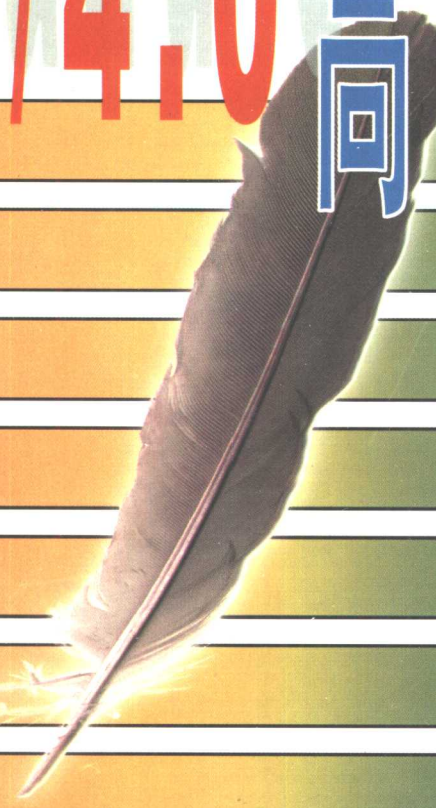
Borland C++ Builder 3.0/4.0 高级类参考详解



王秀娟 孙悦 黎华 编著

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

Borland C++ Builder 3.0/4.0 高级类参考详解

王秀娟 孙 悦 黎 华 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 的高级类参考手册。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 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 高级类参考详解/王秀娟等编著.-北京:清华大学出版社,1999

ISBN 7-302-03793-0

Ⅰ.B… Ⅱ.王… 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 63370 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**:25.5 **字 数**:992 千字
版 次:1999 年 11 月第 1 版 1999 年 11 月第 1 次印刷
书 号:ISBN 7-302-03793-0/TP·2134
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 的环境与 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++, WatcomC++/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++ Builder3.0/4.0 高级类参考详解》。

本书为《Borland C++ Builder 3.0/4.0 高级类参考详解》手册，其以翔实、简练的语言，针对熟练掌握 C++ Builder 3.0/4.0 的读者，详细地介绍了高级类及类中的属性，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 Atlvcl.h

IDataBrokerImpl 单元 Atlvcl.h

执行 IDataBroker 接口，它公布了驻留在 TDataModule 对象中的数据集。

属性列表

m _ DataModule	包含一个 TDataModule 对象

方法列表

~IDataBrokerImpl	释放与 IDataBrokerImpl 对象有关的内存
GetProviderNames	返回远程数据模块上一个可用的提供商列表
IDataBrokerImpl	创建一个 IDataBrokerImpl 对象

参 见 TDataModule。

属 性

IDataBrokerImpl::m _ DataModule

包含一个 TDataModule 对象。

```
DM * m _ DataModule;
```

要 m _ DataModule 的对象在 IDataBrokerImpl 构造函数中创建。

参 见 IDataBrokerImpl::IDataBrokerImpl。

方 法

IDataBrokerImpl::~IDataBrokerImpl

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

```
~IDataBrokerImpl();
```

不要直接调用~IDataBrokerImpl。用 delete 替代，它会自动调用~IDataBrokerImpl。

~IDataBrokerImpl 调用 Bdeprov::UseBdeProv 函数并释放与 m _ DataModule 属性相关的内存。

参 见 IDataBrokerImpl::m _ DataModule。

IDataBrokerImpl.GetProviderNames

返回远程数据模块上一个可用的提供商列表。

存储 Result 中被公开的数据集的 m _ DataModule 的名字。

```
HRESULT STDMETHODCALLTYPE GetProviderNames(System::OleVariant &Result);
```

一旦成功，GetProviderNames 返回 S _ OK；否则，GetProviderNames 就返回 GetTypeInfo 的返回值之一。

IDataBrokerImpl::IDataBrokerImpl

创建一个 IDataBrokerImpl 对象。

```
IDataBrokerImpl();
```

创建一个 TDataModule 对象，将其赋给 m _ DataModule 属性。

参 见 IDataBrokerImpl::m _ DataModule, IDataBrokerImpl::IDataBrokerImpl。

ISimpleFrameSiteImpl 单元 Atlvcl.h

执行一个 ISimpleFrameSite 对象。

关于 ISimpleFrameSite 的信息，参见 Microsoft Active Template Library (ATL)的第三部分文档。

方法列表

PostMessageFilter	执行后信息过滤器
PreMessageFilter	执行前信息过滤器
QueryInterface	执行 QueryInterface

方 法

ISimpleFrameSiteImpl::PostMessageFilter

执行后信息过滤器。

```
STDMETHOD(PostMessageFilter)(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp, LRESULT * plResult, DWORD dwCookie);
```

ISimpleFrameSiteImpl::PreMessageFilter

执行前信息过滤器。

```
STDMETHOD(PreMessageFilter)(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp, LRESULT * plResult, DWORD * pdwCookie);
```

ISimpleFrameSiteImpl::QueryInterface

执行 QueryInterface。

```
STDMETHOD(QueryInterface)(REFIID riid, void * * ppvObject) = 0;
```

TAxControlRegistrar 单元 Atlvcl.h

注册一个没有注册的 ActiveX 控制。

典型地，这个类的实例是由 DECLARE _ ACTIVEXCONTROL _ REGISTRY 宏创建的。

属性列表

m _ BitmapID	包含 ToolboxBitmap32 资源的索引
m _ MiscFlags	包含控制的二义性信息
m _ Verbs	包含控制支持的 OLE verbs

