

Microsoft  
**Visual Basic 6.0**  
控件参考手册

北京希望电脑公司

Microsoft Press

美国微软出版社 Microsoft Visual Studio 中文版系列图书

编程的利器 • 知识的迸发

# Microsoft Visual Basic 6.0

## 控 件 参 考 手 册

〔美〕Microsoft 公司 著

希望图书创作室 译

本书配套光盘包含两部分内容:

1. 与本书配套的电子书
2. 赠送“精通 Visual Basic 5.0”多媒体学习软件

北京希望电脑公司

北京希望电子出版社

1999

## 内 容 简 介

本书是美国微软出版社授权的系列中文版图书之一。本书系统地介绍了 Visual Basic 6.0 中各种控件包括: AdoData, Animation, MSChart, MSComm, CoolBar, Data, DataGrid, DataCombo, DataTimePicker, FlatScrollBar, MSFlexGrid, MSHFlexGrid, CellBackColor, CellForeColor, ImageCombo, Microsoft Internet Transfer, ListView, MAPIMessages, MultimediaMCI, MouthView, MaskedEdit, SSTab, PictureClip, RichTextBox, StatusBar, SysInfo, TabStrip, TreeView, Winsock, 以及对以前版本中的某些控件进行了必要的修改。在每种控件的单元中详细介绍了该控件的属性、方法和事件, 并适当地插入了一些使用示例, 以帮助读者进一步领会其含义和使用方法。本书是完整、权威的有关 Visual Basic 6.0 控件大全, 可供 VB 编程和开发人员快速查阅和使用, 以提高开发效率。无论对于 VB 初学者还是有经验的编程开发人员, 本书都是重要的参考手册。

本书配套光盘包括两部分内容: 1. 与本书配套的电子书; 2. 赠送“精通 Visual Basic 5.0”多媒体学习软件。

需购买本书及配套电子图书或需技术支持的读者, 可直接与北京海淀 8721 信箱书刊部(邮编 100080)联系, 电话: 010-62562329, 62541992, 62531267,

或传真：**010-62579874**。

本书封四贴有微软出版社（**Microsoft Press**）防伪标签，无标签者不得出售，违者必究。

## 版 权 声 明

本书英文版名为“**Microsoft Visual Basic 6.0 Controls Reference**”，由 **Microsoft** 出版社出版，版权归 **Microsoft** 出版社所有。本书中文版由 **Microsoft** 出版授权出版。未经出版者书面许可，本书的任何部分不得以任何手段复制或传播。

**Microsoft Visual Basic 6.0 控件参考手册**

〔美〕**Microsoft** 公司 著

希望图书创作室 译

北京希望电脑公司出品

北京希望电子出版社 出版

北京海淀路 82 号（100080）

责任编辑：周凤明

新华书店、新华书店音像发行所发行      各地新华书店及软件专卖店经售

\* \* \* \* \*

版次：1999 年 2 月

第一版

开本：787×1092

1/16

字数：1633 千字

新出音管[1998]210 号

ISBN 7-980023-47-1/TP • 43

定价：110.00 元（1CD，含配套书）

印次：1999 年 2 月第

一次印刷

印张：77.25

印数：1—5000

## 译 者 序

Visual Basic 是微软公司力推的编程工具，它具有易学易用、编程简单、程序集成化程度高等特点，是许多开发人员的首选开发工具。

为了方便开发人员的编程工作，Visual Basic 提供了许多现成的对象库和控件，极大地提高了程序员的开发效率。然而，在使用 Visual Basic 的过程中，我们发现，如果能对 Visual Basic 所提供的控件体系有一个全面而详细的了解，将会大大缩短开发过程。

本书详细介绍了 Visual Basic 6.0 中的固有控件和 ActiveX 控件。针对数据访问和 Internet 应用，Visual Basic 6.0 在 Visual Basic 5.0 基础上又提供了许多新的控件，同时 Visual Basic 6.0 对 Visual Basic 5.0 中某些控件也进行了必要的改进。所有这些，无疑都将进一步方便广大的 VB 开发人员。

本书中介绍的控件包括 CoolBar 控件、ADO 数据控件、Data 控件、DataGrid 控件、DataCombo 控件、DataRepeater 控件、DateTimeProvider 控件、FlatScrollBar 控件、MSFlexGrid 控件、MSHFlexGrid 控件、ImageCombo 控件、Microsoft Internet Transfer 控件、MAPIMessage 控件、Multimedia MCI 控件、MonthView 控件、Masked Edit 控件、SSTab 控件、PictureClip 控件、RichTextBox 控件、Slider 控件、StatusBar 控件、SysInfo 控件、TabStrip 控

件、ToolBar 控件、TreeView 控件、UpDown 控件和 WinSock 控件。在每一控件单元内详尽地介绍了该控件的属性、方法、事件，而且在必要的地方，给出了详尽的使用示例，可以帮助读者进一步领会含义和使用方法。

参与本书翻译工作的有吴卫、陈永刚、陈建勇、林明、吕翊、冯炜、汤涛、刘云龙、陆桢、连桦、殷秋风和唐春生等，柳超声、叶凡、张洵等对本书进行了仔细地校对。在本书的翻译过程中，各位译者之间通力合作，力求反映出原书的风格和用意。然而，由于译者水平有限，纰漏也在所难免。如有不正确的地方，恳请读者批评指正。



# 引 言

本指南中 Visual Basic ActiveX 控件是按照字母表的顺序排列，并包括以下内容：

- 按字母顺序排列的 ActiveX 控件以及该控件的属性、方法和事件。
- 附录列出了多个 ActiveX 控件所共有的属性、方法和事件。

## 本书中的编程风格

本书中使用下面的编程规定。更详细的信息，请参阅《Microsoft Visual Basic 6.0 程序员指南》第五章“编程基础”。

- 关键字的首字符大写：

`' Sub, If, ChDir, Print, 和 True 是关键字。`

`Print "Title Page"`

- ♦ 使用行标签而不是行数字。在错误处理例程中使用行标签：

`ErrorHandler:`

`Power = conFailure`

End Function

- 引号代表注释：

'这是注释；这两行

'在程序运行时将被忽略。

- Sub, Function 和 Property 过程中的控制流模块和语句都进行缩排：

```
Private Sub cmdRemove_Click ()
```

```
    Dim Ind As Integer
```

```
    Ind = lstClient.ListIndex      ' Get index
```

```
    ' Make sure list item is selected.
```

```
    If Ind >= 0 Then
```

```
        lstClient.RemoveItem Ind    ' Remove It
```

```
                                    ' from list box
```

```
        ' Display number.
```

```
        lblDisplay.Caption = lstClient.ListCount    Else
```

```
        Beep                                     ' If nothing selected , beep.
```

```
    End If
```

```
End Sub
```

- 如果某行太长以致于一行显示不下，可以使用续行符在下一行继续显示，续行符是单个的空格，后跟一个下划线（ \_ ）：

```
Sub Form_MouseDown ( Button As Integer _
```

Shift As Integer, X As Single, Y As Single )

- 常量以两个小写字符做前缀，比如标准的 Visual Basic 常量以“vb”为前缀，数据访问常量以“db”为前缀。例如：

vbTitleHorizontal

dbAppendOnly

在本书中，用户定义的常量通常都以“con”为前缀，而且是大小写字符混用。  
例如：

conYourOwnC

# 目 录

第一部分

第二部分



[返回总目录](#)

## 目 录

|                        |     |
|------------------------|-----|
| ADO Data 控件.....       | 3   |
| Animation 控件.....      | 29  |
| MSChart 控件 .....       | 41  |
| MSComm 控件 .....        | 356 |
| CoolBar 控件 .....       | 403 |
| Data 控件 .....          | 437 |
| DataGrid 控件.....       | 486 |
| DataCombo 控件.....      | 633 |
| DateTimePicker 控件..... | 700 |
| FlatScrollBar 控件 ..... | 728 |
| MSFlexGrid 控件.....     | 735 |

|                                       |     |
|---------------------------------------|-----|
| MSHFlexGrid 控件.....                   | 739 |
| CellBackColor 和 CellForeColor 属性..... | 762 |
| FormatString 属性.....                  | 833 |

## ADO Data 控件

ADO Data 控件与内部 Data 控件以及 Remote Data 控件(RDC)相似。ADO Data 控件使用户能使用 Microsoft ActiveX Data Objects (ADO) 快速地创建一个到数据库的连接。

### 说明

在设计时，您可以通过首先将 `ConnectionString` 属性设置为一个有效的连接字符串，然后将 `RecordSource` 属性设置为一个适合于数据库管理者的语句来创建一个连接。您也可以将 `ConnectionString` 属性设置为定义连接的文件名。该文件是由“数据链接”对话框产生的，当您单击“属性”窗口中的 `ConnectionString`，然后单击“生成”或“选择”时，该对话框出现。

您可以通过将 `DataSource` 属性设置为 ADO Data 控件，把 ADO Data 控件连接到一个数据绑定的控件，例如 `DataGrid`，`DataCombo` 或 `DataList` 控件。

在运行时，您可以动态地设置 `ConnectionString` 和 `RecordSource` 属性来更改数据库。或者，您可以将 `Recordset` 属性直接设置为一个原先已经打开的记录集。

## 属性

CacheSize 属性 (ADO)，CommandTimeout 属性 (ADO)，CommandType 属性 (ADO)，ConnectionString 属性 (ADO)，ConnectionTimeout 属性 (ADO)，CursorLocation 属性 (ADO)，CursorType 属性 (ADO)，LockType 属性 (ADO)，MaxRecords 属性 (ADO)，Mode 属性 (ADO)，Provider 属性 (ADO)，Recordset 属性 (ADO 数据控件)，RecordSource 属性 (ADO 数据控件)，Password 属性 (ADO 数据控件)，UserName 属性 (ADO 数据控件)，BackColor, ForeColor 属性，Height, Width 属性，Left, Top 属性，TabIndex 属性，Tag 属性，Visible 属性，Align 属性，DragIcon 属性，DragMode 属性，CausesValidation 属性，Orientation 属性，TabStop 属性，Appearance 属性，Caption 属性，HelpContextID 属性，Index 属性 (控件矩阵)，Name 属性，Parent 属性，Font 属性，Container 属性，Object 属性，ToolTipText 属性，BOFAction, EOFAction 属性，WhatsThisHelpID 属性。

## 方法

UpdateControls 方法 (ADO 数据控件)，Refresh 方法，SetFocus 方法，Drag 方法，Move 方法，ZOrder 方法，ShowWhatsThis 方法。

## 事件

EndOfRecordSet ( ConnectionEvent ) 方法 ( ADO ) ， FetchComplete (RecordsetEvent) 方法 (ADO)，FetchProgress (RecordsetEvent) 方法 (ADO)，WillChangeField 和 FieldChangeComplete (ConnectionEvent) 方法 (ADO)，



WillChangeRecord 和 RecordChangeComplete (ConnectionEvent) 方法 (ADO), WillMove 和 MoveComplete (ConnectionEvent) 方法 (ADO), Error 事件 (ADO 数据控件), DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, MouseDown, MouseUp 事件, MouseMove 事件, Validate 事件。

请参阅

ADO 对象模型, 数据窗体向导, 数据窗体向导—介绍, DataMember 属性, DataSource 属性, 使用 ADO 数据控件。

## CommandTimeout 属性 (ADO)

指示在终止尝试和产生错误之前执行命令期间需等待的时间。

应用于

Connection 对象 (ADO), Command 对象 (ADO)。

设置和返回值

设置或返回长整型值, 该值指示等待命令执行的时间 (单位为秒)。默认值为 30。

## 说明

使用 `Connection` 对象或 `Command` 上的 `CommandTimeout` 属性，允许由于网络拥塞或服务器负载过重产生的延迟而取消 `Execute` 方法调用。如果在 `CommandTimeout` 属性中设置的时间间隔内没有完成命令执行，将产生错误，然后 ADO 将取消该命令。如果将该属性设置为零，ADO 将无限期等待直到命令执行完毕。请确保正在写入代码的提供者和数据源支持 `CommandTimeout` 功能。

`Connection` 对象的 `CommandTimeout` 设置不会对相同 `Connection` 上 `Command` 对象的 `CommandTimeout` 设置产生影响，即 `Command` 对象的 `CommandTimeout` 属性不继承 `Connection` 对象的 `CommandTimeout` 的值。

在 `Connection` 对象上，打开 `Connection` 后，`CommandTimeout` 属性将保持读/写。

## 请参阅

`ConnectionTimeout` 属性 (ADO)。

## `CommandType` 属性 (ADO)

指示 `Command` 对象的类型。

# 设置和返回值

设置或返回以下某个 CommandTypeEnum 值。

| 常量                 | 说明  |
|--------------------|---|
| AdCmdText          | 将 CommandText 作为命令或存储过程调用的文本化定义进行计算   |
| AdCmdTable         | 将 ommandText 作为其列全部由内部生成的 SQL 查询返回的表格的名称进行计算  |
| AdCmdTableDirect   | 将 CommandText 作为其列全部返回的表格的名称进行计算  |
| AdCmdStoredProc    | 将 CommandText 作为存储过程名进行计算   |
| AdCmdUnknown       | 默认值。CommandText 属性中的命令类型未知  |
| AdCommandFile      | 将 CommandText 作为持久 Recordset 文件名进行计算  |
| AdExecuteNoRecords | 指示 CommandText 为不返回行的命令或存储过程（例如，插入数据的命令）。如果检索任意行，则将丢弃这些行且并不返回。它总是与 adCmdText 或 adCmdStoredProc 进行组合 |

## 说明

使用 CommandType 属性可优化 CommandText 属性的计算。

如果 CommandType 属性的值等于 adCmdUnknown（默认值），系统的性能将会降低，因为 ADO 必须调用提供者以确定 CommandText 属性是 SQL 语句、还是存储过程或表格名称。如果知道正在使用的命令的类型，可通过设置 CommandType 属性指令 ADO 直接转到相关代码。如果 CommandType 属性

