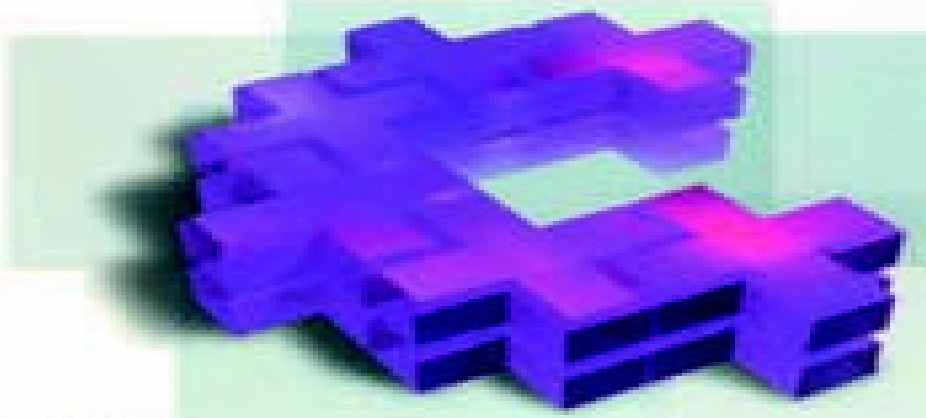
2



Microsoft Visual C++ 6.0

MFC Library Reference Part 2

类库参考手册(上)

北京希望电脑公司

Microsoft Press

返 回

目　　录

COleInsertDialog

COleIPFrameWnd

COleDBRecordView

COleLinkingDoc

COleLinksDialog

COleMessageFilter

COleObjectFactory

COlePasteSpecialDialog

COlePropertiesDialog

COlePropertyPage

COleResizeBar

COleSafeArray

COleServerDoc

COleServerItem

COleStreamFile

COleTemplateServer

COleUpdateDialog

COleVariant

CPageSetupDialog

CPaintDC

CPalette

CPen

CPictureHolder

CPoint

CPrintDialog

CPrintInfo

CProgressCtrl

CPropExchange

CPropertyPage

CPropertyPageEx

CPropertySheet

CPropertySheetEx

CPtrArray

CPtrList

CReBar

CReBarCtrl

CRecentFileList

CRecordset

CRecordView

CRect

CRectTracker

CResourceException

CRgn

CRichEditCntrItem

CRichEditCtrl

CRichEditDoc

CRichEditView

CRuntimeClass

CScrollBar

CScrollView

CSemaphore

CSharedFile

CSingleDocTemplate

CSingleLock

CSize

CSliderCtrl

CSocket

CSocketFile

CSpinButtonCtrl

CSplitterW nd

CStatic

CStatusBar

CStatusBarCtrl

CStdioFile

CString

CStringArray

CStringList

CSyncObject

CTabCtrl

CTime

CTimeSpan

CToolBar

COleDateTimeSpan

COleDateTimeSpan 没有基类。

一个 COleDateTimeSpan 对象表示一个相对的时间，即一个时间段。

COleDateTimeSpan 以日为单位来保存时间。

COleDateTimeSpan 类常常与类 COleDateTime 一起使用。COleDateTime 封装了 OLE 自动化中的 DATE 数据类型。COleDateTime 表示绝对的时间值。所有的 COleDateTime 计算都涉及 COleDateTimeSpan 值。这两个类之间的关系类似于类 CTime 和类 CTimeSpan 之间的关系。

有关类 COleDateTime和 COleDateTimeSpan 的更进一步的信息，可以参考“ Visual C++ 程序员指南”一书中的文章“日期与时间：自动化支持”。

```
# include <afxdisp.h>
```

COleDateTimeSpan 类成员

Constructor	
ColeDateTimeSpan	构造 COleDateTimeSpan 对象

Attributes	
GetStatus	获取 COleDateTimeSpan 对象的状态（有效性）
SetStatus	设置 COleDateTimeSpan 对象的状态（有效性）
GetDays	返回 COleDateTimeSpan 对象所表示的时间段的日部分
GetHours	返回 COleDateTimeSpan 对象所表示的时间段的小时部分
GetMinutes	返回 COleDateTimeSpan 对象所表示的时间段的分钟部分
GetSeconds	返回 COleDateTimeSpan 对象所表示的时间段的秒部分
GetTotalDays	返回 COleDateTimeSpan 对象所表示的天数
GetTotalHours	返回 COleDateTimeSpan 对象所表示的小时数

Attributes	
GetTotalMinutes	返回 COleDateTimeSpan 对象所表示的分钟数
GetTotalSeconds	返回 COleDateTimeSpan 对象所表示的秒数

Operations	
SetDateTimeSpan	设置 COleDateTimeSpan 对象的值
Format	产生一个 COleDateTimeSpan 对象的格式化字符串表达式

Operators	
操作符 double	将 COleDateTimeSpan 的值转换为双精度值
操作符 =	拷贝一个 COleDateTimeSpan 值
操作符 +，-	加、减 COleDateTimeSpan 值，或改变 COleDateTimeSpan 值的符号
操作符 +=，-=	从一个 COleDateTimeSpan 值中加、减这个 COleDateTimeSpan 值
操作符 ==，<，<=	比较两个 COleDateTimeSpan 值

Data Members	
m_span	记录这个 COleDateTimeSpan 对象所代表的基本的双精度值
m_status	记录 COleDateTimeSpan 对象的状态

Dump/Archive

操作符 <<	将 COleDateTimeSpan 值输出到 CArchive 或 CDumpContext
操作符 >>	从 CArchive 输入一个 COleDateTimeSpan 对象

成员函数

COleDateTimeSpan::COleDateTimeSpan

```
COleDateTimeSpan();
COleDateTimeSpan(const COleDateTimeSpan& dateSpanSrc);
COleDateTimeSpan(double dblSpanSrc);
COleDateTimeSpan(long lDays,int nHours,int nMins,int nSecs);
```

参数

dateSpanSrc

要拷贝到新的 COleDateTimeSpan 对象中去的一个已经存在的

COleDateTimeSpan 对象。

dblSpanSrc

要拷贝到新的 COleDateTimeSpan 对象中去的天数。

lDays,nHours,nMins,nSecs

表示要拷贝到新的 COleDateTimeSpan 对象中去的日期和时间值。

说明

所有这些构造函数创建的新的 COleDateTimeSpan 对象都被初始化为指定的值。下面给出了有关每一个构造函数的简短描述：

- COleDateTimeSpan（）构造一个初始值为 0 的 COleDateTimeSpan 对象。
- COleDateTimeSpan（*dateSpanSrc*）根据一个已经存在的 COleDateTimeSpan 对象构造一个新的 COleDateTimeSpan 对象。
- COleDateTimeSpan（*dblSpanSrc*）根据一个浮点值构造一个

COleDateTimeSpan 对象。

- COleDateTimeSpan（*lDays,nHours,nMins,nSecs*）构造一个按给定的数字值初始化的 COleDateTimeSpan 对象。

新的 COleDateTimeSpan 对象的状态被设置为有效的。

有关 COleDateTimeSpan 值的界限问题的更进一步的信息，参见“Visual C++ 程序员指南”一书中的文章“日期和时间：自动化支持”。

示例

```
COleDateTimeSpan spanOne（2.75）;                    // 2 天 18 小时
COleDateTimeSpan spanTwo（2，18，0，0）;             // 2 天 18 小时
COleDateTimeSpan spanThree（3，-6，0，0）；          // 2 天 18 小时
```

请参阅 COleDateTimeSpan::operator =，COleDateTimeSpan::GetStatus，

COleDateTimeSpan::m_span，COleDateTimeSpan::m_status

COleDateTimeSpan::Format

CString Format（LPCTSTR *pFormat*） const;

CString Format（UNIT *nID*） const;

返回值

一个含有格式化的日期/时间间隔值的 CString **值。**

参数

pFormat

一个代表格式的字符串，类似于 printf **中的代表格式的字符串。表示格式的代码，其前面的百分号（%）由相应的** COleDateTimeSpan **成分所代**

替。而表示格式的字符串中的其它字母则原封不动的拷贝到返回字符串中。更多的细节可参见运行时函数 strftime。在 Format 函数中表示格式的符号的值和含义列表如下：

- %H 当前日的小时数
- %M 当前小时中的分钟数
- %S 当前分钟中的秒数
- %% 百分号

nID

格式控制字符串的资源 ID。

说明

调用这些函数可以生成一个有关时间段值的格式化表达式。如果

COleDateTimeSpan 对象的状态为无效，则返回值为一个空的字符串。如果其状态为有效的，则返回的是一个由字符串资源 IDS_INVALID_DATETIMESPAN 确定的字符串。

有关这个函数的几种形式的简短描述如下所示：

Format（*pFormat*）

这种形式的函数用格式字符串来格式化相应的值，格式字符串中用前置的百分号（%）来指定表示格式的符号，就象在 printf 中一样。该格式字符串是一个资源。这个字符串的 ID 是作为参数传递给该函数的。

Format（*nID*）

这种形式的函数用格式字符串来格式化相应的值，格式字符串中用前置的百分号（%）来指定表示格式的符号，就象在 printf 中一样。该格式字符串是一个资源。这个字符串的 ID 是作为指针来传递的。

有关函数中使用的格式化符号的更多的信息,可以参见“Microsoft Visual C++ 6.0 运行时间库参考”。

有关本地 ID 值的列表，可参见“Win32 SDK OLE 程序员参考”一书中的“支持多国语言”章节。

请参阅 COleDateTimeSpan::GetStatus

COleDateTimeSpan::GetDays

long GetDays() const;

返回值

日期/时间段值的日部分。

说明

调用这个成员函数可以跟踪日期/时间段值的日部分。

这个函数的返回值的范围大致在-3,615,000 与 3,615,000 之间。

其它用来查询 COleDateTimeSpan 对象的函数可以参见下面列出的成员函数：

- GetHours
- GetMinutes
- GetSeconds
- GetTotalDays
- GetTotalHours
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::GetHours

```
long GetHours() const;
```

返回值

日期/时间段值的小时部分。

说明

调用这个成员函数可以获取日期/时间段值的小时部分。

这个函数的返回值的范围在-23与 23之间。

其它用来查询 COleDateTimeSpan 对象的函数可以参见下面列出的成员函数：

- GetDays
- GetMinutes
- GetSeconds
- GetTotalDays
- GetTotalHours
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::GetMinutes

long GetMinutes() const;

返回值

日期/时间段值的分钟部分。

说明

调用这个成员函数可以获取日期/时间段值的分钟部分。

这个函数的返回值的范围在-59 **与** 59 **之间。**

其它用来查询 COleDateTimeSpan **对象的函数可以参见下面列出的成员函数：**

- GetDays

- GetHours
- GetSeconds
- GetTotalDays
- GettotalHours
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::GetSeconds

```
long GetSeconds() const;
```

返回值

日期/时间段值的秒部分。

说明

调用这个成员函数可以获取日期/时间段值的秒部分。

这个函数的返回值的范围大致在-59 与 59 之间。

其它用来查询 COleDateTimeSpan 对象的函数可以参见下面列出的成员函数：

- GetDays
- GetHours
- GetMinutes
- GetTotalDays
- GetTotalHours
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::GetStatus

```
DateTimeSpanStatus GetStatus() const;
```

返回值

COleDateTimeSpan 值的状态。

说明

调用这个成员函数可以获取这个 COleDateTimeSpan 对象的状态（有效性）。

这个返回值是用 DateTimeSpanStatus 枚举类型定义的，该枚举类型是在 COleDateTimeSpan 类中定义的。

```
enum DateTimeSpanStatus{
valid = 0,
invalid = 1,
null = 2,
};
```

下面列出了有关这些状态值的简短描述：

- COleDateTimeSpan::valid 表示这个 COleDateTimeSpan 对象是有效的。
- COleDateTimeSpan::invalid 表示这个 COleDateTimeSpan 对象是无效的，也就是说，这个对象的值有可能是错误的。

- COleDateTimeSpan::null 表示这个 COleDateTimeSpan 对象是空的，也就是说，没有值被提供给这个对象。（在数据库中“null”意味着“没有值”，这与 C++的 NULL 相反。）

在下面的示例中一个 COleDateTimeSpan 对象的状态是无效的：

- 该对象在一次算术赋值操作，即+=或-=中，产生了上溢或者下溢。
- 一个无效的值被赋给了这个对象。
- 通过使用 SetStatus 将这个对象的状态显式的设置为无效。

有关其它可以将状态设置为无效的操作的更进一步的信息，参见 COleDateTimeSpan:: operator+，-和 COleDateTimeSpan::operator+=，-=。

有关 COleDateTimeSpan 值的界限的更进一步的信息，参见“Visual C++ 程序员指南”一书中的文章“日期与时间：自动化支持”。

请参阅 COleDateTimeSpan::SetStatus,ColeDateTimeSpan::m_status

COleDateTimeSpan::GetTotalDays

double GetTotalDays() const;

返回值

这个日期/时间段值以日为单位。虽然这个函数的原型是返回一个双精度值，但是它也常常返回一个整型值。

说明

调用这个成员函数可以获取以日为单位表示的日期/时间段值。

这个函数的返回值的范围大致在-3.65e6 **与** 3.65e6 **之间。**

其它用来查询 COleDateTimeSpan **对象的函数可以参见下面列出的成员函数：**

- GetDays
- GetHours
- GetMinutes
- GetSeconds
- GetTotalHours
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan，ColeDateTimeSpan::operator

COleDateTimeSpan::GetTotalHours

```
double GetTotalHours() const;
```

返回值

这个日期/时间段值以小时为单位。虽然这个函数的原型是返回一个双精度值，但是它也常常返回一个整型值。

说明

调用这个成员函数可以获取以小时为单位的日期/时间段值。

这个函数的返回值的范围大致在-8.77e7 与 8.77e7 之间。

其它用来查询 COleDateTimeSpan 对象的函数可以参见下面列出的成员函数：

- GetDays
- GetHours
- GetMinutes
- GetSeconds
- GetTotalDays
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::GetTotalMinutes

```
double GetTotalMinutes() const;
```

返回值

这个日期/时间段值以分钟为单位。虽然这个函数的原型是返回一个双精度值，但是它也常常返回一个整型值。

说明

调用这个成员函数可以获取以分钟为单位的日期/时间段值。

这个函数的返回值的范围大致在-5.26e9 与 5.26e9 之间。

其它用来查询 COleDateTimeSpan 对象的函数可以参见下面列出的成员函数：

- GetDays
- GetHours
- GetMinutes
- GetSeconds
- GetTotalDays
- GetTotalHours

- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::GetTotalSeconds

double GetTotalSeconds() const;

返回值

这个日期/时间段值以秒为单位。虽然这个函数的原型是返回一个双精度值，但是它也常常返回一个整型值。

说明

调用这个成员函数可以获取以秒为单位的日期/时间段值。

这个函数的返回值的范围大致在-3.16e11 **与** 3.16e11 **之间。**

其它用来查询 COleDateTimeSpan **对象的函数可以参见下面列出的成员函数：**

- GetDays
- GetHours
- GetMinutes
- GetSeconds
- GetTotalDays
- GetTotalMinutes
- GetTotalSeconds

请参阅 COleDateTimeSpan::SetDateTimeSpan

COleDateTimeSpan::SetDateTimeSpan

void SetDateTimeSpan(long *lDays*,int *nHours*,int *nMins*,int *nSecs*);

参数

kDays,nHours,nMins,nSecs

表示要被拷贝到 COleDateTimeSpan **对象中去的日期段和时间段值。**

说明

调用这个成员函数可以设置日期/时间段值。

其它用来查询 COleDateTimeSpan 对象的函数可以参见下面列出的成员函数：

- GetDays
- GetHours
- GetMinutes
- GetSeconds
- GetTotalDays
- GetTotalHours
- GetTotalMinutes
- GetTotalSeconds

示例

```
COleDateTimeSpan spanOne;
COleDateTimeSpan spanTwo;
spanOne.SetDateTimeSpan(0,2,45,0);          //2 小时 45 秒
```

spanTwo.SetDateTimeSpan(0,3,-15,0);　　//2 小时 45 秒

请参阅　COleDateTimeSpan::GetStatus,COleDateTimeSpan::m_span

COleDateTimeSpan::SetStatus

void SetStatus(DateTimeSpanStatus *nStatus*);

参数

nStatus

　　表示 COleDateTimeSpan **对象的新的状态值。**

说明

调用这个成员函数可以设置 COleDateTimeSpan **对象的状态（有效性）。参数** *nStatus* **值是用** DateTimeSpanStatus **枚举类型定义的，该枚举类型是在**

COleDateTimeSpan 类中定义的。

```
enum DateTimeSpanStatus{
   valid = 0,
   invalid = 1,
   null = 2,
};
```

下面列出了有关这些状态值的简短描述：

- COleDateTimeSpan::valid 表示这个 COleDateTimeSpan 对象是有效的。
- COleDateTimeSpan::invalid 表示这个 COleDateTimeSpan 对象是无效的，也就是说，这个对象的值有可能是错误的。
- COleDateTimeSpan::null 表示这个 COleDateTimeSpan 对象是空的，也就是说，没有值被提供给这个对象。（在数据库中“null”意味着“没有值”，这与 C++的 NULL 不同。）

警告 这个函数适用于高级编程的情况。这个函数不改变对象中的数据。

它常常被用来将对象的状态设置为 null 或 invalid。注意赋值操作符（operator =）和 SetDateTimeSpan 是基于源值来设置对象的状态。

请参阅 COleDateTimeSpan::GetStatus,COleDateTimeSpan::m_status

操作符

COleDateTimeSpan::operator =

```
const COleDateTimeSpan& operator=( double dblSpanSrc );
const COleDateTimeSpan& operator=( const COleDateTimeSpan& dateSpanSrc );
```

说明

这些重载的赋值操作符将源日期/时间段值拷贝到 COleDateTimeSpan 对象中。

请参阅 COleDateTimeSpan::COleDateTimeSpan

COleDateTimeSpan::operator +,-

COleDateTimeSpan opetator+(const COleDateTimeSpan& *dateSpan*) const;
COleDateTimeSpan operator-(const COleDateTimeSpan& *dateSpan*) const;
COleDateTimeSpan operator-() const;

说明

前两个操作符可以用来加、减日期/时间段值。第三个操作符用来改变日期/时间段值的符号。

如果两个操作数有一个为空，则结果中的 COleDateTimeSpan **值的状态也是空。**

如果两个操作数中一个为无效，另一个不为空，则结果中的 COleDateTimeSpan **值的状态也是无效。**

有关状态值 valid，invalid **和** null **的进一步信息，可参考成员变量** m_status。

请参阅 COleDateTimeSpan::operator+=，-=

COleDateTimeSpan::operator +=,-=

const COleDateTimeSpan opetator+=(const COleDateTimeSpan *dateSpan*) ;
constCOleDateTimeSpan operator-=(const COleDateTimeSpan *dateSpan*);

说明

这两个操作符可以用来从一个 COleDateTimeSpan **对象中加、减这个日期/时间段值。**

如果两个操作数有一个为空，则结果中的 COleDateTimeSpan **值的状态也是空。**

如果两个操作数中一个为无效，另一个不为空，则结果中的 COleDateTimeSpan **值的状态也是无效。**

有关状态值 valid，invalid **和** null **的进一步信息，可参考成员变量** m_status。

请参阅 COleDateTimeSpan::operator +，-

COleDateTimeSpan::operator double

operator double() const

说明

这个操作符将一个 COleDateTimeSpan **值的天数作为一个浮点值返回。**

请参阅 COleDateTimeSpan::GetTotalDays, COleDateTimeSpan::SetDateTimeSpan, COleDateTimeSpan::m_span

COleDateTimeSpan Relational Operators

BOOL operator= = (const COleDateTimeSpan& *dateSpan*) const;

BOOL operator!= (const COleDateTimeSpan& *dateSpan*) const;

BOOL operator< (const COleDateTimeSpan& *dateSpan*) const;

BOOL operator> (const COleDateTimeSpan& *dateSpan*) const;

BOOL operator<= (const COleDateTimeSpan& *dateSpan*) const；

BOOL operator>= (const COleDateTimeSpan& *dateSpan*) const；

说明

这些操作符比较两个日期/时间段值，如果条件满足则返回非零值；否则返回零值。

警告 如果两个操作数中的任何一个的状态为空或者无效，则顺序操作符（<，<=，>，>=）的返回值就是不确定的。等号操作符（= =，!=）也要考虑操作数的状态。

示例

```
COleDateTimeSpan spanOne(3,12,0,0);                    //3 天 12 小时
```

```
COleDateTimeSpan spanTwo(spanOne);          //3 天 12 小时

Bool b;
b = spanOne = = spanTwo;                    //TRUE
spanTwo.SetStatus(COleDateTimeSpan::invalid);
b = spanOne = = spanTwo;                    // FALSE,different status
b = spanOne != spanTwo;                     // TRUE,different status
b = spanOne < spanTwo;                      // FALSE,same value
b = spanOne > spanTwo;                      // FALSE,same value
b = spanOne <= spanTwo;                     // TRUE,same value
b = spanOne >= spanTwo;                     // TRUE,same value
```

警告　上述示例中的后四行将在调试模式中声明。

ColeDateTimeSpan::Operator <<，>>

friend CDumpContext& AFXAPI operator<<(CDumpContext& *dc*,COleDateTimeSpan *dateSp- an*)
friend CArchive& AFXAPI operator<<(CArchive& *ar*,COleDateTimeSpan *dateSpan*);
friend CArchive& AFXAPI operator>>(CArchive& *ar*,COleDateTimeSpan

dateSpan);

说 明

COleDateTimeSpan **插入操作符（**<<**）支持诊断转储并可以保存到一个文件。提取操作符（**>>**）支持从一个文件中提取内容。**

请参阅 CDumpContext,CArchive

数据成员

COleDateTimeSpan::m_span

说明

这是 COleDateTimeSpan **对象的基本的双精度值。其值以日为单位表示日期/时间段值。**

> **警告** **改变这个双精度数据成员的值，将改变该** COleDateTimeSpan **对象的值。它不改变这个** COleDateTimeSpan **对象的状态。**

请参阅 COleDateTimeSpan::COleDateTimeSpan，COleDateTimeSpan::SetDateTimeSpan，COleDateTimeSpan::operator double

COleDateTimeSpan::m_status

说明

这个数据成员的数据类型是枚举类型 DateTimeSpanStatus，**此枚举类型是在** ColeDateTime- Span **类中定义的。**

```
enum DateTimeSpanStatus{
   valid = 0,
   invalid = 1,
   null = 2,
};
```

有关这些状态值的简短描述，参见下面的列表：

- COleDateTimeSpan::valid **表示这个** COleDateTimeSpan **对象是有效的。**
- COleDateTimeSpan::invalid **表示这个** COleDateTimeSpan **对象是无效的，也就是说，这个对象的值有可能是错误的。**

- COleDateTimeSpan::null 表示这个 COleDateTimeSpan 对象是空的，也就是说，没有值被提供给这个对象。（在数据库中“null”意味着“没有值”，这与 C++的 NULL 不同。）

在下面的示例中一个 COleDateTimeSpan 对象的状态是无效的：

- 该对象在一次算术赋值操作，即+=或-=中，产生了上溢或者下溢。
- 一个无效的值被赋给了这个对象。
- 通过使用 SetStatus 将这个对象的状态显式的设置为无效。

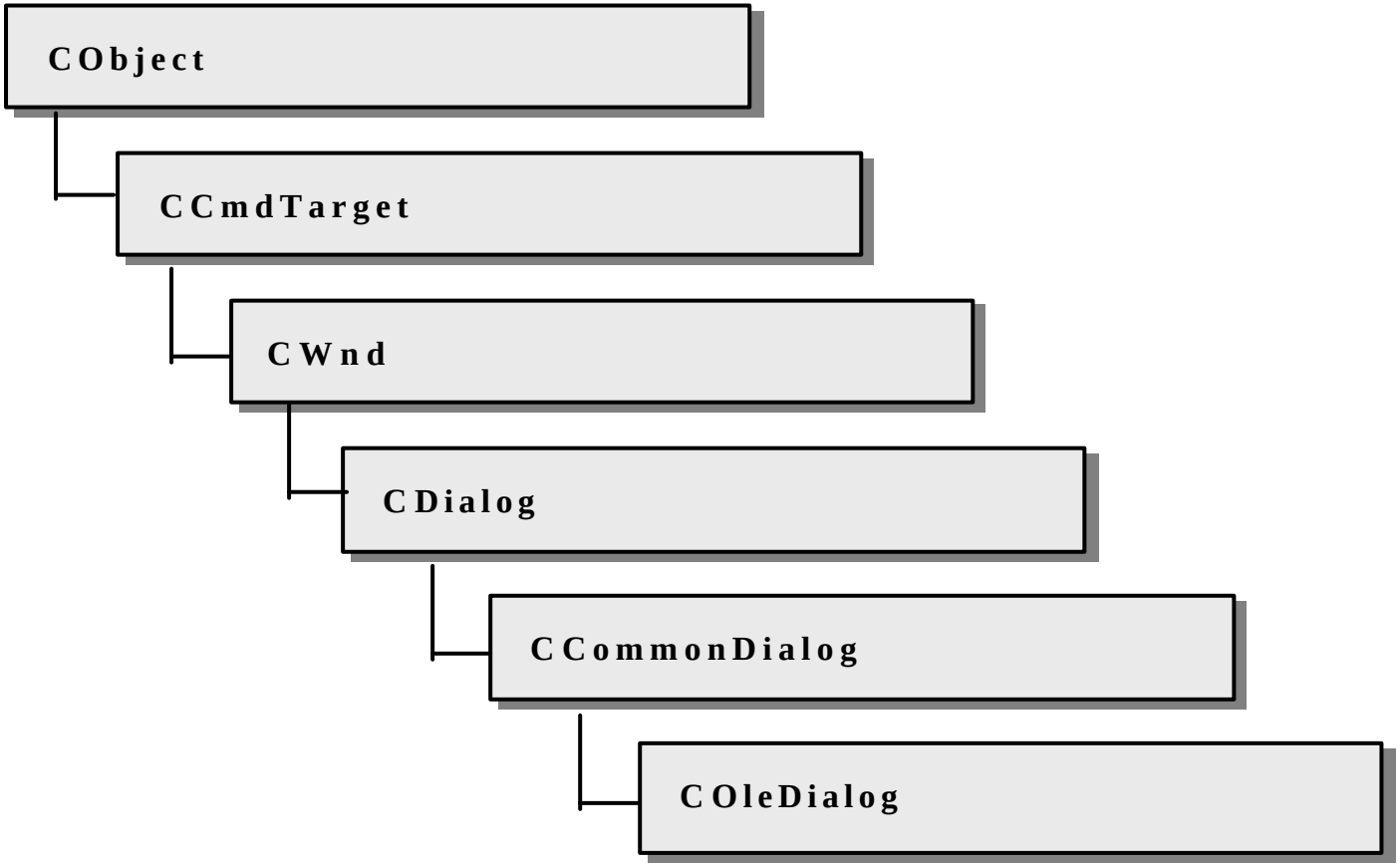
有关其它可以将状态设置为无效的操作的更进一步的信息，参见 ColeDateTimeSp- an::operator+，-和 COleDateTimeSpan::operator+=，-=。

警告 这个数据成员用于高级程序开发情况中。可以使用内联成员函数 GetStatus 和 SetStatus。参见 SetStatus 可以获得有关如何设置这个数据成员的更进一步的警告。

有关 COleDateTimeSpan 值的界限问题的更多信息，参见“Visual C++ 程序员指南”一书中的文章“日期与时间：自动化支持”。

请参阅 COleDateTimeSpan::GetStatus，COleDateTimeSpan::SetStatus

COleDialog



COleDialog 类为 OLE 对话框提供了通用的基本功能。微软基础类库提供了几

个由 COleDialog 派生的类。这些类是：

- COleInsertDialog
- COleConvertDialog
- COleCChangeIconDialog
- COleLinksDialog
- COleBusyDialog
- COleUpdateDialog
- COlePasteSpecialDialog
- COlePropertiesDialog
- COleChangeSourceDialog

有关 OLE 特定对话框的更多信息，参见“Visual C++程序员指南”一书中的文章“OLE 中的对话框”。

#include <afxodlgs.h>

COleDialog 类成员

Operations	
GetLastError	获取对话框返回的错误代码

成员函数

COleDialog::GetLastError

UNIT GetLastError() const;

返回值

由 GetLastError 返回的错误代码是基于所显示的特定的对话框的。

说明

当 DoModal **返回** IDABORT **时，调用** GetLastError **成员函数可以获得额外的错误信息。参见派生类中的** DoModal **成员函数可以获得有关特别错误消息的更进一步的信息。**

请参阅 COleInsertDialog::DoModal，COleConvertDialog::DoModal，
COleChangeIconDialog::DoModal，COleLinksDialog::DoModal，
COleBusyDialog::DoModal，COleUpdateDialog::DoModal，
COlePasteSpecialDialog::DoModal，COlePropertiesDialog::DoModal，
COleChangeSourceDialog::DoModal

COleDispatchDriver

COleDispatchDriver 没有基类。

COleDispatchDriver 类实现 OLE 自动化中的客户方。OLE 调度接口为访问一个对象的方法和属性提供了途径。COleDispatchDriver 的成员函数连接，分离，创建和释放一个 IDispatch 类型的调度连接。其它的成员函数使用变量参数列表来简化调用 IDispatch::Invoke。

要获得更多的信息，参见“Win32 SDK OLE 程序员参考”中的 IDispatch 和 IDispatch::Inv- oke。

这个类可以直接使用，但一般来说，它只是由用 ClassWizard 创建的类使用。

当你通过引用一个类型库来创建一个新的 C++类时，ClassWizard 从 COleDispatchDriver 派生出新类。

有关使用 COleDispatchDriver 的更多信息，参见下面列出的“Visual C++程序员指南”一书中的文章：

- 自动化客户
- 自动化服务器
- ClassWizard:自动化支持

#include <afxdisp.h>

请参阅 CCmdTarget

COleDispatchDriver 成员函数

Data Members	
m_bAutoRelease	在执行 ReleaseDispatch 或析构对象时，指定是否要释放 Idispatch
m_lpDispatch	表示指向附着在这个 ColeDispatchDriver 的 IDispatch 接口的指针

Construction	
ColeDispatchDriver	构造一个 COleDispatchDriver 对象

Operations	
CreateDispatch	创建一个 IDispatch 连接并将它附着于 COleDispatchDriver 对象
AttachDispatch	将一个 IDispatch 连接附着于 ColeDispatchDriver 对象
DetachDispatch	分开一个 IDispatch 连接，但并不释放它
ReleaseDispatch	释放一个 IDispatch 连接
InvokeHelper	用于调用自动化方法的助手
SetProperty	设置一个自动化特性
GetProperty	获得一个自动化特性

成员函数

COleDispatchDriver::AttachDispatch

void AttachDispatch(LPDISPATCH *lpDispatch*,BOOL *bAutoRelease* = TRUE);

参数

lpDispatch

被附着于 COleDispatchDriver **对象的指向一个** OLE IDispatch **对象的指针。**

bAutoRelease

表明当对象超出范围时，是否要释放 dispatch。

说明

调用 AttachDispatch **成员函数可以将一个** IDispatch **指针附着于**

COleDispatchDriver 对象上。这个函数释放任何已经附着于 COleDispatchDriver 对象的 IDispatch 指针。

请参阅 COleDispatchDriver::DetachDispatch，COleDispatchDriver::ReleaseDispatch，COleDispatchDriver::CreateDispatch，COleDispatchDriver::m_lpDispatch，COleDispatchDriverD::m_bAutoRelease

COleDispatchDriver:: COleDispatchDriver

```
COleDispatchDriver();
COleDispatchDriver(LPDISPATCH lpDispatch,BOOL bAutoRelease = TRUE);
COleDispatchDriver(const COleDispatchDriver& dispatchSrc);
```

参数

lpDispatch

指向一个被附着于 COleDispatchDriver **对象的** OLE IDispatch **对象的指针。**

bAboutRelease

当对象超出范围时，表明是否要释放这个 dispatch。

dispatchSrc

对一个已存在的 COleDispatchDriver **对象的引用。**

说明

构造一个 COleDispatchDriver **对象。** COleDispatchDriver(LPDISPATCH *lpDispatch*,BOOL *bAutoRelease* = TRUE)**连接** IDispatch **接口。**

COleDispatchDriver(const COleDispatchDriver& *dispatchSrc*)**拷贝一个已存在的** ColeDispat- chDriver **对象，并增加引用计数。**

COleDispatchDriver( )创建一个 COleDispatchDriver 对象，但不连接 IDispatch 接口。在使用一个没有参数的 COleDispatchDriver( )之前，应该用 COleDispatchDriver::CreateDispatch 或者 COleDispatchDriver::AttachDispatch 为它连接一个 IDispatch。

请参阅 COleDispatchDriver::AttachDispatch，COleDispatchDriver::CreateDispatch

COleDispatchDriver::CreateDispatch

BOOL CreateDispatch(REFCLSID *clsid*,COleException* *pError* = NULL);
BOOL CreateDispatch(LPCTSTR *lpszProgID*,COleException* *pError* = NULL);

返回值

成功则返回非零值；否则返回 0。

参数

clsid

要创建的 IDispatch **连接对象的类** ID**。**

pError

指向一个 OLE **异常对象的指针，这个异常对象将保存在创建中产生的状态码。**

lpszProgID

指向一个自动化对象的标识符的指针，如“ Excel.Document.5 **”。将要创建的调度对象是为这个自动化对象创建的。**

说明

创建一个 IDispatch **对象并将它与** COleDispatchDriver **对象连接。**

请参阅 COleDispatchDriver::DetachDispatch **，**

COleDispatchDriver::ReleaseDispatch，

COleDispatchDriver::AttachDispatch ， COleException ，

COleDispatchDriver::m_
lpDispatch

COleDispatchDriver::DetachDispatch

LPDISPATCH DetachDispatch();

返回值

一个指向以前连接的 IDispatch **对象的指针。**

说明

将当前的 IDispatch **连接从对象上分离。但不释放这个** IDispatch。

有关 LPDISPATCH **类型的更多的信息，参见“** OLE **程序员参考”中的** IDispatch。

请参阅 COleDispatchDriver::ReleaseDispatch，COleDispatchDriver::CreateDispatch，COleDispatchDriver::AttachDispatch，COleDispatchDriver::m_lpDispatch

COleDispatchDriver::GetProperty

void GetProperty(DISPID *dwDispID*,VARTYPE *vtProp*,void* *pvProp*) const;

参数

dwDispID

标识需要获得的属性。这个值通常由 ClassWizard 提供。

vtProp

指定要获取的属性。可能的取值可以参见 COleDispatchDriver::InvokeHelper 的说明部分。

pvProp

用来接收属性值的变量的地址。它必须与 vtProp **所指定的类型相匹配。**

说明

获取由 dwDispID **指定的对象属性。**

请参阅 COleDispatchDriver::InvokeHelper，COleDispatchDriver::SetProperty

COleDispatchDriver::InvokeHelper

void InvokeHelper(DISPID *dwDispID*,WORD *wFlags*,VARTYPE *vtRet*,void* *pvRet*,
const BYTE FAR* *pbParamInfo*,...);
throw(COleException);
throw(COleDispatchException);

参数

dwDispID

标识要激活的方法或者属性。这个值常常由 ClassWizard **提供。**

wFlag

描述调用 IDispatch::Invoke **的前后关系的标志。其可能的取值参见** *Platform* SDK。

vtRet

指定返回值的类型。其可能的取值参见说明部分。

pvRet

是将用来接收属性值或返回值的变量的地址。它必须与 *vtRet* **所指定的类型相匹配。**

pbParamInfo

是一个指向用空字符结尾的字符串的指针，这个字符串有多个字节用来

指定紧跟 *pbParamInfo* 之后的参数的类型。

...

参数变量列表，这些参数的类型在 *pbParamInfo* 中指定。

说明

在由 *wFlags* 指定的前后关系中调用由 *dwDispID* 指定的方法或属性。参数 *pbParamInfo* 指定要传递给方法或属性的参数的类型。在语法说明中...所代表的就是参数的变量列表。

vtRet 参数可能的取值来自于 VARENUM 枚举。

可能的取值如下所示：

符号	返回类型
VT_EMPTY	Void
VT_I2	Short
VT_I4	Long
VT_R4	Float
VT_R8	Double
VT_CY	CY
VT_DATE	DATE
VT_BSTR	BSTR
VT_DISPATCH	LPDISPATCH
VT_ERROR	SCODE
VT_BOOL	BOOL
VT_VARIANT	VARIANT
VT_UNKNOWN	LPUNKNOWN

参数 *pbParamInfo* 是一个由空格分隔的有关 VTS_常量的列表。这些值中的许多是由空格来分隔的（而不是逗号），它们指明了函数的参数列表。可能的取值由宏 EVENT_CUSTOM 列表给出。

此函数将参数转换为 VARIANTARG 值，然后激活 IDispatch::Invoke 方法。如果对 Invoke 的调用失败，这个函数将抛出一个异常。如果由 IDispatch::Invoke 返回的 SCODE（状态码）是 DISP_E_EXCEPTION，此函数抛出一个 COleException 对象；否则它抛出一个 COleDispatchException。

更多的信息请参见“Platform SDK”中的 VARIANTARG，IDispatch，IDispatch::Invoke，以及 COM 错误代码的结构。

请参阅 COleException，COleDispatchException

COleDispatchDriver::ReleaseDispatch

void ReleaseDispatch();

说明

释放 IDispatch 连接。如果这个连接已经设置为自动释放，则在释放这个接口之

前，此函数将调用 Dispatch::Release。

请参阅 COleDispatchDriver::DetachDispatch，COleDispatchDriver::CreateDispatch，COleDispatchDriver::AttachDispatch，COleDispatchDriver::m_lpDispatch，COleDispatchDriver::m_bAutoRelease

COleDispatchDriver::SetProperty

void SetProperty(DISPID *dwDispID*,VARTYPE *vtProp*,...);

参数

dwDispID

标识要设置的属性。这个值常常由 ClassWizard 提供。

vtProp

指明要设置的属性的类型。可能的取值参见COleDispatchDriver::InvokeHelper中的说明部分。

...

由vtProp指明类型的单一参数。

说明

设置由dwDispID指定的OLE对象属性。

请参阅 COleDispatchDriver::InvokeHelper，COleDispatchDriver::GetProperty

数据成员

COleDispatchDriver::m_bAutoRelease

说明

如果为 TRUE，则当调用 ReleaseDispatch 或 COleDispatchDriver 对象被销毁时，由 m_lpDispatch 访问的 COM 对象将被自动释放。

缺省的，在构造函数中 m_bAutoRelease 被设置为 TRUE。

有关释放 COM 对象的更多信息，参见“OLE 2 程序员参考，卷 1”中的“实现引用计数和 IUnknown::Release。”

请参阅 COleDispatchDriver::AttachDispatch，

COleDispatchDriver::ReleaseDispatch，

COleDispatchDriver::m_lpDispatch

COleDispatchDriver::m_lpDispatch

说明

这是指向连接着 COleDispatchDriver **的** IDispatch **接口的指针。** m_lpDispatch **数据成员是一个类型为** LPDISPATCH **的公用变量。**

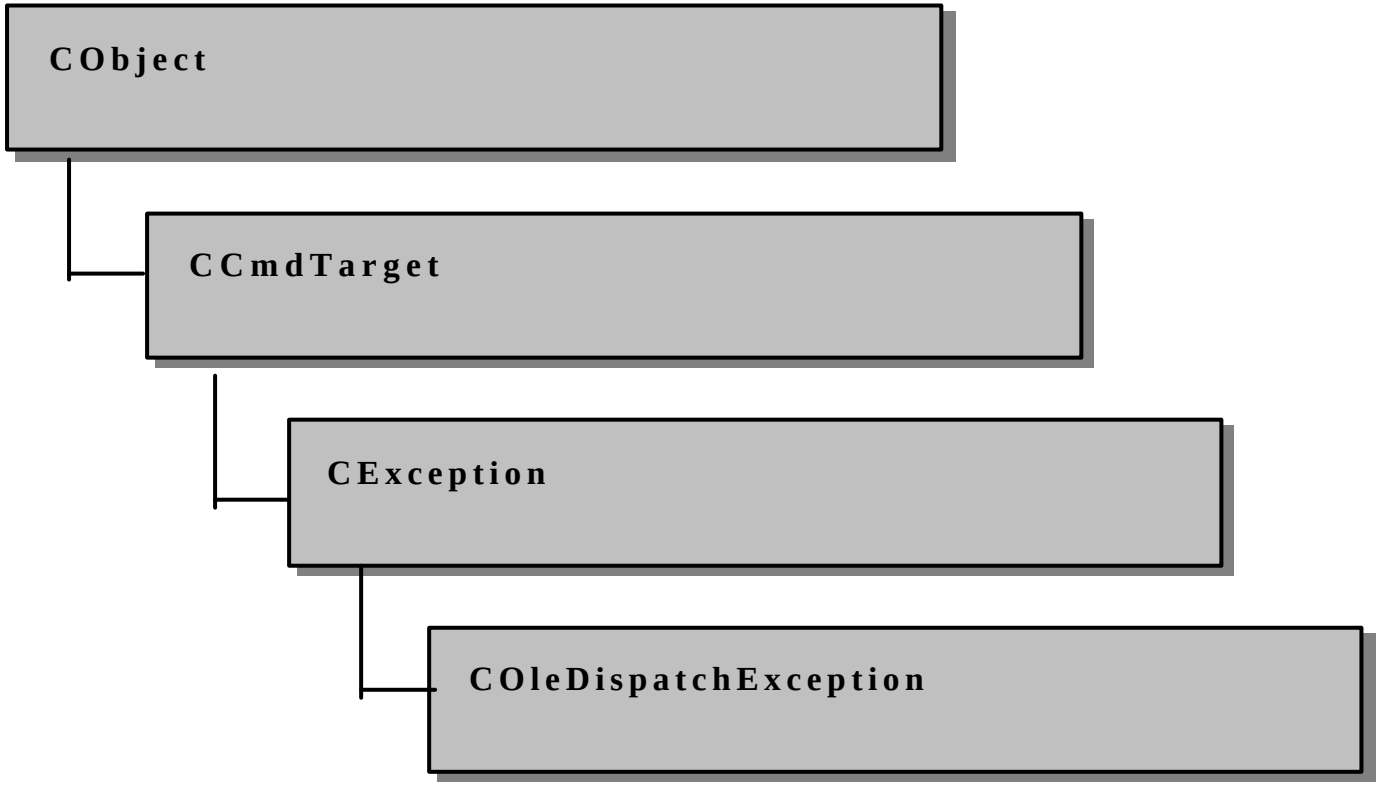
更多的信息可以参见“ OLE **程序员参考”中的** IDispatch**。**

请参阅 COleDispatchDriver::AttachDispatch **，** COleDispatchDriver::ReleaseDispatch，

COleDispatchDriver::CreateDispatch **，**

COleDispatchDriver::DetachDispatch

COleDispatchException



COleDispatchException 类处理对 OLE IDispatch 接口来说特定的异常，这个接口是 OLE 自动化中的关键部分。

象其它由基类 CExecption 派生出来的异常类一样，COleDispatchException 可以与 THROW，THROW_LAST，TRY，CATCH，AND_CATCH 和 END_CATCH 宏一起使用。

一般来说，可以调用 AfxThrowOleDispatchException 来创建和抛出一个 ColeDispatch- Exception 对象。

有关异常的更多信息，参见“Visual C++程序员指南”一书中的“异常”和“异常：OLE 异常”。

#include<afxdisp.h>

请参阅 COleDispatchDriver，COleException

COleDispatchException 类成员

Data Members	
m_wCode	IDispatch 特定错误代码
m_strDescription	动词的错误描述
m_dwHelpContext	错误的帮助上下文

Data Members	
m_strHelpFile	随 m_dwHelpContext 使用的帮助文件
m_strSource	产生异常的应用

数据成员

COleDispatchException::m_dwHelpContext

DWORD m_dwHelpContext;

说明

在你的应用程序的帮助文件（.HLP）中标识帮助的上下文。当出现一个异常时，由函数 AfxThrowOleDispatchException 设置这个成员。

请参阅 COleDispatchException::m_strDescription，COleDispatchException::m_wCode，AfxThrowOleDispatchException

COleDispatchException::m_strDescription

CString m_strDescription;

说明

包含动词的错误描述，如“磁盘满”。当出现一个异常时，由函数 AfxThrowOleDispatch-Exception 设置这个成员。

请 参 阅 COleDispatchException::m_dwHelpContext ，

COleDispatchException::m_wCode，

AfxThrowOleDispatchException

COleDispatchException::m_strHelpFile

CString m_strHelpFile;

说 明

这个字符串的内容是应用程序的帮助文件名。

请参阅 AfxThrowOleDispatchException

ColeDispatchException::m_strSource
Cstring m_strSource

说明

框架将产生异常的应用程序的名字填入该字符串。

请参阅 AfxThrow OleDispatch Excoption

COleDispatchException::m_wCode
WORD m_wCode;

说明

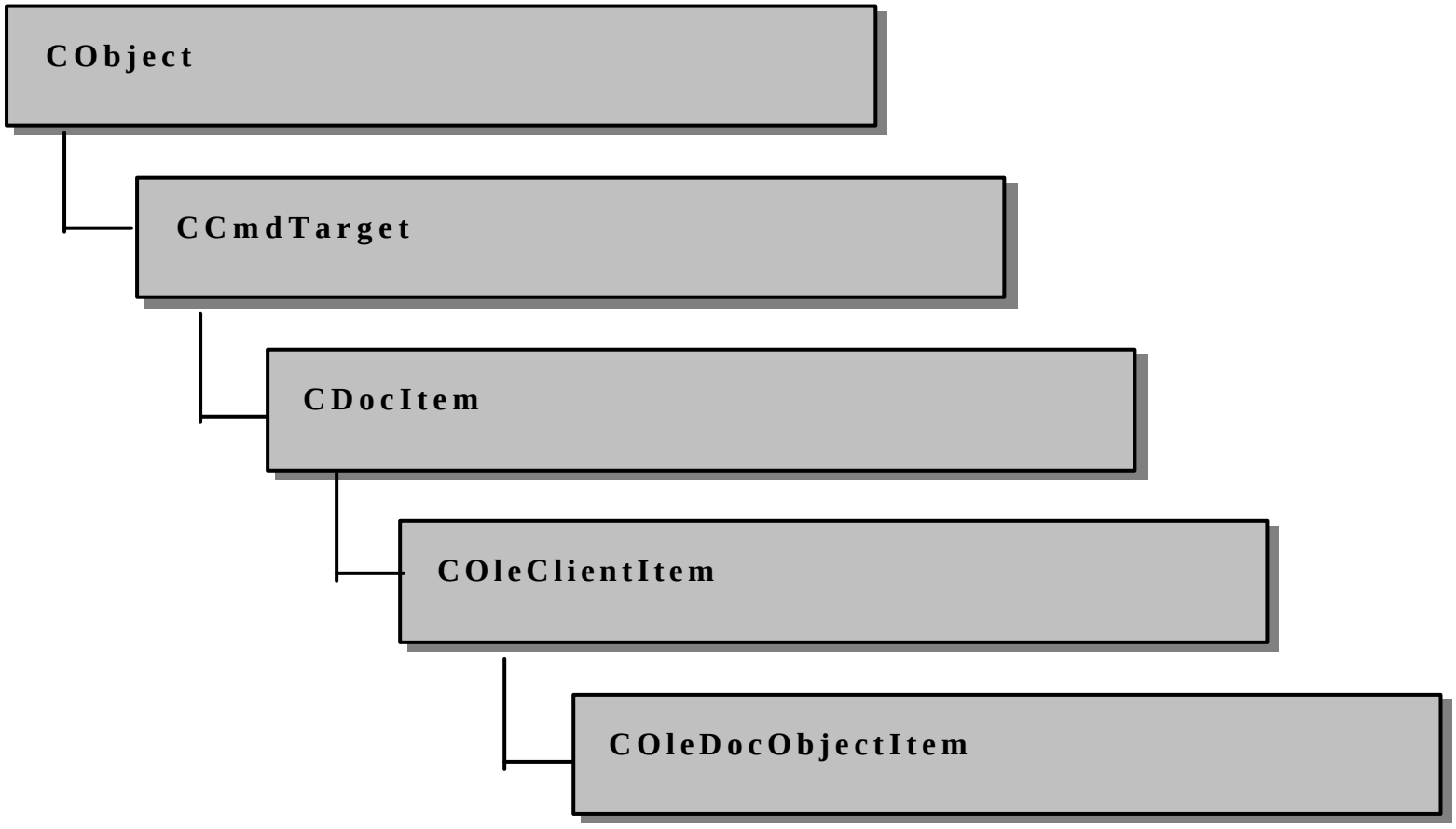
记录一个你的应用程序的特定错误代码。当出现一个异常时，由函数 AfxThrowOleDispatch- Exception 设置这个成员。

请参阅 COleDispatchException::m_strDescription，

COleDispatchException::m_dwHelpContext ，

AfxThrowOleDispatchException

COleDocObjectItem



COleDocObjectItem 类实现了活动文档容器。在 MFC 中，处理一个活动文档类似于处理一个普通的，现场可编辑的嵌入文档，除了下面列出的不同之处：

- COleDocument 派生类仍然保存一个当前内置项的列表；但是这些项有可能是 COleDocObjectItem 派生的项。
- 当一个活动文档是活动的时，当它被现场激活时，它占据窗口中的整个客户区。
- 一个活动文档容器对 Help 菜单具有完全的控制。
- Help 菜单包含适用于活动文档容器和服务器的菜单项。

由于活动文档容器拥有 Help 菜单，因此容器要负责将服务器的 Help 菜单消息向前传送给服务器。这个集成过程由 COleDocObjectItem 来处理。

有关菜单合并和激活活动文档的更多消息，参见“Visual C++程序员指南”一书中的“获得文档容器概述”。

include <afxole.h>

请参阅 COleClientItem，CDocObjectServerItem

COleDocObjectItem 类成员

Constructors	
ColeDocObjectItem	构造一个 COleDocObject 项

Operations	
GetActiveView	获取文档的活动视图
OnPreparePrinting	准备容器应用程序要打印的文档
OnPrint	打印容器应用程序的文档
GetPageCount	获取容器应用程序中的文档的页数
ExecCommand	执行用户指定的命令

成员函数

COleDocObjectItem::COleDocObjectItem

COleDocObjectItem(COleDocument* *pContainerDoc* = NULL);

参数

pContainerDoc

是一个指向 COleDocument 对象的指针，这个对象是作为活动文档容器的。为了使能 IMPLEMENT_SERIALIZE，这个参数必须为 NULL。而一般来说，是必须用一个非 NULL 文档指针来构造一个 OLE 项。

说明

调用这个成员函数来初始化 COleDocObjectItem 项。

COleDocObjectItem::ExecCommand

HRESULT ExecCommand(DWORD *nCmdID*,
    DWORD *nCmdExecOpt* = OLECMDEXECOPT_DONTPROMPTUSER,
    const GUID* *pguidCmdGroup* = NULL);

返回值

如果成功则返回 S_OK**；否则，返回下列错误代码之一：**

Value	Description
E_UNEXPECTED	**发生了没有预想到的错误**
E_FAIL	**发生了错误**
E_NOTIMPL	**表示** MFC **自己尝试翻译和分配命令**
OLECMDERR_E_UNKNOWNGROUP	PguidCmdGroup **不是** NULL**，但没有指明一个公认的命令群**
OLECMDERR_E_NOTSUPPORTED	**在** pGroup **群中** nCmdID **没有被认为是一个有效的命令**
OLECMDERR_DISABLED	**由** nCmdID **标识的命令是无效的，不能被执行**

续表

OLECMDERR_NOHELP	调用者用 nCmdID 标识的命令请求帮助，但是没有可用的帮助
OLECMDERR_CANCELED	使用者取消了执行

参数

nCmdID

待执行命令的标识符。必须是在由 pguidCmdGroup 定义的组中。

nCmdExecOpt

指明任务执行选项。缺省情况下设置为执行任务时不给用户提示。参见 OLECMDEXE-COPT 可获得取值的列表。

pguidCmdGroup

命令组的特有标识符。缺省情况下是 NULL，这表明是标准组。由 nCmdID 传递的命令必须是属于这个组的。

说明

调用这个成员函数执行由用户指定的命令。

参数 *pguidCmdGroup* **和** *nCmdID* **一起唯一确定了要激活的命令。参数** *nCmdExecOpt* **确定了需要执行的确切的动作。**

请参阅 IOleCommandTarget::Exec

COleDocObjectItem::GetActiveView

LPOLEDOCUMENTVIEW GetActiveView() const;

返回值

指向当前活动视窗的 IOleDocumentView **接口的指针。如果当前没有活动视窗，则返回** NULL。

说明

调用这个成员函数可以获得指向当前活动视窗的 IOleDocumentView **接口的指针。有关这个返回的** IOleDocumentView **指针的引用计数在函数返回前不会增加。**

COleDocObjectItem::GetPageCount

BOOL GetPageCount(LPLONG *pnFirstPage*，LPLONG *pcPages*);

返回值

成功则返回非零值；否则返回 0。

参数

pnFirstPage

指向文档的第一页的号码的指针。可以是 NULL，这表明调用者不需要这个号码。

pcPages

指向文档的总页数的指针。可以是 NULL，这表明调用者不需要这个数字。

说明

调用这个成员函数可以获得文档中的页数。

请参阅 IPrint::GetPageInfo

COleDocObjectItem::OnPreparePrinting

static BOOL OnPreparePrinting(CView* *pCaller*,CPrintInfo* *pInfo*,BOOL *bPrintAll* = TRUE);

返回值

如果成功则返回非零值；否则返回 0。

参数

pCaller

指向 CView **对象的指针，该对象用来发送打印命令。**

pInfo

指向 CPrintInfo **对象的指针，该对象用来描述要打印的作业。**

bPrintAll

指明是否是整个文档都要打印。

说明

框架调用这个成员函数来准备一个要打印的文档。

请参阅 COleDocObjectItem::OnPrint

COleDocObjectItem::OnPrint

static void OnPrint(CView* *pCaller*,CPrintInfo* *pInfo*,BOOL *bPrintAll* = TRUE);

参数

pCaller

指向 CView **对象的指针，该对象用来发送打印命令。**

pInfo

指向 CPrintInfo **对象的指针，该对象用来描述要打印的作业。**

bPrintAll

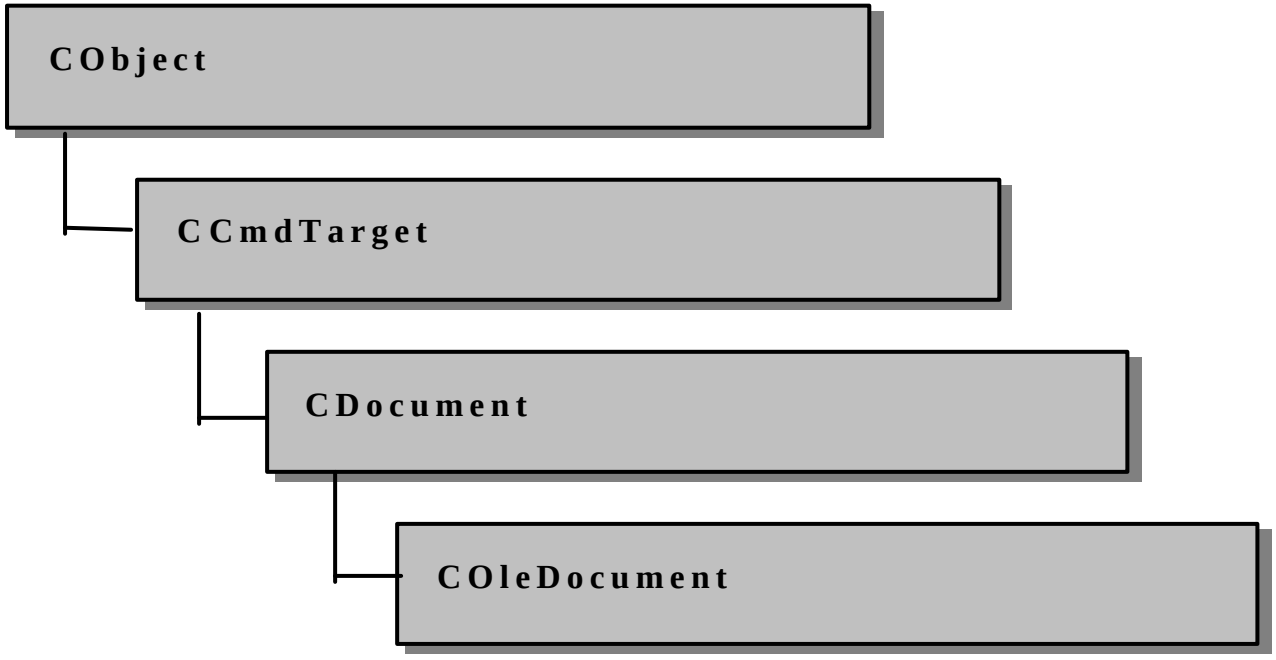
指明是否是整个文档都要打印。

说 明

框架调用这个成员函数来打印一个文档。

请参阅 COleDocObjectItem::OnPreparePrinting

COleDocument



COleDocument 是 OLE 文档的基类，它支持可视化编辑。

COleDocument 从 CDocument 派生来，这使你的 OLE 应用程序能够使用由微软基础类库提供的文档/视结构。

COleDocument 将文档当作一个处理 OLE 项的 CDocItem 对象的集合。容器和服务器应用都需要这样的结构，因为它们的文档必须能够包容 OLE 项。类 COleServerItem 和类 COleClientItem 都从 CDocItem 派生而来，它们管理应用程序与 OLE 项之间的相互作用。

如果你正在编写一个简单的容器应用程序，最好从 COleDocument 派生出你的文档类。如果你正在编写一个支持链接到文档包含的嵌入项的容器应用程序，最好从 COleLinkingDoc 派生出你的文档类。如果你正在编写一个服务器应用程序或者是组合容器/服务器，最好从 COleServerDoc 派生出你的文档类。COleLinkingDoc 和 COleServerDoc 是从 COleDocument 派生而来的，因此这些

类继承了所有在 COleDocument 和 CDocument 中具有的服务。

要使用 COleDocument，可以从它派生出一个类，并添加一些功能用来管理应用中除嵌入或链接项外的其它非 OLE 数据。如果你将从 CDocItem 派生而来的类定义为用来保存应用的内部数据，你可以使用由 COleDocument 定义的工具来保存你的 OLE 数据和非 OLE 数据。也可以设计你自己的数据结构来保存你的非 OLE 数据，以区别于你的 OLE 项。更多的信息，参见“Visual C++程序员指南”一书中的文章“容器：复合文件”。

如果提供了邮件支持（MAPI）的话，CDocument 支持将你的文档通过邮件发送。COleDocument 已经更新了 OnFileSendMail，可以正确处理捆绑在一起的文档。更多的信息，参见“Visual C++程序员指南”一书中的文章“MAPI 主题”和“MFC 中的 MAPI 支持”。

```
#include <afxole.h>
```

COleDocument 类成员

Construction	
ColeDocument	构造一个 COleDocument 对象

Operations	
HasBlankItems	检查文档中的空项
EnableCompoundFile	使文档能够被用 OLE 结构保存文件格式来保存
GetInPlaceActiveItem	返回当前正处于活动状态的 OLE 项
GetStartPosition	获得开始重复的初始位置
GetNextItem	获取下一个要重复的文档项
GetNextClientItem	获取下一个要重复的客户项
GetNextServerItem	获取下一个要重复的服务器项
UpdateModifiedFlag	如果任何一个被包容的 OLE 项被修改，则标记该文档为已被修改
ApplyPrintDevice	为文档中的所有客户项设置打印目标设备
AddItem	向文档保持的项列表中添加一个项
RemoveItem	从文档保持的项列表中移走一个项

Overridables	
GetPrimarySelectedItem	返回文档中最初选择的 OLE 项
OnShowViews	在文档变为可见和不可见时被调用
Mail Function	
OnFileSendMail	发送一个附着有文档的邮件消息
Message Handlers	
OnEditChangeIcon	处理 Change Icon 菜单命令中的事件
OnEditConvert	处理一个嵌入或链接对象从一种类型到另一种类型的转换
OnEditLinks	处理 Edit 菜单上的 Links 命令中的事件
Message Handlers	
OnUpdateEditChangeIcon	由框架调用来更新 Edit/Change Icon 菜单选项的命令 UI
OnUpdateEditLinksMenu	由框架调用来更新 Edit/Links 菜单选项的命令 UI
OnUpdateObjectVerbMenu	由框架调用来更新 Edit/Object Name 菜单选项和从 Edit/ObjectName 访问的 Verb 子菜单的命令 UI

Message Handlers	
OnUpdatePasteLinkMenu	由框架调用来更新 Paste Special 菜单选项的命令 UI
OnUpdatePasteMenu	由框架调用来更新 Paste 菜单选项的命令 UI

成员函数

COleDocument::AddItem

virtual void AddItem(CDocItem* *pItem*);

参数

pItem

指向要增加的文档项的指针。

说明

调用这个函数可以向文档中增加一个项。当它被 COleClientItem **或者** COleServerItem **的构造函数调用时，你不必显式地调用这个函数，上述的构造函数接收一个指向文档的指针。**

请参阅 CDocItem **，** COleDocument::RemoveItem **，** COleServerItem::COleServerItem**，**
COleClientItem::COleClientItem

COleDocument::ApplyPrintDevice

BOOL ApplyPrintDevice(const DVTARGETDEVICE FAR* *ptd*);
BOOL ApplyPrintDevice(const PRINTDLG* *ppd*);

返回值

如果函数执行成功则返回非零值，否则返回 0。

参数

ptd

指向一个 DVTARGETDEVICE 数据结构的指针，该数据结构包含有关新的目标打印设备的信息。它可以为 NULL。

ppd

指向一个 PRINTDLG 数据结构的指针，该数据结构包含有关新的目标打印设备的信息。它可以为 NULL。

说明

调用这个函数可以为你应用程序中的容器文档中的所有嵌入 COleClientItem 项

改变目标打印设备。此函数为所有的项更新目标打印设备，但是不为这些项刷新显示高速缓存。要更新一个项的显示高速缓存，可以调用COleClientItem::UpdateLink。

送往这个函数的参数包含了 OLE 用来标识目标设备的信息。PRINTDLG 结构包含了 Windows 用来初始化通用打印对话框的信息。在用户关闭了对话框之后，Windows 返回用户在这个结构中选择的信息。一个 CPrintDialog 对象的成员 m_pd 是一个 PRINTDLG 结构。

更多的信息，参见 Win32 SDK 文档中的 PRINTDLG 结构。

进一步的信息，参见“OLE 2 程序员参考，卷 1”中的 DVTARGETDEVICE 结构。

请参阅 CPrintDialog

COleDocument::COleDocument

COleDocument();

说明

构造一个 COleDocument **对象。**

COleDocument::EnableCompoundFile

void EnableCompoundFile(BOOL bEnable = TRUE);

参数

bEnable

指明是否能够进行复合文件支持。

说明

如果你希望用复合文件格式来保存文档，请调用这个函数。这也被称作结构化存储。调用这个函数的典型做法是在 COleDocument 派生类的构造函数中进行。有关复合文档的更进一般信息，参见“Visual C++程序员指南”一书中的文章“容器：复合文件”。

在为文档设置好是否能够支持复合文件之后，则在文档的整个生命周期中这个设置是不能改变的。

请参阅 COleClientItem

COleDocument::GetInPlaceActiveItem

COleClientItem* GetInPlaceActiveItem(CWnd* pWnd);

返回值

指向单一的、正处于活动状态的 OLE **项的指针；如果当前没有正处于活动状态的** OLE **项，则返回值为** NULL**。**

参数

pWnd

指向显示容器文档的窗口的指针。

说明

调用这个函数可以获得框架窗口中正处于活动状态的 OLE **项，该框架窗口包含有由** pWnd **标识的视图。**

请参阅 COleClientItem

COleDocument::GetNextClientItem

COleClientItem* GetNextClientItem(POSITION& *pos*) const;

返回值

指向文档中下一个客户项的指针，如果没有更多的客户项则返回 NULL。

参数

pos

是对上一次调用 GetNextClientItem **设置的** POSITION **值的引用；其初值是成员函数** GetStartPosition **返回的。**

说明

调用这个函数可以重复地访问你的文档中的每一个客户项。每一次调用之后，

为文档中的下一个项设置了 pos 的值，这些项有可能是也有可能不是一个客户项。

示例

// COleDocument::GetNextClientItem 示例

```
// pDoc points to a COleDocument object
POSITION pos = pDoc->GetStartPosition();
COleClientItem *pItem;
while((pItem = pDoc->GetNextClientItem(pos)) != NULL)
{
   // Use pItem
}
```

请参阅 COleClientItem，COleDocument::GetStartPosition，COleDocument::GetNextServerItem，COleDocument::GetNextItem

COleDocument::GetNextItem

virtual CDocItem* GetNextItem(POSITION& *pos*) const;

返回值

指向在指定位置的文档项的指针。

参数

pos

是上一次调用 GetNextItem **设置的** POSITION **值的引用；其初值是成员函数** GetStartPosition **的返回值。**

说明

调用这个函数可以重复访问你的文档中的每一个项。在每一次调用之后，pos

的值被设置为文档中下一个项的 POSITION 值。如果被获取的元素是文档中的最后一个元素，则 pos 的新值为 NULL。

示例

```
// Example for COleDocument::GetNextItem
// pDoc points to a COleDocument object
POSITION pos = pDoc->GetStartPosition();
CDocItem *pItem;
while( pos != NULL )
{
   pItem = pDoc->GetNextItem( pos );
   // Use pItem
}
```

请参阅 COleDocument::GetStartPosition，COleDocument::GetNextClientItem，COleDocument::GetNextServerItem

COleDocument::GetNextServerItem

COleServerItem* GetNextServerItem(POSITION& *pos*) const;

返回值

指向文档中下一个服务器项的指针，如果没有更多的项则返回 NULL。

参数

pos

是上一次调用 GetNextServerItem **设置的** POSITION **值的引用；其初值是成员函数** GetStartPosition **的返回值。**

说明

调用这个函数可以重复访问你的文档中的每一个服务器项。在每一次调用之

后，pos 的值为文档中的下一个项而被设置，这个项可能是也可能不是一个服务器项。

示例

```
// Example for COleDocument::GetNextServerItem
// pDoc points to a COleDocument object
POSITION pos = pDoc->GetStartPosition();
COleServerItem *pItem;
while( ( pItem = pDoc->GetNextServerItem( pos ) ) != NULL )
{
   // Use pItem
}
```

请参阅　COleServerItem，COleDocument::GetStartPosition，COleDocument::GetNextClientItem，COleDocument::GetNextItem

COleDocument::GetPrimarySelectedItem

virtual COleClientItem* GetPrimarySelectedItem(CView* *pView*);

返回值

指向一个单一的已被选择的 OLE **项 的指针；如果没有或者没有多于一个** OLE **项被选择则返回** NULL。

参数

pView

指向显示在文档中的活动视图对象的指针。

说明

由框架调用此成员函数来获取指定视中的当前已被选择的 OLE **项。函数的缺**

省实现搜索被包含的 OLE 项列表，以获得一个单一的被选择的项，并返回指向这个项的指针。如果没有被选中的项，或者没有多于一个的被选择的项，则函数返回 NULL。为了使这个函数工作，你必须在你的视类中重载成员函数 CView::IsSelected。如果你有用于保存被包含 OLE 项的自己的方法，请重载这个函数。

请参阅 CView::IsSelected

COleDocument::GetStartPosition

```
virtual POSITION GetStartPosition() const;
```

返回值

一个 POSITION 值，可以用来在整个文档的项中开始重复；如果文档没有项则返回值为 NULL。

说明

调用这个函数可以获得文档中第一个项的位置。可以将返回值传递给 GetNextItem，GetNextClientItem，**或者是** GetNextServerItem。

请参阅 COleDocument::GetNextItem，COleDocument::GetNextClientItem，COleDocument::GetNextServerItem

COleDocument::HasBlankItems

```
BOOL HasBlankItems() const;
```

返回值

如果文档包含任何空项则返回非零值；否则返回 0。

说明

调用这个函数可以确定文档是否包含空项。一个空项是一个其矩形为空的项。

请参阅 CDocItem::IsBlank

COleDocument::OnEditChangeIcon

afx_msg void OnEditChangeIcon();

说明

显示 OLE Change Icon **对话框，并将代表当前选定的** OLE **项的图标改变为用户在对话框中选定的图标。** OnEditChangeIcon **创建并给出一个** COleChangeIconDialog **改变图标对话框。**

请参阅 COleDocument::OnUpdateEditChangeIcon，COleChangeIconDialog

COleDocument::OnEditConvert

afx_msg void OnEditConvert();

说明

显示 OLE Convert **对话框，并根据用户在对话框中的选择转换或激活被选定的** OLE **项。**OnEditConvert **创建并推出一个** COleConvertDialog **转换对话框。**

有关转换的一个例子就是将一个 Microsoft **的** Word **文档转换为一个** WordPad **文档。**

请参阅 COleDocument::OnUpdateObjectVerbMenu，COleConvertDialog

COleDocument::OnEditLinks

afx_msg void OnEditLinks();

说明

显示 OLE Edit/Links 对话框。OnEditLinks 创建并推出一个 COleLinksDialog 链接对话框，让用户改变被链接的对象。

请参阅 COleDocument::OnUpdateEditLinksMenu，COleLinksDialog

COleDocument::OnFileSendMail

afx_msg void OnFileSendMail();

说明

通过一个存在的邮件主机（任何一个）发送一个消息，将文档作为一个附着物。OnFileSendMail 调用 OnSaveDocument 将一个无标题的或已被修改的文档保存到一个临时文件中，这个文件将被通过电子邮件发送。如果文档没有被修改，

则不需要一个临时文件；被发送的是原始的文档。如果 MAPI32.DLL 没有被载入，则 OnFileSendMail 会载入它。

与 OnFileSendMail 在 CDocument 中的实现不同，这个函数正确处理复合文件。

更多的消息，参见“Visual C++程序员指南”一书中的文章“MAPI 主题”和“MFC 中的 MAPI 支持”。

请参阅 CDocument::OnFileSendMail，CDocument::OnUpdateFileSendMail，CDocument::OnSaveDocument

COleDocument::OnShowViews

virtual void OnShowViews(BOOL *bVisible*);

参数

bVisible

表明文档是否是可见的。

说明

在文档的可见性状态改变后，框架就调用这个函数。

这个函数的缺省版本什么也不做。如果当文档的可见性改变时你的应用程序必须执行任何特别的处理，那么你就重载它。

COleDocument::OnUpdateEditChangeIcon

afx_msg void OnUpdateEditChangeIcon(CCmdUI* *pCmdUI*);

参数

pCmdUI

指向一个代表产生更新命令的菜单的 CCmdUI 结构的指针。更新处理者

通过 *pCmdUI* 来调用 CCmdUI 结构的 Enable 成员函数来更新用户界面。

说明

框架调用这个函数来更新 Edit 菜单的 Change Icon 命令。OnUpdateEditChange 基于在文档中是否存在一个有效的图标来更新命令的用户界面。重载这个函数可以改变这些功能。

请参阅 ColeDocument::OnEditChangeIcon,CCmdUI

COleDocument::OnUpdateEditLinksMenu

afx_msg void OnUpdateEditLinksMenu(CCmdUI* *pCmdUI*);

参数

pCmdUI

指向一个代表产生更新命令的菜单的 CCmdUI 结构的指针。更新处理者通过 *pCmdUI* 来调用 CCmdUI 结构的 Enable 成员函数来更新用户界面。

说明

框架调用这个函数来更新 Edit 菜单的 Change Icon 命令。从文档中的第一个 OLE 项开始，OnUpdateEditLinksMenu 访问每一项，测试它们是否是一个链接，并且，如果是一个链接，则使能 Links 命令。重载这个函数可以改变这种情况。

请参阅 COleDocument::OnEditLinks，COleDocument::GetStartPosition，COleDocument::GetNextClientItem，CCmdUI

COleDocument::OnUpdateObjectVerbMenu

```
afx_msg void OnUpdateOjectVerbMenu( CCmdUI* pCmdUI );
```

参数

pCmdUI

指向一个代表产生更新命令的菜单的 CCmdUI 结构的指针。更新处理程序通过 pCmdUI来调用 CCmdUI结构的 Enable成员函数来更新用户界面。

说明

框架调用这个函数来更新 Edit 菜单的 ObjectName 命令以及从 ObjectName 命令访问的 Verb 子菜单，在此，ObjectName 是文档中嵌入 OLE 对象的名字。OnUpdateObjectVerbMenu 基于文档中是否存在一个有效的对象来更新 ObjectName 命令的用户界面。如果一个对象存在，则 Edit 菜单中的 ObjectName 命令是有效的。当选择了这个菜单命令时，Verb 子菜单就被显示出来了。Verb 子菜单包含了所有对这个对象来说可以使用的动词命令，比如说 Edit，Properties

等等。重载这个函数可以改变这种情况。

请参阅 COleDocument::OnEditConvert，CCmdUI

COleDocument::OnUpdatePasteLinkMenu

afx_msg void OnUpdatePasteLinkMenu(CCmdUI* *pCmdUI*);

参数

pCmdUI

指向一个代表产生更新命令的菜单的 CCmdUI 结构的指针。更新处理者通过 *pCmdUI* 来调用 CCmdUI 结构的 Enable 成员函数来更新用户界面。

说明

框架调用这个函数来确定是否能够从剪贴板粘贴一个链接的 OLE 项。Paste

Special 菜单命令是否有效决定于这个项是否能够粘贴到文档中。

请参阅 COleDocument::OnUpdatePasteMenu，CCmdUI

COleDocument::OnUpdatePasteMenu

afx_msg void OnUpdatePasteMenu(CCmdUI* *pCmdUI*);

参数

pCmdUI

指向一个代表产生更新命令的菜单的 CCmdUI 结构的指针。更新处理者通过 *pCmdUI* 来调用 CCmdUI 结构的 Enable 成员函数来更新用户界面。

说明

框架调用这个函数来确定是否能够从剪贴板粘贴一个链接的 OLE 项。Paste 菜

单命令和按钮是否有效决定于这个项是否能够粘贴到文档中。

请参阅 COleDocument::OnUpdatePasteLinkMenu，CCmdUI

COleDocument::RemoveItem

virtual void RemoveItem(CDocItem* *pItem*);

参数

pItem

指向被移动的文档项的指针。

说明

调用这个函数可以从文档中删除一个项。通常你不用显式地调用这个函数；它由 COleClientItem 和 COleServerItem 的析构函数调用。

请参阅 COleServerItem，COleClientItem，COleDocument::AddItem，CDocItem

COleDocument::UpdateModifiedFlag

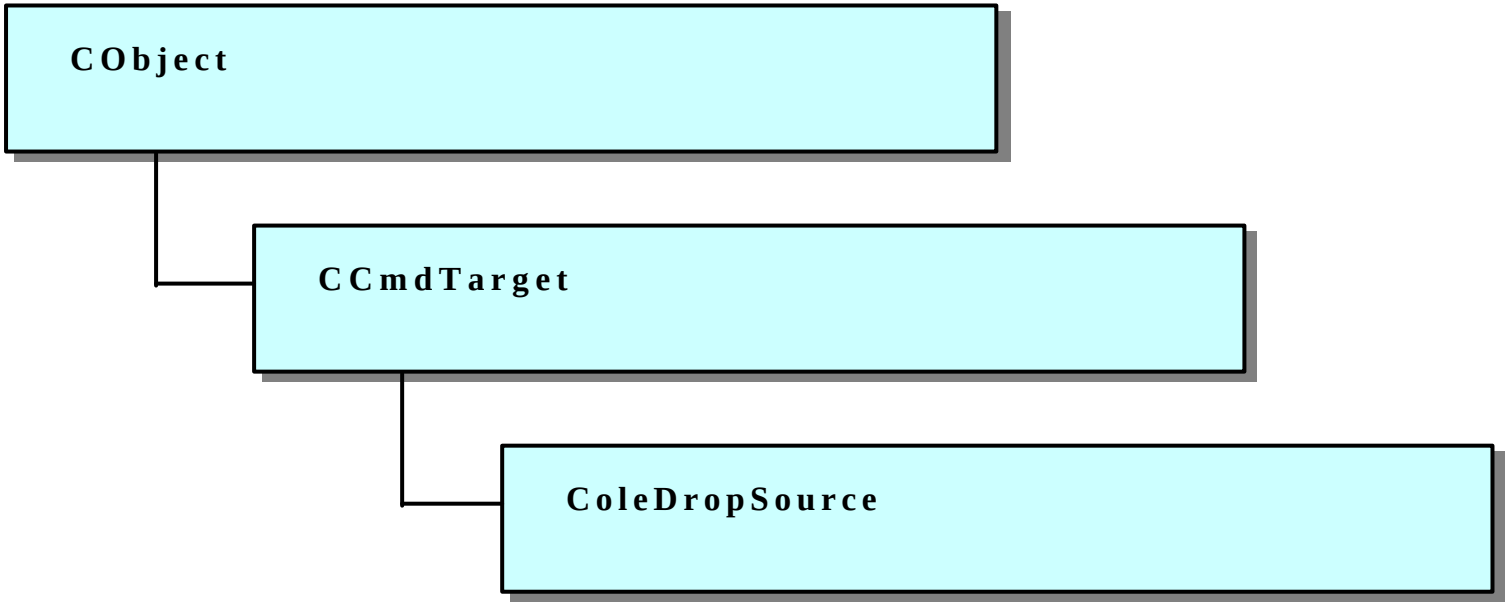
void UpdateModifiedFlag();

说明

如果文档中包含的 OLE **项被修改了，调用这个函数可以将文档标记为被修改的。这就使得框架在文档关闭前提示用户保存文档，即使就是在文档的内部数据没有改变时也可以这样。**

请参阅 CDocument::SetModifiedFlag，COleClientItem::IsModified

COleDropSource



一个 COleDropSource 对象允许将数据拖动到一个拖放目标中。类 COleDropTarget 处理从拖放操作中获得的数据。COleDropSource 对象负责决定何时开始一次拖动操作，并在拖动操作中提供反馈，并决定何时结束一次拖动。只要调用相应的构造函数就可以使用一个 COleDropSource 对象。这简化了确

定是何种事件的过程，比如一次鼠标的点击，用函数 COleDataSource::DoDragDrop，COleClientItem:: DoDragDrop，或者 COleServerItem::DoDragDrop 开始一次拖动操作。这些函数将为你创建一个 COleDropSource 对象。你也许希望修改 COleDropSource 可重载函数的缺省行为。这些成员函数将在适当的时候由框架来调用。

有关使用 OLE 进行拖放操作的更多信息，参见“Visual C++程序员指南”一书中的“拖放(OLE)”。

更多的信息，参见“OLE2 程序员参考，卷 1”中的 IDropSource。

```
#include <afxole.h>
```

COleDropSource 类成员

construction	
COleDropSource	构造一个 COleDropSource 对象
Overridables	
GiveFeedback	在拖放操作中改变光标
OnBeginDrag	在拖动操作中处理鼠标的动作
QueryContinueDrag	查看是否可以继续拖动

成员函数

COleDropSource::COleDropSource

COleDropSource();

说明

构造一个 COleDropSource 对象。

请参阅 COleDropTarget

COleDropSource::GiveFeedBack

virtual SCODE GiveFeedback(DROPEFFECT *dropEffect*);

返回值

如果拖动在进行中，则返回 DRAGDROP_S_USEDEFAULTCURSORS，**如果不在进行中则返回** NOERROR。

参数

dropEffect

你希望显示给用户的效果，它常常表明如果在这一点上放开被选择的数据将会发生什么。通常来说，它是最近调用的 CView::OnDragEnter **或者** CView::OnDragOver **的返回值。它可以是下列值中的一个或几个：**

- DROPEFFECT_NONE 不允许放开操作。
- DROPEFFECT_COPY 将执行一次拷贝操作。
- DROPEFFECT_MOVE 将执行一次移动操作。
- DROPEFFECT_LINK 将建立一个从放开的数据到原始数据的链接。
- DROPEFFECT_SCROLL 在目标中一次拖动滚动条的操作将要发生或正在发生。

说明

在调用 COleDropTarget::OnDragOver 或者 COleDropTarget::DragEnter 后由 framework 调用这个函数。重载这个函数可以向用户提供反馈，及如果在这点放开会发生什么。缺省的实现使用 OLE 缺省光标。有关使用 OLE 进行拖放操作的信息，参见“Visual C++程序员向导”一书的文章“拖放(OLE)”。

更多的信息，参见“OLE 2 程序员参考，卷 1”中的 IDropSource::GiveFeedback，IDropTarget::DragOver 和 IDropTarget::DragEnter。

请参阅 CView::OnDragEnter, CView::OnDragOver

COleDropSource::OnBeginDrag

virtual BOOL OnBeginDrag(CWnd* *pWnd*);

返回值

如果允许拖动则返回非零值，否则返回 0。

参数

pWnd

指向包含被选中数据的窗口的指针。

说明

当开始拖动操作的事件（如按下鼠标左键）发生时，由框架调用此函数。如果你想修改拖动处理开始的方法，你可以重载这个函数。缺省的实现监视鼠标的动作并保持在拖动模式，直到用户点击鼠标的左键或右键，或者敲击 ESC **键，这时它释放鼠标。**

请参阅 COleDropSource::GiveFeedback

COleDropSource::QueryContinueDrag

virtual SCODE QueryContinueDrag(BOOL *bEscapePressed*,DWORD *dwKeyState*);

返回值

在拖动开始之前，如果按下 ESC **键或者鼠标右键或放开鼠标左键，则返回**

DRAGDROP_S_ CANCEL。如果要发生一次放开操作则返回DRAGDROP_S_DROP，否则返回S_OK。

参数

bEscapePressed

自最后一次调用COleDropSource::QueryContinueDrag后，ESC键被按下的状态。

dwKeyState

记录键盘上的修改键的状态。这是下列任何一个组合：MK_CONTROL，MK_SHIFT，MK_ALT，MK_LBUTTON，MK_MBUTTON和MK_RBUTTON。

说明

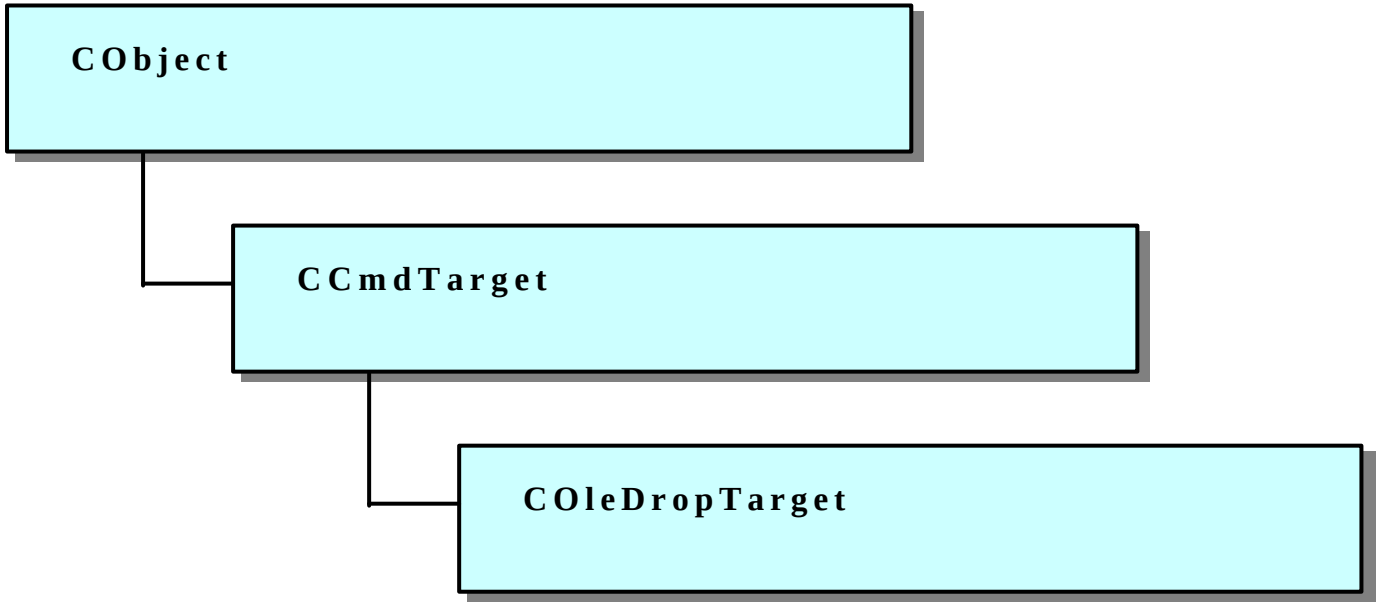
在拖动开始后，这个函数被框架重复调用直到拖动操作被取消或者结束。如果你想改变拖动的取消点或发生一次放开操作，你可以重载这个函数。

缺省的实现方式按下面的描述启动放开或取消一次拖动。当 ESC 键或鼠标右键被按下时，就取消一次拖动操作。在拖动开始之后，当鼠标左键被放起时，就启动一次放开操作。否则，此函数返回 S_OK 并不再执行进一步的操作。

由于这个函数被频繁地调用，所以要尽可能地对它进行优化。

请参阅 COleDropSource::OnBeginDrag，COleDropTarget::OnDrop

COleDropTarget



一个 COleDropTarget 对象提供窗口与 OLE 库之间的通讯机制。创建这个类的对象之后，窗口就可以通过 OLE 拖放机制接收数据。

为了使一个窗口能够接收拖放来的命令，就必须首先创建一个 COleDropTarget 类对象，然后调用 Register 函数，该函数只有一个参数，该参数是一个指向所

需要的 CWnd 对象的指针。

有关使用 OLE 进行拖放操作的更多信息，参见“ Visual C++程序员指南”一书中的文章“ 拖放(OLE)”。

#include <afxole.h>

请参阅 COleDropSource

COleDropTarget 类成员

Contstruction	
ColeDropTarget	构造一个 COleDropTarget 对象

Operations	
Register	将一个窗口注册为一个有效的拖放目标
Revoke	使窗口不再是一个有效的拖放目标
Overridables	
OnDragEnter	当光标第一次进入窗口时被调用
OnDragLeave	当光标被拖动出窗口时被调用
OnDragOver	当光标在窗口中拖动时被重复调用
OnDragScroll	用来确定光标是否被拖进了窗口的滚动条区域
OnDrop	当数据被放在窗口中时被调用，缺省的处理
OnDropEx	当数据被放在窗口中时被调用，最初的处理

成员函数

COleDropTarget::COleDropTarget

COleDropTarget();

说明

构造一个 COleDropTarget **类对象。调用** Register **来使这个对象与窗口产生联系。**

请参阅 COleDropSource，COleDropTarget::Register，COleDropTarget::Revoke

COleDropTarget::OnDragEnter

virtual DROPEFFECT OnDragEnter(CWnd* *pWnd*,COleDataObject* *pDataObject*,
DWORD *dwKeyState*,CPoint *point*);

返回值

如果在 point **指定的位置尝试放开会产生的影响。它可以是下列中的一个或几个：**

- DROPEFFECT_NONE **不允许进行一次放开。**
- DROPEFFECT_COPY **将执行一次拷贝操作。**

- DROPEFFECT_MOVE　将执行一次移动操作。
- DROPEFFECT_LINK　将建立一个从被放开的数据到原始数据的链接。
- DROPEFFECT_SCROLL　在目标中已发生或将要发生一次拖动滚动条的操作。

参数

pWnd

指向光标正进入的窗口。

pDataObject

指向包含可以放开的数据的数据对象。

dwKeyState

包含修饰键的状态。是下列的一些组合：MK_CONTROL，MK_SHIFT，MK_ALT，MK_LBUTTON，MK_MBUTTON 和 MK_RBUTTON。

point

记录光标在客户坐标系中的当前位置。

说明

当光标首次被拖入窗口时由应用框架调用。重载这个函数可以使得在窗口中可以进行放开操作。缺省的实现调用了 CView::OnDragEnter，它只是简单地缺省返回 DROPEFFECT_NONE。

更多的信息请参见“OLE 2 程序员参考，卷 1”中的 IDropTarget::DragEnter。

请参阅 COleDropTarget::OnDragOver，COleDropTarget::OnDragLeave，COleDropTarget::OnDrop，COleDropTarget::OnDropEx，CView::OnDragEnter

COleDropTarget::OnDragLeave

virtual void OnDragLeave(CWnd* *pWnd*);

参数

pWnd

指向光标正离开的窗口。

说明

当一次拖动操作正在起作用，同时光标正离开窗口时，由应用框架调用。当拖动操作离开指定的窗口时，如果你想有一些特别的动作，请你重载这个函数。这个函数的缺省实现调用了 CView::OnDragLeave。

更多的信息请参见“ OLE 2 **程序员参考，卷** 1**”中的** IDropTarget::DragEnter。

请参阅 COleDropTarget::OnDragEnter，COleDropTarget::OnDragOver，

COleDropTarget::OnDrop，COleDropTarget::OnDropEx

COleDropTarget::OnDragOver

virtual DROPEFFECT OnDragOver(CWnd* *pWnd* , COleDataObject* *pDataObject*,

DWORD *dwKeyState*，CPoint *point*);

返回值

返回如果在 point **指定的位置尝试放开会产生的影响。它可以是下列中的一个或几个：**

- DROPEFFECT_NONE **不允许进行一次放开。**
- DROPEFFECT_COPY **将执行一次拷贝操作。**

- DROPEFFECT_MOVE　将执行一次移动操作。
- DROPEFFECT_LINK　将建立一个从放开的数据到原始数据的链接。
- DROPEFFECT_SCROLL　在目标中已发生或将要发生一次拖动滚动条的操作。

参数

pWnd

指向光标正通过的窗口。

pDataObject

指向包含可以放开的数据的数据对象。

dwKeyState

包含修饰键的状态。这是下列的一些组合：MK_CONTROL，MK_SHIFT，MK_ALT，MK_LBUTTON，MK_MBUTTON 和 MK_RBUTTON。

point

记录光标在客户坐标系中的当前位置。

说明

当光标通过窗口时由应用框架调用。为了能够在窗口中进行放开操作，你必须重载这个函数。这个函数的缺省实现调用了 CView::OnDragOver，它缺省地返回 DROPEFFECT_ NONE。由于这个函数被频繁地调用，所以最好能够尽可能的对它进行优化。

更多的信息请参见“OLE 2 程序员参考，卷 1”中的 IDropTarget::DragOver。

请参阅 COleDropTarget::OnDragEnter，COleDropTarget::OnDragLeave，COleDropTarget::OnDrop，COleDropTarget::OnDropEx

COleDropTarget::OnDragScroll

virtual DROPEFFECT OnDragScroll(CWnd* *pWnd*, DWORD *dwKeyState*, CPoint *point*);

返回值

如果在 point **指定的位置尝试放开会产生的影响。它可以是下列中的一个或几个：**

- DROPEFFECT_NONE **不允许进行一次放开。**
- DROPEFFECT_COPY **将执行一次拷贝操作。**
- DROPEFFECT_MOVE **将执行一次移动操作。**
- DROPEFFECT_LINK **将建立一个从放开的数据到原始数据的链接。**
- DROPEFFECT_SCROLL **在目标中已发生或将要发生一次拖动滚动条的**

操作。

参数

pWnd

指向光标正通过的窗口。

dwKeyState

包含修饰键的状态。这是下列的一些组合：MK_CONTROL，MK_SHIFT，MK_ALT，MK_LBUTTON，MK_MBUTTON 和 MK_RBUTTON。

point

以像素为单位记录光标在屏幕上的相对位置。

说明

在调用 OnDragEnter 或者 OnDragOver 之前由框架调用，用来确定 point 指定的

位置是否在滚动条区域内。当你希望这个事件具有特别的行为时，请重载这个函数。这个函数的缺省实现调用了 CView::OnDragScroll，它返回 DROPEFFECT_NONE，并且当光标进入窗口边界内的缺省滚动条区域时滚动窗口。

COleDropTarget::OnDrop

virtual BOOL OnDrop(CWnd* *pWnd*， COleDataObject* *pDateObject*，

DROPEFFECT *dropEffect*， CPoint *point*);

返回值

如果放开成功则返回非零值；否则返回 0。

参数

pWnd

指向光标正通过的窗口。

pDataObject

指向包含可以放开的数据的数据对象。

dropEffect

是用户为放开操作选择的效果。它可以是下列中的一个或几个：

- DROPEFFECT_COPY **将执行一次拷贝操作。**
- DROPEFFECT_MOVE **将执行一次移动操作。**
- DROPEFFECT_LINK **将建立一个从放开的数据到原始数据的链接。**

point

以像素为单位记录光标在屏幕中的相对位置。

说明

当一次放开操作将要发生时由框架调用。框架首先调用 OnDropEx。如果 OnDropEx 函数并不处理放开操作，则框架就接着调用成员函数 OnDrop。一般来说，应用程序要在视类中重载 OnDropEx，以能处理鼠标右键的拖放操作。视类中的 OnDrop 常常是用来处理简单的拖放的。

COleDropTarget::OnDrop 的缺省实现调用了 CView::OnDrop，它只是简单的返回缺省值 FALSE。

更多的信息，参见“OLE 2 程序员参考，卷 1”中的 IDropTarget::Drop。

请参阅 COleDropTarget::OnDragOver，COleDropTarget::OnDragEnter，COleDropTarget::OnDropEx

COleDropTarget::OnDropEx

virtual DROPEFFECT OnDropEx(CWnd* *pWnd*, COleDataObject* *pDataObject*,

DROPEFFECT *dropDefault*, DROPEFFECT *dropList*, CPoint *point*);

返回值

返回当在 point **指定的位置进行放开操作时产生的效果。放开操作的影响将在说明部分进行讨论。**

参数

pWnd

指向光标正通过的窗口。

pDataObject

指向包含可以放开数据的数据对象。

dropDefault

用户选择的缺省放开操作的效果与当前的按键的状态有关。它可以是DROPEFFECT_NONE。放开操作的影响将在书面部分中进行讨论。

dropList

放开操作的数据源支持的效果列表。放开操作的效果值可以是用按位的或（|）操作来进行组合。放开操作的影响将在书面部分进行讨论。

point

以像素为单位记录光标在屏幕中的相对位置。

说明

当要发生一次放开操作的时候有框架调用。框架首先调用这个参数。如果它不处理这次放开操作，则框架接着就调用 OnDrop。通常，为了支持鼠标右键的拖放，你需要重载这个函数。一般来说，视类的 OnDrop 是用来处理简单的拖

放操作的。

COleDropTarget::OnDropEx 的缺省实现调用了 CView::OnDropEx。缺省的，CView:: OnDropEx 只简单地返回一个空值来表明必须调用 OnDrop 成员函数。

放开效果描述了与放开操作相关的动作。下面是放开效果的列表：

- DROPEFFECT_NONE 不允许放开操作。
- DROPEFFECT_COPY 将执行一次拷贝操作。
- DROPEFFECT_MOVE 将执行一次移动操作。
- DROPEFFECT_LINK 将建立一个从被放开的数据到原始数据的链接。
- DROPEFFECT_SCROLL 标识在目标中将发生或正在发生一次拖动滚动条的操作。

更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IDropTarget::Drop。

请参阅 COleDropTarget::OnDragOver，COleDropTarget::OnDragEnter

COleDropTarget::Register

BOOL Register(CWnd* *pWnd*);

返回值

如果注册成功则返回非零值；否则返回 0。

参数

pWnd

指向一个要被注册为放开目标的窗口。

说明

调用这个函数可以以 OLE DLL **将你的窗口注册为一个有效的放开目标。**

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 RegisterDragDrop。

请参阅 COleDropTarget::Revoke，COleDropTarget::COleDropTarget

COleDropTarget::Revoke

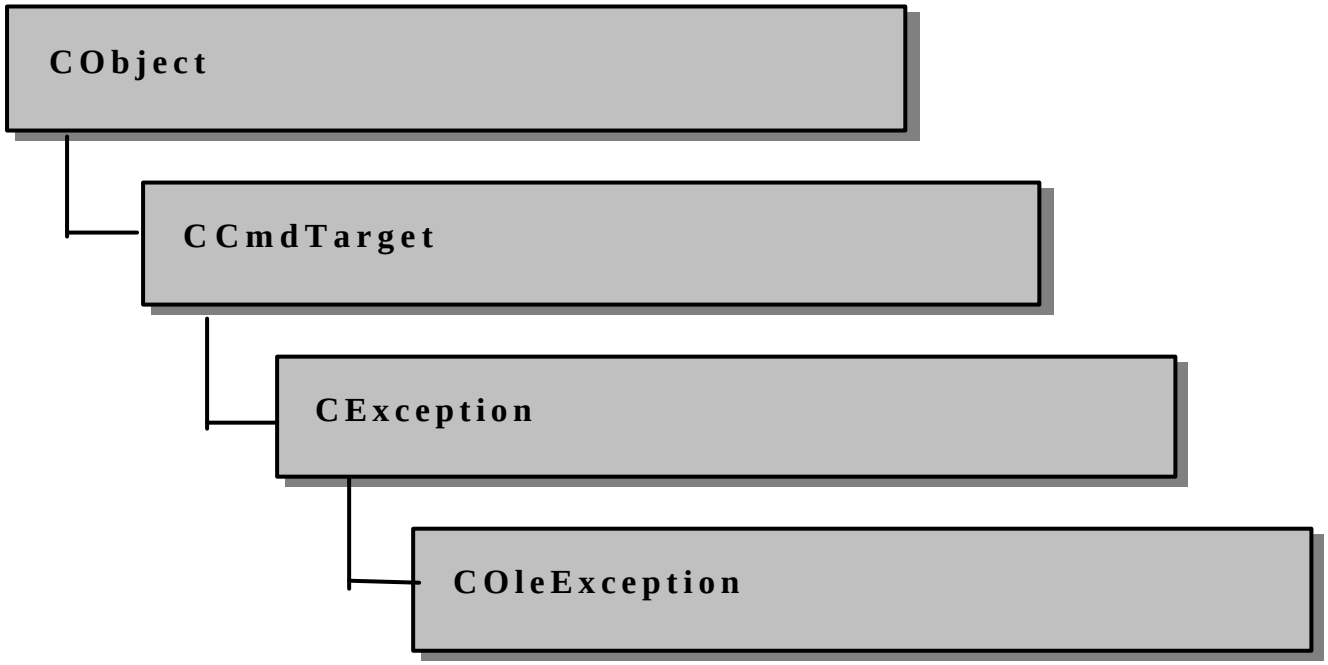
virtual void Revoke();

说明

在销毁任何一个已用 COleDropTarget::Register 调用注册为放开目标的窗口之前，调用 Revoke 函数将该窗口从放开目标列表中删掉。此函数在被注册窗口的 OnDestroy 处理函数中自动被调用，通常不需要显式地调用这个函数。

为了获取更多的信息，参见“OLE 2 程序员参考，卷 1”中的 RevokeDragDrop。

COleException



一个 COleException 对象代表的是与一个 OLE 操作相关的异常情况。此 COleException 类包括一个公用数据成员，该成员包含了用来表明此异常的原因的状态代码。

一般来说，应该直接创建一个 COleException 对象；或者可以调用 AfxThrowOleException 来代替。

有关异常的更多信息，参见“Visual C++程序员指南”中的文章“异常：OLE 异常”。

include <afxole.h>

COleException 类成员

Data Members	
m_sc	包含指明该异常的原因的状态码
Operations	
Process	将一个被捕获的异常翻译为一个 OLE 返回代码。

成员函数

COleException::Process

static SCODE PASCAL Process(const CException* *pAnyException*);

返回值

返回一个 OLE **状态代码。**

参数

pAnyException

指向一个被捕获的异常的指针。

说明

此成员函数用来将一个被捕获的异常转换为一个 OLE **错误代码。**

注意 **这个函数是一个静态函数。**

有关 SCODE **的更多信息，参见“** Platform SDK **”中的“** COM **错误代码”。**

请参阅 CException

数据成员

COleException::m_sc

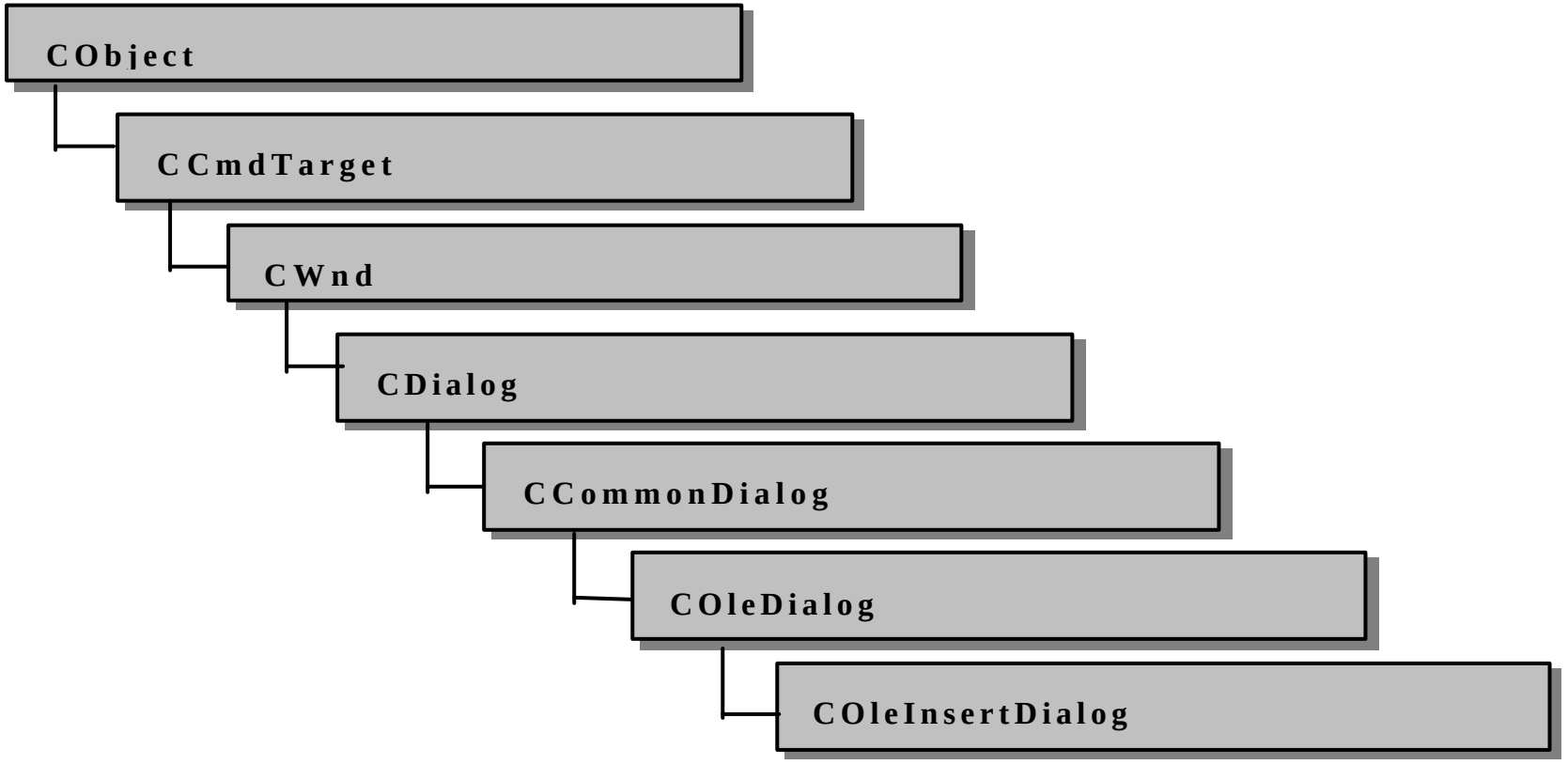
SCODE m_sc;

说明

此数据成员包含了表明异常的原因的 OLE **状态代码。这个变量的值由** AfxThrowOle-Exception **来设置。**

请参阅 AfxThrowOleException

COleInsertDialog



COleInsertDialog 类用于 OLE Insert Object 对话框。当你要调用这个对话框时，创建一个 COleInsertDialog 类对象。在构造了一个 COleInsertDialog 对象后，你就可以使用 m_io 结构来初始化对话框控件中的控件的值或状态。m_io 结构的类型是 OLEUINSERTOBJECT。有关使用这个对话框类的更多信息，参见 DoModal 成员函数。

注意 AppWizard 生成的容器代码使用了这个类。

更多的信息，参见“OLE 2.0 用户接口库”中的 OLEUINSERTOBJECT 结构。

#include <afxodlgs.h>

请参阅 COleDialog

COleInsertDialog 类成员

Data Members	
m_io	一个 OLEUIINSERTOBJECT 类型的结构，用来控制该对话框的行为
Construction	
ColeInsertDialog	构造一个 COleInsertDialog 对象
Operations and Attributes	
DoModal	显示该 OLE Insert Object 对话框
CreateItem	创建在该对话框中选择的项
GetSelectionType	获取被选择对象的类型
GetClassID	获取与被选择项关联的 CLSID
GetDrawAspect	确定是否将该项绘制为一个图标
GetIconicMetafile	获取一个与这个项的图标形式关联的图元文件的句柄
GetPathName	获取在该对话框中选择的文件的全路径

成员函数

COleInsertDialog::COleInsertDialog

COleInsertDialog (DWORD *dwFlags* = IOF_SELECTCREATENEW,
    CWnd* *pParentWnd* = NULL);

参数

dwFlags

创建标志，包含下列值的任意位或（OR）组合：

- IOF_SHOWHELP **指定当对话框被调用时，显示** Help **按钮。**
- IOF_SELECTCREATENEW **指定当对话框被调用时，**Create New **单选按钮将被初始选择。这是缺省的，并且不能与** IOF_SELECTCREATEFROMFILE **一起使用。**

- IOF_SELECTCREATEFROMFILE 指定当对话框则调用时，Create From File 当选按钮被初始选择。不能与 IOF_SELECTCREATENEW 一起使用。
- IOF_CHECKLINK 指定当对话框被调用时，Link 复选框将被初始核选。
- IOF_DISABLELINK 指定当对话框被调用时，Link 复选框将被初始核选。
- IOF_CHECKDISPLAYASICON 指定当对话框被调用时，Display As Icon 复选框将被初始核选，当前图标将被显示，而且 Change Icon 按钮将会有效。
- IOF_VERIFYSERVERSEXIST 指定该对话框将通过确保在对话框显示之前在注册数据库中指定的服务器已经存在来使由它添加到列表框

中的类有效。设置这个标志将将明显削弱性能。

pParentWnd

指向该对话框所属的父或宿主窗口对象（CWnd 类型）。如果它是 NULL，则对话框对象的父窗口被设置为应用程序主窗口。

说明

此成员函数只用来构造一个 COleInsertDialog 对象。要显示这个对话框，调用 DoModal 函数。

请参阅 COleInsertDialog::DoModal

COleInsertDialog::CreateItem

BOOL CreateItem(COleClientItem* *pItem*);

返回值

如果创建了项则返回非零值；否则返回 0。

参数

pItem

指向将要被创建的项。

说明

只有在 DoModal **返回** IDOK **时，此成员函数用来创建一个** COleClientItem **类型的对象。在你调用这个函数之前，你必须分配** COleClientItem **对象。**

请参阅 COleClientItem::CreateLinkFromFile, COleClientItem::CreateFromFile, COleClientItem::CreateNewItem, COleClientItem::SetDrawAspect, COleInsertDialog::GetSelectionType, COleInsertDialog::DoModal

COleInsertDialog::DoModal

virtual int DoModal( );
int DoModal(DWORD dwFlags);

返回值

返回对话框的完成状态。可以是下列值之一：

- IDOK **如果对话框被成功显示。**
- IDCANCEL **如果用户取消了对话框。**
- IDABORT **如果发生了一个错误。如果返回的是** IDABORT**，调用** COleDialog:: GetLastError **成员函数来获取更多有关所发生的错误的类型的信息。可能发生的错误的列表，参见“** OLE 2.01 **用户接口库”中的** OleUIInsertObject。

参数

dwFlags

是下列值之一：

- COleInsertDialog::DocObjectOnly **将只插入** DocObjects。
- COleInsertDialog::ControlsOnly **将只插入** ActiveX **控件。**

如果是零，则 DoModal **将既不插入一个** DocObjects，**也不插入一个** ActiveX **控件；它的返回值与上面所列的第一种原形的返回值一样。**

说明

此函数用来显示该 OLE Insert Object **对话框。**

如果你想通过设置 m_io **结构的成员来初始化不同的对话框控件，你应该在第一** DoModal **之前，但在对话框对象被构造之后进行。**

如果 DoModal **返回的是** IDOK**，则你可以调用其它的成员函数来获取用户输入到此对话框中的设置或信息。**

请参阅 COleDialog::GetLastError, CDialog::DoModal, COleInsertDialog::GetSelectionType, COleInsertDialog::GetClassID, COleInsertDialog::GetDrawAspect, COleInsertDialog::GetIconicMetafile, COleInsertDialog::GetPathName, COleInsertDialog::m_io

COleInsertDialog::GetClassID

```
const CLSID& GetClassID( ) const;
```

返回值

返回与被选择项相关的 CLSID**。**

说明

只有在 DoModal **返回** IDOK **时，并且该选择的类型是** COleInsertDialog::createNewItem **时，才调用此成员函数来获取与所选项关联的** CLSID。

更多的信息，参见“ OLE 2 **程序员参考，卷** 1**”中的** CLSID Key。

请参阅 COleInsertDialog::DoModal, COleInsertDialog::GetSelectionType

COleInsertDialog::GetDrawAspect

DVASPECT GetDrawAspect( ) const;

返回值

返回显示对象所需要的方法。

- DVASPECT_CONTENT　如果 Display As Icon 复选框没有被核选则返回此值。
- DVASPECT_ICON　如果 Display As Icon 复选框被核选则返回此值。

说明

此成员函数用来确定用户是否选择了将所选的项显示为图标。如果 DoModal 返回的是 IDOK 则调用此函数。

有关绘制方面的更多信息，参见“OLE 2 程序员参考，卷 1”中的 FORMATETC 数据结构。

请参阅　COleInsertDialog::DoModal, COleInsertDialog::COleInsertDialog

COleInsertDialog::GetIconicMetafile

HGLOBAL GetIconicMetafile( ) const;

返回值

当通过选择 OK **关闭对话框时，如果** Display As Icon **复选框被核选，则返回包含所选项的图标的图元文件的句柄；否则，返回** NULL。

说明

此成员函数用来获取包含所选项的图标的图元文件的句柄。

请参阅 COleInsertDialog::DoModal, COleInsertDialog::GetDrawAspect

COleInsertDialog::GetPathName

CString GetPathName( ) const;

返回值

返回在对话框中选择的文件的全路径。如果选择的类型是 createNewItem**，则在发行模式下函数返回一个没有意义的** CString**，在调试模式下则给出一个断言。**

说明

只有在 DoModal **返回** IDOK **并且选择的类型不是** COleInsertDialog::CreateNewItem **时，此成员函数用来获取所选文件的全路径。**

请参阅 COleInsertDialog::GetSelectionType, COleInsertDialog::DoModal

COleInsertDialog::GetSelectionType

UINT GetSelectionType( ) const;

返回值

返回所做选择的类型。

说明

当通过选择 OK **关闭** Insert Object **对话框时，调用此成员函数来获取所做选择的类型。**

返回类型的值由在 COleInsertDialog **类中定义的** Selection **枚举类型来指定。**

```
enum Selection
{
   createNewItem,
   insertFromFile,
   linkToFile
};
```

有关这些值的简短描述如下所示：

- COleInsertDialog::createNewItem　Create New 单选按钮被选择。
- COleInsertDialog::insertFromFile　Create From File 单选按钮被选择，并且 Link 复选框没有被核选。
- COleInsertDialog::linkToFile　Create From File 单选按钮被选择，并且 Link 复选框被核选。

请参阅　COleInsertDialog::DoModal, COleInsertDialog::COleInsertDialog

数据成员

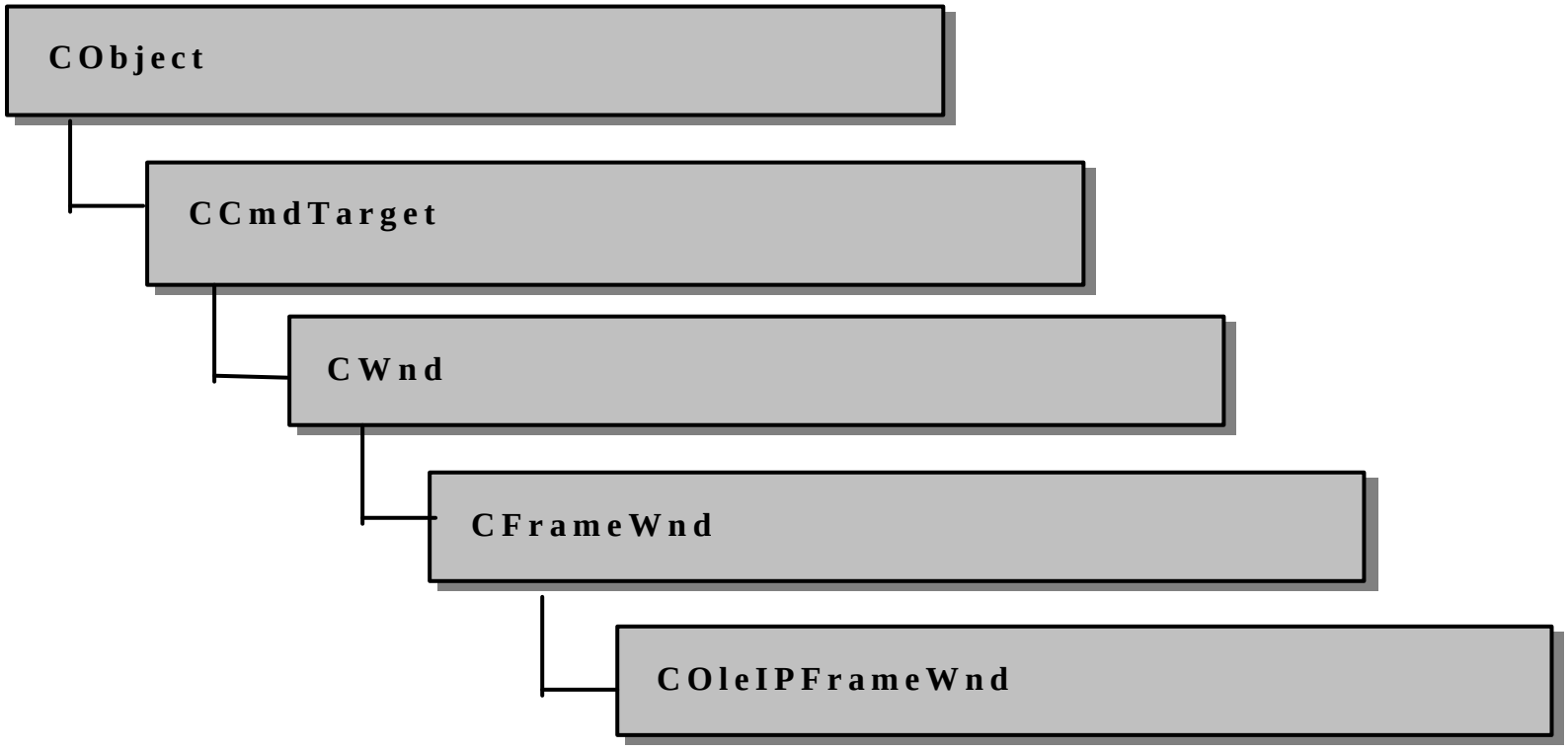
COleInsertDialog::m_io

说明

此数据成员是 OLEUIINSERTOBJECT **类型的结构，用来控制** Insert Object **对话框的行为。这个结构的成员可以被直接修改或通过成员函数来修改。**

请参阅 COleInsertDialog::COleInsertDialog, COleInsertDialog::DoModal

COleIPFrameWnd



使用 COleIPFrameWnd 类作为你的应用程序的现场可编辑窗口的基础。该类创

建工具条并将其安放在你的容器应用程序的文档窗口中。当用户调整现场可编辑窗口的大小时，它还处理由一个嵌入 COleResizeBar 对象产生的通知。

要获取更多有关使用 COleIPFrameWnd，请参见“Visual C++程序员指南”一书中的“激活”。

```
#include <afxole.h>
```

请参阅 CFrameWnd

COleIpFrameWnd 类成员

Construction	
COleIPFrameWnd	构造一个 COleIPFrameWnd 对象

Overridables	
OnCreateControlBars	当激活一个现场可编辑的项时被框架调用
RepositionFrame	由框架调用来改变现场可编辑窗口的位置

成员函数

COleIPFrameWnd::COleIPFrameWnd

COleIPFrameWnd();

说明

此函数构造一个 COleFrameWnd **对象，并初始化它的现场状态信息，此信息保存在一个** OLEINPLACEFRAMEINFO **类型的结构中。**

要获取更多的信息，请参见“ OLE 2 **程序员参考，卷** 1**”中的** OLEINPLACEFRAMEINFO。

请参阅 COleServerDoc::ActivateInPlace

COleIPFrameWnd::OnCreateControlBars

virtual BOOL OnCreateControlBars(CWnd* *pWndFrame*,CWnd* *pWndDoc*);

返回值

如果成功则返回非零值；否则返回 0；

参数

pWndFrame

指向包装应用程序的框架窗口的指针。

pWndDoc

指向包装应用程序的尾矿顶层窗口的指针。如果包装应用程序是一个 SDI **应用程序，则该参数可以为** NULL。

说明

当一个项因为现场编辑而被激活时，框架调用 OnCreateControlBars **函数。该函数的缺省实现不做任何事情。可以重载该函数，以执行控制条创建时所要求的任何特定处理。**

请参阅 COleServerDoc::ActivateInPlace

COleIPFrameWnd::RepositionFrame

virtual void RepositionFrame(LPCRECT *lpPosRect*, LPCRECT *lpClipRect*);

参数

lpPosRect

指向一个 RECT **结构或** CRect **对象的指针，此结构或对象中包含框架窗口的当前位置相对于客户区的像素坐标。**

lpClipRect

指向一个 RECT 结构或 CRect 对象的指针，此结构或对象中包含框架窗口的当前剪裁矩形相对于客户区的坐标。

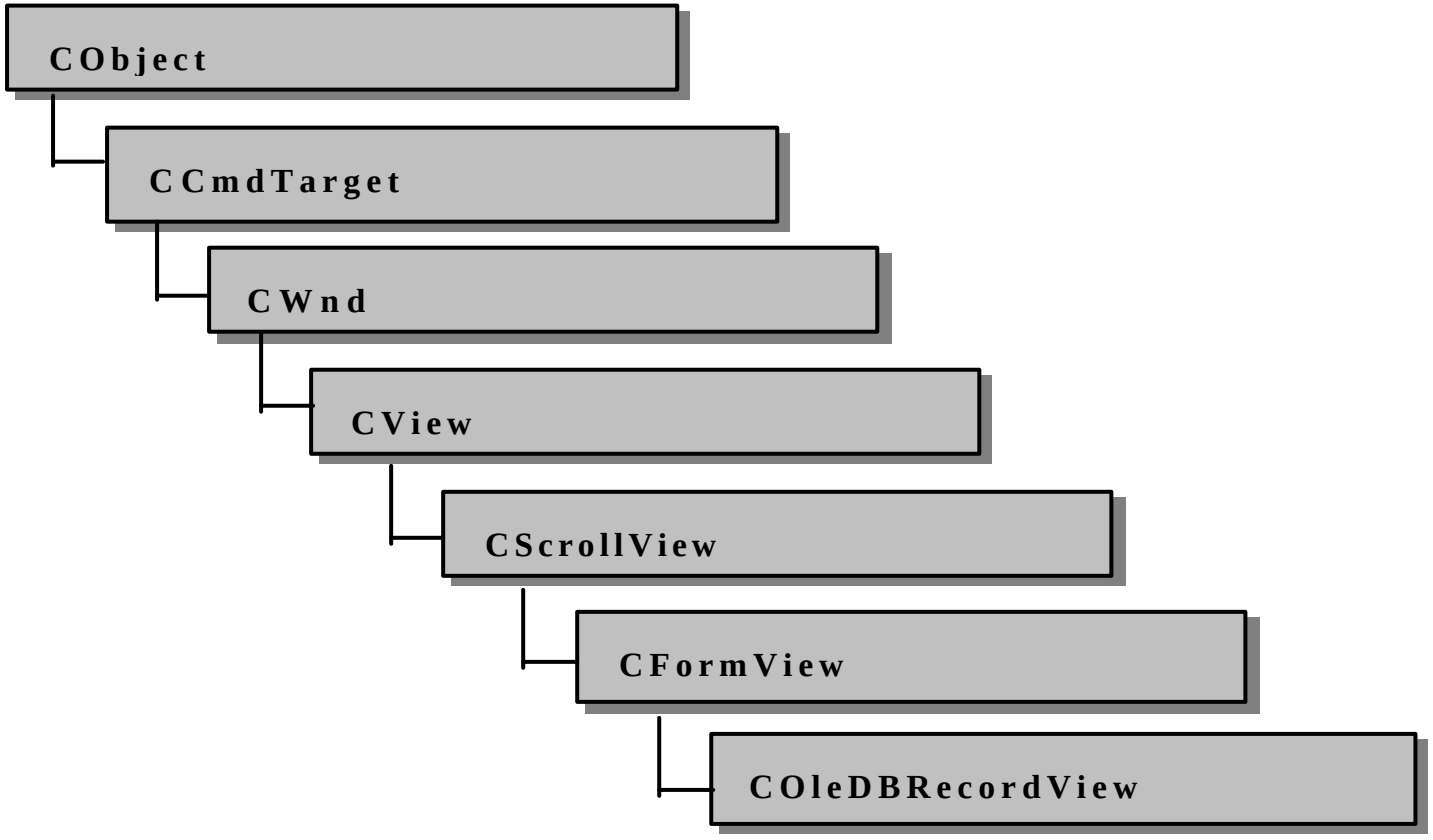
说明

框架调用 RepositionFrame 成员函数来布局控制条，并重定位现场可编辑窗口，使其全部可见。

容器窗口中控制条的布局不同于非 OLE 框架窗口执行的布局。非 OLE 框架窗口从给定框架窗口尺寸计算控制条和其它对象的位置，它与函数 CFrameWnd::RecalcLayout 调用中的处理相同。客户区是减掉控制条和其它对象所占的空间后所剩的区域。从另一方面讲，COleIPFrameWnd 窗口根据给定客户区来定位工具条。换句话来说，CFrameWnd::Recalc- Layout 的处理是“从外至内”，而 COleFrameWnd::RecalcLayout 的处理是“从内至外”。

请参阅 CFrameWnd::RecalcLayout

COleDBRecordView



COleDBRecordView 对象是一个用来显示受控制的数据库记录的视窗。这个视窗是一个直接连接于 CRowset 对象的表格视窗。该视窗是从一个对话框模板资源生成来的，并且在对话框模板的控件中显示 CRowset 对象域。COleDBRecordView 对象使用对话框数据交换（DDX）和 CRowset 中的导航功能，来使数据在表格上的控件与行集之间自动移动。COleDBRecordView 也提供缺省的用于移动到第一个，下一个，前一个或最后一个的实现，并且提供在当前视窗中用于更新记录的界面。

> **注意** 如果你使用 Date Access Objects（DAO）类比使用 OLE DB Consumer Template 类更多，请使用 CDaoRecordView 来代替。如果要获取更多的信息，请参见“Visual C++程序员指南”一书中的文章“数据库主题（总体）”和“DAO 和 MFC”。

COleDBRecordView 在行集中跟踪用户的位置，因此记录视窗可以更新用户界

面。当用户移动到行集的任何一端时，记录视窗就会使用在同一方向上进一步移动的用户界面对象---如菜单项或工具条按钮无效。

如果要获取更多关于行集类的信息，请参见“Microsoft Visual C++ 6.0 类库参考手册”中的“Microsoft Visual C++6.0 模板库参考”卷中的文章“使用 OLE DB Consumer Templates”。

#include <afxoledb.h>

COleDBRecordView 类成员

Construction	
COleDBRecordView	构造一个 COleDBRecordView 对象
Attributes	
OnGetRowset	返回一个指向 CRowset 类对象的指针

Operations	
OnMove	更新数据源上的当前记录（如果变脏了），然后移动到指定的记录（下一个，前一个，第一个或最后一个）

成员函数

COleDBRecordView::COleDBRecordView

COleDBRecordView(LPCTSTR *lpszTemplateName*);
COleDBRecordView(UINT *nIDTemplate*);

参数

lpszTemplateName

记录一个以空字符结尾的字符串，该字符串是一个对话框模板资源的名字。

nIDTemplate

记录一个对话框模板资源的 ID 号。

说明

当你创建一个类型是从 COleDBRecordView 派生而来的对象时，激活其中的一个构造函数来生成一个视对象，并且标识该视所基于的对话框资源。你可以用名字来标识相应的资源（将一个字符串作为参数传递给构造函数），或者用它的 ID 来标识（将一个无符号的整数作为参数传递给构造函数）。

注意　你的派生类必须提供它自己的构造函数。在这个构造函数中，用资源的名字或 ID 作为一个参数来激活构造函数 COleDBRecordView::COleDBRecordView。

COleDBRecordView::OnGetRowset

virtual CRowset* OnGetRowset() = 0;

返回值

如果成功的生成了对象则返回一个指向 CRowset **对象的指针；否则返回一个** NULL **指针。**

说明

此函数返回一个指向 CRowset **对象的指针。为了构造或获得一个行集对象，并返回一个指向该对象的指针，你必须重载这个函数。如果你用** ClassWizard **来声明你的记录视类，则向导将为你进行缺省的重载。**ClassWizard **的缺省实现返回保存在记录视窗中的行集指针，如果这个指针存在的话。如果指针不存在，它就构造一个行集对象，该对象的类型是你用** ClassWizard **指定的，接着调用**

它的成员函数 Open 来打开表或运行一次查询，然后返回一个指向该对象的指针。

COleDBRecordView::OnMove

virtual BOOL OnMove(UINT *nIDMoveCommand*)

返回值

如果移动成功则返回非零值；如果移动请求被忽略则返回 0。

参数

nIDMoveCommand

是下面列出的标准命令 ID 值之一：

- ID_RECORD_FIRST 移动到记录集中的第一个记录。

- ID_RECORD_LAST　　移动到记录集中的最后一个记录。
- ID_RECORD_NEXT　　移动到记录集中的下一个记录。
- ID_RECORD_PREV　　移动到记录集中的前一个记录。

说明

这个成员函数可以实现在行集中移动到不同的记录，并且在记录视窗的控件中显示该记录的字段。其缺省的实现是与记录视窗相关的 CRowset 对象的适当的成员函数 Move。

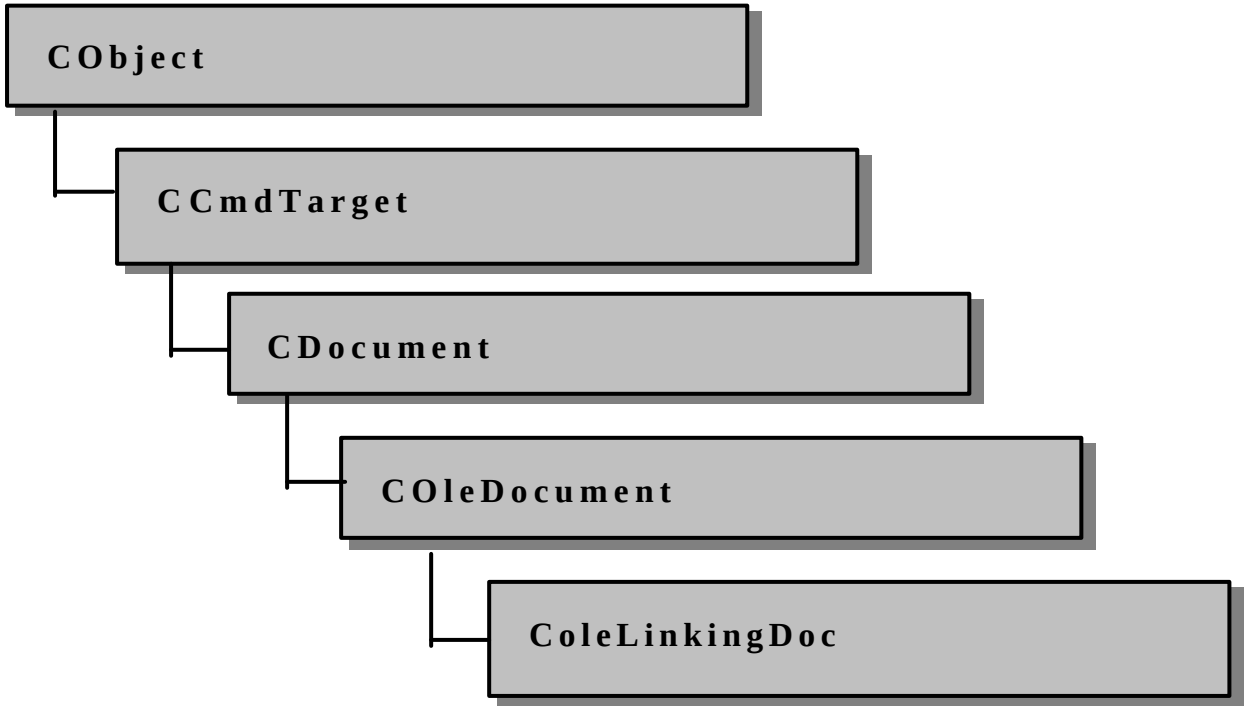
缺省的，如果用户改变了记录视窗中的数据，则 OnMove 更新数据源中的当前记录。

AppWizard 创建一个菜单资源，它具有 First Record，Last Record，Next Record 和 Previous Record 菜单项。如果你选择了 Dockable Toolbar 选项，AppWizard

就创建一个工具条，其上有对应于这些命令的按钮。

如果你在记录集中移动超过了最后一个记录，则在记录视窗中仍然显示最后一个记录。如果你超过了第一个记录，则在视窗中仍然显示第一个记录。

COleLinkingDoc



COleLinkingDoc 类是 OLE 容器文档的基类，它支持链接到它们所包含的嵌入项。一个支持链接到嵌入项的容器应用程序被称为“链接容器”。示例应用程序 OCLIENT 就是一个链接容器的例子。

当一个链接项的源是另一个文档的嵌入项时，则为了使嵌入项可以编辑，那个包容文档必须被载入。由于这个原因，当用户想要编辑链接项的源时，容器程序就必须能够由另一个容器应用程序启动。你的应用程序也必须使用 COleTemplateServer 类以使它在被其它的应用程序启动时能够创建文档。

为了使你的容器应用程序成为一个链接容器，请从 COleLinkingDoc 派生出你的文档类，而不是从 COleDocument 派生。同处理其它任何 OLE 容器程序一样，应用程序必须将自己的类设计为可存储应用程序的本地数据以及嵌入或链接项。而且，你必须为保存你的本地数据设计数据结构。如果你为你的应用程序的本地数据定义了一个 CDocItem 派生类，你就可以使用由 COleDocument 定

义的界面来保存你的本地数据以及 OLE 数据。

为了使你的应用程序能够被另一个容器程序启动，请定义一个 COleTemplateServer 对象作为你的应用程序的 CWinApp 派生类的成员。

```
class COleClientApp : public CWinApp
{
// ...
protected:
   COleTemplateServer m_server;
// ...
};
```

在你的 CWinApp 派生类的成员函数 InitInstance 中，创建一个文档模板并指定你的 COleLinkingDoc-derived 类为一个文档类：

```
// CMainDoc is derived from COleLinkingDoc
CMultiDocTemplate* pDocTemplate = new
CMultiDocTemplate(IDR_OCLIENTTYPE,
   RUNTIME_CLASS(CMainDoc),
   RUNTIME_CLASS(CSplitFrame),
```

```
    RUNTIME_CLASS(CMainView));
pDocTemplate->SetContainerInfo(
    IDR_OCLIENTTYPE_CNTR_IP);
AddDocTemplate(pDocTemplate);
```

通过调用 COleTemplateServer 对象的 ConnectTemplate 成员函数，将此 COleTemplateServer 对象连接到你的文档模板，并通过调用 COleTemplateServer::RegisterAll 函数向 OLE 系统注册所有的类对象。

```
m_server.ConnectTemplate(clsid, pDocTemplate, FALSE);
COleTemplateServer::RegisterAll();
```

CWinApp 派生类的定义和 InitInstance 函数的示例，请参见 MFC 示例 OCLIENT 中的 OCLIENT.H 和 OCLIENT.CPP。

如果要获取有关使用 COleLinkingDoc 的更多信息，请参见“Visual C++程序员指南”一书中的文章“容器：实现一个容器”。

```
#include <afxole.h>
```

请参阅 CDocTemplate

COleLinkingDoc 类成员

Construction	
ColeLinkingDoc	构造一个 COleLinkingDoc 对象
Operations	
Register	向 OLE 系统 DLL 注册此文档
Revoke	撤消文档的注册
Overridables	
OnFindEmbeddedItem	查找指定的嵌入项
OnGetLinkedItem	查找指定的链接项

成员函数

COleLinkingDoc::COleLinkingDoc

COleLinkingDoc();

说明

此函数构造一个 COleLinkingDoc **对象，但不开始与** OLE **系统** DLL **的通信。应用程序必须调用成员函数** Register **通知** OLE **文档已经打开。**

请参阅 COleLinkingDoc::Register

COleLinkingDoc::OnFindEmbeddedItem

virtual COleClientItem* OnFindEmbeddedItem(LPCTSTR *lpszItemName*);

返回值

返回一个指向指定项的指针；如果没有找到该项则返回 NULL。

参数

lpszItemName

指向所请求的嵌入 OLE **项名称的指针。**

说明

框架调用此函数来确定文档中是否包含一个具有指定名称的嵌入 OLE **项。此函数的缺省实现是在嵌入项表中查找具有指定名称的项（名字比较区分大小写）。如果你有你自己的存储或命名嵌入** OLE **项的方式，请重载该函数。**

请参阅 COleClientItem，COleLinkingDoc::OnGetLinkedItem

COleLinkingDoc::OnGetLinkedItem

virtual COleServerItem* OnGetLinkedItem(LPCTSTR *lpszItemName*);

返回值

指向指定项的指针；如果没有找到该项则返回 NULL。

参数

lpszItemName

指向所请求的链接 OLE **项名字的指针。**

说明

框架调用该函数检查文档中是否包含一个具有指定名字的链接服务器项。缺省的 COleLinkingDoc **实现总是返回** NULL。**该格式在派生类** COleServerDoc **中被**

重载，用来在 OLE 服务器项表中查找一个具有指定名称的链接项（名字比较区分大小写）。如果你已经实现了你自己的存储或检索嵌入 OLE 项的方式，则重载该函数。

请参阅 COleServerItem::GetItemName，COleServerItem::SetItemName，COleLinkingDoc::OnFindEmbeddedItem

COleLinkingDoc::Register

BOOL Register(COleObjectFactory* *pFactory*, LPCTSTR *lpszPathName*);

返回值

如果成功注册文档则返回非零值；否则返回 0。

参数

pFactory

指向一个 OLE 库对象的指针（可以是 NULL）。

lpszPathName

指向容器文档的全路径的指针。

说明

此函数通知 OLE 系统 DLL，文档是打开的。该函数在创建或打开一个已命名文件时被调用，用来向 OLE 系统 DLL 注册此文档。如果文档代表一个嵌入项，则不需要调用这个函数。

如果在你的应用程序中正使用 COleTemplateServer，则由 COleLinkingDoc 实现的函数 OnNewDocument，OnOpenDocument 和 OnSaveDocument 来调用 Register。

请参阅 COleTemplateServer，COleObjectFactory，

CDocument::OnNewDocument，

CDocument::OnOpenDocument

COleLinkingDoc::Revoke

void Revoke();

说明

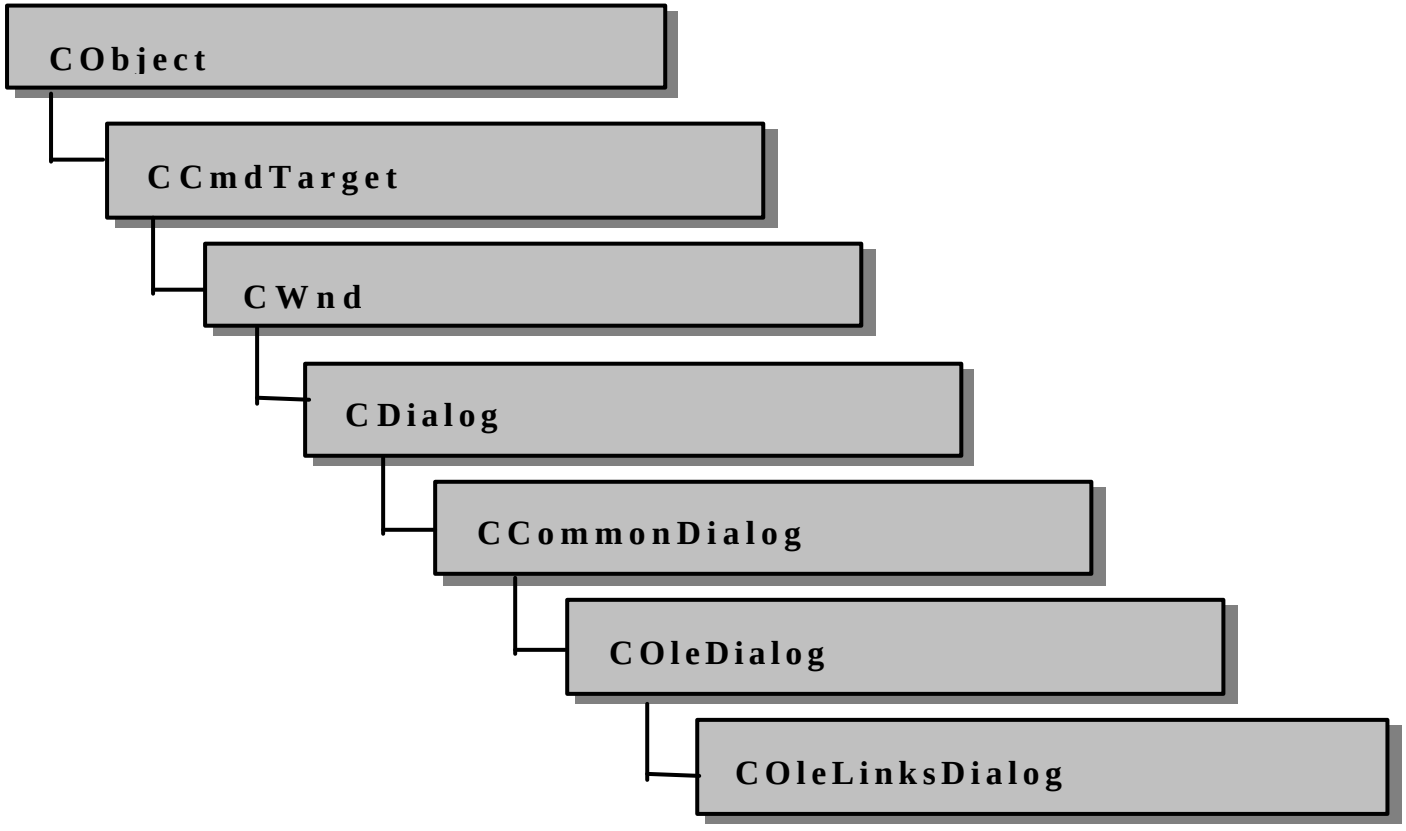
此函数通知 OLE **系统** DLL **文档不再是打开的。该函数从** OLE **系统** DLL **中撤销文档的注册。**

当关闭一个已命名文件时，你必须调用这个函数，但是通常不需要直接调用它。由 COleLinkingDoc **实现的函数** OnNewDocument，OnOpenDocument **和** OnSaveDocument **来调用** Register。

请参阅 COleTemplateServer，CDocument::OnCloseDocument，

CDocument::OnNewDocument, CDocument::OnOpenDocument,
CDocument::OnSaveDocument

COleLinksDialog



COleLinksDialog对象用于OLE Edit Links对话框。当你要调用这个对话框时，可创建一个COleLinksDialog对象。当COleLinksDialog对象被构造之后，你可以使用m_el结构来初始化对话框中各个控件的值或状态。m_el结构是OLEUIEDITLINKS类型。要获取有关使用这个对话框类的更多信息，参见成员函数DoModal。

注意 AppWizard生成的容器代码使用了这个类。

要获取更多的信息，请参见“OLE 2.01用户接口库”中的OLEUIEDITLINKS结构。

要获取更多关于OLE特有对话框的信息，参见“Visual C++程序员指南”一书中的文章“OLE中的对话框”。

#include <afxodlgs.h>

请参阅 COleDialog

COleLinksDialog 类对象

Data Members	
m_el	类型为 OLEUIEDITLINKS 的结构，用于控制对话框的行为
Construction	
ColeLinksDialog	构造一个 COleLinksDialog 对象
Operations	
DoModal	显示 OLE Edit Links 对话框

成员函数

COleLinksDialog::COleLinksDialog

COleLinksDialog(COleDocument* *pDoc*,CView**pView*,DWORD*dwFlags*=0,CWnd*

pParentWnd=NULL);

参数

pDoc

指向 OLE **文档的指针，该文档包含要编辑的链接。**

pView

指向 *pDoc* **上的当前视。**

dwFlags

创建标志，可包含 0 **或** ELF_SHOWHELP**，用于指定当对话框显示时是否显示** Help **按钮。**

pParentWnd

指向对话框对象所属的父或属主窗口对象（CWnd **类型）。如果该参数为** NULL**，则对话框的父窗口设置为主应用程序窗口。**

说明

该函数只构造 COleLinksDialog **对象。要显示对话框，需调用** DoModal **函数。**

请参阅 COleDocument，COleLinksDialog::DoModal，CView，CWnd

COleLinksDialog::DoModal

virtual int DoModal();

返回值

对话框的完成状态。它可以是下面的值：

- IDOK **如果对话框被成功显示则返回此值。**
- IDCANCEL **如果用户取消了对话框则返回此值。**
- IDABORT **如果发生了错误则返回此值。如果返回的是** IDABORT，

