoke 的实现

将对 Automation 方法的调用变换为了对调度映射表内的类成员函数的调用。Automation 属性也是通过调度映射被访问的。在下列调度映射表中定义了 CCmdTarget 派生类 CAutoClass 中的方法 DebitAccount 和属性 Balance：

```
BEGIN_DISPATCH_MAP(CAutoClass, CCmdTarget)
    DISP_FUNCTION(CAutoClass, "DebitAccount", Debit, VT_I4, VTS_I4)
    DISP_PROPERTY_EX(CAutoClass, "Balance", GetBalance, SetBalance,
        VT_I4)
END_DISPATCH_MAP()
```

DISP_FUNCTION 宏命名了 Automation 方法和在调用该方法时调用的成员函数。在宏参数列表中传递的 VT_和 VTS_值标识了方法的返回值类型和接受参数的类型。DISP_PROPERTY_EX 定义了 Automation 属性和用来读取和写入属性值的获取和设置函数。DISP_PROPERTY_EX 的第 5 个参数定义了属性的类型。在本例中，当 Automation 对象的 DebitAccount 方法被调用时就要调用 CAutoClass::Debit。应该调用 CAutoClass::GetBalance 来读取 Balance 的值，调用 CAutoClass::SetBalance 来把一个值赋给它。DISP_FUNCTION 和 DISP_PROPERTY_EX 仅仅是在 Afxdisp.h 内定义了的几个调度映射宏中的两个。

您可能已经注意到了在上段给出的调度映射宏中没有一个接受调度 ID。基于它们在调度映射表中的位置和派生的深度，MFC 使用一种古怪的方式给方法和属性分配调度 ID。MFC 的"Technical Note 39"具有详细的说明。在调度映射表中项目的位置关系对于 Automation 程序员来说具有重要的含义：这些项目的顺序必须与 ODL 文件中的调度 ID 一致。这就意味着如果您手工编辑向导生成的调度映射表并以任何方式修改了项目顺序之后，都必须编辑 ODL 文件，您可以编辑调度映射表并把未修改的 ODL 文件留给使用滞后绑定的客户，但是对于超前绑定的客户，如果类型库和 IDispatch 不一致就会产生可怕的混乱。因此，MFC 提供了可选的接受调度 ID 的调度映射宏；在 Technical Note 39 中对它们也进行了说明。您仍然需要确保调度映射表中的调度 ID 与 ODL 文件保持一致，但是用这些宏创建的调度映射表中的语句顺序就不重要了。ClassWizard 并没有使用调度 ID 宏，因此如果您利用它们，就必须自己编写。

20.2.2 编写 Automation 服务器

您也可以手工编写调度映射表，但是让 ClassWizard 为您编写会更加方便。下面给出编写 MFC Automation 服务器的 3 个基本步骤：

1. 运行 AppWizard 并在 Step 3 对话框中选中 Automation 复选框(如果在 Step 1 中选用了 Dialog Based 则是 Step 2 对话框)，如图 20-3 所示。在 Step 4 对话框中，单击 Advanced按钮并在 File Type ID 框中输入服务器的 ProgID(参见图 20-4)。
2. 使用 ClassWizard 的 Automation 页中的 Add Method 按钮添加 Automation 方法。(参见图20-5。)

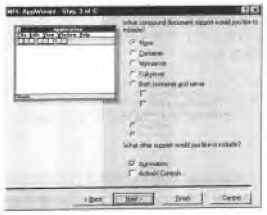


图 20-3 创建一个 MFC Automation 服务器

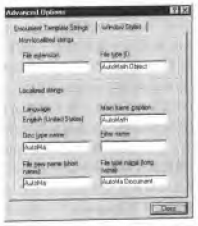


图 20-4 指定 Automation 服务器的 ProgID

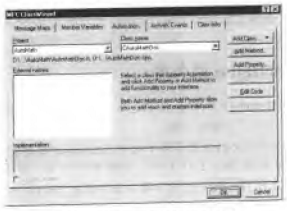


图 20-5 ClassWizard 的 Automation 页

3. 使用 ClassWizard 的 Automation 页中的 Add Property 按钮添加 Automation 属性。

在默认状态下，AppWizard 创建的应用程序所提供的类中只有一个类可以添加 Automation 属性和方法。对于文档/视图应用程序，那个类就是文档类。对于基于对话框的应用程序则是代理类，它们从 CCmdTarget 中派生得到并归属于对话框。为什么这些类是唯一支持 Automation 方法和属性的类呢？因为只有在这些类中 AppWizard 才赋予了作为 Automation 对象必要的基础结构。在本章稍后，您会学习到给 MFC Automation 服务器添加其他 Automatable 类的方法，以便随您的喜好拥有多个 Automation 对象。

20.2.3 添加 Automation 方法

给 MFC Automation 服务器添加 Automation 方法很简单，只要单击 ClassWizard 的 Add Method 按钮并填写 Add Method 对话框即可。（参见图 20-6。）在对话框中，External Name 指的是 Automation 方法的名字，Internal Name 指的是相应成员函数的名字。两个名字不必非得相同，但是大多数时候它们是一样的。Return Type 指的是方法的返回类型；可以是与任何 Automation兼容的数据类型。在 Parameter List 框中定义了方法的参数。MFC 会处理其他事务，如解开包含方法参数的 VARIANTARG，以及在传递给 IDispatch::Invoke 的 VARIANT 中封装方法的返回值。

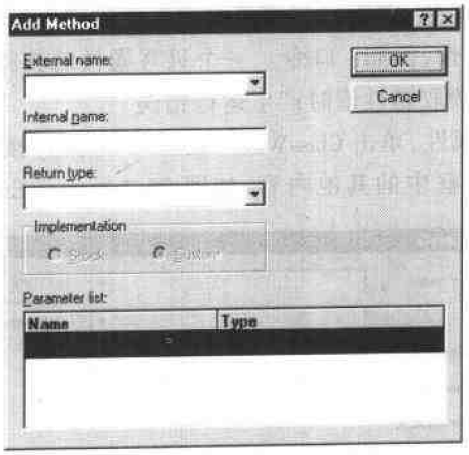


图 20-6 ClassWizard 的 Add Method 对话框

在添加 Automation 方法时，ClassWizard 会对项目的源程序文件进行 4 处改动：

- 在类的头文件中声明实现方法的函数。
- 将一个空的函数实现添加到类的 CPP 文件中。
- 在类的调度映射表中添加一个 DISP_FUNCTION 语句。
- 在项目的 ODL 文件中添加方法及其调度 ID。

在 ClassWizard 完成它的工作以后,您的工作就是在空函数体中填写内容来实现方法。

20.2.4 添加 Automation 属性

还可以使用 ClassWizard 添加 Automation 属性。区分两种类型的属性:

- 成员变量属性
- 获得/设置属性

成员变量属性公开一个成员变量作为 Automation 属性。获得/设置属性是由源程序中获得和设置函数实现的属性。如果属性值允许自身保存在类成员变量中,并且 Automation 服务器不需要控制赋给这些属性的值时,使用成员变量属性较合适。如果满足下列条件之一就应该使用获得/设置属性:

- 属性值不能保存在简单的成员变量中。例如:控制 Automation 服务器窗口的 Visible 属性通常被作为获得/设置属性,以便获得函数可以调用 CWnd::IsWindowVisible,而设置函数可以调用 CWnd::ShowWindow。
- 服务器想要控制赋给属性的值。例如:如果合法的值域是从 1 到 10 的话,设置函数可以将属性值限制在此范围内。
- 属性为只读属性。这种情况下,设置函数调用 Automatable 类从 CCmdTarget 继承来的 SetNotSupported 函数,在客户试图修改属性值时产生运行错误。
- 属性为只能写入属性,例如:口令。一个只写属性的获得函数应该调用 GetNotSupported 在客户试图读取属性值时产生运行错误。

要添加一个成员变量属性,单击 ClassWizard 的 Add Property 按钮并选择 Member Variable。然后填写 Add Property 对话框中的其他内容,如图 20-7 所示。External Name 指定了属性名

