

Delphi

深度探索

陈省

Delphi 深度探索

陈 省

华中科技大学出版社

· 武汉

图书在版编目(CIP)数据

Delphi 深度探索/陈省 主编
武汉:华中科技大学出版社, 2002年2月
ISBN 7-5609-2648-7

- I. D…
- II. 陈…
- III. 计算机软件-Delphi
- IV. TP31

Delphi 深度探索

陈省 主编

责任编辑:周筠 郑兆昭
责任校对:章红

封面设计:潘群
责任监印:张正林

出版发行:华中科技大学出版社

武昌喻家山 邮编:430074 电话:(027)87545012

录排:华中科技大学惠友科技文印中心
印刷:湖北新华印务有限公司

开本:787×1092 1/16

印张:30.75

字数:580 000

版次:2002年2月第1版

印次:2002年2月第1次印刷

印数:1—5 000

ISBN 7-5609-2648-7/TP·454

定价:49.80元(含ICD)

(本书若有印装质量问题,请向出版社发行部调换)

序

Delphi 的书籍在国内已经出版了不少，可是好的作品还是屈指可数。已出的书大致有以下三类：第一类是从入门到精通的系统讲解，内容覆盖面非常广，主要是外版图书，经典的作品有 Charlie Calvert 的《Delphi X Unleashed》，Steve Teixeira & Xavier Pacheco 的《Delphi X Developer's Guide》，Marco Cantù 的《Mastering Delphi X》(X 表示 Delphi 的不同版本)这三本都是大部头的经典著作，有不同版本的中译本，使用 Delphi 的人至少要拥有其中一本，写这种书籍对作者要求很高；第二类是专题领域的著作，李维先生的三本多层应用系列就是其中的代表，这种书籍要求作者在专门领域有很深的研究；第三类就是以应用为主的著作，相对系统性差一点，不过实用性更强，这方面最近出版的陈宽达的《Delphi 深度历险》即是代表作。

非常遗憾，这些好作品基本上都是引进国外和台湾地区的版权。大陆出版的原创 Delphi 书籍，不是作者本身水平有限，就是作者也许技术水平还不错，但却不能把自己的经验和心得很好地表述成文，或者没有花足够的时间去构思创作。

《Delphi 深度探索》属于第三类的作品，比起前两种类型，这种作品应该说容易写作一些，但对作者本身的要求并不低，陈省(网名 Hubdog)在 Delphi 上钻研颇深，国内多个技术论坛上都可以见到他的身影，难得的是他花了大量的时间去精心构思这样一本书籍，内容虽然看起来涉及领域不算广，但却包含了丰富的内涵，很多部分其实是独立成章的，本书有很高的实用价值，充分展示了 Delphi 的强大功能。

这本书对读者要求也不低，读者必须基本掌握 Delphi 编程，并具有一定实际经验。仔细体会本书其中的功能是如何实现的，将大大提高你应用 Delphi 的实力。

我个人还非常喜欢本书的“工具篇”这一部分，GExperts, CodeSite, MemProof 也是我喜欢的工具，Delphi 的强大还在于它丰富的第三方控件和工具。就像好酒的酿造需要好水一样，好的程序员需要最好的开发工具，现在的开发工具具有非常强大的功能，可极大地提高程序员的工作效率。好书的作用就更大了，我衷心希望今后能看到越来越多大陆原创的高质量书籍。

蒋涛

2000年12月于北京

05/68/02

前 言

从编程和软件开发图书市场上来看，Visual C++的书长期一直都占主流，大部头的和深入探索内部机制的书相对很多，而 Delphi 相关的中高层次的图书要少得多，而且精品比较少，大部分是老外写的，国人所写的值得收藏的无疑只有台湾著名的 Delphi 专家李维写的系列丛书了（由机工社出版，共三本）。在内容方面，国内所出版的 Delphi 图书也主要集中在数据库开发方面，这就难免给人以一种印象，好像 Delphi 就是一个简单易学的快速数据库开发工具，用来开发数据库非常方便，但用于其他关键任务就有点不堪重任了。其实不然，Delphi 早已经在应用软件等方面展现了它的强大功能，在整个软件开发领域中，Delphi 早就在网络、游戏、系统开发等各个方面大展拳脚了，最明显的就是 Foxmail、NetVampire 等程序上的应用，但无论是翻译还是国内编著介绍它的书都很稀少，即或稍有涉及，也是浅尝辄止，没有搔到痒处。本书期望能够尽量在适度的篇幅内，展现 Delphi 在一些应用开发方面的强大能力，查缺补漏，期望能让广大程序开发者看了这本书后能够得到一些另类的体验，发现原来 Delphi 还有这么强大的功能呀！

本书分成四个部分，第一部分是 COM 篇。毫无疑问，未来 Windows 操作系统将完全架构于通用对象模型（COM）的基础上，但已有的 Delphi 的书在这方面的论述显得很薄弱、零散，唯一一本老外写的介绍了 COM 的书，也是对于原理讲得比较多，对于应用则讲得比较少，给人以一种“雾里看花”的感觉，本书则从 COM 的更为偏向应用的角度讲述，期望读者通过书中的各个实例，能对 COM 的应用有更为清晰和感性的认识。

第二部分就是外壳（Shell）篇，Windows 之所以能够占有操作系统如此大的份额，其方便和漂亮的外壳操作界面无疑是一个重要的砝码，如能将我们的程序同漂亮外壳无缝地结合起来，无疑会让用户添加很多的印象分，但遗憾的是不知出于什么原因，微软在这方面的资料非常欠缺，这就为这方面的开发造成了很大的困难。本书期望能够通过揭示其冰山之一角，为大家展现操作系统内部更为精彩的世界。

如果说微软公司的文档有所欠缺，Borland 公司的文档就只能说是 very very 欠缺了。程序员们普遍认为 Borland 公司的 VCL 的架构绝对是超先进的，比微软的 MFC 领先了一个时代，但由于其在经济实力上同微软无法相比，使得其无法完成特殊 VCL 的详细开发文档，这无疑极大地限制了程序员们对 VCL 架构的扩展。最具讽刺意味的就是它的集成开发环境（IDE）的扩展的实现 Open Tools API 了，名为开放工具接口，但只提供了一些超长的接口声明单元和简单的注释，一切开发都需要半靠注释，半靠猜测。针对这些问题，本书的第三部分特殊 VCL 篇期望能够揭开覆在其上的那层薄薄轻纱。

工欲善其事，必先利其器，好花还需绿叶扶持。Delphi 虽然很强大，但它也不是万能的，因此，还需要寻找和掌握那些其他人开发的强有力的工具来进一步提高工作效率，第四部分工具篇中提到的 GExperts、CodeSite、MemProof 等无疑是其中的佼佼者。

陈 省

2001 年 10 月于北京

hubdog@263.net

<http://hubdog.myrice.com>

目 录

第一篇 COM 部分

1.1	ACTIVEX 部分.....	(3)
1.1.1	ActiveX 控件之消失的事件.....	(3)
1.1.2	创建 ActiveX 控件之高级编辑界面.....	(10)
1.1.3	数据库明了的 ActiveX 控件.....	(20)
1.2	基于 COM 的 OFFICE 开发.....	(24)
1.2.1	Office 自动化编程.....	(24)
1.2.2	创建 Office 2000 插件.....	(55)
1.3	基于 COM 的拖放技术.....	(62)
1.3.1	基于 COM 的拖放.....	(62)
1.3.2	OLE 相关对话框.....	(90)
1.3.3	使 TRichEdit 支持 COM 拖放.....	(97)
1.4	WEB 相关技术.....	(103)
1.4.1	解析 XML.....	(103)
1.4.2	XML 和 XSLT.....	(113)
1.4.3	Soap 技术应用.....	(120)
1.5	基于 COM 的数据库开发.....	(129)
1.5.1	ADOX 的数据库开发.....	(129)
1.5.2	SQL Server - DMO 数据库开发.....	(148)
1.5.3	OLE 结构化存储及其在公文包型数据库中的应用.....	(166)
1.6	其 他.....	(182)
1.6.1	控制 Script Control 使用.....	(182)
1.6.2	活动目录开发.....	(191)
1.6.3	常见 COM 问题解答.....	(199)

第二篇 SHELL 部分

2.1	外壳扩展.....	(225)
2.1.1	搜索扩展.....	(225)
2.1.2	文件飞跃提示扩展.....	(231)
2.1.3	拖放控制扩展.....	(235)

