Borland C++ Builder 3.0/4.0 类参考详解

张济 袁野 王秀娟 编著

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

Borland C++ Builder 3.0/4.0 类参考详解

张 济 袁 野 王秀娟 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是 Borland C++ Builder 3.0/4.0 的常用类参考手册。其中详尽地介绍了 Borland C++ Builder 3.0/4.0的常用类及类中的属性、方法和事件;还详细介绍了函数、过程以及变量、类型、常量等。

本书内容翔实,与《Borland C++ Bulider 3.0/4.0 高级类参考详解》手册相辅相承,遥相呼应,适合 Borland C++ Builder 开发人员和广大计算机专业人员使用。

图书在版编目(CIP)数据

Borland C++ Builder 3.0/4.0 类参考详解/张济等编著.-北京:清华大学出版社,1999

ISBN 7-302-03774-4

Ⅰ.B… Ⅱ.张… Ⅲ.C 语言,C++ Builder 3.0/4.0-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字 (1999) 第 62020 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者: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**:1014 千字
版 次:1999 年 11 月第 1 版 1999 年 11 月第 1 次印刷
书 号:ISBN 7-302-03774-4/TP·2119
印 数:0001~5000
定 价:46.00 元

前　言

C ++ Builder 3.0/4.0 是 Borland 公司继 Borland C ++ 5.0 之后的又一力作，是新一代面向对象、可视化的快速应用程序开发环境。它工作在 Windows 95/98 操作系统上。使用 C ++ Builder 可以开发通用的或基于客户/服务器模式的 32 位 Windows 应用程序。它比 Borland C ++ 5.0 开发效率更高、功能更强大、使用也更加方便。因为 C ++ Builder 环境与 Visual Basic 或 Delphi 相类似，编程和使用相当方便，而且不影响 C ++ 功能强大的特点。可以说 C ++ Builder 3.0/4.0 是 Visual Basic 外观和 C ++ 内涵相结合的一个优秀产物。

使用传统的基于C ++ 语言的工具软件（如 Visual C ++，Bortland C ++，Watcom C ++/C）进行 Windows 应用程序编程时，要求对 Windows 操作系统和对使用的编程工具的类库（如 MicroSoftR MFC，Borland 公司的 OWL 类库）有较深入的研究。掌握和使用这些类型工具软件难度大且应用程序的开发效率低，不适应当前快速应用程序的开发需求。就在这种背景下，快速应用程序开发工具应运而生，并取得快速进展，当前有代表性的有：MicrosSoft 的 VisualBasic，Borland 的 Delphi，PowerSoft 公司的 PowerBuilder 语言。这些应用程序开发工具通过使用预制的构件，接近口语化的编程语言和可视化的编程界面大大简化了 Windows 应用程序的开发过程，大大提高了开发的效率，满足当前应用构件，使用这些预制的构件可以非常方便地开发 Windows 应用程序。

Borland C ++ Builder 从 Delphi 开发工具继承可视化构件库，又从 Borland C ++ 开发工具继承语言，成为快速应用程序开发模式和可重用构件的一个完美结合，代表着 C ++ 语言的演化和发展方向。

遵循实用、便捷的原则，为使不同层次的读者分门别类地学习、掌握全方位的知识，不仅对 C ++ Builder 3.0/4.0 的 VCL（Visual Component Library，可视组件库）库进行了全面、系统的介绍；而且将其以常用类和高级类为区别分编为两本手册，即《Borland C ++ Builder 3.0/4.0 类参考详解》和《Borland C ++ Builder 3.0/4.0 高级类参考详解》。

本书为《Borland C ++ Builder 3.0/4.0 类参考详解》手册，其以翔实、简练的语言，针对初学者详细地介绍了常用类及类中的属性，方法和事件。其结构层次清晰，在介绍类的基本概念的基础上，介绍了它的继承和层次，说明它的使用方法。帮助读者全面掌握 VCL 组件的属性、方法和事件，为用户掌握大量的 C ++ Builder 类提供方便、快速、实用的途径。

在本书的完成过程中，得到了许多相关人士的大力协助，他们对本书的出版给予了热情的支持，谨在此表示衷心的感谢。

由于作者水平有限，经验不多，再加上时间仓促，书中肯定会有不少缺点和错误，希望得到专家和读者的指正。

作　者

目 录

单元 Axctrls

TActiveForm 单元 Axctrls

TActiveForm 是用作为 ActiveX 控件的 VCL 窗体的基类。

TActiveForm 表示一个通过 ATL (Active Template Library)创建的窗体，在 ActiveX 主应用程序中，作为一个 ActiveX 控件使用。

类关系 TObject→TPersistent→TComponent→TControl→TWinControl → TScrollingWinControl → TCustomForm → TCustomAvtiveForm

TActiveForm 用于创建 ActiveX 控件窗体，从 Web 浏览器上可以进行显示和运行。

属性列表

AxBorderStyle	设置活动窗体的边框风格
~TActiveForm	释放与 TActiveForm 对象有关的内存
TActiveForm	创建 TActiveForm 的一个实例

属 性

TActiveForm::AxBorderStyle

设置活动窗体的边框风格。

enum TActiveFormBorderStyle { afbNone, afbSingle, afbSunken, afbRaised };

__property TActiveFormBorderStyle AxBorderStyle = {read = FAxBorderStyle, write = SetAxBorderStyle, default = 1};

AxBorderStyle 可取下列值之一：

取值	含义
afbNone	活动窗体的周围没有边框
afbSingle	活动窗体的周围有单细线边框
afbSunken	活动窗体有凹下边框
afbRaised	活动窗体有凸起边框但没有文字

参 见 TCustomForm::BorderStyle。

方 法

TActiveForm::~TActiveForm

~TActiveForm 释放与 TActiveForm 对象有关的内存。

__fastcall ~TActiveForm(void);

不要直接调用~TActiveForm。用 delete 会自动引起~TActiveForm 运行的。

TActiveForm::TActiveForm

创建 TActiveForm 的一个实例.

__fastcall virtual TActiveForm(HWND ParentWindow): Forms::TCustomActiveForm(ParentWindow) {}

__fastcall virtual TActiveForm(TComponent * AOwner): Forms::TCustomActiveForm(AOwner) {}

要在运行阶段创建一个活动窗体，用 new 来间接调用 TActiveForm。

把一个组件作为参数传递，给 TActiveForm 对象提供一个负责释放它的拥有者。这是最常用的构造函数语法。把一个窗口句柄作为参数传递，就把活动窗体嵌入到非 VCL 窗口。这种语法用在把 ActiveX 控制嵌入到非 VCL 窗口。

TActiveForm 调用并重设父类的构造函数，来初始化 AxBorderStyle 属性。

参 见 TActiveFormFactory。

TActiveFormControl 单元 Axctrls

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

在 C++Builder 窗体和 ActiveX 控件 API 之间，TActiveFormControl 对象起桥梁作用。

方法列表

~TActiveFormControl	~TActiveFormControl 释放与 TActiveFormControl 对象有关的内存
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
TActiveFormControl	TActiveFormControl 创建一个新的 TActiveFormControl 对象

方 法

TActiveFormControl::~TActiveFormControl

~TActiveFormControl 释放与 TActiveFormControl 对象有关的内存。

不要直接调用~TActiveFormControl。用 delete 会自动使~TActiveFormControl 运行。

TActiveFormControl::FreeOnRelease

FreeOnRelease 过程用于支持 IVCLComObject 接口。

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。

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。

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 参数各自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)并返回对象的请求的 COM 接口。

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。

TActiveFormControl::TActiveFormControl

TActiveFormControl 创建一个新的 TActiveFormControl 对象。

TActiveFormFactory 单元 Axctrls

ActiveFormFactory 对象是 TActiveForm 的类工厂。

ActiveFormFactory 对象,是负责创建 TActiveFormFactory 对象和 TActiveForm 对象实例的类工厂。

方法列表

~TActiveFormFactory	~TActiveFormFactory 释放与 TActiveFormFactory 对象有关的内存
GetIntfEntry	获取调度接口的指针
TActiveFormFactory	TActiveFormFactory 创建一个新的 TActiveFormFactory 对象

方 法

TActiveFormFactory::~TActiveFormFactory

~TActiveFormFactory 释放与 TActiveFormFactory 对象有关的内存。

不要直接调用~TActiveFormFactory。用 delete 会自动使~TActiveFormFactory 运行的。

TActiveFormFactory::GetIntfEntry

GetIntfEntry 方法用于获取调度接口的指针。

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。

TActiveFormFactory::TActiveFormFactory

TActiveFormFactory 创建一个新的 TActiveFormFactory 对象。

TActiveXControl 单元 Axctrls

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 控件链接起来。它允许控件可嵌入到任何一个 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在线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。
- 住留。
- 事件。
- 属性页。
- 属性浏览。

TActiveXControl 对象是一个抽象类,因此不能进行示例。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
- IOleInPlaceActiveObject
- IPersistStorage
- IViewObject
- IOleObject
- IViewObject2
- IOleControl
- IPerPropertyBrowsing
- IOleInPlaceObject
- ISpecifyPropertyPages

注意:参见:::\ActiveX\Delctrls 中的 DelCtrls 例子,它示例了 C++Builder 组件用作 ActiveX 控制。

属性列表

Control	表示 ActiveX 控件的 VCL 实现

方法列表

~TActiveXControl	删除 TActiveXControl 对象的实例
DefinePropertyPages	当控制将要显示其属性页时调用该方法
EventSinkChanged	当控制容器把一个新事件沉没接口，连到对象的事件链接点时调用该方法
GetPropertyString	当 Object Inspector 想获得用户可读字符串来代表属性的当前值时调用该方法。该字符串显示在 Object Inspector 中相邻于属性名处
GetPropertyStrings	当 Object Inspector 希望获得一个可供用户从中选择的属性值列表时调用该方法
GetPropertyValue	当用户从 Object Inspector 中的列表中选了一个属性值时，就调用该方法来把列表选择转化为一个 Variant 值
Initialize	为 TActiveXControl 对象创建类工厂和链接对象
InitializeControl	在控制和 VCL 控制被完全创建后，调用该方法
LoadFromStream	从流中调出 ActiveX 控制的状态
ObjQueryInterface	返回关于对象请求的 COM 接口
PerformVerb	执行定制的 verb 时调用该方法
PropChanged	产生一个 OnChanged 事件
PropRequestEdit	表明能否对指定的属性进行修改
SaveToStream	把 ActiveX 控制的状态保存到提供的流中。该方法的默认实现把 VCL 控制的所有存储属性保存到流中
TActiveXControl	TActiveXControl 创建一个新的 TActiveXControl 对象
WndProc	挂接了 VCL 执行控制的窗口过程

属　性

TActiveXControl::Control

Control 属性表示 ActiveX 控件的 VCL 实现。

__property Controls::TWinControl* Control = {read=FControl};

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::~TActiveXControl

~TActiveXControl 用于删除 TActiveXControl 对象的实例。

__fastcall virtual ~TActiveXControl(void);

不要在应用程序中直接调用~TActiveXControl，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~TActiveXControl。~TActiveXControl 释放把 Control 变成 ActiveX 控件使用的资源。它不删除由 Control 属性引用的对象。

参　见　TActiveXControl::Control, TObject::Free。

TActiveXControl::DefinePropertyPages

当控制将要显示其属性页时调用该方法。

virtual void __fastcall DefinePropertyPages(TDefinePropertyPage &DefinePropertyPage);

如果用户想显示属性页，就重设该方法为每个要显示的属性页调用一次 DefinePropertyPage 回调函数。如果不重设这个方法，就不显示属性页。

DefinePropertyPage 有一个参数，它是要加入属性对话框的属性页对象的类 id。

一个 ActiveX 属性页对话框是一个用来编辑一个或多个 ActiveX 对象的标签对话框。每个属性页作为一个 COM 对象执行，可在相同库中作为控制或在单个注册库中实现。

默认时专家为 ActiveX 控制生成的代码重设该方法，该方法中带有一个备注，其中包含了如果使用 DefinePropertyPage 的例子。

参　见　TPropertyPage。

TActiveXControl::EventSinkChanged

当控制容器把一个新事件沉没接口连到对象的事件链接点时调用该方法。

virtual void __fastcall EventSinkChanged(const __di_IUnknown EventSink);

如果用户想让其对象烧调不是标准事件的事件，就重设该方法保存沉没的私有版。参数 EventSink 就是容器的事件沉没接口。为了把事件烧到容器，该接口可被转换为 IDispatch 或任意的 dispinterface。

当容器希望改变或删除事件沉没接口时也调用该方法。如果 EventSink 为 NULL，事件沉没正被删除或临时关闭，因为容器请求控制不烧调事件。ActiveX 控制专家生成的默认代码将把沉没赋给控制的出界接口类型变量。

参　见　TConnectinPoint。

TActiveXControl::GetPropertyString

当 Object Inspector 想获得用户可读字符串来代表属性的当前值时调用该方法。该字符串显示在 Object Inspector 中相邻于属性名处。

virtual bool __fastcall GetPropertyString(int DispID, System::AnsiString &S);

重设该方法让用户以可读形式而非本地格式来显示一个属性的当前值。如，字体属性可显示为“Arial, 12pt”。

ActiveX 容器重复调用该方法，Object Inspector 的每个属性都调用一次。

参数 DispID 指明了请求的属性。如果想使该属性显示一个特殊字符串，就把参数 S 设为字符串值，该方法返回 True。

返回 False 指明不需要显示特殊字符串的值。这种情

况下，Object Inspector 将以标准方式显示属性值。

ActiveX 控制专家生成的默认实现将自动在类型库中查找来处理枚举类型。

参 见 TActiveXControl::GetPropertyStrings, TActiveXControl::GetPropertyValue。

TActiveXControl::GetPropertyStrings

当 Object Inspector 希望获得一个可供用户从中选择的属性值列表时调用该方法。

virtual bool __fastcall GetPropertyStrings(int DispID, Classes::TStrings * Strings);

当用户单击 Object Inspector 中 property 的下拉箭头但在显示列表之前，为确定放置在列表中的值，Object Inspector 一般要调用这个方法。Object Inspector 在 property 值下面的下拉列表中显示这些值。

参数 DispID 指明了字符串值正被请求的属性值。如果希望 Object Inspector 显示属性的下拉列表，重设该方法，调用 Strings::AddObject 方法把项目加在列表上。

在 Strings::AddObject 的字符串参数中，提供显示在下拉列表中的文字。

在 Strings::AddObject 中的参数 object，提供唯一的"cookie value"，该值可被传回 GetPropertyValue 方法，以标识哪个项被用户选中。尽管可以为 cookie 值选任意的唯一值，通常是要使用一个整数值（0 为列表第一项，1 为列表第二项，依此类推）。

返回 False 表明 DispID 指定的属性不显示下拉列表。

返回 True 表明应该显示下拉列表，列表中项目已被加到 Strings 对象。

该方法与 GetPropertyValue 方法组合使用，在用户选了列表中一项后，把所选的字符串翻译为 Variant 值。如果执行属性的这个方法，也应该执行 GetPropertyValue。

ActiveX 控制专家生成的默认实现将自动查找类型库来处理枚举类型。

参 见 TActiveXControl::GetPropertyValue, TStrings::AddObject。

TActiveXControl::GetPropertyValue

当用户从 Object Inspector 的列表中选了一个属性值时，就调用该方法来把列表选择转化为一个 Variant 值。

virtual void __fastcall GetPropertyValue(int DispID, int Cookie, System::OleVariant &Value);

当用户执行 GetPropertyStrings 方法来定义一个列表时，就应重设这个方法.

参数 DispID 标识了正被修改的属性。参数 Cookie 标识了被用户选择的项目。

实现该方法时，应返回一个对应于参数 Value 中的被选项目的 Variant 值。

参 见 TActiveXControl::GetPropertyString, TActiveXControl::GetPropertyStrings。

TActiveXControl::Initialize

为 TActiveXControl 对象创建类工厂和链接对象。

virtual void __fastcall Initialize(void);

Initialize 方法创建类工厂对象和创造，并封装 VCL 控件。

参 见 TActiveFormControl::InitializeControl。

TActiveXControl::InitializeControl

在控制和 VCL 控制被完全创建后，调用该方法。

virtual void __fastcall InitializeControl(void);

重设该方法可在 VCL 控制被创建后，进一步初始化一个 ActiveX 对象。特别地，应该在 InitializeControl 中把 OLE event _ firing 方法绑接到 VCL 控制的事件属性中。

参 见 TActiveFormControl::Control。

TActiveXControl::LoadFromStream

从流中调出 ActiveX 控制的状态。

virtual void __fastcall LoadFromStream(const _di_IStream Stream);

这个方法的默认实现从从流中调出 VCL 控制的存储属性。

可以重设这个方法来恢复被 SaveToStream 方法保存的其他数据。要检索额外数据，调用 Stream::Read 方法。调出的数据必须匹配以 SaveToStream 方法保存的数据。

如果重设该方法，一定要调用继承方法来恢复 VCL 对象的属性，除非特别地希望从 Stream 中阻止恢复控制属性。

如果希望使用一个 C++Builder 流，可以使用 TOleStream，它在 C++Builder 流中封装了一个 Ole 流。

参 见 TActiveFormControl::SaveToStream, TOleStream, TStream::Read, TVclComControl::IPersistentInit _ Load。

TActiveXControl::ObjQueryInterface

ObjQueryInterface 函数返回关于对象请求的 COM 接口。

virtual HRESULT __stdcall ObjQueryInterface(const GUID &IID, void * Obj);

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::PerformVerb

执行定制的 verb 时调用该方法。

virtual void __fastcall PerformVerb(int Verb);

当利用 TActiveXControlFactory::AddVerb 方法把 verbs 加入 ActiveX 对象工厂时，重设这个方法。

参数 Verb 指明应该执行哪一个 verb。该号匹配工厂的 AddVerb 方法提供标识。

参 见 TActiveXControlFactory::AddVerb。

TActiveXControl::PropChanged

产生一个 OnChanged 事件。

void __fastcall PropChanged(const System::WideString PropertyName);

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。

bool __fastcall PropRequestEdit(const System::WideString PropertyName);

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。当可以对属性进行修改时，IpropertyNotifySink 函数返回 True。

TActiveXControl::SaveToStream

把 ActiveX 控制的状态保存到提供的流中。该方法的默认实现把 VCL 控制的所有存储属性保存到流中。

virtual void __fastcall SaveToStream(const _di_IStream Stream);

可以重设该方法来保存控制要求的其他数据。要保存额外数据，调用 Stream::Write 方法。如果要保存其他数据，重设 LoadFromStream 保证数据被正确恢复。

如果重设这个方法，保证调用继承方法来恢复对象的属性，除非特殊地不保存它们。

参 见 TActiveXControl::LoadFromStream, TStream::Write, TVclComControl::IPersistentStreamInit_Load。

TActiveXControl::TActiveXControl

TActiveXControl 创建一个新的 TActiveXControl 对象。

__fastcall TActiveXControl(void) : Comobj::TAutoObject()

__fastcall TActiveXControl(_di_IUnknown const Controller) : Comobj::TAutoObject(Controller)

__fastcall TActiveXControl(Comobj::TComObjectFactory* Factory, _di_IUnknown const Controller) : Comobj::TAutoObject(Factory, Controller) { }

TActiveXControl::WndProc

挂接了 VCL 执行控制的窗口过程。

virtual void __fastcall WndProc(Messages::TMessage &Message);

一般没必要重设或调用这个方法。该方法被声称为受保护虚方法，允许 C++Builder ActiveX 框架的进一步扩展。

该方法一般把消息传递给控制的原消息句柄，除非 ActiveX 容器指导 ActiveX 对象从处理流中滤出该消息。

参 见 TControl::WndProc。

TActiveXControlFactory 单元 Axctrls

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。在定义 TActiveXControl 对象的单元的 initialization 部分，TActiveXControl 向导加入一个 ActiveX 控件工厂对象。

属性列表

EventIID	控件的事件接口的 GUID
EventTypeInfo	对象的事件接口的类型信息对象
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

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

属 性

TActiveXControlFactory::EventIID

EventIID 是控制的事件接口的 GUID。

__property GUID EventIID = {read=FEventIID};

EventIID 被内部用来创建控制与事件接口之间的链接。

参 见 TActiveXControlFactory::EventTypeInfo。

TActiveXControlFactory::EventTypeInfo

EventTypeInfo 是对象事件接口的类型信息对象。

__property _di_ITypeInfo EventTypeInfo = {read=FEventTypeInfo};

利用这个信息获得对象事件接口的类型信息。

参 见 TActiveXControlFactory::EventIID。

TActiveXControlFactory::MiscStatus

MiscStatus 属性是储存在登记名册中的状态为集合。

__property int MiscStatus = {read=FMiscStatus, nodefault};

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 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

状态位	支持容器所需	备注
IGNOREACTIVATEWHENVISIBLE	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的激活能力),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置属性
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 int ToolboxBitmapID = {read = FToolboxBitmapID, nodefault};

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::TActiveXControlFactory。

TActiveXControlFactory::WinControlClass

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

__property System::TMetaClass* WinControlClass = {read = FWinControlClass};

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::~TActiveXControlFactory

删除 TActiveXControlFactory 的实例。

__fastcall virtual ~TActiveXControlFactory(void);

在应用程序中,不要直接调用 ~TActiveXControlFactory。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 ~TActiveXControlFactory。

参 见 TObject::Free。

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。

void __fastcall AddVerb(int Verb, const System::AnsiString VerbName);

使用 AddVerb 过程加入一个主程序能够对 ActiveX 控件使用动词。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单击鼠标右键时,动词文本显示在一个快捷菜单上。

在字符串中的一个字符前放置一个"&"号,在快捷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::TActiveXControlFactory

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

__fastcall TActiveXControlFactory(Comobj::TComServerObject* ComServer, System::TMetaClass* ActiveXControlClass, System::TMetaClass* WinControlClass, con-

st GUID &ClassID, int ToolboxBitmapID, const System::AnsiStringLicStr, int MiscStatus);

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::UpdateRegistry

UpdateRegistry 过程用于为对象增加或删除登记项目。

virtual void __fastcall UpdateRegistry(bool Register);

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

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 C++Builder 窗体和 ActiveX 属性页 API 之间起桥梁作用。

方法列表

~TActiveXPropertyPage	删除 TActiveXPropertyPage 的实例
Initialize	对子类的实例进行初始化
TActiveXPropertyPage	TActiveXPropertyPage 创建一个新的 TActiveXPropertyPage 对象

方 法

TActiveXPropertyPage::~TActiveXPropertyPage

删除 TActiveXPropertyPage 的实例。

不要在应用程序中直接调用 ~TActiveXPropertyPage，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 ~TActiveXPropertyPage。

参 见 TObject::Free。

TActiveXPropertyPage::Initialize

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

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::TComObject。

TActiveXPropertyPage::TActiveXPropertyPage

TActiveXPropertyPage 创建一个新的 TActiveXPropertyPage 对象。

TActiveXPropertyPageFactory 单元 Axctrls

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

~TactiveXProperty-PageFactory	释放与 TActiveXPropertyPage-Factory 对象有关的内存
TActiveXProperty-PageFactory	创建属性页工厂，并将其与属性页类联编

方 法

TActiveXPropertyPageFactory::~TActiveXPropertyPage-Factory

~TActiveXPropertyPageFactory 释放与 TActiveXPropertyPageFactory 对象有关的内存。

不要直接调用 ~TActiveXPropertyPageFactory。用 delete 会自动引起 ~TActiveXPropertyPageFactory 运行的。

TActiveXPropertyPageFactory::TActiveXPropertyPage-Factory

创建属性页工厂，并将其与属性页类联编。

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::TComObjectFactory。

TAdapterNotifier 单元 Axctrls

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

类关系 TObject→TInterfacedObject

TCustomAdapter 的子类使用 TAdapterNotifier 对象，以便将 C++ Builder 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

~TAdapterNotifier	~TAdapterNotifier 释放与 TAdapterNotifier 对象有关的内存
TAdapterNotifier	创建 TAdapterNotifier 对象的实例

参 见 TCustomAdapter, TFontAdapter, TPictureAdapter。

方 法

TAdapterNotifier::~TAdapterNotifier

~TAdapterNotifier 释放与 TAdapterNotifier 对象有关的内存。

```
__fastcall virtual ~TAdapterNotifier(void) { }
```

不要直接调用~TAdapterNotifier。用 delete 会自动使得~TAdapterNotifier 运行。

TAdapterNotifier::TAdapterNotifier

创建 TAdapterNotifier 对象的实例。

```
__fastcall TAdapterNotifier(TCustomAdapter * Adapter);
```

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::TObject。

TAggregatedObject 单元 Axctrls

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 C++ Builder 类阶层结构中，TAggregatedObject 对象作为创建包含对象并链接对象类的基使用。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

~TAggregatedObject	~TAggregatedObject 释放与 TAggregatedObject 对象有关的内存
TAggregatedObject	构造函数用于实例化集合对象

参 见 TConnectionPoint, TConnectionPoints, TContainedObject。

属 性

TAggregatedObject::Controller

Controller 属性提供对集合的 IUnknown 方法的访问。

```
__property __di_IUnknown Controller = {read=GetController};
```

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::TAggregatedObject。

方 法

TAggregatedObject::~TAggregatedObject

~TAggregatedObject 释放与 TAggregatedObject 对象有关的内存。

```
__fastcall virtual ~TAggregatedObject(void) { }
```

不要直接调用~TAggregatedObject。用 delete 会自动使得~TAggregatedObject 运行。

参 见 AnsiString::Delete, TAggregatedObject::TAggregatedObject, TBaseArray::Delete, TBookmarkList::Delete, TCusotmImageList::Delete, TCusotmListView::Delete, TCusotmTreeView::Delete, TDataSet::Delete, TList::Delete, TListItem::Delete, TListItems::Delete, TMenuItem::Delete, TStringGridStrings::Delete, TStringList::Delete, TStrings::Delete, TTreeNode::Delete, TTreeNodes::Delete。

TAggregatedObject::TAggregatedObject

构造函数用于实例化集合对象。

```
__fastcall TAggregatedObject(__di_IUnknown Controller);
```

在运行时调用 TAggregatedObject 构造函数可以实例化集合对象。

TAggregatedObject 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

TConnectionPoint 是用于为链接点容器创建链接点对象的类。

TConnectionPoint 对象是一个使 IConnectionPoint 接口实现的类。它用于为可链接对象创建链接点对象。可链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 IIDs 链接对象和取消与对象的链接。
- 具有计算存在与一个特定输出接口链接。

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，用户仅需要直接使用这些接口。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

~TConnectionPoint	删除一个链接点对象的实例
TConnectionPoint	实例化一个链接点对象

参 见 TConnectionPoints。

方 法

TConnectionPoint::~TConnectionPoint

删除一个链接点对象的实例。

__fastcall virtual ~TConnectionPoint(void);

链接点对象被自动删除。不需要直接去删除链接点对象。如果链接点容器对象不是 NULL，则 ~TConnectionPoint 访问该对象，以从中删除链接点对象。注意：需要处理一个对象删除的应用程序，不直接调用 ~TConnectionPoint。相反地，它们调用 delete。

参 见 TConnectionPoints。

TConnectionPoint::TConnectionPoint

实例化一个链接点对象。

__fastcall TConnectionPoint(TConnectionPoints * Container, const GUID &IID, TConnectionKind Kind, TConnectEvent&OnConnect);

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

- IID，作为由该链接点管理的 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链接还是多个链接。
- OnConnect，作为在 IConnectionPoint Advise 方法中使用的链接事件。

TConnectEvent 参数是指向链接点 OnConnect 事件的一个过程的指针。TConnectionKind 参数是表示链接点是否有多个链接的一个类型。

参 见 TActiveXControl::Initialize, TConnectionPoint::~TConnectionPoints, TConnectionPoints::TConnectionPoints。

TConnectionPoints 单元 Axctrls

TConnectionPoints 使链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。在一个可链接对象内，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。它也提供对实现 IConnectionPoint 接口的所有链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。

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 链接点。
- 每个输出 IID 的 IConnectionPoint 接口链接点子对象。通过 IConnectionPoint 接口，一个客户可以启动或终止一个可链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，以计算它所知道的链接。

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，用户仅需要直接使用这些接口。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

~TConnectionPoints	实例化一个链接多点对象
CreateConnectionPoint	实例化一个链接点对象
TConnectionPoints	删除一个链接点对象的实例

方 法

TConnectionPoints::~TConnectionPoints

删除一个链接点对象的实例。

__fastcall virtual ~TConnectionPoints(void);

~TConnectionPoints 被自动调用。用户不需要删除链