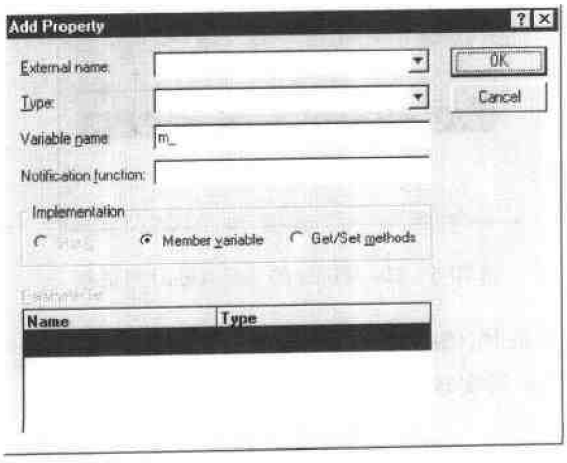


图 20-7 添加一个成员变量 Automation 属性

称。Type 指的是属性的 Automation 兼容数据类型。Variable Name 标识保存属性值的成员变量。ClassWizard 会为您添加此成员变量并把它放在调度映射表中。Notification Function 指定了成员函数的名称，当客户给属性赋值时要调用该函数。您可以在框中填写任意的名字，ClassWizard 会为您添加此函数。如果您不关心属性值是否修改，可以留下此框不填，这样就不会添加任何通知函数了。当您想要立即响应属性值的修改时通知函数就有用了，例如：进行窗口重绘，该窗口的背景颜色被用来作为成员变量属性。

在内部，当添加了不带通知函数的成员变量属性时，ClassWizard 会给类的调度映射表添加 DISP_PROPERTY 语句，而当添加了带有通知函数的成员变量属性时则会添加 DISP_PROPERTY_NOTIFY。它还要在项目的 ODL 文件中声明属性。

如果 Add Property 对话框的 Get/Set Methods 选项被选中，ClassWizard 就会给 Automation 服务器添加一个获得/设置属性。（参见图 20-8。）除了给 Automation 类添加成员函数 GetPropertyName 和 SetPropertyName 以及在 ODL 文件中声明属性以外，ClassWizard 还会在类的调度映射表中添加 DISP_PROPERTY_EX 或 DISP_PROPERTY_PARAM 语句。DISP_PROPERTY_PARAM 定义了带参数的属性；DISP_PROPERTY_EX 定义了不带参数的属性。如果在 Parameter List 中定义了参数，客户在读取或写入属性时要提供这些参数。Automation 服务器有时使用带有参数的获得/设置属性来实现“索引属性”，在本章稍后“更复杂的 Automation 服务器”部分将讨论它。

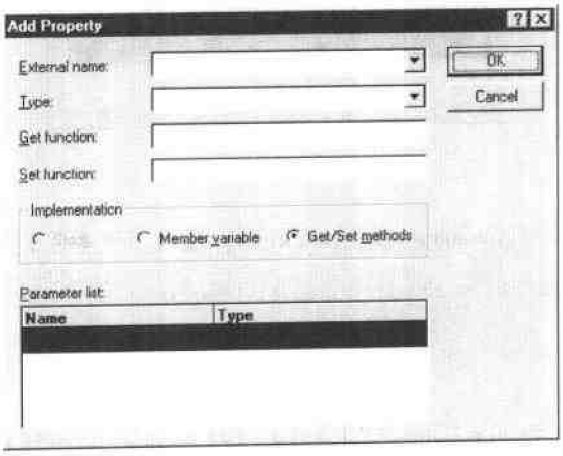


图 20-8 添加一个获得/设置 Automation 属性

20.2.5 简单的 Automation 服务器

为了使您熟悉编写 MFC Automation 服务器，试着做一个简单的练习：

1. 使用 AppWizard 启动一个新项目 AutoMath。在 AppWizard 的 Step 1 对话框中选择

Single Document 使服务器成为单文档界面(SDI)应用程序。在 Step 3 中选中 Automation 框使应用程序成为 Automation 服务器,并在 Step 4 中单击 Advanced 按钮,在 File Type ID 框中输入"AutoMath.Object"。这就是 Automation 对象的 ProgID。

2. 在 ClassWizard 的 Automation 页上,从 Class Name 下拉列表中选择 CAutoMathDoc,单击 Add Method,并在 Add Method 对话框中填写图 20-9 所示内容。单击 OK,然后单击 Edit Code 转移到方法的空函数体部分,输入以下代码:

```
long CAutoMathDoc::Add (long a, long b)
{
    return a + b;
}
```

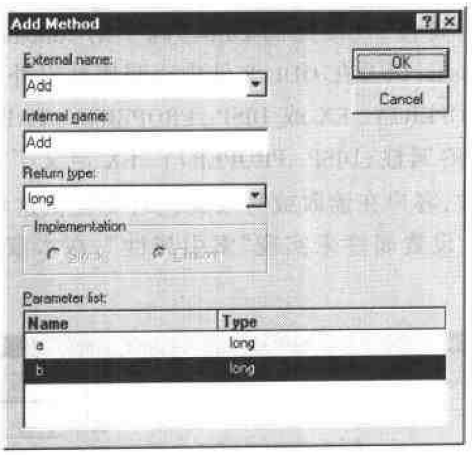


图 20-9 添加 Add 方法

3. 重复第 2 步添加 Automation 方法 Subtract。输入以下程序代码:

```
long CAutoMathDoc::Subtract (long a, long b)
{
    return a - b;
}
```

4. 在 ClassWizard 的 Automation 页上,单击 Add Property 并添加获得/设置属性 Pi。(参见图 20-10。)实现属性的获得和设置函数如下:

```
double CAutoMathDoc::GetPi ()
{
    return 3.1415926;
}

void CAutoMathDoc::SetPi (double newValue)
{
```

```
    SetNotSupported();
}
```

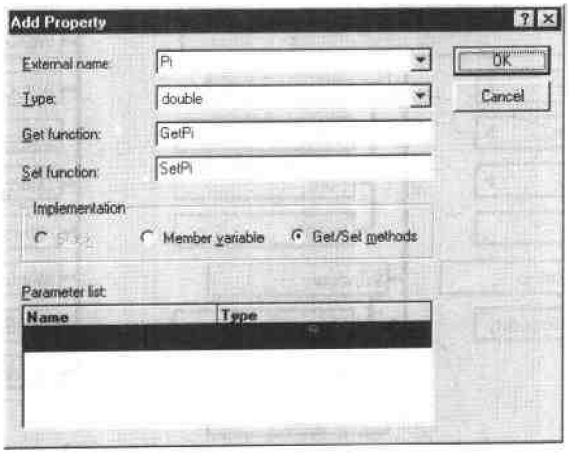


图 20-10 添加 Pi 属性

5. 创建应用程序并运行一次，使得它在系统上得到注册。（MFC Automation 服务器每次运行时都要注册一次。注册涉及到将服务器的 ProgID 和其他信息写入宿主系统的注册表中。）

现在您可以准备测试刚创建的 AutoMath 服务器了。为了进行测试，应该在文本文件 Test.vbs 中输入下列 VBScript 语句：

```
Set Math = CreateObject("AutoMath.Object")
Sum = Math.Add(2, 2)
MsgBox("2 + 2 = " + CStr(Sum))
MsgBox("pi = " + CStr(Math.Pi))
```

然后双击 Test.vbs 文件图标来执行脚本。这样就会在 Windows Scripting Host 的管理下运行脚本了。有两个消息框会出现在屏幕上。第 1 个显示 2 加 2 的和。第 2 个显示 pi 的值。

