Delphi 4.0/5.0 类参考详解

秦绪佳 张济 袁野 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

(京)新登字 158 号

内 容 简 介

本书是 Delphi 4.0/5.0 语言参考手册中的一本，另一本书是《Delphi 4.0/5.0 高级类参考详解》，用来配合本书的使用。

本书内容包括了进行一般应用程序设计所要使用的类。书中详尽地介绍了 Delphi 4.0/5.0 类及类中的属性、方法和事件，还详细介绍了过程和函数以及变量、数据类型、语句等，对读者的编程有很好的参考价值。

本书内容翔实，是一套极好的语言参考手册，适合广大 Delphi 程序开发人员使用。

图书在版编目(CIP)数据

Delphi 4.0/5.0 类参考详解/秦绪佳等编著.-北京：清华大学出版社，1999
ISBN 7-302-03773-6

Ⅰ.D… Ⅱ.秦… Ⅲ.DELPHI 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（1999）第 62018 号

出版者：清华大学出版社(北京清华大学学研楼，邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者：北京市昌平环球印刷厂
发行者：新华书店总店北京发行所
开 本：787×1092 1/16 **印张**：26 **字数**：1016 千字
版 次：1999 年 11 月第 1 版 2000 年 3 月第 2 次印刷
书 号：ISBN 7-302-03773-6/TP・2118
印 数：5001～8000
定 价：48.00 元

前　言

Delphi 4.0/5.0 是 Borland 公司继 Delphi 3 后的又一力作。它比 Delphi 3.0 开发效率更高、功能更强大、使用也更加方便，受到了广大用户的欢迎。目前越来越多的用户已经采用 Delphi 4.0/5.0 来开发他们的软件，但目前市场上全面介绍其语言组成的手册还很少，这就给用户开发应用软件带来了诸多不便。根据这一情况，本书对 Delphi 4.0/5.0 的 VCL 类库进行了详细的介绍。

Delphi 4.0/5.0 与以前的版本相比增加了以下新功能：Advanced Debugging 模块窗口，能够显示当前所有映射到应用程序地址空间的模块；对 CodeInsight 功能作了很多改进，如类自动完成，符号预知，代码模板，类成员提示，参数提示等；增强对象接口技术和 DAX 技术，完全支持 ActiveX，可以很方便地创建、注册、安装、发布和使用 ActiveX 控件、ActiveForm 和 OLE 自动化对象；对 COM 的支持，由于 COM 对象具有语言和平台无关性，用 Delphi 4.0/5.0创建的对象可以与用 Visual Basic，Java，C++ 及其他语言实现的对象交互；在多层 Client/Server 环境中，COM 对象可以封装商业规则，为分布式的客户提供服务。支持 Internet 应用，可方便地实现 WinSock 编程，访问 HTTP，UDP，FTP，SMTP，POP3 和 NNTP 等服务；更具开放性与灵活性，可支持多个数据库引擎，任何一个数据库引擎都能与 Delphi 4.0/5.0的数据显示控件协调工作；MIDAS(Multi - Tiered Distributed Application Services)技术是 Delphi 4.0/5.0 的精髓，是开发多层 Client/Server 应用程序的最佳编程工具。此外，Delphi 4.0/5.0 还支持 Microsoft Transaction Server(MTS)。

鉴于 Delphi 4.0/5.0 的优良特性，为了顺应广大用户的要求，我们在积累大量开发经验的基础上编写了这套书。本套丛书是作为软件开发的参考书而非指导书编写的。

考虑到各种层次用户的需求，本套书共分两册，《Delphi 4.0/5.0 类参考详解》和《Delphi 4.0/5.0 高级类参考详解》。《Delphi 4.0/5.0 类参考详解》主要为初级使用者考虑，介绍了用户在编写 Delphi 应用程序时要用到的一些基本类和函数，对于一般应用程序的设计，如文字处理程序，这本书可以提供良好的帮助，而对高级些的应用，则需要另一本书的帮助了。《Delphi 4.0/5.0 高级类参考详解》主要为高级用户而编写，通过这些类和函数可以实现许多特殊的功能。两本书都列出了 Delphi 4.0/5.0 中的数据类型、变量、语句等内容。两本书相互配合使用，可以产生最好的效果。我们建议读者购买两本书，因为有很多的查阅内容，是在另一本书上才能找到的。而且，通过查阅这些内容，可以更好地把握 Delphi 所包含的内容，有助于从总体上提高编程的能力。

本套书内容翔实，是一套极好的语言参考手册，是从事 Delphi 程序开发和编程的广大技术人员必备的工具书，适合于各个层次的 Delphi 用户。

Delphi 4.0/5.0 的 VCL 类库十分巨大，内容也很丰富，光靠我们几个人的力量是无法完成此书的。在本书的完成过程中，还得到了许多人的大力协助，阚亚斌、陈书伟等对本书的初稿提出了许多很好的建议，修正了书中的许多错误，在此特别感谢他们对本书的出版给

予的热情支持。还有许多其他同行对本书的修订做了大量工作，在此一并表示感谢。

由于作者水平有限、经验不多，再加上时间仓促，书中的错误与不足在所难免，希望广大用户批评指正。

作　者

目　录

Axctrls 单元

TActiveForm 单元 Axctrls

类关系：TObject→TPersistent→TComponent→TControl→TWinControl→TScrollingWinControl→TCustomForm→TCustomActiveForm

TActiveForm 是经由类工厂创建的一个窗体，在 ActiveX 主应用程序中，作为一个 ActiveX 控件使用。

TActiveForm 被用于创建 ActiveX 控件窗体，从 Web 浏览器上可以显示和运行。TActiveForm 对象由 TActiveFormControl 对象创建，它是 TActiveFormControl 的一个子类。创建 ActiveForm 对象时，可以在窗体设计器上进行设计。

属性列表

ActiveFormControl	是 ActiveFormControl 对象的指针

属 性

TActiveForm.ActiveFormControl

ActiveFormControl 属性是 ActiveFormControl 对象的指针。

property ActiveFormControl: TActiveFormControl;

使用 ActiveFormControl 属性，可以对 ActiveForm 对象进行访问，该对象使所有 ActiveX 接口生效。使用 ActiveFormControl 属性，可以创建 TActiveForm 对象。

TActiveFormControl 单元 Axctrls

类关系：TObject→TComObject→TTypedComObject→TAutoObject→TActiveXControl

TActiveFormControl 是一个提供支持 ActiveX 控件所需接口的 COM 对象。

在 Delphi 窗体和 ActiveX 控件 API 之间，TActiveFormControl 对象起桥梁作用。它是 TActiveXControl 对象的一个子类，使 ActiveX 控件所需接口生效。它向 ActiveForm 对象传递适当的信息。TActiveFormControl 对象在 DAX 框架内部使用。正常情况下，不能创建该类的实例。

方法列表

FreeOnRelease	支持 IVCLComObject 接口
InitializeControl	创建 TActiveForm 实例并建立链接关系
Invoke	实现 OLE Automation IDispatch Invoke 功能
ObjQueryInterface	获取接口 ID 并返回对象请求的 COM 接口

方 法

TActiveFormControl.FreeOnRelease

FreeOnRelease 过程用于支持 IVCLComObject 接口。

procedure FreeOnRelease;

FreeOnRelease 过程单独存在以支持 IVCLComObject，并且不能被执行。

TActiveFormControl.InitializeControl

创建 TActiveForm 实例，并且在其与控件属性引用的 TActiveForm 之间建立起链接关系。

procedure InitializeControl; override;

InitializeControl 过程还通知 TActiveForm 对象完成自身的初始化。

TActiveFormControl.Invoke

实现 OLE Automation IDispatch Invoke 功能。

function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer; Flags: Word; var Params; VarResult, ExcepInfo, ArgErr: Pointer): HResult; override;

Invoke 函数可以对由 DispID 参数指定的属性或者方法进行访问。Flags 参数表示 DispID 是引用方法(DISPATCH_METHOD)、还是引用将被读的属性值(DISPATCH_PROPERTYGET)、或者引用将被设置的一个属性值(DISPATCH_PROPERTYPUT 或 DISPATCH_PROPERTYPUTREF)。Params 和 ExcepInfo 参数分别用于保存参数和异常信息，并取决于由 DispID 参数识别的属性或者方法。

如果 Invoke 函数调用成功则返回 S_OK。否则，Invoke 函数将返回一个由 Idispatch 接口定义的错误代码。

TActiveFormControl.ObjQueryInterface

获取接口 ID(IDD 参数)并返回对象请求的 COM 接口。

function ObjQueryInterface(const IID: TGUID; out Obj): HResult; override;

如果 TActiveForm 对象支持请求的接口，则 ObjQueryInterface 方法用于向关联的 TActiveForm 对象传递对请求接口的首次调用。否则，ObjQueryInterface 方法用于对自身接口的查询。

参 见 TActiveForm, TActiveFormFactory。

TActiveFormFactory 单元 Axctrls

类关系：TObject→TComObjectFactory→TTypedComObjectFactory→TAutoObjectFactory→TActiveXControlFactory

ActiveFormFactory 对象是 TActiveForm 的类工厂。

ActiveFormFactory 对象是对响应创建 TActiveFormFactory 对象和 TActiveForm 对象实例的类工厂。ActiveForm 向导在定义 ActiveForm 单元的 initialization 部分加进一个 ActiveForm 工厂对象。

方法列表

GetIntfEntry	获取调度接口的指针

方 法

TActiveFormFactory.GetIntfEntry

GetIntfEntry 方法用于获取调度接口的指针。

function GetIntfEntry(Guid: TGUID): PInterfaceEn-

try; override;

为获取一个在 TActiveForm 类中存在的接口，GetIntfEntry 方法被重设。它是 DAX 框架的内部调用方法。

参 见 TActiveForm。

TActiveXControl 单元 Axctrls

类关系： TObject→TComObject→TTypedComObject→TAutoObject

TActiveXControl 是一个抽象类，它提供了将一个 VCL 控件嵌入到一个 ActiveX 容器窗口的功能。

TActiveXControl 对一个 ActiveX 控件所需的核心行为和接口进行定义，并将这种行为与任何来自 TwinControl 的 VCL 控件连接起来。它允许控件嵌入到任何一个 ActiveX 容器中，主要包括 Internet Explorer, Visual Basic, PowerBuilder, Paradox, Borland C++, IntraBuilder 以及 Delphi。

使用 ActiveX 控件向导，可以创建一个新的 ActiveX 控件类。建立一个 ActiveX 控件的步骤，包含在开发指南或联机帮助文献以及开发中的基本 COM 应用程序中。TActiveXControl 类继承自 TAutoObject，因此，TActiveXControl 对象产生的结果是一个支持以 COM 为基础的自动控制的 COM 对象，或一个类型库，并有一个工厂。此外，TActiveXControl 还实现下列行为：

- 在一个容器(位置)内的嵌入。
- 就地激活。
- 驻留。
- 事件。
- 属性页。
- 属性浏览。

TActiveXControl 对象是一个抽象类，因此不能进行例示。应该从 TActiveXControl 对象获得一个新类，以使 ActiveX 控件生效。获得的这个类叫做执行类。通常情况下，源于 TActiveXControl 对象的执行类可使一个自动控制接口生效，并指定对象的属性和方法。

通常，不需直接创建执行类的实例。但应在包含类声明的单元的 initialization 部分创建一个工厂对象。当系统需要时，工厂对象将创建 ActiveX 控件对象。除了执行从 TautoObject 继承的接口之外，TActiveXControl 对象执行下列接口：

IPersistStreamInit, IPersistPropertyBag, IOleInPlaceActiveObject, IPersistStorage, IViewObject, IObjectSafety, IOleObject, IViewObject2, ISimpleFrameSite, IOleControl, IPerPropertyBrowsing, IOleInPlaceObject, ISpecifyPropertyPages

属性列表

ClientSite	为驻留 ActiveX 控件的容器包含一个客户位置指针
Control	表示 ActiveX 控件的 VCL 实现

方法列表

Destroy	删除 TActiveXControl 对象的实例
Initialize	为 TActiveXControl 对象创建类工厂和连接对象
ObjQueryInterface	返回关于对象请求的 COM 接口
PropChanged	产生一个 OnChanged 事件
PropRequestEdit	表明能否对指定的属性进行修改

属 性

TActiveXControl.ClientSite

ClientSite 属性为驻留 ActiveX 控件的容器包含了一个客户位置指针。

property ClientSite: IOleClientSite;

ClientSite 属性包含了一个对象的客户位置的接口指针。在一个复合的文件中，每一个被嵌入的对象都有各自的客户位置。客户位置是其被显示的地方，通过它可获取关于其存储器、用户接口以及其他资源的信息。

当需要从容器中获取客户位置信息时，使用 ClientSite 属性。

TActiveXControl.Control

Control 属性表示 ActiveX 控件的 VCL 实现。

property Control: TWinControl;

Control 属性是源自 TWinControl 的 VCL 控制器，它使 ActiveX 控件生效。

参 见 TWinControl。

方 法

TActiveXControl.Destroy

Destroy 用于删除 TActiveXControl 对象的实例。

destructor Destroy; override;

不要在应用程序中直接调用 Destroy，而应调用 Free。Free 检查 ActiveX 控件是否已释放，只有在 ActiveX 控件未被释放时才调用 Destroy。Destroy 释放使用的资源，以使 Control 变成 ActiveX 控件。Destroy 不能删除由 Control 属性引用的对象。

参 见 TActiveXControl.Destroy, TObject.Free。

TActiveXControl.Initialize

为 TActiveXControl 对象创建类工厂和连接对象。

procedure Initialize; override;

Initialize 方法用于创建类工厂对象和创造，并封装 VCL 控件。

参 见 TActiveFormControl.InitializeControl。

TActiveXControl.ObjQueryInterface

ObjQueryInterface 函数返回关于对象请求的 COM 接口。

function ObjQueryInterface(const IID: TGUID; out Obj): HResult; override;

ObjQueryInterface 函数试图确定由 IID 参数指定的接口位置，返回值在 Obj 参数中。如果找到并送回请求的接

口，则 ObjQueryInterface 函数返回 S_OK。

参 见 TActiveForm, TActiveFormFactory。

TActiveXControl.PropChanged

产生一个 OnChanged 事件。

procedure PropChanged (const PropertyName: WideString);

当由 PropertyName 参数指定的属性变化时，调用 PropChanged 过程。通过使用 IpropertyNotifySink 接口，IPropertyNotifySink 产生一个 OnChanged 事件。

TActiveXControl.PropRequestEdit

PropRequestEdit 函数表明能否对一个指定的属性进行修改。

function PropRequestEdit(const PropertyName: WideString): Boolean;

调用 PropRequestEdit 函数可以确定：由 PropertyName 参数指定的属性能否进行编辑。PropRequestEdit 函数使用 IpropertyNotifySink 接口，以确定能否对属性进行修改。可以对属性进行修改时，IpropertyNotifySink 函数返回 True。

TActiveXControlFactory 单元 Axctrls

类关系：TObject → TComObjectFactory → TTypedComObjectFactory→TAutoObjectFactory

TActiveXControlFactory 是 ActiveX 控件对象的工厂类。

TActiveXControlFactory 类是一个 ActiveX 控件对象的工厂，通过使用 TActiveXControl 抽象类来实现。定义 TActiveXControl 对象单元的 initialization 部分时，TActiveXControl 向导加入一个 ActiveX 控件工厂对象。

属性列表

MiscStatus	储存在登记名册中的状态
ToolboxBitmapID	是 ActiveX 控件的资源标识符
WinControlClass	为 ActiveX 控件执行 Windows 行为

方法列表

AddVerb	向对象的支持动词的列表中加入动词
Create	创建和初始化新对象
Destroy	释放对象的实例
UpdateRegistry	为对象增加或删除登记项目

属 性

TActiveXControlFactory.MiscStatus

MiscStatus 属性是储存在登记名册中的状态的集合。

property MiscStatus: Integer;

对于类而言，MiscStatus 属性储存在 CLSID 入口下的登记名册项目。该入口提供了关于 ActiveX 服务器对一个容器的信息。入口包含了一个 ActiveX，并定义了具体含义的二进制标志的集合。该入口不是必需的。

ActiveX 控件容器必须认可并支持以下 MiscStatus 属性位：

状态位	支持容器所需	备 注
ACTIVATEWHENVISIBLE	Yes	尽管对象没有可激活用户接口，但当其可视时仍将被激活
IGNORE ACTIVATEWHENVISIBLE	No	对非激活且无窗口控制支持是必需的
INSIDEOUT	No	对象由里至外被激活（具有与可激活用户接口一样的部署适当激活的能力），通常被嵌入的复合文件使用，而不是控件
INVISIBLEATRUNTIME	Yes	指定一个控件在设计阶段可视，在运行阶段不可视
ALWAYSRUN	Yes	表示对象一直处于运行状态，且默认句柄不延迟对象的加载，直至最后片刻
ACTSLIKEBUTTON	No	表示对象具有类似按钮的行为。它允许容器要求控件作为默认按钮或者普通按钮画出
ACTSLIKELABEL	No	表示对象行为类似一个标注。当容器的标注获得焦点时允许处理属性
NOUIACTIVATE	Yes	表示控件没有可激活用户接口
ALIGNABLE	No	表示控件能被用户对齐
SIMPLEFRAME	No	表示该控件对其他控件是一个简单的所有者
SETCLIENTSITEFIRST	No	表示控件在构造期需要很早访问客户位置
IMEMODE	No	表示控件支持输入方法标识符(IMEs)以创建扩充字符。典型地，一个支持 IME 的容器可提供与扩充 IMEMode 属性一样的控制

TActiveXControlFactory.ToolboxBitmapID

ToolboxBitmapID 属性是 ActiveX 控件的资源标识符。

property ToolboxBitmapID: Integer;

使用 ToolboxBitmapID 属性，可以获得位图的资源 ID，它常被用于表示设计选项板上的控件。

参 见 TActiveXControlFactory.Create。

TActiveXControlFactory.WinControlClass

WinControlClass 属性是 VCL 控件类，它为 ActiveX 控件执行 Windows 行为。

type TWinControlClass = class of TWinControl;

property WinControlClass: TWinControlClass;

当创建 ActiveX 控件时，工厂对象对 VCL 控件的类信息保存一个引用，以便它能要求类对象制造一个 VCL 控件。

参 见 TActiveXControl.Control, TWinControl。

方 法

TActiveXControlFactory.AddVerb

AddVerb 过程用于向对象的支持动词的列表中加入一个动词。

procedure AddVerb(Verb: Integer; const VerbName: string);

使用 AddVerb 过程，加入一个主程序能够对 ActiveX 控件使用的动词。Verb 参数传递一个数字以识别动词。用户定义的动词通常为小的正数。

当用户以动词的一个选择表示时，将显示 VerbName 参数传递的文本。在一些 ActiveX(称之为 OLE)容器中，当用户在一个控件上击鼠标右键时，动词文本显示在一个弹出式菜单上。

在字符串中的一个字符前放置一个"&"号，在弹出式菜单上该字符显示为下划线。

ActiveX 定义了以下常用的动词：

动词名称	动词
OLEIVERB_PRIMARY	0
OLEIVERB_SHOW	-1
OLEIVERB_OPEN	-2
OLEIVERB_HIDE	-3
OLEIVERB_UIACTIVATE	-4
OLEIVERB_INPLACEACTIVATE	-5
OLEIVERB_DISCARDUNDOSTATE	-6
OLEIVERB_PROPERTIES	-7

参 见 TActiveXControlFactory.UpdateRegistry。

TActiveXControlFactory.Create

Create 构造函数用于创建和初始化一个新的 TActiveXControlFactory 对象。

constructor Create(ComServer: TComServerObject; ActiveXControlClass: TActiveXControlClass; WinControlClass: TWinControlClass; const ClassID: TGUID; ToolboxBitmapID: Integer; const LicStr: string; MiscStatus: Integer);

对于包含属性页工厂的模块，ComServer 参数是一个 COM 服务器对象的指针。通常它为一个全局变量。ComServer 在 ComServ 单元中声明。

ActiveXControlClass 参数是使 ActiveX 控件生效的控制器对象的类对象。该对象来自于 TActiveXControl。

WinControlClass 参数是使控件的 Windows 行为生效的 VCL 控件对象的类对象，该对象来自于 TWinControl。

ClassID 参数是 ActiveX 控件的 GUID。每一个 COM 对象都有一个唯一的 GUID。可以参见如何产生一个 GUID 部分。

LicStr 参数是由这个工厂创建的控件的许可证字符串。如果由这个工厂创建的控件是未经许可的，LicStr 参数为空。MiscStatus 参数是被储存在登记名册中的状态位。

参 见 TActiveXControlFactory.MiscStatus, TActiveXControlFactory.ToolboxBitmapID。

TActiveXControlFactory.Destroy

删除 TActiveXControlFactory 的实例。

destructor Destroy; override;

在应用程序中不要直接调用 Destroy，应调用 Free。Free 可以检查 TActiveXControlFactory 实例是否已被释放。只有在 TActiveXControlFactory 实例未被释放时，才能调用 Destroy。

参 见 TObject.Free。

TActiveXControlFactory.UpdateRegistry

UpdateRegistry 过程用于为对象增加或删除登记项目。

procedure UpdateRegistry(Register: Boolean); override;

从系统登记名册中增加或删除控件时，调用该方法。该方法的默认行为是增加标准 ActiveX 控件登记钥匙、服务器文件信息以及登记名册的类型库信息。它还可以增加用 AddVerb 方法加入的动词。如果 Register 参数为 True，控件将向系统登记名册中加入其钥匙。如果 Register 参数为 False，控件将向系统登记名册中删除其钥匙。

如果想让控件储存比基本实现储存更多的信息，重设该方法。如果重设了该方法，应确认是用正确的顺序调用继承的方法。当增加登记名册项目时(Register 参数为 True)，应在增加自定义项目前调用继承的方法。当删除登记名册项目时(Register 参数为 False)，应在删除自定义项目后调用继承的方法。

参 见 TActiveXControlFactory.AddVerb。

TActiveXPropertyPage　　单元 Axctrls

类关系：TObject→TComObject

TActiveXPropertyPage 表示一个 COM 对象，它提供了支持一个 ActiveX 属性页的接口。

该类由 TPropertyPage 控件内部使用，在 Delphi 窗体和 ActiveX 属性页 API 之间起桥梁作用。

属性列表

PropertyPageImpl	COM 对象服务器部分的实现

方法列表

Destroy	删除实例
Initialize	对子类的实例进行初始化

属 性

TActiveXPropertyPage.PropertyPageImpl

PropertyPageImpl 属性表示 COM 对象的 COM 服务器部分的实现。

property PropertyPageImpl: TPropertyPageImpl;

PropertyPageImpl 属性是 IpropertyPage 接口实现的

COM 服务器部分。应用程序不用创建 PropertyPageImpl 对象；当用户使用 ActiveX 控件或者 ActiveForms 向导创建一个 ActiveX 控件对象时，PropertyPageImpl 对象被自动创建。只要 PropertyPageImpl 属性使属性页部分生效，TActiveXPropertyPage 就使 COM 对象的 COM 服务器部分生效。

方 法

TActiveXPropertyPage.Destroy

删除 TActiveXPropertyPage 的实例。

destructor Destroy; override;

不要在应用程序中直接调用 Destroy，而应先调用 Free。Free 检查 TPropertyPage 对象是否已被删除，只有当 TPropertyPage 对象未被删除时，才能调用 Destroy。

参 见 TObject.Free。

TActiveXPropertyPage.Initialize:

Initialize 用于对 TActiveXPropertyPage 子类的实例进行初始化。

procedure Initialize; override;

在使用 TActiveXPropertyPage 子类的实例前，重设 Initializ 方法，以便对其进行初始化。

参 见 TComObject.Create。

TActiveXPropertyPageFactory 单元 Axctrls

类关系：TObject→TComObjectFactory

TActiveXPropertyPageFactory 使 ActiveX 属性页的 COM 工厂生效。

属性页是一个 COM 对象，表示模块必须在提供了一个工厂的情况下才能被执行，以便系统能够创建对象。通过对 TPropertyPage 再细分类，ActiveX 属性页被实现，TActiveXPropertyPageFactory 对象是 ActiveX 属性页的工厂。

关于 COM 工厂的更多信息，参见 TComObjectFactory。为向一个程序中加入一个属性页工厂，必须在定义属性页窗体单元的初始化部分创建工厂对象。

方法列表

Create	创建属性页工厂

方 法

TActiveXPropertyPageFactory.Create:

创建属性页工厂，并将其与属性页类联编。

constructor Create(ComServer: TComServerObject; ActiveXControlClass: TActiveXControlClass; WinControlClass: TWinControlClass; const ClassID: TGUID; ToolboxBitmapID: Integer);

ComServer 参数是关于 ActiveX 库的 COM 服务器对象的一个指针。通常为全局变量，在 ComServ 单元中进行声明。

PropertyPageClass 参数是属性页的实现类的类对象。实现类必须来自于窗体 TPropertyPage。该类使出现在属性对话框中的窗体生效。

ClassID 参数是一个识别属性页的类的 GUID。每一个属性页都有其自身的 GUID。

参 见 TComObjectFactory.Create。

TAdapterNotifier 单元 Axctrls

类关系：TObject→TInterfacedObject

在 Delphi 类对象与可比较的 OLE 类对象之间，TAdapterNotifier 对象提供联编。

TCustomAdapter 的子类使用 TAdapterNotifier 对象，以便将 Delphi 对象与相应的 OLE 对象进行联编。例如，TFontAdapter 使用 TAdapterNotifier 对象，将 TFont 对象与 OLE 字体对象进行联编。

方法列表

Create	创建实例

方 法

TAdapterNotifier.Create

创建 TAdapterNotifier 对象的实例。

constructor Create(Adapter: TCustomAdapter);

不要用该类创建一个对象。TAdapterNotifier 对象由 TCustomAdapter 的子类创建，并与之相绑定。

参 见 TObject.Create。

TAggregatedObject 单元 Axctrls

类关系：TObject

通过执行 IUnknown 方法，TAggregatedObject 对象为集合的内部对象提供了控制 IUnknown 方法的机能。

集合的 COM 对象是一个由数个 COM 对象组成的对象。每一个对象均有其自身的行为和接口，但所有对象共享同样的引用问题——控制器对象。在容器模式中，控制器即是容器对象。

TAggregatedObject 对象自身不能支持任何接口。但作为集合的象征，它可以执行由它包含对象使用的 Iunknown 方法。因此，TAggregatedObject 对象可作为使创建对象接口生效的类的基使用。

在 Delphi 类阶层结构中，TAggregatedObject 对象作为创建包含对象并连接对象类的基使用。使用 TAggregatedObject 对象，可以确保对 IUnknown 方法的调用委派于集合的控制 Iunknown。

控制 Iunknown 由 TAggregatedObject 对象的构造函数指定，并由 Controller 属性表示。

属性列表

Controller	提供对集合的 IUnknown 方法的访问

方法列表

Create	实例化集合对象

属 性

TAggregatedObject.Controller

Controller 属性提供对集合的 IUnknown 方法的访问。

property Controller: IUnknown;

Controller 属性是集合的 IUnknown 接口参考。TAggregatedObject 对象被当作是具有单一控制 IUnknown 的集合的一部分使用。在集合对象中执行 IUnknown 方法被授权于 Controller 属性。

当集合对象被创建时，在构造函数中 Controller 属性被初始化为 IUnknown。

参 见 TAggregatedObject.Create。

方 法

TAggregatedObject.Create

Create 构造函数用于实例化集合对象。

constructor Create(Controller: IUnknown);

在运行时调用 Create 构造函数，可以实例化集合对象。Create 构造函数为集合对象分配内存，并通过 Controller 参数设置 Controller 属性。Controller 属性是集合的控制 IUnknown。集合中所有其他接口的实现将委派于 IUnknown 方法对 Controller 属性的调用。

参 见 TAggregatedObject.Controller。

TConnectionPoint 单元 Axctrls

类关系：TObject→TAggregatedObject→TContainedObject

TConnectionPoint 是用于为连接点容器创建连接点对象的类。

TConnectionPoint 对象是一个使 IConnectionPoint 接口实现的类。它用于为可连接对象创建连接点对象。可连接对象具有下列特征：

- 具有输出接口，如事件集合。
- 具有计算输出接口 IIDs 的能力。
- 具有为那些输出接口 IIDs 连接对象和取消与对象的连接。
- 具有计算存在与一个特定输出接口连接。

IConnectionPoint 接口和 IConnectionPointContainer 接口用于 OLE 事件处理。因为 Delphi ActiveX 控件向导为一个控件自动地产生事件，因此，当用户希望修改标准 VCL 实现或者向一个非 ActiveX 控件自动服务程序中加入事件时，用户仅需直接使用这些接口。IConnectionPoint 接口的实现方法：GetConnectionInterface, GetConnectionPointContainer, Advise, Unadvise, EnumConnections。

方法列表

Create	实例化一个连接点对象
Destroy	删除一个连接点对象的实例

方 法

TConnectionPoint.Create

实例化一个连接点对象。

```
type
TConnectionKind[ ] (dkSingle, dkMulti);
TConnectEvent [ ] procedure ( const Sink: Iunknown;
Connecting: Boolean) of object;
```

constructor Create(Container: TConnectionPoints; const IID: TGUID; OnConnect: TConnectEvent);

Create 由连接点容器的 CreateConnectionPoint 方法自动调用，以实例化一个连接点对象。当一个 ActiveX 控件被初始化时，自动调用 Create 构造函数。Create 为一个连接点对象分配内存并调用继承的 Create，传递 Container 参数的 Iunknown 控制。Container 参数为一个连接点容器对象。Create 向 Container 参数中加入连接点对象，然后设置：

- IID，作为由该连接点管理的 GUID 或者输出接口的接口标识符。
- IKind，表示对那个点是一个单个连接还是多个连接。
- OnConnect，作为在 IConnectionPoint Advise 方法中使用的连接事件。

TConnectEvent 参数是指向连接点 OnConnect 事件的一个过程的指针。TConnectionKind 参数是表示连接点是否有多个连接的一个类型。

参 见 TActiveXControl. Initialize, TAggregatedObject. Create, TConnectionPoint. Destroy, TConnectionPoints. CreateConnectionPoint。

TConnectionPoint.Destroy

删除一个连接点对象的实例。

destructor Destroy; override;

连接点对象被自动删除。不需要删除连接点对象。如果连接点容器对象不是 nil，则 Destroy 访问该对象，以从中删除连接点对象。注意：需要处理一个对象删除的应用程序，不直接调用 Destroy。相反地，首先调用 Free，Free 检查到该对象还没有被释放后，才调用 Destroy。

参 见 TConnectionPoints。

TConnectionPoints 单元 Axctrls

类关系：TObject→TAggregatedObject

TConnectionPoints 使连接点对象的一个容器生效。

TConnectionPoints 对象允许一个对象表示输出接口的存在。在一个可连接对象内，IConnectionPointContainer 接口被实现，表示输出接口存在。它提供对一个 IEnumConnectionPoints 接口计数器子对象进行访问的功能，也提供对实现 IConnectionPoint 接口的所有连接点子对象进行访问。IConnectionPoint 接口提供对一个 IEnumConnectionPoints 接口计数器子对象进行访问的功能。

TConnectionPoints 对象用于对下列对象进行访问：

- 一个 IEnumConnectionPoints 接口计数器子对象。IEnumConnectionPoints 接口能够计算每一个输出 IID 连接点。
- 每个输出 IID 的 IConnectionPoint 接口连接点子对象。通过 IConnectionPoint 接口，一个客户可以启动或终止一个可连接对象的咨询循环和客户的拥有汇点。客户使用 IConnectionPoint 接口，也能够获取一个 IEnumConnections 接口计数器对象，以计算它所知道的连接。

IConnectionPoint 接口和 IConnectionPointContainer 接口用于 OLE 事件处理。因为 Delphi ActiveX 控件向导为一个控件自动地产生事件，因此，当用户希望修改标准 VCL 实现或者向一个非 ActiveX 控件自动服务程序中加入事件时，用户仅需直接使用这些接口。

实现接口：IConnectionPointContainer。

方法列表

Create	实例化一个连接多点对象
CreateConnectionPoint	实例化一个连接点对象
Destroy	删除一个连接点对象的实例

方 法

TConnectionPoints.Create

实例化一个连接多点对象。

```
constructor Create(const Controller: IUnknown);
```

它被自动调用，以实例化一个连接多点对象。Create 构造函数被其他对象内部使用，以创建连接点对象。

Create 构造函数为一个连接点对象分配内存，并调用继承的 Create 构造函数，传递其 Controller 参数作为聚集中未知的控制，然后为其将包含的单个连接点对象创建一个列表。

参 见 TAggregatedObject.Controller, TAggregatedObject.Create。

TConnectionPoints.CreateConnectionPoint

实例化一个连接点对象。

```
type
TConnectionKind = (dkSingle, dkMulti);
TConnectEvent = procedure(const Sink: Iunknown;
Connecting: Boolean) of object;
function CreateConnectionPoint(const IID: TGUID;
Kind: TConnectionKind; OnConnec
TConnectEvent): ConnectionPoint;
```

当一个 ActiveX 控件被初始化时，CreateConnectionPoint 函数被自动调用。CreateConnectionPoint 函数创建并返回一个连接点对象。它向连接点构造函数中传递下列参数：

- CreateConnectionPoint 函数自身，作为连接点容器。
- IID，作为该连接点管理的 GUID 或者输出接口的接口标识符。
- OnConnect，作为在 IConnectionPoint Advise 方法中使用的连接事件。

TConnectEvent 参数是一个指向正被创建的连接点对象 OnConnect 事件过程一个的指针。TConnectionKind 参数是表示对那个连接点是否有多个连接的一个类型。

参 见 TActiveXControl.Initialize, TConnectionPoint.Create。

TConnectionPoints.Destroy

删除一个连接点对象的实例。

```
destructor Destroy; override;
```

Destroy 被自动调用。用户不需要删除连接点对象。Destroy 释放在构造函数中创建的连接点对象的列表。注意：需要处理一个对象删除的应用程序，不直接调用 Destroy。相反地，首先调用 Free，Free 检查到该对象还没有被释放后，才调用 Destroy。

TContainedObject 单元 Axctrls

类关系：TObject→TAggregatedObject

作为一个集合内的一个内部对象，TContainedObject 对象实现 IUnknown 接口。

按照适用于能够被集合对象（内部的或被包含的对象）的规则，TContainedObject 对象被实现。TContainedObject 对象能够作为一个基类使用，该基类用于创建内部的或被包含的一个集合对象。使用 TComObject 对象或其子类，可以例示支持集合的一个外部对象。

下面为 TContainedObject 对象支持的可集合（内部的或被包含的）对象的实现规则：

- 当创建该类型的一个对象时，仅对一个请求的 IUnknown 类型的接口进行创建，才能成功。
- IUnknown 方法 AddRef 和 Release 委派于控制 IUnknown。
- QueryInterface 被实现，在 TContainedObject 对象中，仅对在被包含对象中实现的接口。
- 在包含对象中的内部对象的引用计数增加。

在 Delphi 类体系中，TContainedObject 对象被作为一个集合中一个内部对象的一个连接点对象的基类使用。

注意：一个集合的 COM 对象，是由数个 COM 对象组成的一个对象。每一个对象实现其拥有的行为和接口，但所有对象共享相同的控制器对象的引用计数。在容器模板中，控制器是容器对象。

TCustomAdapter 单元 Axctrls

类关系：TObject→TInterfacedObject

TCustomAdapter 是将 VCL 对象绑定到 OLE 接口的对象的一个基类。

不要建立该类对象。调用 GetOleFont，建立一个 TFont 对象与一个 OLE 字体对象之间的连接，或者调用 GetOlePicture，建立一个 TPicture 对象与一个 OLE 图片对象之间的连接。

方法列表

Changed	当 OLE 对象的 VCL 表现变化时，响应 Changed
ConnectOleObject	使适配器与一个 OLE 接口关联
Create	构建定制适配器控件的一个实例
Destroy	删除定制适配器对象
ReleaseOleObject	释放关联的 OLE 接口，允许减少其引用数目
Update	为响应 OLE 对象修改的一个方法提供接口

方 法

TCustomAdapter.Changed

当 OLE 对象的 VCL 表现变化时，响应 Changed。

procedure Changed; virtual;

Changed 被自动调用，以便关联的 OLE 对象能够被更新，以反映对相应 VCL 对象所做出的修改。在 TCustomAdapter 对象中，Changed 调用 ReleaseOleObject，以便能够再次获得 OLE 对象，以反映在 VCL 对象中的修改。TCustomAdapter 的子类重设 Changed，使用 OLE 接口做出适当的修改。

参 见 TCustomAdapter.ReleaseOleObject。

TCustomAdapter.ConnectOleObject

使适配器与一个 OLE 接口关联。

procedure ConnectOleObject(OleObject: IUnknown);

ConnectOleObject 被内部使用，以使适配器与一个 OLE 接口关联。

参 见 TCustomAdapter.ReleaseOleObject。

TCustomAdapter.Create

创建定制适配器控件的一个实例。

constructor Create;

永远不能建立该类的一个对象，仅能建立其子类对象，如 TFontAdapter 或 TPictureAdapter 对象。

参 见 TCustomAdapter.Destroy。

TCustomAdapter.Destroy

删除定制适配器对象。

destructor Destroy; override;

不要直接调用 Destroy，而应调用 Free。Free 检查到适配器对象还没有被释放后，才调用 Destroy。Destroy 释放并取消 Delphi 对象与一个 OLE 对象的连接。

参 见 TCustomAdapter.Create, TObject.Free。

TCustomAdapter.ReleaseOleObject

释放关联的 OLE 接口，允许减少其引用数目。

procedure ReleaseOleObject;

实现 TCustomAdapter 的一个子类时，当适配器不再需要 OLE 对象时调用 ReleaseOleObject。

参 见 TCustomAdapter.ConnectOleObject。

TCustomAdapter.Update

为响应 OLE 对象修改的一个方法提供接口。

procedure Update; virtual; abstract;

TCustomAdapter 的子类重设 Update，以修改关联的 VCL 对象，以便其能够反映在相应的 OLE 对象中所做出的修改。它是 Change 方法的逆方法。

参 见 TCustomAdapter.Changed。

TFontAdapter　　单元 Axctrls

类关系：TObject→TInterfacedObject→TCustomAdapter

TFontAdapter 类将连接一个 TFont 对象和一个 OLE 字体对象。

如果想创建一个 TFontAdapter 实例，可以调用全局 GetOleFont 过程。该过程将在一个 TFont 对象和一个 OLE 字体对象之间创建联系。

方法列表

Changed	更新相联系的 IFont 接口
Create	创建并实例化一个字体适配器控件
GetOleFont	从插入的 OLE 对象中获得 IFontDisp 接口
SetOleFont	将适配器与一个 OLE 对象联系起来
Update	更新 TFont 对象

方 法

TFontAdapter.Changed

使用字体对象的新属性值更新相联系的 IFont 接口。

procedure Changed; override;

方法 Changed 将在 TFont 对象改变后自动调用。它确认与该对象相联系的 OLE 接口是否被更新。

参 见 TFontAdapter.SetOleFont, TFontAdapter.Update。

TFontAdapter.Create

创建并实例化一个字体适配器控件。

Constructor Create(Font: Tfont);

一个 TFont 对象和一个 OLE 字体对象之间创建联系。

参 见 GetOleFont 过程，TFont。

TFontAdapter.GetOleFont

从相联系的 OLE 对象中获得 IFontDisp 接口。

procedure TFontAdapter.GetOleFont(var OleFont: IFontDisp);

使用 TFontAdapter 对象的 ActiveX 控件，将调用 GetOleFont 过程，来获得相应于适配器的 TFont 对象的接口。如果字体的 OLE 对象不存在，则 GetOleFont 过程将创建一个。

IFontDisp 接口将作为 OleFont 参数返回。

参 见 TCustomAdapter.ConnectOleObject, TFontAdapter.SetOleFont。

TFontAdapter.SetOleFont

将适配器与一个指定接口表示的 OLE 对象联系起来。

procedure TFontAdapter.SetOleFont(const OleFont: IFontDisp);

使用 TFontAdapter 对象的 ActiveX 控件，将调用 SetOleFont 过程，把适配器与参数 OleFont 指定的接口联系起来。

参 见 TFontAdapter.GetOleFont。

TFontAdapter.Update

更新 TFont 对象，来反映与该对象相联系的 OLE 字体的变化。

procedure TFontAdapter.Update;

在相联系的 OLE 对象变化时，将自动调用 Update 方法。它将确认相对应的 TFont 对象能反映所出现的变化。

参 见 SetOleFont 过程。

TOleGraphic 单元 Axctrls

类关系：TObject→TPersistent→TGraphic

TOleGraphic 是一个 OLE 绘图接口的包装。

TOleGraphic 是一个在 ActiveX 控件中使用的图标，是图元文件和位图的绘图容器。TOleGraphic 供内部使用，用于给 ActiveX 控件 TPicture 属性图像提供一个 IPicture 接口。因为 TOLEGraphic 是 OLE 接口的包装，因此它不能被用来装载或保存文件。

属性列表

Empty	OLE 图片对象中是否包含有图片
Height	以像素点为单位描述 OLE 图片的高度
MMHeight	以 0.01 毫米为单位描述 OLE 图片的高度
MMWidth	描述 OLE 图片的宽度
Palette	描述 OLE 图片的颜色
Picture	指定 OLE 图片的引用接口
Transparent	描述 OLE 图片是否充满其四边形区域
Width	描述图片的宽度

方法列表

Assign	将 IPicture 拷贝到 OLE 图片对象
LoadFromClipboard-Format	从剪贴板将 OLE 图片装载到 OLE 图片对象
LoadFromFile	防止 OleGraphics 从文件装载
LoadFromStream	将 IPicture 接口从流装入 OLE 图片对象
SaveToClipboardFormat	以 Clipboard 格式保存 OLE 图片
SaveToStream	将 OLE 图片保存到流中

属 性

TOleGraphic.Empty

说明 OLE 图片对象中是否包含有图片。

property Empty: Boolean;

Empty 描述图片是否已装载到 OLE 图片对象中。如果 OLE 图片对象中包含有图片，则 Empty 为 True，否则为 False 。

TOleGraphic.Height

以像素点为单位，描述 OLE 图片的高度。

property Height: Integer;

Height 表示 OLE 图片以像素点为单位的高度。

TOleGraphic.MMHeight

描述 OLE 图片的高度，以 0.01 毫米为单位。

property MMHeight: Integer;

由 MMHeight 可以更精确地得到 OLE 图片的高度。与以屏幕像素为单位的 Height 属性相比，由 MMHeight 得到的值避免了像素点和毫米之间转化的误差。

参 见 TOleGraphic.Height, TOleGraphic.MMWidth。

TOleGraphic.MMWidth

以 0.01 毫米为单位，描述 OLE 图片的宽度。

property MMWidth: Integer;

由 MMWidth 可以更精确地得到 OLE 图片宽度。与以屏幕像素为单位的 Width 属性相比，由 MMWidth 得到的值避免了像素点和毫米之间转化的误差。

参 见 TOleGraphic.MMHeight, TOleGraphic.Width。

TOleGraphic.Palette

描述 OLE 图片的颜色。

property Palette: HPALETTE;

OLE 图片的 Palette 包含有 256 种颜色，用于在屏幕上显示图片。

TOleGraphic.Picture

指定 OLE 图片的引用接口。

property Picture: IPicture;

Picture 只可以内部使用。

TOleGraphic.Transparent

描述 OLE 图片是否充满其四边形区域。

property Transparent: Boolean;

Transparent 描述了 OLE 图片是否充满整个四边形区域，如果 Transparent 为 True，则图片并未充满整个区域，如果 Transparent 为 False，则图片完全充满整个区域。

TOleGraphic.Width

以像素为单位，描述图片的宽度。

property Width: Integer;

通过 Width 可以得到 OLE 图片的宽度。注意：Width 假定屏幕分辨率，因此其数学转化并不精确。

参 见 TOleGraphic.MMWidth。

方 法

TOleGraphic.Assign

将 IPicture 拷贝到 OLE 图片对象。

procedure Assign(Source: TPersistent); override;

Assign 将 Source 中的 IPicture 拷贝到 OLE 图片对象中，IPicture 可以是位图，元文件或图标图像。

参 见 TPersistent.Assign。

TOleGraphic.LoadFromClipboardFormat

从剪贴板将 OLE 图片装载到 OLE 图片对象。

procedure LoadFromClipboardFormat(AFormat: Word; AData: THandle; APalette: HPALETTE); override;

如果已通过 RegisterClipboardFormat 方法，在 TPicture