则调用成员函数 COleDialog::GetLastError 来获取有关所发生错误的类型的进一步信息。可能发生的错误的列表，请参见“OLE 2.01 用户接口库”中的 OleUIEditLinks 函数。

说明

此函数用来显示 OLE Edit Links 对话框。

如果你要通过设置 m_el 结构的成员来初始化不同的对话框控件，你必须要在调用 DoModal 之前，对话框对象被销毁之后进行。

请参阅 COleDialog::GetLastError，CDialog::DoModal，COleLinksDialog::m_el

数据成员

COleLinksDialog::m_el

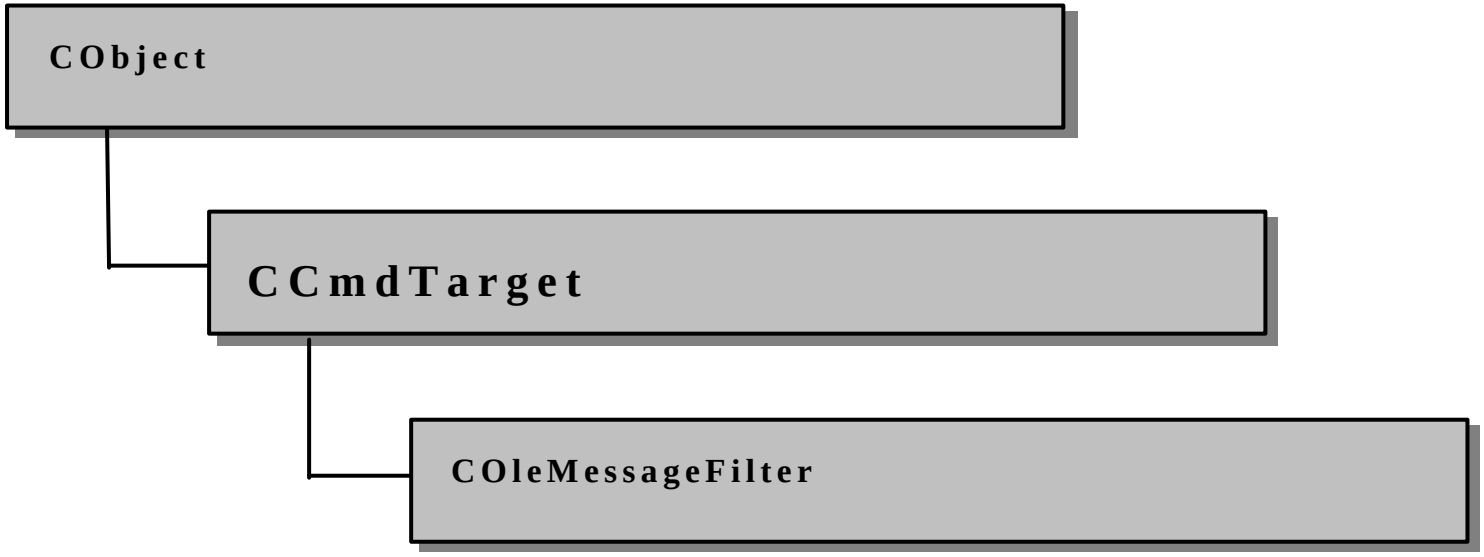
说明

用来控制 Edit Links **对话框行为的** OLEUIEDITILINKS **类型的结构。这个结构的成员可以直接修改，或通过成员函数来修改。**

要获取更多的信息，参见“OLE 2.01 **用户接口库”中的** OLEUIEDITLINKS **结构。**

请参阅 COleLinksDialog::COleLinksDialog，COleLinksDialog::DoModal

COleMessageFilter



COleMessageFilter类管理 OLE 应用程序交互所需要的并行能力。

COleMessageFilter 类用于可视化编辑服务器和容器应用程序中，以及 OLE 自动化应用程序中。对于要调用的服务器应用程序，该类可使应用程序保持“忙”

的状态，这样其他容器应用程序的调用请求就会被取消或稍后重试。当被调用的应用程序忙时，也可以使用该类确定调用者应用程序将采取什么样的动作。

服务器应用程序常用的用法是在文档或其他 OLE 可访问的对象有可能遭到破坏的情况下调用 BeginBusyState 和 EndBusyState。在用户界面更新过程中，CWinApp::OnIdle 将发起这些调用。

缺省地，当应用程序初始化时就分配一个 COleMessageFilter 对象。可以使用 AfxOleGetMessageFilter 检索该对象。

这是一个高级类；你很少需要直接使用该类。

要获得更详细的信息，请参阅“Visual C++ 程序员指南”在线文档中的“服务器：实施一个服务器”一节。

```
#include <afxole.h>
```

请参阅 CCmdTarget

COleMessageFilter 类成员

构造器

ColeMessageFilter	构造一个 COleMessageFilter 对象

操作

Register	注册 OLE 系统 DLL 的消息过滤器
Revoke	激活 OLE 系统 DLL 消息过滤器的注册过程
BeginBusyState	使应用程序处于忙的状态
EndBusyState	结束应用程序忙的状态
SetBusyReply	确定忙的应用程序对 OLE 调用的回答
SetRetryReply	确定调用者应用程序对忙应用程序的回答
SetMessagePendingDelay	确定应用程序要等多长时间才对 OLE 调用做出响应
EnableBusyDialog	使能或禁止在被调用的应用程序处于忙状态时所显示的对话框

续表

EnableNotRespondingDialog	使能或禁止在被调用的应用程序没有响应时所显示的对话框

重载

OnMessagePending	在处理 OLE 调用时框架调用该方法处理消息

成员函数

COleMessageFilter::BeginBusyState

```
virtual void BeginBusySystem();
```

说明

调用该函数将启动忙状态。其与 EndBusyState 函数一起控制应用程序忙的状

态。

SetBusyReply 方法确定在应用程序忙时对调用者应用程序所做的响应。

BeginBusyState 和 EndBusyState 分别增加或减少用于决定应用程序忙状态的计数器。例如，

调用 BeginBusyState 两次和调用 EndBusyState 一次将导致忙状态。要取消忙状态，调用 EndBusyState 的次数与调用 BeginBusyState 的次数必须相等。

缺省地，框架在空闲处理过程中进入忙状态，这是由 CWinApp::OnIdle 来执行的。当应用程序正在处理 ON_COMMANDUPDATEUI 通知时，发来的调用请求将稍后处理，要等到空闲处理过程完成之后。

请参阅 COleMessageFilter::EndBusyState, COleMessageFilter::SetBusyState, CWinApp::OnIdle

COleMessageFilter::COleMessageFilter

COleMessageFilter() ;

说 明

创 建 一 个 COleMessageFilter **对 象 。**

请 参 阅 COleMessageFilter::Register, COleMessageFilter::Remove。

COleMessageFilter::EnableBusyDialog

void EnableBusyDialog (BOOL bEnableBusy = TRUE);

参 数

bEnableBusy

指 明 “ 忙 ” 对 话 框 是 使 能 还 是 禁 止 。

说明

使能或禁止忙对话框，在 OLE **调用过程中当发送消息逾期而没有回应时（请参阅** SetRetry- Reply**），将显示该对话框。**

请参阅 COleMessageFilter::EnableNotRespondingDialog,
COleMessageFilter::BeginBusyState, COleMessageFilter::SetBusyReply,
COleMessageFilter::SetRetryReply, COleBusyDialog

COleMessageFilter::EnableNotRespondingDialog

void EnableNotRespondingDialog (BOOL bEnableNotResponding = TRUE)

参数

bEnableNotResponding

指明“无响应”对话框是使能还是禁止。

说明

使能或禁止“无响应”对话框，在 OLE 调用过程中发送键盘或鼠标消息而该调用过程超时就显示该对话框。

请参阅 COleMessageFilter::EnableBusyDialog,
COleMessageFilter::BeginBusyState,COleMessageFilter::SetBusyReply,
COleBusyDialog

COleMessageFilter::EndBusyState

virtual void EndBusyState () ;

说明

调用该函数结束忙状态。其与 BeginBusyState 函数一起控制应用程序忙的状态。函数 SetBusyReply 决定在其处于忙状态时对调用者应用程序的响应。

BeginBusyState 和 EndBusyState 分别增加或减少用于决定应用程序忙状态的计数器。例如，调用 BeginBusyState 两次和调用 EndBusyState 一次将导致忙状态。要取消忙状态，调用 EndBusyState 的次数与调用 BeginBusyState 的次数必须相等。

缺省地，框架在空闲处理过程中进入忙状态，这是由 CWinApp::OnIdle 来执行的。当应用程序正在处理 ON_UPDATE_COMMAND_UI 通知时，发来的调用请求将稍后处理，要等到空闲处理过程完成之后。

请参阅 COleMessageFilter::BeginBusyState, COleMessageFilter::SetBusyState, CWinApp::OnIdle

COleMessageFilter::OnMessagePending

```
virtual BOOL OnMessagePending ( const MSG* pMsg );
```

返回值

非 0 **代表成功；否则，就返回** 0。

参数

pMsg

要发送消息的指针。

说明

在 OLE **调用过程中框架调用该函数处理消息。**

当调用者应用程序在等待调用完成时，框架就调用 OnMessagePending **方法，其参数是一个指向要发送消息的制止。缺省地，框架发送** WM_PAINT **消息，因此在一个花费很长时间的调用过程中出口仍可以更新。**

在消息过滤器活动之前，你必须调用 Register 对消息过滤器进行注册。

请参阅　COleMessageFilter::Register, AfxOleInit, CWinApp::InitInstance

COleMessageFilter::Register

BOOL Register ();

返回值

非 0 代表成功；否则，就返回 0。

说明

该函数注册 OLE 系统 DLL 的消息过滤器。一个消息过滤器不起作用，除非向系统 DLL 进行注册。通常，你在应用程序初始化代码中注册应用程序的消息过滤器。你的应用程序所注册的任何消息过滤器在应用程序终止之前都应调用

Revoke 函数撤销注册信息。

框架缺省的消息过滤器在初始化时自动进行了注册，并在终止之前自动撤销。

请参阅 COleMessageFilter::Revoke

COleMessageFilter::Revoke

void Revoke () ;

说明

撤销以前调用 Register 所注册的信息。在应用程序终止之前应撤销消息过滤器。

由框架自动创建和注册的缺省消息过滤器也将自动被撤销。

请参阅 COleMessageFilter::Register

COleMessageFilter::SetBusyReply

void SetBusyReply (SERVERCALL nBusyReply)

参数

nBusyReply

SERVERCALL **枚举集合中的一个值，在** COMPOBJ.H **文件中定义。可以是下面的一个值：**

- SERVERCALL_ISHANDLED **应用程序可以接受调用，但在处理某些特殊的调用过程中可能会失败。**
- SERVERCALL_REJECTED **应用程序可能永远都不会处理该调用。**
- SERVERCALL_RETRYLATER **应用程序处于暂时不能处理调用的状态中。**

说明

该函数将应用程序设置为“忙-响应”状态。BeginBusyState 和 EndBusyState 函数控制着应用程序忙的状态。

当调用 BeginBusyState 函数将应用程序设置为忙状态时，其对来自 OLE 系统 DLL 调用的
响应值是由最近调用 SetBusyReply 函数的设置值所决定的。调用者应用程序使用该忙-响应信息决定采取什么动作。
缺省地，忙－响应是 SERVERCALL_RETRYLATER。该响应消息导致调用者应用程序尽快重试该调用过程。

请参阅 COleMessageFilter::BeginBusyState, COleMessageFilter::EndBusyState

COleMessageFilter::SetMessagePendingDelay

void SetMessagePendingDelay (DWORD nTimeout = 5000) ;

参数

nTimeout

消息延迟发送的秒数。

说明

该函数决定在调用者应用程序将在多长时间内等待被调用应用程序的响应，过了该时间，

调用者应用程序将采取进一步的动作。

该函数与 SetRetryRelay **函数协同工作。**

请参阅 COleMessageFilter::SetRetryReply

COleMessageFilter::SetRetryReply

void SetRetryReply (DWORD nRetryReply = 0);

参数

nRetryReply

两次重试之间所等待的秒数。

说明

该函数将决定调用者应用程序在收到被调用应用程序的忙响应信息时所采取的动作。

当被调用的应用程序指明自己忙时，调用者应用程序可能会决定等待直到服务器不再忙、立刻重试或在指定的间隔后重试。也可能决定取消本次调用。

调用者的响应是由 SetRetryReply **和** SetMessagePendingDelay **控制的。**

SetRetryReply 决定调用者应用程序在重试调用时应等待多长时间。SetMessagePendingDelay 决定调用者应用程序在采取进一步动作之前等待服务器做出响应的时间。

通常，缺省值都是可接受的，不需要修改。框架将每隔 nRetryRelay 毫秒就重试调用，直到调用成功或者超出了消息发送延迟时间规定。nRetryRelay 取 0 值则意味着立即重试，-1 则意味着取消本次调用。

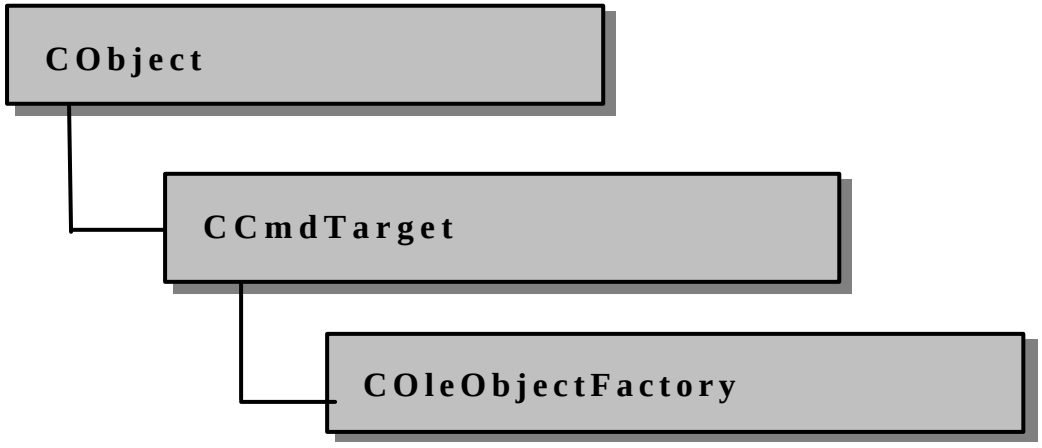
当超出消息延迟发送时间限制时，就显示 OLE“忙-对话框”，这样用户就可以选择是取
消本次调用还是重试。调用 EnableBusyDialog 函数来使能或禁止该对话框。

当在调用过程中发送键盘或鼠标消息而调用超时时（超出了消息发送延迟规定），就显示“无响应”对话框。调用 EnableNotRespondingDialog 来使能或禁止该对话框。通常，这种状态意味着某些东西出现了错误，用户已经不耐烦了。

当禁止该对话框时，对于调用处于忙状态的应用程序总是使用当前“重试响应”。

请参阅 COleBusyDialog, COleMessageFilter::EnableNotRespondingDialog, COleMessageFilter::EnableBusyDialog, COleMessageFilter::SetMessagePendingDelay

COleObjectFactory



COleObjectFactory 类实现了 OLE 类工厂，此工厂创建诸如服务器，自动化对象和文档之类的 OLE 对象。

COleObjectFactory 类有用来执行下列功能的成员函数：

- 管理对象的注册。
- 更新 OLE 系统注册和运行时注册，运行时注册用来通知 OLE 对象正在运行并准备接收信息。
- 通过限制注册用户在设计时的使用和注册应用程序在运行时的使用来强迫注册。
- 向 OLE 系统注册表注册控件对象工厂。

如果要获取更多有关对象创建的信息，请参见文章“数据对象和数据源（OLE）”和“数据对象和数据源：创建和析构”。更多与注册有关的信息，请参见文章“注册”。这些文章都在“Visual C++程序员指南”一书中。

#include <afxdisp.h>

请参阅 COleTemplateServer

COleObjectFactory 类成员

Construction	
ColeObjectFactory	构造一个 COleObjectFactory 对象
Operations	
Register	向 OLE 系统 DLLs 注册这个对象工厂
RegisterAll	向 OLE 系统 DLLs 注册应用程序的所有对象工厂
Revoke	从 OLE 系统 DLLs 中撤销这个对象工厂的注册
RevokeAll	从 OLE 系统 DLLs 中撤销应用程序所有对象工厂的注册
UpdateRegistryAll	向 OLE 系统注册表注册应用程序的所有对象工厂

Attributes	
IsRegistered	指示对象工厂是否已向 OLE 系统 DLL 注册
GetClassID	返回此工厂创建的对象的 OLE 类 ID

Overridables	
OnCreateObject	由框架调用来创建这个工厂对象的一个新对象
UpdateRegistry	向 OLE 系统注册表注册这个工厂对象
VerifyUserLicense	检验控件是否为设计时使用许可
GetLicenseKey	从控件的 DLL 请求一个唯一的键
VerifyLicenseKey	校验控件的嵌入键是否与其容器的嵌入键匹配

成员函数

COleObjectFactory::COleObjectFactory

COleObjectFactory(REFCLSID *clsid*,CRuntimeClass* *pRuntimeClass*,
BOOL *bMultiInstance*,LPCTSTR *lpszProgID*);

参数

clsid

这个对象工厂所代表的 OLE 类 ID 的引用。

pRuntimeClass

指向这个工厂能够创建的 C++对象的运行时类的指针。

bMultiInstance

指示应用程序的单一实例是否能够支持多实例化。如果是 TRUE，则对于每一个创建对象的请求将启动应用程序的多个实例。

lpszProgID

指向一个包含程序的文字标识符的字符串指针，比如说“ Microsoft Excel”。

说明

构造一个 COleObjectFactory 对象，将它初始化为一个没有注册的对象工厂，并

将它添加到工厂列表中。但是为了使用这个对象，你必须注册它。

更多的信息参见“OLE 2 程序员参考，卷 1”中的 CLSID Key。

请参阅　CRuntimeClass

COleObjectFactory::GetClassID

REFCLSID GetClassID() const;

返回值

返回这个工厂所代表的 OLE 类 ID 的引用。

说明

返回这个工厂所代表的 OLE 类 ID 的引用。

要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 CLSID Key。

请参阅 COleObjectFactory::COleObjectFactory

COleObjectFactory::GetLicenseKey

virtual BOOL GetLicenseKey(DWORD *dwReserved*,BSTR **pbstrKey*);

返回值

如果许可键字符串不是 NULL **则返回非零值；否则返回** 0。

参数

dwReserved

为将来的使用保留。

pbstrKey

指向 BSTR **的指针，** BSTR **是用来保存许可键的。**

说明

从控件的 DLL 中请求一个唯一的许可键，并将它保存在有 pbstrKey 指向的 BSTR 中。

这个函数的缺省实现返回 0，而且不向 BSTR 中保存任何东西。如果你使用 MFC ActiveX ControlWizard 来创建你的项目，ControlWizard 提供一个重载后的函数，该函数获取控件的许可键。

请参阅 COleObjectFactory::VerifyUserLicense，COleObjectFactory::VerifyLicenseKey

COleObjectFactory::IsRegistered

BOOL IsRegistered() const;

返回值

如果注册了这个工厂则返回非零值；否则返回 0。

说明

如果已经向 OLE **系统** DLLs **注册了这个工厂则返回非零值。**

请参阅 COleObjectFactory::Register，COleObjectFactory::Revoke

COleObjectFactory::OnCreateObject

virtual CCmdTarget* OnCreateObject();

返回值

指向被创建的东西的指针。如果失败则会抛出一个内存异常。

说明

由框架调用来创建一个新对象。为了从某些不是传递给构造函数的 CRuntimeClass **的东西来创建对象，请重载这个函数。**

请参阅 COleObjectFactory:: COleObjectFactory，CRuntimeClass

COleObjectFactory::Register

```
BOOL Register();
```

返回值

如果成功地注册了这个工厂则返回非零值；否则返回 0。

说明

向 OLE **系统** DLLs **注册这个对象工厂。通常当应用程序被启动时，由**

CWinApp::InitInstance 调用这个函数。

请参阅 COleObjectFactory::Revoke，COleObjectFactory::RegisterAll，CWinApp::InitInstance

COleObjectFactory::RegisterAll

static BOOL PASCAL RegisterAll()

返回值

如果成功地注册了这个工厂则返回非零值；否则返回 0。

说明

向 OLE 系统 DLLs 注册应用程序的所有对象工厂。通常当应用程序被启动时，由 CWinApp:: InitInstance 调用这个函数。

请参阅 COleObjectFactory::Revoke，COleObjectFactory::Register，CWinApp::InitInstance

COleObjectFactory::Revoke

void Revoke();

说明

从 OLE **系统** DLLs **中撤销这个对象工厂的注册。在应用程序终止前框架自动调用这个函数。如果必需的话，可以从重载的** CWinApp::ExitInstance **调用它。**

请参阅 COleObjectFactory::RevokeAll，COleObjectFactory::Register，CWinApp::ExitInstance

COleObjectFactory::RevokeAll

static void PASCAL RevokeAll();

说明

从 OLE **系统** DLLs **中撤销应用程序的所有对象工厂的注册。在应用程序终止前框架自动调用这个函数。如果必须的话，可以从重载的** CWinApp::ExitInstance **调用它。**

请参阅 COleObjectFactory::Revoke，COleObjectFactory::RegisterAll，CWinApp::ExitInstance

COleObjectFactory::UpdateRegistry

void UpdateRegistry (LPCTSTR *lpszProgID* = NULL);
virtual void UpdateRegistry(BOOL *bRegister*) = 0;

参数

lpszProgID

是指向一个字符串的指针，该字符串是人可阅读的程序标识符，比如说

“Excel.Docume- nt.5”。

bRegister

指示控件类的对象工厂是否已经被注册。

说明

下面是有关这个函数的两种形式的简短讨论：

- UpdateRegistry(*lpszProgID*) 向 OLE 系统注册表注册这个对象工厂。这个函数通常由 CWinApp::InitInstance 在应用程序被启动时调用。
- UpdateRegistry(*bRegister*) 该函数的这种形式是可重载的。如果 bRegister 为 TRUE，则此函数向系统注册表注册这个控件类。否则不注册这个类。

如果你使用 MFC ActiveX ControlWizard 来创建你的项目，ControlWizard 支持对这个函数的重载。

请参阅 COleObjectFactory::Revoke，COleObjectFactory::Register，
COleObjectFactory::UpdateRegistryAll，CWinApp::InitInstance

COleObjectFactory::UpdateRegistryAll

static void PASCAL UpdateRegistryAll();

说明

向 OLE **系统注册表注册应用程序的所有的对象工厂。此函数通常在应用程序被启动时被** CWinApp::InitInstance **调用。**

请参阅 COleObjectFactory::Revoke，COleObjectFactory::Register，
COleObjectFactory::UpdateRegistry，CWinApp::InitInstance

COleObjectFactory::VerifyLicenseKey

virtual BOOL VerifyLicenseKey(BSTR *bstrKey*);

返回值

如果运行时许可有效则返回非零值；否则返回 0。

参数

bstrKey

是一个用来保存容器的许可字符串版本的 BSTR。

说明

这个函数检验容器是否允许使用 OLE **控件。缺省的版本调用** GetLicenseKey **来获得一个有关控件的许可字符串的拷贝，并将这个拷贝与** bstrKey **中的字符串**

进行比较。如果两个字符串相匹配，函数就返回一个非零值，否则返回 0。

为了提供可定制的许可检验，你可以重载这个函数。

请参阅 COleObjectFactory::VerifyUserLicense，COleObjectFactory::GetLicenseKey

COleObjectFactory::VerifyUserLicense

virtual BOOL VerifyUserLicense();

返回值

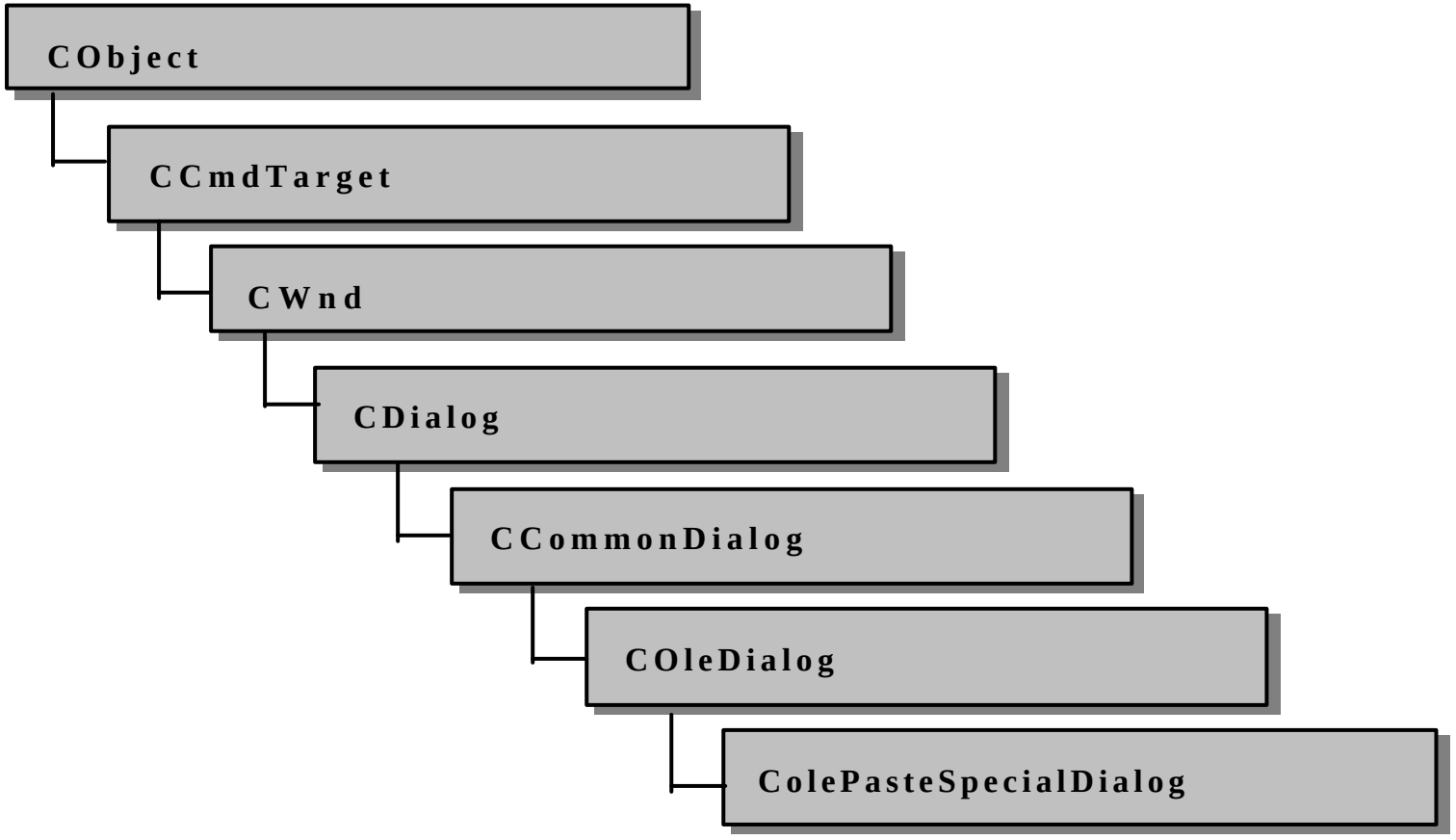
如果设计时许可有效则返回非零值；否则返回 0。

说明

检验 OLE 控件的设计时许可。

请　　参　　阅　COleObjectFactory::VerifyLicenseKey　，COleObjectFactory::GetLicenseKey

COlePasteSpecialDialog



COlePasteSpecialDialog 类用于 OLE Paste Special 对话框。当应用程序想要调用这个对话框时，可创建一个 COlePasteSpecialDialog 对象。当构造了 COlePasteSpecialDialog 对象之后，应用程序可用 m_ps 结构来初始化对话框各个控件的值或状态。m_ps 结构是 OLEUIPASTESPECIAL 类型。

如果要获取有关 OLE 专用对话框的信息，请参阅“Visual C++程序员指南”一书中的文章“OLE 中的对话框”。

#include <afxodlgs.h>

请参阅 COleDialog

COlePasteSpecialDialog 类成员

Data Members	
m_ps	一个控件对话框功能的 OLEUIPASTESPECIAL 类型的结构
Construction	
ColePasteSpecialDialog	构造一个 COlePasteSpecialDialog 对象
Operations and Attributes	
DoModal	显示 OLE Paste Special 对话框
AddFormat	将定制格式增加到应用程序可以粘贴的格式表中
AddStandardFormats	将 CF_BITMAP，CF_DIB，CF_METAFILEPICT 和 CF_LINKSOURCE 增加到应用程序的可粘贴格式表中
CreateItem	用指定格式在容器文档中创建项
GetSelectionType	获取所选择项的类型
GetDrawAspect	判定是否将项绘制成一个图标
GetIconicMetafile	获取与此项的图标格式相关联的源文件的句柄
GetPasteIndex	获取用户选择的可用粘贴项的索引

成员函数

COlePasteSpecialDialog::AddFormat

void AddFormat(const FORMATETC& *fmt*, LPTSTR *lpstrFormat*,
LPTSTR *lpstrResult*, DWORD *flags*);
void AddFormat(UINT *cf*, DWORD *tymed*, UINT *nFormatID*,
BOOL *bEnableIcon*, BOOL *bLink*);

参数

fmt

对要添加的数据类型的引用。

lpstrFormat

向用户描述格式的字符串。

lpstrResult

如果对话框中选择了此格式，则该参数为描述结构的字符串。

flags

指定此格式可用的不同链接和嵌入项。此标志是一个或多个不同的 OLEUIPASTEF- LAG 枚举值的位或组合。

cf

要添加的剪贴板格式。

tymed

格式中可用的媒介类型。它是一个或多个 TYMED 枚举值的位或组合。

nrormatID

标识格式的字符串的 ID。这个格式字符串是两个“ \n ”字符间隔的独立字符串。第一个字符串与 *lpstrFormat* 参数中传递的字符串相同；第二个字符串与 *lpstrResult* 参数相同。

bEnableIcon

一个标志，用于确定当在列表框中选择此格式时，Display As Icon 复选框是否活动。

bLink

一个标志，用于确定当在列表框中选择此格式时，Paste Link 单选按钮是否活动。

说明

这个函数将几种新格式增加到应用程序可在 Paste Special 操作中支持的格式表中。该函数可用于增加标准格式如 CF_TEXT 或 CF_TIFF，也可用于增加应用程序已向 OLE 系统注册的定制格式。有关向应用程序中粘贴数据对象的详细信息，参考“Visual C++程序员指南”一书中的文章“数据对象和数据源：操作”。

要获取更多的信息，请参见“OLE 2.01 用户接口库”中的 OLEUIPASTEFLAG 枚举类型。

请参阅 COlePasteSpecialDialog::AddStandardFormats

COlePasteSpecialDialog::AddStandardFormats

void AddStandardFormats(BOOL *bEnableLink* = TRUE);

参数

bEnableLink

一个标志，用于确定是否将 CF_LINKSOURCE **增加到应用程序可粘贴的格式表中。**

说明

此函数用于将一系列剪贴板格式增加到应用程序在 Paste Special **操作中可支持的格式表中：**

- CF_BITMAP

- CF_DIB
- CF_METAFILEPICT
- **“嵌入对象（Embedded Object）”**
- **（可选）“链接源（Link Source）”**

这些格式是用来支持嵌入和链接的。

请参阅 COlePasteSpecialDialog::AddFormat

COlePasteSpecialDialog::COlePasteSpecialDialog

COlePasteSpecialDialog(DWORD *dwFlags* = PSF_SELECTPASTE,
    COleDataObject* *pDataObject* = NULL, CWnd* *pParentWnd* = NULL);

参数

dwFlags

创建标志，包含下列标志中的任意多个，各值之间用位或 OR **操作符组**

合：

- PSF_SELECTPASTE 表明当调用对话框时，Paste 单选按钮被初始选中。该标志不能和 PSF_SELECTPASTELINK 一起使用。它是缺省值。
- PSF_SELECTPASTELINK 表明当调用对话框时，Paste Link 单选按钮被初始选中。该标志不能与 PSF_SELECTPASTE 一起使用。
- PSF_CHECKDISPLAYASICON 表明当调用对话框时，Display As Icon 复选框被初始选中。
- PSF_SHOWHELP 表明当调用对话框时，将显示 Help 按钮。

pDataObject

指向剪贴板上的 COleDataObject 对象。

pParentWnd

指向对话框对象所属的父或属主窗口对象（CWnd 类型）。如果该参数

为 NULL，则对话框的父窗口设置为主应用程序窗口。

说明

此函数只创建 COlePasteSpecialDialog 对象。要显示此对话框，需调用 DoModal 函数。

要获取更多的信息，请参见“OLE 2.01 用户接口库”中的 OLEUIPASTEFLAG 枚举类型。

请参阅 COleDataObject，COlePasteSpecialDialog::DoModal

COlePasteSpecialDialog::CreateItem

BOOL CreateItem(COleClientItem* *pNewItem*);

返回值

如果创建成功则返回非零值；否则返回 0。

参数

pNewItem

指向一个 COleClientItem **实例。该参数不能是** NULL。

说明

此函数创建在 Paste Special **对话框中所选择的新项。该函数只应在** DoModal **返回** IDOK **之后被调用。**

请参阅 COleClientItem，ColePasteSpecialDialog::DoModal，

COlePasteSpecialDialog::GetSelectionType，

COlePasteSpecialDialog:: COlePasteSpecialDialog

COlePasteSpecialDialog::DoModal

virtual int DoModal();

返回值

对话框完成的状态。它可以是下列值之一：

- IDOK　**如果成功显示了对话框。**
- IDCANCEL　**如果用户取消了对话框。**
- IDABORT　**如果返回的是** IDABORT**，则要调用** COleDialog::GetLastError **成员函数来获取有关所发生的错误类型的进一步信息。可能发生的错误的列表，请参见“**OLE 2.01 **用户接口库”中的** OleUIPasteSpecial **函数。**

说明

此函数用来显示 OLE Paste Special **对话框。**

如果你想通过设置 m_ps 结构的成员函数来初始化不同的对话框控件，则你必须在调用 DoModal 之前，但在构造了对话框对象之后进行。

如果 DoModal 返回 IDOK，则你可以调用其它成员函数来获取各个设置或由用户在对话框中输入的信息。

请参阅 COleDataObject，COleDialog::GetLastError，CDialog::DoModal，
COlePasteSpecialDialog:: COlePasteSpecialDialog，
COlePasteSpecialDialog::GetDrawAspect，
COlePasteSpecialDialog::GetIconicMetafile，
COlePasteSpecialDialog::GetPasteIndex，
COlePasteSpecialDialog::GetSelectionType

COlePasteSpecialDialog::GetDrawAspect

DVASPECT GetDrawAspect() const;

返回值

返回用来显示对象的方法。

- DVASPECT_CONTENT 如果当对话框消失时，Display As Icon 复选框未被选择，则返回此值。
- DVASPECT_ICON 如果当对话框消失时，Display As Icon 复选框被选择，则返回此值。

说明

此函数用来确定用户是否选择了将所选项显示为一个图标。仅在 DoModal 返回 IDOK 后才调用这个函数。

要获取更多有关绘制方面的信息，请参见“OLE 2 程序员参考，卷 1”中的

FORMATETC 结构。

请参阅 COlePasteSpecialDialog::DoModal

COlePasteSpecialDialog::GetIconicMetafile

HGLOBAL GetIconicMetafile() const;

返回值

如果在选择 OK **取消对话框时，**Display As Icon **复选框被选择，则返回值为包含所选项的图标特征的源文件句柄；否则返回值为** NULL。

说明

此函数获取与用户所选项相关联的源文件。

请参阅 COlePasteSpecialDialog::GetDrawAspect **，**

COlePasteSpecialDialog::DoModal

COlePasteSpecialDialog::GetPasteIndex

int GetPasteIndex() const;

返回值

返回值为用户所选项在 OLEUIPASTEENTRY **结构数组中的索引。在执行粘贴操作时，将使用对应于所选索引的格式。**

说明

此函数获取与用户所选项相关的索引值。

如果要获取更多的信息，请参见“ OLE 2.01 **用户接口库”中的** OLEUIPASTEENTRY **结构。**

请参阅 COlePasteSpecialDialog::DoModal

COlePasteSpecialDialog::GetSelectionType

UINT GetSelectionType() const;

返回值

返回所作的选择的类型。

说明

此函数用来决定用户所作的选择的类型。

返回类型的值由 Selection **枚举类型指定，该枚举类型是在** COlePasteSpecialDialog **类中声明的。**

```
enum Selection
{
```

```
    pasteLink,
    pasteNormal,
    pasteOther,
    pasteStatic,
};
```

有关这些值的简短说明如下所示：

- COlePasteSpecialDialog::pasteLink　Paste Link **单选按钮被选择，所选的格式是一个标准的** OLE **格式。**
- COlePasteSpecialDialog::pasteNormal　Paste **单选按钮被选择，所选的格式是一个标准的** OLE **格式。**
- COlePasteSpecialDialog::pasteOther　**被选择的格式不是一个标准的格式。**
- COlePasteSpecialDialog::pasteStatic　**被选择的格式是一个图元文件。**

请参阅　COlePasteSpecialDialog::DoModal

数据成员

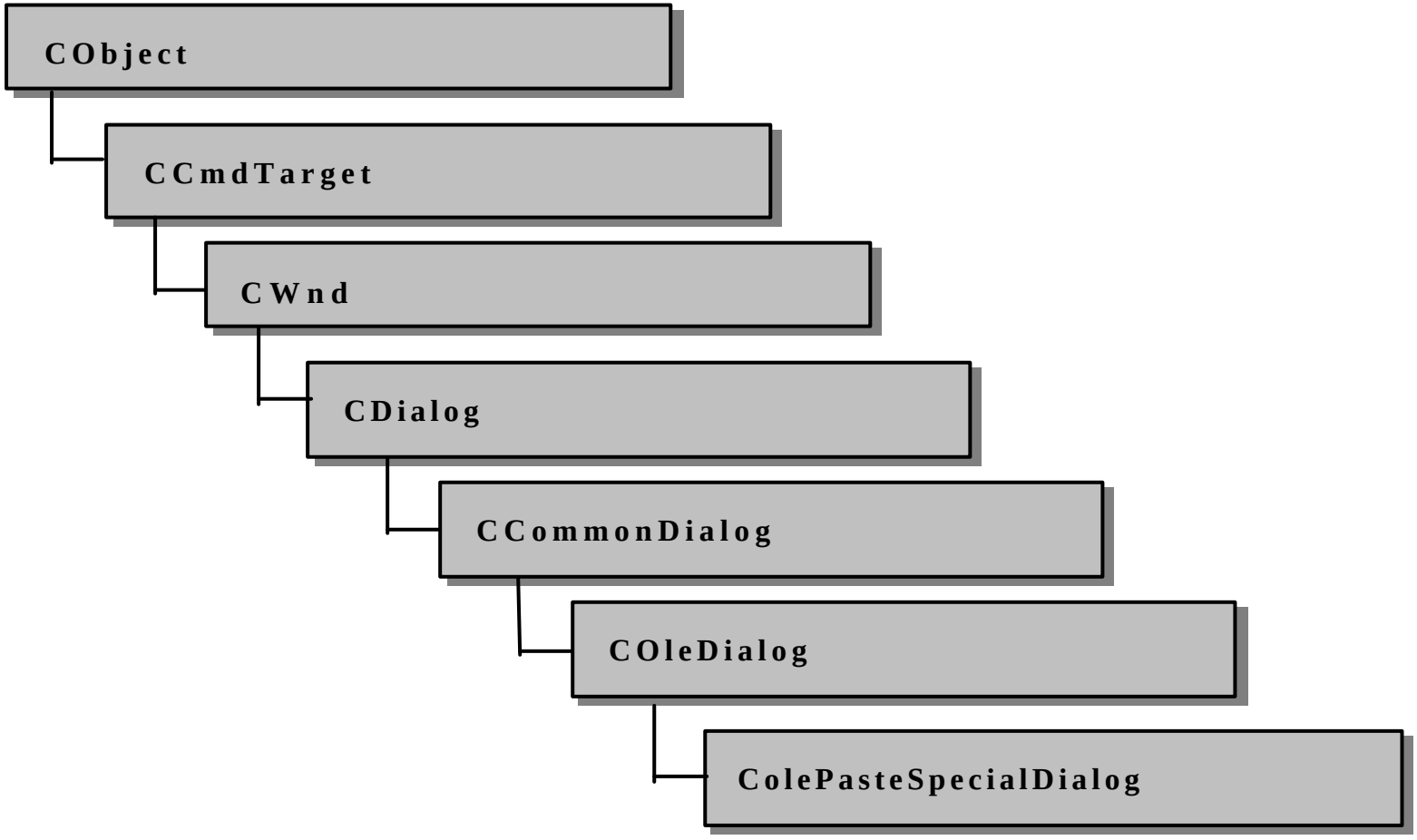
COlePasteSpecialDialog::m_ps

说明

此数据成员是一个 OLEUIPASTESPECIAL 类型的结构，用于控制 Paste Special 对话框的行为。该结构的成员可以被直接修改，也可以通过成员函数修改。

请参阅 COlePasteSpecialDialog:: COlePasteSpecialDialog，
COlePasteSpecialDialog::DoModal

COlePropertiesDialog



COlePropertiesDialog类封装了Windows通用的OLE 对象属性对话框。通用OLE对象属性对话框提供了用与 Windows 一致的风格来显示和修改 OLE 文档项属性的简单方法。这些属性包括，文档项所代表的文件的信息，用于显示按比例缩放的图标和图像的属性，以及有关项的链接的信息（如果该项是链接项的话），还有其它一些。

为了使用一个 COlePropertiesDialog 对象，首先要用 COlePropertiesDialog 构造函数来创建这个对象。在对话框被构造出来以后，调用 DoModal 成员函数来显示这个对话框并使用户能够修改这个项的任何属性。不管选择的是 OK（IDOK）按钮还是 Cancel（IDCANCEL）按钮 DoModal 都返回。除了 OK 和 Cancel 按钮，还有一个 Apply 按钮。当用户选择了 Apply 按钮时，任何有关这个文档项的属性的变化都应用到这个项中，其图像就会自动被更新，但是它仍然是活动的。

m_psh 数据成员是一个指向 PROPSHEETHEADER 结构的指针，在许多情况下，你不需要显式地访问它。但是当你在除了缺省的 General，View 和 Link 页外需要额外的页时，你就必须显式地访问它了。在这种情况下，你可以在调用 DoModal 成员函数之前来修改 m_psh 数据成员，使它包括你的定制页。

如果要获取更多的有关 OLE 对话框的信息，请参见“Visual C++程序员指南”一书中的文章“OLE 中的对话框”。

#include <afxodlgs.h>

请参阅 COleDialog，CPropertyPage

COlePropertiesDialog 类成员

Construction	
ColePropertiesDialog	构造一个 COlePropertiesDialog 对象

Data Members

m_gp	用来初始化 COlePropertiesDialog 对象的“General“页面的结构
m_lp	用来初始化 COlePropertiesDialog 对象的“Link“页面的结构
m_op	用来初始化 COlePropertiesDialog 对象的结构
m_psh	用来增加额外的定制属性页面的结构
m_vp	用来定制 COlePropertiesDialog 对象的“View“页面的结构

Operation

DoModal	显示对话框并允许用户作一个选择

Overridables

OnApplyScale	当文档项的缩放比例改变时由框架调用

成员函数

COlePropertiesDialog:: COlePropertiesDialog

COlePropertiesDialog(COleClientItem* *pItem*, UINT *nScaleMin* = 10,
UINT *nScaleMax* = 500, CWnd* *pParentWnd* = NULL);

参数

pItem

指向一个正被访问的文档项的指针。

nScaleMin

文档项图像的最小缩放比例百分比。

nScaleMax

文档项图像的最大缩放比例百分比。

pParentWnd

指向对话框的父或属主的指针。

说明

此成员函数用来创建一个 COlePropertiesDialog 对象。为了使你的文档项能够实现按比例缩放，请从 COlePropertiesDialog 类来派生你的通用 OLE Object Properties 对话框类。任何用这个类的实例实现的对话框都不能支持文档项的按比例缩放。

缺省的，通用 OLE Object Properties 对话框具有三个缺省的页面：

- General
  此页面包含了选中的文档项所代表的文件的系统信息。在这一页上，用户可以把选中的项转换为其他的类型。
- View
  此页面包含了显示这个文档项，改变图标，以及改变图像的缩放比例等选

项。

- Link
  此页面包含了改变链接项的位置和更新链接项的选项。在这个页面上，用户可以中断所选项的链接。

要增加除那些缺省页面之外的页面，在退出你的 COlePropertiesDialog 派生类的构造函数之前修改 m_psh 成员变量就可以了。这是 COlePropertiesDialog 构造函数的高级实现。

请参阅 COlePropertiesDialog::OnApplyScale

COlePropertiesDialog::DoModal

virtual int DoModal();

返回值

如果成功则返回 IDOK 或 IDCANCEL；否则返回 0。IDOK 和 IDCANCEL 都是常量，它们指示用户是选择了 OK 按钮还是 Cancel 按钮。

如果返回的是 IDCANCEL，你可以调用函数 Windows CommDlgExtendedError 来确定是否发生了一个错误。

说明

这个成员函数用来显示 Windows 通用 OLE Object Properties 对话框，并允许用户查看或改变文档项的各个属性。

请参阅 COlePropertiesDialog::OnApplyScale，ColePropertiesDialog::m_psh

COlePropertiesDialog::OnApplyScale

virtual BOOL OnApplyScale(COleClientItem* *pItem*, int *nCurrentScale*,
    BOOL *bRelativeToOrig*);

返回值

pItem

指向其属性正在被访问的文档项的指针。

nCurrentScale

对话框比例的数字值。

bRelativeToOrig

表明缩放比例是否适用于文档项的原始大小。

说明

当缩放比例值被改变，并且没有选择 OK **或** Apply **时，由框架调用这个函数。**

缺省的实现什么也没做。为了可以进行缩放比例控制，你必须重载这个函数。

> **注意** 在通用 OLE Object Properties 对话框显示之前，框架将 pItem 置为 NULL，将 nCurrentScale 置为-1 来调用这个函数。这用来决定是否可以进行缩放比例控制。

请参阅 COlePropertiesDialog::DoModal

数据成员

COlePropertiesDialog::m_gp

说明

是一个 OLEUIGNRLPROPS **类型的结构，它用来初始化** OLE Object Properties

对话框中的 General 页面。此页面显示了一个嵌入项的类型和大小，并且允许用户访问 Convert 对话框。如果对象是一个链接，这个页面也显示链接目标。

如果要获取更多有关 OLEUIGNRLPROPS 结构的信息，请参见 OLE 文档。

COlePropertiesDialog::m_lp

说明

一个 OLEUILINKPROPS 类型的结构，用来初始化 OLE Object Properties 对话框中的 Link 页面。此页面显示了被链接项的位置，并允许用户更新或中断指向这个项的链接。

要获取更多的有关 OLEUILINKPROPS 结构的信息，请参见 OLE 文档。

COlePropertiesDialog::m_op

说明

一个 OLEUIOBJECTPROPS **类型的结构，用来初始化** OLE Object Properties **对话框中的** Link **页面。这个结构包含了用来初始化** General，Link **和** View **页面的成员。**

要获取更多的信息，请参见 OLE **文档中的** OLEOBJECTPROPS **和** OLEUILINKPROPS **结构。**

COlePropertiesDialog::m_psh

说明

一个 PROPSHEETHEADER **类型的结构，它的成员保存了对话框对象的特征。**

在构造了一个 COlePropertiesDialog 对象之后，在调用 DoModal 成员函数之前，你可以使用 m_psh 来设置对话框的不同方面。

要获取更多有关 PROPSHEETHEADER 结构的信息，参见 Win32 SDK 文档。

请参阅 COlePropertiesDialog::DoModal

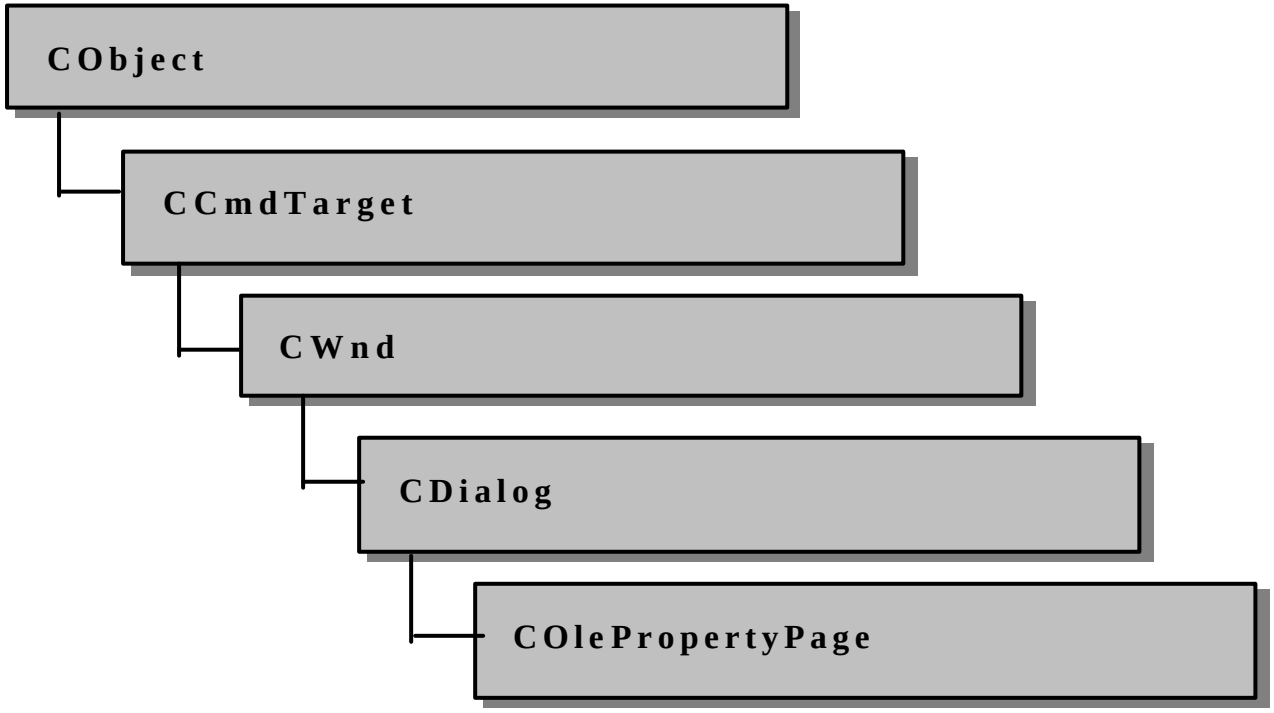
COlePropertiesDialog::m_vp

说明

一个 OLEUIOBJECTPROPS 类型的结构，用来初始化 OLE Object Properties 对话框中的 View 页面。此页面允许用户在“内容”和“图标”之间切换对对象的查看，并在容器内改变它的缩放比例。当对象显示为一个图标时，它也允许用户访问 Change Icon 对话框。

要获取更多有关 OLEUIVIEWPROPS 结构的信息，请参见 OLE 文档。

COlePropertyPage



COlePropertyPage 类以类似于对话框的图形化界面来显示定制控件的属性。例如，一个属性页可能会包含一个编辑控件，允许用户查看和修改控件的标题属

性。

每一个定制的或普通的控件属性都可以有一个对话框控件来允许控件的用户查看当前的属性值，如果有必要的话还可以修改这个值。

要获取更多有关使用 COlePropertyPage 的信息，请参见“Visual C++程序员指南”一书中的文章“ActiveX 控件：属性页”，和“Visual C++教程”中的“修改缺省的属性页”。

#include <afxctl.h>

请参阅 CDialog

COlePropertyPage 类成员

Construction	
ColePropertyPage	构造一个 ColePropertyPage 对象

Operations	
GetObjectArray	返回由属性页编辑的对象数组
Operations	
SetModifiedFlag	设置一个标志，该标志用来表明用户是否修改了属性页
IsModified	表明用户是否修改了属性页
GetPageSite	返回一个指向属性页的 IpropertyPageSite 接口的指针
SetDialogResource	设置属性页的对话框资源
SetPageName	设置属性页的名称（标题）
SetHelpInfo	设置属性页的简短帮助文本，帮助文件的名称，以及帮助的上下文
GetControlStatus	表明用户是否修改了控件中的值
SetControlStatus	设置一个标志来表明用户是否修改了控件中的值
IgnoreApply	确定是哪一个控件没有使能 Apply 按钮

Overridables	
OnEditProperty	当用户编辑一项属性时由框架调用
OnHelp	当用户激活帮助时由框架调用
OnInitDialog	当初始化属性页时由框架调用
OnObjectsChanged	当选择了其它具有新属性的 OLE 控件时由框架调用
OnSetPageSite	当属性框架提供页面的位置时由框架调用

成员函数

COlePropertyPage::COlePropertyPage

COlePropertyPage(UINT *idDlg*, UINT *idCaption*);

参数

idDlg

对话框模板的资源 ID。

idCaption

属性页的标题的资源 ID。

说明

当你实现 COlePropertyPage 的一个子类的时候，这个子类的构造函数应该使用 COlePropertyPage 的构造函数来标识对话框模板资源，这个资源是实现页的基础，并且字符串资源包含了它的标题。

COlePropertyPage::GetControlStatus

BOOL GetControlStatus(UINT *nID*);

返回值

如果控件值被改变则返回 TRUE；否则返回 FALSE。

参数

nID

一个实现页控件的资源 ID。

说明

此函数用来确定用户是否修改了属性控件的值，这个属性页控件由资源 ID **确定。**

请参阅 COlePropertyPage::SetControlStatus

COlePropertyPage::GetObjectArray

LPDISPATCH FAR* GetObjectArray(ULONG FAR* *pnObjects*);

返回值

指向一个 IDispatch **指针数组的指针，该指针数组用来访问属性页面上的每一个控件的属性。调用者不用释放这些接口指针。**

参数

pnObjects

指向一个无符号长整型数的指针，此整型数用来接收被这个页面编辑的对象数。

说明

每个属性页面对象维持一个指针数组，这些指针指向被页面编辑的对象的 IDispatch **接口。这个函数将它的参数** pnObjects **设置为指针数组的元素个数，并返回一个指向数组中的第一个元素的指针。**

COlePropertyPage::GetPageSite

LPPROPERTYPAGESITE GetPageSite();

返回值

指向属性页的 IPropertyPageSite **接口的指针。**

说明

此函数用来获取一个指向属性页的 IPropertyPageSite **接口的指针。**

控件和容器互相配合，以使用户能够浏览和修改控件的属性。控件提供属性页面，每一个页面都是一个 OLE **对象，允许用户编辑一个相关的属性集合。容器提供一个用来显示属性页面的属性框架。对应每一个页面，属性框架提供一个页面位置，它支持** IPropertyPageSite **接口。**

请参阅 COlePropertyPage::OnSetPageSite

COlePropertyPage::IgnoreApply

void IgnoreApply(UINT *nID*);

参数

nID

将被忽略的控件的 ID。

说明

只有当属性页控件的值被改变时，属性页上的 Apply **按钮才会被使能。这个函数用来指定一些控件，即使这些控件的值被改变了，也不会引起** Apply **按钮被使能。**

请参阅 COlePropertyPage::GetControlStatus

COlePropertyPage::IsModified

BOOL IsModified();

返回值

如果属性页被修改则返回 TRUE。

说明

此函数用来确定用户是否改变了属性页上的值。

请参阅 COlePropertyPage::SetModifiedFlag

COlePropertyPage::OnEditProperty

virual BOOL OnEditProperty(DISPID *dispid*);

返回值

缺省实现返回 FALSE。**重载这个函数后将返回** TRUE。

参数

dispid

被编辑的属性的分配 ID。

说明

当一个特定的属性被编辑时框架调用这个函数。为了将焦点设置到页面上特定的控件中，你可以重载这个函数。缺省的实现什么也不做，并返回 FALSE。

COlePropertyPage::OnHelp

virtual BOOL OnHelp(LPCTSTR *lpszHelpDir*);

返回值

缺省实现返回 FALSE。

参数

lpszHelpDir

包含属性页的帮助文件的目录。

说明

当用户请求联机帮助的时候框架调用这个函数。在用户访问帮助的时候，如果你的属性页面需要执行任何特定的操作，请重载这个函数。缺省的实现什么也

不做并返回 FALSE，这指示框架去调用 WinHelp。

COlePropertyPage::OnInitDialog

virtual BOOL OnInitDialog();

返回值

缺省实现返回 FALSE。

说明

当属性页的对话框被初始化时，框架调用这个函数。如果在对话框被初始化时需要执行任何特定的操作，请重载这个函数。缺省的实现调用了 CDialog::OnInitDialog，返回 FALSE。

请参阅　CDialog::OnInitDialog

COlePropertyPage::OnObjectsChanged

virtual void OnObjectsChanged();

说明

当在开发环境中查看一个 OLE 控件的属性时，一个无模式对话框被用来显示它的属性页面。如果选择了另一个控件，就必须显示一个具有不同设置的属性页面用来显示一个新的属性集合。框架调用这个函数来通知属性页面这个改变。

COlePropertyPage::OnSetPageSite

virtual void OnSetPageSite();

说明

当属性框架提供了属性页的页面位置时，框架调用这个函数。缺省的实现载入页面的标题，并尝试从对话框资源来确定页面的大小。如果你的属性页需要任何进一步的动作，请重载这个函数。

请参阅 COlePropertyPage::GetPageSite

COlePropertyPage::SetControlStatus

BOOL SetControlStatus(UINT *nID*, BOOL *IsDirty*);

返回值

如果指定的控件已被设置则返回 TRUE**；否则返回** FALSE**。**

参数

nID

包含一个属性页控件的 ID**。**

IsDirty

表明属性页的某个域是否被修改。如果域被修改则被设置为 TRUE**，如果还没有被修改则被设置为** FALSE**。**

说明

这个函数用来改变属性页控件的状态。

当属性页被关闭或 Apply **按钮被选择时，如果属性页控件的状态是变脏了的，则控件的属性将会被用适当的值更新。**

请参阅 COlePropertyPage::GetControlStatus

COlePropertyPage::SetDialogResource

void SetDialogResource(HGLOBAL *hDialog*);

参数

hDialog

属性页的对话框资源的句柄。

说明

此函数用来设置属性页的对话框资源。

COlePropertyPage::SetHelpInfo

void SetHelpInfo(LPCTSTR *lpszDocString*, LPCTSTR *lpszHelpFile* = NULL,
DWORD *dwHelpContext* = 0);

参数

lpszDocString

是一个包含简短帮助信息的字符串，用来显示在状态条或其他的位置中。

lpszHelpFile

属性页的帮助文件的名称。

dwHelpContext

属性页的帮助上下文。

说明

这个函数用来为属性页指定“工具提示”信息，帮助文件名，和帮助上下文。

请参阅 COlePropertyPage::OnHelp

COlePropertyPage::SetModifiedFlag

void SetModifiedFlag(BOOL *bModified* = TRUE);

参数

bModified

为属性页的修改标志指定新值。

说明

此函数用来表明用户是否修改了属性页面。

请参阅 COlePropertyPage::IsModified

COlePropertyPage::SetPageName

void SetPageName(LPCTSTR *lpszPageName*);

参数

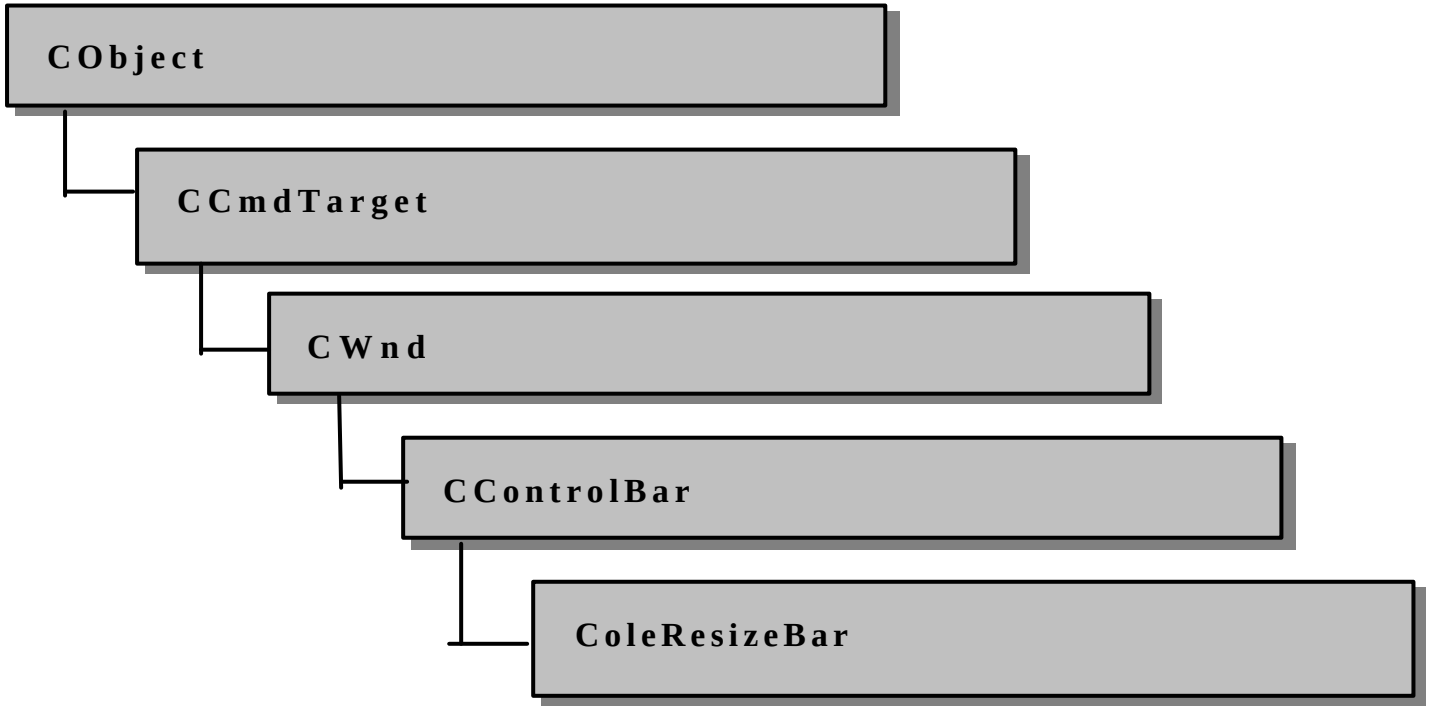
lpszPageName

是一个字符串指针，该字符串包含属性页的名称。

说明

此函数用来设置属性页的名字，这个名字是属性框架特别显示在页面标签上的。

COleResizeBar



COleResizeBar对象是一种支持改变现场OLE项的大小的控制条。COleResizeBar

对象看起来是一个具有带阴影边框和能处理外部大小改变的 CRectTracker 大小。

COleResizeBar 对象通常是从 COleIPFrameWnd 类派生的框架窗口对象的嵌入成员。

更多信息，请参见“Visual C++ 程序员指南”中的“Activation”。

#include <afxole.h>

请参阅 COleServerDoc

COleResizeBar 类成员

Construction	
ColeResizeBar	构造一个 COleResizeBar 对象
Create	创建并初始化一个 Windows 子窗口，并使它与 COleResizeBar 对象相联系

成员函数

COleResizeBar:: COleResizeBar

COleResizeBar();

说明

构造一个 COleResizeBar **对象。调用** Create **创建可改变大小的条对象。**

请参阅 COleResizeBar::Create

COleResizeBar::Create

BOOL Create(CWnd* *pParentWnd*, DWORD *dwStyle* = WS_CHILD|WS_VISIBLE,
UINT *nID* = AFX_IDW_RESIZE_BAR);

返回值

pParentWnd

如果可改变大小的条创建成功则返回非零值；否则返回 0。

dwStyle

指定窗口风格属性。

nID

可改变大小的条的子窗口 ID。

说明

创建一个子窗口并将其与 COleResizeBar **对象相联系。**

请参阅 CWnd::Create，CControlBar

COleSafeArray

COleSafeArray 类是用于处理任意类型和维数的数组的类。COleSafeArray 是从 OLE VARIANT 结构派生而来的。OLE SAFEARRAY 成员函数在可以通过 COleSafeArray 来访问，就象是特别为一维的字节数组所设计的一个成员函数集。

#include <afxdisp.h>

请参阅 COleVariant，CRecordset，CDatabase

COleSafeArray 类成员

Construction	
COleSafeArray	构造一个 COleSafeArray 对象

Operations	
Attach	给 COleSafeArray 对象以存在的 VARIANT 数组的控制
Clear	释放基 VARIANT 中的所有数据
Detach	将 VARIANT 数组从 COleSafeArray 对象中分离出来（这将使数据不会被释放）

Win32 API Wrappers	
AccessData	获取一个指向数组数据的指针
AllocData	为数组分配内存
AllocDescriptor	为安全数组描述符分配内存
Copy	创建一个已存在的数组的拷贝
Create	创建一个安全数组
Destroy	销毁一个已经存在的数组
DestroyData	销毁一个安全数组中的数据
DestroyDescriptor	销毁一个安全数组的描述符
GetDim	返回数组的维数
GetElement	获取安全数组中的一个单一元素
GetElemSize	返回安全数组中一个元素的按字节表示的大小
GetLBound	返回一个安全数组任一维的下界

续表

GetUBound	返回一个安全数组任一维的上界
Lock	增加一个数组的加锁计数，并将一个指向数组数据的指针放到数组描述符中
PtrOfIndex	返回一个指向被索引的元素的指针
PutElement	将一个单一的元素放入数组中
Redim	改变一个安全数组的最不重要（最右边）的边界
UnaccessData	减小一个数组的加锁计数，并使由 AccessData 获得的指针无效
Unlock	减小一个数组的加锁以使它能被释放或改变大小
One-Dimensional Array Operations	
CreateOneDim	创建一个一维的 COleSafeArray 对象
GetOneDimSize	返回一个一维的 COleSafeArray 对象中的元素个数
ResizeOneDim	改变一个一维的 COleSafeArray 对象中的元素个数

Operators	
Operator =	将一些值（包括 SAFEARRAY，VARIANT，ColeVariant，或 ColeSafeArray 对象）拷贝到 COleSafeArray 对象中
Operator ==	比较两个不同的数组（SAFEARRAY，VARIANT，ColeVariant，或 COleSafeArray 对象）
Operator <<	向转储描述表输出一个 COleSafeArray 对象的内容
Operator LPVARIANT	访问 COleSafeArray 对象的基 VARIANT 结构
Operator LPCVARIANT	访问 COleSafeArray 对象的基 VARIANT 结构

成员函数

COleSafeArray::AccessData

void AccessData(void* *ppvData*);

参数

ppvData

指向数组数据的指针的指针。

说明

此成员函数用来返回一个指向数组数据的指针。如果出错，则函数抛出一个 CMemoryException **或** COleException**。**

请参阅 COleSafeArray::UnaccessData**，** SafeArrayAccessData

COleSafeArray::AllocData

void AllocData();

说明

此成员函数用来为一个安全数组分配内存。如果出错，则函数抛出一个 CMemoryException **或** COleException。

请参阅 COleSafeArray::AllocDescriptor，SafeArrayAllocData

COleSafeArray::AllocDescriptor

void AllocDescriptor(DWORD *dwDims*);

参数

dwDims

安全数组中的维数。

说明

此函数用来为一个安全数组的描述符分配内存。如果出错，则函数抛出一个 CMemoryException **或** COleException。

请参阅 COleSafeArray::AllocData，SafeArrayAllocDescriptor

COleSafeArray::Attach

void Attach(VARIANT& *varSrc*);

参数

varSrc

是一个 VARIANT **对象。参数** *varSrc* **必须有** VARTYPE VT_ARRAY。

说明

此成员函数将对一个已有的VARIANT数组中的数据的控制传给COleSafeArray对象。源VARIANT的类型被设置为VT_EMPTY。如果当前数组有数据的话，函数将清掉这些数据。

请参阅 COleSafeArray::Detach

COleSafeArray::Clear

void Clear();

说明

此函数用来清除安全数组。函数通过将对象的VARTYPE设置为VT_EMPTY来清除一个安全数组。数组中的当前内容被释放，数组也被释放了。

请参阅 VariantClear

COleSafeArray::COleSafeArray

```
COleSafeArray();
COleSafeArray( const SAFEARRAY& saSrc,VARTYPE vtSrc );
COleSafeArray（ LPCSAFEARRAY psaSrc, VARTYPE vtSrc ）;
COleSafeArray( const COleSafeArray& saSrc );
COleSafeArray( const VARIANT& varSrc );
COleSafeArray( LPCVARIANT pSrc );
COleSafeArray( const COleVariant& varSrc );
```

参数

saSrc

要被拷贝到新的 COleSafeArray **对象中去的已经存在的** COleSafeArray **对象或** SAFEARRAY。

vtSrc

新的 COleSafeArray 对象的 VARTYPE。

psaSrc

一个指向要被拷贝到新的 COleSafeArray 对象中去的 SAFEARRAY 的指针。

varSrc

要被拷贝到新的 COleSafeArray 对象中去的已经存在的 VARIANT 或者 COleVariant。

pSrc

一个指向要被拷贝到新的 COleSafeArray 对象中去的 VARIANT 对象的指针。

说明

所有这些构造函数都创建一个新的 COleSafeArray 对象。如果没有参数，则创

建的是一个空的 COleSafeArray 对象（VT_EMPTY）。如果 COleSafeArray 是从另一个数组拷贝来的，并且这个数组的 VARTYPE 并不是完全确定的（一个 COleSafeArray，COleVariant，或者 VARIANT），则源数组中的 VARTYPE 被保留，并且不需要说明。如果 COleSafeArray 是从另一个数组拷贝而来，并且该数组的 VARTYPE 是不知道的，则 VARTYPE 必须用 vtSrc 参数来指定。

如果出错，则函数抛出一个 CMemoryException 或 COleException。

请参阅 VariantCopy

COleSafeArray::Copy

void Copy(LPSAFEARRAY* *ppsa*);

参数

ppsa

指向一个特定区域的指针，在这个特定区域中将返回新的数组描述符。

说明

创建一个已经存在的安全数组的拷贝。如果出错，则函数抛出一个 CMemoryException 或 COleException。

请参阅 SafeArrayCopy

COleSafeArray::Create

void Create(VARTYPE *vtSrc*, DWORD *dwDims*, DWORD* *rgElements*);
void Create(VARTYPE *vtSrc*, DWORD *dwDims*, SAFEARRAYBOUND* *rgsabounds*);

参数

vtSrc

是数组的基本类型（也就是说，数组的每一个元素的 VARTYPE）。VARTYPE 被限制在可变类型的一个子集。不能设置为 VT_ARRAY 和 VT_BYREF 标志。对数组来说 VT_ARRAY 和 VT_BYREF 是无效的基本类型。所有其它的类型都是合法的。

dwDims

数组中的维数。在数组创建之后，可以用 Redim 来改变这个值。

rgElements

指向一个数组的指针。该数组中记录着被创建的数组的每一维的元素个数。

rgsabounds

指向一个分配给数组的界线向量（每一维一个）的指针。

说明

此函数用来给数组分配和初始化数据。如果必要的话，这个函数将清除当前的数组数据。如果出错，则函数抛出一个 CMemoryException。

请参阅 SafeArrayCreate

COleSafeArray::CreateOneDim

```
void CreateOneDim( VARTYPE vtSrc, DWORD dwElememts,
    const void* pvSrcData = NULL,long nLBound = 0 );
```

参数

vtSrc

是数组的基本类型（也就是说，数组的每一个元素的 VARTYPE）。

dwElements

数组的维数。在数组被创建之后可以用 ResizeOneDim **来改变这个值。**

pvSrcData

指向要拷贝到数组中去的数据的指针。

nLBound

数组的下界。

说明

此函数用来创建一个一维的 COleSafeArray 对象。函数为数组分配并初始化数据，如果指针 pvSrcData 不是 NULL，则将指定的数据拷贝到数组中。

如果出错，则函数抛出一个 CMemoryException。

请参阅 COleSafeArray::GetOneDimSize，COleSafeArray::ResizeOneDim，COleSafeArray::Create

COleSafeArray::Destroy

```
void Destroy();
```

说明

此函数用来销毁一个已经存在的数组描述符和数组中的所有数据。如果数组中保存有对象，则每一个对象都被释放。如果出错，则函数抛出一个 CMemoryException 或 COleException。

请参阅 COleSafeArray::DestroyData，COleSafeArray::DestroyDescriptor，SafeArrayDestroy

COleSafeArray::DestroyData

void DestroyData();

说明

此函数销毁一个安全数组中的所有数据。如果数组中保存有对象，则每一个都被释放。如果出错，则函数抛出一个 CMemoryException 或 COleException。

请参阅 COleSafeArray::DestroyData，COleSafeArray::DestroyDescriptor，SafeArrayDestroyData

COleSafeArray::DestroyDescriptor

void DestroyDescriptor();

说明

此函数用来销毁一个安全数组的描述符。如果出错，则函数抛出一个 CMemoryException **或** COleException。

请参阅 COleSafeArray::DestroyData，COleSafeArray::DestroyDescriptor，SafeArrayDestroyDescriptor

COleSafeArray::Detach

VARIANT Detach();

返回值

返回 COleSafeArray **对象中的基** VARIANT **值。**

说明

此成员函数用来将 VARIANT **数据从** COleSafeArray **对象中分离出来。函数通过将对象的** VARTYPE **设置为** VT_EMPTY **来分离出安全数组中的数据。调用者通过调用** Windows**函数** VariantClear **来负责释放数组。**

如果出错，则函数抛出一个 COleException**。**

请参阅 COleSafeArray::Attach，VariantClear

COleSafeArray::GetDim

DWORD GetDim();

返回值

安全数组的维数。

说明

此函数返回 COleSafeArray **对象中的维数。**

请参阅 COleSafeArray::Create，COleSafeArray::Redim，SafeArrayGetDim

COleSafeArray::GetElement

void GetElement(long* *rgIndices*, void* *pvData*);

参数

rgIndices

指向一个数组的指针。该数组包含安全数组的每一维的索引。

pvData

指向用来放置数组的元素的特定区域的指针。

说明

此函数用来获取安全数组中的一个单一的元素。在获取元素之前和获取元素之后，此函数自动调用 Windows 函数 SafeArrayLock 和 SafeArrayUnlock。如果此数据元素是一个字符串，对象或可变类型，则函数会用正确的方法来拷贝这个元素。参数 pvData 应该指向足够大的缓冲区以保存这个元素。

如果出错，函数将抛出一个 CMemoryException 和 COleException。

请参阅 COleSafeArray::PutElement，SafeArrayGetElement

COleSafeArray::GetElemSize

DWORD GetElemSize();

返回值

以字节表示的安全数组中的元素的大小。

说明

此函数用来获取一个 COleSafeArray **大小中的一个元素的大小。**

请参阅 COleSafeArray::GetDim**，** SafeArrayGetElemSize

COleSafeArray::GetBound

void GetBound(DWORD *dwDim*, long* *pLBound*);

参数

dwDim

要获得其下界的数组的维数。

b pLBound

指向要返回下界的位置的指针。

说明

此成员函数返回一个 COleSafeArray **对象的任何维的下界。如果出错，则函数抛出一个** COleException。

请参阅 COleSafeArray::GetUBound，SafeArrayGetLBound

COleSafeArray::GetOneDimSize

DWORD GetOneDimSize();

返回值

一维安全数组中的元素个数。

说明

此成员函数用来返回一个一维的 COleSafeArray **对象的元素个数。**

请参阅 COleSafeArray::CreateOneDimSize，COleSafeArray::ResizeOneDim，SafeArrayRedim

COleSafeArray::GetUBound

void GetUBound (DWORD *dwDim*, long* *pUBound*);

参数

dwDim

要获得其上界的数组的维数。

pUBound

指向要返回上界的位置的指针。

说明

此成员函数返回一个 COleSafeArray 对象的任何维的上界。如果出错，则函数抛出一个 COleException。

请参阅 COleSafeArray::GetLBound，SafeArrayGetUBound

COleSafeArray::Lock

void Lock();

说明

此成员函数用来增加一个数组的加锁计数，并在数组描述符中放置一个指向数组数据的指针。如果出错，则函数抛出一个 COleException。

数组描述符中的指针一直有效，直到调用了 Unlock。对 Lock 的调用可以是嵌

套的；这时就需要对 Unlock 进行相同次数的调用。

当一个数组被锁住时是不能被删除的。

请参阅 COleSafeArray::Unlock，SafeArrayLock

COleSafeArray::PtrOfIndex

void PtrOfIndex(long* *rgIndices*, void** *ppvData*);

参数

rgIndices

一个索引值数组。这些索引值用来标识数组中的一个元素。元素的索引都必须被指明。

ppvData

在返回时，指向由 rgIndices 中的值标识的元素。

说明

此函数返回一个指向由索引值指定的元素的指针。

请参阅 SafeArrayPtrOfIndex

COleSafeArray::PutElement

void PutElement(long* *rgIndices*，LPVOID *pvData*);

参数

rgIndices

指向一个数组的指针。该数组包含安全数组的每一个元素的索引。

pvData

指向要赋给数组的数据的指针。VT_DISPATCH，VT_UNKNOWN，**和** VT_BSTR **可变类型都是指针，它们不需要其它级别的迂回。**

说明

这个函数用来将一个单一的元素赋给数组。在分配元素之前和分配元素之后，此函数自动调用 Windows 函数 SafeArrayLock 和 SafeArrayUnlock。如果数据元素是一个字符串，对象或可变类型，则函数会正确拷贝它们，并且如果已经存在的元素是一个字符串，对象，或可变类型，它们将被清除。

注意你可以对一个数组进行多重锁定，这样当数组被其它操作加锁时，你仍然可以将元素放入这个数组中。

如果出错，函数将抛出一个 CMemoryException 或 COleException。

请参阅 COleSafeArray::GetElement，SafeArrayPutElement

COleSafeArray::Redim

void Redim(SAFEARRAYBOUND* *psaboundNew*);

参数

psaboundNew

指向一个新的安全数组边界结构的指针，这个边界结构包含新数组的边界。只有数组的最不重要的维是可以改变的。

说明

此函数用来改变一个安全数组的最不重要的（最右边的）边界。如果出错，函数将抛出一个 COleException。

请参阅 COleSafeArray::Create，COleSafeArray::GetDim，COleSafeArray::ResizeOneDim，SafeArrayRedim

COleSafeArray::ResizeOneDim

void ResizeOneDim(DWORD *dwElements*);

参数

dwElements

一维安全数组中的元素个数。

说明

此函数用来改变一个一维 COleSafeArray **对象的元素个数。如果错误，函数将抛出一个** COleException。

请参阅 COleSafeArray::Redim，COleSafeArray::GetOneDimSize，COleSafeArray::CreateOneDim，SafeArrayRedim

COleSafeArray::UnaccessData

void UnaccessData();

说明

此函数用来减小一个数组中的锁定计数，并使由 AccessData **获取的指针无效。**

如果出错，函数将抛出一个 COleException**。**

请参阅 COleSafeArray::AccessData**，** SafeArrayUnaccessData

COleSafeArray::Unlock

void Unlock();

说明

此函数用来减小一个数组的锁定计数，以使数组能够被释放或改变大小。此函

数在对数组中的数据的访问完成之后被调用。如果出错，函数将抛出一个 COleException。

请参阅 COleSafeArray::Lock，SafeArrayUnlock

操作符

COleSafeArray::operator =

```
COleSafeArray& operator = ( const COleSafeArray& saSrc );
COleSafeArray& operator = ( const VARIANT& varSrc );
COleSafeArray& operator = ( LPCVARIANT pSrc );
COleSafeArray& operator = ( const COleVariant& varSrc );
```

说明

这些重载的赋值操作符将源值拷贝到 COleSafeArray 对象。一个有关每个操作

符的简短描述如下所示：

- operator = (saSrc) 将一个已经存在的 COleSafeArray 对象拷贝到这个对象中。
- operator = (varSrc) 将一个已经存在的 VARIANT 或 COleVariant 数组拷贝到这个对象中。
- operator = (pSrc) 将一个由 pSrc 访问的 VARIANT 数组拷贝到这个对象中。

请参阅 VariantCopy

COleSafeArray::operator ==

```
BOOL operator = = ( const SAFEARRAY& saSrc ) const;
BOOL operator = = ( LPCSAFEARRAY pSrc ) const;
BOOL operator = = ( const COleSafeArray& saSrc ) const;
BOOL operator = = ( const VARIANT& varSrc ) const;
BOOL operator = = ( LPCVARIANT pSrc ) const;
```

BOOL operator = = (const COleVariant& *varSrc*) const;

说明

这个操作符比较两个数组（SAFEARRAY**，**VARIANT**，**COleVariant**，或**COleSafeArray **数组），如果它们相等则返回非零值；否则就返回零。如果两个数组具有相同的维数，每一维的大小相等，且元素值相等，则它们就是相等。**

COleSafeArray::operator <<

CDumpContext& AFXAPI operator<<(CDumpContext& *dc* , COleSafeArray& *saSrc*);

说明

此强制转换操作符用来访问 COleSafeArray **对象的基** VARIANT **结构。**

COleSafeArray::operator LPVARIANT

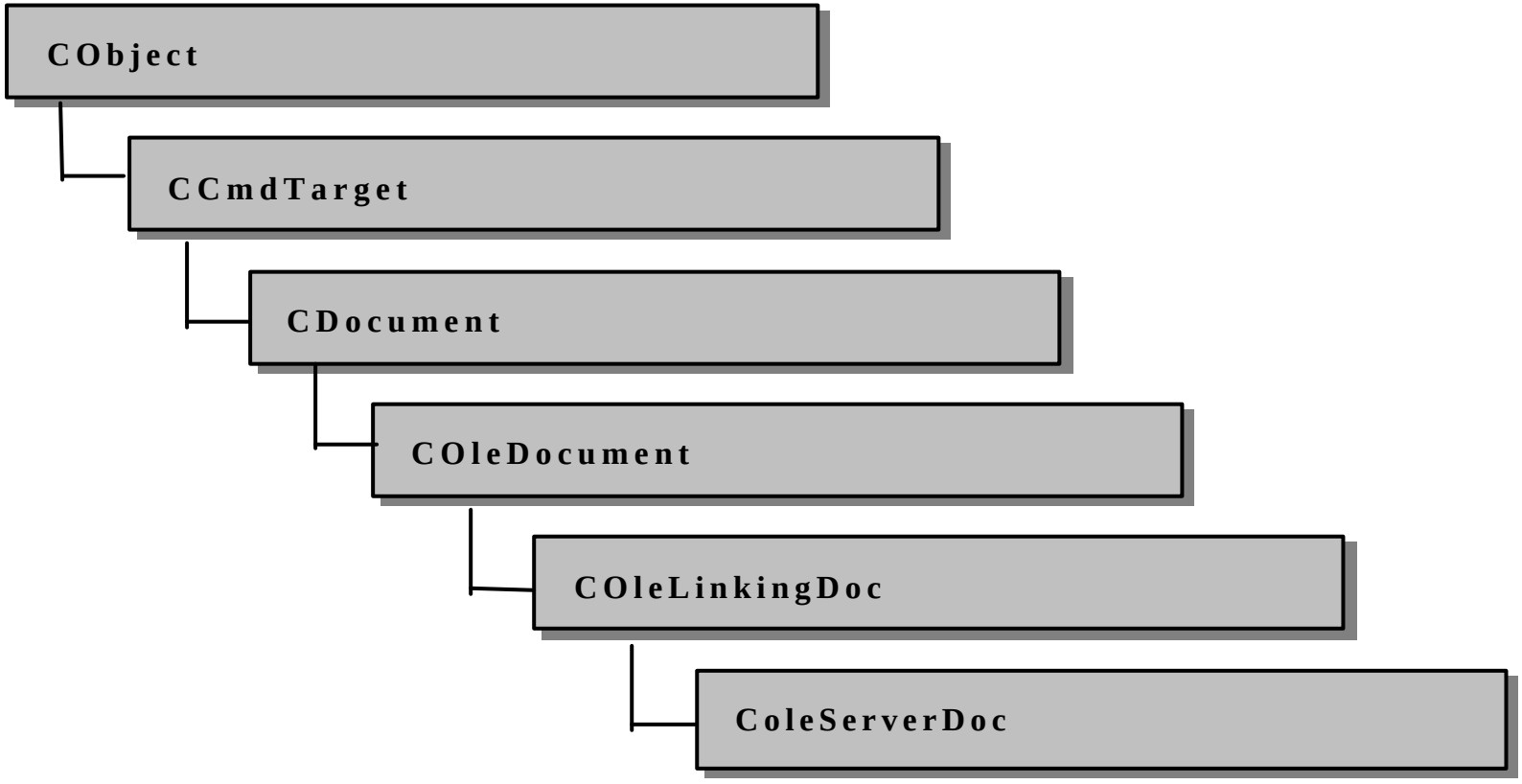
operator LPVARIANT();

说明

此强制转换操作符用来访问 COleSafeArray **对象的基** VARIANT **结构。**

注意，改变由这个函数的返回指针访问的 VARIANT **结构的值，将改变这个** COleSafeArray **对象的值。**

COleServerDoc



COleServerDoc 是 OLE 服务器文档的基类。服务器文档可包含 COleServerItem

对象，这些对象表示提供给嵌入或链接项的服务器接口。当服务器应用程序由容器应用程序启动来编辑嵌入项时，则将该项加载为服务器自己的服务器文档；COleServerDoc 对象只包含一个 COleServerItem 对象，构成了整个文档。当服务器应用程序由容器应用程序启动来编辑链接项时，则从磁盘装载一个现有的文档，一部分文档内容被加亮以指示链接项。

COleServerDoc 对象也可包含 COleClientItem 类的项。这使得你可以创建容器－服务器应用程序。框架在为 COleServerDoc 对象提供服务的同时，也提供了用于存储 COleClientItem 项的函数。

如果服务器应用程序不支持链接，则服务器程序文档将总是只包含一个服务器项，该项将整个嵌入对象表示为一个文档。如果服务器应用程序支持链接，则每当将一个选择拷贝到剪贴板中时，它都必须创建一个服务器项。

要使用 COleServerDoc，可从 COleServerDoc 类派生一个新类，并实现

OnGetEmbeddedItem 成员函数，该函数使得服务器应用程序能够支持嵌入项。你可以从 COleServerItem 派生一个新类来实现文档中的嵌入项，并从 OnGetEmbeddedItem 函数返回该类的这些对象。

为支持链接项，COleServerDoc 提供了 OnGetLinkedItem 成员函数。应用程序可以利用该函数的缺省实现，或者当应用程序有自己的管理文档项的方式时，可以重载这个函数。

应用程序需要为它所支持的每种服务器文档类型派生一个 COleServerDoc 派生类。例如，如果服务器应用程序支持工作表格和图表，则它需要两个 COleServerDoc 派生类。

#include <afxole.h>

请参阅 COleDocument，COleLinkingDoc，COleTemplateServer

COleServerDoc 类成员

Construction	
ColeServerDoc	构造一个 COleServerDoc 对象

Attributes	
IsEmbedded	表明文档是嵌入在一个容器文档中的还是独立运行的
IsInPlaceActive	如果此项是当前现场被激活的则返回 TRUE
GetEmbeddedItem	返回一个指向代表整个文档的项的指针
GetItemPosition	返回当前位置矩形用于现场编辑，该矩形是相对于容器应用程序的客户区的
GetItemClipRect	返回当前用于现场编辑的剪贴板矩形
GetZoomFactor	返回用像素表示的缩放因子

Operations	
OnExecOleCmd	执行一个指定的命令或显示与该命令相关的帮助
NotifyChanged	通知容器应用程序用户已经改变了文档
NotifyRename	通知容器应用程序用户已经将文档改名了
NotifySaved	通知容器应用程序用户已经保存了文档

续表

NotifyClosed	通知容器应用程序用户已经关闭了文档
SaveEmbedding	通知容器应用程序保存文档
ActivateInPlace	激活文档用于现场编辑
DeactivateAndUndo	使服务器的用户界面不再是活动的
DiscardUndoState	丢弃取消状态信息
RequestPositionChange	改变现场可编辑框架的位置
ScrollContainerBy	滚动容器程序文档
UpdateAllItems	通知容器应用程序用户已经改变了文档
Overridables	
GetDocObjectServer	重载这个函数可以创建一个新的CDocObjectServer对象，并表明这个文档是一个DocObject容器
OnUpdateDocument	当保存一个作为嵌入项的服务器文档时，框架调用该函数来更新此项的容器应用程序的备份
OnGetEmbeddedItem	调用此函数可获取一个表示整个文档的COleServerItem；此函数可用来获取一个嵌入项。需要实现

续表

OnClose	当容器应用程序请求关闭文档时，框架调用这个函数
OnSetHostNames	当一个容器为一个嵌入对象设置窗口标题时，框架调用这个函数
OnShowDocument	框架调用此函数来显示或隐藏文档
OnDeactivate	当用户使一个现场被激活的项变为不活动时，框架调用此函数
OnDeactivateUI	框架调用此函数来销毁控件和其它为现场激活所创建的用户界面元素。
OnSetItemRects	框架调用此函数来定位现场编辑框架窗口在容器应用程序窗口中的位置
OnReactivateAndUndo	框架调用此函数来取消在现场编辑期间所做的改变
OnFrameWindowActivate	当容器的框架窗口被激活或成为不活动时，框架调用这个函数
OnDocWindowActivate	当容器应用程序的文档框架窗口被激活或成为不活动时，框架调用这个函数
OnShowControlBars	框架调用此函数来显示或隐藏用于现场编辑的控制条

续表

OnResizeBorder	当容器应用程序的框架窗口或文档窗口被改变大小时，框架调用这个函数
CreateInPlaceFrame	框架调用此函数来创建一个用于现场编辑的框架窗口
DestroyInPlaceFrame	框架调用此函数来销毁一个用于现场编辑的框架窗口

成员函数

COleServerDoc::ActivateInPlace

BOOL ActivateInPlace();

返回值

如果成功则返回非零值；否则返回 0，**这表明该项已经被完全打开了。**

说明

此成员函数用来激活用于现场编辑的项。

此函数执行在现场激活中必须的所有操作。它创建一个现场框架窗口，激活它并设置此项的大小，设置共享菜单和其它控件；将此项滚动到视窗中，并将焦点设置到现场框架窗口。

COleServerItem::OnShow 的缺省实现调用这个函数。如果你的应用程序支持其它用于现场激活的动词（比如 Play），则请你调用这个函数。

请参阅 COleServerItem::OnShow

COleServerDoc::COleServerDoc

COleServerDoc();

说明

当不与 OLE 系统 DLLs 连接，此成员函数用来构造一个 COleServerDoc 对象。应用程序必须调用 COleLinkingDoc::Register 函数来打开与 OLE 的通讯。如果应用程序中使用的是 COleTemplateServer，则由 COleLinkingDoc 实现的 OnNewDocument，OnOpenDocument 和 OnSaveDocument 为应用程序调用 COleLinkingDoc::Register 函数。

请参阅 COleServerItem::OnShow

COleServerDoc::CreateInPlaceFrame

virtual COleIPFrame Wnd* CreatInPlaceFrame(CWnd* *pParentWnd*);

返回值

如果函数成功则返回一个指向现场框架窗口的指针；否则返回 NULL。

参数

pParentWnd

指向容器应用程序的父窗口的指针。

说明

框架调用此函数来创建一个用于现场编辑的框架窗口。该函数的缺省实现是使用文档模板中指定的信息来创建框架窗口。所使用的视是为文档创建的第一个视。这个视暂时从原来的框架分离，并连接到这个刚创建的框架窗口上。

此函数是一个高级的可重载函数。

请参阅 COleServerDoc::DestroyInPlaceFrame

COleServerDoc::DeactivateAndUndo

BOOL DeactivateAndUndo();

返回值

如果成功则返回非零值；否则返回 0。

说明

如果你的应用程序支持 Undo，**并且用户在激活一个项之后，但在对它进行编辑之前选择了** Undo，**则调用这个函数。如果容器应用程序是用微软基础类库编写的，则调用** DeactivateAndUndo **函数会导致** COleClientItem::OnDeactivateAndUndo **被调用，后者使服务器应用程序的用户**

界面变成不活动的。

请参阅 COleClientItem::OnDeactivateAndUndo

COleServerDoc::DestroyInPlaceFrame

virtual void DestroyInPlaceFrame(COleIPFrameWnd* *pFrame*);

参数

pFrame

指向将要被销毁的现场框架窗口的指针。

说明

框架调用这个函数来销毁一个现场框架窗口，并将服务器应用程序的文档窗口返回到其在现场激活之前的状态。

此函数是一个高级的可重载函数。

请参阅 COleServerDoc::CreateInPlaceFrame

COleServerDoc::DiscardUndoState

BOOL DiscardUndoState();

返回值

如果成功则返回非零值；否则返回 0。

说明

如果用户执行一个不能被取消的编辑操作，则调用该函数来强制容器应用程序丢弃它的取消状态信息。

这个函数使得支持 Undo 的服务器应用程序可以释放资源，否则这些资源就要

被不能再使用的取消状态信息浪费掉。

请参阅 COleServerDoc::OnReactivateAndUndo

COleServerDoc::GetDocObjectServer

virtual CDocObjectServer* GetDocObjectServer(LPOLEDOCUMENTSITE *pSite*);

返回值

如果操作成功则返回一个指向 CDocObjectServer 的指针；否则返回 NULL。

参数

pSite

指向 IOleDocumentSite 界面的指针，此界面将这个文档与服务器连接起来。

说明

重载这个函数可以创建一个新的 CDocObjectServer **项，并返回一个指向这个项的指针。当一个** DocObject **服务器被激活时，就返回一个非** NULL **指针，它表明客户可以支持** DocObjects**。函数的缺省实现返回** NULL**。**

一个支持 DocObjects **的文档的典型实现将简单地分配一个新的** CDocObjectServer **对象，并将它返回给调用者。例如：**

```
CDocObjectServer*
COleServerDoc::GetDocObjectServer( LPOLEDOCUMENTSITE pSite )
{
   return new CDocObjectServer( this,pSite);
}
```

请参阅 CDocObjectServer::CDocObjectServer

COleServerDoc::GetEmbeddedItem

COleServerItem* GetEmbeddedItem();

返回值

如果成功则返回一个指向代表整个文档的项的指针；否则返回 NULL。

说明

此函数用来获取一个指向代表整个文档的项的指针。它调用了 COleServerDoc::OnGetEmb- eddedItem **函数，这是一个没有缺省实现的虚函数。**

请参阅 COleServerDoc::OnGetEmbeddedItem

COleServerDoc::GetItemClipRect

void GetItemClipRect(LPRECT *lpClipRect*) const;

参数

lpClipRect

指向一个 RECT 结构或 CRect 对象的指针，此结构或对象用来接收此项的剪贴矩形坐标。

说明

调用这个函数来获取正在进行现场编辑的项的剪贴矩形坐标。坐标是相对于容器应用程序窗口的客户区，用像素表示的。

绘制不应该发生在剪贴矩形之外。通常，绘制会自动受到限制，利用这个函数可以确定用户是否滚动到了文档的可见部分之外；如果是，则通过调用 ScrollContainerBy 函数按需要滚动容器文档。

请参阅 COleServerDoc::GetItemPosition，COleServerDoc::ScrollContainerBy

COleServerDoc::GetItemPosition

void GetItemPosition(LPRECT *lpPosRect*) const;

参数

lpPosRect

指向一个 RECT **结构或** CRect **对象的指针，此结构或对象用来接收此项的坐标。**

说明

调用这个成员函数来获取正被现场编辑的项的坐标。坐标相对于容器应用程序窗口的客户区，用像素表示。

项的位置可以与当前剪贴矩形相比较，用于确定此项在屏幕上可见（或不可见）的程度。

请参阅 COleServerDoc::GetItemClipRect

COleServerDoc::GetZoomFactor

BOOL GetZoomFactor(LPSIZE *lpSizeNum* = NULL, LPSIZE *lpSizeDenom* = NULL,
LPCRECT *lpPosRect* = NULL) const;

返回值

如果此项是为现场编辑而被激活的，并且它的缩放因子不是 100%（1:1），**则返回一个非零值；否则返回值为** 0。

参数

lpSizeNum

指向一个 CSize **类对象的指针，此对象将用来保存缩放因子的分子。该参数可为** NULL。

lpSizeDenom

指向一个 CSize 类对象的指针，此对象将用来保存缩放因子的分母。该参数可为 NULL。

lpPosRect

指向一个 CRect 类对象的指针，此对象用于描述此项的新位置。如果该参数为 NULL，则该函数使用此项的当前位置。

说明

GetZoomFactor 成员函数确定一个为现场编辑而激活的“缩放因子”。缩放因子是此项的大小与其当前范围之比（用像素表示），如果容器应用程序没有设置此项的范围，则使用它的固有范围（由 COleServerItem::OnGetExtent 函数确定）。

GetZoomFactor 函数将它的前两个参数设为此项的“缩放因子”的分子和分母。

如果此项未被现场编辑，则该函数将这些参数设置为缺省值 100%（或 1:1），并返回零。如果要获取更进一步的信息，请参见联机技术注释 40，“MFC/OLE 现场改变大小和缩放”。

请参阅 COleServerDoc::GetItemPosition，COleServerDoc::GetItemClipRect，COleServerDoc::OnSetItemRects

COleServerDoc::IsEmbedded

```
BOOL IsEmbedded() const;
```

返回值

如果 COleServerDoc 对象是一个代表容器中的嵌入对象的文档，则返回非零值；否则返回 0。

说明

调用 IsEmbedded **成员函数来确定文档表示的是否是一个嵌入在容器应用程序中的对象。从文件装载的文档不是嵌入式的，尽管它可以由容器应用程序链接来操作。嵌入在容器文档中的文件被认为是嵌入式的。**

COleServerDoc::IsInPlaceActive

BOOL IsInPlaceActive() const;

返回值

如果 COleServerDoc **对象是现场活动的，则返回非零值；否则返回** 0。

说明

调用 IsInPlaceActive **成员函数来确定此项当前是否处于现场活动状态。**

请参阅 COleServerDoc::ActiveInPlace，COleServerDoc::OnReactivateAndUndo，COleServerDoc::ActivateInPlace

COleServerDoc::NotifyChanged

```
void NotifyChanged();
```

说明

此函数将文档改变了的信息通知给与此文档链接的所有链接项，一般的，应用程序在用户改变某些全局属性（比如服务器文档的大小）之后，调用 NotifyChanged 函数。如果用一个自动链接将一个 OLE 项链接到此文档上，则将更新此项以反映这些改变。在用微软基础类库编写的容器应用程序中，将调用 COleClientItem 的 OnChange 函数。

注意 这个函数是为与 0LE 兼容而提供的。新应用程序应该使用

UpdateAllItems 函数。

请参阅 COleServerDoc::NotifyClosed，COleServerDoc::NotifySaved，
COleServerDoc::OnChange

COleServerDoc::NotifyClosed

```
void NotifyClosed();
```

说明

此函数通知容器应用程序文档已经关闭。当用户从 File **菜单中选择** Close **命令时，由** COleServerDoc **实现的** OnCloseDocument **成员函数调用** NotifyClosed **函数。在用微软基础类库编写的容器应用程序中，将调用** COleClientItem **的** OnChange **成员函数。**

请参阅 COleServerDoc::NotifyChanged，COleServerDoc::NotifySaved，

COleServerDoc::OnChange，COleServerDoc::OnCloseDocument

COleServerDoc::NotifyRename

void NotifyRename(LPCTSTR *lpszNewName*);

参数

lpszNewName

一个字符串指针，此字符串指定服务器文档的新名字；它通常是一个全路径。

说明

在用户改换服务器文档的名字之后，调用此函数。当用户从 File **菜单中选择** Save As **命令时，由** COleServerDoc **实现的** OnSaveDocument **成员函数调用** NotifyRename **函数。**NotifyRename **函数通知** OLE **系统** DLLs**，后者再接着通知**

容器应用程序。在用微软基础类库容器应用程序中，将调用 COleClientItem 的 OnChange 成员函数。

请参阅 COleServerDoc::NotifySaved，CDocument::OnSaveDocument

COleServerDoc::NotifySaved

void NotifySaved();

说明

在用户保存了服务器文档之后，调用该函数。当用户从 File 菜单中选择 Save 命令时，由 COleServerDoc 实现的 OnSaveDocument 成员函数为应用程序调用 NotifySaved 函数。NotifySaved 函数通知 OLE 系统 DLL，后者再接着通知容器应用程序。在用微软基础类库编写的容器应用程序中，将调用 COleClientItem 的 OnChange 成员函数。

请参阅 COleServerDoc::NotifyChanged，COleServerDoc::NotifyClosed，COleClientItem::OnChange，CDocument::OnSaveDocument

COleServerDoc::OnClose

virtual void OnClose(OLECLOSE *dwCloseOption*);

参数

dwCloseOption

一个 OLECLOSE 枚举值。这个参数的值可以是下列值之一：

- OLECLOSE_SAVEIFDIRTY 如果文件已被修改则将该文件保存。
- OLECLOSE_NOSAVE 将文件关闭而不保存。
- OLECLOSE_PROMPTSAVE 如果文件已被修改，则向用户提示保存文件。

说明

当容器应用程序请求关闭服务器文档时，框架调用此函数。函数的缺省实现调用 CDocument::OnCloseDocument 函数。要获取更多有关 OLECLOSE 和其它值的信息，请参见 OLE 文档的 OLECLOSE。

请参阅 COleException，CDocument::OnCloseDocument

COleServerDoc::OnDeactivate

virtual void OnDeactivate();

说明

当用户使一个当前现场活动的嵌入或链接项变为不活动的时候，框架调用此函数。该函数将容器应用程序的用户界面恢复到其原始状态，并销毁视窗为现场

激活所创建的任何菜单和其它控件。

此时，取消状态信息应当被无条件释放。

要获取更多的信息，请参见“Visual C++程序员指南”中的文章“激活”。

请参阅 COleServerDoc::ActivateInPlace，COleServerDoc::OnDeactivateUI，
COleServerDoc::DestroyInPlaceFrame

COleServerDoc::OnDeactivateUI

virtual void OnDeactivateUI(BOOL *bUndoable*);

参数

bUndoable

指示编辑改变是否可以被取消。

说明

当用户使一个现场活动的项变为不活动时，调用此函数。函数将容器应用程序的用户界面恢复到其原始状态，隐藏由现场激活所创建的任何菜单和其它控件。

框架总是设置 bUndoable **为** FALSE**。如果服务器应用程序支持取消操作，但是不存在一个可被取消的操作，则** bUndoable **设置为** TRUE **来调用由基类实现的** OnDeactivateUI **函数。**

请参阅 COleServerDoc::OnDeactivate

COleServerDoc::OnDocWindowActivate

virtual void OnDocWindowActivate(BOOL *bActivate*);

参数

bActivate

指示是将文档窗口激活还是使它成为不活动的。

说明

框架调用这个函数来激活用于现场编辑的一个文档窗口，或使此窗口变成不活动的。该函数的缺省实现是删除或增加相应的框架顶层用户界面元素。如果当包含项的文档被激活或成为不活动时，应用程序希望执行附加的动作，则可以重载这个函数。

请参阅 COleServerDoc::ActivateInPlace，COleServerDoc::OnReactivateAndUndo，COleServerDoc::OnShowControlBars，

COleServerDoc::OnDeactivateUI，

COleServerDoc::OnFrameWindowActivate，COleIPFrameWnd

COleServerDoc::OnExecOleCmd

HRESULT OnExecOleCmd(const GUID* *pGroup*, DWORD *nCmdID*,
DWORD *nCmdExecOut*, VARIANTARG* *pvaIn*, VARIANTARG* *pvaOut*);

返回值

如果成功则返回 S_OK**；否则，返回下列错误代码之一：**

Value	Description
E_UNEXPECTED	**发生了意外的错误**
E_FAIL	**发生了错误**
E_NOTIMPL	**表明** MFC **自身要尝试翻译和分派命令**

Value	Description
OLECMDERR_E_UNKNOWNGROUP	pGroup **不是** NULL**，但是不表示一个被认可的命令组**

续表

OLECMDERR_E_NOTSUPPORTED	nCmdID 不被认可为是 pGroup 群中的有效命令
OLECMDERR_DISABLED	由 nCmdID 标识的命令被变为无效，不能被执行
OLECMDERR_NOHELP	调用者请求有关由 nCmdID 标识的命令的帮助，但是没有可用的帮助
OLECMDERR_CANCELED	用户取消了执行

参数

pGroup

一个指向 GUID 的指针，这个 GUID 标识了一个命令集。它可以是 NULL，这表示缺省的命令组。

nCmdID

要执行的命令。必须是在由 pGroup 标识的组中。

nCmdExecOut

对象执行命令的方式，可以是下列 OLECMDEXECOPT 枚举值中的一个或某几个：

- OLECMDEXECOPT_DODEFAULT
- OLECMDEXECOPT_PROMPTUSER
- OLECMDEXECOPT_DONTPROMPTUSER
- OLECMDEXECOPT_SHOWHELP

pvaIn

指向一个 VARIANTARG 的指针，这个 VARIANTARG 包含了命令的输入参数。它可以是 NULL。

pvaOut

指向一个 VARIANTARG 的指针，这个 VARIANTARG 用来接收来自命令的输出返回值。它可以是 NULL。

说明

框架调用这个函数来执行一个指定的命令，或为命令显示帮助。

COleCmdUI 可以用来使能，更新和设置 DocObject 用户界面命令的其它属性。在命令被初始化之后，你可以用 OnExecOleCmd 来执行它们。在框架尝试翻译和分派一个 OLE 文档命令之前，它调用这个函数。在处理标准的 OLE 文档命令时，你不必重载这个函数。但是如果你想处理你自己的定制命令，或处理需要接收参数或返回结果的命令时，你必须要重载这个函数。

大多数命令不需要接收参数或返回结果。对于大多数的命令来说，调用者可以将 NULL 赋给 pvaIn 和 pvaOut。对于那些需要接收输入参数的命令，调用者可以声明和初始化一个 VARIANTARG 变量，并将一个指针传递给 pvaIn 中的变量。对需要一个单一值的命令，参数被直接保存在 VARIANTARG 中，并传递

给函数。而多个参数则必须要在 VARIANTARG 中使用某个允许的类型（如 IDispatch 和 SAFEARRAY）来打包。

类似的，如果一个命令要返回参数，则调用者需要定义一个 VARIANTARG，将它初始化为 VT_EMPTY，并将它的地址传递给 pvaOut。如果命令返回一个单一的值，则对象可以将这个值直接保存在 pvaOut 中。如果有多个输出值则必须用某种适用于 VARIANTARG 的方法来打包。

这个函数的基类实现将使 OLE COMMAND_MAP 结构与命令目标相联系，并尝试为命令分派一个合适的句柄。函数的基类实现只处理不接收参数和不返回结果的命令。如果你需要处理接收参数或返回结果的命令，你就必须重载这个函数，并且自己来处理 pvaIn 和 pvaOut。

请参阅 COleCmdUI

COleServerDoc::OnFrameWindowActivate

virtual void OnFrameWindowActivate(BOOL *bActivate*);

参数

bActivate

指示框架窗口是要被激活还是变为不活动的。

说明

当容器应用程序的框架窗口被激活或变为不活动时，框架调用这个函数。此函数的缺省实现是取消框架窗口可能处于的任何帮助方式。如果当框架窗口被激活或成为不活动时，应用程序想要执行特定的处理，则可以重载 OnFrameWindowActive **函数。**

如果要获取更多的信息，请参见“ Visual C++**程序员指南”中的文章“激活”。**

请参阅 COleServerDoc::OnDocWindowActivate

COleServerDoc::OnGetEmbeddedItem

virtual COleServerItem* OnGetEmbeddedItem() = 0;

返回值

如果成功则返回一个指向代表着整个文档的项的指针；否则返回 NULL。

说明

当容器应用程序调用服务器应用程序来创建或编辑一个嵌入项时，框架调用该函数。该函数没有缺省实现。应用程序必须重载这个函数以返回一个表示整个文档的项。这个返回值应该是一个 COleServerItem **派生类对象。**

请参阅 COleLinkingDoc::OnGetLinkedItem，COleServerItem

COleServerDoc::OnReactivateAndUndo

virtual BOOL OnReactivateAndUndo();

返回值

如果成功则返回一个非零值；否则返回 0。

说明

当用户选择取消对一个已被现场激活，后来又变为不活动的项所作的改变时，框架调用这个函数。该函数的缺省实现除了返回 FALSE 来表明失败之外，它不做任何事情。

如果你的应用程序要支持取消，请重载这个函数。通常应用程序会执行取消操作，然后调用 ActivateInPlace 函数来激活项。如果容器应用程序是用微软基础类库编写的，则调用 COleClientItem::ReactivateAndUndo 会导致调用

OnReactivateAndUndo 函数。

请参阅 COleServerDoc::ActivateInPlace，COleServerDoc::IsInPlaceActive，
COleClientItem::ReactivateAndUndo

COleServerDoc::OnResizeBorder

virtual void OnResizeBorder(LPCRECT *lpRectBorder*,
LPOLEINPLACEUIWINDOW *lpUIWindow*, BOOL *bFrame*);

参数

lpRectBorder

指向一个 RECT 结果或 CRect 对象的指针，此结构或对象指定边框的坐标。

lpUIWindow

指向一个 IOleInPlaceUIWindow 类对象，此对象拥有当前的现场编辑会话。

bFrame

如果 lpUIWindow 指向容器应用程序的顶层框架窗口，则设该参数为

TRUE；如果 lpUIWindow 指向容器应用程序的文档顶层框架窗口，则设该参数为 FALSE。

说明

当容器应用程序的框架窗口改变大小时，框架调用这个函数。该函数根据新的窗口大小来改变和调整工具条和其它用户界面元素的大小。

此函数是一个高级的可重载函数。

请参阅 COleServerDoc::OnShowControlBars

COleServerDoc::OnSetHostNames

virtual void OnSetHostNames(LPCTSTR *lpszHost*, LPCTSTR *lpszHostObj*);

参数

lpszHost

指向一个字符串的指针，该字符串指定了容器应用程序的名字。

lpszHostObj

指向一个字符串的指针，该字符串指定了文档的容器名称。

说明

当容器应用程序设置或改变此文档的主机名时，框架调用此函数。此函数的缺省实现改变所有引用此文档的视的文档标题。

如果你的应用程序想通过一种不同的机制来设置标题，则需要重载这个函数。

请参阅 COleClientItem::SetHostNames

COleServerDoc::OnSetItemRects

virtual void OnSetItemRects(LPCRECT *lpPosRect*, LPCRECT *lpClipRect*);

参数

lpPosRect

指向一个 RECT **结构或** CRect **对象的指针，此结构或对象指定框架窗口相对于容器应用程序的客户区的位置。**

lpClipRect

指向一个 RECT **结构或** CRect **对象的指针，此结构或对象指定在适当位置的框架窗口的剪贴矩形（相对于容器应用程序的客户区）。**

说明

框架调用这个函数来在容器应用程序的框架窗口内定位现场编辑创建窗口。如

果有必要更新视的缩放因子，则可以重载这个函数。

尽管 OnSsetItemRects 可以在任何时候由容器应用程序调用来请求改变现场可编辑项的位置，但它通常还是用于响应 RequestPositionChange 调用。

请参阅 COleServerDoc::RequestPositionChange，COleIPFrameWnd::RepositionFrame，COleClientItem::SetItemRects，COleServerDoc::GetZoomFactor

COleServerDoc::OnShowControlBars

virtual void OnShowControlBars(CFrameWnd *_pFrameWnd_, BOOL _bShow_);

参数

pFrameWnd

指向框架窗口的指针，此框架窗口的控制条将被隐藏或显示。

bShow

确定控制条是显示还是隐藏。

说明

框架调用这个函数来显示或隐藏与 pFrameWnd 所标识的框架窗口相关联的服务器应用程序的控制条。该函数的缺省实现是枚举该框架窗口所拥有的所有控制条，并隐藏或显示它们。

请参阅 COleServerDoc::ActivateInPlace，COleServerDoc::OnReactivateAndUndo，COleServerDoc::OnFrameWindowActivate，COleServerDoc::IsInPlaceActive

COleServerDoc::OnShowDocument

virtual void OnShowDocument(BOOL *bShow*);

参数

bShow

指示是显示还是隐藏文档的用户界面。

说明

当服务器文档必须被隐藏或显示时，框架调用 OnShowDocument **函数。如果** bShow **为** TRUE**，则该函数的缺省实现是在响应时激活服务器拥有的程序，并使容器应用程序滚动它的窗口以使此项可见。如果** bShow **为** FALSE**，则函数的缺省实现是通过一个** OnDeactivate **调用使此项不活动，然后销毁或隐藏为该文档创建的所有框架窗口（除了第一个）。如果没有可见文档，则该函数的缺**

省实现隐藏服务器应用程序。

请参阅 COleServerDoc::ActivateInPlace，COleServerItem::OnDoVerb，
COleServerDoc::IsInPlaceActive，COleServerDoc::OnDeactivateUI

COleServerDoc::OnUpdateDocument

```
virtual BOOL OnUpdateDocument();
```

返回值

如果文档被成功地更新则返回非零值；否则返回 0。

说明

当保存一个作为复合文档中的嵌入项的文档时，框架调用该函数。该函数的缺省实现调用 COleServerDoc::NotifySaved 和 COleServerDoc::SaveEmbedding 成

员函数，然后将文档标记为干净的（未修改的）。如果应用程序想要在更新一个嵌入项时执行特定的处理，则必须重载这个函数。

请参阅 COleServerDoc::NotifySaved，COleServerDoc::SaveEmbedding，CDocument::OnSaveDocument

COleServerDoc::RequestPositionChange

void RequestPositionChange(LPCRECT *lpPosRect*);

参数

lpPosRect

指向一个 RECT 结构或 CRect 对象的指针，此结构或对象中包含此项的新位置。

说明

此成员函数使容器应用程序改变此项的位置。当现场活动的项中的数据改变时，通常调用 RequestPositionChange 函数（与 UpdateAllItems 一起调用）。在此调用之后，容器应用程序可能调用也可能不调用 OnSetItemRects 函数来执行这个改变。最终位置可能与请求的位置并不相同。

请参阅 COleServerDoc::ScrollContainerBy

COleServerDoc::SaveEmbedding

void SaveEmbedding();

说明

这个函数用来告诉容器应用程序保存嵌入对象。OnUpdateDocument 会自动调

用此函数。值得注意的是这个函数会导致项被在磁盘上更新，因此，它通常只是作为一次特定用户动作的结果被调用。

请参阅 COleServerDoc::NotifyClosed

COleServerDoc::ScrollContainerBy

BOOL ScrollContainerBy (CSize *sizeScroll*);

返回值

如果成功则返回非零值；否则返回 0。

参数

sizeScroll

指示容器文档要滚动多远。

说明

ScrollContainerBy **成员函数将容器文档滚动** sizeScroll **中所指定的像素数量。正值表示向下和向右滚动；负值表示向上和向左滚动。**

请参阅 COleClientItem::OnScrollBy

COleServerDoc::UpdateAllItems

void UpdateAllItems(COleServerItem* *pSender*, LPARAM *lHint* = 0L**,**
CObject* *pHint* = NULL, DVASPECT *nDrawAspect* = DVASPECT_CONTENT);

参数

pSender

指向修改文档的项的指针；如果所有项都要被更新，则设置该参数为

NULL。

lHint

包含了有关修改的信息。

pHint

指向一个包含有关修改的信息的对象的指针。

nDrawAspect

确定如果绘制此项。该参数是一个 DVASPECT 枚举值。它可以是下列值之一：

- DVASPECT_CONTENT 用一种可将项显示为其容器文档中一个嵌入对象的方式来显示此项。
- DVASPECT_THUMBNAIL 以“拇指甲”的方式来显示项，以使它能在浏览工具中显示。
- DVASPECT_ICON 将项显示为一个图标。

- DVASPECT_DOCPRINT 将项显示为就好象是在利用 File 菜单中的 Print 命令打印它。

说明

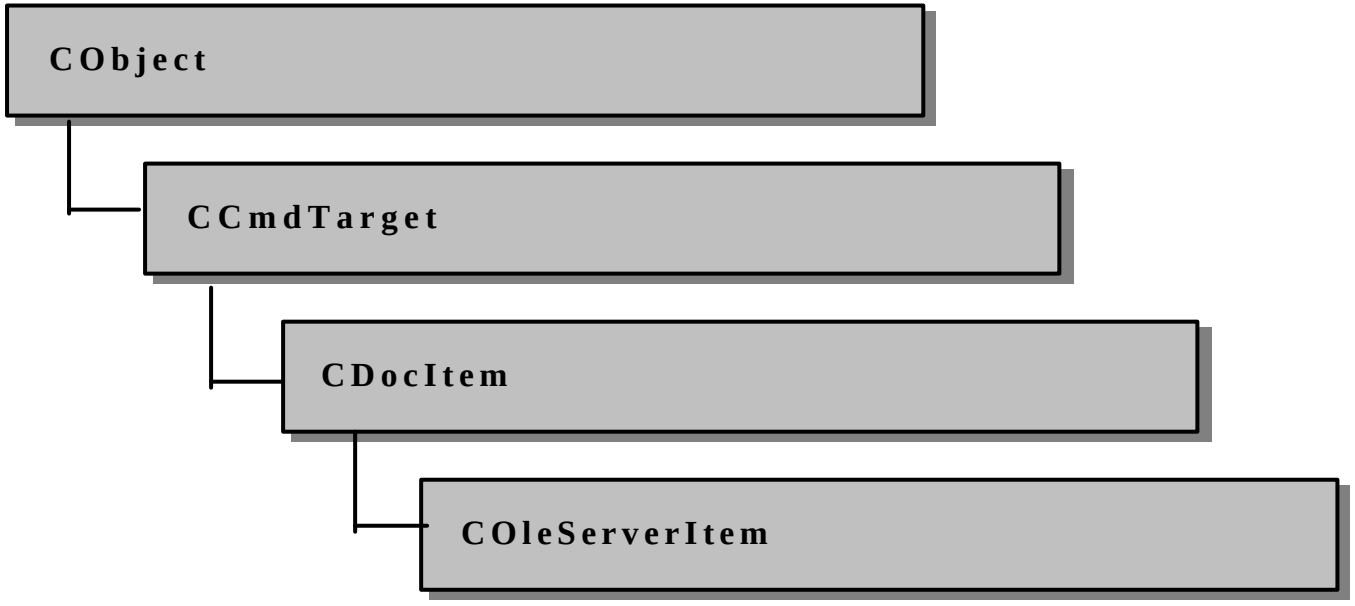
此函数将文档改变了的信息通知给连接在此文档上的所有链接项。应用程序一般在用户改变服务器文档之后调用这个函数。如果一个 OLE 项以一个自动链接链接到此文档上，则该项被更新以反映这些变化。在用微软基础类库编写的容器应用程序中，将调用 COleClientItem 的 OnChange 成员函数。

除了发送项外，UpdateAllItems 函数为文档的每个项调用 OnUpdate 成员函数，传递 pHint，lHint 和 nDrawAspect 参数。利用这些参数将有关对文档所做修改的信息传递给各个项。应用程序可以利用 lHint 对信息进行编码，或者定义一个 CObject 派生类来存储有关修改的信息并用 pHint 传递一个该类的对象。可

在应用程序的 COleServerItem 派生类中重载 OnUpdate 成员函数，根据每个项的显示是否改变了来优化对各个项的更新。

请参阅 COleServerDoc::NotifyChanged，COleServerItem::OnUpdate，COleServerDoc::NotifySaved，COleClientItem::OnChange

COleServerItem



COleServerItem 类提供对 OLE 项的服务器接口。一个链接项可以表示某些或全部服务器文档。一个嵌入项则总是表示整个服务器文档。

COleServerItem 类定义了几个可重载的成员函数，这些函数由 OLE 系统动态链接库（DLL）调用，通常用于响应来自容器应用程序的请求。这些成员函数允许容器应用程序以各种方式间接地对项进行操作，例如显示它，执行它的动词，或以各种格式获取它的数据。

要使用 COleServerItem，可从 COleServerItem 中派生一个新类，并实现 OnDraw 和 Serialize 成员函数。OnDraw 函数提供项的元文件表示，使它可在容器应用程序打开一个命令文档时显示。CObject 的 Serialize 函数提供项的本机表示，使得一个嵌入项可在服务器应用程序和容器应用程序之间直接传输。OnGetExtent 函数提供项的固有对象给容器应用程序，使得容器应用程序可以设置项的大小。

要获取更多有关服务器及其相关话题的信息，请参见“Visual C++程序员指南”中的文章“服务器：实现一个服务器”和文章“容器：高级特征”中的“创建

一个容器/服务器应用程序”。

#include <afxole.h>

请参阅 COleClientItem，COleServer，COleTemplateServer

COleServerItem 类成员

Status	
GetDocument	返回包含此项的服务器文档
GetItemName	返回此项的名称。只用于链接项
Status	
SetItemName	设置此项的名称。只用于链接项
IsConnected	指示此项当前是否连接到一个活动的容器应用程序上
IsLinkedItem	指示此项是否表示一个链接式 OLE 项

Operations	
CopyToClipboard	将此项拷贝到剪贴板上
NotifyChanged	用自动链接更新来更新所有的容器
DoDragDrop	执行一次拖放操作
GetClipboardData	获取用于数据传输（拖放或剪贴）的数据源
GetEmbedSourceData	获取一个 OLE 项的 CF_EMBEDSOURCE 数据
AddOtherClipboardData	将显示和转换格式放进一个 COleDataSource 对象
GetLinkSourceData	获取一个 OLE 项的 CF_LINKSOURCE 数据
GetObjectDescriptorData	获取一个 OLE 项的 CF_OBJECTDESCRIPTOR 数据

Construction	
ColeServerItem	构造一个 COleServerItem 对象
GetDataSource	获取用于存储转换格式的对象

Overridables	
OnDraw	当容器应用程序请求绘制此项时，调用该函数；需要实现
OnDrawEx	用于绘制专用化的项
OnUpdate	当此项所属的文档的某部分改变时，调用该函数
OnInitFromData	由框架调用该函数，该函数用指定的数据传输对象的内容来初始化一个 OLE 项
OnGetExtent	由框架调用此函数来获取此 OLE 项的大小
OnSetExtent	由框架调用此函数来设置此 OLE 项的大小
OnGetClipboardData	由框架调用此函数来获取将要拷贝到剪贴板中的数据
OnSetColorScheme	用于设置项的颜色表
OnSetData	用于设置项的数据
OnDoVerb	用于执行一个动词
OnQueryUpdateItems	用于确定是否有链接项需要更新
OnRenderData	获取数据为延迟提供的一部分
OnRenderFileData	将数据检取到一个 CFile 中，作为延迟提供的一部分
OnRenderGlobalData	将数据检取到一个 HGLOBAL 中，作为延迟提供的一部分

续表

OnUpdateItems	用于更新服务器文档中所有项的显示高速缓存
OnOpen	由框架调用此函数来将此 OLE 项显示在它自己的顶层窗口中
OnShow	当容器应用程序请求显示此项时，调用该函数
OnHide	由框架调用该函数来隐藏此 OLE 项

Data.Members

m_sizeExtent	通知服务器此 OLE 项有多少是可见的

成员函数

COleServerItem::AddOtherClipboardData

void AddOtherClipboardData(COleDataSource* *pDataSource*);

参数

pDataSource

指向一个 COleDataSource 对象的指针，数据将放在此对象中。

说明

此函数用来将 OLE 项的显示和转换格式放进指定的 COleDataSource 对象中。应用程序必须已经实现了 OnDraw 成员函数，这样才能提供此项的显示格式（一个图元文件图像）。要支持其它的转换格式，可以向 GetDataSource 返回的 COleDataSource 对象注册这些格式，并重载 OnRenderData 成员函数以应用程序希望支持的格式来提供数据。

请参阅 COleDataSource，COleServerItem::GetDataSource，COleServerItem::GetEmbedSourceData，COleServerItem::OnDraw

COleServerItem::COleServerItem

COleServerItem(COleServerDoc* *pServerDoc*, BOOL *bAutoDelete*);

参数

pServerDoc

指向包含此项的文档的指针。

bAutoDelete

一个标志，指示当连接到此对象的一个链接被释放时，此对象是否可被删除。如果此 COleServerItem **对象是必须删除的文档数据的组成部分，则将该参数设置为** FALSE**。如果此对象是用于标识文档中可被删除的数据范围的辅助结构，则将该参数设置为** TRUE**。**

说明

此函数构造一个 COleServerItem **项，并将其添加到服务器文档的文档项集合中。**

请参阅 COleDocument::AddItem

COleServerItem::CopyToClipboard

void CopyToClipboard(BOOL *bIncludeLink* = FALSE);

参数

bIncludeLink

如果链接数据将被拷贝到剪贴板中，则将此参数设置为 TRUE。**如果服务器应用程序不支持链接，则将此参数设置为** FALSE。

说明

该函数将此 OLE 项拷贝到剪贴板中。它利用 OnGetClipboardData 成员函数来创建一个 COleDataSource 对象，此对象包含了以所支持的格式表示的 OLE 项数据。然后函数利用 COleDataSource::SetClipboard 函数将此 COleDataSource 对象放到剪贴板上。这个 COleDataSource 对象中包括了此项的本机数据和它的 CF_METAFILEPICT 格式的表示，以及应用程序选择支持的任何转换格式中的数据。应用程序必须已经实现了 Serialize 和 OnDraw 函数，以使此 CopyToClipboard 成员函数可以工作。

请参阅 COleDataSource::SetClipboard，COleDataSource，
COleServerItem::AddOtherClipboardData，
COleServerItem::GetClipboardData，
COleServerItem::OnDraw，CObject::Serialize

COleServerItem::DoDragDrop

DROPEFFECT DoDragDrop(LPCRECT *lpItemRect*, CPoint *ptOffset*,
BOOL *bIncludeLink* = FALSE,DWORD *dwEffects* = DROPEFFECT_COPY|
DROPEFFECT_MOVE,LPCRECT *lpRectStartDrag* = NULL);

返回值

是一个 DROPEFFECT **枚举值。如果它是** DROPEFFECT_MOVE**，则原始数据将被删除。**

参数

lpItemRect

指定项在屏幕上的矩形，相对于客户区，用像素表示。

ptOffset

拖动时鼠标位置距离 lpItemRect **的偏移。**

bIncludeLink

如果链接数据将被拷贝到剪贴板上，则将此参数设置为 TRUE。如果应用程序不支持链接，则将此参数设置为 FALSE。

dwEffects

确定数据源在拖动操作中允许的效果（Copy，Move 和 Link 的组合）。

lpRectStartDrag

指向调用拖动实际开始处的矩形的指针。更多的信息，请参见下面的说明部分。

说明

此成员函数用来执行一次拖放操作。拖放操作并不是立即开始。直到鼠标光标离开由 lpRectStartDrag 指定的矩形后，或者直到已经过去了指定的毫秒数后，拖放才开始。如果 lpRectStartDrag 为 NULL，则使用缺省拖动矩形，这样拖动

在鼠标光标至少移动一个像素时才开始。

延迟时间是由一个注册表键设置来确定的。你可以通过调用 CWinApp::WriteProfileString 或者 CWinApp::WriteProfileInt 来改变延迟时间。如果你没有指定延迟时间，则使用缺省的 200 毫秒延迟设置。拖动延迟时间是按下面的方式保存的：

- Windows NT 拖动延迟时间被保存在 HKEY_LOCAL_MACHINE\SOFTWARE\Mic- rosoft \Windows\NT\CurrentVersion\IniFileMapping\win.ini\Windows\DragDelay。
- Windows 3.x 拖动延迟时间被保存在[Windows]部分下的 WIN.INI 文件中。
- Windows 95 拖动延迟时间被保存在 WIN.INI 的缓冲版本中。

如果要获取更多有关拖动延迟时间是如何保存在注册表或.INI 文件中的信息，

请参见 Platform SDK **中的**::WriteProfileString。

请参阅 COleDataSource::DoDragDrop，COleServerItem::CopyToClipboard

COleServerItem::GetClipboardData

void GetClipboardData(COleDataSource* *pDataSource*, BOOL *bIncludeLink* = FALSE,
    LPPOINT *lpOffset* = NULL, LPSIZE *lpSize* = NULL);

参数

pDataSource

指向一个 COleDataSource **对象的指针，此对象将接收** OLE **项的所有用所支持的格式表示的数据。**

bIncludeLink

如果链接数据可以被拷贝到剪贴板中，则将此参数设置为 TRUE。**如果**

你的服务器应用程序不支持链接，则将此参数设置为 FALSE。

lpOffset

鼠标光标距离此对象原点的以像素表示的偏移。

lpSize

此对象以像素表示的大小。

说明

此函数用将拷贝到剪贴板中的所有数据（如果调用 CopyToClipboard 函数）填充到指定的 COleDataSource 对象中（如果调用 DoDragDrop，同样的数据也将被传输）。GetClipboardData 成员函数调用 GetEmbedSourceData 成员函数来获取此 OLE 项的本机数据，并调用 AddOtherClipboardData 成员函数来获取显示格式和任何所支持的转换格式。如果 bIncludeLink 为 TRUE，则 GetClipboardData 函数也调用 GetLinkSourceData 函数来获取此项的链接数据。

如果你希望在 CopyToClipboard 提供这些格式之前或之后将它们放进一个 COleDataSource 对象中，则可重载这个函数。

请参阅 COleDataSource，COleServerItem::AddOtherClipboardData，COleServerItem::CopyToClipboard，COleServerItem::DoDragDrop，COleServerItem::GetEmbedSourceData，COleServerItem::GetLinkSourceData

COleServerItem::GetDataSource

```
COleDataSource* GetDataSource();
```

返回值

一个指向 COleDataSource 对象的指针，该对象用来保存转换格式。

说明

此函数用来获取用于存储服务器应用程序所支持的转换格式的 COleDataSource 对象。如果你希望你的服务器应用程序可以在数据传输操作期间提供多种格式的数据，则可以向此函数返回的 COleDataSource 对象注册这些格式。例如，如果应用程序想要提供 OLE 项的 CF_TEXT 表示以用于剪贴板或拖放操作，则要向 GetDataSource 返回的 COleDataSource 对象注册这种格式，然后重载 OnRenderXxxData 成员函数来提供数据。

请参阅 COleDataSource，COleDataSource::DelayRenderData，

COleServerItem::CopyToClipboard，COleServerItem::DoDragDrop，

COleServerItem::OnRenderData，COleServerItem::OnRenderFileData，

ColeServerItem::OnRenderGlobalData

COleServerItem::GetDocument

COleServerDoc* GetDocument() const;

返回值

一个指向包含此项的文档的指针；如果此项不是文档的一部分，则返回值为 NULL。

说明

此函数获取指向包含此项的文档的指针。它支持对作为参数传递给 COleServerItem **构造函数的服务器文档的访问。**

请参阅 COleServerItem:: COleServerItem，COleServerDoc

COleServerItem::GetEmbedSourceData

void GetEmbedSourceData(LPSTGMEDIUM *lpStgMedium*);

参数

lpStgMedium

指向一个 STGMEDIUM **结构的指针，此结构将接收此** OLE **项的** CF_EMBEDSOURCE **数据。**

说明

此函数用来获取一个 OLE **项的** CF_EMBEDSOURCE **数据。这种格式中包括此项的本机数据。应用程序必须已经实现了** Serialize **成员函数，以使** GetEmbedSourceData **函数能够正确运行。**

该函数的结果可以用 COleDataSource::CacheData **函数增加到一个数据源中。**

GetEmbedSourceData 函数由 OnGetClipboard 函数自动调用。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 STGMEDIUM。

请参阅 COleServerItem::GetLinkSourceData，COleServerItem::GetObjectDescriptorData，COleDataSource::CacheData，CObject::Serialize

COleServerItem::GetItemName

```
const CString& GetItemName() const;
```

返回值

返回值是项的名字。

说明

此函数用来获取项的名字。应用程序一般只对链接项调用这个函数。

请参阅 COleServerItem::SetItemName，COleLinkingDoc::OnGetLinkedItem

COleServerItem::GetLinkSourceData

BOOL GetLinkSourcedData(LPSTGMEDIUM *lpStgMedium*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpStgMedium

指向一个 STHGMEDIUM **结构的指针，此结构将接收此** OLE **项的**

CF_LINKSOURCE 数据。

说明

此函数用来获取一个 OLE 项的 CF_LINKSOURCE 数据。这种格式中包括用于描述此 OLE 项类型的 CLSID，以及定位包含此 OLE 项的文档所需的信息。

此函数的结果可用 COleDataSource::CacheData 函数增加到一个数据源中。

GetLinkSourceData 函数由 OnGetClipboard 函数自动调用。

要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 STGMEDIUM。

请参阅 COleServerItem::GetEmbedSourceData，
COleServerItem::GetObjectDescriptorData

COleServerItem::GetObjectDescriptorData

void GetObjectDescriptorData(LPPOINT* *lpOffset*, LPSIZE* *lpSize*,

LPSTGMEDIUM *lpStgMedium*);

参数

lpOffset

鼠标单击距离此 OLE **项左上角的偏移。该参数可以为** NULL**。**

lpSize

此 OLE **项的大小。该参数可以为** NULL**。**

lpStgMedium

指向一个 STGNEDIUM **结构的指针，此结构将接收此** OLE **项的** CF_OBJECTDESCRIP- TOR **数据。**

说明

此函数用来获取一个 OLE **项的** CF_OBJECTDESCRIPTOR **数据。这个信息被拷贝到** lpStgMedium **所指向的** STGMEDIUM **结构中。这种格式包括** Paste Special

对话框所需要的信息。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 STGMEDIUM。

请参阅 COleServerItem::AddOtherClipboardData，

COleServerItem::GetEmbedSourceData，

COleServerItem::GetLinkSourceData，

COlePasteSpecialDialog

COleServerItem::IsConnected

```
BOOL IsConnected() const;
```

返回值

如果此项被连接则返回一个非零值；否则返回 0。

说明

此函数用来判断 OLE **项是否被连接。如果有一个或多个容器文档引用一个** OLE **项，则认为此** OLE **项是被连接的。如果一个项的引用计数大于零或者若该项是一个嵌入项，则认为此项是被连接的。**

请参阅 COleServerItem::IsLinkedItem，COleLinkingDoc::OnGetLinkedItem

COleServerItem::IsLinkedItem

BOOL IsLinkedItem() const;

返回值

如果此项是一个链接项，则返回一个非零值；否则返回 0。

说明

此函数用来判断此 OLE **项是否是一个链接项。如果一个项是合法的，并且未返回在文档的嵌入项表中，则此项是链接项。链接项可以链接也可以不链接到容器文档上。**

通常对链接和嵌入项使用同一个类。IsLinkedItem **函数支持应用程序将链接项的行为有别于嵌入项，尽管它们的代码常常是通用的。**

请参阅 COleServerItem::IsConnected，COleLinkingDoc::OnGetLinkedItem

COleServerItem::NotifyChanged

void NotifyChanged(DVASPECT *nDrawAspect* = DVASPECT_CONTENT);

参数

nDrawAspect

一个 DVASPECT 枚举值，用于指示此 OLE 项的哪一特征改变了。这个参数可以取下列任何值：

- DVASPECT_CONTENT 用这种方式表示的项可以作为一个嵌入对象显示在它的容器中。
- DVASPECT_THUMBNAIL 用“拇指甲”方式表示项，以使它可以显示在一个浏览工具中。
- DVASPECT_ICON 用图标来表示项。
- DVASPECT_DOCPRINT 项被表示为就象它被用 File 菜单中的 Print 命令打印一样。

说明

当链接项改变后，调用此函数。如果一个容器应用程序项用一个自动链链接到文档上，则此项被更新以反映这些变化。在用微软基础类库编写的容器应用程序中，将调用 COleClientItem **的** OnChange **成员函数来作为响应。**

请参阅 COleClientItem::OnChange，COleServerItem::OnUpdate，COleServerDoc::NotifyChanged

COleServerItem::OnDoVerb

virtual void OnDoVerb(LONG *iVerb*);

参数

iVerb

指定要执行的动词。它可以是下列值中的任何一个。

值	含义	符号
0	主动词	OLEIVERB_PRIMARY
1	此动词	（无）
-1	显示用于编辑的项	OLEIVERB_SHOW
-2	在独立窗口中编辑项	OLEIVERB_OPEN
-3	隐藏项	OLEIVERB_HIDE

值-1 通常是另一个动词的别名。如果打开编辑不支持，则-2 值与-1 值作用相同。有关其它的取值，参看“OLE 2 程序员参考，卷 1”中的 IOleObject::DoVerb。

说明

框架调用这个函数来执行指定的动词。如果容器应用程序是用微软基础类库编写的，则当对应的 COleClientItem 对象的 COleClientItem::Activate 成员函数被调用时，调用 OnDoVerb 函数。如果指定主动词或 OLEIVERB_SHOW，则 OnDoVerb 的缺省实现是调用 On_Show 成员函数；如果指定次动词或

OLEIVERB_OPEN，则缺省实现是调用 OnOpen 成员函数；如果指定 OLEIVERB_HIDE，则缺省实现是调用 OnHide 成员函数。如果 iVerb 不是上面所列中的任何一个，则缺省实现是调用 OnShow 成员函数。

如果指定的主动词不显示此项，则可重载这个函数。例如，如果此项是一个声音记录，并且其主动词是 Play，则应用程序不必显示服务器应用程序来播放此项。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IOleObject::DoVerb。

请参阅 COleClientItem::Activate，COleServerItem::OnShow，COleServerItem::OnOpen，COleServerItem::OnHide

COleServerItem::OnDraw

virtual BOOL OnDraw(CDC* *pDC*, CSize& *rSize*) = 0;

返回值

如果此项绘制成功，则返回一个非零值；否则返回 0。

参数

pDC

指向一个 CDC **对象的指针，要在此对象上绘制此项。显示描述表自动连接到属性显示描述表，因此应用程序可以调用属性函数，尽管这样做会使图元文件设备专用化。**

rSize

指定绘制图元文件的大小（以 HIMETRIC **为单位）。**

说明

框架调用此函数来将此 OLE **项转换到一个图元文件中。此** OLE **项的图元文件表示用于在容器应用程序中显示此项。如果容器应用程序是用微软基础类库编写的，则由对应的** COleClientItem **对象的** Draw **成员函数使用此图元文件。**Draw **函数没有缺省实现。要将此项在指定的设备环境中绘制，应用程序必须重载** OnDraw **函数。**

请参阅 COleClientItem::Draw

COleServerItem::OnDrawEx

virtual BOOL OnDrawEx(CDC* *pDC*, DVASPECT *nDrawAspect*, CSize& *rSize*);

返回值

如果此项绘制成功则返回一个非零值；否则返回值为 0。

参数

pDC

指向一个 CDC **对象的指针，要在此对象上绘制该项。**DC **自动连接到属性** DC **上，因此应用此项可以调用属性函数，尽管这样做会使元文件设备专用化。**

nDrawAspect

一个 DVASPECT **枚举值。此参数可以取下列任何值：**

- DVASPECT_CONTENT **用这种方式表示的项可以作为一个嵌入对象显示在它的容器中。**

- DVASPECT_THUMBNAIL 用“拇指甲”方式表示项，以使它可以显示在一个浏览工具中。
- DVASPECT_ICON 用图标来表示项。
- DVASPECT_DOCPRINT 项被表示为就象它被用 File 菜单中的 Print 命令打印一样。

rSize

此项的大小（以 HIMETRIC 为单位）。

说明

框架调用此函数来进行所有绘制。当 DVASPECT 等于 DVASPECT_CONTENT 时，缺省实现是调用 OnDraw 函数，否则函数失败。

可重载此函数来为非 DVASPECT_CONTENT 的特征如 DVASPECT_ICON 或

DVASPECT_THUMBNAIL **提供数据。**

请参阅 COleServerItem::OnDraw

COleServerItem::OnGetClipboardData

virtual COleDataSource* OnGetClipboardData(BOOL *bIncludeLink*,
LPPOINT *lpOffset*, LPSIZE *lpSize*);

返回值

一个指向包含剪贴板数据的 COleDataSource **对象的指针。**

参数

bIncludeLink

如果链接数据将被拷贝到剪贴板上，则设置该参数为 TRUE**；如果服务器应用程序不支持链接，则将该参数设置为** FALSE**。**

lpOffset

鼠标光标距离此对象原点的偏移（以像素计算）。

lpSize

此对象的用像素表示的大小。

说明

框架调用这个函数来获取一个 COleDataSource 对象。此对象中包含将要由 CopyToClipboard 成员函数调用放到剪贴板上去的所有数据。OnGetClipbordData 的缺省实现调用 GetClipboardData。

请参阅 COleDataSource，COleDataSource::SetClipboard，COleServerItem::CopyToClipboard，COleServer::GetClipboardData

COleServerItem::OnGetExtent

virtual BOOL OnGetExtent(DVASPECT *nDrawAspect*, CSize& *rSize*);

返回值

如果成功则返回非零值；否则返回 0。

参数

nDrawAspect

指定 OLE **项的特征，此** OLE **项的边界将要被获取。此参数的值可以是下列任意值：**

- DVASPECT_CONTENT **用这种方式表示的项可以作为一个嵌入对象显示在它的容器中。**
- DVASPECT_THUMBNAIL **用“拇指甲”方式表示项，以使它可以**

显示在一个浏览工具中。

- DVASPECT_ICON 用图标来表示项。
- DVASPECT_DOCPRINT 项被表示为就像它被用 File 菜单中的 Print 命令打印一样。

rSize

一个 CSize 对象的引用，此对象将接收此 OLE 项的大小。

说明

框架调用此函数来获取此 OLE 项的以 HIMETRIC 为单位的大小。

如果容器应用程序是用微软基础类库编写的，则在调用对应的 COleClientItem 对象的 GetExtent 成员函数时，调用 OnGetExtent 函数。此函数的缺省实现不做任何事情。你必须自己来实现它。如果你希望在处理此 OLE 项的尺寸请求时

执行特殊的处理，则需要重载这个函数。

请参阅 COleClientItem::Draw，COleClientItem::GetExtent

COleServerItem::OnHide

virtual void OnHide();