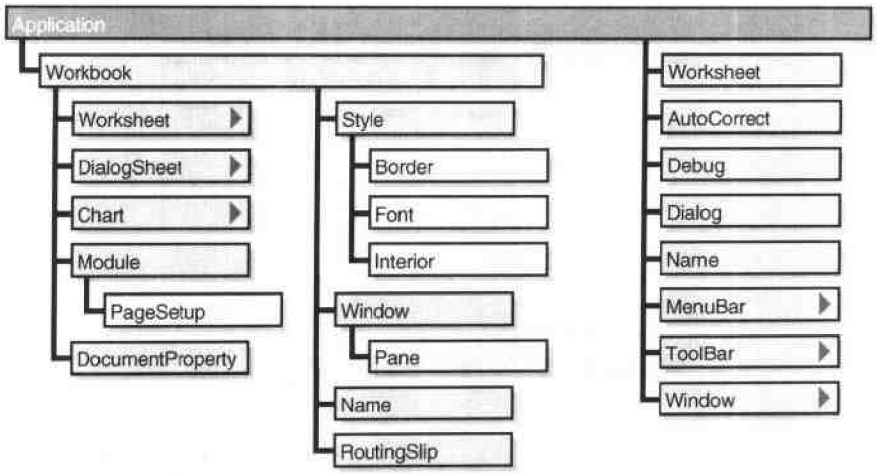
怎么样？使用 MFC 编写 Automation 够容易吧！

20.2.6 Automation 的分层结构

您可以创建任意复杂的 Automation 服务器，只要给它无限地添加方法和属性即可。但是如果负担了太多的方法和属性，Automation 服务器就会变得非常难于处理。这就是 Automation 程序员通常采用 Automation 分层结构“目标化”其服务器特性设置的原因。

Automation 分层结构就是一组 Automation 对象结合起来形成的树形结构对象模型。图 20-11 给出了 Microsoft Excel 的 Automation 分层结构上面四层的结构。Excel 并没有把所有方法和属性都集中在一个对象中，而是把它们划分在了顶层 Application 对象和子对象中。下面的 Visual Basic

程序启动 Excel 并打开了 Caps Lock Correct 功能,使 Excel 可以更正字的大小写错误:



▶ 表示该对象还包括一个或多个子对象没有显示

图 20-11 Excel 对象模型

```
Dim Excel as Object
Set Excel = CreateObject ("Excel.Application")
Excel.AutoCorrect.CorrectCapsLock = 1
```

Caps Lock Correct 作为 AutoCorrect 对象的属性提供给了 Automation 客户。而 AutoCorrect 又是 Application 对象的子对象。像这样的分层结构对象模型组织管理了服务器的调度接口,使得程序设计模型更易于被理解。

在 MFC Automation 服务器中实现 Automation 分层结构困难吗? 如果您知道具体方法的话,一点儿也不困难。其中秘密有两个:首先,为每个要实现的子对象在应用程序中添加一个 Automatable 类。对每个 Automatable 类,再添加 Automation 方法和属性。其次,将子对象连结到它们的父亲上实现分层结构。通过给父对象添加类型 LPDISPATCH 的获得/设置属性,并返回子对象的 IDispatch 接口指针来实现获得函数,这样就可以把一个对象作为另一个对象的子对象了。通过调用子对象从 CCmdTarget 继承来的 GetIDispatch 函数,您可以获得子对象的 IDispatch 指针。

添加 Automatable 类也很容易。只要单击 ClassWizard 的 Add Class 按钮,选择 New,输入类名,选择 CCmdTarget 作为基类,然后选中对话框底部的 Automation 选项。(参见图 20-12。)为使类可以在外部实现(也就是提供自己的 ProgID,以便可以被 Automation 客户创建),可选中 Createable By Type ID 并在其右边的编辑框内输入相应的 ProgID。

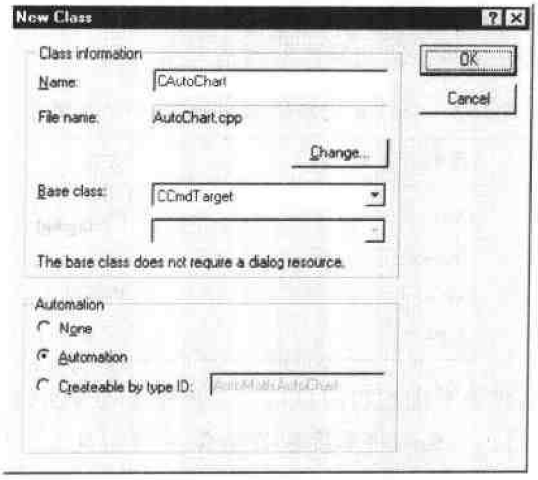


图 20-12 添加一个 Automatable 类

20.2.7 更复杂的 Automation 服务器

图 20-13 所示的 AutoPie 应用程序是一个 MFC Automation 服务器，采用了图 20-14 所示的两级对象模型。AutoPie 绘制的饼图说明了季度税收值。税收值是通过索引属性 Revenue 公开提供的，该属性属于 Chart 对象。称其为"索引属性"的原因是要访问它必须提供 1 到 4 中的一个数来指定季度（第 1 季度、第 2 季度等等。）在内部，Revenue 是被作为获得/设置 Automation属性实现的，在其参数列表中只有一个参数。

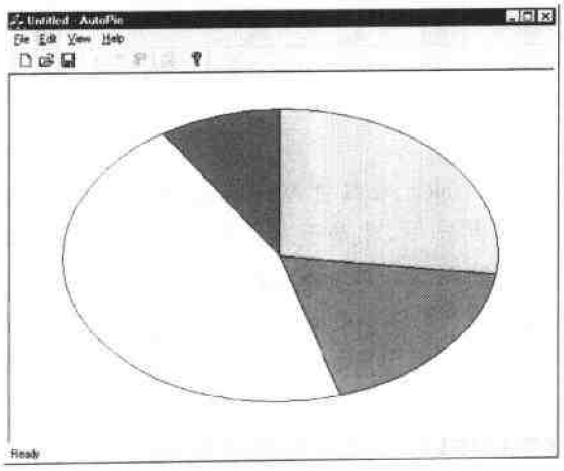


图 20-13 AutoPie 窗口

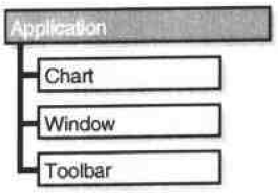


图 20-14 AutoPie 的对象模型

Revenue 只是 AutoPie 提供的几个属性之一。表 20-3 中给出了 AutoPie 支持的所有 Automation方法和属性,以及这些方法和属性所属的对象。

表 20-3 AutoPie 支持的 Automation 方法和属性

对象	属性	方法
Application	N/A	Quit ()
Chart	Revenue (quarter)	Save (pathname)
Window	Visible	Refresh ()
Toolbar	Visible	N/A

顶层 Application 对象代表了应用程序自己。它的唯一一个方法 Quit 执行结束应用程序的任务。Chart 对象代表饼图。Save 将季度税收值保存到磁盘上。Window 代表了应用程序的窗口,其 Visible 属性用来隐藏或显示窗口,Refresh 强迫窗口(以及其中显示的饼图)重绘。最后,Toolbar 对象代表窗口的工具栏,通过将 Visible 设置为 0(关)或非 0(开)值可以使工具栏在开关状态间切换。

可以使用下列 VBScript 小应用程序测试 AutoPie :

```
Set Pie = CreateObject ("AutoPie.Application")
Pie.Chart.Revenue (1) = 420
Pie.Chart.Revenue (2) = 234
Pie.Chart.Revenue (3) = 380
Pie.Chart.Revenue (4) = 640
Pie.Window.Visible = 1
MsgBox ("Click OK to double third-quarter revenues")
Pie.Chart.Revenue (3) = Pie.Chart.Revenue (3) * 2
Pie.Window.Refresh
Pie.Chart.Save ("C:\Chart.pie")
MsgBox ("Test completed")
```

在执行时,脚本将 AutoPie 的 ProgID 传递给 CreateObject 来启动 Automation 服务器。然后赋给它季度税收值并使 AutoPie 窗口可见。(在默认状态下,非基于对话框的 MFC Automation 服务器在被 Automation 客户启动时并不显示它们的窗口。)接下来,脚本显示一个消息框。在消息框释放后,第 3 季度的税收值被读取,乘 2 后再写入 Automation 服务器。以后 Refresh 会被调用来更新饼图。最后 Chart 对象的 Save 方法被调用来将饼图保存到文件中,消息框出现宣布测试结束。

在图 20-15 中给出了 AutoPie 源程序中相关的部分。顶层 Application 对象是由应用程序的文档类代表的。在我使用 AppWizard 创建项目时,输入了"AutoPie. Application"作为 ProgID。因为 AppWizard 自动生成了文档类,所以对于顶层的 Application 对象,CAutoPieDoc 便成为某种代理了。子对象分别由 CAutoChart、CAutoWindow 和 CAutoToolbar 代表,我是使用

ClassWizard 从 CCmdTarget 派生来的。每个都是 Automatable 类(参见图 20-12。)在生成这些类之后,我使用 ClassWizard 给它们添加了 Automation 方法和属性。

AutoPie.h

```
// AutoPie.h : main header file for the AUTOPIE application
//
#if !defined(
    AFX_AUTOPIE_H__3B5BA30B_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_AUTOPIE_H__3B5BA30B_3B72_11D2_AC82_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h" // main symbols

/////////////////////////////////////////////////////////////////////////////
// CAutoPieApp:
// See AutoPie.cpp for the implementation of this class
//

class CAutoPieApp : public CWinApp
{
public:
    CAutoPieApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAutoPieApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    COleTemplateServer m_server;
        // Server object for document creation
    //{{AFX_MSG(CAutoPieApp)
    afx_msg void OnAppAbout();
        // NOTE - the ClassWizard will add and remove member functions here.
        // DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

```
/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(AFX_AUTOPIE_H__3B5EA30B_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

AutoPie.cpp

```
// AutoPie.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "AutoPie.h"

#include "MainFrm.h"
#include "AutoPieDoc.h"
#include "AutoPieView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAutoPieApp

BEGIN_MESSAGE_MAP(CAutoPieApp, CWinApp)
    //{{AFX_MSG_MAP(CAutoPieApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        // DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoPieApp construction

CAutoPieApp::CAutoPieApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
```

```
};

/////////////////////////////////////////////////////////////////////////////
// The one and only CAutoPieApp object

CAutoPieApp theApp;

// This identifier was generated to be statistically unique for your app.
// You may change it if you prefer to choose a specific identifier.

// {3B5BA306-3B72-11D2-AC82-006008A8274D}
static const CLSID clsid =
{ 0x3b5ba306, 0x3b72, 0x11d2,
    { 0xac, 0x82, 0x0, 0x60, 0x8, 0xa8, 0x27, 0x4d } };

/////////////////////////////////////////////////////////////////////////////
// CAutoPieApp initialization

BOOL CAutoPieApp::InitInstance()
{
    // Initialize OLE libraries
    if (!AfxOleInit())
    {
        AfxMessageBox(IDP_OLE_INIT_FAILED);
        return FALSE;
    }

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.

    // Change the registry key under which our settings are stored.
    // TODO: You should modify this string to be something appropriate
    // such as the name of your company or organization.
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings(); // Load standard INI file options
                              // (including MRU)

    // Register the application's document templates. Document templates
    // serve as the connection between documents, frame windows and views.

    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CAutoPieDoc),
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CAutoPieView));
    AddDocTemplate(pDocTemplate);
```

```
    // Connect the COleTemplateServer to the document template.
    // The COleTemplateServer creates new documents on behalf
    // of requesting OLE containers by using information
    // specified in the document template.
    m_server.ConnectTemplate(clsid, pDocTemplate, TRUE);
        // Note: SDI applications register server objects only if /Embedding
        // or /Automation is present on the command line.

    // Enable DDE Execute open
    EnableShellOpen();
    RegisterShellFileTypes(TRUE);

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Check to see if launched as OLE server
    if (cmdInfo.m_bRunEmbedded cmdInfo.m_bRunAutomated)
    {
        // Register all OLE server (factories) as running. This enables
        // the OLE libraries to create objects from other applications.
        COleTemplateServer::RegisterAll();

        // Application was run with /Embedding or /Automation.
        // Don't show themain window in this case.
        return TRUE;
    }

    // When a server application is launched stand-alone, it is a good idea
    // to update the system registry in case it has been damaged.
    m_server.UpdateRegistry(OAT_DISPATCH_OBJECT);
    COleObjectFactory::UpdateRegistryAll();

    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so show and update it.
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();

    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
```

```
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CAutoPieApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
```

```
}
/////////////////////////////////////////////////////////////////////////////
// CAutoPieApp message handlers
```

AutoPieDoc.h

```
// AutoPieDoc.h : interface of the CAutoPieDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_AUTOPIEDOC_H__3B5BA312_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_AUTOPIEDOC_H__3B5BA312_3B72_11D2_AC82_006008A8274D__INCLUDED_

#include "AutoChart.h" // Added by ClassView
#include "AutoWindow.h" // Added by ClassView
#include "AutoToolbar.h" // Added by ClassView

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CAutoPieDoc : public CDocument
{
protected: // create from serialization only
    CAutoPieDoc();
    DECLARE_DYNCREATE(CAutoPieDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAutoPieDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    void SetRevenue(int nQuarter, int nNewValue);

    int GetRevenue(int nQuarter);
    virtual ~CAutoPieDoc();
#ifdef _DEBUG
```

```
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    CAutoToolbar m_autoToolbar;
    CAutoWindow m_autoWindow;
    CAutoChart m_autoChart;
    int m_nRevenues[4];
    //{{AFX_MSG(CAutoPieDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CAutoPieDoc)
    afx_msg LPDISPATCH GetChart();
    afx_msg void SetChart(LPDISPATCH newValue);
    afx_msg LPDISPATCH GetWindow();
    afx_msg void SetWindow(LPDISPATCH newValue);
    afx_msg LPDISPATCH GetToolbar();
    afx_msg void SetToolbar(LPDISPATCH newValue);
    afx_msg void Quit();
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//    AFX_AUTOPIEDOC_H__3B5BA312_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

AutoPieDoc.cpp

```
// AutoPieDoc.cpp : implementation of the CAutoPieDoc class
//
```

```
#include "stdafx.h"
#include "AutoPie.h"
#include "AutoPieDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAutoPieDoc

IMPLEMENT_DYNCREATE(CAutoPieDoc, CDocument)

BEGIN_MESSAGE_MAP(CAutoPieDoc, CDocument)
    //{{AFX_MSG_MAP(CAutoPieDoc)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CAutoPieDoc, CDocument)
    //{{AFX_DISPATCH_MAP(CAutoPieDoc)
    DISP_PROPERTY_EX(CAutoPieDoc, "Chart", GetChart, SetChart, VT_DISPATCH)
    DISP_PROPERTY_EX(CAutoPieDoc, "Window", GetWindow,
        SetWindow, VT_DISPATCH)
    DISP_PROPERTY_EX(CAutoPieDoc, "Toolbar", GetToolbar,
        SetToolbar, VT_DISPATCH)
    DISP_FUNCTION(CAutoPieDoc, "Quit", Quit, VT_EMPTY, VTS_NONE)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IAutoPie to support typesafe binding
// from VBA.  This IID must match the GUID that is attached to the
// dispinterface in the .ODL file.

// {3B5BA308-3B72-11D2-AC82-006008A8274D}
static const IID IID_IAutoPie =
{ 0x3b5ba308, 0x3b72, 0x11d2,
    { 0xac, 0x82, 0x0, 0x60, 0x8, 0xa8, 0x27, 0x4d } };

BEGIN_INTERFACE_MAP(CAutoPieDoc, CDocument)
    INTERFACE_PART(CAutoPieDoc, IID_IAutoPie, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoPieDoc construction/destruction
```

```
CAutoPieDoc::CAutoPieDoc()
{
    EnableAutomation();

    AfxOleLockApp();
}

CAutoPieDoc::~CAutoPieDoc()
{
    AfxOleUnlockApp();
}

BOOL CAutoPieDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    m_nRevenues[0] = 1;
    m_nRevenues[1] = 1;
    m_nRevenues[2] = 1;
    m_nRevenues[3] = 1;
    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAutoPieDoc serialization

void CAutoPieDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        for (int i = 0; i < 4; i++)
            ar << m_nRevenues[i];
    }
    else
    {
        for (int i = 0; i < 4; i++)
            ar >> m_nRevenues[i];
    }
}

/////////////////////////////////////////////////////////////////////////////
// CAutoPieDoc diagnostics

#ifdef _DEBUG
void CAutoPieDoc::AssertValid() const
{
    CDocument::AssertValid();
}
```