方法列表

TAxControlRegistrar	创建一个 TAxControlRegistrar 对象
UpdateRegistry	利用 TAxControlRegistrar 对象中包含的数据注册控制，包括其类型库；会反注册该控制

属 性

TAxControlRegistrar::m _ BitmapID

包含 ToolboxBitmap32 资源的索引。

```
int m _ BitmapID;
```

TAxControlRegistrar::m_MiscFlags

包含控制的二义性信息。

DWORD m_MiscFlags;

当创建 TAxControlRegistrar 对象时，m_MiscFlags 被初始化为 dwDefaultControlMiscFlags 常数。

参 见 dwDefaultControlMiscFlags 常数。

TAxControlRegistrar::m_Verbs

包含控制支持的 OLE verbs。

const OLEVERB * m_Verbs;

方 法

TAxControlRegistrar::TAxControlRegistrar

创建一个 TAxControlRegistrar 对象.

TAxControlRegistrar(): m_BitmapID(0), m_MiscFlags(dwDefaultControlMiscFlags), m_Verbs(0) {}

TAxControlRegistrar(const CLSID& clsid, AnsiString ProgID, int BitmapID, DWORD MiscFlags = dwDefaultControlMiscFlags, const OLEVERB * verbs = 0) : m_BitmapID(BitmapID), m_MiscFlags(MiscFlags), m_Verbs(verbs), TTypedComServerRegistrar(clsid, ProgID) {}

Initializes 语法中的数据成员。

不要直接调用 TAxControlRegistrar；用 new 会使得 TAxControlRegistrar 方法自动运行。

TAxControlRegistrar::UpdateRegistry

利用 TAxControlRegistrar 对象中包含的数据注册控制，包括其类型库，会反注册该控制。

virtual HRESULT UpdateRegistry(bool Register);

Register 指定是否注册或反注册 COM 服务器。

TComModule　　单元 Atlvcl.h

扩展 TComModule，可允许它在进程内及进程外和 VCL 应用内工作。

关于 TComModule 对象的信息，参见 CBuilder3 \ Include \ Atl 目录中的头文件以及 Microsoft Active Template Library (ATL)的第三部分文档。

属性列表

m_bAutomationServer	指定模块是否为一个自动化服务器
m_bExe	指定模块是否为本地（EXE）或进程内部（DLL)服务器
m_bRun	指定服务器是否运行
m_InitOle	包含初始化 OLE 的数据
m_InitProc	包含 VCL 的初始化过程
m_ThreadID	模块的线程标识符

方法列表

~TComModule	删除 COM 对象的一个实例
DoFileAndObjectRegistration	根据命令行开关和实例设置来注册服务器
TComModule	创建一个新的 TComModule 对象
Unlock	如果模块代表一个本地服务器，发送 WM_QUIT 消息

TComModule::m_bAutomationServer

指定模块是否为一个自动化服务器。

bool m_bAutomationServer;

在 TComServerRegistrar::UpdateRegistry 方法中，被用来确定模块是否为一个自动化服务器。

参 见 TComServer::UpdateRegistry。

TComModule::m_bExe

指定模块是否为本地（EXE）或进程内部（DLL)服务器。

bool m_bExe;

m_bExe 在 Unlock，TComServerRegistrar::Init 以及 TComServerRegistrar::UpdateRegistry 方法中，用来确定模块是否为本地（EXE）或进程内部（DLL)服务器。

参 见 TComModule::~TComModule, TComModule::Unlock, TComServerRegistrar::Init, TComServerRegistrar::UpdateRegistry。

TComModule::m_bRun

指定服务器是否运行。

bool m_bRun;

m_bRun 在~TComModule 方法中，被用来确定某对象的内存是否需要释放。

参 见 TComModule::~TComModule。

TComModule::m_InitOle

包含初始化 OLE 的数据。

TInitOle m_InitOle;

m_InitOle 在 TComModule 构造函数中，被用来确认 OLE 被适当初始化。

参 见 TComModule::TComModule。

TComModule::m_InitProc

包含 VCL 的初始化过程。

TProcedure m_InitProc;

m_InitProc 与 InitProcedure 一样，它被作为 TComModule 构造函数的参数传递。它被赋给 System::InitProc。System::InitProc 的旧值被保存在 SaveInitProc 变量中。

参 见 TComModule::TComModule, SaveInitProc 变量。

TComModule:: m_ThreadID

模块的线程标识符。

DWORD m_ThreadID;

线程标识符在 Unlock 方法中，被用来确定往哪里发送 WM_QUIT 消息。

参 见 TComModule::Unlock。

方 法

TComModule:: ~TCom Module

删除 COM 对象的一个实例。

__fastcall virtual ~TComModule(void);

释放与 TComModule 对象有关的内存。不要直接调用～TComModule,用 delete 替代,它会自动调用～TComModule。

参 见 TComModule::TComModule。

TComModule::DoFileAndObjectRegistration

根据命令行开关和实例设置来注册服务器。

void DoFileAndObjectRegistration()

根据/automation,/embedding,/regserver 以及/unregserver 命令行开关来注册服务器。也注册单用和多用服务器。

TComModule::TComModule

创建一个新的 TComModule 对象。

TComModule(): m_ThreadID(0), m_bRun(true), m_bExe(false), m_InitProc(0), m_bAutomationServer(false) {}

TComModule(TProcedure InitProcedure): m_ThreadID(0), m_bRun(true), m_bExe(true), m_InitProc(InitProcedure), m_bAutomationServer(false);

初始化函数原型中的数据成员。如果指定了 InitProcedure,TComModule 就完成下列任务:

- 确认 OLE 被适当地初始化。
- 把 SaveInitProc 变量设置为 System::InitProc(如果 m_bExe 和 m_InitProc 为 true)。
- 把 System::InitProc 设置为 InitProcedure(如果 m_bExe 和 m_InitProc 为 true)。

TComModule::Unlock

如果模块代表一个本地服务器,发送 WM_QUIT 消息。

LONG TComModule::Unlock();

调用 CComModule::Unlock。如果 CComModule::Unlock 返回 0 以及 m_bExe 为 true,Unlock 把 WM_QUIT 消息,发送给 m_ThreadID 数据成员指定的线程。

参 见 TComModule::～TComModule。

TComServerRegistrar 单元 Atlvcl.h

注册或反注册一个 COM 服务器,不包括其类型库。

典型地,这个类的一个实例是用 DECLARE_COMSERVER_REGISTRY 宏创建的。

属性列表

m_ClassID	包含 COM 服务器的 ClassID
m_ClassKey	包含 COM 服务器的 ClassKey
m_Description	包含 COM 服务器的 description
m_ModuleName	包含 COM 服务器的模块名
m_ProgID	包含 COM 服务器的 ProgID
m_ServerType	包含 COM 服务器的类型

方法列表

～TComServerRegistrar	释放与 TComServerRegistrar 对象有关的内存
CreateRegKey	创建一个注册键
DeleteRegKey	删除一个注册键
Init	初始化与 TComServerRegistrar 对象有关的数据
NukeRegKey	删除一个注册键及其子键
TComServerRegistrar	TComServerObject 创建一个新的 TComServerObject 对象
UpdateRegistry	注册或反注册 COM 服务器

参 见 DECLAER_COMSERVER_REGISTRY 宏,TTypedComServerRegistrar。

属 性

TComServerRegistrar::m_ClassID

包含 COM 服务器的 ClassID。

CLSID m_ClassID;

TComServerRegistrar::m_ClassKey

包含 COM 服务器的 ClassKey。

CLSID m_ClassID;

当 TComServerRegistrar 对象被创建时,m_ClassKey 被自动初始化。m_ClassKey 可用 Init 方法重新初始化。

参 见 TComServerRegistrar::Init, TComServerRegistrar::TComServerRegistrar。

TComServerRegistrar::m_Description

包含 COM 服务器的 description。

AnsiString m_Description;

TComServerRegistrar::m_ModuleName

包含 COM 服务器的模块名。

WideString m_ModuleName。

当 TComServerRegistrar 对象被创建时,m_ModuleName 被自动初始化。m_ModuleName 可用 Init 方法重新初始化。

参 见 TComServerRegistrar::Init, TComServerRegistrar::TComServerRegistrar。

TComServerRegistrar::m_ProgID

包含 COM 服务器的 ProgID。

AnsiString m_ProgID;

TComServerRegistrar::m_ServerType

包含 COM 服务器的类型。

AnsiString m_ServerType;

TComServerRegistrar 对象被创建时,m_ServerType 被自动初始化。m_ServerType 可用 Init 方法重新初始化。

参 见 TComServerRegistrar::Init, TComServerRegistrar::TComServerRegistrar。

方 法

TComServerRegistrar::～TComServerRegistrar

释放与 TComServerRegistrar 对象有关的内存。

～TComServerRegistrar() {}

不要直接调用～TComServerRegistrar。用 delete 替代,它会自动调用～TComServerRegistrar。

TComServerRegistrar::CreateRegKey

创建一个注册键。

static void CreateRegKey(AnsiString Key, AnsiString ValueName, AnsiString Value);

Key 是要加入新值的键名。

ValueName 是与新值有关的名字。

Value 为新值。

参 见 TComServerRegistrar::DeleteRegKey, TComServerRegistrar::NukeRegKey。

TComServerRegistrar::DeleteRegKey

删除一个注册键。

static void DeleteRegKey(AnsiString Key);

Key 是要删除的键名。

在 Windows95, DeleteRegKey 也删除子键。在 Windows NT, DeleteRegKey 只对没有子键的键起作用。

参 见 TComServerRegistrar::CreateRegKey, TComServerRegistrar::NukeRegKey。

TComServerRegistrar::Init

初始化与 TComServerRegistrar 对象有关的数据。

void Init();

初始化 m_ModuleName, m_ClassKey, 以及 m_ServerType 属性。

Init 由 TComServerRegistrar 方法调用。

参 见 TComServerRegistrar::m_ClassKey, TComServerRegistrar::m_ModuleName, TComServerRegistrar::m_ServerType, TComServerRegistrar::TComServerRegistrar。

TComServerRegistrar::NukeRegKey

删除一个注册键及其子键。

static void NukeRegKey(AnsiString Key);

Key 是要删除的键名。Key 的子键也被删除。

参 见 TComServerRegistrar::DeleteRegKey。

TComServerObject::TComServerObject

TComServerObject 创建一个新的 TComServerObject 对象。

TComServerRegistrar(const CLSID& clsid, AnsiString progID, AnsiString description) : m_ClassID(clsid), m_ProgID(progID), m_Description(description) { Init(); }

不要直接调用 TComServerRegistrar。用 new 替代,它会自动调用 TComServerRegistrar 方法。TComServerRegistrar 调用 Init 方法。

参 见 TComServerRegistrar::~TComServerRegistrar, TComServerRegistrar::Init。

TComServerRegistrar::UpdateRegistry

利用 TComServerRegistrar 对象中包含的数据注册 COM 服务器或反注册 COM 服务器。

virtual HRESULT UpdateRegistry(bool Register);

Register 指定是否注册或反注册 COM 服务器。

TLicenseString 单元 Atlvcl::h

检查 ActiveX 控制是否在当前机器上被许可的对象模板。

template <class T>

T 为 ActiveX 控制类。

对于要求许可检查的 ActiveX 控制,ActiveX 向导为 TLicenseString 对象生成代码,并被传递给 DECLARE_CLASSFACTORY2 宏。

关于 DECLARE_CLASSFACTORY2 宏的信息,参见 CBuilder3\Include\Atl 目录的头文件,以及 Microsoft Active Template Library (ATL)中的第三部分文档。

TLicenseString 对象的方法仅调用 ActiveX 控制类的相应方法。利用 Use the methods in the ActiveX 控制类的方法,可重设 ActiveX 控制许可检查的默认实现。

方法列表

GetLicenseKey	检索 ActiveX 控制的许可钥匙
IsLicenseValid	允许 ActiveX 控制可确定许可是否有效
VerifyLicenseKey	把一个字符串与 ActiveX 控制的许可钥匙进行比较

参 见 TValidateLicences。

方 法

TLicenseString::GetLicenseKey

检索 ActiveX 控制的许可钥匙。

static BOOL GetLicenseKey(DWORD /* dwReserved */, BSTR *pStr)

DWORD 参数保留为以后使用。

pStr 为保存钥匙的对象。

GetLicenseKey 总是返回 TRUE。

参 见 TLicenseString::IsLicenceValid, TLicenseString::VerifyLicenseKey。

TLicenseString::IsLicenseValid

允许 ActiveX 控制可确定许可是否有效。

static BOOL IsLicenseValid();

调用 ActiveX 控制的 IsLicenseValid 方法,返回结果。

参 见 TLicenseString::GetLicenseKey, TLicenseString::VerifyLicenseKey。

TLicenseString::VerifyLicenseKey

把一个字符串与 ActiveX 控制的许可钥匙进行比较。

static BOOL VerifyLicenseKey(BSTR str);

如果 str 匹配 ActiveX 控制的许可钥匙,就返回 true;否则就返回 otherwise。

参 见 TLicenseString::GetLicenseKey, TLicenseString::VerifyLicenseKey。

TRemoteDataModuleRegistrar 单元 Atlvcl.h

登记或反登记远程数据模块,包括其类型库。

典型地,这个类的实例是由 DECLARE _ REMOTEDATAMODULE _ REGISTRY 宏创建的。

方法列表

~TComServerRegistrar	释放与 TComServerRegistrar 对象有关的内存
CreateRegKey	创建一个登记键
DeleteRegKey	删除一个登记键
Init	初始化与 TComServerRegistrar 对象有关的数据
NukeRegKey	删除一个登记键及其子键
TComServerRegistrar	创建一个 TComServerRegistrar 对象
UpdateRegistry	利用 TComServerRegistrar 对象中包含的数据,登记 COM 服务器;或反登记 COM 服务器

参 见 DECLARE _ COMSERVER _ REGISTRY 宏, TTypedComServerRegistrar。

方 法

TComServerRegistrar::~TComServerRegistrar

释放与 TComServerRegistrar 对象有关的内存。

~TComServerRegistrar() {}

不要直接调用~TComServerRegistrar。用 delete 替代,它会自动调用~TComServerRegistrar 方法。

参 见 TComServerRegistrar::TComServerRegistrar。

TComServerRegistrar::CreateRegKey

创建一个登记键。

static void CreateRegKey(AnsiString Key, AnsiString ValueName, AnsiString Value);

Key 为要加入新值的键名。ValueName 为与新值有关的名字。Value 为新值。

参 见 TComServerRegistrar:: DeleteRegKey, TComServerRegistrar::NukeRegKey。

TComServerRegistrar::DeleteRegKey

删除一个登记键。

static void DeleteRegKey(AnsiString Key);

Key 为要删除的键名。在 Windows 95 下,DeleteRegKey 也删除子键。在 Windows NT 下,DeleteRegKey 只对没有子键的键起作用。

参 见 TComServerRegistrar:: CreateRegKey, TComServerRegistrar::NukeRegKey。

TComServerRegistrar::Init

初始化与 TComServerRegistrar 对象有关的数据。

void Init();

初始化 m _ ModuleName, m _ ClassKey, 以及 m _ ServerType 属性。

Init 是从 TComServerRegistrar 方法中调用的。

参 见 TComServerRegistrar::m _ ClassKey, TComServerRegistrar:: m _ ModuleName, TComServerRegistrar:: m _ ServerType, TComServerRegistrar:: TComServerRegistrar。

TComServerRegistrar::NukeRegKey

删除一个登记键及其子键。

static void NukeRegKey(AnsiString Key);

Key 为要删除的键名。其子键也被删除。

参 见 TComServerRegistrar::DeleteRegKey。

TComServerRegistrar:: TComServerRegistrar

创建一个 TComServerRegistrar 对象。

TComServerRegistrar(const CLSID& clsid, AnsiString progID, AnsiString description) : m _ ClassID(clsid), m _ ProgID(progID), m _ Description(description) { Init(); }

不要直接调用 TComServerRegistrar。用 new 替代,它会自动调用 TComServerRegistrar 方法。

TComServerRegistrar 调用 Init 方法。

参 见 TComServerRegistrar:: ~ TComServerRegistrar, TComServerRegistrar::Init。

TComServerRegistrar::UpdateRegistry

利用 TComServerRegistrar 对象中包含的数据登记 COM 服务器;或反登记 COM 服务器。

virtual HRESULT UpdateRegistry(bool Register);

Register 指定是否登记或反登记 COM 服务器。

TTypedComServerRegistrar 单元 Atlvcl.h

注册或反注册一个 COM 服务器,包括其类型库。

典型地,该类的实例由 DECLARE _ TYPED _ COMSERVER _ REGISTRY 宏创建。

方法列表

TTypedComServerRegistrar	创建一个 TTypedComServerRegistrar 对象
UpdateRegistry	注册或反注册 COM 服务器

参 见 DECLARE _ TYPED _ COMSERVER _ REGISTRY 宏。

方 法

TTypedComServerRegistrar::TTypedComServerRegistrar

创建一个 TTypedComServerRegistrar 对象。

TTypedComServerRegistrar(): TComServerRegistrar() {}

TTypedComServerRegistrar(const CLSID& clsid, AnsiString ProgID): TComServerRegistrar(clsid, ProgID, NULL) {}

不要直接调用 TTypedComServerRegistrar。用 new 进行,它会自动调用 TTypedComServerRegistrar。

TTypedComServerRegistrar::UpdateRegistry

注册 COM 服务器,包括其类型信息,使用 TTypedCom-

ServerRegistrar 对象中包含的数据，或反注册 COM 服务器。

HRESULT UpdateRegistry(bool Register);

Register 指明是否注册或反注册 COM 服务器。

TValidateLicense 单元 Atlvcl.h

包含了一个方法，IsGUIDInFile 的对象，它确认一个 license 字符串 GUID 是否在一个 license (.LIC)文件中。

对于要求许可确认的 ActiveX 控制，ActiveX 控制向导利用 TValidateLicense::IsGUIDInFile，提供了 ActiveX 控制的 IsLicenseValid 方法的默认实现。

关于 ActiveX 控制的许可的详细信息，参见 Microsoft Active Template Library (ATL)的第三部分文档。

方法列表

IsGUIDInFile	如果指定的 GUID 在指定的许可文件中，就返回 true

方 法

TValidateLicense::IsGUIDInFile

如果指定的 GUID 在指定的许可文件中，就返回 true。

static BOOL IsGUIDInFile(const WCHAR * szGUID, const TCHAR * szLicFileName);

szGUID 是一个 GUIDl。

szLicFileName 为许可文件的名字。

IsGUIDInFile 会从前往后查找下列目录的许可文件：

- 包含 ActiveX 控制的目录。
- 默认目录。
- 容器目录。
- Windows 目录。
- 系统路径目录。

TVclComControl 单元 Atlvcl.h

从一个 VCL 控制中创建一个 ActiveX 控制的模板。

template <class T, class TVCL>

TVclComControl 被用在 ActiveX 向导生成的代码中。T 为被创建的 ActiveX 控制。TVCL 为 ActiveX 控制被从中创建的 VCL 控制。

属性列表

m_AmbientDriver	包含容器的外界属性
m_hWnd	包含 ActiveX 控制的窗口句柄
m_VclCtl	包含 VCL 控制的包装体
m_VclWinCtl	包含反映 VCL 控制的消息的窗口的包装体

方法列表

~TVclComControl	释放与 ~TVclComControl 对象有关的内存
ControlQueryInterface	检索被请求接口(iid)的指针(ppv)
ControlWndProc	就如 ActiveX 消息一样处理 ActiveX 控制的 Windows 消息
Create	创建 ActiveX 控制的窗口，返回窗口句柄
CreateControlWindow	调用 Create 方法
FireOnChanged	通知所有的被连接的 IPropertyNotifySink 接口，指定的控制属性将要改变
FireOnRequestEdit	通知 IPropertyNotifySink 接口，控制属性将要改变
Initialize	设置 ActiveX 控制的客户场所
InitializeControl	从一个流中把 VCL 组件调入 m_VclCtl 属性
IOleObject_SetClientSite	设置 ActiveX 控制的客户场所
IPersistStreamInit_Load	从一个流中把 VCL 组件调入 m_VclCtl 属性
IPersistStreamInit_Save	把 m_VclCtl 属性中的 VCL 组件保存到一个流中
ISimpleFrameSite_PostMessageFilter	处理单帧场所的后消息过滤
ISimpleFrameSite_PreMessageFilter	处理单帧场所的前消息过滤
RecreateWnd	重新创建 ActiveX 控制的窗口
ShowWindow	设置窗口的 show 状态
TVclComControl	创建一个 TVclComControl 对象
WndProc	处理 ActiveX 控制的 Windows 消息

参 见 TVclComControlImpl。

属 性

TVclComControl::m_AmbientDriver

包含容器的外界属性。

TAutoDriver<IDispatch> m_AmbientDriver;

m_AmbientDriver 被在 IOleObject_SetClientSite 方法中初始化。m_AmbientDriver 在 OnAmbientPropertyChange 方法中定义。

TVclComControl::m_hWnd

包含 ActiveX 控制的窗口句柄。

HWND m_hWnd;

TVclComControl::m_VclCtl

包含 VCL 控制的包装体。

TWinControlAccess<TVCL> * m_VclCtl;

TVclComControl::m_VclWinCtl

包含反映 VCL 控制的消息的窗口的包装体。

TWinControlAccess<TWinControl> * m_VclWinCtl;

参 见 TVclComControl::m_VclCtl。

方 法

TVclComControl::~TVclComControl

释放与～TVclComControl 对象有关的内存。

～TVclComControl(void);

不要直接调用～TVclComControl，利用 delete 进行，它会自动调用～TVclComControl。

参 见 TVclComControl::TVclComControl。

TVclComControl::ControlQueryInterface

检索被请求接口(iid)的指针 (ppv)。

virtual HRESULT ControlQueryInterface(const IID& iid, void * * **ppv);**

被请求接口必须被列在 COM 映射表中。

TVclComControl::ControlWndProc

就如 ActiveX 消息一样处理 ActiveX 控制的 Windows 消息。

virtual void ControlWndProc(Messages::TMessage& Message);

ControlWndProc 被从 WndProc 方法中调用。

参 见 TVclComControl::WndProc。

TVclComControl::Create

创建 ActiveX 控制的窗口，返回窗口句柄。

HWND Create(HWND hWndParent, RECT& rcPos);

Create 完成下列任务：

- 把 m_VclWinCtl->ParentWindow 初始化为 hWndParent。
- 把 m_VclWinCtl->BoundsRect 初始化为 rcPos。
- 把 m_hWnd 初始化为 mVclWinCtl->GetWindowHandle()。
- 强迫窗口被创建和显示。

参 见 TVclComControl::CreateControlWindow。

TVclComControl::CreateControlWindow

调用 Create 方法。

HWND CreateControlWindow(HWND hWndParent, RECT& rcPos);

参 见 TVclComControl::Create。

TVclComControl::FireOnChanged

通知所有的被连接的 IPropertyNotifySink 接口，指定的控制属性将要改变。

HRESULT FireOnChanged(DISPID dispID);

调用 CFirePropNotifyEvent::FireOnChanged，返回结果。如果控制不是从 IPropertyNotifySink 派生的，就返回 S_OK。

参 见 TVclComControl::FireOnRequestEdit

TVclComControl::FireOnRequestEdit

通知所有的被连接的 IPropertyNotifySink 接口，指定的控制属性将要改变。

HRESULT FireOnRequestEdit(DISPID dispID);

调用 CFirePropNotifyEvent::FireOnRequestEdit，返回结果。如果控制不是从 IPropertyNotifySink 派生的，就返回 S_OK。

参 见 TVclComControl::FireOnChanged

TVclComControl::IOleObject_SetClientSite

设置 ActiveX 控制的客户场所。

HRESULT IOleObject_SetClientSite(IOleClientSite * pClientSite);

IOleObject_SetClientSite 完成下列任务：

- 调用 CComModule::IOleObject_SetClientSite。
- 确认 ActiveX 控制被正确登记。
- 初始化 m_AmbientDriver 属性。
- 使 Ctl3D 风格生效。

TVclComControl::IPersistStreamInit_Load

从一个流中把 VCL 组件调入 m_VclCtl 属性。

HRESULT IPersistStreamInit_Load(LPSTREAM pStm, ATL_PROPMAP_ENTRY * pMap);

IPersistStreamInit_Load 利用 LoadVCLControlFromStream 函数。

参 见 TVclComControl::IPersistStreamInit_Save。

TVclComControl::IPersistStreamInit_Save

把 m_VclCtl 属性中的 VCL 组件保存到一个流中。

HRESULT IPersistStreamInit_Save(LPSTREAM pStm, BOOL /* fClearDirty */, ATL_PROPMAP_ENTRY * pMap);

IPersistStreamIni_Save 利用 SaveVCLControlToStream 函数。

参 见 TVclComControl::IPersistStreamInit_Load。

TVclComControl::ISimpleFrameSite_PostMessageFilter

处理单帧场所的后消息过滤。

HRESULT ISimpleFrameSite_PostMessageFilter(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp, LRESULT * plResult, DWORD dwCookie);

SimpleFrameSite_PostMessageFilter 是从 ISimpleFrameSite::PostMessageFilter 中被调用。

TVclComControl::ISimpleFrameSite_PreMessageFilter

处理单帧场所的前消息过滤。

HRESULT ISimpleFrameSite_PreMessageFilter(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp, LRESULT * plResult, DWORD * pdwCookie)

ISimpleFrameSite_PreMessageFilter 是从 ISimpleFrameSite::PreMessageFilter 中被调用。

TVclComControl::RecreateWnd

重新创建 ActiveX 控制的窗口。

void RecreateWnd();

参 见 TVclComControl::Create。

TVclComControl::ShowWindow

设置窗口的 show 状态。

BOOL ShowWindow(int nCmdShow);

关于 nCmdShow 的信息，参见 Win32 Reference 中的::ShowWindow。

TVclComControl::TVclComControl

创建一个 TVclComControl 对象。

TVclComControl(void); m_VclCtl(0), m_VclWinCtl(0), m_hWnd(0), m_CtlWndProc(0), CComControlBase(m_hWnd);

不要直接调用 TVclComControl，利用 new 进行，它会自动调用 TVclComControl 方法。TVclComControl 调用 Init 方法。

TVclComControl::WndProc

处理 ActiveX 控制的 Windows 消息。

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

把消息发送给 ControlWndProc 方法。

参 见 TVclComControl::ControlWndProc

TVclControlImpl 单元 Atlvcl.h

是一个基于 VCL 控制的 ActiveX 控制的模板。

template <class T, class TVCL, const CLSID* pclsid, const IID* piid, const IID* peventsid, const GUID* plibid>

TVclControlImpl 被用在 ActiveX 向导生成的代码中。

TVCL 为 ActiveX 控制基于的 VCL 控制。

pclsid 为 ActiveX 控制的类 ID。

piid 为 ActiveX 控制的主接口。

peventsid 为 ActiveX 控制的输出事件接口。

plibid 为 ActiveX 控制的类型库的 GUID。

TVclControlImpl 是从下列类中派生的：

CComObjectRootEx, CComCoClass, TVclComControl, IProvideClassInfo2Impl, IPersistStreamInitImpl, IPersistStorageImpl, IQuickActivateImpl, IOleControlImpl, IOleObjectImpl, IOleInPlaceActiveObjectImpl, IViewObjectExImpl, IOleInPlaceObjectWindowlessImpl, IDataObjectImpl, ISpecifyPropertyPagesImpl, IConnectionPointContainerImpl, ISupportErrorInfo, ISimpleFrameSiteImpl。

方法列表

~TVclControlImpl	释放与 ~TVclControlImpl 对象有关的内存
FinalConstruct	初始化与 TVclComControlImpl 对象有关的数据
FinalRelease	不作操作
GetControllingUnknown	返回 ActiveX 控制的外部 IUnknown 接口
GetViewStatus	检索 ActiveX 控制视图的信息
InterfaceSupportsErrorInfo	确定一个接口是否支持错误信息
OnAmbientPropertyChange	把 ActiveX 控制的背景色、前景色以及字体属性，改变为匹配容器的相应属性
OnDraw	处理 ActiveX 控制的窗口的绘画
SetObjectRects	设置 ActiveX 控制的窗口的位置
TVclControlImpl	创建一个 TVclControlImpl 对象

参 见 TVclComControl。

方 法

TVclControlImpl::~TVclControlImpl

释放与 ~TVclControlImpl 对象有关的内存。

~TVclControlImpl() {}

不要直接调用 ~TVclControlImpl，利用 delete 进行，它会自动调用 ~TVclControlImpl。

TVclControlImpl::FinalConstruct

初始化与 TVclComControlImpl 对象有关的数据。

HRESULT FinalConstruct()

TVclControlImpl 调用 TVclComControl::Initialize，然后返回 S_OK。

参 见 TVclComControl::Initialize。

TVclControlImpl::FinalRelease

不作操作。

void FinalRelease();

TVclControlImpl::GetControllingUnknown

返回 ActiveX 控制的外部 IUnknown 接口。

virtual IUnknown* GetControllingUnknown();

TVclControlImpl::GetViewStatus

检索 ActiveX 控制视图的信息。

STDMETHOD(GetViewStatus)(DWORD* pdwStatus);

GetViewStatus 把 pdwStatus 的值设置为 VIEWSTATUS_SOLIDBKGND | VIEWSTATUS_OPAQUE，然后返回 S_OK。

TVclControlImpl::InterfaceSupportsErrorInfo

确定一个接口是否支持错误信息。

STDMETHOD(InterfaceSupportsErrorInfo)(REFIID riid);

如果接口支持错误信息，返回 S_OK；否则就返回 S_FALSE。

TVclControlImpl::OnAmbientPropertyChange

把 ActiveX 控制的背景色、前景色以及字体属性，改变为匹配容器的相应属性。

STDMETHOD(OnAmbientPropertyChange)(DISPID dispid);

对容器的字体和前景色的改变将被忽略，除非 ActiveX 控制的 ParentFont 属性为 true。

对容器的字体和背景色的改变将被忽略，除非 ActiveX 控制的 ParentFont 属性为 true。

TVclControlImpl::OnDraw

处理 ActiveX 控制的窗口的绘画。

HRESULT OnDraw(ATL_DRAWINFO& di);

TVclControlImpl::SetObjectRects

设置 ActiveX 控制的窗口的位置。

STDMETHOD(SetObjectRects)(LPCRECT prcPos, LPCRECT prcClip);

TVclControlImpl::TVclControlImpl

创建一个 TVclControlImpl 对象。

TVclControlImpl(){}

不要直接调用 TVclControlImpl，利用 new 进行，它会自动调用 TVclControlImpl 方法。

参 见 TVclComControlImpl::~TVclComControlImpl

TVclPropertyPage 单元 Atlvcl.h

template <class T, const CLSID * pclsid, class ppclass>

ActiveX 属性页的一个模板。

TVclPropertyPage 被用在 PropertyPage 向导生成的代码中。

T 为属性页类。pclsid 为属性页类的类 id。ppclass 为属性页的 VCL 组件类。

属性列表

m_InnerUnk	包含属性页实现的一个接口
m_PPImpl	包含属性页执行

方法列表

~TVclPropertyPage	释放与 TVclPropertyPage 对象有关的内存
FinalConstruct	初始化与 TVclPropertyPage 对象有关的数据
FinalRelease	释放与 TVclPropertyPage 对象有关的内存
TVclPropertyPage	创建一 TVclPropertyPage 对象

参 见 TPropertyPage。

属 性

TVclPropertyPage::m_InnerUnk

包含属性页实现的一个接口。

IUnknown * m_InnerUnk;

m_InnerUnk 由 FinalConstruct 方法来初始化。属性页实现被包含在 m_PPImpl 属性中。

参 见 TVclPropertyPage::FinalConstruct, TVclPropertyPage::m_PPImpl。

TVclPropertyPage::m_PPImpl

包含属性页执行。

TPropertyPageImpl * m_PPImpl;

m_PPImpl 由 FinalConstruct 方法来创建和初始化。m_PPImpl 的接口被包含在 m_InnerUnk 属性中。

参 见 TVclPropertyPage::FinalConstruct, TVclPropertyPage::m_InnerUnk。

方 法

TVclPropertyPage::~TVclPropertyPage

释放与 TVclPropertyPage 对象有关的内存。

~TVCLPropertyPage(){}

不要直接调用~TVclPropertyPage，利用 delete 进行，它会自动调用~TVclPropertyPage 方法。

TVclPropertyPage::FinalConstruct

初始化与 TVclPropertyPage 对象有关的数据。

HRESULT FinalConstruct(void);

初始化 m_PPImpl 和 m_InnerUnk 属性。

参 见 TVclPropertyPage::m_PPImpl, TVclPropertyPage::m_InnerUnk

TVclPropertyPage::FinalRelease

释放与 TVclPropertyPage 对象有关的内存。

void FinalRelease(void)

释放与 m_PPImpl 和 m_InnerUnk 属性有关的内存。调用基类的 FinaleRelease 方法。

参 见 TVclPropertyPage::m_PPImpl, TVclPropertyPage::m_InnerUnk。

TVclPropertyPage::TVclPropertyPage

创建一 TVclPropertyPage 对象。

TVCLPropertyPage(void): m_PPImpl(0), m_InnerUnk(0){}

不要直接调用 TVclPropertyPage，利用 new 进行，它会自动调用 TVclPropertyPage 方法。

参 见 TVclPropertyPage::~TVclPropertyPage。

TVclPtr 单元 Atlvcl.h

一个指针模板。该指针封装了一个标准指针，处理其内存管理。例如，删除指针时，诸如超出范围，与指针有关的内存就被释放。

TVclPtr 被用在 SaveVCLComponentToStream 和 LoadVCLComponentFromStream 函数中。

方法列表

~TVclPtr	释放与 TVclPtr 对象有关的内存
get	返回封装的指针
operator->()	实现右箭头(->)运算
operator!	实现感叹号(!)运算
operator!=	实现感叹号-等于号(!=)运算
operator&	实现与号(&)运算
operator*	实现星号(&)运算
operator=	实现等号(&)运算
operator==	实现双等号(==)运算
operatorT*	实现 T-星号(T*)运算，T 为 TVclPtr 对象的类名
release	释放封装的指针
reset	重置封装的指针
TVclPtr	创建一个新的 TVclPtr 对象

参 见 LoadVCLComponentFromStream 函数，SaveVCLComponentFromStream 函数。

方 法

TVclPtr::~TVclPtr

释放与 TVclPtr 对象有关的内存。

```
~TVclPtr() { delete m_ptr; }
```

不要直接调用~TVclPtr。利用 delete 进行,它会自动调用~TVclPtr。

TVclPtr::get

返回封装的指针。

```
T* get() const { return m_ptr;}
```

参 见 TVclPtr::release。

TVclPtr::operator->

实现右箭头(->)运算。

```
T* operator->() const { _ASSERTE(m_ptr); return m_ptr;}
```

TVclPtr::operator!

实现感叹号(!)运算。

```
bool operator ! () const{ return m_ptr == 0; }
```

TVclPtr::operator!=

实现感叹号-等于号(!=)运算。

```
bool operator != (T* rhs) const { return m_ptr != rhs;}
bool operator != (const TVclPtr<T>& rhs) const { return m_ptr != rhs.m_ptr; }
```

TVclPtr::operator&

实现与号(&)运算。

```
T** operator &()/*const*/ { return &m_ptr; }
```

TVclPtr::operator*

实现星号(&)运算。

```
T& operator * () const { _ASSERTE(m_ptr); return *m_ptr; }
```

TVclPtr::operator=

实现等号(&)运算。

```
TVclPtr& operator = (TVclPtr<T>& rhs) { reset(rhs.release()); return *this; }
```

TVclPtr::operator==

实现双等号(==)运算。

```
bool operator == (T* rhs) const { return m_ptr == rhs;}
bool operator == (const TVclPtr<T>& rhs) const { return m_ptr == rhs.m_ptr; }
```

TVclPtr::operatorT*

实现 T-星号(T*)运算,T 为 TVclPtr 对象的类名。

```
operator T* () const { return m_ptr;}
```

TVclPtr::release

释放封装的指针。

```
T* release() { return reset(0); }
```

参 见 TVclPtr::get。

TVclPtr::reset

重置封装的指针。

```
T* reset(T* p = 0) { T* tmp = m_ptr; m_ptr = p; return tmp; }
```

参 见 TVclPtr::get。

TVclPtr::TVclPtr

创建一个新的 TVclPtr 对象。

```
TVclPtr(T* p = 0) : m_ptr(p) {}
TVclPtr(TVclPtr<T>& src) : m_ptr(src.release()) {}
TVclPtr& operator = (TVclPtr<T>& rhs) { reset(rhs.release()); return *this; }
```

TWinControlAccess 单元 Atlvcl.h

```
template <class T>
```

包装了一个 ActiveX 控制的窗口句柄的类模板。

TWinControlAccess is 用在 ActiveX 控制向导生成的代码中。

T 为被创建的包装类。

方法列表

DoDestroyHandle	调用窗口的 DestroyHandle 方法
GetWindowHandle	返回窗口的 WindowHandle 属性
TWinControlAccess	创建一个 TWinControlAccess 对象

方 法

TWinControlAccess::DoDestroyHandle

调用窗口的 DestroyHandle 方法。

```
void DoDestroyHandle(void) {DestroyHandle();}
```

TWinControlAccess::GetWindowHandle

返回窗口的 WindowHandle 属性。

```
HWND GetWindowHandle(void) {return WindowHandle;}
```

TWinControlAccess::TWinControlAccess

创建一个 TWinControlAccess 对象。

```
__fastcall virtual TWinControlAccess(Classes::TComponent* AOwner): T(AOwner) {}
__fastcall TWinControlAccess(HWND ParentWindow): T(ParentWindow) {}
```

AOwner 为 ActiveX 控制的拥有者。

ParentWindow 为 ActiveX 控制的父窗口的句柄。

不要直接调用 TWinControlAccess,利用 new 进行,它会自动调用 TWinControlAccess 方法。