说明

由框架调用此函数来隐藏 OLE 项。其缺省实现调用了 COleServerDoc::OnShowDocument（FALSE）。此函数还通知容器应用程序此 OLE 项已经被隐藏。如果你希望在隐藏一个 OLE 项时执行特殊的处理，则可以重载这个函数。

请参阅 COleServerItem::OnOpen，COleServerItem::OnShow，
COleServerItem::OnShowDocument

COleServerItem::OnInitFromData

virtual BOOL OnInitFromData(COleDataObject* *pDataObject*, BOOL *bCreation*);

返回值

如果成功则返回非零值；否则返回 0。

参数

pDataObject

指向一个 OLE **数据对象的指针，该对象包含了用来初始化该** OLE **项的各种格式的数据。**

bCreation

如果调用此函数来初始化一个刚由容器应用程序创建的 OLE **项，则将此参数设置为** TRUE。**如果调用此函数来替换一个已有的** OLE **项的内容，**

则将此参数设置为 FALSE。

说明

由框架调用这个函数来以 pDataObject 的内容初始化一个 OLE 项。如果 bCreation 的值是 TRUE，则当容器应用程序基于当前选择实现了 Insert New Object 时调用这个函数。所选数据用于创建这个新的 OLE 项。例如，当在一个电子数据表格程序中选择一段单元范围，然后用 Insert New Object 创建一个基于所选范围中的值的图表时。此函数的缺省实现不做任何事情。要实现从 pDataObject 所提供的格式中选择一种可以接收的格式，然后根据该格式中所提供的数据初始化此 OLE 项，则需要重载这个函数。这是一个高级的可重载函数。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IOleObject::InitFromData。

COleServerItem::OnOpen

virtual void OnOpen();

说明

由框架调用此函数，用来在服务器应用程序的一个独立实例中，而不是现场显示此 OLE **项。**

此函数的缺省实现是激活第一个显示包含此 OLE **项的文档的框架窗口；如果应用程序是一个小服务器应用程序，则此函数的缺省实现是显示主窗口。此函数还通知容器应用程序此** OLE **项已经打开。**

如果想要在打开一个 OLE **项时执行特殊的处理，则可重载这个函数。这尤其常用于链接项，例如在链接项打开时，将选择设置到链接上。**

请参阅 COleServerItem::OnShow

COleServerItem::OnQuieryUpdateItems

virtual BOOL OnQueryUpdateItems();

返回值

如果文档有需要更新的项则返回非零值；否则返回 0。

说明

框架调用此函数来确定当前服务器文档中是否有过时了的链接项。如果一个项的源文档已经改变了，而链接项还没有更新来反映文档中的变化，则认为此项已经过时了。

请参阅 COleServerItem::OnUpdate，COleServerItem::OnUpdateItems

COleServerItem::OnRenderData

virtual BOOL OnRenderData(LPFORMATETC *lpFormatEtc*,
LPSTGMFDIUM *lpStgMedium*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpFormatEtc

指向一个 FORMATETC **结构的指针，此结构用来指定所请求信息的格式。**

lpStgMedium

指向一个 STGMEDIUM **结构的指针，此结构将接收返回的数据。**

说明

由框架调用此函数来获取指定格式的数据。指定格式是由先前调用 DelayRenderData 或 DelayRenderFileData 成员函数进行延迟提交时放入 COleDataSource 对象中的格式。若提供的存储介质是一个文件或内存，则函数的缺省实现分别调用 OnRenderFileData 或 OnRenderGlobalData 函数。如果这两种格式都未提供，则缺省实现返回零，并且不做任何事情。

如果 lpStgMedium->tymed 是 TYMED_NULL，则将分配 STGMEDIUM 结构，并用 lpFormatEtc->tymed 所指定的内容来填充此 STGMEDIUM 结构。如果 lpStgMedium->tymed 不是 TYMED_NULL，则用数据来填充此 STGMEDIUM 结构。

可以重载这个函数来提供所请求格式和介质中的数据。根据数据的不同，应用

程序可能希望重载此函数的另一个版本来代替。如果数据很小且大小固定，则可重载 OnRenderGlobalData 函数。若数据放在一个文件中，或若数据的大小可变，则可重载 OnRerderFileData 函数。OnRenderData 是一个高级的可重载函数。

请参阅 COleServerItem::OnRenderFileData

COleServerItem::OnRenderFileData

virtual BOOL OnRenderFileData(LPFORMATETC *lpFormatEtc*, CFile* *pFile*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpFormatEtc

指向一个 FORMATETC 结构的指针，此结构指定所请求信息的格式。

pFile

指向一个 CFile 对象的指针，此对象中存放要提交的数据。

说明

当存储介质是一个文件时，由框架调用这个函数来获取指定格式的数据。指定的格式是先前用 DelayRenderData 成员函数进行延迟提交时放进 COleDataSource 对象中的格式。此函数的缺省实现只是简单返回 FALSE。

此函数是一个高级的可重载函数。可以重载这个函数来提供所请求格式和介质中的数据。根据数据的不同，应用程序可能想要重载该函数的其它一种版本来代替。如果想要处理多种存储介质，可以重载 OnRenderData 函数；如果数据放在文件中，或者数据的大小可变，则重载 OnRenderFileData 函数。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IDataObject::GetData 和 FORMATETC。

请参阅 COleServerItem::OnRenderData

COleServerItem::OnRenderGlobalData

virtual BOOL OnRenderGlobalData(LPFORMATETC *lpFormatEtc*, HGLOBAL* *phGlabal*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpFormatEtc

指向一个 FORMATETC **结构，此结构指定所请求信息的格式。**

phGlobal

指向一个全局内存句柄，此全局内存将存放所返回的数据。如果没有分配内存，则该参数可以是 NULL。

说明

当指定的存储介质是全局内存时，框架调用此函数以检取指定格式的数据。指定格式是先前用 DelayRenderData 成员函数进行延迟提交时放入 COleDataSource 对象中的格式。OnRenderGlobalData 函数的缺省实现只是简单返回 FALSE。

如果 phGlobal 为 NULL，则将分配一个新 HGLOBAL，并在 phGlobal 中返回此 HGLOBAL。否则，将用数据填充 phGlobal 所指定的 HGLOBAL。放入 HGLOBAL 中的数据量不能超过此内存块的当前大小。并且，此内存块不能再分配成更大的尺寸。

OnRenderGlobalData 是一个高级的可重载函数。可以重载此函数来提供所请求格式和介质中的数据。根据数据的不同，应用程序可能想要加载该函数的其它一种版本来代替。如果想要处理多种存储介质，可以重载 OnRenderData 函数；如果数据放在文件中，或者数据的大小可变，则重载 OnRenderFileData 函数。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 STGMEDIUM，FORMATETC 和 ReleaseStgMedium。

请参阅 COleDataSource::OnRecordData

COleServerItem::OnSetColorScheme

virtual BOOL OnSetColorScheme(const LOGPALETTE FAR* lpLogPalette);

返回值

如果使用了调色板则返回非零值；否则返回 0。

参数

lpLogPalette

指向一个 Windows 的 LOGPALETTE 结构的指针。

说明

由框架调用这个函数，用于指定编辑此 OLE 项时使用的调色板。如果容器应用程序是用微软基础类库编写的，则当对应的 COleClientItem 对象的 IOleObject::SetColorScheme 函数被调用时，此函数被调用。这个函数的缺省实现是返回 FALSE。如果想要使用推荐的调色板，则可以重载这个函数。服务器应用程序不需要使用建议的调色板。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IOleObject::SetColorScheme。

COleServerItem::OnSetData

virtual BOOL OnSetData(LPFORMATETC *pFormatEtc*,
    LPSTGMEDIUM *pStgMedium*, BOOL *bRelease*);

返回值

如果成功则返回非零值；否则返回 0。

参数

pFormatEtc

指向一个 FORMATETC **结构的指针，此结构用来指定数据的格式。**

pStgMedium

指向一个 STGMEDIUM **结构的指针，此结构中存放数据。**

bRelease

指示当完成此函数调用后，谁拥有存储介质的所有权。调用者决定谁负

责释放为存储介质分配的资源。调用者提供设置 bRelease 的值来实现这一功能。如果 bRelease 为一非零值，则服务器应用程序取得所有权，负责在使用完时释放介质。当 bRelease 为零时，调用者保留所有权，服务器项只能在此函数调用期间使用存储介质。

说明

由框架调用这个函数，用指定数据替换此 OLE 项的数据。服务器项不具备此数据的所有权，直到它成功地获取此数据。也就是说，如果服务器项返回零，则它不具有所有权。如果数据源取得所有权，则它通过调用 ReleaseStgMedium 函数来释放存储介质。

此函数的缺省实现不做任何事情。可重载此函数来实现用指定的数据替换此 OLE 项的数据。该函数是一个高级的可重载函数。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 STGMEDIUM，FORMATETC 和 ReleaseStgMedium。

请参阅 COleDataSource::OnSetData

COleServerItem::OnSetExtent

```
virtual BOOL OnSetExtent( DVASPECT nDrawAspect, const CSize& size );
```

返回值

如果成功则返回非零值；否则返回 0。

参数

nDrawAspect

指定一个 OLE 项的特征，此 OLE 项的边界要被检取。此参数的值可以

是下列任意值：

- DVASPECT_CONTENT 用这种方式表示的项可以作为一个嵌入对象显示在它的容器中。
- DVASPECT_THUMBNAIL 用“拇指甲”方式表示项，以使它可以显示在一个浏览工具中。
- DVASPECT_ICON 用图标来表示项。
- DVASPECT_DOCPRINT 项被表示为就象它被用 File 菜单中的 Print 命令打印一样。

size

一个 CSize 结构，用来指定 OLE 项的新尺寸。

说明

由框架调用这个函数，用来通知此 OLE **项在容器文档中有多少空间可以用于它。如果容器应用程序是用微软基础类库编写的，则当对应的** COleClientItem **对象的** SetExtent **成员函数被调用时，调用此函数。如果** nDrawAspect **是** DVASPECT_CONTENT**，则** OnSetExtent **函数的缺省实现是设置** m_sizeExtent **成员为所指定的尺寸；否则，该函数返回零。可以重载此函数来实现在改变此项的尺寸时执行某种特殊处理。**

请参阅 COleClientItem::SetExtent，COleServerItem::OnGetExtent，
COleServerItem::m_sizeExtent

COleServerItem::OnShow

```
virtual void OnShow();
```

说明

由框架调用这个函数，用来指导服务器应用程序在适当的地方显示这个 OLE **项。当容器应用程序的用户创建一个项或执行一个需要此项的动词（例如** Edit**）时，需要显示这个项，则通常调用此函数。该函数的缺省实现是试图进行现场激活。如果尝试失败了，则函数调用** OnOpen **成员函数来在一个独立的窗口中显示这个** OLE **项。**

如果你希望在显示一个 OLE **项时执行特殊的处理，则需要重载这个函数。**

请参阅 COleServerItem::OnOpen，COleClientItem::Activate

COleServerItem::OnUpdate

virtual void OnUpdate(COleServerItem* *pSender*, LPARAM *lHint*,
    CObject* *pHint*, DVASPECT *nDrawAspect*);

参数

pSender

指向修改此文档的项的指针。它可以是 NULL**。**

lHint

包含与修改有关的信息。

pHint

指向一个存储有关修改的信息的对象。

nDrawAspect

一个 DVASPECT_**枚举值。此参数的值可以是下列任意值：**

- DVASPECT_CONTENT **用这种方式表示的项可以作为一个嵌入对象显示在它的容器中。**
- DVASPECT_THUMBNAIL **用“拇指甲”方式表示项，以使它可以显示在一个浏览工具中。**

- DVASPECT_ICON　用图标来表示项。
- DVASPECT_DOCPRINT　项被表示为就象它被用 File 菜单中的 Print 命令打印一样。

说明

当一个项被修改时，框架调用这个函数。该函数的缺省实现是调用 NotifyChanged，而不考虑提示或发送者。

请参阅　COleServerItem::NotifyChangd

COleServerItem::OnUpdateItems

virtual void OnUpdateItems();

说明

由框架调用这个函数，用来更新服务器文档中的所有项。该函数的缺省实现是对文档中的所有 COleClientItem **对象调用** UpdateLink **函数。**

请参阅 COleServerItem::OnUpdate，COleServerItem::OnQueryUpdateItems

COleServerItem::SetItemName

void SetItemName(LPCTSTR *lpszItemName*)

参数

lpszItemName

指向项的新名字的指针。

说明

当你创建一个链接项时，调用这个函数来设置它的名字。该名字在文档内必须是唯一的。当调用服务器应用程序来编辑一个链接项时，应用程序利用这个名字来查找该项。不需要对嵌入项调用此函数。

请参阅 COleServerItem::GetItemName，COleLinkingDoc::OnGetLinkedItem

数据成员

COleServerItem::m_sizeExtent

```
CSize m_sizeExtent;
```

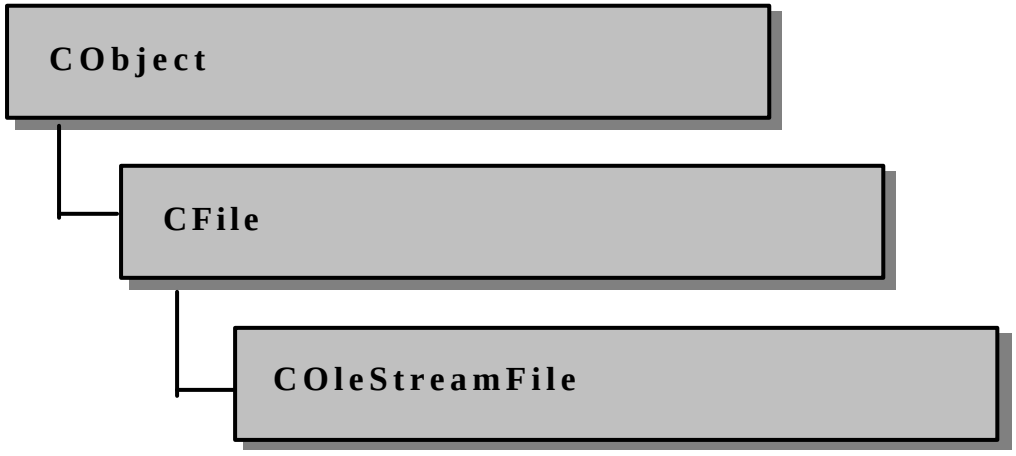
说明

此成员函数告知服务器应用程序在容器文档中对象有多少是可见的。

OnSetExtent **的缺省实现设置此成员。**

请参阅 COleServerItem::OnSetExtent

COleStreamFile



一个 COleStreamFile 对象将一个复合文件中的数据流（IStream）表示为 OLE 结构式存储的一部分。除非是一个内存流，否则在此流可打开或创建之前，必须存在一个 IStorage 对象。

COleStreamFile 对象的操作与 CFile 对象极其相似。

如果要获取更多有关操作流和存储器的信息，请参见“Visual C++程序员指南”中的文章“容器：复合文件”。

更多的信息，参见“OLE 2 程序员参考，卷 1”中的 IStream 和 IStorage。

```
#include <afxole.h>
```

COleStreamFile 类成员

Construction	
ColeStreamFile	构造一个 COleStreamFile 对象

Attributes and Operations	
Attach	将一个流与此对象相联系
CreateMemoryStream	从全局内存创建一个流，并将此流与此对象相关联
CreateStream	创建一个流，并将此流与此对象相关联
Detach	将此流从此对象中分离
GetStream	返回当前的流
OpenStream	安全地打开一个流，并将此流与此对象相关联

成员函数

COleStreamFile::Attach

void Attach(LPSTREAM *lpStream*);

参数

lpStream

指向要与此对象关联的 OLE 流，该参数不能是 NULL。

说明

此函数将所提供的 OLE **流与此** COleStreamFile **对象相关联。该对象必须已经与一个** OLE **流相关联。**

如果要获取更多的信息，请参见“ OLE 2 **程序员参考，卷** 1**”中的** IStream**。**

请参阅 COleStreamFile::Detach

COleStreamFile::COleStreamFile

COleStreamFile(LPSTREAM *lpStream* = NULL);

参数

lpStream

指向一个要连接到此对象上的 OLE **流的指针。**

说明

此函数创建一个 COleStreamFile **对象。如果** lpStream **为** NULL**，则此对象不与一个** OLE **项连接；否则，此对象与提供的** OLE **流相关联。**

如果要获取更多的信息，请参见“ OLE 2 **程序员参考，卷** 1**”中的** IStream**。**

请参阅 COleStreamFile::Attach，CFile

COleStreamFile::CreateMemoryStream

BOOL CreateMemoryStream(CFileException* *pError* = NULL);

返回值

如果成功地创建了流，则返回非零值；否则返回 0**。**

参数

pError

指向一个 CFileException **对象的指针或者是** NULL**，**NULL **表示创建操作的完成。如果你希望监视因试图创建流而可能产生的异常，则为此函数提供一个值。**

说明

此函数从全局的共享内存中安全地创建一个新的流，在此函数调用中失败是一种正常的、意料之中的情况。内存由 OLE **子系统分配。**

如果要获取更多的信息，请参见“ OLE 2 **程序员参考，卷** 1**”中的** CreateStreamOnHGlobal**。**

请参阅 COleStreamFile::OpenStream**，**COleStreamFile::CreateStream**，**

CFileException

COleStreamFile::CreateStream

BOOL CreateStream(LPSTORAGE *lpStorage*, LPCTSTR *lpszName*,
DWORD *nOpenFlags* = modeReadWrite|shareExclusive|modeCreate,
CFileException* pError = NULL);

返回值

如果成功地创建了流则返回非零值；否则返回 0。

参数

lpStorage

指向一个 OLE **存储器对象，此对象中包含要创建的流。**

lpszStreamName

要创建的流的名称。该参数不能为 NULL。

nOpenFlags

打开流时所用的存取方式。缺省时使用互斥，读/写及创建方式。有关可用方式的完整列表，参见 CFile::CFile。

pError

指向一个 CFileException 对象的指针，或者是 NULL。如果你希望监视因试图创建流而可能产生的异常，则为此函数提供一个值。

说明

此函数在所提供的存储器对象中安全地创建一个新的流，在此函数调用中，失败是一种正常的、意料之中的情况。如果打开失败，并且 *pError* 不是 NULL，则将抛出一个文件异常。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IStorage::CreateStream。

请　　参　　阅

COleStreamFile::OPenStream,COleStreamFile::CreateMemoryStream,CFileException

COleStreamFile::Detach

LPSTREAM Detach();

返回值

返回一个指向与此对象相关联的流的指针。

说明

此函数将流与对象分离，但是不关闭流。在程序终止之前，流必须以其它某种方式关闭。

如果要获取更多的信息，请参见“OLE2 **程序员参考，卷** 1**”中的** IStream**。**

请参阅 COleStreamFile::Attach

COleStreamFile::GetStream

IStream* GetStream() const;

返回值

是一个指向当前流界面（IStream**）的指针。**

说明

此函数用来返回一个指向当前流的指针。

COleStreamFile::OpenStream

BOOL OpenStream(LPSTORAGE *lpStorage*, LPCTSTR *lpszName*,
DWORD *nOpenFlags* = modeReadWrite|shareExclusive,
CFileException* *pError* = NULL);

返回值

如果成功地打开了流则返回非零值；否则返回 0。

参数

lpStorage

指向一个 OLE **存储器对象，此对象中包含要打开的流。该参数不可为** NULL。

lpszName

要打开的流的名称。该参数不可以为 NULL。

nOpenFlags

指定打开流时使用的存取方式。缺省使用互斥方式和读/写方式。有关可用方式的完整列表，参见 CFile::CFile。

pError

指向一个 CFileException 对象的指针，或者是 NULL。如果你希望监视因试图创建流而可能产生的异常，则为此函数提供一个值。

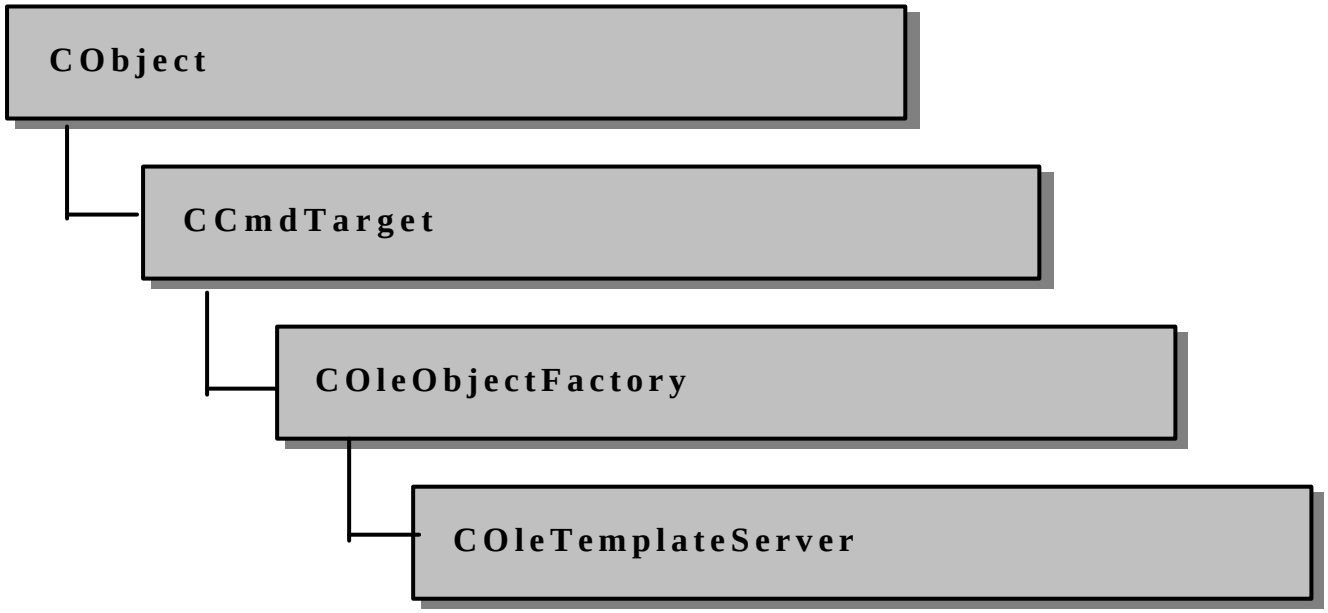
说明

此函数用来打开一个已经存在的流。如果打开失败，并且 *pError* 不为 NULL，则将抛出一个文件异常。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 IStorage::OpenStream。

请参阅 COleStreamFile::CreateStream，COleStreamFile::CreateMemoryStream，CfileException

COleTemplateServer



COleTemplateServer 类用于 OLE 可视编辑服务器、自动化服务器和链接容器（支持链接到嵌入项的应用程序）。该类从 COleObjectFactory 类中派生，通常应用程序可直接使用 COleTemplateServer 而无需使用一个 CDocTemplate 对象来管

理服务器程序文档。可使用 COleTemplateServer 来实现一个服务器应用程序，即一个可作为独立的应用程序运行的服务器应用程序。全服务器应用程序一般是多文档界面（MDI）应用程序，尽管单文档界面（SDI）应用程序也被支持。必须为应用程序所支持的每种服务器程序文档类型提供一个 COleTemplateServer 对象；即，如果服务器应用程序支持电子数据表格和图标，那么必须有两个 COleTemplateServer 对象。

COleTemplateServer 重载了由 COleObjectFactory 所定义的 OnCreateInstance 成员函数。这个成员函数由框架调用来创建一个适当类型的 C++对象。

#include <afxdisp.h>

请参阅 COleServerDoc，COleServerItem

COleTemplateServer 类成员

Construction	
ColeTemplateServer	构造一个 COleTemplateServer 对象

Operations	
ConnectTemplate	将一个文档模板连接到基础 COleObjectFactory 对象上
UpdateRegistry	向 OLE 系统注册表中注册这个文档类型

成员函数

COleTemplateServer::COleTemplateServer

COleTemplateServer();

说明

构造一个 COleTemplateServer **对象。**

有关使用 COleTemplateServer **类的简短描述，参见** COleLinkingDoc **概述。**

COleTemplateServer::ConnectTemplate

void ConnectTemplate(REFCLSID *clsid*, CDocTemplate* *pDocTemplate*,
    BOOL *bMultiInstance*);

参数

clsid

对模板所请求的 OLE **类** ID **的引用。**

pDocTemplate

指向文档模板的指针。

bMultiInstance

指示单个应用程序是否能支持多个实例。如果该参数为 TRUE，则为每一个创建对象的请求，启动应用程序的多个实例。

说明

此函数将 pDocTemplate 所支持的文档模板连接到基 COleObjectFactory 对象。

如果要获取更多的信息，请参见“OLE 2 程序员参考，卷 1”中的 CLSID Key。

请参阅 CDocTemplate

COleTemplateServer::UpdateRegistry

```
void UpdateRegistry( OLE_APPTYPE nAppType = OAT_INPLACE_SERVER,
    LPCSTR* rglpszRegister = NULL, LPCSTR FAR* rglpszOverwrite = NULL );
```

参数

nAppType

一个 OLE_APPTYPE 枚举类型值，该枚举是在 AFXDISP.H 中定义的。它可以是下列值之一：

- OAT_INPLACE_SERVER 服务器具有全服务器用户界面。
- OAT_SERVER 服务器值支持嵌入。
- OAT_CONTAINER 容器支持到嵌入项的链接。
- OAT_DISPATCH_OBJECT 对象是能够 IDispatch 的。
- OAT_DOCOBJECT_SERVER 服务器支持嵌入和复合模式的文档对象。

rglpszRegister

一个项的列表，只有在没有项存在时才要写入注册表中。

rglpszOverwrite

一个项的列表，不管是否有项存在都要写入注册表中。

说明

此函数从文档模板串中装载文件类型的信息，并将该信息放进 OLE 系统注册表中。

注册信息通过调用 CDocTemplate::GetDocString 来装载。所获取的子串是由索引 regFileTypeId，regFileTypeName 和 fileNewName 所标识的那些子串，如 GetDocString 参考页上所描述的。

如果 regFileTypeId 子串为空，或 GetDocString 调用由于某种原因失败，则 UpdateRegistry 函数失败，文件信息未写入注册表中。

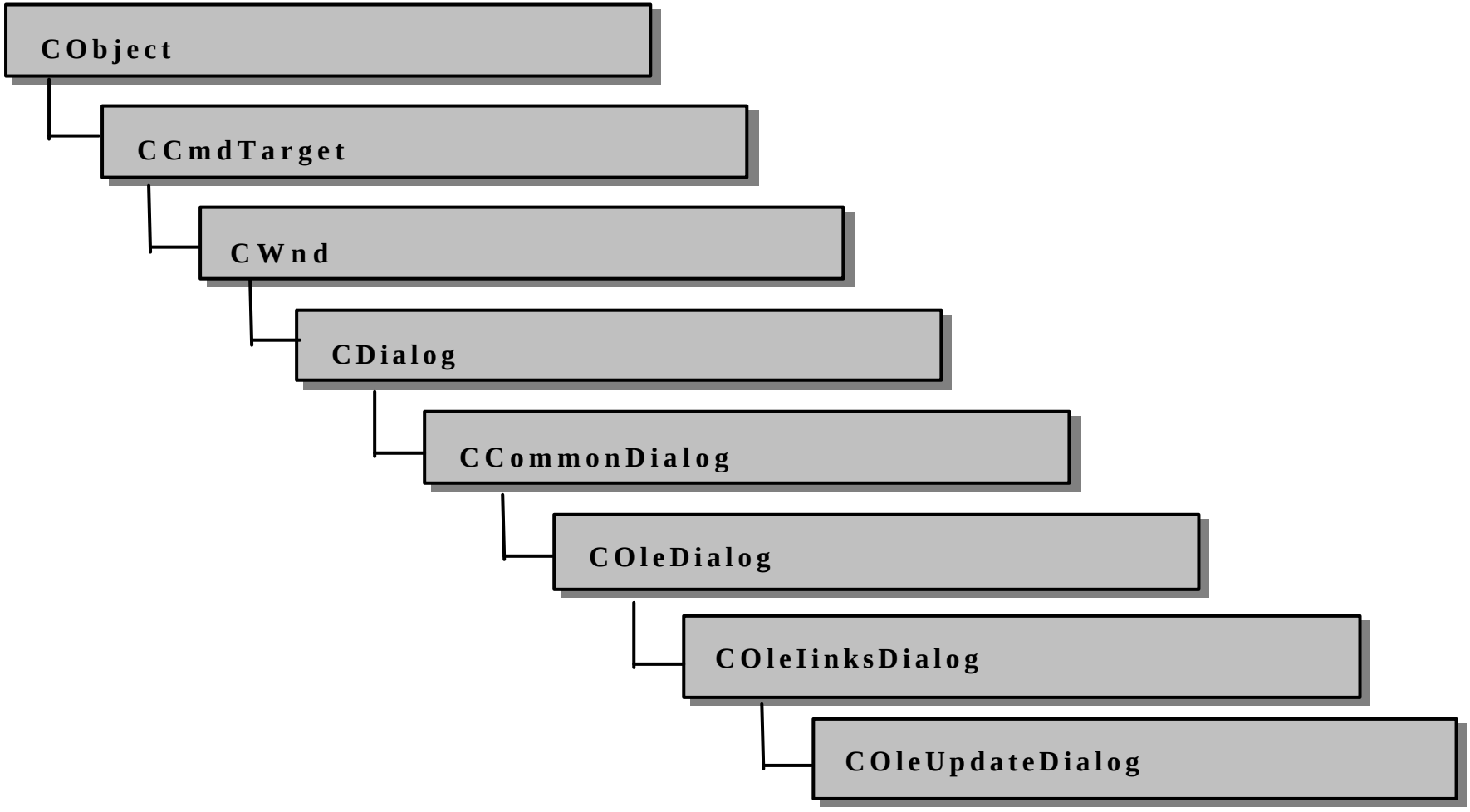
参数 *rglpszRegister* 和 *rglpszOverwrite* 中的信息通过调用

AfxOleRegisterServerClass 函数来写入注册表。当这两个参数为 NULL 时，注册表的缺省信息适用于多数应用程序。有关这些参数中的信息结构的信息，参见 AfxOleRegisterServerClass。

如果要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 IDispatch。

请参阅 CDocTemplate::GetDocString，AfxOleRegisterServerClass

COleUpdateDialog



COleUpdateDialog 类用于 OLE Edit Links 对话框的某种特殊情形，用于只需要更新文档中已经存在的链接或嵌入对象。

#include <afxodlgs.h>

请参阅 ColeLinksDialog

COleUpdateDialog 类成员

Construction	
ColeUpdateDialog	构造一个 COleUpdateDialog 对象
Operations	
DoModal	以一种更新方式显示 Edit Links 对话框

成员函数

COleUpdateDialog::COleUpdateDialog

COleUpdateDialog(COleDocument* *pDoc*, BOOL *bUpdateLinks* = TRUE,
BOOL *bUpdateEmbeddings* = FALSE, CWnd* = *pParentWnd* = NULL);

参数

pDoc

指向包含可能需要更新的链接的文档。

bUpdateLinks

一个标志，用于确定链接对象是否要更新。

bUpdateEmbeddings

一个标志，用于确定嵌入对象是否要更新。

pParentWnd

指向此对话框对象所属的父或属主窗口对象（CWnd 类型）。如果该参数为 NULL，则此对话框的父窗口属主为主应用程序窗口。

说明

此函数只构造 COleUpdateDialog 对象。要显示此对话框，需要调用 DoModal 函数。当应用程序想要只更新已经存在的链接或嵌入项时，应当用该类来代替 COleLinksDialog 类。

请参阅 COleDialog，COleLinksDialog，COleDocument，CWnd，CDialog，COleUpdateDialog::DoModal

COleUpdateDialog::DoModal

```
virtual int DoModal();
```

返回值

对话框的完成状态。是下列值之一：

- IDOK　如果对话框成功返回。
- IDCANCEL　如果当前文档中不存在需要更新的链接项或嵌入项。
- IDABORT　如果发生了一个错误。如果返回的是 IDABORT，调用 COleDialog::GetLast- Error 成员函数来获取更多有关所发生的错误类型的信息。可能发生的错误的列表，参见“ OLE 2.01 用户接口库”中的 OleUIEditLinks 函数。

说明

此函数以更新方式显示 Edit Links 对话框。除非用户选择了 Cancel 按钮，所有的链接项和/或嵌入项都会被更新。

请参阅 COleDialog::GetLastError，COleLinksDialog::DoModal

COleVariant

COleVariant **没有基类。**

一个 COleVariant **对象封装了** VARIANT **数据类型。这个数据类型用于** OLE **自动化。特别的，**DISPPARAMS **结构包含了一个指向** VARIANT **结构数组的指针。一个** DISPPARAMS**结构被用来将参数传递给** IDispatch::Invoke。

> **注意** 这个类是从 VARIANT 结构派生而来的。这意味着你可以在调用需要一个 VARIANT 参数时将一个 COleVariant 传递给它，并且 VARIANT 结构的数据成员也是在 COleVariant 中可以访问的数据成员。

两个相关联的 MFC **类** COleCurrency **和** COleDateTime **封装了不同的数据类型**

CURRENCY（VT_CY）和 DATE（VT_DATE）。更广的，COleVariant 类被用在 DAO 类中；参考这些类可以获得有关 COleVariant 类的典型用法，比如说 CDaoQueryDef 和 CDaoRecordset。

#include <afxdisp.h>

COleVariant 类成员

Construction	
ColeVariant	构造一个 COleVariant 对象
Operations	
Attach	将一个 VARIANT 附着在一个 ColeVariant 上
ChangeType	改变此 COleVariant 对象的可变类型
Clear	清除这个 COleVariant 对象
Detach	将一个 VARIANT 从一个 COleVariant 上分离，并返回该 VARIANT
SetString	将一个字符串设置为某种特定类型，比如说 ANSI

Operators	
Opetator LPCVARIANT	将一个 COleVariant 值转换为一个 LPCVARIANT

Operators	
Opetator LPVARIANT	将一个 COleVariant 值转换为一个 LPVARIANT
Opetator =	拷贝一个 COleVariant 值
Opetator ==	比较两个 COleVariant 值

Archive/Dump	
opetator<<	输出一个 COleVariant 值到 CArchive 或 CDumpContext
Opetator>>	从 CArchive 输入一个 COleVariant 对象

成员函数

COleVariant::Attach

void Attach(VARIANT& *varSrc*);

参数

varSrc

一个已存在的 VARIANT 对象，它将被附着到当前的 COleVariant 对象上。

说明

此函数用来将给定的 VARIANT 对象附着到当前的 COleVariant 对象上。此函数将 varSrc 的 VARTYPE 设置为 VT_EMPTY。

要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 VARIANT 和 VARTYPE 项。

请参阅 COleVariant::operator LPCVARIANT，COleVariant::operator LPVARIANT

COleVariant::COleVariant

```
COleVariant();
COleVariant(const VARIANT& varSrc);
COleVariant(const COleVariant& varSrc);
COleVariant(LPCVARIANT pSrc);
COleVariant(LPCTSTR lpszSrc);
COleVariant(LPCTSTR lpszSrc, VARTYPE vtSrc);
COleVariant(CString& strSrc);
COleVariant(BYTE nSrc);
COleVariant(short nSrc,VARTYPE vtSrc = VT_I2);
COleVariant(long lSrc,VARTYPE vtSrc = VT_I4);
COleVariant(const COleCurrency& curSrc);
COleVariant(float fltSrc);
COleVariant(double dblSrc);
COleVariant(const COleDateTime& dateSrc);
COleVariant(const CByteArray& arrSrc);
COleVariant(const CLongBinary& lbSrc);
```

参 数

varSrc

一个已存在的 COleVariant 或 VARIANT 对象，将要被拷贝到新的 COleVariant 对象中。

pSrc

一个指向将要被拷贝到新的 COleVariant 对象中的 VARIANT 对象的指针。

lpszSrc

一个以空字符结尾的，将要被拷贝到新的 COleVariant 对象中的字符串。

vtSrc

新的 COleVariant 对象的 VARTYPE。

strSrc

一个将要被拷贝到新的 COleVariant 对象中去的 CString 对象。

nSrc, lSrc

一个将要被拷贝到新的 COleVariant 对象中去的数字值。

vtSrc

新的 COleVariant 对象的 VARTYPE。

curSrc

一个将要被拷贝到新的 COleVariant 对象中去的 COleCurrency 对象。

fltSrc, dblSrc

一个将要被拷贝到新的 COleVariant 对象中去的数字值。

dateSrc

一个将要被拷贝到新的 COleVariant 对象中去的 COleDataTime 对象。

arrSrc

一个将要被拷贝到新的 COleVariant 对象中去的 CByteArray 对象。

lbSrc

一个将要被拷贝到新的 COleVariant 对象中去的 CLongBinary 对象。

说明

所有的这些构造函数都创建新的 COleVariant 对象，并将其初始化为指定的值。

有关这些构造函数的简短描述如下所示：

- COleVariant() 创建一个空的 COleVariant 对象，VT_EMPTY。
- COleVariant(*varSrc*) 拷贝一个已存在的 VARIANT 或 COleVariant 对象。保留可变类型。
- COleVariant(*pSrc*) 拷贝一个已存在的 VARIANT 或 COleVariant 对象。保留可变类型。
- COleVariant(*lpszSrc*) 将一个字符串拷贝到新对象中，VT_BSTR(UNICODE)。
- COleVariant(*lpszSrc*, *vtSrc*) 将一个字符串拷贝到新对象中。参数 *vtSrc* 必须是 VT_BSTR(UNICODE)或 VT_BSTRT(ANSI)。
- COleVariant(*strSrc*) 将一个字符串拷贝到新对象中，VT_BSTR(UNICODE)。
- COleVariant(*nSrc*) 将一个 8 位的整数拷贝到新对象中，VT_UI1。

- COleVariant(*nSrc*, *vtSrc*) 将一个 16 位整数（或布尔值）拷贝到新对象中。参数 *vtSrc* 必须是 VT_I2 或 VT_BOOL。
- COleVariant(*lSrc*, *vtSrc*) 将一个 32 位整数（或 SCODE 值）拷贝到新对象中。参数 *vtSrc* 必须是 VT_I4，VT_ERROR 或 VT_BOOL。
- COleVariant(*curSrc*) 将一个 COleCurrency 值拷贝到新对象中，VT_CY。
- COleVariant(*fltSrc*) 将一个 32 位的浮点数拷贝到新对象中，VT_R4。
- COleVariant(*dblSrc*) 将一个 64 位的浮点数拷贝到新对象中，VT_R8。
- COleVariant(*dateSrc*) 将一个 COleDataTime 值拷贝到新对象中，VT_DATE。
- COleVariant(*arrSrc*) 将一个 CByteArray 对象拷贝到新对象中，VT_EMPTY。
- COleVariant(*lbSrc*) 将一个 CLongBinary 对象拷贝到新对象中，

VT_EMPTY。

如果要获取更多的信息，请参见“ Platform SDK **”中的** VARIANT **和** VARTYPE **项。**

要获取更多关于 SCODE **的信息，参见“** Platform SDK **”中的“** COM **错误代码结构”。**

请参阅 COleVariant::operator =，CString，COleCurrency，COleDateTime

COleVariant::ChangeType

void ChangeType(VARTYPE *vartype*, LPVARIANT *pSrc* = NULL);

参数

vartype

此 COleVariant **对象的** VARTYPE。

pSrc

一个指向要被转换的 VARIANT 对象的指针。如果这个值是 NULL，则此 COleVariant 对象是用来作为转换源的。

说明

此函数用来转换这个 COleVariant 对象中的可变值的类型。

要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 VARIANT，VARTYPE 和 VariantChangeType 项。

请参阅 COleVariant::operator =

COleVariant::Clear

```
void Clear ();
```

说明

此函数用来清除 VARIANT。**函数为此对象将** VARTYPE **设置为** VT_EMPTY。COleVariant **的析构函数调用这个函数。**

如果要获取更多的信息，请参见“ Win32 SDK OLE **程序员参考”中的** VARIANT，VARTYPE **和** VariantClear **项。**

COleVariant::Detach

VARIANT Detach();

返回值

返回这个 COleVariant **对象的基础** VARIANT **值。**

说明

此函数用来将基础 VARIAINT 对象从 COleVariant 对象上分离。函数为此 COleVariant 对象将 VARTYPE 设置为 VT_EMPTY。

> 注意　在调用 Detach 后，由调用者负责基于结果中的 VARIANT 结构来调用 VariantClear。

如果要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 VARIANT，VARTYPE，和 VariantClear 项。

请参阅　COleVariant::operator LPCVARIANT，COleVariant::operator LPVARIANT

COleVariant::SetString

void SetString(LPCTSTR *lpszSrc*, VARTYPE *vtSrc*);

参数

lpszSrc

一个以 null **结尾的字符串，将被拷贝到新的** COleVariant **对象中。**

vtSrc

新的 COleVariant **对象的** VARTYPE**。**

说明

此函数用来将字符串设置为特别的类型。参数 *vtSrc* **必须是** VT_BSTR **（**UNICODE**）或** VT_BSTRT**（**ANSI**）。**SetString **通常被用来将字符串设置为** ANSI**，这是因为** COleVariant:: COleVariant **构造函数有一个字符串或字符串指针参数，并且没有** VARTYPE **是** UNICODE**。**

在一个非 UNICODE **构造中，一个** DAO **记录集希望字符串是** ANSI**。因此，对于使用** COleVariant **对象的** DAO **函数，如果你不是在创建一个** UNICODE **记录**

集，你就必须使用 COleVariant:: COleVariant(*lpszSrc, vtSrc*)形式的构造函数，并且将 *vtSrc* 设置为 VT_BSTRT（ANSI）；或使用 SetString 并且将 *vtSrc* 设置为 VT_BSTRT 来产生一个 ANSI 字符串。例如，CDaoRecordset 函数 CDaoRecordset::Seek 和 CDaoRecordset::SetFieldValue 使用 COleVariant 对象作为参数。如果这些 DAO 记录集不是 UNICODE 的，则这些对象必须是 ANSI。

请参阅 COleVariant:: COleVariant，CDaoRecordset::Seek，CDaoRecordset::SetFieldValue

操作符

COleVariant::operator =

const COleVariant& operator = (const VARIANT& *varSrc*);
const COleVariant& operator = (LPCVARIANT *pSrc*);

```
const COleVariant& operator = ( const COleVariant& Src );
const COleVariant& operator = (const LPCTSTR lpszSrc );
const COleVariant& operator = (const CString& strSrc );
const COleVariant& operator = (const BYTE nSrc );
const COleVariant& operator = (const short nSrc );
const COleVariant& operator = (const long lSrc );
const COleVariant& operator = (const COleCurrency& curSrc );
const COleVariant& operator = (const float fltSrc );
const COleVariant& operator = (const double dblSrc );
const COleVariant& operator = (const COleDateTime& dateSrc );
const COleVariant& operator = (const CByteArray& arrSrc );
const COleVariant& operator = (const CLongBinary& lbSrc );
```

说明

这些重载的赋值操作符都将源数据拷贝到 COleVariant **对象中。对于每一个操作符的简短描述如下所示：**

- operator = (*varSrc*) **将一个已存在的** VARIANT **或** COleVariant **对象拷贝到此对象中。**

- operator = (*pSrc*) 将由 pSrc 访问的 VARIANT 对象拷贝到此对象中。
- operator = (*lpszSrc*) 将一个以空字符结尾的字符串拷贝到此对象中，并将 VARTYPE 设置为 VT_BSTR。
- operator = (*strSrc*) 将一个 CStrin 对象拷贝到此对象中，并将 VARTYPE 设置为 VT_BSTR。
- operator = (*nSrc*) 将一个 8 位或 16 位的整数值拷贝到此对象中。如果 nSrc 是一个 8 位的值，则将 VARTYPE 设置为 VT_UI1。如果 nSrc 是一个 16 位的值并且 VARTYPE 的值是 VT_BOOL，则保持它；否则，将其设置为 VT_I2。
- operator = (*lSrc*) 将一个 32 位的整数值拷贝到此对象中。如果 VARTYPE 是 VT_ERROR，则保持它；否则，将其设置为 VT_I4。
- operator = (*curSrc*) 将一个 COleCurrency 对象拷贝到此对象中，并将

VARTYPE 设置为 VT_CY。

- operator = (*fltSrc*) 将一个 32 位的浮点值拷贝到此对象中，并将 VARTYPE 设置为 VT_R4。
- operator = (*dblSrc*) 将一个 64 位的浮点值拷贝到此对象中，并将 VARTYPE 设置为 VT_R8。
- operator = (*dateSrc*) 将一个 COleDateTime 对象拷贝到此对象中，并将 VARTYPE 设置为 VT_DATE。
- operator = (*arrSrc*) 将一个 CByteArray 对象拷贝到此对象中。
- operator = (*lbSrc*) 将一个 CLongBinary 对象拷贝到此对象中。

如果要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 VARIANT 和 VARTYPE 项。

请参阅 COleVariant::COleVariant，COleCurrency，COleDateTime

COleVariant::operator ==

BOOL operator == (const VARIANT& *varSrc*) const;
BOOL operator == (LPCVARIANT *pSrc*) const;

说明

此操作符比较两个不同的值，如果它们相等则返回非零值；否则返回 0。

请参阅 COleVariant::operator =

COleVariant::operator LPCVARIANT

operator LPCVARIANT() const;

说明

此操作符返回一个 VARIANT **结构，该结构的值是从此** COleVariant **对象中拷贝获得的。**

如果要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 VARIANT 项。

请参阅 COleVariant::operator LPVARIANT

COleVariant::operator LPVARIANT

operator LPVARIANT();

说明

此操作符用来访问 COleVariant 对象中的基础 VARIANT 结构。

> **警告** 改变由这个函数返回的指针所指的 VARIANT 结构中的值，就会导致改变此 COleVariant 对象中的值。

如果要获取更多的信息，请参见“Win32 SDK OLE 程序员参考”中的 VARIANT

项。

请参阅 COleVariant::operator LPCVARIANT

COleVariant::operator <<,>>

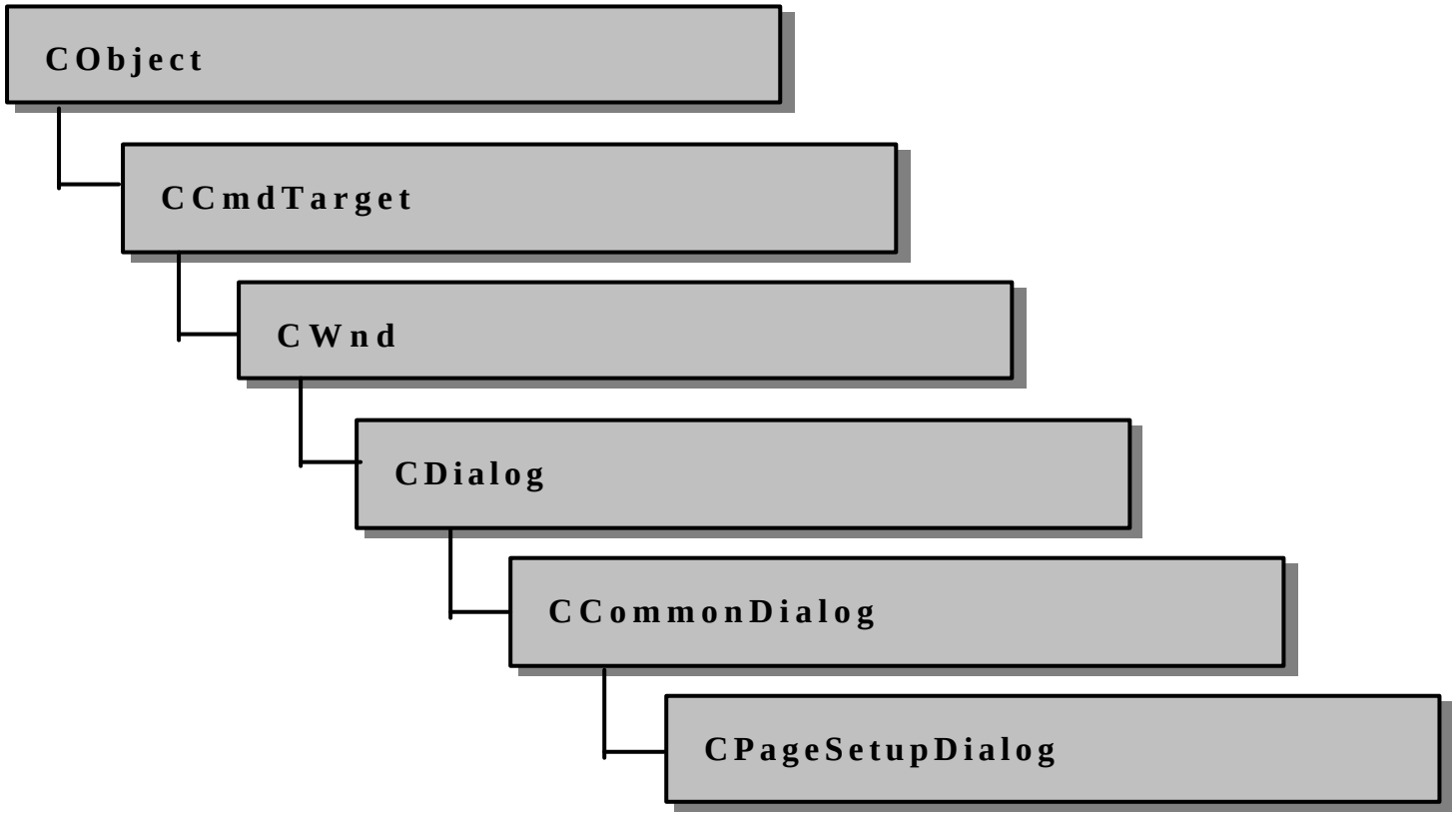
```
friend CDumpContext& AFXAPE operator<<( CDumpContext& dc, OleVariant varSrc );
friend CArchive& AFXAPI operator << ( CArchive& ar, COleVariant varSrc );
friend CArchive& AFXAPI operator >> ( CArchive& ar, COleVariant& varSrc );
```

说明

COleVariant **的插入操作符（**<<**）支持诊断转储并可以保存到一个文件。提取操作符（**>>**）支持从一个文件中提取内容。**

请参阅 CDumpContext，Carchive

CPageSetupDialog



CPageSetupDialog 类封装了 Windows 通用 OLE Page SetUp 对话框提供的服务，并且还添加了对设置和修改打印页边距的支持。这个类是设计来代替 Print Setup

对话框的。

要使用一个 CPageSetupDialog 对象，首先用 CPageSetupDialog 构造函数创建对象。一旦已经构造了对话框，就可以设置或修改数据成员 m_psd 的任意值来初始化对话框的控件。m_psd 结构是 PAGESETUPDLG 类型。Win32 SDK 文档中的“设置打印页”主题中给出了初始化这个结构的例子。

在初始化对话框控件之后，调用 DoModal 成员函数来显示对话框，并允许用户选择打印选项。DoModal 返回值说明了用户是选择了 OK（IDOK）还是选择了 Cancel（IDCANCEL）按钮。

如果 DoModal 返回的是 IDOK，你就可以用几个 CPageSetupDialog 的成员函数，或访问 m_psd 数据成员来获取用户输入的信息。

注意　在通用的 0LE Page Setup 对话框被取消后，用户作出的任何改变

都不会被框架保存。应用程序必须自己负责来将对话框中的所有值保存到一个永久的地方，比如应用程序文档或应用程序类的成员。

```
#include <afxdlgs.h>
```

CPageSetupDialog 类成员

Attributes	
CreatePrinterDC	为打印创建一个设备环境
GetDeviceName	返回打印机的设备名称
GetDevMode	返回打印机的当前 DEVMODE
GetDiverName	返回打印机使用的驱动程序
GetMargins	返回打印机当前的页边距设置
GetPortName	返回输出端口名
GetPaperSize	返回打印机的纸张大小

Construction	
CPageSetupDialog	构造一个 CPageSetupDialog 对象

Data Members	
m_psd	一个用来定制 CPageSetupDialog 对象的结构

Operations	
DoModal	显示对话框并让用户做一次选择

Overridables	
OnDrawPage	框架调用来给出一个打印页的屏幕图像。
PreDrawPage	在给出一个打印页的屏幕图像之前由框架调用

成员函数

CPageSetupDialog:: CPageSetupDialog

```
CPageSetupDialog( DWORD dwFlags = PSD_MARGINS |
PSD_INWININIINTLMEASURE,
    CWnd* pParentWnd = NULL );
```

参数

dwFlags

你可以使用一个或多个标志来定制对话框的设置。这些值可以用位或操作符来组合。这些值的意义如下所示：

- PSD_DEFAULTMINMARGINS 将允许的页边距的最小宽度设置得与打印机允许的最小宽度一样。如果 PSD_MARGINS 和 PSD_MINMARGINS 标志也被指定了，则这个标志可以忽略。
- PSD_INWININIINTLMEASURE 不执行。
- PSD_MINMARGINS 使系统使用在 rtMinMargin 成员中指定的值作为允许的左、上、右和下边距的最小宽度。系统将禁止用户输入的宽度小于指定的最小值。如果 PSD_MINMARGINS 没有被指定，则系统将允许的最小宽度设置得与打印机允许的一样。

- PSD_MARGINS 激活边距控制区域。
- PSD_INTHOUSANDTHSOFINCHES 使对话框的单元按 1/1000 英寸来测量。
- PSD_INTHUNDREDTHSOFMILLIMETERS 使对话框的单元按 1/100 毫米来测量。
- PSD_DISABLEMARGINS 使边距对话框控件无效。
- PSD_DISABLEPRINTER 使 Printer 按钮无效。
- PSD_NOWARNING 当没有缺省的打印机时，禁止显示警告消息。
- PSD_DISABLEORIENTATION 使页面方向控件无效。
- PSD_RETURNDEFAULT 不显示对话框就使 CPageSetupDialog 返回 DEVMODE 和 DEVNAMES 结构，这两个结构是为系统缺省的打印机而初始化的。它假设 hDevNames 和 hDevMode 都是 NULL；否则，函

数就返回一个错误。如果系统的缺省打印机是由一个老的打印机驱动程序(早于 Windows 3.0 版本)驱动的,则只返回 hDevNames;hDevMode 是 NULL。

- PSD_DISABLEPAPER 使页面选择控件无效。
- PSD_SHOWHELP 使对话框显示 Help 按钮。如果这个标志被指定,则 hwndOwner 成员必须是 NULL。
- PSD_ENABLEPAGESETUPHOOK 使 hook 函数在 lpfnSetupHook 中被指定。
- PSD_ENABLEPAGESETUPTEMPLATE 使操作系统用由 hInstance 和 lpSetupTem- plateName 标识的对话框模板来创建对话框。
- PSD_ENABLEPAGESETUPTEMPLATEHANDLE 表明 hInstance 标识一个包含预先载入的对话框模板的数据块。如果此标志被指定,则系

统忽略 lpSetupTemplate- Name。

- PSD_ENABLEPAGEPAINTHOOK 使在 lpfnPagePaintHook 中指定的钩子函数有效。
- PSD_DISABLEPAGEPAINTING 使对话框的绘画区无效。

pParentWnd

指向对话框的父或属主的指针。

说明

此函数用来构造一个 CPageSetupDialog 对象。使用 DoModal 函数来显示此对话框。

请参阅 CPrintDialog，CPageSetupDialog

CPageSetupDialog::CreatePrinterDC

HDC CreatePrinterDC();

返回值

返回新创建的打印机设备环境（DC）的句柄。

说明

从 DEVMODE **和** DEVNAMES **结构创建一个打印机设备环境。**

请参阅 CPageSetupDialog::GetDevMode，CPageSetupDialog::GetDeviceName，
CPageSetupDialog::GetDriverName

CPageSetupDialog::DoModal

virtual int DoModal();

返回值

返回 IDOK 或 IDCANCEL。如果返回的是 IDCANCEL，则调用 Windows CommDlgExtended- Error 函数来确定是否发生了一个错误。

IDOK 和 IDCANCEL 都是常量,它们用来表明用户选择的是 OK 按钮还是 Cancel 按钮。

说明

此函数用来显示 Windows 通用 OLE Page Setup 对话框，并允许用户选择不同的打印设置选项，比如打印边距、页面大小、页面方向，和打印机目标。另外，用户页可以访问如网络位置和所选打印机的属性等打印机设置选项。

如果你希望通过设置 m_psd 结构的成员来初始化不同的 Page Setup 对话框选

项，你必须在调用 DoModal 之前和构造此对话框之后进行。在调用 DoModal 之后，调用其它的成员函数来获取用户在对话框中输入的设置和信息。

如果你想传送用户输入的当前设置，请调用 CWinApp::SelectPrinter。这个函数读取来自 CPageSetupDialog 对象的信息并初始化和选择一个具有适当属性的新打印机 DC。

```
AfxGetApp() – >SelectPrinter(dlg.m_psd.hDevNames, dlg.m_psd.hDevMode );
```

请参阅 CPageSetupDialog::m_psd

CPageSetupDialog::GetDeviceName

```
CString GetDeviceName() const;
```

返回值

由 CPageSetupDialog 对象使用的设备名称。

说明

在 DoModal **之后调用此函数来获取当前选择的打印机名称。**

CPageSetupDialog::GetDevMode

LPDEVMODE GetDevMode() const;

返回值

返回 DEVMODE **数据结构，该结构包含了有关设备初始化和打印机驱动程序环境的信息。你必须用** Windows GlobalUnlock **函数解锁这个结构所占的内存，这在“** Platform SDK **”中有描述。**

说明

在 DoModal **之后调用此函数来获取有关** CPageSetupDialog **对象的打印机设备**

环境的信息。

CPageSetupDialog::GetDriverName

CString GetDriverName()Const；

返回值

返回当前选择的打印机设备驱动程序的名称。

说明

在 DoModal 之后调用此函数来获取当前选择的打印机设备驱动程序的名称。

请参阅　CPageSetupDialog::GetDeviceName，CPageSetupDialog::GetDevMode，
CPageSetupDialog::GetPortName

CPageSetupDialog::GetMargins

void GetMargins(LPRECT *lpRectMargins*, LPRECT *lpRectMinMargins*) const;

参数

lpRectMargins

指向 RECT **结构或** CRect **结构的指针，这两个结构描述（用** 1/1000 **英寸或** 1/100 **毫米）了当前选择的打印机的打印边距。如果对这个矩形并不感兴趣，可以将此参数设置为** NULL。

lpRectMinMargins

指向 RECT **结构或** CRect **结构的指针，这两个结构描述（用** 1/1000 **英寸或** 1/100 **毫米）了当前选择的打印机的最小打印边距。如果对这个矩形并不感兴趣，可以将此参数设置为** NULL。

说明

在 DoModal **之后调用这个函数来获取打印机设备驱动程序的边距。**

CPageSetupDialog::GetPaperSize

CSize GetPaperSize() const；

返回值

一个包含打印纸张尺寸（以 1/1000 **英寸或** 1/100 **毫米为单位）的** CSize **对象。**

说明

此函数用来获取打印纸张的尺寸。

CPageSetupDialog::GetPortName

CString GetPortName() const;

返回值

返回当前所选打印端口的名称。

说明

在 DoModal **之后调用此函数来获取当前所选打印端口的名称。**

请参阅 CPageSetupDialog::GetDeviceName，CPageSetupDialog::GetDriverName

CPageSetupDialog::OnDrawPage

virtual UINT OnDrawPage(CDC* *pDC*, UINT *nMessage*, LPRECT *lpRect*);

返回值

如果被处理则返回非零值；否则返回 0。

参数

pDC

指向打印机设备环境的指针。

nMessage

指明一个消息，表示当前绘画页面的区域。可以是下列值之一：

- WM_PSD_FULLPAGERECT **整个页面区。**
- WM_PSD_MINMARGINRECT **当前最小边距。**
- WM_PSD_MARGINRECT **当前边距。**
- WM_PSD_GREEKTEXTRECT **页面的内容。**

- WM_PSD_ENVSTAMPRECT 为贴邮票而保留的区域。
- WM_PSD_YAFULLPAGERECT 用于返回地址的区域。这个区域延伸到示例页区域的边界。

lpRect

指向一个 CRect 或 RECT 对象的指针，这两个对象包含了绘画区的坐标。

说明

由框架调用来画一个打印页面的屏幕图像。然后这个图像作为通用 OLE Page Setup 对话框的一部分来显示。函数的缺省实现是画一个文本页的图像。

重载这个函数可以定制是画图像的指定区域，还是画整个图像。你可以使用 switch 和 case 语句来检查 *nMessage* 的值。例如，为了定制获取表示图像的内容，你可以使用下面的例子代码：

```
Switch( nType )
```

```
{
   case WM_PSD_GREEKTEXTRECT:
      DrawMyImage( pDC, lpRect);              //画我的指定图像。
      return 1;
   default:
      return   ::Draw(CDC* pDC, UINT nDrawType, LPRECT lpRect);
};
```

注意：你不需要处理 *nMessage* 的每一个取值。可以选择处理图像的某一部分，某几个部分或整个区域。

请参阅 CPageSetupDialog::PreDrawPage

CPageSetupDialog::PreDrawPage

virtual UINT PreDrawPage(WORD *wPaper*, WORD *wFlags*,LPPAGESETUPDLG *pPSD*);

返回值

如果被处理则返回非零值；否则返回 0。

参数

wPaper

指定一个用来表明纸张尺寸的值。这个值可以是一个 DMPAPER_**值，在** DEVMODE **结构的描述中有这些值的列表。**

wFlags

表明纸张或信封的方向,以及表明打印机是点阵式的还是 HPPCL(Hewlett Packard Printer Control Language ）**设备。此参数可以是下列值之一：**

- 0x001 **纸张是横向放置（点阵式）。**
- 0x003 **纸张是横向放置（**HPPCL）。

- 0x005 纸张是纵向放置（点阵式）。
- 0x007 纸张是纵向放置（HPPCL）。
- 0x00b 信封是横向放置（HPPCL）。
- 0x00d 信封是纵向放置（点阵式）。
- 0x019 信封是横向放置（HPPCL）。
- 0x01f 信封是纵向放置（点阵式）。

pPSD

指向一个 PAGESETUPDLG 结构的纸张。要获取有关这个结构的更多消息，请参见 Win32 文档。

说明

在画用于页面的屏幕图像之前由框架调用。重载这个函数来定制图像的绘画。如果你重载这个函数并返回 TRUE，则你必须画整个图像。如果你重载这个函

数并返回 FALSE，则框架会画出整个缺省的图像。

请参阅　CPageSetupDialog::OnDrawPage

数据成员

CPageSetupDialog::m_psd

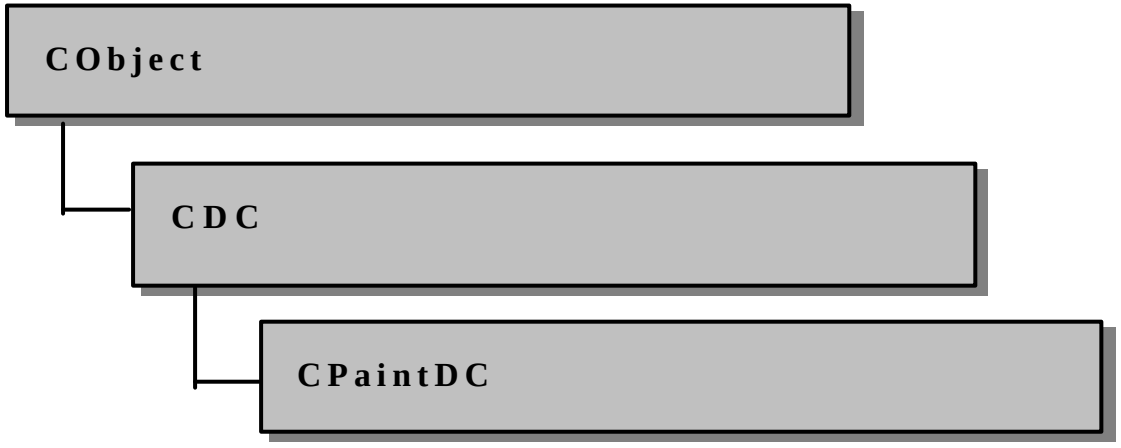
说明

一个 PAGESETUPDLG 类型的结构，它的成员保存了对话框对象的特征。在构造了一个 CPageSetupDialog 对象之后，在调用 DoModal 之前，你可以使用 m_psd 来设置对话框的不同方面。

如果你直接修改 m_psd 数据成员，则你将重载任何缺省的行为。

要获取有关 PAGESETUPDLG 结构的更多消息，请参见 Win32 文档。

CPaintDC



CPaintDC 类是一个来自 CDC 的设备环境类。它在构造期间执行 CWnd::BeginPaint，在析构期间执行 CWnd::EndPaint。

一个 CPaintDC 对象只在响应一个 WM_PAINT 消息的时候被使用，通常是在你的 OnPaint 消息处理成员函数中。

include <afxwin.h>

CPaintDC 类成员

Data Members	
m_ps	包含了用于画客户区的 PAINTSTRUCT
m_hWnd	CPaintDC 对象所附着的 HWND

Construction	
CpaintDC	构造一个连接到指定的 CWnd 上的 CPaintDC 对象

成员函数

CPaintDC:: CpaintDC

CPaintDC（CWnd* *pWnd*）;

throw(CResourceException);

参数

pWnd

指向一个 CPaintDC **对象所属的** CWnd **对象。**

说明

构造一个 CPaintDC **对象，准备用于绘画的应用程序窗口，并将** PAINTSTRUCT **结构保存在** m_ps **成员变量中。**

如果 Windows GetDC **调用失败，则抛出一个异常（**CResourceException **类型）。如果** Windows **已经分配了它所有的可利用的设备环境，则没有可用的设备环境了。在** Windows **下，你的应用程序在给定时刻竞争五个可用的公共显示环境。**

数据成员

CPaintDC::m_hWnd

说明

是 CPaintDC 对象所附着的 HWND。m_hWnd 是一个 HWND 类型的被保护变量。

CPaintDC::m_ps

说明

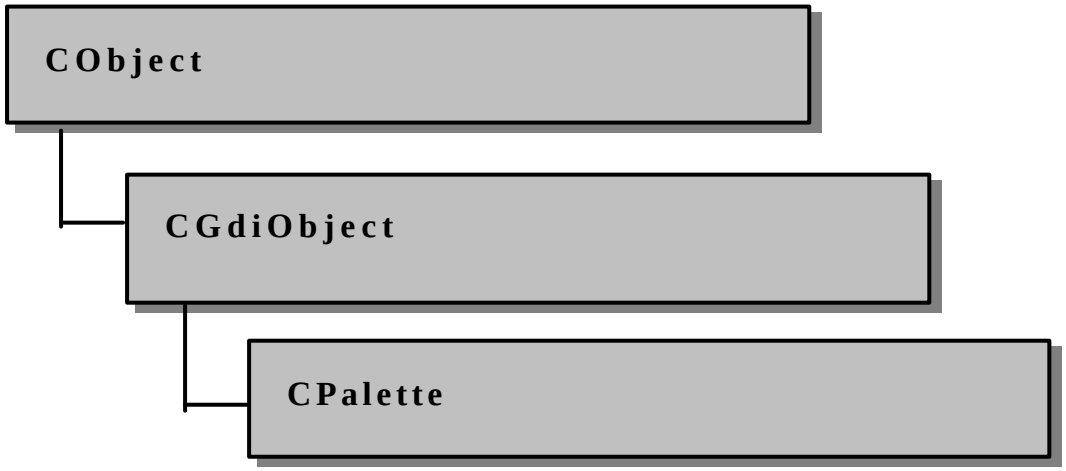
m_ps 是一个 PAINTSTRUCT 类型的公有成员变量。它是一个由

CWnd::BeginPaint 传递和填充的 PAINTSTRUCT。

这个 PAINTSTRUCT 包含了应用程序用来绘制与 CPaintDC 对象相关的窗口的客户区信息。

注意，你可以通过 PAINTSTRUCT 来访问设备环境句柄。但是，你可以更直接地通过 m_hDC 成员变量来访问这个句柄，m_hDC 是 CPaintDC 从 CDC 继承来的。

CPalette



CPalette 类封装了 Windows 的调色板。调色板在一个应用程序和一个颜色输出设备（比如一个显示设备）之间提供了一个接口。这个接口允许此应用程序充分使用输出设备的颜色处理能力，而不会干涉其它应用程序显示的颜色。

Windows使用应用程序的逻辑调色板（一个所需颜色的列表）和系统调色板（定义了可以使用的颜色）来确定使用的颜色。

一个 CPalette 对象提供了用于操作对象所涉及的调色板的成员函数。构造一个 CPalette 对象，并使用它的成员函数来创建实际的调色板，一个图形设备接口（GDI）对象，并操作他的项和其它的属性。

#include <afxwin.h>

请参阅 CPalette::GetPaletteEntries，CPalette::SetPaletteEntries

CPalette 类成员

Construction	
Cpalette	构造一个 CPalette 对象，没有被附着的 Windows 调色板。你必须用一个初始化成员函数来初始化这个 CPalette 对象后才能使用它

Initialization	
CreatePalette	创建一个 Windows 调色板并将它附着在 CPalette 对象上
CreareHalftonePalette	创建一个用于设备环境的半调调色板，并将它附着在此 CPalette 对象上

Operations	
FromHandle	当给予一个 Windows 调色板对象时返回一个指向一个 CPalette 对象的指针
AnimatePalette	替换由 CPalette 对象标识的逻辑调色板中的项。应用程序不需要更新它的客户区，因为 Windows 会立即将新的项映射到系统调色板
GetNearestPaletteIndex	返回逻辑调色板中最匹配某个颜色值的项 0 的索引
ResizePalette	将 CPalette 对象所指定的逻辑调色板的大小改变为指定的项数

Attributes	
GetEntryCount	获取一个逻辑调色板中的调色板项数目
GetPaletteEntries	获取一个逻辑调色板中一段范围内的调色板项
SetPaletteEntries	设置逻辑调色板中一段表项范围内的 RGB 颜色值和标志
Operator HPALETTE	返回附着在 CPalette 上的 HPALETTE

成员函数

CPalette::AnimatePalette

```
void AnimatePalette( UINT nStartIndex, UINT nNumEntries,
        LPPALETTEENTRY lpPaletteColors );
```

参数

nStartIndex

指定要激活的调色板中的第一项。

nNumEntries

指定要激活的调色板中的项数。

lpPaletteColors

指向一个 PALETTEENTRY 结构数组中的第一个成员，此结构用于替换由 *nStartIndex* 和 *nNumEntries* 所标识的调色板项。

说明

此函数替换附着在 CPalette 对象上的逻辑调色板。当一个应用程序定义 AnimatePalette 时，它不用更新它的客户区，因为 Windows 会立即将新项映射到系统调色板。

AnimatePalette 函数只改变这样的项:在此 CPalette 对象所连接的 LOGPALETTE 结构中，此项所对应的 palPaletteEntry 成员中设置了 PC_RESERVED 标志的调色板项。要获取有关这个结构的更多信息，请参见“Win 32 SDK 程序员参考”

中的 LOGPALETETTE。

请参阅 CPalette::CreatePalette，::AnimatePalette

CPalette::CPalette

CPalette();

说明

构造一个 CPalette **对象。这个对象没有连接调色板，直到你调用了** CreatePalette **来将一个调色板连接到此对象上。**

请参阅 CPalette::CreatePalette

CPalette::CreateHalftonePalette

BOOL CreateHalftonePalette(CDC* *pDC*);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

pDC

标识设备环境。

说明

为设备环境创建一个半调的调色板。当设备环境的拉伸方式被设置为 HALFTONE **时，应用程序应该创建半调调色板。在调用** CDC::StretchBlt **或**::StretchDIBits **函数之前，应当将** CreateHalftonePalette **成员函数所返回的逻辑半调调色板选入并实现到设备环境。**

要获取更多关于 CreateHalftonePalette **和** StretchDIBits **的信息，请参见“** Win32 SDK **程序员参考”。**

请参阅 CDC::RealizePalette，CDC::SelectPalette，CDC::SetStretchBltMode，:: CreateHalftonePalette，::StretchDIBits

CPalette::CreatePalette

BOOL CreatePalette(LPLOGPALETTE *lpLogPalette*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpLongPalette

指向有关 LOGPALETTE 结构，此结构包含了有关逻辑调色板中的颜色的信息。

说明

通过创建一个 Windows 逻辑调色板并将它连接到 CPalette 对象上来初始化这个 CPalette 对象。

要获取更多关于 LOGPALETTE 的信息，请参见“Win 32 SDK 程序员参考”。

请参阅 ::CreatePalette，LOGPALETTE

CPalette::FromHandle

static CPalette* PASCAL FormHandle(HPALETTE *hPalette*);

返回值

如果成功则返回一个指向 CPalette **对象的指针；否则返回** NULL**。**

参数

hPalette

一个 Windows GDI **调色板句柄。**

说明

当给予一个 Windows**调色板对象时，该函数返回指向一个** CPalette **对象的指针。如果此** Windows **调色板要连接一个** CPalette **对象，则创建一个临时的** CPalette **对象，并将此对象连接到此** Windows**调色板上。这个临时** CPalette **对象只会有效到下次应用程序的事件循环中有空闲时间时，到那时所有临时图形对象都将被删除。也就是说，临时对象仅在处理一条窗口消息的期间有效。**

CPalette::GetEntryCount

int GetEntryCount();

返回值

返回一个逻辑调色板中的项数。

说明

调用这个成员函数来获取一个给定的逻辑调色板中的项数。

请参阅 CPalette::GetPaletteEntries，CPalette::SetPaletteEntries

CPalette::GetNearestPaletteIndex

UINT GetNearestPaletteIndex(COLORREF *crColor*) const;

返回值

返回值为逻辑调色板某一项的索引。该项中包含最匹配指定颜色的调色板中的颜色。

参数

crColor

指定要匹配的颜色。

说明

此函数返回逻辑调色板中最匹配指定颜色值的项的索引。

请参阅 ::GetNearestPaletteIndex

CPalette::GetPaletteEntries

UINT GetPaletteEntries(UINT nStartIndex, UINT nNumEntries,
LPPALETTEENTRY *lpPaletteColors*) const;

返回值

如果函数成功，则返回从逻辑调色板中获取的项数；否则，返回 0。

参数

nStartIndex

指定在逻辑调色板中要被获取的第一项。

nNumEntries

指定在逻辑调色板中要获取的项数。

lpPaletteColors

指向一个 PALETTEENTRY **数据结构数组，此结构用来接收调色板项。**

数组中必须至少包含与 nNumEntries 所指定的项数一样多的数据结构。

说明

此函数返回逻辑调色板中一定范围内的调色板项。

请参阅 ::GetPaletteEntries，CPalette::SetPaletteEntries

CPalette::operator HPALETTE

operator HPALETTE() const;

返回值

如果成功则返回一个由 CPalette 对象表示的 Windows GDI 对象句柄；否则返回 0。

说明

此操作符用来获取与 CPalette 对象连接的 Windows GDI 句柄。此操作符是一个强制转换操作符，它支持直接使用一个 HPALETTE 对象。

要获取更多关于使用图形对象的信息，请参见“Win32 SDK 程序员参考”中的文章“图形对象”。

CPalette::ResizePalette

BOOL ResizePalette(UINT *nNumEntries*);

返回值

如果成功地改变了调色板的大小，则返回非零值；否则返回 0。

参数

nNumEntries

指定调色板在改变大小后的项数。

说明

此函数将一个连接于 CPalette **对象的逻辑调色板的大小改变为** *nNumEntries* **所指定的项数。如果一个应用程序调用** ResizePalette **来减小一个调色板的大小，则保留在改变大小后的调色板中的项是没有变化的。如果应用程序调用** ResizePalette **来增大调色板的大小，则增加的调色板项被设置为黑（即红、绿、蓝的值均为** 0**），并且所有增加项的标志均为** 0**。**

要获取更多关于 Windows API ResizePalette **的信息，请参见“** Win32 SDK **程序员参考”中的**::ResizePalette**。**

请参阅 ::ResizePalette

CPalette::SetPaletteEntries

UINT SetPaletteEntries(UINT *nStartIndex*, UINT *nNumEntries*,
LPPALETTEENTRY *lpPaletteColors*);

返回值

如果成功则返回在逻辑调色板中设置的项数；否则返回 0。

参数

nStartIndex

指定在逻辑调色板中要设置的第一项。

nNumEntries

指定在逻辑调色板中要设置的项数。

lpPaletteColors

指向一个用来接收调色板项的 PALETTEENTRY 数据结构数组。该数组必须包含至少与 nNumEntries 所指定项数一样多的数据结构。

说明

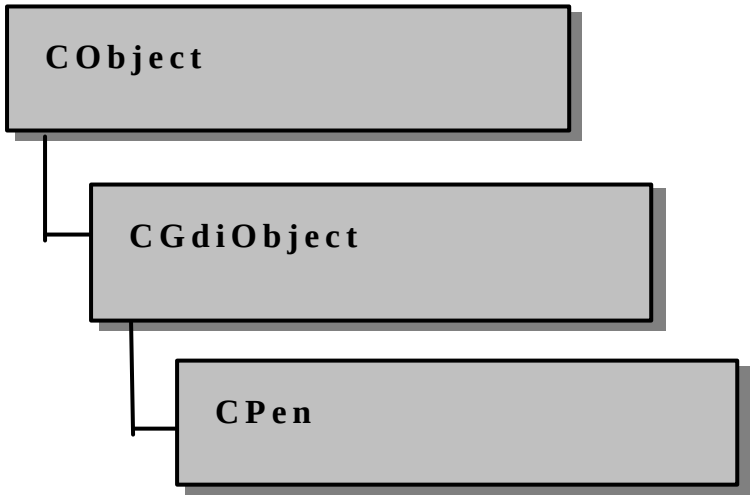
此函数用来设置一个逻辑调色板中一定范围内的项的 RGB 颜色值和标志。

如果当应用程序调用 SetPaletteEntries 函数时此逻辑调色板被选入一个设备环境中，则该调用所做的变化在应用程序调用 CDC::RealizePalette 之前不会起作用。

要获取更多关于 Windows PALETTEENTRY 结构的信息，请参见“Win32 SDK 程序员参考”中的 PALETTEENTRY。

请参阅 CDC::RealizePalette，CPalette::GetPaletteEntries，::SetPaletteEntries

CPen



CPen 类封装了 Windows 图形设备接口（GDI）画笔。

#include <afxwin.h>

请参阅 CBrush

CPen 类成员

Construction	
Cpen	构造一个 CPen 对象
Initialization	
CreatePen	用指定风格、宽度和画刷属性创建一个逻辑装饰画笔或几何画笔，并将它连接到 CPen 对象上
CreatePenIndirect	用在一个 LOGPEN 结构中给出的风格、宽度和颜色来创建一只画笔，并将它连接到 CPen 对象上
Operations	
FromHandle	当给予一个 Windows HPEN 时，返回一个指向 CPen 对象的指针
Attributes	
Operator HPEN	返回连接到 CPen 对象上的 Windows 句柄
Attributes	
GetLogPen	获取一个 LOGPEN 基础结构
GetExtLogPen	获取一个 EXTLOGPEN 基础结构

成员函数

CPen::CPen

```
CPen();
CPen( int nPenStyle, int nWidth, COLORREF crColor );
   throw( CResourceException );
CPen( int nPenStyle, int nWidth , const LOGBRUSH* pLogBrush, int nStyleCount =
0,
          const DWORD* lpStyle = NULL );
          throw( CResourceException );
```

参数

nPenStyle

指定画笔的风格。在构造函数的第一个版本中，此参数可以取下列值之一：

- PS_SOLID　创建一支实线画笔。
- PS_DASH　创建一支虚线画笔。只有当画笔宽度为 1 或更小（以设备单位计算）时才有效。
- PS_DOT　创建一支点线画笔。只有当画笔宽度为 1 或更小（以设备单位计算）时才有效。
- PS_DASHDOT　创建一支虚线和点交替的画笔。只有当画笔宽度为 1 或更小（以设备单位计算）时才有效。
- PS_DASHDOTDOT　创建一支虚线和两点交替的画笔。只有当画笔宽度为 1 或更小（以设备单位计算）时才有效。
- PS_NULL　创建一支空画笔。
- PS_INSIDEFRAME　创建一支画笔，该画笔在 Windows GDI 输出函数所产生的封闭形状的框架内画线，此输出函数指定一个限定矩形（例

如，Ellipse，Rectangle，RoundRect，Pie 和 Chord 成员函数），当此风格用于没有指定限定矩形的 Windows GDI 输出函数（例如 LineTo 成员函数）时，此画笔的绘制区域不受框架的限制。

第二种版本的 CPen 构造器指明了类型、风格、尾帽和连接等属性组合。来自每个类别的值应使用位操作符（1）组合起来，笔的风格可以是下列之一：

- PS_GEOMETRIC 创建一支几何画笔。
- PS_COSMETIC 创建一支装饰画笔。

CPen 构造函数的第二个版本为 nPenStyle 添加了下列一些画笔风格：

- PS_ALTERNATE 创建一支交替设置像素的画笔（此风格只用于装饰画笔）。
- PS_USERSTYLE 创建一支使用用户提供的风格数组的画笔。

尾帽可以是下列值之一：

- PS_ENDCAP_ROUND　尾帽是圆的。
- PS_ENDCAP_SQUARE　尾帽是方的。
- PS_ENDCAP_FLAT　尾帽是平面的。

连接可以是下列值之一：

- PS_JOIN_BEVEL　连接是斜截式的。
- PS_JOIN_MITER　当连接在::SetMiterLimit 函数所设置的当前限制之内时，连接是斜接式的。如果连接超出这个限制则成为斜截式的。
- PS_JOIN_ROUND　连接是圆的。

nWidth

指定画笔的宽度。

- 对于构造函数的第一个版本来说，如果这个值是 0，则不管是什么映

射方式，以设备单位计算的宽度总是一个像素。

- 对于构造函数的第二个版本来说，如果 *nPenStyle* 是 PS_GEOMETRIC，则宽度以逻辑单位给出。如果 *nPenStyle* 是 PS_COSMETIC，则宽度必须设置为 1。

crColor

包含了画笔的 RGB 值。

pLogBrush

指向一个 LOGBRUSH 结构。如果 *nPenStyle* 是 PS_COSMETIC，则此 LOGBRUSH 结构的 *lbColor* 成员指定画笔的颜色，此 LOGBRUSH 结构的 *lbStyle* 成员必须设置为 BS_SOLID。如果 *nPenStyle* 为 PS_GEOMETRIC，则此结构的所有成员都必须用于指定画笔画刷属性。

nStyleCount

指定 *lpStyle* 数组的长度（以双字为单位）。如果 *nPenStyle* 不是

PS_USERSTYLE，这个值必须是零。

lpStyle

指向一个双字值的数组。第一个值指定一个用户定义的风格中第一段虚线的长度，第二个值指定第一段空白的长度，如此等等。如果 nPenStyle 不是 PS_USERSTYLE，则这个指针必须为 NULL。

说明

如果你使用的是没有参数的构造函数，你就必须用 CreatePen，CreatePenIndirect 或 CreateStockObject 成员函数来初始化所生成的 CPen 对象。如果你使用的是带参数的构造函数，则不再需要进一步的初始化。如果有错误发生，则带参数的构造函数可能会抛出一个异常，但是不带参数的构造函数则总是会成功。

请参阅 CPen::CreatePen，CPen::CreatePenIndirect，CGdiObject::CreateStockObject

CPen::CreatePen

BOOL CreatePen(int *nPenStyle*, int *nWidth*, COLORREF *cfColor*);
BOOL CreatePen(int *nPenStyle*, intz, const LOGBRUSH* *pLogBrush*,
int *nStyleCount* = 0, const DWORD* *lpStyle* = NULL);

返回值

如果成功则返回非零值或逻辑画笔的句柄；否则返回 0。

参数

nPenStyle

指定画笔的风格。其可能取值的列表，请参见 CPen **构造函数中的** *nPenStyle* **参数。**

nWidth

指定画笔的宽度。

- 对于 CreatePen 的第一个版本来说，如果这个值为 0，则不管是什么映射模式，以设备单位表示的宽度总是一个像素。
- 对于 CreatePen 的第二个版本，如果 nPenStyle 是 PS_GEOMETRIC，则宽度以逻辑单位给出。如果 *nPenStyle* 是 PS_COSMETIC，则宽度必须设置为 1。

crColor

包含画笔的一个 RGB 颜色。

pLogBrush

指向一个 LOGBRUSH 结构。如果 *nPenStyle* 是 PS_COSMETIC，则此 LOGBRUSH 结构的 lbColor 成员指定画笔的颜色，此 LOGBRUSH 结构的 lbStyle 成员必须设置为 BS_SOLED。如果 nPenStyle 为 PS_GEOMETRIC，则此结构的所有成员都必须用于指定画笔画刷属性。

nStyleCount

指定 *lpStyle* 数组的长度（以双字为单位）。如果 *nPenStyle* 不是 PS_USERSTYLE，这个值必须是零。

lpStyle

指向一个双字值的数组。第一个值指定一个用户定义的风格中第一段虚线的长度，第二个值指定第一段空白的长度，如此等等。如果 *nPenStyle* 不是 PS_USERSTYLE，则这个指针必须为 NULL。

说明

CreatePen 的第一个版本用指定的风格、宽度和颜色来初始化一支画笔。此画笔以后可选作为任何设备环境的当前画笔。

宽度大于 1 个像素的画笔总是具有 PS_NULL，PS_SOLED 或 PS_INSIDEFRAME 风格。

如果画笔具有 PS_INSIDEFRAME 风格和某一不匹配逻辑调色板中颜色的颜色，则此画笔用一个混合色来进行绘制。PS_SOLID 画笔风格不能用一个混合色来创建一支画笔。当画笔的宽度小于或等于 1 时，PS_INSIDEFRAME 风格等同于 PS_SOLID 风格。

CreatePen 的第二个版本初始化一支具有指定的风格、宽度和刷子属性的逻辑装饰画笔或几何画笔。一支装饰画笔的宽度总是 1；几何画笔的宽度总是由世界单位指定。当应用程序创建了一支逻辑画笔之后，它可通过调用 CDC::SelectObject 函数来将此画笔选入一个设备环境中。当一支画笔被选入一个设备环境之后，它就可以用于画直线或曲线。

- 如果 *nPenStyle* 为 PS_COSMETIC 和 PS_USERSTYLE，则 *lpStyle* 数组中的项指定风格单位中虚线和空白的长度。一个风格单位由使用此画笔画线的设备定义。

- 如果 *nPenStyle* 为 PS_GEOMETRIC 和 PS_USERSTYLE，则 *lpStyle* 数组中的项指定虚线和空白的长度（用逻辑单位表示）。
- 如果 *nPenStyle* 为 PS_ALTERNATE，则忽略风格单位而交替设置像素。

当一个应用程序不再需要一支给定的画笔时,它将调用 CGdiObject::DeleteObject 成员函数或销毁 CPen 对象，以使资源不再使用。当一支画笔被选入一个设备环境后，应用程序就不能删除这支画笔了。

请参阅 CPen::CreatePenIndirect，CPen::CPen，CGdiObject::DeleteObject，LOGBRUSH

CPen::CreatePenIndirect

BOOL CreatePenIndirect(LPLOGPEN *lpLogPen*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpLogPen

指向 Windows LOGPEN **结构，此结构包含了此画笔的信息。**

说明

此函数初始化一支画笔，此画笔具有 *lpLogPen* **所指向的结构中给定的风格、宽度和颜色。**

宽度大于 1 **个像素的画笔总是具有** PS_NULL，PS_SOLID **或** PS_INSIDEFRAME **风格。**

如果画笔具有 PS_INSIDEFRAME 风格和某种不匹配逻辑调色板中颜色的颜色，则画笔用一支混合色来绘制。如果画笔宽度小于或等于 1，则 PS_INSIDEFRAME 风格等同于 PS_SOLID 风格。

请参阅 CPen::CreatePen，CPen::CPen

CPen::FormHandle

static CPen* PASCAL FromHandle(HPEN *hPen*)；

返回值

如果成功则返回一个指向 CPen 对象的指针；否则返回 NULL。

参数

hPen

Windows GDI 画笔的 HPEN 句柄。

说明

当给予一个 Windows GDI 画笔对象时，此函数返回指向一个 CPen 对象的指针。如果此句柄没有连接一个 CPen 对象，则将创建一个临时 CPen 对象，并将此 CPen 对象连接到此 HPEN 句柄上。这个临时 CPen 对象只能有效到下次应用程序的事件循环有空闲时间时，到那时所有临时图形对象都将被删除。也就是说，临时对象仅在处理一条 Windows 消息的期间有效。

CPen::GetExtLogPen

int GetExtLogPen(EXTLOGPEN* *pLogPen*);

返回值

如果成功则返回非零值；否则返回 0。

参数

pLogPen

指向一个 EXTLOGPEN **结构，此结构包含此画笔的消息。**

说明

此函数用来获取一个 EXTLOGPEN **基础结构。该** EXTLOGPEN **结构定义了画笔的风格、宽度和刷子属性。例如，调用** GetExtLogPen **来匹配一支画笔的特别的风格。**

参见“ Win32 SDK **程序员参考”中的下列主题，可以获取有关画笔属性的信息：**

- GetObject
- EXTLOGPEN
- LOGPEN
- ExtCreatePen

示例

下面的例子代码演示了调用 GetExtLogPen **来获取一支画笔的属性，然后创建一支新的、具有相同颜色的装饰画笔。**

```
EXTLOGPEN extlogpen;
penExisting.GetExtLogPen( &extlogpen );
CPen penOther;
LOGBRUSH LogBrush = { extlogpen.elpBrushStyle,extlogpen.elpColor,
extlogpen.elpHatch };
penOther.CreatePen( PS_COSMETIC, 1, &LogBrush );
```

请参阅 CPen::GetLogPen，CPen::CreatePen

CPen::GetLogPen

int GetLogPen(LOGPEN* *pLogPen*);

返回值

如果成功则返回非零值；否则返回 0。

参数

pLogPen

指向一个包含画笔信息的 LOGPEN **结构。**

说明

此函数用来获取一个 LOGPEN **基础结构。该** LOGPEN **结构定义了画笔的风格、宽度和画笔模式。**

例如，调用 GetLogPen 来匹配一支画笔的特别风格。

参见“Win32 SDK 程序员参考”中的下列主题，可以获取有关画笔属性的信息：

- GetObject
- LOGPEN

示例

下面的例子代码演示了调用 GetLogPen 来获取一支画笔的特征，然后创建一支新的、具有相同颜色的装饰画笔。

```
LOGPEN logpen;
penExisting.GetLogPen( &logpen );
CPen penOther( PS_SOLID, 0, logpen.lopnColor );
```

请参阅 CPen::GetExtLogPen

CPen::operator HPEN

operator HPEN() const;

返回值

如果成功，则返回一个 CPen **对象所代表的** Windows GDI **对象句柄；否则返回** 0。

说明

此操作符用来获取连接到 CPen **对象上的** Windows GDI **句柄。此操作符是一个强制转换操作符，它支持直接使用一个** HPEN **对象。**

要获取更多有关使用图形对象的信息，请参见“ Win32 SDK **程序员参考”中的文章“图形对象”。**

CPictureHolder

CPictureHolder 没有基类。

CPictureHolder 类的目的是 Picture 属性的实现，这使得用户可以在其控件中显示一幅图画。通过固有的 Picture 属性，开发者可以指定用于显示的位图、图标或图元文件。

#include <afxctl.h>

请参阅 CFontHolder

CPictureHolder 类成员

DataMembers	
m_pPict	一个指向图画对象的指针

Construction	
CPictureHolder	构造一个 CPictureHolder 对象

Operations	
GetDisplayString	获取一个显示在控件容器的属性浏览器中的字符串
CreateEmpty	创建一个空的 CPictureHolder 对象
CreateFromBitmap	从位图创建一个 CPictureHolder 对象
CreateFromMetafile	从图元文件创建一个 CPictureHolder 对象
CreateFromIcon	从图标创建一个 CPictureHolder 对象
GetPictureDispatch	返回 CpictureHolder 对象的 IDispatch 接口
SetPictureDispatch	设置 CpictureHolder 对象的 IDispatch 接口
GetType	表明 CpictureHolder 对象是位图、图元文件还是图标
Render	画出此图画

成员函数

CPictureHolder:: CPictureHolder

CPictureHolder();

说明

此函数用来构造一个 CPictureHolder **对象。**

请参阅 CPictureHolder::CreateEmpty

CPictureHolder::CreateEmpty

BOOL CreateEmpty();

返回值

如果成功地创建了对象，则返回非零值；否则返回 0。

说明

此函数创建一个空的 CPictureHolder **对象，并将它与一个** IPicture **接口连接。**

请参阅 CPictureHolder::CreateFromBitmap，CPictureHolder::CreateFromIcon，CPictureHolder::CreateFromeMetafile

CPictureHolder::CreateFromBitmap

```
BOOL CreateFromBitmap( UINT idResource );
BOOL CreateFromBitmap( CBitmap* pBitmap, CPalette* pPal = NULL,
   BOOL bTransferOwnership = TRUE );
BOOL CreateFromBitmap( HBITMAP hbm, HPALETTE hpal = NULL );
```

返回值

如果成功地创建了对象则返回非零值；否则返回 0。

参数

idResource

一个位图资源的资源 ID。

pBitmap

指向一个 CBitmap **对象的指针。**

pPal

指向一个 CPalette **对象的指针。**

bTransferOwnership

表明图画对象是否会获取位图和调色板对象的所有权。

hbm

位图句柄，CPictureHolder对象是由此创建的。

hpal

调色板句柄，该调色板用来绘制位图。

说明

此函数用一个位图来初始化一个 CPictureHolder 中的图画对象。如果 bTransferOwnership 是 TRUE，则在这次调用返回之前，调用者不应该以任何方式来使用这个位图或调色板。如果 *bTransferOwnership* 是 FALSE，则在图画对象的生命周期内由调用者来负责保证位图和调色板保持有效。

请参阅 CPictureHolder::CreateEmpty，CPictureHolder::CreateFromIcon，CPictureHolder::CreateFromMetafile

CPictureHolder::CreateFromIcon

BOOL CreateFromIcon(UINT *idResource*);
BOOL CreateFromIcon(HICON *hIcon*,BOOL *bTransferOwnership* = FALSE);

返回值

如果创建对象成功则返回非零值；否则返回 0。

参数

idResource

一个位图资源的资源 ID。

hIcon

图标句柄，CPictureHolder 对象由此创建。

bTransferOwnership

表明这个图画对象是否会获得图标对象的所有权。

说明

此函数用一个图标来初始化一个 CPictureHolder 中的图画对象。如果 *bTransferOwnership* 是 TRUE，则在这次调用返回之前，调用者不应该以任何方式来使用这个图标。如果 *bTransferOwnership* 是 FALSE，则在图画对象的生命周期内由调用者来负责保证图标保持有效。

请参阅 CPictureHolder::CreateEmpty，CPictureHolder::CreateFromBitmap，CPictureHolder::CreateFromeMetafile

CPictureHolder::CreateFromMetafile

```
BOOL CreateFromMetafile( HMETAFILE hmf, int xExt, int yExt,
    BOOL bTransferOwnership = FALSE );
```

返回值

如果成功地创建了对象则返回非零值；否则返回 0。

参数

hmf

图元文件的句柄，此图元文件用来创建 CPictureHolder **对象。**

xExt

图画的 X **范围。**

yExt

图画的 Y **范围。**

bTransferOwnership

表明这个图画对象是否会获得图标对象的所有权。

说明

此函数用一个图元文件来初始化一个 CPictureHolder **中的图画对象。如果** *bTransferOwnership* **是** TRUE**，则在这次调用返回之前，调用者不应该以任何方式来使用这个图元文件。如果** *bTransferOwnership* **是** FALSE**，则在图画对象的生命周期内由调用者来负责保证这个图元文件保持有效。**

请参阅 CPictureHolder::CreateEmpty，CPictureHolder::CreateFromBitmap，CPictureHolder::CreateFromIcon

CPictureHolder::GetDisplayString

BOOL GetDisplayString(CString& *strValue*);

返回值

如果成功地获取了字符串则返回非零值；否则返回 0**。**

参数

strValue

用来保存显示字符串的 CString **的引用。**

说明

此函数用来获取被显示在一个容器的属性浏览器中的字符串。

CPictureHolder::GetPictureDispatch

LPPICTUREDISP GetPictureDispatch();

返回值

返回一个指向 CPictrureHolder **对象的** IPictureDisp **接口的指针。**

说明

此函数返回一个指向 CPictrureHolder **对象的** IPictureDisp **接口的指针。当使用完这个指针后，用户必须对它调用** Release。

请参阅 CPictureHolder::SetPictureDispatch

CPictureHolder::GetType

short GetType();

返回值

是一个表明图画类型的值。它可能的取值及含义如下所示：

Value	Meaning
PICTYPE_UNINITIALIZED	CpictureHolder **对象没有被初始化**

续表

PICTYPE_NONE	CpictureHolder 对象是空的
PICTYPE_BITMAP	图画是一个位图
PICTYPE_METAFILE	图画是一个图元文件
PICTYPE_ICON	图画是一个图标

说明

此函数用来表明图画是位图、图元文件还是一个图标。

CPictureHolder::Render

void Render(CDC* *pDC*, const CRect& *rcRender*, const CRect& *rcWBounds*);

参数

pDC

指向一个显示环境的指针，图画被显示在这个环境中。

rcRender

一个矩形，图画被显示在这个矩形中。

rcWBounds

一个矩形，它代表显示该图画的对象的边界矩形。对一个控件来说，这个矩形是被传递给一个重载的 COleControl::OnDraw 函数的 rcBounds 参数。

说明

此函数将图画显示在由 *rcRender* 引用的矩形中。

CPictureHolder::SetPictureDispatch

void SetPictureDispatch(LPPICTUREDISP *pDisp*);

参数

pDisp

指向新的 IPictureDisp 接口的指针。

说明

此函数将 CPictureHolder 对象连接到 IPictureDisp 接口。

请参阅 CPictureHolder::GetPictureDispatch

数据成员

CPictureHolder::m_pPict

说明

此数据成员是一个指向 CPictureHolder 对象的 IPicture 接口的指针。

CPoint

CPoint类与Windows POINT结构类似。它还包括用来操纵CPoint和POINT结构的成员函数。

只要POINT结构可以使用的地方，CPoint对象也可以使用。这个类与“大小”有关的操作符可以接受CSize对象或SIZE结构，因为这两者是可以互换的。

> **注意** 这个类派生于tagPOINT结构（名字tagPOINT是POINT结构的不太常用的有关名字）。这意味着POINT结构的数据成员，x和y，也是CPoint的可以访问的数据成员。

```
#include <afxwin.h>
```

请参阅 CRect，CSize

CPoint 类成员

Construction	
Cpoint	构造一个 CPoint

Operations	
Offset	给 CPoint 的 x 和 y 成员增加值
Operator ==	检查两个点是否相等
Operator !=	检查两个点是否不等

Operators Returning CPoint Values	
Operator +=	通过增加一个尺寸或点来使 CPoint 偏移
Operator -=	通过减去一个尺寸或点来使 CPoint 偏移
Operator +	返回一个 CPoint 和一个尺寸或点的和
Operator –	返回一个 CPoint 和一个尺寸或点的偏差，或一个点的不存在

Operators Returning CSize Values	
Operator –	返回两点之间的大小差别

Operators Returning CRect Values	
Operator +	返回偏移一个大小的 CRect
Operator –	返回偏移一个负大小的 Crect

成员函数

CPoint:: CPoint

```
CPoint();
CPoint( int initX, int initY );
CPoint( POINT initPt );
CPoint( SIZE initSize );
CPoint( DWORD dwPoint );
```

参数

initX

指定 CPoint 的成员 x 的值。

initY

指定 CPoint 的成员 y 的值。

initPt

用于初始化此 CPoint 的一个 POINT 结构或 CPoint 对象。

initSize

用于初始化 CPoint 值的 SIZE 结构或 CSize。

dwPoint

设此参数的低位字为 x 成员，高位字为 y 成员。

说明

构造一个 CPoint 对象。如果不给参数，则不初始化 x 和 y 成员。

CPoint::Offset

void Offset(int *xOffset*, int *yOffset*)
void Offset(POINT *point*);
void Offset (SIZE *size*);

参数

xOffset

指定 CPoint **的** x **成员的偏移量。**

yOffset

指定 CPoint **的** y **成员的偏移量。**

point

指定偏移此 CPoint **的数量（**POINT **或** CPoint**）。**

size

指定偏移此 CPoint **的数量（**SIZE **或** CSize**）。**

说明

此函数将偏移值增加到 CPoint **的** x **和** y **成员。**

请参阅 CPoint::operator +=，CPoint::operator -=

操作符

CPoint::operator ==

BOOL operator ==(POINT *point*) const;

返回值

如果两点相等则返回非零值；否则返回 0。

参数

point

包含一个 POINT **结构或** CPoint **对象。**

说明

此函数用来检查两个点是否相等。

请参阅 CPoint::Operator !=

CPoint::operator !=

BOOL operator != (POINT *point*) const;

返回值

如果两点不相等则返回非零值；否则返回 0**。**

参数

point

包含一个 POINT **结构或** CPoint **对象。**

说明

此成员函数用来检查两个点是否不等。

请参阅 CPoint::Operator ==

CPoint::operator +=

void operator +=(SIZE *size*);
void operator +=(POINT *point*);

参数

size

包含一个 SIZE 结构或 CSize 对象。

point

包含一个 POINT 结构或 CPoint 对象。

说明

第一种形式给 CPoint 增加一个大小。

第二种形式给 CPoint 增加一个点。

在这两种格式中，都是将右边操作数的 x（或 cx）成员加到 CPoint 的 x 成员，将右边操作数的 y（或 cy）成员加到 CPoint 的 y 成员。

例如，将 CPoint(5, -7)加到一个包含 CPoint(30,40)的变量上，则变量的值变为 CPoint(35，33)。

请参阅 CPoint::operator -=，CPoint::operator +，CPoint::Offset

CPoint::operator -=

```
void operator -=( SIZE size );
void operator -=( POINT point);
```

参数

size

包含一个 SIZE 结构或 CSize 对象。

point

包含一个 POINT 结构或 CPoint 对象。

说明

第一种形式从 CPoint 减去一个大小。

第二种形式给 CPoint 减去一个点。

在这两种格式中，都是从 CPoint 的 x 成员中减去右边操作数的 x（或 cx）成员，

从 CPoint 的 y 成员中减去右边操作数的 y（或 cy）成员。

例如，将 CPoint(5, -7)从一个包含 CPoint(30,40)的变量中减去，则变量的值变为 CPoint(25, 47)。

请参阅 CPoint::operator -，CPoint::operator +=，CPoint::Offset

CPoint::operator +

```
CPoint operator +( SIZE size ) const;
CPoint operator +( POINT point ) const;
CRect operator +( const RECT* lpRect ) const;
```

返回值

返回一个偏移一个大小的 CPoint，一个偏移一个点的 CPoint，或者是一个偏移一个点的 CRect。

参数

size

包含一个 SIZE **结构或** CSize **对象。**

point

包含一个 POINT **结构或** CPoint **对象。**

lpRect

包含一个指向 RECT **结构或** CRect **对象的指针。**

说明

此操作符用来将一个 CPoint **偏移一个** CPoint **或一个** CSize **对象，或者是将一个** CRect **对象偏移一个** CPoint**。**

例如，使用前两种格式可以做到，将点 CPoint(25,-19)**偏移一个点** CPoint(15,5) **或偏移一个大小** CSize(15,5)**，其结果都是返回** CPoint(40,-14);

将一个矩形与一个点相加，返回的是偏移了点中所指定的 x 和 y 值之后的矩形。例如，使用最后一种格式可以将矩形 CRect(125,219,325,419)偏移一个点 CPoint(25,-19)，其结果是返回一个矩形 CRect(150,200,350,400)。

请参阅 CPoint::operator -= ， CPoint::operator- ， CPoint::operator += ， CSize::operator +，

CRect::operator +， CPoint::Offset，CRect::OffsetRect

CPoint::operator -

CSize operator – (POINT *point*) const;
CPoint operator – (SIZE *size*) const;
CRect operator – (const RECT* *lpRect*) const;
CPoint operator – () const;

返回值

返回表示两点之间偏差的 CSize，或是加上了负的 size 偏移的 CPoint，或是加上了负的 point 偏移的 CRect，或是 CPoint 等于负的 point。

参数

size

包含一个 SIZE 结构或 CSize 对象。

point

包含一个 POINT 结构或 CPoint 对象。

lpRect

包含一个指向 RECT 结构或 CRect 对象的指针。

说明

使用前两种格式来从 CPoint 中减去一个 CPoint 或 CSize。第三种格式在 Rect 上加上负的 CPoint 所表示的偏移。最后一种格式使用一元操作符来对 CPoint 取负值。

例如，用第一种格式来查找两个点 CPoint(25,-19)和 CPoint(15,5)之间的差异，则返回的是 CSize(10,-24)。

从 CPoint 中减去 CSize，所进行的计算与上面是一样的，但是返回的是一个 CPoint 对象，而不是一个 CSize 对象。例如，用第二种格式来查找点 CPoint(25,-19)和大小 CSize(15,5)之间的差异，则返回的是 CPoint(10,-24)。

从一个点中减去一个矩形，返回的是偏移了点中所指定的 x 和 y 的负值之后的矩形。例如，用最后一种格式来将矩形 CRect(125,200,325,400)偏移一个点

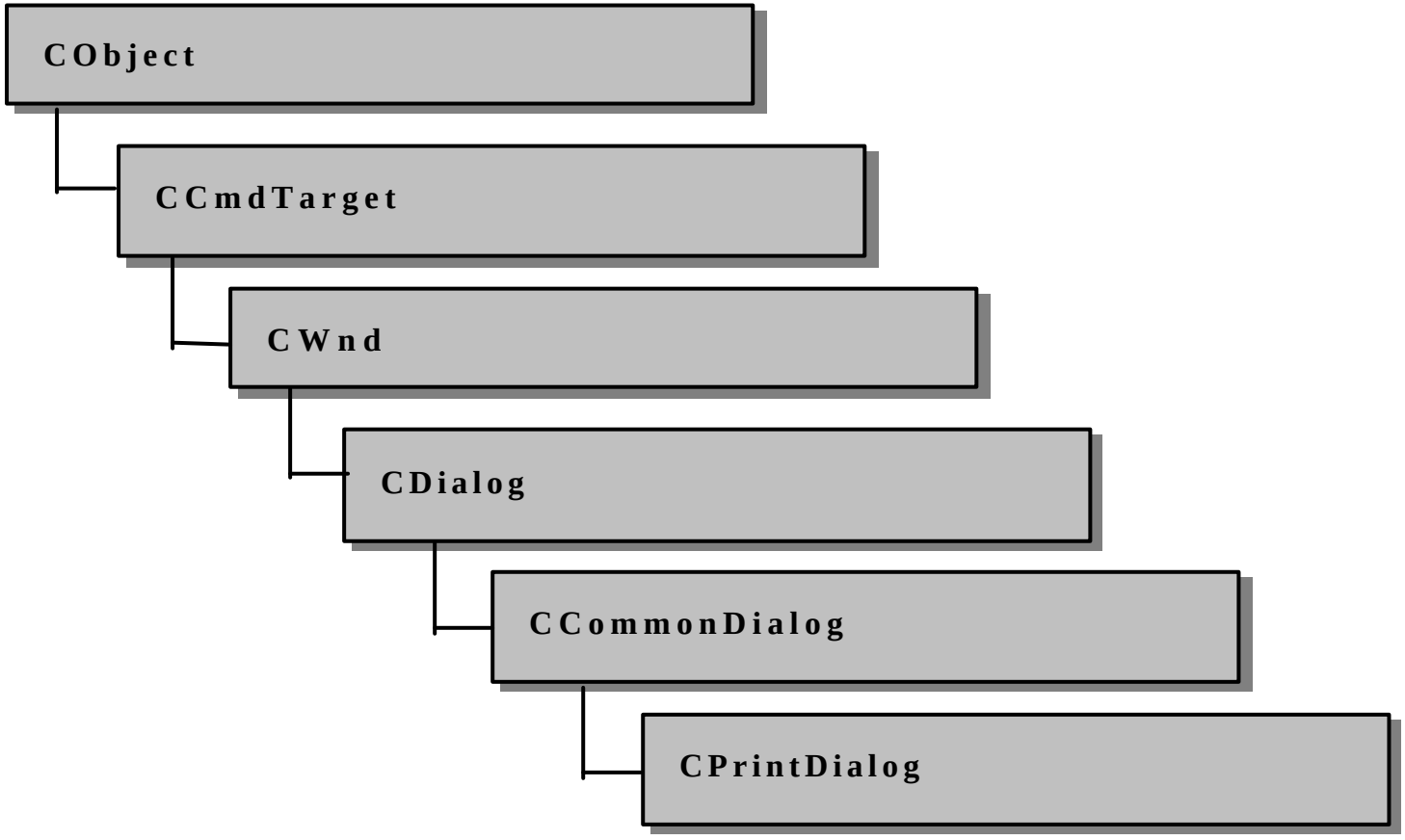
CPoint(25,一 19)，则返回的是矩形 CRect(100,219,300,419)。

使用一个一元操作符来取一个点的负值。例如，对点 CPoint(25,-19)使用一元操作符，则返回的是 CPoint(-25,19)。

请参阅 CPoint::operator -= ， CPoint::operator += ， CPoint::operator + ，CSize::operator +，

CRect::operator +，CPoint::Offset，CRect::OffsetRect

CPrintDialog



CPrintDialog 类封装了 Windows 通用打印对话框为打印提供的服务。通用打印对话框以一种与 Windows 标准一致的方式提供了一种实现 Print 和 Print Setup 对话框的简单方法。

如果你愿意，可以依靠框架来为应用程序处理许多打印处理方面的事情。在这种情况下，框架自动显示打印用的 Windows 通用对话框。也可以让框架为应用程序处理打印操作，但用自己的打印对话框来加载通用的 Print 对话框。

如果你希望你的应用程序在没有框架干涉的情况下来处理打印，则可用所提供的构造函数使用“原样”CPrintDialog 类，或者从 CPrintDialog 派生自己的对话框类，并编制构造函数以满足自己的需要。在这两种情况下，由于这些对话框都是从类 CCommondDialog 类派生而来的，因此它们的行为将都类似于标准 MFC 对话框。

为了使用 CPrintDialog 对象，首先要用 CPrintDialog 构造函数来创建此对象。

一旦已经构造了对话框，你可以设置或修改 m_pd 结构中的任何值来初始化对话框的控件的值。m_pd 结构是 PRINTDLG 类型的。要获取更多有关这个结构的信息，请参见 Win32 SDK 文档。

如果你在 m_pd 中没有为 hDevMode 和 hDevName 成员提供自己的句柄，则要确保在应用程序使用完此对话框时，调用 Windows 函数 GlobalFree 释放这些句柄。当使用由 CWinApp::OnFilePrintSetup 函数提供的框架的 Print Setup 实现时，应用程序不必释放这些句柄。这些句柄由 CWinApp 维护，并在 CWinApp 的析构函数中释放。唯一必要的是在独立使用 CPrintDialog 时，释放这些句柄。

在初始化对话框之后，调用 DoModal 成员函数来显示这个对话框，并允许用户选择打印选项。DoModal 返回用户选择的是 OK（IDOK）还是 Cancel（IDCANCEL）按钮。

如果 DoModal 返回 IDOK，则应用程序可以用一个 CPrintDialog 成员函数来获

取用户输入的信息。

要在不显示一个对话框的情况下获取当前打印机的缺省状态，CPrintDialog::GetDefaults 成员函数是一个有用的函数。这个成员函数不需要用户的交互作用。

你可以调用 Windows 的 CommDlgExtendedError 函数来确定在对话框的初始化过程中是否发生了错误，并获得有关错误的进一步消息。要获取有关这个函数的更多信息，请参见 Win32 SDK 文档。

CPrintDialog 依赖由 Windows 3.1 及其更高版本引进的 COMMDLG.DLL 文档。

要定制此对话框，先从 CPrintDialog 派生一个新类，提供一个定制对话框模板，并增加一个处理来自扩展控件的通知消息的消息映射。任何未处理消息将传递给基类处理。不需要定制 hook 函数。

要想根据对话框是 Print 还是 Print Setup 来不同地处理同一条消息，应用程序必须为每个对话框派生一个新类。还必须重载 Windows 函数 AttachOnSetup，它负责处理当在 Print 对话框内选择 Print Setup 按钮时新对话框的创建。

如果要获取有关使用 CPrintDialog 的更多信息，请参见“ Visual C++程序员指南”中的文章“通用对话框类”。

#include <afxdlgs.h>

请参阅 CPrintInfo

CPrintDialog 类成员

Data Members	
m_pd	用来定制 CPrintDialog 对话框的结构

Construction	
CPrintDialog	构造一个 CPrintDialog 对话框

Operations	
CreatePrinterDC	创建一个打印机设备环境，但不显示 Print 对话框
DoModal	显示对话框，并允许用户进行选择
GetCopies	获取所请求的备份数目
GetDefaults	检取设备缺省项，但不显示对话框
GetDeviceName	检取当前所选打印设备的名字
GetDevMode	检取 DEVMODE 结构
GetDriverName	检取当前所选打印机驱动器的名字
GetFormPage	检取打印范围的起始页
GetToPage	检取打印范围的结束页
GetPortName	检取当前所选打印机端口名
GetPrinterDC	检取设备环境的句柄
PrintAll	确定是否请求打印文档的所有页
PrintCollate	确定是否请求校对备份
PrintRange	确定是否只打印指定范围内的页
PrintSelection	确定是否只打印当前所选项

成员函数

CPrintDialog:: CPrintDialog

CPrintDialog(BOOL *bPrintSetupOnly*, DWORD *dwFlags* = PD_ALLPAGES | PD_USEDEVMODECOPIES | PD_NOPAGENUMS | PD_HIDEPRINTTOFILE | PD_NOSELECTION, CWnd* *pParentWnd* = NULL);

参数

bPrintSetupOnly

指定是显示标准的 Windows Print **对话框还是** Print Setup **对话框。设置该参数为** TRUE**，则显示标准** Windows Print Setup **对话框。设置该参数为** FALSE**，则显示** Windows Print **对话框。如果** *bPrintSetupOnly* **为** FALSE**，则** Print Setup **选项按钮仍显示在** Print **对话框中。**

dwFlags

一个或多个标志，可用于定制对话框的设置，这些标志可以用位或 OR 操作符来组合。例如，PD_ALLPAGES 标志将缺省打印范围设置为文档的所有页。有关这些标志的详细信息，请参见 Windows SDK 中的 PRINTDLG 结构。

pParentWnd

指向对话框的父或属主窗口的指针。

说明

此函数用来构造一个 Windows Print 或 Print Setup 对话框。这个成员函数只构造此对象。利用 DoModal 成员函数可显示此对话框。

值得注意的是，当你将 bPrintSetupOnly 设置为 FALSE 来调用构造函数时，会自动使用 PD_RETURNDC 标志。在调用 DoModal，GetDefaults 或 GetPrinterDC

后，一个打印机 DC 将会被放在 m_pd.hdc 中返回。这个 DC 必须由 CPrintDialog 的调用者来释放。

请参阅　CPrintDialog::DoModal，::PrintDlg

CPrintDialog::CreatePrinterDC

HDC CreatePrinterDC();

返回值

返回一个新创建的打印机设备环境的句柄。

说明

此函数从 DEVMODE 和 DEVNAMES 结构创建一个打印机设备环境（DC）。这个 DC 被假定为当前打印机 DC，其它任何先前获得的打印机 DC 都必须由

用户删除。不必显示 Print 对话框就能调用此函数，并使用创建得到的 DC。

请参阅　CPrintDialog::GetDevModel

CPrintDialog::DoModal

virtual int DoModal();

返回值

返回 IDOK 或 IDCANCEL。如果返回的是 IDCANCEL，则要调用 Windows CommDlgExtendedError 函数来确定是否发生了一个错误。IDOK 和 IDCANCEL 都是常量，它表明用户选择的是 OK 按钮还是 Cancel 按钮。

说明

此函数用来显示 Windows 的通用打印对话框，并允许用户选择各个打印选项，

例如备份的数目、页范围和备份是否需要整理。

如果你想要提供设置 m_pd 结构的成员来初始化各个打印对话框选项，则应当在打印 DoModal 之前，但在对话框对象构造之后进行。

在调用了 DoModal 之后，你就可以调用其它的成员函数来检取各个设置或用户在对话框中输入的信息了。

请参阅 CPrintDialog:: CPrintDialog，CDialog::DoModal

CPrintDialog::GetCopies

int GetCopies() const;

返回值

返回所请求的备份数目。

说明

在调用 DoModal 之后调用此函数来获取所请求的备份数目。

请参阅　CPrintDialog::PrintCollate

CPrintDialog::GetDefaults

BOOL GetDefaults();

返回值

如果函数成功则返回非零值；否则返回 0。

说明

此函数用来检取缺省打印机的设备缺省值，不需要显示对话框。所检取的值放在 m_pd 结构中。

在某些情况下，调用这个函数会导致调用 CPrintDialog 的构造函数，并将 bPrintSetupOnly 设置为 FALSE。在这些情况下，一个打印机 DC 和 hDevNames 和 hDevMode（m_pd 数据成员中的两个句柄）将会被自动分配。

如果 CPrintDialog 的构造函数是在 bPrintSetupOnly 设置为 FALSE 的情况下被调用的，则此函数将不仅仅是把 hDevNames 和 hDevMode（在 m_pd.hDevNames 和 m_pd.hDevMode 中）返回给调用者，而是还要返回一个在 m_pd.hDC 中的打印机 DC。由调用者负责删除这个打印机 DC，并在完成了 CPrintDialog 对象时以此句柄调用 Windows GlobalFree 函数。

请参阅　CPrintDialog::m_pd

CPrintDialog::GetDeviceName

```
CString GetDeviceName() const;
```

返回值

返回当前所选打印机的名字。

说明

在调用 DoModal 之后调用此函数来检取当前所选打印机的名字。

请参阅 CPrintDialog::GetDriverName，CPrintDialog::GetDevMode，
CPrintDialog::GetPortName

CPrintDialog::GetDevMode

LPDEVMODE GetDevMode() const;

返回值

返回 DEVMODE 数据结构，该结构包含了有关设备初始化和打印机驱动器环

境的信息。你必须调用 Windows GlobalUnlock 函数来解锁这个结构所占的内存，在“Platform SDK”中有关于这个函数的描述。

说明

在调用 DoModal 之后调用此函数来检取有关打印机设备的信息。

请参阅 CDC::GetDeviceCaps

CPrintDialog::GetDriverName

CString GetDriverName() const;

返回值

返回当前所选的打印机设备驱动器的名字。

说 明

在 调 用 DoModal **之 后 调 用 此 函 数 来 获 取 当 前 所 选 打 印 机 设 备 驱 动 器 的 名 字 。**

请 参 阅 CPrintDialog::GetDeviceName，CPrintDialog::GetDevMode，
CPrintDialog::GetPortName

CPrintDialog::GetFromPage

int GetFromPage() const;

返 回 值

返 回 所 选 打 印 范 围 内 的 首 页 号 码 。

说 明

在 调 用 DoModal **之 后 调 用 此 函 数 来 检 取 所 选 打 印 范 围 内 的 首 页 号 码 。**

请参阅 CPrintDialog::GetToPage，CPrintDialog::PrintRange

CPrintDialog::GetPortName

CString GetPortName() const;

返回值

返回当前所选打印机端口的名字。

说明

在调用 DoModal **之后调用此函数来检取当前所选打印机端口的名字。**

请参阅 CPrintDialog::GetDriverName，CPrintDialog::GetDeviceName

CPrintDialog::GetPrinterDC

HDC GetPrinterDC() const;

返回值

如果成功则返回一个打印机设备环境的句柄；否则返回 NULL。

说明

如果 CPrintDialog**构造函数的参数** *bPrintSetupOnly* **是** FALSE**（表明显示的是** Print **对话框），则** GetPrinterDC **返回一个打印机设备环境句柄。当你使用完这个设备环境时，你必须调用** Windows DeleteDC **函数来删除它。**

CPrintDialog::GetToPage

int GetToPage() const;

返回值

返回所选打印范围内的结束页号码。

说明

在调用 DoModal **之后调用此函数来检取所选打印范围内的结束页号码。**

请参阅 CPrintDialog::GetFromPage，CPrintDialog::PrintRange

CPrintDialog::PrintAll

BOOL PrintAll() const;

返回值

如果文档中的所有页都要打印则返回非零值；否则返回 0。

说明

在调用 DoModal **之后调用此函数来确定是否要文档中的所有页。**

请参阅 CPrintDialog::PrintRange，CPrintDialog::PrintSelection

CPrintDialog::PrintCollate

BOOL PrintCollate() const;

返回值

如果用户选择了对话框中的整理选择框，则返回非零值；否则返回 0。

说明

在调用 DoModal **之后调用此函数来确定打印机是否要整理所有要打印的文档备份。**

请参阅 CPrintDialog::GetCopies

CPrintDialog::PrintRange

BOOL PrintRange() const;

返回值

如果文档中只有某一个范围的页要打印，则返回非零值；否则返回 0。

说明

在调用 DoModal **之后调用此函数来确定是否只打印文档中某一范围内的页。**

请参阅 CPrintDialog::PrintAll，CPrintDialog::PrintSelection，

CPrintDialog::GetFromPage，CPrintDialog::GetToPage

CPrintDialog::PrintSelection

BOOL PrintSelection() const;

返回值

如果只要打印所选的项，则返回非零值；否则返回 0。

说明

在调用 DoModal **之后调用此函数来确定是否只要打印当前所选的项。**

请参阅 CPrintDialog::PrintRange，CPrintDialog::PrintAll

数据成员

CPrintDialog::m_pd

PRINTDLG& m_pd;

说明

一个结构，它的成员保存了对话框对象的特征。在构造一个 CPrintDialog 对象之后，在调用 DoModal 之前，你可以使用 m_pd 来设置对话框的各个方面。有关 m_pd 结构的更多信息，请参见 Win32 SDK 文档。

如果要直接修改 m_pd 数据成员，你将重载任何缺省的行为。

CPrintInfo

CPrintInfo 没有基类。

CPrintInfo 存储有关一次打印或打印预览作业的信息。每次选择 Print 或 Print Preview 命令，框架就创建一个 CPrintInfo 对象，并在命令完成时删除此对象。

CPrintInfo 包含有关打印作业的一般信息，例如要打印的页范围，以及打印作业的当前状态，例如当前正打印的页。某些信息存放在一个相关联的 CPrintInfo 对象中；此对象中包含用户在 Print 对话框中输入的值。

在打印期间，一个 CPrintInfo 对象在框架和应用程序的视类之间传递，并用于在两者之间交换信息。例如，框架通过对 CPrintInfo 和 m_nCurPage 成员赋值，来通知视类要打印文档的哪一页；视类检取此值，并执行指定页的实际打印。

另一个例子就是文档的长度直到打印时也不知道的情况。在这种情况下，视类在每打印一页时都测试是否是文档的结尾了。当到达文档结尾时，视类将 CPrintInfo 的 m_bContinuePrinting 成员设置为 FALSE，它通知框架停止打印循环。

CPrintInfo 由“请参阅”中所列的 CView 的成员函数调用。

请参阅 CView::OnBeginPrinting，CView::OnEndPrinting，CView::OnEndPrintPreview，CView::OnPrepareDC，CView::OnPreparePrinting，CView::OnPrint

CPrintInfo 类成员

Data Members	
m_bDocObject	包含一个标志，该标志表明被打印的文档是否是一个 DocObject
m_dwFlags	指定 DocObject 的打印选择
m_nOffsetPage	在一个组合 DocObject 打印作业中指定一个特别的 DocObject 的第一页的偏移
m_pPD	包含一个指针，该指针指向用于 Print 对话框的 CPrintInfo 对象
m_bDirect	包含一个标志，该标志表明是否直接打印这个文档（不显示 Print 对话框）
m_bPreview	包含一个标志，该标志表明是否预览文档
m_bContinuePrinting	包含一个标志，该标志表明框架是否要继续打印循环
m_nCurPage	表明当前打印的页码
m_NumPreviewPages	表明在预览窗口中显示的页数；1 或者 2
m_lpUserData	包含一个指针，该指针指向一个用户创建的结构
m_rectDraw	指定一个用于定义当前可用页区的矩形
m_strPageDesc	包含一个用于页码显示的格式字符串

Attributes	
SetMinPage	设置文档第一页的页码
SetMaxPage	设置文档最后一页的页码
GetMinPage	返回文档第一页的页码
GetMaxPage	返回文档最后一页的页码
GetOffsetPage	返回在一次组合 DocObject 打印作业中被打印的 DocObject 项的第一页前面的页数
GetFromPage	返回要打印的第一页的页码
GetToPage	返回要打印的最后一页的页码

成员函数

CPrintInfo::GetFromPage

UINT GetFromPage() const;

返回值

要打印的第一页的页码。

说明

此函数用来获取要打印的第一页的页码。这个值由用户在 Print **对话框中指定，存放在由** m_pPD **成员引用的** CPrintInfo **对象中。如果用户没有指定这个值，则缺省的是文档的第一页。**

请参阅 CPrintInfo::m_nCurPage，CPrintInfo::m_pPD，CPrintInfo::GetToPage

CPrintInfo::GetMaxPage

UINT GetMaxPage() const;

返回值

返回文档的最后一页的页码。

说明

此函数用来获取文档的最后一页的页码。这个值存放在由 m_pPD **成员所引用的** CPrintDialog **对象中。**

请参阅 CPrintInfo::m_nCurPage，CPrintInfo::m_pPD，CPrintInfo::GetMinPage，CPrintInfo::SetMaxPage，CPrintInfo::SetMinPage

CPrintInfo::GetMinPage

UINT GetMinPage() cost;

返回值

返回文档第一页的页码。

说明

此函数用来获取文档第一页的页码。这个值存放在由 m_pPD **所引用的** CPrintDialog **对象中。**

请参阅 CPrintInfo::m_nCurPage，CPrintInfo::m_pPD，CPrintInfo::GetMaxPage，CPrintInfo::SetMaxPage，CPrintInfo::SetMinPage

CPrintInfo::GetOffsetPage

UINT GetOffsetPage() const;

返回值

返回在一次组合 DocObject 打印作业中被打印的 DocObject 项的第一页的前面的页数。

说明

当从一个 DocObject 客户打印多个 DocObject 项时，此函数用来获取偏移。这个值由 m_nOffsetPage 成员引用。当要打印的是 DocObject 和其它活动文档时，文档的第一页将被编码为：m_nOffsetPage 值＋1。只有在 m_bDocObject 的值为 TRUE 时，m_nOffsetPage 成员才是有效的。

请参阅 CPrintInfo::m_nOffsetPage，CPrintInfo::m_bDocObject

CPrintInfo::GetToPage

UINT GetToPage() const;

返回值

返回要打印的最后一页的页码。

说明

此函数用来获取要打印的最后一页的页码。这个值是用户在 Print **对话框中指定的，并且它被保存在由** m_pPD **成员所引用的** CPrintDialog **对象中。如果用户没有指定这个值，则缺省值是文档的最后一页。**

请参阅 CPrintInfo::m_nCurPage，CPrintInfo::m_pPD，CPrintInfo::GetFromPage

CPrintInfo::SetMaxPage

void SetMaxPage(UINT *nMaxPage*);

参数

nMaxPage

文档最后一页的页码。

说明

此函数用来指定文档最后一页的页码。这个值保存在由 m_pPD **所引用的** CPrintDialog **对象中。如果文档的长度在打印之前是已知的，则从重载的** CView::OnPreparePrinting **中调用这个函数。如果文档的长度依赖于用户在** Print **对话框中的某个设置指定，则从你的重载的** CView::OnBeginPrinting **中调用这个函数。如果文档的长度直到打印时还是未知的，则使用** m_bContinuePrinting

成员来控制打印循环。

请参阅 CPrintInfo::m_bContinuePrinting，CPrintInfo::m_nCurPage，CPrintInfo::m_pPD，CPrintInfo::GetMinPage，CPrintInfo::GetToPage，CPrintInfo::SetMinPage，CView::OnBeginPrinting，CView::OnPreparePrinting

CPrintInfo::SetMinPage

void SetMinPage(UINT *nMinPage*);

参数

nMinPage

文档的第一页的页码。

说明

此成员函数用来指定文档的第一页的页码。通常页码开始于 1**。这个值被保存在由** m_pPD **成员所引用的** CPrintDialog **对象中。**

请参阅 CPrintDialog::m_nCurPage**，** CPrintDialog::m_pPD**，** CPrintDialog::GetMaxPage**，** CPrintInfo::GetMinPage**，** CPrintInfo::SetMaxPage

数据成员

CPrintInfo::m_bContinuePrinting

说明

此数据成员包含了一个标志，该标志用来表明框架是否继续进行打印循环。如果你正在进行打印时分页，则一旦已到达文件尾，即可在 CView::OnPrepareDC 的重载中设置该成员为 FALSE。如果应用程序已经在打印作业开始时用 SetMaxPage 成员函数指定了文档的长度，则不必修改这个变量。m_bContinuePrinting 成员是 BOOL 类型的公有变量。

请参阅 CPrintDialog::SetMaxPage，CView::OnPrepareDC

CPrintInfo::m_bDirect

说明

如果要省略 Print **对话框直接进行打印，则框架将此成员设置为** TRUE**；否则将其设置为** FALSE**。当应用程序从外壳进行打印，或打印是通过使用** ID_FILE_PRINT_DIRECT **命令** ID **进行时，通常** Print **对话框是被忽略的。**

通常来说，你不会改变这个成员，但如果你要改变它，则在你的 CView::OnPreparePrinting **重载中调用** CView::DoPreparePrinting **之前改变它。**

请参阅 CView::DoPreparePrinting，CView::OnPreparePrinting

CPrintInfo::m_bDocObject

说明

此成员中包含一个标志，该标志表明被打印的文档是否是一个 DocObject。除非这个标志是 TRUE，否则数据成员 m_dwFlags 和 m_nOffsetPage 是无效的。

请参阅 CPrintInfo::m_dwFlags，CPrintInfo::m_nOffsetPage

CPrintInfo::m_bPreview

说明

此成员包含一个标志，该标志表明文档是否正在预览。这个标志的值由框架根据用户执行的命令来设置。不为打印预览作业显示 Print 对话框。m_bPreview 是一个 BOOL 类型的公有变量。

请参阅 CView::DoPreparePrinting，CView::OnPreparePrinting

CPrintInfo::m_dwFlags

说明

此成员包含一些标志的组合，这些标志指定了 DocObject **打印操作。只有当成员** m_bDocObject **是** TRUE **的时候才有效。**

这些标志可以是下列值中的一个或几个：

PRINTFLAG_MAYBOTHERUSER
PRINTFLAG_PROMPTUSER
PRINTFLAG_USERMAYCHANGEPRINTER
PRINTFLAG_RECOMPOSETODEVICE
PRINTFLAG_DONTACTUALLYPRINT
PRINTFLAG_FORCEPROPERTIES
PRINTFLAG_PRINTTOFILE

请参阅 CPrintInfo::bDocObject，CPrintInfo::m_nOffsetPage

CPrintInfo::m_lpUserData

说明

此成员包含一个指针，该指针指向一个用户创建的结构。你可以用这个成员来保存那些你不想保存在你的应用程序的视类中的打印专用数据。m_lpUseData **成员是** LPVOID **类型的公有变量。**

CPrintInfo::m_nCurPage

说明

此成员包含当前页的页码。框架为文档的每一页调用 CView::OnPripareDC **函**

数，每次为这个成员指定一个不同的值；该成员的取值范围从 GetFromPage 返回的值到 GetToPage 返回的值，可以在重载的 CView::OnPrepareDC 和 CView::OnPrint 函数中使用这个成员来打印文档的指定页。

当预览方式第一次被激活时，框架读出此成员的值来确定初始时预览文档的哪一页。应用程序可在重载的 CView::OnPreparePrinting 中设置此成员的值，用来在进入预览方式时维护用户在文档中的当前位置。m_nCurPage 成员是 UINT 类型的公有变量。

请参阅 CPrintInfo::GetFromPage，CPrintInfo::GetToPage，CView::OnPrepareDC，

CView::OnPreparePrinting，CView::OnPrint

CPrintInfo::m_nNumPreviewPages

说明

此成员包含了在预览模式下显示的页数，它可以是 1 **或** 2。m_nNumPreviewPages **成员是** UINT **类型的公有变量。**

请参阅 CPrintInfo::m_strPageDesc

CPrintInfo::m_nOffsetPage

说明

此成员包含在一个组合的 DocObject **打印作业中，在一个特定的** DocObject **的第一页之前的页数。**

请参阅 CPrintInfo::m_bDocObject，CPrintInfo::m_dwFlags

CPrintInfo::m_pPD

说明

此成员中包含一个指向 CPrintDialog **对象的指针，这个** CPrintDialog **对象用于为打印作业显示** Print **对话框。**m_pPD **成员是一个声明为** CPrintDialog **指针的公有变量。**

请参阅 CPrintDialog

CPrintInfo::m_rectDraw

说明

此成员指定页的可用绘制区域的逻辑坐标。你也许想要在重载的 CView::OnPrint **中引用该成员。应用程序可以使用这个成员来记录当打印完头、尾等之后留下的可用区域。**

请参阅 CView::OnPrint

CPrintInfo::m_strPageDesc

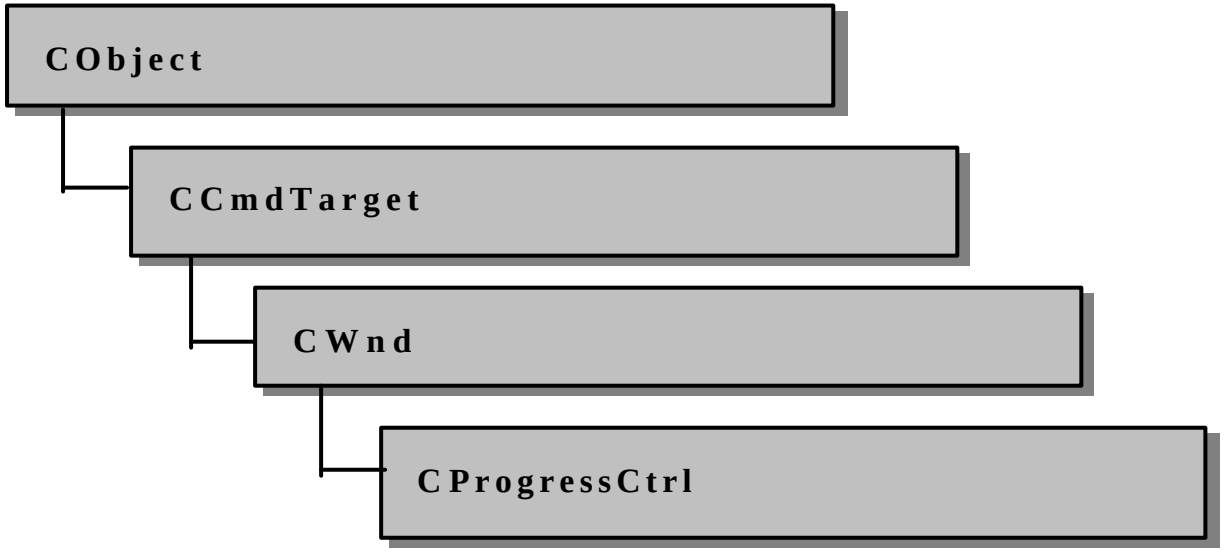
说明

此成员中包含了一个在打印预览期间用来显示页码的格式字符串；这个字符串

由两个子串组成，一个用于单页显示，一个用于双页显示，每个子串用一个“\n”字符结尾。框架用“Page %u\nPages %u-%u\n”作为缺省值。如果应用程序想要为页码指定一种不同的格式，则可在CView::OnPreparePrinting的重载中指定一个格式字符串。m_strPageDesc成员是CString类型的公有变量。

请参阅　CView::OnPreparePrinting

CProgressCtrl



“进度条控件”是一个窗口，应用程序可以使用这个窗口来表明一个冗长操作

的进度。它由一个从左到右，用系统高亮色逐渐填充的矩形组成。

CProgressCtrl 类提供了 Windows 通用进度条控件的机能。这个控件（也就是 CProgressCtrl 类）只有对运行在 Windows 95 和 Windows NT 3.51 或更高版本下的程序才是有效的。

进度条控件具有一个范围和一个当前位置。范围代表了操作的整个期间，当前位置代表了应用程序为完成操作已经做完的部分。窗口进程用范围和当前位置来确定用高亮色填充进度条的百分比，以及确定在进度条中所显示的文本（如果有的话）。由于范围和当前位置值是用有符号整数表示的，所以可能的当前位置值的范围是从 -217483648 到 217483647。

```
#include <afxcmn.h>
```

CProgressCtrl 类成员

Construction	
CprogressCtrl	构造一个 CProgressCtrl 对象
Create	创建一个进度条控件并将它与一个 CProgressCtrl 对象连接

Attributes	
SetRange	为进度条控件设置范围的最小值和最大值，并重画进度条来反映新的范围
SetRange32	为进度条控件设置范围的最小值和最大值，并重画进度条来反映新的范围
GetRange	获取进度条控件范围的下限和上限
GetPos	获取进度条的当前位置
SetPos	设置进度条的当前位置并重画进度条来反映新的位置
OffsetPos	用一个指定的增量来增加进度条控件的当前位置，并重画此进度条来反映新的位置
SetStep	为一个进度条控件指定每一步的增量

Operations	
StepIt	用每一步的增量（参见 SetStep）来增加一个进度条控件的当前位置，并重画此进度条来反映新的位置

成员函数

CProgressCtrl::CProgressCtrl

CProgressCtrl();

说明

此成员函数用来构造一个 CProgressCtrl 对象。

在构造一个 CProgressCtrl 对象后，调用 CProgressCtrl::Create 来创建进度条控件。

请参阅 CProgressCtrl::Create

CProgressCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果 CProgressCtrl **对象被成功创建则返回** TRUE**；否则返回** FALSE。

参数

dwStyle

指定进度条控件的风格。除了 Windows **风格的任何组合，附加下列进度条控件风格：**

- PBS_VERTICAL **垂直显示进度信息，从顶到底。没有这个标志，进**

度条控件从左到右水平显示。

- PBS_SMOOTH 逐渐显示，平滑地填充进度条控件。没有这个标志，控件将用块来填充。

rect

指定进度条控件的尺寸和位置。它可以是一个 CRect 对象或者是一个 RECT 结构。由于此控件必须是一个子窗口，所以指定的坐标是相对于 *pParentWnd* 的客户区的。

pParentWind

指定进度条控件的父窗口，通常是一个 CDialog。它不能是 NULL。

nID

指定进度条控件的 ID。

说明

你可以分两步来构造一个 CProgressCtrl **对象。首先调用构造函数来创建** CProgressCtrl **对象；然后调用** Create **来创建进度条控件。**

请参阅 CProgressCtrl:: CProgressCtrl

CProgressCtrl::GetPos

int GetPos();

返回值

返回进度条控件的位置。

说明

此成员函数用来检取进度条的当前位置。进度条的这个位置不是它在屏幕上的

物理位置，而是在 SetRange 中的上限和下限范围之间的位置。

请参阅 CProgressCtrl::SetPos

CProgressCtrl::GetRange

void GetRange(int & *nLower*, int& *nUpper*);

参数

nLower

一个指向整数的引用，它用来接收进度条控件的下限。

nUpper

一个指向整数的引用，它用来接收进度条控件的上限。

说明

此成员函数用来获取当前进度条控件的上限和下限，或者说是范围。此函数将

上限和下限值拷贝到由 *nLower* 和 *nUpper* 各自引用的整数中。

请参阅 PBRANGE，PBM_GETRANGE

CProgressCtrl::OffsetPos

int OffsetPos(int *nPos*);

返回值

返回进度条控件的早先的位置。

参数

nPos

用来增加位置的数量。

说明

此函数用由 *nPos* **指定的增量来增加进度条控件的当前位置，并且重画此进度条来反映新的位置。**

请参阅 CProgressCtrl::SetPos，CProgressCtrl::SetRange，CProgressCtrl::StepIt

CProgressCtrl::SetPos

int SetPos(int *nPos*);

返回值

返回进度条控件的早先的位置。

参数

nPos

进度条控件的新位置。

说明

此成员函数根据 *nPos* 指定的位置来设置进度条控件的当前位置，并重画此进度条来反映新的位置。

进度条的这个位置不是它在屏幕上的物理位置，而是在 SetRange 中的上限和下限范围之间的位置。

请参阅 CProgressCtrl::OffsetPos，CProgressCtrl::SetRange，CProgressCtrl::StepIt

CProgressCtrl::SetRange

void SetRange(short *nLower*, short *nUpper*);
void SetRange32(int *nLower*, int *nUpper*);

参数

nLower

指定范围的下限（缺省值是零）。

nUpper

指定范围的上限（缺省值是 100**）。**

说明

此函数用来设置进度条控件范围的上限和下限，并重画此进度条来反映新的范围。

成员函数 SetRange32 **为进度条设置** 32 **位的范围。**

请参阅 CProgressCtrl::OffsetPos，CProgressCtrl::SetPos，CProgressCtrl::StepIt

CProgressCtrl::SetStep

int SetStep(int *nStep*);

返回值

返回原来的步增量。

参数

nStep

新的步增量。

说明

此函数为进度条控件指定步增量。步增量就是每调用一次 CProgressCtrl::StepIt **所增加进度条控件的当前位置的数量。**

缺省的步增量是 10。

请参阅　CProgressCtrl::OffsetPos，CProgressCtrl::SetPos，CProgressCtrl::StepIt

CProgressCtrl::StepIt

int StepIt();

返回值

返回进度条控件的原来的位置。

说明

此函数用步增量来增加一个进度条控件的当前位置。该步增量由成员函数 CProgressCtrl::SetStep 来设定。

请参阅　CProgressCtrl::SetPos，CProgressCtrl::SetRange，CProgressCtrl::SetStep

CPropExchange

CPropExchange 没有基类。

建立一个属性交换的环境和方向。

CPropExchange 类支持应用程序的 OLE 控件的永久性实现。永久性是控件的状态信息在控件本身与某种媒介之间的交换，通常这些状态信息由控件的属性来表现。

当框架获知要从存储器载入一个 OLE 控件的属性，或是要将这些属性存入永久存储器时，它就构造一个从 CPropExchange 派生的对象。

框架将一个指向这个 CPropExchange 对象的指针传递给你的控件的

DoPropExchange 函数。如果你使用 ClassWizard 来为你的控件创建开始文件，则控件的 DoPropExchange 函数调用 COleControl::DoPropExchange。这个基类版本交换控件的固有属性；你可以修改你的派生类版本来交换那些你给控件添加的属性。

在载入或创建一个控件的时候，可以使用 CPropExchange 来使控件的属性连续，或初始化控件的属性。CPropExchange 的成员函数 ExchangeProp 和 ExchangeFontProp 可以用来将控件的属性存入不同的介质，并从不同的介质载入属性。

#include <afxctl.h>

请参阅 COleControl:: DoPropExchange

CPropExchange 类成员

Operations	
ExchangeFontProp	交换一个字体属性
ExchangeProp	交换任何内建的类型属性
ExchangBlobProp	交换一个二进制的大对象（BLOB）属性
ExchangePersistertProp	在一个控件和一个文件之间交换一个属性
ExchangeVersion	交换一个 OLE 控件的版本号
IsLoading	表明属性是被载入一个控件还是从控件被保存
GetVersion	获取一个 OLE 控件的版本号

成员函数

CPropExchange::ExchangeBlobProp

virtual BOOL ExchangeBlobProp(LPCTSTR *pszPropName*, void** *ppvBlob*,
    const void* *pvBlobDefault* = NULL) = 0;

返回值

如果交换成功则返回非零值；否则返回 0**。**

参数

pszPropName

要交换的属性的名字。

ppvBlob

指向一个变量的指针，该变量指向属性所保存的地方（变量通常是你的类的成员）。

pvBlobDefault

属性的缺省值。

说明

此函数用来使一个保存二进制大对象（BLOB**）数据的属性连续。**

属性的值从由 *ppvBlob* **引用的变量中读出，或在适当的时候将属性值写入这个变量。如果指定了** *pvBlobDefault*，**它将被用来作为属性的缺省值。如果由于某种原因，控件的连续失败了，则使用这个值。**

函数 CArchivePropExchange::ExchangeBlobProp，CResetPropExchange::ExchangeBlobProp **和** CPropsetPropExchange::ExchangeBlobProp **加载这个纯虚函数。**

请参阅 COleControl::DoPropExchange，CPropExchange::ExchangeFontProp，CPropExchange::ExchangePersistentProp，CPropExchange::ExchangeProp

CPropExchange::ExchangeFontProp

virtual BOOL ExchangeFontProp(LPCTSTR *pszPropName*, CFontHolder& *font*,
    const FONTDESC FAR* *pFontDesc*,LPFONTDISP *pFontDispAmbient*) = 0;

返回值

如果交换成功则返回非零值；否则返回 0。

参数

pszPropName

要交换的属性的名字。

font

一个 CFontHolder **对象的引用，该对象包含了字体属性。**

pFontDesc

指向一个 FONTDESC **结构的指针，当** *pFontDispAmbient* **是** NULL **时，**

该结构包含了用来初始化字体属性的缺省状态的值。

pFontDispAmbient

一个指向字体的 IFontDisp 接口的指针，用来初始化字体属性的缺省状态。

说明

此函数用来在某种存储介质和控件之间交换字体属性。

如果字体属性正在从介质载入控件，则从介质中获取字体的特征，并用来初始化由 font 引用的 CFontHolder 对象。如果正在保存字体属性，则字体对象的特征被写入介质。

函数 CArchivePropExchange::ExchangeFontProp，CRestPropExchange::ExchangeFontProp 和 CPropsetPropExchange::ExchangeFontProp 加载这个纯虚函数。

请参阅 COleControl::DoPropExchange，CPropExchange::ExchangeBlobProp，
CPropExchange::ExchangePersistentProp，
CPropExchange::ExchangeProp

CPropExchange::ExchangePersistentProp

virtual BOOL ExchangePersistentProp(LPCTSTR *pszPropName*,
LPUNKNOWN FAR* *ppUnk*, REFIID *iid*, LPUNKNOWN *pUnkDefault*) = 0;

返回值

如果交换成功则返回非零值；否则返回 0。

参数

pszPropName

要交换的属性的名字。

ppUnk

指向一个变量的指针，该变量包含了一个指向属性的 IUnknown 接口的指针（此变量通常是你的类的成员）。

iid

控件要使用的有关这个属性的接口的接口指针。

pUnkDefault

这个属性的缺省值。

说明

此函数用来在控件和一个文件之间交换属性。

如果属性正被从文件载入到控件，则属性被从文件创建并初始化。如果正保存属性，则属性的值被写入文件。

函数 CArchivePropExchange::ExchangePersistentProp ，

CResetPropExchange::ExchangePersist- entProp 和 CPropsetPropExchange::ExchangePersistentProp 加载这个纯虚函数。

请参阅 COleControl::DoPropExchange，CPropExchange::ExchangeBlobProp，CPropExchange::ExchangeFontProp，CPropExchange::ExchangeProp

CPropExchange::ExchangeProp

virtual BOOL ExchangeProp(LPCTSTR *pszPropName*, VARTYPE *vtProp*, void* *pvProp*,
    const void* *pvDefault* = NULL) = 0;

返回值

如果成功则返回非零值；否则返回 0。

参数

pszPropName

要交换的属性的名字。

vtProp

一个符号，用来指定要交换的属性的类型。可能的取值如下：

Symbol	PropertyType
VT_I2	**短型**（short）
VT_I4	**长型**（long）
VT_BOOL	**布尔型**（BOOL）
VT_BSTR	CString
VT_CY	CY
VT_R4	**浮点型**（float）
VT_R8	**双精度型**（double）

pvProp

一个指向属性值的指针。

pvDefault

指向属性的一个缺省值的指针。

说明

此函数用来在某个存储介质和控件之间交换属性。

如果属性正被从介质载入控件，则从介质中获取属性的值，并将这个值存入由 *pvProp* **指向的控件中。如果正将属性存入介质，则由** *pvProp* **所指向的控件的值被写入介质中。**

函数 CArchivePropExchange::ExchangeProp，CResetPropExchange::ExchangeProp **和** CPropsetPropExchange::ExchangeProp **加载这个纯虚函数。**

请参阅 COleControl::DoPropExchange，CPropExchange::ExchangeBlobProp，CPropExchange::ExchangeFontProp，CPropExchange::ExchangePersisentProp

CPropExchange::ExchangeVersion

BOOL ExchangeVersion (DWORD& *dwVersionLoaded*,DWORD *dwVersionDefault*,
BOOL *bConvert*);

返回值

如果成功则返回非零值；否则返回 0。

参数

dwVersionLoaded

指向一个变量的引用，该变量用来保存被载入的永久数据的版本号。

dwVersionDefault

控件的当前版本号。

bConvert

用来表明是将永久数据转换为当前的版本号，还是保持它载入时的版本

号。

说明

框架调用此函数来处理一个版本号的持续性。

请参阅 COleControl::ExchangeVersion

CPropExchange::GetVersion

DWORD GetVersion();

返回值

返回控件的版本号。

说明

此函数用来获取控件的版本号。

CPropExchange::IsLoading

BOOL IsLoading();

返回值

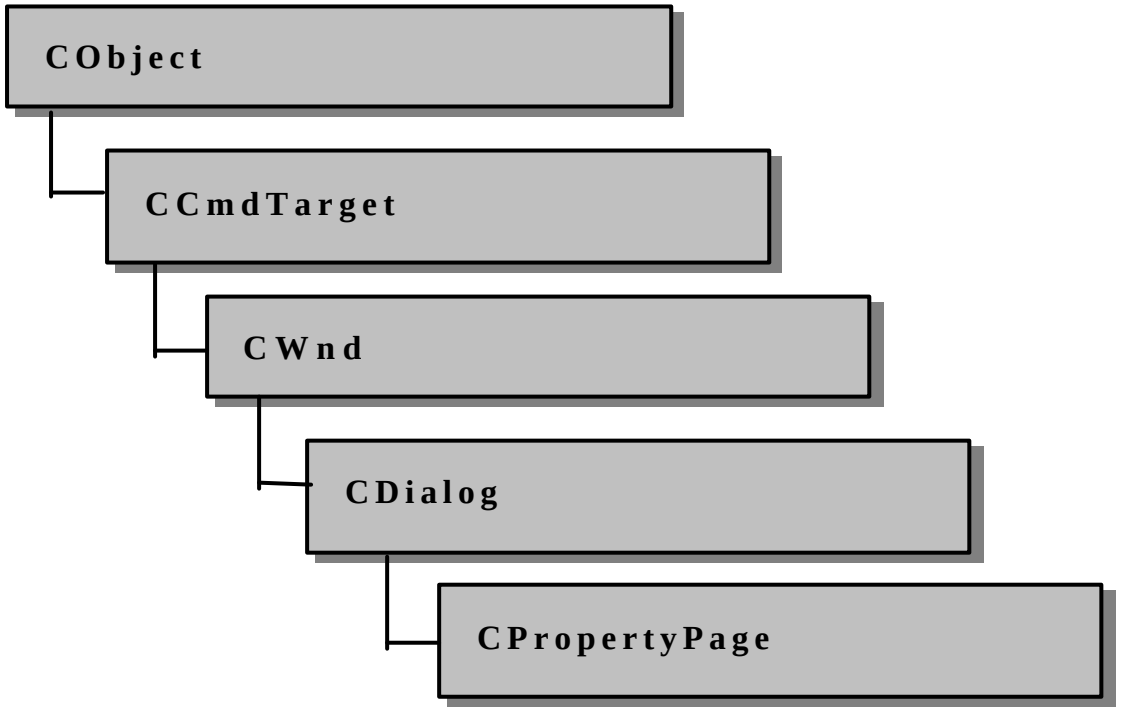
如果属性被载入则返回非零值；否则返回 0**。**

说明

此函数用来确定属性是被载入控件，还是被从控件保存。

请参阅 COleControl::DoPropExchange

CPropertyPage



类 CPropertyPage 的对象表示一张属性表的各页，或者说是被看作是标签对话框。同标准对话框一样，要为属性表中每一页从 CPropertyPage 类中派生一个新类。要使用 CPropertyPage 派生对象，首先要创建一个 CPropertySheet 对象，

然后为属性表中的每一页创建一个对象。为表中的每一页调用 CPropertySheet::AddPage 函数，然后对一个模式属性表调用 CPropertySheet::DoModal 函数来显示此属性表。对一个非模式属性表调用 CPropertySheet::Create 来显示此属性表。

你可以创建一种被称为向导的标签对话框，这种对话框包括一个属性表，该表有一系列属性页来引导用户进行一项操作的每一个步骤，比如说设置一个设备或创建一个时事通讯。在一个向导类型的标签对话框中属性页没有标签，每次只有一个属性页是可见的。而且，一个向导类型的对话框没有 OK 和 Apply Now 按钮，代替它们的是一个 Back 按钮，一个 Next 或 Finish 按钮和一个 Cancel 按钮。

如果要获取更多有关建立一个作为向导的属性表的信息，请参见 CPropertySheet::SetWizard- Mode。要获取更多有关使用 CPropertyPage 对象的

信息，请参见“Visual C++程序员指南”中的文章“属性表”。

#indlude <afxdlgs.h>

请参阅 CPropertySheet，CDialog，CPropertySheet::SetWizardMode

CPropertyPage 类成员

Data Members	
m_psp	Windows PROPSHEETPAGE 结构。提供对基本属性页参数的访问

Construction	
CpropertyPage	构造有关 CPropertyPage 对象
Construct	构造有关 CPropertyPage 对象。如果你想指定在运行时的参数，或者是你使用的是数组，就使用 Construct 函数

Operations	
CancelToClose	将 OK 按钮改变为读取 Close，并在一个模式属性表的页中进行了一次不可恢复的改变之后，使 Cancel 按钮无效
SetModified	用来激活一个 Apply Now 按钮，或使它成为不活动
QuerySiblings	向前传送消息到属性表的每一页

Overridables	
OnCancel	当单击 Cancel 按钮时由框架调用
OnKillActive	当当前页不再是活动页时由框架调用。在此执行数据确认
OnOK	当 OK，Apply Now 或 Close 按钮被单击时由框架调用
OnSetActive	当某页成为活动页时由框架调用
OnApply	当 Apply Now 按钮被单击时由框架调用
OnReset	当 Cancel 按钮被单击时由框架调用
OnQueryCancel	当 Cancel 按钮被单击，并在发生取消操作之前由框架调用

续表

OnWizardBack	在使用一个向导类型的属性表的情况下，当 Back 按钮被单击时由框架调用
OnWizardNext	在使用一个向导类型的属性表的情况下，当 Next 按钮被单击时由框架调用
OnWizardFinish	在使用一个向导类型的属性表的情况下，当 Finish 按钮被单击时由框架调用

成员函数

CPropertyPage::CancelToClose

```
void CancelToClose();
```

说明

在对一个模式属性表的页中的数据进行了不可恢复的改变之后，调用此函数。

此函数将把 OK 按钮改变为 Close 按钮，并使 Cancel 按钮无效。这种改变警告用户，刚才所作的改变是永久性的，并且这种修改不能被取消。

在一个无模式属性表中，CancelToClose 成员函数不做任何事情，因为一个无模式属性表不会有一个缺省的 Cancel 按钮。

请参阅 CPropertyPage::OnKillActive，CPropertyPage::SetModified

CPropertyPage::Construct

```
void Construct( UINT nIDTemplate, UINT nIDCaption = 0 );
void Construct( LPCTSTR lpszTemplateName, UINT nIDCaption = 0 );
```

参数

nIDTemplate

用于这个页的模板 ID。

nIDCaption

用来放在此页的标签中的名字的 ID。如果是 0，则将从对话框模板中取出用于此页的名字。

lpszTemplateName

包含一个空结尾的字符串，该字符串是模板资源的名字。

说明

此成员函数用来构造一个 CPropertyPage 对象。在满足了下列所有条件后，此对象会被显示：

- 已经用 CPropertySheet::AddPage 将此页添加到一个属性表中。
- 已经调用了属性表的 DoModal 或 Create 函数。
- 用户已经选择了（调整到）这一页。

如果没有调用另一个构造函数，则调用 Construct。Construct 成员函数是灵活的，

因为你可以让参数为空，然后在你的代码中的任何地方来指定多个参数和构造。

当你是在处理数组时，你就必须使用 Construct，而且你必须为数组的每一个成员调用 Construct，以使数据成员可以被分配适当的值。

请参阅 CPropertyPage:: CPropertyPage，CPropertySheet::DoModal，
CPropertySheet::AddPage

CPropertyPage::CPropertyPage

CPropertyPage();
CPropertyPage(UINT *nIDTemplate*, UINT *nIDCaption* = 0);
CPropertyPage(LPCTSTR *lpszTemplateName*, UINT *nIDCaption* = 0);

参数

nIDTemplate

用于此页的模板 ID。

nIDCaption

用来放在此页的标签中的名字 ID。如果是 0，则名字将从此页的对话框模板中取得。

lpszTemplateName

指向一个字符串，该字符串包含了用于此页的模板的名字。不可以是 NULL。

说明

此函数用来构造一个 CPropertyPage 对象。在满足下列条件之后，此对象将被显示：

- 已经用 CPropertySheet::AddPage 将此页添加到一个属性表中。
- 已经调用了属性表的 DoModal 或 Create 函数。
- 用户已经选择了（调整到）这一页。

如果你有多个参数（例如，你正使用数组），则使用 CPropertySheet::Construct **来代替** CPropertyPage。

请参阅 CPropertySheet::Create，CPropertySheet::DoModal，CPropertySheet::AddPage，CPropertyPage::Construct

CPropertyPage::OnApply

virtual BOOL OnApply();

返回值

如果改变被接受则返回非零值；否则返回 0。

说明

当用户选择 OK 或 Apply Now 按钮后，框架调用此函数。当框架调用此函数时，对属性表中的所有属性页做的改变都被接受了，属性表保持焦点，OnApply 返回 TRUE(值 1)。在框架可以调用 OnApply 之前,你必须已经调用了 SetModified 并将它的参数设置为 TRUE。这将会使得在用户一进行有关属性页的改变时就激活 Apply Now 按钮。

重载这个函数来指定当用户单击 Apply Now 按钮时，你的程序会进行什么动作。在重载时，函数应该返回 TRUE 来表示接受改变，返回 FALSE 来表示禁止改变。

OnApply 的缺省实现调用 OnOK。

如果要获取更多有关当用户在一个属性表中按下 Apply Now 或 OK 按钮时发送

的通知消息的信息，请参见 Win32 文档中的 PSN_APPLY。

请参阅 CPropertyPage::SetModified，CPropertyPage::OnOK

CPropertyPage::OnCancel

virtual void OnCancel();

说明

当选择 Cancel 按钮时，框架调用此成员函数。重载这个成员函数来执行 Cancel 按钮的动作。缺省的实现忽略所做的任何改变。

请参阅 CPropertyPage::OnApply，CDialog::OnCancel，CPropertyPage::OnOK

CPropertyPage::OnKillActive

virtual BOOL OnKillActive();

返回值

如果数据被成功更新则返回非零值；否则返回 0。

说明

当页不再是活动页时，框架调用此函数。重载这个成员函数来执行指定的数据确认任务。

此成员函数的缺省实现将对属性页中的控件所做的设置拷贝到属性页的成员变量中。如果数据没有被成功地更新，则属于一个对话框数据确认（DDV）错误，该页保持焦点。

在此成员函数成功返回之后，框架将调用此页的 OnOK 函数。

请参阅 CWnd::UpdateData，CPropertyPage::OnOK，CPropertyPage::OnSetActive

CPropertyPage::OnOK

virtual void OnOK();

说明

当用户选择了 OK **或** Apply Now **按钮时，在框架调用了** OnKillActive **之后，框架立即调用此函数。重载这个成员函数来实现当用户解散整个属性表时，执行对当前活动页指定的行为。**

此成员函数的缺省实现将该页标记为“清除（clear**）”来反映数据已经在** OnKillActive **函数中更新了。**

请参阅 CDialog::OnOK，CPropertyPage::OnKillActive

CPropertyPage::OnQueryCancel

virtual BOOL OnQueryCancel();

返回值

如果禁止取消操作则返回 FALSE**；如果允许取消操作则返回** TRUE**。**

说明

当用户单击 Cancel **时，并在取消动作发生之前，框架调用此成员函数。**

重载此成员函数来指定当用户单击 Cancel **按钮时程序要做的动作。**

OnQueryCancel **的缺省实现返回** TRUE**。**

CPropertyPage::OnReset

virtual void OnReset();

说明

当用户选择 Cancel **按钮时，可以调用此成员函数。当框架调用此函数时，用户向前对所有属性页所做的选择** Apply Now **按钮确认的改变都将被放弃，而属性表保持焦点。**

重载这个成员函数来指定当用户单击 Cancel **按钮时程序将执行的动作。**

OnReset **的缺省实现不做任何事情。**

请参阅 CPropertyPage::OnCancel，CPropertyPage::OnApply

CPropertyPage::OnSetActive

virtual BOOL OnSetActive();

返回值

如果成功地激活了此页则返回非零值；否则返回 0。

说明

当用户选择了此页并将其变为活动页时，框架调用这个成员函数。重载此成员函数来执行当一个页被激活时要执行的任务。你的关于这个成员函数的重载应该在做任何其它处理之前调用缺省的版本。

缺省的实现为该页创建窗口（如果在此之前没有创建的话），并使它成为活动页。

请参阅 CPropertyPage::OnKillActive

CPropertyPage::OnWizardBack

virtual LRESULT OnWizardBack();

返回值

如果自动前进到下一页，则返回 0；如果禁止页改变，则返回-1。如果所到的页是除下一页之外的其它页，则返回要显示的对话框的标识符。

说明

当用户在一个向导中单击 Back 按钮时，框架调用这个成员函数。

重载此成员函数来指定当按下 Back 按钮时用户必须做的一些操作。

要获取更多有关如何创建一个向导类型的属性表的信息，请参见 CPropertySheet::SetWizard- Mode。

请参阅 CPropertySheet::SetWizardMode

CPropertyPage::OnWizardFinish

virtual BOOL OnWizardFinish();

返回值

当向导完成时如果属性表被销毁，则返回非零值；否则返回零。

说明

当用户在一个向导中单击 Finish **按钮时框架调用此成员函数。当框架调用这个成员函数时，在** Finish **按钮被按下并且** OnWizardFinish **返回** TRUE **（一个非零值）时，属性表被销毁。**

重载这个函数来指定当 Finish **按钮被按下时，用户必须执行的一些动作。当重**

载这个函数时，返回 FALSE 来禁止属性表被销毁。

要获取更多有关当用户在一个向导属性表中按下 Finish 按钮时发送的通知消息的信息，请参见 Win32 文档中的 PSN_WIZFINISH。

如果要获取更多关于如何创建一个向导类型的属性表的信息，请参见 CPropertySheet::Set- WizardMode。

请参阅　CPropertyPage::SetWizardMode

CPropertyPage::OnWizardNext

virtual LRESULT OnWizardNext();

返回值

如果自动前进到下一页，则返回 0；如果禁止页改变，则返回-1。如果所到的

页是除下一页之外的其它页，则返回要显示的对话框的标识符。

说明

当用户在一个向导中单击 Next 按钮时，框架调用这个成员函数。

重载这个成员函数来指定当 Next 按钮被按下时用户必须做的某些动作。

如果要获取更多关于如何创建一个向导类型的属性表的信息，请参见 CPropertySheet::Set- WizardMode。

请参阅 CPropertySheet::SetWizardMode

CPropertyPage::QuerySiblings

LRESULT QuerySiblings(WPARAM *wParam*, LPARAM *lParam*);

返回值

如果在属性表中的所有页都返回 0，则函数返回 0；否则，返回属性表中的每一页返回的非零值。

参数

wParam

指定附加的与消息相关的信息。

lParam

指定附加的与消息相关的信息。

说明

此成员函数用来将一个消息向前传递给属性表中的每一页。如果有一个页返回一个非零值，则属性表不再将消息发送给后续页。

CPropertyPage::SetModified

void SetModified(BOOL *bChanged* = TRUE);

参数

bChanged

如果是 TRUE**，则表明在最后一次应用后属性页的设置被修改了；如果是** FALSE**，则表明属性页的设置被应用或被忽略。**

说明

此成员函数用来使 Apply Now **按钮有效或无效，这要根据属性页中的设置是否应该被应用到适当的外部对象。**

框架知道哪一页是“脏的”，就是那些你对它们调用了 SetModified(TRUE)**的属性页。如果你对某一属性页调用了** SetModified(TRUE)**，则** Apply Now **按钮**

将总是有效的。当你对某一属性页调用了 SetModified(FALSE)时，则 Apply Now 按钮将总是无效的，但是只有在没有任何一个属性页是“脏的”的时候。

请参阅 CPropertyPage::CancelToClose

数据成员

CPropertyPage::m_psp

说明

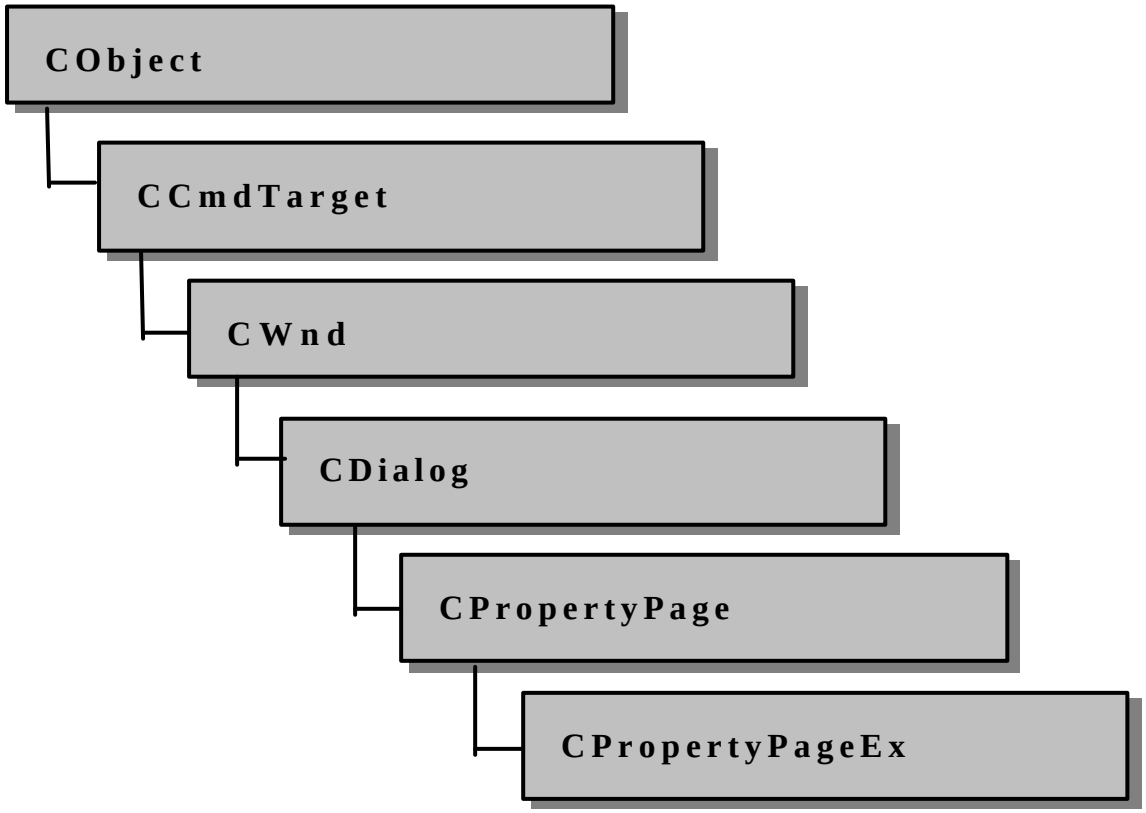
m_psp 是一个结构，它的成员保存了 PROPSHEETPAGE 的特征。在一个属性页被构造之后，使用这个结构来初始化该属性页的外观。

如果要获取更多关于这个结构的消息，包括它的成员的列表，请参见“Windows

SDK 程序员参考”中的 PROPSHEETPAGE。

请参阅 CPropertySheet，PROPSHEETPAGE

CPropertyPageEx



CPropertyPageEx 支持在 Windows 98 和 Windows NT 5 中引进的扩展的 PROPSHEETPAGE 结构。此结构中包含了附加的标志和成员，它们支持下列特征：

- 一个可以容纳标题和子标题的宽的页眉区。

在你的属性页对象中自动显示新的页眉，调用 CPropertyPageEx::Contstruct 或 CpropertyPage- Ex:: CPropertySheetEx 来为标题和子标题传递有效的值。

在其它的方面，CPropertyPageEx 与它的基类 CPropertyPage 具有一样的行为。

请参阅 CPropertySheet，CPropertySheetEx

CPropertyPageEx 类成员

Construction	
CpropertyPageEx	**构造一个** CPropertyPageEx **对象**
Construct	**构造一个** CPropertyPageEx **对象**

请参阅 CPropertyPage, CPropertySheetEx

成员函数

CPropertyPageEx::Construct

```
void Construct( UINT nIDTemplate, UINT nIDCaption = 0, UINT nIDHeaderTitlt =
0,
      UINT nIDHeaderSubTitle = 0 );
void Construct( LPCTSTR lpszTemplateName, UINT nIDCaption = 0,
    UINT nIDHeaderTitle = 0, UINT nIDHeaderSubTitle = 0 )
```

参数

nIDTemplate

用于此页的模板的 ID。

lpszTemplateName

指向一个字符串的指针，该字符串包含用于此页的模板的名字。

nIDCaption

放在此页的标签中的名字的 ID。如果是 0，则名字从此页的对话框模板中取出。其缺省值为 0。

nIDHeaderTitle

要被放在此属性页的页眉区的标题位置的名字的 ID。缺省值为 0。

nIDHeaderSubTile

要被放在此属性页的页眉区的子标题位置的名字的 ID。缺省值为 0。

说明

此成员函数用来构造一个 CPropertyPageEx **对象。**

如果你想要在运行时指定你的参数，或你要处理数组，请使用 Construct。

请参阅 CPropertyPage::Construct，CPropertyPageEx:: CPropertyPageEx

CPropertyPageEx:: CPropertyPageEx

```
CPropertyPageEx();
CPropertyPageEx( UINT nIDTemplate, UINT nIDCaption = 0, UINT nIDHeaderTitle = 0,
    UINT nIDHeaderSubTitle = 0 );
CPropertyPageEx( LPCTSTR lpszTemplateName, UINT nIDCaption = 0,
    UINT nIDHeaderTitle = 0, UINT nIDHeaderSubTitle = 0 )
```

参数

nIDTemplate

用于此页的模板的 ID。

nIDCaption

放在此页标签中的名字的 ID。如果是 0，则名字从此页的对话框模板中取出。

nIDHeaderTitle

要被放在此属性页的页眉区的标题位置的名字的 ID。

nIDHeaderSubTile

要被放在此属性页的页眉区的子标题位置的名字的 ID。缺省值为 0。

lpszTemplateName

指向一个字符串的指针，该字符串包含用于此页的模板的名字。

说明

此成员函数用来构造一个 CPropertyPageEx 对象。在满足了下列所有条件后，

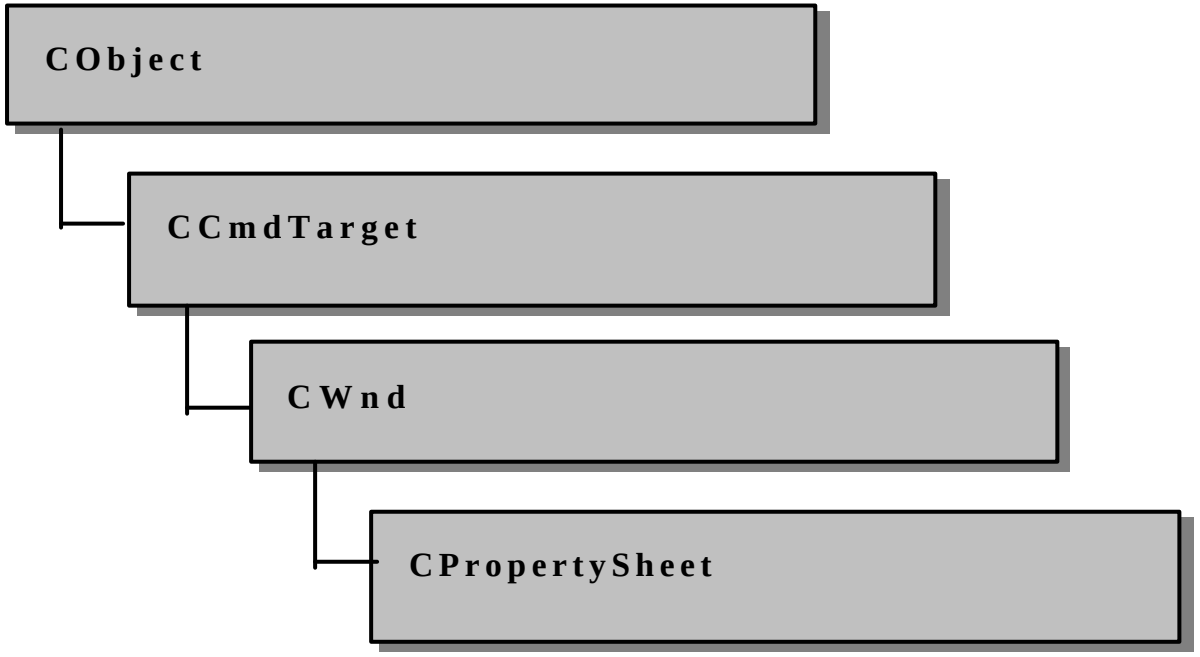
此对象会被显示：

- 已经用 CPropertyExSheet::AddPage 将此页添加到一个属性表中。
- 已经调用了属性表的 DoModal 或 Create 函数。
- 用户已经选择了（调整到）这一页。

如果你有多个参数（例如，你使用了一个数组），请使用 Constrcuct 来代替 CPropertyPageEx。

请参阅 CPropertyPage:: CPropertyPage

CPropertySheet



CPropertySheet 类对象表示属性表，或者说是标签对话框。一个属性表由一个 CPropertySheet 对象和一个或多个 CPropertyPage 对象构成。一个属性表由框架

来显示，就象是一个具有一系列标签索引的窗口。用户通过这些标签索引来选择当前的页，和一块用于当前所选页的区域。

虽然 CPropertySheet 不是从 CDialog 派生而来的，但是管理一个 CPropertySheet 对象类似于管理一个 CDialog 对象。例如，一个属性表的创建需要分两部分构造：调用构造函数，然后对模式属性表调用 DoModal，或对非模式属性表调用 Create。CPropertySheete 有两种类型的构造函数：CPropertySheet::Construct 和 CPropertySheet:: CPropertySheet。

在一个 CPropertySheet 对象和某个外部对象之间交换数据，类似于与一个 CDialog 对象交换数据。两者之间的重要差别是：一个属性表的设置通常是 CPropertyPage 对象的成员变量，而不是 CPropertySheet 对象本身。

你可以创建一种被称为向导的标签对话框，这种对话框包括一个属性表，该表有一系列属性页来引导用户进行一项操作的每一个步骤，比如说设置一个设备

或创建一个时事通讯。在一个向导类型的标签对话框中属性页没有标签，每次只有一个属性页是可见的。而且，一个向导类型的对话框没有 OK 和 Apply Now 按钮，代替它们的是一个 Back 按钮，一个 Next 或 Finish 按钮和一个 Cancel 按钮。

要创建一个向导类型的对话框，其步骤与创建一个标准的属性表的步骤是一样的，但是要在调用 DoModal 之前调用 SetWizardMode。为了使向导按钮有效，调用 SetWizardButtons，使用标志来定制它们的功能和外观。为了使 Finish 按钮有效，在用户完成了在向导的最后一页中的动作之后调用 SetFinishText。

```
#include <afxdlgs.h>
```

CPropertySheet 类成员

Data Members	
m_psh	Windows PROPSHEETHEADER 结构。提供对基本属性表参数的访问

Construction	
CpropertySheet	构造一个 CPropertySheet 对象
Construct	构造一个 CPropertySheet 对象

Attributes	
GetActiveIndex	获取属性表的活动页的索引
GetPageIndex	获取属性表指定页的索引
GetPageCount	获取属性表中的页数
GetPage	获取指向指定页的指针
GetActivePage	返回活动页对象
SetActivePage	可设计地设置活动页对象
SetTitle	设置属性表的标题
GetTabControl	获取一个指向一个标签控件的指针
SetFinishText	设置 Finish 按钮的文本
SetWizardButtons	使向导按钮有效
SetWizardMode	使向导模式有效
EnableStacdedTabs	代码属性表是使用堆叠标签还是滚动标签

Operations	
DoModal	显示一个模式属性表
Create	显示一个无模式属性表
AddPage	向属性表中添加一个页
RemovePage	从属性表中移去一页
PressButton	在一个属性表中模拟对指定按钮的选择
EndDialog	终止属性表

成员函数

CPropertySheet::AddPage

void Addpage(CPropertyPage *pPage);

参数

pPage

指向要被添加到属性表中去的页。不能是 NULL。

说明

此成员函数用来将所提供的页添加到属性表中，并使它具有最右边的标签。此函数按你所希望的从左至右的顺序来添加页。

AddPage 将 CPropertyPage 对象添加到 CPropertySheet 对象的页列表中，但是并不为这些页实际地创建窗口。直到用户选择了一页，框架才为此页创建窗口。

当你用 AddPage 来添加一个属性页时，CPropertySheet 就是 CPropertyPage 的父。为了获得从属性页对属性表的访问，可以调用 CWnd::GetParent。

如果你在显示属性页之后调用 AddPage，则标签行将反映新添加的页。

请参阅 CPropertySheet::RemovePage

CPropertySheet::Construct

void Construct(UINT *nIDCaption*, CWnd* *pParentWnd* = NULL, UINT *iSelectPage* = 0):
void Construct(LPCTSTR *pszCaption*, CWnd* *pParentWnd* = NULL, UINT *iSelectPage* = 0);

参数

nIDCaption

用于属性表的标题的 ID。

pParentWnd

指向此属性表的父窗口的指针。如果是 NULL，**则其父窗口就是应用程序的主窗口。**

iSelectPage

最初在最顶上的页的索引。缺省值是添加到表中的第一页的索引。

pszCaption

指向一个字符串的指针，该字符串包含了用于属性表的标题。不能是NULL。

说明

此成员函数用来构造一个 CPropertySheet 对象。如果还没有调用一个类构造函数，则调用此成员函数。例如，当你声明或分配 CPropertySheet 对象数组时，就调用 Construct。在有数组的情况下，你必须为数组中的每一个成员调用Construct。

示例

下面的示例说明了在什么情况下你要调用 Construct。

```
int i;
CPropertySheet    grpropsheet[4];
```

CPropertySheet　　someSheet;　　//　　不需要为它调用 Construct。

```
UINT rgID[4] = {IDD_SHEET1, IDD_SHEET2, IDD_SHEET3, IDD_SHEET4};
for ( i= 0; i < 4; i++)
   grpropsheet[i].Construct( rgID[i] );
```

请参阅　CPropertySheet:: CPropertySheet，CPropertySheet::DoModal，

CPropertySheet::Create

CPropertySheet::CPropertySheet

```
CPropertySheet();
CPropertySheet( UINT nIDCaption, CWnd* pParentWnd = NULL, UINT
iSelectPage = 0 );
CPropertySheet( LPCTSTR pszCaption, CWnd* pParentWnd = NULL,
      UINT iSelectPage = 0 );
```

参数

nIDCaption

用于属性表的标题的 ID。

pParentWnd

指向此属性表的父窗口的指针。如果是 NULL，则其父窗口就是应用程序的主窗口。

iSelectPage

最初在最顶上的页的索引。缺省值是添加到表中的第一页的索引。

pszCaption

指向一个字符串的指针，该字符串包含了用于属性表的标题。不能是 NULL。

说明

此函数用来构造一个 CPropertySheet 对象。要显示此 CPropertySheet，调用 DoModal 或 Create。第一个参数中包含的字符串将被放置在属性表的标题条中。如果你有多个参数（例如，你使用的是数组），则使用 Construct 来代替

CPropertySheet。

请参阅 CPropertySheet::Construct，CPropertySheet::DoModal，

CPropertySheet::Create，CPropertyPage

CPropertySheet::Create

BOOL Create(CWnd* *pParentWnd* = NULL, DWORD *dwStyle* = (DWORD)-1,
DWORD *dwExStyle* = 0);

返回值

如果成功地创建了属性表则返回非零值；否则返回 0。

参数

pParentWnd

指向一个父窗口。如果是 NULL，**则父是桌面。**

dwStyle

属性表窗口的风格。要获取可用风格的完整列表，参见“Windows风格”。

dwExStyle

属性表的扩展窗口风格。要获取可用风格的完整列表，参见“扩展的窗口风格”。

说明

此成员函数用来显示一个无模式的属性表。可以在构造函数的内部调用 Create，也可以在激活构造函数之后调用它。

当传递给参数 *dwStyle* 的值是-1 时，表示的是缺省风格，这种风格实际是：WS_SYSMENU | WS_POPUP | WS_CAPTION | DS_MODALFRAME | DS_CONTEXT_HELP | WS_VISIBLE。当传递给参数 dwExStyle 的值是 0 时，表示是缺省的扩展风格，实际是：WS_EX_DLGMODALFRAME。

在创建属性表之后，Create 成员函数立即返回。要销毁这个属性表，可以调用 CWnd::DestroyWindow。

调用 Create 来显示的无模式属性表，没有象模式属性表那样的 OK，Cancel，Apply Now 和 Help 按钮。必须由用户来创建所需要的按钮。

要显示一个模式属性表，可以调用 DoModal。

请参阅　CDialog::Create，CPropertySheet::DoModal

CPropertySheet::DoModal

virtual int DoModal();

返回值

如果函数成功则返回 IDOK 或 IDCANCEL；否则返回 0 或-1。如果此属性表是

作为一个向导（参见 SetWizardMode）建立的，DoModal 返回 ID_WIZFINISH 或 IDCANCEL。

说明

此成员函数用来显示一个模式属性表。其返回值对应于用来关闭属性表的控件的 ID。此函数返回后，Windows 响应这个属性表，所有的属性页都会被销毁。而这些对象本身仍然存在。通常，你将在 DoModal 返回 IDOK 之后从 CPropertyPage 对象检取数据。

要显示一个无模式属性表，请调用 Create 来代替此函数。

注意　在一个属性页被第一次从它相应的对话框资源创建时，它有可能引发一个第一次机会（first-chance）异常。这是在创建此页之前属性页将对话框资源的风格改变成了所需的风格的结果。因为资源通常来说

是只读的，所以这导致了一个异常。这个异常由系统处理，系统会自动拷贝修改后的资源。这样，第一次机会（first-chance）异常就被忽略了。

由于这个异常必须由操作系统来处理，所以不要用一个 C++ try/catch 块来隐藏调用 CPropertySheet::DoModal，因为在一个 C++ try/catch 块中 catch 会处理所有的异常，比如，catch(...)。它将处理那些属于操作系统的异常，这将导致不可预料的行为发生。但是，通过指定异常类型或结构化异常处理来处理 C++异常就是安全的，在结构化异常处理中，访问非法异常被传递给操作系统。

请参阅 CDialog::DoModal，CPropertySheet::Create

CPropertySheet::EnableStackedTabs

void EnableStackedTabs(BOOL *bStacked*);

参数

bStacked

表明在属性表中堆叠式标签是否是有效的。通过设置 *bStacked* **为** FALSE**，可以使堆叠式行标签无效。**

说明

此成员函数用来表明在一个属性表中是否使用堆叠式标签行。缺省的，如果一个属性表有很多标签，它们不能在属性表的宽度中按一个单行放下，则这些标签将按多行堆叠。要使用滚动标签来代替堆叠标签，请在调用 DoModal **或** Create **之前将** *bStacked* **设置为** FALSE **来调用** EnableStackedTabs。

当你创建一个模式或无模式的属性表时，你必须调用 EnableStackedTabs。为了在一个 CPropertySheet 派生类中混合这种风格，请为 WM_CREATE 写一个消息句柄。在 CWnd::OnCreate 的重载版本中，在调用基类实现之前调用 EnableStackedTabs(FALSE)。

示例

```
int CMyPropertySheet::OnCreate( LPCREATESTRUCT lpCreateStruct )
{
    // 设置为滚动标签风格
    EnableStackedTabs( FALSE );
    // 调用基类
    if(CPropertySheet::OnCreate( lpCreateStruct ) == -1)
        return –1;
    // TODO：在此添加你的指定创建代码
    return 0;
```

}

CPropertySheet::EndDialog

void EndDialog(int *nEndID*);

参数

nEndID

用来作为属性表的返回值的标识符。

说明

此成员函数用来终止属性表。当按下 OK，Cancel **或** Close **按钮时，框架调用这个成员函数。如果发生了一个要改变此属性表的事件，则调用此成员函数。**

请参阅 CPropertySheet::OnOK，CPropertySheet::OnCancel，CWnd::DestroyWindow

CPropertySheet::GetActiveIndex

int GetActiveIndex() const;

返回值

返回活动页的索引号。

说明

此成员函数用来获取属性表窗口中的活动页的索引号，然后用这个返回的索引号作为 GetPage **的参数。**

请参阅 CPropertySheet::GetPage，CPropertySheet::GetActivePage

CPropertySheet::GetActivePage

CPropertyPage* GetActivePage() const;

返回值

返回指向活动页的指针。

说明

此成员函数用来获取属性表窗口中的活动页。使用这个函数可以对活动页执行某些动作。

请参阅 CPropertySheet::GetPage

CPropertySheet::GetPage

CPropertyPage* GetPage(int *nPage*) const;

返回值

返回指向 *nPage* **参数对应的页的指针。**

参数

nPage

所希望的页的索引，开始于 0**。其值必须是包含在** 0 **和属性表的页数之间。**

说明

此成员函数返回一个指向此属性表中的指定页的指针。

请参阅 CPropertySheet::AddPage，CPropertySheet::GetActivePage，

CPropertySheet::GetPageCount，CPropertySheet::RemovePage，

CPropertySheet::SetTitle

CPropertySheet::GetPageIndex

int GetPageIndex(CPropertyPage* *pPage*) const;

返回值

返回一个页的索引号。

参数

pPage

指向具有要找的索引的页。不能是 NULL。

说明

此成员函数用来获取指定页在属性表中的索引号。例如，为了使用 SetActivePage **或** GetPage **你要用** GetPageIndex **来获取页的索引。**

请参阅 CPropertySheet::SetActivePage，CPropertySheet::GetPage

CPropertySheet::GetPageCount

int GetPageCount();

返回值

返回属性表中的页数。

说明

此成员函数用来确定当前在属性表中的页数。

请参阅 CPropertySheet::GetPage，CPropertySheet::AddPage，
CPropertySheet::RemovePage

CPropertySheet::GetTabControl

CTabCtrl* GetTabControl();

返回值

返回一个指向标签控件的指针。

说明

此成员函数用来获取一个指向标签控件的指针，并执行某些特定于这个标签控件的操作（也就是说，使用 CTabCtrl **中的任何** API**）。例如，如果你希望在初始化期间给每一个标签添加一个位图，你就可以调用这个函数。**

请参阅 CTabCtrl::CTabCtrl

CPropertySheet::PressButton

BOOL PressButton(int *nButton*);

返回值

如果成功则返回非零值；否则返回 0。

参数

nButton

此参数表明被按下的按钮。它可以是下列值之一：

- PSBTN_BACK　选择的是 Back 按钮。
- PSBTN_NEXT　选择的是 Next 按钮。
- PSBTN_FINISH　选择的是 Finish 按钮。
- PSBTN_OK　选择的是 OK 按钮。
- PSBTN_APPLYNOW　选择的是 Apply Now 按钮。
- PSBTN_CANCEL　选择的是 Cancel 按钮。

- PSBTN_HELP　选择的是 Help 按钮。

说明

此成员函数用来模拟在一个属性表中选择了某个指定的按钮。参见 PSM_PRESSBUTTON 可以获得更多有关 Windows SDK Pressbutton 消息的信息。

CPropertySheet::RemovePage

```
void RemovePage( CPropertyPage *pPage );
void RemovePage( int nPage );
```

参数

pPage

指向要被从属性表中移走的页。不能是 NULL。

nPage

要被移走的页的索引。其值必须是包含在 0 和属性表中的页数减 1 之间。

说明

此成员函数从属性表中移走一页，并销毁与此页相关的窗口。但 CPropertySheet 对象本身要直到自己的 CPropertySheet 窗口被关闭时才会被销毁。

请参阅 CPropertySheet::AddPage

CPropertySheet::SetActivePage

```
BOOL SetActivePage( int nPage );
BOOL SetActivePage(CPropertyPage* pPage );
```

返回值

如果成功地激活了属性表，则返回非零值；否则返回 0。

参数

nPage

要设置的页的索引。它的值必须是包含在 0 **和属性表中的页数减** 1 **之间。**

pPage

指向属性表中要设置的页。它不能是 NULL**。**

说明

此成员函数用来改变活动页。例如，如果一个用户对某一页的动作要导致另一页成为活动页，则可以使用 SetActivePage**。**

CPropertySheet::SetFinishText

void SetFinishText(LPCTSTR *lpszText*);

参数

lpszText

指向要显示在 Finish **命令按钮上的文本。**

说明

此成员函数用来设置 Finish **命令按钮中的文本。调用** SetFinishText **来在** Finish **命令按钮上显示文本，并在用户完成了在向导的最后一页上的动作后隐藏** Next **和** Back **按钮。**

CPropertySheet::SetTitle

void SetTitle(LPCTSTR *lpszText*, UINT *nStyle* = 0);

参数

nStyle

指定属性表标题的风格。此风格必须被指定为 0 或者是 PSH_PROPTITLE。如果将风格设置为 PSH_PROPTITLE，则单词“Properties for”显示在指定作为标题的文本的前面。

lpszText

指向用来在属性表的标题条中作为标题的文本。

说明

此成员函数用来指定属性表的标题（显示在一个框架窗口的标题条中的文本）。缺省的，一个属性表在属性表的构造函数中使用这个标题参数。

请参阅 CPropertySheet::GetPage，CPropertySheet::GetActivePage

CPropertySheet::SetWizardButtons

void SetWizardButtons(DWORD *dwFlags*);

参数

dwFlags

是一套用来定制向导按钮的功能和外观的标志。这个参数可以是下列值的组合：

- PSWIZB_BACK　Back **按钮。**
- PSWIZB_NEXT　Next **按钮。**
- PSWIZB_FINISH　Finish **按钮。**
- PSWIZB_DISABLEDFINISH　Disabled Finish **按钮。**

说明

此成员函数用来使一个向导属性表中的 Back，Next 或 Finish 按钮有效或者无效。只有在对话框打开之后才调用 SetWizardButtons；你不能在调用 DoModal 之前调用 SetWizardButtons。通常来说，应该从 CPropertySheet::OnSetActive 调用 SetWizardButtons。

如果你想改变在 Finish 按钮上的文本，或者想在用户完成向导的各个步骤之后隐藏 Back 和 Next 按钮，请调用 SetFinishText。值得注意的是，Finish 和 Next 是共享同一个按钮的。在某一个时间你可以显示 Finish 或 Next 按钮，但是不能同时显示它们两个。

CPropertySheet::SetWizardMode

void SetWizardMode();

说明

此成员函数用来建立一个作为向导的属性页。一个作为向导的属性页的一个关键特征就是：用户使用 Next 或 Finish，Back 和 Cancel 按钮而不是标签来导航。

在调用 DoModal 之前调用 SetWizardMode。在调用 SetWizardMode 之后，DoModal 将返回 ID_WIZFINISH（如果用户用 Finish 按钮来关闭）或者是 IDCANCEL。

SetWizardMode 设置 PSF_WIZARD 标志。

示例

```
CPropertySheet dlg;
CPropertySheet page1,page2;
dlg.AddPage( &page1 );
dlg.AddPage( &page2 );
dlg.SetWizardMode();
```

dlg.DoModal();

请参阅 CPropertySheet::DoModal

数据成员

CPropertySheet::m_psh

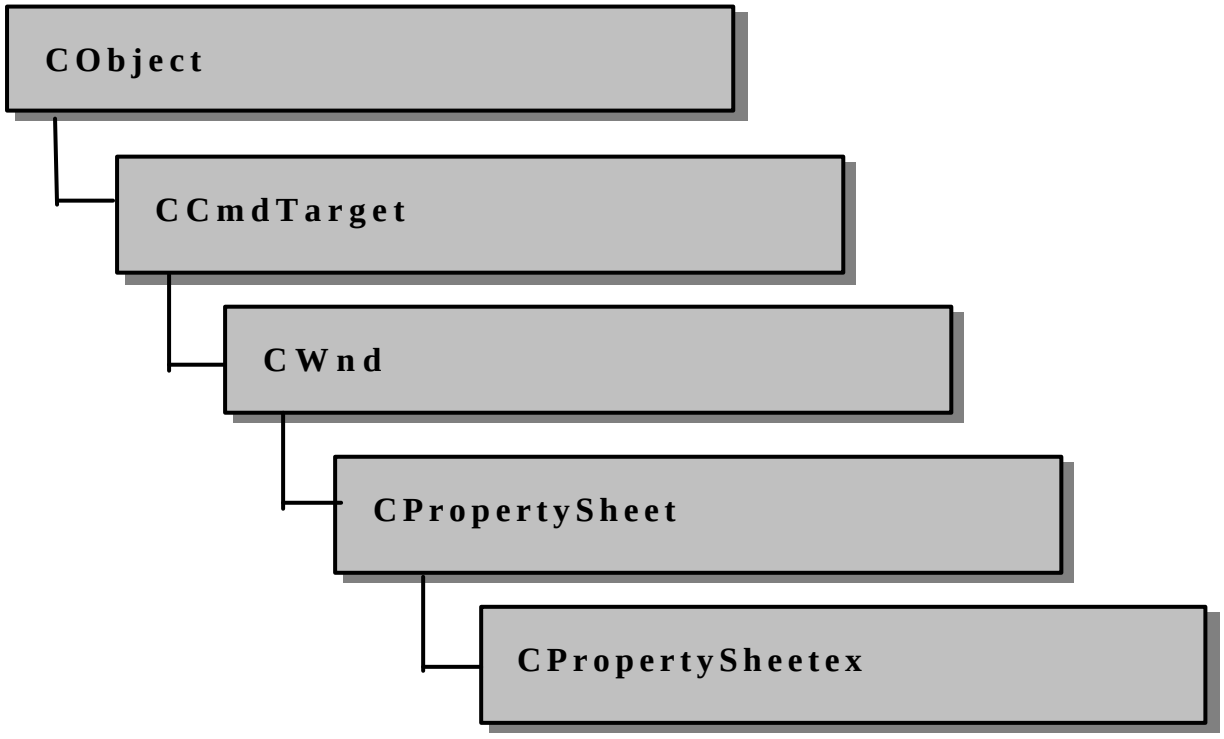
说明

m_psh **是一个结构，它的成员保存了** PROPSHEETHEADER **的特征。在一个属性表被构造之后，并在用** DoModal **成员函数来显示它之前，此结构可以用来初始化这个属性表的外观。例如，将** m_psh **的成员** dwSize **设置为你的属性表所要求的尺寸。**

如果要获取更多有关这个结构的信息，包括它的成员列表，请参见“Windows SDK 程序员参考”中的 PROPSHEETHEADER。

请参阅 CPropertySheet::DoModal

CPropertySheetEx



CPropertySheetEx 支持在 Windows 98 和 Windows NT 5 中引进的

PROPSHEETHEADER 结构。该结构包含了附加的标志和成员，支持下面的特征：

- 一个“水印”背景位图。

为了在你的属性表对象中自动显示这些新的图像，在调用 CPropertySheetEx::Construct 或 CPropertySheetEx:: CPropertySheetEx 时要给位图和剪贴板图像传递有效的值。

在所有其它的方面，CPropertySheetEx 具有与它的基类 CPropertySheet 相同的行为。

请参阅 CPropertyPage，CPropertyPageEx

CPropertySheetEx 类成员

Construction	
CPropertySheetEx	构造一个 CPropertySheetEx 对象
Construct	构造一个 CPropertySheetEx 对象。由具有多个参数的属性页使用

Data Members	
m_psh	PROPSHEETHEADER 结构，它的成员包含了要创建的属性表的特征

Operation	
AddPage	给属性表添加一个页

请参阅 CPropertyPage，CPropertyPageEx

成员函数

CPropertySheetEx::AddPage

void AddPage(CPropertyPageEx* *pPage*);

参数

pPage

指向一个要被添加到属性表中的页。不能是 NULL。

说明

此成员函数用来向属性表中添加一个属性页。按你所希望的从左至右的顺序来添加页。

CPropertySheetEx::Construct

void Construct(UINT *nIDCaption*, CWnd* *pParentWnd* = NULL, UINT *iSelectPage* = 0,
HBITMAP *hbmWatermark* = NULL, HPALETTE *hpalWatermark* = NULL,
HBITMAP *hbmHeader* = NULL);
void Construct(LPCTSTR *pszCaption*, CWnd* *pParentWnd* = NULL, UINT *iSelectPage* = 0,
HBITMAP *hbmWatermark* = NULL, HPALETTE *hpalWatermark* = NULL,
HBITMAP *hbmHeader* = NULL);

参数

nIDCaption

要用在属性表中的标题的 ID。

pParentWnd

指向属性表的父窗口的指针。如果是 NULL，**则父窗口将是应用程序的主窗口。**

iSelectPage

最初将要在最顶上的页的索引。缺省的是被添加到表中的第一页。

hbmWatermark

属性页的水印位图的句柄。

hpalWatermark

水印位图和/或页眉位图的调色板的句柄。

hbmHeader

属性页的页眉位图的句柄。

pszCaption

指向一个字符串的指针，该字符串包含了用于属性表的标题。它不能是NULL。

说明

此成员函数用来构造一个 CPropertySheetEx 对象。要获取更多的信息，请参见

CPropertySheet::Construct。

要显示这个属性表，请调用 DoModal **或** Create。**包含在第一个参数中的字符串将被放在属性表的标题条中。**

如果在调用 Construct **时传递的值是有效的，则水印和/或页眉图像被自动显示。**

请参阅 CPropertySheetEx:: CPropertySheetEx

CPropertySheetEx::CPropertySheetEx

```
CPropertySheetEx();
CPropertySheetEx( UINT nIDCaption, CW nd* pParentWnd = NULL, UINT iSelectPage = 0,
    HBITMAP hbmWatermark = NULL, HPALETTE hpalWatermark = NULL,
    HBITMAP hbmHeader = NULL );
CPropertySheetEx(LPCTSTR pszCaption,CWnd* pParentWnd = NULL,UINT iSelectPage= 0,
    HBITMAP hbmWatermark = NULL, HPALETTE hpalWatermark = NULL,
    HBITMAP hbmHeader = NULL );
```

参数

nIDCaption

要用在属性表中的标题的 ID。

pParentWnd

指向属性表的父窗口的指针。如果是 NULL，**则父窗口将是应用程序的主窗口。**

iSelectPage

最初将要在最项上的页的索引。缺省的是被添加到表中的第一页。

hbmWatermark

属性页的水印位图的句柄。

hpalWatermark

水印位图和/或页眉位图的调色板的句柄。

hbmHeader

属性页的页眉位图的句柄。

pszCaption

指向一个字符串的指针，该字符串包含了用于属性表的标题。它不能是NULL。

说明

此成员函数用来构造一个 CPropertySheetEx 对象。要显示这个属性表，请调用 DoModal 或 Create。包含在第一个参数中的字符串将被放置在属性表的标题条中。

如果你有多个参数（例如，如果你正使用一个数组），使用 CPropertySheetEx::Construct 来代替 CPropertySheetEx。

如果在调用 Construct 时传递的值是有效的，则水印和/或页眉图像被自动显示。

请参阅 CPropertySheetEx::Construct

数据成员

CPropertySheetEx::m_psh

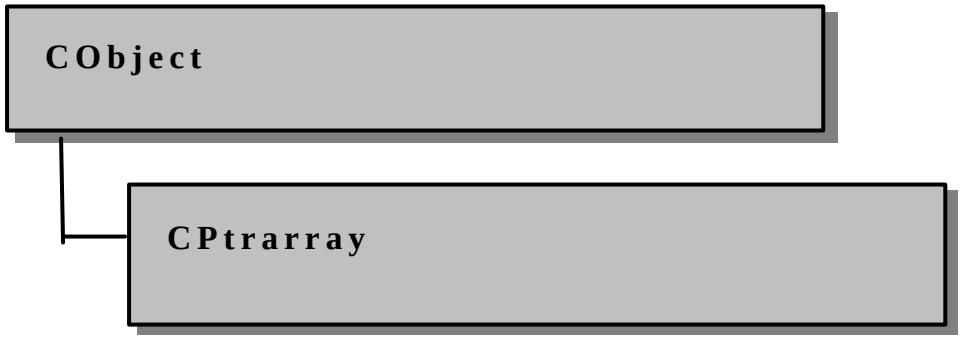
说明

此数据成员是一个结构，它的成员保存了 PROPSHFETHEADER 的特征。在创建了一个属性表之后，并在调用 CPropertySheetEx::DoModal 来显示它之前，可以使用这个结构来初始化这个属性表的外观。例如，将 m_psh 的成员 dwSize 的设置为你的属性表想要具有的尺寸。

如果要获取更多有关这个结构的信息，包括它的成员的列表，请参见“Platform SDK”中的 PROPSHEETHEADER。

请参阅 CPropertySheetEx::Construct

CPtrArray



CPtrArray 类支持 void 指针数组。

CPtrArray 的成员函数类似于 CObArray 类的成员函数。由于这种相似，你可以利用 CObArray 参考文档作为成员函数的说明。无论在何处使用一个 CObject 指针作为函数参数或返回值，都可以将它替换成 void。例如：

CObject* CObArray::GetAt(int <nIndex>) const;

可以转换成：

void* CPtrArray::GetAt(int <nIndex>) const;

CPtrArray 嵌入 IMPLEMENT_DYNAMIC 宏来支持运行时类型访问和转储到一个 CDumpContext 对象。如果你需要转储个别的指针数组元素，你必须将转储环境的深度设置为 1 或更大。

> **注意** 在使用一个数组之前，先用 SetSize 函数建立数组的大小，并为数组分配内存。如果应用程序没有使用 SetSize 函数，则向数组中增加元素会使它被频繁地再分配和拷贝。频繁再分配和拷贝的效率很低，并且可能使内存变得很零碎。

指针数组不能被串行化。

当一个指针数组被删除时，或当其元素被删掉时，只删掉指针，而这些指针所引用的实体不被删除。

#include <afxcoll.h>

请参阅 CObArray

CPtrArray 类成员

Construction	
CptrArray	构造一个空的 void 指针数组
Bounds	
GetSize	获取这个数组中的元素个数
GetUpperBound	返回最大的有效索引
SetSize	设置此数组中要包含的元素个数
Operations	
FreeExtra	释放超出当前上界的所有未用内存
RemoveAll	删除此数组中的所有元素
Element Access	
GetAt	返回给定索引处的值
SetAt	设置给定索引处的值；数组不运行扩展
ElementAt	返回对数组内的一个元素指针的一个临时引用
GetData	允许访问数组中的元素。不能是 NULL

Growing the Array	
SetAtGrow	设置给定索引处的值，如果必要，允许数组扩展
Add	将一个元素增加到数组尾；在必要时扩展数组
Append	将另一个数组添加到此数组；如果必要，则扩展此数组
Copy	将另一个数组拷贝给此数组；如果必要，则扩展此数组

Insertion/Removal	
InsertAt	将一个元素（或另一数组中的所有元素）插入到指定索引处
RemoveAt	删除指定索引处的元素

Operators	
Operator[ ]	设置或获取给定索引处的元素

CPtrList

CPtrList 类支持 void 指针列表。

CPtrList 类的成员函数类似于 CObList 类的成员函数。由于这种类似性，可以使用 CObList 参考文档作为成员函数的说明。无论在任何时候使用一个 CObject 指针作为函数参数或返回值，都可以将指针替换为 void。例如，

CObject* & CObList::Gethead() const;

可以替换成：

void * CPtrList::Gethead() const;

CPtrList 嵌入了 IMPLEMENT_DYNAMIC 宏来支持运行类型访问和转储到 CDumpContext 对象。如果应用程序需要转储各个指针列表元素，则必须将转储环境的深度设置为 1 或大于 1。

指针列表不能为 NULL。

当一个 CPtrList 对象被删除时，或当其元素被删除时，只删除指针，而指针所引用的实体并未被删除。

#include <afxcoll.h>

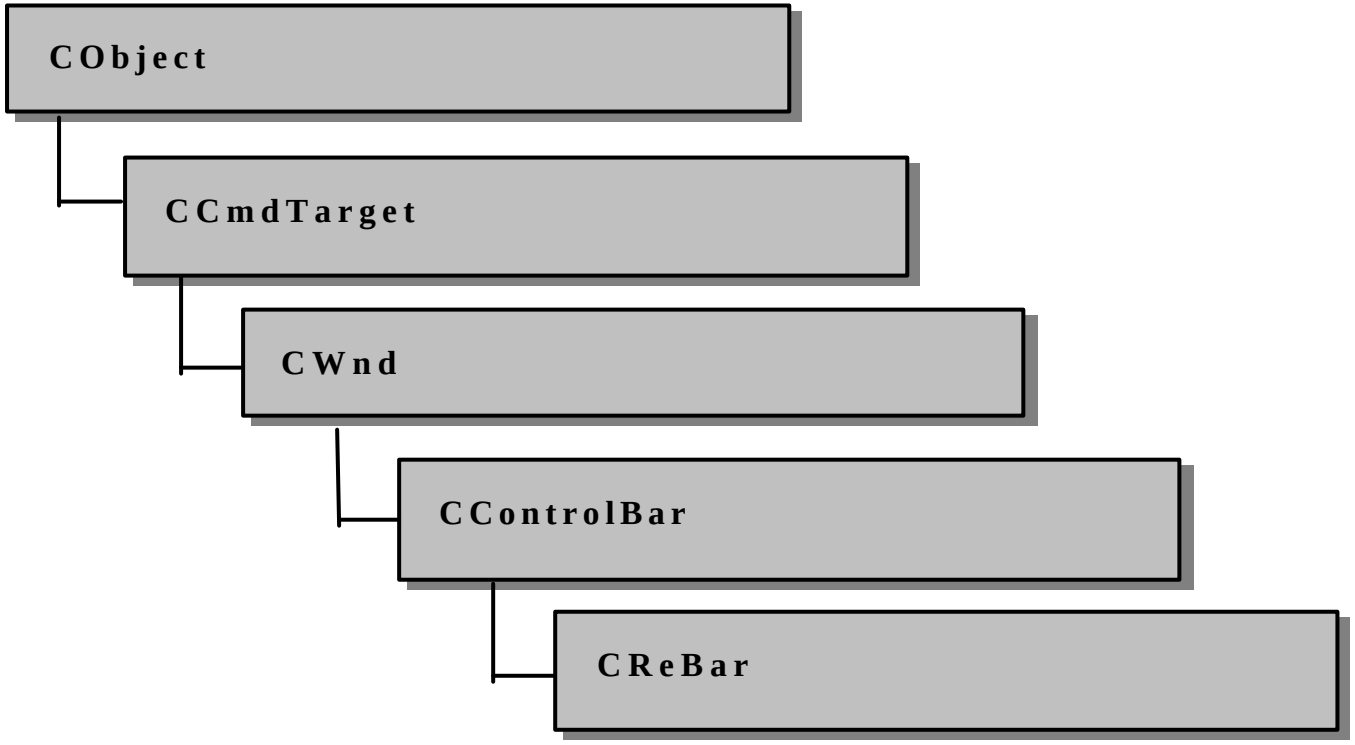
请参阅 CObList

CPtrList 类成员

Construction	
CPtrList	构造一个空的 void 指针列表
Head/Tail Access	
GetHead	返回列表（不能为空）的头元素
GetTail	返回列表（不能为空）的尾元素
Operations	
RemoveHead	移走列表的头元素
RemoveTail	移走列表的尾元素
AddHead	将一个元素（或另一列表中的所有元素）增加到列表头（成为新的列表头）
AddTail	将一个元素（或另一列表中的所有元素）增加到列表尾（成为新的列表尾）
RemoveAll	从此列表中移走所有元素
Iteration	
GetHeadPosition	返回列表头元素的位置
GetTailPosition	返回列表尾元素的位置
GetNext	获取用于重复的下一个元素
GetPrev	获取用于重复的上一个元素

Retrieval/Modification	
GetAt	获取在给定位置的元素
SetAt	设置在给定位置的元素
RemoveAt	从此列表中移走给定位置的元素
Insertion	
InsertBefore	在一个给定位置之前插入一个新元素
InsertAfter	在一个给定位置之后插入一个新元素
Searching	
Find	获取一个由指针值指定的元素的位置
FindIndex	获取由一个索引（从零开始）指定的元素的位置
Status	
GetCount	返回此列表中的元素数目
IsEmpty	测试列表是否为空（无元素）

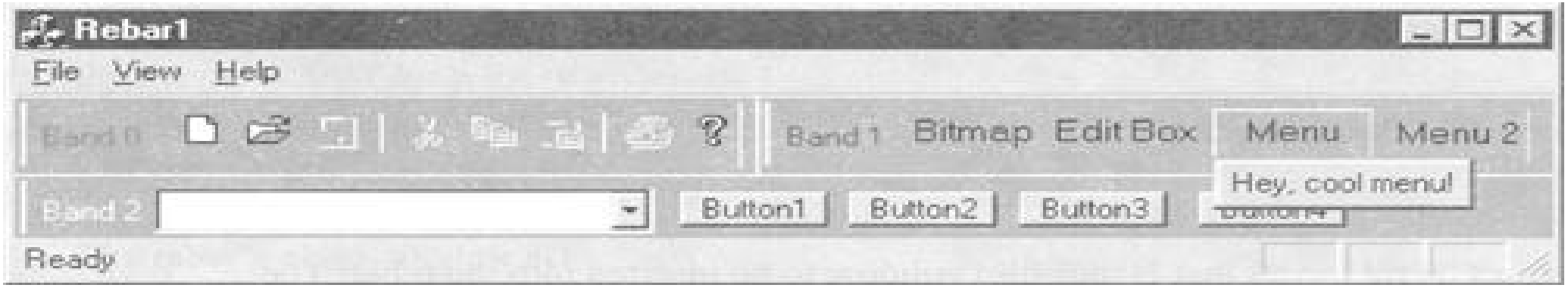
CReBar



CReBar对象是一个控制条，它为rebar控件提供页面布局、持续性和状态信息。

rebar对象可以包含多种子窗口，通常是其它控件，包括编辑框、工具条和列表

框。rebar 对象可以把它的子窗口显示在一个指定的位图上面。应用程序可以自动地改变 rebar 的大小，或者是用户通过单击或拖动 rebar 的控制条来手动改变它的大小。



Rebar 控件

rebar 对象的行为类似于一个工具条对象的行为。一个 rebar 使用单击－拖动机制来调整它的带的尺寸。一个 rebar 控件可以包含一个或多个带，每一个带可以是一个控制条、一个位图、一个文本标签和一个子窗口的任意组合。但是，

带不能包含多于一个的子窗口。

CReBar使用 CReBarCtrl类来提供它的实现。你可以通过 CReBarCtrl来访问 rebar 控件，以利用控件的定制选项。要获取有关 rebar 控件的给定选项，请参见 CReBarCtrl。

警告 Rebar 和 rebar 控件对象不支持 MFC 固定控制条。如果调用了 CReBar::EnableDocking，则你的应用程序将给出断言。

```
#include <afxext.h>
```

CReBar 类成员

Construction	
Create	创建一个 rebar 控件，并将它连接到 CReBar 对象上

Attributes	
GetReBarCtrl	允许直接访问基础通用控件
AddBar	将一个带添加到一个 rebar 中

成员函数

CReBar::AddBar

BOOL AddBar(CWnd* *pBar*, LPCTSTR *lpszText* = NULL, CBitmap* *pbmp* = NULL,
    DWORD *dwStyle* = RBBS_GRIPPERALWAYS | RBBS_FIXEDBMP);
BOOL AddBar(CWnd* *pBar*, COLORREF *clrFore*, COLORREF *clrBack*,
    LPCTSTR *pszText* = NULL, DWORD *dwStyle* = RBBS_GRIPPERALWAYS);

返回值

如果成功则返回非零值；否则返回 0。

参数

pBar

一个指向 CWnd 对象的指针，该对象将要被插入 rebar 的子窗口。被引用的对象必须有一个 WS_CHILD。

lpszText

一个指向字符串的指针，该字符串包含要显示在 rebar 上的文本。其缺省值是 NULL。*pszText* 中包含的文本并不是子窗口的一部分；它是在 rebar 本身上的。

pbmp

一个指向 CBitmap 对象的指针，该对象被显示在 rebar 的背景上。其缺省值为 NULL。

dwStyle

一个包含要应用于 rebar 的风格的 DWORD。参见 Win32 结构中的 fStyle

函数的描述，可以获得有关带风格的完整列表。

clrFore

一个 COLORREF 值，它代表了 rebar 的前景颜色。

clrBack

一个 COLORREF 值，它代表了 rebar 的背景颜色。

说明

此成员函数用来给 rebar 添加一个带。

请参阅 CReBarCtrl

CReBar::Create

```
BOOL Create( CWnd* pParentWnd, DWORD dwCtrlStyle = RBS_BANDBORDERS,
    DWORD dwStyle = WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS |
```

WS_CLIPCHILDREN | CBRS_TOP, UINT *nID* = AFX_IDW_REBAR);

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

指向 CWnd **对象的指针，该对象的** Windows **窗口是状态条的父窗口。通常是你的框架窗口。**

dwCtrlStyle

rebar **控件风格。缺省的，是** RBS_BANDBORDERS，**这种风格显示窄线来分隔** rebar **控件中相邻的带。参见“** Platform SDK**”中的“** Rebar **控件风格”，可以获得一个风格列表。**

dwStyle

rebar 窗口风格。

nID

rebar 的子窗口 ID。

说明

此成员函数用来创建一个 rebar。

请参阅 CReBarCtrl

CReBar::GetReBarCtrl

CReBarCtrl& GetReBarCtrl() const;

返回值

一个指向一个 CReBarCtrl 对象的引用。

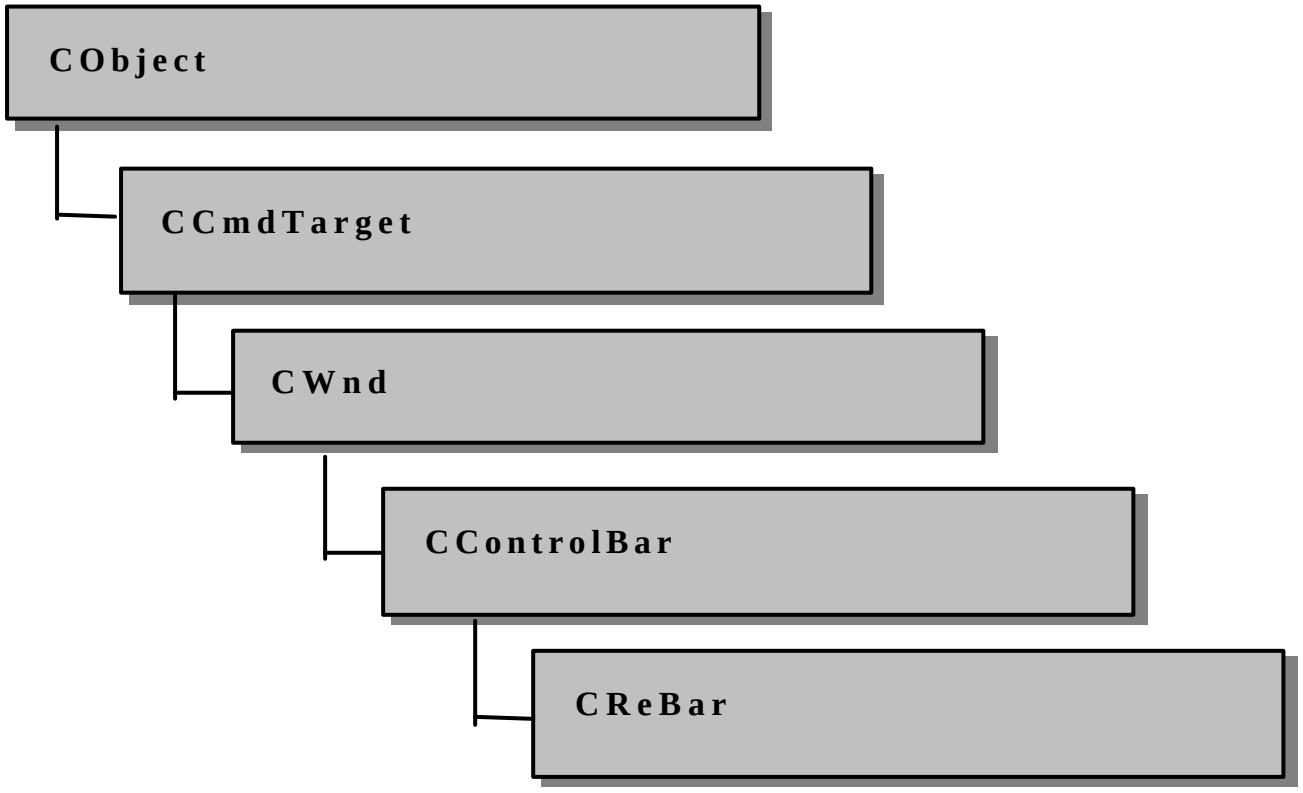
说明

此成员函数允许直接访问基础通用控件。

调用此成员函数来利用 Windows rebar 通用控件的机能来定制你的 rebar。当你调用 GetReBarCtrl 时，它返回一个指向 CReBarCtrl 对象的引用，这样你就可以使用任何一套成员函数。

如果要获取更多有关使用 CReBarCtrl 来定制你的 rebar 的信息，请参见“Visual C++程序员指南”中的“使用 CReBarCtrl”。

CReBarCtrl



CReBarCtrl 类封装了 rebar 控件的机能，rebar 控件是一个子窗口容器。rebar 控

件所属的应用程序将 rebar 控件包含的子窗口分配给 rebar 带。子窗口是另外一种通用的控件。

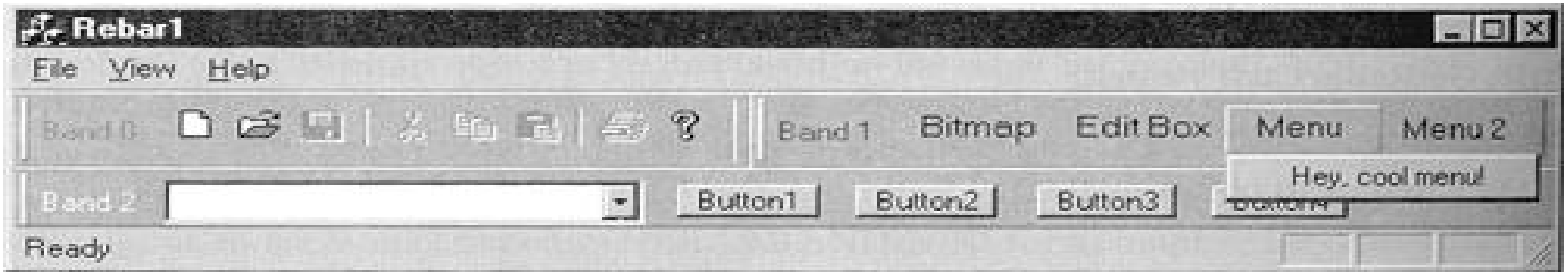
rebar 控件包含一个或多个带。每一个带可以包含一个控制条，一个位图，一个文本标签和一个子窗口的组合。但一个带只能包含每一种项的一个。

rebar 控件可以在指定的位图之上显示子窗口。所有的 rebar 控件带都可以调整大小，除了那些使用 RBBS_FIXEDSIZE 风格的以外。当你调整一个 rebar 控件带的位置和大小的时候，rebar 控件控制对应于该带的子窗口的位置和大小。单击和拖动一个带的控制条，就可以调整控件中的带的大小或改变它们的顺序。

下面的例子说明了一个具有三个带的 rebar 控件：

- 带 0 包含了一个平坦的、透明（transparent）的工具条控件。
- 带 1 包含了透明的标准按钮和下拉按钮。

- 带 2 包含了一个组合框和四个标准按钮。



Rebar 控件

Rebar 控件支持：

- 图像列表。
- 消息处理。
- 定制绘画功能。
- 除标准的窗口风格之外，附加的各种控件风格。要获取这些风格的列表，参见“ Platform SDK ”中的“ Rebar 控件风格 ”。

#include <afxcmn.h>

CReBarCtrl 类成员

Constructors	
CreBarCtrl	构造一个 CReBarCtrl 对象

Initialization	
Create	创建 rebar 控件，并将它与 CReBarCtrl 对象连接

Attributes	
GetBandCount	获取在 rebar 控件中的当前带数
GetBandInfo	获取 rebar 控件中的指定带的信息
GetBarHeight	获取 rebar 控件的高度
GetBarInfo	获取有关 rebar 控件的信息以及它使用的图像列表
GetBkColor	获取 rebar 控件的缺省背景颜色
GetDropTarget	获取一个 rebar 控件的 IDropTarget 接口指针
GetRect	获取一个 rebar 控件中的给定带的边界矩形
GetRowCount	获取一个 rebar 控件中的带行的数目
GetRowHeight	获取一个 rebar 控件中的指定行的高度

续表

GetTextColor	获取一个 rebar 控件的缺省文本颜色
GetToolTips	获取与 rebar 控件相关的任何工具提示控件的句柄
IDToIndex	将一个带的标识符（ID）转换成一个 rebar 控件中的带索引
SetBandInfo	设置一个 rebar 控件中的已存在的带的特征
SetBarInfo	设置一个 rebar 控件的特征
SetBkColor	设置一个 rebar 控件的缺省背景颜色
SetOwner	设置一个 rebar 控件的属主窗口
SetTextColor	设置一个 rebar 控件的缺省文本颜色
SetToolTips	使一个工具提示控件与 rebar 控件相关联
GetImageList	获取与 rebar 控件相关的图像列表
SetImageList	设置一个 rebar 控件的图像列表
GetBandBorders	获取一个带的边界
GetPalette	获取 rebar 控件的当前调色板
SetPalette	设置 rebar 控件的当前调色板

Operations	
BeginDrag	设置 rebar 控件进入拖－放模式
DeleteBand	从一个 rebar 控件中删除一个带
DragMove	在调用 BeginDrag 后，更新在 rebar 控件中的拖动位置
EndDrag	终止 rebar 控件的拖放操作
HitTest	如果一个 rebar 带在屏幕上的给定点存在的话，则确定 rebar 带的哪一部分是在这个点上
InsertBand	向一个 rebar 控件中插入一个新带
MaximizeBand	将 rebar 控件中的一个带调整到它的理想或最大尺寸
MinimizeBand	将 rebar 控件中的一个带调整到它的理想或最小尺寸
ShowBand	显示或隐藏一个 rebar 控件中的给定带
SizeToRect	使一个 rebar 控件符合一个指定的矩形
MoveBand	将一个带从一个索引移动到另一个索引

成员函数

CReBarCtrl::BeginDrag

void BeginDrag(UINT *uBand*, DWORD *dwPos* = (DWORD)-1);

参数

uBand

从零开始的带索引值，这些带将被拖放操作影响。

dwPos

一个 DWORD **值，包含了开始的鼠标坐标。水平坐标包含在** LOWORD **中，垂直坐标包含在** HIWORD **中。如果你传递的是（**DWORD**）**-1**，则** rebar **控件将通过调用**::GetMessage **或**::PeekMessage **来使用鼠标最后一次的位置。**

说明

此成员函数实现 Win32 **消息** RB_BEGINDRAG **的行为，就像在“** Platform SDK **”中描述的一样。**

请参阅 ::GetMessage，::PeekMessage

CReBarCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果成功地创建了对象，则返回非零值；否则返回 0。

参数

dwStyle

指定应用于控件的 rebar 控件风格的组合。参见“ Platfor SDK ”中的“ rebar 控件风格 ”，可以获得所支持的风格的列表。

rect

一个指向 CRect 对象或 RECT 结构的引用，它们是 rebar 控件的位置和尺寸。

pParentWnd

一个指向 CWnd 对象的指针。它不能是 NULL。

nID

指定 rebar 控件的控制 ID。

说明

分两步来创建一个 rebar 控件：

1. 调用 CReBarCtrl 来构造一个 CReBarCtrl 对象。
2. 调用 Create 成员函数来创建 Windows rebar 控件，并将它与 CReBarCtrl 对象连接。

当你调用 Create 时，通用的控件被初始化。

CReBarCtrl::CReBarCtrl

CReBarCtrl();

说明

此成员函数用来创建一个 CReBarCtrl 对象。

请参阅 CReBarCtrl::Create

CReBarCtrl::DeleteBand

BOOL DeleteBand(UINT *uBand*);

返回值

如果成功地删除了带，则返回非零值；否则返回 0。

参数

uBand

要被删除的带的从零开始的索引值。

说明

此成员函数用来实现 Win32 **消息** RB_DELETEBAND **的行为，就像在“** Platform

SDK”中所描述的一样。

CReBarCtrl::DragMove

void DragMove(DWORD *dwPos* = (DWORD)-1);

参数

dwPos

一个 DWORD 值，该值包含了新的鼠标位置。水平坐标包含在 LOWORD 中，垂直坐标包含在 HIWORD 中。如果你传递的是（DWORD）－1，则 rebar 控件将通过调用::GetMessage 或::PeekMessage 来使用鼠标最后一次的位置。

说明

此成员函数用来实现 Win32 消息 RB_DRAGMOVE 的行为，就像在“ Platform

SDK”中所描述的一样。

请参阅 CReBarCtrl::BeginDrag，CReBarCtrl::EndDrag，::GetMessage，::PeekMessage

CReBarCtrl::EndDrag

void EndDrag();

说明

此成员函数用来实现 Win32 消息 RB_ENDDRAG 的行为，就像在“Platform SDK”中所描述的一样。

请参阅 CReBarCtrl::BeginDrag，CReBarCtrl::DragMove

CReBarCtrl::GetBandBorders

void GetBandBorders(UINT *uBand*, LPRECT *prc*) const;

参数

uBand

将要被获取边界的带的从 0 开始的索引。

prc

一个指向 RECT **结构的指针，该结构将用来接收带的边界。如果该** rebar **控件具有** RBS_BANDBORDERS **风格，则此结构的每一个成员都将接收相似的数目，从相应的带的一面来说，这构成了边界。如果该** rebar **控件不具有** RBS_BANDBORDERS **风格，则此结构只有左边的成员接收有效的信息。要获取有关** rebar **控件风格的描述，请参见“** Platform SDK **”中的“** rebar **控件风格”。**

说明

此成员函数用来实现 Win32 **消息** RB_GETBANDBORDERS **的行为，就像在“** Platform SDK **”中所描述的一样。**

CReBarCtrl::GetBandCount

UINT GetBandCount() const;

返回值

被分配给控件的带的数目。

说明

此成员函数用来实现 Win32 **消息** RB_GETBANDCOUNT **的行为，就像在“** Platform SDK **”中所描述的一样。**

请 参 阅 CReBarCtrl::GetBandInfo ， CReBarCtrl::SetBandInfo ，
CReBarCtrl::DeleteBand，
CReBarCtrl::InsertBand，CReBarCtrl::ShowBand

CReBarCtrl::GetBandInfo

BOOL GetBandInfo(UINT *uBand*, REBARBANDINFO* *prbbi*) const;

返 回 值

如 果 成 功 则 返 回 非 零 值 ； 否 则 返 回 0。

参 数

uBand

带 的 从 零 开 始 的 索 引 ， 将 从 这 些 带 获 取 信 息 。

prbbi

一个指向 REBARBANDINFO 结构的指针，该结构用来接收带的信息。你必须在发送此消息之前将此结构的成员 cbSize 设置为 sizeof(REBARBANDINFO)，并将 fMask 成员设置为你要获取的项。

说明

此成员函数用来实现 Win32 消息 RB_GETBANDINFO 的行为，就像在“ Platform SDK ”中所描述的一样。

请参阅 CReBarCtrl::SetBandInfo ， CReBarCtrl::GetBandCount ， CReBarCtrl::DeleteBand，

CReBarCtrl::InsertBand，CReBarCtrl::ShowBand

CReBarCtrl::GetBarHeight

UINT GetBarHeight() const;

返回值

以像素表示的代表控件高度的值。

说明

此成员函数用来获取 rebar **控件的高度。**

请参阅 CReBarCtrl::GetBarInfo，CReBarCtrl::SetBarInfo

CReBarCtrl::GetBarInfo

BOOL GetBarInfo(REBARINFO* *prbi*) const;

返回值

如果成功则返回非零值；否则返回 0。

参数

prbi

一个指向 REBARINFO 结构的指针，该结构将用来获取此 rebar 控件的信息。你必须在发送这个消息之前设置此结构的 cbSize 成员为 sizeof(REBARINFO)。

说明

此成员函数用来实现 Win32 消息 RB_GETBARINFO 的行为，就像在“Platform SDK”中所描述的一样。

请参阅 CReBarCtrl:SetBarInfo

CReBarCtrl::GetBkColor

COLORREF GetBkColor() const;

返回值

返回一个 COLORREF 值，该值代表当前缺省的背景颜色。

说明

此成员函数用来实现 Win32 消息 RB_GETBKCOLOR 的行为，就像在“ Platform SDK ”中所描述的一样。

请参阅　CReBarCtrl::SetBkColor

CReBarCtrl::GetDropTarget

IDropTarget* GetDropTarget() const;

返回值

返回一个指向 IDropTarget 接口的指针。

说明

此成员函数用来实现 Win32 **消息** RB_GETDROPTARGET **的行为，就像在“** Platform SDK **”中所描述的一样。**

CReBarCtrl::GetImageList

CImageList* GetImageList();

返回值

返回一个指向 CImageList **对象的指针。如果没有控件的图像列表则返回** NULL**。**

说明

此成员函数用来获取与一个 rebar **控件相关的** CImageList **对象。此成员函数使用** REBARINFO **结构中的** size **和** mask **信息，就像在“** Platform SDK **”中描述**

的一样。

请参阅 CReBarCtrl::SetImageList

CReBarCtrl::GetPalette

CPalette* GetPalette() const;

返回值

返回一个指向 CPalette 对象的指针，该对象指定 rebar 控件的当前调色板。

说明

此成员函数用来获取 rebar 控件的当前调色板。

值得注意的是，此成员函数使用一个 CPalette 对象作为它的返回值，而不是一个 HPALETTE。

请参阅 CReBarCtrl::SetPalette

CReBarCtrl::GetRect

BOOL GetRect(UINT *uBand*, LPRECT *prc*) const;

返回值

如果成功则返回非零值；否则返回 0。

参数

uBand

在 rebar **控件中的带的从** 0 **开始的索引。**

prc

一个指向 RECT **结构的指针，该结构将用来接收** rebar **带的边界。**

说明

此成员函数用来实现 Win32 **消息** RB_GETRECT **的行为，就像在“** Platform SDK **”中所描述的一样。**

请参阅 CReBarCtrl::SizeToRect

CReBarCtrl::GetRowCount

UINT GetRowCount() const;

返回值

返回一个 UINT **值，它代表控件中的带行的数目。**

说明

此成员函数用来实现 Win32 **消息** RB_GETROWCOUNT **的行为，就像在“** Platform

SDK”中所描述的一样。

请参阅 CReBarCtrl::GetRowHeight

CReBarCtrl::GetRowHeight

UINT GetRowHeight(UINT *uRow*) const;

返回值

一个 UINT **值，该值代表行的以像素为单位的高度。**

参数

uRow

带的从 0 **开始的索引，此索引所代表的带的高度将被获取。**

说明

此成员函数用来实现 Win32 **消息** RB_GETROWHEIGHT **的行为，就像在“** Platform SDK **”中所描述的一样。**

请参阅 CReBarCtrl::GetRowCount

CReBarCtrl::GetTextColor

COLORREF GetTextColor() const;

返回值

返回一个 COLORREF **值，该值代表当前缺省的文本颜色。**

说明

此成员函数用来实现 Win32 **消息** RB_GETTEXTCOLOR **的行为，就像在**

“Platform SDK”中所描述的一样。

请参阅 CReBarCtrl::SetTextColor，CReBarCtrl::GetBkColor，CReBarCtrl::SetBkColor

CReBarCtrl::GetToolTips

```
CToolTipCtrl* GetToolTips() const;
```

返回值

返回一个指向 CToolTipCtrl 对象的指针。

说明

此成员函数用来实现 Win32 消息 RB_GETTOOLTIPS 的行为，就像在“Platform SDK”中所描述的一样。

值得注意的是，GetToolTips 的 MFC 实现返回一个指向 CToolTipCtrl 的指针，而不是一个 HWND。

请参阅　CReBarCtrl::SetToolTips

CReBarCtrl::HitTest

int HitTest(RBHITTESTINFO* *prbht*);

返回值

返回在给定点的带的从 0 开始的索引，如果在该指定点上没有 rebar 带，则返回 -1。

参数

prbht

是一个指向 RBHITTESTINFO 结构的指针。在发送此消息之前，此结构的 pt 成员布线在客户区坐标被初始化为指向要被测试的项。

说明

此成员函数用来实现 Win32 消息 RB_HITTEST 的行为，就像在“ Platform SDK ”中所描述的一样。

CReBarCtrl::IDToIndex

int IDToIndex(UINT *uBandID*) const;

返回值

如果成功则返回从 0 开始的带索引；否则返回-1。如果有复制的带索引存在，则返回第一个。

参数

uBandID

应用程序定义的指定带的标识符，当带被插入时，由 REBARBANDINFO **中的** wID **成员传递。**

说明

此成员函数用来实现 Win32 **消息** RB_IDTOINDEX **的行为，就像在“** Platform SDK **”中所描述的一样。**

请参阅 CReBarCtrl::InserBand

CReBarCtrl::InsertBand

BOOL InsertBand(UINT *uIndex*, REBARBANDINFO* *prbbi*);

返回值

如果成功则返回非零值；否则返回 0。

参数

uIndex

从零开始的位置索引，带将在这个位置插入。如果你设置这个参数为-1，**则控件会将新带插入在最后。**

prbbi

一个指向 REBARBANDINFO **结构的指针，此结构定义了要被插入的带。在调用此成员函数之前，你必须将此结构的** cbSize **成员设置为** sizeof(REBARBANDINFO)。

说明

此成员函数用来实现 Win32 **消息** RB_INSERTBAND **的行为，就像在“** Platform SDK **”中所描述的一样。**

CReBarCtrl::MaximizeBand

void MaximizeBand(UINT *uBand*);

参数

uBand

返回要被最大化的带的从零开始的索引值。

说明

此成员函数用来实现 Win32 **消息** RB_MAXIMIZEBAND **的行为，就像在**

“Platform SDK”中所描述的一样。

请参阅 CReBarCtrl::MinimizeBand

CReBarCtrl::MinimizeBand

void MinimizeBand(UINT *uBand*);

参数

uBand

返回要被最大化的带的从 0 **开始的索引值。**

说明

此成员函数用来实现 Win32 **消息** RB_MINIMIZEBAND **的行为，就像在“** Platform SDK”中所描述的一样。

请参阅 CReBarCtrl::MaximizeBand

CReBarCtrl::MoveBand

BOOL MoveBand(UINT *uFrom*, UINT *uTo*);

返回值

如果成功则返回非零值；否则返回 0。

参数

uFrom

要被移动的带的从 0 **开始的索引。**

uTo

新的带位置的从 0 **开始的索引。此参数的值必须不大于总带数减** 1 **的值。**

要得到总带数，可以调用 GetBandCount。

说明

此成员函数用来实现 Win32 **消息** RB_MOVEBAND **的行为，就像在“** Platform SDK **”中所描述的一样。**

CReBarCtrl::SetBandInfo

BOOL SetBandInfo(UINT *uBand*, REBARBANDINFO* *prbbi*);

返回值

如果成功则返回非零值；否则返回 0**。**

参数

uBand

要接收新设置的带的从 0 **开始的索引。**

prbbi

指向一个 REBARBANDINFO 结构的指针，此结构定义了要被插入的带。在发送此消息之前，你必须将此结构的 cbSize 成员设置为 sizeof(REBARBANDINFO)。

说明

此成员函数用来实现 Win32 消息 RB_SETBANDINFO 的行为，就像在“ Platform SDK ”中所描述的一样。

请参阅 CReBarCtrl::GetBandInfo

CReBarCtrl::SetBarInfo

BOOL SetBarInfo(REBANRINFO* *prbi*);

返回值

如果成功则返回非零值；否则返回 0。

参数

prbi

一个指向 REBARINFO **结构的指针，该结构包含了要设置的信息。在发送此消息之前，你必须将此结构的** cbSize **成员设置为** sizeof(REBARINFO)。

说明

此成员函数用来实现 Win32 **消息** RB_SETBARINFO **的行为，就像在“** Platform SDK **”中所描述的一样。**

请参阅 CReBarCtrl::GetBarInfo

CReBarCtrl::SetBkColor

COLORREF SetBkColor(COLORREF *clr*);

返回值

返回一个 COLORREF **值，它代表先前的缺省背景颜色。**

参数

clr

一个 COLORREF **值，它代表新的缺省背景颜色。**

说明

此成员函数用来实现 Win32 **消息** RB_SETBKCOLOR **的行为，就像在“** Platform

SDK”中所描述的一样。在“Platform SDK”中参见此主题，就可以获得更多关于何时设置背景颜色和如何设置缺省值的信息。

请参阅 CReBarCtrl::GetBkColor

CReBarCtrl::SetImageList

BOOL SetImageList(CImageList* *pImageList*);

返回值

如果成功则返回非零值；否则返回 0。

参数

pImageList

一个指向 CImageList **对象的指针，该对象包含了要分配给** rebar **控件的图**

像列表。

说明

此成员函数用来给 rebar 控件分配图像列表。

请参阅 CReBarCtrl::GetImageList

CReBarCtrl::SetOwner

CWnd* SetOwner(CWnd* *pWnd*);

返回值

返回一个指向 CWnd 对象的指针，该对象是此 rebar 控件的当前属主。

参数

pWnd

一个指向 CWnd 对象的指针，该对象要被设置为此 rebar 控件的属主。

说明

此成员函数用来实现 Win32 消息 RB_SETPARERN 的行为，就像在“Platform SDK”中所描述的一样。

值得注意的是，不管是对 rebar 控件的当前属主还是被选中的属主，都使用指向 CWnd 对象的指针，而不是窗口句柄。

注意 此成员函数并不改变在创建对象时所设置的父；它只是向你指定的窗口发送通知消息。

CReBarCtrl::SetPalette

CPalette* SetPalette(HPALETTE *hPal*);

返回值

返回一个指向 CPalette **对象的指针，该对象指定了** rebar **控件先前调色板。**

参数

hPal

一个 HPALETTE**，它指定了** rebar **控件将要使用的新的调色板。**

说明

此成员函数用来实现 Win32 **消息** RB_SETPALETTE **的行为，就像在“** Platform SDK**”中所描述的一样。**

值得注意的是，此成员函数用一个 CPalette 对象作为它的返回值，而不是一个 HPALETTE。

请参阅 CReBarCtrl::GetPalette

CReBarCtrl::SetTextColor

COLORREF SetTextColor(COLORREF *clr*);

返回值

返回 COLORREF **值，该值代表了与** CReBarCtrl **对象相关的先前文本颜色。**

参数

clr

是一个 COLORREF **值，它代表** CReBarCtrl **对象中的新的文本颜色。**

说明

此成员函数用来实现 Win32 **消息** RB_SETTEXTCOLOR **的行为，就像在“** Platform SDK **”中所描述的一样。此函数用来支持一个** rebar **控件中的文本颜色的灵活性。**

请参阅 CReBarCtrl::GetTextColor

CReBarCtrl::SetToolTips

void SetToolTips(CToolTipCtrl* *pToolTip*);

参数

pToolTip

一个指向 CToolTipCtrl **对象的指针。**

说明

此成员函数用来使一个工具提示控件和一个 rebar **控件相联系。**

当你使用完后，你必须销毁此 CRoolTipCtrl **对象。**

请参阅 CReBarCtrl::GetToolTips

CReBarCtrl::ShowBand

BOOL ShowBand(UINT *uBand*, BOOL *fShow* = TRUE) ;

返回值

如果成功则返回非零值；否则返回 0。

参数

uBand

此 rebar 控件中的一个带的从零开始的索引值。

fShow

表明是显示还是隐藏这个带。如果这个值是 TRUE，则显示此带。否则，隐藏这个带。

说明

此成员函数用来实现 Win32 消息 RB_SHOWBAND 的行为，就像在“Platform SDK”中所描述的一样。

请参阅 CReBarCtrl::SetBandInfo，CReBarCtrl::GetBandCount，CReBarCtrl::DeleteBand，CReBarCtrl::InsertBand，CReBarCtrl::GetBandInfo

CReBarCtrl::SizeToRect

BOOL SizeToRect(CRect& *rect*);

返回值

如果成功则返回非零值；否则返回 0。

参数

rect

一个指向 CRect **对象的引用，该对象指定了一个** rebar **对象必须适合的矩形。**

说明

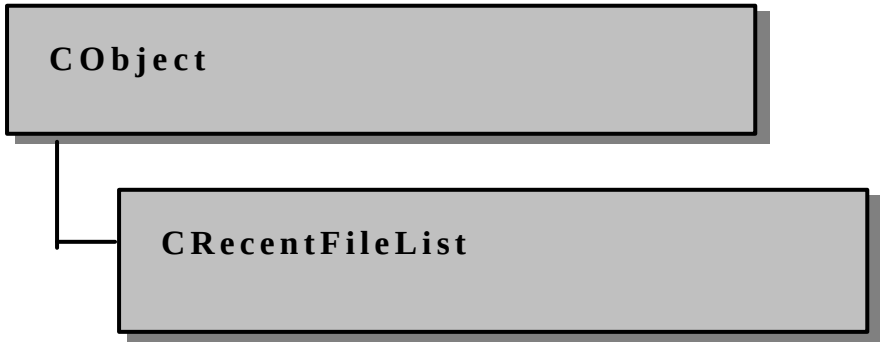
此成员函数用来实现 Win32 **消息** RB_SIZETORECT **的行为，就像在“** Platform

SDK”中所描述的一样。

值得注意的是，此成员函数使用一个 CRect 对象作为参数，而不是一个 RECT 结构。

请参阅　CReBarCtrl::GetRect

CRecentFileList



CRecentFileList 是一个 CObject 类，它支持对最近使用过（MRU）的文件列表

的控件。可以从 MRU 文件列表中添加或删除文件，该文件列表可以从注册表或一个.INI文件中读出或写入，显示 MRU 文件列表的菜单可以被更新。

#include <afxadv.h>

CRecentFileList 类成员

COnstruction	
CRecentFileList	构造一个 CrecentFileList 对象
Attributes	
GetSize	获取 MRU 文件列表中的文件数目
Operations	
Remove	从 MRU 文件列表中移走一个文件
Add	向 MRU 文件列表中添加一个文件
GetDisplayName	为一个 MRU 文件名的菜单显示提供一个显示名称
UpdateMenu	更新 MRU 文件列表的菜单显示
ReadList	从注册表或.INI文件中读出 MRU 文件列表
WriteList	将 MRU 文件列表写入注册表或.INI文件

Operators	
Operator[ ]	在一个给定位置返回一个 CString 对象

成员函数

CRecentFileList::Add

virtual void Add(LPCTSTR *lpszPathName*);

参数

lpszPathName

要添加到列表中的路径名。

说明

此成员函数用来将路径在 lpszPathName **给定的文件添加到最近使用过（**MRU**）文件列表中。此文件名将被添加到** MRU **列表的最顶端。如果此文件名在** MRU **列表中已经存在，则它将会被移动到顶端。**

请参阅 CRecentFileList::Remove，CRecentFileList::UpdateMenu

CRecentFileList::CRecentFileList

CRecentFileList(UINT *nStart*, LPCTSTR *lpszSection*, LPCTSTR *lpszEntryFormat*,
int *nSize*, int *nMaxDispLen* = AFX_ABBREV_FILENAME_LEN);

参数

nStart

在 MRU**（最近使用过的）文件列表的菜单显示中的数字偏移。**

lpszSection

指向注册表或应用程序的.INI 文件中用于 MRU 文件列表读出或写入部分的名字。

lpszEntryFormat

指向一个格式字符串，此格式字符串用来作为保存在注册表或应用程序的.INI文件中的项的名字。

nSize

MRU 文件列表中的最大文件数目。

nMaxDispLen

可用于 MRU 文件列表中的文件名的菜单显示的按字符计算的最大长度。

说明

此成员函数用来构造一个 CRecentFileList 对象。

由 *lpszEntryFormat* 指向的格式字符串应该包含“%d”，这将被用来代替每一个 MRU 项的索引。例如，如果此格式字符串是“file%d”，则这些项将被命名为 file0，file1 等。

CRecentFileList::GetDisplayName

BOOL GetDisplayName(CString& *strName*, int *nIndex*, LPCTSTR *lpszCurDir*,
int *nCurDir*, BOOL *bAtLeastName* = TRUE) conse;

返回值

如果在 MRU（最近使用过的）文件列表中的值的索引处没有文件名，则返回 FALSE。

参数

strName

文件的全路径，该文件的名字将要被显示在 MRU 文件的菜单列表中。

nIndex

在 MRU 文件列表中的文件的从零开始的索引。

lpszCurDir

包含当前目录的字符串。

bAtLeastName

如果是非零值，则表明应该返回文件的低级名字，即使是这个名字超过了最大的显示长度（作为 *nMaxDispLen* 参数传递给 CrecentFileList 构造函数）。

说明

此成员函数用来获取 MRU 文件列表中的一个文件的显示名。如果该文件在当前目录下，则函数不显示该目录。如果文件名太长了，则不显示目录和扩展名。

如果这样的文件名仍然太长，则除非 bAtLeastName 是非零值，否则将显示名置为一个空字符串。

请参阅 CRecentFileList::ReadList，CRecentFileList::WriteList

CRecentFileList::GetSize

int GetSize() const;

返回值

在当前 MRU（最近使用的）文件列表中的文件数目。

请参阅 CRecentFileList::Add，CRecentFileList::Remove

CRecentFileList::ReadList

virtual void ReadList();

说明

此函数用来从注册表或应用程序的.INI **文件中读取最近使用过的（**MRU**）文件列表。**

请参阅 CRecentFileList::WriteList

CRecentFileList::Remove

virtual void Remove(int *nIndex*);

参数

nIndex

要被从 MRU **（最近使用过的）文件列表中移走的文件的从零开始的索引。**

说明

此函数用来从 MRU **文件列表中移走一个文件。**

请参阅 CRecentFileList::Add，CRecentFileList::UpdateMent

CRecentFileList::UpdateMenu

virtual void UpdateMenu(CCmdUI* *pCmdUI*);

参数

pCmdUI

一个指向 MRU **（最近使用过的）文件列表菜单的指针，该菜单是一个** CCmdUI **对象。**

说明

此函数用来更新 MRU **文件列表的菜单显示。**

请参阅 CRecentFileList::Add，CRecentFileList::Remove

CRecentFileList::WriteList

virtual void WriteList();

说明

此函数用来将最近使用过的（MRU**）文件列表写入注册表或应用程序的**.INI **文件。**

请参阅 CRecentFileList::ReadList

操作符

CRecentFileList::operator[]

CString& operator[](int *nIndex*);

参数

nIndex

在一个 CString 集合中的一个 CStrings 的从零开始的索引。

说明

此重载的下标（[]）操作符返回一个由 nIndex 中的从零开始的索引指定的单一的 CString。

CRecordset

CRecordset 对象代表从一个数据源选择的一组记录的集合，被称作“记录集”。CRecordset 对象可以以两种形式使用：动态集和快照。动态集是与其它用户的更新保持同步的动态数据集。快照是数据的静态视图。每一种形式都代表打开记录集时固定的一组记录。但是当滚动到动态集中的一个记录时，动态集将反映后来由其它用户或由应用程序中其它记录集对此记录所做的改变。

注意 如果你正在使用数据访问对象（DAO）类，而不是打开数据库连接（ODBC）类，请使用类 CDaoRecordset 来代替。

要使用任何一种记录集，通常需要从 CRecordset 派生一个应用程序指定的记录集类。记录集从一个数据源中选择记录，然后你就可以：

- 在这些记录中滚动。
- 更新记录并指定一种加锁模式。
- 过滤记录集，以获得那些从数据源中选择出来的可利用的记录。
- 给记录集排序。
- 参数化该记录集以定制它的具有要直到运行时才知道的信息的选项。

要使用你的类，打开一个数据库并构造一个记录集对象，给构造函数传递一个指向你的 CDatabase 对象的指针。然后调用记录集的 Open 成员函数，在此你可以指定该对象是一个动态集还是一个快照。调用 Open 来从数据源中选择数据。在记录集对象被打开之后，用它的成员函数和数据成员来滚动和操作记录。可用的操作根据对象是一个动态集还是一个快照（这依赖于打开数据库连接

（ODBC）数据源的性能），是可更新的还是只读的，你是否实现了成组行检取而不同。为了刷新从调用 Open 以来可能被改变或添加的记录，可以调用对象的 Requery 成员函数。当你使用完对象之后，调用对象的 Close 成员函数，并销毁此对象。

在一个派生的 CRecordset 类中，使用记录字段交换（RFX）或成组记录字段交换（Bulk RFX）来支持读取和更新记录字段。

```
#include <afxdb.h>
```

请参阅 CDatabase，CRecordView

CRecordset 类成员

Data Members	
m_hstmt	包含记录集的 ODBC 语句句柄。类型为 HSTMT
m_nFields	包含记录集中的字段数据成员数目。类型为 UINT
m_nParams	包含记录集中参数数据成员的数目。类型为 UINT
m_pDatabase	包含一个指向 CDatabase 对象的指针，提供该指针将记录集连接到一个数据源
m_strFilter	包含一个 CString，此对象指定一条结构式查询语言（SQL）的 WHERE 子句。此成员可用作一个过滤器，只选择符合某一标准的那些记录
m_strSort	包含一个 CString，此对象指定一条 SQL ORDER BY 子句。此成员可用于控制记录的排序

Construction	
Crecordset	构造一个 CRecordset 对象。应用程序的派生类必须提供一个调用此函数的构造函数
Open	通过检取记录集表示的表格或执行查询来打开记录集
Close	关闭记录集和与此记录集相关联的 ODBC HSTMT

Recordset Attributes	
CanAppend	如果新记录可以通过 AddNew 成员函数增加到记录集中，则该函数返回一个非零值
CanBookmark	如果记录集支持书签则函数返回一个非零值
CanRestart	如果可以调用 Requery 来再次运行记录集的查询，则该函数返回一个非零值
CanScroll	如果应用程序可以滚动记录，则该函数返回一个非零值
CanTransact	如果数据源支持事务，则该函数返回一个非零值
CanUpdate	如果记录集可修改（应用程序可以增加、修改或删除记录），则该函数返回一个非零值
GetODBCFieldCount	返回记录集中的字段数目
GetRecordCount	返回记录集中的记录数目
GetStatus	获取记录集的状态：读取记录的索引，以及是否已获取到记录的最终计数
GetTableName	获取此记录集基于的表的名字
GetSQL	获取用于选择记录集的记录的 SQL 字符串
IsOpen	如果前面已经调用了 Open 函数，则此函数返回一个非零值

续表

IsBOF	如果记录集已经定位在第一个记录前，则此函数返回一个非零值
IsEOF	如果记录集已经定位在最后一个记录后，则此函数返回一个非零值
IsDeleted	如果记录集定位在一个已删除的记录上，则该函数返回一个非零值

Recordset Operations	Update
AddNew	为增加新记录作准备。调用 Update 来完成增加
CancelUpdate	取消任何用 AddNew 或 Edit 操作指定的未决定的更新
Delete	从记录集中删除当前记录。删除之后，应用程序必须显式地滚动到另一个记录
Edit	为改变当前记录作准备。调用 Update 来完成编辑
Update	通过将新数据或所编辑的数据保存到数据源上，来完成一次 AddNew 或 Edit 操作

Recordset Navigation Operations	
GetBookMark	将一个记录的标签值分配给该参数对象
Move	将记录集双向定位到距离当前记录指定数目的记录的位置
MoveFirst	定位当前记录为记录集中的第一个记录。该函数首先测试 IsBOF
MoveLast	定位当前记录为记录集中的最后一个记录。该函数首先测试 IsEOF
MoveNext	定位当前记录为记录集中的下一个记录。该函数首先测试 IsEOF
MovePrev	定位当前记录为记录集中的第一个记录。该函数首先测试 IsBOF
SetAbsolutePosition	将记录集定位到与指定的记录数相对应的位置
SetBookmark	定位记录集到书签指定的位置

Other Recordset Operations	
Cancel	取消一次异步操作或一次来自第二线程的处理
FlushResultSet	当使用一个预定义的查询时，如果有另外一个结果被获取，返回非零值
GetFieldValue	返回记录集中的一个字段的值
GetODBCFieldInfo	返回记录集中各字段的指定类别的信息
GetRowsetSize	返回在一次单个获取中你要获取的记录数目
GetRowsFetched	返回在一次获取中实际获取的行数
GetRowStatus	返回在一次获取中行的状态
IsFieldDirty	如果在当前记录中的指定字段被改变，则返回一个非零值
IsFieldNull	如果当前记录中的指定字段是 Null（没有值），则返回非零值
IsFieldNullable	如果当前记录中的指定字段可被设置为 Null（没有值），则返回非零值
RefreshRowset	刷新指定行的数据和状态
Requery	再次运行记录集的查询来刷新所选择的记录
SetFieldDirty	标记当前记录中的指定字段是被改变的

续表

SetFieldNull	设置当前记录中的指定字段的值为 Null（没有值）
SetLockingMode	将加锁模式设置为“乐观”加锁（缺省值）或“悲观”加锁。确定任何更新加锁记录
SetParamNull	将指定的参数设置为 Null（没有值）
SetRowsetCursorPosition	将游标定位在记录集中的指定行上

Recordset Overridables

Check	用来检查从一个 ODBC API 函数返回的代码
CheckRowSetError	用来处理在获取记录期间产生的错误
DoBulkFieldExchange	用来将一组数据行从数据源中交换到记录集中。实现成组记录交换（Bulk RFX）
DoFieldExchange	用来在此记录集的字段数据成员和数据源上对应的记录之间交换数据（双向）。双向记录字段交换（RFX）
GetDefaultConnect	用来获取缺省的字符串
GetDefaultSQL	用来获取要执行的缺省的 SQL 字符串
OnSetOptions	用来为指定的 ODBC 语句设置选项
SetRowsetSize	指定在一次获取中你希望获取的记录数目

成员函数

CRecordset::AddNew

```
virtual void AddNew();
    throw( CDBException );
```

说明

此函数用来为向表中增加一个新记录作准备。你必须调用 Requery 成员函数来查看刚增加的记录。该记录的字段初始为 Null（在数据库术语中，Null 意味着“没有值”，与 C++中的 NULL 不一样）。要完成增加操作，你必须调用 Update 成员函数。Update 将应用程序的改变保存到数据源中。

注意 如果你已经实现了成组读取，则你不能调用 AddNow。这将导致失败的断言。虽然类 CRecordset 不提供用于更新成组数据行的机制，但是

你可以使用 ODBC API 函数 SQLSetPos 来编写你自己的函数。有关如何做到这一点的例子，请参见 DBEFETCH 示例。

AddNew 利用记录集的字段数据成员准备了一个新的空记录。在应用程序调用 AddNew 之后，你可以在记录集的字段数据成员中设置所想要的值（应用程序不必为此调用 Edit 函数；Edir 仅用于已有记录）。当应用程序后来调用 Update 函数时，此字段数据成员中已改变的数据将被保存到数据源上。

警告　如果你在调用 Update 之前滚动到一个新的记录，则该新记录丢失，并且不会给出警告。

如果数据源支持事务，则应用程序可以使 AddNew 调用成为一次事务的一部分。要获取更多有关事务的信息，请参见类 CDatabase。值得注意的是，在调用 AddNew 之前你必须调用 CDatabase::BeginTrans。

重点 对于动态集，新记录增加到记录集中作为最后一个记录。新增加的记录不增加到快照中---应用程序必须调用 Requery 函数来刷新此记录集。

对一个未调用其 Open 函数的记录集，调用 AddNew 函数是不合法的。如果应用程序对一个不可向其插入新记录的记录集调用 AddNew 函数，则将抛出一个 CDBException 异常。应用程序可通过调用 CanAppend 函数来确定此记录集是否可以更新。

请参阅 CRecordset::Edit，CRecordset::Delete，CRecordset::Update，CRecordset::Requery，CDatabase::BeginTrans，CDBException

CRecordset::CanAppend

```
BOOL CanAppend() const;
```

返回值

如果此记录集允许增加新记录，则该函数返回一个非零值；否则返回值为 0。

如果应用程序以只读方式打开此记录集，则 CanAppend()将返回 0。

说明

此成员函数用来确定先前打开的记录集是否允许应用程序通过调用 AddNew 成员函数来增加新记录。

请参阅 CRecordset::AddNew，CRecordset::Requery

CRecordset::CanBookmark

BOOL CanBookmark() const;

返回值

如果记录集支持书签则返回非零值；否则返回 0。

说明

此成员函数用来确定此记录集是否允许用书签来标记记录。此函数独立于 Open 成员函数的 dwOptions 参数中 CRecordset::useBookmarks 选项。CanBookmark 表明给定的 ODBC 驱动器和游标类型是否支持书签。

如果支持书签，CRecordset::useBookmark 表明书签是否会是有效的。

注意　单正向记录集不支持书签。

请参阅　CRecordset::GetBookmark，CRecordset::SetBookmark

CRecordset::Cancel

void Cancel();

说明

此成员函数用来请求数据源取消一次正在进行的异步操作或者是一次来自第二线程的处理。值得注意的是，MFC ODBC **类不再使用异步处理；要执行一次异步操作，你必须直接调用** ODBC API **函数** SQLSetConnectOption**。要获取更多的信息，请参见“**ODBC SDK **程序员指南”中的“异步执行函数”。**

CRecordset::CancelUpdate

void CancelUpdate();

说明

此成员函数用来在调用 Update 之前，取消任何由 Edit 或 AddNew 引起的未决定的更新。

> 注意　对于使用成组行检取的记录集，此成员函数不能使用，因为这样的记录集不能调用 Edit, AddNew 或 Update。

如果能够进行自动检查脏字段，则 CancelUpdate 将把成员变量的值恢复到调用 Edit 或 AddNew 之前的值；如果不支持自动检查，则改变后的任何值都被保留。缺省的，当记录集被打开时，自动字段检查是有效的。为了使它无效，你必须指定 Open 成员函数的 dwOption 参数中的 CRecordset::noDirtyFieldCheck。

请参阅　CRecordset::AddNew，CRecordset::Edit，CRecordset::Update

CRecordset::CanRestart

BOOL CanRestart() const;

返回值

如果允许查询则返回非零值；否则返回 0。

说明

此成员函数用来确定记录集是否允许通过调用 Requery **成员函数来重新开始它的查询（刷新它的记录）。**

请参阅 CRecordset::Requery

CRecordset::CanScroll

BOOL CanScroll() const;

返回值

如果记录集允许滚动则返回非零值；否则返回 0。

说明

此成员函数用来确定记录集是否允许滚动。

CRecordset::CanTransact

BOOL CanTransact() const;

返回值

如果记录集支持事务，则返回一个非零值；否则，返回值为 0。

说明

此成员函数用来确定此记录集是否支持事务。

请参阅 CDatabase::BeginTrans，CDatabase::CommitTrans，CDatabase::Rollback

CRecordset::CanUpdate

BOOL CanUpdate() const;

返回值

如果记录集可更新，则返回一个非零值；否则，返回值为 0。

说明

此成员函数用来确定此记录集是否可更新。

如果基本数据资源只读或如果当你打开记录集在 *dwOptions* 参数中指定 CRecordset:: readOnly 时，一个记录集可能为只读。

请参阅 CRecordset::Open，CRecordset::AddNew，CRecordset::Edit，CRecordset::Delete，CRecordset::Update

CRecordset::Check

virtual BOOL Check(RETCODE *nRetCode*) const;

返回值

如果 *nRetCode* 的值是 SQL_SUCCESS，SQL_SUCCESS_WITH_INFO，SQL_NO_DATA_ FOUND 或 SQL_NEED_DATA，则返回非零值；否则返回 0。

参数

nRetCode

是一个 ODBC API **函数的返回代码。有关的细节，请参见说明部分。**

说明

此成员函数用来检查从一个 ODBC API **函数返回的代码。下面的列表给出了** *nRetCode* **可能的值：**

NretCode	Description
SQL_SUCCESS	**函数成功完成；没有其它的可用信息**
SQL_SUCCESS_WITH_INFO	**函数成功完成，可能有一个非致命的错误。通过调用** SQLError **可以获得更多的信息**
SQL_NO_DATA_FOUND	**获取了来自结果集的所有行**
SQL_ERROR	**函数失败。调用** SQLError **可以获得更多的信息**

续表

SQL_INVALID_HANDLE	函数失败，这是由于一个无效的环境句柄、连接句柄或语句句柄。这表明有一个程序错误。从 SQLError 无法获得其它的信息
SQL_STILL_EXECUTING	一个异步开始的函数仍然在执行。值得注意的是，缺省的 MFC 从来都不会将这个值传递给 Check，因为 MFC 只使用同步处理
SQL_NEED_DATA	当处理一个语句时，驱动器决定应用程序需要发送参数数据值

如果要获取更多关于 SQLError 的信息，请参见“ODBC SDK 程序员指南。”

示例

参见宏 AFX_ODBC_CALL。

请参阅 AFX_ODBC_CALL

CRecordset::CheckRowsetError

virtual void CheckRowsetError(RETCODE *nRecCode*)；

参数

nRetCode

一个 ODBC API **函数返回代码。要获取细节，请参见说明部分。**

说明

此虚成员函数用来处理当记录被获取时发生的错误，在成组行处理中它也适用。你可以重载 CheckRowsetError **来执行你自己的错误处理。**

在一次游标导航操作（如 Open，Requery **或是任何** Move **操作）中，** CheckRowsetError **是被自动调用的。它被作为** ODBC API **函数** SQLEetendedFetch **的返回值传递。下面的列表给出了** nRetCode **参数的可能取值：**

NretCode	Description
SQL_SUCCESS	函数成功完成；没有其它的可用信息
SQL_SUCCESS_WITH_INFO	函数成功完成，可能有一个非致命的错误。通过调用 SQLError 可以获得更多的信息
SQL_NO_DATA_FOUND	获取了来自结果集的所有行
SQL_ERROR	函数失败。调用 SQLError 可以获得更多的信息
SQL_INVALID_HANDLE	函数失败，这是由于一个无效的环境句柄、连接句柄或语句句柄。这表明有一个程序错误。从 SQLError 无法获得其它的信息
SQL_STILL_EXECUTING	一个异步开始的函数仍然在执行。值得注意的是，缺省的，MFC 从来都不会将这个值传递给 CheckRowsetError，MFC 将继续调用 SQLEetendedFetch，直到它不再返回 SQL_STILL_EXECUTING

请参阅 CRecordset::DoBulkFieldExchange，CRecordset::GetRowsetSize，

CRecordset::SetRowsetSize，CRecordset::Move

CRecordset::Close

virual void Close();

说明

此成员函数用来关闭记录集。ODBC HSTMT **和框架分配给这个记录集的内存都将被释放。通常在调用** Close **之后，如果此** C++**记录集对象是用** new **分配的，则由应用程序来负责删除它。**

在调用 Close **之后，应用程序可再次调用** Open **函数。这使应用程序可复用此对象。另一方式是调用** Requery**。**

示例

```
//    CRecordset::Close 的示例
//    构造一个快照对象
```

```
CCutSet rsCustSet( NULL );
if ( !rsCustSet.Open( ) )
   return FALSE;
//   适用快照…

//   关闭快照

rsCustSet.Clost( ) ;
//   当函数退出时，调用析构函数
```

请参阅　CRecordset:: CRecordset，CRecordset::Open，CRecordset::Requery

CRecordset::CRecordset

CRecordset(CDatabase* *pDatabase* = NULL);

参数

pDatabase

包含一个执行 CDatabase **对象的指针，或者是值为** NULL。**如果不是**

NULL，并且还没有调用 CDatabase 对象的 Open 成员函数来将它连接到数据源上，则记录集试图在它自己的 Open 调用期间为应用程序打开此 CDatabase 对象。如果此参数为 NULL，则当应用程序利用 ClassWizard 来派生自己的记录集类时，将使用应用程序所指定的数据源信息来构造并连接一个 CDatabase 对象。

说明

此成员函数用来构造一个 CRecordset 对象。应用程序的记录集对象必须是从 CRecordset 派生的应用程序专用类的对象。可以利用 ClassWizard 来派生应用程序自己的记录集类。

注意　应用程序的派生类必须提供它自己的构造函数。在此构造函数中，需要传递适当的参数来调用构造函数 CRecordset::CRecordset。

传递 NULL 参数给你的记录集构造函数，可以为应用程序自动构造并连接一个 CDatabase 对象。这是一种有用的简捷方式，它不需要应用程序在构造自己的记录集之前构造并连接一个 CDatabase 对象。

请参阅 CRecordset::Open，CRecordset::Close

CRecordset::Delete

```
virtual void Delete();
throw(CDBException);
```

说明

此成员函数用来删除当前记录。在成功删除之后，此记录集的字段数据成员被设置为 NULL 值，并且应用程序必须显式地调用一个 Move 函数来移走所删除的记录。一旦移走了被删除的记录，就不能再恢复它。如果数据源支持事务，

则可以使 Delete 调用成为一次事务的一部分。

> **注意** 如果你已经实现了成组读取，则不能调用 Delete。这会导致一个失败断言。虽然类 CRecordset 不提供用于更新成组数据行的机制，但是你可以使用 ODBC API 函数 SQLSetPos 来编写你自己的函数。有关如何做到这一点的例子，请参见 DBEFETCH 示例。

> **警告** 当你调用 Delete 时，记录集必须被更新，并且当前在记录集中必须有有效的记录；否则将发生错误。例如，如果你删除了一个记录，但是在你再次调用 Delete 之前没有滚动到一个新的记录，则 Delete 会抛出一个 CDBException。

与 AddNew 和 Edit 不同，Delete 调用后不跟 Update 调用。如果 Delete 调用失败，字段数据成员将保持不变。

示例

这个例子说明了如何在一个函数框架上创建一个记录集。该例子假定 m_dbCust 存在，一个 CDatabase 类型的成员变量已经连接到数据源上。

```
//  创建一个派生的 CRecordset 对象
CCustSet rsCustSet( &m_dbCust );
rsCustSet.Open( );
if ( rsCustSet.IsEOF( ) || !rsCustSet.CanUpdate( ) || !rsCustSet.CanTransact( ) )
   return ;
if( !m_dbCust.BeginTrans() )
{
   // 做某些事情来处理一个失败
}
else
{
   // 可能是滚动到一个新记录...
   // 删除当前记录
```

```
   rsCustSet.Delete( );
   //…

   //  完成这次事务的命令

   if ( <the user Confir ms the transaction> )
      m_dbCust.CommitTrans( );
   else      //  用户改变了主意

      .m_dbCust.Rollback( );
}
//   …
```

请参阅　Database::BeginTrans，CDatabase::CommitTrans，CDatabase::Rollback，CDBException

CRecordset::DoBulkFieldExchange

```
virtual void DoBulkFieldExchange( CFieldExchange* pFX );
   throw( CDBException );
```

参数

pFX

一个指向一个 CFileExchange 对象的指针。框架已经设置了这个对象来为字段交换操作指定一个环境。

说明

当实现了成组行处理时，框架调用这个成员函数来将数据从数据源自动传输到你的记录集对象。DoBulkFieldExchange 还将应用程序的参数数据成员（如果有的话）与记录集选择的 SQL 语句字符串中的参数占位符绑定。

如果还没有实现成组行读取，则框架调用 DoFieldExchange。为了实现成组行读取，你必须指定 Open 成员函数中的 dwOptions 参数的 CRecordset::useMultiRowFetch 选项。

注意 只有在你正使用一个从 CRecordset 派生来的类时，DoBulkFieldExchange 才是可用的。如果你已经直接从 CRecordset 创建了一个记录集对象，则你必须调用 GetFieldValue 成员函数来获取数据。

成组记录字段交换（Bulk RFX）与记录字段交换（RFX）类似。数据被自动地从数据源传输到一个记录集对象。但是，你不能调用 AddNew，Edit，Delete 或 Update 来将改变传输回数据源。目前类 CRecordset 不提供用于更新数据成组行的机制；但是，你可以用 ODBC API 函数 SQLSetPos 来编写你自己的函数。

值得注意的是，ClassWizard 不支持成组记录字段交换（RFX）；因此，你必须通过调用 Bulk RFX 函数来手动重载 DoBulkFieldExchange。如果要获取更多关于这些函数的信息，请参见“Visual C++联机文档”中的“记录字段交换函数”。

如果要得到一个有关如何实现成组记录字段交换的例子，请参见 DBFETCH 示例。要获取更多关于成组行读取的信息，请参见“Visual C++程序员指南”中

的文章“记录集：按组读取记录”（ODBC）。

请参阅 CRecordset::m_nFields，CRecordset::m_nParams，

CRecordset::DoFieldExchange ， CRecordset::GetFieldValue ，

CFieldExchange，

Record Field Exchange Functions

CRecordset::DoFieldExchange

virtual void DoFieldExchange(CFieldExchange* *pFX*);
throw(CDBException);

参数

pFX

是一个指向 CFielsExchange 对象的指针。框架已经为字段交换操作设置了这个对象来指定一个环境。

说明

当还没有实现成组行读取时，框架调用这个成员函数来在记录集的字段数据成员和数据源的当前记录的相应列之间自动进行数据交换。DoFieldExchange 还将应用程序的参数数据成员（如果有的话）与记录集选择的 SQL 语句字符串中的参数占位符绑定。

如果已经实现了成组行读取，则框架调用 DoBulkFieldExchange。为了实现成组行读取，你必须指定 Open 成员函数中的 dwOptions 的 CRecordset::useMultiRowFetch 选项。

> **注意** 只有在你正使用一个从 CRecordset 派生来的类时 DoBulkFieldExchange 才是有效的。如果你已经直接从 CRecordset 创建了一个记录集对象，你必须调用 GetFieldValue 成员函数来检取数据。

字段数据的交换，被称为记录字段交换（RFX），可以双向工作：从记录集对象的字段数据成员到数据源的记录的字段，或者从数据源中的记录到记录集对象。

应用程序为实现应用程序派生记录集类的 DoFieldExchange 函数，一般必须执行的唯一动作是用 ClassWizard 创建此派生记录集类，并指定字段数据成员的名字和数据类型。应用程序也可以增加代码到 ClassWizard 编写的代码中，以指定参数数据成员或处理应用程序动态连接的任何列。

当你用 ClassWizard 声明你的记录集类的时候，该向导会为你重载 DoFieldExchange，这类似于下面的例子：

```
void CCustSet::DoFieldExchange( CFieldExchange* pFX )
{
   // { { AFX_FIELD_MAP( CCustSet )
   PFX->SetFieldType( CFieldExchange::outputDColum );
   RFX_Text( pFX, “Name”, m_strName );
```

```
   RFX_Int( pFX, “Age”, m_wAge );
   //} }AFX_FIELD_MAP
}
```

要获取更多有关 RFX **函数的信息，请参见“** Visual C++**联机文档”中的文章“记录字段交换：**RFX **是如何工作的”。**

请参阅 CRecordset::m_nFields，CRecordset::m_nParams，CRecordset::DoBulkfieldExchange，CRecordset::GetFieldValue，CFieldExchange，Record Field Exchange Functions

CRecordset::Edit

```
virtual void Edit();
   throw( CDBException, CMemoryException );
```

说明

此成员函数用来支持对当前记录的修改。当应用程序调用 Edit 之后，就可以通过直接重新设置字段数据成员的值来改变它们。当后来应用程序重新调用 Update 成员函数来将这些改变保存到数据源上时，这次操作才算完成。

> 注意　如果你已经实现了成组行读取，你就不能调用 Edit。这将导致一个失败断言。虽然类 CRecordset 不提供用于更新成组数据行的机制，但是你可以使用 ODBC API 函数 SQLSetPos 来编写你自己的函数。有关如何做到这一点的例子，请参见 DBEFETCH 示例。

Edit 保存记录集的数据成员的值。如果你调用了 Edit，进行了改变，然后再调用 Edit，则记录的值被恢复到第一次调用 Edit 之前的值。

在某些情况下，你可能会想通过将一个列置为 Null 来更新它。要实现这一点，

可以使用 TRUE 为参数调用 SetFieldNull 函数，来标记此字段为 Null；这样也使此字段表列被更新。如果你想要将某一字段写入数据源，即使是它的值没有改变，则可以使用 TRUE 为参数调用 SetFieldDirty 函数。即使此字段的值为 Null，这也同样可行。

如果数据源支持事务，则应用程序可以让 Edit 成为事务的一部分。值得注意的是，你应当在调用 Edit 之前，但在打开记录集之后，调用 CDatabase::BeginTrans 函数。还要注意，调用 CDatabase::CommitTrans 并不能代替调用 Update 来完成 Edit 操作。要获取有关事务的更多信息，请参见 CDatabase 类。

根据当前的加锁方式，正被更新的记录可能被 Edit 加锁，直到应用程序调用 Update 或滚动到另一个记录；也可能它只在 Edit 调用期间被加锁。你可以利用 SetLocking Mode 来改变加锁方式。

如果应用程序在调用 Update 之前滚动到了一个新的记录，则恢复当前记录的先

前值。如果对一个不可修改的记录集调用 Edit，或没有当前记录，则将抛出一个 CDBException 异常。

示例

```
// CRecordset::Edit 的示例
// 编辑一个记录
// 首先设置编辑缓存
// rsCustSet.Edit();
// 然后编辑记录的字段数据成员
rsCustSet.m_dwCustID = 2795;
rsCustSet.m_strCustomer = “Jones Mfg”;
// 最后，完成此操作
if ( !rsCustSet.Update() )
   //  处理更新失败
```

请参阅 CRecordset::Update，CRecordset::AddNew，CRecordset::Delete，CRecordset::SetFieldDirty，CRecordset::SetFieldNull，CRecordset::CanUpdate，CRecordset::CanTransact，CRecordset::SetLockingMode

CRecordset::FlushResultSet

```
BOOL FlushResultSet() const;
   throw( CDBException );
```

返回值

如果获取了多个结果集，则返回非零值；否则返回 0。

说明

此成员函数用来获取一个预定义的查询（被保存的过程）的下一个结果集，如

果有多个结果集的话。只有在你完全完成了对在当前结果集上的游标的处理时，你才应该调用 FlushResultSet。值得注意的是，当你通过调用 FlushResultSet 获取下一个结果集时，则在那一个结果集上的游标就无效了；你应当在调用 FlushResultSet 之后调用 MoveNext 成员函数。

如果一个预定义的查询使用了输出参数或者是输入/输出参数，则为了得到这些参数的值，你必须一直调用 FlushResultSet 直到它返回 FALSE（值 0）。

FlushResultSet 调用了 ODBC API 函数 SQLMoreResults。如果 SQLMoreResults 返回 SQL_ERROR 或 SQL_INVALID_HANDLE，则 FlushResultSet 将抛出一个异常。要获取有关 SQLMoreResults 的更多信息，请参见“ODBC SDK 程序员参考”。

示例

下面的例子假定 COutParamRecordset **是一个基于预定义查询的** Crecordset **的派生对象，该查询具有一个输入参数和一个输出参数，并且有多个结果集。注意** DoFieldExchange **重载的结构。**

```
//    DoFieldExchange 重载
//
//    只有在处理参数装订时才有必要。
//    不要使用有边界字段的 CRecordset 派生类，
//    除非所有的结果集具有相同的概要
      //或者是有条件的装订代码。
void COutParamRecordset::DoFieldExchange( CFieldExchange* pFX );
{
   pFX_>SetFieldType( CFieldExchange::outputParam );
   RFX_Long( pFX, “Paraml”, m_nOutParamInstructorCount );
```

```
        //  在此“Paraml”名字是一个从未使用过的假名。
pFX->SetFieldType( CFieldExchange::inputParam );
RFX_Text( pFX, “Param2”, m_strInParamName );
}
//  现在实现 COurParamRecordset。
//  假定 db 是一个已经打开的数据库对象。
COutParamRecordset rs( &db );
rs.m_strInParamName = _T(“Some_Input_Param_value”);
//  获取第一个结果集
//  注意：对于返回多行集合的存储过程来说，SQL Server 需要只能前进类型
的游标。
rs.Open(CRecordset::forwardOnly,”
        {? = CALL GetCourses( ? ) }”,
        CRecordset::readOnly );

//  在第一个结果集中循环通过所有的数据
```

```
while( !rs.IsEOF( ) )
{
   CString strFieldValue;
   for( int nIndex = 0 ;
      nIndex < rs.GetODBCFieldCount( );
      nIndex++ )
   {
      rs.GetFieldValue( nIndex, strFieldValue );

      //  TO DO：使用字段值字符串

   }
   rs.MoveNext( );
}
//  获取其它的结果集...
while ( rs.FlushResultSet( ) )
{
   //  由于游标无效，必须定义 MoveNext
   rs.MoveNext( );
   while( !rs.IsEOF( ) )
   {
```

```
        CString strFieldValue;
        for( int nIndex = 0;
           nIndex < rs.GetODBCFieldCount( ) ;
           nIndex++ )
        {
         rs.GetFieldValue( nIndex, strFieldValue );
         TO DO：使用的字段值字符串。
         }
         rs.MoveNext( );
    }
}
//  所有的结果集都满了。不能使用该游标了，
//  但是已经写入了输出参数 m_nOutParamInstructorCount。
//  注意，要直到 CRecordset::FlushResultSet 返回 FALSE，
//  这表明不会再返回更多的结果集，这时 m_nOutParamInstructorCount 才无
效。
//  TO DO：使用 m_nOutParamInstructorCount
```

```
//  清除
rs.Close( );
db.Close( );
```

请参阅 CFieldExchange::SetFieldType

CRecordset::GetBookmark

```
void GetBookmark( CDBVariant& varBookmark );
   throw( CDBException, CMemoryException );
```

参数

varBookmark

一个指向 CDBVariant **对象的引用，该对象代表在当前记录上的书签。**

说明

此成员函数用来获得当前记录的书签值。要确定此记录集是否支持书签，调用

CanBookmark。如果支持书签，应用程序要使书签有效，就必须设置 Open 成员函数中的 *dwOptions* 的 CRecordset::useBookmarks 选项。

> **注意** 如果不支持书签或者是书签无效，调用 GetBookmark 将导致抛出一个异常。在只向前的记录集中是不支持书签的。

GetBookmark 将当前记录的书签值分配给一个 CDBVariant 对象。在移动到另一个记录之后，要返回原来的记录，可以用相应的 CDBVariant 对象来调用 SetBookmark。

> **注意** 在进行了一定的记录集操作后，书签也许就不再有效了。例如，如果你在调用 GetBookmark 之后接着调用了 Requery，你也许就不能再用 SetBookmark 返回原来的记录了。调用 CDatabase::GetBookmarkPersistence 来检查你是否能够安全地调用 SetBookmark。

请参阅 CRecordset::CanBookmark，CRecordset::SetBookmark，CDatabase::GetBookmarkPersistence

CRecordset::GetDefaultConnect

virtual CString GetDefaultConnect( );

返回值

返回一个包含缺省连接字符串的 CString。

说明

框架调用此成员函数来获取此记录集所基于的数据源的缺省连接字符串。 ClassWizard **通过在** ClassWizard **中标识与应用程序所使用的相同数据源来获取有关表和列的信息，为应用程序实现** GetDefaultConnect **函数。你可能会发现依赖这样的缺省连接来开发应用程序是很方便的。但是缺省连接可能不适合于你**

的用户。如果是这种情况，你就应该重新实现 GetDefaultConnect，放弃 ClassWizard 的版本。

CRecordset::GetDefaultSQL

virtual CString GetDefaultSQL( );

返回值

返回一个包含缺省的 SQL 语句的 CString。

说明

框架调用此成员函数来获取此记录集基于的缺省的 SQL 语句。它可能是一个表名，也可能是一条 SQL SELECT 语句。

你可以通过使用 ClassWizard 来声明自己的记录集类，来间接地定义缺省的 SQL

语句。ClassWizard 会为你实现这项工作。

如果你需要自己的 SQL 语句串，就调用 GetSQL，它返回用来选择记录集的记录的 SQL 语句，如果记录集是打开的话。你可以在你的类的 GetDefaultSQL 重载中编辑你的 SQL 字符串。例如，你可以使用一个 CALL 语句来指定对一个预定义的查询的调用。

> 警告　如果框架不能标识一张表的名字或提供了多个表名，亦或不能插入一个 CALL 语句，则表名将是空的。注意，当使用 CALL 语句时，你不能在弯括号和 CALL 关键字之间插入空格，也不能在弯括号之前或者是 SELECT 语句中的 SELECT 关键字之前插入空格。

请参阅　CRecordset::GetSQL

CRecordset::GetFieldValue

```
void GetFieldValue ( LPCTSTR lpszName, CDBVariant& varValue,
   short nFieldType = DEFAULT_FIELD_TYPE );
   throw( CDBException, CMemoryException );
void GetFieldValue ( short nIndex, CDBVariant& varValue,
   short nFieldType = DEFAULT_FIELD_TYPE );
   throw( CDBException, CMemoryException );
void GetFieldValue ( LPCTSTR lpszName, CString& strValue );
   throw( CDBException, CMemoryException );
void GetFieldValue ( short nIndex, CString& strValue );
   throw( CDBException, CMemoryException );
```

参数

lpszName

一个字段的名字。

varValue

一个指向 CDBVariant **对象的引用，该对象将用来保存字段的值。**

nFieldType

字段的 ODBC C 数据类型。缺省值 DEFAULT_FIELD_TYPE 强迫 GetFieldValue 根据下面的表格确定与 SQL 数据类型相对应的 C 数据类型。或者，你可以直接指定数据类型或选择一个兼容的数据类型；例如，你可以将任何数据类型保存到 SQL_C_CHAR 中。

C 数据类型	SQL 数据类型
SQL_C_BIT	SQL_BIT
SQL_C_UTINYINT	SQL_TINYINT
SQL_C_SSHORT	SQL_SMALLINT
SQL_C_SLONG	SQL_INTEGER
SQL_C_FLOAT	SQL_REAL
SQL_C_DOUBLE	SQL_FLOAT
	SQL_DOUBLE
SQL_C_TIMESTAMP	SQL_DATE
	SQL_TIME
	SQL_TIMESTAMP
SQL_C_CHAR	SQL_NUMERIC
	SQL_DECIMAL

续表

	SQL_BIGINT
	SQL_CHAR
	SQL_VARCHAR
	SQL_LONGVARCHAR
SQL_C_BINARY	SQL_BINARY
	SQL_VARBINARY
	SQL_LONGVARBINARY

有关 ODBC 数据类型的更多信息，参见“ODBC SDK 程序员参考”的附录 D 中的“SQL 数据类型”和“C 数据类型”。

nIndex

字段从零开始的索引。

strValue

一个指向 CString 对象的引用，该对象将把字段的值保存为文本，而不管字段的数据类型。

说明

此成员函数用来获取当前记录中的字段数据。你可以用名字或者索引来查找一个字段。也可以把字段值保存在一个 CDBVariant 对象或一个 CString 对象中。

如果你已经实现了成组行读取，则在一个行集中的当前记录总是被定位在第一个记录。要在一个给定的行集中对一个记录使用 GetFieldValue，必须首先调用 SetRowsetCursorPosition 成员函数来将游标移动到行集中所希望的行上。然后为这一行调用 GetFieldValue。要实现成组行读取，你必须指定 Open 成员函数中的 *dwOptions* 参数的 CRecordset::useMultiRow- Fetch 选项。

你可以使用 GetFieldValue 在运行时动态地读取字段，而不是在设计时静态地装订它们。例如，如果你已经直接从 CRecordset 声明了一个记录集对象，你就必须使用 GetFieldValue 来获取字段数据；记录字段交换（RFX），或者是成组记

录字段交换（Bulk RFX）还没有实现。

> **注意** 如果你声明了一个记录集对象，而不是从 CRecordset 派生，则不被装载 ODBC 游标库。此游标库要求记录集至少有一个边界列；但是，当你直接使用 CRecordset 时，没有一个列是边界。成员函数 CDatabase::OpenEx 和 CDatabase::Open 控制游标库是否被装载。

GetFieldValue 调用 ODBC API 函数 SQLGetData。如果你的驱动器为字段值的实际长度输出值 SQL_NO_TOTAL，GetFieldValue 抛出一个异常。关于 SQLGetData 的更多消息，参见“ODBC SDK 程序员参考”。

示例

下面的例子代码说明如何为一个直接从 CRecordset 声明的记录集对象调用 GetFieldValue。

```
// 创建并打开一个数据库对象；

// 不要装载游标库

CDatabase db;
db.OpenEx( NULL, CDatabase::forceOdbcDialog );
// 直接从 CRecordset 创建并打开一个记录集对象。

// 注意在一个被连接的数据库中必须存在一个表。

// 使用仅向前类型的记录集以获得最佳的性能，因为值需要 MoveNext 函数。

CRecordset rs( &db );
rs.Open( CRecordset::forwardOnly,
      _T( "SELECT * FROM SomeTable" ) );
// 创建一个 CDBVariant 对象来保存字段数据

CDBVariant varValue;
// 在记录集中循环，使用 GetFieldValue 和 GetODBCFieldCount 来获取所有列
中的数据

short nFields = rs.GetODBCFieldCount( );
while( !rs.IsEOF( ) )
```

```
{
   for( short index = 0; index < nFields; index++ )
   {
      rs.GetFieldValue( index, varValue );
      // 对 varValue 作某些操作。
   }
   rs.MoveNext( );
}
rs.Close( );
db.Close( );
```

注意 与 DAO 类 CDaoRecordset 不一样，CRecordset 没有 SetFieldValue 成员函数。如果你直接从 CRecordset 创建了一个对象，它实际上是只读的。

请参阅 CRecordset::DoFieldExchange, CRecordset::DoBulkFieldExchange, CRecordset::GetODBCFieldCount, CRecordset::GetODBCFieldInfo, CRecordset::SetRowsetCursorPosition

CRecordset::GetODBCFieldCount

short GetODBCFieldCount() const;

返回值

数据集中的字段数。

说明

此成员函数用来获取记录集中的字段总数。

请参阅 CRecordset::GetFieldValue

CRecordset::GetODBCFieldInfo

void GetODBCFieldInfo(LPCTSTR *lpszName*, CODBCFieldInfo& *fieldinfo*);
throw(CDBException);
void GetODBCFieldInfo(short *nIndex*, CODBCFieldInfo& *fieldinfo*);

throw(CDBException);

参数

lpszName

字段的名字。

fieldinfo

一个指向 CODBCFieldInfo **结构的引用。**

nIndex

字段从零开始的索引。

说明

此成员函数用来获取记录集中的字段的信息。此函数的一个版本可以让你用名字来查找一个字段。另一个版本让你用索引来查找一个字段。

有关返回信息的描述，参见 CODBCFieldInfo **结构。**

请参阅 CRecordset::GetFieldValue, CODBCFieldInfo

CRecordset::GetRecordCount

long GetRecordCount() const;

返回值

记录集中记录的数目。如果记录集中不包含记录，则为 0，**如果不能确定记录的个数，则为－**1。

说明

调用这个函数以确定记录集的大小。

警告 **记录计数保持为“顶点标记”----当用户移动记录时可以看见的最高编号的记录。记录总数仅在用户已经移动过最后一个记录时才可知。**

由于性能原因，在应用程序中调用 MoveLast 时，记录计数不被更新。要想自己更新计数记录，可重复调用 MoveNext 函数，直到 IsEOF 返回一个非零值。通过 CRecordset:AddNew 来加入一个新的记录，并用 Update 来增加计数。通过 CRecordset:Delete 来删除一个记录并减小计数。

请参阅 **CRec**ordset::MoveLast, CRecordset::MoveNext, CRecordset::IsEOF, CRecordset::GetStatus

CRecordset::GetRowsetSize

DWORD GetRowsetSize() const;

返回值

返回在一次给定的检取操作中所获取的行数。

说明

此成员函数用来获取关于你希望在一次给定的检取操作中获取的行数的设置。

如果你使用成组行检取，记录集被打开时的缺省行集大小为 25**；否则为** 1**。**

为了实现成组行检取，你必须在 Open **成员函数的** *dwOptions* **参数中指定** CRecordset::use- MultiRowFetch **选项。如果要改变关于行集大小的设置，调用** SetRowsetSize**。**

请参阅 CRecordset::Open, CRecordset::SetRowsetSize, CRecordset::CheckRowsetError, CRecordset::DoBulkFieldExchange

CRecordset::GetRowsFetched

```
DWORD GetRowsFetched( ) const;
```

返回值

在执行了指定的检取操作后从数据源获取的行数。

说明

调用这个成员函数以确定在检取后实际获得了多少记录。当你实现成组行检取的时候，这是非常有用的。行集大小通常指定了在一次检取中会获得多少行记录，但是，记录集中记录的总数也会影响能在行集中获取的行数。例如，如果你的记录集中有 10 **条记录，行集大小被设为** 4**，那么在记录集中循环调用** MoveNext **会导致最后一个行集中只有** 2 **条记录。**

为了实现成组行检取，你必须在 Open **成员函数的** *dwOptions* **参数中指定** CRecordset::use MultiRowFetch **选项。如果要指定行集大小，调用** SetRowsetSize**。**

示例

```
MultiRowSet rs;
// 设置行集大小
rs.SetRowsetSize( 5 );
// 打开记录集
rs.Open( CRecordset::dynaset, NULL,
         CRecordset::useMultiRowFetch );
// 在记录集中循环
while( !rs.IsEOF( ) )
{
   for( int rowCount = 0;
       rowCount < (int)rs.GetRowsFetched( );
       rowCount++ )
   {
      // 作某些操作
   }
   rs.MoveNext( );
}
rs.Close( );
```

请参阅 CRecordset::SetRowsetSize, CRecordset::CheckRowsetError

CRecordset::GetRowStatus

WORD GetRowStatus(WORD *wRow*) const;

返回值

行的状态值。有关细节参见说明部分。

参数

wRow

该行在记录集中的位置，从 1 **开始计算。这个值的范围是从** 1 **到行集的大小。**

说明

调用这个函数以获得当前行集中某行的状态。GetRowStauts **返回一个值，指明了自最近一次从数据源中获取数据以来，行的状态是否有了变化，或者指明没有检取到与** *wRow* **对应的记录。下面的表格列出了可能的返回值。**

状态值	**描述**
SQL_ROW_SUCCESS	**该行没有发生变化**
SQL_ROW_UPDATED	**该行已经被更新**
SQL_ROW_DELETED	**该行已经被删除**
SQL_ROW_ADDED	**该行已经被加入**
SQL_ROW_ERROR	**由于发生了错误，无法获得该行**
SQL_ROW_NOROW	**没有与** *wRow* **对应的行**

更多的信息参见“ODBC SDK **程序员参考”中的** ODBC API **函数** SQLExtendedFetch**。**

请参阅 CRecordset::CheckRowsetError, CRecordset::GetRowsFetched, CRecordset::RefreshRowset

CRecordset::GetStatus

void GetStatus(CRecordsetStatus& *rStatus*) const;

参数

rStatus

对一个 CRecordsetStatus **对象的引用。更多的信息参见说明部分。**

说明

调用这个函数以确定记录集中的当前记录的索引，并（或）确定是否已看见最后一个记录。CRecordset **试图跟踪索引，但是在某些情况下可能无法做到。有关解释参见** GetRecordCount **函数。**

CRecordsetStatus **结构具有如下形式：**

```
struct CRecordsetStatus
{
```

```
    long m_lCurrentRecord;
    BOOL m_bRecordCountFinal;
};
```

CRecordsetStatus的两个成员的含义如下：

- m_lCurrentRecord 包含了记录集中当前记录的索引（从零开始），如果可知的话。如果不能确定索引，则该成员中包含AFX_CURRENT_RECORD_UNDEFINED（-2）。如果IsBOF为TRUE（空记录集或试图滚动到第一个记录之前），m_lCurrentRecord将被设为AFX_CURRENT_RECORD_BOF（－1）。如果是在第一个记录，它被设为0，第二个记录被设为1，等等。
- m_bRecordCountFinal 如果记录集中记录的总数已经确定，则为非零值。通常这个必须通过在记录集头开始并调用MoveNext，直到IsEOF返回非零值。如果这个成员为0，则该记录计数就是GetRecordCount返回的值，如

果不是-1，就只是记录的“顶点标记”计数。

请参阅 CRecordset::GetRecordCount

CRecordset::GetSQL

const CString& GetSQL( ) const;

返回值

对包含了 SQL 语句的 CSting 的 const 引用。

说明

调用这个成员函数以获取 SQL 语句，该语句用于在记录集打开时选择记录集中的记录。这通常是一个 SQL SELECT 语句。由 GetSQL 返回的字符串是只读的。

由 GetSQL 返回的字符串通常与你在 Open 成员函数的 *lpszSQL* 参数中传递给记

录集的字符串不同。这是因为记录集根据你传递给 Open 的选项、通过 ClassWizard 指定的选项、在 m_strFilter 和 m_strSort 数据成员中指定的选项以及指定的其它任何参数来构造一个完整的 SQL 语句。

重点 在调用 0pen 以后调用这个成员函数。

请参阅 CRecordset::GetDefaultSQL, CRecordset::Open, CRecordset::m_strFilter, CRecordset::m_strSort

CRecordset::GetTableName

```
const CString& GetTableName( ) const;
```

返回值

如果记录集是基于一个表格的,则返回值为对包含了表格名字的 CString 的 const 引用。否则是一个空字符串。

说明

调用这个函数以获取SQL表的名字，记录集查询是基于这个表的。只有当记录集是基于一个表，而不是多个表的连接或者预定义的查询（存储过程）时，GetTableName才是有效的。这个表名是只读的。

重点　在调用了0pen以后调用这个成员函数。

CRecordset::IsBOF

BOOL IsBOF( ) const;

返回值

如果记录集不包含记录或者程序已经向后滚动到第一个记录之前，则返回一个非零值；否则返回值为0。

说明

在记录之间滚动之前调用这个函数，以确定是否已经到了记录集的第一条记录之前。你也可以与 IsEOF 一起使用 IsBOF 函数来确定记录集中是否包含记录，或者为空。在调用 Open 之后，如果记录集不包含记录，IsBOF 返回非零值。当你打开一个至少具有一条记录的记录集时，第一个记录成为当前记录，IsBOF 返回 0。

如果当前记录为第一条记录并且你调用了 MovePrev，IsBOF 将随后返回一个非零值。如果 IsBOF 返回非零值并且你调用了 MovePrev，将会发生错误。如果 IsBOF 返回非零值，则当前记录没有定义，所有需要当前记录的操作将会导致错误。

示例

这个例子在双向滚动记录集时使用 IsBOF 和 IsEOF 来检测记录集的界限。

```
// 打开一个记录集，第一条记录是当前记录。
CCustSet rsCustSet( NULL );
rsCustSet.Open( );
if( rsCustSet.IsBOF( ) )
   return;
   // 记录集为空。
// 滚动到记录集的尾部，越过最后一条记录，
// 因此没有当前记录。
while ( !rsCustSet.IsEOF( ) )
   rsCustSet.MoveNext( );
// 移动到最后一条记录
rsCustSet.MoveLast( );
// 滚动到记录集的头部，在第一条记录之前，
```

```
// 因此没有当前记录。
while( !rsCustSet.IsBOF( ) )
   rsCustSet.MovePrev( );
// 第一条记录又成了当前记录。
rsCustSet.MoveFirst( );
```

请参阅 CRecordset::IsEOF, CRecordset::MoveFirst, CRecordset::MovePrev

CRecordset::IsDeleted

```
BOOL IsDeleted( ) const;
```

返回值

如果记录集被定位在一个被删除的记录上则返回非零值；否则返回 0。

说明

调用这个函数以确定当前记录是否已被删除。如果你滚动到一条记录，IsDeleted 返回 TRUE（非零值），在你能够执行任何记录集操作之前，你必须先滚动到其它记录。

注意 IsDeleted 的结构依赖于许多因素，例如你的记录集的类型，记录集是否可更新，在打开记录集时是否指定了 CRecordset::skipDeletedRecords 选项，驱动程序是否压缩被删除的记录，以及是否有多个用户。

有关 CRecordset::skipDeletedRecords 和驱动程序压缩的更多信息参见 Open 成员函数。

> **注意** 如果你实现了成组行读取，你不能调用 IsDeleted，而应该调用 GetRowStatus 成员函数。

请参阅 CRecordset::Delete, CRecordset::IsBOF, CRecordset::IsEOF

CRecordset::IsEOF

BOOL IsEOF( ) const;

返回值

如果记录集中不包含记录或者程序已经滚动到最后一个记录之后，则返回非零值；否则返回 0。

说明

当应用程序在记录之间滚动时，调用这个成员函数来确定应用程序是否已滚动超出了记录集的最后一个记录。你也可以使用 IsEOF **来确定记录集中是包含了记录还是为空。在调用了** Open **以后，如果记录集不包含记录，**IsEOF **返回非**

零值。当你打开一个至少具有一条记录的记录集的时候，第一条记录成为当前记录，IsEOF 返回 0。

如果在你调用 MoveNext 的时候，当前记录是最后一条记录，IsEOF 将会返回一个非零值。如果 IsEOF 返回非零值并且你调用了 MoveNext，将会产生一个错误。如果 IsEOF 返回非零值，则当前记录未定义，并且任何需要当前记录的操作将会导致错误。

示例

参见 IsBOF 的例子。

请参阅 CRecordset::IsBOF, CRecordset::MoveLast, CRecordset::MoveNext

CRecordset::IsFieldDirty

BOOL IsFieldDirty(void* *pv*);
throw(CMemoryException);

返回值

如果由于调用了 AddNew **或** Edit**，指定的字段数据成员被改变了，则返回非零值；否则返回** 0**。**

参数

pv

指向想要检查其状态的字段数据成员的指针；或者为 NULL**，用来确定某些字段是否被更改了。**

说明

调用这个成员函数以确定指定的字段数据成员是否因为调用了 Edit 或 AddNew 而改变。当调用 CRecordset 的 Update 成员函数（跟在 Edit 或 AddNew 调用之后）更新当前记录时，所有被修改了的字段数据成员中的数据都将被传送给数据源上的对应记录。

> 注意　这个成员函数对使用了成组行检取的记录集不起作用。如果你实现了成组行检取，那么 IsFieldDirty 将总是返回 FALSE 并导致一个失败的断言。

调用 IsFieldDirty 之后，前面调用 SetFieldDirty 的效果将会被复位，因为字段的状态将被重新计算。在使用 AddNew 的时候，如果当前字段值与 null 值不同，字段状态将被设为改变过的。在使用 Edit 的时候，如果字段值与缓冲的值不同，

则字段状态被设为改变过的。

IsFieldDirty 是通过 DoFieldExchange 实现的。

请参阅 CRecordset::SetFieldDirty, CRecordset::IsFieldNull

CRecordset::IsFieldNull

```
BOOL IsFieldNull( void* pv );
   throw( CMemoryException );
```

返回值

如果指定的字段数据成员被标记为 Null，则返回一个非零值；否则返回 0。

参数

pv

指向一个要检查其状态的字段数据成员的指针，或者是 NULL，以确定

是否所有的字段都是 Null。

说明

调用这个成员函数以确定记录集中指定的字段数据成员是否被标记为 Null。（在数据库术语中，Null 表示“没有值”，与 C++中的 NULL 不同）如果一个字段数据成员被标记为 Null，它可以解释为当前记录的某一列没有值。

注意　这个成员函数对使用了成组行检取的记录集不起作用。如果你实现了成组行检取，IsFieldNull 将总是返回 FALSE，并将导致断言失败。

IsFieldNull 是通过 DoFieldExchange 实现的。

请参阅　CRecordset::SetFieldNull, CRecordset::IsFieldDirty

CRecordset::IsFieldNullable

BOOL IsFieldNullable(void* *pv*);
throw(CDBException)

参数

pv

指向要检查其状态的字段数据成员的指针。或者是 NULL**，表示要确定是否所有的字段都可以被设置为** Null **值。**

说明

调用这个函数以确定指定的字段数据成员是否可以为空（可以被设为 Null **值，**C++**的** NULL **与数据库术语中的** Null **并不相同，后者意味着“没有值”）。**

注意 **如果你实现了成组行检取，你不能调用** IsFieldNullable**，而是应**

当调用 GetODBCFieldInfo 成员函数以确定一个字段是否可以被设为 Null 值。注意你在任何时候都可以调用 GetODBCFieldInfo，而不用考虑你是否实现了成组行检取。

不能为空的字段必须具有值。如果你试图在增加或更新一个记录时将这样一个字段的值设置为 Null，则数据源将拒绝增加或更新，并且 Update 将抛出一个异常。异常发生在调用 Update 时，而不是发生在调用 SetFieldNull 时。

用 NULL 作为 IsFieldNullable 函数的第一个参数，将使该函数只应用于 outputCloumns，而不作用于 params。例如，调用

```
SetFieldNull( NULL );
```

这只将设置 outputColumns 为 NULL，而 Params 将不受影响。

要作用于 params，你必须为所想要设置的各个 params 提供实际地址，例如：

```
SetFieldNull( &m_strParam );
```

这意味着应用程序不能像对 outputColumns 一样，将所有 params 都设置为 NULL。

IsFieldNullable 通过 DoFieldExchange 实现。

请参阅 CRecordset::IsFieldNull, CRecordset::SetFieldNull

CRecordset::IsOpen

BOOL IsOpen() const;

返回值

如果先前已经调用了记录集对象的 Open 或 Requery 成员函数，并且记录集还未关闭，则返回一个非零值，否则返回值为零。

说明

此成员函数用来确定记录集是否已经打开。

CRecordset::Move

virtual void Move(long *nRows*, WORD *wFetchType* = SQL_FETCH_RELATIVE);
    throw(CDBException, CMemoryException);

参数

nRows

要向前或向后移动的行数。正值表示向前移动，直至移动到记录集的尾部。负值表示向后移动，直至移动到开始处。

wFetchType

确定 Move **将要获取的行集。其细节参见说明部分。**

说明

此成员函数用来在记录集中向前或向后移动当前记录指针。如果你给 *nRows* 传递一个为 0 的值，则 Move 刷新当前记录；Move 将终止当前的 AddNew 或 Edit 模式，并且将把当前记录的值恢复到调用 AddNew 或 Edit 之前的值。

> **注意** 当在记录集中移动时，不能略过被删除的记录。有关的细节可以参见 IsDeleted 成员函数。

Move 用行集来定位记录集。根据传递给 *nRows* 和 *wFetchType* 的值，Move 检取相应的行集，然后将此行集中的第一个记录作为当前记录。如果你还没有实现成组行检取，则行集的大小总是 1。当检取一个行集时，Move 直接调用 CheckRowsetError 成员函数来处理检取中发生的任何错误。

Move 与其他的 CRecordset 成员函数是等同的，这得看你所传递的值。尤其是，

数值 WFetchType 指明了更为直观的成员函数，对于移动当前记录来说，可能更喜欢使用该方法。

下面的表列出了 *wFetchType* 可能的取值，Move 根据 *wFectchType* 和 *nRows* 将获取的行集，和其它对应于 wFetchType 的相当的成员函数。

WfetchType	获取的行集	等效成员函数
SQL_FETCH_RELATIVE	从当前行集的第一行开始的 *nRows* 行的行集，(缺省值)	
SQL_FETCH_NEXT	下一个行集；*nRows* 被忽略	MoveNext
SQL_FETCH_PRIOR	前一个行集；*nRows* 被忽略	MovePrev
SQL_FETCH_FIRST	记录集中的第一个行集，*nRows* 被忽略	MoveFirst

续表

SQL_FETCH_LAST	记录集中的最后一个完整的行集；*nRows* 被忽略	MoveLast
SQL_FETCH_ABSOLUTE	如果 *nRows* > 0，行集从记录集的开始处开始 *nRows* 行。如果 *nRows* < 0，则行集从记录集的结尾处开始 *nRows* 行。如果 *nRows* = 0，则返回一个 BOF 条件	SetAbsolutePosition
SQL_FETCH_BOOKMARK	行集开始于书签值与 *nRows* 一致的行	SetBookmark

注意 对于只向前的记录集，Move 只在 *wFetchType* 的值为 SQL_FETCH_NEXT 才有效。

警告 如果记录集没有记录则调用 Move 将抛出一个异常。要确定记录集是否有记录，可以调用 IsBOF 和 IsEOF。

如果你已经滚动过了记录集的开始或结尾（IsBOF 或 IsEOF 返回非零值），则调用 Move 函数将有可能抛出一个 CDBException。例如，如果 IsEOF 返回非零

值而 IsBOF 没有返回非零值，则 MoveNext 将抛出一个异常，而 MovePrev 就不会抛出异常。

如果你在当前记录被更新或增加时调用 Move，则更新的值将丢失，并且不会给出警告。

有关的信息，参见“ODBC SDK 程序员参考”中的 ODBC API 函数 SQLExtendedFetch。

示例

```
// rs 是一个 CRecordset 或一个 CRecordset 派生对象。
// 将行集的大小改变为 5
rs.SetRowsetSize( 5 );
// 移动到记录集中的第一个记录
rs.MoveFirst( );
```

```
// 移动到第六个记录
rs.Move( 5 );
// 用其它相当的方法移动到第六个记录
// rs.Move( 6, SQL_FETCH_ABSOLUTE );
// rs.SetAbsolutePosition( 6 );
// 在这种情况下，第六个记录是下一个行集的第一个记录。
// 所以下面的方法也是相当的。
// rs.Move( 1, SQL_FETCH_NEXT );
// rs.MoveNext( );
```

请参阅 CRecordset::MoveNext, CRecordset::MovePrev, CRecordset::MoveFirst, CRecordset::MoveLast, CRecordset::SetAbsolutePosition, CRecordset::SetBookmark, CRecordset::IsBOF, CRecordset::IsEOF, CRecordset::CheckRowsetError

CRecordset::MoveFirst

```
void MoveFirst();
```

throw(CDBException, CMemoryException);

说明

此成员函数用来使第一个行集中的第一个记录成为当前记录。不管是否实现了成组行检取，此记录都是记录集中的第一个记录。

你不用在打开记录集之后就立刻调用 MoveFirst。在打开记录集的时候，第一个记录（如果有的话）会自动成为当前记录。

注意 对于仅向前的记录集来说，此成员函数是无效的。

注意 当在一个记录集中移动时，删除的记录不能被忽略。参见 IsDelete 成员函数可以获得有关的细节。

警告 如果记录集没有记录，则调用任何 Move 函数都将抛出一个异常。要确定记录集中是否有记录，可以调用 IsBOF 和 IsEOF。

在当前记录正在被更新或增加时，如果你调用任何 Move 函数，则更新将丢失，并且不给出警告。

示例

参见 IsBOF 的例子。

请参阅 CRecordset::Move, CRecordset::MoveLast, CRecordset::MoveNext, CRecordset::MovePrev, CRecordset::IsBOF, CRecordset::IsEOF

CRecordset::MoveLast

```
void MoveLast();
   throw( CDBException, CMemoryException );
```

说明

此成员函数用来使最后一个完整行集中的第一个记录成为当前记录。如果你还

没有实现成组行检取，则你的记录集的行集大小是 1，所以 MoveLast 只简单地移动到记录集中的最后一个记录。

注意 对于仅向前记录集，此成员函数是无效的。

注意 当在一个记录集中移动时，不能忽略被删除的记录。参见 IsDeleted 成员函数会获得有关的细节。

警告 如果记录集没有记录，则调用任何 Move 函数都将抛出一个异常。要确定记录集中是否有记录，可以调用 IsBOF 和 IsEOF。

在当前记录正在被更新或增加时，如果你调用任何 Move 函数，则更新将丢失，并且不给出警告。

请参阅 CRecordset::Move, CRecordset::MoveFirst, CRecordset::MoveNext, CRecordset::MovePrev, CRecordset::IsBOF, CRecordset::IsEOF

CRecordset::MoveNext

```
void MoveNext();
   throw ( CDBException, CMemoryException );
```

说明

此成员函数用来使下一个记录集中的第一个记录成为当前记录。如果你还没有实现成组行检取，则你的记录集的行集的大小为 1，所以 MoveNext 只简单地移动到下一个记录。

> **注意** 当在一个记录集中移动时，不能忽略被删除的记录。参见 IsDeleted 成员函数会获得有关的细节。
>
> **警告** 如果记录集没有记录，则调用任何 Move 函数都将抛出一个异常。要确定记录集中是否有记录，可以调用 IsBOF 和 IsEOF。

还要忠告你，必须在调用 MoveNext 之前调用 IsEOF。例如，如果你已经滚动过了记录集的结尾，则 IsEOF 将返回非零值；则随后调用 MoveNext 将抛出一个异常。

在当前记录正在被更新或增加时，如果你调用任何 Move 函数，则更新将丢失，并且不给出警告。

示例

参见 IsBOF 的例子。

请参阅 CRecordset::Move, CRecordset::MovePrev, CRecordset::MoveFirst, CRecordset::MoveLast, CRecordset::IsBOF, CRecordset::IsEOF

CRecordset::MovePrev

void MovePrev();

throw(CDBException, CMemoryException);

说明

此成员函数用来使上一个记录集中的第一个记录成为当前记录。如果你还没有实现成组行检取，则你的记录集的行集的大小为 1，所以 MovePrev 只简单地移动到前一个记录。

注意 此成员函数对于只向前记录集无效。

注意 当在一个记录集中移动时,不能忽略被删除的记录。参见 IsDeleted 成员函数会获得有关的细节。

警告 如果记录集没有记录，则调用任何 Move 函数都将抛出一个异常。要确定记录集中是否有记录，可以调用 IsBOF 和 IsEOF。

还要忠告你，必须在调用 MovePrev 之前调用 IsBOF。例如，如果你已经滚动

过了记录集的开始，则 IsBOF 将返回非零值；则随后调用 MovePrev 将抛出一个异常。

在当前记录正在被更新或增加时，如果你调用任何 Move 函数，则更新将丢失，并且不给出警告。

示例

参见 IsBOF 的例子。

请参阅 CRecordset::Move, CRecordset::MoveNext, CRecordset::MoveFirst, CRecordset::MoveLast, CRecordset::IsBOF, CRecordset::IsEOF

CRecordset::OnSetOptions

virtual void OnSetOptions(HSTMT *hstmt*);

参数

hstmt

要设置其选项的 ODBC **语句的** HSTMT**。**

说明

框架调用此成员函数来设置记录集的初始选项。OnSetOption **确定数据源是否支持可滚动游标和游标并行特性，并相应设置记录集的选项。**

可以重载 OnSetOptions **函数来设置专用于驱动器或数据源的附加项。例如，如果数据源支持打开进行独占存取，则应用程序可以重载** OnSetOptions **来利用这种功能。**

请参阅 CDatabase::OnSetOptions

CRecordset::Open

virtual BOOL Open(UINT *nOpenType* = AFX_DB_USE_DEFAULT_TYPE,
LPCTSTR *lpszSQL* = NULL, DWORD *dwOptions* = none);
throw(CDBException, CMemoryException);

返回值

如果 CRecordset **对象被成功打开，则返回非零值；否则，如果** CDatabase::Open **（如果被调用了）返回** 0**，则返回** 0**。**

参数

nOpenType

接收缺省值 AFX_DB_USE_DEFAULT_TYPE**，或使用下列** enum OpenType **枚举值之一：**

- CRecordset::dynaset **可以双向滚动的记录集。当记录集被打开时，记**

录集的全体成员和记录的顺序都被确定，但是伴随一次获取操作，其它用户对数据值所做的改变是可见的。

- CRecordset::snapshot 可以双向滚动的静态记录集。当记录集被打开时，记录集的全体成员和记录的顺序都被确定了；当记录被获取时，其数据值是确定的。其它用户在记录集被关闭，然后在打开之前是不可见的。
- CRecordset::dynamic 可以双向滚动的记录集。伴随一次获取操作，其它用户对成员,顺序和数据值所做的改变是可见的。注意,许多 ODBC 驱动器是不支持这种类型的记录集的。
- CRecordset::forwardOnly 只能向前滚动的只读的记录集。

对于 CRecordset 来说，缺省的值是 CRecordset::snapshot。这种缺省值机制，允许 Visual C++ 向导与具有不同缺省值的 ODBC CRecordset 和 DAO

CDaoRecordset进行交互。

与此有关的信息，参见“ODBC SDK程序员参考”中文章的“使用块和可滚动游标”。

警告 如果不支持要求的类型，框架将抛出一个异常。

lpszSQL

一个包含下列值之一的字符串指针：

- 一个NULL指针。
- 一个表名。
- 一条SQL SELECT语句（可选择带一条SQL WHERE或ORDER BY子句）。
- 一条CALL语句，指定一个预定义查询（存储过程）名。注意，你不能在卷括号和CALL关键字之间插入空格。

有关这个字符串的更多信息，请参见说明下的表，以及对 ClassWizard 的作用的讨论。

注意 在你的结果集中，列的顺序必须与在你的重载的 DoFieldExchange 或 DoBulkFieldExchange 函数中的 RFX 和 Bulk RFX 函数调用的顺序相匹配。

dwOptions

一个 bitmask，它可以指定下列值的组合。这些值中的某些是相互独立的。缺省的值是 none。

- CRecordset::none 未设置选项。此参数值与所有其它的值是相互独立的。缺省的，记录集可以用 Edit 或 Delete 来更新，并允许用 AddNew 来添加新记录。这种可更新性依赖于数据源，就像你所指定的 *nOpenType* 选项。成组添加的优化是没有用的。成组行检取不会被实

现。在记录集中导航时，不能略过被删除的记录。书签是没有用的。实现了自动脏字段检查。

- CRecordset::appendOnly 不允许在记录集中进行 Edit 或 Delete。值允许 AddNew。此选项与 CRecordset::readOnly 相互独立。
- CRecordset::readOnly 用只读方式打开记录集。此选项与 CRecordset::appendOnly 相互独立。
- CRecordset::optimizeBulkAdd 使用预备的 SQL 语句来优化一次添加多个记录。只有在你不使用 ODBC API 函数 SQLSetPos 来更新记录集时才使用。第一次更新确定将哪一个字段标记为脏的。此选项与 CRecordset::useMultiRowFetch 相互独立。
- CRecordset::useMultiRowFetch 实现成组行检取，允许在一次检取操作中获取多行。这是一个高级特征，是设计来提高性能的；但是，

ClassWizard 不支持成组记录字段交互。此选项与 CRecordset::optimiazBulkAdd 相互独立。注意，如果你指定了 CRecordset::useMultiRowFetch，则选项 CRecordset::noDirtyFieldCheck 将自动被返回（双缓冲将无效）；对于只向前记录集，选项 CRecordset::useExtendedFetch 将被自动返回。

- CRecordset::skipDeletedRecords 当在记录集中导航时，略过所有已被删除的记录。这将使某个相关获取的性能下降。在一个只向前的记录集中，此选项是无效的。注意，CRecordset::skipDeletedRecords 与“驱动器包装”相似，驱动器包装意味着被删除的行被从记录集中移走。但是，如果你的驱动器包装记录，则它将只略过那些被你删除的记录；而不会略过在记录集打开时被其它用户删除的记录。CRecordset::skipDeletedRecords 将会略过被其它用户删除的记录。

- CRecordset::useBookarks 允许在记录集中使用书签，如果支持书签的话。书签会使数据检取变慢，但是会提高数据导航的性能。在只向前的记录集中此选项是无效的。
- CRecordset::noDirtyFieldCheck 关闭自动脏数据检查（双缓冲）。这将提高性能；但是，你必须通过调用 SetFieldDirty 和 SetFieldNull 成员函数来手动标记变脏的字段。注意，CRecordset 类中的双缓冲类似于 CDaoRecordset 中的双缓冲。但是，在 CRecordset 中，你不能在独立字段中使双缓冲有效；你只能对所有字段使它有效或无效。注意，如果你指定了选项 CRecordset::useMultiRowFetch，则 CRecordset::noDirtyFieldCheck 将被自动返回；但是，SetFieldDirty 和 SetFieldNull 不能在实现成组行检取的记录集中使用。
- CRecordset::executeDirect 不要使用预备的 SQL 语句。如果永远不会

调用 Requery 成员函数，则指定此选项来提高性能。

- CRecordset::useExtendedFetch 实现 SQLException 来代替 SQLFetch。这是设计来在只向前的记录集中实现成组行检取的。如果你在一个只向前的记录集中指定了选项 CRecordset::userMultiRowFetch，则 CRecordset::useExtendedFetch 将被自动返回。
- CRecordset::userAllocMultiRowBuffers 用户将为数据分配存储缓存。如果你想分配自己的存储区，将此选项用与 CRecordset::useMultiRowFetch 连接；否则，框架将自动分配必要的存储区。注意，指定了 CRecordset::userAllocMutiRowBuffers，而没有指定 CRecordset::useMultiRowFetch，将导致一个失败断言。

说明

你必须调用此成员函数来运行记录集定义的查询。在调用 Open 之前，你必须构造记录集对象。

此记录集与数据源的连接依赖于在调用 Open 之前你是如何构造这个记录集的。如果你将一个没有连接到一个数据源的 CDatabase 对象传递给记录集构造函数，此成员函数使用 GetDefaultConnect 来尝试打开该数据库对象。如果你将 NULL 传递给记录集构造函数，则此成员函数为你构造一个 CDatabase 对象，并且 Open 试图连接给数据库对象。有关关闭记录集和在这些不同的环境下的连接的细节，请参见 Close。

注意　通过一个 CRecordset 对象对数据源的访问总是被共享的。不像 CDaoRecordset 类，不能使用一个 CRecordset 对象来打开一个具有独占

访问的数据源。

当你调用 Open 时，一条查询语句，通常是一条 SQL SELECT 语句，基于下表给出的标准来选择记录：

lpszSQL 参数的值	选择的记录所决定于	示例
NULL	由 GetDefaultSQL 返回的字符串	
SQL 表名	在 DoFieldExchange 或 DoBulkFieldExchange 中的列表的所有列	“Customer”
预定义的查询（存储过程）名称	该查询被定义的要返回的列	“{call OverDueAccts}”
SELECT 列表 FROM 列表	从指定的表中选出的指定的列	SELECT CustId, CustName FROM Customer”

警告　注意，在 SQL 字符串中，你不能插入额外的空格。例如，如果你在卷括号和 CALL 关键字之间插入了空格，MFC 将错误地将这个 SQL 串解释为一个表名，并将它合并到 SELECT 语句中，这将导致抛出一个异常。

类似地，如果使用输出参数的预定义查询没有在卷括号和‘？’号之间插入空格，则也会导致抛出异常。最后，你不能在一个 CALL 语句中的卷括号之前和在一个 SELECT 语句中的 SELECT 关键字之前插入空格。

通常的程序是将 NULL 传递给 Open；在这种情况下，Open 调用 GetDefaultSQL。如果你正在使用一个 CRecordset 派生类，GetDefaultSQL 给出你在 ClassWizard 中指定的表名。你可以代替在 *lpszSQL* 参数中指定其它的信息。

不管传递的是什么，Open 都为查询构造一个最后的 SQL 字符串（该字符串可能有 SQL WHERE 和 ORDER BY 子串添加在你传递的 *lpszSQL* 串中），然后执行这个查询。你可以在调用 Open 之后通过调用 GetSQL 来检查这个构造出来的字符串。

记录集类的字段数据成员与所选择数据的列相关联。如果有记录被返回，则第一个记录成为当前记录。

如果你想为记录集设置选项，比如一个过滤器或排序，则应在构造了此记录集对象之后，但在调用 Open 之前指定这些选项。如果你要在记录集被打开之后刷新记录集中的记录，可以调用 Requery。

示例

下面的代码例子说明了 Open 调用的不同形式。

```
// rs 是一个 CRecordset 或 CRecordset 派生对象
// 用缺省的 SQL 语句打开 rs
// 实现书签，并关闭自动脏数据检查
rs.Open( CRecordset::snapshot, NULL,
    CRecordset::useBookmarks |
    CRecordset::noDirtyFieldCheck );
// 传递一个完整的 SELECT 语句并打开作为一个动态集
rs.Open( CRecordset::dynaset,
```

```
    _T( "Select L_Name from Customer" ) );
// 接收所有的缺省值
rs.Open( );
```

请参阅 CRecordset::CRecordset, CRecordset::Close, CRecordset::GetDefaultSQL, CRecordset::GetSQL, CRecordset::m_strFilter, CRecordset::m_strSort, CRecordset::Requery

CRecordset::RefreshRowset

void RefreshRowset(WORD *wRow*, WORD *wLockType* = SQL_LOCK_NO_CHANGE);

参数

wRow

在当前行集中一行的从 1 开始的位置。这个值的范围可以是零到此行集的大小之间。

wLockType

一个值，表明在行被刷新之后如何去加锁这个行。有关的细节可参见说明部分。

说明

此成员函数用来为当前行集中的一行更新数据和状态。如果你传递一个为零的值给 *wRow*，则此行集中的每一行都将被刷新。

要使用 RefreshRowset,你必须已经通过指定 Open 成员函数中的 CRecordset::use-MulitRowFetch 选项实现了成组行检取。

RefreshRowset 调用 ODBC API 函数 SQLSetPos。参数 *wLockType* 指定在执行 SQLSetPos 之后此行的加锁状态。下面的表格描述了 *wLockType* 可能的取值。

WlockType	描述
SQL_LOCK_NO_CHANGE（缺省值）	驱动器和数据源保证该行的加锁或解锁状态与调用 RefreshRowset 之前一致
SQL_LOCK_EXCLUSIVE	驱动器或数据源各自独立的加锁该行。不是所有的数据源都支持这种加锁类型
SQL_LOCK_UNLOCK	驱动器或数据源解锁此行。不是所有的数据源都支持这种加锁类型

有关 SQLSetPos 的更多信息，请参见“ODBC SDK 程序员参考”。

请参阅 CRecordset::SetRowsetCursorPosition, CRecordset::SetRowsetSize

CRecordset::Requery

```
virtual BOOL Requery();
   throw ( CDBException, CMemoryException );
```

返回值

如果记录集重建成功则返回非零值；否则返回 0。

说明

此成员函数用来重建（刷新）一个记录集。如果有记录返回，则第一个记录成为当前记录。

为了使记录集可反映你或其它用户正对数据集进行的添加或删除，你必须调用 Requery 函数来重建记录集。如果记录集是一个动态集，则它将自动反映你或其它用户对其现有记录（而非所添加的记录）的更新。如果记录集是一个快照，则应用程序必须调用 Requery 来反映其它用户的标记及添加和删除。

无论是对一个动态集还是对一个快照，都可以在应用程序想要用一个新的过滤器或排序，或新的参数值来重建记录集的任何时候，调用 Requery 函数。可以在调用 Requery 之前，通过赋新值给 m_strFilter 和 m_strSort 成员来设置新的过滤器或排序特征；赋新值给参数数据成员来设置新的参数。如果过滤器和排序

串不变，则可以复用该查询，这将会提高性能。

如果重建记录集的尝试失败，则记录集被关闭。在调用 Requery 之前，你可以通过调用 CanRestart 成员函数来确定记录集是否可被重新查询。CanRestart 不保证 Requery 会成功。

注意 只能在调用 Open 之后调用 Requery 函数。

示例

此示例重建一个记录集，使其采用另一种不同的排列顺序。

```
// CRecordset::Requery 示例
CCustSet rsCustSet( NULL );
// 打开一个记录集
rsCustSet.Open( );
// 使用该记录集...
```

```
// 设置排列顺序并查询记录集
rsCustSet.m_strSort = "District, Last_Name";
if( !rsCustSet.CanRestart( ) )
   return;      // 不能查询
if( !rsCustSet.Requery( ) )
   // 查询失败，因此进行某种行动
```

请参阅 CRecordset::CanRestart, CRecordset::m_strFilter, CRecordset::m_strSort

CRecordset::SetAbsolutePosition

```
void SetAbsolutePosition( long nRows );
   throw( CDBException, CMemoryException );
```

参数

nRows

此记录集中的当前记录的从一开始的顺序位置。

说明

此成员函数用来将记录集定位在指定记录号对应的记录上。SetAbsolutePosition 根据这个顺序位置来移动当前的记录指针。

注意　此成员函数在只向前的记录集中是无效的。

对于 ODBC 记录集，一个为 1 的绝对位置设置指向记录集中的第一个记录；设置 0 指向文件开始（BOF）位置。

你也可以传递负值给 SetAbsolutePosition。在这种情况下，记录集的位置是从记录集的结尾开始计算的。例如，SetAsolutePosition(-1)当前记录指针移动到记录集中的最后一个记录上。

注意　绝对位置并不是用来作为记录号的替代品。书签仍然是用来保持和返回到给定记录的最受欢迎的方式，因为当一个记录前面的记录被删

除时，该记录的绝对位置就改变了。另外，如果记录集被再次重建，你就不能保证一个给定记录将具有相同的绝对位置，因为在记录集中单个记录的顺序是不被保证的，除非该记录集是用一个使用 ORDER BY 子句的 SQL 语句创建的。

请参阅 CRecordset::SetBookmark

CRecordset::SetBookmark

```
void SetBookmark( const CDBVariant& varBookmark );
throw( CDBException, CMemoryException );
```

参数

varBookmark

一个指向 CDBVariant 对象的引用，该对象包含了一个指定记录的书签值。

说明

此成员函数用来将记录集定位在包含指定书签的记录上。为了确定在此记录集中是否支持书签，可以调用 CanBookmark。如果支持书签，为了使书签可以使用，你必须设置 Open 成员函数中的 *dwOptions* 参数的 CRecordset::useBookmarks 选项。

> **注意** 如果书签不被支持或不能使用，调用 SetBookmark 将导致抛出一个异常。在只向前的记录集中是不支持书签的。

第一次获取当前记录的书签，可调用 GetBookmark，它将书签值保存在一个 CDBVariant 对象中，用保存的书签值调用 SetBookmark 可以返回该记录。

> **注意** 在进行了一定的记录集操作之后，你必须在调用 SetBookmark 之前检查书签是否持续可用。例如，如果你用 GetBookmark 获取一个书签，

然后调用 Requery，则该书签也许已经不再有效了。调用 CDabase::GetBookmarkPersistence 来检查你是否能够安全地调用 SetBookmark。

请参阅 CRecordset::CanBookmark, CRecordset::GetBookmark, CRecordset::SetAbsolutePosition, CDatabase::GetBookmarkPersistence

CRecordset::SetFieldDirty

void SetFieldDirty(void* *pv*, BOOL *bDirty* = TRUE);

参数

pv

指向记录集中一字段数据成员的地址，或为 NULL。如果该参数为 NULL，则标记记录集中所有字段数据成员。（C++ NULL 与数据库术语中的 Null 不相同，后者表示“没有值”）。

bDirty

如果要将字段数据成员标记为“脏的”（被修改了），则设置此参数为TRUE；否则如果要将字段数据成员标记为“干净的”（未被修改过），则设置此参数为FALSE。

说明

此成员函数用来标记记录集中的一个字段数据成员为被修改了或未被修改过。标记字段为未被修改过，可以确保该字段不被更新，并产生较少的SQL语句。

注意　此成员函数对使用成组行检取的记录集是不适用的。如果你已经实现了成组行检取，则SetFieldDirty将导致一个失败断言。

框架标记改变过的字段数据成员来确保记录字段交换机制（RFX）将它们写入数据源中对应的记录中。改变一个字段的值通常自动地将该自动设置为被修改

了，因此应用程序本身很少有必要调用 SetFieldDirty 函数，但有时可能想要确保记录的列显式地被更新或插入，而不管自动数据成员中是什么值。

重点 只能在调用 Edit 或 AddNew 之后调用 SetFieldDirty 函数。

使用 NULL 作为 SetFieldDirty 函数的第一个参数，将使该函数只应用于 outputColumns，而不作用于 params。例如，调用

```
SetFieldNull( NULL );
```

将只设置 outputColumns 为 NULL，而 params 将不受影响。

要想作用于 params，应用程序必须对所想要作用的各个 param 提供实际地址，例如：

```
SetFieldNull( &m_strParam );
```

这意味着应用程序不能像对 outputColmns 一样将所有 params 都设置为 NULL。

请参阅 CRecordset::IsFieldDirty, CRecordset::SetFieldNull, CRecordset::Edit,

CRecordset::Update

CRecordset::SetFieldNull

void SetFieldNull(void* *pv*, BOOL *bNull* = TRUE);

参数

pv

指向记录集中一字段数据成员的地址，或为 NULL**。如果该参数为** NULL**，则标记记录集中所有字段数据成员。（**C++ NULL **与数据库术语中的** Null **不相同，后者表示“没有值”）。**

bNull

如果要将此字段数据成员标记为没有值（Null**），则设置该参数为一个非零值。如果要将此字段数据成员标记为非** Null**，则设置该参数为零。**

说明

此成员函数用来将记录集的一个字段数据成员标记为 Null（专指没有值）或非空。当应用程序将一个新记录增加到一个记录集中时，其所有字段数据成员初始都设置为 Null 值，并标记为“脏的”（被修改了）。当应用程序从一个数据源中检取一个记录时，该记录的列或是已有值，或是 Null。

> **注意** 此成员函数对使用成组行检取的记录集是不适用的。如果你已经实现了成组行检取，则 SetFieldDirty 将导致一个失败断言。

如果你特别希望将当前记录的某个字段设计为没有值，可将 *bNull* 设置为 TRUE 来调用 SetFieldNull 函数，以将该字段标记为 Null。如果一个字段先前标记为 Null，而现在应用程序想要给它一个值，则只需要简单地设置该字段的新值。应用程序不需要用 SetFieldNull 来删掉 Null 标志。要确定该字段是否可以为

Null，可以调用 IsFieldNullable 函数。

重点 只能在调用 Edit 或 AddNew 之后才能调用 SetFieldNull 函数。

使用 NULL 作为 SetFieldNull 函数的第一个参数，将使该函数只应用于 outputColumns，而不作用于 params。例如，调用

```
SetFieldNull( NULL );
```

将只设置 outputColumns 为 NULL，而 params 将不受影响。

要想作用于 params，应用程序必须对所想要作用的各个 param 提供实际地址，例如：

```
SetFieldNull( &m_strParam );
```

这意味着应用程序不能像对 outputColumns 一样将所有 params 都设置为 NULL。

注意 当设置参数为 Null 时，在记录集打开之前调用 SetFieldNull 将

导致一个断言。在这种情况下，可以调用 SetParamNull。

SetFieldNull 通过 DoFieldExchange 实现。

请参阅 CRecordset::IsFieldNull, CRecordset::SetFieldDirty, CRecordset::Edit, CRecordset::Update, CRecordset::IsFieldNullable

CRecordset::SetLockingMode

void SetLockingMode(UINT *nMode*);

参数

nMode

包含下列 enum LockMode 枚举之一：

- optimistic 乐观加锁方式，仅在调用 Update 期间加锁正被更新的记录。

- pessimistic 悲观加锁方式，在一调用 Edit 时即加锁，并保持加锁直到 Update 调用完成，或移动到一个新记录。

说明

如果应用程序需要指定记录集用于更新的两种记录加锁方式之一，则调用此成员函数。缺省时，记录集的加锁方式是 optimistic。应用程序可将它改变成一种更为谨慎的 pessimistic 加锁方式。SetLockMode 调用应该在应用程序构造并打开一个记录集对象之后，但在调用 Edit 之前。

请参阅 CRecordset::Edit, CRecordset::Update

CRecordset::SetParamNull

void SetParamNull(int *nIndex*, NOOL *bNull* = TRUE);

参数

nIndex

参数从零开始的索引。

bNull

如果是 TRUE（**缺省值**），**则参数被标记为** Null。**否则，参数被标记为非** Null。

说明

此成员函数用来标记参数为 Null（**专指没有值**）**或非** Null。**与** SetFieldNull **不一样，你可以在打开记录集之前调用** SetParamNull。

SetParamNull **通常是用来进行预定义查询的。**

请参阅 CRecordset::FlushResultSet

CRecordset::SetRowsetCursorPosition

void SetRowsetCursorPosition(WORD *wRow*, WORD *wLockType* =
SQL_LOCK_NO_CHANGE);

参数

wRow

当前记录集中一行的从 1 开始的位置。这个值的范围可以是从 1 到此行集的大小之间。

wLockType

一个值，指定在行被刷新之后如何对它加锁。有关的细节，参见说明部分。

说明

此成员函数用来将游标移动到当前行集中的一行。当实现成组行检取时，通过行集来获取记录，在所获取的行集中的第一个记录就是当前记录。为了使行集

中的另一个记录成为当前记录，可以调用 SetRowsetCursorPosition。例如，你可以联合 SetRowsetCursorPosition 和 GetFieldValue 成员函数来动态地获取你的记录集中的任何记录的数据。

要使用 SetRowset Cursor Position，你必须在 Open 成员函数的 *dwOptions* 参数中指明 CRecordset::useMultiRowFetch 选项，从而实现多行块存取操作。

SetRowsetCursorPosition 调用 ODBC API 函数 SQLSetPos。*wLockType* 参数指定了在执行 SQLSetPos 之后该行的加锁状态。下面的表格描述了 *wLockType* 的可能取值。

WlockType	描述
SQL_LOCK_NO_CHANGE（缺省值）	驱动器和数据源保证该行的加锁或解锁状态与调用 RefreshRowset 之前一致
SQL_LOCK_EXCLUSIVE	驱动器或数据源各自独立的加锁该行。不是所有的数据源都支持这种加锁类型
SQL_LOCK_UNLOCK	驱动器或数据源解锁此行。不是所有的数据源都支持这种加锁类型

有关 SQLSetPos 的更多信息，请参见“ODBC SDK 程序员参考”。

请参阅 CRecordset::RefreshRowset, CRecordset::SetRowsetSize

CRecordset::SetRowsetSize

virtual void SetRowsetSize(DWORD *dwNewRowsetSize*);

参数

dwNewRowsetSize

在一次给定的检取中要获取的行数。

说明

在使用成组行检取时，此虚成员函数用来指定在一次单一的检取中你希望获取的行数。要实现成组行检取，你必须在 Open 成员函数的 *dwOptions* 参数中设置 CRecordset::use- MultiRowFetch 选项。

注意 如果在没有实现成组行获取的情况下调用 SetRowsetSize，将导致一个失败断言。

在调用 Open 之前调用 SetRowsetSize 来初始设置记录集的行集的大小。在实现成组行检取时，缺省的行集大小是 25。

注意 在调用 SetRowsetSize 时要小心。如果你手动为数据分配存储区（就像在 0pen 中的 dw0ptions 参数的

CRecordset::userAllocMultiRowBuffers 选项所指定的一样），你应该在调用 SetRowsetSize 之后，但在执行任何游标导航操作之前，检查是否需要重新分配这些存储缓冲区。

要获得对行集大小的当前设置，可以调用 GetRowsetSize。

请参阅 CRecordset::Open, CRecordset::GetRowsetSize, CRecordset::CheckRowsetError, CRecordset::DoBulkFieldExchange

CRecordset::Update

```
virtual BOOL Update();
    throw ( CDBException );
```

返回值

如果一个记录更新成功则返回一个非零值；否则如果没有列改变，则返回值为

零。如果没有记录被更新，或有多个记录被更新，则抛出一个异常。对数据源上的其它任何失败也都将抛出一个异常。

说明

在调用 AddNew 或 Edit 成员函数之后，调用 Update 成员函数，必须用该调用来完成 AddNew 或 Edit 操作。

> **注意** 如果你已经实现了成组行检取，你就不能调用 Update。这将导致一个异常。虽然类 CRecordset 不提供更新成组行数据的机制，但是你可以提供使用 ODBC API 函数 SQLSetPos 来编写你自己的函数。如何做到这一点，请参见例子 DBFETCH。

AddNew 或 Edit 都准备了一个编辑缓冲区，被增加或被编辑的数据都放在此缓冲区中，等待存入数据源中。Update 保存这些数据。只有标记为或检测为被修

改了的那些字段才被更新。

如果数据源支持事务，你可以使 Update 调用（以及它的相应的 AddNew 或 Edit 调用）成为事务的一部分。

> **警告** 如果你没有先调用 AddNew 或 Edit 函数就调用 Update 函数，则 Update 函数将抛出一个异常。如果应用程序调用 AddNew 或 Edit，则在调用 MoveNext 之前，或改变记录集或数据源之前，必须调用 Update 函数。否则，所做的改变将丢失，并且不给出通知。

请参阅 CRecordset::Edit, CRecordset::AddNew, CRecordset::SetFieldDirty, CDBException

数据成员

CRecordset::m_hstmt

说明

此数据成员包含与此记录集相关联的 ODBC 语句数据结构的一个句柄，类型为 HSTMT。对一个 ODBC 数据源的每个查询都与一个 HSTMT 相关联。

警告 在调用 Open 前不要使用 m_hstmt。

通常不需要直接访问 HSTMT，但是在直接执行 SQL 语句时可能需要它。类 CDatabase 的 ExectueSQL 成员函数提供了使用 m_hstmt 的示例。

请参阅 CDatabase::ExecuteSQL

CRecordset::m_nFields

说明

此成员中包含了记录集类中字段数据成员的数目---记录集从数据源中所选择的列数。记录集必须用当前数目初始化 m_nFields 成员。当应用程序使用 ClassWizard 来声明应用程序的记录集类时，ClassWizard 将为应用程序编写这段初始代码。你也可人工编写这段代码。

框架利用这个数目来管理字段数据成员和数据源上当前记录的对应列之间的交互。

> 重点　在利用参数 CFieldExchange::outputColumn 调用 SetFieldType 之后，这个数目必须对应于 DoFieldExchange 中注册的“输出列”的数目。

你可以动态连接列，就像“Visual C++程序员指南”中的文章“记录集：动态连接数据列”所描述的一样。如果这样做，你就必须增加 m_nFields 中的计数，以反映 DoFieldExchange 成员函数中为动态连接列调用的 RFX 函数的数目。

请参阅 CRecordset::DoFieldExchange, CRecordset::DoBulkFieldExchange, CRecordset::m_nParams, CFieldExchange::SetFieldType

CRecordset::m_nParams

说明

此成员中包含记录集类中参数数据成员的数目---传递给记录集的程序的参数数目。如果记录集类中有参数数据成员，则该类的构造函数必须用当前数目来初始化 m_nParams 成员。m_nParams 的缺省值为零。如果应用程序增加参数数据成员---这必须手动进行---则也必须在类构造函数中手动地增加一个初始

化，以反映参数数目（它必须至少与 m_strFilter 或 m_strSort 字符串中的占位符‘？’数目一样多）。

当框架参数化记录集的查询时，使用这个成员中的数目。

重点 在用 CFieldExchange::inputParam, CFieldExchange::param, CFieldExchange:: outputParam, 或 CFieldExchange::inoutParam 为参数调用 SetFieldType 之后，这个成员的数目必须对应于 DoFieldExchange 或 DoBulkFieldExchange 中注册的“params”的数目。

请参阅 CRecordset::DoFieldExchange, CRecordset::DoBulkFieldExchange, CRecordset::m_nFields, CFieldExchange::SetFieldType

CRecordset::m_pDatabase

说明

此成员包含了一个指向 CDatabase 对象的指针，记录集通过此对象连接到一个数据源上。这个成员变量可用两种方式设置。通常来说，在应用程序构造记录集对象时，传递一个指向已经连接的 CDatabase 对象的指针。如果应用程序传递的是 NULL，则 CRecordset 为应用程序重建一个 CDatabase 对象并连接该对象。在这两种情况下，CRecordset 都将指针存放在此成员变量中。

一般来说，你不需要直接使用 m_pDatabase 中所存储的指针。不过，如果你想要编制自己对 CRecordset 的扩展，则可能会使用这个指针。例如，如果你抛出自己的 CDBException 异常，则你也许会需要这个指针。或者，如果你需要用相同的 CRecordset 对象来做一些事情，例如运行事务，设置超时中断，或调用

CDatabase 类的 ExecuteSQL 成员函数来直接执行 SQL 语句，则也可能需要用到这个数据成员中的指针。

CRecordset::m_strFilter

说明

在应用程序构造了记录集对象之后，但在调用该对象的 Open 成员函数之前，使用此数据成员来存放一个 CString，其中包含一条 SQL WHERE 子句。记录集利用这个字符串来限制---或过滤---它在 Open 或 Requery 调用期间可选择的记录。这可用于选择记录的一个子集，例如“ all salespersons based in California”（“ state = CA”）。一条 WHERE 子句的 ODBC SQL 语法是

WHERE serch-condition（查询条件）。

注意，在该字符串中不要包括 WHERE 关键字。框架会提供这个关键字的。

应用程序也可以通过在过滤串中放‘？’占位字符，在记录集类中为每个占位字符声明一个参数数据成员，并在运行时传递参数给记录集，来参数化过滤串。这使应用程序可以在运行时构造过滤串。

示例

```
// CRecordset::m_strFilter示例
CCustSet rsCustSet( NULL );
// 设置过滤串
rsCustSet.m_strFilter = "state = 'CA'";
// 运行过滤后的查询
rsCustSet.Open( CRecordset::snapshot, "Customers" );
```

请参阅　CRecordset::m_strSort, CRecordset::Requery

CRecordset::m_strSort

说明

在应用程序构造了记录集对象之后，但在调用此类的 Open 成员函数之前，使用此数据成员来存储一个 CString，其中包含一条 SQL ORDERBY 子句。记录集利用这个字符串来排序它在 Open 或 Requery 调用期间选择的记录。应用程序可以利用这个特性，对记录集按一列或多列进行排序。一条 ORDER BY 子句的 ODBC 语法是

ORDER BY sort-specification[,sort-specification]...

在此一个 sort-specification（排序规范）是一个整数或一个列名。应用程序也可以通过在排序字符串中的列表后添加上“ ASC ”或“ DESC ”来指定是升序还是降序（缺省是升序）。所选记录首先按第一列排序，然后按第二列排序，如

此等等。例如，可以先按人名的尾名对“Customers”记录集进行排序，再按人名的首名对该记录集进行排序。更多的信息，可参见“ODBC SDK 程序员参考”。

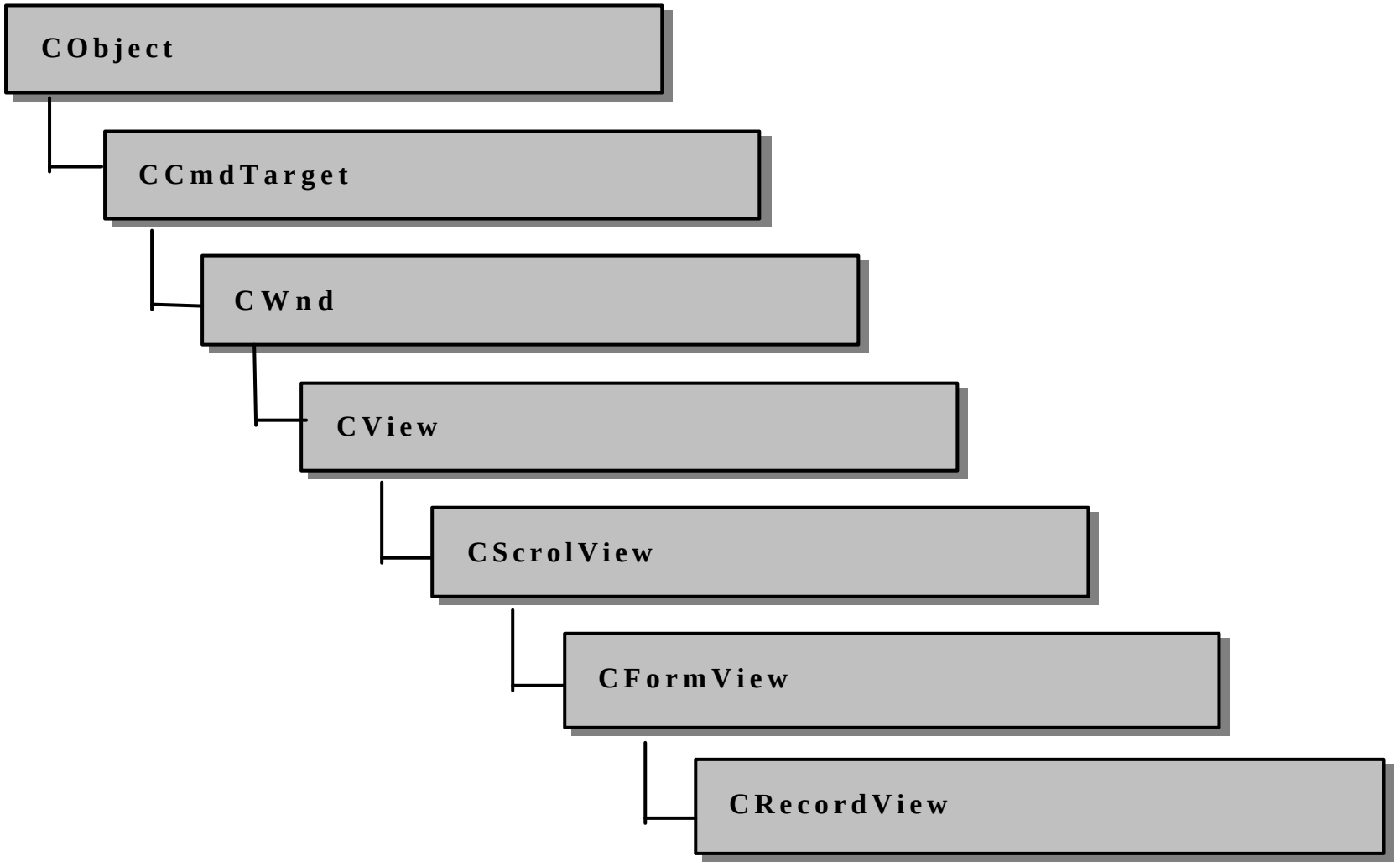
注意，在排序字符串中不要包括 ORDER BY 关键字，框架会提供这个关键字。

示例

```
// CRecordset::m_strSort 示例
CCustSet rsCustSet( NULL );
// 设置排序字符串
rsCustSet.m_strSort = "District, Last_Name";
// 运行整理后的查询
rsCustSet.Open( CRecordset::snapshot, "Customers" );
```

请参阅　CRecordset::m_strFilter, CRecordset::Requery

CRecordView



一个 CRecordView 对象是在控件中显示数据库记录的视。这种视是一种直接连

接到一个 CRecordset 对象上的格式视。此视是从一个对话框模板资源创建的，并将 CRecordSet 对象的字段显示在对话框模板的控件中。CRecordView 对象利用对话框数据交换（DDX）和记录字段交换（RFX）机制，使格式上的控件和记录集的字段之间的数据移动自动化。CRecordView 还提供了完成移动到第一个，下一个，上一个或最后一个记录的缺省实现，和一个用于更新视上面的当前记录的接口。

> **注意** 如果你正在使用数据访问对象（DAO）类，而不是使用开放数据库连接（ODBC）类，则使用 CDaoRecordView 来代替。

创建应用程序的最常用的方法是利用 AppWizard。AppWizard 创建记录视类及与其相关联的记录集类，作为基本起始程序的一部分。如果没有用 AppWizard 来创建记录视类，可以用 ClassWizard 在以后再创建记录视类。如果只是简单地需要一种格式，那么 AppWizard 方式要简单些。ClassWizard 可以让你决定

在以后开发过程中使用一个记录视。利用 ClassWizard 来分别创建一个记录视和一个记录集，然后连接它们，是一种灵活的方式，因为它在命名记录集类及其.H 或.CPP 文件时给予你较多的控制。这种方式也使得在同一个记录类上可以用多个记录视。

为使得在记录视中记录之间的移动对终端用户变得比较容易，AppWizard 为移动到第一个，下一个，前一个或最后一个记录的操作创建菜单（和可选工具条）资源。如果是用 ClassWizard 创建一个记录视类，应用程序需要自己利用菜单和位图编辑器来创建这些资源。

CRecordView 记载用户在记录集中的位置，以便记录视可以更新用户界面。当用户移动到记录集任何一端时，记录集即禁止用户界面对象---例如菜单项或工具条---以阻止朝同一个方向再移动下去。

```
#include <afxdb.h>
```

请参阅 CRecordset，CFormView

CRecordView 类成员

Construction	
CrecordView	构造一个 CRecordView 对象
Attributes	
OnGetRecordset	返回一个指向 CRecordset 派生类对象的指针。ClassWizard 可为应用程序重载此函数，并在需要时创建记录集
IsOnFirstRecord	如果当前记录是相关记录集中的第一个记录，则该函数返回非零值
IsOnLastRecord	如果当前记录是相关记录集中的最后一个记录，则该函数返回非零值
Operations	
OnMove	如果当前记录已经改变，则在数据源上更新该记录，然后移动到指定记录（下一个，前一个，第一个或最后一个）

成员函数

CRecordView::CRecordView

CRecordView(LPCSTR *lpszTemplateName*);
CRecordView(UINT *nIDTemplate*);

参数

lpszTemplateName

包含一个以 NULL **结尾的字符串，此字符串是一个对话框模板资源的名称。**

nIDTemplate

包含一个对话框模板资源的 ID **号。**

说明

当应用程序创建一个 CRecordView 派生类对象时，可调用此构造函数的任何一种形式来初始化此视对象，并标识该视所基于的对话框资源。应用程序可以按名称（传递字符串参数给构造函数）也可以按其 ID（传递一个无符号整数为参数给构造函数）来标识这个对话框资源。建议使用资源 ID。

> **注意** 应用程序的派生类必须提供自己的构造函数。在派生类的构造函数中，需要用资源名或 ID 为参数来调用 CRecordView:: CRecordView 构造函数，如下面的示例所示。

CRecordView::OnInitialUpdate 调用 UpdateData，后者调用 DoDataExchange。这个初始 DoDataExchange 调用连接 CRecordView 控件（间接地）与 ClassWizard 所创建的 CRecordset 字段数据成员。这些数据成员直到应用程序调用了

CFormView::OnInitialUpdate 成员函数之后才可使用。

注意 如果使用 ClassWizard，此向导定义了一个 enum 值 CRecordView::IDD，并在构造函数的成员初始化列表中说明此值，参看示例中的 IDD_MYFORM。示例说明了如果你利用自己的代码而不使用 ClassWizard 时，如何指定对话框模板资源。

示例

```
CMyRecordView::CMyRecordView()
  :CRecordView( IDD_MYFORM )
{
  //{{AFX_DATA_INIT( CMyRecordView )
    // 注意：ClassWizard 将在此添加成员初始化
  //}}AFX_DATA_INIT
  // 其它的构造代码，比如数据初始化
}
```

请参阅 CRecordset::DoFieldExchange, CView::OnInitialUpdate, CWnd::UpdateData

CRecordView::IsOnFirstRecord

BOOL IsOnFirstRecord();

返回值

如果当前记录是记录集中的第一个记录，则返回非零值；否则返回 0。

说明

此成员函数用来确定当前记录是否是与此记录视相关联的记录集对象中的第一个记录。该函数可以用于编写应用程序自己实现的命令更新处理函数，代替 ClassWizard **编制的缺省命令更新处理函数。**

如果用户移动到了第一个记录，则框架将禁止用于移动到第一个记录或前一个记录的用户界面对象。

请参阅 CRecordView::OnMove, CRecordView::IsOnLastRecord, CRecordset::IsBOF, CRecordset::GetRecordCount

CRecordView::IsOnLastRecord

BOOL IsOnLastRecord();

返回值

如果当前记录是记录集中的最后一个记录，则返回非零值；否则返回 0。

说明

此成员函数用来确定当前记录是否是与此记录视相关联的记录集对象中的最后

一个记录。该函数可以用来编写应用程序实现的缺省命令更新处理函数，使ClassWizard能支持在记录间移动的用户界面。

警告　此函数的结果是可靠的，但是在这样的情况下，即只有当用户已经移动到了记录集的结尾记录后，记录视才能检测到记录集的结尾，函数的结果是不可靠的。在此情况下，用户必须移过最后一个记录后，记录视才能获知它必须禁止用于移动到下一个记录或最后一个记录的任何界面对象。如果用户移过最后一个记录，又移回到最后一个记录（或最后一个记录的前面），记录视可以跟踪用户在记录集中的位置，并正确地禁止用户界面对象。在调用实现函数 OnRecordLast（该函数处理ID_RECORD_LAST 命令）或 CRecordset::MoveLast 函数之后，IsOnLastRecord也不可靠。

请参阅　CRecordView::OnMove, CRecordView::IsOnFirstRecord,

CRecordset::IsEOF,
CRecordset::GetRecordCount

CRecordView::OnGetRecordset

virtual CRecordset* OnGetRecordset() = 0;

返回值

如果成功地创建了对象，则返回一个指向 CRecordset **派生对象的指针；否则返回一个** NULL **指针。**

说明

此函数返回一个指向与此记录视相关联的 CRecordset **派生对象的指针。应用程序必须重载这个成员函数来构造或获取一个记录集对象，并返回一个指向该对象的指针。如果应用程序利用** ClassWizard **来声明自己的记录视类，则该向导**

为应用程序编写一个缺省的重载。ClassWizard 缺省实现的 OnGetRecordset 函数返回存储在记录视中的记录集指针（如果存在的话）。如果不存在，则它构造一个具有你在 ClassWizard 所指定的类型的记录集对象,并调用该对象的 Open 成员函数打开表或运行查询，然后返回一个指向此记录集对象的指针。

请参阅 CRecordset, CRecordset::Open

CRecordView::OnMove

```
virtual BOOL OnMove( UINT nIDMoveCommand );
   throw( CDBException );
```

返回值

如果移动成功则返回非零值；否则，如果移动请求被否决则返回 0。

参数

nIDMoveCommand

是下列标准命令 ID **值之一：**

- ID_RECORD_FIRST **移动到记录集中的第一个记录。**
- ID_RECORD_LAST **移动到记录集中的最后一个记录。**
- ID_RECORD_NEXT **移动到记录集中的下一个记录。**
- ID_RECORD_PREV **移动到记录集中的前一个记录。**

说明

此成员函数用来移动到记录集中的一个不同的记录，并将该记录的字段显示在记录视的控件中。缺省实现是调用与此记录视相关联的 CRecordset **对象的相应的** Move **成员函数。**

缺省的，如果用户在记录视中改变了当前记录，则 OnMove 将更新数据源上的当前记录。

AppWizard 创建一个带有 First Record（第一个记录），Last Record（最后一个记录），Next Record（下一个记录），和 Previous Record（前一个记录）等菜单项的菜单。如果选择 Initial Toolbar 选项，则 AppWizard 还创建一个带有对应于这些命令的工具条。

如果你移动过了记录集中的最后一个记录，记录视仍显示最后一个记录。如果向前移动超过了第一个记录，则记录视仍显示第一个记录。

> 警告　如果记录集中没有记录，则调用 OnMove 将抛出一个异常。在执行任何移动操作之前，可以先调用相应的适当用户界面更新处理函数---OnUpdateRecordFirst，OnUpdateRecordLast，OnUpdateRecord 或 OnUpdateRecordPrev 来确定记录集中是否还有记录。

请参阅 CRecordset::Move

CRect

CRect 类与 Windows RECT 结构相似，并且还包括操作 CRect 对象和 Windows RECT 结构的成员函数。

在传递 LPRECT,LPCRECT 或 RECT 结构作为参数的任何地方，都可以传递 CRect 对象来代替。

> **注意** 这个类是从 tagRECT 结构派生而来的。（tagRECT 是 RECT 结构的不太常用的别名。）这意味着 RECT 结构的数据成员（left, top, right, 和 bottom）也是 CRect 的可访问数据成员。

一个 CRect 包含用于定义矩形的左上角和右下角点的成员变量。

当指定一个 CRect 时，必须谨慎地构造它，以使它符合规范---也就是说，使其左坐标值小于右坐标值，使顶坐标值小于底坐标值。例如，左上角为（10，10），右下角为（20，20）就定义了一个符合规范的矩形，但是左上角为（20，20）而右下角为（10，10）的值就定义了一个不符合规范的矩形。如果矩形是不符合规范的，则 CRect 的许多成员函数都会符合不正确的结果。（参见 CRect::NormalizeRect 可以得到这些函数的列表。）在你调用一个要求符合规范的矩形的函数之前，你可以通过调用 NormalizeRect 函数来使不符合规范的矩形成为符合规范的矩形。

当用成员函数 CDC::DPtoLP 和 CDC::LPtoDP 来处理 CRect 时要小心。如果显示环境的映射模式 y-extent 是负的，就像在 MM_LOENGLISH 中一样，则 CDC::DPtoLP 将转换 CRect，以使它的顶部坐标大于底部坐标。然后像 Height 和 Size 这样的函数将返回负值作为转换后的矩形的高度，则此矩形将是不符合

规范的。

当使用重载的 CRect 操作符时，第一个操作数必须是一个 CRect；第二个操作数可以是一个 RECT 结构或一个 CRect 对象。

include < afxwin.h >

请参阅 CPoint，CSize，RECT

CRect 类成员

构造	
Crect	构造一个 Crect 对象
运算	
Width	计算 CRect 的宽度
Height	计算 CRect 的高度
Size	计算 CRect 的大小

续表

TopLeft	返回 CRect 的左上角点
BottomRigh	返回 CRect 的右下角点
CenterPoint	返回 CRect 的中心点
IsRectEmpty	确定 CRect 是否是空的。如果 CRect 的宽度和/或高度为 0，则它是空的
IsRectNull	确定 CRect 的 top，bottom，left，和 right 是否都等于 0
PtInRect	确定指定的点是否在 CRect 之内
SetRect	设置 CRect 的尺寸
SetRectEmpty	设置 CRect 为一个空的矩形（所有的坐标都等于 0）
CopyRect	将一个源矩形的尺寸拷贝到 CRect
EqualRect	确定 CRect 是否等于给定的矩形
InflateRect	增加 CRect 的宽度和高度
DeflateRect	减少 CRect 的宽度和高度
NormalizeRect	使 CRect 的高度和宽度返回规范
OffsetRect	将 CRect 移动到指定的偏移
SubtractRect	从一个矩形中减去另一个矩形
IntersectRect	设置 CRect 等于两个矩形的交集
UnionRect	设置 CRect 等于两个矩形的并集

运算符	
Operator LPCRECT	将一个 CRect 转换为一个 LPCRECT
Operator LPRECT	将一个 CRect 转换为一个 LPRECT
Operator =	将一个矩形的尺寸拷贝到 CRect
Operator ==	确定 CRect 是否与一个矩形相等
Operator !=	确定 CRect 是否不等于另一个矩形
Operator +=	使 CRect 增加指定的偏移，或使 CRect 放大
Operator -=	从 CRect 中减去指定的偏移，或缩小 CRect
Operator &=	设置 CRect 等于 CRect 和某个矩形的交
Operator |=	设置 CRect 等于 CRect 和某个矩形的并
Operator +	增加给定偏移量到 CRect，并返回得到的 CRect 对象
Operator –	从 CRect 中减去给定偏移量，并返回得到的 CRect 对象
Operator &	创建 CRect 与某个矩形的交，并返回得到的 CRect 对象
Operator |	创建 CRect 与某个矩形的并，并返回得到的 CRect 对象

成员函数

CRect::BottomRight

```
CPoint& BottomRight( );
const CPoint& BottomRight( ) const;
```

返回值

返回矩形的右下角点的坐标。

说明

返回的右下角坐标是一个指向 CPoint 对象的引用，该对象包含在 CRect 对象中。

可以使用这个函数来获取或设置矩形的右下角坐标。要用这个函数来设置右下角。可以将此函数放在赋值操作符的左边。

请参阅 CRect::TopLeft, CPoint, CRect::CenterPoint

CRect::CenterPoint

CPoint CenterPoint() const;

返回值

返回一个 CPoint **对象，该对象是** CRect **的中心点。**

说明

要计算 CRect **的中心点，将其左、右坐标值相加再除** 2**，将其上、下值相加再除** 2**。**

请参阅 CRect::Width, CRect::Height, CRect::Size, CRect::TopLeft, CRect::BottomRight, CRect::IsRectNull, CPoint

CRect::CopyRect

void CopyRect(LPCRECT *lpSrcRect*);

参数

lpSrcRect

指向要拷贝的 RECT **结构或** CRect **对象的指针。**

说明

此成员函数用来将 *lpSrcRect* **矩形拷贝到** CRect **中。**

请参阅 CRect::CRect, CRect::operator =, CRect::SetRect, CRect::SetRectEmpty

CRect::CRect

CRect( );
CRect(int *l*, int *t*, int *r*, int *b*);

CRect(const RECT& *srcRect*);
CRect(LPCRECT *lpSrcRect*);
CRect(POINT point, SIZE *size*);
CRect(POINT *topLeft*, POINT *bottomRight*);

参数

l

指定 CRect **的左边位置。**

t

指定 CRect **的上边位置。**

r

指定 CRect **的右边位置。**

b

指定 CRect **的底部位置。**

srcRect

对指定的 CRect **对象的坐标的** RECT **结构的引用。**

lpSrcRect

指向给定的 CRect 对象的坐标的 RECT 结构。

point

指定要构造的矩形的原点。该参数值对应于矩形的左上角。

size

指定要构造的矩形的从左上角到右下角的偏移量。

topLeft

指定 CRect 的左上角位置。

bottomRight

指定 CRect 的右下角位置。

说明

此成员函数用来构造一个 CRect 对象。如果没有给出参数，则不初始化 left，top，right，和 bottom 成员。

CRect(const RECT&)和 CRect(LPRECT)构造函数指向一个 CopyRect。其它的构造函数直接初始化对象的成员变量。

请参阅 CRect::SetRect, CRect::CopyRect, CRect::operator =, CRect::SetRectEmpty

CRect::DeflateRect

void DeflateRect(int *x*, int *y*);
void DeflateRect(SIZE *size*);
void DeflateRect(LPCRECT *lpRect*);
void DeflateRect(int *l*, int *t*, int *r*, int *b*);

参数

x

指定缩小 CRect 的左和右边的单位数。

y

指定缩小 CRect 的上、下边的单位数。

size

一个指定缩小 CRect 的单位数的 SIZE 或 CSize。*cx* 值指定缩小左、右边的单位数，*cy*` ` 指定缩小上、下边的单位数。

lpRect

指向一个 RECT 结构或 CRect，指定缩小每一边的单位数。

l

指定缩小 CRect 左边的单位数。

t

指定缩小 CRect 上边的单位数。

r

指定缩小 CRect 右边的单位数。

b

指定缩小 CRect 下边的单位数。

说明

DeflateRect 通过将 CRect 的边向其中心移动来缩小它。为了做到这一点，DeflateRect 将单位数增加到矩形的左边和上边，从右边和下边减去单位数。DeflateRect的参数是有符号的值；正值缩小 CRect，而负值则放大它。

前两个重载函数使 CRect 相对的两对边都缩小，因此 CRect 的总宽度减小了两倍 *x*（或 *cx*），总高度减小了两倍 *y*（或 *cy*）。其它两个重载函数使 CRect 的边相对独立的缩小。

请参阅 CRect::InflateRect, CRect::operator -, CRect::operator -=, ::InflateRect

CRect::EqualRect

BOOL EqualRect(LPCRECT *lpRect*) const;

返回值

如果两个矩形具有相同的上、左、下、右边值，则返回非零值；否则返回 0。

注意 两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

参数

lpRect

指向一个 RECT **结构或** CRect **对象，该对象或结构包含了一个矩形的左上角和右下角的坐标。**

请参阅 CRect::operator ==, CRect::operator !=, CRect::NormalizeRect, ::EqualRect

CRect::Height

int Height() const;

返回值

返回 CRect **的高度。**

说明

此成员函数通过从 CRect **的下边值中减去上边值来计算** CRect **的高度。结果值可以是负数。**

注意　这个矩形必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect **来使矩形规范化。**

请参阅　CRect::Width, CRect::Size, CRect::CenterPoint, CRect::IsRectEmpty, CRect::IsRectNull, CRect::NormalizeRect

CRect::InflateRect

void InflateRect(int *x*, int *y*);
void InflateRect(SIZE *size*);
void InflateRect(LPCRECT *lpRect*);
void InflateRect(int *l*, int *t*, int *r*, int *b*);

参数

x

指定扩大 CRect **左和右边的单位数。**

y

指定扩大 CRect **上、下边的单位数。**

size

一个指定扩大 CRect **的单位数的** SIZE **或** CSize**。***cx* **值指定扩大左、右边的单位数，***cy* **指定扩大上、下边的单位数。**

lpRect

指向一个 RECT 结构或 CRect，指定扩大每一边的单位数。

l

指定扩大 CRect 左边的单位数。

t

指定扩大 CRect 上边的单位数。

r

指定扩大 CRect 右边的单位数。

b

指定扩大 CRect 下边的单位数。

说明

InflateRect 通过将 CRect 的边向远离其中心的方向移动来扩大它。为了做到这一点，InflateRect 从矩形的左边和上边减去单位数，将单位数增加到右边和下

边。InflateRect 的参数是有符号的值；正值扩大 CRect，而负值则缩小它。

前两个重载函数使 CRect 相对的两对边都扩大，因此 CRect 的总宽度增加了了两倍 *x*（或 *cx*），总高度增加了两倍 *y*（或 *cy*）。其它两个重载函数使 CRect 的边则是相对独立的扩大。

请参阅 CRect::DeflateRect, CRect::operator +, CRect::operator +=, ::InflateRect

CRect::IntersectRect

BOOL IntersertRect (LPCRECT *lpRect1*, LPCRECT *lpRect2*);

返回值

如果交不为空，则返回非零值；否则，如果交为空则返回 0。

参数

lpRect1

指向一个 RECT **结构或** CRect **对象，该对象或结构包含了一个源矩形。**

lpRect2

指向一个 RECT **结构或** CRect **对象，该对象或结构包含了一个源矩形。**

说明

此函数使 CRect **等于两个现有矩形的交。交是同时包含在两个现有矩形中的最大矩形。**

注意 **两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用** NormalizeRect **来使矩形规范化。**

请参阅 CRect::operator &=, CRect::operator &, CRect::UnionRect,

CRect::SubtractRect,
CRect::NormalizeRect, ::IntersectRect

CRect::IsRectEmpty

BOOL IsRectEmpty() const;

返回值

如果 CRect **是空的则返回非零值；否则，如果** CRect **不是空的则返回** 0。

说明

此函数用来确定 CRect **是否是空的。如果一个矩形的宽度和/或高度是** 0 **或负值，则称这个矩形为空的。与** IsRectNull **不同，**IsRectNull **用来确定是否矩形的所有坐标都是零。**

注意　这个矩形必须是符合规范的，否则此函数将失败。你可以在调用

此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅 CRect::IsRectNull, CRect::SetRectEmpty, CRect::NormalizeRect, ::IsRectEmpty

CRect::IsRectNull

BOOL IsRectNull() const;

返回值

如果 CRect 的上、左、下和右边的值都等于 0，则返回非零值；否则返回 0。

说明

此函数用来确定 CRect 的上、左、下和右边的值是否都等于 0。与 IsRectEmpty 不同，IsRectEmpty 用来确定矩形是否为空。

请参阅 CRect::IsRectEmpty, CRect::SetRectEmpty

CRect::NormalizeRect

void NormalizeRect( );

说明

此函数用来使 CRect **符合规范，使其高度和宽度均为正值。矩形被从四个象限规范化，通常这四个象限就是** Windows **用来确定坐标的。**NormalizeRect **比较上、下坐标的值，如果上坐标值大于下坐标的值，则将它们互换。类似的，如果左坐标的值大于右坐标的值，则将左、右坐标互换。当处理不同的映射模式和转化的矩形时，这个函数是很有用的。**

注意 **下面的** CRect **成员函数为了正确地工作，需要规范化的矩形，它们是：**Height, Width, Size, IsRectEmpty, PtInRect, EqualRect,

UnionRect, IntersectRect, SubtractRect, operator ==, operator !=, operator |, operator |=, operator &, 和 operator &=。

CRect:: OffsetRect

```
void OffsetRect( int x, int y );
void OffsetRect( POINT point );
void OffsetRect( SIZE size );
```

参数

x

指定要往左或右移动的数量。要往左时此值必须是负值。

y

指定要往上或下移动的数量。要往上时此值必须是负值。

point

包含一个 POINT 结构或 CPoint 对象，该结构或对象指定了要移动的尺寸。

size

包含一个 SIZE 结构或 CSize 对象，指定要移动的尺寸。

说明

此函数用来将 CRect 移动指定的偏移量。将 CRect 沿 x 轴移动 x 个单位，沿 y 轴移动 y 个单位。x 和 y 参数都是有符号的值，因此 CRect 可以向左或向右，向上或向下移动。

请参阅 CRect::operator +, CRect::operator +=, CRect::operator -, CRect::operator -=

CRect::PtInRect

BOOL PtInRect(POINT *point*) const;

返回值

如果点位于 CRect **中，则返回非零值；否则返回** 0。

参数

point

包含一个 POINT **结构或** CPoint **对象。**

说明

此函数用来确定一个指定的点是否在 CRect **内。如果这个点在** CRect **的左边和上边上，或在四个边之内，则称此点在** CRect **之内。点在** CRect **的右边或底边上，则称点在** CRect **外。**

注意 **这个矩形必须是符合规范的，否则此函数将失败。你可以在调用**

此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅 CRect::NormalizeRect, ::PtInRect

CRect::SetRect

void SetRect(int *x1*, int *y1*, int *x2*, int *y2*);

参数

x1

指定左上角的 x 坐标。

y2

指定左上角的 y 坐标。

x2

指定右下角的 x 坐标。

y2

指定右下角的 y 坐标。

说明

此函数将 CRect 的各个坐标设置为指定的坐标值。

请参阅 CRect::CRect, CRect::operator =, CRect::CopyRect, CRect::SetRectEmpty, ::SetRect

CRect::SetRectEmpty

void SetRectEmpty();

说明

此函数通过将 CRect 的所有坐标设置为零来使 CRect 成为一个空矩形。

请参阅 CRect::CRect, CRect::SetRect, CRect::CopyRect, CRect::operator =,

CRect::IsRectEmpty, CRect::IsRectNull, ::SetRectEmpty

CRect::Size

```
CSize Size() const;
```

返回值

返回一个包含 CRect **的尺寸的** CSize **对象。**

说明

返回值的 *cx* **和** *cy* **成员包含了** CRect **的高度和宽度。高度和宽度都可以是负值。**

> **注意** **这个矩形必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用** NormalizeRect **来使矩形规范化。**

请参阅 CRect::Height, CRect::Width, CRect::IsRectEmpty, CRect::IsRectNull,

CRect::NormalizeRect

CRect::SubtractRect

BOOL SubtractRect(LPCRECT *lpRectSrc1*, LPCRECT *lpRectSrc2*);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

lpRectSrc1

指向一个 RECT **结构或** CRect **对象，将从这个结构或对象中减去某个矩形。**

lpRectSrc2

指向一个 RECT **结构或** CRect **对象，将从由** *lpRectSrc1* **参数指定的矩形中减去这个结构或对象。**

说明

该函数使一个 CRect 对象的尺寸等于从 *lpRectSrc1* 中减去 *lpRectSrc2* 的差，这个差值是一个包含所有在 *lpRectScr1* 内，而不在 *lpRectScr1* 和 *lpRectScr2* 的交之中的点的最小矩形。

如果由 *lpRectScr2* 指定的矩形和由 *lpRectScr1* 指定的矩形没有在 x 或 y 方向之一的方向上重叠，则由 *lpRectScr1* 指定的矩形将没有改变。

例如，如果 *lpRectScr1* 是（10，10，100，100），*lpRectScr2* 是（50，50，150，150，）则当函数返回时，由 *lpRectScr1* 指定的矩形将没有改变。如果 *lpRectScr1* 是（10，10，100，100，），*lpRectScr2* 是（50，10，150，150），则当函数返回时，由 *lpRectScr1* 指定的矩形将包含坐标（10，10，50，100）。

SubtractRect 与操作符 - 和操作符 -= 是不一样的。这两个操作符都不调用

SubtractRect。

注意　这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅　CRect::operator -, CRect::operator -=, CRect::IntersectRect, CRect::UnionRect,
CRect::NormalizeRect, ::SubtractRect

CRect::TopLeft

```
CPoint& TopLeft( );
const CPoint& TopLeft( ) const;
```

返回值

返回矩形的左上角坐标。

说明

此函数返回的坐标是一个对 CPoint **对象的引用，该对象包含在** CRect **中。**

你可以使用此函数来获取或设置矩形的左上角。通过在赋值操作符的左边使用这个函数，可以设置矩形的左上角。

请参阅 CRect::BottomRight, CPoint, CRect::CenterPoint

CRect::UnionRect

BOOL UnionRect(LPCRECT *lpRect1*, LPCRECT *lpRect2*);

返回值

如果两个矩形的并不是空，则返回非零值；否则，如果并是空，则返回 0**。**

参数

lpRect1

指向一个 RECT **或** CRect**，包含了一个源矩形。**

lpRect2

指向一个 RECT **或** CRect**，包含了一个源矩形。**

说明

此函数使 CRect **的尺寸等于两个源矩形的并。并就是包含两个源矩形的最小矩形。**Windows**忽略空矩形，即没有高度也没有宽度的矩形。**

注意 **这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用** NormalizeRect **来使矩形规范化。**

请参阅 CRect::operator |=, CRect::operator |, CRect::IntersectRect, CRect::SubtractRect,

CRect::NormalizeRect, ::UnionRect

CRect::Width

int Width() const;

返回值

返回 CRect **的宽度。**

说明

此函数用右坐标值减去左坐标值，来计算 CRect **的宽度。这个宽度可以是负值。**

注意 **这个矩形必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用** NormalizeRect **来使矩形规范化。**

请参阅 CRect::Height, CRect::Size, CRect::CenterPoint, CRect::IsRectEmpty,

CRect::IsRectNull, CRect::NormalizeRect

操作符

CRect::operator LPCRECT

operator LPCRECT( ) const;

说明

此操作符将一个 CRect **转化为一个** LPCRECT**。当使用此函数时，不需要使用地址操作符（**&**）。当你传递一个** CRect **对象给一个需要** LPCRECT **的函数时，此操作符被自动使用。**

请参阅 CRect::operator LPRECT

CRect::operator LPRECT

Operator LPRECT();

说明

此操作符将一个 CRect **转化为一个** LPRECT**。当使用此函数时，不需要使用地址操作符（**&**）。当你传递一个** CRect **对象给一个需要** LPRECT **的函数时，此操作符被自动使用。**

请参阅 CRect::operator LPCRECT

CRect:: operator =

void operator =(const RECT& *srcRect*);

参数

srcRect

指向一个源矩形。可以是一个 RECT 或 CRect。

说明

此操作符将 *srcRect* 分配给 CRect。

请参阅 CRect::CRect, CRect::SetRect, CRect::CopyRect, CRect::SetRectEmpty, ::CopyRect

CRect:: operator ＝＝

BOOL operator ==(const RECT& *rect*) const;

返回值

如果相等则返回非零值；否则返回 0。

参数

rect

指向一个源矩形。可以是一个 RECT **或** CRect**。**

说明

此操作符通过比较 rect **与** CRect **左上角和右下角的坐标值来确定它们是否相等。**

注意 **这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用** NormalizeRect **来使矩形规范化。**

请参阅 CRect::operator !=, CRect::NormalizeRect, ::EqualRect

CRect:: operator !=

BOOL operator !=(const RECT& *rect*) const;

返回值

如果不相等则返回非零值；否则返回 0。

参数

rect

指向一个源矩形。可以是一个 RECT **或** CRect。

说明

此操作符通过比较 *rect* **与** CRect **左上角和右下角的坐标值来确定它们是否不相等。**

注意 这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅 CRect::operator ==, CRect::NormalizeRect, ::EqualRect

CRect:: operator +=

```
void operator +=( POINT point );
void operator +=( SIZE size );
void operator +=( LPCRECT lpRect );
```

参数

point

一个 POINT 结构或 CPoint 对象，指定了矩形要移动的单位数。

size

一个 SIZE 结构或 CSize 对象，指定了矩形要移动的单位数。

lpRect

指向一个 RECT 或 CRect 对象，包含了要扩大 CRect 的每一边的单位数。

说明

前两种重载形式将 CRect 移动指定的偏移量。参数的 *x* 和 *y*（或 *cx* 和 *cy*）被增加到 CRect。

第三种重载形式将 CRect 扩大参数中每一个成员所指定的单位数。

请参阅 CRect::OffsetRect, CRect::InflateRect, CRect::operator +, CRect::operator -=

CRect:: operator -=

```
void operator -=( POINT point );
void operator -=( SIZE size );
void operator -=( LPCRECT lpRect );
```

参数

point

一个 POINT 结构或 CPoint 对象，指定了矩形要移动的单位数。

size

一个 SIZE 结构或 CSize 对象，指定了矩形要移动的单位数。

lpRect

指向一个 RECT 或 CRect 对象，包含了要扩大 CRect 的每一边的单位数。

说明

前两种重载形式将 CRect 移动指定的偏移量。从 CRect 中减去参数的 x 和 y（或 cx 和 cy）。

第三种重载形式将 CRect 缩小参数中每一个成员所指定的单位数。注意，这个重载函数像 DeflateRet。

请 参 阅 CRect::OffsetRect, CRect::DeflateRect, CRect::SubtractRect, CRect::operator -,
CRect::operator +=

CRect:: operator &=

void operator &=(const RECT& *rect*);

参 数

rect

包 含 一 个 RECT **或** CRect。

说 明

此 操 作 符 用 来 使 CRect **等 于** CRect **和** *rect* **的 交 。 交 是 被 两 个 源 矩 形 都 包 含 的 最 大 矩 形 。**

注意　这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅　CRect::operator &, CRect::operator |=, CRect::IntersectRect, CRect::NormalizeRect, ::IntersectRect

CRect:: operator |=

void operator |=(const RECT& *rect*);

参数

rect

包含一个 RECT 或 CRect。

说明

此操作符用来使 CRect 等于 CRect 和 *rect* 的并。并是包含两个源矩形的最小矩

形。

注意 这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅 CRect::operator |, CRect::operator &=, CRect::UnionRect, CRect::NormalizeRect, ::UnionRect

CRect:: operator +

```
CRect operator +( POINT point ) const;
CRect operator +( LPCRECT lpRect ) const;
CRect operator +( SIZE size ) const;
```

返回值

返回一个 CRect，**它是将** CRect **移动或扩大参数中指定的单位数之后的结果。**

参数

point

一个 POINT 结构或 CPoint 对象，指定了返回值要移动的单位数。

size

一个 SIZE 结构或 CSize 对象，指定了返回值要移动的单位数。

lpRect

指向一个 RECT 或 CRect 对象，包含了要扩大返回值的每一边的单位数。

说明

前两种重载形式返回的 CRect 对象是 CRect 移动指定偏移后的结果。参数的 *x* 和 *y*（或 *cx* 和 *cy*）被增加到 CRect 的位置上。

第三种重载形式返回一个新的 CRect 对象，此对象是扩大了参数的每一个成员指定的单位数之后的 CRect。

请 参 阅 CRect::operator +=, CRect::operator -, CRect::OffsetRect, CRect::InflateRect

CRect:: operator -

CRect operator – (POINT *point*) const;
CRect operator – (SIZE *size*) const;
CRect operator – (LPCRECT *lpRect*) const;

返 回 值

返回一个 CRect**，它是将** CRect **移动或缩小参数中指定的单位数之后的结果。**

参 数

point

一个 POINT **结构或** CPoint **对象，指定了返回值要移动的单位数。**

size

一个 SIZE 结构或 CSize 对象，指定了返回值要移动的单位数。

lpRect

指向一个 RECT 或 CRect 对象，包含了要扩大返回值的每一边的单位数。

说明

前两种重载形式返回的 CRect 对象是 CRect 移动指定偏移后的结果。参数的 *x* 和 *y*（或 *cx* 和 *cy*）被从 CRect 的位置上减去。

第三种重载形式返回一个新的 CRect 对象，此对象是缩小了参数的每一个成员指定的单位数之后的 CRect。注意，这个重载函数类似于 DeflateRect，而不是 SubtractRect。

请参阅 CRect::operator -=, CRect::operator +, CRect::OffsetRect, CRect::DeflateRect,
CRect::SubtractRect

CRect:: operator &

CRect operator &(const RECT& *rect2*) const;

返回值

返回一个 CRect，**它是** CRect **和** *rect2* **的交。**

参数

rect2

包含一个 RECT **或** CRect。

说明

此操作符返回一个 CRect，**它是** CRect **和** *rect2* **的交。交是被两个矩形包含的最大矩形。**

注意 这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅 CRect::IntersectRect, CRect::operator &=, CRect::operator |, CRect::NormalizeRect

CRect:: operator |

CRect operator |(const RECT& *rect2*) const;

返回值

返回一个 CRect，**它是** CRect **和** *rect2* **的并**。

参数

rect2

包含一个 RECT **或** CRect。

说明

此操作符返回一个 CRect，它是 CRect 和 *rect2* 的并。并是包含两个矩形的最小矩形。

注意 这两个矩形都必须是符合规范的，否则此函数将失败。你可以在调用此函数之前，调用 NormalizeRect 来使矩形规范化。

请参阅 CRect::UnionRect, CRect::operator |=, CRect::operator &, CRect::NormalizeRect

CRectTracker

CRectTracker 没有基类。

CRectTracker 类允许一个项被显示，移动，以不同的方式改变大小。虽然 CRectTracker 类是设计来支持用户以图形化界面与 OLE 项交互的，但是它的使用不仅限于支持 OLE 的应用程序。它可以使用在任何需要用户界面的地方。

CRectTracker 的边框可以是实线，也可以是点线。可给予项一种阴影式边框或用一种阴影样式覆盖项，用来指示项的不同状态。你可以在项的外界或内部放置八个调整大小把手。（有关八个调整大小把手的解释，参见 GetHandleMask。）最后，一个 CRectTracker 允许你在调整项的大小时改变项的方向。

要使用 CRectTracker，首先要构造一个 CRectTracker 对象，并指定用哪种显示状态来初始化。然后，应用程序就可以使用这个界面，提供给用户有关与

CRectTracker 对象相关联的 OLE 项当前状态的直观反馈了。

#include <afxext.h>

请参阅 COleResizeBar, CRect, CRectTracker::GetHandleMask

CRectTracker 类成员

Data Members	
m_nHandleSize	确定调整大小把手的尺寸
m_rect	矩形的以像素表示的当前位置
m_sizeMin	确定矩形宽度和高度的最小值
m_nStyle	跟踪器的当前风格

Construction	
CrectTracker	构造一个 CRectTracker 对象

Operations	
Draw	显示矩形

续表

GetTrueRect	返回矩形的宽度和高度，包括改变大小句柄
HitTest	返回与 CRectTracker 对象关联的光标的当前位置
NormalizeHit	规范化一个单击测试代码
SetCursor	根据光标在矩形上方的位置来设置光标
Track	支持用户操作矩形
TrackRubberBand	支持用户“橡皮筋”似的拉伸选择
Overridables	
AdjustRect	当矩形被改变大小时此函数被调用
DrawTrackerRect	当画一个 CRectTracker 对象的边框时此函数被调用
OnChangedRect	当矩形被改变大小或被移动时，此函数被调用
GetHandleMask	调用此函数来获得一个 CRectTracker 项的调整大小把手的掩码

成员函数

CRectTracker::AdjustRect

virtual void AdjustRect(int *nHandle*, LPRECT *lpRect*);

参数

nHandle

所使用的句柄的索引。

lpRect

指向矩形的当前尺寸的指针。（矩形的尺寸由它的高度和宽度给出。）

说明

当通过使用调整大小把手来改变跟踪矩形的大小时，框架调用此成员函数。只

有当用允许反转的形式来调用 Track 和 TrackRubberBand 时，这个函数的缺省行为才允许改变矩形的方向。

可以重载这个函数来控制在一次拖动操作期间对跟踪矩形进行调整。一种方法就是在返回之前调整由 *lpRect* 指定的坐标。

特殊的特征，如对齐至网格或保持长宽比例，不是由 CRectTracker 直接支持的，可以重载这个函数来实现。

请参阅 CRectTracker::Track, CRectTracker::TrackRubberBand, CRectTracker::OnChangedRect

CRectTracker::CRectTracker

```
CRectTracker( );
CRectTracker( LPCRECT lpSrcRect, UINT nStyle );
```

参数

lpSrcRect

矩形对象的坐标。

nStyle

指定 CRectTracker **对象的风格。下面的风格都是被支持的：**

- CRectTracker::solidLine **矩形的边框用实线。**
- CRectTracker::dottedLine **矩形的边框用点线。**
- CRectTracker::hatchedBorder **矩形的边框用阴影模式。**
- CRectTracker::resizeInside **调整大小把手位于矩形的内部。**
- CRectTracker::resizeOutside **调整大小把手位于矩形的外界。**
- CRectTracker::hatchInside **整个矩形都覆盖了阴影模式。**

说明

此成员函数用来创建并初始化一个 CRectTracker **对象。**

缺省的构造函数用 lpSrcRect **的值来初始化** CRectTracker **对象，并将其它的尺寸初始化为系统的缺省值。如果该对象创建时没有使用参数，则不初始化** m_rect **和** m_nStyle **数据成员。**

请参阅 CRect::CRect

CRectTracker::Draw

void Draw(CDC* *pDC*) const;

参数

pDC

指向绘制所用的设备环境的指针。

说明

此成员函数用来绘制矩形的轮廓线和内部区域。跟踪器矩形的风格决定了如何进行绘制。参见 CRectTracker **的构造函数，可以获得有关可用风格的更多信息。**

请参阅 CRectTracker::DrawTrackerRect, CRectTracker::CRectTracker, CRect::NormalizeRect

CRectTracker::DrawTrackerRect

virtual void DrawTrackerRect(LPCRECT *lpRect*, CWnd* *pWndClipTo*, CDC* *pDC*, CWnd* *pWnd*);

参数

lpRect

指向包含要绘制的矩形的 RECT **的指针。**

pWndClipTo

指向窗口的指针，该窗口用于剪贴矩形。

pDC

指向用于绘制的设备环境的指针。

pWnd

指向窗口的指针，绘制将在此窗口中发生。

说明

在 Track 或 TrackRubberBand 成员函数内，无论何时跟踪矩形的位置发生改变，框架都将调用此成员函数。此成员函数的缺省实现调用了 CDC::DrawFocusRect 函数，该函数用来绘制一个打点矩形。

重载这个函数，可以实现在跟踪期间提供不同的反馈。

请参阅 CRectTracker::Track, CRectTracker::TrackRubberBand, CDC::DrawFocusRect

CRectTracker::GetHandleMask

virtual UINT GetHandleMask( ) const;

返回值

返回一个 CRectTracker **项的调整大小把手的掩码。**

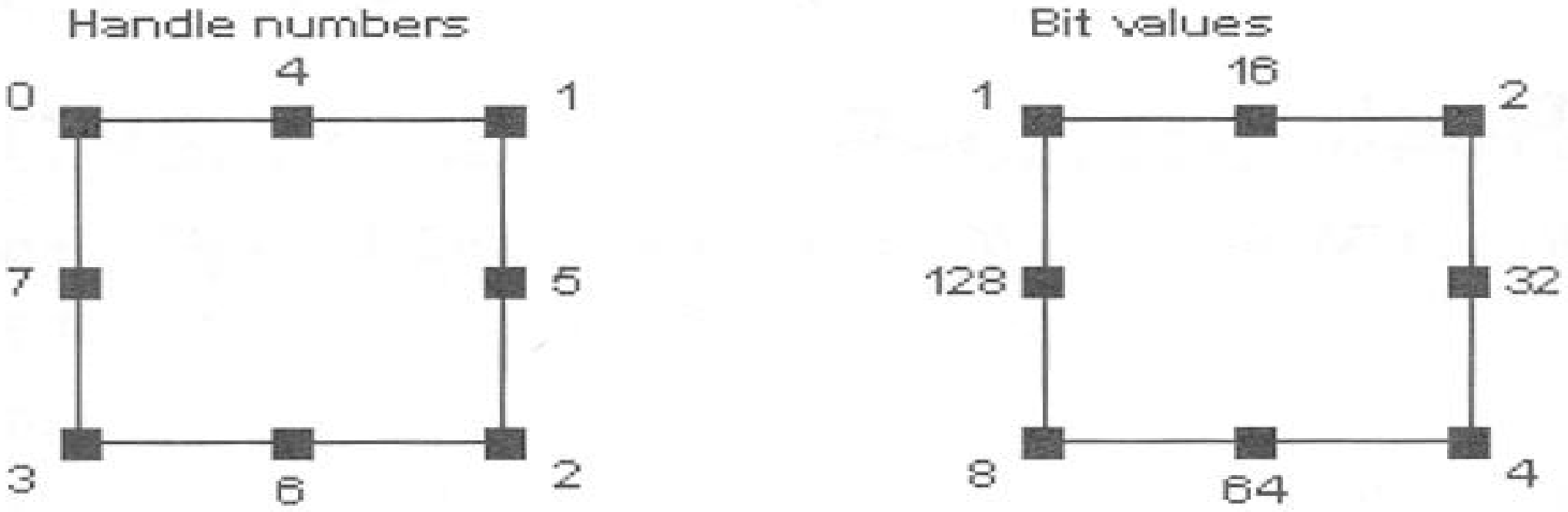
说明

框架调用此成员函数来获取一个矩形的调整大小把手的掩码。

调整大小把手形式在矩形的边和角上，用来让用户控制矩形的形状和尺寸。

每个矩形由 8 **个调整大小把手，标号为** 0－7**。每一个调整大小把手由掩码中的一位代表；该位的值是** 2^n**，这里的** n **是调整大小把手号。位** 0－3 **对应于角上的调整大小把手，从左上角开始，按顺时针方向转。位** 4－7 **对应于边上的调**

整大小把手，从上边开始，按顺时针方向转。下面的图例说明了一个矩形的调整大小把手和它们对应的调整大小把手号和值：



GetHandleMask 的缺省实现返回位的掩码，以显示调整大小把手。如果某一单个的位有效，则将绘制相应的调整大小把手。

重载这个函数，可以实现隐藏或显示指定的句柄。

请参阅 CRectTracker::AdjustRect

CRectTracker::GetTrueRect

void GetTrueRect(LPRECT *lpTrueRect*) const;

参数

lpTrueRect

指向 RECT **结构的指针，该结构将包含** CRectTracker **对象的设备坐标。**

说明

此成员函数用来获取矩形的坐标。矩形的尺寸包括位于外边框上的任何调整大小把手的高度和宽度。当函数返回时，*lpTrueRect* **总是一个用设备坐标表示的规范矩形。**

请参阅 CRect::NormalizeRect

CRectTracker::HitTest

int HitTest(CPoint *point*) const;

返回值

返回的值取决于 CRectTracker::TrackerHit **枚举类型，可以是下列值之一：**

- CRectTracker::hitNothing -1
- CRectTracker::hitTopLeft 0
- CRectTracker::hitTopRight 1
- CRectTracker::hitBottomRight 2
- CRectTracker:hitBottomLeft 3
- CRectTracker:hitTop 4
- CRectTracker:hitRight 5
- CRectTracker:hitBottom 6
- CRectTracker:hitLeft 7
- CRectTracker:hitMiddle 8

参数

point

以设备坐标表示的要测试的点。

说明

此成员函数用来检查用户是否已经捕获一个调整大小把手。

请参阅 CRectTracker::NormalizeHit, CRectTracker::SetCursor

CRectTracker::NomalizeHit

int NormalizeHit(int *nHandle*) const;

返回值

返回规范化句柄的索引。

参数

nHandle

标识用户选择的句柄。

说明

此函数用于转换潜在的翻转句柄。

当在允许反转的情况下调用 CRectTracker::Track **或** CRectTracker::TrackRubberBand **时，矩形有可能按** x **轴，**y **轴或两个轴进行反转。当这种情况发生时，**HitTest **将返回也相对于矩形而反转了的句柄。这不适用于绘制光标反馈，因为反馈取决于矩形的屏幕位置，而不是矩形数据结构中将要被修改的部分。**

请参阅 CRectTracker::HitTest, CRectTracker::Track,

CRectTracker::TrackRubberBand

CRectTracker::OnChangeRect

virtual void OnChangedRect(const CRect& *rectOld*);

参数

rectOld

包含 CRectTracker **对象的旧设备坐标。**

说明

在调用 Track **期间，无论何时跟踪矩形发生变化，框架都将调用** OnChangedRect **函数。在调用该函数时，所有用** DrawTrackerRect **绘制的反馈都已经被删除。此函数的缺省实现不做任何事情。**

当你希望在矩形改变大小后执行某些动作时，你可以重载这个函数。

请 参 阅 CRectTracker::AdjustRect, CRectTracker::Track, CRectTracker::TrackRubberBand

CRectTracker:: SetCursor

BOOL SetCursor(CWnd* *pWnd*, UINT *nHitTest*) const;

返回值

如果前一次单击是在跟踪矩形的上方，则返回非零值；否则返回 0。

参数

pWnd

指向当前包含光标的窗口。

nHitTest

前一次单击测试的结果，从 WM_SETCURSOR **消息返回。**

说明

当光标处于 CRectTracker **对象的区域上方时，此成员函数用来改变光标的形状。**

从处理 WM_SETCURSOR **消息的窗口函数（通常是** OnSetCursor**）内调用此** SetCursor **函数。**

请参阅 CRectTracker::NormalizeHit, CRectTracker::HitTest, CWinApp::LoadCursor, CWnd::OnSetCursor

CRectTracker::Track

BOOL Track(CWnd* *pWnd*, CPoint *point*, BOOL *bAllowInvert* = FALSE, CWnd* *pWndClipTo* = NULL);

返回值

如果按下的是 ESC **键，则跟踪过程停止，跟踪矩形内存储的矩形不变，并且返**

回 0。如果通过移动鼠标并释放鼠标按钮将改变提交，则新的位置和/或大小被记录在跟踪矩形内，并且返回一个非零值。

参数

pWnd

包含矩形的窗口对象。

point

当前鼠标位置相对于客户区的鼠标坐标。

bAllowInvert

如果此参数为 TRUE，则矩形可沿 x 轴，或 y 轴反转；否则，此参数为 FALSE。

pWndClipTo

绘制操作将要被剪贴到的窗口。如果该参数为 NULL，则 *pWnd* 用作剪贴

矩形。

说明

此成员函数显示用于改变矩形大小的用户界面。它通常从应用程序中处理 WM_LBUTTONDOWN 消息的函数（一般是 OnLButtonDown）中调用。

这个函数将捕捉鼠标，直到用户释放鼠标左按钮，单击 ESC 键，或单击鼠标右按钮。当用户移动鼠标光标时，反馈由调用 DrawTracderRect 和 OnChangedRect 来更新。

如果 *bAllowInvert* 时 TRUE，则跟踪矩形可以沿 x 轴或 y 轴反转。

请参阅 CRectTracker::DrawTrackerRect, CRectTracker::OnChangedRect, CRectTracker::CRectTracker, CRectTracker::TrackRubberBand

CRectTracker::TrackRubberBand

BOOL TrackRubberBand(CWnd* *pWnd*, CPoint *point*, BOOL *bAllowInvert* = TRUE);

返回值

如果鼠标被移动并且矩形不是空的，则返回非零值；否则返回 0。

参数

pWnd

包含矩形的窗口对象。

point

当前鼠标相对于客户区的设备坐标。

bAllowInvert

如果是 TRUE，**则矩形可以沿** x **轴或** y **轴反转；否则，其值就是** FALSE。

说明

此成员函数用来进行“橡皮筋似”选择。它通常从处理 WM_LBUTTONDOWN 消息的应用程序函数（一般时 OnLButtonDown）中调用。

这个函数将捕捉鼠标，直到用户释放鼠标左按钮，单击 ESC 键，或单击鼠标右按钮。当用户移动鼠标光标时，反馈由调用 DrawTracderRect 和 onChangedRect 来更新。

通过从右下方选择一个 rubber-band-type 选择来实现跟踪。如果允许反转的话，则可以通过向上至左或向下至右拖动来改变矩形的大小。

请参阅 CRectTracker::DrawTrackerRect, CRectTracker::OnChangedRect, CRectTracker::CRectTracker

数据成员

CRectTracker::m_nHandleSize

说明

此数据成员是 CRectTracker 的调整大小把手的按像素表示的尺寸。用缺省的系统值初始化。

CRectTracker::m_rect

说明

用客户坐标（以像素计）表示的矩形的当前位置。

请 参 阅 CRectTracker::CRectTracker, CRectTracker::Track, CRectTracker::TrackRubberBand

CRectTracker::m_sizeMin

说明

矩形的最小尺寸。缺省值 cx **和** cy **都从边框宽度的系统缺省值计算而来。该数据成员只用于** AdjustRect **成员函数。**

请 参 阅 CRectTracker::Track, CRectTracker::TrackRubberBand, CRectTracker::AdjustRect

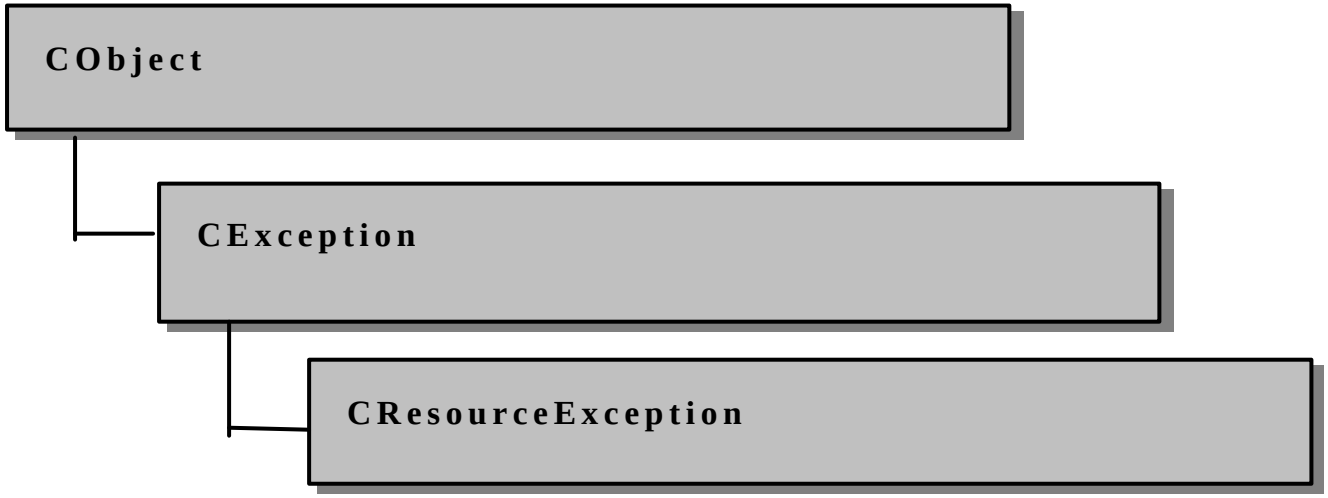
CRectTracker::m_nStyle

说明

矩形的当前风格，各种风格的列表参见 CRectTracker::CRectTracker。

请参阅 RectTracker::CRectTracker, CRectTracker::Draw

CResourceException



当 Windows **不能找到或分配一个被申请的资源时就生成一个** CResourceException **对象。**

#include <afxwin.h>

CresourceException 类成员

构造	
CresourceException	构造一个 CResourceException 对象

成员函数

CResourceException:: CResourceException

CResourceException();

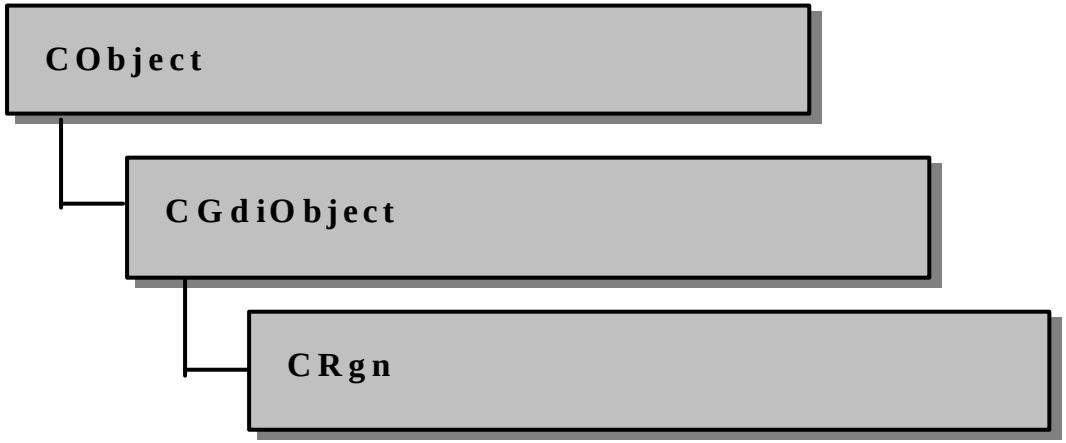
说明

此成员函数用来构造一个 CResourceException **对象。**

不要直接使用这个构造函数，应该直接调用全局函数 AfxThrowResourceException。

请参阅 AfxThrowResourceException，Exception Processing

CRgn



CRgn 类封装了一个 Windows 图形设备接口（GDI）区域。这一区域是某一窗口中的一个椭圆或多边形区域。要使用这个区域，你可以使用类 CRgn 的成员函数以及被定义为类 CDC 的成员函数的剪贴函数。

调用 CRgn 的成员函数就可以创建，修改和检取关于此区域对象的信息。

#include <afxwin.h>

CRgn 类成员

Construction	
CRgn	构造一个 CRgn 对象
Initialization	
CreateRectRgn	用一个矩形区域来初始化一个 CRgn 对象
CreateRectRgnIndirect	用由一个 RECT 结构定义的矩形区域来初始化一个 CRgn 对象
CreateEllipticRgn	用一个椭圆形区域来初始化一个 CRgn 对象
CreateEllipticRgnIndirect	用由一个 RECT 结构定义的椭圆形区域来初始化一个 CRgn 对象
CreatePolygonRgn	用一个多边形区域来初始化一个 CRgn 对象。如果有必要，系统通过在第一个顶点和最后一个顶点之间绘制直线来自动关闭该多边形

续表

CreatePolyPolygonRgn	用一系列封闭的多边形组成的区域来初始化一个CRgn对象。这些多边形可能互不相交或相互重叠
CreateRoundRectRgn	用一个圆角的矩形区域来初始化一个CRgn对象
CombineRgn	设置一个CRgn对象，使它等效于两个指定的CRgn对象的联合
CopyRgn	设置一个CRgn对象，使它为一个指定的CRgn对象的拷贝
CreateFromPath	从被选入给定设备环境的路径创建一个区域
CreateFromData	根据给定的区域和变换数据创建一个区域
Operations	
EqualRgn	检查两个CRgn对象，确定它们是否相等
FromHandle	当给定了一个Windows区域的句柄时返回指向一个CRgn对象的指针
GetRegionData	用描述给定区域的数据来填充指定的缓冲区
GetRgnBox	检取一个CRgn对象的限定矩形的坐标
OffsetRgn	用指定的偏移量移动一个CRgn对象
PtInRegion	确定一个指定的点是否在矩形内

续表

RectInRegion	确定一个指定矩形的任何部分是否都在区域的边界内
SetRectRgn	将 CRgn 对象设置为指定的矩形区域
Operators	
OperatorHRGN	返回包含在 CRgn 对象中的 Windows 句柄

成员函数

CRgn::CombineRgn

int CombineRgn(CRgn* *pRgn1*, CRgn* *pRgn2*, *int nCombineMode*);

返回值

返回最后结果区域的类型。它可以是下列值之一：

- COMPLEXREGION　新区域具有相互重叠的边界。
- ERROR　未创建新区域。
- NULLREGION　新区域为空。
- SIMPLEREGION 新区域没有相互重叠的边界。

参数

pRgn1

表示一个已经存在的区域。

pRgn2

表示一个已经存在的区域。

nCombineMode

指定当组合两个源区域时要执行的操作。它可以是下列值之一：

- RGN_AND　使用两个区域互相重叠的区域（相交区域）。

- RGN_COPY　创建区域 1（由 pRgn1 标识）的一个拷贝
- RGN_DIEF　创建一个区域，该区域由区域 1（由 *pRgn1* 标识）的不是区域 2（由 *pRgn2* 标识）的那一部分区域组成。
- RGN_OR　组合两个区域的整个部分（两个区域的并）。
- RGN_XOR　组合两个区域，但去掉相互重叠的区域。

说明

此成员函数通过组合两个已有的区域来创建一个新的 GDI 区域。所组合成的区域就像 *nCombineMode* 所指定的一样。

两个指定的区域被组合，所得到的结果区域的句柄被保存在 CRgn 对象中。因此，不管 CRgn 中原来保存的区域是什么，都将被合并后的区域所代替。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的

那个值的范围内。

使用 CopyRgn 只是简单地将一个区域拷贝到另一个区域。

请参阅 CRgn::CopyRgn, ::CombineRgn

CRgn::CopyRgn

int CopyRgn(CRgn* *pRgnSrc*);

返回值

返回最后结果区域的类型。它可以是下列值之一：

- COMPLEXREGION 新区域具有相互重叠的边界。
- ERROR 未创建新区域。
- NULLREGION 新区域为空。

- SIMPLEREGION　新区域没有相互重叠的边界。

参数

pRgnSrc

标识一个已有的区域。

说明

此成员函数用来将一个由 *pRgnSrc* 定义的区域拷贝到 CRgn 对象中。新的区域将代替 *CRgn* 对象中原来保存的区域。此函数是 CombineRgn 成员函数的一个特例。

请参阅　CRgn::CombineRgn, ::CombineRgn

CRgn::CreateEllipticRgn

BOOL CreateEllipticRgn(int *x1*, int *y1*, int *x2*, int *y2*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

x1

指定椭圆的边界矩形的左上角的逻辑 x **坐标。**

y1

指定椭圆的边界矩形的左上角的逻辑 y **坐标。**

x2

指定椭圆的边界矩形的右下角的逻辑 x **坐标。**

y2

指定椭圆的边界矩形的右下角的逻辑 y 坐标。

说明

此成员函数用来创建一个椭圆形区域。该区域由用 *x1*，*y1*，*x2*，和 *y2* 指定的边界矩形来定义。该区域保存在 CRgn 对象中。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的那个值的范围内。

当使用完了由 CreateEllipticRgn 函数创建的区域时，应用程序应该将此区域选择出设备环境外，并用 DeleteObject 函数来删除它。

请参阅 CRgn::CreateEllipticRgnIndirect, ::CreateEllipticRgn

CRgn::CreateEllipticRgnIndirect

BOOL CreateEllipticRgnIndirect(LPCRECT *lpRect*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

lpRect

指向一个 RECT **结构或一个** CRect **对象，该结构或对象包含了椭圆的边界矩形的左上角和右下角的逻辑坐标。**

说明

此成员函数用来创建一个椭圆形区域。该区域由用 *lpRect* **指向的结构或对象定**

义，它保存在 CRgn 对象中。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的那个值的范围内。

当使用完了由 CreateEllipticRgnIndirect 函数创建的区域时，应用程序应该将此区域选择出设备环境外，并用 DeleteObject 函数来删除它。

请参阅 CRgn::CreateEllipticRgn, ::CreateEllipticRgnIndirect

CRgn::CreateFromData

BOOL CreateFromData(const XFORM* *lpXForm*, int *nCount*, const RGNDATA* *pRgnData*);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

lpXForm

指向一个 XFORM 数据结构，该数据结构定义了要在区域中实现的变换。

如果这个指针是 NULL，则将使用相同的变换。

nCount

指定由 pRgnData 指向的结构的字节数。

pRgnData

指向一个 RGNDATA 数据结构，该数据结构包含了区域数据。

说明

此成员函数从给定的区域和变换数据创建一个区域。应用程序可以通过调用函数 CRgn::GetRegionData 来获取一个区域的数据。

请参阅 CRgn::GetRegionData, ::ExtCreateRegion

CRgn::CreateFromPath

BOOL CreateFromPath(CDC* *pDC*);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

pDC

标识包含有一条闭合路径的设备环境。

说明

此成员函数从被选入给定设备环境的路径创建一个区域。由 *pDC* **标识的设备环境中必须包含有一条闭合的路径。在** CreateFromPath **把一条路径转换为一个区**

域后，Windows从该设备环境中丢弃此闭合路径。

请参阅　CDC::BeginPath, CDC::EndPath, CDC::SetPolyFillMode

CRgn::CreatePolygonRgn

BOOL CreatePolygonRgn(LPPOINT *lpPoints*, int *nCount*, int *nMode*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

lpPoints

指向一个 POINT 结构数组，或指向一个 CPoint 对象数组。每一个结构指定多边形的一个顶点的 x 坐标和 y 坐标。POINT 结构具有下面这样的

形式：

```
typedef struct tagPOINT{
int x;
int y;
} POINT;
```

nCount

指定由 *lpPoints* 指向的数组中的 POINT 结构或 CPoint 对象的数目。

nMode

指定区域的填充模式。这个参数可以是 ALTERNATE 或 WINDING。

说明

此成员函数创建一个多边形区域。如果必要，系统通过从多边形的最后一个顶点到第一个顶点绘制一条直线来关闭此多边形。创建出的多边形区域被保存在 CRgn 对象中。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的那个值的范围内。

当多边形的填充模式是 ALTERNATE 时，系统在每条扫描线上填充第奇数条边与第偶数条边之间的区域。也就是说，系统将填充第 1 条边和第 2 条边之间的区域；第 3 条边和第 4 条边之间的区域，等等。

当多边形的填充模式时 WINDING 时，系统使用绘图的方向来决定是否填充一个区域。多边形中的线段不是以顺时针方向就是以逆时针方向绘制的。一旦从一个封闭区域内向一个图形之外画的假想线穿过了一条顺时针线段时，便将某一计数值加 1；若这条线穿过了一条逆时针线段，则将计数值减 1。当假想线到达图形之外时，若计数值为非零，则区域被填充。

当使用完了由 CreatePolygonRgn 函数创建的区域时，应用程序应该将此区域选择出设备环境外，并用 DeleteObject 函数来删除它。

请参阅 CRgn::CreatePolyPolygonRgn, ::CreatePolygonRgn

CRgn::CreatePolyPolygonRgn

BOOL CreatePolyPolygonRgn(LPPOINT *lpPoints*, LPINT *lpPolyCounts*, int *nCount*,
int *nPolyFillMode*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

lpPoints

指向一个 POINT **结构数组，或指向一个** CPoint **对象数组，该结构或对象定义了多边形的顶点。每一个多边形都必须被显式地关闭，因为系统不会自动地关闭它们。多边形被连续指定。**POINT **结构具有下面这样的**

形式：

```
typedef struct tagPOINT{
int x;
int y;
} POINT;
```

lpPolyCounts

指向一个整数数组。第一个整数指定 lpPoints 数组中的第一个多边形的顶点数，第二个整数指定 *lpPoints* 数组中的第二个多边形的顶点数，如此等等。

nCount

指定 *lpPolyCounts* 数组中的整数的总数。

nPolyFillMode

指定多边形填充模式。这个值可以是 ALTERNATE 或 WINDING。

说明

此成员函数用来创建一个包含一系列封闭多边形的区域。获得的结果矩形被保存在 CRgn 对象中。

这些多边形可能是互相分开的，或可能时相互重叠的。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的那个值的范围内。

当多边形的填充模式是 ALTERNATE 时，系统在每条扫描线上填充第奇数条边与第偶数条边之间的区域。也就是说，系统将填充第 1 条边和第 2 条边之间的区域；第 3 条边和第 4 条边之间的区域，等等。

当多边形的填充模式是 WINDING 时，系统使用绘图的方向来决定是否填充一个区域。多边形中的线段不是以顺时针方向就是以逆时针方向绘制的。一旦从

一个封闭区域内向一个图形之外画的假想线穿过了一条顺时针线段时，便将某一计数值加 1；若这条线穿过了一条逆时针线段，则将计数值减 1。当假想线到达图形之外时，若计数值为非零，则区域被填充。

当使用完了由 CreatePolyPolygonRgn 函数创建的区域时，应用程序应该将此区域选择为设备环境外，并用 CGDIObject::DeleteObject 函数来删除它。

请参阅 CRgn::CreatePolygonRgn, CDC::SetPolyFillMode, ::CreatePolyPolygonRgn

CRgn::CreateRectRgn

BOOL CreateRectRgn(int *x1*, int *y1*, int *x2*, int *y2*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

x1

指定区域的左上角的逻辑 x 坐标。

y1

指定区域左上角的逻辑 y 坐标。

x2

指定区域右下角的逻辑 x 坐标。

y2

指定区域的右下角的逻辑 y 坐标。

说明

此成员函数用来创建一个矩形区域，该区域被保存在 CRgn 对象中。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的

那个值的范围内。

当使用完了由 CreateRectRgn 函数创建的区域时，应用程序应该将此区域选择出设备环境外，并用 CGDIObject::DeleteObject 函数来删除它。

请参阅 CRgn::CreateRectRgnIndirect, CRgn::CreateRoundRectRgn, ::CreateRectRgn

CRgn::CreateRectRgnIndirect

BOOL CreateRectRgnIndirect(LPCRECT *lpRect*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

lpRect

指向一个 RECT **结构或** CRect **对象，该结构或对象包含了区域的左上角和右下角的逻辑坐标。**RECT **结构具有下面的格式：**

```
typedef struct tagRECT {
    int left;
    int top;
    int right;
    int bottom;
} RECT;
```

说明

此成员函数用来创建一个矩形区域，该区域被保存在 CRgn **对象中。**

区域的大小被限制在 32767×32767 **个逻辑单位和** 64K **内存这两个值中较小的那个值的范围内。**

当使用完了由 CreateRectRgnIndirect 函数创建的区域时，应用程序应该将此区域选择出设备环境外，并用 CGDIObject::DeleteObject 成员函数来删除它。

请参阅 CRgn::CreateRectRgn, CRgn::CreateRoundRectRgn, ::CreateRectRgnIndirect

CRgn::CreateRoundRectRgn

BOOL CreateRoundRectRgn(int *x1*, int *y1*, int *x2*, int *y2*, int *x3*, int *y3*);

返回值

如果操作成功则返回非零值；否则返回 0。

参数

x1

指定区域的左上角的逻辑 x 坐标。

y1

指定区域左上角的逻辑 y 坐标。

x2

指定区域右下角的逻辑 x 坐标。

y2

指定区域的右下角的逻辑 y 坐标。

x3

指定用来创建圆角的椭圆的宽度。

y3

指定用来创建圆角的椭圆的高度。。

说明

此成员函数用来创建一个具有圆角的矩形区域。

区域的大小被限制在 32767×32767 个逻辑单位和 64K 内存这两个值中较小的

那个值的范围内。

当使用完了由 CreateRoundRectRgn 函数创建的区域时，应用程序应该将此区域选择出设备环境外，并用 CGDIObject::DeleteObject 函数来删除它。

请参阅 CRgn::CreateRectRgn, CRgn::CreateRectRgnIndirect, ::CreateRoundRectRgn

CRgn::CRgn

CRgn();

说明

此成员函数用来构造一个 CRgn 对象。在未使用 CRgn 中的某一个或几个成员函数来初始化该 CRgn 对象之前，m_hObject 数据成员中不含由一个有效的 Windows GDI 区域。

CRgn::EqualRgn

BOOL EqualRgn(CRgn* *pRgn*) const;

返回值

如果两个区域相等则返回非零值；否则返回 0。

参数

pRgn

标识一个区域。

说明

此成员函数用来确定给定的矩形与 CRgn **对象中保存的矩形是否相等。**

请参阅 ::EqualRgn

CRgn::FormHandle

static CRgn* PASCAL FromHandle(HRGN *hRgn*);

返回值

如果函数成功则返回一个指向 CRgn **对象的指针。如果函数不成功，则返回值为** NULL**。**

参数

hRgn

指定一个 Windows **区域的句柄。**

说明

当给出了一个 Windows **区域句柄时，此成员函数返回一个指向** CRgn **对象的指**

针。如果还没有一个 CRgn 对象连接到这个句柄上，则创建一个临时 CRgn 对象并将它连接到句柄上。这一临时的 CRgn 对象将一直有效，直到应用程序在其事件循环中再次有空闲时间为止。此时，所有的临时图形对象都将被删除。也就是说，该临时对象仅在处理一条窗口消息的过程中有效。

CRgn::GetRegionData

Int GetRegionData(LPRGNDATA *lpRgnData*, int *nCount*) const;

返回值

指定结果区域的类型。它可以是下列值之一：

- COMPLEXREGION　新区域具有相互重叠的边界。
- ERROR　未创建新区域。
- NULLREGION　新区域为空。

- SIMPLEREGION　新区域没有相互重叠的边界。

参数

lpRgnData

指向一个 RGNDATA 数据结构，该结构用来接收信息。如果这个参数是 NULL，则返回值包含了此区域数据的字节数。

nCount

指定 lpRgnData 缓冲的用字节表示的大小。

说明

此成员函数用描述区域的数据来填充指定的缓存。这些数据包括组成该区域的矩形的尺寸。此函数常与 CRgn::CreateFromData 函数连用。

请参阅　CRgn::CreateFromData

CRgn::GetRgnBox

int GetRgnBox(LPRECT *lpRect*) const;

返回值

指定结果区域的类型。它可以是下列值之一：

- COMPLEXREGION **新区域具有相互重叠的边界。**
- ERROR **未创建新区域。**
- NULLREGION **新区域为空。**
- SIMPLEREGION **新区域没有相互重叠的边界。**

参数

lpRect

指向一个 RECT **结构或** CRect **对象，该结构或对象包含了区域的左上角**

和右下角的逻辑坐标。RECT 结构具有下面的格式：

```
typedef struct tagRECT {
    int left;
    int top;
    int right;
    int bottom;
} RECT;
```

说明

此成员函数用来获取 CRgn 对象的边界矩形的坐标。

请参阅 ::GetRgnBox

CRgn::OffsetRgn

int OffsetRgn(int *x*, int *y*);
int OffsetRgn(POINT *point*);

返回值

指定结果区域的类型。它可以是下列值之一：

- COMPLEXREGION　新区域具有相互重叠的边界。
- ERROR　未创建新区域。
- NULLREGION　新区域为空。
- SIMPLEREGION 新区域没有相互重叠的边界。

参数

x

指定向左或向右移动的单位数。

y

指定向上或向下移动的单位数。

point

此参数的 x 坐标指定向左或向右移动的单位数。其 y 坐标指定向上或向下移动的单位数。此参数可以是一个 POINT 结构或一个 CPoint 对象。

说明

此成员函数用来将保存在 CRgn 对象中的区域移动指定的偏移。此函数将区域沿 x 轴移动 x 个单位，沿 y 轴移动 y 个单位。

区域的坐标值必须小于或等于 32,767，但大于或等于-32,768。要谨慎地选择 x 和 y 参数，以避免无效的区域坐标。

请参阅 ::OffsetRgn

CRgn::PtInRegion

BOOL PtInRegion(int *x*, int *y*) const;

BOOL PtInRegion(POINT *point*) const;

返回值

如果点在区域内则返回非零值；否则返回 0。

参数

x

指定要测试的点的 x **坐标。**

y

指定要测试的点的 y **坐标。**

point

此参数的 x **坐标和** y **坐标指定了要测试的点的** x **坐标和** y **坐标。此** *point* **参数或者是一个** point **结构或是一个** CPoint **对象。**

说明

此成员函数用来检查由 x **和** y **给定的点是否在保存在** CRgn **对象中的区域的内部。**

请参阅 ::PtInRegion

CRgn::RectInRegion

BOOL RectInRegion(LPCRECT *lpRect*) const;

返回值

如果指定矩形的任何一部分在区域的边界内，则返回非零值；否则返回 0。

参数

lpRect

指向一个 RECT 结构或 CRect 对象，该结构或对象包含了区域的左上角和右下角的逻辑坐标。RECT 结构具有下面的格式：

```
typedef struct tagRECT {
    int left;
    int top;
    int right;
    int bottom;
} RECT;
```

说明

此成员函数用来确定由 *lpRect* 指定的矩形的任何一部分是否在保存在 CRgn 对象中的区域的边界之内。

请参阅 ::RectInRegion

CRgn:: SetRectRgn

void SetRectRgn(int *x1*, int *y1*, int *x2*, int *y2*);
void SetRectRgn(LPCRECT *lpRect*);

参数

x1

指定矩形区域的左上角的逻辑 x **坐标。**

y1

指定矩形区域左上角的逻辑 y **坐标。**

x2

指定矩形区域右下角的逻辑 x **坐标。**

y2

指定矩形区域的右下角的逻辑 y **坐标。**

lpRect

指定一个矩形区域。该参数可以是一个指向 RECT **结构或一个** CRect **对象**

的指针。

说明

此成员函数用来创建一个矩形区域。但是它与 CreateRectRgn 不一样，此函数不从局部 Windows 应用程序堆中分配任何附加的内存。此函数使用为存放在 CRgn 对象中的区域分配的空间。这就意味着在调用 SetRectRgn 函数之前必须使用一个有效的 Windows区域将该 CRgn 对象初始化。由参数 *x1*，*x2*，*y1*，和 *y2* 指定的两个点指定已分配的空间的最小尺寸。

用这个函数代替 CreateRectRgn 成员函数可以避免调用局部内存管理程序。

请参阅 CRgn::CreateRectRgn, ::SetRectRgn

操作符

CRgn::operator HRGN

operator HRGN( ) const;

返回值

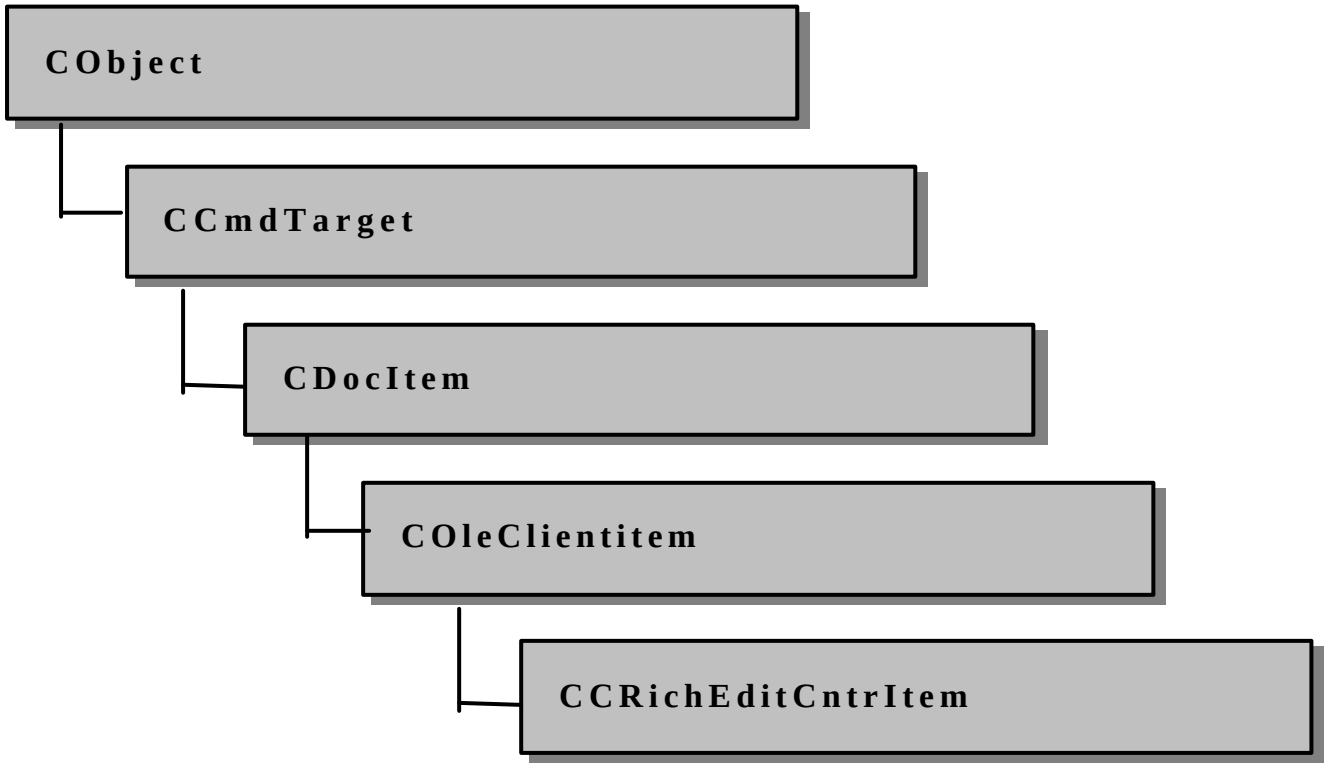
如果成功则返回一个 WindowsGDI 对象句柄，该对象由 CRgn 对象代表；否则返回 NULL。

说明

此操作符用来获取 CRgn 对象的与 Windows GDI 连接的句柄。此操作符是一个强制转换操作符，它支持对一个 HRGN 对象的直接使用。

要获取更多有关使用图形对象的信息，请参见“Win32 SDK程序员参考”中的文章“图形化对象”。

CRichEditCntrItem



一个“带格式编辑控件”是一个窗口，在这个窗口中用户可以输入和编辑文本。文本可以是字母或段落格式，也可以包括嵌入的 OLE 对象。带格式编辑控件为格式化文本提供了设计界面。但是，应用程序必须实现需要的用户部件，以使格式化操作对用户来说可用。

在 MFC 的文档视结构的环境中，CRichEditCntrItem 类与 CRichEditView 和 CRichEditDoc 一起提供带格式编辑控件的功能。CRichEditCntrView 保存文本和文本的格式化特征。CRichEditCntrDoc 保存视中的 OLE 客户项的列表。CRichEditCntrItem 提供容器方对 OLE 客户项的访问。

这个 Windows 通用控件（也就是 CRichEditCntr 及其相关类）只对于运行在 Window95 和 Windows NT 3.51 及更新版本下的程序是可用的。

有关在一个 MFC 应用程序中使用带格式编辑容器项的例子,请参见 WORDPAD 示例应用程序。

#include <afxrich.h>

请参阅 CRichEditDoc,CRichEditView

CRichEditCntrItem 类成员

Constructor	
CrichEditCntrItem	构造一个 CRichEditCntrItem 对象

Operations	
SyncToCRichEditObject	用另一个类型激活项

成员函数

CRichEditCntrItem:: CRichEditCntrItem

CRichEditCntrItem(REOBJECT* *preo* = NULL, CRichEditDoc* *pContainer* =

NULL);

参数

preo

指向一个 REOBJECT **结构的指针，该结构描述了一个** OLE **项。微软这个** OLE **项构造** CRichEditCntrItem **对象。如果** *preo* **时** NULL**，则客户项是空的。**

pContainer

指向一个容器文档的指针，该容器文档将包含这个项。如果 *pContainer* **是** NULL**，则你必须显式地调用** COleoDcument::AddItem **来将这个客户项增加到一个文档中。**

说明

此成员函数用来创建一个 CRichEditCntrItem **对象，并将它添加到容器文档中。**

此函数不实现任何 OLE **初始化。**

更多的信息，请参见 Win32 **文档中的** REOBJECT **结构。**

请参阅 COleDocument::AddItem, CRichEditDoc

CRichEditCntrItem::SyncToCRichEditObject

void SyncToRichEditObject(REOBJECT& *reo*);

参数

reo

指向一个 REOBJECT **结构的引用，该结构描述了一个** OLE **项。**

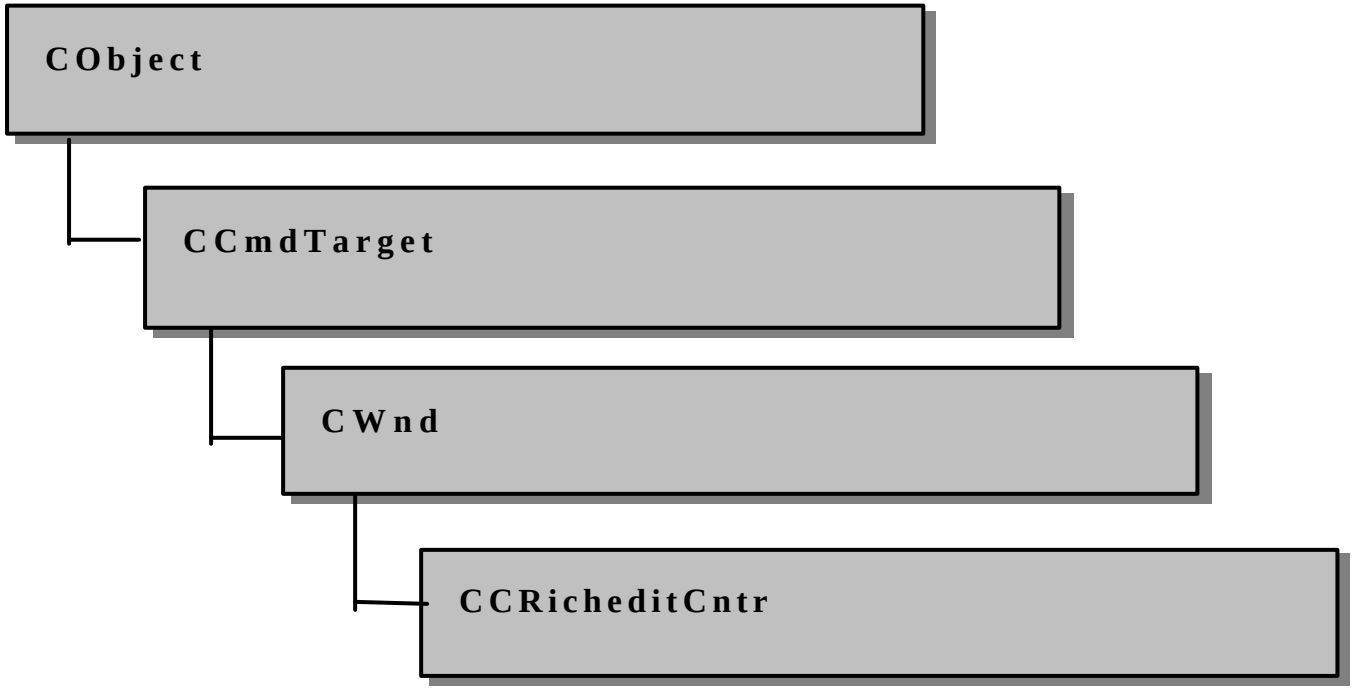
说明

此成员函数用来使这个 CRichEditCntrItem **的设备外观，即** DVASPECT **与** *reo*

所指定的结构同步。

更多的信息请参见 OLE 文档中的 DVASPECT。

CRichEditCtrl



一个“带格式编辑控件”是一个窗口，在这个窗口中用户可以输入和编辑文本。文本可以是字母或段落格式，也可以包括嵌入的 OLE 对象。带格式编辑控件为格式化文本提供了设计界面。但是，应用程序必须实现需要的用户部件，以使格式化操作对用户来说可用。

CRichEditCntr 类提供了带格式编辑控件的功能。这个 Windows 通用控件（也就是 CRichEditCtrl 类）只对于运行在 Window95 和 Windows NT 3.51 及更新版本下的程序是可用的。

> **重点** 如果你正在使用一个对话框中的带格式编辑控件（不管你的应用程序是 SDI, MDI, 或是基于对话框的），你必须在显示该对话框之前调用 AfxInitRichEdit 一次。通常是在你的应用程序的 IniInstance 成员函数中调用这个函数。你不用在每一次显示这个对话框的时候都调用此 AfxInitRichEdit 函数，只要在第一次时调用它就可以了。如果你是在

使用 CRichEditView，则不用调用 AfxInitRichEdit。

有关在一个 MFC 应用程序中使用带格式编辑容器项的例子，请参见 WORDPAD 示例应用程序。

```
#include <afxcmn.h>
```

请参阅 CEdit, CRichEditView

CRichEditCtrl 类成员

Construction	
CrichEditCtrl	构造一个 CRichEditCtrl 对象
Create	创建 Windows 带格式编辑控件并将它与这个 CRichEditCtrl 对象相联系

LineOperations	
GetLineCount	获取这个 CRichEditCtrl 对象中的行数目
GetLine	从这个 CRichEditCtrl 对象中获取一行文本
GetFirstVisibleLine	确定这个 CRichEditCtrl 对象的最上面的可见行
LineIndex	获取此 CRichEditCtrl 对象中一个给定行的字符索引
LineFromChar	确定是哪一行包含了给定字符
LineLength	获取此 CRichEditCtrl 对象中的给定行的长度
LineScroll	在此 CRichEditCtrl 对象中滚动文本

Selection Operations	
GetSel	获取此 CRichEditCtrl 对象中的当前选择的开始和结束位置
GetSelText	获取此 CRichEditCtrl 对象中的当前选择的文本
GetSelectionType	获取此 CRichEditCtrl 对象中的当前选择中内容的类型
Clear	清除当前选择
SetSel	设置此 CRichEditCtrl 对象中的选择
ReplaceSel	用指定的文本替换此 CRichEditCtrl 对象中的当前选择
HideSelection	显示或隐藏当前的选择

Formatting Operations		
GetDefaultCharFormat	获取此属性	CRichEditCtrl 对象中当前缺省的字符格式
GetSelectionCharFormat	获取此属性	CRichEditCtrl 对象中当前选择的字符格式
GetParaFormat	获取此式属性	CRichEditCtrl 对象中的当前选择的段落格
SetDefaultCharFormat	设置此的属性	CRichEditCtrl 对象中的当前缺省字符格式
SetSelectionCharFormat	设置此属性	CRichEditCtrl 对象中当前选择的字符格式
SetWordCharFormat	设置此式属性	CRichEditCtrl 对象中的当前单词的字符格
SetParaFormat	设置此式属性	CRichEditCtrl 对象中的当前选择的段落格

EditingOperations	
Undo	取消最后一次编辑操作
CanUndo	确定是否可以取消一次编辑操作
EmptyUndoBuffer	重置（清除）此 CRichEditCtrl 对象的取消标志
StreamIn	将来自一个输入流的文本插入此 CRichEditCtrl 对象中
StreamOut	将来自此 CRichEditCtrl 对象的文本保存到输出流中
General Operations	
GetModify	确定在最后一次保存后此 CRichEditCtrl 对象的内容是否已经被改变了
SetModify	为这个 CRichEditCtrl 对象设置或清除修改标志
FindText	在这个 CRichEditCtrl 对象中定位文本
GetRect	为此 CRichEditCtrl 对象获取格式化矩形
SetRect	为此 CRichEditCtrl 对象设置格式化矩形
GetCharPos	确定此 CRichEditCtrl 对象中的一个给定字符的位置
SetOptions	为这个 CRichEditCtrl 对象设置选项
SetReadOnly	为这个 CRichEditCtrl 对象设置只读选项
GetTextLength	获取此 CRichEditCtrl 对象中的文本的长度

续表

GetLimitText	获取一个用户可以输入这个 CrichEditCtrl 对象的文本数量的限制
LimitText	限制一个用户可以输入此 CRichEditCtrl 对象的文本数量
GetEventMask	获取此 CRichEditCtrl 对象的事件掩码
SetEventMask	设置此 CRichEditCtrl 对象的事件掩码
RequestResize	强迫此 CRichEditCtrl 对象发送请求改变大小的通知
SetBackgroundColor	设置此 CRichEditCtrl 对象中的背景颜色
SetTargetDevice	设置此 CRichEditCtrl 对象的目标输出设备
FormatRange	为目标输出设备格式化一个文本范围
DisplayBand	显示此 CRichEditCtrl 对象的一部分内容
Clipboard Operations	
Copy	将当前选项拷贝到剪贴板上
Cut	将存取选择剪下到剪贴板上
Paste	剪贴板上的内容插入到此带格式编辑控件中
PasteSpecial	将剪贴板上的内容按指定的数据格式插入到此带格式编辑控件中

续表

CanPaste	确定剪贴板上的内容是否可以粘贴到此带格式编辑控件中
OLE Operations	
GetIRichEditOle	为此带格式编辑控件获取一个直线 IrichEditOle 接口的指针
SetOLECallback	为此带格式编辑控件设置 IrichEditOleCallback COM 对象

成员函数

CRichEditCtrl::CanPaste

BOOL CanPaste(UINT *nFormat = 0*) const;

返回值

如果剪贴板的格式可以被粘贴，则返回非零值；否则返回 0。

参数

nFormat

要查询的剪贴板数据格式。此参数是某个预定义的剪贴板格式，或是由 RegisterClip- boardFormat **返回的值。**

说明

此成员函数用来确定一个带格式编辑控件是否可以粘贴指定的剪贴板格式。如果 *nFormat* **是** 0**，** CanPaste **将尝试当前剪贴板中的任何格式。**

请参阅 CRichEditCtrl::Paste, CRichEditCtrl::PasteSpecial

CRichEditCtrl::CanUndo

BOOL CanUndo( ) const;

返回值

如果最后一次编辑操作可以被通过调用 Undo 成员函数来取消，则返回非零值；否则，如果不能取消，则返回 0。

说明

此成员函数用来确定最后一次编辑操作是否可以被取消。

更多信息，请参见 Win32 文档中的 EM_CAUNDO。

请参阅 CRichEditCtrl::Undo, CRichEditCtrl::EmptyUndoBuffer

CRichEditCtrl::Clear

void Clear( );

说明

此成员函数用来删除（清除）带格式编辑控件中的当前选择（如果有的话）。

由 Clear 执行的删除，可以通过调用 Undo 成员函数来取消。

要删除当前的选择，并将其作为剪贴板的内容，可以调用 Cut 成员函数。

更多的信息，请参见 Win32 文档中的 WM_CLEAR。

请参阅 CRichEditCtrl::Undo, CRichEditCtrl::Cut, CRichEditCtrl::Copy, CRichEditCtrl::Paste

CRichEditCtrl::Copy

void Copy( );

说明

此成员函数用来将带格式编辑控件中的当前选择（如果有的话）拷贝到剪贴板中。

更多的信息，请参见 Win32 **文档中的** WM_COPY**。**

请参阅 CRichEditCtrl::Paste, CRichEditCtrl::Cut

CRichEditCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果初始化成功，则返回非零值；否则返回 0**。**

参数

dwStyle

指定编辑控件的风格。可以将编辑风格的任何组合应用到此控件。

rect

指定编辑控件的大小和位置。它可以是一个 CRect **对象或** RECT **结构。**

pParentWnd

指定编辑控件的父窗口（通常是一个 CDialog**）。它不能是** NULL**。**

nID

指定编辑控件的 ID**。**

说明

分两步来构造一个 CRichEditCtrl **对象。首先，调用** CRichEditCtrl **构造函数，然后调用** Create**，该函数创建** Windows **编辑控件并将其连接到** CRichEditCtrl

对象上。

当你用这个函数来创建一个带格式编辑控件时，首先你必须加载必须的通用控件库。为了加载此库，可以调用全局函数 AfxInitRichEdit，它将按顺序初始化通用控件库。在你的进程中你只需要调用 AfxInitRichEdit 一次。

当执行 Create 时，Windows 发送 WM_NCCREATE，WM_NCCALCSIZE，WM_CREATE，和 WM_GETMINMAXINFO 消息给这个编辑控件。

缺省的，这些消息由 CWnd 基类中的 OnNcCreate，OnNcCalcSize，OnCreate，和 OnGetMinMaxInfo 成员函数来处理。为了扩展这些缺省的消息处理，可以从 CRichEditCtrl 派生一个类，向此新类中添加一个消息映射，并重载上述消息处理成员函数。例如，重载 OnCreate 来为新类实现需要的初始化。

可以将下列的窗口风格应用到一个编辑控件上。

- WS_CHILD　**总是使用。**
- WS_VISIBLE　**经常使用。**
- WS_DISABLED　**很少使用。**
- WS_GROUP　**可用于成组控件。**
- WS_TABSTOP　**将编辑控件包括在** TABLE **键的顺序中。**

请参阅　CRichEditCtrl::CRichEditCtrll

CRichEditCtrl::CRichEditCtrl

CRichEditCtrl( );

说明

此成员函数用来构造一个 CRichEditCtrl **对象。使用** Create **来构造** Windows **带格式编辑控件。**

请参阅 CRichEditCtrl::Create

CRichEditCtrl::Cut

void Cut();

说明

此成员函数用来删除（剪切）带格式编辑控件中的当前选择（如果有），并将其拷贝到剪贴板中。

用 Cut **执行的删除可以通过调用** Undo **成员函数来取消。**

要删除当前的选择而又不将被删除的内容放到剪贴板中，可以调用 Clear **成员函数。**

更多的信息，请参见 Win32 **文档中的** WM_CUT**。**

请参阅 CRichEditCtrl::Copy, CRichEditCtrl::Undo, CRichEditCtrl::Clear

CRichEditCtrl::DisplayBand

BOOL DisplayBand(LPRECT *pDisplayRect*);

返回值

如果显示格式化文本成功，则返回非零值；否则返回 0。

参数

pDisplayRect

指向一个 RECT **或** CRect **对象的指针，用来指定显示文本的设备的区域。**

说明

此成员函数用来显示带格式编辑控件的内容（文本和 OLE **项）的一部分，用**

前面的 FormatRange 指定的格式。文本和 OLE 项被粘贴到由 *pDisplayRect* 指针指定的区域。

更多的信息，请参见 Win32 文档中的 WM_DISPLAYBAND。

请参阅　CRichEditCtrl::FormatRange

CRichEditCtrl::EmptyUndoBuffer

void EmptyUndoBuffer();

说明

此成员函数用来重置此带格式编辑控件的 undo 标志。此时控件将不能取消最后一次编辑操作。无论什么时候当带格式编辑控件中的一项操作是可以被取消的时，取消标志就被设置。

当你调用 CWnd 的成员函数 SetWindowText 时，取消标志被自动清除。

更多的信息，请参见 Win32 文档中的 EM_EMPTYUNDOBUFFER。

请参阅　CRichEditCtrl::CanUndo,CRichEditCtrl::Undo, CWnd::SetWindowText

CRichEditCtrl::FindText

long FindText(DWORD *dwFlags*, FINDTEXTEX* *pFindText*) const;

返回值

返回下一次匹配的从零开始的字符位置；如果没有匹配则返回 -1。

参数

dwFlags

匹配标准的标志。可以是下列值的零个或多个：

- FR_MATCHCASE　指明查找是大小写敏感的。
- FR_WHOLEWORD　指明在查找中要考虑单词的边界。

pFindText

指向一个 FINDTEXTEX 结构的指针，该结构给出了用于查找的参数并返回找到匹配的范围。

说明

此成员函数用来在带格式编辑控件中查找文本。你可以通过在 FINDTEXTEX 结构中设置 CHARRANGE 中的适当范围参数来进行向上或向下搜索。

更多的信息，请参见 Win32 文档中的 EM_FINDTEXTEX 消息和 FINDTEXTEX 结构。

请参阅　CRichEditCtrl::SetSel

CRichEditCtrl::FormatRange

long FormatRange(FORMATRANGE* *pfr*, BOOL *bDisplay* = TRUE);

返回值

返回值为适应该区域的最后一个字符的索引再加 1**。**

参数

pfr

指向 FORMATRANGE **结构的指针，该结构包含了有关输出设备的信息。如果是** NULL**，则表明此带格式编辑控件中的高速缓存中的信息可以释放了。**

bDisplay

指示是否应该提交文本。如果是 FALSE**，则表明文本正好是适量的。**

说明

此成员函数用来为一个指定的设备格式化一个带格式编辑控件中的一定范围的文本。通常，在调用这个函数之后会调用 DisplayBand。

更多的信息，请参见 Win32 **文档中的** EM_FORMATRANGE **消息和** FORMATRANGE **结构。**

请参阅 CRichEditCtrl::DisplayBand

CRichEditCtrl::GetCharPos

CPoint GetCharPos(long *lChar*) const;

返回值

返回由 *lChar* **指定的字符的左上角的位置。**

参数

lChar

字符的从零开始的索引。

说明

此成员函数用来获取在一个 CRichEditCtrl **对象内的给定字符的位置（左上角）。该字符由给出的它的从零开始的索引指定。如果** *lChar* **大于此** CRichEditCtrl **对象中的最后一个字符的索引，则返回值指示的坐标是此** CRichEditCtrl **对象中刚好超过其最后一个字符的位置的坐标。**

更多的信息，参见 Win32 **文档中的** EM_POSFROMCHAR.**。**

请参阅 CRichEditCtrl::FindText

CRichEditCtrl::GetDefaultCharFormat

DWORD GetDefaultCharFormat(CHARFORMAT& *cf*) const;

返回值

返回 *cf* **的** dwMask **数据成员。它指定了缺省的字符格式化属性。**

参数

cf

指向一个 CHARFORMAT **结构的指针，该结构将包含缺省的字符格式化属性。**

说明

此成员函数用来获取此 CRichEditCtrl **对象的缺省的字符格式化属性。**

更多的信息，参见 Win32 文档中的 EM_GETCHARFORMAT 消息和 CHARFORMAT 结构。

请参阅 CRichEditCtrl::SetDefaultCharFormat, CRichEditCtrl::GetSelectionCharFormat, CRichEditCtrl::GetParaFormat

CRichEditCtrl::GetEventMask

```
long GetEventMask() const;
```

返回值

返回此 CRichEditCtrl 对象的事件掩码。

说明

此成员函数用来获取此 CRichEditCtrl 对象的事件掩码。这个事件掩码指定了此

CRichEditCtrl 对象要将哪一个消息发送给它的父窗口。

更多的信息，参见 Win32 文档中的 EM_GETEVENTMASK。

请参阅 CRichEditCtrl::SetEventMask, CRichEditCtrl::CRichEditCtrl

CRichEditCtrl::GetFirstVisibleLine

```
int GetFirstVisibleLine() const;
```

返回值

返回此 CRichEditCtrl 对象中的最顶上的可见行的从零开始的索引。

说明

此成员函数用来确定此 CRichEditCtrl 对象中的最顶上的可见行。

更多的信息，参见 Win32 文档中的 EM_GETFIRSTVISIBLELINE。

请参阅 CRichEditCtrl::GetLine, CRichEditCtrl::GetLineCount

CRichEditCtrl::GetIRichEditOle

IRichEditOle* GetIRichEditOle( ) const;

返回值

返回指向 IRichEditOle **接口的指针，该接口用来访问此** CRichEditCtrl **对象的** OLE **机能；如果接口不能被访问，则返回的是** NULL。

说明

此成员函数用来访问此 CRichEditCtrl **对象的** IRichEditOle **接口。使用此接口可以访问这个** CRichEditCtrl **对象的** OLE **机能。**

更多的信息，参见 Win32 **文档中的** EM_GETOLEINTERFACE **消息和**

IRichEditOle 接口。

请参阅 CRichEditCtrl::SetOLECallback

CRichEditCtrl::GetLimitText

long GetLimitText( ) const;

返回值

返回此 CRichEditCtrl **对象的以字节表示的文本限制。**

说明

此成员函数用来获取这个 RichEditCtrl **对象的文本限制。文本限制是指带格式编辑控件可以接收的以字节表示的最大的文本数量。**

更多的信息，参见 Win32 **文档中的** EM_GETLIMITTEXT。

请参阅 CRichEditCtrl::LimitText

CRichEditCtrl::GetLine

int GetLine(int *nIndex*, LPTSTR *lpszBuffer*) const;
int GetLine(int *nIndex*, LPTSTR *lpszBuffer*, int *nMaxLength*) const;

返回值

返回要被拷贝到 *lpszBuffer* **中去的字符数。**

参数

nIndex

要获取的行的从零开始的索引。

lpszBuffer

指向用来接收文本的缓冲区。缓冲区中的第一个单词必须指定可以被拷

贝到缓冲区中去的最大的字节数目。

nMaxLength

可以被拷贝到 *lpszBuffer* 中去的最大的字符数。GetLine 的第二种格式将这个值放在由 *lpszBuffer* 指定的缓冲区的第一个单词中。

说明

此成员函数用来从此 CRichEditCtrl 对象中获取一行文本。此拷贝的行不包含用于结尾的空字符。

注意　因为此缓冲区的第一个单词保存了要被拷贝的字符数目，所以必须确保你的缓冲区至少有 4 字节长。

更多的信息，参见 Win32 文档中的 EM_GETLINE。

请参阅　CRichEditCtrl::LineLength

CRichEditCtrl::GetLineCount

int GetLineCount( ) const;

返回值

返回此 CRichEditCtrl **对象中的行数。**

说明

此成员函数用来获取 CRichEditCtrl **对象中的行数。**

更多信息，请参见 Win32 **文档中的** EM_GETLINELOUNT。

请参阅 CRichEditCtrl::GetLine

CRichEditCtrl::GetModify

BOOL GetModify( ) const;

返回值

如果此 CRichEditCtrl **对象中的文本被修改了，则返回非零值；否则返回** 0。

说明

此成员函数用来确定 CRichEditCtrl **对象中的内容是否被修改了。**

Windows **保持一个内部标志来表明带格式编辑控件中的内容是否被改变了。当编辑控件第一次被创建时，这个标志被清零，也可以通过调用** SetModify **成员函数来清零。**

更多的信息，参见 Win32 **文档中的** EM_GETMODIFY。

请参阅 CRichEditCtrl::SetModify

CRichEditCtrl::GetParaFormat

DWORD GetParaFormat(PARAFORMAT& *pf*) const;

返回值

返回 *pf* **的数据成员** dwMask。**它指定在整个当前选择中一致的段落格式化属性。**

参数

pf

指向一个 PARAFORMAT **结构的指针，该结构包含了当前选择的段落格式化属性。**

说明

此成员函数用来获取当前选择的段落格式化属性。如果选择的不止一段， *pf* **接**

收的是第一个被选择段的属性。返回值指明了在整个当前选择中一致的属性。

更多的信息，参见 Win32 文档中的 EM_GETPARAFORMAT 消息和 PARAFORMAT 结构。

请参阅 CRichEditCtrl::SetParaFormat, CRichEditCtrl::GetSelectionCharFormat

CRichEditCtrl::GetRect

void GetRect(LPRECT *lpRect*) const;

参数

lpRect

是 CRect 或指向一个 RECT 的指针，用来接收此 CRichEditCtrl 对象的格式化矩形。

说明

此成员函数用来获取此 CRichEditCtrl **对象的格式化矩形。格式化矩形是文本的边界矩形。这个值不依赖** RichEditCtrl **对象的大小。**

更多的信息，参见 Win32 **文档中的** EM_GETRECT。

请参阅 CRichEditCtrl::SetRect

CRichEditCtrl::GetSel

```
void GetSel( CHARRANGE& cr ) const;
void GetSel( long& nStartChar, long& nEndChar ) const;
```

参数

cr

对一个 CHARRANGE **结构的引用，该结构用来接收当前选择的界线。**

nStartChar

当前选择中的第一个字符的从零开始的索引。

nEndChar

当前选择中的最后一个字符的从零开始的索引。

说明

此成员函数用来获取在此 CRichEditCtrl 对象中的当前选择的界线。

这个函数的两种形式提供了用来获取选择的界线的两种可相互替代的方法。下面就是这些形式的简单描述：

- GetSel(*cr*) 这种形式使用 CHARRANGE 结构的 cpMin 和 cpMax 成员来返回界线。
- GetSel(*nStartChar, nEndChar*) 这种形式使用参数 *nStartChar* 和 *nEndChar* 来返回界线。

如果开始值（cpMin 或 *nStartChar*）是 0，而结尾值（cpMax 或 *nEndChar*）是-1，则选择包括了所有的文本。

更多的信息，参见 Win32 文档中的 EM_EXGETSEL 消息和 CHARRANGE 结构。

请参阅 CRichEditCtrl::SetSel, CRichEditCtrl::GetSelText, CRichEditCtrl::GetParaFormat, CRichEditCtrl::GetSelectionCharFormat

CRichEditCtrl::GetSelectionCharFormat

DWORD GetSelectionCharFormat(CHARFORMAT& *cf*) const;

返回值

返回 *cf* 的 dwMask 数据成员。它指定了在整个当前选择中都一致的字符格式化

属性。

参数

cf

这是一个 CHARFORMAT 结构的指针，该结构用来接收当前选择的字符格式化属性。

说明

此成员函数用来获取当前选择的字符格式化属性。参数 *cf* 接收当前选择中的第一个字符的属性。返回值指明了在整个当前选择中都一致的属性。

更多的信息，参见 Win32 文档中的 EM_GETCHARFORMAT 消息和 CHARFORMAT 结构。

请参阅 CRichEditCtrl::GetDefaultCharFormat, CRichEditCtrl::GetParaFormat,

CRichEditCtrl::SetSelectionCharFormat, CRichEditCtrl::GetSelText

CRichEditCtrl::GetSelectionType

WORD GetSelectionType( ) const;

返回值

返回表明当前选择内容的标志。它可以是下列标志的组合：

- SEL_EMPTY　**表明没有当前选择。**
- SEL_TEXT　**表明当前选择包含文本。**
- SEL_OBJECT　**表明当前选择至少包含了一个** OLE **项。**
- SEL_MULTICHAR　**表明当前选择包含了不止一个字符的文本。**
- SEL_MULTIOBJECT　**表明当前选择包含了不止一个** OLE **项。**

说明

此成员函数用来确定此 CRichEditCtrl **对象中的选择的类型。**

更多的信息，参见 Win32 **中的** EM_SELECTIONTYPE。

请参阅 CRichEditCtrl::GetSel, CRichEditCtrl::GetSelText

CRichEditCtrl::GetSelText

long GetSelText(LPTSTR *lpBuf*) const;
CString GetSelText( ) const;

返回值

返回值决定于函数的形式：

- GetSelText(*lpBuf*) **返回要拷贝到** *lpBuf* **中去的字符数，不包括结尾的空字**

符。

- GetSelText( ) 返回包含当前选择的字符串。

参数

lpBuf

指向用来接收当前选择中的文本的缓冲区的指针。

说明

此成员函数用来接收来自 CRichEditCtrl 对象中当前选择的文本。

如果你使用第一种形式 GetSelText(*lpBuf*)，则你必须保证缓冲区对于将要接收的文本来说足够大。调用 GetSel 来确定当前选择中的字符数。

更多的信息，参见 Win32 文档中 EM_GETSELTEXT。

请参阅 CRichEditCtrl::GetSel, CRichEditCtrl::GetSelectionType

CRichEditCtrl::GetTextLength

long GetTextLength();

返回值

返回此 CRichEditCtrl **对象中的文本的长度。**

说明

此成员函数用来获取此 CRichEditCtrl **对象中的文本的长度。**

更多的信息，参见 Win32 **文档中** EM_GETTEXTLENGTH。

请参阅 CRichEditCtrl::LimitText, CRichEditCtrl::GetLimitText

CRichEditCtrl::HideSelection

void HideSelection(BOOL *bHide*, BOOL *bPerm*);

参数

bHide

表明是要显示还是要隐藏选择，如果是 TRUE **就是要隐藏选择。**

bPerm

表明有关这个选择的可见性的改变是否是永久的。

说明

此成员函数用来改变此选择的可见性。

当 *pPerm* **是** TRUE **时，则函数改变此** CRichEditCtrl **对象的** ECO_NOHIDESEL **选项。有关这个选项的简短描述，参见** SetOptions**。可以使用此函数来设置**

CRichEditCtrl 对象的所有选项。

更多的信息，参见 Win32 文档的 EM_HIDESELECTION。

请参阅 CRichEditCtrl::SetSel, CRichEditCtrl::GetSelectionType

CRichEditCtrl::LimitText

void LimitText(long *nChars* = 0);

参数

nChars

指定用户可输入的文本的长度（以字节表示）。如果这个常数是 0，则文本的长度被设置为 UINT_MAX 字节。这是缺省的行为。

说明

此成员函数用来限制用户可以输入到一个编辑控件中的文本的长度。

改变这个文本限制只会约束用户可以输入的文本。它对编辑控件中已经存在的文本是没有影响的，它也不影响通过调用 CWnd 中的 SetWindowText 函数拷贝到编辑控件中去的文本的长度。如果一个应用程序要用 SetWindowText 函数来将比在 LimiText 中指定的文本更多的文本放入一个编辑控件中，则用户可以编辑控件中的任何文本。但是，这个文本限制将禁止用户用新的文本来代替控件中已经存在的文本，除非已经删除了当前的选择，使得控件中的文本低于了文本的限制值。

注意　对于文本极限来说，将每一个 0LE 项计算为一个字符。

更多的信息，参见 Win32 文档中的 EM_EXLIMITTEXT。

请参阅 CRichEditCtrl::GetLimitText

CRichEditCtrl::LineFromChar

long LineFromChar(long *nIndex*) const;

返回值

返回包含由 *nIndex* **指定的字符索引的行的从零开始的行号。如果** *nIndex* **是**-1**，则返回的是包含选择的第一个字符的行的行号。如果没有选择，则返回当前行的行号。**

参数

nIndex

包含了编辑控件的文本中的期望字符的从零开始的索引值，或者时-1**。如果** *nIndex* **是**-1**，则它指定的是当前行，也就是包含插字符（**^**）的行。**

说明

此成员函数用来获取包含指定字符索引的行的行号。字符索引就是带格式编辑控件中从开始计算的字符数目。对于字符计数，一个 OLE 项被计算为一个单字符。

更多的信息，参见 Win32 文档中的 EM_EXLINEFROMCHAR。

请参阅 CRichEditCtrl::GetLineCount, CRichEditCtrl::GetLine, CRichEditCtrl::LineIndex

CRichEditCtrl::LineIndex

int LineIndex(int *nLine* = -1) const;

返回值

返回在 *nLine* 中指定的行的字符索引；如果指定的行号大于编辑控件中的行数，

则返回值为-1。

参数

nLine

包含编辑控件中的指定行的索引值，或者包含-1。如果 *nLine* 是-1，则它指定的是当前行，也就是包含脱字符号（^）的行。

说明

此成员函数用来获取此 CRichEditCtrl 对象中的一行的字符索引。字符索引就是带格式编辑控件中从开始计算的字符数目。对于字符计数，一个 OLE 项被计算为一个单字符。

更多的信息，参见 Win32 文档中的 EM_LINEINDEX。

请参阅 CRichEditCtrl::LineFromChar, CRichEditCtrl::GetLineCount

CRichEditCtrl::LineLength

int LineLength(int *nLine* = -1) const;

返回值

当 LineLength **是对一个多行的编辑控件调用时，返回的是由** *nLine* **指定的行的长度（以字节计算）。当** LineLength **是对一个单行编辑控件调用时，返回值是编辑控件中的文本的长度。**

参数

nLine

指定其长度要被返回的行中的一个字符的字符索引。如果这个参数是-1**，则返回当前行（包含脱字符号的行）的长度，不包括此行中任何被选择文本的长度。当** LineLength **是对一个单行编辑控件调用时，这个参数被**

忽略。

说明

此成员函数用来获取一个带格式编辑控件中的一行的长度。

使用 LineIndex 成员函数可以获取此 CRichEditCtrl 对象中的一个给定行号的字符索引。

更多的信息，参见 Win32 文档中的 EM_LINELENGTH。

请参阅 CRichEditCtrl::LineIndex

CRichEditCtrl::LineScroll

void LineScroll(int *nLines*, int *nChars* = 0);

参数

nLines

指定垂直滚动的行数。

nChars

指定水平滚动的字符数。如果带格式编辑控件是 ES_RIGHT 或 ES_CENTER 风格，则这个值被忽略。编辑风格在 Create 中指定。

说明

此成员函数用来滚动一个多行编辑控件的文本。

编辑控件不能用来垂直滚动过编辑控件的最后一行。如果当前行的行号再加上 *nLines* 指定的行数超过了编辑控件中的总行数，则将调整这个值，以使在滚动到编辑控件中的最后一行后再滚动到控件顶部的行。

LineScroll 可以水平滚动过任意行的最后一个字符。

更多的信息，参见 Win32 文档中的 EM_LINESCROLL。

请参阅　CRichEditCtrl::LineIndex

CRichEditCtrl::Paste

void Paste();

说明

此成员函数用来将来自剪贴板的数据插入到 CRichEditCtrl 中的插入点处，即脱字符号所在的位置。只有当剪贴板上的数据是能够被认可的格式时，数据才会被插入。

更多的信息，参见 Win32 文档中的 WM_PASTE。

请参阅 CRichEditCtrl::Copy, CRichEditCtrl::Cut, CRichEditCtrl::PasteSpecial

CRichEditCtrl::PasteSpecial

void PasteSpecial(UINT *nClipFormat*, DWORD *dvAspect* = 0, HMETAFILE *hMF* = 0);

参数

nClipFormat

要粘贴到此 CRichEditCtrl **对象中的剪贴板格式。**

dvAspect

要从剪贴板获取的数据的设备外观。

hMF

一个图元文件的句柄，该图元文件包含了要被粘贴的对象的图标视。

说明

此成员函数用来将特定剪贴板格式的数据粘贴到此 CRichEditCtrl **对象中。新的材料被在插入点，即脱字符号所在处插入。**

更多的信息，参见 Win32 **文档中的** EM_PASTESPECIAL**。**

请参阅 CRichEditCtrl::Paste, CRichEditCtrl::Copy, CRichEditCtrl::Cut

CRichEditCtrl::ReplaceSel

void ReplaceSel(LPCTSTR *lpszNewText*, BOOL *bCanUndo* = FALSE);

参数

lpszNewText

指向一个以空字符结尾的字符串的指针，该字符串包含了用于替换的文本。

bCanUndo

要指定这个功能是可以取消的，就把这个参数设置为 TRUE。缺省的值是 FALSE。

说明

此成员函数使用 CWnd::SetWindowText 来用指定的文本替换此 CRichEditCtrl 对象中的当前选择的文本。

如果没有当前选择，则替换文本被插入当前脱字符号所在的位置。

更多的信息，参见 Win32 文档中的 EM_REPLACESEL。

请参阅 CRichEditCtrl::CanUndo, CRichEditCtrl::Undo, CWnd::SetWindowText

CRichEditCtrl::RequestResize

```
void RequestResize();
```

说明

此成员函数用来强迫此 CRichEditCtrl **对象向它的父窗口发送** EN_REQUESTRESIZE **通知消息。在** CWnd::OnSize **处理一个无底的** CRichEditCtrl **对象时，这个函数是很有用的。**

更多的信息，参见 Win32 **文档中的** EM_REQUESTRESIZE **消息和文章“无底的带格式控件”。**

请参阅 CWnd::OnSize, CRichEditCtrl::Create

CRichEditCtrl::SetBackgroundColor

COLORREF SetBackgroundColor(BOOL *bSysColor*, COLORREF *cr*);

返回值

返回这个 CRichEditCtrl **对象先前的背景颜色。**

参数

bSysColor

表明背景色是否要设置为系统值。如果这个值是 TRUE**，则** *cr* **被忽略。**

cr

要求的背景色。只有在 *bSysColor* **是** FALSE **时才使用。**

说明

此成员函数用来设置这个 CRichEditCtrl **对象的背景色。背景色可以设置为系统值，或者是设置为指定的** COLORREF **值。**

更多的信息，参见 Win32 **文档中的** EM_SETBKGNDCOLOR **消息和** COLORREF

结构。

请参阅 CDC::SetBkColor

CRichEditCtrl::SetDefaultCharFormat

BOOL SetDefaultCharFormat(CHARFORMAT& *cf*);

返回值

如果成功则返回非零值；否则返回 0。

参数

cf

一个包含新的缺省字符格式化属性的 CHARFORMAT 结构。

说明

此成员函数用来为这个 CRichEditCtrl **对象中的新文本设置字符格式化属性。这个函数只改变** *cf* **中的** dwMask **成员指定的属性。**

更多的信息，参见 Win32 **文档中的** EM_SETCHARFORMAT **消息和** CHARFORMAT **结构。**

请参阅 CRichEditCtrl::GetDefaultCharFormat, CRichEditCtrl::SetSelectionCharFormat

CRichEditCtrl::SetEventMask

DWORD SetEventMask(DWORD *dwEventMask*);

返回值

返回先前的事件 Mask。

参数

dwEventMask

这个 CRichEditCtrl **对象的新的事件** Mask。

说明

此成员函数用来为这个 CRichEditCtrl **对象设置新的事件** Mask。**此事件掩码指定了** CRichEditCtrl **对象将哪一个通知消息发送给它的父窗口。**

更多的信息，参见 Win32 **文档中的** EM_SETEVENTMASK。

请参阅 CRichEditCtrl::GetEventMask

CRichEditCtrl::SetModify

void SetModify(BOOL *bModified* = TRUE);

参数

bModified

如果是 TRUE，则表明文本被修改了；如果是 FALSE，则表明文本没有被修改。这个修改标志被缺省地设置。

说明

此成员函数用来为一个编辑控件设置或清除修改标志。此修改标志表明编辑控件中的文本是否被修改了。不管用户什么时候改变文本，这个标志都自动被设置。可以用 GetModify 成员函数来获取这个标志的值。

更多的信息，参见 Win32 文档中的 EM_SETMODIFY。

请参阅 CRichEditCtrl::GetModify

CRichEditCtrl::SetOLECallback

BOOL SetOLECallback(IRichEditOleCallback* *pCallback*);

参数

pCallback

指向一个 IRichEditOleCallback **对象的指针。** CRichEditCtrl **对象将利用这个对象来获取与** OLE **相关的资源和信息。**

返回值

如果成功则返回非零值；否则返回 0**。**

说明

此成员函数用来给 CRichEditCtrl **对象提供一个** IRichEditOleCallback **对象，用**

来访问与 OLE 相关的资源和信息。此 CRichEditCtrl 对象将调用 IUnknown::AddRef 来增加 *pCallback* 指定的 COM 对象的使用计数。

更多的信息，参见 Win32 文档中的 EM_SETOLEINTERFACE 消息和 IRichEditOleCallback 接口。

请参阅 CRichEditCtrl::GetIRichEditOle

CRichEditCtrl::SetOptions

void SetOptions(WORD *wOp*, DWORD *dwFlags*);

参数

wOp

用来表明存在的类型。可以是下列值之一：

- ECOOP_SET 将选项设置为 *dwFlags* 所指定的那样。

- ECOOP_OR　将当前选项与 *dwFlags* 所指定的组合。
- ECOOP_AND　只保留那些与 *dwFlags* 所指定的一样的当前选项。
- ECOOP_XOR　值保留那些没有由 *dwFlags* 指定的当前选项。

dwFlags

是带格式编辑选项。在说明部分给出了列出这个标志的可能取值。

说明

此成员函数用来为 CRichEditCtrl 对象设置选项。

选项可以是下列值的组合：

- ECO_AUTOWORDSELECTION　双击进行自动单词选择。
- ECO_AUTOVSCROLL　当用户在行的最末段输入一个字符时，文本自动向右滚动 10 个字符。当用户按下 ENTER 键时，控件将所有文本滚动到零

位。

- ECO_AUTOHSCROLL 当用户在最后一行按下 ENTER 键时，文本自动向上滚动一页。
- ECO_NOHIDESEL 取消一个编辑控件的缺省行为。当控件失去输入焦点时缺省行为隐藏选择，当控件结束输入焦点时则显示选择。如果你指定 ECO_NOHIDESEL，则存在的文本被转换，即使是没有焦点的情况下。
- ECO_READONLY 禁止用户在编辑控件中输入或编辑文本。
- ECO_WANTRETURN 当用户在一个对话框中的多行带格式编辑控件中输入文本按下 ENTER 键时，指定插入一个回车换行符。如果你没有指定这个风格，按下 ENTER 键将给带格式编辑控件的父窗口发送一个命令，模拟点击了父窗口的缺省按钮（例如，对话框中的 OK 按钮）。这个风格对单行编辑控件来说是没有影响的。

- ECO_SAVESEL　当控件失去焦点时，保存选择。缺省的，当焦点返回时，控件的所有内容都被选择。
- ECO_VERTICAL　在垂直方向上绘制文本和对象。只对亚洲人的语言有效。

更多的信息，参见 Win32 文档中的 EM_SETOPTIONS。

请参阅　CRichEditCtrl::HideSelection, CRichEditCtrl::SetReadOnly

CRichEditCtrl::SetParaFormat

BOOL SetParaFormat(PARAFORMAT& *pf*);

返回值

如果成功则返回非零值；否则返回 0。

参数

pf

一个包含新的缺省段落格式化属性的 PARAFORMAT 结构。

说明

此成员函数用来为 CRichEditCtrl 对象中的当前选择设置段落格式化属性。这个函数只改变 *pf* 中 dwMask 成员指定的属性。

更多的信息，参见 Win32 文档中的 EM_SETPARAFORMAT 消息和 PARAFORMAT 结构。

请参阅 CRichEditCtrl::GetParaFormat, CRichEditCtrl::SetSelectionCharFormat

CRichEditCtrl::SetReadOnly

BOOL SetReadOnly(BOOL *bReadOnly* = TRUE);

返回值

如果成功则返回非零值；否则返回 0。

参数

bReadOnly

表明此 CRichEditCtrl **对象是否应该是只读的。**

说明

此成员函数用来改变这个 CRichEditCtrl **对象的** ECO_READONLY **消息。有关这个消息的简短描述，参见** SetOptions。**你可以使用这个函数来设置这个**

CRichEditCtrl 对象的所有选项。

更多的信息，参见 Win32 文档中的 EM_SETREADONLY。

请参阅　CRichEditCtrl::Create, CRichEditCtrl::SetOptions

CRichEditCtrl::SetRect

void SetRect(LPCRECT *lpRect*);

参数

lpRect

是一个 CRect 或一个指向 RECT 的指针。它表明了格式化矩形的新的界线。

说明

此成员函数用来为这个 CRichEditCtrl 对象设置新的格式化矩形。这个格式化矩

形是文本的限制矩形。此限制矩形与带格式编辑控件的大小无关。当此 CRichEditCtrl 对象第一次被创建时，格式化矩形与窗口的客户区具有相同的尺寸。使用 SetRect 可以使格式化矩形变得比带格式编辑窗口大或小。

更多的信息，参见 Win32 文档中的 EM_SETRECT。

请参阅 CRichEditCtrl::GetRect

CRichEditCtrl::SetSel

```
void SetSel( long nStartChar, long nEndChar );
void SetSel( CHARRANGE& cr );
```

参数

nStartChar

选择中的第一个字符的从零开始的索引。

nEndChar

选择中的最后一个字符的从零开始的索引。

cr

一个 CHARRANGE 结构，包含了当前选择的界线。

说明

此成员函数用来设置这个 CRichEditCtrl 对象中的选择。

这个函数的两种形式都用来设置选择的界线，它们可以相互替换。有关这两种形式的简短描述如下所示：

- SetSel(*cr*) 这种形式用 CHARRANGE 结构的 cpMin 和 cpMax 成员来设置界线。
- SetSel(*nStarChar, nEndChar*) 这种形式用参数 *nStartChar* 和 *nEndChar* 来设置界线。

脱字符号被放置在由开始（cpMin 或 *nStartChar*）和结束（cpMax 或 *nEndChar*）索引中的较大者所指定的选择的结尾处。此函数不滚动 CRichEditCtrl 的内容，以使脱字符号是可见的。

要选择此 CRichEditCtrl 对象中的索引文本，可以用一个为 0 的开始索引和一个为 -1 的结束索引来调用 SetSel。

更多的信息，参见 Win32 文档中的 EM_EXSETSET 消息和 CHARRANGE 结构。

请参阅 CRichEditCtrl::GetSel, CRichEditCtrl::GetSelectionType

CRichEditCtrl::SetSelectionCharFormat

BOOL SetSelectionCharFormat(CHARFORMAT& *cf*);

返回值

如果成功则返回非零值；否则返回 0。

参数

cf

一个 CHARFORMAT **结构，包含了当前选择的字符格式化属性。**

说明

此成员函数用来设置这个 CRichEditCtrl **对象中的当前选择的文本的字符格式化属性。这个函数只改变由** *cf* **中的** dwMask **成员指定的属性。**

更多的信息，参见 Win32 **文档中的** EM_SETCHARFORMAT **消息和** CHARFORMAT **结构。**

请　　参　　阅　　CRichEditCtrl::GetSelectionCharFormat, CRichEditCtrl::SetDefaultCharFormat

CRichEditCtrl::SetTargetDevice

BOOL SetTargetDevice(HDC *hDC*, long *lLineWidth*);
BOOL SetTargetDevice(CDC& *dc*, long *lLineWidth*);

返回值

如果成功则返回非零值；否则返回 0。

参数

hDC

新的目标设备的设备环境句柄。

lLineWidth

用于格式化的线宽。

dc

新的目标设备的 CDC。

说明

此成员函数用来设置目标设备和此 CRichEditCtrl 对象中的用于 WYSIWYG（所见即所得）的线宽。

如果这个函数成功，则该带格式编辑控件就拥有了作为参数传递来的设备环境。在这种情况下，调用函数不应该销毁这个设备环境。

更多的信息，参见 Win32 文档中的 EM_SETTARGETDEVICE。

请参阅 RichEditCtrl::FormatRange, CRichEditCtrl::DisplayBand

CRichEditCtrl::SetWordCharFormat

BOOL SetWordCharFormat(CHARFORMAT& *cf*);

返回值

如果成功则返回非零值；否则返回零。

参数

cf

一个 CHARFORMAT **结构，包含了当前选择单词的字符格式化属性。**

说明

此成员函数用来设置这个 CRichEditCtrl **对象中的当前选择单词的字符格式化属性。这个函数只影响** *cf* **中的** dwMask **成员指定的属性。**

更多的信息，参见 Win32 **文档中的** EM_SETCHARFORMAT **消息和** CHARFORMAT **结构。**

请参阅 CRichEditCtrl::SetSelectionCharFormat

CRichEditCtrl::StreamIn

long StreamIn(int *nFormat*, EDITSTREAM& *es*);

返回值

返回从输入流中读到的字符数。

参数

nFormat

用来指定输入数据格式的标志。参见说明部分可以获得更多的信息。

es

一个 EDITSTREAM **结构，用来指定输入流。参见说明部分可以获得更多的信息。**

说明

此成员函数用从指定的输入流中获得的文本来替换 CRichEditCtrl **对象中的文本。**

nFormat **的值必须是下列值之一：**

- SF_TEXT **表明只读取文本。**
- SF_RTF **表明读取文本并进行格式化。**

这两个值中的任何一个都可以与 SFF_SELECTION **组合。如果指定了** SFF_SELECTION**，则** StreamIn **就替换这个** CRichEditCtrl **对象的所有内容。**

在 EDITSTREAM **参数** *es* **中，你可以指定一个用文本来填充缓存的收回函数。这个收回函数被重复调用，直至输入流被用完为止。**

更多的信息，参见 Win32 **文档中的** EM_STREAMIN **消息和** EDITSTREAM **结**

构。

请参阅 CRichEditCtrl::StreamOut

CRichEditCtrl::StreamOut

long StreamOut(int *nFormat*, EDITSTREAM& *es*);

返回值

返回要写到输出流中去的字符数。

参数

nformat

用来指定输出数据格式的标志。参见说明部分可以获得更多的信息。

es

用来指定输出流的 EDITSTREAM **结构。参见说明部分可以获得更多的信**

息。

说明

此成员函数用来将此 CRichEditCtrl 对象中的内容写到指定的输出流中。

nFormat 的值必须是下列值之一：

- SF_TEXT　表示只是写文本。
- SF_RTF　表明写文本并进行格式化。
- SF_RTFNOOBJS　表明是写文本并进行格式化，用空格来替换 OLE 项。
- SF_TEXTIZED　表明写文本并进行格式化，使用 OLE 项的原文表示。

这些值中的任何一个都可以与 SFF_SELECTION 组合。如果指定了 SFF_SELECTION，则 StreamOut 将读取选择写入输出流。如果没有指定，则 StreamOut 将 CRichEditCtrl 对象的所有内容都写入输出流中。

在 EDITSTREAM 参数 *es* 中，你可以指定一个用文本来填充缓存的收回函数。这个收回函数被重复调用，直至输入流被用完为止。

更多的信息，参见 Win32 文档中的 EM_STREAMOUT 消息和 EDITSTREAM 结构。

请参阅 CRichEditCtrl::StreamIn

CRichEditCtrl::Undo

BOOL Undo();

返回值

如果取消操作成功则返回非零值；否则返回 0。

说明

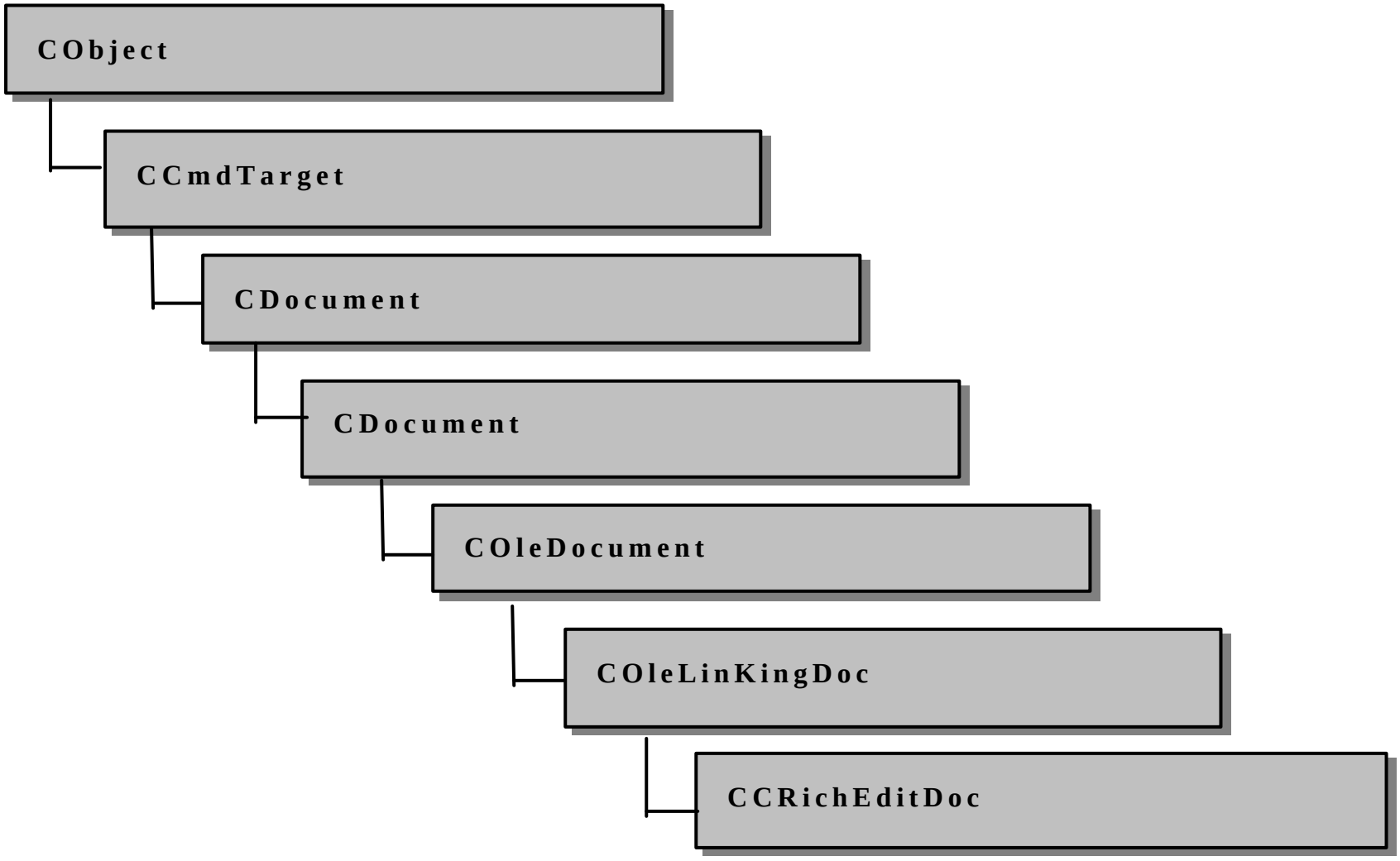
此成员函数用来取消带格式编辑控件中的最后一次操作。

一次取消操作也可以被取消。例如，你可以用对 Undo 的第一次调用来恢复被删除的文本。当还没有进行其它的编辑操作时，你可以通过对 Undo 的第二次调用来再次删除这些文本。

更多的信息，参见 Win32 文档中的 EM_UNDO。

请参阅 CRichEditCtrl::CanUndo, CRichEditCtrl::EmptyUndoBuffer

CRichEditDoc



一个“带格式编辑控件”是一个窗口，在这个窗口中用户可以输入和编辑文本。文本可以被赋予字母或段落的格式化，也可以包括嵌入的 OLE 对象。带格式编辑控件为格式化文本提供了设计界面。但是，应用程序必须实现需要的用户部件，以使格式化操作对用户来说可用。

CRichEditDoc 类与 CRichEditView 和 CRichEditCntrItem 一起，提供了 MFC 文档视环境中的带格式编辑控件的功能。CRichEditView 保存文本和文本的格式化特征。CRichEditDoc 保持视中的客户项的列表。CRichEditCntrItem 支持从容器方对 OLE 客户项的访问。

这个 Windows 通用控件（也就是 CRichEditCtrl 及其相关类）只对于运行在 Window95 和 Windows NT 3.51 及更新版本下的程序是可用的。

有关在一个 MFC 应用程序中使用带格式编辑文档的例子，请参见 WORDPAD 示例应用程序。

#include <afxrich.h>

请参阅 CRichEditView, CRichEditCntrItem, COleDocument, CRichEditCtrl

CRichEditDoc 类成员

属性	
GetStreamFormat	指示输入/输出流是否要包含格式化信息
GetView	获取相关的 CrichEditView 对象

数据成员	
m_bRTF	指示 I/O 流应该包含格式化

可重载	
CreateClientItem	此函数用来实现文档的清除

成员函数

CRichEditDoc::CreateClientItem

virtual CRichEditCntrItem* CreateClientItem(REOBJECT* *preo* = NULL) const = 0;

返回值

返回指向添加到文档中的新 CRichEditCntrItem 对象的指针。

参数

preo

指向一个 REOBJECT 结构的指针，该结构描述一个 OLE 项。行的 CRichEditCntrItem 对象就是围绕这个 OLE 项构造的。如果 *preo* 是 NULL，

则新的客户项是空的。

说明

此成员函数用来创建一个 CRichEditCntrItem 对象，并将它添加到这个文档中。

此函数不实现任何 OLE 初始化。

更多的信息，参见 Win32 文档中的 REOBJECT 结构。

请参阅 CRichEditCntrItem::CRichEditCntrItem, COleDocument::AddItem

CRichEditDoc::GetStreamFormat

int GetStreamFormat( ) const;

返回值

返回下列标志之一：

- SF_TEXT 表示此带格式编辑控件没有包含格式化信息。
- SF_RTF 表示此带格式编辑控件包含格式化信息。

说明

此成员函数用来确定带格式编辑内容的用于流式传输的文本格式。返回值是基于 m_bRTF 数据成员的。如果 m_bRTF 是 TRUE，则函数返回 SF_RTF；否则函数返回 SF_TEXT。

请参阅 CRichEditDoc::m_bRTF, CRichEditCtrl::StreamIn, CRichEditCtrl::StreamOut

CRichEditDoc::GetView

CRichEditView* GetView( ) const;

返回值

返回指向与文档相关的 CRichEditView **对象的指针。**

说明

此成员函数用来访问与这个 CRichEditDoc **对象相关联的** CRichEditView **对象。文本和格式化信息被包含在** CRichEditView **对象中。**CRichEditDoc **对象包含了用于连载的** OLE **项。每一个** CRichEditDoc **只能有一个** CRichEditView **与之相对应。**

请参阅 CRichEditView, CDocument::GetNextView

数据成员

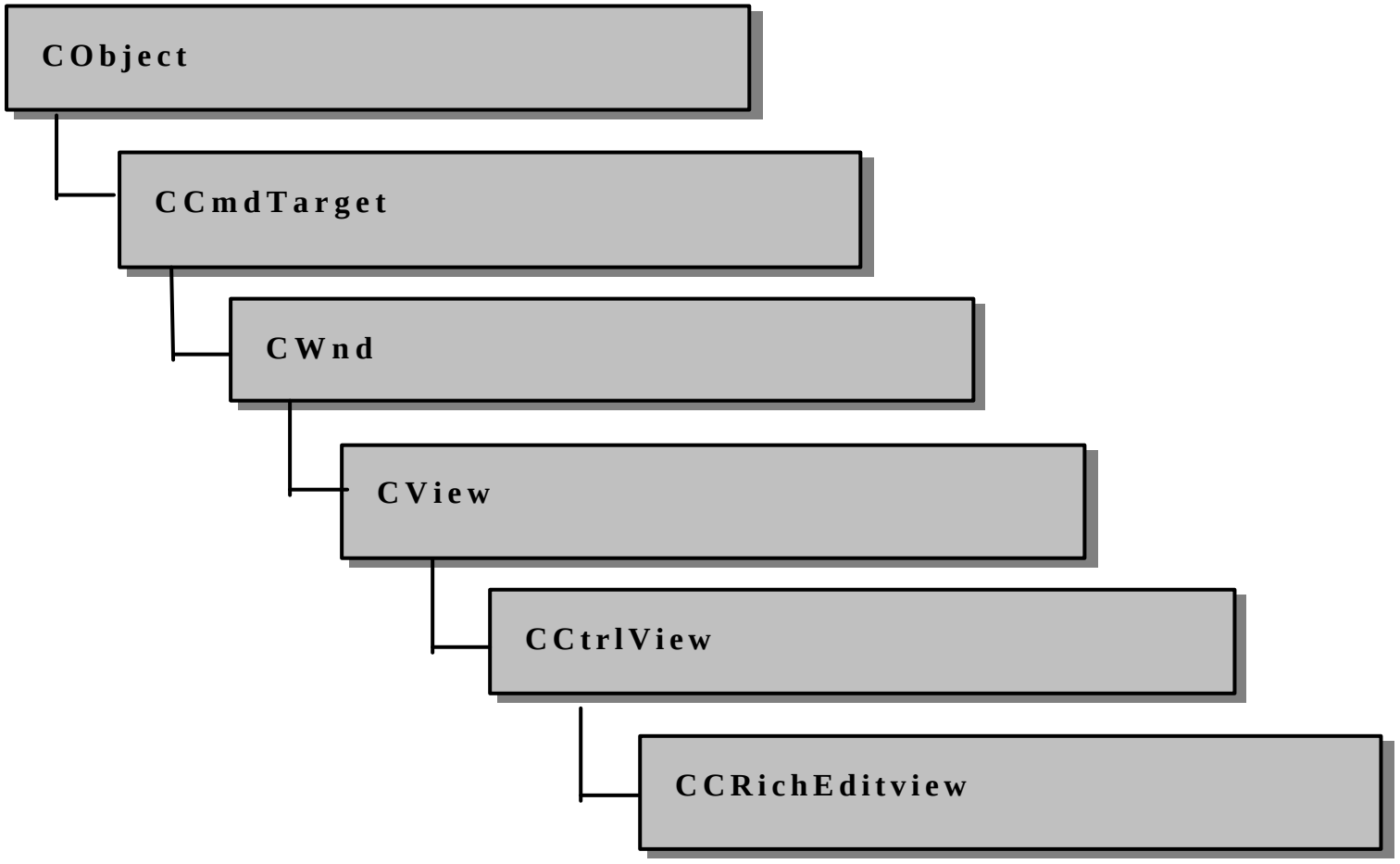
CRichEditDoc::m_bRTF

说明

当此数据成员为 TRUE 时，表示 CRichEditCtrl::StreamIn 和 CRichEditCtrl::StreamOut 应该保存段落和字符格式化的特征。

请参阅 CRichEditDoc::GetStreamFormat

CRichEditView



一个“带格式编辑控件”是一个窗口，在这个窗口中用户可以输入和编辑文本。文本可以被赋予字母或段落的格式化，也可以包括嵌入的 OLE 对象。带格式编辑控件为格式化文本提供了设计界面。但是，应用程序必须实现需要的用户部件，以使格式化操作对用户来说可用。

CRichEditView 类与 CRichEditDoc 和 CRichEditCntrItem 一起，提供了 MFC 文档-视环境中的带格式编辑控件的功能。CRichEditView 保存文本和文本的格式化特征。CRichEditDoc 保持视中的客户项的列表。CRichEditCntrItem 支持从容器方对 OLE 客户项的访问。

这个 Windows 通用控件（也就是 CRichEditCtrl 及其相关类）只对于运行在 Window95 和 Windows NT 3.51 及更新版本下的程序是可用的。

有关在一个 MFC 应用程序中使用带格式编辑文档的例子，请参见 WORDPAD 示例应用程序。

#include <afxrich.h>

请参阅 CRichEditDoc, CRichEditCntrItem

CRichEditView 类成员

Constructor	
CrichEditView	构造一个 CrichEditView 对象

Attributes	
GetDocument	获取一个指向相关的 CRichEditDoc 对象的指针
GetCharFormatSelection	获取当前选择的字符格式化属性
SetCharFormat	设置当前选择的字符格式化属性
GetParaFormatSelection	获取当前选择的段落格式化属性
SetParaFormat	设置当前选择的段落格式化属性
GetTextLength	获取 rich 编辑视中的文本的长度
GetPaperSize	获取此 rich 编辑视的纸张大小

续表

SetPaperSize	设置此 rich 编辑视的纸张大小
GetMargins	获取此 rich 编辑视的页边距
SetMargins	设置此 rich 编辑视的的页边距
GetPrintWidth	获取此 rich 编辑视的的打印宽度
GetPrintRect	获取此 rich 编辑视的的打印矩形
GetPageRect	获取此 rich 编辑视的的页面矩形
GetSelectedItem	从此 rich 编辑视中获取被选择的项
GetInPlaceActiveItem	获取此 rich 编辑视中的当前现场激活的 OLE 项
GetRichEditCtrl	获取此带格式编辑控件

Data Members	
m_nBulletIndent	表示公告列表的缩进量
m_nWordWrap	表示单词的折行约束

Operations	
FindText	激活等待光标，查找指定的文本
FindTextSimple	查找指定文本
IsRichEditFormat	确定剪贴板中是否包含一个 rich 编辑中的数据或文本格式
CanPaste	确定剪贴板包含的数据是否能够被粘贴到此 rich 编辑视中
DoPaste	将一个 OLE 项粘贴到此 rich 编辑视中
InsertItem	插入一个新项作为一个 OLE 项
InsertFileAsObject	插入一个文件作为一个 OLE 项
AdjustDialogPosition	移动一个对话框以使它不遮蔽当前的选择
OnCharEffect	改变当前选择的字符格式
OnParaAlign	改变段落的对齐方式
OnUpdateCharEffect	更新字符公有成员函数的命令 UI
OnUpdateParaAlign	更新段落公有成员函数的命令 UI
PrintInsideRect	格式化在给定矩形中的指定文本
PrintPage	格式化给定页中的指定文本

Overridables	
OnInitialUpdate	在一个视第一次与一个文档连接时刷新这个视
IsSelected	表明给定的 OLE 项是否被选择了
OnFindNext	搜索一个子串的下一次发现
OnTextNotFound	在没有找到需要的文本时处理用户通知
OnReplaceAll	用一个新的字符串来替换所有的给定字符串
OnReplaceSel	替换当前的选择
QueryAcceptData	查询 IDataObject 上的数据
OnPasteNativeObject	从一个 OLE 项获取本地数据
OnPrinterChanged	设置给定设备的打印特征
WrapChanged	根据 m_nWordWrap 的值为此 rich 编辑视调整目标输出设备
GetClipboardData	为此 rich 编辑视中的一个范围获取一个剪贴板对象
GetContextMenu	获取一个用于按下右鼠标按钮的上下文菜单

成员函数

CRichEditView::AdjustDialogPosition

void AdjustDialogPosition(CDialog* *pDlg*);

参数

pDlg

指向一个 CDialog **对象的指针。**

说明

此函数用来移动给定的对话框以使它不会隐蔽当前的选择。

请参阅 CRichEditCtrl::GetSel

CRichEditView::CanPaste

BOOL CanPaste( ) const;

返回值

如果剪贴板中包含的数据的格式是此 rich **编辑视可以接收的，则返回非零值；否则返回** 0。

说明

此成员函数用来确定剪贴板包含的信息是否可以粘贴到此 rich **编辑视中。**

请参阅 CRichEditCtrl::Paste, CRichEditView::DoPaste, CRichEditView::IsRichEditFormat

CRichEditView::CRichEditView

CRichEditView( );

说明

此成员函数用来创建一个 CRichEditView **对象。**

请参阅 CRichEditDoc, CRichEditCtrl

CRichEditView::DoPaste

void DoPaste(COleDataObject& *dataobj*, CLIPFORMAT *cf*, HMETAFILEPICT *hMetaPict*);

参数

dataobj

要粘贴的 COleDataObject **包含的数据。**

cf

所希望的剪贴板格式。

hMetaPict

代表要被粘贴的项的图元文件。

说明

此成员函数用来将 *dataobj* 中的 OLE 项粘贴到 rich 编辑文档/视中。可以调用这个函数作为 QueryAcceptData 实现的一部分。

这个函数根据 PasteSpecial 的处理者的结果来决定粘贴类型。如果 *cf* 是 0，则新项使用当前的图标来代表。如果 *cf* 是非零值并且 hMetaPict 不是 NULL，则新项使用 *hMetaPict* 作为它的代表。

请参阅 CRichEditCtrl::Paste, CRichEditView::IsRichEditFormat, CRichEditView::InsertItem

CRichEditView::FindText

BOOL FindText(LPCTSTR *lpszFind*, BOOL *bCase* = TRUE, BOOL *bWord* =

TRUE);

返回值

如果找到了 *lpszFind* **文本，则返回非零值；否则返回** 0。

参数

lpszFind

包含要搜索的字符串。

bCase

指明查找是否是大小写敏感的。

bWord

指明查找只是全词匹配，还是匹配词的一部分。

说明

此成员函数还是用来查找指定的文本，并将其设置为当前文本。这个函数在查找期间显示等待鼠标。

请参阅 CRichEditCtrl::FindText, CRichEditCtrl::SetSel, CRichEditView::FindTextSimple, CWaitCursor

CRichEditView::FindTextSimple

BOOL FindTextSimple(LPCTSTR *lpszFind*, BOOL *bCase* = TRUE, BOOL *bWord* = TRUE);

返回值

如果找到了 *lpszFind* **文本则返回非零值；否则返回** 0。

参数

lpszFind

包含要搜索的字符串。

bCase

指明查找是否是大小写敏感。

bWord

指明查找只是全词匹配，还是匹配词的一部分。

说明

此成员函数用来查找指定的文本，并将其设置为当前选择。

请参阅 CRichEditCtrl::FindText, CRichEditCtrl::SetSel, CRichEditView::FindText

CRichEditView::GetCharFormatSelection

CHARFORMAT& GetCharFormatSelection( );

返回值

返回一个 CHARFORMAT **结构，该结构包含了当前选择的字符格式化属性。**

说明

此成员函数用来获取当前选择的字符格式化属性。

更多的信息，参见 Win32 **文档中的** EM_GETCHARFORMAT **消息和** CHARFORMAT **结构。**

请参阅 CRichEditView::SetCharFormat, CRichEditView::GetParaFormatSelection, CRichEditCtrl::GetSelectionCharFormat

CRichEditView::GetClipboardData

virtual HRESULT GetClipboardData(CHARRANGE* *lpchrg*, DWORD *dwReco*,
LPDATAOBJECT *lpRichDataObj*, LPDATAOBJECT* *lplpdataobj*);

返回值

返回一个 HRESULT **值，该值用来报告操作的成功。有关** HRESULT **的更多信息，参见“** Platform SDK **”中的“** COM **错误代码结构”。**

参数

lpchrg

指向 CHARRANGE **结构的指针，该结构指定了要被拷贝到由** *lplpdataobj* **指定的数据结构中去的字符（和** OLE **项）的范围。**

dwReco

剪贴板操作标志。可以是下列值之一：

- RECO_COPY　拷贝到剪贴板。
- RECO_CUT　剪切到剪贴板。
- RECO_DRAG　拖动操作（拖放）。
- RECO_DROP　放开操作（拖放）。
- RECO_PASTE　从剪贴板粘贴。

lpRichDataObj

指向一个 IDataObject 对象的指针，该对象包含了来自带格式编辑控件（IRichEditOle:: GetClipboardData）的剪贴板数据。

lplpdataobj

指向一个指针变量的指针，该指针变量用来接收代表在 *lpchrg* 参数中指定的范围的 IDataObject 对象的地址。如果返回的是一个错误，则 *lplpdataobj* 的值被忽略。

说明

框架调用这个函数作为 IRichEditOleCallback::GetClipboardData **处理的一部分。如果返回值表示成功，则** IRichEditOleCallback::GetClipboardData **返回由** *lplpdataobj* **访问的** IDataObject**；否则，它返回由** *lpRichDataObj* **访问的对象。这个函数的缺省实现返回** E_NOTIMPL**。**

这是一个高级的可重载函数。

更多的信息，参见 Win32 **文档中的** IRichEditOle::GetClipboardData**，**IrichEditOleCallback:: GetClipbuardpata **和** CHARRANGE**，及参见** OLE **文档中的** IDataObject**。**

请参阅 COleServerItem::GetClipboardData

CRichEditView::GetContextMenu

virtual HMENU GetContextMenu(WORD *seltyp*, LPOLEOBJECT *lpoleobj*,
CHARRANGE* *lpchrg*);

返回值

返回上下文菜单的句柄。

参数

seltyp

选择的类型。在说明部分描述了选择的类型值。

lpoleobj

指向一个 OLEOBJECT **结构的指针，该结构指定了第一个被选择的** OLE **对象，如果选择中包含一个或更多的** OLE **项的话。如果选择没有包含项，则** *lpoleobj* **就是** NULL**。**OLEOBJECT **结构保存一个指向一个** OLE **对象**

v-table 的指针。

lpchrg

指向一个 CHARRANGE 结构的指针，该结构包含了当前的选择。

说明

框架调用这个成员函数作为 IRichEditOleCallback::GetContextMenu 处理的一部分。此函数是鼠标右键处理程序的典型组成部分。

选择类型可以时下列标志的任意组合：

- SEL_EMPTY 表明没有当前选择。
- SEL_TEXT 表明当前选择包含了文本。
- SEL_OBJECT 表明当前选择包含了至少一个 OLE 项。
- SEL_MULTICHAR 表明当前选择包含的文本不止一个字符。

- SEL_MULTIOBJECT 表明当前选择包含了不止一个的 OLE 对象。

此函数的缺省实现返回 NULL。这是一个高级的可重载函数。

更多的信息，参见 Win32 文档中的 IRichEditOleCallback::GetContextMenu 和 CHARRAN- GE。

更多有关 OLEOBJECT 类型的信息，参见“OLE 知识库”中的“OLE 数据结构和结构配置”。

请参阅 CRichEditCtrl::GetSelectionType

CRichEditView::GetDocument

```
CRichEditDoc* GetDocument( ) const;
```

返回值

返回一个指向 CRichEditDoc 对象的指针，该对象与应用程序的 CRichEditView 对象相关。

说明

此成员函数用来获取一个指向与这个视相关联的 CRichEditDoc 对象的指针。

请参阅 CRichEditDoc, CView::GetDocument, COleClientItem::GetDocument

CRichEditView::GetInPlaceActiveItem

CRichEditCntrItem* GetInPlaceActiveItem( ) const;

返回值

返回一个指向 CRichEditCntrItem 对象的指针,该对象是此 rich 编辑视中单一的，

可现场激活的 CRichEditCntrItem 对象。

说明

此成员函数用来获取此 CRichEditView 对象中的当前被现场激活的 OLE 项。

请参阅 COleDocument::GetInPlaceActiveItem, CRichEditCntrItem, CRichEditView::GetSelectedItem

CRichEditView::GetMargins

CRect GetMargins( ) const;

返回值

返回在打印中使用的页边距，此页边距是以 MM_TWIPS 来度量的。

说明

此成员函数用来接收在打印中使用的当前页边距。

请参阅 CRichEditView::SetMargins, CRichEditView::GetPrintWidth,
CRichEditView::GetPrintRect, CRichEditView::GetPaperSize,
CRichEditView::PrintPage, CRichEditView::WrapChanged

CRichEditView::GetPageRect

CRect GetPageRect( ) const;

返回值

返回在打印中使用的页面范围，以 MM_TWIPS **来度量。**

说明

此成员函数用来获取用于打印的页面尺寸。这个值是根据纸张大小来确定的。

请参阅 CRichEditView::GetMargins, CRichEditView::GetPrintWidth,
CRichEditView::GetPrintRect, CRichEditView::GetPaperSize,
CRichEditView::PrintPage

CRichEditView::GetPagerSize

CSize GetPaperSize( ) const;

返回值

返回用于打印的纸张大小，以 MM_TWIPS **度量。**

说明

此成员函数用来获取当前的纸张大小。

请参阅 CRichEditView::SetPaperSize, CRichEditView::GetMargins,
CRichEditView::GetPrintWidth, CRichEditView::GetPrintRect,
CRichEditView::GetPageRect, CRichEditView::PrintPage

CRichEditView::GetParaFormatSelection

```
PARAFORMAT& GetParaFormatSelection( );
```

返回值

返回一个 PARAFORMAT **结构，该结构包含了当前选择的段落格式化属性。**

说明

此成员函数用来获取当前选择的段落格式化属性。

更多的信息，参见 Win32 **文档中的** EM_GETPARAFORMAT **消息和** PARAFORMAT **结构。**

请参阅 CRichEditView::GetCharFormatSelection, CRichEditView::SetParaFormat, CRichEditCtrl::GetParaFormat

CRichEditView::GetPrintRect

CRect GetPrintRect( ) const;

返回值

用于打印的图像区域的界线。

说 明

此 成 员 函 数 用 来 获 取 页 面 矩 形 中 用 于 打 印 的 区 域 的 界 线 。

请 参 阅 CRichEditView::GetMargins, CRichEditView::GetPrintWidth,
CRichEditView::GetPaperSize, CRichEditView::GetPageRect,
CRichEditView::PrintPage

CRichEditView::GetPrintWidth

int GetPrintWidth( ) const;

返 回 值

返 回 打 印 区 域 的 宽 度 ， 以 MM_TWIPS **度 量 。**

说 明

此 成 员 函 数 用 来 确 定 打 印 区 域 的 宽 度 。

请参阅 CRichEditView::GetMargins, CRichEditView::GetPrintRect,
CRichEditView::GetPaperSize, CRichEditView::GetPageRect,
CRichEditView::PrintPage, CRichEditView::WrapChanged

CRichEditView::GetRichEditCtrl

```
CRichEditCtrl& GetRichEditCtrl( ) const;
```

返回值

返回用于这个视的 CRichEditCtrl **对象。**

说明

此成员函数用来获取与这个 CRichEditView **对象相关联的** CRichEditCtrl **对象。**

请参阅 CRichEditCtrl, CEditView::GetEditCtrl, CTreeView::GetTreeCtrl,
CListView::GetListCtrl

CRichEditView::GetSelectedItem

CRichEditCntrItem* GetSelectedItem( ) const;

返回值

返回指向一个在 CRichEditView **对象中选择的** CRichEditCntrItem **对象的指针；如果在这个视中没有被选择的项，则返回** NULL。

说明

此成员函数用来获取 CRichEditView **中的当前选择的** OLE **项（一个** CRichEditCntrItem **对象）。**

请参阅 CRichEditCntrItem, CRichEditView::GetInPlaceActiveItem

CRichEditView::GetTextLength

long GetTextLength( ) const;

返回值

返回这个 CRichEditView **对象中的文本的长度。**

说明

此成员函数用来获取这个 CRichEditView **对象中的文本长度。**

请参阅 CRichEditCtrl::GetTextLength

CRichEditView::InsertFileAsObject

void InsertFileAsObject(LPCTSTR *lpszFileName*);

参数

lpszFileName

是一个字符串，包含了要被插入的文件的名字。

说明

此成员函数用来将一个指定的文件（作为一个 CRichEditCntrItem **对象）插入到一个** rich **编辑视。**

请参阅 CRichEditView::InsertItem, CRichEditCntrItem

CRichEditView::InsertItem

HRESULT InsertItem(CRichEditCntrItem* *pItem*);

返回值

返回有关 HRESULT **值表示插入的成功。**

参数

pItem

指向要插入的项的指针。

说明

此成员函数用来将一个 CRichEditCntrItem **对象插入到一个** rich **编辑视。**

更多有关 HRESULT **的信息，参见“** Platform SDK **”中的“** COM **错误代码结构”。**

请参阅 CRichEditView::InsertFileAsObject, CRichEditCntrItem

CRichEditView::IsRichEditFormat

BOOL IsRichEditFormat(CLIPFORMAT *cf*);

返回值

如果 *cf* **是一个** rich **编辑或文本剪贴板格式，则返回非零值。**

参数

cf

关心的剪贴板格式。

说明

此成员函数用来确定 *cf* **剪贴板格式是文本，带格式文本，或是有** OLE **项的带格式文本。**

请参阅 CRichEditCtrl::CanPaste, CRichEditCtrl::Paste, CRichEditView::DoPaste

CRichEditView::IsSelected

virtual BOOL IsSelected(const CObject* *pDocItem*) const;

返回值

如果对象被选择则返回非零值；否则返回 0。

参数

pDocItem

指向视中的一个对象的指针。

说明

此成员函数用来确定指定的 OLE **项是否是视中当前被选择的项。**

如果你的派生视类有处理 OLE **项选择的不同方法，可以重载这个函数。**

请参阅 CRichEditView::GetSelectedItem, CRichEditView::GetInPlaceActiveItem

CRichEditView::OnCharEffect

void OnCharEffect(DWORD *dwMask*, DWORD *dwEffect*);

参数

dwMask

在当前选择中要修改的字符格式化效果。

dwEffect

想要的字符格式化效果的列表。

说明

此成员函数用来为当前选择改变字符格式化效果。

有关 *dwMask* 和 *dwEffect* 参数和它们的潜在值的更多信息，参见 Win32 文档中 CHARFORMAT 的相应数据成员。

请参阅 CRichEditView::SetCharFormat

CRichEditView::OnFindNext

virtual void OnFindNext(LPCTSTR *lpszFind*, BOOL *bNext*, BOOL *bCase*, BOOL *bWord*);

参数

lpszFind

要查找的字符串。

bNext

搜索的方向：如果是 TRUE 则是向下搜索；如果时 FALSE 则向上搜索。

bCase

表明查找是否是大小写敏感的。

bWord

表明查找是否需要匹配整个单词。

说明

当处理来自 Find/Replace 对话框的命令时，由框架调用此成员函数。此函数用来在 CRichEditView 中查找文本。重载这个函数可以为你的派生视类改变搜索特征。

请参阅 CRichEditView::FindText, CRichEditView::FindTextSimple

CRichEditView::OnInitialUpdate

virtual void OnInitialUpdate( );

说明

在第一次将视与文档连接之后，但在视被初始化显示之前，由框架打印此成员函数。这个函数的缺省实现打印了 CView::OnUpdate **成员函数，没有提示信息（即，参数使用缺省值，**lHint **为** 0**，**pHint **为** NULL**）。重载这个函数可以实现任何需要有关文档信息的一次初始化。例如，如果你的应用程序有固定大小的文档，你就可以根据文档的大小使用这个函数来初始化一个视的滚动极限。如果你的应用程序支持可变大小的文档，则可以在每次文档改变时使用** OnUpdate **来更新视的滚动极限。**

请参阅 CView::OnUpdate

CRichEditView::OnPasteNativeObject

virtual BOOL OnPasteNativeObject(LPSTORAGE *lpStg*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpStg

指向一个 *IStorage* **对象的指针。**

说明

此成员函数用来从一个嵌入项加载本地数据。通常，你可以通过创建一个围绕 IStorage **的** COleStreamFile **来实现这一点。这个** COleStreamFile **可以连接到一个文档，可以调用** CObject::Serialize **来加载数据。**

这是一个高级的可重载函数。

更多的信息，参见 OLE **文档中的** IStorage。

请参阅 ColeStreamFile,CObject::Serialize, CArchive

CRichEditView::OnParaAlign

void OnParaAlign(WORD *wAlign*);

参数

wAlign

想要的段落对齐方式。可以是下列值之一：

- PFA_LEFT **段落按左边距对齐。**
- PFA_RIGHT **段落按右边距对齐。**
- PFA_CENTER **段落居中对齐。**

说明

此函数用来改变选择段的段落对齐方式。

请参阅 CRichEditView::OnUpdateParaAlign

CRichEditView::OnPrinterChanged

virtual void OnPrinterChanged(const CDC& *dcPrinter*);

参数

dcPrinter

新打印机的 CDC **对象。**

说明

当打印机改变时，重载这个函数来改变带格式编辑视的特征。缺省的实现将纸

张的尺寸设置为输出设备（打印机）的物理高度和宽度。如果没有与 *dcPrinter* 相关联的设备环境。则缺省实现将纸张的尺寸设置为 8.5×11 寸。

请参阅 CRichEditView::SetPaperSize, CRichEditView::WrapChanged

CRichEditView::OnReplaceAll

virtual void OnReplaceAll(LPCTSTR *lpszFind*, LPCTSTR *lpszReplace*, BOOL *bCase*,
BOOL *bWord*);

参数

lpszFind

要被替换的文本。

lpszReplace

用来替换的文本。

bCase

表明查找是否是大小写敏感的。

bWord

表明查找是否需要选择整个单词。

说明

当处理来自 Replace 对话框的 Replace All 命令时框架调用此成员函数。此函数用另外的给定文本来替换所有出现的指定文本。重载这个函数可以改变这个视的查找特征。

请参阅 CRichEditView::OnReplaceSel, CRichEditView::OnFindNext

CRichEditView::OnReplaceSel

virtual void OnReplaceSel(LPCTSTR *lpszFind*, BOOL *bNext*, BOOL *bCase*, BOOL *bWord*,

LPCTSTR *lpszReplace*);

参数

lpszFind

要被替换的文本。

bNext

表明搜索的方向：如果是 TRUE **则表示向下搜索；如果是** FALSE **则表示向上搜索。**

bCase

表明搜索是否是大小写敏感的。

bWord

表明搜索是否必须选择整个单词。

lpszReplaced

用来替换的文本。

说明

当处理来自 Replac **对话框的** Replace **命令时框架调用此成员函数。此函数用另一个字符串来替换某个给定文本的一次出现。重载这个函数可以改变这个视的搜索特征。**

请参阅 CRichEditView::OnReplaceAll

CRichEditView::OnTextNotFound

virtual void OnTextNotFound(LPCTSTR *lpszFind*);

参数

lpszFind

没有查找到的文本。

说明

当一次搜索失败时框架调用此成员函数。重载这个函数可以改变来自一个 MessageBeep **的输出通知。**

更多的信息，参见 Win32 **文档的** MessageBeep。

请参阅 CRichEditView::FindText, CRichEditView::FindTextSimple, CRichEditView::OnFindNext

CRichEditView::OnUpdateCharEffect

void OnUpdateCharEffect(CCmdUI* *pCmdUI*, DWORD *dwMask*, DWORD *dwEffect*);

参数

pCmdUI

指向一个 CCmdUI **对象的指针。**

dwMask

表示字符格式化掩码。

dwEffect

表示字符格式化的效果。

说明

框架调用此成员函数来更新字符效果命令的命令 UI。掩码 *dwMask* 指定了要检查的字符格式化属性。标志 *dwEffect* 列出了要设置/清除的字符格式化属性。

有关 *dwMask* 和 *dwEffect* 参数和它们的潜在值的更多信息，参见 Win32 文档中的 CHARFORMAT 的相应数据成员。

CRichEditView::OnUpdateParaAlign

void OnParaAlign(CCmdUI* *pCmdUI*, WORD *wAlign*);

参数

pCmdUI

指向一个 CCmdUI **对象的指针。**

wAlign

要检查的段落对齐方式。可以是下列值之一：

- PFA_LEFT **段落按左边距对齐。**
- PFA_RIGHT **段落按右边距对齐。**
- PFA_CENTER **段落居中对齐。**

说明

框架调用此成员函数来更新段落效果命令的命令 UI。

请参阅 CRichEditView::GetParaFormatSelection, CRichEditView::OnParaAlign,

CRichEditView::SetParaFormat

CRichEditView::PrintInsideRect

long PrintInsideRect(CDC* *pDC*, RECT& *rectLayout*, long *nIndexStart*, long *nIndexStop*,
BOOL *bOutput*);

返回值

返回值为适合输出区域的最后一个字符的索引值再加 1。

参数

pDC

指向一个输出区域的设备环境的指针。

rectLayout

用来定义输出区域的 RECT **或** CRect。

nIndexStart

要被格式化的第一个字符的从零开始的索引。

nIndexStop

要被格式化的最后一个字符的从零开始的索引。

bOutput

表明是否要丢弃文本。如果时 FALSE，则文本正好适量。

说明

此成员函数用来为 *pDC* 指定的设备格式化一个带格式编辑控件中的一定范围的文本，使之正好在 *rectLayout* 之内。通常，在调用这个函数之后，会调用生成输出的 CRichEditCtrl::DisplayBand。

请参阅 CRichEditCtrl::FormatRange, CRichEditView::PrintPage, CrichEditCtrl::DisplayBand

CRichEditView::PrintPage

long PrintPage(CDC* *pDC*, long *nIndexStart*, long *nIndexStop*);

返回值

返回值为此页面中的最后一个字符的索引值加 1**。**

参数

pDC

指向一个用于页面输出的设备环境的指针。

nIndexStart

要被格式化的第一个字符的从零开始的索引。

nIndexStop

要被格式化的最后一个字符的从零开始的索引。

说明

此成员函数用来为 *pDC* 指定的设备格式化一个带格式编辑控件中的一定范围的文本。每一个页面的版面都由 GetPageRect 和 GetPrintRect 控制。通常，在调用这个函数之后会调用生成输出的 CRichEditCtrl::DisplayBand。

> **注意** PrintPase 使用的边距是与物理页相关的，而不是与逻辑页相关。因此，由于许多打印机在纸张上具有不打印的区域，所以为零的页边距将会剪裁文本。为了避免剪裁文本,你必须在打印前调用 SetMargins 来设置合理的打印机页边距。

请参阅 CRichEditView::PrintInsideRect, CRichEditView::GetPageRect, CRichEditView::GetPrintRect, CRichEditView::SetMargins

CRichEditView::QueryAcceptData

virtual HRESULT QueryAcceptData(LPDATAOBJECT *lpdataobj*, CLIPFORMAT FAR *
lpcfFormat, DWORD *dwReco*, BOOL *bReally*, HGLOBAL *hMetaFile*);

返回值

返回一个报告操作成功的 HRESULT **值。**

参数

lpdataobj

指向要查询的 IDataObject **的指针。**

lpcfFormat

指向可以接受的数据格式的指针。

dwReco

没有使用。

bReally

表明是否应该继续进行粘贴操作。

hMetaFile

一个用来绘制项的图标的图元文件的句柄。

说明

框架调用此成员函数来将一个对象粘贴到带格式编辑中。重载这个函数以在你的派生文档类中处理不同的 COM 项的组织。

有关 HRESULT 和 IDataObject 的更多信息,分别参见“ Platform SDK ”中的“ COM 错误代码结构 ”和 IDataObject。

CRichEditView::SetCharFormat

void SetCharFormat(CHARFORMAT *cf*);

参数

cf

包含新的缺省字符格式化属性的 CHARFORMAT **结构。**

说明

此成员函数用来为 CRichEditView **的新文本设置字符格式化属性。这个函数只改变** *cf* **的** dwMask **成员指定的属性。**

更多的信息，参见 Win32 **文档中的** EM_SETCHARFORMAT **消息和** CHARFORMAT **结构。**

请参阅 CRichEditView::GetCharFormatSelection, CRichEditView::SetParaFormat

CRichEditView::SetMargins

void SetMargins(const CRect& *rectMargin*);

参数

rectMargin

用于打印的新的页边距值，以 MM_TWIPS **度量。**

说明

此成员函数用来设置这个带格式编辑控件的打印页边距。如果 m_nWordWrap **是** WrapToTargetDevice**，则你应该在使用这个函数之后调用** WrapChanged **来调整打印特性。**

注意，PrintPage **使用的边距是与物理页相关的，而不是与逻辑页相关。因此，由于许多打印机在纸张上具有不打印的区域，所以为零的页边距将会剪裁文**

本。为了避免剪裁文本，你必须在打印前调用 SetMargins 来设置合理的打印机页边距。

请参阅 CRichEditView::GetMargins, CRichEditView::GetPrintWidth, CRichEditView::GetPrintRect, CRichEditView::GetPaperSize, CRichEditView::GetPageRect, CRichEditView::PrintPage, CRichEditView::WrapChanged

CRichEditView::SetPaperSize

void SetPaperSize(CSize *sizePaper*);

参数

sizePaper

用于打印的新的纸张尺寸值，以 MM_TWIPS 度量。

说明

此成员函数用来为打印这个带格式编辑视设置纸张的大小。如果 m_nWordWrap **是** WrapToTargetDevice**，则你应该在使用这个函数之后调用** WrapChanged **来调整打印特性。**

请参阅 CRichEditView::GetPaperSize, CRichEditView::GetMargins, CRichEditView::GetPrintWidth, CRichEditView::GetPrintRect, CRichEditView::GetPageRect, CRichEditView::PrintPage, CRichEditView::WrapChanged

CRichEditView::SetParaFormat

BOOL SetParaFormat(PARAFORMAT& *pf*);

返回值

如果成功则返回非零值；否则返回 0**。**

参数

pf

包含新的缺省度量格式化属性的 PARAFORMAT **结构。**

说明

此成员函数用来为这个 CRichEditView **对象中的当前选择设置段落格式化属性。这个函数只改变** *pf* **的** dwMask **成员指定的属性。**

更多的信息，参见 Win32 **文档中的** EM_SETPARAFORMAT **消息和** PARAFORMAT **结构。**

请参阅 CRichEditView::GetParaFormatSelection, CRichEditView::SetCharFormat

CRichEditView::WrapChanged

virtual void WrapChanged( );

说明

当打印特性被改变时（SetMargins **或** SetPaperSize**）调用此成员函数。**

重载这个函数可以修改这个带格式编辑视对 m_nWordWrap **中的变化或打印特性的变化（**OnPrinterChanged**）做出反应的方式。**

请参阅 CRichEditView::m_nWordWrap, CRichEditView::OnPrinterChanged, CRichEditView::SetMargins, CRichEditView::SetPaperSize

数据成员

CRichEditView::m_nBulletIndent

说明

此数据成员表示公告项在一个列表中的缩进；缺省的，为 720 个单位，就是 1/2 寸。

CRichEditView::m_nWordWrap

说明

此数据成员表示这个带格式编辑视的 word wrap 的类型。可以是下列值之一：

- WrapNone 表示没有自动的 word wrap。
- WrapToWindow 表示根据窗口的宽度来进行 word wrap。
- WrapToTargetDevice 表示根据目标设备的特性来进行 word wrap。

请参阅 CRichEditView::WrapChanged

CRuntimeClass

CRuntimeClass 没有基类。

每个由 CObject 派生的类都与一个 CRuntimeClass 结构相联系，用户可以使用该结构获取一个对象及其基类的运行时信息。当需要额外的函数参数检查时，或当用户必须根据一个对象的类编写特殊目的代码时，在运行时确定该对象的

类就非常有用。C++并不直接支持运行时类的信息。

此结构具有下列成员：

LPCSTR m_lpszClassName

存放 ASCII 类名的以空字符结尾的字符串。

int m_nObjectSize

以字节为单位给出对象的大小。若此对象具有指向被分配的内存的数据成员，则此值不包含该内存的大小。

UINT m_wSchema

分类编号（对不可分类的类，该值为-1）。对于此分类编号的详细说明，参见 IMPLEMENT_SERIAL 宏。

CObject* (PASCAL* m_pfnCreateObject)( )

是一个指向缺省的构造函数的函数指针，该构造函数创建一个你的类的对象（只有在类支持动态创建时才有效；否则，返回 NULL）。

CRuntimeClass* (PASCAL* m_pfn_GetBaseClass)( )

如果你的应用程序是动态地链接到 MFC 的 AFXDLL 版本，则是一个指向函数的指针，该函数返回基类的 CRuntimeClass 结构。

CRuntimeClass* m_pBaseClass

如果你的应用程序是静态地链接到 MFC 的，则是一个指向基类的 CRuntimeClass 结构的指针。

Feature Only in Professional and Enterprise Editions 只有在 Visual C++ 的专业版和企业版中才支持对 MFC 的静态链接。

CObject* CreateObject( );

从 CObject 派生的类可以支持动态创建，这是在运行时创建一个指定类的对象的能力。例如，文档，视和框架类就应该支持动态创建。CreateObject 成员函数可以用来实现这个功能，在运行时为这些类创建对象。

BOOL IsDerivedFrom(const CRuntimeClass* pBaseClass) const;

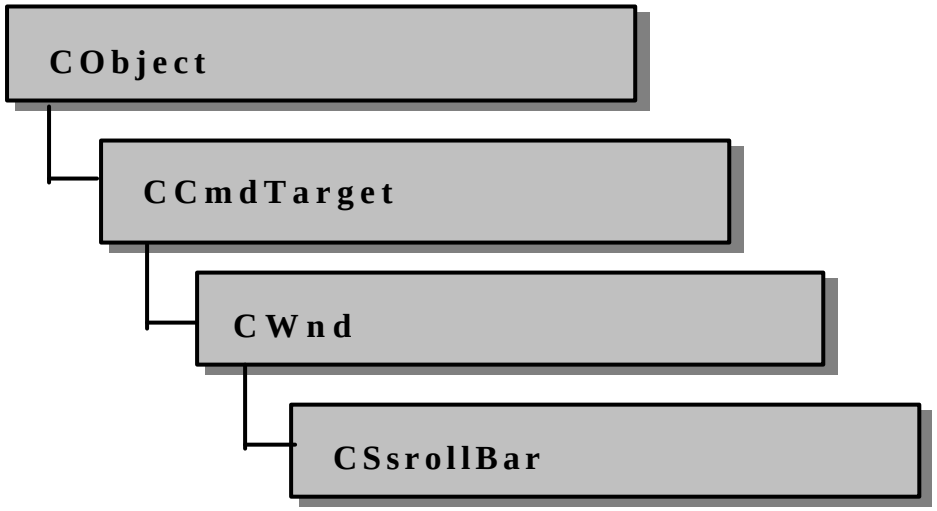
如果 IsDerivedFrom 类成员的类是从基类派生而来，该基类的

CRuntimeClass 结构作为一个参数给出，则返回 TRUE。IsDerivedFrom 从该成员的类开始向上沿派生类链经过所有的类直到顶端，并且只有在没有与基类匹配的类时才返回 FALSE。

注意 要使用 CRuntimeClass 结构，你必须在你想要获取运行时对象信息的类的实现中包括 IMPLEMENT_DYNAMIQ，IMPLEMENT_DYNCREATE，或 IMPLEMENT_SERIAL 宏。

请参阅 CObject::GetRuntimeClass, CObject::IsKindOf, RUNTIME_CLASS, IMPLEMENT_DYNAMIC, IMPLEMENT_DYNCREATE, IMPLEMENT_SERIAL

CScrollBar



CScrollBar 类支持一个 Windows 滚动条控件的功能。

创建一个滚动条控件分为两步。首先，调用构造函数 CScrollBar 来构造一个 CScrollBar 对象，然后调用 Create 成员函数来创建 Windows 滚动条控件并将它与 CScrollBar 对象相联系。

如果你是在一个对话框（通过一个对话框资源）中创建一个 CScrollBar 对象的，

则当用户关闭这个对话框时，CScrollBar 自动销毁。

如果你是在一个窗口中创建一个 CScrollBar 对象，则你必须负责销毁该对象。

如果你是堆栈中创建 CScrollBar 对象，则该对象将会自动销毁。如果你是通过使用 New 函数在数据堆中创建 CScrollBar 对象，则当用户终止使用该 Windows 滚动条时，你必须调用 delete 来销毁该对象。

如果你在此 CScrollBar 对象中分配了任何内存，则应重载 CScrollBar 析构函数来释放这些内存。

#include <afxwin.h>

请参阅 CWnd, CButton, CComboBox, CEdit, CListBox, CStatic, CDialog

CScrollBar 类成员

Construction	
CscrollBar	构造一个 CScrollBar 对象

Initialization	
Create	创建 Windows 滚动条，并将它连接到 CScrollBar 对象上

Operations	
GetScrollPos	获取一个滚动框的当前位置
SetScrollPos	设置一个滚动框的当前位置
GetScrollRange	获取给定滚动条的当前最大和最小滚动条位置
SetScrollRange	设置给定滚动条的最小和最大位置
ShowScrollBar	显示或隐藏一个滚动条
EnableScrollBar	使一个滚动条的一个或两个箭头有效或无效
SetScrollInfo	设置有关滚动条的信息
GetScrollInfo	获取有关滚动条的信息
GetScrollLimit	获取滚动条的极限

成员函数

CScrollBar::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果成功则返回非零值；否则返回 0。

参数

dwStyle

指定滚动条的风格。任何滚动条风格的组合都适用于滚动条。

rect

指定滚动条的大小和位置。可以是一个 RECT **结构或是一个** CRect **对象。**

pParentWnd

指定滚动条的父窗口，通常是一个 CDialog **对象。它不能是** NULL。

NID

该滚动条的控件 ID。

说明

可以分两步构造一个 CScrollBar 对象。首先调用构造函数来构造一个 CScrollBar 对象；然后调用 Create 来创建并初始化关联的 Windows 滚动条，并将它连接到该 CScrollBar 对象上。

可以将下列窗口风格使用到一个滚动条：

- WS_CHILD　总要使用。
- WS_VISIBLE　常用。
- WS_DISABLED　少用。
- WS_GROUP　将控件分组。

请参阅 CScrollBar::CScrollBar

CScrollBar::CScrollBar

CScrollBar( );

说明

此成员函数用来构造一个 CScrollBar **对象。在构造对象之后，调用** Create **成员函数来创建并初始化** Windows **滚动条。**

请参阅 CScrollBar::Create

CScrollBar::EnableScrollBar

BOOL EnableScrollBar(UINT *nArrowFlags* = ESB_ENABLE_BOTH);

返回值

如果按指定的要求使箭头有效或无效，则返回非零值；否则返回 0，这表明箭头已经处于需要的状态，或者是发生了一个错误。

参数

nArrowFlags

指定是要使给定箭头有效还是无效，及是使哪一个箭头有效或无效。这个参数可以是下列值之一：

- ESB_ENABLE_BOTH　使滚动条的两个箭头都有效。
- ESB_DISABLE_LTUP　使水平滚动条的左箭头或垂直滚动条的上箭头无效。
- ESB_DISABLE_RTDN　使水平滚动条的右箭头或垂直滚动条的下箭

头无效。

- ESB_DISABLE_BOTH　使滚动条的两个箭头都无效。

说明

此成员函数用来使滚动条的一个或两个箭头有效或无效。

请参阅　CWnd::EnableScrollBar, ::EnableScrollBar

CScrollBar::GetScrollInfo

BOOL GetScrollInfo(LPSCROLLINFO *lpScrollInfo*, UINT *nMask*);

返回值

如果消息获取了任何值，则返回 TRUE；否则返回 FALSE。

参数

lpScrollInfo

指向一个 SCROLLINFO 结构的指针。参见“Win32 程序员参考”可以获得更多有关这个结构的信息。

nMask

指定要获取的滚动条参数。通常使用的是：SIF_ALL，表示是 SIF_PAGE，SIF_POS，SIF_TRACKPOS 和 SIF_RANGE 的组合。参见 SCROLLINFO 可以获得有关 nMask 值的更多信息。

说明

此成员函数用来获取由 SCROLLINFO 结构保存的关于一个滚动条的信息。GetScrollInfo 允许应用程序使用 32 位的滚动位置信息。

SCROLLINFO 结构包含了有关一个滚动条的信息，包括滚动位置的最小和最大值，页面的大小和滚动框（拇指）的位置。参见“Win32 SDK 程序员参考”中的 SCROLLINFO 结构主题，可以获得有关改变结构的缺省值的更多信息。

MFC Windows 消息句柄表示了滚动条的位置，CWnd::OnHScroll 和 CWnd::OnVScroll 只提供 16 位的位置数据。GetScrollInfo 和 SetScrollInfo 提供 32 位的滚动条位置数据。因此，一个应用程序在处理 GetScrollInfo 或 SetScrollInfo 时，可以调用 GetScrollInfo 来获取 32 位的滚动条位置数据。

请参阅 CScrollBar::SetScrollInfo, CWnd::SetScrollInfo, CWnd::SetScrollPos, CWnd::OnVScroll, CWnd::OnHScroll, SCROLLINFO

CScrollBar::GetScrollLimit

```
int GetScrollLimit( );
```

返回值

如果成功则返回值指定滚动条最大位置；否则返回值为 0。

说明

此成员函数用来获取滚动条的最大滚动位置。

请参阅 CWnd::GetScrollLimit

CScrollBar::GetScrollPos

int GetScrollPos( ) const;

返回值

如果成功则返回指定滚动框当前位置的值；否则返回 0。

说明

此成员函数用来获取一个滚动框的当前位置。这个当前位置是一个相对值，它依赖于当前的滚动范围。例如，如果滚动范围是 100 **到** 200**，而滚动框是在滚动条的中间，则这个当前位置是** 150**。**

请参阅 CScrollBar::SetScrollPos, CScrollBar::GetScrollRange, CScrollBar::SetScrollRange, ::GetScrollPos

CScrollBar::GetScrollRange

void GetScrollRange(LPINT *lpMinPos*, LPINT *lpMaxPos*) const;

参数

lpMinPos

指向一个用来接收最小位置的整型变量。

lpMaxPos

指向一个用来接收最大位置的整型变量。

说明

此成员函数将滚动条的当前滚动条位置的最小和最大值拷贝为 *lpMinPos* 和 *lpMaxPos* 指定的值。

一个滚动条控件的缺省范围是空（两个值都为零）。

请参阅 ::GetScrollRange, CScrollBar::SetScrollRange, CScrollBar::GetScrollPos, CScrollBar::SetScrollPos

CScrollBar::SetScrollInfo

BOOL SetScrollInfo(LPSCROLLINFO *lpScrollInfo*, BOOL *bRedraw* = TRUE);

返回值

如果成功则返回 TRUE；**否则返回** FALSE。

参数

lpScrollInfo

指向一个 SCROLLINFO **结构的指针。**

bRedraw

指示这个滚动条是否应重画以反映新的信息。如果 *bRedraw* **是** TRUE，**则滚动条被重画。如果它是** FALSE，**则不重画滚动条。缺省的，滚动条将被重画。**

说明

此成员函数用来设置 SCROLLINFO **结构保持的有关一个滚动条的信息。你必**

须提供 SCROLLINFO 结构参数需要的值，包括标志值。

SCROLLINFO 结构包含了有关一个滚动条的信息，包括滚动位置的最小和最大值，页面的大小和滚动框（拇指）的位置。参见“Win32 SDK 程序员参考”中的 SCROLLINFO 结构主题，可以获得有关改变此结构的缺省值的更多信息。

请参阅 CScrollBar::GetScrollInfo, CWnd::SetScrollInfo, CWnd::SetScrollPos, CWnd::OnVScroll, CWnd::OnHScroll, CWnd::GetScrollInfo, SCROLLINFO

CScrollBar::SetScrollPos

int SetScrollPos(int *nPos*, BOOL *bRedraw* = TRUE);

返回值

如果成功则返回滚动框的先前位置；否则返回 0。

参数

nPos

指定滚动框的新位置。它必须是在滚动范围之内。

bRedraw

指示滚动条是否应该被重画来反映新的位置。如果 *bRedraw* **是** TRUE**，则滚动条被重画。如果它是** FALSE**，则不重画滚动条。缺省的，滚动条将被重画。**

说明

此成员函数用来将一个滚动框的当前位置设置为 *nPos* **指定的位置，并且，如果指定了重画，则重画滚动条来反映新的位置。**

请参阅 CScrollBar::GetScrollPos, CScrollBar::GetScrollRange, CScrollBar::SetScrollRange, ::SetScrollPos

CScrollBar::SetScrollRange

void SetScrollRange(int *nMinPos*, int *nMaxPos*, BOOL *bRedraw* = TRUE);

参数

nMinPos

指定滚动位置的最小值。

nMaxPos

指定滚动位置的最大值。

bRedraw

指示是否要重画滚动条来反映改变。如果 *bRedraw* **是** TRUE**，则滚动条被重画。如果它是** FALSE**，则不重画滚动条。缺省的，滚动条将被重画。**

说明

此成员函数用来为给定的滚动条设置滚动位置的最小和最大值。如果将 *nMinPos*

和 *nMaxPos* 都设置为零，则隐藏此标准滚动条。

当处理一个滚动条通知消息时，不要调用这个函数来隐藏一个滚动条。

如果在调用 SetScrollRange 之后，立即调用 SetScrollPos 成员函数，则在 SetScrollPos 中将 *bRedraw* 设置为 0，以禁止滚动条被重画两次。

由 *nMinPos* 和 *nMaxPos* 指定的间隔值必须不大于 32,767。一个滚动条控件的缺省范围是空（*nMinPos* 和 *nMaxPos* 都为 0）。

请参阅 CScrollBar::GetScrollPos, CScrollBar::SetScrollPos, CScrollBar::GetScrollRange, ::SetScrollRange

CScrollBar::ShowScrollBar

```
void ShowScrollBar( BOOL bShow = TRUE );
```

参数

bShow

指示是显示还是隐藏滚动条。如果这个参数是 TRUE**，则显示滚动条；否则隐藏滚动条。**

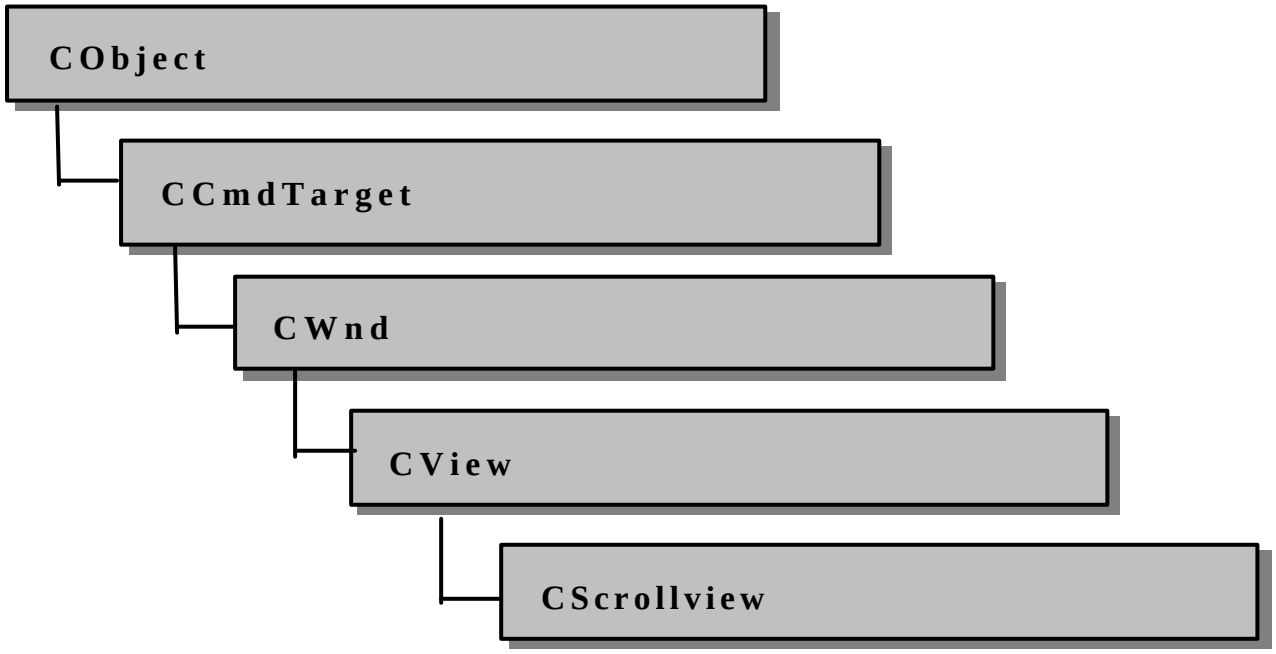
说明

此成员函数用来显示或隐藏一个滚动条。

当应用程序在处理一个滚动条通知消息时，不要调用这个函数来隐藏滚动条。

请参阅 CScrollBar::GetScrollPos, CScrollBar::GetScrollRange, CWnd::ScrollWindow, CScrollBar::SetScrollPos, CScrollBar::SetScrollRange

CScrollView



CScrollView 类是一个具有滚动性能的 CView。你可以在任何一个从 CView 派生的类中通过重载消息映射的 OnHScroll 和 OnVScroll 成员函数来自己处理标

准的滚动。但是 CScrollView 给它的 CView 性能添加了下列特征：

- 它管理窗口和视口的大小和映射模式。
- 它响应滚动条消息自动滚动。
- 它响应来自键盘，非滚动鼠标或 IntelliMouse 滚轮的消息自动滚动。

你可以通过重载消息映射的 OnMouseWheel 和 OnRegisteredMouseWheel 成员函数来自己处理鼠标轮的滚动。就象对于 CScrollView 一样，这些成员函数支持为轮旋转消息 WM_MOUSEWHEEL 推荐的行为。

要利用自动滚动的优点，可以从 CScrollView 派生你的视类来代替 CView 派生。当视第一次被创建时，如果你希望根据文档的大小来计算可滚动视的大小，可以从重载的 CView::OnInitialUpdate 或 CView::OnUpdate 中调用成员函数 SetScrollSizes。（你必须编写自己的代码来查询文档的大小。参见在“ Visual C++ 教程 ”中的“ 增强视 ”可以获得有关的示例。）

调用成员函数 SetScrollSize 可以设置视的映射模式，可滚动视的总尺寸，以及可在水平或垂直方向上滚动的量。所有的尺寸都是以逻辑单位给出的。视的逻辑大小通常是由保存在文档中的数据计算出来的。但在某些情况下，你可能希望为视指定一个固定的尺寸。有关这两种方法的例子，参见 CScrollView::SetScrollSizes。

你应该以逻辑单位指定在水平或垂直方向上滚动的量。缺省情况下，如果用户在滚动框之外单击了滚动条的条身，则 CScrollView 滚动一“页”。如果用户单击了滚动条两端的滚动箭头，则 CScrollView 滚动一“行”。缺省情况下，一页是指整个视的大小的 1/10；一行是指页大小的 1/10。你可以通过传递给 SetScrollSize 成员函数定制的尺寸来代替这些缺省值。例如，你可以将水平大小设置为宽度总尺寸的一部分，并将垂直大小设置为当前字体的一行的高度。

除了滚动之外，CScrollView 还可自动将视的大小按比例变换成当前窗口的大

小。在这种方式下，视没有滚动条，这个逻辑视被扩大或缩小以精确符合窗口的客户区。要使用这种“ scale-to-fit ”的功能，可以调用 CScrollView::SetScaleToFitSize。（调用 SetScaleToFitSize 或 SetScrollSizes，但不要都调用。）

在调用你的派生类的 OnDraw 成员函数之前，CScrollView 自动调整传递给 OnDraw 的 CPaintDC 设备环境对象的视口原点。

为了为滚动窗口调整视口原点，CScrollView 重载 CView::OnPrepareDC。这种调整对于 CScrollView 传递给 OnDraw 的 CPaintDC 设备环境来说是自动的，但你必须为自己所使用的其它设备环境（例如 CClientDC）自己调用 CScrollView::OnPrepareDC。你可以重载 CScrollView::OnPrepareDC 来设置画笔，背景颜色，和其它绘画属性，但是要调用基类来进行比例变换。

滚动条可以显示在与一个视相关的三个位置，如下所示：

- 可以使用 WS_HSCROLL 和 WS_VSCROLL 风格为视设置标准的窗口风格滚动条。
- 也可以将滚动条加在包含视的框架上，在这种情况下，框架将来自框架窗口的 WM_HSCROLL 和 WM_VSCROLL 消息向前传递给当前的活动视。
- 框架也可以将来自 CSplitterWnd 分隔器控件的滚动信息转发给当前活动的分隔格（一个视）。当在一个 CSplitterWnd 中放置了一个可共享的滚动条时，CScrollView 对象将使用这个可共享的滚动条而不再去创建它自己的滚动条。

```
#include <afxwin.h>
```

请参阅 CView, CSplitterWnd

CScrollView 类成员

Operations	
FillOutsideRect	填充视滚动区域之外的其它区域
GetDeviceScrollPositio n	获取以设备单位表示的当前滚动位置
GetDeviceScrollSizes	获取可滚动视的用设备单位表示的当前映射模式，总尺寸，以及行和页的大小
GetScrollPosition	获取用逻辑单位表示的当前滚动位置
GetTotalSize	获取用逻辑单位表示的滚动视的总尺寸
ResizeParentToFit	使视的大小可以支配它的框架的大小
ScrollToPosition	将视滚动到用逻辑单位指定的给定点
SetScaleToFitSize	将视设置为“scale-to-fit”模式
SetScrollSizes	设置滚动视的映射模式，总尺寸，以及水平和垂直滚动量

Construction	
CscrollView	构造一个 CScrollView 对象

成员函数

CScrollView::CScrollView

CScrollView( );

说明

此成员函数用来构造一个对象。在此滚动视可用之前你必须调用 SetScrollSize **或** SetScaleToFitSize。

请参阅 CScrollView::SetScrollSizes, CScrollView::SetScaleToFitSize

CScrollView::FillOutsideRect

void FillOutsideRect(CDC* *pDC*, CBrush* *pBrush*);

参数

pDC

将在其中完成填充操作的设备环境。

pBrush

将被填充的区域的画刷。

说明

调用 FillOutsideRect 来填充视的滚动区域之外的区域。在滚动视的 OnEraseBkgnd 处理函数中使用 FillOutsideRect 可以避免连续的背景重画。

示例

```
BOOL CScaleView::OnEraseBkgnd( CDC* pDC )
{
    CBrush br( GetSysColor( COLOR_WINDOW ) );
```

```
    FillOutsideRect( pDC, &br );
    return TRUE;                        // 擦除
}
```

请参阅 CWnd::OnEraseBkgnd

CScrollView::GetDeviceScrollPosition

CPoint GetDeviceScrollPosition( ) const;

返回值

返回一个 CPoint **对象，包含了滚动框的水平和垂直位置（以设备单位表示）。**

说明

当你需要滚动条中的滚动框的当前水平和垂直位置时，可以调用 GetDeviceScrollPosition。**此函数返回的坐标对应于已滚动的视的左上角在文档**

中的位置。在将鼠标设备的位置偏移到滚动视的设备位置时，这一点是很有用的。

GetDeviceScrollPosition 的返回值是以设备单位表示的。如果你需要的是逻辑单位值，你可以使用 GetScrollPosition 来代替。

请参阅 CScrollView::GetScrollPosition

CScrollView::GetDeviceScrollSizes

void GetDeviceScrollSizes(int& *nMapMode*, SIZE& *sizeTotal*, SIZE& *sizePage*,
SIZE& *sizeLine*) const;

参数

nMapMode

返回这个视的当前映射模式。参见 SetScrollSizes 可以获得可能取值的列

表。

sizeTotal

返回这个滚动视的当前以设备单位表示的总尺寸。

sizePage

返回在响应滚动条的条体中发生的鼠标单击时，水平和垂直方向上的当前滚动量。cx 成员中存放水平量；cy 成员中存放垂直量。

sizeLine

返回当响应鼠标在某个滚动箭头上单击时水平和垂直方向上的当前滚动量。cx 成员中存放水平量；cy 成员中存放垂直量。

说明

GetDeviceScrollSizes 用来获取当前的映射模式，总尺寸，以及这个可滚动视的行和页的大小。各个尺寸都是以设备单位表示的。这个成员函数很少被调用。

请参阅 CScrollView::SetScrollSizes, CScrollView::GetTotalSize

CScrollView::GetScrollPosition

CPoint GetScrollPosition( ) const;

返回值

返回一个 CPoint **对象，包含滚动框的水平和垂直位置（以逻辑单位表示）。**

说明

当你需要滚动条中的滚动框的当前水平和垂直位置时可以调用 GetScrollPosition。**这个坐标对是对应于已滚动的视的左上角在文档中的位置。**

GetScrollPosition **的返回值是以逻辑单位表示的。如果你需要的是以设备单位表示的值，你可以调用** GetDeviceScrollPosition **来代替。**

请参阅 CScrollView::GetDeviceScrollPosition

CScrollView::GetTotalSize

CSize GetTotalSize( ) const;

返回值

返回以逻辑单位表示的滚动视的总尺寸。水平尺寸被保存在 CSize **返回值的** cx **成员中。垂直尺寸被保存在** cy **成员中。**

说明

调用 GetTotalSize **来获取滚动视的当前水平和垂直尺寸。**

请参阅 CScrollView::GetDeviceScrollSizes, CScrollView::SetScrollSizes

CScrollView::ResizeParentToFit

void ResizeParentToFit(BOOL *bShrinkOnly* = TRUE);

参数

bShrinkOnly

要实现的尺寸调整操作。缺省值是 TRUE**，表示要根据需要缩小框架窗口。对大的视或小的框架窗口，滚动条仍然是可见的。该值为** FALSE**，表示视总是要精确地调整框架窗口。这样做可能会有危险，这是因为框架窗口可能会变得太大以致于不能被放在多文档界面（**MDI**）的框架窗口中或屏幕内。**

说明

调用 ResizeParentToFit **将使视的大小支配它的框架窗口的大小。仅对** MDI **子窗**

口才推荐使用此函数。你应该在 CScrollView 派生类的 OnInitialUpdate 处理函数中使用 ResizeParentToFit 函数。关于这个成员函数的例子，参见 CScrollView::SetScrollSize。

ResizeParentToFit 假设视图窗口的尺寸已被设置好。如果 ResizeParentToFit 函数被调用时视图窗口的尺寸还没有被设置好，则屏幕上显示一条断言。为了保证不发生此类事件，应确保在调用 ResizeParentToFit 函数之前进行如下调用：

```
GetParentFrame()->RecalcLayout();
```

请参阅 CView::OnInitialUpdate, CScrollView::SetScrollSizes

CScrollView::ScrollToPosition

void ScrollToPosition(POINT *pt*);

参数

pt

用逻辑单位表示的要滚动到的点。x 成员必须是正值（大于或等于 0，小于视的总大小值）。当映射模式时 MM_TEXT 时，对 y 成员有同样的要求。当映射模式不是 MM_TEXT 时，y 成员是一个负值。

说明

调用 ScrollToPosition 来滚动到视中的一个给定点。视也将被滚动以使此点位于窗口的左上角。当视是“scale-to-fit”的视时，不能调用此函数。

请参阅 CScrollView::GetDeviceScrollPosition, CScrollView::SetScaleToFitSize, CScrollView::SetScrollSizes

CScrollView::SetScaleToFitSize

void SetScaleToFitSize(SIZE *sizeTotal*);

参数

sizeTotal

指出这个视将被比例变换至的水平和垂直尺寸。可滚动视的尺寸是以逻辑单位测量的。cx **成员中存放水平尺寸，**cy **成员中存放垂直尺寸。**cx **和** cy **都必须大于或等于** 0**。**

说明

当你希望将视口的尺寸自动按比例变换为当前窗口的尺寸时，可以调用 SetScaleToFitSize **函数。使用滚动条时，在任一时刻可能只能看到逻辑视的一部分。但是如果具有“**scale-to-fit**”的功能，则视没有滚动条，并且逻辑视被扩**

大或缩小以精确吻合窗口的客户区。当窗口被重新调整大小时，该视将根据窗口的尺寸以新的比例绘制其数据。

通常你可以在视的重载的 OnInitialUpdate 成员函数中调用 SetScaleToFitSize。如果你不需要自动按比例变换，调用 SetScrollSizes 成员函数来代替。

SetScaleToFitSize 可以用来实现一个“Zoom to fit”操作。使用 SetScrollSizes 来重新初始化滚动。

SetScaleToFitSize 假设视图窗口已经被设置好了。如果在调用 SetScaleToFitSize 时视图窗口的尺寸还没有被设置，你将会得到一个断言。为了保证不出现这种情况，在调用 SetScaleToFitSize 之前要进行以下调用：

```
GetParentFrame()->RecalcLayout();
```

请参阅 CScrollView::SetScrollSizes, CView::OnInitialUpdate

CScrollView::SetScrollSizes

void SetScrollSizes(int *nMapMode*, SIZE *sizeTotal*, const SIZE& *sizePage* = sizeDefault,
const SIZE& *sizeLine* = sizeDefault);

参数

nMapMode

要为这个视设置的映射模式。可能的取值包括：

Mapping Mode	Logical Unit	Positive Extends...	y-axis
MM_TEXT	1 **个像素**	**向下**	
Mapping Mode	Logical Unit	Positive Extends...	y-axis
MM_HIMETRIC	0.01 **毫米**	**向上**	
MM_TWIPS	1/1440 **英寸**	**向上**	
MM_HIENGLISH	0.001 **英寸**	**向上**	
MM_LOMETRIC	0.1 **毫米**	**向上**	
MM_LOENGLISH	0.01 **英寸**	**向上**	

所有的这些模式都是 Windows 定义的。两个标准的映射模式，MM_ISOTROPIC 和 MM_ANISOTROPIC，在 CScrollView 中没有使用。类库提供了 SetScaleToFitSize 成员函数来将视图的尺寸按比例变换为窗口的尺寸。

sizeTotal

滚动视的总尺寸。cx 成员包含了水平分量。cy 成员包含了垂直分量。这些尺寸是以逻辑单位表示的。cx 和 cy 都必须大于或等于 0。

sizePage

当鼠标在滚动条的条体上单击时，要在水平和垂直方向上滚动的量。cx 中存放水平滚动量，cy 成员中存放垂直滚动量。

sizeLine

当响应鼠标单击滚动条的滚动箭头时要在水平或垂直方向上滚动的量。cx 中存放水平滚动量，cy 成员中存放垂直滚动量。

说明

当视要被更新时，调用 SetScrollSizes。在你的重载的 OnUpdate 成员函数中调用这个函数来调整滚动特性，例如，当文档第一次被显示或是被改变大小时。

通常，通过调用你的派生文档类的一个文档成员函数，可能是 GetMyDocSize，你可以获得视的相关文档的尺寸信息。下面的代码给出了这种方法：

```
SetScrollSizes( nMapMode, GetDocument( )->GetMyDocSize( ) );
```

另外，你有时候也许需要设置一个固定的尺寸，就像下面的代码：

```
SetScrollSizes( nMapMode, CSize(100, 100) );
```

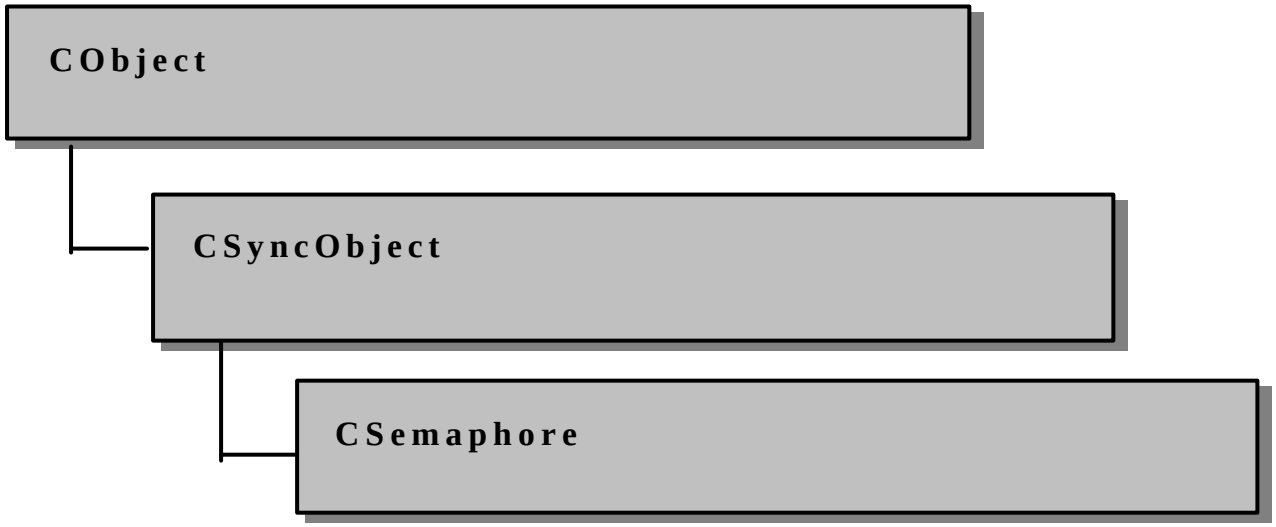
你必须将映射模式设置为除 MM_ISOTROPIC 或 MM_ANISOTROPIC 之外的任何 Windows 映射方式。如果你希望使用一种非限定的映射方式，可以调用 SetScaleToFitSize 成员函数来代替 SetScrollSizes。

示 例

```
void CScaleView::OnUpdate( )
{
   // ...
   // Implement a GetDocSize( ) member function in
   // your document class; it returns a CSize.
   SetScrollSizes( MM_LOENGLISH, GetDocument( )->GetDocSize( ) );
   ResizeParentToFit( );     // Default bShrinkOnly argument
   // ...
}
```

请 参 阅 CScrollView::SetScaleToFitSize, CScrollView::GetDeviceScrollSizes, CScrollView::GetTotalSize

CSemaphore



一个 CSemaphore 类对象代表一个“信号”---一个同步对象，它允许有限数目的线程在一个或多个进程中访问同一个资源。一个 CSemaphore 对象保持了对当前访问某一指定资源的线程的计数。

对于一个只能支持有限数目用户的共享资源来说，CSemaphore 是很有用的。

CSemaphore 对象的当前计数是还可以允许的其它用户的数目。当这个计数达到零的时候，所有对这个由 CSemaphore 对象控制的资源的访问尝试都将被插入到一个系统队列中等待，直到它们的时间用完或计数值不再为零。这个被控制的资源可以同时接受访问的最大用户数目是在 CSemaphore 对象的构造期间被指定的。

要使用一个 CSemaphore 对象，则在需要的时候构造这个对象。指定你想要等待的信号的名字，应用程序应该在最初就拥有它。然后你就可以在构造函数返回时访问这个信号。当你要访问这个被控制的资源时，调用 CSyncObject::Unlock。

使用 CSemaphore 对象的另一种方法是，将一个 CSemaphore 类型的变量添加到你想要控制的类中作为一个数据成员。在被控制对象的构造期间，调用 CSemaphore 数据成员的构造函数来指定访问计数的初始值，访问计数的最大

值，信号的名字（如果它要在整个进程中使用），以及需要的安全标志。

要访问由 CSemaphore 对象用这种方式控制的资源，首先要在你的资源的访问成员函数中创建一个 CSingleLock 类型或 CMultiLock 类型的变量。然后调用加锁对象的 Lock 成员函数（例如，CSingleLock::Lock）。这时，你的线程将达到对资源的访问，等待资源被释放并访问它，或者是在等待资源被释放的过程中超过了时间，对资源的访问失败。不管是哪一种情况，你的资源都是以一种线程安全（thread-safe）方式被访问的。要释放资源，可以使用加锁对象的 Unlock 成员函数（例如，CSingleLock::Unlock），或者是让加锁对象超越范围。

另外，你可以单独创建一个 CSemaphore 对象，并且在尝试访问被控制的资源之前显式地访问 CSemaphore 对象。这种方法虽然对阅读你的源代码的人来说更加清楚，但是却更易于出错。

```
#include <afxmt.h>
```

CSemaphore 类成员

Construction	
Csemaphore	构造一个 CSemaphore 对象

成员函数

CSemaphore:: CSemaphore

CSemaphore(LONG *lInitialCount* = 1, LONG *lMaxCount* = 1, LPCTSTR *pstrName* = NULL,
LPSECURITY_ATTRIBUTES *lpsaAttributes* = NULL);

参数

lInitialCount

信号的初始使用计数。必须是大于或等于 0，并且小于或等于 lMaxCount。

lMaxCount

信号的使用计数的最大值。必须大于 0。

pstrName

信号的名字。如果此信号将在整个进程中被访问，则必须提供这个名字。如果是 NULL，则对象将是没有名字的。如果这个名字与一个已经存在的信号的名字一样，则构造函数创建一个新的 CSemaphore 对象，此对象引用具有这个名字的对象。如果这个名字与一个已经存在的但不是一个信号的同步对象的名字一样，则构造函数会失败。

lpsaAttributes

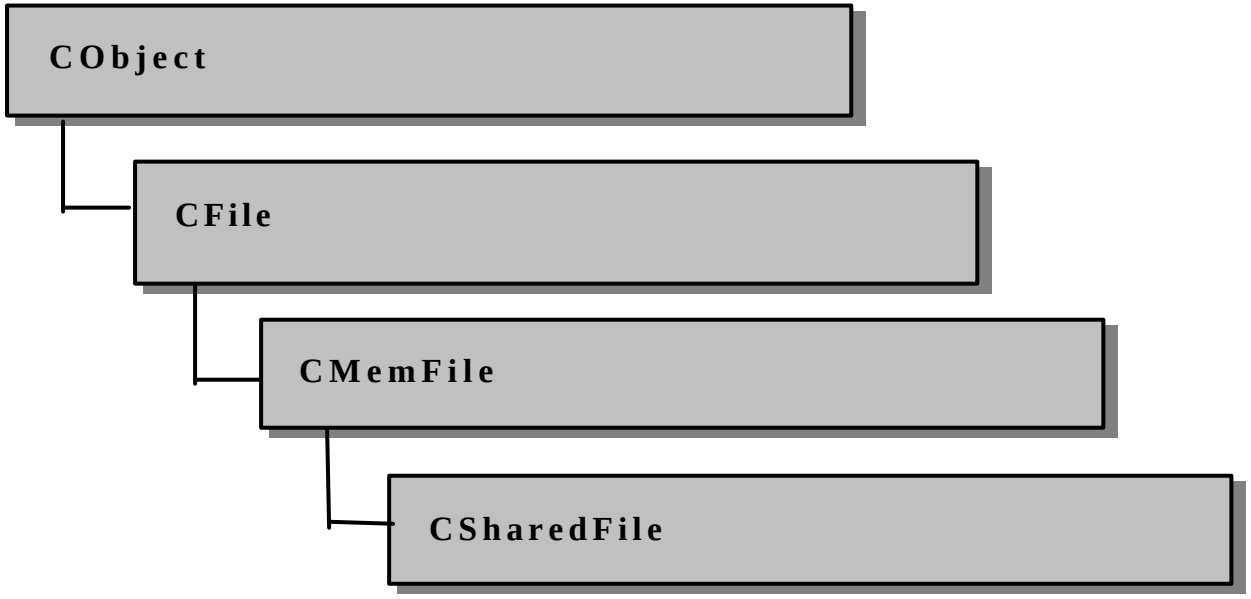
此信号对象的安全标志。有关这个结构的详细描述，参见“Win32 SDK 程序员参考”中的 SECURITY_ATTRIBUTES。

说明

此成员函数用来构造一个有名字或没有名字的 CSemaphore 对象。要访问或释放一个 CSemaphore 对象，可以创建一个 CMultiLock 或 CSingleLock 对象，并调用它们的 Lock 和 Unlock 函数。

请参阅 CMutex, CEvent, CMultiLock, CSingleLock

CSharedFile



CSharedFile 是 CMemFile 派生类，它支持共享的内存文件。内存文件的行为类似于磁盘文件，但是文件是保存在 RAM 中而不是磁盘上。在快速暂存或传输未加工字节时，或在独立进程之间使对象串行化时，一个内存文件是很有用的。

共享内存文件与其它的内存文件不同，共享内存文件的内存是由 GlobalAlloc Windows 函数分配的。CSharedFile 类将数据保存在一个全局分配的内存块（用 GlobalAlloc 创建）中，并且这个内存块可以使用 DDE，剪贴板，或其它 OLE/COM 统一数据传输操作（例如，使用 IDataObject）来共享。

GlobalAlloc 返回一个 HGLOBAL 句柄，而不是像 malloc 那样返回一个内存指针。在某些应用程序中需要 HGLOBAL 句柄。例如，要将数据放入剪贴板中，你就需要一个 HGLOBAL 句柄。

请注意，CSharedFile 不使用内存映射文件，并且数据不能在进程之间共享。

CSharedFile 对象可以自动分配它们自己的内存，或是你可以通过调用 CSharedFile:: SetHandle 来将你自己的内存块连接到 CSharedFile 对象上。不管是哪一种情况，如果 *nGrowBytes* 不为零，则用于增长内存文件的内存被自动按 *nGrowBytes-sized* 增量分配。。

更多的信息，参见“Microsoft Visual C++ 6.0参考库”中的“Microsoft Visual C++6.0运行库参考手册”卷中的“文件处理”。

#include <afxadv.h>

请参阅 CMemFile, GlobalAlloc, GlobalFree, GlobalRealloc

CSharedFile 类成员

Construction	
CsharedFile	构造一个 CSharedFile 对象
Operations	
Detach	关闭共享内存文件并返回它的内存块的句柄
SetHandle	将共享内存文件与一个内存块连接

成员函数

CSharedFile:: CSharedFile

CSharedFile(UINT *nAllocFlags* = GMEM_DDESHARE | GMEM_MOVEABLE,
UINT *nGrowBytes* = 4096);

参数

nAllocFlags

表示内存是如何分配的标志。参见 GlobalAlloc **可以获得一个有效标志值的列表。**

nGrowBytes

用字节表示的内存分配增量。

说明

此成员函数用来构造一个 CSharedFile 对象并为它分配内存。

请参阅 CSharedFile::Detach CSharedFile::SetHandle

CSharedFile::Detach

HGLOBAL Detach( );

返回值

返回包含此内存文件内容的内存块的句柄。

说明

此成员函数用来关闭内存文件并使它与内存块分离。你可以通过使用 Detach 返回的句柄来调用 SetHandle 来重新打开它。

请参阅 CSharedFile::CSharedFile, CSharedFile::SetHandle

CSharedFile::SetHandle

void SetHandle(HGLOBAL *hGlobalMemory*, BOOL *bAllowGrow* = TRUE);

参数

hGlobalMemory

与 CSharedFile **连接的全局内存的句柄。**

bAllowGrow

指示内存块是否被允许增长。

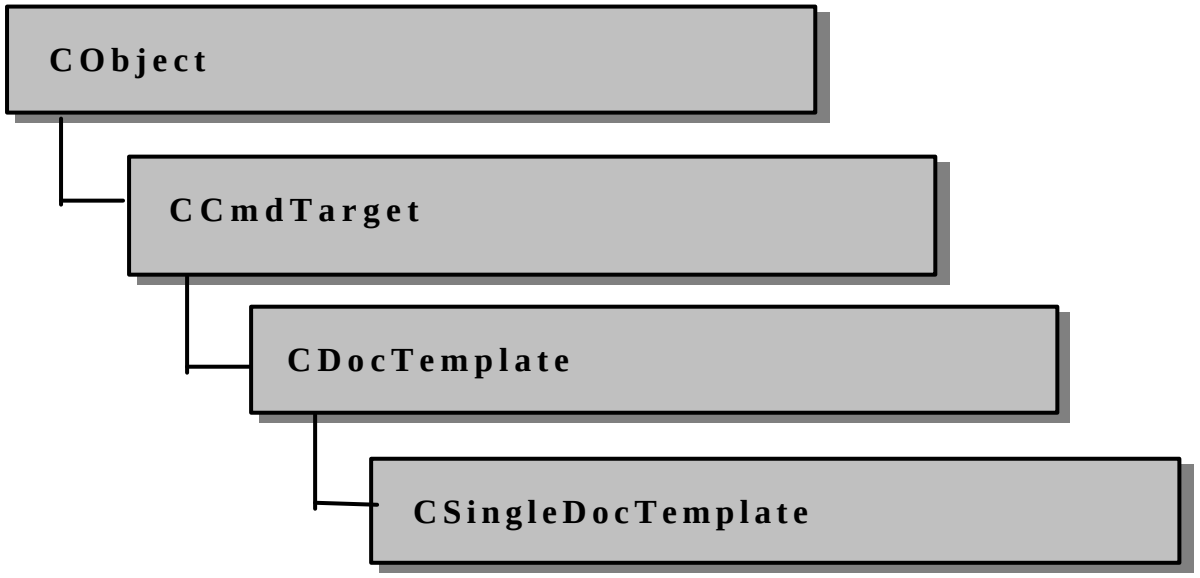
说明

此成员函数用来将一个全局内存块连接到 CSharedFile **对象上。如果** *bAllowGrow* **是非零值，则内存块按需要增长，例如，如果要写入此内存文件的字节数大于**

分配的内存块，则可以增长内存块。

请参阅 CSharedFile::CSharedFile, CSharedFile::Detach

CSingleDocTemplate



CSingleDocTemplate 类定义了一个文档模板用于实现单文档界面（SDI）。一个 SDI 应用程序使用主框架窗口来显示一个文档；一次只能打开一个文档。

一个文档模板定义了类的三个类型之间的关系：

- 一个从 CDocument 派生而来的文档类。
- 一个视类，用来显示来自上面提到的文档类的数据。你可以从 CView，CScrollView，CFormView，或 CEditView 派生这个类。（你也可以直接使用 CEditView。）
- 一个框架窗口类，用来包容视。对于一个 SDI 文档模板，你可以从 CFrameWnd 派生这个类；如果你不需要定制主框架窗口的行为，你可以直接使用 CFrameWnd，而不派生你自己的类。

一个 SDI 应用程序通常支持一种类型的文档，因此它只有一个 CSingleDocTemplate 模板。一次只打开一个文档。

除了 CSingleDocTemplate 的构造函数，你不需要调用它的任何其它成员函数。框架在内部处理 CSingleDocTemplate 对象。

请参阅 CDocTemplate, CDocument, CFrameWnd, CMultiDocTemplate, CView, CWinApp

CSingleDocTemplate 类成员

Construction	
CsingleDocTemplate	构造一个 CSingleDocTemplate 对象

成员函数

CSingleDocTemplate:: CSingleDocTemplate

CSingleDocTemplate(UINT *nIDResource*, CRuntimeClass* *pDocClass*,
CRuntimeClass* *pFrameClass*, CRuntimeClass* *pViewClass*);

参数

nIDResource

指定文档类型使用的资源 ID。它可能包括菜单，图标，加速键和字符串资源。

字符串资源最多可以包含七个子串，子串由‘\n’字符（如果有一个子串没有被包括，则需要‘\n’字符作为一个占位符；尾部的‘\n’字符不是必须的）；这些子串描述了文档的类型。有关子串的信息，参见 CDocTemplate::GetDocString。这个字符串资源可以在应用程序的资源文件中找到。例如：

```
// MYCALC.RC
STRINGTABLE PRELOAD DISCARDABLE
BEGIN
   IDR_MAINFRAME "MyCalc Windows Application\nSheet\nWorksheet\n
      Worksheets (*.myc)\n.myc\nMyCalcSheet\n MyCalc Worksheet"
```

END

你可以使用字符串编辑器来编辑这个字符串；在字符串编辑器中整个字符串是作为一个单一的项显示的，而不是作为七个分离的项。

有关这些资源类型的更多信息，参见子串编辑器（在“Visual C++用户指南”中被描述）。

pDocClass

指向一个文档类的 CRuntimeClass 对象。这个类是一个 CDocument 派生类，是你定义来代表你的文档的。

pFrameClass

指向框架窗口类的 CRuntimeClass 对象。这个类可以是一个 CFrameWnd 派生类，或者如果你想要你的主框架窗口具有缺省的行为，它可以是一个 CFrameWnd 自身，

pViewClass

指向一个视类的 CRuntimeClass 对象。这个类是一个 CView 派生类，是你定义来显示你的文档的。

说明

此成员函数用来构造一个 CSingleDocTemplate 对象。动态分配一个 CSingleDocTemplate 对象，并将它从你的应用程序类的 InitInstance 成员函数传递给 CWinApp::AddDoc- Template。

示例

```
// example for CSingleDocTemplate::CSingleDocTemplate
BOOL CMyApp::InitInstance()
{
    // ...
    // 建立应用程序支持的文档类
    AddDocTemplate( new CSingleDocTemplate( IDR_MAINFRAME,
```

```
        RUNTIME_CLASS( CSheetDoc ),
        RUNTIME_CLASS( CFrameWnd ),
        RUNTIME_CLASS( CSheetView ) ) );

    // ...
}
```

请 参 阅 CDocTemplate::GetDocString, CWinApp::AddDocTemplate, CWinApp::InitInstance, CRuntimeClass, RUNTIME_CLASS

CSingleLock

CSingleLock **没有基类。**

一个 CSingleLock **类对象代表一种访问控制机制，这种机制用于控制在一个多线程程序中对一个资源的访问。为了使用同步类** CSemaphore，CMutex，

CCriticalSection，和 CEvent，

你必须创建一个 CSingleLock 或 CMultiLock 对象来等待和释放这个同步对象。当你只需要每次等待一个对象时，可以使用 CSingleLock。当在一个特别的时候你可以使用多个对象时，可以使用 CMultiLock。

要使用一个 CSingleLock 对象，在被控制资源的类中的一个成员函数内部调用 CSingleLock 的构造函数。然后调用 ISLock 成员函数来确定这个资源是否可用。如果资源是可用的，则继续该成员函数的其余部分。如果资源不能使用，可以在一个指定的时间内等待资源被释放，或者是返回失败。在使用完资源后，如果 CSingleLock 对象要被再次使用,可以调用 Unlock 函数,或者销毁 CSingleLock 对象。

CSingleLock 对象需要有一个从 CSyncObject 派生的对象存在。这通常是一个被控制资源的类的数据成员。

#include <afxmt.h>

请参阅 CMultiLock

CSingleLock 类成员

Construction	
CsingleLock	构造一个 CSingleLock 对象

Methods	
IsLocked	确定此对象是否被加锁
Lock	等待一个同步对象
Unlock	释放一个同步对象

成员函数

CSingleLock:: CSingleLock

CSingleLock(CSyncObject* *pObject*, BOOL *bInitialLock* = FALSE);

参数

pObject

指向要被访问的同步对象。不能是 NULL。

bInitialLock

指示是否要在最初尝试访问所提供的对象。

说明

此成员函数用来构造一个 CSingleLock 对象。通常是从被控制资源的一个访问

成员函数中来调用这个函数。

CSingleLock::IsLocked

BOOL IsLocked( );

返回值

如果对象被加锁则返回非零值；否则返回 0。

说明

此成员函数用来确定与 CSingleLock 对象相关的对象是否没有发信号（不能使用）。

CSingleLock::Lock

BOOL Lock(DWORD *dwTimeOut* = INFINITE);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

dwTimeOut

指定等待要被利用的同步对象的时间数量。如果是 INFINITE，**则** Lock **等待直到该对象在返回之前可用。**

说明

此成员函数用来获取对由同步对象控制的资源的访问，这个访问要提供给 CSingleLock **构造函数。如果同步对象是可用的，**Lock **将成功返回，而且线程拥有了该对象。如果此同步对象是不可用的，则** Lock **将等待此同步对象在** *dwTimeOut* **参数指定的时间内变为可用。如果此同步对象在指定的时间内没有**

变为可用的，则 Lock 返回失败。

CSingleLock::Unlock

BOOL Unlock( );
BOOL Unlock(LONG lCount, LPLONG *lPrevCount* = NULL);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

LCount

要释放的访问数目。必须要大于 0。如果指定的数量要导致对象的计数超过它的最大值，则计数不改变，并且函数返回 FALSE。

LPrevCount

指向一个用来接收同步对象的先前计数的变量。如果是 NULL，则不返回先前的计数。

说明

此成员函数用来释放由 CSingleLock 拥有的同步对象。由 CSingleLock 的析构函数类调用这个函数。

如果你需要释放一个信号的多于一个的访问计数，可以使用 Unlock 的第二种形式，并指定要释放的访问数目。

CSize

CSize 类与 Windows 中表示相对坐标或位置的的 SIZE 结构类似。

注意 这个类是从 SIZE 结构派生而来的。这意味着在需要一个 SIZE 参数的调用中你可以传递一个 CSize，并且 SIZE 结构的数据成员也是 CSize 中可以访问的数据成员。

SIZE（和 CSize）的 cx 和 cy 成员是公有成员。另外，CSize 实现了用来处理 SIZE 结构的成员函数。

#include <afxwin.h>

请参阅 CRect，CPoint

CSize 类成员

Construction	
Csize	构造一个 CSize 对象

Operators	
Operator ==	检查 CSize 是否等于一个尺寸
Operator !=	检查 CSize 是否不等于一个尺寸
Operator +=	将 CSize 与一个尺寸相加
Operator -=	从 CSize 中减去一个尺寸

Operators Returning CSize Values	
Operator +	将两个尺寸相加
Operator -	将两个尺寸相减

成员函数

CSize:: CSize

```
CSize( );
CSize( int initCX, int initCY );
CSize( SIZE initSize );
```

CSize(POINT *initPt*);
CSize(DWORD *dwSize*);

参数

initCX

设置 CSize **的** cx **成员。**

initCY

设置 CSize **的** cy **成员。**

initSize

用来初始化 CSize **的** SIZE **结构或** CSize **对象。**

initPt

用来初始化 CSize **的** POINT **结构或** CPoint **对象。**

dwSize

用来初始化 CSize **的** DWORD**。双字中的低位字是** cx **成员，高位字是** cy **成员。**

说明

此成员函数用来构造一个 CSize **对象。如果没有给出参数，则不初始化** cx **和** cy **成员。**

操作符

CSize::operator ==

BOOL operator ==(SIZE *size*) const;

说明

此操作符用来检查两个尺寸是否相等。如果相等则返回非零值，否则返回 0**。**

请参阅 CSize::operator !=

CSize::operator !=

BOOL operator !=(SIZE *size*) const;

说明

此操作符用来检查两个尺寸是否不相等。如果不相等则返回非零值，否则返回 0。

请参阅 CSize::operator ==

CSize::operator +=

void operator +=(SIZE size);

说明

此操作符将 CSize **与一个尺寸相加。**

请参阅 CSize::operator +

CSize::operator -=

void operator -= (SIZE *size*);

说明

此操作符从 CSize **中减去一个尺寸。**

请参阅 CSize::operator -

CSize::operator +

CSize operator +(SIZE *size*) const;
CPoint operator +(POINT *point*) const;
CRect operator +(const RECT* *lpRect*) const;

说明

这些操作符将此 CSize **值与参数值相加。参见下面有关每一个操作符的描述：**

- operator + (*size*) **此操作将两个** CSize **值相加。**
- operator + (*point*) **此操作将一个** POINT **（或** CPoint**）值偏移（移动）**CSize **值。此** CSize **值的** cx **和** cy **成员被增加到** POINT **值的** x **和** y **成员中。它与需要一个** SIZE **参数的** CPoint::operator + **版本是类似的。**
- operator + (*lpRect*) **此操作将一个** RECT **（或** CRect**）值偏移（移动）**CSize **值。此** CSize **值的** cx **和** cy **成员被增加到** RECT **值的** left**,**top**,**right**,和** bottom **数据成员中。它与需要一个** SIZE **参数的** CRect::operator + **版本是类似的。**

请参阅 CPoint::operator +, CRect::operator +

CSize::operator -

CSize operator –(SIZE *size*) const;
CPoint operator –(POINT *point*) const;
CRect operator –(const RECT* *lpRect*) const;
CSize operator –( ) const;

说明

这些操作的前三个将此 CSize **值从参数值中减去。第四个操作是一个一元的负号，用来改变此** CSize **值的符号。参见下面对每一个操作的描述：**

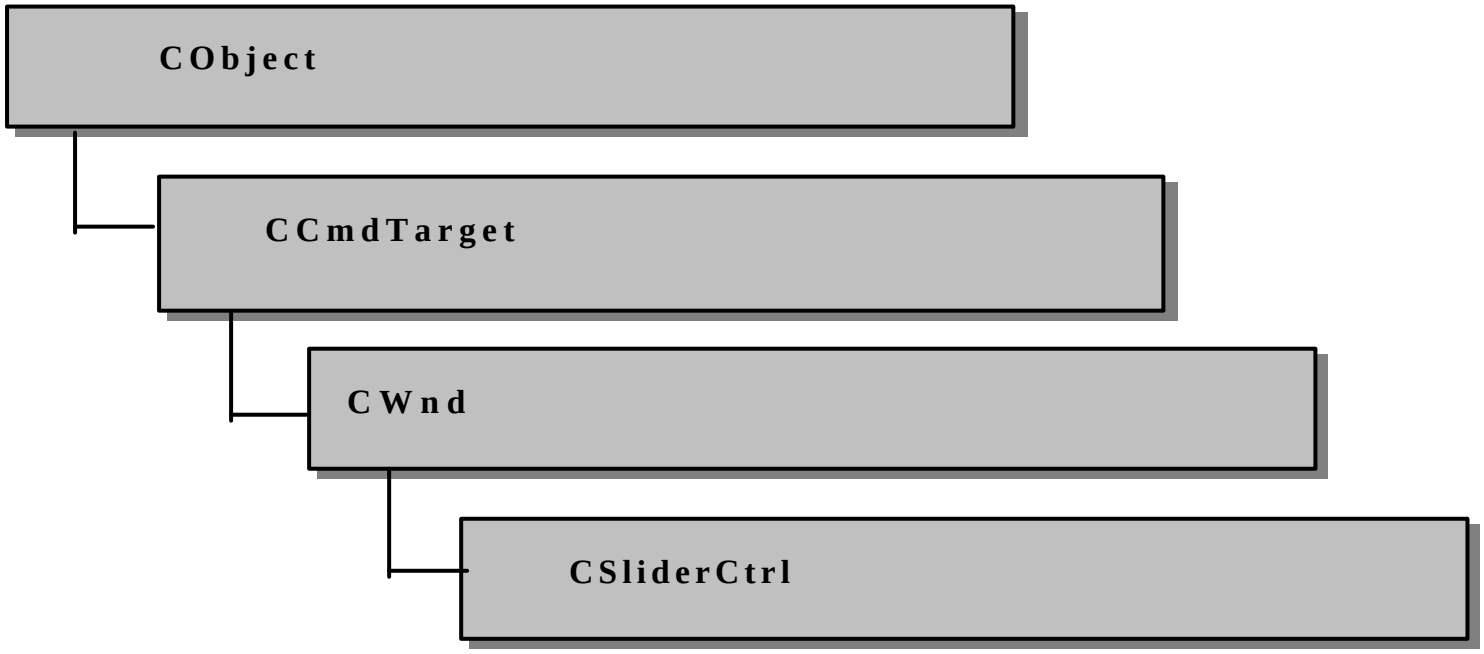
- operator –(*size*) **此操作将两个** CSize **值相减。**
- operator –(*point*) **此操作将一个** POINT **（或** CPoint**）值偏移（移动）** CSize **值的负值。此** CSize **值的** cx **和** cy **成员被从** POINT **值的** x **和** y **成员中减去。它与需要一个** SIZE **参数的** CPoint::operator -**版本是类似的。**
- operator –(*lpRect*) **此操作将一个** RECT **（或** CRect**）值偏移（移动）** CSize

值的负值。此 CSize 值的 cx 和 cy 成员被从 RECT 值的 left，top，right 和 bottom 数据成员中减去。它与需要一个 SIZE 参数的 CRect::operator -版本是类似的。

- operator –( ) 此操作符返回 CSize 值的负值。

请参阅 CPoint::operator -, CRect::operator –

CSliderCtrl



一个“滑动块控件”（也称为一个跟踪器）是一个包含一个滑动块和可选的刻度线的窗口。当用户用鼠标或方向键移动滑动块时，该控件发送通知消息来表

明这些改变。

当你想要用户选择不连续的值或是某一范围内的连续值的集合时，滑动块控件是很有用的。例如，你可以让用户通过移动滑动块到一个给定的刻度线来设置键盘的重复速度。

CSliderCtrl 类提供了 Windows 通用滑动块控件的功能。这个控件（也就是 CSliderCtrl 类）只对在 Windows95 和 Windows NT 3.51 或更高版本下运行的程序是可用的。

滑动块按你在创建它时指定的增量来移动。例如，如果你指定此滑动块的范围为五，则滑动块只能有六个位置：在滑动块控件最左边的一个位置和另外五个在此范围内每隔一个增量的位置。通常，这些位置的每一个由一个刻度线来标识。

通过使用 CSliderCtrl 的构造函数和 Create 成员函数来创建一个滑动块。一旦你已经创建了一个滑动块控件，你就可以使用 CSliderCtrl 中的成员函数来改变它的许多属性。这些你可以做的改变包括设置滑动块的最小和最大位置，绘制刻度线，设置一个选择范围，以及响应该滑动块。

#include <afxcmn.h>

请参阅 CProgressCtrl

CSliderCtrl 类成员

Construction	
CsliderCtrl	构造一个 CSliderCtrl 对象
Create	创建一个滑动块控件并将它与一个 CSliderCtrl 对象连接

Attributes	
GetLineSize	获取一个滑动块控件的行大小
SetLineSize	设置一个滑动块控件的行大小
GetPageSize	获取一个滑动块控件的页大小
SetPageSize	设置一个滑动块控件的页大小
GetRangeMax	获取一个滑动块的位置的最大值
GetRangeMin	获取一个滑动块的位置的最小值
GetRange	获取一个滑动块的位置的最大值和最小值
SetRangeMin	设置一个滑动块的位置的最小值
SetRangeMax	设置一个滑动块的位置的最大值
SetRange	设置一个滑动块的位置的最小值和最大值
GetSelection	获取当前选择的范围
SetSelection	设置当前选择的范围
GetChannelRect	获取滑动块控件的通道的尺寸
GetThumbRect	获取滑动块控件的拇指的尺寸
GetPos	获取滑动块的当前位置
SetPos	设置滑动块的当前位置
GetNumTics	获取一个滑动块控件中的刻度线的数目
GetTicArray	获取一个滑动块控件的刻度线位置的数组
GetTic	获取指定刻度线的位置

续表

GetTicPos	获取指定刻度线的以客户坐标表示的位置
SetTic	设置指定刻度线的位置
SetTicFreq	设置对每一个滑动块控件的增量，刻度线的频率
GetBuddy	在一个指定位置获取一个滑动块控件的伙伴窗口句柄
SetBuddy	为一个滑动块控件分配一个伙伴窗口
GetToolTips	获取分配给一个滑动块控件的工具提示（如果有）句柄
SetToolTips	将一个工具提示赋给一个滑动块控件
SetTipSide	定位跟踪器控件使用的工具提示
Operations	
ClearSel	清除在一个滑动块控件中的当前位置
VerifyPos	检验滑动块控件的位置是否在最小值和最大值之间
ClearTics	将当前刻度线从滑动块控件中移走

成员函数

CSliderCtrl::ClearSel

void ClearSel(BOOL *bRedraw* = FALSE);

参数

bRedraw

重画标志。如果这个参数是 TRUE**，则在选择被清除后重画滑动块；否则不重画滑动块。**

说明

此成员函数用来清除滑动块中的当前位置。

请参阅 CSliderCtrl::GetSelection, CSliderCtrl::SetSelection

CSliderCtrl::ClearTics

void ClearTics(BOOL bRedraw = FALSE);

参数

bRedraw

重画标志。如果这个参数是 TRUE**，则在选择被清除后重画滑动块；否则不重画滑动块。**

说明

此成员函数用来从一个滑动块控件中删除当前的刻度线。

请参阅 CSliderCtrl::GetTicArray, CSliderCtrl::GetTic, CSliderCtrl::GetNumTics

CSliderCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT

nID);

返回值

如果初始化成功则返回非零值；否则返回 0。

参数

dwStyle

指定滑动块控件的风格。滑动块控件风格的任意组合都适用于这个控件。

rect

指定滑动块控件的大小和位置。它可以是一个 CRect **对象或一个** RECT **结构。**

pParentWnd

指定此滑动块控件的父窗口，通常是一个 CDialog。**它必须不是** NULL。

nID

指定此滑动块控件的 ID。

说明

构造一个 CSliderCtrl 控件可以分两步。首先是调用构造函数，然后调用 Create，该函数创建滑动块控件并将它与此 CSliderCtrl 对象连接。

滑动块控件可以是垂直或水平方向的。它们可以在任何一边或两边具有刻度线，或者是两边都没有。它们也可以用来指定某个范围的连续值。这些属性通过使用滑动块风格来控件，这些风格是在你创建此滑动块控件的时候指定的：

- TBS_HORZ 滑动块为水平方向。这个方向是缺省方向。
- TBS_VERT 滑动块为垂直方向。如果你没有指定方向，滑动块被认为是水平方向。
- TBS_AUTOTICKS 创建的滑动块在其取值范围内对每一个增量都有一个

刻度线。当应用程序调用 SetRange 成员函数时这些刻度线被自动添加。如果你使用了这个风格，你就不能再使用 SetTic 和 SetTicFreq 成员函数来指定刻度线的位置。可以使用 ClearTics 成员函数来代替。

- TBS_NOTICKS 创建一个不显示刻度线的滑动块控件。
- TBS_BOTTOM 在一个水平滑动块的底部显示刻度线。可以与 TBS_TOP 一起使用，表示在滑动块的两边都显示刻度线。
- TBS_TOP 在一个水平滑动块的顶部显示刻度线。可以与 TBS_BOTTOM 一起使用，表示在滑动块的两边都显示刻度线。
- TBS_RIGHT 在一个垂直滑动块的右边显示刻度线。可以与 TBS_LEFT 一起使用，表示在滑动块的两边都显示刻度线。
- TBS_LEFT 在一个垂直滑动块的左边显示刻度线。可以与 TBS_RIGHT 一起使用，表示在滑动块的两边都显示刻度线。

- TBS_BOTH　在任意方向的滑动块的两边都显示刻度线。
- TBS_ENABLESELRANGE　显示一个选择范围。当一个滑动块控件使用了这个风格时，在选择范围的开始和结束位置的刻度线被显示为三角形（而不是垂直的破折号），而且选择范围被用高亮色显示。例如，在一个简单的日程安排应用程序中，选择范围也许是很有用的。用户可以对应于一天的小时选择一个刻度线范围来标识一个预定的会议时间。

请参阅　CSliderCtrl::CSliderCtrl

CSliderCtrl::CSliderCtrl

CSliderCtrl( );

说明

此成员函数用来构造一个 CSliderCtrl **对象。**

请参阅 CSliderCtrl::Create

CSliderCtrl::GetBuddy

CWnd* GetBuddy(BOOL *fLocation* = TRUE) const;

返回值

返回一个指向 CWnd **对象的指针，该对象是在由** fLocation **指定的位置的一个伙伴窗口。如果在那个位置不存在伙伴窗口，则返回** NULL。

参数

fLocation

这个值是用来指示在相关位置上将获取哪一个伙伴窗口的句柄。它可以是下列值之一：

- TRUE　获取在滑动块左边的伙伴窗口的句柄。如果该滑动块控件使用了 TBS_VERT 风格，则将获取滑动块上面的伙伴窗口。
- FALSE　获取在滑动块右边的伙伴窗口的句柄。如果该滑动块控件使用了 TBS_VERT 风格，则将获取滑动块下面的伙伴窗口。

说明

此成员函数用来实现 Win32 消息 TBM_GETBUDDY 的行为，就象在“Platform SDK”中所描述的一样。

有关滑动块控件风格的描述，参见“Platform SDK”中的“跟踪器控件风格”。

请参阅　CSliderCtrl::SetBuddy

CSliderCtrl::GetChannelRect

void GetChannelRect(LPRECT *lprc*) const;

参数

lprc

一个指向 CRect **对象的指针，该对象包含了当函数返回时通道的边界矩形。**

说明

此成员函数用来获取一个滑动块控件的通道的边界矩形的大小和位置。通道是这样一个区域，滑动块在它上面移动，并且当选择了一个范围后，它用高亮色显示。

请参阅 CSliderCtrl::GetThumbRect

CSliderCtrl::GetLineSize

int GetLineSize( ) const;

返回值

返回此滑动块控件的一行的大小。

说明

此成员函数用来获取一个滑动块控件的行的大小。这个行的大小表明对于 TB_LINEUP **和** TB_LINEDOWN **通知，滑动块移动多少。行的大小的缺省设置是** 1。

请参阅 CSliderCtrl::SetLineSize, CSliderCtrl::GetPageSize

CSliderCtrl::GetNumTics

UINT GetNumTics( ) const;

返回值

返回在滑动块中的刻度线的数目。

说明

此成员函数用来获取一个滑动块中的刻度线。

请参阅 SliderCtrl::GetTicArray, CSliderCtrl::GetTic, CSliderCtrl::GetTicPos, CSliderCtrl::SetTicFreq, CSliderCtrl::ClearTics

CSliderCtrl::GetPageSize

int GetPageSize( ) const;

返回值

返回此滑动块控件的一页的大小。

说明

此成员函数用来获取一个滑动块控件中的一页的大小。页的大小表明在响应 TB_PAGEUP 和 TB_PAGEDOWN 通知时滑动块会移动多少。

请参阅 CSliderCtrl::GetLineSize, CSliderCtrl::SetPageSize

CSliderCtrl::GetPos

int GetPos( ) const;

返回值

返回当前的位置。

说明

此成员函数用来获取一个滑动块控件中的滑动块的当前位置。

请参阅 CSliderCtrl::SetPos, CSliderCtrl::GetTicPos

CSliderCtrl::GetRange

void GetRange(int& *nMin*, int& *nMax*) const;

参数

nMin

一个用来接收最小位置的整数的引用。

nMax

一个用来接收最大位置的整数的引用。

说明

此成员函数用来获取在一个滑动块控件中的滑动块的最大和最小位置。这个函数将值拷贝到由 *nMin* **和** *nMax* **引用的整数中。**

请参阅 SliderCtrl::GetRangeMin, CSliderCtrl::GetRangeMax, CSliderCtrl::SetRange

CSliderCtrl::GetRangeMax

int GetRangeMax( ) const;

返回值

返回控件的最大值。

说 明

此 成 员 函 数 用 来 获 取 一 个 滑 动 块 控 件 中 的 滑 动 块 的 最 大 位 置 。

请 参 阅 CSliderCtrl::GetRangeMin, CSliderCtrl::GetRange, CSliderCtrl::SetRange

CSliderCtrl::GetRangeMin

int GetRangeMin( ) const;

返 回 值

返 回 控 件 的 最 小 位 置 。

说 明

此 成 员 函 数 用 来 获 取 一 个 滑 动 块 控 件 中 的 滑 动 块 的 最 小 位 置 。

请参阅 CSliderCtrl::GetRange, CSliderCtrl::GetRangeMax, CSliderCtrl::SetRange

CSliderCtrl::GetSelection

void GetSelection(int& *nMin*, int& *nMax*) const;

参数

nMin

一个用来接收当前选择的开始位置的整数的引用。

nMax

一个用来接收当前选择的结束位置的整数的引用。

说明

此成员函数用来获取一个滑动块控件中的当前选择的开始和结束位置。

请参阅 CSliderCtrl::SetSelection, CSliderCtrl::ClearSel

CSliderCtrl::GetThumbRect

void GetThumbRect(LPRECT lprc) const;

参数

lprc

一个指向 CRect **对象的指针，该对象包含了当函数返回时此滑动块的边界矩形。**

说明

此成员函数用来获取一个滑动块控件中的滑动块（拇指）的边界矩形的大小和位置。

请参阅 CSliderCtrl::GetChannelRect

CSliderCtrl::GetTic

int GetTic(int nTic) const;

返回值

返回指定刻度线的位置。如果 nTic **没有指定一个有效索引，则返回**-1**。**

参数

nTic

一个刻度线的从零开始的索引。

说明

此成员函数用来获取一个滑动块控件中的刻度线的位置。

请参阅 CSliderCtrl::SetTic, CSliderCtrl::GetTicArray, CSliderCtrl::GetTicPos,

CSliderCtrl::SetTicFreq, CSliderCtrl::ClearTics

CSliderCtrl::GetTicArray

DWORD* GetTicArray( ) const;

返回值

返回包含此滑动块控件的刻度线位置的数组的地址。

说明

此成员函数用来获取包含一个滑动块控件的刻度线的数组的地址。

请参阅 CSliderCtrl::SetTic, CSliderCtrl::GetTic, CSliderCtrl::GetTicPos, CSliderCtrl::SetTicFreq, CSliderCtrl::ClearTics

CSliderCtrl::GetTicPos

int GetTicPos(int nTic) const;

返回值

返回指定刻度线的用客户坐标表示的物理位置。如果 *nTic* **没有指定一个有效的索引则返回**-1。

参数

nTic

一个刻度线的从零开始的索引。

说明

此成员函数用来获取一个滑动块控件中的一个刻度线的当前物理位置。

请参阅 CSliderCtrl::SetTic, CSliderCtrl::GetTic, CSliderCtrl::SetTicFreq, CSliderCtrl::ClearTics

CSliderCtrl::GetToolTips

```
CToolTipCtrl* GetToolTips( ) const;
```

返回值

返回一个指向 CToolTipCtrl **对象的指针。如果没有使用工具提示则返回** NULL。**如果此滑动块控件没有使用** TBS_TOOLTIPS **风格，则返回值是** NULL。

说明

此成员函数用来实现 Win32 **消息** TBM_GETTOOLTIPS **的行为，就象在“** Platform SDK **”中描述的那样。**

注意这个成员函数返回一个 CToolTipCtrl **对象。而不是一个控件句柄。**

有关滑动块控件风格的描述，参见“ Platform SDK **”中的“跟踪器风格”。**

请参阅 CSliderCtrl::SetToolTips

CSliderCtrl::SetBuddy

CWnd* SetBuddy (CWnd* *pWndBuddy*, BOOL *fLocation* = TRUE);

返回值

返回一个指向 CWnd **对象的指针，该对象是先前在那个位置上分配给此滑动块控件的。**

参数

pWndBuddy

一个指向 CWnd 对象的指针，该对象将被设置为滑动块控件的伙伴。

fLocation

是用来指定显示伙伴窗口的位置的值。这个值可以是下列值之一：

- TRUE 如果滑动块控件使用 TBS-HORZ 风格，则伙伴将显示在滑动块的左边。如果滑动块控件使用 TBS-VERT 风格，伙伴就出现在滑动块控件的上方。
- FALSE 如果滑动块控件使用 TBS-HORZ 风格，则伙伴就出现在滑动块的右边。如果滑动块控件使用 TBS-VERT 风格，伙伴就出现在滑动块控件的下方。

说明

此成员函数用来实现 Win32 消息 TBM_SETBUDDY 的行为，就象在“ Platform

SDK”中描述的那样。

注意，这个成员函数在它的返回值和参数中都使用了指向 CWnd 对象的指针，而不是窗口句柄。

有关滑动块控件风格的描述，参见“Platform SDK”中的“跟踪器风格”。

请参阅 CSliderCtrl::GetBuddy

CSliderCtrl::SetLineSize

int SetLineSize(int *nSize*);

返回值

返回先前的行大小。

参数

nSize

滑动块控件的新的行大小。

说明

此成员函数用来设置一个滑动块控件的行的大小。这个行大小表示在响应 TB_LINEUP **和** TV_LINEDOWN **通知时，滑动块移动多少。**

请参阅 CSliderCtrl::GetLineSize, CSliderCtrl::SetPageSize

CSliderCtrl::SetPageSize

int SetPageSize(int *nSize*);

返回值

返回先前的页大小。

参数

nSize

滑动块控件的新的页大小。

说明

此成员函数用来设置一个滑动块控件的页的大小。这个页大小表示在响应 TB_PAGEUP **和** TB_PAGEDOWN **通知时，滑动块移动多少。**

请参阅 CSliderCtrl::GetPageSize, CSliderCtrl::GetLineSize

CSliderCtrl::SetPos

void SetPos(int *nPos*);

参数

nPos

指定新的滑动块位置。

说明

此成员函数用来设置一个滑动块控件中的滑动块的当前位置。

请参阅 CSliderCtrl::GetPos, CSliderCtrl::SetTic, CSliderCtrl::VerifyPos

CSliderCtrl::SetRange

void SetRange(int *nMin*, int *nMax*, BOOL *bRedraw* = FALSE);

参数

bMin

滑动块的最小位置。

nMax

滑动块的最大位置。

bRedraw

重画标志。如果这个参数是 TRUE**，则在范围被重新设置之后滑动块被重画；否则不重画滑动块。**

说明

此成员函数用来设置一个滑动块控件的滑动块的范围（位置的最小值和最大值）。

请参阅 CSliderCtrl::GetRange, CSliderCtrl::SetRangeMax,

CSliderCtrl::SetRangeMin

CSliderCtrl::SetRangeMax

void SetRangeMax(int *nMax*, BOOL *bRedraw* = FALSE);

参数

nMax

滑动块的最大位置。

bRedraw

重画标志。如果这个参数是 TRUE**，则在范围被设置之后，重画滑动块；否则不重画滑动块。**

说明

此成员函数用来设置一个滑动块控件中的滑动块的最大位置。

请 参 阅 CSliderCtrl::SetRange, CSliderCtrl::GetRangeMax, CSliderCtrl::SetRangeMin

CSliderCtrl::SetRangeMin

void SetRangeMin(int *nMin*, BOOL *bRedraw* = FALSE);

参数

nMin

滑动块的最小位置。

bRedraw

重画标志。如果这个参数是 TRUE**，则在范围被设置之后，重画滑动块；否则不重画滑动块。**

说明

此成员函数用来设置一个滑动块控件中的滑动块的最大位置。

请参阅 CSliderCtrl::SetRange, CSliderCtrl::GetRangeMin, CSliderCtrl::SetRangeMax

CSliderCtrl::SetSelection

void SetSelection(int *nMin*, int *nMax*);

参数

nMin

滑动块的开始位置。

nMax

滑动块的结束位置。

说明

此成员函数用来设置一个滑动块控件中当前选择的开始和结束位置。

请参阅 CSliderCtrl::GetSelection, CSliderCtrl::ClearSel

CSliderCtrl::SetTic

BOOL SetTic(int *nTic*);

返回值

如果设置了刻度线则返回非零值；否则返回 0。

参数

nTic

刻度线的位置。这个参数必须指定一个正值。

说 明

此成员函数用来设置一个滑动块控件中的一个刻度线的位置。

请参阅 CSliderCtrl::GetTic, CSliderCtrl::GetTicArray, CSliderCtrl::GetTicPos, CSliderCtrl::SetTicFreq, CSliderCtrl::ClearTics

CSliderCtrl::SetTicFreq

void SetTicFreq(int *nFreq*);

参 数

nFreq

刻度线的频率。

说明

此成员函数用来设置显示在一个滑动块中的刻度线的的频率。例如，如果频率被设置为 2**，则在滑动块的范围中每两个增量显示一个刻度线。缺省的频率设置是** 1**（即，范围中每一个增量都有一个与之对应的刻度线）。**

要使用这个函数，你必须创建具有 TBS_AUTOTICKS **风格的控件。更多的信息，参见** CSliderCtrl::Create**。**

请参阅 CSliderCtrl::Create, CSliderCtrl::SetTic, CSliderCtrl::GetTicArray

CSliderCtrl::SetTipSide

int SetTipSide (int *nLocation*);

返回值

返回一个代表工具提示控件的先前位置的值。这个返回值等于 *nLocation* 的可能取值之一。

参数

nLocation

是代表要显示工具提示控件的位置的值。其可能取值的列表，参见“Platform SDK”中所描述的 Win32 消息 TBM_SETTIPSIDE。

说明

此成员函数用来实现“Platform SDK”中所描述的 Win32 消息 TBM_SETTIPSIDE 的行为。

滑动块控件使用 TBS_TOOLTIPS 风格来显示工具提示。有关滑动块控件风格的描述，参见“Platform SDK”中的“跟踪器控件风格”。

CSliderCtrl::SetToolTips

void SetToolTips(CToolTipCtrl* *pWndTip*);

参数

pWndTip

是一个指向 CToolTipCtrl 对象的指针，该对象包含了这个滑动块控件的工具提示。

说明

此成员函数用来实现“Platform SDK”中所描述的 Win32 消息 TBM_SETOOLTIPS 的行为。

当创建了一个具有 TBS_TOOLTIPS 风格的滑动块控件时，它就创建一个缺省的显示在滑动块的旁边的工具提示控件，它显示滑动块的当前位置。有关滑动块控件风格的描述，参见“Platform SDK”中的“跟踪器控件风格”。

请参阅 CSliderCtrl::GetToolTips

CSliderCtrl::VerifyPos

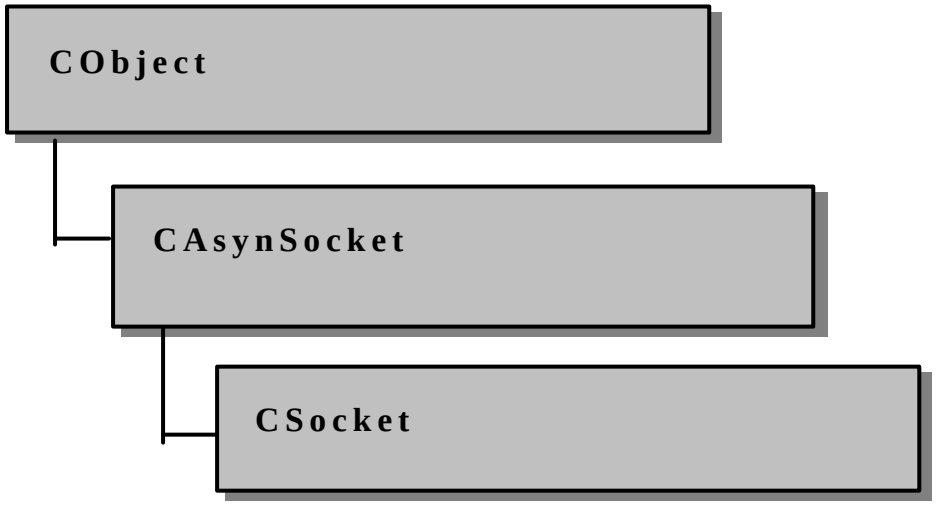
void VerifyPos( );

说明

此成员函数用来检验一个滑动块控件中的滑动块的当前位置是否在最小和最大值之间。

请参阅 CSliderCtrl::GetRange, CSliderCtrl::SetPos, CSliderCtrl::GetTicPos

CSocket



CSocket 类是从 CAsyncsocket 派生而来的，它继承了 CAsyncsocket 对 Windows Sockets API 的封装。与 CAsyncsocket 对象相比，CSocket 对象代表了 Windows Sockets API 的更高一级的抽象化。CSocket 与类 CSocketFile 和 CArchive 一起来管理对数据的发送和接收。

一个 CSocket 对象也支持阻塞，这对于 CArchive 的同步操作来说是必要的。块

操作函数，比如 Receive，Send，ReceiveFrom，SendTo，和 Accept（都是从 CAsyncsocket 继承来的），都不返回一个 CSocket 对象中的 WSAEWOULDBLOCK 错误。取而代之，这些函数等待，直到操作完成。另外，当这些函数中的某一个是阻塞的时，如果调用了 CancelBlockingCall，则原来的调用将因为 WSAEINTR 错误而终止。

要使用一个 CSocket 对象，调用构造函数，然后调用 Create 来创建基础 插槽句柄（插槽类型）。Create 的缺省参数创建一个插槽，但是如果你不是用一个 CArchive 对象来使用这个插槽，则你可以指定一个参数来创建一个数据包 插槽来代替，或者是结合一个指定的端口来创建一个服务器插槽。在客户方使用 Connect，则服务器方使用 Accept 来与一个客户插槽连接。然后再创建一个 CSocketFile 对象，并在 CSocketFile 的构造函数中将它连接到 CSocket 对象上。再接着，创建一个 CArchive 对象用来发送数据，一个用来接收数据（如果需要），

然后在 CArchive 构造函数中将它们与 CSocketFile 对象连接。当通讯完成后，销毁 CArchive，CSocketFile，CSocket 对象。

更多的信息，参见 Win32 SDK 文档中的“Windows 插槽 2 概述”和“Windows 插槽设计思考”。

#include <afxsock.h>

请参阅 CAsyncsocket, CSocketFile

CSocket 类成员

Construction	
Csocket	构造一个 CSocket 对象
Create	创建一个插槽

Attributes	
IsBlocking	确定一个阻塞调用是否在进行中
FromHandle	返回一个指向 CSocket 对象的指针，给出一个插槽句柄
Attach	将一个插槽句柄与一个 CSocket 对象连接
Operations	
CancelBlockingCall	取消一个当前在进行中的阻塞调用
Overridables	
OnMessagePending	当等待完成一个阻塞调用时调用此函数来处理悬而未决的消息

成员函数

CSocket::Attach

BOOL Attach(SOCKET hsocket);

返回值

如果函数成功则返回非零值。

参数

hsocket

包含一个插槽句柄。

说明

此成员函数用来将一个 hsocket **句柄与一个** CSocket **对象连接。这个插槽句柄被保存在对象的** m_socket **数据成员中。**

更多的信息，参见 Win32 SDK **文档中的“** Windows Sockets **设计思考”。**

请参阅 CAsyncsocket::Attach

CSocket::CancelBlockingCall

void CancelBlockingCall( );

说明

此成员函数用来取消一个当前在进行中的阻塞调用。这个函数取消该插槽的任何未完的阻塞操作。原来的阻塞调用一有可能就终止，并给出 WSAEINTR **错误。**

在一次阻塞的 Connect **操作中，**Windows **插槽实现一有可能就终止这个阻塞的调用，但是这对于释放** socket **资源来说是不可能的，要直到连接完成（然后被重置）或时间到时，资源才被释放。只有在应用程序立即尝试打开一个新的插槽（如果没有插槽可利用），或连接到相同的同级者时，这一点才是需要注意的。**

除了 Accept，取消任何操作都会使该插槽处于一种不确定的状态。如果一个应用程序取消了在一个插槽上的阻塞操作，则应用程序要能够在该插槽上显示，唯一能依赖的操作就是调用 Closet，虽然其它的操作可能会对某些 Windows 插槽起作用。如果你希望你的应用程序具有最大的可移植性，则你必须小心不要在一次取消后依赖于执行某些操作。

更多的信息，参见 Win32 SDK 文档中的“ Windows Sockets 设计思考 ”。

请参阅 CAsyncsocket::Accept, CAsyncsocket::Close, CAsyncsocket::Connect, CSocket::IsBlocking, ::WSASetBlockingHook

CSocket::Create

```
BOOL Create( UINT nSocketPort = 0, int nSocketType = SOCK_STREAM,
    LPCTSTR lpszSocketAddress = NULL );
```

返回值

如果函数成功则返回非零值；否则返回 0，**并通过调用** GetLastError **可以获得特定的错误代码。**

参数

nSocketPort

一个要被此插槽使用的特别的端口，如果你希望 MFC **来选择一个端口，则这个值是** 0。

nSocketType

是 SOCK_STREAM **或** SOCK_DGRAM。

lpszSocketAddress

一个指向字符串的指针，该字符串包含了被连接插槽的网络地址。一个带点的数字，如“ 128.56.22.8 **”。**

说明

在构造一个插槽对象之后，调用 Create 成员函数来创建 Windows 插槽并连接它。然后 Create 调用 Bind 来将此插槽与指定的地址结合。下面就是所支持的插槽类型：

- SOCK_STREAM　提供连续的，可靠的，两种方法（two-way）的，基于连接的字节流。使用 Internet 地址家族的传输控制协议（TCP）。
- SOCK_DGRAM　支持无连接的数据包，它有具有固定（通常较小）最大长度的不可靠的缓冲。使用 Internet 地址家族的 User Datagram Protocol（UDP）。要使用这个选项，你必须不用一个 CArchive 对象来使用插槽。

  注意　Accept 成员函数将一个新的，空的 CSocket 对象的引用作为它的参数。你必须在调用 Accept 之前构造这个对象。记住，如果这个插槽对象超

出了范围，则连接关闭。不要对这个新的 sock 对象调用 Create 函数。

有关流和数据包 插槽的更多信息，参见 Win32 SDK 文档中的“ Windows Sockets 设计思考”。

请参阅 CAsyncsocket::Create, CAsyncsocket::Bind

CSocket::CSocket

CSocket( );

说明

此成员函数用来构造一个 CSocket 对象。在构造之后，你必须调用 Create 成员函数。

更多的信息，参见 Win32 SDK 文档中的“ Windows Sockets 设计思考”。

请参阅 CAsyncSocket::Create

CSocket::FromHandle

static CSocket* PASCAL FromHandle(SOCKET *hSocket*);

返回值

返回一个指向 CSocket **对象的指针。如果没有与** *hSocket* **连接的** CSocket **对象，则返回** NULL。

参数

hSocket

包含一个插槽句柄。

说明

此成员函数返回一个指向 CSocket **对象的指针。当给予一个插槽句柄，如果没有一个** CSocket **对象连接到这个句柄，则此成员函数返回** NULL**，并且不会创建一个临时对象。**

更多的信息，参见 Win32 SDK **文档中的“** Windows Sockets **设计思考”。**

请参阅 CAsyncsocket::FromHandle

CSocket::IsBlocking

BOOL IsBlocking( );

返回值

如果该插槽是阻塞的，则返回非零值。否则返回 0**。**

说明

此成员函数用来确定一个阻塞的调用是否正在进行中。

更多的信息，参见 Win32 SDK **文档中的“** Windows Sockets **设计思考”。**

请参阅 CSocket::CancelBlockingCall

CSocket::OnMessagePending

virtual BOOL OnMessagePending( );

返回值

如果消息被处理了则返回非零值；否则返回 0**。**

说明

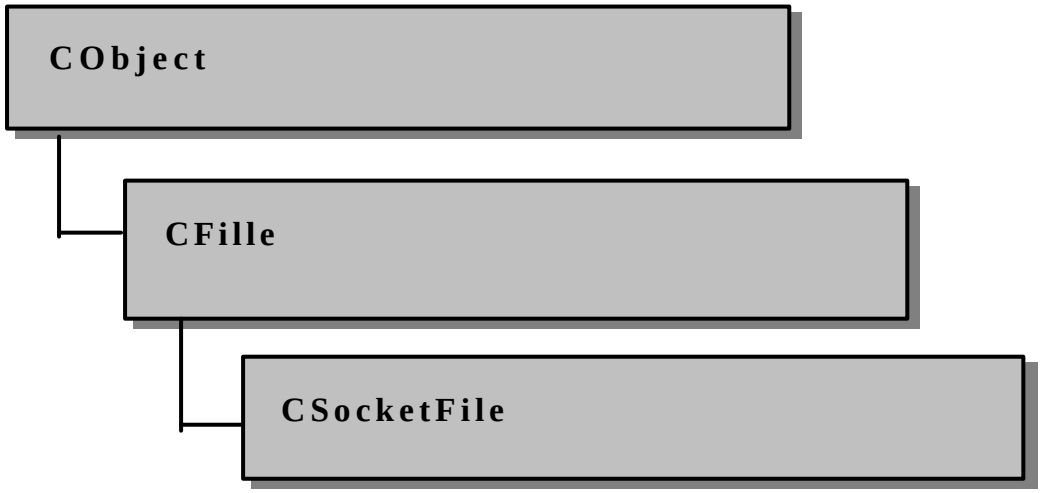
重载这个函数来查找来自 Windows **的特别消息，并在你的插槽中对它们作出响应。这是一个高级的可重载函数。**

当插槽在转发 Windows **消息时，框架调用** OnMessagePending **来给你一个机会去处理你的应用程序感兴趣的消息。**

更多的信息，参见 Win32 SDK **文档中的“** Windows Sockets **设计思考”。**

请参阅 CSocket::CancelBlockingCall, CSocket::IsBlocking

CSocketFile



一个 CSocketFile 对象是一个用来通过 Windows Sockets 在网络中发送和接收数据的 CFile 对象。为了这个目的，你可以将 CSocketFile 对象与一个 CSocket 对象连接。你也可以---并且通常是这样做---将 CSocketFile 对象与一个 CArchive 对象连接，以使用 MFC 系列来简化发送和接收数据。

对于连续（发送）的数据，你可以把它们插入到档案中，该档案调用 CSocketFile 成员函数来把数据写到 CSocket 对象中。对于不连续（接收）的数据，你从档案中提取它们。这导致该档案调用 CSocketFile 成员函数来从 CSocket 对象中读取数据。

> **提示** 除了象这儿所描述的一样使用 CSocketFile，你也可以将它作为一个标准文件对象，就象你使用它的基类 CFile 一样。你可以将 CSocketFile 与其它任何基于档案的 MFC 系列函数一起使用。因为 CSocketFile 并不支持 CFile 的所有性能，某些缺省的 MFC 串行化函数与 CSocketFile 是不兼容的。这一点对于 CEditView 类来说完全正确。你不要尝试使用 CEditView::SerializeRaw 并通过一个与 CSocketFile 对象连接的 CArchive 对象来串行化 CEditView 数据；而应当使用 CEditView::Serialize。SerializeRaw 函数希望文件对象具有一些

CSocketFile 所不拥有的函数，比如说 Seek。

更多的信息，参见 Win32 SDK 文档中的“ Windows Sockets 2 概述 ”和“ Windows Sockets 设计思考 ”。

#include <afxsock.h>

请参阅 CAsyncsocket, CSocket

CSocketFile 类成员

Construction	
CsocketFile	构造一个 CSocketFile 对象

成员函数

CSocketFile:: CSocketFile

CSocketFile(CSocket* *pSocket*, BOOL *bArchiveCompatible* = TRUE);

参数

pSocket

连接到 CSocketFile **对象的插槽。**

bArchiveCompatible

指示该文件对象是否与一个 CArchive **对象一起使用。只有当你希望在单机方式下来使用这个** CSocketFile **对象时，才传递** FALSE**。这种方式就象单机的** CFile **对象，具有一定的限制。这个标志将改变与** CSocketFile **对象连接的** CArchive **对象管理读缓冲区的方式。**

说明

此成员函数用来构造一个 CSocketFile 对象。当此对象超出范围或被删除时，它的析构函数将使它自己从插槽对象上分离。

> **注意** 一个 CSocketFile 对象也可以在没有 CArchive 对象的情况下作为一个（受限制）的文件来使用。缺省的，CSocketFile 构造函数的 *bArchiveCompatible* 参数是 TRUE。这表明此文件对象是与一个档案一起使用的。要在没有档案的情况下使用这个文件对象，给 *bArchiveCompatible* 参数传递 FALSE。

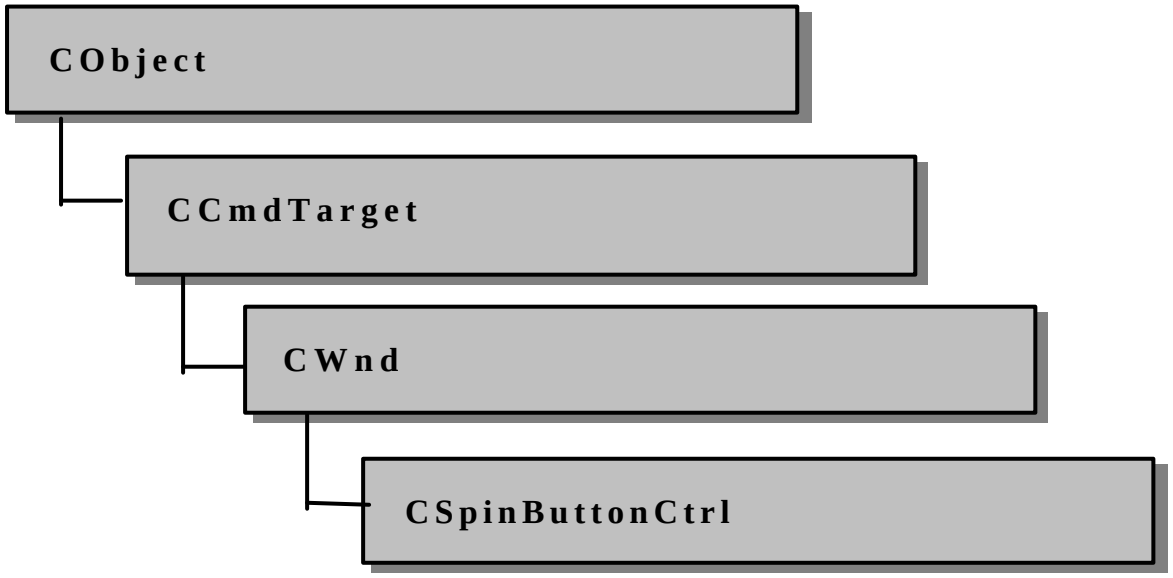
在“档案兼容”模式下，一个 CSocketFile 对象可以提供更好的表现，并减少出现“死锁”的危险。当两个发送和接收插槽相互等待时，或是同时等待一个公用的资源时，就会发生死锁。如果与 CSocketFile 对象一起工作的 CArchive

对象按它与 CFile 对象一起工作的方式来工作时，这种情况就会发生。在与 CFile 一起工作时，档案可以假定如果它接收到的字节少于它所请求的，则文件已经到达文件结尾了。

但是，在使用 CSocketFile 工作时，数据是基于消息的；缓冲区可以保存多条消息，因此收到比所请求的要少的字节数并不表示到达了文件结尾。在这种情况下，应用程序不会象使用 CFile 时那样阻塞，它可以继续从缓冲区中读取数据，直到缓冲区为空。在这种情况下，CArchive::IsBufferEmpty 用于监视档案的缓冲区的状态是很有用的。

请参阅 CFile::CFile, CFile::Read

CSpinButtonCtrl



一个“旋转按钮控件”（也称为上下控件）是一对箭头按钮，用户点击它们来增加或减小一个值，比如一个滚动位置或显示在相应控件中的一个数字。与一

个旋转按钮控件相联系的值被称为它的当前位置。一个旋转控件通常是与一个相伴的控件一起使用的，称为“伙伴窗口”。

CSpinButtonCtrl类提供了Windows通用旋转按钮控件的功能。这个控件（也就是CSpinButtonCtrl类）只对运行在Windows95和Windows NT3.51或更高版本下的程序来说是可用的。

对用户来说，一个旋转按钮控件和它的伙伴窗口看起来通常就象一个单一的控件。你可以指定一个旋转按钮控件自动将它自己定位在它的伙伴窗口的旁边，并且它自动将它的伙伴窗口的标题设置为它的当前位置。可以将一个旋转按钮控件与一个编辑控件一起使用，以提示用户进行数字输入。

点击向上箭头使当前位置向最大值方向移动，而点击向下箭头使当前位置向最小值的方向移动。缺省的，最小值是100，最大值是0。任何时候，最小值的设置都大于最大值（例如，当使用缺省值时），点击向上箭头减少位置值，而

点击向下箭头则增加它。

一个没有伙伴窗口的旋转按钮控件就象简化了的滚动条。例如，一个 tab 控件有时显示一个旋转按钮控件来使它的用户能够滚动其它的 tab 进入视。

有关使用 CSpinButtonCtrl 的更多信息，参见“ Visual C++程序员指南 ”中的“ 控件主题 ”和“ 使用 CSpinButtonCtrl ”。

#include <afxcmn.h>

请参阅 CSliderCtrl

CSpinButtonCtrl 类成员

Construction	
CspinButtonCtrl	构造一个 CSpinButtonCtrl 对象
Create	创建一个旋转按钮控件并将它连接到一个 CSpinButtonCtrl 对象

Attributes	
SetAccel	为一个旋转按钮控件设置加速
GetAccel	获取一个旋转按钮控件的加速信息
SetBase	为一个旋转按钮控件设置基数
GetBase	获取一个旋转按钮控件的当前基数
SetBuddy	为一个旋转按钮控件设置伙伴窗口
GetBuddy	获取指向当前伙伴窗口的指针
SetPos	设置控件的当前位置
GetPos	获取一个旋转按钮控件的当前位置
SetRange	设置一个旋转按钮控件的上限和下限（范围）
GetRange	获取一个旋转按钮控件的上限和下限（范围）
SetRange 32	设置旋转按钮控件的 32 位范围
GetRange 32	获取旋转按钮控件的 32 位范围

成员函数

CSpinButtonCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果初始化成功则返回非零值；否则返回 0。

参数

dwStyle

指定旋转按钮控件的风格。可以将旋转控件风格的任何组合应用到这个控件。

rect

指定旋转按钮控件的大小和位置。它可以是一个 CRect 对象或一个 RECT 结构。

pPrentWnd

一个指向旋转控件的父窗口的指针，该窗口通常是一个 CDialog。它必须不是 NULL。

nID

指定旋转控件的 ID。

说明

构造一个 CSpinButtonCtrl 对象要分两步。首先调用构造函数，然后调用 Create，该函数创建旋转按钮控件并将它与 CSpinButtonCtrl 对象连接。

可以指定下面的风格给旋转按钮控件：

- UDS_HORZ 控件的箭头指向左和右，而不是指向上和下。
- UDS_WRAP 如果控件的增加或减小超过了结尾或开始，使控件可以“环绕”。
- UDS_ARROWKEYS 当按下向上和向下键时，使控件可以增加或减小。
- UDS_SETBUDDYINT 当位置改变时，使控件设置伙伴窗口的文本（使用WM_SETTEXT消息）。文本是按十进制或十六进制格式化后的位置值。
- UDS_NOTHOUSANDS 不要在每隔三个十进制数字的地方加千分隔符。
- UDS_AUTOBUDDY 自动选择一个 Z-order 中的前一个窗口作为控件的伙伴窗口。
- UDS_ALIGNRIGHT 将旋转按钮窗口定位在伙伴窗口的右边。伙伴窗口的宽度被减小来适应此控件的宽度。
- UDS_ALIGNLEFT 将旋转按钮窗口定位在伙伴窗口的右边。伙伴窗口被

移动到右边，并且它的宽度被减小来适应此控件的宽度。

请参阅 CSpinButtonCtrl::CSpinButtonCtrl

CSpinButtonCtrl::CSpinButtonCtrl

CSpinButtonCtrl( );

说明

此成员函数用来构造一个 CSpinButtonCtrl **对象。**

请参阅 CSpinButtonCtrl::Create

CSpinButtonCtrl::GetAccel

UINT GetAccel(int *nAccel*, UDACCEL* *pAccel*) const;

返回值

返回所获取的加速器结构的数目。

参数

nAccel

由 pAccel **指定的数组中的元素数目。**

pAccel

指向一个 UDACCEL **结构数组的指针，该数组用来接收加速信息。**

说明

此成员函数用来获取一个旋转按钮控件的加速信息。

请参阅 CSpinButtonCtrl::SetAccel

CSpinButtonCtrl::GetBase

UINT GetBase( ) const;

返回值

返回当前的基数值。

说明

此成员函数用来获取一个旋转按钮控件的当前基数值。

请参阅 CSpinButtonCtrl::SetBase

CSpinButtonCtrl::GetBuddy

CWnd* GetBuddy( ) const;

返回值

返回一个指向当前伙伴窗口的指针。

说明

此成员函数用来获取一个指向当前伙伴窗口的指针。

请参阅 CSpinButtonCtrl::SetBuddy

CSpinButtonCtrl::GetPos

int GetPos( ) const;

返回值

返回值的低位字中是当前位置。如果发生了一个错误，则高位字是非零值。

说明

此成员函数用来获取一个旋转按钮控件的当前位置。控件根据伙伴窗口的标题来更新它的当前位置，获得要返回的值。如果没有伙伴窗口或者如果标题指示的是一个无效或超出范围的值，则返回一个错误信息。

请参阅 CSpinButtonCtrl::SetPos

CSpinButtonCtrl::GetRange

DWORD GetRange( ) const;
void GetRange(int &*lower*, int& *upper*) const;
void GetRange32(int &*lower*, int& *upper*) const;

返回值

第一个版本返回一个 32 **位的值，包含了上限和下限值。低位字是控件的上限，**

高位字是控件的下限。

参数

lower

对一个用来接收控件的下限的整数的引用。

upper

对一个用来接收控件的上限的整数的引用。

说明

此成员函数用来获取一个旋转按钮控件的上限和下限（范围）。

成员函数 GetRange32 获取此旋转按钮控件的用一个 32 位的整数表示的范围。

请参阅 CSpinButtonCtrl::SetRange

CSpinButtonCtrl::SetAccel

BOOL SetAccel(int *nAccel*, UDACCEL* *pAccel*);

返回值

如果成功则返回非零值；否则返回 0。

参数

nAccel

由 *pAccel* **指定的** UDACCEL **结构的数目。**

pAccel

指向一个 UDACCEL **结构数组的指针，该数组包含了加速信息。元素根据** nSec **成员按升序保存。**

说明

此成员函数用来设置一个旋转按钮控件的加速。

请参阅 CSpinButtonCtrl::GetAccel

CSpinButtonCtrl::SetBase

int SetBase(int *nBase*);

返回值

如果成功则返回先前的基数值，如果给出的是一个无效的基数则返回一个非零值。

参数

nBase

控件的新的基数。

说明

此成员函数用来设置一个旋转按钮控件的基数。这个基数值决定了伙伴窗口显示的数字是十进制的还是十六进制的。十六进制的数总是无符号的；十进制的数是有符号的。

请参阅 CSpinButtonCtrl::GetBase

CSpinButtonCtrl::SetBuddy

CWnd* SetBuddy(CWnd* *pWndBuddy*);

返回值

一个指向先前的伙伴窗口的指针。

参数

pWndBuddy

指向新的伙伴窗口的指针。

说明

此成员函数用来为一个旋转控件设置伙伴窗口。

请参阅 CSpinButtonCtrl::GetBuddy

CSpinButtonCtrl::SetPos

int SetPos(int *nPos*);

返回值

返回原来的位置。

参数

nPos

控件的新位置。这个值必须在控件的上限和下限指定的范围之内。

说明

此成员函数用来设置一个旋转按钮控件的当前位置。

请参阅 CSpinButtonCtrl::SetRange, CSpinButtonCtrl::GetPos

CSpinButtonCtrl::SetRange

void SetRange(int *nLower*, int *nUpper*);
void SetRange32(int *nLower*, int *nUpper*);

参数

nLower and nUpper

控件的上限和下限。任何一个界限值都不能大于 UD_MAXVAL 或小于 UD_ MINVAL。另外，两个界限值之间的差值必须不超过 UD_MAXVAL。

说明

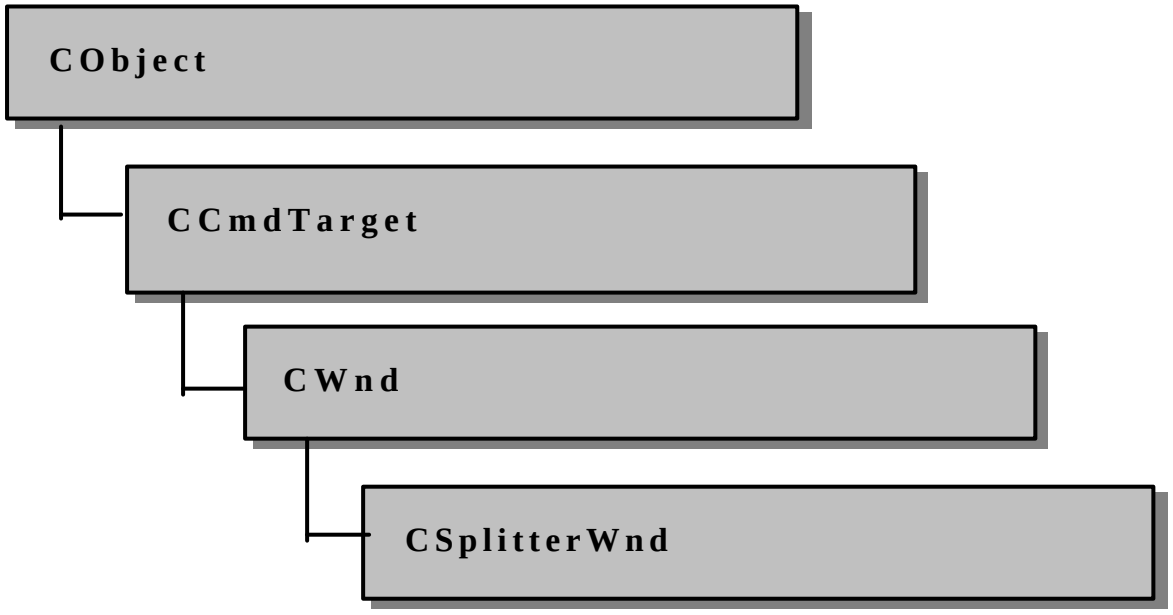
此成员函数用来设置一个旋转按钮控件的上限和下限（范围）。

成员函数 SetRange32 为此旋转按钮控件设置 32 位的范围。

注意　旋转按钮控件的缺省范围将最大值设置为零（0），而最小值设置为 100。由于最大值小于最小值，所以点击向上箭头将减小位置值，而点击向下箭头则增加位置值。使用 CSpinButtonCtrl::SetRange 来调整这些值。

请参阅　CSpinButtonCtrl::GetRange, CSpinButtonCtrl::GetPos, Using CSpinButtonCtrl

CSplitterWnd



CSplitterWnd 类提供一个分隔器窗口的功能，分隔器窗口是一个包含有多个窗格的窗口。窗格通常是应用程序特定的由 CView 派生的对象，但它也可以是具有适当子窗口 ID 的任何 CWnd 对象。

一个 CSplitterWnd 对象通常被嵌入 CFrameWnd 或 CMDIChildWnd 父对象。你

应按如下步骤创建一个 CSplitterWnd 对象：

1. 在父框架中嵌入一个 CSplitterWnd 成员变量。
2. 重载父框架的 CFrameWnd::OnCreateClient 成员函数。
3. 从重载的 OnCreateClient 函数中调用类 CSplitterWnd 的 Create 或 CreateStatic 成员函数。

调用 Create 成员函数可以创建一个动态的分隔器窗口。动态的分隔器窗口通常用于创建和滚动同一文档的多个窗格或视。框架将自动为该分隔器创建一个起始窗格；然后，当用户操纵该分隔器窗口的控件时，框架创建，调整大小，并且排列其它的窗格。

当你调用 Create 时，应指定一个最小行高度和最小列宽度，这两个最小值被用来确定窗格什么时候太小以致于不能被完全显示。在调用了 Create 之后，你可以通过调用 SetColumnInfo 和 SetRowInfo 来调整这些最小值。

你还可以使用 SetColumnInfo 和 SetRowInfo 成员函数来给一列设置一个“理想的”宽度，以及给一行设置一个“理想的”高度。当框架显示一个分隔器窗口时，首先显示父框架，然后才显示分隔器窗口。然后，框架从分隔器窗口的客户区左上角至右下角，根据窗格的理想尺寸以行或列来排放各个窗格。

动态分隔器窗口中的所有窗格都必须是同一个类的窗格。读者熟悉的支持动态分隔器窗口的应用程序包括 Microsoft Word 和 Microsoft Excel.

使用 CreateStatic 成员函数可以创建一个静态分隔器窗口。用户只能修改静态分隔器窗口中的窗格的尺寸，但是不能改变其中的窗格序号和次序。

在创建静态分隔器时必须专门创建该静态分隔器的所有窗格。你必须在父框架的 OnCreateClient 成员函数返回之前确保创建了所有窗格，否则，框架将不能正确显示该窗口。

CreateStatic 成员函数将使用为 0 的最小行高度和最小列宽度来自动初始化一个静态分隔器。在调用了 Create 之后，可以通过调用 SetColumnInfo 和 SetRowInfo 成员函数来调整这两个最小值。在调用 CreateStatic 之后，你也可以使用 SetColumnInfo 和 SetRowInfo 成员函数来指定你所希望的理想窗格值。

静态分隔器中的窗格通常属于不同的类。给予静态分隔器窗口的示例，参见图形编辑器和 Windows 文件管理器。分隔器窗口支持特殊的滚动条（除窗格可能带有的滚动框之外）。这些滚动条是 CSplitterWnd 对象的子对象并且被窗格共享。

当你创建这个分隔器窗口时，你创建了这些特殊的滚动条。例如，如果一个 CSplitterWnd 具有一行，两列，则 WS_VSCROLL 风格将显示两个窗格共享的垂直滚动条。当用户移动这个滚动条时，WM_VSCROLL 消息将被发送给两个窗格。当窗格设置滚动条位置时，这个共享滚动条将被设置。

有关动态分隔器窗口的更多信息，参见“Visual C++程序员指南”中的文章“多文档类型，视和框架窗口”中的“分隔器窗口”；和 Visual C++联机文档中的“TN029：分隔器窗口”；以及 CSplitterWnd 类概述。有关如何创建动态分隔器窗口的更多信息，参见“Visual C++联机教程”中的“在增强视的 Scribble 中添加分隔器窗口”；和 MFC 常规示例 VIEWEX。

```
#include <afxext.h>
```

请参阅 CView，CWnd

CSplitterWnd 类成员

Construction	
CsplitterWnd	构造一个 CSplitterWnd 对象
Create	创建一个动态的分隔器窗口并将它与一个 CSplitterWnd 对象连接
CreateStatic	创建一个静态的分隔器窗口并将它与一个 CSplitterWnd 对象连接
CreateView	在一个分隔器窗口中创建一个窗格

Operations	
GetRowCount	返回当前窗格行的计数值
GetColumnCount	返回当前窗格列的计数值
GetRowInfo	返回指定行的信息
SetRowInfo	设置指定行的信息
GetColumnInfo	返回指定列的信息
SetColumnInfo	设置指定列的信息
GetPane	返回位于指定行和列处的窗格
IsChildPane	确定窗口是否是此分隔器窗口的当前子窗格
IdFromRowCol	返回位于指定行和列处的窗格的子窗口 ID
RecalcLayout	在调整行或列尺寸后调用此函数来重新显示该分隔器窗口

续表

GetScrollStyle	返回共享滚动条的风格
SetScrollStyle	为分隔器窗口的共享滚动条指定新的滚动条风格
Overridables	
OnDrawSplitter	绘制一个分隔器窗口的图像
OnInvertTracker	绘制一个分隔器窗口的图像，使它具有与框架窗口相同的大小和形状
CreateScrollBarCtrl	创建一个共享的滚动条控件
DeleteView	从分隔器窗口中删除一个视
SplitRow	表明一个框架窗口是否是水平分隔的
SplitColumn	表明一个框架窗口是否是垂直分隔的
DeleteRow	从分隔器窗口中删除一行
DeleteColumn	从分隔器窗口中删除一列
GetActivePane	根据焦点或框架中的活动视来确定活动窗格
SetActivePane	在框架中设置一个活动窗格
CanActivateNext	检查 Next Pane 或 Previous Pane 命令当前是否有效
ActivateNext	执行 Next Pane 或 Previous Pane 命令
DoKeyboardSplit	执行键盘分隔命令，通常是“Window Split”

续表

DoScroll	执行分隔窗口的同步滚动
DoScrollBy	将分隔窗口滚动给定的像素数

成员函数

CSplitterWnd::ActivateNext

```
virtual void ActivateNext( BOOL bPrev = FALSE );
```

参数

bPrev

表明要激活哪一个窗口。TRUE 表示激活前一个窗口，FALSE 表示激活下一个窗口。

说明

框架调用此成员函数来执行 Next Pane **或** Previous Pane **命令。**

这个成员函数是一个高级命令，它由 CView **类用来作为** CSplitterWnd **实现的代表。**

请参阅 CView, CSplitterWnd::CanActivateNext, CSplitterWnd::SetActivePane

CSplitterWnd::CanActivateNext

virtual BOOL CanActivateNext(BOOL *bPrev* = FALSE);

返回值

如果成功则返回非零值；否则返回 0**。**

参数

bPrev

表明要激活哪一个窗口。TRUE **表示激活前一个窗口，**FALSE **表示激活下一个窗口。**

说明

框架调用此成员函数来检查 Next Pane **和** Previous Pane **命令当前是否有效。**

这个成员函数是一个高级命令，它由 CView **类用来作为** CSplitterWnd **实现的代表。**

请参阅 CSplitterWnd::ActivateNext, CSplitterWnd::SetActivePane

CSplitterWnd::Create

```
BOOL Create( CWnd* pParentWnd, int nMaxRows, int nMaxCols, SIZE sizeMin,
   CCreateContext* pContext, DWORD dwStyle = WS_CHILD | WS_VISIBLE
   |WS_HSCROLL | WS_VSCROLL | SPLS_DYNAMIC_SPLIT,    UINT nID =
AFX_IDW_
   PANE_FIRST );
```

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

分隔器窗口的父框架窗口。

nMaxRows

分隔器窗口中的最大行数。这个值不能超过 2。

nMaxCols

分隔器窗口中的最大列数。这个值不能超过 2。

sizeMin

指出显示一个窗格所需的最小尺寸。

pContext

指向一个 CCreateContext 结构的指针。多数情况下，这个值可以是传递给父框架窗口的 pContext。

dwStyle

指定窗口的风格。

nID

此窗口的子窗口 ID。如果这个分隔器窗口不是嵌套在另一个分隔器窗口中的，则这个 ID 可以是 AFX_IDW_PANE_FIRST。

说明

要创建一个动态分隔器窗口，应调用 Create 成员函数。

你可以通过如下步骤将一个 CSplitterWnd 嵌入一个 CFrameWnd 或 CMDIChildWnd 父对象：

1. 在父框架中嵌入一个 CSplitterWnd 成员变量。
2. 重载父框架的 CFrameWnd::OnCreateClient 成员函数。
3. 从重载的 OnCreateClient 中调用 Create 成员函数。

当你从一个父框架内创建一个分隔器窗口时，将父框架的 *pContext* 参数传递给此分隔器窗口。否则，Create 函数的 *pContext* 可以是 NULL。

一个动态分隔器窗口的初始最小行高度和列宽度由 sizeMin 参数设置。这些最小值被用来确定窗格是否太小以至于不能将其完整地显示出来。

有关动态分隔器窗口的更多信息，参见“Visual C++程序员指南”中的文章“多文档类型，视，和框架窗口”中的“分隔器窗口”；“Visual C++联机文档”中的“TN029：分隔器窗口”；以及 CSplitterWnd 类概述。

请参阅 CSplitterWnd::CreateStatic, CFrameWnd::OnCreateClient, CSplitterWnd::SetRowInfo, CSplitterWnd::SetColumnInfo, CSplitterWnd::CreateView

CSplitterWnd::CreateScrollBarCtrl

virtual BOOL CreateScrollBarCtrl(DWORD *dwStyle*, UINT *nID*);

返回值

如果成功则返回非零值；否则返回 0。

参数

dwStyle

指定窗口的风格。

nID

此窗口的子窗口 ID。**如果这个分隔器窗口不是嵌套在另一个分隔器窗口中的，则这个** ID **可以是** AFX_IDW_PANE_FIRST。

说明

框架调用此成员函数来创建一个共享的滚动条控件。重载 CreateScrollBarCtrl **来包括一个滚动条附近的其它控件。缺省的行为是创建标准的** Windows **滚动条控件。**

请参阅 AfxGetInstanceHandle

CSplitterWnd::CreateStatic

```
BOOL CreateStatic( CWnd* pParentWnd, int nRows, int nCols,
   DWORD dwStyle = WS_CHILD | WS_VISIBLE, UINT nID =
AFX_IDW_PANE_FIRST );
```

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

分隔器窗口的父框架窗口。

nRows

行数。这个值必须不超过 16。

nCols

列数。这个值必须不超过 16。

dwStyle

指定窗口的风格。

nID

此窗口的子窗口 ID。如果这个分隔器窗口不是嵌套在另一个分隔器窗口中的，则这个 ID 可以是 AFX_IDW_PANE_FIRSH。

说明

要创建一个静态的分隔器窗口，应调用 CReateStatic 成员函数。

你可以通过如下步骤将一个 CSplitterWnd 嵌入一个 CFrameWnd 或 CMDIChildWnd 父对象：

1. 在父框架中嵌入一个 CSplitterWnd 成员变量。
2. 重载父框架的 CFrameWnd::OnCreateClient 成员函数。
3. 从重载的 OnCreateClient 中调用 Create 成员函数。

一个静态的分隔器窗口包含了固定数目的窗格，这些窗格是不同类的。

当你创建一个静态分隔器窗口时，你必须同时创建它的所有窗格。CReateView 成员函数通常就是用于这个目的，但你也可以创建其它非视类。

一个静态分隔器窗口的初始最小行高度和列高度是 0。这些最小值用来确定窗格是否太小以至于不能将其完整地显示出来。

要给静态分隔器窗口添加滚动条，就应在 *dwStyle* 参数中指定 WS_HSCROLL 或 WS_VSCROLL 风格。

有关静态分隔器窗口的更多信息，参见“Visual C++程序员指南”中的文章“多文档类型，视，和框架窗口”中的“分隔器窗口”；“Visual C++联机文档”中的“TN029：分隔器窗口”；以及 CSplitterWnd 类概述。

请参阅 CSplitterWnd::Create, CFrameWnd::OnCreateClient,

CSplitterWnd::SetRowInfo,
CSplitterWnd::SetColumnInfo, CSplitterWnd::CreateView

CSplitterWnd::CreateView

virtual BOOL CreateView(int *row*, int *col*, CRuntimeClass* *pViewClass*, SIZE *sizeInit*,
CCreateContext* *pContext*);

返回值

如果成功则返回非零值；否则返回 0。

参数

row

指定用来放置新视的分隔器窗口行。

col

指定用来放置新视的分隔器窗口列。

pViewClass

指定新视的 CRuntimeClass。

sizeInit

指定新视的初始尺寸。

pContext

指向用来创建此视的创建环境的指针（通常，该 *pContext* 被传递给在其中创建此分隔器窗口的父框架的重载的 OnCreateClient 成员函数）。

说明

此成员函数用来为一个静态分隔器窗口创建窗格。一个静态分隔器窗口的所有窗格必须在框架显示这个分隔器之前被创建。

当一个动态分隔器的用户分隔一个窗格，行或列时，框架也调用这个成员函数来创建新的窗格。

请参阅 CSplitterWnd::Create

CSplitterWnd::CSplitterWnd

CSplitterWnd( );

说明

构造一个 CSplitterWnd **分两步。首先调用这个构造函数，它创建一个** CSplitterWnd **对象，然后调用** Create **成员函数来创建分隔器窗口并将它与该** CSplitterWnd **对象连接。**

请参阅 CSplitterWnd::Create

CSplitterWnd::DeleteColumn

virtual void DeleteColumn(int *colDelete*);

参数

colDelete

指定要被删除的列。

说明

当要删除一个列时调用此成员函数。

框架调用此成员函数来实现动态分隔器窗口的逻辑（如果此分隔器窗口具有 SPLS_DYNAMIC_SPLIT **风格）。它可以用虚函数** CreateView **来定制，以实现更高级的动态分隔器。**

请参阅 CSplitterWnd::DeleteRow, CSplitterWnd::CreateView, CSplitterWnd::DeleteView

CSplitterWnd::DeleteRow

virtual void DeleteRow(int *rowDelete*);

参数

rowDelete

指定要被删除的行。

说明

当要删除一个行时调用此成员函数。

框架调用此成员函数来实现动态分隔器窗口的逻辑（如果此分隔器窗口具有 SPLS_ DYNAMIC_SPLIT **风格）。它可以用虚函数** CreateView **来定制，以实现更高级的动态分隔器。**

请参阅 CSplitterWnd::DeleteColumn, CSplitterWnd::CreateView,

CSplitterWnd::DeleteView

CSplitterWnd::DeleteView

virtual void DeleteView(int *row*, int *col*);

参数

row

指定要在分隔器窗口的哪一行中删除视。

col

指定要在分隔器窗口的哪一列中删除视。

说明

当要删除一个视时调用 DeleteView。**如果删除的是活动视，则下一个将成为活动视。缺省的实现假定这个视将在** PostNcDestroy **中自动删除。**

框架调用此成员函数来实现动态分隔器窗口的逻辑（如果此分隔器窗口具有SPLS_DYNAMIC_SPLIT风格）。它可以用虚函数CreateView来定制，以实现更高级的动态分隔器。

请参阅 CWnd::PostNcDestroy, CSplitterWnd::CreateView, CSplitterWnd::DeleteColumn, CSplitterWnd::DeleteRow

CSplitterWnd::DoKeyboardSplit

virtual BOOL DoKeyboardSplit( );

返回值

如果成功则返回非零值；否则返回0。

说明

框架调用此成员函数来实现一个键盘分隔命令，通常是 Window Split。

这个成员函数是一个高级命令，被 CView **用来作为** CSplitterWnd **实现的代表。**

请参阅 CView

CSplitterWnd::DoScroll

virtual BOOL DoScroll(CView* *pViewFrom*, UINT *nScrollCode*,BOOL *bDoScroll* = TRUE);

返回值

如果发生了同步的滚动则返回非零值；否则返回 0。

参数

pViewFrom

一个指向视的指针，滚动消息就是从这个视中发出的。

nScrollCode

一个滚动条代码，它表明了用户的滚动要求。这个参数包括两个部分：一个是低字节，用来确定发生的水平滚动的类型，一个是高字节，用来确定发生的垂直滚动的类型：

- SB_BOTTOM　滚动到底。
- SB_LINEDOWN　向下滚动一行。
- SB_LINEUP　向上滚动一行。
- SB_PAGEDOWN　向下滚动一页。
- SB_PAGEUP　向上滚动一页。

- SB_TOP　滚动到顶。

bDoScroll

确定是否可以发生指定的滚动动作。如果 *bDoScroll* 是 TRUE（就是说，如果一个子窗口存在，并且此分隔窗口有一个滚动范围），则指定的滚动动作可以发生；如果 *bDoScroll* 是 FALSE（就是说，如果不存在一个子窗口，或此分隔视没有滚动范围），则滚动将不会发生。

说明

当视接收到一个滚动消息时，框架调用此成员函数来执行分隔窗口的同步滚动。

重载这个函数，在同步滚动允许之前请求一个由用户指定的动作。

请参阅　CSplitterWnd::DoScrollBy, CView::OnScroll

CSplitterWnd::DoScrollBy

virtual BOOL DoScrollBy(CView* *pViewFrom*, CSize *sizeScroll*,BOOL *bDoScroll* = TRUE);

返回值

如果发生了同步滚动则返回非零值；否则返回 0。

参数

pViewFrom

指向一个视的指针，滚动消息就是从这个视发出的。

sizeScroll

要水平和垂直滚动的像素数。

bDoScroll

确定是否可以发生指定的滚动动作。如果 bDoScroll **是** TRUE**（就是说，**

如果一个子窗口存在，并且此分隔窗口有一个滚动范围），则指定的滚动动作可以发生；如果 bDoScroll 是 FALSE（就是说，如果不存在一个子窗口，或此分隔视没有滚动范围），则滚动将不会发生。

说明

在响应一个滚动消息时，框架调用此成员函数来执行此分隔窗口的同步滚动，滚动由 *sizeScroll* 指定的像素数。正值表示向下和向右滚动，负值表示向上和向左滚动。

重载这个函数，在滚动允许之前请求一个由用户指定的动作。

请参阅 CSplitterWnd::DoScroll, CView::OnScroll

CSplitterWnd::GetActivePane

virtual CWnd* GetActivePane(int* *pRow* = NULL, int* *pCol* = NULL);

返回值

返回指向活动窗格的指针。如果不存在活动窗格则返回 NULL**。**

参数

pRow

指向一个用来接收此活动窗格的行号的 int **的指针。**

pCol

指向一个用来接收此活动窗格的列号的 int **的指针。**

说明

框架调用此成员函数来确定一个分隔器窗口中的活动窗格。

重载这个函数，在获得此活动窗格之前请求一个由用户指定的动作。

请参阅 CSplitterWnd::SetActivePane, CFrameWnd::GetActiveView,CWnd::GetParentFrame, CWnd::GetFocus

CSplitterWnd::GetColumnCount

int GetColumnCount( );

返回值

返回此分隔器中的当前列数。对一个静态分隔器，这也是行数的最大值。

请参阅 CSplitterWnd::GetRowCount

CSplitterWnd::GetColumnInfo

void GetColumnInfo(int *col*, int& *cxCur*, int& *cxMin*);

参数

col

指定一列。

cxCur

一个要用来设置列的当前宽度的 int **的引用。**

cxMin

一个要用来设置列的当前最小宽度的 int **的引用。**

说明

此成员函数用来获取指定列的信息。

请参阅 CSplitterWnd::SetColumnInfo, CSplitterWnd::GetRowInfo

CSplitterWnd::GetPane

CWnd* GetPane(int *row*, int *col*);

返回值

返回位于指定行和列处的窗格。此返回窗格通常是一个 CView **派生类。**

参数

row

指定一行。

col

指定一列。

请参阅 CSplitterWnd::GetActivePane, CSplitterWnd::IdFromRowCol, CSplitterWnd::IsChildPane

CSplitterWnd::GetRowCount

int GetRowCount( );

返回值

返回此分隔器窗口中的当前行数。对一个静态分隔器窗口，这也是最大行数。

请参阅 CSplitterWnd::GetColumnCount

CSplitterWnd::GetRowInfo

void GetRowInfo(int *row*, int& *cyCur*, int& *cyMin*);

参数

row

指定一行。

cyCur

一个要用来设置行的当前高度的 int 的引用。

cyMin

一个要用来设置行的当前最小高度的 int 的引用。

说明

此成员函数用来获取指定行的信息。*cyCur* 参数用指定行的当前高度来填充，*cyWin* 用该行的最小高度来填充。

请参阅 CSplitterWnd::SetRowInfo, CSplitterWnd::GetColumnInfo

CSplitterWnd::GetScrollStyle

```
DWORD GetScrollStyle( ) const;
```

返回值

如果成功，则返回下列窗口风格标志的一个或多个：

- WS_HSCROLL　如果分隔器当前管理共享的水平滚动条。
- WS_VSCROLL　如果分隔器当前管理共享的垂直滚动条。

如果分隔器窗口当前不管理任何共享滚动条，则返回零。

说明

此成员函数返回此分隔器窗口的共享滚动条的风格。

请参阅　CSplitterWnd::SetScrollStyle

CSplitterWnd::IdFromRowCol

int IdFromRowCol(int *row*, int *col*);

返回值

返回此窗格的子窗口的 ID。

参数

row

指定分隔器窗口的行。

col

指定分隔器窗口的列。

说明

此成员函数用来获取位于指定行和列处的窗格的子窗口 ID。**这个成员函数用于创建非视类的窗格，并且可以在该窗格存在之前被调用。**

请参阅 CSplitterWnd::GetPane, CSplitterWnd::IsChildPane

CSplitterWnd::IsChildPane

BOOL IsChildPane(CWnd* *pWnd*, int& *row*, int& *col*);

返回值

若返回值是非零值，则 *pWnd* **是此分隔器窗口的子窗格，并且该函数将此分隔器窗口的窗格的位置填写到** *row* **和** *col* **中。如果** *pWnd* **不是此分隔器窗口的子窗口，则返回** 0。

参数

pWnd

指向一个要被测试的 CWnd **对象的指针。**

row

是对一个将要被设置为行号的 int **的引用。**

col

是对一个将要被设置为列号的 int 的引用。

说明

此成员函数用来确定 pWnd 是否是此分隔器窗口的子窗口。

请参阅 CSplitterWnd::GetPane

CSplitterWnd::OnDrawSplitter

virtual void OnDrawSplitter(CDC* *pDC*, ESplitType *nType*, const CRect& *rect*);

参数

pDC

一个指向设备环境的指针，将要在这个设备上绘画。如果 *pDC* 是 NULL，则框架调用 CWnd::RedrawWindow 并且没有分隔窗口被绘制。

nType

一个 enum ESplitType 值，可以是下列值之一：

- splitBox　分隔器拖拉框。
- splitBar　显示在两个分隔窗口之间的条。
- splitIntersection　分隔窗口的交。当在 Windows95 下允许时，不会调用这个元素。
- splitBorder　分隔窗口边界。

rect

是对一个用来指定分隔窗口的大小和形状的 CRect 对象的引用。

说明

框架调用此成员函数来绘制并指定一个分隔器窗口的精确的特征。

要对一个分隔器窗口的各个图形部件的图像进行高级定制，可以重载

OnDrawSplitter。缺省的图像与 Microsoft Works for Windows 或 Microsoft Windows 95 中的分隔器类似。在 Microsoft Works for Windows 或 Microsoft Windows 95 中分隔器条的交接处是混合在一起的。

有关静态分隔器窗口的更多信息，参见“Visual C++程序员指南”中的文章“多文档类型，视和框架窗口”中的“分隔器窗口”；“Visual C++文档”中的“TN029：分隔器窗口”；以及 CSplitterWnd 类概述。

请参阅 CSplitterWnd::OnInvertTracker

CSplitterWnd::OnInvertTracker

virtual void OnInvertTracker(const CRect& *rect*);

参数

rect

是对一个指定跟踪矩形的 CRect 对象的引用。

说明

在调整分隔器大小期间，框架调用此成员函数。

要对一个分隔器窗口的图像进行高级定制，可以重载 OnInvertTracker。缺省的图像与 Microsoft Works for Windows 或 Microsoft Windows 95 中的分隔器类似。在 Microsoft Works for Windows 或 Microsoft Windows 95 中分隔器条的交接处是混合在一起的。

有关静态分隔器窗口的更多信息，参见“Visual C++程序员指南”中的文章“多文档类型，视和框架窗口”中的“分隔器窗口”；“Visual C++文档”中的“TN029：分隔器窗口”；以及 CSplitterWnd 类概述。

请参阅 CSplitterWnd::OnDrawSplitter

CSplitterWnd::RecalcLayout

void RecalcLayout( );

说明

在你用 SetRowInfo **和** SetColumnInfo **成员函数调整了行和列的大小之后，调用此成员函数来正确地重新显示此分隔器窗口。如果你是在分隔器窗口可见之前改变行和列的大小作为创建过程的一部分，则不需要调用这个成员函数。**

无论什么时候用户调整此分隔器窗口的大小或移动一个分隔条，框架都将调用这个成员函数。

请参阅 CSplitterWnd::SetRowInfo, CSplitterWnd::SetColumnInfo

CSplitterWnd::SetActivePane

virtual void SetActivePane(int *row*, int *col*, CWnd* *pWnd* = NULL);

参数

row

如果 *pWnd* **是** NULL，**则此参数指定将要被激活的窗格中的行。**

col

如果 *pWnd* **是** NULL，**则此参数指定将要被激活的窗格中的列。**

pWnd

一个指向 CWnd **对象的指针。如果是** NULL，**则由** *row* **和** *col* **指定的窗格被置为活动窗格。如果不是** NULL，**则它指定了要被激活的窗格。**

说明

当用户将焦点改变到框架窗口中的某一个窗格时，框架调用此成员函数来设置一个活动窗格。你可以显式地调用 SetActivePane **来将焦点改变到指定的视。可以通过提供行和列，或** *pWnd* **来指定窗格。**

请 参 阅 CSplitterWnd::GetActivePane, CSplitterWnd::GetPane, CFrameWnd::SetActiveView

CSplitterWnd::SetColumnInfo

void SetColumnInfo(int *col*, int *cxIdeal*, int *cxMin*);

参数

col

指定一个分隔器窗口列。

cxIdeal

指定分隔器窗口列的以像素表示的理想宽度。

cxMin

指定分隔器窗口列以像素表示的最小宽度。

说明

此成员函数用来设置一列的新的最小宽度和理想宽度。列的最小值确定了什么时候列将太小以至于不能被完全显示。

当框架显示分隔器窗口时，它从分隔器窗口的客户区的左上角至右下角，根据窗格的理想尺寸按行和列来排放各个窗格。

请参阅 CSplitterWnd::GetRowInfo, CSplitterWnd::RecalcLayout

CSplitterWnd::SetRowInfo

void SetRowInfo(int *row*, int *cyIdeal*, int *cyMin*);

参数

row

指定一个分隔器窗口行。

cyIdeal

指定分隔器窗口行的以像素表示的理想高度。

cyMin

指定分隔器窗口行以像素表示的最小高度。

说明

此成员函数用来设置一行的新的最小高度和理想高度。行的最小值确定了什么时候行将太小以至于不能被完全显示。

当框架显示分隔器窗口时，它从分隔器窗口的客户区的左上角至右下角，根据窗格的理想尺寸按行和列来排放各个窗格。

请参阅 CSplitterWnd::GetRowInfo, CSplitterWnd::SetColumnInfo, CSplitterWnd::RecalcLayout

CSplitterWnd::SetScrollStyle

void SetScrollStyle(DWORD *dwStyle*);

参数

dwStyle

此分隔器窗口的共享滚动条所指出的滚动条风格，可以是下列值之一：

- WS_HSCROLL **创建/显示水平共享滚动条。**
- WS_VSCROLL **创建/显示垂直共享滚动条。**

说明

此成员函数用来为分隔器窗口的共享滚动条指定新的滚动条风格。一旦创建了一个滚动条，即使调用了没有这个风格的 SetScrollStyle**，此滚动条也不会被销毁；而是隐藏这些滚动条。在调用** SetScrollStyle **之后，为了使所有的改变见效，**

必须调用 RecalcLayout。

请参阅 CSplitterWnd::GetScrollStyle

CSplitterWnd::SplitColumn

virtual BOOL SplitColumn(int *cxBefore*);

返回值

如果成功则返回非零值；否则返回 0。

参数

cxBefore

在分隔出现点前面的以像素表示的位置。

说明

当创建了一个垂直的分隔器窗口时调用此成员函数。SplitColumn 指明分隔要出现的缺省位置。

框架调用 SplitColumn 来实现动态分隔器窗口（即，分隔器窗口具有 SPLS_DYNAMIC_ SPLIT 风格）的逻辑。可以用 CreateView 函数来定制它，以实现更为高级的动态分隔器。

请参阅 CSplitterWnd::CreateView, CSplitterWnd::SplitRow, CSplitterWnd::RecalcLayout

CSplitterWnd::SplitRow

virtual BOOL SplitRow(int *cyBefore*);

返回值

如果成功则返回非零值；否则返回 0。

参数

cyBefore

在分隔出现点前面的以像素表示的位置。

说明

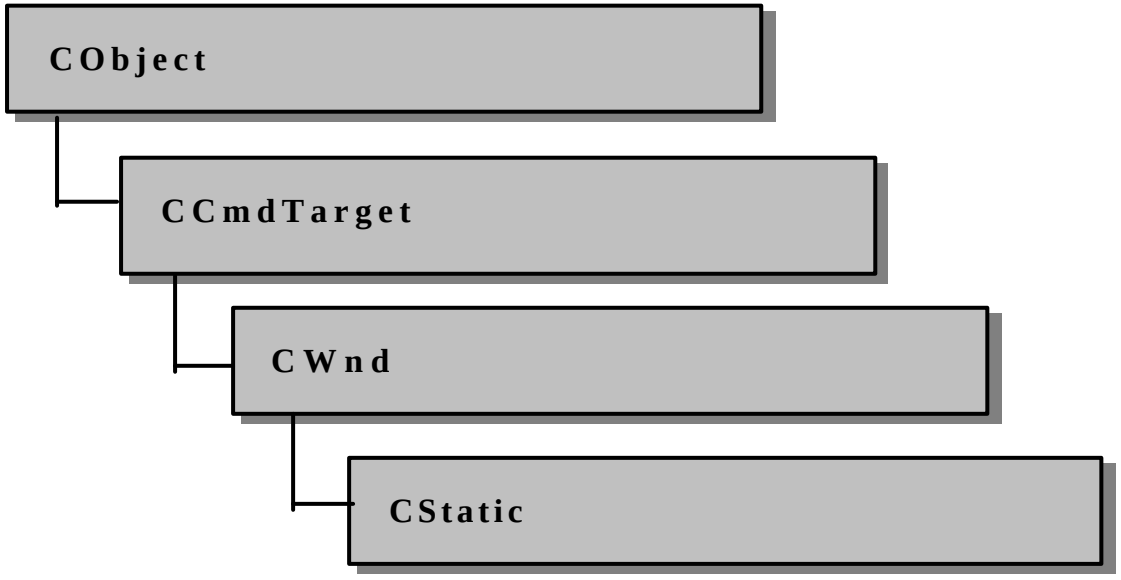
当创建了一个水平的分隔器窗口时调用此成员函数。SplitRow **指明分隔要出现的缺省位置。**

框架调用 SplitRow **来实现动态分隔器窗口（即，分隔器窗口具有** SPLS_DYNAMIC_SPLIT **风格）的逻辑。可以用** CreateView **函数来定制它，以**

实现更为高级的动态分隔器。

请参阅 CSplitterWnd::SplitColumn, CSplitterWnd::CreateView, CSplitterWnd::RecalcLayout

CStatic



CStatic 类提供了一个 Windows 静态控件的性能。一个静态控件用来显示一个文本字符串，框，矩形，图标，光标，位图，或增强的图元文件。它可以被用来作为标签，框，或用来分隔其它的控件。一个静态控件不接收输入，也不提供输出；但是，如果它是用 SS_NOTIFY 风格创建的，则它可以通知其父有关设备点击的消息。

创建一个静态控件分两步。首先，调用构造函数来构造此 CStatic 对象，然后调用 Create 成员函数来创建此静态控件并将它与该 CStatic 对象连接。

如果你是在一个对话框中创建了一个静态控件（通过一个对话框资源），则当用户关闭这个对话框时，此 CStatic 对象被自动销毁。

如果你是在一个窗口中创建了一个 CStatic 对象，则必须由你来销毁它。在一个窗口的堆栈中创建的 CStatic 对象将自动被销毁。如果你是使用 new 函数在堆中创建 CStatic 对象，则当你使用完后，必须调用 delete 来销毁这个 CStatic

对象。

#include <afxwin.h>

请参阅 CWnd, CButton, CComboBox, CEdit, CListBox, CScrollBar, CDialog

CStatic 类成员

Construction	
Cstatic	构造一个 CStatic 对象
Initialization	
Create	创建 Windows 静态控件并将它与该 CStatic 对象连接

Operations	
SetBitmap	指定要在此静态控件中显示的位图
GetBitmap	获取先前用 SetBitmap 设置的位图的句柄
SetIcon	指定一个要在此静态控件中显示的图标
GetIcon	获取先前用 SetIcon 设置的图标的句柄
SetCursor	指定要显示在此静态控件中的光标图像
GetCursor	获取先前用 SetCursor 设置的光标图像的句柄
SetEnhMetaFile	指定要显示在此静态控件中的增强的图元文件
GetEnhMetaFile	获取先前用 SetEnhMetaFile 设置的增强图元文件的句柄

成员函数

CStatic::Create

BOOL Create(LPCTSTR *lpszText*, DWORD *dwStyle*, const RECT& *rect*,
CWnd* *pParentWnd*, UINT *nID* = 0xffff);

返回值

如果成功则返回非零值；否则返回 0；

参数

lpszText

指定要放置在控件中的文本。如果是 NULL，**则表示没有文本是可见的。**

dwStyle

指定静态控件的窗口风格。任何静态控件风格的组合都可以用于这个控件。

rect

指定静态控件的位置和大小。它可以是一个 RECT **结构或一个** CRect **对象。**

pParentWnd

指定 CStatic **父窗口，通常是一个** CDialog **对象。它不能是** NULL。

nID

指定静态控件的控件 ID。

说明

可以将下列窗口风格用于一个静态控件：

- WS_CHILD　总要使用。
- WS_VISIBLE　经常使用。
- WS_DISABLED　很少使用。

如果你准备在此静态控件中显示一个位图，光标，图标，或图元文件，你必须使用下列风格之一：

- SS_BITMAP　此风格用于位图。
- SS_ICON　此风格用于光标和图标。

- SS_ENHMETAFILE　此风格用于增强的图元文件。

对于光标，位图，或图标，你也可以使用下面的风格：

- SS_CENTERIMAGE　用来使静态控件中的图像居中。

请参阅　CStatic::CStatic

CStatic::CStatic

CStatic( );

说明

此成员函数用来构造一个 CStatic 对象。

请参阅　CStatic::Create

CStatic::GetBitmap

HBITMAP GetBitmap( ) const;

返回值

返回一个当前位图的句柄，如果没有设置位图，则返回 NULL。

说明

此成员函数用来获取先前用 SetBitmap **设置的与** CStatic **关联的位图句柄。**

请参阅 CStatic::SetBitmap, STM_GETIMAGE, Bitmaps

CStatic::GetCursor

HCURSOR GetCursor( );

返回值

返回一个当前图标句柄。如果没有设置图标则返回 NULL。

说明

此成员函数用来获取先前用 SetCursor **设置的与** CStatic **关联的光标句柄。**

请参阅 CStatic::SetCursor, STM_GETIMAGE, Cursors

CStatic::GetEnhMetaFile

HENHMETAFILE GetEnhMetaFile( ) const;

返回值

返回一个当前增强图元文件句柄。如果没有设置增强的图元文件则返回 NULL。

说明

此成员函数用来获取先前用 SetEnhMetaFile **设置的与** CStatic **关联的增强的图元文件句柄。**

请参阅 CStatic::SetEnhMetafile, STM_GETIMAGE

CStatic::GetIcon

HICON GetIcon( ) const;

返回值

返回一个当前图标句柄。如果没有设置增强的图标则返回 NULL。

说明

此成员函数用来获取先前用 SetIcon **设置的与** CStatic **关联的图标句柄。**

请参阅 CStatic::SetIcon, STM_GETICON, Icons

CStatic::SetBitmap

HBITMAP SetBitmap(HBITMAP *hBitmap*);

返回值

返回先前与此静态控件关联的位图的句柄。如果没有与此静态控件关联的位图，则返回 NULL。

参数

hBitmap

要绘制在此静态控件中的位图句柄。

说明

此成员函数用来将一个新的位图与此静态控件关联。

这个位图将被自动绘制在此静态控件中。缺省的，它将被绘制在左上角，并且此静态控件将根据位图的大小来调整尺寸。

你可以使用不同的窗口和静态控件风格，包括下列值：

- SS_BITMAP　此风格总是用于位图。
- SS_CENTERIMAGE　用来在此静态控件中居中。如果图像比静态控件大，则它将被剪切掉。如果它比静态控件小，则图像周围的空间将被用位图左上角的像素的颜色填充。

请参阅　CStatic::GetBitmap, STM_SETIMAGE, Bitmaps

CStatic::SetCursor

HCURSOR SetCursor(HCURSOR *hCursor*);

返回值

返回先前与此静态控件关联的光标的句柄。如果没有与此静态控件关联的光标，则返回 NULL。

参数

hCursor

要绘制在此静态控件中的光标句柄。

说明

此成员函数用来将一个新的光标与此静态控件关联。

这个光标将被自动绘制在此静态控件中。缺省的，它将被绘制在左上角，并且此静态控件将根据光标的大小来调整尺寸。

你可以使用不同的窗口和静态控件风格，包括下列值：

- SS_ICON　此风格总是用于位图。
- SS_CENTERIMAGE　用来在此静态控件中居中。如果图像比静态控件大，则它将被剪切掉。如果它比静态控件小，则图像周围的空间将被用位图左上角的像素颜色填充。

请参阅　CStatic::GetCursor, STM_SETIMAGE, Cursors

CStatic::SetEnhMetaFile

HENHMETAFILE SetEnhMetaFile(HENHMETAFILE *hMetaFile*);

返回值

返回先前与此静态控件关联的增强图元文件的句柄。如果没有与此静态控件关联的增强图元文件，则返回 NULL。

参数

hMetaFile

要绘制在此静态控件中的增强图元文件句柄。

说明

此成员函数用来将一个新的增强图元文件与此静态控件关联。

这个增强图元文件将被自动绘制在此静态控件中。缺省的，它将被绘制在左上角，并且此静态控件将根据增强图元文件的大小来调整尺寸。

你可以使用不同的窗口和静态控件风格，包括下列值：

- SS_ENHMETAFILE　**此风格总是用于位图。**

请参阅　CStatic::GetEnhMetafile, STM_SETIMAGE

CStatic::SetIcon

HICON SetIcon(HICON *hIcon*);

返回值

返回先前与此静态控件关联的图标的句柄。如果没有与此静态控件关联的图标，则返回 NULL。

参数

hIcon

要绘制在此静态控件中的图标句柄。

说明

此成员函数用来将一个新的图标与此静态控件关联。

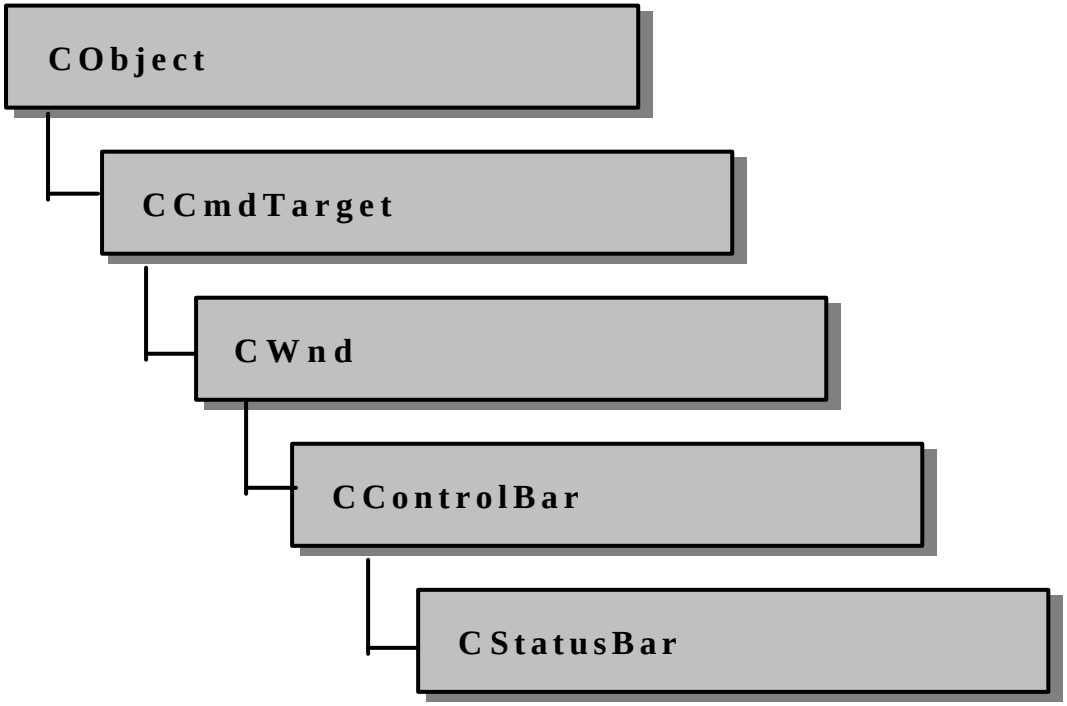
这个图标将被自动绘制在此静态控件中。缺省的，它将被绘制在左上角，并且此静态控件将根据图标的大小来调整尺寸。

你可以使用不同的窗口和静态控件风格，包括下列值：

- SS_ICON　此风格总是用于图标。
- SS_CENTERIMAGE　用来在此静态控件中居中。如果图像比静态控件大，则它将被剪切掉。如果它比静态控件小，则图像周围的空间将被用位图左上角的像素颜色填充。

请参阅 CStatic::GetIcon, STM_SETICON, Icons

CStatusBar



一个 CStatusBar 对象是一个带有一行文本输出窗格的控件，或者称为“指示器”。这些输出窗格常被用作消息行和状态指示器。例如：简单解释所选菜单命令的菜单帮助消息行，作为显示 SCROLL LOCK，NUMLOCK 以及其它键状态的指示器。

在 MFC4.0 之后新增加的成员函数 CStatusBar:: GetStatusBarCtrl，允许你利用 Windows 通用控件对状态条的定制和其它功能的支持。CStatusBar 成员函数提供了 Windows 通用控件的大多数功能；但是，当你调用 GetStatusBarCtrl 时，你可以赋予你的状态条更多的 Windows95 状态条的特性。当你调用 GetStatusBarCtrl 时，它将返回一个对 CStatusBarCtrl 对象的引用。参见 CStatusBarCtrl 可以获得有关使用 Windows 通用控件设计工具条的更多信息。有关通用控件的更多的一般信息，请参见“Windows 95 SDK 程序员参考”中的“通用控件”。

框架将指示器的信息保存在一个数组中，且最左边的的指示器位于 0 位置上。当创建一个工具条时，可以使用一个字符串 ID 数组，框架把这一组 ID 与对应的指示器关联起来。此后，你就可以使用字符串 ID 或索引值访问一个指示器。

缺省的，第一个指示器是“可伸缩的”：该指示器占据了其它指示器窗格未用到的状态条长度，因此其它窗格是右对齐的。

可以按下列步骤创建一个状态条：

1. 构造 CStatusBar 对象。
2. 调用 Create（或 CreateEx）函数来创建状态条窗口并将它连接到 CStatusBar 对象。
3. 调用 SetIndicators 函数将字符串 ID 与每一个指示器联系起来。

有三种办法可以更新状态条窗口中的文本：

1. 调用 CWnd::SetWindowText 只更新窗格 0 中的文本。
2. 在状态条的 ON_UPDATE_COMMAND_UI 处理程序中调用 CCmdUI::SetText 函数。
3. 调用 SetPaneText 函数更新任何窗格中的文本。

调用 SetPaneStyle 可以更新一个状态条窗格的风格。

有关使用 CStatusBar 的更多信息，参见“Visual C++程序指南”中的文章“状态条”，以及技术注释 31，“控制条”。

```
#include <afxext.h>
```

请参阅 CStatusBarCtrl, CControlBar, CWnd::SetWindowText, CStatusBar::SetIndicators

CStatusBar 类成员

Construction	
CstatusBar	构造一个 CStatusBar 对象
Create	创建状态条，并将它与 CstatusBar 对象连接，且设置初始字体和条高度
CreateEx	创建一个具有嵌入 CStatusBarCtrl 对象附加风格的 CStatusBar 对象
SetIndicators	设置指示器 ID

Attributes	
CommandToIndex	获取给定指示器 ID 的索引
GetItemID	获取给定索引的指示器 ID
GetItemRect	获取给定索引值的显示矩形
GetPaneInfo	获取一个给定索引的指示器 ID，风格和宽度
GetPaneStyle	获取一个给定索引的指示器风格
GetPaneText	获取一个给定索引的指示器文本
GetStatusBarCtrl	允许直接访问基础通用控件
SetPaneStyle	设置一个给定索引的指示器风格
SetPaneText	设置一个给定索引的指示器文本
SetPaneInfo	设置一个给定索引的指示器 ID，风格和宽度

续表

Overridables	
DrawItem	当一个属主绘制的状态条控件的外观改变时，此函数被调用

成员函数

CStatusBar::CommandToIndex

int CommandToIndex(UINT *nIDFind*) const;

返回值

如果成功则返回指示器的索引；否则返回-1。

参数

nIDFind

要获取其索引的指示器的字符串 ID。

说明

此成员函数用来获取一个给定 ID **的指示器索引值。第一个指示器的索引值为** 0。

请参阅 CStatusBar::GetItemID

CStatusBar::Create

BOOL Create(CWnd* *pParentWnd*, DWORD *dwStyle* = WS_CHILD | WS_VISIBLE |
CBRS_BOTTOM, UINT *nID* = AFX_IDW_STATUS_BAR);

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

指向一个 CWnd **对象的指针，该对象的** Windows **窗口是此状态条的父窗口。**

dwStyle

状态条风格。除标准的 Windows**风格之外，还支持下列风格：**

- CBRS_TOP **控制条在框架窗口的顶部。**
- CBRS_BOTTOM **控制条在框架窗口的底部。**
- CBRS_NOALIGN **当父窗口的尺寸发生变化时不重新定位控制条。**

nID

工具条的子窗口 ID。

说明

此成员函数用来创建一个状态条（一个子窗口）并将它与 CStatusBar 对象连接。

还将状态条的初始字体和高度设置为缺省值。

请参阅 CStatusBar::SetIndicators

CStatusBar::CreateEx

```
BOOL CreateEx( CWnd* pParentWnd, DWORD dwCtrlStyle = 0,
    DWORD dwStyle = WS_CHILD | WS_VISIBLE | CBRS_BOTTOM,
    UINT nID = AFX_IDW_STATUS_BAR );
```

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

指向 CWnd **对象的指针，该对象的** Windows**窗口是状态条的父窗口。**

dwCtrlStyle

用于嵌入 CStatusBarCtrl **对象创建的附加风格。有关所支持的风格的完整列表，参见** dwStyle。

被支持的状态条风格有：

- SBARS_SIZEGRIP **在状态条的右端有一个调整大小的把手的状态条控件。一个调整大小的把手就类似于一个调整大小的边；它是一个特别的区**

域，用户可以点击并拖动它来调整父窗口的大小。

- SBT_TOOLTIPS　状态条支持工具提示。

dwStyle

状态条风格。所支持的风格包括：

- SBARS_SIZEGRIP　在状态条的右端有一个调整大小的把手的状态条控件。一个调整大小的把手就类似于一个调整大小的边；它是一个特别的区域，用户可以点击并拖动它来调整父窗口的大小。
- SBT_TOOLTIPS　状态条支持工具提示。

nID

状态条的子窗口 ID。

说明

此成员函数用来创建一个状态条（一个子窗口），并将它与 CStatusBar 对象连

接。此函数也将状态条的初始字体和高度设置为缺省值。

在创建需要具有一定风格的嵌入状态条控件时，使用 CreateEx 来代替 Creat。例如，设置 dwCtrlStyle 为 SBT_TOOLTIPS 来在状态条对象中显示工具提示。

CStatusBar::CStatusBar

CStatusBar();

说明

此成员函数用来构造一个 CStatusBar 对象，如果必要的话，创建一种缺省的状态条字体，并将字体特征设置为缺省值。

请参阅 CStatusBar::Create

CStatusBar::DrawItem

virtual void DrawItem(LPDRAWITEMSTRUCT *lpDrawItemStruct*);

参数

lpDrawItemStruct

一个指向 DRAWITEMSTRUCT **结构的指针，该结构包含了有关所需要的绘画类型的信息。**

说明

当一个属主绘制的状态条的外观改变时，框架调用此成员函数。

DRAWITEMSTRUCT **结构的** itemAction **成员定义了要执行的绘画动作。重载这个成员函数可以实现一个属主绘制的** CStatusBar **对象的绘画。在此成员函数终止之前，应用程序应该恢复在** *lpDrawItemStruct* **中提供的显示环境所选择的**

所有图形设备接口（GDI）。

请参阅 CWnd::OnDrawItem

CStatusBar::GetItemID

UINT GetItemID(int *nIndex*) const;

返回值

返回由 *nIndex* **指定的指示器的** ID。

参数

nIndex

要获取其 ID **的指示器的索引。**

说明

此成员函数返回由 *nIndex* **指定的指示器的** ID。

请参阅 CStatusBar::CommandToIndex

CStatusBar::GetItemRect

void GetItemRect(int *nIndex*, LPRECT *lpRect*) const;

参数

nIndex

要获取其矩形坐标的指示器的索引。

lpRect

执行一个 RECT **结构或一个** CRect **对象，该结构或对象将用来接收由** *nIndex* **指定的指示器的坐标。**

说明

此成员函数用来将由 *nIndex* **指定的指示器的坐标拷贝到由** *lpRect* **指定的结构中去。坐标是相对于此状态条的左上角的以像素表示的坐标。**

请参阅 CStatusBar::CommandToIndex, CStatusBar::GetPaneInfo

CStatusBar::GetPaneInfo

void GetPaneInfo(int *nIndex*, UINT& *nID*, UINT& *nStyle*, int& *cxWidth*) const;

参数

nIndex

要获取其信息的窗格的索引。

nID

对一个要被设置为此窗格的 ID **的** UINT **值的引用。**

nStyle

对一个要被设置为此窗格的风格的 UINT 值的引用。

cxWidth

对一个要被设置为此窗格的宽度的整数值的引用。

说明

此成员函数用 *nID*，*nStyle* 和 *cxWidth* 来设置位于由 *nIndex* 指定的位置的指示器窗格的 ID，风格和宽度。

请参阅 CStatusBar::SetPaneInfo, CStatusBar::GetItemID, CStatusBar::GetItemRect

CStatusBar::GetPaneStyle

UINT GetPaneStyle(int *nIndex*) const;

返回值

返回由 *nIndex* **指定的状态条窗格的风格。**

参数

nIndex

要获取其风格的窗格的索引。

说明

此成员函数用来获取一个状态条窗口的风格。一个窗格的风格确定窗格显示的外观。

有关一个状态条的可用风格的列表，参见 Create。

请参阅 CStatusBar::Create, CStatusBar::SetPaneStyle

CStatusBar::GetPaneText

CString GetPaneText(int *nIndex*) const;
void GetPaneText(int *nIndex* , CString& *rString*) const;

返回值

返回一个包含窗格的文本的 CString **对象。**

参数

nIndex

要获取其文本的窗格的索引。

rString

对一个包含要获取的文本的 CString **对象的引用。**

说明

此成员函数用来获取显示在一个状态条窗格中的文本。此成员函数的第二种形式用字符串文本来填充一个 CString **对象。**

请参阅 CStatusBar::SetPaneText

CStatusBar::GetStatusBarCtrl

CStatusBarCtrl& GetStatusBarCtrl( ) const;

返回值

返回一个对 CStatusBarCtrl **对象的引用。**

说明

使用 CStatusBarCtrl **可以利用** Windows **状态条通用控件的性能，并且可以利用**

CStatusBarCtrl 提供的对状态条定制的支持。例如，通过使用这个通用控件，你可以指定一种风格，使得状态条上有一个调整大小的把手，或者你可以指定一种风格，使得状态条显示在父窗口的客户区的顶端。

有关通用控件的更多信息，参见“Windows95 SDK 程序员参考”中的通用控件。

CStatusBar::SetIndicators

BOOL SetIndicators(const UINT* *lpIDArray*, int *nIDCount*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpIDArray

指向一个 ID 数组的指针。

nIDCount

由 lpIDArray 指向的数组的元素数目。

说明

此成员函数用来将每一个指示器的 ID 设置为由数组 *lpIDArray* 的相应元素指定的值，加载由每一个 ID 指定的字符串资源，并将这个字符串设置为此指示器的文本。

请参阅 CStatusBar::CStatusBar, CStatusBar::Create, CStatusBar::SetPaneInfo, CStatusBar::SetPaneText

CStatusBar::SetPaneInfo

void SetPaneInfo(int *nIndex*, UINT *nID*, UINT *nStyle*, int *cxWidth*);

参数

nIndex

要设置其风格的指示器窗格的索引。

nID

该指示器窗格的新 ID。

nStyle

该指示器窗格的新风格。

cxWidth

该指示器窗格的新宽度。

说明

此成员函数用来设置指示器窗格的新 ID，风格和宽度。

下面的指示器风格是被支持的：

- SBPS_NOBORDERS　在窗格周围没有 3-D 边框。
- SBPS_POPOUT　反转边界以使文字“凸出来”。
- SBPS_DISABLED　不绘制文本。
- SBPS_STRETCH　扩大窗格以填充不用的空白空间。没有状态条只能有一个窗格具有这种风格。
- SBPS_NORMAL　没有扩展，边界或“凸出来”。

请参阅　CStatusBar::GetPaneInfo

CStatusBar::SetPaneStyle

void SetPaneStyle(int *nIndex*, UINT *nStyle*);

参数

nIndex

要设置其风格的窗格的索引。

nStyle

要设置的窗格的风格。

说明

此成员函数用来设置一个状态条窗格的风格。窗格的风格决定了窗格是怎么显示的。

有关状态条可用风格的列表，参见 SetPaneInfo。

请参阅 CStatusBar::Create, CStatusBar::GetPaneStyle

CStatusBar::SetPaneText

BOOL SetPaneText(int *nIndex*, LPCTSTR *lpszNewText*, BOOL *bUpdate* = TRUE);

返回值

如果成功则返回非零值；否则返回 0。

参数

nIndex

要设置其文本的窗格的索引。

lpszNewText

指向新的窗格文本的指针。

bUpdate

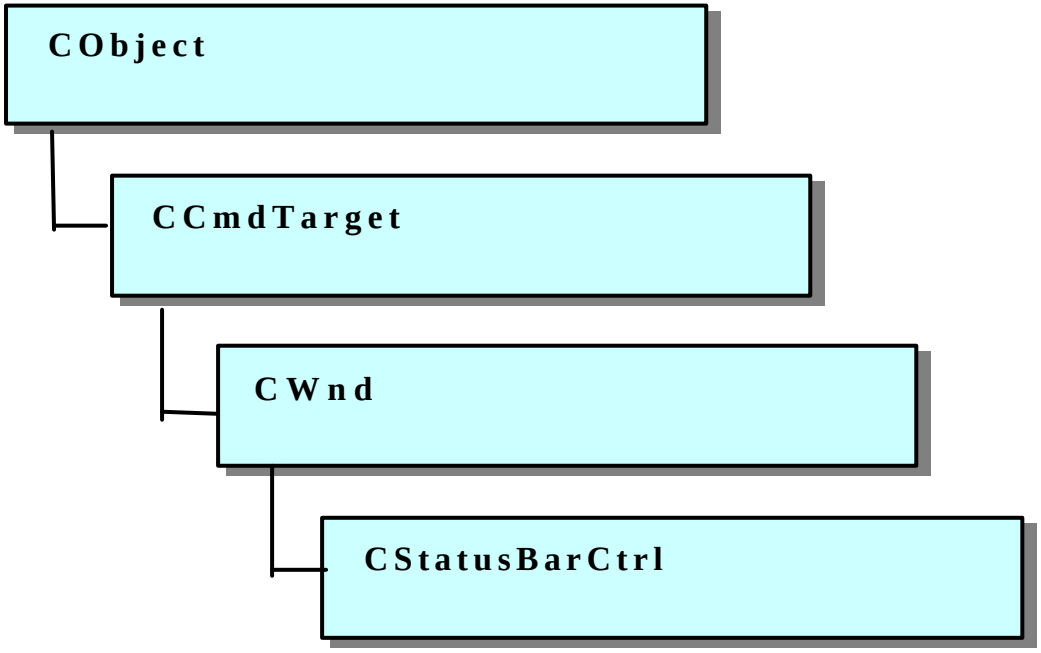
如果是 TRUE，则在文本被设置之后，窗格是无效的。

说明

此成员函数用来将窗格文本设置为由 *lpszNewText* 指定的字符串。

请参阅 CStatusBar::GetPaneText

CStatusBarCtrl



一个“状态条控件”是一个水平的窗口，通常显示在一个父窗口的底部，在其

中应用程序可以显示不同类型的状态信息。可以将此状态条控件分割为多个部分，用来显示多种类型的信息。

CStatusBarCtrl 类提供了 Windows 通用状态条控件的性能。这个控件（也就是 CStatusBarCtrl 类）只对运行在 Windows 95 和 Windows NT3.51 或更新版本下的程序来说是可用的。

有关使用 CStatusBarCtrl 的更多信息，参见“Visual C++程序员指南”中的“控件主题”和“使用 CStatusBarCtrl”。

#include <afxcmn.h>

请参阅 CtoolBarCtrl

CStatusBarCtrl 类成员

Construction	
CstatusBarCtrl	构造一个 CStatusBarCtrl 对象
Create	创建一个状态条控件并将它与一个 CstatusBarCtrl 对象连接

Attributes	
SetText	设置一个状态条控件的给定部分中的文本
GetText	从一个状态条控件的给定部分中获取文本
GetTextLength	从一个状态条控件的给定部分获取文本的用字符数目表示的长度
SetParts	设置一个状态条控件中的部分数目，以及每一个部分的右边缘的坐标
GetParts	获取一个状态条控件中的部分数目
GetBorders	获取一个状态条控件的水平或垂直边界的对齐宽度
SetMinHeight	设置一个状态条控件的绘制区域的最小高度
SetSimple	指定一个状态条控件是显示简单的文本，还是显示由前一次调用 SetParts 设置的所有控件部分

续表

GetRect	获取一个状态条控件中的一个部分的边界矩形
IsSimple	检查一个状态条窗口控件以确定它是否处于简单模式
GetTipText	获取状态条中的一个窗格的工具提示文本
SetTipText	设置状态条中的一个窗格的工具提示文本
SetBkColor	设置一个状态条控件中的背景颜色
SetIcon	设置状态条控件中的窗格的图标
Overridables	
DrawItem	当一个属主绘制的状态条控件的外观改变时调用此函数

成员函数

CStatusBarCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果成功则返回非零值；否则返回零。

参数

dwStyle

指定状态条控件的风格。状态条控件风格的任意组合都适用于这个控件。这个参数必须包括 WS_CHILD **风格。它也必须包括** WS_VISIBLE **风格。参见说明部分可以获得更多信息。**

rect

指定状态条控件的大小和位置。它可以是一个 CRect **对象或一个** RECT **结构。**

pParentWnd

指定状态条控件的父窗口，通常是一个 CDialog**。它不能是** NULL**。**

nID

指定状态条控件的 ID。

说明

构造一个 CStatusBarCtrl 对象可以分两步。首先调用构造函数，然后调用 Create 来创建状态条控件并将它与 CStatusBarCtrl 对象连接。

dwStyle 参数可以是下列值的任意组合：

- CCS_BOTTOM　使控件将它自己定位在父窗口的客户区的底端，并将宽度设置为与父窗口的宽度一样。状态条控件将此作为它的缺省风格。
- CCS_NODIVIDER　禁止在控件的顶部绘制两个像素的高亮区。
- CCS_NOHILITE　禁止在控件的顶部绘制一个像素的高亮区。
- CCS_NOMOVEY　使控件响应 WM_SIZE 消息，调整自己的大小并水平移

动自己，但不是垂直移动。如果使用了 CCS_NORESIZE 风格，则此风格不能使用。

- CCS_NOPARENTALIGE 禁止控件自动移动到父窗口的顶部或底部。不管控件的父窗口的尺寸怎么改变，控件都保持它在父窗口中的位置。如果也使用了 CCS_TOP 和 CCS_BOTTOM 风格，则高度被调整为缺省值，但位置和宽度仍然保持不变。
- CCS_NORESIZE 当控件设置它自己的初始尺寸和新尺寸时，禁止控件使用缺省的宽度和高度。而是用在创建或调整大小的请求中指定的宽度和高度。
- CCS_TOP 使控件将自己定位在其父窗口的顶部，并将自己的宽度设置为与父窗口的宽度一样。

一个状态窗口的缺省位置是沿着父窗口的底部，但是你也可以指定 CCS_TOP

风格来使它显示在父窗口的客户区的顶部。还可以指定 SBARS_SIZEGRIP 风格来使它包括一个位于状态窗口右端的调整大小的把手。并不建议组合 CCS_TOP 和 SBARS_SIZEGRIP 风格，因为这样获得的调整大小把手是没有用的，尽管系统将它绘制在了状态窗口中。

请参阅 CStatusBarCtrl::CStatusBarCtrl

CStatusBarCtrl::CStatusBarCtrl

CStatusBarCtrl( );

说明

此成员函数用来构造一个 CStatusBarCtrl 控件。

请参阅 CStatusBarCtrl::Create

CStatusBarCtrl::DrawItem

virtual void DrawItem(LPDRAWITEMSTRUCT *lpDrawItemStruct*);

参数

lpDrawItemStruct

一个指向 DRAWITEMSTRUCT **结构的长指针。该结构包含了有关所要求的绘制的类型的信息。**

说明

当一个属主绘制的状态条控件的外观发生改变时框架调用此成员函数。 DRAWITEMS- TRUCT **结构的** itemAction **成员定义了要执行的绘制动作。**

缺省的，这个成员函数不做任何事情。重载这个成员函数来实现对一个属主绘制的 CStatusBarCtrl **对象的绘制。**

在此成员函数终止之前，应用程序应该恢复在 lpDrawItemStruct 中提供的显示环境所选择的所有图形设备接口（GDI）。

请参阅 CWnd::OnDrawItem

CStatusBarCtrl::GetBorders

```
BOOL GetBorders( int* pBorders ) const;
BOOL GetBorders( int& nHorz, int& nVert, int& nSpacing ) const;
```

返回值

如果成功则返回非零值；否则返回零。

参数

pBorders

一个有三个元素的整数数组的地址。第一个元素用来接收水平边界的宽

度，第二个元素用来接收垂直边界的宽度，第三个元素用来接收矩形之间的边界的宽度。

nHorz

是对一个用来接收水平边界宽度的整数的引用。

nVert

是对一个用来接收垂直边界宽度的整数的引用。

nSpacing

是对一个用来接收矩形之间的边界宽度的整数的引用。

说明

此成员函数用来获取状态条控件的水平和垂直边界，以及矩形之间的空间的当前宽度。这些边界决定了控件的外部边缘和控件内部包含文本的矩形之间的间隔。

请参阅 CStatusBarCtrl::GetParts, CStatusBarCtrl::SetParts

CStatusBarCtrl::GetParts

int GetParts(int *nParts*, int* *pParts*) const;

返回值

如果成功则返回控件中的部分数；否则返回零。

参数

nParts

要获取其坐标的部分数。如果这个参数大于控件中的部分数，则只返回现有部分的坐标。

pParts

一个整数数组的地址，该数组的元素个数与 *nParts* **指定的部分数一样。**

数组中的每一个元素都用来接收相应部分的右边缘的客户区坐标。如果一个元素被设置为-1，则表示该部分的右边缘扩展到了状态条的右边缘。

说明

此成员函数用来获取一个状态条控件中的部分数。这个成员函数也用来获取给定数目的部分的右边缘的坐标。

请参阅 CStatusBarCtrl::GetBorders, CStatusBarCtrl::SetParts

CStatusBarCtrl::GetRect

BOOL GetRect(int *nPane*, LPRECT *lpRect*) const;

返回值

如果成功则返回非零值；否则返回零。

参数

nPane

要获取其边界矩形的部分的从零开始的索引。

lpRect

一个 RECT **结构的地址，该结构接收边界矩形。**

说明

此成员函数用来获取状态条控件中的一个部分的边界矩形。

请参阅 CStatusBarCtrl::GetParts

CStatusBarCtrl::GetText

int GetText(LPCTSTR *lpszText*, int *nPane*, int* *pType*) const;

返回值

文本的用字符数表示的长度。

参数

lpszText

用来接收文本的缓冲的地址。这个参数是一个用空字符结尾的字符串。

nPane

要获取其文本的部分的从零开始的索引。

pType

指向一个用来接收信息类型的整数的指针。这个类型可以是下列值中的一个：

- 0　绘制的文本的一个边界看起来低于状态条平面。
- SBT_NOBORDERS　绘制的文本没有边界。

- SBT_POPOUT　绘制的文本的边界显示比状态条的平面高。
- SBT_OWNERDRAW　如果文本具有 SBT_OWNERDRAW 绘制风格，则 *pType* 接收这个消息并返回与文本关联的 32 位值，而不是长度和操作类型。

说明

此成员函数用来从一个状态条控件的给定部分获取文本。

请参阅　CStatusBarCtrl::SetText, CStatusBarCtrl::GetTextLength

CStatusBarCtrl::GetTextLength

int GetTextLength(int *nPane*, int* *pType*) const;

返回值

返回文本的用字符数表示的长度。

参数

nPane

要获取其文本的部分的从零开始的索引。

pType

指向一个用来接收信息类型的整数的指针。这个类型是下列值中的一个：

- 0　绘制的文本的一个边界看起来低于状态条平面。
- SBT_NOBORDERS　绘制的文本没有边界。
- SBT_OWNERDRAW　文本是父窗口绘制的。
- SBT_POPOUT　绘制的文本的边界显示比状态条的平面高。

说明

此成员函数用来从一个状态条控件中的给定部分获取文本的用字符数表示的长

度。

请参阅 CStatusBarCtrl::GetText, CStatusBarCtrl::SetText

CStatusBarCtrl::GetTipText

CString GetTipText(int *nPane*) const;

返回值

返回一个 CString **对象，该对象包含了要被使用在工具提示中的文本。**

参数

nPane

要获取工具提示文本的状态条窗格的从零开始的索引。

说明

此成员函数用来实现 Win32 **消息** SB_GETTIPTEXT **的行为，就像在“** Platform SDK **”中描述的一样。**

请参阅 CStatusBarCtrl::SetTipText

CStatusBarCtrl::IsSimple

BOOL IsSimple( ) const;

返回值

如果状态窗口控件是在简单模式下则返回非零值；否则返回零。

说明

此成员函数用来实现 Win32 **消息** SB_ISSIMPLE **行为，就像在“** Platform SDK **”**

中描述的一样。

请参阅 CStatusBarCtrl::SetSimple

CStatusBarCtrl::SetBkColor

COLORREF SetBkColor(COLORREF *cr*);

返回值

返回一个代表先前的缺省背景颜色的 COLORREF **值。**

参数

cr

是指定新的背景颜色的 COLORREF **值。如果指定的是** CLR_DEFAULT **值，则会导致状态条使用它的缺省背景颜色。**

说明

此成员函数用来实现 Win32 **消息** SB_SETBKCOLOR **的行为，就像在“** Platform SDK **”中描述的一样。**

CStatusBarCtrl::SetIcon

BOOL SetIcon(int *nPane*, HICON *hIcon*);

返回值

如果成功则返回非零值；否则返回零。

参数

nPane

将用来接收图标的窗格的从零开始的索引。如果此参数是-1**，则状态条**

被假定为是一个简单状态条。

hIcon

是要被设置的图标的句柄。如果这个值是 NULL，则图标被从此部分中移走。

说明

此成员函数用来实现 Win32 消息 SB_SETICON 的行为，就像在“ Platform SDK ”中描述的一样。

CStatusBarCtrl::SetMinHeight

void SetMinHeight(int *nMin*);

参数

nMin

控件的以像素表示的最小高度。

说明

此成员函数用来设置一个状态条控件的绘画区的最小高度。这个最小高度是 *nMin* 和两倍的状态条控件的以像素表示的垂直边界的宽度的和。

请参阅 CStatusBarCtrl::GetRect, CStatusBarCtrl::GetBorders

CStatusBarCtrl::SetParts

BOOL SetParts(int *nParts*, int* *pWidths*);

返回值

如果成功则返回非零值；否则返回零。

参数

nParts

要设置的部分的数目。部分数不能大于 255。

pWidths

一个整数数组的地址，该数组的元素个数与 *nParts* 指定的数目一样。数组中的每一个元素以客户区坐标指定了相应部分的右边缘位置。如果某一个元素是-1，则相应部分的右边缘位置扩展到了该控件的右边缘。

说明

此成员函数用来设置一个状态条控件中的部分数，以及每一部分的右边缘的坐标。

请参阅 CStatusBarCtrl::GetBorders, CStatusBarCtrl::GetParts

CStatusBarCtrl::SetSimple

BOOL SetSimple(BOOL *bSimple* = TRUE);

返回值

如果发生了错误则返回零。

参数

bSimple

显示类型标志。如果这个参数是 TRUE**，则控件显示简单的文本；如果是** FALSE**，则它显示多个部分。**

说明

此成员函数用来指定一个状态条控件是显示简单文本，还是显示前面调用

SetParts 设置的所有控件部分。

如果状态条控件是从非简单模式改变到简单模式，或者是从简单到非简单，控件都会被重画。

请参阅 CStatusBarCtrl::SetParts

CStatusBarCtrl::SetText

BOOL SetText(LPCTSTR *lpszText*, int *nPane*, int *nType*);

返回值

如果成功则返回非零值；否则返回零。

参数

lpszText

一个以空字符结尾的字符串的地址，该字符串指定了要设置的文本。如果 *nType* 是 SBT_OWNERDRAW，则 *lpszText* 表示 32 位的数据。

nPane

nType

绘制操作的类型。这个类型可以是下列值中的一个：

- 0　绘制的文本的一个边界看起来低于状态条平面。
- SBT_NOBORDERS　绘制的文本没有边界。
- SBT_OWNERDRAW　文本是父窗口绘制的。
- SBT_POPOUT　绘制的文本的边界显示比状态条的平面高。

说明

此成员函数用来在一个状态条控件的给定部分中设置文本。当控件下一次接收到 WM_PAINT 消息时，这是表示控件的该部分被改变的消息，导致该部分显

示新的文本。

请参阅 CStatusBarCtrl::GetText, CStatusBarCtrl::GetTextLength

CStatusBarCtrl::SetTipText

void SetTipText(int *nPane*, LPCTSTR *pszTipText*);

参数

nPane

要接收工具提示文本的状态条窗格的从零开始的索引。

pszTipText

一个指向包含工具提示文本的字符串的指针。

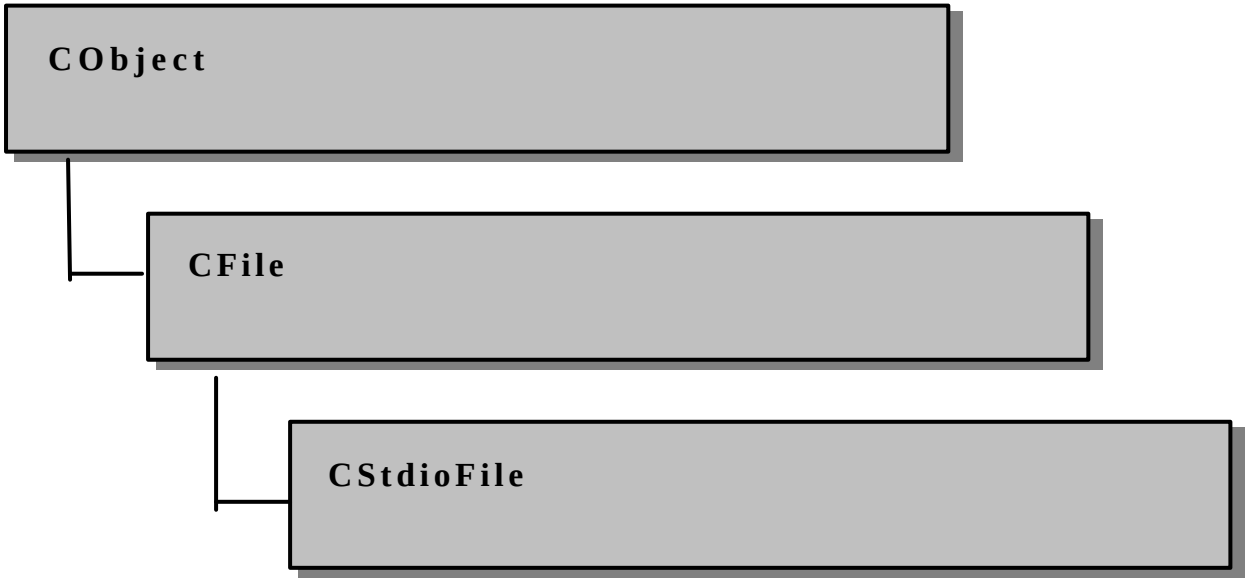
说明

此成员函数用来实现 Win32 消息 SB_SETTIPTEXT 的行为，就像在“ Platform

SDK”中描述的一样。

请参阅 CStatusBarCtrl::GetTipText

CStdioFile



一个 CStdioFile 对象代表一个用运行时函数 fopen 打开的 C 运行时流式文件。

流式文件是被缓冲的，而且可以以文本方式（缺省）或二进制方式打开。

文本方式提供对硬回车—换行符对的特殊处理。当你将一个换行符（0x0A）写入一个文本方式的 CStdioFile 对象时，字节对（0x0A，0x0D）被发送给该文件。当你读一个文件时，字节对（0x0A，0x0D）被翻译为一个字节（0x0A）。

CStdioFile 不支持 Duplicate，LockRange，和 UnlockRange 这几个 CFile 函数。

如果在 CStdioFile 中调用了这几个函数，将会出现 CNoSupported 异常。

有关使用 CStdioFile 的更多信息，参见“Visual C++程序员指南”中的文章“MFC 文件”，和“Microsoft Visual C++库参考”中的“Microsoft Visual C++ 6.0 运行库参考”。

#include <afx.h>

请参阅 CFile, CFile::Duplicate, CFile::LockRange, CFile::UnlockRange, CNotSupportedException

CStdioFile 类成员

Data Members	
m_pStream	包含了一个指向打开文件的指针
Construction	
CstdioFile	从一个路径或文件指针构造一个 CStdioFile 对象
Text Read/Write	
ReadString	读取一行文本
WriteString	写一行文本

成员函数

CStdioFile::CStdioFile

```
CStdioFile( );
CStdioFile( FILE* pOpenStream );
```

CStdioFile(LPCTSTR *lpszFileName*, UINT *nOpenFlags*);
throw(CFileException);

参数

pOpenStream

指定由 C **运行时函数** fopen **调用返回的文件指针。**

lpszFileName

指定表示需要文件的路径的字符串。该路径可以是相对路径，也可以是绝对路径。

nOpenFlags

共享和访问方式。指定当文件打开时要进行的动作。可以通过使用位或 OR（|）**操作符来组合选项。需要一个访问允许和一个文本—二进制标识符；**create **和** noInherit **方式是可选的。参见** CFile::CFile **可以获得方式选项和其它标志的列表。在** MFC 3.0 **及以后版本中，共享标志是允许的。**

说明

此构造函数的缺省版本与 CFile::Open 成员函数联合工作以测试错误。

一个参数的构造函数版本从一个已经打开的文件的指针构造一个 CStdioFile 对象。允许指针值包括预定义的输入/输出文件指针 stdin，stdout，和 stderr。

两个参数的构造函数版本构造一个 CStdioFile 对象并根据给定的路径打开相应的操作系统文件。

如果不能被打开或创建则抛出 CFileException。

示例

```
// CStdioFile::CStdioFile 示例
char* pFileName = "test.dat";
CStdioFile f1;
if( !f1.Open( pFileName, CFile::modeCreate
```

```
      | CFile::modeWrite | CFile::typeText ) ) {
      #ifdef _DEBUG
      afxDump << "Unable to open file" << "\n";
   #endif
   exit( 1 );
}
CStdioFile f2( stdout );
TRY
{
   CStdioFile f3( pFileName,
      CFile::modeCreate | CFile::modeWrite | CFile::typeText );
}
CATCH( CFileException, e )
{
   #ifdef _DEBUG
      afxDump << "File could not be opened "
         << e->m_cause << "\n";
   #endif
}
END_CATCH
```

CStdioFile::ReadString

virtual LPTSTR ReadString(LPTSTR *lpsz*, UINT *nMax*);
throw(CFileException);
BOOL ReadString(CString& *rString*);
throw(CFileException);

返回值

返回一个指向包含文本数据的缓存的指针。如果在到达文件结尾后没有读到任何数据，则返回 NULL**；或者是返回布尔值** FALSE**。**

参数

lpsz

指定一个指向用户提供的缓冲区的指针；该缓冲区将接收一个以空字符结尾的文本字符串。

nMax

指定要读取的最大字符数，不计算结尾的空字符。

rString

是一个对 CString 对象的引用，当函数返回时，该对象将包含了所读取的字符串。

说明

此成员函数用来从与 CStdioFile 对象关联的文件中将文本数据读入一个缓冲区，读取的字符数不超过 *nMax*-1。在碰到第一个换行符时停止读取。在这种情况下，如果读取的字符数小于 *nMax*-1，则将一个换行符保存在缓冲区中。在任何情况下，都在最后添加一个空字符（“0”）。

对于文本方式的输入，CFile::Read 也是可用的，但它不会在一个回车换行符处停止。

注意 这个函数的 CString 版本将“/n”删除（如果有的话）；LPTSTR 版本则不删除。

示例

```
// example for CStdioFile::readString
extern CStdioFile f;
char buf[100];

f.ReadString(buf,99);
```

请参阅 CStdioFile::WriteString, CFile::Read

CStdioFile::WriteString

```
virtual void WriteString( LPCTSTR lpsz );
    throw( CFileException );
```

参数

lpsz

指定一个指向存放了以空字符结尾的文本字符串的缓冲区的指针。

说明

此成员函数将一个缓冲区中的数据写入与 CStdioFile **对象关联的文件中。结束的空字符（“** \0 **”）不被写入该文件。***lpsz* **中的所有换行符都被以一个硬回车—换行符对写入该文件。**

在响应某些条件（如磁盘已满条件）时， WirteString **抛出一个异常。**

这是一个面向文本的写函数，它对 CStdioFile **及其后代，以及** CArchive **来说是可用的。** CFile::Write **也是可用的，但是它不是在一个空字符处终止，而是将要求的字节数写入文件中。**

示例

```
// example for CStdioFile::WriteString
extern C StdioFile f;
char buf [ ] = “test string”;

f.WriteString(buf);
```

请参阅 CArchive::ReadString, CFile::Write

数据成员

CStdioFile::m_pStream

说明

m_pStream **数据成员是指向一个打开文件的指针，该文件是由** C **运行时函数**

fopen 返回的。如果文件从来没有被打开过或者已经被关闭了，则它是 NULL。

CString

CString 没有基类。

一个 CString 对象由可变长度的一队字符组成。CString 使用类似于 Basic 的语法提供函数和操作符。连接和比较操作符以及简化的内存管理使 CString 对象比普通字符串数组容易使用。

CString 是基于 TCHAR 数据类型的对象。如果在你的程序中定义了符号 _UNICODE，则 TCHAR 被定义为类型 wchar_t，即 16 位字符类型；否则，TCHAR 被定义为 char，即 8 位字符类型。在 UNICODE 方式下，CString 对象由 16 位

字符组成。非 UNICODE 方式下，CString 对象由 8 位字符组成。

当不使用_UNICODE 时，CString 是多字节字符集（MBCS，也被认为是双字节字符集，DBCS）。注意，对于 MBCS 字符串，CString 仍然基于 8 位字符来计算，返回，以及处理字符串，并且你的应用程序必须自己解释 MBCS 的开始和结束字节。

CString 对象还具有下列特征：

- CString 可作为连接操作的结果而增大。
- CString 对象遵循“值语义”。应将 CString 看作是一个真实的字符串而不是指向字符串的指针。
- 你可以使用 CString 对象任意替换 Coust char*和 LPCTSTR 函数参数。
- 转换操作符使得直接访问该字符串的字符就像访问一个只读字符（C-风格的字符）数组一样。

提示　如果可能的话，应在框架中而不是堆中分配这个 CString 对象。这可以节省内存并简化参数的传递。

CString 允许两个具有相同值的字符串共享相同的缓冲空间，这有助于你节省内存空间。但是，如果你初始直接改变该缓冲的内容（不使用 MFC），则有可能在无意中改变了两个字符串。CString 提供了两个成员函数 CString::LockBuffer 和 CString::UnlockBuffer 来帮助你保护你的数据。当你调用 LockBuffer 时，你就创建了一个字符串的一个拷贝，然后将引用计数设置为-1，这就“加锁”了该缓冲区。当缓冲区被加锁时，就没有其它的字符串可以引用该字符串中的数据，被加锁的字符串也不能引用其它字符串的数据。通过加锁该缓冲区内的字符串，就可以保证该字符串对数据的持续独占。当你使用完数据后，调用 UnlockBuffer 来将引用计数恢复为 1。

更多的信息，参见“Visual C++程序员指南”中的“MFC 字符串”和“字符串：

Unicode 和多字节字符集（MBCS）支持”，以及“ Microsoft Visual C++库参考”中的“ Microsoft Visual C++ 6.0 运行库参考”。

#include <afx.h>

请参阅 在“ Visual C++ Programmer's 指南”中的：字符串：基本的 CString 操作，

字符串：CString 语义，字符串：CString 与 C-Style Strings 相关的操作，

字符串：CString 异常清除，字符串：CString 参数传递

CString 类成员

Construction	
CString	以各种方法构造一个 CString 对象
The String as an Array	
GetLength	返回 CString 对象中的字符数。对多字节字符，按 8 位字符计算；即在一个多字节字符中一个开始和结束字节算作两个字符
IsEmpty	测试一个 CString 对象中是否不含有字符
Empty	强制一个字符串的长度为 0
GetAt	返回在给定位置的字符
Operator[]	返回在给定位置的字符---它是代替 GetAt 的操作符
SetAt	设置给定位置上的字符
Operator LPCTSTR	象访问一个 C 风格的字符串一样，直接访问保存在一个 CString 对象中的字符

Assignment/Concatenation	
Operator =	给 CString 对象赋一个新值
Operator +	连接两个字符串并返回一个新字符串
Operator +=	把一个新字符串连接到一个已经存在的字符串的末端

Comparison	
Operator == <, etc.	比较操作符（大小写敏感）
Compare	比较两个字符串（大小写敏感）
CompareNoCase	比较两个字符串（不区分大小写）
Collate	比较两个字符串（大小写敏感，使用现场特别信息）
CollateNoCase	比较两个字符串（不区分大小写，使用现场特别信息）

Extraction	
Mid	提取一个字符串的中间一部分（类似于 BASIC 的 MID$函数）
Left	提取一个字符串的左边一部分（类似于 BASIC 的 LEFT$函数）
Right	提取一个字符串的右边一部分（类似于 BASIC 的 RIGHT$函数）
SpanIncluding	提取一个字符串，该子字符串中仅含有某一字符集合中的字符。
SpanExcluding	提取一个字符串，该子字符串中不含有某一字符集合中的字符。

Other Conversions	
MakeUpper	将字符串中的所有字符转换为大写字符
MakeLower	将字符串中的所有字符转换为小写字符
MakeReverse	将字符串中的字符以倒序排列
Replace	用其它字符替换指定的字符
Remove	从一个字符串中移走指定的字符
Insert	在字符串中的给定索引处插入一个字符或一个子字符串
Delete	从一个字符串中删除一个或多个字符
Format	像 sprintf 函数一样格式化该字符串
FormatV	像 vprintf 函数一样格式化该字符串
TrimLeft	将字符串中前面的空格整理出字符串
TrimRight	将字符串中结尾的空格整理出字符串
FormatMessage	格式化一个消息字符串

Searching	
Find	在一个较大的字符串中查找字符或子字符串
ReverseFind	在一个较大的字符串中从末端开始查找某个字符
FindOneOf	查找与某个字符集合中的字符相匹配的第一个字符

Archive/Dump	
Operator <<	把一个 CString 对象插入一个存档或转储的环境中
Operator >>	从一个存档中提取一个 CString 对象

Buffer Access	
GetBuffer	返回一个指向 CString 对象的指针
GetBufferSetLength	返回一个指向 CString 对象的指针，字符串被截断为指定的长度
ReleaseBuffer	释放对 GetBuffer 所返回的缓冲区的控制权
FreeExtra	通过释放原先为此字符串分配的额外内存来删除此字符串对象的额外开销
LockBuffer	使引用计数无效并保护缓冲区内的数据
UnlockBuffer	使引用计数有效并释放缓冲区中的数据

Windows-Specific	
AllocSysString	由 CString 数据分配一个 BSTR
SetSysString	使用 CString 对象中的数据来设置一个已存在的 BSTR
LoadString	从一个 Windows 资源中加载一个已经存在的 CString 对象
AnsiToOem	实现由 ANSI 字符集到 OEM 字符集的对应转换
OemToAnsi	实现由 OEM 字符集到 ANSI 字符集的对应转换

成员函数

CString::AllocSysString

```
BSTR AllocSysString ( ) const;
   throw( CMemoryException );
```

返回值

指向新分配的字符串。

说明

此成员函数分配一个 BSTR 类型的新的 OLE 的兼容 Automation 字符串，并把包括空结束符在内的此 CString 对象的内容拷贝到其中，若内存不足，则发出 CMemoryException 异常。这个函数通常是用来为 OLE 自动化返回字符串。

在很少的情况下，你需要使用::SysFreeString 来重新分配此函数返回的字符串的存储单元。

注意　如果你使用的是一个共享库的 MFC 并且是在调试模式下生成的，则你必须与 MFCO42D.DLL 连接来获得这个函数。通过在 STDAFX.H 文件中

包括<afxdisp.h>，可以使连接程序自动获得 MFC042D.DLL。

有关 Windows 中的 OLE 分配函数的更多信息，参见“Win32 SDK OLE 程序员参考”中的::SysAllocString 和::SysFreeString。

请参阅 ::SysAllocString, ::SysFreeString, CMemoryException

CString::AnsiTiOem

void AnsiToOem( );

说明

此成员函数将 CString 对象中的所有字符由 ANSI 字符集转换为 OEM 字符集。参见“Microsoft Visual C++6.0 参考库”中的“Microsoft Visual C++6.0 语言参考”卷中的“ANSI 字符代码表”。

请参阅 CString::OemToAnsi

CString::Collate

int Collate(LPCTSTR *lpsz*) const;

返回值

如果两个字符串是一样的，则返回非零值；如果 CString **对象小于** *lpsz* **则返回值**<0**；如果此** CString **对象大于** *lpsz* **则返回值**>0**。**

参数

lpsz

要用于比较的另一个字符串。

说明

此成员函数通过使用通用文本函数_tcscoll来比较这个CString对象和另一个字符串。此通用文本函数_tcscoll是在TCHAR.H中定义的，根据在编译时设置的字符来与strcoll，wcscoll，或_mbscoll对应。这些函数的每一个都根据当前使用的代码页来进行一次区分大小写的比较。更多的信息，参见“Microsoft Visual C++ 6.0运行时库参考”中的strcat，wcscat，_mbscat。

请参阅 CString::Compare, CString::CompareNoCase

CString::CollateNoCase

int CollateNocase(LPCTSTR *lpsz*) const;

返回值

如果字符串是一样的（不区分大小写）则返回非零值；如果 CString 对象小于 *lpsz*（不区分大小写）则返回值<0，如果 CString 对象大于 *lpsz*（不区分大小写）则返回值>0。

参数

lpsz

指向要用于比较的另一个字符串的指针。

说明

此成员函数通过使用通用文本函数_tcscoll 来比较这个 CString 对象和另一个字符串。此通用文本函数_tcscoll 是在 TCHAR.H 中定义的，根据在编译时设置的字符来与 stricoll，wcsicoll，或_mbsicoll 对应。这些函数的每一个都根据当前

使用的代码页来进行一次不区分大小写的比较。

请参阅 CString::Collate, CString::CompareNoCase, CString::Compare

CString::Compare

int Compare(LPCTSTR *lpsz*) const;

返回值

如果字符串是一样的则返回非零值；如果 CString 对象小于 *lpsz* 则返回值<0，如果 CString 对象大于 *lpsz* 则返回值>0。

参数

lpsz

要用于比较的另一个字符串。

说明

此成员函数通过使用通用文本函数_tcscmp来比较这个CString对象和另一个字符串。此通用文本函数_tcscmp是在TCHAR.H中定义的，根据在编译时设置的字符来与strcmp，wcscmp，或_mbscmp对应。这些函数的每一个都根据当前使用的代码页来进行一次区分大小写的比较，而且不会被现场影响。更多的信息，参见“Microsoft Visual C++ 6.0运行库参考”中的strcmp，wcscmp，_mbscmp。

示例

下面的例子说明了如何使用CString::Compare。

```
// CString::Compare 示例
CString s1( "abc" );
CString s2( "abd" );
ASSERT( s1.Compare( s2 ) == -1 ); // 与另一个 CString 比较。
ASSERT( s1.Compare( "abe" ) == -1 ); // 与 LPTSTR 字符串比较。
```

请参阅 CString::CompareNoCase,CString::Collate,CString::CollateNoCase

CString::CompareNoCase

int CompareNoCase(LPCTSTR lpsz) const;

返回值

如果字符串是一样的（不区分大小写）则返回非零值；如果 CString **对象小于** *lpsz* **（不区分大小写）则返回值**<0**，如果** CString **对象大于** lpsz**（不区分大小写）则返回值**>0**。**

说明

此成员函数通过使用通用文本函数_tcsicmp **来比较这个** CString **对象和另一个字符串。此通用文本函数**_tcsicmp **是在** TCHAR.H **中定义的，根据在编译时设**

置的字符来与_stricmp，_wcsicmp，或_mbsicmp 对应。这些函数的每一个都根据当前使用的代码页来进行一次区分大小写的比较，而且不会被现场影响。更多的信息，参见“Microsoft Visual C++ 6.0 运行库参考”中的_stricmp，_wcsicmp，_mbsicmp。

示例

下面的例子说明了如何使用 CString::CompareNoCase。

```
// CString::CompareNoCase 示例
CString s1( "abc" );
CString s2( "ABD" );
ASSERT( s1.CompareNoCase( s2 ) == -1 ); // 与一个 CString 比较。
ASSERT( s1.Compare( "ABE" ) == -1 ); // 与 LPTSTR 字符串比较。
```

请参阅 CString::Compare, CString::Collate,Cstring::CollateNoCase

CString::CString

```
CString( );
CString( const CString& stringSrc );
   throw( CMemoryException );
CString( TCHAR ch, int nRepeat = 1 );
   throw( CMemoryException );
CString( LPCTSTR lpch, int nLength );
   throw( CMemoryException );
CString( const unsigned char* psz );
   throw( CMemoryException );
CString( LPCWSTR lpsz );
   throw( CMemoryException );
CString( LPCSTR lpsz );
   throw( CMemoryException );
```

参数

stringSrc

一个已经存在的 CString **对象，它要被拷贝到此** CString **对象中。**

ch

要被重复 *nRepeat* 次的单个字符。

nRepeat

要对 *ch* 重复的次数。

lpch

一个指向字符数组的指针，该字符数组的长度是 *nLength*，不包括结尾的空字符。

nLength

pch 中的字符的个数。

psz

一个要被拷贝到此 CString 对象中去的以空字符结尾的字符串。

lpsz

一个要被拷贝到此 CString 对象中去的以空字符结尾的字符串。

说明

这些构造函数的每一个都用来以指定的数据初始化一个新的 CString **对象。**

由于构造函数是将输入数据拷贝到新分配的存储区，所以你应该注意可能会导致的内存异常。注意，这些构造函数中的某些函数的作用相当于一个转换函数。这就允许你使用替换物，例如在一个需要 CString **对象的地方用一个** LPTSTR **来代替。**

此构造函数的某几种形式具有特殊的目的：

- CString(LPCSTR *lpsz*) **从一个** ANSI **字符串构造一个** Unicode CString**。你可以象下面的例子那样用这个函数来加载一个字符串资源。**
- CString(LPCWSTR *lpsz*) **从一个** Unicode **字符串构造一个** CString**。**
- CString(const unsigned char* *psz*) **从一个指向** unsigned char **的指针构造一**

个 CString。

想了解更多的信息，请参阅“ Visual C++ 程序员指南”随机文中的“ String:CString Exception Cleanup”一节。

示例

下面的例子说明了如何使用 CString::CString。

```
// CString::CString 示例
CString s1;                                    // 空字符串
CString s2( "cat" );                           // 从一个文字的 C 字符串
CString s3 = s2;                               // 拷贝构造函数
CString s4( s2 + " " + s3 );                   // 从一个字符串表达式
CString s5( 'x' );                             // s5 = "x"
CString s6( 'x', 6 );                        // s6 = "xxxxxx"
CString s7((LPCSTR)ID_FILE_NEW);    // s7 = "Create a new document"
CString city = "Philadelphia";               // 不是赋值操作符
```

请参阅 CString::operator =

CString::Delete

```
int Delete( int nIndex, int nCount = 1);
   throw( CMemoryException ) ;
```

返回值

返回已改变的字符串的长度。

参数

nIndex

要删除的第一个字符的索引。

nCount

要删除的字符数。

说明

此成员函数用来从一个字符串中从 nIndex 开始的地方删除一个或多个字符。如果 nCount 比此字符串还要长，则字符串的其余部分都将被删除。

示例

```
// 下面的例子说明了如何使用 CString::Delete。
   str2 = “Hockey is best!”;
   printf( “Before: %s\n, (LPCTSTR) str2 );
   int n = str2.Delete(6, 3 );
   printf( “After: %s/n”, (LPCTSTR) str2 );
   ASSERT( n == str2.Getlength ( ) );
// 这些代码产生下面的删除行：

Before: Hockey is best!
After: Hockey best!
```

请参阅 CString::Insert

CString::Empty

void Empty( );

说明

此成员函数用来使 CString **对象成为一个空字符串，并释放相应的内存。**

更多的信息，参见“ Visual C++**程序员指南”中的“字符串：**CString **异常清除”。**

示例

下面的例子说明了如何使用 CString::Empty**。**

```
// CString::Empty 示例
CString s( "abc" );
s.Empty();
```

```
ASSERT( s.GetLength( ) == 0 );
```

请参阅 CString::IsEmpty

CString::Find

```
int Find( TCHAR ch ) const;
int Find( LPCTSTR lpszSub ) const;
int Find( TCHAR ch, int nStart ) const;
int Find( LPCTSTR lpszSub, int nStart ) const;
```

返回值

返回此 CString 对象中与需要的子字符串或字符匹配的第一个字符的从零开始的索引；如果没有找到子字符串或字符则返回-1。

参数

ch

要搜索的单个字符。

lpszSub

要搜索的子字符串。

nStart

字符串中开始搜索的字符的索引，如果是 0，则是从头开始搜索。如果 nStart 不是 0，则位于 nStart 处的字符不包括在搜索之内。

pstr

指向要搜索的字符串的指针。

说明

此成员函数用来在此字符串中搜索子字符串的第一个匹配的字符。函数的重载可以接收单个字符（类似于运行时函数 strchr）和字符串（类似于 strstr）。

示例

//下面演示第一个例子

```
// CString::Find（TCHAR ch）
CString s( "abcdef" );
ASSERT( s.Find( 'c' ) == 2 );
ASSERT( s.Find( "de" ) == 3 );
```

// 下面演示第二个例子

```
// Cstring::Find(TCHAR ch,int nStart)
Cstring str(“The stars are aligned”);
Ing n = str.Find('e',5);
ASSERT(n == 12)
```

请参阅 CString::ReverseFind, CString::FindOneOf

CString::FindOneOf

int FindOneOf(LPCTSTR *lpszCharSet*) const;

返回值

返回此字符串中第一个在 *lpszCharSet* 中也包括的字符的从零开始的索引。

参数

lpszCharSet

包含用于匹配字符的字符串。

说明

此成员函数在此字符串中搜索与 *lpszCharSet* 中任意字符匹配的第一个字符。

示例

下面的例子说明了如何使用 CString::FindOneOf。

// CString::FindOneOf 示例

```
CString s( "abcdef" );
ASSERT( s.FindOneOf( "xd" ) == 3 ); // 'd' is first match.
```

请参阅 CString::Find

CString::Format

```
void Format( LPCTSTR lpszFormat, ... );
void Format( UINT nFormatID, ... );
```

参数

lpszFormat

一个格式控制字符串。

nFormatID

包含格式控制字符串的字符串资源标识符。

说明

此成员函数用来将格式化数据写入一个 CString 中，其方法就像 sprintf 函数向一个 C-风格的字符数组中格式化输出数据一样。这个成员函数在 CString 中格式化并存储一系列字符和值。根据 *lpszFormat* 中指定的格式或 *nFormatID* 标识的字符串资源，函数中的每一个可选参数（如果有）都被转换并输出。

如果此字符串对象本身是作为 Format 的一个参数，则调用将失败。例如象下面的代码：

```
CString str = “Some Data”;
str.Format(“%s%d”,str, 123);   //注意：在参数列表中也使用了 str。
```

将导致不可预期的结果。

当你传递一个字符串作为一个可选择的参数时，你必须显式地将它转换为 LPCTSTR。这个格式与 printf 函数中的格式参数具有相同的形式和函数。（有

关格式和参数的描述，参见“ Microsofr Visual C++ 6.0 运行库参考”中的 printf。）

在被写的字符串结尾将添加一个空字符。

更多的信息，参见“ Microsofr Visual C++ 6.0 运行时库参考”中的 sprintf。

请参阅　CString::GetBuffer,CString::FormatV

CString::FormatV

void FormatV(LPCTSTR *lpszFormat*, va_list *argList*);

参数

lpszFormat

一个格式控制字符串。

argList

要传递的参数列表。

说明

此成员函数用来将格式化数据和一个参数变量列表写入一个 CString 中，其方法就像 vsprintf 函数向一个 C-风格的字符数组中格式化输出数据一样。这个成员函数在 CString 中格式化并存储一系列字符和值。根据 *lpszFormat* 中指定的格式，函数中的每一个可选参数都被转换并输出。

如果此字符串对象本身是作为 FormatV 的一个次数，则调用将失败。例如象下面的代码：

```
CString str = “Some Data”;
str.FormatV(“%s%d”,str, 123);    //注意：在次数列表中也使用了 str。
```

将导致不可预期的结果。

更多的信息，参见“ Microsoft Visual C++ 6.0 运行库参考 ”中的 vsprintf。

示例

```
//  使用 CString::FormatV，你可以像下面这样来编写函数：

void WriteLogEntry( CStdioFile& refFile, LPCTSTR pstrFormat, ... )
{
   CTime timeWrite;
   timeWrite = CTime:GetCurrentTime( );

   //  输出时间
   CString str = timeWrite.Format ( "%d %b %y %H:%M:%S – ");
   refFile.Write( str, str.GetLength( ));

   //  格式化并输出给定的数据
   va_list args;
   va_start( args, pstrFormat );
   str.FormatV( pstrFormat, args );
   refFile.Write( str, str.GetLength( ) );

   //  插入一个换行符
   refFile.Write( "\n", 1 );
   return;
}
```

你可以用任意个数的参数来调用上面的函数，例如：

```
WriteLogEntry( fileLog , “Program started”);
WriteLogEntry( fileLog , “Processed %d bytes” 91341 );
WriteLogEntry( fileLog , “%d error(s) found in %d line(s)”, 10, 1351 );
WriteLogEntry( fileLog , “Program completed”);
```

上面的例子将把输出添加到你的 fileLog **文件，就像下面给出的一样：**

```
17 Apr 97 12:34:53 – Program started
17 Apr 97 12:34:59 – Processed 91341 bytes
17 Apr 97 12:35:22 – 10 error(s) found in 1351 line(s)
17 Apr 97 12:35:23 – Program completed
```

请参阅 CString::Format, va_start

CString::FormatMessage

void FormatMessage(LPCTSTR *lpszFormat*, ...);
void FormatMessage(UINT *nFormatID*, ...);

参数

lpszFormat

指向格式控制字符串。它将被扫描来进行插入，从而进行格式化。这个格式字符串类似于运行时函数 pritf-style **格式字符串，除了这个字符串允许参数可以按任意顺序插入。**

nFormatID

包含还没有格式化的消息文本的字符串资源标识符。

说明

此成员函数用来格式化一个消息字符串。此函数需要一个消息定义作为输入。该消息定义是由 *lpszFormat* **决定的，或者是来自由** *nFormatID* **标识的字符串资源。函数将这个格式化的消息文本拷贝到** CString **中，如果需要的话就处理任何嵌入的插入顺序。**

每一次插入都必须有相应的参数跟随 *lpszFormat* 或 *nFormatID* 参数。在消息文本中，为了动态格式化消息，要支持几个转义序列。有关这些转义序列的描述和它们的意义，参见“Win32 SDK 程序员参考”中的 Windows::FormatMessage 函数。

请参阅 ::FormatMessage,CString::LoadString,CString::Format

CString::FreeExtra

void FreeExtra( );

说明

此成员函数用来释放原先被此字符串分配的但不再使用的额外的内存。这将减小此字符串的内存的额外开销。此函数按 GetLength 返回的准确长度来重新分配缓冲区。

CString::GetAt

TCHAR GetAt(int *nIndex*) const;

返回值

返回一个包含字符串中的指定位置的字符串的 TCHAR**。**

参数

nIndex

CString **对象中的某个字符的从零开始的索引。***nIndex* **参数必须大于或等于** 0**，并小于** GetLength **函数的返回值。**Microsoft **基础类库的测试版验证** *nIndex* **的边界；但是发行版不验证。**

说明

你可以把一个 CString **对象看作是一个字符数组。** GetAt **成员函数返回一个由一个索引号指定的单个字符。重载的下标（**[]**）是** GetAt **常用的代用符。**

示例

下面的例子说明了如何使用 CString::GetAt**。**

```
// CString::GetAt 示例
CString s( "abcdef" );
ASSERT( s.GetAt(2) == 'c' );
```

请参阅 CString::GetAt, CString::GetLength, CString::operator []

CString::GetBuffer

```
LPTSTR GetBuffer( int nMinBufLength );
```

```
throw( CMemoryException );
```

返回值

一个指向对象的（以空字符结尾的）字符缓冲区的 LPTSTR **指针。**

参数

nMinBufLength

字符缓冲区的以字符数表示的最小容量。这个值不包括一个结尾的空字符的空间。

说明

此成员函数返回一个指向 CString **对象的内部字符缓冲区的指针。返回的** LPTSTR **不是** const**，因此可以允许直接修改** CString **的内容。**

如果你使用由 GetBuffer 返回的指针来改变字符串的内容，你必须在使用其它的 CString 成员函数之前调用 ReleaseBuffer 函数。

在调用 ReleaseBuffer 之后，由 GetBuffer 返回的地址也许就无效了，因为其它的 CString 操作可能会导致 CString 缓冲区被重新分配。如果你没有改变此 CString 的长度，则缓冲区不会被重新分配。

当此 CString 对象被销毁时，其缓冲区内存将被自动释放。

注意，如果你自己知道字符串的长度，则你不应该添加结尾的空字符。但是，当你用 ReleaseBuffer 来释放该缓冲区时，你必须指定最后的字符串长度。如果你添加了结尾的空字符，你应该给 ReleaseBuffer 的长度参数传递 -1，ReleaseBuffer 将对该缓冲区执行 strlen 来确定它的长度。

示例

下面的例子说明了如何用 CString::GetBuffer。

```
// CString::GetBuffer 例子
CString s( "abcd" );
#ifdef _DEBUG
afxDump << "CString s " << s << "\n";
#endif
LPTSTR p = s.GetBuffer( 10 );
strcpy( p, "Hello" );      // 直接访问 CString 对象。
s.ReleaseBuffer( );
#ifdef _DEBUG
afxDump << "CString s " << s << "\n";
#endif
```

请参阅 CString::GetBufferSetLength, CString::ReleaseBuffer

CString::GetBufferSetLength

LPTSTR GetBufferSetLength(int *nNewLength*);

throw(CMemoryException);

返回值

返回一个指向对象的（以空字符结尾的）字符缓冲区的 LPTSTR **指针。**

参数

nNewLength

此 CString **字符缓冲区的以字符数表示的精确容量。**

说明

此成员函数返回一个指向 CString **对象的内部字符缓冲区的指针，如果需要的话，切断或增长缓冲区的长度以精确匹配由** *nNewLength* **指定的长度。返回的** LPTSTR **不是** const**，因此可以允许直接修改** CString **的内容。**

如果你使用由 GetBuffer 返回的指针来改变字符串的内容，你必须在使用其它的 CString 成员函数之前调用 ReleaseBuffer 函数。

在调用 ReleaseBuffer 之后，由 GetBuffer 返回的地址也许就无效了，因为其它的 CString 操作可能会导致 CString 缓冲区被重新分配。如果你没有改变此 CString 的长度，则缓冲区不会被重新分配。

当此 CString 对象被销毁时，其缓冲区内存将被自动释放。

注意，如果你自己知道字符串的长度，则你不应该添加结尾的空字符。但是，当你用 ReleaseBuffer 来释放该缓冲区时，你必须指定最后的字符串长度。如果你添加了结尾的空字符，你应该给 ReleaseBuffer 的长度参数传递 -1，ReleaseBuffer 将对该缓冲区执行 strlen 来确定它的长度。

有关引用计数的更多信息，参见下面的文章：

- “Win 32 SDK 程序员参考”中的“通过引用计数来管理对象的生命周期”。
- “Win 32 SDK 程序员参考”中的“实现引用计数”。
- “Win 32 SDK 程序员参考”中的“管理引用计数的规则”。

请参阅 CString::GetBuffer, CString::ReleaseBuffer

CString::GetLength

int GetLength( ) const;

返回值

返回字符串中的字节计数。

说明

此成员函数用来获取这个 CString **对象中的字节计数。这个计数不包括结尾的**

空字符。

对于多字节字符集（MBCS），GetLength 按每一个 8 位字符计数；即，在一个多字节字符中的开始和结尾字节被算作两个字节。

示例

下面的例子说明了如何使用 CString::GetLength。

```
// CString::GetLength 示例
CString s( "abcdef" );
ASSERT( s.GetLength() == 6 );
```

请参阅　CString::IsEmpty

CString::Insert

```
int Insert( int nIndex, TCHAR ch );
   throw( CMemoryException );
```

```
int Insert( int nIndex, LPCTSTR pstr );
   throw( CMemoryException );
```

返回值

返回被改变的字符串的长度。

参数

nIndex

某个字符的索引，在这个字符的前面将要进行插入操作。

ch

要插入的字符。

pstr

一个指向要被插入的子字符串的指针。

说明

此成员函数用来在字符串中的给定索引处插入一个单个字符或一个子字符串。nIndex 参数标识了将要被移走以给字符或子串空间的第一个字符。如果 *nIndex* 是零，则插入将方式在这个字符串之前。如果 *nIndex* 大于字符串的长度，则函数将把当前的字符串与由 *ch* 或 *pstr* 指定的新字符串连接起来。

示例

```
//  下面的例子说明了如何使用 CString::Insert。
   CString str( “HockeyBest”);
   int n = str.Insert( 6, “is” );
   ASSERT( n == str.GetLength( ) );
   printf( “1: %s\n”, ( LPCTSTR ) str );
   n = str.Insert( 6, ' ');
   ASSERT( n == str.GetLength( ) );
   printf ( “2: %s\n”, (LPCTSTR) STR );

   n = str.Insert(555, ‘1’);
```

```
ASSERT( n == str.GetLength ( ) );
printf ( “3: %s\n”, ( LPCTSTR ) str );
```

// 上面的代码将产生下面这些输出行：

1. Hockeyis Best
2. Hockey is Best
3. Hockey is Best!

请参阅 CString::Delete, CString::operator +

CString::IsEmpty

```
BOOL IsEmpty( ) const;
```

返回值

如果 CString **对象的长度为** 0，**则返回非零值；否则返回** 0。

说明

此成员函数用来测试一个 CString **对象是否是空的。**

示例

下面的例子说明了如何使用 CString::IsEmpty**。**

```
// CString::IsEmpty 示例
CString s;
ASSERT( s.IsEmpty() );
```

请参阅 CString::GetLength

CString::Left

```
CString Left( int nCount ) const;
   throw( CMemoryException );
```

返回值

返回一个包含指定范围字符的拷贝的 CString 对象。注意，返回的 CString 对象可能是空的。

参数

nCount

要从这个 CString 对象中提取的字符数。

说明

此成员函数用来从此 CString 对象中提取最前面的 *nCount* 个字符，并返回被提取的子字符串的拷贝。如果 *nCount* 超过了字符串的长度，则整个字符串都被提取。Left 类似与 Basic LEFT$函数（除了索引是从零开始的）。

对于多字节字符集（MBCS），*nCount* 指的是每 8 位字符的个数，即，在一个

多字节字符中开始和结尾字节被算作两个字符。

示例

下面的例子说明了如何使用 CString::Left。

```
// CString::Left 示例
CString s( _T("abcdef") );
ASSERT( s.Left(2) == _T("ab") );
```

请参阅 CString::Mid, CString::Right

CString::LoadString

```
BOOL LoadString( UINT nID );
   throw( CMemoryException );
```

返回值

如果加载资源成功则返回非零值；否则返回 0。

参数

nID

一个 Windows 字符串资源 ID。

说明

此成员函数用来读取一个由 *nID* 标识的 Windows 字符串资源，并放入一个已有的 CString 对象中。

示例

下面的例子说明了如何使用 CString::LoadString。

```
// CString::LoadString 示例
#define IDS_FILENOTFOUND 1
CString s;
if (! s.LoadString( IDS_FILENOTFOUND ))
```

```
{
   AfxMessageBox("Error Loading String: IDS_FILENOTFOUND");
   ...
}
```

CString::LockBuffer

```
LPTSTR LockBuffer( );
```

返回值

返回一个指向 CString 对象的指针，或者是一个以 NULL 结尾的字符串。

说明

此成员函数用来加锁缓冲区内的一个字符串。

通过调用 LockBuffer，可以创建一个字符串的拷贝，然后将引用计数设置为-1。当引用计数被设置为-1 时，缓冲区中的字符串被认为是处于“加锁”状态。当

该字符串处于加锁状态时，字符串被从两个方面得到保护：

- 没有其它的字符串能够获得对此加锁字符串中的数据的引用，即使是该字符串被赋予了加锁字符串。
- 加锁字符串将不能引用其它的字符串，即使另一个字符串被拷贝到该加锁字符串中。

通过加锁缓冲区中的字符串，可以保证该字符串对缓冲区的独占保持完整。

在你完成了对 LockBuffer 的使用之后，调用 UnlockBuffer 来将该引用计数恢复到 1。

有关引用计数的更多信息，参见下面的文章：

- “Win 32 SDK 程序员参考”中的“通过引用计数来管理对象的生命周期”。
- “Win 32 SDK 程序员参考”中的“实现引用计数”。

- “Win 32 SDK 程序员参考”中的“管理引用计数的规则”。

请参阅 CString::UnlockBuffer, CString::GetBuffer, CString::ReleaseBuffer

CString::MakeLower

void MakeLower( );

说明

此成员函数将此 CString **对象转换为一个小写字符串。**

示例

下面的例子说明了如何使用 CString::MakeLower**。**

```
// CString::MakeLower 示例
CString s( "ABC" );
s.MakeLower();
```

```
ASSERT( s == "abc" );
```

请参阅 CString::MakeUpper

CString::MakeReverse

```
void MakeReverse( );
```

说明

此成员函数将此 CString **对象中的字符的顺序颠倒过来。**

示例

下面的例子说明了如何使用 CString::MakeReverse**。**

```
// CString::MakeReverse 示例
CString s( "abc" );
s.MakeReverse();
```

```
ASSERT( s == "cba" );
```

CString::MakeUpper

```
void MakeUpper( );
```

说明

此成员函数将此 CString **对象转换为一个大写字符串。**

示例

下面的例子说明了如何使用 CString::MakeUpper**。**

```
// CString::MakeUpper 示例
CString s( "abc" );
s.MakeUpper();
ASSERT( s == "ABC" );
```

请参阅 CString::MakeLower

CString::Mid

```
CString Mid( int nFirst ) const;
   throw( CMemoryException );
CString Mid( int nFirst, int nCount ) const;
   throw( CMemoryException );
```

返回值

返回一个包含指定范围字符的拷贝的 CString **对象。注意，这个返回的** CString **对象可能是空的。**

参数

nFirst

此 CString **对象中的要被提取的子串的第一个字符的从零开始的索引。**

nCount

要从此 CString **对象中提取的字符数。如果没有提供这个参数，则字符串**

的其余部分都被提取。

说明

此成员函数从此 CString 对象中提取一个长度为 *nCount* 个字符的子串，从 *nFirst*（从零开始的索引）指定的位置开始。此函数返回一个对所提取的字符串的拷贝。Mid 类似于 Basic MID$函数（除了索引是从零开始的）。

对于多字节字符集（MBCS），*nCount* 指的是每 8 位字符的数目；也就是说，在一个多字节字符中开始和结尾字节被算作两个字符。

示例

下面的例子说明了如果如何使用 CString::Mid。

```
// CString::Mid 示例
CString s( _T("abcdef") );
```

```
ASSERT( s.Mid( 2, 3 ) == _T("cde") );
```

请参阅 CString::Left, CString::Right

CString::OemToAnsi

```
void OemToAnsi( );
```

说明

此成员函数将此 CString **对象中的所有字符由** OEM **字符集转换成** ANSI **字符集。参见“** Microsoft Visual C++ 6.0 **库参考”中的“** Microsoft Visual C++ 6.0 **语言参考手册”卷中的“** ANSI **字符代码表”。**

如果定义了_UNICODE**，则这个函数是无效的。**

请参阅 CString::AnsiToOem

CString::ReleaseBuffer

void ReleaseBuffer(int *nNewLength* = -1);

参数

nNewLength

此字符串的以字符数表示的新长度，不计算结尾的空字符。如果这个字符串是以空字符结尾的，则参数的缺省值－1 将把 CString **的大小设置为字符串的当前长度。**

说明

使用 ReleaseBuffer **来结束对由** GetBuffer **分配的缓冲区的使用。如果你知道缓冲区中的字符串是以空字符结尾的，则可以省略** *nNewLength* **参数。如果字符串不是以空字符结尾的，则可以使用** *nNewLength* **指定字符串的长度。在调用**

ReleaseBuffer 或其它 CString 操作之后，由 GetBuffer 返回的地址是无效的。

示例

下面的例子说明了如何使用 CString::ReleaseBuffer。

```
// CString::ReleaseBuffer 示例
CString s;
s = "abc";
LPTSTR p = s.GetBuffer( 1024 );
strcpy(p, "abc");       // 直接使用该缓冲区
ASSERT( s.GetLength() == 3 ); // 字符串长度 = 3
s.ReleaseBuffer();   // 释放多余的内存，现在 p 无效。
ASSERT( s.GetLength() == 3 ); // 长度仍然是 3
```

请参阅 CString::GetBuffer

CString::Remove

int CString::Remove (TCHAR ch);

返回值

返回从字符串中移走的字符数。如果字符串没有改变则返回零。

参数

ch

要从一个字符串中移走的字符。

说明

此成员函数用来将 *ch* 实例从字符串中移走。与这个字符的比较是区分大小写的。

示例

```
// 从一个句子中移走小写字母'c':
```

```
CString str (“This is a test.”);
int n = str.Remove( 't' );
ASSERT( n == 2 );
ASSERT( str ==“This is a es.” );
```

请参阅 CString::Replace

CString::Replace

int Replace(TCHAR *chOld*, TCHAR *chNew*);
int Replace(LPCTSTR *lpszOld*, LPCTSTR *lpszNew*);

返回值

返回被替换的字符数。如果这个字符串没有改变则返回零。

参数

chOld

要被 *chNew* **替换的字符。**

chNew

要用来替换 *chOld* 的字符。

lpszOld

一个指向字符串的指针，该字符串包含了要被 *lpszNew* 替换的字符。

lpszNew

一个指向字符串的指针，该字符串包含了要用来替换 *lpszOld* 的字符。

说明

此成员函数用一个字符替换另一个字符。函数的第一个原形在字符串中用 *chNew* 现场替换 *chOld*。函数的第二个原形用 *lpszNew* 指定的字符串替换 *lpszOld* 指定的子串。

在替换之后，该字符串有可能增长或缩短；那是因为 *lpszNew* 和 *lpszOld* 的长度不需要是相等的。两种版本形式都进行区分大小写的匹配。

示例

```
//  第一个例子，old 和 new 具有相同的长度。

CString strZap( “C - -” );
int n = strZap.Replace('-', '+' );
ASSERT( n == 2 );
ASSERT(strZap == “C++” );

//  第二个例子，old 和 new 具有不同的长度。

CString strBang( “Everybody likes ice hockey” );
n = strBang.Replace( “hockey”, “golf” );
ASSERT( n ==1 );
n = strBang.Replace ( “likes” , “plays” );
ASSERT( n == 1 );
n = strBang.Replace( “ice”, NULL );
ASSERT( n == 1 );
ASSERT( strBang == “Everybody plays golg” );

//  注意，现在在你的句子中有了一个额外的空格。
//  要移走这个额外的空格，可以将它包括在要被替换的字符串中，例如，“ ice ”。
```

请参阅　CString::Remove

CString::ReverseFind

int ReverseFind(TCHAR ch) const;

返回值

返回此 CString **对象中与要求的字符匹配的最后一个字符的索引；如果没有找到需要的字符则返回**-1。

参数

ch

要搜索的字符。

说明

此成员函数在此 CString **对象中搜索与一个子串匹配的最后一个字符。此函数**

类似于运行时函数 strrchr。

示例

```
//    CString::ReverseFind 示例
CString s( "abcabc" );
ASSERT( s.ReverseFind( 'b' ) == 4 );
```

请参阅　CString::Find, CString::FindOneOf

CString::Right

```
CString Right( int nCount ) const;
   throw( CMemoryException );
```

返回值

返回一个包含指定字符拷贝的 CString **对象。注意，这个返回的** CString **对象可能是空的。**

参数

nCount

要从这个 CString **对象中提取的字符数目。**

说明

此成员函数用来从此 CString **对象中提取最后（最右边）的** *nCount* **个字符，并返回此提取字符串的一个拷贝。如果** *nCount* **超过了字符串的长度，则提取整个字符串。**Right **类似于** Basic **的** RIGHT$**函数（除了索引是从零开始的）。**

对于多字节字符集（MBCS**），** *nCount* **指的是每** 8 **位字符的数目；也就是说，在一个多字节字符中开始和结尾字节被算作两个字符。**

示例

下面的例子说明了如何使用 CString::Right。

```
//    CString::Right 示例
CString s( _T("abcdef") );
ASSERT( s.Right(2) == _T("ef") );
```

请参阅　CString::Mid, CString::Left

CString:: SetAt

void SetAt(int *nIndex*, TCHAR *ch*);

参数

nIndex

CString **对象中的某个字符的从零开始的索引。***nIndex* **参数必须大于或等于** 0，**小于由** Getlength **的返回的值。**Microsoft **基础类库的测试版本将检**

验 *nIndex* 的边界，而 Release 版本则不检验。

ch

要插入的字符。

说明

你可以将一个 CString 对象看作是一个字符数组。SetAt 成员函数重写由一个索引值指定的单个字符。如果索引超出了已有字符串的边界，SetAt 不会扩大这个字符串。

请参阅 CString::GetAt, CString::operator [ ]

CString::SetSysString

BSTR SetSysString(BSTR* *pbstr*) const;

返回值

返回新的字符串。

参数

pbstr

一个指向字符串的指针。

说明

此成员函数重新分配由 *pbstr* **所指向的** BSTR**，并将包括结尾的空字符在内的** CString **对象中的内容拷贝到该** BSTR **中。由** *pbstr* **所引用的** BSTR **的值可能会发生变化。如果内存不足，此函数将抛出一个** CMemoryException **异常。**

这个函数通常被用来为 OLE **自动化改变通过引用传递的字符串的值。**

有关 Windows 中的 OLE 重分配函数的更多信息，参见“Win32 SDK OLE 程序员参考”中的::SysReallocStringLen 和::SysFreeString。

CString::SpanExcluding

```
CString SpanExcluding( LPCTSTR lpszCharSet ) const;
   throw( CMemoryException );
```

返回值

返回一个子字符串，该子串保存了在 CString 字符串中但又不在 *lpszCharSet* 字符集合中的字符，该子串以字符串的第一个字符作为开始，以在此字符串中发现的第一个也在 lpszCharSet 字符集中字符前面的字符作为结尾（也就是说，从此字符串的第一个字符开始，直到第一个也包括在 *lpszCharSet* 字符集的字符为止，但不包括这个也在 *lpszCharSet* 字符集中的字符）。如果 CString 中的每一

个字符都不在 *lpszCharSet* 字符集合中，则返回整个字符串。

参数

lpszCharSet

一个被解释为字符集的字符串。

说明

此成员函数用来查找整个 CString 对象中的字符串，找出 *lpszCharSet* 指定的字符集中任何字符第一次出现的位置。SpanExcluding 提取并返回这个 *lpszCharSet* 字符集中的字符第一次出现之前的所有字符（换句话说，来自 *lpszCharSet* 字符集的字符以及在它之后的字符串都不会被返回）。如果在此字符串中没有发现来自 *lpszCharSet* 字符集的字符，则 SpanExcluding 返回整个字符串。

示例

下面的函数返回 src **参数的第一个部分。**

```
//  由一个分号（；），一个逗号(,)，一个句号(.)，一个破折号(-)，
//  或一个冒号(:)划定界线的部分。

CString GetFirstPart( CString src)
{
   return src.SpanExcluding( ";,.- :");
}
```

请参阅 CString::SpanIncluding

CString::SpanIncluding

```
CString SpanIncluding( LPCTSTR lpszCharSet ) const;
   throw( CMemoryException );
```

返回值

返回此字符串中的一个子字符串，该子串中的字符也在 *lpszCharSet* 字符集中字符集中。该子串开始于字符串中的第一个字符，当发现第一个不是 *lpszCharSet* 字符集中的字符时结束。如果字符串中的第一个字符就不是指定字符集中的字符，则 SpanIncluding 返回一个空的子字符串。

参数

lpszCharSet

一个被解释为字符集的字符串。

说明

此成员函数用来从字符串中提取字符，提取的子字符串开始于字符串中的第一个字符，其字符都是由 *lpszCharSet* 指定的字符集中的。如果字符串的第一个字

符就不在此字符集中，则 SpanIncluding 返回一个空的字符串。否则，它返回一个连续字符序列，它们都是该字符集中的字符。

示例

下面的例子说明了如何使用 CString::SpanIncluding。

```
// CString::SpanIncluding 示例
CString str( "cabbage" );
CString res = str.SpanIncluding( "abc" );
ASSERT( res == "cabba" );
res = str.SpanIncluding( "xyz" );
ASSERT( res.IsEmpty( ) );
```

请参阅 CString::SpanExcluding

CString::TrimLeft

```
void TrimLeft( );
void CString::TrimLeft( TCHAR chTarget );
```

void CString::TrimLeft(LPCTSTR *lpszTargets*);

参数

chTarget

要被整理的目标字符。

lpszTargets

指向一个字符串的指针，该字符串包含了要被整理的目标字符。

说明

这个成员函数的没有参数的版本用来将字符串最前面的空格修整掉。当在没有参数的情况下调用时，TrimLeft **删除换行符，空格和** tab **字符。**

这个成员函数的需要参数的版本用来将一个特定的字符或一群特定的字符从字符串的开始处删除。

参见 TrimRight 可以获得一个代码例子。更多的信息，参见“Visual C++程序员指南”中的“字符串主题”。

请参阅 CString::Mid, CString::Left, CString::Right, CString::MakeUpper, CString::MakeLower, CString::MakeReverse, CString::Format

CString::TrimRight

```
void TrimRight( );
void CString::TrimRight( TCHAR chTarget );
void CString::TrimRight( LPCTSTR lpszTargets );
```

参数

chTarget

要被整理的目标字符。

lpszTargets

一个指向字符串的指针，该字符串中包含了要被整理的目标字符。

说明

这个成员函数的没有参数的版本用来将字符串最后面的空格修整掉。当在没有参数的情况下调用时，TrimRight 从字符串中删除换行符，空格和 tab 字符。

这个成员函数的需要参数的版本用来将一个特定的字符或一群特定的字符从字符串的结尾处删除。

示例

```
CString strBefore;
CString strAfter;

   strBefore ="Hockey is Best!!!!" ;
   strAfter = strBefore;
   str.TrimRight('!' );
   printf ("Before: \"%s\"\n",(LPCTSTR) strBefore );
```

```
printf ("After: \"%s\"\n",(LPCTSTR) strBefore );

strBefore ="Hockey is Best?!?!?!?!" ;
strAfter = strBefore;
str.TrimRight( '?!' );
printf ("Before: \"%s\"\n",(LPCTSTR) strBefore );
printf ( "After: \"%s\"\n",(LPCTSTR) strAfter );
```

在上面的第一个例子中，字符串“ Hockey is Best!!!! **”变成了“** Hockey is Best **”。**

在上面的第二个例子中，字符串“ Hockey is Best?!?!?!?! **”变成了“** Hockey is Best **”。**

更多的信息，参见“ Visual C++**程序员指南”中的“字符串主题”。**

请参阅 CString::TrimLeft, CString::Mid, CString::Left, CString::Right, CString::MakeUpper, CString::MakeLower, CString::MakeReverse, CString::Format

CString::UnlockBuffer

void UnlockBuffer( );

说明

此成员函数用来解锁先前通过调用 LockBuffer **加锁的的缓冲区。**UnlockBuffer **将引用计数恢复为** 1**。**

CString **析构函数使用** UnlockBuffer **来保证当析构函数被调用时缓冲区不会被加锁。**

请参阅 CString::LockBuffer, CString::GetBuffer, CString::ReleaseBuffer

操作符

CString::operator =

```
const CString& operator =( const CString& stringSrc );
    throw( CMemoryException );
const CString& operator =( TCHAR ch );
    throw( CMemoryException );
const CString& operator =( const unsigned char* psz );
    throw( CMemoryException );
const CString& operator =( LPCWSTR lpsz );
    throw( CMemoryException );
const CString& operator =( LPCSTR lpsz );
    throw( CMemoryException );
```

说明

这个 CString **赋值操作符（**=**）用新的数据来初始化一个已经存在的** CString **对**

象。如果目标字符串（即左边的字符串）已经具有足够大的空间来保存新数据，则不用进行新的内存分配。不管你什么时候使用这个赋值操作符，都要小心可能出现的内存异常，因为常常要为保存新的结果给 CString 对象分配内存。

示例

下面的例子说明了如何使用 CString::operator =。

```
// CString::operator =示例
CString s1, s2;              // 空的 CString 对象

s1 = "cat"                   // s1 = "cat"
s2 = s1;                     // s1 and s2 each = "cat"
s1 = "the " + s1;            // 或者是表达式
s1 = 'x';                    // 或者只是单个的字符
```

请参阅 CString::CString

CString::operator LPCTSTR

operator LPCTSTR ( ) const;

返回值

返回一个指向字符串的数据的字符指针。

说明

这个非常有用的强制操作符提供了一种非常有效的方法来访问存放在 CString **对象中的以空字符结尾的** C **字符串。此函数不拷贝字符而仅返回一个指针。如果你在获取指针之后修改此** CString **对象，则可能导致内存被重新分配，并使该指针无效。**

CString::operator <<, >>

```
friend CArchive& operator <<( CArchive& ar, const CString& string );
      throw( CArchiveException );
friend CArchive& operator >>( CArchive& ar, CString& string );
      throw( CArchiveException );
friend CDumpContext& operator <<( CDumpContext& dc, const CString& string );
```

说明

CString **插入操作符（<<）支持向一个存档进行诊断转储和存储。提取操作符支持从一个文档中进行装入。**

CDumpContext **操作符只有在** Microsoft **基础类库的调试版中才有效。**

示例

下面的例子说明了如何使用 CString::operator <<, >>**。**

```
// CString::operator <<, >>示例
   extern CArchive ar;
   CString s( "abc" );
#ifdef _DEBUG
   afxDump << s;    // Prints the value (abc)
   afxDump << &s;    // Prints the address
#endif

   if( ar.IsLoading() )
      ar >> s;
   else
      ar << s;
```

请参阅 CDumpContext

CString::operator +

```
friend CString operator +( const CString& string1, const CString& string2 );
      throw( CMemoryException );
friend CString operator +( const CString& string, TCHAR ch );
      throw( CMemoryException );
friend CString operator +( TCHAR ch, const CString& string );
```

```
    throw( CMemoryException );
friend CString operator +( const CString& string, LPCTSTR lpsz );
    throw( CMemoryException );
friend CString operator +( LPCTSTR lpsz, const CString& string );
throw( CMemoryException );
```

返回值

返回一个 CString **对象，该对象是此连接操作的临时结果。这个返回值使得在一个表达式中组合多次连接成为可能。**

参数

string，*string1*，*string2*

要连接的 CString **对象。**

ch

将被连接到一个字符串之后的或将在其后连接一个字符串的字符。

lpsz

指向一个以空字符结尾的字符串的指针。

说明

此+操作符将两个字符串连接起来并返回一个 CString 对象。两个参数中的一个必须是一个 CString 对象，而另一个可以是指向字符的指针或是一个字符。不管你什么时候使用这个连接操作符，都要小心可能出现的内存异常，因为为了保存这个临时结果常常要分配新内存。

示例

下面的例子说明了如何使用 CString::operator +。

```
// CString::operator +示例
CString s1( "abc" );
CString s2( "def" );
```

```
ASSERT( (s1 + s2 ) == "abcdef" );
CString s3;
s3 = CString( "abc" ) + "def" ; // Correct
s3 = "abc" + "def";
// 错了！第一个参数必须是一个 CString。
```

请参阅 CString::operator +=

CString::oeprator +=

```
const CString& operator +=( const CString& string );
      throw( CMemoryException );
const CString& operator +=( TCHAR ch );
      throw( CMemoryException );
const CString& operator +=( LPCTSTR lpsz );
      throw( CMemoryException );
```

参数

string

将连接到此字符串之后的 CString **对象。**

ch

将连接到此字符串之后的一个字符。

lpsz

指向将连接到此字符串之后的的字符串的指针。

说明

此+=连接操作符将字符连接到此 CString 字符串之后。+=操作符可以接受另一个 CString 对象，一个指向字符的指针或一个字符作为参数。不管你什么时候使用这个连接操作符，都要小心可能出现的内存异常，因为常常要为添加到这个 CString 对象的字符分配内存。

示例

下面的例子说明了如何使用 CString::operator +=。

```
// CString::operator +=示例
CString s( "abc" );
ASSERT( ( s += "def" ) == "abcdef" );
```

请参阅 CString::operator +

CString Comparison Operators

```
BOOL operator ==( const CString& s1, const CString& s2 );
BOOL operator ==( const CString& s1, LPCTSTR s2 );
BOOL operator ==( LPCTSTR s1, const CString& s2 );
BOOL operator !=( const CString& s1, const CString& s2 );
BOOL operator !=( const CString& s1, LPCTSTR s2 );
BOOL operator !=( LPCTSTR s1, const CString& s2 );
BOOL operator <( const CString& s1, const CString& s2 );
BOOL operator <( const CString& s1, LPCTSTR s2 );
BOOL operator <( LPCTSTR s1, const CString& s2 );
BOOL operator >( const CString& s1, const CString& s2 );
BOOL operator >( const CString& s1, LPCTSTR s2 );
BOOL operator >( LPCTSTR s1, const CString& s2 );
BOOL operator <=( const CString& s1, const CString& s2 );
BOOL operator <=( const CString& s1, LPCTSTR s2 );
```

BOOL operator <=(LPCTSTR *s1*, const CString& *s2*);
BOOL operator >=(const CString& *s1*, const CString& *s2*);
BOOL operator >=(const CString& *s1*, LPCTSTR *s2*);
BOOL operator >=(LPCTSTR *s1*, const CString& *s2*);

返回值

如果字符串满足比较条件则返回非零值；否则返回 0。

参数

s1, *s2*

要比较的 CString **对象。**

说明

这些比较操作符用来比较两个字符串。它们是区分大小写的 Compare **成员函数的方便的代用符。**

示例

```
//  CString 比较操作符示例
CString s1( "abc" );
CString s2( "abd" );
ASSERT( s1 < s2 ); // Operator is overloaded for both.
ASSERT( "ABC" < s1 ); // CString and char*
ASSERT( s2 > "abe" );
```

CString::operator [ ]

TCHAR operator [](int *nIndex*) const;

参数

nIndex

字符串中的一个字符的从零开始的索引。

说明

你可以将一个 CString 对象看作是一个字符数组。此重载下标操作符（[]）返回由 nIndex 中的从零开始的索引值指定的一个字符。此操作符是 GetAt 成员函数的惯用的代用符。

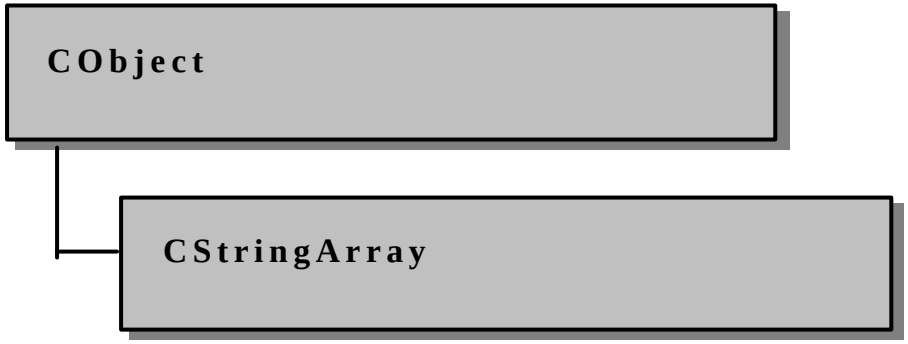
示例

下面的例子说明了如何使用 CString::operator [ ]操作符。

```
//    CString::operator [ ]示例
CString s( "abc" );
ASSERT( s[1] == 'b' );
```

请参阅　CString::GetAt, CString::SetAt

CStringArray



CStringArray 类支持 CString 对象数组。

CStringArray 的成员函数类似于 CObArray 类的成员函数。由于具有这些相似性，你可以参考关于 CObArray 的参考文件来获取 CStringArray 成员函数的详细说明。如果在说明中看到某一函数的返回值是一个指向 CObject 对象的指针，则可以用一个 CString（而不是一个 CString 指针）来代替它。如果看到某一函数的参数是一个指向 CObject 的指针，则可以用 LPCTSTR 来代替它。

例如，可以将

CObject* CObArray::GetAt(int <nIndex>) const;

转换为

CString CStringArray::GetAt(int <nIndex>) const;

和将

void SetAt(int <nIndex>, CObject* <newElement>)

转换为

void SetAt(int <nIndex>, LPCTSTR <newElement>)。

CStringArray 与 IMPLEMENT_SERIAL 宏联合起来支持其元素的连续和转储。如果一个 CString 对象数组被用一个重载的插入操作符或 Serialize 成员函数保存到一个存档中，则它的每一个元素都按顺序连续。

注意　在使用一个数组之前，使用 SetSize 来建立它的大小并给它分配

内存。如果你不使用 SetSize，则向数组中添加元素将导致数组被频繁地拷贝和分配内存。频繁分配内存和拷贝会导致效率低和内存零碎。

如果你需要数组中个别字符串元素的转储，则应该将转储环境的深度设置为 1 或更大。当一个 CString 数组被删除时，或当其中的个别元素被删除时，字符串内存被根据需要释放。

有关使用 CStringArray 的更多信息，参见“Visual C++程序员指南”中的文章“集合”。

```
#include <afxcoll.h>
```

CStringArray 类成员

Construction	
CStringArray	构造一个空的 CString 对象数组

Bounds	
GetSize	获取这个数组中的元素数目
GetUpperBound	返回最大的有效索引
SetSize	设置这个数组中包含的元素数目

Operations	
FreeExtra	释放当前数组边界之外的未使用的所有内存
RemoveAll	从数组中删除所有元素

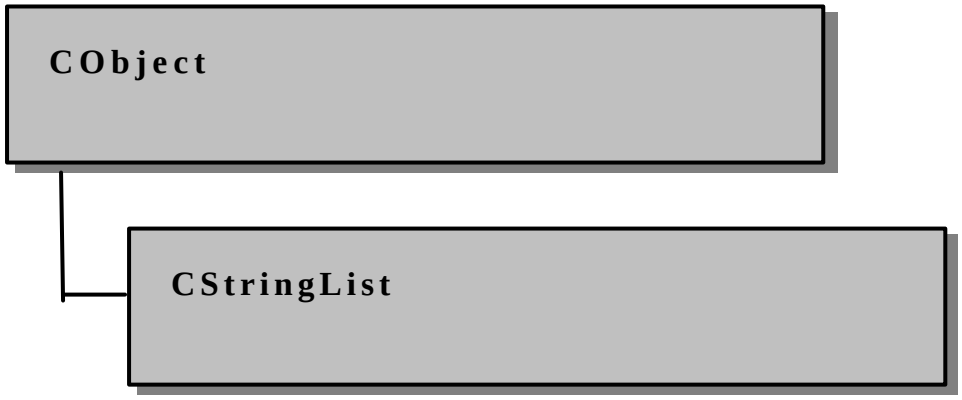
Element Access	
GetAt	返回位于给定索引处的值
SetAt	设置给定索引处的元素的值；不得将数组增大
ElementAt	返回对数组中的某一元素指针的临时引用
GetData	对数组中的元素允许的访问。可以是 NULL

Growing the Array	
SetAtGrow	设置给定索引处的值，如果必要的话可以增长数组
Add	在数组的末尾添加一个元素；可根据需要增长数组
Append	向数组中添加另一个数组；如果必要的话可增长数组
Copy	将另一个数组拷贝到此数组中；如果必要的话可增长数组

Insertion/Removal	
InsertAt	在指定索引处插入一个元素（或者是另一个数组中的所有元素）
RemoveAt	删除指定索引处的一个元素

Operators	
operator	设置或获取在指定索引处的元素

CStringList



CStringList 类支持 CString 对象的列表。所有的比较都是通过值比较来完成的，这意味着不是比较字符串的地址而是比较字符串中的字符。

CStringList 的成员函数类似于类 CObList 类的成员函数。由于具有这些相似性，你可以参考关于 CObList 的参考文件来获取 CStringList 成员函数的详细说明。如果在说明中看到某一函数的返回值是一个指向 CObject 对象的指针，则可以用一个 CString（而不是一个 CString 指针）来代替它。如果看到某一函数的参

数是一个指向 CObject 的指针，则可以用 LPCTSTR 来代替它。

例如，可以将

```
CObject*& CObList::GetHead() const;
```

转换为

```
CString& CStringList::GetHead() const;
```

和将

```
POSITION AddHead( CObject* <newElement> );
```

转换为

```
POSITION AddHead( LPCTSTR <newElement> );
```

CStringList 与 IMPLEMENT_SERIAL 宏联合起来支持其元素的连续和转储。如果一个 CString 对象列表被用一个重载的插入操作符或 Serialize 成员函数保存到一个存档中，则它的每一个元素都按顺序连续。

如果你需要数组中个别字符串元素的转储，则应该将转储环境的深度设置为 1 或更大。

当一个 CStringList 对象被删除时，或当它的元素被删除时，则相应的 CString 对象被删除。

有关使用 CStringList 的更多信息，参见“Visual C++程序员指南”中的文章“集合”。

#include <afxcoll.h>

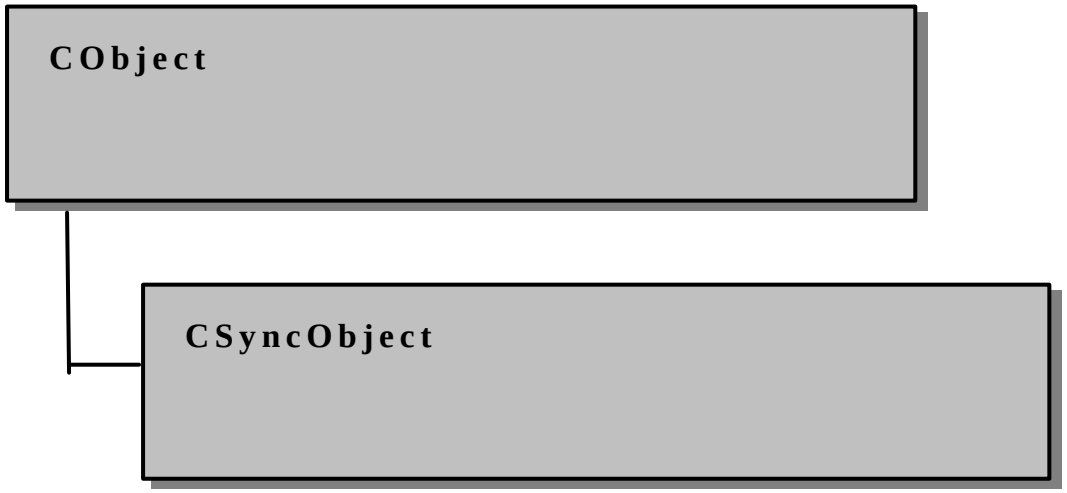
CStringList 类成员

Construction	
CstringList	构造一个空的 CString 对象列表

Head/Tail Access	
GetHead	返回此列表（不能是空的）中头部的元素
GetTail	返回此列表（不能是空的）中尾部的元素
Operations	
RemoveHead	从列表的头部删除元素
RemoveTail	从列表的尾部删除元素
AddHead	在列表的头部添加一个元素（或者是另一个列表中的所有元素），即产生一个新的头部
AddTail	在列表的尾部添加一个元素（或者是另一个列表中的所有元素），即产生一个新的尾部
RemoveAll	删除此列表中的所有元素
Iteration	
GetHeadPosition	返回列表中的头部元素的位置
GetTailPosition	返回列表中尾部元素的位置
GetNext	获取用于重复的下一个元素
GetPrev	获取用于重复的前一个元素

Retrieval/Modification	
GetAt	获取给定位置处的元素
SetAt	设置给定位置处的元素
RemoveAt	从此列表中删除由位置指定的一个元素
Insertion	
InsertBefore	在一个给定位置之前插入一个新元素
InsertAfter	在一个给定位置之后插入一个新元素
Searching	
Find	获取由字符串值指定的元素的位置
FindIndex	获取由一个从零开始的索引指定的元素的位置
Status	
GetCount	返回此列表中的元素个数
IsEmpty	测试列表是否为空（没有元素）

CSyncObject



CSyncObject 类是一个纯虚类，它提供了对于 Win32 中的同步对象来说通用的性能。Microsoft 基础类库提供了几个从 CSyncObject 派生的类。这些类是 CEvent，CMutex，CCriticalSection，和 CSemaphore。

有关如何使用同步对象的信息，请参见“Visual C++程序员指南”中的文章“多线程：如何使用同步类”。

```
#include <afxmt.h>
```

CSyncObject 类对象

Construction	
CsyncObject	构造一个 CSyncObject 对象
Methods	
Lock	获得对该同步对象的访问
Unlock	释放对该同步对象的访问
Attributes	
OperatorHANDLE	获得对该同步对象的访问

成员函数

CSyncObject:: CSyncObject

```
CSyncObject( LPCTSTR pstrName );
virtual ~CSyncObject( );
```

参数

pstrName

对象的名字。如果是 NULL，则 pstrName 将是空。

说明

此成员函数用来构造一个具有给定名字的同步对象。

CSyncObject::Lock

virtual BOOL Lock(DWORD *dwTimeout* = INFINITE);

返回值

如果函数成功则返回非零值；否则返回 0。

参数

dwTimeout

指定等待此同步对象变为可用（发信号的）的时间量。如果是 INFINITE，**则** Lock **一直等待直到对象在返回前变为有效。**

说明

此成员函数用来获得对该同步对象控制的资源的访问。如果该同步对象是可用

的，则 Lock 将成功地返回，并且线程从现在起拥有了这个对象。如果同步对象不可用，则 Lock 将在 *dwTimeOut* 参数指定的毫秒数内等待此同步对象变为可用。如果这个同步对象在指定量的时间内没有变为可用的，则 Lock 返回失败。

CSyncObject::operator HANDLE

operator HANDLE( ) const;

返回值

如果成功，则返回此同步对象的句柄；否则返回 NULL。

说明

此操作符用来获取此 CSyncObject 对象的句柄。你可以用这个句柄来直接调用

Windows API。

CSyncObject::Unlock

```
virtual BOOL Unlock( ) = 0;
virtual BOOL Unlock( LONG lCount, LPLONG lpPrevCount = NULL );
```

返回值

此函数的缺省实现总是返回 TRUE。

参数

lCount

缺省实现不使用这个参数。

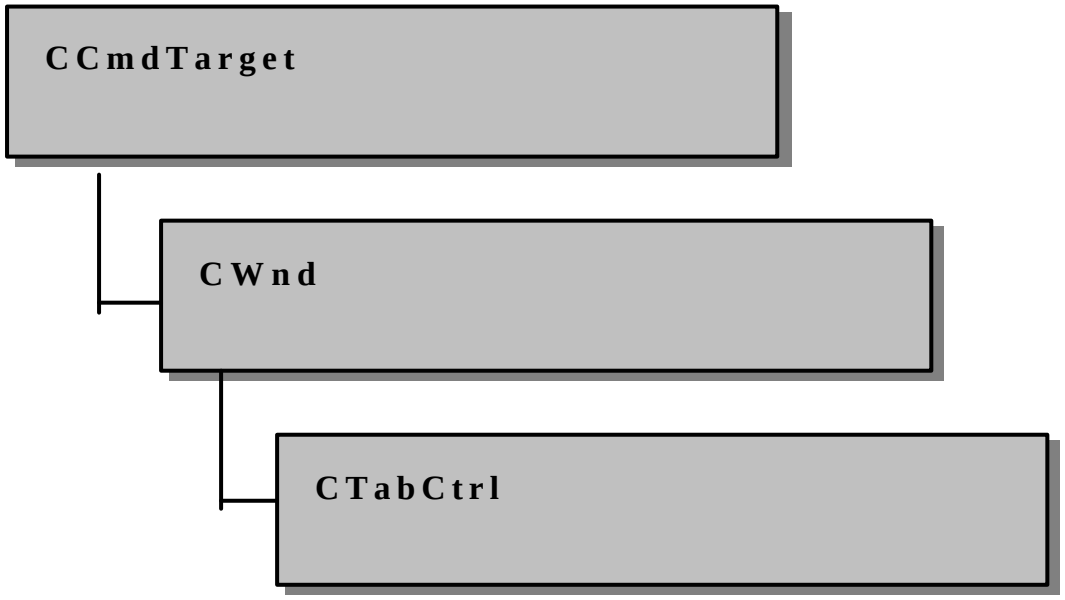
lpPrevCount

缺省实现不使用这个参数。

说明

Unlock 的不带参数的声明是一个纯虚函数，所有从 CSyncObject 派生的类都必须重载它。带有两个参数的缺省实现总是返回 TRUE。这个成员函数用来释放由调用线程所拥有的同步对象的访问。第二个声明是提供给像信号（semaphores）那样的同步对象的，这些对象允许对一个被控制资源进行多个访问。

CTabCtrl



一个“标签控件”类似于一个笔记本中的分隔器，或一个文件柜上的标签。通过使用标签控件，应用程序可以将一个窗口或对话框的相同区域定义为多个页

面。每一页包含了一套信息或一组控件，当用户选择了相应的标签时应用程序就会显示相应的信息或控件。一种特殊类型的标签控件把标签显示得像按钮一样。点击一个按钮将立即执行一条命令而不是显示一个页。

CTabCtrl 类提供了 Windows 通用标签控件的性能。这个控件（也就是 CTabCtrl 类）只对运行在 Windows 95 和 Windows NT 3.51 或更新版本下的程序来说是可用的。

有关使用 CTabCtrl 的更多信息，参见“Visual C++程序员指南”中的“控件主题”和“使用 CTabCtrl”。

#include <afxcmn.h>

请参阅 CHeaderCtrl, CListCtrl

CTabCtrl 类成员

Construction	
CtabCtrl	构造一个 CTabCtrl 对象
Create	创建一个标签控件并将它与一个 CTabCtrl 对象连接

Attributes	
GetImageList	获取与一个标签控件相关的图像列表
SetImageList	将一个图像列表分配给一个标签控件
GetItemCount	获取此标签控件中的标签的数目
GetItem	获取此标签控件中的某一个标签的信息
SetItemExtra	设置一个标签控件中的每一个标签为应用程序定义的数据所保留的字节数
GetItemRect	获取一个标签控件中的一个标签的边界矩形
GetCurSel	确定在一个标签控件中当前选择的标签
SetCurSel	在一个标签控件中选择一个标签
SetCurFocus	将焦点设置到一个标签控件中的指定标签上
SetItemSize	设置某个项的宽度和高度

续表

SetPadding	设置一个标签控件中的每一个标签的图标和标签周围的空间（填料）
GetRowCount	获取一个标签控件中的标签的当前行数
GetToolTips	获取与一个标签控件相关联的工具提示控件的句柄
SetToolTips	将一个工具提示控件赋给一个标签控件
GetCurFocus	获取一个标签控件的具有当前焦点的标签
SetMinTabWidth	设置一个标签控件中的项的最小宽度
GetExtendedStyle	获取标签控件当前使用的扩展风格
SetExTendedStyle	设置一个标签控件的扩展风格
GetItemState	获取指定标签控件项的状态
SetItemState	设置指定标签控件项的状态

Operations	
InsertItem	在一个标签控件中插入一个新的标签
DeleteItem	从一个标签控件中删除一项
DeleteAllItems	从一个标签控件中删除所有的项
AdjustRect	根据一个给定的窗口矩形来估算一个标签控件的的显示区域，或根据一个给定的显示区域来估算与之对应的窗口矩形
RemoveImage	从一个标签控件的图像列表中删除一个图像
HitTest	确定哪一个标签（如果有的话）位于指定的屏幕位置
DeselectAll	重新设置一个标签控件中的项，清除任何被按下的项
HighlightItem	设置一个标签项的加亮状态

Overridables	
DrawItem	绘制一个标签控件的指定项

成员函数

CTabCtrl::AdjustRect

void AdjustRect(BOOL bLarger, LPRECT lpRect);

参数

bLarger

指明要执行的选项。如果这个参数是 TRUE**，则** *lpRect* **指定一个显示矩形并接收相应的窗口矩形。如果这个参数是** FALSE**，则** *lpRect* **指定一个窗口矩形并接收相应的显示矩形。**

lpRect

指向一个 RECT **结构的指针，该结构指定给定的矩形并接收计算后的矩形。**

说明

此成员函数根据给定的窗口矩形计算一个标签控件的显示区域，或根据一个给定的显示区域计算相应的窗口矩形。

请参阅 CTabCtrl::SetItemSize, CTabCtrl::GetItemRect, CTabCtrl::AdjustRect

CTabCtrl::Create

BOOL Create(DWORD *dwStyle*, const RECT& *rect*, CWnd* *pParentWnd*, UINT *nID*);

返回值

如果成功地初始化了对象则返回 TRUE**；否则返回** FALSE**。**

参数

dwStyle

指定标签控件的风格。标签控件风格的任意组合都适用于这个控件。

rect

指定控件的尺寸和位置。它可以是一个 CRect **对象或一个** RECT **结构。**

pParentWnd

指定该标签控件的父窗口，通常是一个 CDialog**。它不能是** NULL**。**

nID

指定标签控件的 ID**。**

说明

分两步构造一个 CTabCtrl **对象。首先调用构造函数，然后调用** Create **函数来创建标签控件并将它与此** CTabCtrl **对象连接。**

下面的风格适用于一个标签控件：

- TCS_BUTTON 修改标签的外观，使它们看起来像按钮一样。
- TCS_FIXEDWIDTH 使所有的标签都具有相同的宽度。（缺省的，标签控件自动调整每一个标签的尺寸使之适合于它的图标。）不能将此风格与 TCS_RIGHTJUSTIFY 风格要求使用。
- TCS_FOCUNEVER 表示一个标签永远也不会接收输入焦点。
- TCS_FOCUSONBUTTONDOWN 表示当点击一个标签时它接收输入焦点。通常这个风格只与 TCS_BUTTONS 风格一起使用。
- TCS_FORCEICONLEFT 将图标集中在左边，但标签标签仍然在中间。（缺省的，控件将图标和标签都放在中间，图标在标签的左边。）
- TCS_FORCELABELLEFT 左对齐图标和标签。
- TCS_MULTILINT 使一个标签控件显示多行标签。因此所有的标签都可

以同时显示。（缺省的，一个标签控件只显示一行标签。）

- TCS_OWNERDRAWFIXED 指定由父窗口在该控件中绘制标签。
- TCS_RIGHTJUSTIFY 向右调整标签。（缺省的，标签在每一行中是向左调整的。）
- TCS_SHAREIMAGELISTS 表示当一个标签控件被销毁时，不销毁它的图像列表。
- TCS_TOOLTIPS 表示该标签控件具有一个与之关联的工具提示控件。
- TCS_TABS 标签就显示为标签，并且在显示区域的周围绘制了边框。这个风格是缺省的。
- TCS_SINGLELINE 只显示一行标签。用户可以根据需要通过滚动来查看更多的标签。这个风格是缺省的。
- TCS_RAGGEDRIGHT 不拉伸每一行标签来使之适合控件的整个宽度。这

个风格是缺省的。

另外，你还可以将下面的窗口风格应用于一个标签控件：

- WS_CHILD 创建一个代表标签控件的子窗口。不能与 WS_POPUP 风格一起使用。
- WS_VISIBLE 创建一个初始可见的标签控件。
- WS_DISABLED 创建一个初始无效的窗口。
- WS_GROUP 指定一组控件中的第一个控件，在这组控件中用户可以使用箭头键从一个控件移动到另一个控件。在第一个控件后面的所有用 WS_GROUP 风格定义的控件都属于同一组。下一个具有 WS_GROUP 风格的控件结束这个风格组并开始下一个组（即，一组结束的地方就是另一组开始的地方）。
- WS_TABSTOP 指定用户可以通过使用 TAB 键到达的控件。用户按 TAB

键可以到达下一个具有 WS_TABSTOP 风格的控件。

请参阅 CTabCtrl::CTabCtrl

CTabCtrl::CTabCtrl

CTabCtrl( );

说明

此成员函数用来构造一个 CTabCtrl **对象。**

CTabCtrl::DeleteAllItems

BOOL DeleteAllItems( );

返回值

如果成功则返回非零值；否则返回 0。

说明

此成员函数用来从一个标签控件中删除所有的项。

CTabCtrl::DeleteItem

BOOL DeleteItem(int *nItem*);

返回值

如果成功则返回非零值；否则返回 0**。**

参数

nItem

要删除零基项的从零开始的值。

说明

此成员函数用来从一个标签控件中删除指定的项。

CTabCtrl::DeselectAll

void DeselectAll(BOOL *fExcludeFocus*);

参数

fExcludeFocus

用来指定未被选择项的范围的标志。如果这个参数被设置为 FALSE**，则所有的标签按钮都将被重新安排。如果它被设置为** TRUE**，则除了当前操作选择的项，其它所有的标签项都将被重新安排。**

说明

此成员函数用来实现 Win32 **消息** TCM_DESELECTALL **的行为，就像在“** Platform SDK **”中描述的一样。**

CTabCtrl::DrawItem

void DrawItem(LPDRAWITEMSTRUCT *lpDrawItemStruct*);

参数

lpDrawItemStruct

一个指向 DRAWITEMSTRUCT **结构的指针。该结构描述了要被绘制的项。**

说明

当一个设置绘制的标签控件的可视外观改变时，可以调用这个成员函数。

DRAWITEMST- RUCT 的成员 itemAction 定义了要被执行的绘制动作。

缺省的，这个成员函数不做任何事情。重载这个成员函数可以实现对一个属主绘制的 CTabCtrl 对象的绘制。

在这个成员函数终止之前，应用程序必须恢复为 *lpDrawItemStruct* 中提供的显示环境选择的所有图形设备接口（GDI）。

请参阅　CWnd::OnDrawItem

CTabCtrl::GetCurFocus

int GetCurFocus( ) const;

返回值

拥有当前焦点的标签的从零开始的索引。

说明

此成员函数用来获取拥有当前焦点的标签的索引。

请参阅 CTabCtrl::GetCurSel

CTabCtrl::GetCurSel

int GetCurSel( ) const;

返回值

如果成功则返回所选标签的从零开始的索引，如果没有标签被选择则返回-1。

说明

此成员函数用来获取一个标签控件中当前选择的标签。

请参阅 CTabCtrl::SetCurSel, CTabCtrl::GetCurFocus

CTabCtrl::GetExtendedStyle

DWORD GetExtendedStyle( );

返回值

返回值代表了此标签控件当前使用的控件风格。这个值是标签控件扩展风格的一个组合，就像在“ Platform SDK **”中描述的一样。**

说明

此成员函数用来实现 Win32 **消息** TCM_GETEXTENDEDSTYLE **的行为，就像在“** Platform SDK **”中描述的一样。**

CTabCtrl::GetImageList

HIMAGELIST GetImageList( ) const;

返回值

如果成功则返回此标签控件的图形列表；否则返回 NULL。

说明

此成员函数用来获取与一个标签控件相关联的图形列表。

请参阅 CTabCtrl::SetImageList, CImageList

CTabCtrl::GetItem

BOOL GetItem(int *nItem*, TC_ITEM* *pTabCtrlItem*) const;

返回值

如果成功则返回 TRUE；**否则返回** FALSE。

参数

nItem

标签的从零开始的索引。

pTabCtrlItem

指向一个 TCITEM **结构的指针，该结构用来指定要获取的信息。也用来获取该标签的信息。此结构也被** InsertItem，GetItem **和** SetItem **成员函数使用。**

说明

此成员函数用来获取一个标签控件中的某个标签的信息。

当消息被发送后，mask 成员指定要返回哪一个属性。如果 mask 成员指定 TCIF_TEXT 值，则 pszText 成员必须包含用来接收该项文本的缓冲区的地址，cchTextMax 成员必须指定缓冲区的大小。

mask

指定要获取或设置的是哪些 TCITEM 结构成员。这个成员可以是零或下列值的一个组合：

- TCIF_TEXT　pszText 成员是有效的。
- TCIF_IMAGE　iImage 成员是有效的。
- TCIF_PARAM　lParam 成员是有效的。
- TCIF_RTLREADING　用从右到左的读取顺序在 Hebrew 或 Arabic 系统中显示 pszText 的文本。
- TCIF_STATE　dwState 成员是有效的。

pszText

指向一个以空字符结尾的字符串的指针，如果此结构包含了一个标签的信息，则该字符串包含了该标签文本。如果此结构正在接收信息，则这个成员指定了用来接收该标签文本的缓冲区的地址。

cchTextMax

pszText 指向的缓冲区的大小。如果此结构不是在接收信息，则忽略这个成员。

iImage

是此标签控件的图像列表中的索引，如果此标签没有图像则这个成员的值为－1。

lParam

应用程序定义的与此标签相关联的数据。如果每一个标签有多于四个字节的应用程序定义的数据，则应用程序必须定义一个结构并用这个结构

来代替 TCITEM 结构。应用程序定义的结构的第一个成员必须是一个 TCITEMHEADER 结构。这个 TCITEMHEADER 结构与 TCITEM 结构一样，但是没有 lParam 成员。你的结构与 TCITEMHEADER 结构的大小差别必须等于每一个标签的额外的字节数。

请参阅 CTabCtrl::InsertItem, CTabCtrl::SetItem

CTabCtrl::GetItemCount

```
int GetItemCount( ) const;
```

返回值

返回此标签控件中的项数。

说明

此成员函数用来获取此标签控件中的项数。

请参阅 CTabCtrl::GetItem, CTabCtrl::SetItem

CTabCtrl::GetItemRect

BOOL GetItemRect(int *nItem*, LPRECT *lpRect*) const;

返回值

如果成功则返回非零值；否则返回 0。

参数

nItem

标签项的从零开始的索引。

lpRect

指向一个 RECT 结构的指针，该结构用来接收该标签的边界矩形。这些结构使用的是视口的当前映射模式。

说明

此成员函数用来获取一个标签控件中的指定标签的边界矩形。

请参阅 CTabCtrl::SetItemSize, CTabCtrl::AdjustRect

CTabCtrl::GetItemState

BOOL GetItemState (int *nItem*, DWORD *dwMask*, DWORD& *dwState*) const;

返回值

如果成功则返回非零值；否则返回 0。如果成功，则 *dwState* 用来接收状态信

息。

参数

nItem

要获取其状态信息的项的从零开始的索引号。

dwMask

用来指定要返回该项的哪些状态标志。可能值的列表，参见 TCITEM 结构的 mask 成员，就像“Platform SDK”中描述的一样。

dwState

一个对一个用来接收状态信息的 DWORD 值的引用。可以是下列值之一：

值	描述
TCIS_BUTTONPRESSED	该标签控件项被选择
TCIS_HIGHLIGHTED	该标签控件项被加亮，并且该标签和文本被用当前的加亮色绘制。当使用加亮色时，将是一个真正的颜色，而不是一个抖动色

说明

此成员函数用来获取由 *nItem* **标识的标签控件项的状态。一个项的状态由** TCITEM **结构中的** dwState **成员来指定。**

请参阅 CTabCtrl::SetItemState

CTabCtrl::GetRowCount

int GetRowCount( ) const;

返回值

返回此标签控件中的标签行数。

说明

此成员函数用来获取一个标签控件中的当前行数。只有具有 TCS_MULTILINT

风格的标签控件可以有多行标签。

请参阅 CTabCtrl::Create

CTabCtrl::GetToolTips

CTool TipCtrl* GetTooltips( ) const;

返回值

如果成功则返回工具提示控件的句柄；否则返回 NULL。

说明

此成员函数用来获取与一个标签控件相关联的工具提示控件的句柄。如果一个标签控件具有 TCS_TOOLTIPS 风格，则它创建一个工具提示控件。你也可以提供使用 SetToolTips 成员函数来给一个标签控件分配一个工具提示控件。

请参阅 CTabCtrl::SetTooltips, CTabCtrl::Create

CTabCtrl::HighlightItem

BOOL HighlightItem(int *idItem*, BOOL *fHighlight* = TRUE);

返回值

如果成功则返回非零值；否则返回 0。

参数

idItem

一个标签控件项的从零开始的索引。

fHighlight

指定要设置的加亮状态。如果这个值是 TRUE，**则该标签是被加亮的；如果这个值是** FALSE，**则该标签被设置为它的缺省状态。**

说明

此成员函数用来实现 Win32 **消息** TCM_HIGHLIGHTITEM**，就像在“** Platform SDK**”中描述的一样。**

CTabCtrl::HitTest

int HitTest(TC_HITTESTINFO* *pHitTestInfo*) const;

返回值

返回位于指定位置的标签的从零开始的索引，如果在指定的位置没有标签，则返回 -1**。**

参数

pHitTestInfo

指向一个 TCHITTESTINFO 结构的指针，就象在“Platform SDK”中描述的一样，该结构指定了要测试的屏幕位置。

说明

此成员函数用来确定是哪一个标签（如果有的话）位于指定的屏幕位置。

CTabCtrl::InsertItem

BOOL InsertItem(int *nItem*, TC_ITEM* *pTabCtrlItem*);
BOOL InsertItem(int *nItem*, LPCTSTR *lpszItem*);
BOOL InsertItem(int *nItem*, LPCTSTR *lpszItem*,int *nImage*);
BOOL InsertItem(UINT *nMask* int *nItem*, LPCTSTR *lpszItem*, int *nImage*, LPARAM *lParam*);

返回值

如果成功则返回从零开始的索引；否则返回-1。

参数

nItem

新标签的从零开始的索引。

pTabCtrlItem

指向一个 TCITEM **结构的指针，该结构指定了该标签的属性。**

lpszItem

指向要插入的项的指针。

nImage

一个将要被插入的图像的在一个图像列表中的从零开始的索引。

nMask

指定要设置的是哪些 TCITEM **结构属性。可以是零或下列值的一个组合：**

- TCIF_TEXT pszText**成员是有效的。**
- TCIF_IMAGE iImage**成员是有效的。**

- TCIF_PARAM　lParam 成员是有效的。
- TCIF_RTLREADING　用从右到左的读取顺序在 Hebrew 或 Arabic 系统中显示 pszText 的文本。
- TCIF_STATE　dwState 成员是有效的。

lParam

与此标签相关联的应用程序定义的数据。

说明

此成员函数用来将一个新的标签插入到一个已有的标签控件中。

请参阅　CTabCtrl::GetItem, CTabCtrl::SetItem

CTabCtrl::RemoveImage

void RemoveImage(int *nImage*);

参数

nImage

要删除的图像的从零开始的索引。

说明

此成员函数用来从一个标签控件的图像列表中删除指定的图像。该标签控件更新每一个标签的图像索引以使每一个标签保持与原来的相同图像的关联。

请参阅 CTabCtrl::GetImageList, CTabCtrl::SetImageList

CTabCtrl::SetCurFocus

void SetCurFocus(int *nItem*);

参数

nItem

指定获得焦点的标签的索引。

说明

此成员函数用来实现 Win32 **消息** TCM_SETCURFOCUS**，就像在“** Platform SDK**”中描述的一样。**

请参阅 CTabCtrl::SetCurSel

CTabCtrl::SetCurSel

int SetCurSel(int *nItem*);

返回值

如果成功则返回先前选择的标签的从零开始的索引；否则返回-1。

参数

nItem

要选择项的从零开始的索引。

说明

此成员函数用来从一个标签控件中选择一个标签。当一个标签是通过使用这个函数被选择的时候，标签控件就不会发送 TCN_SELCHANGING 或 TCN_SELCHANGE 通知消息。当用户提供点击或使用键盘来改变标签时，将使用 WM_NOTIFY 来发送这些通知消息。

请参阅 CTabCtrl::GetCurSel, CTabCtrl::GetCurFocus

CTabCtrl::SetExtendedStyle

DWORD SetExtendedStyle(DWORD *dwNewStyle*, DWORD *dwExMask* = 0);

返回值

返回一个包含先前的标签控件的扩展风格的 DWORD **值，就像在“** Platform SDK**”中描述的一样。**

参数

dwNewStyle

表示标签控件的扩展风格的一个组合的值。

dwExMask

一个用来指明 *dwNewStyle* **中的哪些风格要被影响的** DWORD **值。只有**

dwNewStyle 中的扩展风格会被改变。所有其它的风格都将保持原来的样子。如果这个参数是零，则 *dwNewStyle* 中的所有风格都将被影响。

说明

此成员函数用来实现 Win32 消息 TCM_SETEXTENDEDSTYLE，就像在“Platform SDK”中描述的一样。

请参阅 CTabCtrl::GetExtendedStyle

CTabCtrl::SetImageList

CImageList * SetImageList(CImageList * *pImageList*);

返回值

返回一个指向先前的图像列表的指针；如果没有先前的图像列表则返回 NULL。

参数

pImageList

指向要被分配给此标签控件的图像列表的指针。

说明

此成员函数用来将一个图像列表分配给一个标签控件。

请参阅 CTabCtrl::GetImageList, CImageList

CTabCtrl::SetItem

BOOL SetItem(int *nItem*, TCITEM* *pTabCtrlItem*);

返回值

如果成功则返回非零值；否则返回 0。

参数

nItem

该项的从零开始的索引。

pTabCtrlItem

指向一个 TCITEM **结构的指针，该结构包含了新的项属性。**mask **成员指定要设置哪些属性。如果** mask **成员指定的是** TCIF_TEXT **值，则** pszText **就是一个以空字符结尾的字符串的地址，而** cchTextMax **成员被忽略。**

说明

此成员函数用来设置一个标签的某些或所有属性。

请参阅 CTabCtrl::InsertItem, CTabCtrl::GetItem

CTabCtrl::SetItemExtra

BOOL SetItemExtra(int *nBytes*);

返回值

如果成功则返回非零值；否则返回零。

参数

nBytes

要设置的额外字节的数目。

说明

此成员函数用来实现 Win32 **消息** TCM_SETITEMEXTRA**，就像在“** Platform SDK**”中描述的一样。**

请参阅 CTabCtrl::SetItem

CTabCtrl::SetItemSize

CSize SetItemSize(CSize size);

返回值

返回标签控件项的原来的宽度和高度。

参数

size

标签控件项的以像素表示的新宽度和高度。

说明

此成员函数用来设置标签控件项的宽度和高度。

请参阅 CTabCtrl::AdjustRect, CTabCtrl::GetItemRect, CTabCtrl::SetItemSize

CTabCtrl::SetItemState

BOOL SetItemState(int *nItem*, DWORD *dwMask*, DWORD *dwState*);

返回值

如果成功则返回非零值；否则返回 0**。**

参数

nItem

要设置其状态信息的项的从零开始的索引号。

dwMask

用来指定哪个项的状态标志要被设置的 mask。可能值的列表，参见 TCITEM 结构的 mask 成员，就像在“Platform SDK”中描述的一样。

dwState

一个对包含状态信息的 DWORD 值的引用。可以是下列值之一：

值	描述
TCIS_BUTTONPRESSED	该标签控件项被选择
TCIS_HIGHLIGHTED	该标签控件项被加亮，并且该标签和文本被用当前的加亮色绘制。当使用加亮色时，将是一个真正的颜色，而不是一个抖动色

说明

此成员函数用来设置由 nItem 标识的标签控件项的状态。

请参阅 CTabCtrl::GetItemState

CTabCtrl::SetMinTabWidth

int SetMinTabWidth(int *cx*);

返回值

返回先前的最小标签宽度。

参数

cx

要设置给一个标签控件项的最小宽度。如果这个参数被设置为-1**，则该控件将使用缺省的标签宽度。**

说明

此成员函数用来实现 Win32 **消息** TCM_SETMINTABWIDTH**，就像在“** Platform

SDK”中描述的一样。

CTabCtrl::SetPadding

void SetPadding(CSize *size*);

参数

size

表示一个标签控件中的每一个标签的图标和标签周围的空间（填料）数量。

说明

此成员函数用来设置一个标签控件中的每一个标签的图标和标签周围的空间（填料）数量。

CTabCtrl::SetToolTips

void SetToolTips(CToolTipCtrl* *pWndTip*);

参数

pWndTip

工具提示控件的句柄。

说明

此成员函数用来将一个工具提示控件分配给一个标签控件。通过调用 GetToolTips **函数可以获得与一个标签控件相关联的工具提示控件。**

请参阅 CTabCtrl::GetToolTips

CTime

CTime 没有基类。

一个 CTime 对象代表一个绝对的时间和日期。CTime 类引入了 ANSI time_t 数据类型以及其相关的运行时函数，其中包括向或自一个 Gregorian 日期和 24-小时时间的转换功能。

CTime 值是基于世界标准时间（UCT）的，UCT 时间等于格林威治（Greenwich）时间（GMT）。本地时区是由 TZ 环境变量控制的。

当创建一个 CTime 时，将 *nDST* 参数设置为 0 表示有效的是标准时间，或将其设置为大于 0 表示有效的白天保留时间，将其设置为小于零的值表示由 C 运行时库代码来计算有效的是标准时间还是白天保留时间。如果没有设置这个参数，则它的值是不明确的，而从 mktime 返回的值是不可预知的。如果 timeptr

指向一个由先前调用 asctime，gmtime，或 localtime 返回的 tm 结构，则 tm_isdst 域包含了适当的值。

参见“Microsoft Visual C++ 6.0 参考库”的“Microsoft Visual C++ 6.0 运行时参考库”卷可以获得有关 time_t 数据类型和 CTime 使用的运行时函数的更多信息。

与 CTime 类对应的类是 CTimeSpan 类，它代表了一段时间间隔---两个 CTime 对象之间的差值。

CTime 和 CTimeSpan 类都是不可派生的。因为没有虚函数，CTime 和 CTimeSpan 对象的大小都正好是四个字节。多数成员函数都是内联函数。

有关使用 CTime 的更多信息，参见“Visual C++程序员指南”中的文章“日期和时间”，以及“Microsoft Visual C++ 6.0 运行时库参考”中的“时间管理”。

```
#include <afx.h>
```

请参阅　运行时函数：asctime, _ftime, gmtime, localtime, strftime, time

CTime 类成员

Construction	
Ctime	用各种方法构造一个 Ctime 对象
GetCurrentTime	创建一个代表当前时间的 CTime 对象（静态成员函数）
Extraction	
GetTime	返回一个与此 CTime 对象对应的 time_t
GetYear	返回此 CTime 对象代表的年
GetMonth	返回此 CTime 对象代表的月（1 至 12）
GetDay	返回此 CTime 对象代表的日（1 至 31）
GetHour	返回此 CTime 对象代表的小时（0 至 23）
GetMinute	返回此 CTime 对象代表的分钟（0 至 59）
GetSecond	返回此 CTime 对象代表的秒（0 至 59）
GetDayOfWeek	返回此日是星期几（1 代表星期日，2 代表星期一，以此类推）

Conversion	
GetGmtTm	根据 UCT 将一个 CTime 对象分解为它的组成部分
GetLocalTm	根据本地时区时将一个 CTime 对象分解为它的组成部分
GetAsSystemTime	将 CTime 对象中保存的时间信息转换为一个与 Win32 兼容的 SYSTEMTIME 结构
Format	根据本地时区时将一个 CTime 对象转换为一个格式化字符串
FormatGmt	根据 UCT 将一个 CTime 对象转换为一个格式化字符串

Operators	
Operator =	赋一个新的时间值
Operator +-	将 CTime 对象加上或减去一个 CTimeSpan 对象
Operator +=, -=	将此 CTime 对象加上或减去一个 CTimeSpan 对象
Operator ==, <, etc.	比较两个绝对时间

Archive/Dump	
operator<<	向一个 CArchive 或 CDumpContext 输出一个 CTime 对象
operator>>	从 CArchive 输入一个 CTime 对象

成员函数

CTime::CTime

CTime( );
CTime(const CTime& *timeSrc*);
CTime(time_t *time*);
CTime(int *nYear*, int *nMonth*, int *nDay*, int *nHour*, int *nMin*, int *nSec*, int *nDST* = -1);
CTime(WORD *wDosDate*, WORD *wDosTime*, int *nDST* = -1);
CTime(const SYSTEMTIME& *sysTime*, int *nDST* = -1);
CTime(const FILETIME& *fileTime*, int *nDST* = -1);

参数

timeSrc

表示一个已经存在的 CTime 对象。

time

表示一个时间值。

nYear,nMonth, nDay, nHour, nMin, nSec

表示要拷贝到新的 CTime 对象中去的日期和时间值。

nDST

表明有效的是否是 daylight saving time。可以是下列三个值中的某一个：

- *nDST* 被设置为 0　表示起作用的是标准时间。
- *nDST* 被设置为一个大于 0 的值　表示起作用的是 daylight saving time。
- *nDST* 被设置为一个小于 0 的值　表示要自动计算起作用的是标准时间还是 daylight saving time。

wDosDate, wDosTime

要被转换为一个日期/时间值并被拷贝到新的 CTime 对象中去的 MSDOS 日期和时间。

sysTime

要被转换为一个日期/时间值并被拷贝到新的 CTime 对象中去的 SYSTEMTIME 结构。

fileTime

要被转换为一个日期/时间值并被拷贝到新的 CTime 对象中去的 FILETIME 结构。

说明

所有这些构造函数都创建一个新的 CTime 对象，并用指定的基于当前时区的绝对时间值来初始化这个对象。

下面是对每一个构造函数的描述：

- CTime( ); 构造一个没有初始化的 CTime 对象。这个构造函数允许你定义 CTime 对象数组。在使用此对象之前，你应该用有效的时间来初始化这个

数组。

- CTime(const CTime&); 从另一个 CTime 值构造一个 CTime 对象。
- CTime(time_t); 从一个 time_t 类型构造一个 CTime 对象。
- CTime(int, int, etc.); 从本地时间成分构造一个 CTime 对象，每一个时间成分被约束在下列范围：

Component	Range
NYear	1970––2038*
NMonth	1––12
NDay	1––31
NHour	没有约束
NMin	没有约束
NSec	没有约束

*最大的日期限是 1/18/2038。更宽的日期范围，参见 COleDateTime。

这个构造函数进行到 UTC 的相应转换。如果一个或多个年，月，或日成分超

出了范围，则 Microsoft 基础类库的调试版会给出断言。在调用之前由你负责检验参数。

- CTime(WORD, WORD); 根据指定的 MSDOS 日期和时间构造一个 CTime 对象。
- CTime(const SYSTEMTIME&); 根据一个 SYSTEMTIME 结构构造一个 CTime 对象。
- CTime(const FILETIME&); 根据一个 FILETIME 结构构造一个 CTime 对象。你最好不要直接使用 CTime FILETIME 初始化。如果你使用一个 CTime 对象来操纵一个文件，则 CFile::GetStatus 通过使用一个用 FILETIME 结构初始化的 CTime 对象来为你获取该文件的时间标记。

有关 time_t 数据类型的更多信息，参见“ Microsoft Visual C++ 6.0 运行库参考”中的时间函数。

更多的信息，参见“Win32 SDK 程序员参考”中的 SYSTEMTIME 和 FILETIME 结构。

更多的信息，参见 Win32 SDK 文档中的 MSDOS 日期和时间项。

示例

```
// CTime::CTime 示例
time_t osBinaryTime;  // C 运行时时间(在<time.h>中定义)
time( &osBinaryTime ) ;   // 从操作系统中获取当前时间
CTime time1; // 空的 CTime. (0 是不合法的时间值.)
CTime time2 = time1; // 拷贝构造函数
CTime time3( osBinaryTime );   // 根据 C 运行时时间构造 CTime
CTime time4( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
```

请参阅　CTime::GetTime, GetCurrentTime, operator =

CTime::Format

CString Format(LPCTSTR *pFormat*) const;

CString Format(UINT *nFormatID*) const;

返回值

返回一个包含了格式化时间的 CString**。**

参数

pFormat

一个类似于 printf **格式化字符串的格式化字符串。前面有一个百分号（**%**）标记的格式化代码，被相应的** CTime **成分替换。格式化字符串中的其它字符被不作变动地拷贝到返回字符串中。参见运行时函数** strftime **可以获得详细的信息。**Format **的格式化代码的值和意义如下所示：**

- **%**D　**此** CTime **中的总天数。**
- %H　**当前天的小时。**

- %M　当前小时中的分钟。
- %S　当前分钟中的秒。
- %%　百分号。

nFormatID

用来表示这个格式的字符串的 ID。

说明

此成员函数用来创建一个日期/时间值的格式化表达式。如果此 CTime 对象的状态是空，则返回值是一个空字符串。如果 CTime 对象的状态是无效，则返回值是一个空字符串。

示例

// CTime::Format 和 CTime::FormatGmt 的示例

```
CTime t( 1999, 3, 19, 22, 15, 0 );
// 10:15PM March 19, 1999
CString s = t.Format( "%A, %B %d, %Y" );
ASSERT( s == "Friday, March 19, 1999" );
```

请参阅 CTime::FormatGmt

CTime::FormatGmt

CString FormatGmt(LPCTSTR *pFormat*) const;
CString FormatGmt(UINT *nFormatID*) const;

返回值

一个包含格式化时间的 CString**。**

参数

pFormat

一个类似于 printf **格式化字符串的格式化字符串。细节参见运行时函数**

strftime。

nFormatID

用来表示这个格式的字符串的 ID。

说明

此成员函数用来生成一个对应于这个 CTime 对象的格式化字符串。这个时间值没有被转换，因此是反映 UTC 的。

请参阅 CTime::Format

CTime::GetAsSystemTime

BOOL GetAsSystemTime(SYSTEMTIME& timeDest) const;

返回值

如果成功则返回非零值；否则返回 0。

参数

timeDest

是一个对 SYSTEMTIME **结构的引用，该结构要被转换为一个日期/时间值并被拷贝到新的** CTime **对象中。**

说明

此成员函数用来将保存在 CTime **对象中的时间信息转换为一个** Win32 **兼容的** SYSTEMTIME **结构。**GetAsSystemTime **将结果时间保存在引用的** timeDest **对象中。由这个函数初始化的** SYSTEMTIME **数据结构把它的** wMilliseconds **成员设置为零。**

CTime::GetCurrentTime

static CTime PASCAL GetCurrentTime( );

说明

此成员函数返回一个代表当前时间的 CTime **对象。**

示例

```
// CTime::GetCurrentTime 示例
CTime t = CTime::GetCurrentTime();
```

CTime::GetDay

int GetDay( ) const;

说明

此成员函数根据本地时间返回范围在 1--31 **之间的该月的天。这个函数调用**

GetLocalTm，该函数使用了一个内部的、静态分配的缓冲区。调用其它的 CTime 成员函数会导致这个缓冲区中的数据被覆盖。

示例

```
// CTime::GetDay, CTime::GetMonth, 和 CTime::GetYear 的示例
CTime t( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
ASSERT( t.GetDay() == 19 );
ASSERT( t.GetMonth() == 3 );
ASSERT( t.GetYear() == 1999 );
```

请参阅 CTime::GetDagOfWeek

CTime::GetDayOfWeek

int GetDayOfWeek( ) const;

说明

此成员函数根据本地时间返回该星期的天；1 就是星期日，2 就是星期一，...，7 就是星期六。这个函数调用 GetLocalTm，该函数使用了一个内部的，静态分配的缓冲区。调用其它的 CTime 成员函数会导致这个缓冲区中的数据被覆盖。

CTime::GetGmtTm

struct tm* GetGmtTm(struct tm* *ptm* = NULL) const;

返回值

返回一个指向已填好的 struct tm 的指针，struct tm 是在包含文件 TIME.H 中定义的。该结构的成员和值如下所示：

- tm_sec　秒

- tm_min　分钟
- tm_hour　小时（0--23）
- tm_mday　月中的日（1--31）
- tm_mon　月（0－11；一月＝0）
- tm_year　年（实际的年减1900的差）
- tm_wday　星期几（1－7；星期日＝1）
- tm_yday　年中的日（0－365；一月一日＝0）
- tm_isdst　总是0

注意　struct tm中的年是在范围70至138之间的；在CTime中，年的范围是1970到2038（包含2038）。

参数

ptm

指向一个将用来接收实际数据的缓冲区。如果这个指针是 NULL，则使用一个内部的，静态分配的缓冲区。调用其它的 CTime 成员函数将导致此缓冲区中的数据被改写。

说明

此成员函数用来获取一个包含在此 CTime 对象中所含的时间分解 struct tm。GetGmtTm 返回 UTC。

这个函数调用 GetLocalTm，该函数使用了一个内部的、静态分配的缓冲区。调用其它的 CTime成员函数会导致这个缓冲区中的数据被覆盖。

示例

参见 GetLocalTm **的示例。**

CTime::GetHour

int GetHour( ) const;

说明

此成员函数根据本地时间返回范围在 0 **至** 23 **之间的小时。这个函数调用** GetLocalTm**，该函数使用了一个内部的、静态分配的缓冲区。调用其它的** CTime **成员函数会导致这个缓冲区中的数据被覆盖。**

示例

// CTime::GetHour, CTime::GetMinute, **和** CTime::GetSecond **的示例**

```
CTime t( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
ASSERT( t.GetSecond() == 0 );
ASSERT( t.GetMinute() == 15 );
ASSERT( t.GetHour() == 22 );
```

CTime::GetLocalTm

```
struct tm* GetLocalTm( struct tm* ptm = NULL ) const;
```

返回值

返回一个指向已填好的 struct tm 的指针，struct tm 是在包含文件 TIME.H 中定义的。关于该结构的成员安排，参见 GetGmtTm。

参数

ptm

指向一个将用来接收实际数据的缓冲区。如果这个指针是 NULL，则使

用一个内部的、静态分配的缓冲区。调用其它的 CTime 成员函数将导致此缓冲区中数据被改写。

说明

此成员函数用来获取一个包含此 CTime 对象的分解后的各个成分的 struct tm。

GetLocalTm

返回本地数据。

示例

```
// CTime::GetLocalTm 示例
CTime t( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
struct tm* osTime;   // 指向一个包含数据元素的指针。
osTime = t.GetLocalTm( NULL );
ASSERT( osTime->tm_mon == 2 ); // Note zero-based month!
```

CTime::GetMinute

int GetMinute( ) const;

说明

此成员函数根据本地时间返回范围在 0 **至** 59 **之间的分钟。这个函数调用** GetLocalTm**，该函数使用了一个内部的、静态分配的缓冲区。调用其它的** CTime **成员函数会导致这个缓冲区中的数据被覆盖。**

CTime::GetMonth

int GetMonth( ) const;

说明

此成员函数根据本地时间返回范围在 1 **至** 12 **之间的月（**1**＝一月）。这个函数**

调用 GetLocalTm，该函数使用了一个内部的、静态分配的缓冲区。调用其它的 CTime 成员函数会导致这个缓冲区中的数据被覆盖。

示例

参见 GetDay 的示例。

CTime::GetSecond

int GetSecond( ) const;

说明

此成员函数根据本地时间返回范围在 0 至 59 之间的秒。这个函数调用 GetLocalTm，该函数使用了一个内部的、静态分配的缓冲区。调用其它的 CTime 成员函数会导致这个缓冲区中的数据被覆盖。

CTime::GetTime

time_t GetTime( ) const;

说明

此成员函数返回一个给定 CTime **对象的** time_t **值。**

示例

```
// CTime::GetTime示例
CTime t( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
time_t osBinaryTime = t.GetTime(); // time_t在<time.h>中定义
printf( "time_t = %ld\n", osBinaryTime );
```

请参阅 CTime::CTime

CTime::GetYear

int GetYear( ) const;

说明

此成员函数根据本地时间返回范围在 1970 年 1 月 1 日至 2038 年 1 月 18 日之间的年。这个函数调用 GetLocalTm，该函数使用了一个内部的，静态分配的缓冲区。调用其它的 CTime 成员函数会导致这个缓冲区中的数据被覆盖。

示例

参见 GetDay 的示例。

请参阅 CTime::CTime

操作符

CTime::operator =

const CTime& operator =(const CTime& *timeSrc*);
const CTime& operator =(time_t *t*);

说明

这些重载的赋值操作符将源时间拷贝到此 CTime 对象中。

保存在一个 CTime 对象中的内部时间与时区无关。在赋值操作中不用进行时区转换。

示例

// CTime::operator = 示例

```
time_t osBinaryTime;  // C 运行时时间 (在<time.h>中定义)
CTime t1 = osBinaryTime; // Assignment from time_t
CTime t2 = t1; // Assignment from CTime
```

请参阅 CTime::CTime

CTime::operator +, -

```
CTime operator +( CTimeSpan timeSpan ) const;
CTime operator - CTimeSpan timeSpan ) const;
CTimeSpan operator - CTime time ) const;
```

说明

CTime **对象表示绝对时间。**CTimeSpan **对象表示相对时间。前两个操作符允许你向或从** CTime **对象中加上或减去一个** CTimeSpan **对象。第三个操作符允许你将两个** CTime**对象相减产生一个** CTimeSpan **对象。**

示例

```
//    CTime::operator +, - 示例
CTime t1( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
CTime t2( 1999, 3, 20, 22, 15, 0 ); // 10:15PM March 20, 1999
CTimeSpan ts = t2 - t1;    // Subtract 2 CTimes
ASSERT( ts.GetTotalSeconds() == 86400L );
ASSERT( ( t1 + ts ) == t2 );    // Add a CTimeSpan to a CTime.
ASSERT( ( t2 - ts ) == t1 );    // Subtract a CTimeSpan from a Ctime.
```

CTime::operator +=, -=

```
const CTime& operator +=( CTimeSpan timeSpan );
const CTime& operator ?( CTimeSpan timeSpan );
```

说明

这些操作符允许你从此 CTime **对象中加上或减去一个** CTimeSpan **对象。**

示例

```
// CTime::operator -= 示例
CTime t( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
t += CTimeSpan( 0, 1, 0, 0 ); // 1 hour exactly
ASSERT( t.GetHour() == 23 );
```

CTime Comparison Operators

```
BOOL operator ==( CTime time ) const;
BOOL operator !=( CTime time ) const;
BOOL operator <( CTime time ) const;
BOOL operator >( CTime time ) const;
BOOL operator <=( CTime time ) const;
BOOL operator >=( CTime time ) const;
```

说明

这些操作符比较两个绝对时间，如果测试条件为真则返回非零值；否则返回 0。

示例

```
// Ctime比较操作符示例
CTime t1 = CTime::GetCurrentTime();
CTime t2 = t1 + CTimeSpan( 0, 1, 0, 0 );       // 1 hour later
ASSERT( t1 != t2 );
ASSERT( t1 < t2 );
ASSERT( t1 <= t2 );
```

CTime::operators <<, >>

```
friend CDumpContext& AFXAPI operator<<( CDumpContext& dc, CTime time );
friend CArchive& AFXAPI operator<<( CArchive& ar, CTime time );
friend CArchive& AFXAPI operator>>( CArchive& ar, CTime& rtime );
```

说明

CTime 插入操作符（<<）支持向一个存档中进行诊断存档和转储。提取操作符（>>）支持从一个存档装入。

示例

```
// CTime::operators <<, >> 示例
CTime t( 1999, 3, 19, 22, 15, 0 ); // 10:15PM March 19, 1999
afxDump << t << "\n"; // Prints 'CTime("Fri Mar 19 22:15:00 1999")'.

extern CArchive ar;
if( ar.IsLoading() )
ar >> t;
else
ar << t;
```

请参阅 CArchive, CdumpContext

CTimeSpan

CTimeSpan 没有基类。

一个 CTimeSpan 对象代表一个相对的时间段。CTimeSpan 引入了 ANSI time_t 数据及与之相关的运行时函数。这些函数将秒转换为日，时，分和秒的各种组合。

一个 CTimeSpan 对象以秒为单位保存时间。由于 CTimeSpan 对象以带符号的四字节数存储，所以最大的时间跨度近似为+-68 年。

另外一个与 CTimeSpan 对应的类是 CTime，它描述的是绝对时间。CTimeSpan 是两个 CTime 对象之间的差。

CTime 和 CTimeSpan 类不可派生。因为没有虚函数，所以 CTime 和 CTimeSpan 对象的大小都正好是四字节。大多数成员函数都是内联函数。

有关使用 CTime 的更多信息，参见“Visual C++程序员指南”中的文章“日期和时间”，以及“Microsoft Visual C++ 6.0 参考库”的“Microsoft Visual C++ 6.0 运行时库参考”卷中的“时间管理”。

```
#include <afx.h>
```

请参阅 Run-time functions: asctime, _ftime, gmtime, localtime, strftime, time

CTimeSpan 类成员

Construction	
CTimeSpan	用不同的方法构造 CtimeSpan 对象

Extraction	
GetDays	返回此 CTimeSpan 对象中的完整的天数
GetHours	返回当前日中的小时数（-23 至+23）
GetTotalHours	返回此 CTimeSpan.对象中的完整的小时数
GetMinutes	返回当前小时中的分钟数（-59 至+59）
GetTotalMinutes	返回此 CTimeSpan.对象中的完整的分钟数
GetSeconds	返回当前分钟中的秒数（-59 至+59）
GetTotalSeconds	返回此 CTimeSpan.对象中的完整的秒数

Conversion	
Format	将一个 CTimeSpan 转换为一个格式化的字符串

Operators	
Operator =	赋一个新的时间段值
Operator +-	加或减去 CTimeSpan 对象
Operator += -=	向或从此 CTimeSpan 中加上或减去一个 CTimeSpan 对象
Operator == < etc.	比较两个相对的时间值

Archive/Dump	
operator<<	向 CArchive 或 CDumpContext 输出一个一个 CTimeSpan 对象
operator>>	从 CArchive 输入一个 CTimeSpan 对象

成员函数

CTimeSpan::CTimeSpan

```
CTimeSpan( );
CTimeSpan( const CTimeSpan& timeSpanSrc );
CTimeSpan( time_t time );
CTimeSpan( LONG lDays, int nHours, int nMins, int nSecs );
```

参数

timeSpanSrc

一个已存在的 CTimeSpan 对象。

time

一个 time_t 时间值。

lDays, nHours, nMins, nSecs

分别代表日，时，分，秒

说明

所有这些构造函数都创建一个新的 CTimeSpan 对象，并用指定的相对时间值来对其进行初始化。下面是对每一个构造函数的描述：

- CTimeSpan( ); 构造一个没有被初始化的 CTimeSpan 对象。
- CTimeSpan(const CTimeSpan&); 从另一个 CTimeSpan 值构造一个 CTimeSpan 对象。
- CTimeSpan(time_t); 从一个 time_t 类型构造一个 CTimeSpan 对象。这个 time_t 值应该是两个绝对 time_t 值的差。

- CTimeSpan(LONG, int, int, int); 由时间跨度的组成部分构造一个 CTimeSpan 对象。每一个成分都受下列范围的限制：

Component	Range
LDays	0 至 25,000（近似值）
NHours	0 至 23
NMins	0 至 59
NSecs	0 至 59

注意，如果有一个或多个时间–日成分超出了范围，则 Microsoft 基础类库将给出断言。在调用之前由你负责检验参数是否有效。

示例

```
// CTimeSpan::CTimeSpan 示例
CTimeSpan ts1;  // 没有初始化的时间值
CTimeSpan ts2a( ts1 ); // Copy constructor
CTimeSpan ts2b = ts1; // Copy constructor again
CTimeSpan ts3( 100 ); // 100 seconds
CTimeSpan ts4( 0, 1, 5, 12 );      // 1 hour, 5 minutes, and 12 seconds
```

CTimeSpan::Format

CString Format(LPCSTR *pFormat*) const;
CString Format(LPCTSTR *pFormat*) const;
CString Format(UINT *nID*) const;

返回值

返回一个包含格式化的时间的 CString **对象。**

参数

pFormat

一个类似于 printf **格式化字符串的格式化字符串。前面有一个百分号（**%**）标记的格式化代码，将被相应的** CTimeSpan **成分替换。格式化字符串中的其它字符被不作变动地拷贝到返回字符串中。**Format **的格式化代码的值和意义如下所示：**

- %D　此 CTime 中的总天数。
- %H　当前天的小时。
- %M　当前小时中的分钟。
- %S　当前分钟中的秒。
- %%　百分号。

nID

用来表示这个格式的字符串的 ID。

说明

生成一个对应于此 CTimeSpan 的格式化字符串。

库的调试版检查格式化代码，如果代码不在上面的列表中，则将给出断言。

示例

```
// CTimeSpan::Format 示例
CTimeSpan ts( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
CString s = ts.Format( "Total days: %D, hours: %H, mins: %M, secs: %S" );
ASSERT( s == "Total days: 3, hours: 01, mins: 05, secs: 12" );
```

CTimeSpan::GetDays

LONG GetDays( ) const;

说明

此成员函数返回完整的天数。如果该时间段是负的，则这个值可能是负的。

示例

```
// CTimeSpan::GetDays 示例
CTimeSpan ts( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
ASSERT( ts.GetDays() == 3 );
```

CTimeSpan::GetHours

int GetHours( ) const;

说明

此成员函数返回当前日的小时数。其范围是-23 **到**+23**。**

示例

```
// CTimeSpan::GetHours 示例
CTimeSpan ts( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
ASSERT( ts.GetHours() == 1 );
ASSERT( ts.GetMinutes() == 5 );
ASSERT( ts.GetSeconds() == 12 );
```

CTimeSpan::GetMinutes

int GetMinutes( ) const;

说明

此成员函数返回当前小时中的分钟数。范围是-59 **到** 59**。**

示例

参见 GetHours **的示例。**

CTimeSpan::GetSeconds

int GetSeconds( ) const;

说明

此成员函数返回当前分钟中的秒数。范围是-59 **到** 59**。**

示例

参见 GetHours 的示例

CTimeSpan::GetTotalHours

LONG GetTotalHours( ) const;

说明

此成员函数返回此 CTimeSpan 中的完整小时数。

示例

```
// CTimeSpan::GetTotalHours 示例
CTimeSpan ts( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
ASSERT( ts.GetTotalHours() == 73 );
ASSERT( ts.GetTotalMinutes() == 4385 );
ASSERT( ts.GetTotalSeconds() == 263112 );
```

CTimeSpan::GetTotalMinutes

LONG GetTotalMinutes( ) const;

说明

此成员函数返回此 CTimeSpan **中的完整分钟数。**

示例

参见 GetTotalHours **的示例。**

CTimeSpan::GetTotalSecond

LONG GetTotalSeconds( ) const;

说明

此成员函数返回此 CTimeSpan **中的完整秒数。**

示例

参见 GetTotalHours **的示例。**

操作符

CTimeSpan::operator =

const CTimeSpan& operator =(const CTimeSpan& *timeSpanSrc*);

说明

这些重载的赋值操作符将源 CTimeSpan *timeSpanSrc* **对象拷贝到此** CTimeSpan **对象中。**

示例

```
// CTimeSpan::operator =示例
CTimeSpan ts1;
CTimeSpan ts2( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
ts1 = ts2;
ASSERT( ts1 == ts2 );
```

请参阅 CTimeSpan::CTimeSpan

CTimeSpan::operator + -

```
CTimeSpan operator +( CTimeSpan timeSpan ) const;
CTimeSpan operator - LCTimeSpan timeSpan ) const;
```

说明

这两个操作符允许你将两个 CTimeSpan **对象相加或相减。**

示例

```
// CTimeSpan::operator +, - 示例
CTimeSpan ts1( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
CTimeSpan ts2( 100 ); // 100 seconds
CTimeSpan ts3 = ts1 + ts2;
```

CTimeSpan::operator += -=

```
const CTimeSpan& operator +=( CTimeSpan timeSpan );
const CTimeSpan& operator -=( CTimeSpan timeSpan );
```

说明

这些操作符允许你向或从此 CTimeSpan **中加上或减去一个** CTimeSpan **对象。**

示例

```
// CTimeSpan::operator +=, -=示例
CTimeSpan ts1( 10 ); // 10 seconds
```

```
CTimeSpan ts2( 100 ); // 100 seconds
ts2 -= ts1;
ASSERT( ts2.GetTotalSeconds() == 90 );
```

CTimeSpan Comparison Operators

BOOL operator ==(CTimeSpan *timeSpan*) const;
BOOL operator !=(CTimeSpan *timeSpan*) const;
BOOL operator <(CTimeSpan *timeSpan*) const;
BOOL operator >(CTimeSpan *timeSpan*) const;
BOOL operator <=(CTimeSpan *timeSpan*) const;
BOOL operator >=(CTimeSpan *timeSpan*) const;

说明

这些操作符比较两个相对时间值。如果测试条件为真，则返回非零值；否则返回 0。

示例

```
// CTimeSpan 比较操作符示例
CTimeSpan ts1( 100 );
CTimeSpan ts2( 110 );
ASSERT( ( ts1 != ts2 ) && ( ts1 < ts2 ) && ( ts1 <= ts2 ) );
```

CTimeSpan::operator <<, >>

friend CDumpContext& AFXAPI operator<<(CDumpContext& *dc*, CTimeSpan *timeSpan*);
friend CArchive& AFXAPI operator<<(CArchive& *ar*, CTimeSpan *timeSpan*);
friend CArchive& AFXAPI operator>>(CArchive& *ar*, CTimeSpan& *rtimeSpan*);

说明

CTimeSpan 插入操作符（<<）支持向一个存档进行诊断转储和存储。提取操作符（>>）支持从一个存档中装入。

当你向一个转储环境发送 CTimeSpan 对象时，时间跨度以日，时，分和秒的数

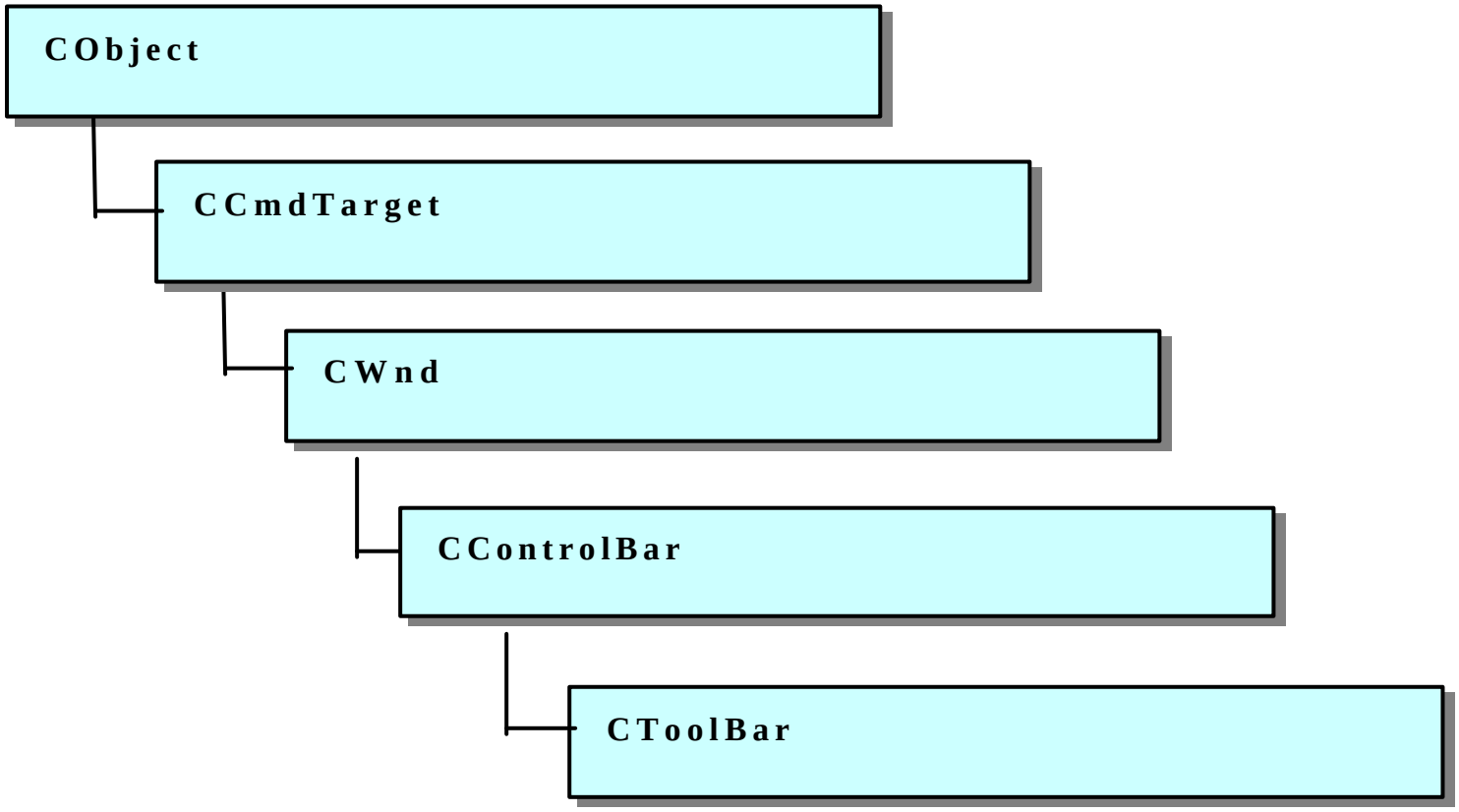
字字符格式显示出来。

示例

```
// CTimeSpan::operators <<, >> 示例
CTimeSpan ts( 3, 1, 5, 12 ); // 3 days, 1 hour, 5 min, and 12 sec
#ifdef _DEBUG
afxDump << ts << "\n";
#endif
// Prints 'CTimeSpan(3 days, 1 hours, 5 minutes and 12 seconds)'

extern CArchive ar;
if( ar.IsLoading( ))
   ar >> ts;
else
   ar << ts;
```

CToolBar



类 CToolBar 的对象是带有一行位图按钮和可选分隔线的控件条。这些按钮可以像普通按钮、核选按钮或单选按钮那样动作。CToolBar 对象通常是由

CFrameWnd 或 CMDIFrameWnd 派生的框架窗口对象的嵌入成员。

CToolBar::GetToolBarCtrl 是 MFC4.0 后新增加的成员函数，它允许你利用 Windows 通用控件对工具条的定制及其它性能的支持。CToolBar 的成员函数为你提供了 Windows 通用控件的大多数性能；但是，当你调用 GetToolBarCtrl 时，你可以使你的工具条具有更多 Windows 95 工具条的特征。参见 CToolBarCtrl 可以获得更多有关如何使用 Windows 通用控件来设计工具条的信息。更多有关通用控件的一般信息，参见“Windows 95 SDK 程序员参考”中的“通用控件”。

Visual C++提供了两种创建一个工具条的方法。要使用资源编辑器（Resource Editor）创建一个工具条，应遵循下面的步骤：

1. 创建一个工具条资源。
2. 构造 CToolBar 对象。
3. 调用 Create(或 CreateEx)函数来创建 Windows 工具条并将它与该 CToolBar

对象连接。

4. 调用 LoadToolBar 来装入工具条资源。

另外，也可以依据下面的步骤：

1. 构造 CToolBar 对象。
2. 调用 Create（或 CreateEx）函数来创建 Windows 工具条并将它与 CToolBar 对象连接。
3. 调用 LoadBitmap 来装入包含工具条按钮图像的位图。
4. 调用 SetButtons 来设置按钮风格并使每一个按钮与位图中的一幅图像关联。

此工具条中的所有按钮图像都位于同一个位图，该位图必须为每一个按钮包含一个图像。所有的图像都必须是同样大小的。缺省的尺寸是 16 个像素宽和 15 个像素高。这些图像必须一幅接一幅的放在位图中。

SetButton 函数以指向一个控制 ID 数组的指针和一个指定数组中元素数目的整数作为参数。该函数将每个按钮的 ID 值设置为对应的数组元素的值，并赋予每个按钮一个图像索引值，每个索引值指定对应按钮图像在位图中的位置。如果某一数组元素的值为 ID_SEPARATOR，则不为其赋图像索引值。

按钮图像在位图中的次序通常就是图像在屏幕上的绘制次序，但你也可以使用 SetButtonInfo 函数来修改图像次序和绘制次序的关系。

一个工具条中的所有按钮都具有相同的尺寸。按照“Windows 界面软件设计指南”中所说，缺省尺寸为 24×22 个像素。图像和按钮之间的任何空白尺寸都可用来在图像周围形成一个边界。

每个按钮具有一幅图像。按钮的各种状态和风格（被压住，弹起，按下，无效，无效按下，以及不定）都可以由这幅图像产生。虽然位图可以是任何颜色，但是使用灰色加黑色文字及阴影可实现最好的效果。

缺省情况下，工具条按钮模拟普通按钮。但是，工具条按钮也可以模拟复选框按钮和单选按钮。复选框按钮具有三种状态：核选，清除和不定。单选按钮值具有两种状态：核选和清除。

要设置单个按钮或不指向一个数组的分隔线风格，可以调用 GetButtonStyle 来获取这个风格，然后调用 SetButtonStyle（而不是调用 SetButton）。当你要在运行时改变按钮的风格时，SetButtonStyle 是最有用的。

要分配显示在一个按钮上的文本，可以调用 GetButtonText 来获取要显示在该按钮上的文本，然后调用 SetButtonText 来设置这些文本。

要创建一个复选框按钮，给它赋予 TBBS_CHECKBOX 风格或在一个 ON_UPDATE_ COMMAND_UI 处理程序中使用某个 CCmdUI 对象的 SetCheck 成员函数。调用 SetCheck 将一个普通按钮变成一个复选框按钮。传递给 SetCheck 函数的参数为 0，则按钮是未核选的，传递 1 则按钮被核选，传递 2 则按钮为

不定状态。

要创建一个单选按钮，在一个 ON_UPDATE_COMMAND_UI 处理程序中调用某个 CCmdUI 对象的 SetRadio 成员函数。给 SetRadio 传递参数 0，则按钮为未核选的，传递非零值，则按钮是被核选的。为了提供一个单选按钮组的互不相容性，则组中的每个按钮都必须具有 ON_UPDAT_COMMAND_UI 处理程序。

有关使用 CToolBar 的更多信息，参见“ Visual C++程序员指南”中的文章“工具条”和“ Visual C++联机文件”中的技术注释 31，控制条。

include <afxext.h>

请参阅 CToolBarCtrl, CControlBar, CToolBar::Create, CToolBar::LoadBitmap, CToolBar::SetButtons, CCmdUI::SetCheck, CCmdUI::SetRadio

CToolBar 类成员

Construction	
CtoolBar	创建一个 CtoolBar 对象
Create	创建 Windows 工具条并将它与该 CToolBar 连接
CreateEx	为嵌入的 CtoolBarCtrl 对象创建一个具有附加风格的 CToolBar 对象
SetSizes	设置按钮及其位图的尺寸
SetHeight	设置工具条的高度
LoadToolBar	装入一个用资源编辑器创建的工具条资源
LoadBitmap	装入包含位图-按钮图像的位图
SetBitmap	设置一个位图中的图像
SetButtons	设置按钮风格和按钮图像在位图中的索引

Attributes	
CommandToIndex	返回具有给定的命令 ID 的按钮的索引
GetItemID	返回具有给定索引值的按钮或分隔线的命令 ID
GetItemRect	获取具有给定索引值的项的显示矩形
GetButtonStyle	获取一个按钮的风格
SetButtonStyle	设置一个按钮的风格
GetButtonInfo	获取一个按钮的 ID，风格和图像号
SetButtonInfo	设置一个按钮的 ID，风格和图像号

GetButtonText	获取要显示在一个按钮上的文本
SetButtonText	设置要显示在一个按钮上的文本
GetToolBarCtrl	允许直接访问基本的通用控件

成员函数

CToolBar::CommandToIndex

int CommandToIndex(UINT *nIDFind*);

返回值

返回按钮的索引，如果没有按钮具有这个给定的命令 ID**，则返回**-1**。**

参数

nIDFind

一个工具条按钮的命令 ID。

说明

此成员函数返回第一个工具条按钮的从位置 0 开始的索引，该按钮的命令 ID 与 nIDFind 匹配。

请参阅 CToolBar::GetItemId

CToolBar::Create

```
BOOL Create( CWnd* pParentWnd, DWORD dwStyle = WS_CHILD | WS_VISIBLE |
    CBRS_TOP, UINT nID = AFX_IDW_TOOLBAR );
```

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

指向工具条的父窗口的指针。

dwStyle

工具条的风格。下面列出的是此参数支持的附加风格：

- CBRS_TOP **控制条位于框架窗口的顶部。**
- CBRS_BOTTOM **控制条位于框架窗口的底部。**
- CBRS_NOALIGN **当父窗口调整尺寸时不重定位控制条。**
- CBRS_TOOLTIPS **控制条显示工具提示。**
- CBRS_SIZE_DYNAMIC **控制条是动态的。**
- CBRS_SIZE_FIXED **控制条是固定的。**
- CBRS_FLOATING **控制条是浮动的。**

- CBRS_FLYBY　状态条显示按钮的信息。
- CBRS_HIDE_INPLACE　控件不显示给用户。

ID

工具条子窗口的 ID。

说明

此成员函数用来创建一个 Windows 工具条（一个子窗口），并将它与 CToolBan 对象连接。它还将工具条的高度设置为缺省值。

请参阅　CToolBar::CToolBar, CToolBar::LoadBitmap, CToolBar::SetButtons, CToolBar::LoadToolBar,CControlBar::CalcDynamicLayout, CControlBar::CalcFixedLayout

CToolBar::CreateEx

BOOL CreateEx(CWnd* *pParentWnd*, DWORD *dwCtrlStyle* = TBSTYLE_FLAT,

DWORD *dwStyle* = WS_CHILD | WS_VISIBLE | CBRS_ALIGN_TOP,
CRect *rcBorders* = CRect(0, 0, 0), UINT *nID* = AFX_IDW_TOOLBAR);

返回值

如果成功则返回非零值；否则返回 0。

参数

pParentWnd

指向工具条的父窗口的指针。

dwCtrlStyle

嵌入 CToolBarCtrl **对象的附加风格。缺省的，这个值被设置为** TBSTYLE_FLAT。**工具条风格的完整列表，参见** *dwStyle*。

dwStyle

工具条风格。参见“ Platform SDK **”的“工具条控件和按钮风格”可以**

获得一个相关风格的列表。

rcBorders

一个定义工具条窗口边框的宽度的 CRect 对象。缺省的这些边框被设置为 0，0，0，0，这样的工具条窗口是没有边界的。

nID

工具条的子窗口 ID。

说明

此成员函数用来创建一个 Windows 工具条（一个子窗口）并将它与该 CToolBar 对象连接。它还将工具条的高度设置为一个缺省值。

当在嵌入工具条控件的创建期间需要指定某个特定的风格时，使用 CreateEx 来代替 Create。例如，将 *dwCtrlStyle* 设置为 TBSTYLE_FLAT | TBSTYLE_TRANSPARENT 来创建一个类似于 Internet Explorer 4 工具条的工

具条。

CToolBar::CToolBar

CToolBar( );

说明

这个成员函数构造一个 CToolBar **对象并将它设置为缺省尺寸。**

请参阅 CToolBar::Create

CToolBar::GetButtonInfo

void GetButtonInfo(int *nIndex*, UINT& *nID*, UINT& *nStyle*, int& *iImage*) const;

参数

nIndex

要获取其信息的工具条按钮或分隔线的索引。

nID

对一个被设置为按钮的命令 ID 的 UINT 值的引用。

nStyle

对一个被设置为按钮的风格的 UINT 值的引用。

iImage

对一个被设置为按钮的图像在位图中的索引值的整数的引用。

说明

此成员函数用来获取位于由 *nIndex* 指定的位置的工具条按钮或分隔线的控制 ID，风格和图像索引。这些值被分别赋给由 *nID*，*nStyle* 和 *iImage* 引用的变量。图像索引是图像在位图中的位置，该位图包含了所有工具条按钮的图像。第一个图像位于位置 0。

如果 *nIndex* **指定的是一个分隔线，则** *iImage* **被设置为以像素表示的分隔线宽度。**

请参阅 CToolBar::SetButtonInfo, CToolBar::GetItemID

CToolBar::GetButtonStyle

UINT GetButtonStyle(int *nIndex*) const;

返回值

返回由 *nIndex* **指定的按钮或分隔线的风格。**

参数

nIndex

要获取其风格的工具条按钮或分隔线的索引。

说明

此成员函数用来获取工具条上的一个按钮或分隔线的风格。一个按钮的风格确定了按钮显示的样子，以及按钮对用户输入作出的反应。参见 SetButtonStyle **可以获得有关按钮风格的例子。**

请参阅 CToolBar::SetButtonStyle

CToolBar::GetButtonText

CString GetButtonText(int *nIndex*) const;
void GetButtonText(int *nIndex*, CString& *rString*) const;

返回值

返回包含按钮的文本的 CString **对象。**

参数

nIndex

要获取的文本的索引。

rString

一个对 CSing **对象的引用，该对象将包含要被获取的文本。**

说明

此成员函数用来获取显示在一个按钮上的文本。这个函数的第二种形式用字符串文本来填充一个 CString **对象。**

请参阅 CToolBar::SetButtonText

CToolBar::GetItemID

UINT GetItemID(int *nIndex*) const;

返回值

返回由 nIndex **指定的按钮或分隔线的命令** ID**。**

参数

nIndex

要获取其 ID **的项的索引。**

说明

此成员函数返回由 *nIndex* **指定的按钮或分隔线的命令** ID**。分隔线返回** ID_SEPARATOR**。**

请参阅 CToolBar::CommandToIndex, CControlBar::GetCount

CToolBar::GetItemRect

virtual void GetItemRect(int *nIndex*, LPRECT *lpRect*);

参数

nIndex

要获取其矩形坐标的项（按钮或分隔线）的索引。

lpRect

将包含项的坐标的 RECT **结构的地址。**

说明

此成员函数用由 *nIndex* **指定的按钮或分隔线的坐标来填充由** *lpRect* **指定地址的** RECT **结构。坐标是以像素为单位的相对于工具条的左上角的值。**

使用 GetItemRect **函数可以获取要用组合框或其它控件替换的分隔线的坐标。**

请参阅 CToolBar::CommandToIndex

CToolBar::GetToolBarCtrl

CToolBarCtrl& GetToolBarCtrl( ) const;

返回值

返回一个对 CToolBarCtrl **对象的引用。**

说明

此成员函数允许对基本通用控件的直接访问。

使用 GetToolBarCtrl **可以利用** Windows **工具条通用控件的性能，并且可以利用** CToolBarCtrl **为工具条定制提供的支持。**

有关使用命令控件的更多信息，参见“ Visual C++ **程序员指南”中的文章“控**

件主题”和“Windows 95 SDK程序员参考”中的“通用控件”。

请参阅 CToolBarCtrl

CToolBar::LoadBitmap

BOOL LoadBitmap(LPCTSTR *lpszResourceName*);
BOOL LoadBitmap(UINT *nIDResource*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpszResourceName

指向要被装入的位图的资源名称的指针。

nIDResource

要装入的位图的资源 ID。

说明

此成员函数用来装入由 *lpszResourceName* 或 *nIDResource* 指定的位图。该位图应该为每一个工具条按钮包含一个图像。如果该图像不具有标准尺寸（16 像素宽，15 像素高），则定义 SetSizes 来设置按钮的尺寸和它们的图像。

请参阅 CToolBar::Create, CToolBar::SetButtons, CToolBar::SetSizes, CToolBar::LoadToolBar

CToolBar::LoadToolBar

BOOL LoadToolBar(LPCTSTR *lpszResourceName*);
BOOL LoadToolBar(UINT *nIDResource*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpszResourceName

指向要被装入的工具条的资源名称的指针。

nIDResource

要装入的工具条的资源 ID。

说明

此成员函数用来装入由 *lpszResourceName* **或** *nIDResource* **指定的工具条。**

参见“ Visual C++**用户指南”中的“工具条编辑器”可以获得有关创建一个工具条资源的更多信息。**

请参阅 CToolBar::Create, CToolbar::LoadBitmap, CToolBar::SetButtons

CToolBar::SetBitmap

BOOL SetBitmap(HBITMAP *hbmImageWell*);

返回值

如果成功则返回非零值；否则返回 0。

参数

hbmImageWell

一个与工具条相关联的位图图像的句柄。

说明

此成员函数用来为工具条设置位图图像。例如，在用户对文档采取了可以改变

按钮动作的行动之后，调用 SetBitmap 可以改变位图方式的图像。

CToolBar::SetButtonInfo

void SetButtonInfo(int *nIndex*, UINT *nID*, UINT *nStyle*, int *iImage*);

参数

nIndex

要设置其信息的按钮或分隔线的索引。

nID

要给按钮的命令 ID 设置的值。

nStyle

新的按钮风格。下面的按钮风格是被支持的：

- TBBS_BUTTON 标准按钮（缺省的）
- TBBS_SEPARATOR 分隔线。

- TBBS_CHECKBOX 自动复选框按钮。
- TBBS_GROUP 标记一个按钮组的开始。
- TBBS_CHECKGROUP 标记一个复选框按钮组的开始。

iImage

按钮的图像在位图中的新索引。

说明

此成员函数用来设置按钮的命令 ID，风格和图像号。对于具有 TBBS_SEPARATOR 风格的分隔线，这个函数将分隔线的以像素为单位的宽度设置为保存在 iImage 中的值。

有关位图图像和按钮的更多信息，参见 CToolBar 概述和 CToolBar::LoadBitmap。

请参阅 CToolBar::GetButtonInfo

CToolBar::SetButtons

BOOL SetButtons(const UINT* *lpIDArray*,int *nIDCount*);

返回值

如果成功则返回非零值；否则返回 0。

参数

lpIDArray

指向一个命令 Ids **数组的指针。它可以是** NULL**，用于分配空的按钮。**

nIDCount

由 *lpIDArray* **指向的数组的元素数目。**

说明

此成员函数用来将每一个工具条按钮的命令 ID 设置为由数组 *lpIDArray* 的相应元素指定的值。如果该数组的某个元素的值是 ID_SEPARATOR，则在工具条的相应位置创建一个分隔线。这个函数还将每个控件的风格设置为 TBBS_BUTTON，将每个分隔线的风格设置为 TBBS_SEPARATOR，并为每一个按钮分配一个图像索引。图像索引指定了按钮的图像在位图中的位置。

你不用在位图中说明分隔线，因为这个函数不给分隔线分配图像索引。如果你的工具条有位于位置 0，1 和 3 的按钮，有位于位置 2 的分隔线，则位于你的位图中的位置 0，1，和 2 的图像将被分别分配给位置 0，1 和 3 的位图。

如果 *lpIDArray* 是 NULL，则此函数按 *nIDCount* 指定的项数分配空间。使用 SetButtonInfo 可以设置每一个项的属性。

请参阅 CToolBar::Create, CToolBar::SetButtonInfo, CToolBar::SetButtonStyle, CToolBar::LoadToolBar

CToolBar::SetButtonStyle

void SetButtonStyle(int *nIndex*, UINT *nStyle*);

参数

nIndex

要设置其风格的按钮或分隔线的索引。

nStyle

按钮风格。下面的按钮风格都是被支持的：

- TBBS_BUTTON **标准按钮（缺省的）**
- TBBS_SEPARATOR **分隔线。**
- TBBS_CHECKBOX **自动复选框按钮。**

- TBBS_GROUP　标记一个按钮组的开始。
- TBBS_CHECKGROUP　标记一个复选框按钮组的开始。

说明

此成员函数用来设置一个按钮或分隔线或按钮组的风格。一个按钮的风格决定了按钮是如何显示的，以及它对用户的输入是如何反应的。

在调用 SetButtonStyle 之前，调用 GetButtonStyle 成员函数来获取该按钮或分隔线的风格。

请参阅　CToolBar::GetButtonStyle

CToolBar::SetButtonText

BOOL SetButtonText(int *nIndex*, LPCTSTR *lpszText*);

返回值

如果成功则返回非零值；否则返回 0。

参数

nIndex

要设置其文本的按钮的索引。

lpszText

指向要被设置到一个按钮上的文本。

说明

此成员函数用来设置一个按钮上的文本。

请参阅 CToolBar::GetButtonText

CToolBar::SetHeight

void SetHeight(int *cyHeight*);

参数

cyHeight

是工具条的以像素表示的高度。

说明

此成员函数用来将工具条的高度设置为在 *cyHeight* **中指定的以像素表示的值。**

在调用 SetSize **之后，可以使用这个成员函数来重新设置这个标准工具条的高度。如果这个高度太小，则按钮将被从底部剪切。**

如果没有调用这个函数，则框架使用该按钮的尺寸来确定此工具条的高度。

请参阅 CToolBar::SetSizes, CToolBar::SetButtonInfo, CToolBar::SetButtons

CToolBar::SetSizes

void SetSizes(SIZE *sizeButton*, SIZE *sizeImage*);

参数

sizeButton

每个按钮的用像素表示的尺寸。

sizeImage

每个图像的以像素表示的尺寸。

说明

此成员函数用来将工具条的按钮的尺寸设置为在 *sizeButton* **中指定的以像素表示的尺寸。** *sizeImage* **参数必须包含在工具条位图中的图像的以像素表示的尺寸。**sizeButton **中的空间必须足够大以能存放在宽度上加** 7 **个像素，在高度上**

加 6 个像素的按钮图像。此函数还用来设置工具条的高度以使它能与按钮相适合。

只有当工具条不采用“Windows 界面软件设计”中推荐的按钮和图像尺寸时，才调用这个函数。

请参阅 CToolBar::LoadBitmap, CToolBar::SetButtonInfo, CToolBar::SetButtons,