```
void CAutoPieDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CAutoPieDoc commands

int CAutoPieDoc::GetRevenue(int nQuarter)
{
    ASSERT(nQuarter >= 0 && nQuarter <= 3);
    return m_nRevenues[nQuarter];
}

void CAutoPieDoc::SetRevenue(int nQuarter, int nNewValue)
{
    ASSERT(nQuarter >= 0 && nQuarter <= 3);
    m_nRevenues[nQuarter] = nNewValue;
}

void CAutoPieDoc::Quit()
{
    AfxGetMainWnd()->PostMessage(WM_CLOSE, 0, 0);
}

LPDISPATCH CAutoPieDoc::GetChart()
{
    return m_autoChart.GetIDispatch(TRUE);
}

void CAutoPieDoc::SetChart(LPDISPATCH newValue)
{
    SetNotSupported();
}

LPDISPATCH CAutoPieDoc::GetWindow()
{
    return m_autoWindow.GetIDispatch(TRUE);
}

void CAutoPieDoc::SetWindow(LPDISPATCH newValue)
{
    SetNotSupported();
}

LPDISPATCH CAutoPieDoc::GetToolbar()
{
    return m_autoToolbar.GetIDispatch(TRUE);
```

```
}

void CAutoPieDoc::SetTcolbar(LPDISPATCH newValue)
{
    SetNotSupported();
}
```

AutoChart.h

```
#if !defined(
    AFX_AUTOCHART_H__3B5BA31E_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_AUTOCHART_H__3B5BA31E_3B72_11D2_AC82_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000

// AutoChart.h : header file
//

#define ID_ERROR_OUTOFRANGE 100

/////////////////////////////////////////////////////////////////////////////
// CAutoChart command target

class CAutoChart : public CCmdTarget
{
    DECLARE_DYNCREATE(CAutoChart)

    CAutoChart();    // protected constructor used by dynamic creation

// Attributes
public:
    virtual ~CAutoChart();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAutoChart)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CAutoChart)
```

```
        // NOTE - the ClassWizard will add and remove member functions here.
    //!!AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CAutoChart)
    afx_msg BOOL Save(LPCTSTR pszPath);
    afx_msg long GetRevenue(short nQuarter);
    afx_msg void SetRevenue(short nQuarter, long nNewValue);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_AUTOCHART_H__3B5BA31E_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

AutoChart.cpp

```
// AutoChart.cpp : implementation file
//

#include "stdafx.h"
#include "AutoPie.h"
#include "AutoChart.h"
#include "AutoPieDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAutoChart

IMPLEMENT_DYNCREATE(CAutoChart, CCmdTarget)

CAutoChart::CAutoChart()
{
    EnableAutomation();
```

```
:

CAutoChart::~CAutoChart()
{
}

void CAutoChart::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called. The base class will automatically
    // deletes the object. Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}

BEGIN_MESSAGE_MAP(CAutoChart, CCmdTarget)
    //{{AFX_MSG_MAP(CAutoChart)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CAutoChart, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CAutoChart)
    DISP_FUNCTION(CAutoChart, "Save", Save, VT_BOOL, VTS_BSTR)
    DISP_PROPERTY_PARAM(CAutoChart, "Revenue", GetRevenue,
        SetRevenue, VT_I4, VTS_I2)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IAutoChart to support typesafe binding
// from VBA. This IID must match the GUID that is attached to the
// dispinterface in the .ODL file.

// {3B5BA31D-3B72-11D2-AC82-006008A8274D}
static const IID IID_IAutoChart =
{ 0x3b5ba31d, 0x3b72, 0x11d2,
    { 0xac, 0x82, 0x0, 0x60, 0x8, 0xa8, 0x27, 0x4d } };

BEGIN_INTERFACE_MAP(CAutoChart, CCmdTarget)
    INTERFACE_PART(CAutoChart, IID_IAutoChart, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoChart message handlers

BOOL CAutoChart::Save(LPCTSTR pszPath)
{
    CFrameWnd* pFrame = (CFrameWnd*)AfxGetMainWnd();
```

```
    CAutoPieDoc * pDoc = (CAutoPieDoc * ) pFrame -> GetActiveDocument ();
    return pDoc -> OnSaveDocument (pszPath);
}

long CAutoChart::GetRevenue(short nQuarter)
{
    long lResult = -1;

    if (nQuarter >= 1 && nQuarter <= 4)
        CFrameWnd * pFrame = (CFrameWnd * ) AfxGetMainWnd ();
        CAutoPieDoc * pDoc = (CAutoPieDoc * ) pFrame -> GetActiveDocument ();
        lResult = (long) pDoc -> GetRevenue (nQuarter - 1);
    }
    else {
        //
        // If the quarter number is out of range, fail the call
        // and let the caller know precisely why it failed.
        //
        AfxThrowOleDispatchException (ID_ERROR_OUTOFRANGE,
            _T ("Invalid parameter specified when reading Revenue"));
    }
    return lResult;
}

void CAutoChart::SetRevenue(short nQuarter, long nNewValue)
{
    if (nQuarter >= 1 && nQuarter <= 4) {
        CFrameWnd * pFrame = (CFrameWnd * ) AfxGetMainWnd ();
        CAutoPieDoc * pDoc = (CAutoPieDoc * ) pFrame -> GetActiveDocument ();
        pDoc -> SetRevenue (nQuarter - 1, nNewValue);
    }
    else {
        //
        // If the quarter number is out of range, fail the call
        // and let the caller know precisely why it failed.
        //
        AfxThrowOleDispatchException (ID_ERROR_OUTOFRANGE,
            _T ("Invalid parameter specified when setting Revenue"));
    }
}
```

AutoWindow.h

```
#if !defined(
    AFX_AUTOWINDOW_H__335BA321_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_AUTOWINDOW_H__3B5BA321_3B72_11D2_AC82_006008A8274D__INCLUDED_
```

```
#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000

// AutoWindow.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CAutoWindow command target

class CAutoWindow : public CCmdTarget
{
    DECLARE_DYNCREATE(CAutoWindow)

    CAutoWindow();    // protected constructor used by dynamic creation

// Attributes
public:
    virtual ~CAutoWindow();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAutoWindow)
    public:

virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CAutoWindow)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CAutoWindow)
    afx_msg BOOL GetVisible();
    afx_msg void SetVisible(BOOL bNewValue);
    afx_msg void Refresh();
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_AUTOWINDOW_H__3B5BA321_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

AutoWindow.cpp

```
// AutoWindow.cpp : implementation file
//

#include "stdafx.h"
#include "AutoPie.h"
#include "AutoWindow.h"
#include "AutoPieDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAutoWindow

IMPLEMENT_DYNCREATE(CAutoWindow, CCmdTarget)

CAutoWindow::CAutoWindow()
{
    EnableAutomation();
}

CAutoWindow::~CAutoWindow()
{
}

void CAutoWindow::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called. The base class will automatically
    // deletes the object. Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}

BEGIN_MESSAGE_MAP(CAutoWindow, CCmdTarget)
```

```
    //{{AFX_MSG_MAP(CAutoWindow)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}} AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CAutoWindow, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CAutoWindow)
    DISP_PROPERTY_EX(CAutoWindow, "Visible", GetVisible,
        SetVisible, VT_BOOL)
    DISP_FUNCTION(CAutoWindow, "Refresh", Refresh, VT_EMPTY, VTS_NONE)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IAutoWindow to support typesafe binding
// from VBA. This IID must match the GUID that is attached to the
// dispinterface in the .ODL file.

// {3B5BA320-3B72-11D2-AC82-006008A8274D}
static const IID IID_IAutoWindow =
{ 0x3b5ba320, 0x3b72, 0x11d2,
    { 0xac, 0x82, 0x0, 0x60, 0x8, 0xa8, 0x27, 0x4d } };
BEGIN_INTERFACE_MAP(CAutoWindow, CCmdTarget)
    INTERFACE_PART(CAutoWindow, IID_IAutoWindow, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoWindow message handlers

void CAutoWindow::Refresh()
{
    CFrameWnd * pFrame = (CFrameWnd * ) AfxGetMainWnd ();
    CAutoPieDoc * pDoc = (CAutoPieDoc * ) pFrame -> GetActiveDocument ();
    pDoc -> UpdateAllViews (NULL);


BOOL CAutoWindow::GetVisible()

    return AfxGetMainWnd () -> IsWindowVisible ();
}

void CAutoWindow::SetVisible(BOOL bNewValue)
{
    AfxGetMainWnd () -> ShowWindow (bNewValue ? SW_SHOW : SW_HIDE);
}
```

AutoToolbar.h

```
#if !defined(
```

```
    AFX_AUTOTOOLBAR_H__3B5BA324_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_AUTOTOOLBAR_H__3B5BA324_3B72_11D2_AC82_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000

// AutoToolbar.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CAutoToolbar command target

class CAutoToolbar : public CCmdTarget
{
    DECLARE_DYNCREATE(CAutoToolbar)

    CAutoToolbar();    // protected constructor used by dynamic creation

// Attributes
public:
    virtual ~CAutoToolbar();

// Operations
public:

//Overrides
// ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAutoToolbar)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CAutoToolbar)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CAutoToolbar)
    afx_msg BOOL GetVisible();
    afx_msg void SetVisible(BOOL bNewValue);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};
```

```
/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_AUTOTOOLBAR_H__3B5BA324_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

AutoToolbar.cpp

```
// AutoToolbar.cpp : implementation file
//

#include "stdafx.h"
#include "AutoDie.h"
#include "AutoToolbar.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAutoToolbar

IMPLEMENT_DYNCREATE(CAutoToolbar, CCmdTarget)

CAutoToolbar::CAutoToolbar()
{
    EnableAutomation();
}

CAutoToolbar::~CAutoToolbar()
{
}

void CAutoToolbar::OnFinalRelease()
{
    // When the last reference for an automation object is released
    // OnFinalRelease is called. The base class will automatically
    // deletes the object. Add additional cleanup required for your
    // object before calling the base class.

    CCmdTarget::OnFinalRelease();
}
```

```
BEGIN_MESSAGE_MAP(CAutoToolbar, CCmdTarget)
    //{{AFX_MSG_MAP(CAutoToolbar)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CAutoToolbar, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CAutoToolbar)
    DISP_PROPERTY_EX(CAutoToolbar, "Visible", GetVisible,
        SetVisible, VT_BOOL)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_IAutoToolbar to support typesafe binding
// from VBA. This IID must match the GUID that is attached to the
// dispinterface in the .ODL file.

// {3B5BA323-3B72-11D2-AC82-006008A8274D}
static const IID IID_IAutoToolbar =
{ 0x3b5ba323, 0x3b72, 0x11d2,
    { 0xac, 0x82, 0x0, 0x60, 0x8, 0xa8, 0x27, 0x4d } };

BEGIN_INTERFACE_MAP(CAutoToolbar, CCmdTarget)
    INTERFACE_PART(CAutoToolbar, IID_IAutoToolbar, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoToolbar message handlers


BOOL CAutoToolbar::GetVisible()
{
    CMainFrame* pFrame = (CMainFrame*) AfxGetMainWnd();
    return (pFrame->m_wndToolBar.GetStyle() & WS_VISIBLE) ?
        TRUE : FALSE;
}

void CAutoToolbar::SetVisible(BOOL bNewValue)
{
    CMainFrame* pFrame = (CMainFrame*) AfxGetMainWnd();
    pFrame->ShowControlBar(&pFrame->m_wndToolBar, bNewValue, FALSE);
}
```

AutoPieView.h

```
// AutoPieView.h : interface of the CAutoPieView class
//
```

```
////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_AUTOPIEVIEW_H__3B5BA314_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_AUTOPIEVIEW_H__3B5BA314_3B72_11D2_AC82_006008A8274D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define PI 3.1415926

class CAutoPieView : public CView
{
protected: // create from serialization only
    CAutoPieView();
    DECLARE_DYNCREATE(CAutoPieView)

// Attributes
public:
    CAutoPieDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAutoPieView)

    public:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CAutoPieView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CAutoPieView)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG // debug version in AutoPieView.cpp
inline CAutoPieDoc* CAutoPieView::GetDocument()
   { return (CAutoPieDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////
//{ AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//    AFX_AUTOPIEVIEW_H__3B5BA314_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

AutoPieView.cpp

```
// AUTOPIEVIEW.CPP : IMPLEMENTATION OF THE CAUTOPIEVIEW CLASS
//

#include "stdafx.h"
#include "AutoPie.h"
#include "AutoPieDoc.h"
#include "AutoPieView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAutoPieView

IMPLEMENT_DYNCREATE(CAutoPieView, CView)

BEGIN_MESSAGE_MAP(CAutoPieView, CView)
    //{{AFX_MSG_MAP(CAutoPieView)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAutoPieView construction/destruction
```

```
CAutoPieView::CAutoPieView()
{
    // TODO: add construction code here

}

CAutoPieView::~CAutoPieView()
{
}

BOOL CAutoPieView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CAutoPieView drawing

void CAutoPieView::OnDraw(CDC * pDC)
{
    CAutoPieDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    CRect rect;
    GetClientRect (&rect);
    //
    // Initialize the mapping mode.
    //
    pDC->SetMapMode (MM_ANISOTROPIC);
    pDC->SetWindowExt (500, 500);
    pDC->SetWindowOrg (-250, -250);
    pDC->SetViewportExt (rect.Width (), rect.Height ());

    //
    // Create a set of brushes.
    //
    CBrush brFillColor[4];
    brFillColor[0].CreateSolidBrush (RGB (255, 0, 0)); // Red
    brFillColor[1].CreateSolidBrush (RGB (255, 255, 0)); // Yellow
    brFillColor[2].CreateSolidBrush (RGB (255, 0, 255)); // Magenta
    brFillColor[3].CreateSolidBrush (RGB ( 0, 255, 255)); // Cyan

    //
    // Draw the pie chart.
    //
    int nTotal = 0;
```

```
    for (int i = 0; i < 4; i++)
        nTotal += pDoc->GetRevenue(i);

    int x1 = 0;
    int y1 = -1000;
    int nSum = 0;

    for (i = 0; i < 4; i++) {
        int nRevenue = pDoc->GetRevenue(i);
        if (nRevenue != 0) {
            nSum += nRevenue;
            int x2 = (int)(sin((((double)nSum * 2 * PI) /
                (double)nTotal) + PI) * 1000);
            int y2 = (int)(cos((((double)nSum * 2 * PI) /
                (double)nTotal) + PI) * 1000);
            pDC->SelectObject(&brFillColor[i]);
            pDC->Pie(-200, -200, 200, 200, x1, y1, x2, y2);
            x1 = x2;
            y1 = y2;
        }
    }
    pDC->SelectStockObject(WHITE_BRUSH);
}

/////////////////////////////////////////////////////////////////////////////
// CAutoPieView diagnostics

#ifdef _DEBUG
void CAutoPieView::AssertValid() const
{
    CView::AssertValid();
}
void CAutoPieView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CAutoPieDoc* CAutoPieView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CAutoPieDoc)));
    return (CAutoPieDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CAutoPieView message handlers
```

图 20-15 AutoPie 程序

为了使 CAutoWindow、CAutoChart 和 CAutoToolbar 成为 Application 对象的子对象，我给文档类中添加了 CAutoWindow、CAutoChart 和 CAutoToolbar 数据成员 m_autoWindow、m_autoChart 和 m_autoToolbar。然后给文档类添加了 LPDISPATCH 获得/设置属性，名字为 Window、Chart 和 Toolbar，并调用嵌入对象的 GetIDispatch 来实现获得函数。如果客户试图给这些属性写入值，设置函数中的 SetNotSupported 就会提示通知属性为只读的。

```
LPDISPATCH CAutoPieDoc::GetChart()
{
    return m_autoChart.GetIDispatch(TRUE);
}

void CAutoPieDoc::SetChart(LPDISPATCH newValue)
{
    SetNotSupported();
}

LPDISPATCH CAutoPieDoc::GetWindow()
{
    return m_autoWindow.GetIDispatch(TRUE);
}

void CAutoPieDoc::SetWindow(LPDISPATCH newValue)
{
    SetNotSupported();
}

LPDISPATCH CAutoPieDoc::GetToolbar()
{
    return m_autoToolbar.GetIDispatch(TRUE);
}

void CAutoPieDoc::SetToolbar(LPDISPATCH newValue)
{
    SetNotSupported();
}
```

将 TRUE 传递给 GetIDispatch 确保了对 AddRef 的调用要通过从子对象中获取到的 IDispatch 指针。这样就防止了子对象被过早地删除。删除 IDispatch 指针则是客户的责任。幸运的是，VBScript 客户可以自动执行此任务。

AfxThrowOleDispatchException 函数

SetNotSupported 使用了 MFC 的 AfxThrowOleDispatchException 函数来阻止对只读 Automation 属性的写入。有时您也可以亲自调用 AfxThrowOleDispatchException。AutoPie 只是在读取或写入 Chart 对象的 Revenue 属性时，在客户指定了非法的季度编号（1 到 4 范围之外的值）的情况下才调用它。下面给出一段从 AutoChart.cpp 节选出的程序：

```
AfxThrowOleDispatchException (ID_ERROR_OUTOFRANGE,
    _T ("Invalid parameter specified when reading Revenue"));
```

AfxThrowOleDispatchException 使调用失败并给客户提供了一个说明出错的消息。大多数客户,特别是 VBScript 客户,会给用户显示此信息。

20.3 MFC Automation 客户

MFC 极大地简化了 Automation 服务器的编写,但是对编写 Automation 客户怎么样呢? 一个好消息是:只要借助一点 ClassWizard 的帮助,用 MFC 编写 Automation 客户几乎和用 Visual Basic 编写一样简单。

最重要的是名为 COleDispatchDriver 的类,它对正在运行的 Automation 服务器所公开的 IDispatch 指针进行了友好的封装。COleDispatchDriver 的辅助函数 InvokeHelper、SetProperty 和 GetProperty 简化了对方法和属性的访问,但是使用这些函数与 Automation 对象进行交互仅仅比直接调用 IDispatch::Invoke 好一些。COleDispatchDriver 真正的价值在于创建类型安全的类,它的成员函数提供了对 Automation 方法和属性进行访问的简易方法。毕竟,对于 C++ 程序员来说,调用成员函数要比调用 IDispatch::Invoke 更容易些。

要从为特定的 Automation 服务器定做的 COleDispatchDriver 中派生类,可以单击 ClassWizard 的 Add Class 按钮,选择 From A Type Library,并将 ClassWizard 指向服务器类型库。ClassWizard 将阅读类型库并生成一个新类。在这个类中,您会发现调用服务器方法的成员函数以及访问服务器属性的获得/设置函数。例如:如果服务器支持方法 Add 和属性 Pi,那么 ClassWizard 生成的类将包含成员函数 Add 和访问函数 GetPi 和 SetPi。如果封装类名称是 CAutoMath 并且对象的 ProgID 为"Math.Object",则可用以下语句对对象进行实例化和编程:

```
CAutoMath math;
math.CreateDispatch (_T ("Math.Object"));
int sum = math.Add (2, 2);
double pi = math.GetPi ();
```

CreateDispatch 使用 ::CoCreateInstance 来创建 Automation 对象。它在成员变量 m_lpDispatch 中保存了对象的 IDispatch 指针。通过 CAutoMath 成员函数执行的方法调用和属性访问被 InvokeHelper 和其他 COleDispatchDriver 函数变换为了对对象的 IDispatch 调用。

20.3.1 PieClient 应用程序

让我们用一个 MFC Automation 客户程序 PieClient 来结束本章内容。图 20-16 显示了它的外观,而在图 20-17 中给出了它的源程序。这是一个基于对话框的应用程序,其主窗口具有编辑控件用来输入和编辑季度税收值。AutoPie 以图形方式表示出在控件中输入的值。

PieClient 通过 Automation 来驱动 AutoPie。

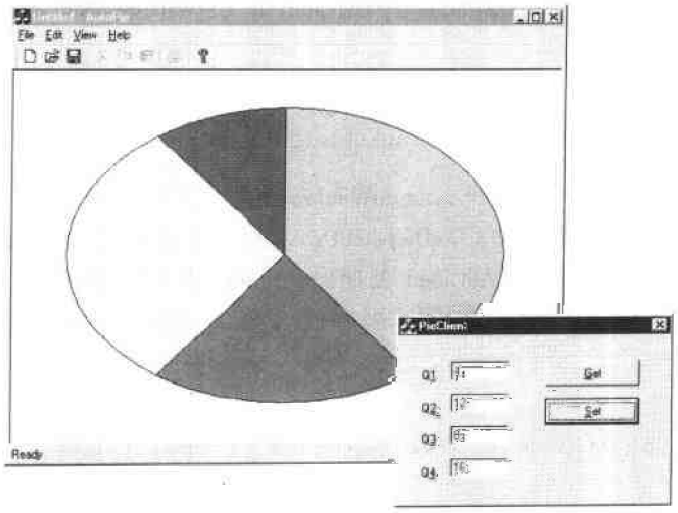


图 20-16 PieClient 作为 AutoPie 的 Automation 客户

当启动时，PieClient 调用 CAutoPie 对象名字为 m_autoPie 中的 CreateDispatch 来启动 Automation 服务器：

```
BOOL bSuccess = m_autoPie.CreateDispatch (_T ("AutoPie.Application"));
```

当 Set 按钮被按下时，PieClient 从编辑控件中得到税收值并通过将它们写入 Chart 对象的 Revenue 属性而将其传送给服务器：

```
m_autoChart.SetRevenue (1, GetDlgItemInt (IDC_Q1));
m_autoChart.SetRevenue (2, GetDlgItemInt (IDC_Q2));
m_autoChart.SetRevenue (3, GetDlgItemInt (IDC_Q3));
m_autoChart.SetRevenue (4, GetDlgItemInt (IDC_Q4));
```

然后调用 Window 对象的 Refresh 方法来重绘饼图：

```
m_autoWindow.Refresh ();
```

相反，如果 Get 按钮被单击时，PieClient 会从 Automation 对象中读取属性值并把它们显示在编辑控件中。

m_autoChart 和 m_autoWindow 是 CAutoChart 和 CAutoWindow 的实例。这些类和其他类，也就是 CAutoPie 和 CAutoToolbar，是 COleDispatchDriver 派生类，是由 ClassWizard 从 AutoPie 的类型库中创建的。CAutoPie 代表了服务器顶层 Application 对象。剩下的类分别代表了 Chart、Window 和 Toolbar 子对象。m_autoPie 是由 CreateDispatch 初始化的，而 m_autoChart 和

m_autoWindow 必须单独初始化,因为它们相应的子对象是在服务器启动以后自动创建的。这些初始化的执行是通过把由 CAutoPie 的 GetChart 和 GetWindow 函数返回的 IDispatch 指针传递给 AttachDispatch 完成的:

```
m_autoChart.AttachDispatch(m_autoPie.GetChart());
m_autoWindow.AttachDispatch(m_autoPie.GetWindow());
```

由于 m_autoPie、m_autoChart 和 m_autoWindow 是嵌入型数据成员,所以当对话框被销毁时它们也会被自动销毁。并且在 COleDispatchDriver 对象被销毁时,它封装的 IDispatch 指针会被类析构函数释放。这就是 PieClient 关闭时 AutoPie 也关闭的原因。当最后一个指向 MFC Automation 服务器的 dispinterface 的指针释放后,服务器就会顺从地关闭自身。

PieClient.h

```
// PieClient.h : main header file for the PIECLIENT application
//

#if !defined(
    AFX_PIECLIENT_H__3B5BA32A_3B72_11D2_AC82_006008A8274D__INCLUDED_)
#define AFX_PIECLIENT_H__3B5BA32A_3B72_11D2_AC82_006008A8274D__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000
#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"    // main symbols

/////////////////////////////////////////////////////////////////////////////
// CPieClientApp:
// See PieClient.cpp for the implementation of this class
//

class CPieClientApp : public CWinApp
{
public:
    CPieClientApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPieClientApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
```

```
    //{{AFX_MSG(CPieClientApp)
        // NOTE - the ClassWizard will add and remove member functions here.
        //      DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_PIECLIENT_H__3B5BA32A_3B72_11D2_AC82_006008A82F4D__INCLUDED_)
```

PieClient.cpp

```
// PieClient.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "PieClient.h"
#include "PieClientDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPieClientApp

BEGIN_MESSAGE_MAP(CPieClientApp, CWinApp)
    //{{AFX_MSG_MAP(CPieClientApp)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPieClientApp construction

CPieClientApp::CPieClientApp()
{
    // TODO: add construction code here,
```

```
    // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CPieClientApp object

CPieClientApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CPieClientApp initialization

BOOL CPieClientApp::InitInstance()
{
    if (!AfxOleInit ()) {
        AfxMessageBox (_T ("AfxOleInit failed"));
        return FALSE;
    }
    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.

    CPieClientDlg dlg;
    m_pMainWnd = &dlg;
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with Cancel
    }

    // Since the dialog has been closed, return FALSE so that we exit the
    // application, rather than start the application's message pump.
    return FALSE;
}
```

PieClientDlg.h

```
// PieClientDlg.h : header file
//

#if !defined(
    AFX_PIECLIENTDLG_H__3B5BA32C_3B72_11D2_AC82_006008A8274D__INCLUDED_)
```

```
#define AFX_PIECLIENTDLG_H__3B5BA32C_3B72_11D2_AC82_006008A8274D__INCLUDED_

#include "autopie.h" // Added by ClassView
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CPieClientDlg dialog

class CPieClientDlg : public CDialog
{
// Construction
public:
    CPieClientDlg(CWnd * pParent = NULL); // standard constructor
// Dialog Data
    //{{AFX_DATA(CPieClientDlg)
    enum { IDD = IDD_PIECLIENT_DIALOG };
    CButton  m_wndSet;
    CButton  m_wndGet;
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPieClientDlg)
    protected:
    virtual void DoDataExchange(CDataExchange * pDX); // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    CAutoWindow m_autoWindow;
    CAutoChart m_autoChart;
    CAutoPie m_autoPie;
    HICON m_hIcon;

    // Generated message map functions
    //{{AFX_MSG(CPieClientDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnGet();
    afx_msg void OnSet();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
```

```
// immediately before the previous line.

#endif
// !defined(
//    AFX_PIECLIENTDLG_H__3B5BA32C_3B72_11D2_AC82_006008A8274D__INCLUDED_ )
```

PieClientDlg.cpp

```
// PieClientDlg.cpp : implementation file
//

#include "stdafx.h"
#include "PieClient.h"
#include "PieClientDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPieClientDlg dialog

CPieClientDlg::CPieClientDlg(CWnd* pParent /* = NULL */)
    : CDialog(CPieClientDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CPieClientDlg)
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent
    // DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CPieClientDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CPieClientDlg)
    DDX_Control(pDX, IDC_SET, m_wndSet);
    DDX_Control(pDX, IDC_GET, m_wndGet);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CPieClientDlg, CDialog)
    //{{AFX_MSG_MAP(CPieClientDlg)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
```

```
    ON_BN_CLICKED(IDC_GET, OnGet)
    ON_BN_CLICKED(IDC_SET, OnSet)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPieClientDlg message handlers

BOOL CPieClientDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    SetIcon(m_hIcon, TRUE);    // Set big icon
    SetIcon(m_hIcon, FALSE);    // Set small icon

    //
    // Start the Automation server.
    //
    BOOL bSuccess = m_autoPie.CreateDispatch(_T("AutoPie.Application"));

    //
    // If CreateDispatch succeeded, initialize the m_autoChart and
    // m_autoWindow data members to represent the Chart and Window
    // subobjects, respectively. Then initialize the controls in
    // the dialog and make the server window visible.
    //
    if (bSuccess)
    {
        m_autoChart.AttachDispatch(m_autoPie.GetChart());
        ASSERT(m_autoChart.m_lpDispatch != NULL);
        m_autoWindow.AttachDispatch(m_autoPie.GetWindow());
        ASSERT(m_autoWindow.m_lpDispatch != NULL);
        OnGet();
        m_autoWindow.SetVisible(TRUE);
    }

    //
    // If CreateDispatch failed, let the user know about it.
    //
    else
    {
        MessageBox(_T("Error launching AutoPie. Run it once to "
        "register it on this system and then try again."),_T("Error"));
        m_wndGet.EnableWindow(FALSE);
        m_wndSet.EnableWindow(FALSE);
    }
    return TRUE; // return TRUE unless you set the focus to a control
}

void CPieClientDlg::OnPaint()
```

```
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle.
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon.
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

HCURSOR CPieClientDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CPieClientDlg::OnGet()
{
    //
    // Retrieve revenue values from the Automation server and display them.
    //
    SetDlgItemInt (IDC_Q1, m_autoChart.GetRevenue (1));
    SetDlgItemInt (IDC_Q2, m_autoChart.GetRevenue (2));
    SetDlgItemInt (IDC_Q3, m_autoChart.GetRevenue (3));
    SetDlgItemInt (IDC_Q4, m_autoChart.GetRevenue (4));
}

void CPieClientDlg::OnSet()
{
    //
    // Retrieve the revenue values displayed in the edit controls
    // and provide them to the Automation server.
    //
    m_autoChart.SetRevenue (1, GetDlgItemInt (IDC_Q1));
    m_autoChart.SetRevenue (2, GetDlgItemInt (IDC_Q2));
```

```
    m_autoChart.SetRevenue (3, GetDlgItemInt (IDC_Q3));
    m_autoChart.SetRevenue (4, GetDlgItemInt (IDC_Q4));

    //
    // Repaint the pie chart.
    //
    m_autoWindow.Refresh ();
}
```

AutoPie.h

```
// Machine generated IDispatch wrapper class(es) created with ClassWizard
/////////////////////////////////////////////////////////////////////////////
// CAutoPie wrapper class

class CAutoPie : public COleDispatchDriver
{
public:
    CAutoPie() {}    // Calls COleDispatchDriver default constructor
    CAutoPie(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    CAutoPie(const CAutoPie& dispatchSrc) :
        COleDispatchDriver(dispatchSrc) {}

// Attributes
public:
    LPDISPATCH GetChart();
    void SetChart(LPDISPATCH);
    LPDISPATCH GetWindow();
    void SetWindow(LPDISPATCH);
    LPDISPATCH GetToolbar();
    void SetToolbar(LPDISPATCH);

// Operations
public:
    void Quit();
};
/////////////////////////////////////////////////////////////////////////////
// CAutoChart wrapper class

class CAutoChart : public COleDispatchDriver
{
public:
    CAutoChart() {}    // Calls COleDispatchDriver default constructor
    CAutoChart(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    CAutoChart(const CAutoChart& dispatchSrc) :
        COleDispatchDriver(dispatchSrc) {}
```

```
// Attributes
public:

// Operations
public:
    BOOL Save(LPCTSTR pszPath);
    long GetRevenue(short nQuarter);
    void SetRevenue(short nQuarter, long nNewValue);
};
/////////////////////////////////////////////////////////////////////////////
// CAutoWindow wrapper class

class CAutoWindow : public COleDispatchDriver
{
public:
    CAutoWindow() {}    // Calls COleDispatchDriver default constructor
    CAutoWindow(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    CAutoWindow(const CAutoWindow& dispatchSrc) :
        COleDispatchDriver(dispatchSrc) {}

// Attributes
public:
    BOOL GetVisible();
    void SetVisible(BOOL);

// Operations
public:
    void Refresh();
};
/////////////////////////////////////////////////////////////////////////////
// CAutoToolbar wrapper class

class CAutoToolbar : public COleDispatchDriver
{
public:
    CAutoToolbar() {}    // Calls COleDispatchDriver default constructor
    CAutoToolbar(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    CAutoToolbar(const CAutoToolbar& dispatchSrc) :
        COleDispatchDriver(dispatchSrc) {}

// Attributes
public:
    BOOL GetVisible();
    void SetVisible(BOOL);

// Operations
public:
};
```