2.1.4	命名空间扩展	(241)
2.1.5	实现 AutoComplete	(268)
2.1.6	属性页扩展	(273)
2.1.7	外壳执行操作记录器	(282)
2.2	未经公开的外壳奥秘	(288)
2.2.1	PItemIDList 的基本概念	(288)
2.2.2	用外壳接口对系统进行管理	(300)
2.2.3	外壳对话框	(310)
2.2.4	外壳事件通知	(324)

第三篇 特殊 VCL 部分

3.1	属性分类及其实现	(339)
3.2	OPEN TOOLS API	(345)
3.2.1	简介	(345)
3.2.2	Hello World	(347)
3.2.3	主要接口概述	(349)
3.2.4	消息通知器	(355)
3.2.5	将 Winamp 集成到 Delphi 中	(358)
3.2.6	编辑器增强功能	(375)
3.2.7	自动规范控件前缀命名的专家	(384)
3.2.8	To-Do List 的增强	(391)
3.2.9	其他 OTA 相关问题	(398)
3.3	ACTION 开发之七种武器	(413)
3.4	自绘画的属性编辑器	(427)

第四篇 工具部分

4.1	NO.1 之 GEXPERTS	(437)
4.1.1	GExperts 应用指南	(437)
4.1.2	GExperts 编辑器的增强功能	(440)
4.1.3	其他专家	(441)
4.2	内存泄漏清道夫—MEMPROOF	(454)
4.3	CODESITE 应用指南	(462)
4.4	异常杀手—EXCEPTIONALMAGIC	(469)
4.5	代码格式修正专家	(471)

第一篇

COM 部分

1.1 ActiveX 部分

1.1.1 ActiveX 控件之消失的事件

Delphi ActiveX 框架

Delphi ActiveX (DAX) 框架使我们能够很容易使用转换向导从 Delphi 本身的控件 (VCL) 出发创建 ActiveX 控件, 而这一切不需要我们具备很多的 COM 知识, 生成的 ActiveX 控件可以应用在 VC、VB 等不同的开发环境中, 这就可以避免在不同的开发环境中重复开发组件, 实现真正的代码重用。

创建 ActiveX 控件涉及到生成一个 VCL 的 COM 封装, 封装类继承于 TActiveXControl 实现了一个对 Delphi 控件的引用参考, Delphi 的 VCL 控件要想能够转换为 ActiveX 控件, 必须是继承于 TWinControl。这是因为 COM 封装和 VCL 控件之间的通信需要一个窗口句柄, 而只有 TWinControl 的子类才有句柄。

封装类通过属性和方法向其他的对象公开了 Delphi 控件的功能。同时它也负责创建管理和销毁嵌入的 VCL 控件。COM 封装类通常还负责响应控件产生的事件, 并提交给 ActiveX 控件容器。

听起来好像有点复杂, 幸好 Delphi 提供了 ActiveX 控件向导可以极大地简化这一过程。下面让我们把一个最常见的 TListBox 控件转化成 ActiveX 控件。首先选 File|New|ActiveX 页, 如图 1.1 所示。

然后选择 ActiveX Control 项, 显示一个对话框, 第一步是选择 VCL 的类名, VCL 类名下拉框中列出了全部 TWinControl 的子类 (不包括使用 RegisterNonActiveX 函数注册的控件), 从列表我们可以看到 TListBox, 但你找不到 TLabel, 因为 TLabel 继承于 TGraphicControl (它是没有窗口句柄的), 自然像那种 TTimer 的非可视控件就更是没有了, 如图 1.2 所示。

当我们选好了类名, 向导会自动给其他编辑框添上缺省值。这里我们使用它的缺省值, 同时选中 ActiveX Control Options 的三个选项, 然后点 OK 按钮。

ActiveX 控制选项

图 1.2 显示了全部可选的 ActiveX 控制选项, Include Design-Time License 选项表明设计时许可证将被创建, 它可以防止控件在设计环境中被非法使用, 如果选择了这个选项, 向导会生成一个 LIC 文件, 里面包含了许可证信息。用户必须有这个 .LIC 文件才能在开发环境中使用相应的 ActiveX 控件。Include Version Information 选项使你可以在项目中添加像版权说明、版本号等版本信息。具体版本信息的设定可以在创

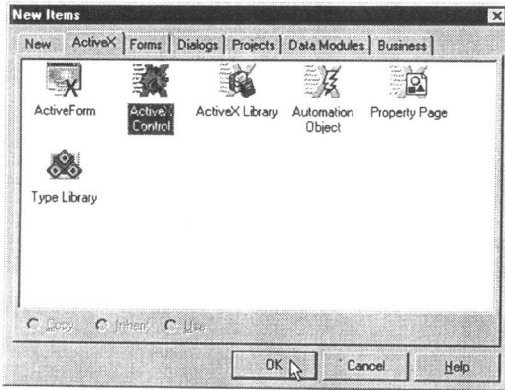


图 1.1

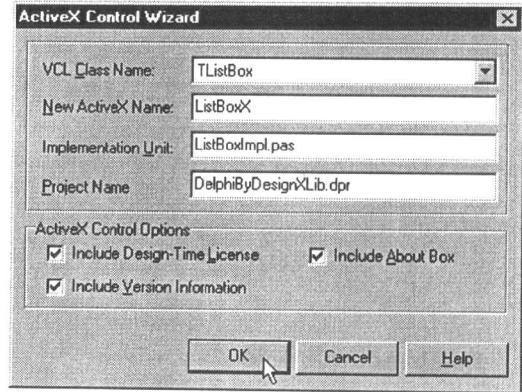


图 1.2

建好项目后，选 Project|Options 然后切换到 Version-Info 页面来进行。(注意：尽管版本信息是一个可选的选项，但如果你打算在 Visual Basic 4.0 及以上中使用 Delphi 生成的 ActiveX 控件，你必须选定它)。Include About Box 选项表明你是否为 ActiveX 控件创建一个“关于对话框”。“关于对话框”是一个独立的窗体单元，你可以编辑它来显示你需要的版本信息或其他信息。关于信息可以通过在开发环境中点击 About 属性来调用。

生成单元文件

一切设定好后，点 OK 按钮。向导首先创建一个 ActiveX Library 项目。接着创建一个实现单元，来实现对原生的 Delphi 控件的 COM 封装。封装类是 TActiveXControl 一个简单的子类，然后是类型库和类型库接口单元（类型库实际上是一个二进制文件，里面定义了数据类型、接口、方法以及 ActiveX Library 要公开的对象。类型库接口单元包含了对应于类型库中信息的 Pascal 声明）。最后是许可证文件，关于对话框和源码文件。

编译和注册 ActiveX 控制

现在我们可以编译 ActiveX Library 项目来生成包含 ActiveX 控件的 OCX 文件。我们可以直接调用 Run|Register Sever 来注册 ActiveX Sever。然而在这之前，我建议保存全部的项目文件。如果你没有先保存项目文件，那么储存在注册表中的路径就是你当前的路径而不一定是你项目的真正路径。

注册后我们就可以在 Visual Basic 等程序中使用 Delphi 生成的 ActiveX 控件了，如图 1.3 所示。

ActiveX 转换过程中的问题

虽然生成的 ListBoxX 控件使我们能在 Visual Basic 中使用 TListBox 的多项功能。但 ListBoxX 控件并没有实现 TListBox 的全部功能。比如，TListBox 的 OnDrawItem 的事件定义在 ListBoxX 中是找不到的。回想一下在 Delphi 里我们经常

使用 OnDrawItem 事件来支持用户自绘画功能。没有了这个事件，ActiveX 版本的 ListBox 就无法支持这一特性，即使 Style 属性设定为 lsOwnerDrawFixed。

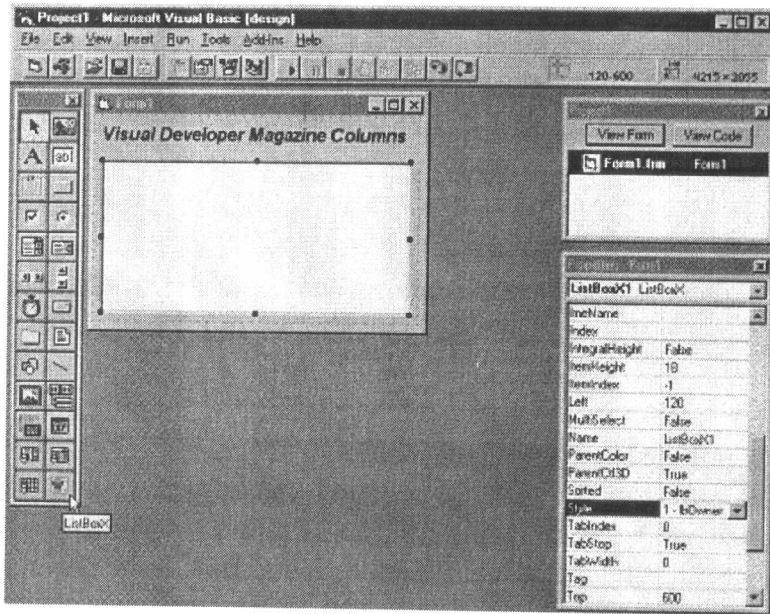


图 1.3

要理解为什么这个事件没有出现在 ActiveX 中，我们需要研究一下事件类型，特别是要研究一下传递给事件的参数类型。OnDrawItem 事件声明为 TDrawItemEvent 类型，定义如下：

```
TDrawItemEvent = procedure( Control: TWinControl;
                             Index: Integer; Rect: TRect;
                             State: TOwnerDrawState) of object;
```

我们看到 OnDrawItem 事件处理函数接受 4 个参数。其中第三个参数阻止了事件出现在 ActiveX Control 中。因为 OLE 自动化不知道如何列集 (marshall) TRect 类型的数据。

原因在于 ActiveX 控制实际上是小型的 OLE 自动化服务器。包含 ActiveX 控件的 ActiveX 容器使用自动化来和控件通信。因此要想使一个属性、方法或事件出现在 ActiveX 版本的控件中，全部参数和返回类型必须是自动化兼容的。图 1.4 列出了全部的兼容类型。

在这个表中虽没有列出，但 ActiveX 转换向导还可以正确处理 TColor, TFont, TPicture 和 TStrings 类型的

表 1
自动化兼容类型
预定义类型
SmallInt, Integer, Single, Double, Currency, TDateTime, WideString, IDispatch, WordBool, Variant, OleVariant, SCode, Byte, and IUnknown
Enumerations defined in a type library
Interface types defined in a type library
Dispatch interfaces defined in a type library

图 1.4

属性。Color 属性也可以很容易转换，因为它实际上就是一个整数类型。而对于其他属性类型，Delphi 提供了一个定制化的接口来处理对应的 Delphi 属性。比如 IStrings 接口提供了处理 TStrings 属性的途径。

同时，一个属性是自动化兼容的并不意味着它就一定会出现在 ActiveX 控件中，向导不会转化对 ActiveX 控件无意义的属性、方法或事件。因而很多属性就没有被转化，比如：Height、HelpContext、Hint、Left、Name、ParentFont、ParentShowHint、PopupMenu、ShowHint、TabOrder、Tag、Top 和 Width 等。

此外，有两种特殊类型的属性向导也不会将其转化。这就是对象引用和数据明了属性，对象引用（就是指 Delphi 的某个属性对应于一个对象，比如 TTable 的 DataSource 实际上是引用了另外一个对象 TDataSource）在 Delphi 里是通过指针来实现的，因此是自动化不兼容的。此外，ActiveX 本身也没有提供一个标准的方式来使一个控件同容器中的其他控件相关联。而数据明了属性 DataSource 和 DataField 没有被映射到 ActiveX 控件中去是因为 Delphi 实现的数据明了采用了完全不同于 ActiveX 的机制。当然如果非常想利用 Delphi 中好用的数据库控件，这也是可以的，但需要一些额外的工作，这将在后面介绍。必须指出的是 ActiveX 的数据控件和 Delphi 的数据控件是不一样的。这也就是为什么没有一个 Delphi 的数据明了控件出现在 ActiveX 转换向导的类名列表中的缘故，即使它可能是继承于 TwinControl 的控件。

正确的转换的关键

到目前为止 ListBoxX 其他特性都没问题，我们也可以把 Style 变成 IsOwnerDrawFixed，但是没有 OnDrawItem 事件我们无法改变它的外观显示。让我们试一种不同的方法，这就是定义一个自动化兼容的接口来实现自绘画的外观（Owner-Draw）。例如我们可以创建一个新的事件 OnColorItem，使我们能改变每一个列表项的颜色。

添加新事件

添加新的事件到 ActiveX 中需要修改类型库中的信息，修改有两种方法：通过选择

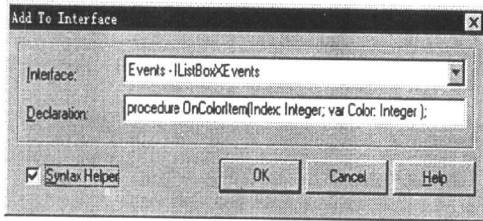


图 1.5

Edit|Add To Interface 菜单或用内置的类型库编辑器都可以，Edit|Add To Interface 菜单只有当当前文件为实现单元时才能使用。如图 1.5 所示。

图 1.5 显示了 OnColorItem 是如何加入到 IListBoxEvents 接口中的。定义一个 ActiveX 接口的新事件同 Delphi 中有点不同。不像 Delphi 中那样给事件声明一个

属性类比如 TNotifyEvent，你必须写一个将会传递给事件处理函数的带参数的过程。类似于写一个控件中的事件分配方法。OnColorItem 事件定义如下：

```
procedure OnColorItem(Index: Integer; var Color: Integer);
```

点 OK 按钮, Delphi 就把这个声明加入了类型库。这使得我们可以给 OnColorItem 事件写处理函数了。但是在这之前,还必须实现产生事件的代码,这有点类似于在 Delphi 控件中定义新的事件。生成事件属性和事件分配方法仅仅是一部分工作,我们还必须在合适的时间调用事件分配方法来产生事件。最好的产生 OnColorItem 事件的位置就是在 Delphi 控件的 OnDrawItem 事件中,实现的方法见源码清单 1。

TListBoxX 封装类定义了一个事件处理函数 DrawItemEvent 来响应嵌入的 ListBox 的 OnDrawItem 事件。DrawItemEvent 函数在 InitializeControl 方法中进行初始化,同时 DrawItemEvent 函数负责产生 OnColorItem 事件。为了产生事件,要使用 FEvents 接口来激发 OnColorItem 事件过程。Index 参数来自于 DrawItemEvent 参数列表,而 Color 参数是在 DrawItemEvent 里定义的。当 OnColorItem 事件返回时,ItemColor 变量将包括缺省的颜色或用户定义的颜色,然后 ItemColor 变量就被用来画列表项。如图 1.6 所示。

图 1.6 显示了 ListBoxX 控件在 VB 中的运行状况。虽然 ActiveX 不可能像 Delphi 中的 VCL 控件那样可以随意使用 Canvas 来画图,但是这个例子证明了利用 ActiveX 控件的灵活性,我们还是可以作出一些有趣的效果的。

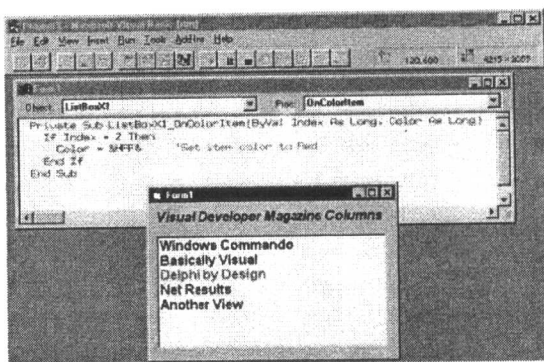


图 1.6

正如我们在上面这个例子中所看到的,实际上把 Delphi 控件转化为 ActiveX 控件并不是很容易,这是由于我们转换的是一个现成的 Delphi 控件,特别是这个控件使用了某些 VCL 的高级特性。从这方面来说,Active 控制框架不如 VCL 灵活,然而 ActiveX 是一个公认的标准,可以在除了 Delphi 外的其他环境中使用,并且它是一个二进制级别的组件,无须编译就可以被应用程序嵌入使用,它必然要牺牲灵活性来获得通用性。

结论

如果你想从头开始创建一个 ActiveX 控件,首先你的控件必须从 TWinControl 开始继承,如果想要提供自绘画的功能,必须从 TCustomControl 开始。另外,应该使用自动化兼容的数据类型来定义属性、方法、参数。如果你想转换一个已有的 VCL 控件为 ActiveX,注意一定要改变参数类型,否则方法、属性等等是不会被转换的。

源码清单 1 - ListBoxImpl.pas

```
unit ListBoxImpl;
interface
uses
  Windows, ActiveX, Classes, Controls, Graphics, Menus, Forms,
  StdCtrls, ComServ, StdVCL, AXCtrls, DelphiByDesignXLib_TLB;
type
```

```
TListBoxX = class(TActiveXControl, IListBoxX)
private
    { Private declarations }
    FDelphiControl: TListBox;
    FEvents: IListBoxXEvents;
    procedure ClickEvent(Sender: TObject);
    procedure DblClickEvent(Sender: TObject);
    procedure KeyPressEvent(Sender: TObject; var Key: Char);
    // Add a custom event handler for the OnDrawItem event
    procedure DrawItemEvent( Control: TWinControl; Index: Integer;
        Rect: TRect; State: TOwnerDrawState );
protected
    { Protected declarations }
    procedure InitializeControl; override;
    procedure EventSinkChanged(const EventSink: IUnknown); override;
    procedure DefinePropertyPages(
        DefinePropertyPage: TDefinePropertyPage); override;
    function Get_BorderStyle: TxBorderStyle; safecall;
    function Get_Color: TColor; safecall;
    function Get_Columns: Integer; safecall;
    function Get_Ctl3D: WordBool; safecall;
    function Get_Cursor: Smallint; safecall;
    function Get_DragCursor: Smallint; safecall;
    function Get_DragMode: TxDragMode; safecall;
    function Get_Enabled: WordBool; safecall;
    function Get_ExtendedSelect: WordBool; safecall;
    function Get_Font: Font; safecall;
    function Get_ImeMode: TxImeMode; safecall;
    function Get_ImeName: WideString; safecall;
    function Get_IntegralHeight: WordBool; safecall;
    function Get_ItemHeight: Integer; safecall;
    function Get_ItemIndex: Integer; safecall;
    function Get_Items: IStrings; safecall;
    function Get_MultiSelect: WordBool; safecall;
    function Get_ParentColor: WordBool; safecall;
    function Get_ParentCtl3D: WordBool; safecall;
    function Get_SelCount: Integer; safecall;
    function Get_Sorted: WordBool; safecall;
    function Get_Style: TxListBoxStyle; safecall;
    function Get_TabWidth: Integer; safecall;
    function Get_TopIndex: Integer; safecall;
    function Get_Visible: WordBool; safecall;
    procedure AboutBox; safecall;
    procedure Clear; safecall;
    procedure Set_BorderStyle(Value: TxBorderStyle); safecall;
```



```
procedure Set_Color(Value: TColor); safecall;
procedure Set_Columns(Value: Integer); safecall;
procedure Set_Ctl3D(Value: WordBool); safecall;
procedure Set_Cursor(Value: Smallint); safecall;
procedure Set_DragCursor(Value: Smallint); safecall;
procedure Set_DragMode(Value: TxDragMode); safecall;
procedure Set_Enabled(Value: WordBool); safecall;
procedure Set_ExtendedSelect(Value: WordBool); safecall;
procedure Set_Font(const Value: Font); safecall;
procedure Set_ImeMode(Value: TxImeMode); safecall;
procedure Set_ImeName(const Value: WideString); safecall;
procedure Set_IntegralHeight(Value: WordBool); safecall;
procedure Set_ItemHeight(Value: Integer); safecall;
procedure Set_ItemIndex(Value: Integer); safecall;
procedure Set_Items(const Value: IStrings); safecall;
procedure Set_MultiSelect(Value: WordBool); safecall;
procedure Set_ParentColor(Value: WordBool); safecall;
procedure Set_ParentCtl3D(Value: WordBool); safecall;
procedure Set_Sorted(Value: WordBool); safecall;
procedure Set_Style(Value: TxListBoxStyle); safecall;
procedure Set_TabWidth(Value: Integer); safecall;
procedure Set_TopIndex(Value: Integer); safecall;
procedure Set_Visible(Value: WordBool); safecall;
end;
implementation
uses AboutListBox;
{ TListBoxX }
procedure TListBoxX.InitializeControl;
begin
  FDelphiControl := Control as TListBox;
  FDelphiControl.OnClick := ClickEvent;
  FDelphiControl.OnDblClick := DblClickEvent;
  FDelphiControl.OnKeyPress := KeyPressEvent;
  //添加一个定制的事件来处理 OnDrawItem 事件
  FDelphiControl.OnDrawItem := DrawItemEvent;
end;
procedure TListBoxX.EventSinkChanged(const EventSink: IUnknown);
begin
  FEvents := EventSink as IListBoxXEvents;
end;
function TListBoxX.Get_Enabled: WordBool;
begin
  Result := FDelphiControl.Enabled;
end;
procedure TListBoxX.Set_Enabled(Value: WordBool);
```