与 `CommandText` 属性中的命令类型不匹配，调用 `Execute` 方法时将产生错误。

`adExecuteNoRecords` 常量通过最小化内部处理来提高性能。该常量不独立使用，它总是与 `adCmdText` 或 `adCmdStoredProc` 组合（如 `adCmdText+adExecuteNoRecords`）一起使用。如果与 `Recordset.Open` 方法一起使用 `adExecuteNoRecords`，或者该方法使用 `Command` 对象都将产生错误。

## ConnectionTimeout 属性 (ADO)

指示在终止尝试和产生错误前建立连接期间所等待的时间。

### 设置和返回值

设置或返回指示等待连接打开时间的 `Long`(长整型)值（单位为秒）。默认值为 15。

### 说明

如果由于网络拥塞或服务器负载过重导致的延迟使得必须放弃连接尝试时，请使用 `Connection` 对象的 `ConnectionTimeout` 属性。如果打开连接前所经过的时间超过 `ConnectionTimeout` 属性上设置的时间，将产生错误，并且 ADO 将取消该尝试。如果将该属性设置为零，ADO 将无限等待直到连接打开。请确认正在对其编写代码的提供者会支持 `ConnectionTimeout` 功能。

连接关闭时 `ConnectionTimeout` 属性为读/写，而打开时其属性为只读。

请参阅

`CommandTimeout` 属性（ADO）。

## CursorLocation 属性 (ADO)

设置或返回游标引擎的位置。

设置和返回值

设置或返回可设置为以下某个常量的长整型值。

| 常量                     | 说明                           |
|------------------------|------------------------------|
| <code>adUseNone</code> | 没有使用游标服务（该常量已过时并且只为了向后兼容才出现） |

| 常量                       | 说明  |
|--------------------------|---|
| <code>adUseClient</code> | 使用由本地游标库提供的客户端游标。本地游标引擎通常允许使用的许多功能可能是驱动程序提供的游标无法使用的，因此使用该设置对于那些将要启用的功能是有好处的。 <code>adUseClientBatch</code> 与 <code>adUseClient</code> 同义，也支持向后兼容性 |

续表

|             |   |
|-------------|---|
| adUseServer | 默认值。使用数据提供者或驱动程序提供的游标。这些游标有时非常灵活，对于其他用户对数据源所作的更改具有额外的敏感性。但是，Microsoft Client Cursor Provider（如已断开关联的记录集）的某些功能无法由服务器端游标模拟，通过该设置将无法使用这些功能 |
|-------------|---|

## 说明

该属性允许在可用于提供者的各种游标库中进行选择。通常，可以选择使用客户端游标库或位于服务器上的某个游标库。

该属性设置仅对属性已经设置后才建立的连接有影响。更改 **CursorLocation** 属性不会影响现有的连接。

对于 **Connection** 或关闭的 **Recordset** 该属性为读/写，而对打开的 **Recordset** 该属性为只读。

**Connection.Execute** 游标将继承该设置。**Recordset** 将自动从与之关联的连接中继承该设置。

**Remote Data Service 用法** 当用于客户端 (ADOR) **Recordset** 或 **Connection** 对象时，只能将 **CursorLocation** 属性设置为 **adUseClient**。

请参阅

通过 ADO 使用 OLE DB 提供者。

## EndOfRecordset (ConnectionEvent)方法(ADO)

移动行时如果超过记录集结尾，则调用 EndOfRecordset 方法。

语法

EndOfRecordset *pfMoreData*, *adStatus*, *pRecordset*

参数

*pfMoreData*

VARIANT\_BOOL，在处理该事件期间有可能将新记录追加到 *pRecordset*。在 EndOfRecordset 返回前添加数据，然后将该参数设置为 True 以指示 Recordset 的新结尾。

*adStatus*

EventStatusEnum 状态值。

当调用 EndOfRecordset 时，如果引发事件的操作成功，将该参数设置为 adStatusOK。如果该方法无法请求取消引发该事件的操作，则设置为 adStatusCantDeny。

在 `EndOfRecordset` 返回前，将该参数设置为 `adStatusUnwantedEvent` 可避免后续的通知。

### *pRecordset*

`Recordset` 对象，发生该事件所针对的 `Recordset`。

### 说明

如果 `Recordset.MoveNext` 操作失败，则可能发生 `EndOfRecordset` 事件。

当用户可能因调用 `MoveNext` 而移过 *pRecordset* 结尾时，将调用该事件的处理程序。使用该方法用户可以从数据库中检索到更多记录并将其追加到 *pRecordset* 的结尾。在这种情况下，用户要将 *pfMoreData* 设置为 `VARIANT_TRUE`，并从 `EndofRecordset` 返回。在此之后用户可以再次调用 `MoveNext` 以访问新检索到的记录。

## Error 事件（ADO Data 控件）

只有在没有执行任何 `Visual Basic` 代码而发生了一个数据访问错误的情况下，才会发生这个事件。

### 语法

```
object_Error([Index As Integer,] ByVal ErrorNumber As Long, Description As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As
```



String, ByVal *HelpContext* As Long, *fCancelDisplay* As Boolean)

Error 事件的语法包括下面这些部分：

| 部分                    | 描述                         |
|-----------------------|----------------------------|
| <i>object</i>         | 一个对象表达式，其值为“应用于”列表中的一个对象   |
| <i>index</i>          | 如果控件在一个控件数组中，此值可以标识该控件     |
| <i>errorNumber</i>    | 本地错误号码                     |
| <i>description</i>    | 对错误的描述                     |
| <i>scode</i>          | 服务器返回的错误代码                 |
| <i>source</i>         | 错误的来源                      |
| <i>helpFile</i>       | 包含该错误详细信息的帮助文件的路径          |
| <i>helpContext</i>    | 帮助主题的上下文号码                 |
| <i>fCancelDisplay</i> | 一个布尔值，可以设置这个值来取消对错误消息的显示。如 |
| <i>y</i>              | “设置值”中所描述的                 |

设置

*fCancelDisplay* 的设置值为：

| 常量    | 值  | 描述         |
|-------|----|------------|
| False | 0  | 继续         |
| True  | -1 | （缺省）显示错误消息 |

# 说明

无论何时，只要某个不是由 Visual Basic 引起的错误中止了一个操作，就会发生 Error 事件。错误可以分为两类：由 ADO 产生的错误和一般性错误（例如，内存不足错误）。如果错误的原因是 ADO，则 ADO Data 控件就会在描述字符串中添加与上下文相关的文本。

## Mode 属性 (ADO)

指示用于更改在 Connection 中的数据的数据的可用权限。

### 设置和返回值

设置或返回以下某个 ConnectModeEnum 的值。

| 常量                  | 说明                |
|---------------------|-------------------|
| adModeUnknown       | 默认值。表明权限尚未设置或无法确定 |
| adModeRead          | 表明权限为只读           |
| adModeWrite         | 表明权限为只写           |
| adModeReadWrite     | 表明权限为读/写          |
| adModeShareDenyRead | 防止其他用户使用读权限打开连接   |

续表

|                      |                  |
|----------------------|------------------|
| adModeShareDenyWrite | 防止其他用户使用写权限打开连接  |
| adModeShareExclusive | 防止其他用户打开连接       |
| adModeShareDenyNone  | 防止其他用户使用任何权限打开连接 |

说明

使用 Mode 属性可设置或返回当前连接上提供者正在使用的访问权限。Mode 属性只能在关闭 Connection 对象时方可设置。

**Remote Data Service 的用法** 当用于客户端 Connection 对象时，Mode 属性只能设置为 adModeUnknown。

Password 属性（ADO Data 控件）

设置 ADO Recordset 对象创建过程中所使用的口令。

语法

```
object.Password [= string]
```

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>strings</i> | 为在 UserName 属性中命名的用户设置口令 |

## 说明

该属性设置值是只写的，它只能在代码中提供，而不能从 **Password** 属性中读出。

## Recordset 属性（ADO Data 控件）

返回或设置对下一级 ADO Recordset 对象的引用。

## 语法

*object*.Recordset [= *recordset*]

Recordset 属性的语法包括下面这些部分：

| 部分               | 描述                       |
|------------------|--------------------------|
| <i>object</i>    | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>recordset</i> | 一个记录集对象                  |

## 说明

利用 Recordset 属性，可以使用 ADO 的 ADODB.Recordset 对象的方法、属性和事件。

必须在 Set 语句中使用 Recordset 属性，如下例所示：

```
Dim rsNwind As New ADODB.Recordset
```

'此处没有显示用于打开此 ADO Recordset 对象的代码。

```
Set ADODC1.Recordset = rsNwind
```

如果要对 Recordset 对象的事件进行编程，请使用 WithEvents 关键字声明一个对象变量，如下例所示。

```
Option Explicit
```

```
Dim WithEvents rsNames As ADODB.Recordset
```

```
Private Sub Form_Load()
```

```
    Set rsNames = New ADODB.Recordset
```

```
    ' 此处没有显示用于创建此记录集的代码。
```

```
    rsNames.MoveFirst ' 移动到记录集的开始位置。
```

```
    Set ADODC1.Recordset = rsNames
```

```
    With Text1
```

```
        Set .DataSource = ADODC1
```

```
        .DataField = "Name"
```

```
    End With
```

```
End Sub
```

```
Private Sub rsNames_FieldChangeComplete(ByVal cFields As Long, _  
    ByVal Fields As Variant, ByVal pError As ADODB.Error, adStatus As _  
    ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
```

```
    Debug.Print "New Name", pRecordset!Name
```

End Sub

## RecordSource 属性（ADO Data 控件）

返回或设置语句或返回一个记录集的查询。

### 语法

*object*.RecordSource [= *value* ]

RecordSource 属性的语法包括下述部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象       |
| <i>value</i>  | 一个字符串表达式，它指定了一个记录源，如“设置值”中所描述的 |

### 设置

*value* 的设置值为：

| 设置值 | 描述        |
|-----|-----------|
| 表名称 | 一个数据库表的名称 |

续表

SQL 查询

一个有效的 SQL 字符串，该字符串使用了适合于数据源的语法

## UpdateControls 方法（ADO Data 控件）

从控件的 ADO Recordset 对象中获取当前行，并在绑定到此控件的控件中显示相应的数据。

语法

*object*.UpdateControls

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

说明

使用这个方法将绑定控件的内容恢复为其初始值，如同用户更改了数据，然后又决定取消了这些更改。

除了不会导致任何事件的发生之外，这个方法产生的效果与使当前行再次成为当前行一样。由于此方法不会调用任何事件，您可以使用这个方法来简化一个更新操作，因为它不会触发任何多余的验证和更改事件过程。

## UserName 属性（ADO Data 控件）

返回或设置一个值，该值代表了 ADO Recordset 对象的一个用户。

### 语法

*object.UserName [= value]*

UserName 属性的语法包括下面这些部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象        |
| <i>value</i>  | 一个字符串表达式，其中包含了一个用户名，如“设置值”中所描述的 |

### 设置

用户名的语法取决于数据源。

### 说明

如果 UserName 和 Password 都已提供，则控件会使用这些值来创建一个连接字符串（ConnectionString 属性）。



## WillChangeField 和 FieldChangeComplete (ConnectionEvent)方法 (ADO)

WillChangeField 方法在更改 Recordset 中的一个或多个 Field 对象值的挂起操作前调用。FieldChangeComplete 方法在一个或多个 Field 对象值已经更改后调用。

### 语法

WillChangeField *cFields*, *Fields*, *adStatus*, *pRecordset*

FieldChangeComplete *cFields*, *Fields*, *pError*, *adStatus*, *pRecordset*

### 参数

*cFields*

长整型值，*Fields* 中的 Field 对象数目。

*Fields*

变体型数组，包含带有挂起更改内容的 Field 对象。

*pError*

Error 对象，说明当 adStatus 值为 adStatusErrorsOccured 时所发生的错误，否则将不对它进行设置。

*adStatus*

EventStatusEnum 状态值。

当调用 WillChangeField 时，如果引发事件的操作成功，该参数设置为 adStatusOK；如果该方法无法请求取消挂起操作，则设置为 adStatusCantDeny。

当调用 FieldChangeComplete 时，如果引发事件的操作成功，该参数设置为 adStatusOK，如果操作失败，则设置为 adStatusErrorsOccurred。

在 WillChangeField 返回前，将该参数设置为 adStatusCancel 可请求取消挂起的操作。

在 FieldChangeComplete 返回前，将该参数设置为 adStatusUnwantedEvent 避免后续的通知。

*pRecordset*

Recordset 对象，发生该事件所针对的 Recordset 。

说明

WillChangeField 或 FieldChangeComplete 事件可因下列 Recordset 操作而发生：Value 和带有字段及数组参数值的 Update。

## WillChangeRecord 和 RecordChangeComplete(ConnectionEvent)方法 (ADO)

WillChangeRecord 方法在 Recordset 中的一个或多个记录（行）更改前调用。RecordChangeComplete 方法在一个或多个记录更改后调用。

### 语法

WillChangeRecord *adReason*, *cRecords*, *adStatus*, *pRecordset*

RecordChangeComplete *adReason*, *cRecords*, *pError*, *adStatus*, *pRecordset*

### 参数

*adReason*

EventReasonEnum 值，指定该事件的原因。它的值可以是 adRsnAddNew, , adRsnDelete, adRsnUpdate, adRsnUndoUpdate, adRsnUndoAddNew, adRsnUndoDelete。

*cRecords*

Long(长整型)值，更改（影响）的记录数目。

*pError*

Error 对象，说明当 *adStatus* 值为 adStatusErrorsOccurred 时所发生的错误，否则将不对它进行设置。

## *adStatus*

EventStatusEnum 状态值。

当调用 WillChangeRecord 时，如果引发事件的操作成功，该参数设置为 adStatusOK。如果该方法无法请求取消挂起的操作，则设置为 adStatusCantDeny。

当调用 RecordChangeComplete 时，如果引发事件的操作成功，则该参数设置为 adStatusOK，如果操作失败，则设置为 adStatusErrorsOccurred。

在 WillChangeRecord 返回前，将该参数设置为 adStatusCancel 可请求取消引发该事件的操作。

在 RecordChangeComplete 返回前，将该参数设置为 adStatusUnwantedEvent 可避免后续的通知。

## *pRecordset*

Recordset 对象，发生该事件所针对的记录集。

## 说明

WillChangeRecord 或 RecordChangeComplete 事件可因下列 Recordset 操作而发生：Update,Delete,CancelUpdate,AddNew,UpdateBatch 和 CancelBatch。

在 WillChangeRecord 事件中，Recordset Filter 属性设置为 adFilterAffectedRecords。在处理事件时更改该属性是不合法的。

## WillChangeRecordset 和 RecordsetChangeComplete(ConnectionEvent) 方法 (ADO)

WillChangeRecordset 方法在挂起的操作更改 Recordset 前调用。  
RecordsetChangeComplete 方法在 Recordset 更改后调用。

### 语法

WillChangeRecordset *adReason, adStatus, pRecordset*  
RecordsetChangeComplete *adReason, pError, adStatus, pRecordset*

### 参数

#### *adReason*

EventReasonEnum 值，指定该事件的原因。它的值可以是  
adRsnReQuery,adRsnReSynch,adRsnClose,adRsnOpen。

#### *adStatus*

EventStatusEnum 状态值。

当调用 WillChangeRecordset 时，如果引发事件的操作成功，则该参数设置为 adStatusOK。如果该方法无法请求取消挂起的操作，则设置为 adStatusCantDeny。

当调用 RecordsetChangeComplete 时，如果引发事件的操作成功，则该参数设置为 adStatusOK；如果操作失败，则设置为

adStatusErrorsOccurred; 如果与以前接受的 WillChangeRecordset 事件关联的操作已经取消, 则设置为 adStatusCancel。

在 WillChangeRecordset 返回前, 将该参数设置为 adStatusCancel 以请求取消挂起操作。

在 WillChangeRecordset 或 RecordsetChangeComplete 返回前, 将该参数设置为 adStatusUnwantedEvent 可避免后续的通知。

#### *pError*

Error 对象, 说明当 adStatus 值为 adStatusErrorsOccurred 时所发生的错误, 否则将不对它进行设置。

#### *pRecordset*

Recordset 对象, 发生该事件所针对的 Recordset。

#### 说明

WillChangeRecordset 或 RecordsetChangeComplete 事件可因下列 Recordset 操作而发生: Requery, Resync, Close, Open 和 Filter。

### WillMove 和 MoveComplete (ConnectionEvent)方法(ADO)

WillMove 方法在挂起操作更改 Recordset 中的当前位置前调用。MoveComplete 方法则在 Recordset 的当前位置更改后调用。

## 语法

*WillMove* *adReason*, *adStatus*, *pRecordset*

*MoveComplete* *adReason*, *pError*, *adStatus*, *pRecordset*

## 参数

*adReason*

EventReasonEnum 值，指定该事件的原因。它的值可以是 adRsnMoveFirst, adRsnMoveLast, adRsnMoveNext, adRsnMovePrevious, adRsnMove 或 adRsnRequery。

*pError*

Error 对象，说明当 adStatus 值为 adStatusErrorsOccurred 时所发生的错误，否则将不对它进行设置。

*adStatus*

EventStatusEnum 状态值。

当调用 WillMove 时，如果引发事件的操作成功，则该参数设置为 adStatusOK。如果该方法无法请求取消挂起的操作，则设置为 adStatusCantDeny。

当调用 MoveComplete 时，如果引发事件的操作成功，则该参数设置为 adStatusOK。如果操作失败，则设置为 adStatusErrorsOccurred。

在 **WillMove** 返回前，将该参数设置为 **adStatusCancel** 可请求取消挂起的操作。在 **MoveComplete** 返回前，将该参数设置为 **adStatusUnwantedEvent** 可避免后续的通知。

*pRecordset*

**Recordset** 对象。发生该事件所针对的记录集。

说明

**WillMove** 或 **MoveComplete** 事件可因下列 **Recordset** 操作而发生：  
**Open,Move,MoveFirst** 、  
**MoveLas,MoveNext,MovePrevious,Bookmark,AddNew,Delete,Requery** 和 **Resync**。



## Animation 控件

**Animation** 控件允许你创建显示动画的按钮，如点击一个按钮播放一个.avi 文件。该控件只能播放没有声音的.avi 文件，并且 **Animation** 控件只能播放解压缩的或使用 **RLE**（行程长度编码，**Run-Length Encoding**）压缩的.avi 文件。

### 说明

如果试图加载包含声音的.avi 文件或该控件不支持的格式，将返回一个错误（错误码 35752）。

该控件的一个例子是 **Windows 95** 中的文件拷贝过程。该过程在拷贝时使用了一个

**Animation** 控件，显示了一个文件从一个文件夹飞入另一个文件夹。

### 属性

**AutoPlay** 属性，**BackStyle** 属性(**Animation** 控件)，**Center** 属性，**Height**,**Width** 属性，**TabIndex** 属性，**DragIcon** 属性，**DragMode** 属性，**hWnd** 属性，**TabStop** 属性，**HelpContextID** 属性，**Name** 属性，**Parent** 属性，**Container** 属性，**ToolTipText** 属性，**WhatsThisHelpID** 属性，**OLEDropMode** 属性（**ActiveX** 控件），**Height**,**Width** 属性（**ActiveX** 控件），**Index** 属性（**ActiveX** 控件），**Left**,**Top** 属性（**ActiveX**

控件），Tag 属性（ActiveX 控件），Visible 属性（ActiveX 控件），Object 属性（ActiveX 控件），BackColor,ForeColor 属性（ActiveX 控件），Enabled 属性（ActiveX 控件），hWnd 属性（ActiveX 控件）。

## 方法

Close 方法（Animation 控件），Open 方法（Animation 控件），Play 方法，Stop 方法（Animation 控件），SetFocus 方法，Drag 方法，Move 方法，ZOrder 方法，ShowWhatsThis 方法，OLEDrag 方法（ActiveX 控件）。

## 事件

DragDrop 事件，DragOver 事件，GotFocus 事件，LostFocus 事件，MouseDown 事件，MouseUp 事件，MouseMove 事件，Validate 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedBack 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Click 事件（ActiveX 控件），DbClick 事件（ActiveX 控件）。

## 请参阅

Multimedia MCI 控件，PictureBox 控件，Using the Animation 控件。

## 示例

下面的例子使用 Open 对话框打开一个.avi 文件并开始自动播放。要使用该

例子，在一个窗体中放置一个 **Animation** 控件和一个 **CommonDialog** 控件，将该例子代码加入到窗体的 **Declarations** 段。要运行该例子，选择要打开的.avi 文件。

```
Private Sub Animation1_Click ()  
    With CommonDialog1  
        .Filter = "avi (*.avi)|*.avi"  
        .ShowOpen  
    End With  
    With Animation1  
        .Autoplay = True  
        .Open CommonDialog1.FileName  
    End With  
End Sub
```

## AutoPlay 属性

当一个.avi 文件加载到该控件时，返回或设置一个值以确定 **Animation** 控件是否将自动播放该.avi 文件。

应用于

**Animation** 控件

# 语法

```
object.Animation [ = boolean ]
```

Animation 属性的语法包括如下两部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值为一个 Animation 控件 |
| <i>boolean</i> | 布尔表达式指明是否打开 Animation 属性   |

# 设置

*boolean* 的设置如下：

| 设置    | 描述                                 |
|-------|------------------------------------|
| True  | 一旦.avi 文件加载到 Animation 控件中将连续不断地播放 |
| False | 加载了.avi 文件后，使用了 Play 方法后才开始播放      |

# 数据类型

Integer (Boolean)

# 说明

使用 Animation 属性播放的.avi 文件将一直重复播放，直到 Animation 设置为 False。

请参阅

Center 属性。

## BackStyle 属性（Animation 控件）

返回或设置指示 **Animation** 控件在绘制动画时是在透明的背景或动画片断中指定的背景色。在程序运行时该属性为只读。

应用于

Animation 控件

语法

*object*.BackStyle[ = *value* ]

BackStyle 属性的语法包括下列部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>object</i> | 一个对象表达式，其值为一个 <b>Animation</b> 控件 |
| <i>value</i>  | 表示透明的数值表达式，如“设置”所示                |

# 设置

*value* 的值如下:

| 设置 | 描述                               |
|----|----------------------------------|
| 0  | (缺省值) 透明——控件的背景色是可见的             |
| 1  | 不透明——在动画片断文件中指定的背景色填充控件，覆盖任何其他颜色 |

# 说明

在播放动画时，可以使用 **BackStyle** 属性指定使用控件的背景色而不是使用动画自己的背景色。

# 数据类型

Integer(Boolean)

请参阅

**Center** 属性。

# Center 属性

确定该动画是否在 **Animation** 控件的中央位置。当该属性设置为 **Ture**(缺省值)时，.avi 文件就在 **Animation** 控件的中央播放。当设置为 **False** 时，.avi 文件

的位置从控件的(0,0)坐标开始计算。

应用于

Animation 控件

语法

*object.Center*[=*boolean*]

Center 属性的语法有如下几个部分：

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 要求的。一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>boolean</i> | 要求的。一个布尔表达式，确定控件中的 AVI 文件是否居中 |

数据类型

Integer(Boolean)

说明：

如果 Center 属性设置为 True，而.avi 帧的尺寸又比控件大，则动画的边将不显示。

请参阅

Animatin 控件，AutoPlay 属性，BackStyle 属性（Animation 控件），Close 方法（Animation 控件），Open 方法（Animation 控件），Play 方法，BackStyle 属性（UserControl 对象）。

## Close 方法（Animation 控件）

Close 方法导致 Animation 控件关闭当前播放的.avi 文件。如果 Animation 控件没有加载文件，Close 方法不做任何事情，也不报告错误。

应用于

Animation 控件

语法

*object*.Close

object 置换元表示一个对象表达式，其值为一个 Animation 控件。

说明

要停止所播放的.avi 文件可以使用 Stop 方法。然而，如果 AutoPlay 属性已经设置为 True，将该属性设置为 False 将停止播放该文件。



请参阅

Center 属性，Stop 方法（Animation 控件）。

## Open 方法（Animation 控件）

打开要播放的.avi 文件。如果 AutoPlay 属性设置为 True，则文件一加载就开始播放。其将重复播放，直到关闭了.avi 文件或将 AutoPlay 属性设置为 False。

应用于

Animation 控件

语法

*object.Open file*

Open 方法的语法包含了如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 要求的。一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>file</i>   | 要求的。要播放的文件名                  |

## 说明

**Animation** 控件不能播放带声音数据的.avi 文件，并且 **Animation** 控件只能播放没有压缩的.avi 文件或使用行程长度编码的.avi 文件。如果使用 **Open** 打开一个包含声音数据的.avi 文件或不支持的压缩文件，则返回出错信息。

## 请参阅

**AutoPlay** 属性。

## Play 方法

在 **Animation** 控件中播放.avi 文件。

## 应用于

**Animation** 控件

## 语法

*object.Play[=repeat, start, end]*

**Play** 方法的语法包含了如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 要求的。一个对象表达式，其值为一个 <b>Animation</b> 控件     |
| <i>repeat</i> | 可选参数，指定该动画重复次数的整数。缺省值-1 将导致无限循环播放         |
| <i>start</i>  | 可选参数，指定起始帧位置的整数。缺省值 0，表示从头开始播放。最大值是 65535 |
| <i>end</i>    | 可选参数，指定结束帧位置的整数。缺省值-1，表示最后一帧。最大是 65535    |

## 数据类型

### Integer

### 说明

使用 **Stop** 方法结束.avi 文件的播放。

请参阅

**Stop** 方法（**Animation** 控件）。

## Stop 方法（**Animation** 控件）

停止在 **Animation** 控件中播放的.avi 文件。

应用于

Animation 控件

语法

*object*.Stop

*object* 置换元表示一个对象表达式，其值为 Animation 控件。

说明

Stop 方法停止的动画只是用 Play 方法播放的动画。当 AutoPlay 属性设置为 True 时，使用 Stop 方法将返回一个错误（错误码 35759）。

请参阅

Animation 控件, Close 方法(Animation 控件), Open 方法(Animation 控件), Play 方法。

# MSChart 控件

以图形方式显示数据的图表。

语法

MSChart

说明

MSChart 控件支持下面特征：

- 真正三维显示。
- 支持所有主要的图表类型。
- 数据网格，随机数据和数据矩阵。

MSChart 控件与一个数据网格（DataGrid 对象）关联。该数据网格存放了要显示的数据。数据网格也可以包含用于图表中标识系列或类别的标签。图表应用程序设计者在数据网格中插入数据或从报表或矩阵中输入数据。

属性

ActiveSeriesCount 属性, AllowDithering 属性, AllowDynamicRotation 属性, AllowSelections 属性, AllowSeriesSelection 属性, AutoIncrement 属性, Visible

属性 (MSChart), Backdrop 属性, Chart3d 属性, ChartData 属性, ChartType 属性, Column 属性, ColumnCount 属性, ColumnLabel 属性 (MSChart), ColumnLabelCount 属性, ColumnLabelIndex 属性, Data 属性 (MSChart), DataGrid 属性, DoSetCursor 属性, DrawMode 属性 (MSChart), Footnote 属性, FootnoteText 属性, Legend 属性, MousePointer 属性, Plot 属性, RandomFill 属性, Repaint 属性, Row 属性 (MSChart), RowCount 属性 (MSChart), RowLabel 属性 (MSChart), RowLabelCount 属性, RowLabelIndex 属性, SerieColumn 属性, SeriesType 属性, ShowLegend 属性, Stacking 属性, TextLengthType 属性, Title 属性 (MSChart), TitleText 属性, DataBindings 属性, Drag 方法, TabIndex 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), Height, Width Properties 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), DataMember 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), DataSource 属性 (ActiveX 控件), HWnd 属性 (ActiveX 控件), ActiveSeriesCount 属性, AllowDithering 属性, AllowDynamicRotation 属性, AllowSelections 属性, AllowSeriesSelection 属性, AutoIncrement 属性, Visible 属性 (MSChart), Backdrop 属性, Chart3d 属性, ChartData 属性, ChartType 属性, Column 属性, ColumnCount 属性, ColumnLabel 属性 (MSChart), ColumnLabelCount 属性, ColumnLabelIndex 属性, Data 属性 (MSChart), DataGrid 属性, DoSetCursor 属性, DrawMode 属性

(MSChart),Footnote 属性,FootnoteText 属性,Legend 属性,MousePointer 属性 (MSChart),Plot 属性 ,RandomFill 属性 ,Repaint 属性 ,Row 属性 (MSChart),RowCount 属性(MSChart),RowLabel 属性(MSChart),RowLabelCount 属性,RowLabelIndex 属性,SeriesColumn 属性,SeriesType 属性,ShowLegend 属性,Stacking 属性,TextLengthType 属性,Title 属性 (MSChart),TitleText 属性,DataBindings 属性,Drag 方法,TabIndex 属性,DragIcon 属性,DragMode 属性,CausesValidation 属性,TabStop 属性,HelpContextID 属性,Name 属性,Parent 属性,Container 属性,ToolTipText 属性,WhatsThisHelpID 属性,OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), Height,Width Properties 属性(ActiveX 控件),Index 属性(ActiveX 控件),Left,Top Properties 属性(ActiveX 控件),Tag 属性(ActiveX 控件),Visible 属性(ActiveX 控件),Object 属性(ActiveX 控件),DataMember 属性(ActiveX 控件),BorderStyle 属性(ActiveX 控件),Enabled 属性(ActiveX 控件),DataSource 属性(ActiveX 控件),hWnd 属性 (ActiveX 控件)

## 方法

EditCopy 方法, EditPaste 方法, GetSelectedPart 方法, Layout 方法, SelectPart 方法, ToDefaults 方法, TwipsToChartPart 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法(ActiveX 控件), Refresh 方法(ActiveX 控件), EditCopy 方法, EditPaste 方法, GetSelectedPart 方法, Layout 方法, SelectPart 方法, ToDefaults 方法, TwipsToChartPart 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方

法(ActiveX 控件), Refresh 方法(ActiveX 控件)

## 事件

AxisActivated 事件, AxisLabelActivated 事件, AxisLabelSelected 事件, AxisLabelUpdated 事件, AxisSelected 事件, AxisTitleActivated 事件, AxisTitleSelected 事件, AxisTitleUpdated 事件, AxisUpdated 事件, ChartActivated 事件, CharSelected 事件, ChartUpdated 事件,

DataUpdated 事件, DonePainting 事件, FootnoteActivated 事件, FootnoteSelected 事件, FootnoteUpdated 事件, LegendActivated 事件, LegendSelected 事件, LegendUpdated 事件, PlotActivated 事件, PlotSelected 事件, PlotUpdated 事件, PointActivated 事件, PointLabelActivated 事件, PointLabelSelected 事件, PointLabelUpdated 事件, PointSelected 事件, PointUpdated 事件, SeriesActivated 事件, SeriesSelected 事件, SeriesUpdated 事件, TitleActivated 事件, TitleSelected 事件, TitleUpdated 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, Validate 事件, OLECompleteDrag 事件(ActiveX 控件), OLEDragDrop 事件(ActiveX 控件), OLEDragOver 事件(ActiveX 控件), OLEGiveFeedback 事件(ActiveX 控件), OLESetData 事件(ActiveX 控件), OLEStartDrag 事件(ActiveX 控件), Click 事件(ActiveX 控件), DblClick 事件(ActiveX 控件), KeyDown,KeyUp 事件(ActiveX 控件), KeyPress 事件(ActiveX 控件), MouseDown,MouseUp 事件(ActiveX 控件), MouseMove 事件(ActiveX 控件)



请参阅

AngleUnits 常量, Axis 对象, AxisGrid 对象, AxisScale 对象, AxisTitle 对象, Backdrop 对象, Brush 对象, CategoryScale 对象, Coor 对象, DataGrid 对象, DataPoint 对象, DataPointLabel 对象, Fill 对象, Footnote 对象, Frame 对象, Intersection 对象, Label 对象, LCoor 对象, Legend 对象, Light 对象, LightSource 对象, Location 对象, Marker 对象, Pen 对象, Plot 对象, PlotBase 对象, Rect 对象, Series 对象, SeriesMarker 对象, SeriesPosition 对象, Shadow 对象, StatLine 对象, TextLayout 对象, Tick 对象, Title 对象, ValueScale 对象, View3D 对象, VtColor 对象, VtFont 对象, Wall 对象, Weighting 对象, Error Messages (MSChart 控件), 使用 MSChart 控件。

示例

下面的例子显示了 8 行 8 列的三维数据图表，并设置了图例参数。

```
Private Sub Command1_Click()
```

```
    With MSChart1
```

```
        Displays a 3d chart with 8 columns and 8 rows  
        data.
```

```
        .ChartType = VtChChartType3dBar
```

```
        .ColumnCount = 8
```

```
        .RowCount = 8
```

```
        For column = 1 To 8
```

```

        For row = 1 To 8
            .Column = column
            .Row = row
            .Data = row * 10
        Next row
    Next column
    ' Use the chart as the backdrop of the legend.
    .ShowLegend = True
    .SelectPart VtChPartTypePlot, index1, index2, _
    index3, index4
    ' EditCopy
    ' SelectPart VtChPartTypeLegend, index1, _
    index2, index3, index4
    .EditPaste
End With
End Sub

```

## ActiveSeriesCount 属性

根据 DataGrid 对象的列数和显示的图表类型，返回在图表中显示的系列数。

应用于

MSChart 控件。

语法

*object.ActiveSeriesCount*

*object* 置换元素表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

DataGrid 对象。

## Add 方法（LightSources 集合）

给 LightSources 集合增加一个 LightSoure 对象。

应用于

LightSoures 集合。

语法

*object.Add(x,y,z,intensity)*

Add 方法的语法有如下几个部分：

| 部分               | 描述                     |
|------------------|------------------------|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>x,y,z</i>     | 整数，指定光源的位置             |
| <i>intensity</i> | 单精度数据。指定光源密度           |

### 说明

将 *x,y* 和 *z* 都设置为 0 将产生 **VtChInvalidArgument** 错误。

## AllowDithering 属性

返回或设置一个值，该值确定在 8bit 的彩色显示器上为了使用 **MSChart** 控件自己的调色板和增强图表显示效果是否禁止颜色抖动。

### 应用于

**MSChart** 控件。

### 语法

*object*. AllowDithering [=*boolean*]

AllowDithering 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定是否允许颜色抖动，如“设置”所示 |

## 设置

*boolean* 值的设置如下：

| 设置    | 描述                              |
|-------|---------------------------------|
| True  | 允许颜色抖动                          |
| False | (缺省)使用 MSChart 控件的调色板来增强颜色匹配和显示 |

## AllowDynamicRotation 属性

返回或设置一个值，该值指明用户是否可以按下 Ctrl 键显示旋转光标以便交互式地旋转三维图表。

应用于

MSChart 控件。

语法

*object.AllowDynamicRotation [=boolean]*

AllowDynamicRotation 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定是否允许动态旋转，如“设置”所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述               |
|-------|------------------|
| True  | (默认)用户可以使用光标旋转图表 |
| False | 用户不能交互式地旋转图表     |

AllowSelections 属性

返回或设置一个值，该值指明用户是否可以选择图表对象。

应用于  
MSChart 控件。

语法

*object.AllowSelection [=boolean]*

AllowSelection 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定是否允许选择对象，如“设置”所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                 |
|-------|--------------------|
| True  | (默认)用户可以交互式地选择图表对象 |
| False | 用户不能交互式地选择图表对象     |

## AllowSeriesSelection 属性

返回或设置一个值，该值指明当用户单击单个的图表数据点时是否选择一个系列。

应用于

MSChart 控件。

语法

*object*.AllowSeriesSelection [=*boolean*]

AllowSeriesSelection 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定是否选择一个系列，如“设置”所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                       |
|-------|--------------------------|
| True  | (默认)用户可以单击一个数据点而选择一个系列   |
| False | 用户单击一个数据点只选择该数据点，不选择整个系列 |



## AmbientIntensity 属性

返回或设置照亮一个三维图表周围光线的密度。

应用于

Light 对象。

语法

*object.AmbientIntensity [=intensity]*

AmbientIntensity 属性的语法有如下几个部分：

| 部分               | 描述  |
|------------------|---|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>intensity</i> | 单精度数据。图表光源密度。有效值是 0 到 1。如果设置为 1，则不管打开的是什么光源，图表元素的所有边框都全部照亮。如果设置为 0，则周围光线不起作用；只有图表元素正对光源的边框被照亮 |

## AngleUnit 属性

返回或设置所有图表角度的度量单位。

应用于  
Plot 对象。  
语法

```
object.AngleUnit [=unit]
```

AngleUnit 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                         |
| <i>unit</i>   | 整数。描述度量单位的 <b>VtAngleUnits</b> 常量。角度可以是度，弧度或梯度 |

请参阅  
AngleUnits 常量。

### Auto 属性（CategoryScale）

返回或设置一个值，该值指明坐标轴是否自动改变尺度。  
应用于  
CategoryScale 对象。

语法

*object*.Auto [=*boolean*]

Auto 属性的语法有如下几个部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，指明是否显示部件，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | (默认)坐标轴根据显示的数据自动改变尺度   |
| False | 坐标轴尺度不自动更改。使用 DivisionsPerLabel 和 DivisionsPerTick 属性中的值决定尺度 |

请参阅

DivisionsPerLabel 属性，DivisionsPerTick 属性，LabelTick 属性，Auto 属性（Intersection），Auto 属性（Label），Auto 属性（SeriesMarker），Auto 属性（ValueScale）。

# Auto 属性（Intersection）

返回或设置一个值，该值确定 **Intersection** 对象是否使用 **Point** 属性值来放置坐标轴。

应用于

**Intersection** 对象。

语法

*object*.Auto [=*boolean*]

Auto 属性的语法有如下几个部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，指明是否显示部件，如“设置”中所 |

示

设置

*boolean* 值的设置如下：

| 设置    | 描述                        |
|-------|---------------------------|
| True  | 坐标轴显示在标准的位置               |
| False | 坐标轴放置在 <b>Point</b> 指定的位置 |

请参阅

Auto 属性 (CategoryScale)，Auto 属性 (Label)，Auto 属性 (SeriesMarker)，Auto 属性 (ValueScale)。

## Auto 属性 (Label)

返回或设置一个值，该值确定坐标轴标签是否自动旋转以便增强图表外观效果。

应用于

Label 对象。

语法

*object*.Auto [=*boolean*]

Auto 属性的语法有如下几个部分：

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>boolean</i> | 一个布尔表达式，指明旋转坐标轴标签显示状态，如“设置”中所示 |

## 设置

*boolean* 值的设置如下：

| 设置    | 描述                    |
|-------|-----------------------|
| True  | 坐标轴标签可以旋转             |
| False | 坐标轴标签不能旋转，长的标签显示可能不正确 |

## 请参阅

Auto 属性（CategoryScale），Auto 属性（Intersection），Auto 属性（SeriesMarker），Auto 属性（ValueScale）。

## Auto 属性（SeriesMarker）

返回或设置一个值，该值确定 SeriesMarker 对象是否将下一个可以标记分配给系列中的所有数据点。

应用于

SeriesMarker 对象。

语法

*object*.Auto [=boolean]

Auto 属性的语法有如下几个部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，指明标记如何分配，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                      |
|-------|-------------------------|
| True  | (默认)SeriesMarker 对象分配标记 |
| False | 可以分配一个自定义的标记            |

说明

如果想改变序列标记的类型，就将该属性设置为 **False**。如果 **DataPoint** 对象设置了 **Marker** 属性值，则该属性就自动设置为 **False**。

请参阅

Auto 属性 (CategoryScale)，Auto 属性 (Intersection)，Auto 属性 (Label)，Auto 属性 (ValueScale)。

### Auto 属性 (ValueScale)

返回或设置一个值，该值确定在绘制坐标轴的值时是否自动更改尺度。

应用于

ValueScale 对象。

语法

*object*.Auto [=*boolean*]

Auto 属性的语法有如下几个部分：

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>boolean</i> | 一个布尔表达式，指明是否自动放缩，如“设置”中所<br>示 |



## 设置

*boolean* 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 根据显示的数据自动放缩尺度  |
| False | 使用 <b>Minimum,Maximum,MajorDivisions</b> 和 <b>MinorDivisions</b> 属性的值放缩坐标轴 |

## 请参阅

Auto 属性（**CategoryScale**），Auto 属性（**Intersection**），Auto 属性（**Label**），Auto 属性（**SeriesMarker**）。

## AutoIncrement 属性

返回或设置一个值，该值确定在数据输入时设置当前数据点的属性是否增大而不必手工修改 **Column** 和 **Row** 属性。

## 应用于

MSChart 控件。

语法

*object*.AutoIncrement [=*boolean*]

AutoIncrement 属性的语法有如下几个部分：

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>boolean</i> | 一个布尔表达式，指明当前数据点是否自动增加，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | 当 Data 属性改变时，Row 属性改变为该列中的下一行。<br>如果你在一列的末尾，则 Column 属性增加到下一列 |
| False | （默认）当前数据点不增加  |

AutoLayout 属性

返回或设置一个值，该值确定一个 Plot 对象是手工还是自动布局。

应用于  
Plot 对象。

语法

*object*.AutoLayout [=*boolean*]

AutoLayout 属性的语法有如下几个部分：

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>boolean</i> | 一个布尔表达式，指定布局格式，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                                   |
|-------|--------------------------------------|
| True  | (默认)根据其他元素的大小和位置，Plot 对象自动确定相应的大小和位置 |
| False | 使用 Plot 对象规定的 LocationRect 属性值定位该对象  |

## Automatic 属性

返回或设置一个值，该值确定是否自动计算颜色值。这只用在绘制图表元素边框的画笔中。

应用于

VtColor 对象。

语法

*object*.Automatic[=*boolean*]

Automatic 属性的语法有如下几个部分：

| 部分             | 描述                               |
|----------------|----------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象           |
| <i>boolean</i> | 一个布尔表达式，指定是否自动计算颜色值，如“设置”<br>中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述              |
|-------|-----------------|
| True  | 自动取用图表系列的刷子颜色   |
| False | 颜色由 Value 的设置确定 |

## Axis 对象

图表中的一个坐标轴。

### 语法

Axis

### 说明

有三条数轴可以使用： $x$ ,  $y$  和  $z$ 。只有是 3D 图表的情况下  $z$  轴才可用。

### 属性

AxisTitle 属性, LabelLevelCount 属性, AxisGrid 属性, AxisScale 属性, CategoryScale 属性, Intersection 属性, Labels 属性, Pen 属性, Tick 属性, ValueScale 属性。

### 请参阅

AxisGrid 对象, AxisScale 对象, AxisTitle 对象, CategoryScale 对象, Intersection 对象, Labels 集合。

## 示例

下面的例子使用 *X Axis* 对象读取出现在 *X* 坐标轴上的标签级数。

```
Private Sub Command1_Click()  
    Dim XAxis As Object  
    Dim NumberOfLevels As Integer  
    ' Read the number of label level present on the X  
    ' Axis.  
    Set XAxis = MSChart1.Plot.Axis(VtChAxisIdX, 1)  
    NumberOfLevels = XAxis.LabelLevelCount  
    MsgBox "Number of Label Levels = " _  
        & Str(NumberOfLevels)  
End Sub
```

## Axis 属性

返回描述图表坐标的一个 Axis 对象引用。

应用于

Plot 对象。

语法

*object*.Axis(*axisID*)

Axis 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 要求的。对象表达式，其值是“应用于”列表中的一个对象                         |
| <i>axisID</i> | 要求的。一个整数式 VtChAxisId 常量，标识一个特定的坐标轴                 |
| <i>index</i>  | 可选的。保留，以便将来使用。当有多个坐标轴都有同一个 <i>axisID</i> 时来标识某个坐标轴 |

说明

有三条数轴可以使用：x,y 和 z。只有是 3D 图表的情况下 z 轴才可用。

请参阅

AxisID 常量。

AxisActivated 事件

当用户双击一个坐标轴时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisActivated(axisId As Integer ,axisIndex As Integer ,  
    mouseFlags As Integer, cancel As Integer )
```

AxisActivated 事件的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                    |
| <i>axisId</i>     | 整数，标识某个坐标轴，如“设置”中所示                       |
| <i>axisIndex</i>  | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有<br>1 是有效值 |
| <i>mouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中<br>所示          |
| <i>cancel</i>     | 整数，现在尚未使用                                 |

设置

事件处理器确定激活了哪个坐标轴，并将 *axisId* 设置为：



| 常量           | 描述              |
|--------------|-----------------|
| VtChAxisIdX  | 如果 $x$ 轴受到影响    |
| VtChAxisIdY  | 如果 $y$ 轴受到影响    |
| VtChAxisIdY2 | 如果第二个 $y$ 轴受到影响 |
| VtChAxisIdZ  | 如果 $z$ 轴受到影响    |

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

## AxisGrid 对象

一个图表坐标轴周围的平面区域。

语法

AxisGrid

属性

MajorPen 属性，MinorPen 属性。

## 示例

下面的例子将 AxisGrid 线条风格改为虚线。

```
Private Sub Command1_Click()  
    With MSChart1.Plot.Axis(VtChAxisIdY)  
        .CategoryScale.Auto = False  
        .ValueScale.MajorDivision = 10  
        .ValueScale.MinorDivision = 5  
        .AxisGrid.MajorPen.Style = VtPenStyleSolid  
        .AxisGrid.MajorPen..VtColor.Green = 255  
        .AxisGrid.MajorPen.Widte = 2  
        .AxisGrid.MajorPen.Style = VtPenStyleDashed  
        .AxisGrid.MajorPen.VtColor.Red = 255  
        .AxisGrid.MajorPen.Width = 1  
        .AxisGrid.MajorPen.Cpa = VtPenCapRound  
    End With  
End Sub
```

## AxisGrid 属性

返回描述图表坐标轴周围平面区域的 AxisGrid 对象的引用。

应用于

Axis 对象。

语法

*object*.AxisGrid

*object* 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

## AxisId 属性

返回与当前坐标轴相交的坐标轴。

应用于

Intersection 对象。

语法

*object*.AxisId

*object* 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

返回值

标识相交坐标轴的整数。

请参阅  
AxisId 常量。

## AxisLabelActivated 事件

当用户双击坐标轴标签时产生该事件。  
应用于  
MSChart 控件。

### 语法

```
Private Sub object_AxisLabelActivated (axisId As Integer,axisIndex As Integer,labelSetIndex As Integer,labelIndex As Integer,mouseFlags As Integer,cancel As Integer)
```

AxisLabelActivated 事件的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>axisId</i> | 整数，标识某个坐标轴，如“设置”中所示    |

续表

|                      |  |
|----------------------|--|
| <i>axisIndex</i>     | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>labelSetIndex</i> | 整数，标识双击鼠标的标签级。标签级从 1 开始标识              |
| <i>labelIndex</i>    | 整数，当前尚未使用                              |
| <i>mouseFlagss</i>   | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中所示           |
| <i>cancel</i>        | 整数。现在尚未使用                              |

设置

事件处理器确定激活了哪个坐标轴标签，并将 *axisId* 设置为：

| 常量           | 描述                   |
|--------------|----------------------|
| VtChAxisIdX  | 如果 <i>x</i> 轴受到影响    |
| VtChAxisIdY  | 如果 <i>y</i> 轴受到影响    |
| VtChAxisIdY2 | 如果第二个 <i>y</i> 轴受到影响 |
| VtChAxisIdZ  | 如果 <i>z</i> 轴受到影响    |

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

AxisLabelSelected 事件， AxisLabelUpdated 事件。

## AxisLabelSelected 事件

当用户单击坐标轴标签时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisLabelSelected (axisId As Integer, axisIndex As Integer,  
    labelSetIndex As Integer, labelIndex As Integer, mouseFlags As Integer, cancel  
As Integer)
```

AxisLabelSelected 事件的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>axisId</i> | 整数，标识某个坐标轴，如“设置”中所示    |

续表

|                      |  |
|----------------------|--|
| <i>axisIndex</i>     | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>labelSetIndex</i> | 整数，标识单击鼠标的标签级。标签级从 1 开始                |
| <i>labelIndex</i>    | 整数，当前尚未使用                              |
| <i>mouseFlags</i>    | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中所示           |
| <i>cancel</i>        | 整数。现在尚未使用                              |

设置

事件处理器确定选择了哪个坐标轴标签，并将 *axisId* 设置为：

| 常量           | 描述                   |
|--------------|----------------------|
| VtChAxisIdX  | 如果 <i>x</i> 轴受到影响    |
| 常量           | 描述                   |
| VtChAxisIdY  | 如果 <i>y</i> 轴受到影响    |
| VtChAxisIdY2 | 如果第二个 <i>y</i> 轴受到影响 |
| VtChAxisIdZ  | 如果 <i>z</i> 轴受到影响    |

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                          | 描述              |
|-----------------------------|-----------------|
| VtChMouseFlagShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

AxisLabelActivated 事件，AxisLabelUpdated 事件。

## AxisLabelUpdated 事件

当坐标轴标签改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisLabelUpdated (axisId As Integer, axisIndex As Integer, labelSetIndex As Integer, labelIndex As Integer, updateFlags As Integer)
```

AxisLabelUpdated 事件的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>axisId</i> | 整数，标识某个坐标轴，如“设置”中所示    |



续表

|                    |   |
|--------------------|---|
| <i>axisIndex</i>   | 整数，保留将来使用。对于现在版本的 <b>MSChart</b> 控件，只有 1 是有效值 |
| <i>labelIndex</i>  | 整数，标识单击鼠标的标签级。标签级从 1 开始                       |
| <i>updateFlags</i> | 整数，提供更新标签的信息，如“设置”中所示                         |

设置

事件处理器确定更新了哪个坐标轴标签，并将 **axisId** 设置为：

| 常量                  | 描述                   |
|---------------------|----------------------|
| <b>VtChAxisIdX</b>  | 如果 <i>x</i> 轴受到影响    |
| <b>VtChAxisIdY</b>  | 如果 <i>y</i> 轴受到影响    |
| <b>VtChAxisIdY2</b> | 如果第二个 <i>y</i> 轴受到影响 |
| <b>VtChAxisIdZ</b>  | 如果 <i>z</i> 轴受到影响    |

事件处理器确定更新标签的影响，并将 *updateFlags* 设置为：

| 常量                       | 描述                     |
|--------------------------|------------------------|
| <b>VtChNoDisplay</b>     | 没有更新标记；图表显示不受影响（缺省为 0） |
| <b>VtChDisplayPlot</b>   | 更新将导致图表重新绘制            |
| <b>VtChLayoutPlot</b>    | 更新将导致图表重新布局            |
| <b>VtChDisplayLegend</b> | 更新将导致说明部分重新绘制          |

续表

VtChLayoutLegend

更新将导致说明部分重新布局

VtChLayoutSeries

更新将导致系列重新布局

VtChPostionSection

图表的一部分已被移动或改变了大小

请参阅

AxisLabelSelected 事件，AxisLabelActivated 事件。

## AxisScale 对象

控制如何在坐标轴上绘制图表。

语法

AxisScale。

属性

Hide 属性，LogBase 属性，PercentBasis 属性，Type 属性（MSChart）。

请参阅

Axis 对象，AxisScale 属性。

## 示例

下面的例子将为一个二维线性图表的  $x$  和  $y$  轴设置为百分比坐标或对数坐标。要使用该例子，在一个窗体上绘制一个 **MSChart** 控件和两个 **CommandButton** 控件。将下面的代码加入到 **Form** 对象的代码模块中并执行该程序。

```
Private Sub Command1_Click()  
    Dim axisID as VtChAxisId  
    ' Change both x and y axes to Percent scale for 2D  
    ' Line chart.  
    MSChart1.ChartType = VtChChartType2dLine  
    For axisID = VtChAxisIdX To VtChAxisIdY  
        With MSChart1.Plot.Axis(axisID).AxisScale  
            .Type = VtChScaleTypePercent  
            .PercentBasis = VtChPercentAxisBasisSumChart  
        End With  
    Next  
End Sub  
  
Private Sub Command2_Click()  
    Dim axisID as VtChAxisId  
    ' Change both x and y axes to Logarithmic scale for 2D  
    ' Line chart.  
    MSChart1.ChartType = VtChChartType2dLine
```

```
For AxisID = VtChAxisIdX To VtChAxisIdY
    With MSChart1.Plot.Axis(AxisID).AxisScale
        .Type = VtChScaleTypeLogarithmic
        .LogBase = 12
    End With
Next
End Sub
```

## AxisScale 属性

返回对描述如何在坐标轴上绘制数据的 **AxisScale** 对象的引用。

应用于

**Axis** 对象。

语法

*object*.AxisScale

**object** 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

请参阅

**AxisScale** 对象。

# AxisSelected 事件

当用户单击图表坐标轴时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisSelected( axisId As Integer , axisIndex As Integer ,  
                                MouseFlags As Integer, cancel As Integer )
```

AxisSelected 事件的语法有如下几个部分：

| 部分                 | 描述                                     |
|--------------------|--|
| <i>object</i>      | 对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>axisId</i>      | 整数，标识某个坐标轴，如“设置”中所示                    |
| <i>axisIndex</i>   | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>mMouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中所示           |
| <i>cancel</i>      | 整数。现在尚未使用                              |

设置

事件处理器确定选择了哪个坐标轴，并将 *axisId* 设置为：

| 常量             | 值 | 描述              |
|----------------|---|-----------------|
| VtChAxisIdX    | 0 | 如果 $x$ 轴受到影响    |
| VtChAxisIdY    | 1 | 如果 $y$ 轴受到影响    |
| VtChAxisIdY2   | 2 | 如果第二个 $y$ 轴受到影响 |
| VtChAxisIdZ    | 3 | 如果 $z$ 轴受到影响    |
| VtChAxisIDNone | 4 | 没有轴             |

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

## AxisTitle 对象

图表上的坐标标题。

语法

AxisTitle

## 属性

Text 属性（MSChart 控件）， TextLength 属性， Visible 属性（MSChart 控件）， Backdrop 属性， Font 属性（MSChart 控件）， TextLayout 属性， VtFont 属性。

请参阅

Axis 对象， Backdrop 对象。

## 示例

下面的例子使得坐标标题对三维图表的每个坐标轴都可见。

```
Private Sub Command1_Click()  
    ' Makes Axis title visible for all axes of a  
    ' 3D chart.  
    Dim axisID As VtChAxId  
    MSChart1.chartType = VtChChartType3dBar  
    For axisId = VtChAxisIdX To VtChAxisIdZ  
        With MSChart1.Plot.axis(axisId, 1)  
            .AxisTitle.VtFont.Size = 14  
            AxisTitle.Visible = True  
            .AxisTitle  
        Select Case axisId
```

```
Case 0
    .AxisTitle.text = "X Axis Title"
Case 1
    .AxisTitle.text = "Y Axis Title"
Case 2
    .AxisTitle.text = "2nd Y Axis Title"
Case 3
    .AxisTitle.text = "Z Axis Title"
End Select
End With
Next
End Sub
```

## AxisTitle 属性

返回与一个图表坐标关联的 AxisTitle 对象引用。

应用于

Axis 对象。



语法

*object*.AxisTitle

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

### AxisTitleActivated 事件

当用户单击坐标标题时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisTitleActivated( axisId As Integer , axisIndex As Integer ,  
                                         mouseFlags As Ingerger, cancel As Integer )
```

AxisTitleActivated 事件的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>AxisId</i> | 整数，标识某个坐标轴             |

续表

|                   |  |
|-------------------|--|
| <i>AxisIndex</i>  | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>MouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下                    |
| <i>Cancel</i>     | 整数。现在尚未使用                              |

设置

事件处理器确定激活了哪个坐标轴标题，并将 *axisId* 设置为：

| 常量             | 值 | 描述                    |
|----------------|---|-----------------------|
| VtChAxisIdX    | 0 | 如果 <i>x</i> 轴受到影响。    |
| VtChAxisIdY    | 1 | 如果 <i>y</i> 轴受到影响。    |
| VtChAxisIdY2   | 2 | 如果第二个 <i>y</i> 轴受到影响。 |
| VtChAxisIdZ    | 3 | 如果 <i>z</i> 轴受到影响     |
| VtChAxisIDNone | 4 | 没有轴                   |

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

AxisTitleSelected 事件，AxisTitleUpdated 事件。

# AxisTitleSelected 事件

当用户单击坐标标题时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisTitleSelected( axisId As Integer , axisIndex As Integer ,  
                                         mouseFlags As Ingerger, cancel As Integer )
```

AxisTitleSelected 事件的语法有如下几个部分：

| 部分                | 描述                                     |
|-------------------|--|
| <i>Object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>AxisId</i>     | 整数，标识某个坐标轴                             |
| <i>AxisIndex</i>  | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>MouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下                    |
| <i>Cancel</i>     | 整数。现在尚未使用                              |

设置

事件处理器确定选择了哪个坐标轴标题，并将 *axisId* 设置为：

| 常量             | 值 | 描述              |
|----------------|---|-----------------|
| VtChAxisIdX    | 0 | 如果 $x$ 轴受到影响    |
| VtChAxisIdY    | 1 | 如果 $y$ 轴受到影响    |
| VtChAxisIdY2   | 2 | 如果第二个 $y$ 轴受到影响 |
| VtChAxisIdZ    | 3 | 如果 $z$ 轴受到影响    |
| VtCHAxisIDNone | 4 | 没有轴             |

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

AxisTitleActivated 事件，AxisTitleUpdated 事件。

## AxisTitleUpdated 事件

当坐标标题改变时产生该事件。

应用于  
MSChart 控件。  
语法

```
Private Sub object_AxisTitleUpdated( axisId As Integer , axisIndex As Integer ,  
                                         UpdateFlags As Ingerger)
```

AxisTitleUpdated 事件的语法有如下几个部分:

| 部分                | 描述                                     |
|-------------------|--|
| <i>Object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>AxisId</i>     | 整数，标识某个坐标轴                             |
| <i>AxisIndex</i>  | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>UpdateFlag</i> | 整数，提供标题更新信息                            |

设置

事件处理器确定更新了哪个坐标轴标题，并将 *axisId* 设置为:

| 常量          | 值 | 描述                |
|-------------|---|-------------------|
| VtChAxisIdX | 0 | 如果 <i>x</i> 轴受到影响 |
| VtChAxisIdY | 1 | 如果 <i>y</i> 轴受到影响 |

续表

|                  |   |            |
|------------------|---|------------|
| VtChAxisIdY2     | 2 | 如果 y 轴受到影响 |
| VtChAxisIdZ      | 3 | 如果 z 轴受到影响 |
| VtChAxisIdIDNone | 4 | 没有轴        |

下表列出了 *updateFlags* 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

请参阅

AxisTitleSelected 事件，AxisTitleActivated 事件。

### AxisUpdated 事件

当坐标轴改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_AxisUpdated( axisId As Integer , axisIndex As Integer ,  
                                updateFlags As Ingerger)
```

AxisUpdated 事件的语法有如下几个部分：

| 部分                | 描述                                     |
|-------------------|--|
| <i>Object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>AxisId</i>     | 整数，标识某个坐标轴                             |
| <i>AxisIndex</i>  | 整数，保留将来使用。对于现在版本的 MSChart 控件，只有 1 是有效值 |
| <i>UpdateFlag</i> | 整数，提供标题更新信息                            |

设置

事件处理器确定更新哪个坐标轴，并将 *axisId* 设置为：

| 常量          | 值 | 描述                |
|-------------|---|-------------------|
| VtChAxisIdX | 0 | 如果 <i>x</i> 轴受到影响 |

续表

|                  |   |               |
|------------------|---|---------------|
| VtChAxisIdY      | 1 | 如果 y 轴受到影响    |
| VtChAxisIdY2     | 2 | 如果第二个 y 轴受到影响 |
| VtChAxisIdZ      | 3 | 如果 z 轴受到影响    |
| VtChAxisIdIDNone | 4 | 没有轴           |

下表列出了 *updateFlags* 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

## Backdrop 对象

图表元素后面的阴影或图案。

说明

plot 对象的 Backdrop 对象，为 VtFillStyleBrush 的可见效果设置 Style 属性。



语法

Backdrop

属性

Fill 属性, Frame 属性, Shadow 属性。

请参阅

Backdrop 属性, Fill 对象, Frame 对象。

## Backdrop 属性

返回描述图表后面的阴影、图案或图片的 Backdrop 对象的引用。

应用于

AxisTitle 对象, DataPointLabel 对象, Footnote 对象, Label 对象, Legend 对象, MSChart 控件, Plot 对象, Title 对象。

语法

*object*.Backdrop

object 置换元表示一个对象表达式, 其值为“应用于”列表中的一个列表。

## BarGap 属性

返回或设置二维棒图或一栏中的三维棒图之间的间距。

应用于

Plot 对象。

语法

*object*.BarGap [=value]

BarGap 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>value</i>  | 单精度数据。棒图的间距值。以棒图宽度的百分比来度量。<br>0 意味着棒图之间没有间距。100 意味着棒图之间的间距就是一个棒图宽度 |

## BaseHeight 属性

返回或设置一个三维图表以磅表示的基准线高度。

应用于  
PlotBase 对象。

语法

*object.BaseHeight* [=height]

BaseHeight 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>Height</i> | 单精度数据。基准线的高度           |

## Basis 属性

返回或设置饼图图表中各部分权重类型。

应用于  
Weighting 对象。

语法

*object.Basis* [=type]

Basis 属性的语法有如下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的一个对象          |
| <i>Type</i>   | 一个 VtChPieWeightBasis 常量，表明权重类型 |

请参阅

PieWeightBasis 常量。

## Blue 属性

返回或设置图表的 RGB 值的 R 部分。

应用于

VtColor 对象。

语法

*object*.Blue [=*b*]

Blue 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>B</i>      | 整数。蓝色值                 |

## 说明

RGB 指定红、绿、蓝相关颜色的相对密度，以导致显示指定颜色。标准 RGB 颜色有效范围是从 0 到 16777215。RGB 参数中任何大于 255 的值都当作 255。

## Brush 对象

图表元素的填充类型。

## 语法

Brush

## 属性

FillColor 属性, Index 属性(Brush), PatternColor 属性, Style 属性(MSChart)。

请参阅

Brush 属性。

## 示例

下面的例子使用 **Brush** 对象给图表背景设置垂直粗线型。

```
Private Sub Command1_Click()  
    ' Sets Backdrop to Fill - Brush Style.  
    MSChart1.Backdrop.Fill.Style = VtFillStyleBrush  
    ' Sets a pattern for the chart backdrop using the  
    ' Brush object.  
    With MSChart1.Backdrop.Fill.Brush  
        .Style = VtBrushStylePattern  
        .Index = VtBrushPatternBoldVertical  
    ' Sets Pattern to Bold Vertical lines.  
        .FillColor.Set 255, 0, 0           ' Fill Color = Red.  
        .PatternColor.Set 0, 0, 255       ' Pattern Color =  
                                           ' Blue.  
    End With  
End Sub
```

## Brush 属性

返回描述图表元素填充类型的 **Brush** 对象。

应用于

DataPoint 对象， Fill 对象， PlotBase 对象， Shadow 对象， Wall 对象。

语法

*object*.Brush

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

Brush 对象。

## Cap 属性

返回或设置一个值，该值确定线段如何结尾。

应用于

Pen 对象。

语法

*object*.Cap [=type]

Cap 属性的语法有如下几个部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象             |
| <i>type</i>   | 整数。一个 <b>VtPenCap</b> 常量，确定线段笔尖的格式 |

请参阅

**PenCap** 常量。

## CategoryScale 对象

隔栏坐标的尺度。

语法

**CategoryScale**

属性

**Auto** 属性（**CategoryScale**），**DivisionsPerLabel** 属性，**DivisionsPerTick** 属性，**LabelTick** 属性。

请参阅

**CategoryScale** 属性。



## 示例

下面的例子设置了隔栏坐标的尺度：

```
Private Sub Command1_Click()  
    ' Sets scaling attributes for a category axis.  
    MSChart1.ChartType = VtChChartType2dLine  
    With MSChart1.Plot.Axis(VtChAxisIdX)  
        .ValueScale.MajorDivision = 10  
        .ValueScale.MinorDivision = 5  
        .CategoryScale.Auto = False           ' Sets manual scaling.  
        .CategoryScale.DivisionsPerLabel = 2   ' Label appears every two  
                                                ' divisions.  
        .CategoryScale.DivisionsPerTick = 2    ' Ticks appear every two  
                                                ' divisions.  
        .CategoryScale.LabelTick = True        ' Labels displayed on top of  
                                                ' Tick marks.  
    End With  
End Sub
```

## CategoryScale 属性

返回一个描述隔栏坐标尺度信息的 CategoryScale 对象。

应用于

Axis 对象。

语法

*object*.CategoryScale

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

ValueScale 属性。

## Chart3d 属性

返回一个确定图表是否是三维的值。

应用于

MSChart 控件。

语法

*object*.Char3D

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

返回值

| 设置    | 描述      |
|-------|---------|
| True  | 图表是三维的  |
| False | 图表不是三维的 |

说明

一个图表是否是三维的要看 **ChartType** 属性的设置。

ChartActivated 事件

当用户双击 **Microsoft Chart** 控件而不是图表中的元素时产生该事件。

应用于

**MSChart** 控件。

语法

Private Sub *object*\_ChartActivated(*mouseFlags* As Integer, *cancel* As Integer)

AxisTitleActivated 事件的语法有如下几个部分：

| 部分                | 描述                              |
|-------------------|---------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象          |
| 部分                | 描述                              |
| <i>mouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”<br>所示 |
| <i>cancel</i>     | 整数。现在尚未使用                       |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

ChartSelected 事件，ChartUpdated 事件。

## ChartData 属性

返回或设置一个矩阵，该矩阵包含图表显示的值。

应用于

MSChart 控件。

语法

*object*.ChartData [=*data*]

ChartData 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>data</i>   | 可变。一个二维矩阵保存了绘制图表所需的数据  |

说明

如果多维矩阵的第一列包含字符串，这些字符串将成为图表的标签。

ChartData 是 MSChart 控件的缺省属性。

示例

下面的例子使用一个 Visual Basic 矩阵直接加载图表数据。要使用该例子，需要在一个窗体上绘制一个 MSChart 控件，将下面的代码拷贝到 Form 对象的代码模块中，执行该程序。

Option Explicit

Option Base 1

Private Sub Form\_Load()

Dim arrDataX( 3, 1 To 3)

arrData(1,1) = "Jan" ' Set the labels in the first series.

arrData(2,1) = "Feb"

arrData(3,1) = "Mar"

arrData(1,2) = 8

arrData(2,2) = 4

arrData(3,2) = 0.3

arrData(1,3) = 0.2

arrData(2,3) = 3

arrData(3,3) = 6.3

MSChart1.ChartData = arrData

End Sub

在这个例子中，使用 **Option Base 1** 语句将矩阵下界定义为 1 而不是缺省的 0。我们使用一个 **Variant** 矩阵，第一列的值设置为字符串变量，第二和第三列的值都设置为数值。这允许同时设置图表的标签和数据。注意，将矩阵定义为 **String** 类型也可以，只要第二和第三列中保存的是数值。如果你只设置图表的数据，可以定义矩阵为 **Integer**, **Long**, **Single** 或 **Double** 类型的。注意，这样做将导致图表标签是缺省行和列标签值。一维矩阵也可以跟二维矩阵一样工作，

只要后面的元素是数值或数值的字符串表达式。

下面的例子从图表中返回数据。本例中包含了一个循环语句，打印从图表中返回的矩阵。注意，使用了 **Lbound** 和 **Ubound** 函数确定图表矩阵的边界。要使用该例子，需要在一个窗体上绘制一个 **MSChart** 控件和一个 **CommandButton** 控件。将下面的代码拷贝到代码模块中，执行该例子。

Option Explicit

Private Sub Command1\_Click()

Dim y() As Variant

Dim i,j As Integer

y = MSChart1.ChartData

For i = LBound(y,2) To Ubound(y,2)

Debug.Print

For j = Lbound(y,1) To UBound(y,1)

Debug.Print y(j, i)

Next j

Next i

end sub

返回矩阵的下界是 0。返回的矩阵总是二维的 **Variant** 类型矩阵。因为 **ChartData** 是图表的缺省属性，对象的名称 **MSChart1** 可以替换为 **MSChart1.ChartData**。因此，你可以使用 **MSChart1 = data** 或 **data = MSChart1** 这样的语句。

## ChartSelected 事件

当用户双击 Microsoft Chart 控件而不是图表中的元素时产生该事件。

应用于

MSChart 控件。

语法

Private Sub *object*\_ChartSelected(*mouseFlags* As Integer, *cancel* As Integer )

AxisTitleActivated 事件的语法有如下几个部分：

| 部分                | 描述                           |
|-------------------|------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>mouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 整数。现在尚未使用                    |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：



| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| 常量                           | 描述              |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

ChartActivated 事件，ChartUpdated 事件。

## ChartType 属性

返回或设置用于显示图表的图表类型。

应用于

MSChart 控件。

语法

*object*.ChartType [=*type*]

ChartType 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>type</i>   | 整数。一个 VtChChartType 常量，指定图表类型 |

请参阅  
ChartType 常量。

## ChartUpdated 事件

当图表改变时产生该事件。  
应用于  
MSChart 控件。

### 语法

Private Sub *object*\_ChartUpdated(*updateFlags* As Integer)

| 部分                | 描述                     |
|-------------------|------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>updateFlag</i> | 整数，提供图表更新信息，如“设置”中所示   |

### 设置

下表列出了 *updateFlags* 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

请参阅

ChartActivated 事件， ChartSelected 事件。

### Clockwise 属性

返回或设置一个值，该值指明饼图是否按顺时针方向绘制。

应用于

Plot 对象。

语法

*object*.Clockwise [=boolean]

Clockwise 属性的语法有如下几个部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，控制饼图的绘制方向，如“设置”所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述              |
|-------|-----------------|
| True  | （缺省值）饼图按顺时针方向绘制 |
| False | 饼图按逆时针方向绘制      |

## Column 属性（MSChart 控件）

返回或设置数据格子中的当前列的数据。

应用于

MSChart 控件。

语法

```
object.Column [=col]
```

Column 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>col</i>    | 整数。当前数据列               |

说明

在你使用其他属性修改某列相应的图表系列或修改系列中任何的数据点之前，你必须选择该列。

ColumnCount 属性

返回或设置与图表相关的当前数据格子的列数。

应用于

MSChart 控件，DataGrid 对象。

语法

```
object.ColumnCount [=count]
```

ColumnCount 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>count</i>  | 数据列数                   |

请参阅

SetSize 方法。

## ColumnLabel 属性（DataGrid 对象）

返回或设置与图表相关的格子中数据列的标签。

应用于

DataGrid 对象。

语法

*object*.ColumnLabel( *column*, *labelIndex* ) [=text]

ColumnLabel 属性的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>column</i>     | 整数。标识数据列。列的计算从左到右，从 1 开始计算。包含标签的列不在列数计算范围之内 |
| 部分                | 描述  |
| <i>labelIndex</i> | 整数。标识一个标签。如果某列有多级标签，你必须指定一个。列标签从下到上计算，下面是 1 |
| <i>text</i>       | 字符串。列的标签文本                                  |

请参阅

ColumnLabel 属性（MSChart 控件）， RowLabel 属性（DataGrid）。

示例

下面的例子将给图表 *z* 轴增加三个列标签。要使用该例子，需要在一个窗体上绘制一个 MSChart 控件。将下面的代码拷贝到 Form 对象的代码模块中，并执行该程序。单击窗体就可以看到标签。

```
Option Explicit
Option Base 1
Private Sub Form_Click()
    With MSChart1
        .chartType = VtChChartType3dLine
```

```
.plot.Axis(VtChAxisIdZ).Labels(1).VtFont.Size = 24
```

```
.DataGrid.ColumnLabel(1,1) = "Label Z1"
```

```
.DataGrid.ColumnLabel(2,1) = "Label Z2"
```

```
.DataGrid.ColumnLabel(3,1) = "Label Z3"
```

```
End With
```

```
End Sub
```

```
Private Sub Form_Load ()
```

```
    ' The first series contains labels for the X axis.
```

```
    Dim arrData(4, 1 to 4)
```

```
    arrData(1,1) = "Jan"
```

```
    arrData(2,1) = "Feb"
```

```
    arrData(3,1) = "Mar"
```

```
    arrData(4,1) = "Apr"
```

```
    arrData(1,2) = 8
```

```
    arrData(2,2) = 4
```

```
    arrData(3,2) = 0.3
```

```
    arrData(4,2) = 3
```

```
    arrData(1,3) = 0.2
```

```
    arrData(2,3) = 3
```

```
    arrData(3,3) = 6.3
```

```
    arrData(4,3) = 2
```



```
arrData(1,4) = 5  
arrData(2,4) = 7  
arrData(3,4) = 2  
arrData(4,4) = 9  
MSChart1.ChartData = arrData
```

End Sub

## ColumnLabel 属性（MSChart）

返回或设置图表数据格子中与某列关联的标签文本。

应用于

MSChart 控件。

语法

*object*.ColumnLabel [=text]

ColumnLabel 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>text</i>   | 字符串。数据格子中与列关联的标签文本     |

说明

该属性设置由 Column 属性标识列的标签。

请参阅

ColumnLabel 属性（DataGrid）， RowLabel 属性（MSChart）。

ColumnLabelCount 属性

返回或设置图表数据格子中列标签的级数。

应用于

MSChart 控件， DataGrid 对象。

语法

*object*.ColumnLabelCount [=count]

ColumnLabelCount 属性的语法有如下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象           |
| <i>count</i>  | 整数。列标签级数。设置该属性增加或删除数据格子上的<br>标签级 |

说明

列标签的级数从下向上算起，以 1 开始。级从上面增加或删除。

请参阅

SetSize 方法。

ColumnLabelIndex 属性

返回或设置图表列标签的指定级。

应用于

MSChart 控件。

语法

*object*.ColumnLabelIndex [=*index*]

ColumnLabelIndex 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 整数。标识列的标签级数            |

# 说明

要在列中设置多级标签或返回标签的当前值，你必须先指定对哪个标签进行操作。列标签的级数从下向上算起，开始是 1。

## Component 属性

返回或设置标识数据点的标签类型。

应用于

DataPointLabel 对象。

### 语法

*object*.Component [=*type*]

Component 属性的语法有如下几个部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象             |
| <i>type</i>   | 整数。一个 VtChLabelComponent 常量，指定标签类型 |

请参阅

LabelComponent 常量。

## CompositeColumnLabel 属性

返回多级标签字符串，该字符串标识图表数据格子的列。

应用于

DataGrid 对象。

语法

*object*.CompositeColumnLabel(*column*)

CompositeColumnLabel 属性的语法有如下几个部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>column</i> | 整数。标识某个列。列计数从左到右，左边为 1。包含标签的列不计算在数据列之内 |

请参阅

CompositeRowLabel 属性。

## CompositeRowLabel 属性

返回多级标签字符串，该字符串标识图表数据格子的行。

应用于

DataGrid 对象。

语法

*object*.CompositeRowLabel(*row*)

CompositeColumnLabel 属性的语法有如下几个部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>row</i>    | 整数。标识某个行。行计数从上到下，开始为 1。包含标签的行不计算在数据行之内 |

请参阅

CompositeColumnLabel 属性。

## Coor 对象

描述图表的浮点数  $x$  和  $y$  坐标。

语法

Coor

属性

X 属性，Y 属性。

方法

Set 方法（Coor,LCoor）。

请参阅

LCoor 对象。

## Count 方法

返回集合中对象的数量。

应用于

SeriesCollection 集合, Series 集合。

语法

*object.Count*

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Custom 属性

返回或设置一个值，该值确定是否使用自定义的文本来标签图表的数据点。

应用于

DataPointLabel 对象。

语法

*object.Custom* [=*boolean*]

Custom 属性的语法有如下几个部分：



| 部分             | 描述                               |
|----------------|----------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象           |
| <i>boolean</i> | 一个布尔表达式，指定是否使用自定义文本，如“设置”<br>中所示 |

## 设置

*boolean* 值的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | 标签包含自定义文本   |
| False | 使用 <code>DataPointLabel</code> 对象的 <code>Components</code> 属性指定的信息<br>标签数据点 |

## Data 属性（MSChart）

返回或设置一个值，该值将插入到图表数据格子当前的数据点上。

## 应用于

MSChart 控件。

# 语法

*object.Data* [=value]

Data 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 整数。数据点的值               |

# 说明

如果当前的数据点已经有值，就替换为新值。重新绘制图表以显示新的数据。

## DataGrid 对象

# 语法

DataGrid

# 说明

DataGrid 对象配有行和列。你可以增减行、列以及阵列的标签，从而改变图表的外观。

## 属性

ColumnCount 属性, ColumnLabelCount 属性, ColumnLabel 属性 (DataGrid), CompositeColumnLabel 属性, CompositeRowLabel 属性, RowLabel 属性 (DataGrid),

RowCount 属性 (MSChart), RowLabelCount 属性。

## 方法

DeleteColumns 方法, DeleteColumnLabels 方法, DeleteRows 方法, DeleteRowLabels 方法, GetData 方法 (MSChart), InitializeLabels 方法, InsertColumns 方法, InsertColumnLabels 方法, InsertRows 方法, InsertRowLabels 方法, MoveData 方法, RandomDataFill 方法, RandomFillColumns 方法, RandomFillRows 方法, SetData 方法 (MSChart), SetSize 方法。

## 请参阅

DataGrid 属性。

## 示例

下面的例子设置了三维棒图图表参数，使用随机数据填充图表，并标签数据格子列。

Option Explicit

Option Base 1

```

Private Sub Command1_Click()
    Dim rowLabelCount , columnLabelCount, rowCount As Integer
    Dim columnCount , labelIndex, Column, Row As Integer
    MSChart1.ChartType = VtChChartType3dBar
    With MSChart1.DataGrid
        ' Set Chart parameters using methods.
        rowLabelCount = 2
        columnLabelCount = 2
        rowCount = 6
        columnCount = 6
        .SetSize RowLabelCount, RolumnLabelCount, _
        RowCount, ColumnCount
        ' Randomly fill in the data.
        .RandomDataFill
        ' Then assign labels to second Level.
        labelIndex = 2
        column = 1
        .ColumnLabel(column, labelIndex) = "Product 1"
        column = 4
        .ColumnLabel(column, labelIndex) = "Product 2"
        row = 1
        .RowLabel(row, labelIndex) = "1994"
    End With
End Sub

```

```
row = 4
```

```
.RowLabel(row, labelIndex) = "1995"
```

```
End With
```

```
End Sub
```

表示了一个包含 **MSChart** 控件标签和数据点的虚拟矩阵。

## DataGrid 属性

返回描述图表数据格子的 **DataGrid** 对象的引用。

应用于

**MSChart** 控件。

语法

*object*.DataGrid

**object** 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

# DataPoint 对象

## 语法

DataPoint

## 说明

通过 SeriesCollection 对象访问 DataPoints 集合。

**重点** 目前，DataPoints 集合只包含了一个成员。要访问该成员，必须使用-1，如下所示。

```
MSChart1.Plot.SeriesCollection(1).DataPoints(-1)_  
.Brush.FillColor.Set 0, 255, 255
```

## 属性

Bursh 属性，Offset 属性，DataPointLabel 属性，Marker 属性，EdgePen 属性。

## 方法

ResetCustom 方法，Select 方法。

请参阅

Brush 对象, DataPoints 属性, DataPoints 集合, DataPointLabel 对象。

示例:

下面的例子给数据点分配一个变量, 设置数据点的颜色和标记。

```
Private Sub Command1_Click()  
    ' Change the color and marker of First DataPoint in  
    ' the First Series.  
    With MSChart1.Plot.SeriesCollection(1).DataPoints(-1)  
        ' Change Data Point color to blue.  
        .Brush.Style = VtBrushStyleSolid  
        ' Set Color=Blue.  
        .Brush.FillColor.Set 0, 255, 255  
        ' Set DataPoint marker visible.  
        .Marker.Visible = True  
    End With  
End Sub
```

描述图表上单个数据点属性的 DataPionts 集合的一个成员。

## DataPointLabel 对象

图表数据点的标签。

语法

DataPointLabel

属性

Text 属性(MSChart),TextLength 属性,Backdrop 属性,Offset 属性,Component 属性, Custom 属性,LineStyle 属性,LocationType 属性, PercentFormat 属性, ValueFormat 属性,Font 属性 (MSChart) ,TextLayout 属性, TextLength 属性, VtFont 属性。

方法

ResetCustomLabel 方法, Select 方法。

请参阅

Coor 对象, DataPointLabel 属性。



## DataPointLabel 属性

返回描述单个图表数据点标签的 DataPointLabel 对象的引用。

应用于

DataPoint 对象。

语法

*object.DataPointLabel*

*object* 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

## DataPoints 集合

一组图表数据点。

语法

*object.DataPoints.(index)*

DataPoints 集合的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>index</i>  | 标识当前系列中某个数据点。在图表当前版本中，-1 是唯一有效的值。这允许你对系列中所有数据点的缺省设置作两次修改。但不能修改系列中单个数据点的设置 |

## 属性

Item 属性（MSChart）， Count 属性（ActireX 控件）。

请参阅

DataPoint 对象， DataPoints 属性。

## 示例

下面的例子设置了图表中每个数据点的标记。

```
Private Sub Command1_Click()
    With MSChart1.Plot.SeriesCollection(1).DataPoints(-1)
        ' Set DataPoint marker visible.
        .Marker.Visible = True
        .Marker.Style = VtMarKerStyleDiamond
    End With
End Sub
```

Datapoints 集合的 Item 方法使用 -1 参数给该系列中的所有数据点设置缺省属性。由 Item(-1) 返回的 Datapoint 对象可以与标准的 Datapoint 对象一样进行操作。对该缺省数据点属性的设置将影响到系列中所有数据点，除了那些单独设置属性的数据点。在下面的例子中，每一系列的数据点标签被设置在三个不同的位置。单个数据点的设置在缺省设置之前还是之后都没有分别；单独的设置总是覆盖缺省设置。可以使用 ResetCustom 方法删除单个数据点的设置，从而使用系列缺省设置。

Option Explicit

Option Base 1

Private Sub Command1\_Click ()

With MSChart1.Plot

.SeriesCollection.DataPoints.Item(-1)\_

.DataPointLabel.LocationType = VtChLabelLocationTypeAbovePoint

.SeriesCollection(1).DataPoints.Item(-1)\_

.DataPointLabel.LocationType = VtChLabelLocationTypeCenter

.SeriesCollection(2).DataPoints.Item(-1)\_

.DataPointLabel.LocationType = VtChLabelLocationTypeBase

End With

End Sub

## DataPoints 属性

返回一个描述图表系列数据点的 **DataPoint** 集合对象的引用。

应用于

**Series** 对象。

语法

*object*.DataPoints

**object** 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

## DataSeriesInRow 属性

返回或设置一个值，该值指明系列数据是否从图表的数据格子的一行或一列读取。

应用于

**Plot** 对象。

语法

*object*.DataSeriesInRow [=boolean]

DataSeriesInRow 属性的语法有如下几个部分：

| 部分             | 描述                              |
|----------------|---------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象          |
| <i>boolean</i> | 一个布尔表达式，控制系列数据如何读取，如“设置”<br>中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                 |
|-------|--------------------|
| True  | 系列数据从数据格子的行读取      |
| False | (缺省值)系列数据从数据格子的列读取 |

## DataUpdated 事件

当图表数据格子改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_DataUpdated (row As Integer, column As Integer, labelRow
As Integer,
    labelColumn As Integer, labelSetIndex As Integer, updateFlags As
Integer)
```

DataUpdated 事件的语法有如下几个部分：

| 部分                   | 描述                                  |
|----------------------|-------------------------------------|
| <i>object</i>        | 对象表达式，其值是“应用于”列表中的一个对象              |
| <i>row</i>           | 整数。指示数据格子所在的行                       |
| <i>column</i>        | 整数。指示数据格子所在的列                       |
| <i>labelRow</i>      | 整数。指示行标签                            |
| <i>labelColumn</i>   | 整数。指示列标签                            |
| <i>labelSetIndex</i> | 整数。标明标签的级数。标签级数从数轴开始，往外<br>计数，开始是 1 |
| <i>updateFlags</i>   | 整数，提供数据更新信息，如“设置”中所示                |

设置

下表列出了 *updateFlags* 常量。

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

## 说明

如果行和列非 0，将修改指定的数据单元。如果 *labelRow* 或 *labelColumn* 和 *labelSetIndex* 非 0，则修改指定的行或列标签。如果没有一个非 0，则没有可以修改的信息。

## 请参阅

MSChart 控件。

## DefaultPercentBasis 属性

返回坐标轴对图表的缺省百分比基数。

应用于

Plot 对象。

语法

*object.DefaultPercentBasis*

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

返回值

返回值是一个整数，指明缺省的坐标轴百分比基数。返回值可能是 `UtChPercentAxisBasis` 常量。

## DeleteColumnLabels 方法

删除与图表关联的数据格子中数据列的标签级数。

应用于

DataGrid 对象。

语法

*object.DeleteColumnLabels(labelIndex, count)*



DeleteColumnLabels 方法的语法有如下几个部分：

| 部分                | 描述   |
|-------------------|--|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                           |
| <i>labelIndex</i> | 整数。确定你要删除的第一级标签数。列标签级数是从下向上计算的，从 1 开始            |
| <i>count</i>      | 整数。指定你要删除的标签级数。删除的列数从 <i>labelIndex</i> 指定的列开始算起 |

请参阅

DeleteColumns 方法， DeleteRows 方法， DeleteRowLabels 方法。

## DeleteColumns 方法

从图表的数据格子中删除数据列及其关联的标签。

应用于

DataGrid 对象。

语法

*object.DeleteColumns(column, count)*

DeleteColumns 方法的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>column</i> | 整数。确定一个指定数据列。列数是从左向右计算的，从 1 开始 |
| <i>count</i>  | 整数。指定你要删除的列数                   |

请参阅

DeleteColumnLabels 方法， DeleteRows 方法， DeleteRowLabels 方法。

## DeleteRowLabels 方法

从图表的数据格子中删除数据行的标签级数。

应用于

DataGrid 对象。

语法

*object.DeleteRowLabels(labelIndex , count)*

DeleteRowLabels 方法的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                            |
| <i>labelIndex</i> | 整数。确定你要删除的第一级标签数。行标签级数是从右向左计算的，开始是 1              |
| <i>count</i>      | 整数。指定你要删除的标签级数。删除的行标签从 <i>labelIndex</i> 指定的行向左算起 |

请参阅

DeleteColumns 方法，DeleteColumnLabels 方法，DeleteRows 方法。

## DeleteRows 方法

从图表的数据格子中删除数据行及其关联的标签。

应用于

DataGrid 对象。

语法

*object.DeleteColumns(row, count)*

DeleteRows 方法的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>row</i>    | 整数。确定一个特定的数据行。行数是从上向下计算的，开始是 1 |
| <i>count</i>  | 整数。指定你要删除的行数                   |

请参阅

DeleteColumns 方法， DeleteColumnLabels 方法， DeleteRowLabels 方法。

## DepthToHeightRatio 属性

返回或设置用于表示图表深度的图表高度百分比。

应用于

Plot 对象。

语法

*object*.DepthToHeightRadio [=pctg]

DepthToHeightRatio 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>pctg</i>   | 单精度数据。图表高度百分比          |

## DivisionsPerLabel 属性

返回或设置标签之间间隔单位数。

应用于

CategoryScale 对象。

语法

*object*.DivisionsPerLabel [=num]

DivisionsPerLabel 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>num</i>    | 整数。指定间隔单位数量            |

## 说明

如果设置了该属性，则对象的 `Auto` 属性自动设置为 `False`。

## 请参阅

`Auto` 属性（`CategoryScale`），`DivisionsPerTick` 属性。

## DivisionsPerTick 属性

返回或设置主信号（`major tick`）标记之间的间隔单位数。

## 应用于

`CategoryScale` 对象。

## 语法

*object*.DivisionsPerTick [=*num*]

`DivisionsPerTick` 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>num</i>    | 整数。指定单位数量              |

## 说明

如果设置了该属性，则对象的 Auto 属性自动设置为 False。

## 请参阅

Auto 属性（CategoryScale）， DivisionsPerLabel 属性。

## DonePainting 事件

图表重新绘画后就产生该事件。

## 应用于

MSChart 控件。

## 语法

Private Sub *object*\_DonePainting()

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## DoSetCursor 属性

返回或设置一个值，该值指示图表是否设置鼠标。DoSetCursor 属性决定应用程序是否可以控制鼠标指针的外观。

应用于

MSChart 控件。

语法

*object*.DoSetCursor [*boolean*]

DoSetCursor 属性的语法有如下几个部分：

| 部分             | 描述                                |
|----------------|-----------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象            |
| <i>boolean</i> | 一个布尔表达式，指定是否使用自定义的鼠标，如“设置”<br>中所示 |

设置

*boolean* 值的设置如下：



| 设置    | 描述                 |
|-------|--------------------|
| True  | （缺省）应用程序可以控制鼠标指针外观 |
| False | 应用程序不能控制鼠标指针外观     |

## DrawMode 属性（MSChart）

返回或设置一个值，该值确定图表何时以及如何重新绘制。

应用于

MSChart 控件。

语法

*object*.DrawMode [=*mode*]

DrawMode 属性的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>mode</i>   | 整数。确定图表何时以及如何重新绘制，如“设置”中所示 |

设置

*mode* 值的设置如下：

| 常量               | 值 | 描述                 |
|------------------|---|--------------------|
| VtChDrawModeDraw | 0 | 直接在显示设备上绘制         |
| VtChDrawModeBlit | 1 | 在内存中绘制完毕后再绘制到显示设备上 |

请参阅

DrawMode 常量。

### EdgeIntensity 属性

返回或设置在三维图表中绘制对象边框所使用光线的密度。

应用于

Light 对象。

语法

*object*.EdgeIntensity [=edgeint]

EdgeIntensity 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

续表

*edgeint*

单精度数据。边缘光线密度。有效值是 0 到 1。0 值将关闭光线，将边框绘制为黑线；1 值将使用元件的画笔颜色全部照亮边框

## 说明

如果设置了该属性，则 **Light** 对象的 **EdgeVisible** 属性将自动设置为 **True**。

## EdgePen 属性

返回用于绘制图表数据点边框的 **Pen** 对象。

应用于

**DataPoint** 对象。

## 语法

*object*.EdgePen

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

# EdgeVisible 属性

返回或设置一个值，该值确定在三维图表中是否显示元素的边框。

应用于

Light 对象。

语法

*object*.EdgeVisible [=*boolean*]

EdgeVisible 属性的语法有如下几个部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，指定是否绘制边框，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述            |
|-------|---------------|
| True  | 边框可见          |
| False | 在三维图表中不显示元素边框 |

## EditCopy 方法

以 Windows 元文件格式将图表当前的图片拷贝到剪贴板上。也拷贝用于创建图表的数据到剪贴板上。

应用于

MSChart 控件。

语法

*object*.EditCopy

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

说明

该方法允许你将图表的数据或图表本身的图片拷贝到另一个应用程序。因为数据和图片都存储在剪贴板上，应用程序所获得的内容由图片应用程序的类型决定。例如，如果你在代码中执行了 EditCopy 方法，然后在 Excel 中选择 Edit Paste 命令，图表数据集合就被放置在 Excel 数据报表中了。要在报表中插入图表的图片，选择 Edit Paste Special，然而选择 Picture 类型。

请参阅

EditPaste 方法。

## EditPaste 方法

将剪贴板上的 Windows 元文件图片或一段文本拷贝到图表的当前位置。

应用于

MSChart 控件。

语法

*object*.EditPaste

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

说明

图表可以从剪贴板获取多种类型信息，这需要视调用 **EditPaste** 方法时所选择的图表元件。如果选择了整个图表，图表就查找剪贴板上的数据，试图使用新数据重新绘制。如果选择的标记可以接受图片，如棒图或图表背景，图表就在剪贴板上查找元文件。如果找到了元文件，就使用该元文件填充所选择的对象。

请参阅

**EditCopy** 方法。

## Effect 属性

返回或设置图表中字体的效果

应用于

VtFont 对象。

语法

*object*.Effect [=*effects*]

Effect 属性的语法有如下几个部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>effects</i> | 整数。一个 VtFontEffect 常量，描述字体效果 |

请参阅

FontEffect 常量

## Elevation 属性

返回或设置一个值，该值描述三维图表正视图的程度。

应用于  
View3D 对象。

语法

*object.Elevation [=degree]*

Elevation 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>degree</i> | 单精度数据。正视图的程度<br><br>正视图的角度可以从 0 到 90 度。如果是 90 度，则从图表的正上方看；如果是 0 度，则从视图的正边看。缺省值是 30 度。缺省的是使用度来度量正视图的角度。然而，你也可以使用弧度或梯度 |

## Excluded 属性

返回或设置一个值，该值确定图表中是否包含一个系列。

应用于

SeriesPosition 对象。



语法

*object.Excluded* [=*boolean*]

Excluded 属性的语法有如下几个部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>boolean</i> | 一个布尔表达式，指明图表中是否包含系列，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                                       |
|-------|--|
| True  | 图表不包含系列                                  |
| False | （缺省）绘制图表时包含系列。一个系列可以包含在图表中，但并不显示，因为其是隐藏的 |

请参阅

Hidden 属性（MSChart）。

## Fill 对象

描述图表中对象的背景类型和外观。

语法

**Fill**

属性

**Brush** 属性， **Style** 属性（**MSChart**）。

请参阅

**Brush** 对象， **Fill** 属性。

示例

下面的例子使用 **Fill** 对象给图表设置了一个梯度背景。

```
Private Sub Command1_Click()  
    With MSChart1.backdrop.Fill  
        ' Set a brush pattern backdrop.  
        .Style = VtFillStyleBrush  
        .Brush.Style. = VtBrushPattern50Percent  
    End With  
End Sub
```

## Fill 属性

返回描述图表中对象的背景类型和外观的 Fill 对象的引用。

应用于

Backdrop 对象。

语法

*object.Fill*

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## FillColor 属性

返回一个描述填充图表元素颜色的 VtColor 对象的引用。

应用于

Brush 对象，Marker 对象。

语法

*object.FillColor*

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## 说明

如果设置了 **Marker** 对象的 **FillColor** 属性，则 **Marker** 对象的 **Visible** 属性自动设置为 **True**。

## 示例

下面的例子为图表的背景刷子设置了颜色。**FillColor** 属性返回一个 **VtColor** 对象的引用。

```
Private Sub Command1_Click()  
    ' Sets Backdrop to Fill - Brush Style.  
    MSChart1.Backdrop.Fill.Style = VtFillStyleBrush  
    ' Sets chart fill color to red.  
    With MSChart1.Backdrop.Fill.Brush.FillColor  
        .Red = 255      ' Use properties to set color.  
        .Green = 0  
        .Blue = 0  
    End With  
End Sub
```

# Flag 属性

返回或设置系列中显示哪条统计线。

应用于

StatLine 对象。

语法

*object*.Flag [=lines]

Flag 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                            |
| <i>lines</i>  | 整数。描述统计线的 VtChStats 常量。如果显示多条统计线，常量应使用 OR 操作符组合起来 |

请参阅

StatsType 常量。

## Font 属性 (MSChart)

返回描述图表中显示文本字体的标准 Font 对象的引用。

应用于

AxisTitle 对象, DataPointLabel 对象, Footnote 对象, Lable 对象, Legend 对象, Title 对象。

语法

*object*.Font

object 置换元表示一个对象表达式, 其值为“应用于”列表中的一个对象。

## Footnote 对象

语法

Footnote

属性

Text 属性(MSChart), TextLength 属性, Backdrop 属性, