第 8 章　OLE 编程

OLE 自动化的技术使得程序员能控制其他的应用程序或者 DLL 中的对象，不仅能够使用驻留在自己程序里的对象，而且能使用驻留在系统上的其他程序中的对象。特别是，拥护能够访问这些对象的属性和方法，而不是它们的原始数据，而且，分布式 OLE 的到来，已经能使程序员把这些功能扩展到了网络上。

8.1　在 Exc　和 Word 间共享图表

程序员在 Dephi　编程过程中，有时候会要处理复杂的文档、报表之类的任务。如果单纯地用 Dephi 自身所带的功能来实现，其难度是非常大的，而现成已经有了像 Exce，Word 等优秀的处理软件，那么我们所需要做的就是在 Dephi 中控制 Exc　和 Word，使他们按照程序员的意愿实现要求。

下面这个例子详细地介绍了如何在 Dephi 中创建并控制 Exc　和 Word，并实现 Exce 和 Word 之间图表的共享。用户必须在机器上已经安装了 Mirosoft 的 Word 和 Exce 应用程序。

- 打开 Fi　菜单下的 Ne　选项，新建一个工程；
- 在 Form 上放置一个 Button，所有属性均采用默认属性；
- 保存 Unit 为 main.pas，保存工程为 projet1.dpr 。界面如图 8.1 所示：

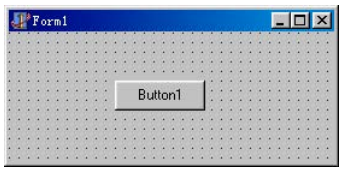

图 8.1 主界面

程序源代码如程序清单 8.1 所示：

（程序清单 8.1）

```
unt mai;

iterfac

us
  Wido, Meage,        SysUti, Cas, Graph, Ctro, Form, Dialgs,
  StdCtrl;
```

```
type
  TForm1 = cas(  TForm)
    Butto1:  TButto;
    produre Butto1Ck(Seder:          TObjet);
    produre   FormDetroy(Seder:    TObjet);
  private
    { Private dearati }
    XLAPP:variant; //Exc   对象
    WordAPP:variant;//Word 对象
  publ
    { Publ dearati }
    produre   HandlData;
    produre
    produre
    produre
    produre
    produre   MaiDoumt;
  ed;

var
  Form1: TForm1;
ct
  { xlttype}
    xlart =-4109;
    xldialgst = -4116;
    xlxc4itlacrot =4;
    xlxc4macrot = 3;
    xlrkst = -4167;
    { xlbaTeate}
    xlWBATCart = -4109;
    xlWBATExc4IntlMacrot=4;
    xlbarexc4macrot =3;
    xlbatwrkst =-4167;
ipltati
us
{$R *.DFM}

produre TForm1.Butto1Ck(Seder:          TObjet);
begi
  //创建 exc   对象
  XLAPP:=CreateOlObjet('   Excapplati');
  //设置为可见
  XLAPP.vibl   :=True;
  //增加一个工作薄
  XLAPP.rkboksadd      [xlbatwrkst];
  //增加一个工作表
  xlapp.rkboks[1].rksts[1].am:='              deph data';
  HandlData;
  CartData;
  CpyData;
  MaiDoumt;
```

```
ed;
//此过程用于向工作表中插入数据
produre TForm1.HandlData ;
var
   st:variant;
   i:Integer;
begi
   //把工作表赋值给临时变量
   st:=    xlapp.rkboks[1].rksts['            deph data'];
   //向工作表中插入数据
   for i:=1 to 10 do
   begi
                          i;
   ed;
ed;
//此过程用于创建图表
produre TForm1.artData ;
var
   ARange:variant;
   sts:variant;
begi
   //插入一个工作表，类型为 xlart
   xlapp.rkboks[1].
   sts:=    xlapp.ts;
   //设定一组数据
   arange:=stste['        deph data'].range['a1:a0'];
   stste['cart1'].                                      arange;
   //设置图表类型
   stste['cart1'].arttype:=xl3dpi;
   stste['cart1'].
   xlapp.rkboks[1].
   stste['cart2'].                                      arange;
   stste['cart2'].                              arange);
   stste['cart2'].
   stste['cart2'].                                      vararrayof([1,2,3,4,5,6,7,8,9,10])
   stste['cart2'].arttype:=xl3dcum;

ed;
produre TForm1.pyData ;
var
   sts:variant;
begi

   sts:=    xlapp.ts;
   //激活工作表
   stste['        deph data'].activate;
   //选择数据
   stste['        deph data'].range['a1:a10'].t;
   //把所选择的数据复制到剪贴板
   stste['        deph data'].  usdrangepy;
   //复制数据到 wrd
```

```
    cpyctord;
    //选择图表
    stste['cart1'].t;
    //复制图表到剪贴板
    xlapp.tipy;
    //把图表复制到 wrd
    cpycarttord;
ed;
produre TForm1.pyCartToWord ;
var
    range:variant;
    i,numpars:iteger;
begi
  //得到文档中有几个段落
    numpars:=wrdapp.doumtste(1).          paragraphunt;
    //创建一个覆盖最后一段的范围
    range:=wrdapp.doumtste(1).range(
    wrdapp.doumtste(1).          paragraphte(      numpars).rangetart,
    wrdapp.doumtste(1).          paragraphte(      numpars).ranged);
    rangetext:='th graph:';
    for i:=1 to 3 do
                                    paragraphadd;
    range:=wrdapp.doumtste(1).range(
    wrdapp.doumtste(1).          paragraphte(numpars+1).      rangetart,
    wrdapp.doumtste(1).          paragraphte(numpars+1).      ranged);
  //插入 o   对象
    rangepastepeal(,,,,     wdpastebjet);

ed;
produre TForm1.pyctord;
var
    range:variant;
    i:iteger;
begi
    //创建 wrd   对象
    wrdapp:=  createbjet('       wrd.applati');
    //设置为可见
    wrdapp.vibl:=true;
    //加入一个单一文档
    wrdapp.doumtsadd;
    //添加一些初始文本
    range:=wrdapp.doumtste(1).range;
    rangetext:='th a cum fropreadst:';
    //可以把 Exc   的数据直接粘贴到文档中，不过我们要对放置单元格的位置有所
    //控制，而要实现该控制，需要文档内的一些空间，我门加入一系列回车
    for i:=1 to 3 do
                                    paragraphadd;
      //选择一个文档中第三段的范围
    range:=wrdapp.doumtste(1).range(
    wrdapp.doumtste(1).          paragraphte(3).      rangetart);
    //  从剪贴板中粘贴到文档
```


rdapp.doumtste(1).

```
    rangepaste;
    //加入回车，控制位置
    for i:=1 to 3 do
                                    paragraphadd;
ed;
produre Tform1.MaiDoumt;
begi
    //保存文档
    wrdapp.doumtste(1).                    tet.do');
    //如果机器上安装了 MAPI 客户程序，可以从 Exc   直接把一个文档发给另外一个用户
    wrdapp.ptidmaiattac:=true;
    wrdapp.doumtste(1).
ed;
produre TForm1.FormDetroy(Seder:      TObjet);
begi
//退出 exc    和 wrd
    if nt   varipty(     xlapp) th
    begi
       xlapp.diplayalrts:=fal;
       xlapp.quit;
    ed;
    if nt   varipty(     wrdapp) th
    begi
                                          wddotsaveange);
       wrdapp.quit;
    ed;
ed;

ed.
```

在这个程序中，只涉及到了 Exc　和 Word 的一些常规操作。如果读者要进行一些很复杂的操作，则必须要了解微软对 Exce 和 Word 所提供的接口和属性和方法等，而根据笔者的经验，读者可以在 Exce 或者是 Word 的 VBA 环境中得到 Exc　和 Word 本身对那些方法和属性的调用，读者可以从中得到借鉴，可以以图 8.2 所示方式启动 VBA（以 Word 为例）。

在用 Dephi 控制 Exce 和 Word 的程序中，一个很大的不便之处就是调试。只能在程序运行过程中，我们才能知道我们所调用的方法和属性以及方法的参数设置是否正确，所以要开发一个很复杂的程序有一定的难度，不过现在资源很多，有很多例子可以参考。

图 8.2　调用 VBA

8.2 OLE 文档

OLE 文档允许不同文档在一起协同工作，并共享数据。它也有客户和服务器之分，能够提供 OLE 文档服务的应用程序即为服务器，能够容纳来自于其他应用程序数据的即为客户。

许多应用程序支持 OLE 2.0，微软的 Offi 程序全部支持 OLE 2.0。不过 OLE 是一种极其消耗内存的链接方式，如果内存不足，那么 OLE 文档和 OLE 服务器的链接就显得有缺陷了，有时不能使用在位方式打开，有时导致保护性错误，终止了服务器运行。

客户程序访问 OLE 文档有两种：链接和嵌入。如果是链接方式，OLE 文档仍然存储在它的源文档中，并且随着源文档的改变而自动刷新，容器内的 OLE 文档仅仅保存了源文档的位置；如果是嵌入方式，容器内的 OLE 文档便是源文档的内容，嵌入之后，源文档发生变化，数据不能自动刷新。

OLE 容器还可以链接剪贴板上不同类型的对象，也可以嵌入其他应用程序才可以处理的文档。使用 TOLEContair 控件可以完全地实现对象的嵌入和链接。

OLE 容器的编程，需要 TOLEContair 控件的大量方法。我们通过一个例子来介绍 TOLEContair 的简单编程。

- 打开 Fi 菜单中的 Ne ，翻到 Projets 页，选择 MDI Applation ，如图 8.3 所示。
- MDI 主窗体的界面设计如图 8.4 所示。

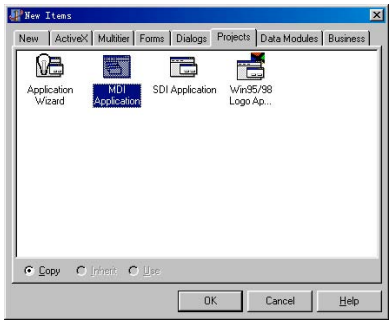

图 8.3 创建 MDI 应用程序

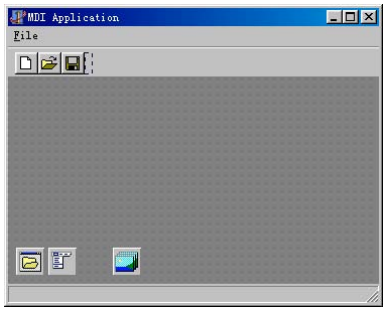

图 8.4 MDI 主窗口界面

主窗体菜单设置如表 8.1 所示。

表 8.1 菜单项设计

Capti	Nam	事件
&Ne		MnuNek
&Ope		MnuOpek
&Save		MnuSavek
Save &A		MnuSavek
&C		MnuCk
&C		MnuCalk

&Exit	muExit	MnuExitCk

♦　子窗体界面设计如图 8.5 所示。

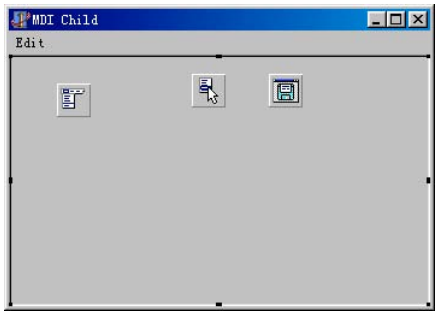

图 8.5　子窗体界面

在窗体上放置一主菜单，一弹出式菜单，一个 SaveDialog，再放置一个 TOLEContair　，弹出式菜单的 Nam　属性设置为 ppmMenu，OLECntair　　的 algn　属性设置 al　　。主菜单的设置如表 8.2 所示。

表 8.2　　菜单项设计

Capti	Nam	事件
&Cut	muCut	MnuCutCk
&Cpy	muCpy	MnuCpyCk
&Paste		MnuPastek
Paste &Speal		MnuPasteSpealk
C&lar	muCar	MnuCarCk
&Objet properti		MnuObjetpropCk
&Inrt Objet	muInrtObjet	MnuInrtObjetCk

♦　按 Dephi 默认就保存主窗体、子窗体单元文件以及项目文件。

主窗体的源代码如程序清单 8.2 所示。

（程序清单 8.2）

```
unt Mai;

iterfac

us Wido,        SysUti, Cas, Graph, Form, Ctro, Meus,
   StdCtrl, Dialgs, Butto, Meage,        ExtCtrl,  Ctrl,       StdAtn,
   AtnLit,   ToWi,     ImgLit;

type
   TMaiForm =   cas(   TForm)
      MaiMeu1:    TMaiMeu;
      muFi:     TMeuIte;
```


nPasteSpeal

```
    muNe:     TMeuIte;
    muOpe:    TMeuIte;
    muC:      TMeuIte;
    N1: TMeuIte;
    muExit:   TMeuIte;
    OpeDialg1:    TOpeDialg;
    muSave:   TMeuIte;
    muSave:     TMeuIte;
    StatusBar: TStatusBar;
    ToBar2:   TToBar;
    ToButto1:     TToButto;
    ToButto2:     TToButto;
    ToButto3:     TToButto;
    ToButto9:     TToButto;
    ImageLit1:  TImageLit;
    muC:          TMeuIte;
    produre                          TObjet);
    produre                          TObjet);
    produre                          TObjet);
    produre                             TObjet);
    produre                         TObjet);
    produre                          TObjet);
    produre                             TObjet);
    produre ToButto9Ck(Seder:           TObjet);
    produre ToButto1Ck(Seder:           TObjet);
    produre ToButto2Ck(Seder:           TObjet);


  private
    { Private dearati }

  publ
    { Publ dearati }
  ed;

var
  MaiForm:  TMaiForm;

ipltati

{$R *.DFM}

us
produre   TMaiFormuNek(Seder:              TObjet);
begi
  TMDICd.reateNeFi(sf);   //          新建 OLE 文档
ed;

produre   TMaiFormuOpek(Seder:              TObjet);
begi
```

```
      TMDICd.reateFroFi(sf,opedialg1.fiam,fal); //                打开存在的 Ol  文档
ed;

produre    TMaiFormuCk(Seder:                TObjet);
begi
     activedid<>n th          TMDICd(    Ativedid).; //          关闭当前子窗体
ed;

produre    TMaiFormuCk(Seder:                TObjet);
var i:iteger;
begi
   for i:=0 to MDIcdcunt-1 do //     关闭所有子窗体
      mdidre[       i]. ;
ed;

produre    TMaiFormuExitCk(Seder:          TObjet);
begi

ed;

produre    TMaiFormuSavek(Seder:           TObjet);
begi
     activedid<>n th                    //          保存 OLE 文档
     TMDIcd    (activedid).
ed;

produre    TMaiFormuSavek(Seder:           TObjet);
begi
     activedid<>n th
     TMDIcd    (activedid).
ed;

produre TMaiFormToButto9Ck(Seder:          TObjet);
begi
                          新建
ed;

produre TMaiFormToButto1Ck(Seder:          TObjet);
begi
                           打开
ed;

produre TMaiFormToButto2Ck(Seder:          TObjet);
begi
                          //保存
ed;
ed.
```

在主窗体中，使用了子窗体的两个不同的构造函数 CreateNeFi　　和 CreateFroFi　　。CreateFroFi　　的最后一个参数是控制引入的 OLE 文档是链接还是嵌入方式。另外还引用

了子窗体的 Savefi 方法，布尔参数用于控制是否要重新命名。对当前子窗体（AtiveMDICd ）操作之前，都要判断其是否为 n ，否则，将出现地址错误。

子窗体的源代码如程序如清单 8.3 所示。

（程序清单 8.3）

```
unt  Cdw;

iterfac

us Wido,        Cas,SysUti, Graph, Form, Ctro,                StdCtrl, Meus, Dialgs,
  Oltnrs;

type
  TMDICd =    cas(  TForm)
    MaiMeu1:    TMaiMeu;
    ppmMeu:  TPopupMeu;
    Edit1: TMeuIte;
    muCut:  TMeuIte;
    muCpy:    TMeuIte;
    muPaste:  TMeuIte;
    muPasteSpeal:    TMeuIte;
    muCar:    TMeuIte;
    N1: TMeuIte;
    muObjetProp:    TMeuIte;
    muInrtObjet:      TMeuIte;
    Oltair1:          TOltair;
    SaveDialg1:  TSaveDialg;
    produre    Form(Seder:         TObjet;  var Ati:     TCti);
    produre Edit1Ck(Seder:        TObjet);
    produre                        TObjet);
    produre                          TObjet);
    produre                          TObjet);
    produre                                TObjet);
    produre                                TObjet);
    produre                               TObjet);
    produre                          TObjet);
    produre    ppmMeuPopup(Seder:    TObjet);
    produre    FormQuery(Seder:         TObjet;  var Can: Boan);
  private
    { Private dearati }
    FiNam    :TFiNam;
    saved:boan;
    produre    verbck(    sder:Tobjet);
  publ
    { Publ dearati }
    ctructor     createfrofi(     aor:Twtro;FFiam:Tfiam;lked:boan);
    ctructor     createfi(        aor:Twtro);
    produre              nfi:boan);
```

```
  ed;
var
  mdid:Tmdid;
ipltati

{$R *.DFM}

produre    TMDICd.Form(Seder:              TObjet;   var Ati:      TCti);
begi
               caFre;
ed;
ctructor     TMDICd.reatefi(             AOwr:Twtro);
begi
                      Ar);
  muirtobjetck(sf); //             插入新对象
  if otair1.NeInrted    thtair1.DoVerb(otair1.priaryverb);
  saved:=fal;
  fiam:='';
ed;
ctructor     TMDICd.reatefrofi(          aor:Twtro;FFiam:Tfiam;lked:boan);
begi
                     aor);
  if lked th //      通过 lked    变量来判断是链接还是嵌入
      otair1.reateLikToFi(              ffiam,fal)
  e
                                          FFiam,fal);
  capti:=    FFiNam;
  FiNam    :=FFiam;
  saved:=fal;

ed;
produre    TMDICd.avefi(       nFi:boan);
begi
                                nfi th


  otair1.aveasdoumt(fiam); //                  保存 OLe 文档
  capti:=    FiNam;
  Oltair1.Modifid :=fal;
ed;
produre TMDICd.Edit1Ck(Seder:            TObjet);
begi
 //初始化菜单各项的显示状态
  mucut.Enabld :=otair1.State <>            oEmpty;
  muCpy.Enabld :=otair1.State <>              oEmpty;
  muPasteEnabld :=otair1.anpaste;
  mupastepealEnabld :=otair1.anpaste;
  muobjetprop.Enabld :=otair1.State <>              oEmpty;
  mucar.Enabld :=otair1.State <>              oEmpty;

ed;
```



```
                                  th fiam:=savedialg1.fiam
```

```
produre    TMDICd.uCutCk(Seder:              TObjet);
begi
                                    到剪贴板并 detroy  对象
   otair1.DetroyObjet ;
   fiam:='';
ed;

produre    TMDICd.uCpyCk(Seder:              TObjet);
begi
                                    到剪贴板
ed;

produre    TMDICd.uPastek(Seder:             TObjet);
begi
                          opty)or(     magedlg('    Deteurreto
                                    mrok) th
                               粘贴 o   文档
ed;

produre    TMDICd.uPasteSpealk(Seder:                TObjet);
begi
                           opty)or(     magedlg('    Deteurreto
                                    mrok) th

ed;

produre    TMDICd.uInrtObjetCk(Seder:                TObjet);
begi
                           opty)or(     magedlg('    Deteurreto
                                    mrok) th

ed;

produre    TMDICd.uObjetPropCk(Seder:                TObjet);
begi
                                          属性对话框
ed;

produre    TMDICd.uCarCk(Seder:              TObjet);
begi
                           opty)or(     magedlg('    Deteurreto
                                    mrok) th
   begi

      ppmu.tag:=0;
      w      ppmu.teunt<>0 do
         ppmu.te[0].fre;
   ed;
ed;
```

```
produre    TMDICd.ppmMeuPopup(Seder:        TObjet);
var i:iteger;
begi
     ppmu.tag>0 thxit;
   if(otair1.State <>          oEmpty) and (otair1.ObjetVerbsunt >0) th
   begi
     //得到当前 OLE 服务程序所能提供的命令，然后逐个写到 ppmMeu   中
      for i:=0 totair1.ObjetVerbsunt -1 do
         ppmMeu.IteInrt(          i,nte(otair1.bjetverbs[                    i],0,fal,true,verbck,0,''));
      ppmu.tag:=1;
   ed;
ed;
produre    TMDICd.verbck(       sder:Tobjet);
begi
                                                      TMeuIte(sder).apti));
ed;
produre    TMDICd.FormQuery(Seder:             TObjet;   var Can: Boan);
begi
                                            //如果已经被修改过，则换名保存
ed;

ed.
```

在子窗体的代码中，读者要注意以下几点。

♦ 我们根据主窗体建立子窗体的方法，编写了两个构造函数：一种方法是用 NEW 命令新建 OLE 文档，使用构造函数 CreateNeFi ：另外一种是打开已经有的 OLE 文档，使用构造函数 CreateFromFi ；

♦ 构造方法 CreateNeFi 中，调用了过程 mnuInsertObjetCk 来插入新对象。而 CreateFroFi 中通过了 lnked 变量来判断是链接还是嵌入方式，并且调用了 TOLECntair 的不同方法来实现这个功能；

♦ 在过程 SaveFi 中，存盘使用的是 SaveDocument 方法，而不是 SaveToFi 方法，如果使用后者，保存的是一个 OLE 对象包，并非是 OLE 服务器所要求的格式，下次就不能使用 OLE 服务器环境打开此文件了；

♦ 在子窗体内有个弹出式菜单 ppmMenu，在设计时并没有使用。运行期间，这个弹出式菜单是将 OLE 服务器提供的命令引出来，过程 ppmMenuPopup 就是响应事件 OnPopup 的。在这个过程中，首先通过属性 Objetverbs 得到当前 OLE 服务程序能提供的命令，然后逐个填写到 ppmMenu 上，当 ppmMenu 上的命令被选中的时候，调用过程 VerbCk 来执行，执行时调用了方法 DoVerb。

8.3　结构化存储

结构化存储是一项向磁盘写对象或者数据的技术。这项技术提供了以前用过的标准文件输入/输出提供的所有服务，可以在盘中生成文件，也可以生成子目录，也可以向文件或者子目录中读写数据，这些都可通过许多技术来完成。结构化存储和标准文件输入/输出的

faxaifdtoMdbhoidid1.bjetverbsdexof(

不同之处在于：每一套结构化存储的文件和目录都处于一个大文件内部，很像单个 GDB 文件中包含的 InterBas 表。

一个结构化存储文件称为混合文件。在混合文件中的“目录”称为存储器，文件被称为流，其他类型的对象也可以存入混合文件中。

我们用一个小例子来说明结构化存储，它展示了如何生成一个结构化存储文件以及如何读取它的内容。

♦ 选择 Fi 菜单中的 Ne ，创建一个新的工程，主界面上放置两个按钮。设计界面如图 8.6 所示。

图 8.6 主界面

其中 Button1 的 Caption 为“writetream ”，nam 为 bWriteStream， Button2 的 Caption 为“Read Stream”，Nam 为“bReadStream”。

♦ 保存 Unit1 为 main.pas，保存项目为 projet1.dpr 。

整个程序的源代码如程序清单 8.4 所示。

（程序清单 8.4）

```
unt mai;

iterfac

us
  Wido, Meage,          SysUti, Cas, Graph, Ctro, Form, Dialgs,
  StdCtrl ,sapi,CObj,          AtiveX,StdVc;

type
  TForm1 = cas(  TForm)
    bWritetream:  TButto;
    bReadStream: TButto;
    produre   Formreate(Seder:    TObjet);
    produre   Form(Seder:         TObjet;  var Ati:     TCti);
    produre   bWritetreamk(Seder:         TObjet);
    produre   bReadStreamk(Seder:        TObjet);
  private
    { Private dearati }
  publ
    { Publ dearati }
  ed;

var
  Form1: TForm1;

ipltati
```

```
{$R *.DFM}

produre TForm1.Formreate(Seder:        TObjet);
begi
   OlIntialze(n);   //      初始化 o


produre TForm1.Form(Seder:             TObjet;   var Ati:      TCti);
begi
   OlUntialze;
ed;

produre TForm1.bWritetreamk(Seder:             TObjet);
var
   hr:HReult;
   Storage:IStorage;
   Stream:IStream;
   Size:LogInt;
   S:strig;
begi
 //创建混合文件
   hr:=StgCreateDoFi('c:\    samtg',STGM_DIRECT    or STGM_CREATE or
STGM_WRITE or STGM_SHARE_EXCLUSIVE, 0,Storage);
   if hr<>s_ok th

   hr:=Storagereatetream('    mystream',STGM_DIRECT or STGM_CREATE or
STGM_WRITE or STGM_SHARE_EXCLUSIVE,
                                0,0,Stream);
     //createtream   在一个存储器对象里面生成一个文件.好象由
     //stgcreaedoFi      生成的存储器是一个新硬盘分区，由 IStoragereatetream      返回的流//是这个
新区的第一个文件
   ok(hr); //         检查合法性
   s:='sam';
   streamrite(    pcar(s),4,@sze); //   把字符串写入流中
   if sze<>4 th

ed;

produre TForm1.bReadStreamk(Seder:             TObjet);
var
   Storage:IStorage;
   Stream:Istream;
   hr:Hreult;
   s:Pcar;
   sze:lgInt;
begi
 //stgOpetorage    用于打开一个已经存在的混合文件
   hr:=stgOpetorage('c:\    samtg',n,STGM_DIRECT      or STGM_READ or
STGM_SHARE_EXCLUSIVE, NIL,0,Storage);
   Olk        (hr);
```


dge('wrog'szeritte');

```
    //storagepetream      用于打开文件的一个特定流
    hr:=storagepetream('      MyStream',n,STGM_DIRECT   or STGM_READ or
STGM_SHARE_EXCLUSIVE, 0,STREAM);
    Olk        (HR);
    GetMe(s   ,4);
    Streamread(s,4,@sze);//  得到字符串
    sage(s);

ed;

ed.
```

在本程序中，读者应该注意以下几点。

♦ 在它的 OnCreate 和 Onose 处理函数中，分别调用了 OlInitialze 和 OlUninitialze 函数。结构化存储是 OLE 属性的一部分，所以必须调用 OlInitialze, 而不是 CoInitialze 。传给 OlInitialze 的参数是一个 Imaloc 的实例，通常可设置为 nil；

♦ 生成一个混合文件的代码在 produre TForm1.bWritetreamk(Sender: TObjet) 中， 这个方法里的两个关键函数是 stgCreateDocFi 和 Istorage.CreateStream，前者用于返回一个初始化的 Istorage 对象，后者用于返回一个初始化的 Istream 对象；

♦ 当使用混合文件时，进入的方式是非常重要的。关键是可以用直接方式或者处理化方式。处理化方式是使用数据库进行处理。当发现有错时，可返回。实际上，如果采用处理化方式，可以对一个混合文件进行较大的改动，然后撤消。

♦ 流可以被写入、读取和复制。所有的混合对象都有多个流，所以必须确认自己想用的流。

8.4 本章小结

OLE 现在是一个很大的技术问题，并不再明确地指对象的链接和嵌入，而是指对象的链接与嵌入、OCX、结构化存储、自动化和拖放技术等一系列技术。OLE 技术是基于组件对象模型(COM)上的，绝大部分内容与获取多态性方法、二进制文件、进程、机器或操作系统之间的进程通信有关。不管这种交换是在 DLL 和可执行程序之间、两个可执行文件之间、对象和对象之间或是机器和机器之间，OLE 技术通常以一些边界的形式进行通信。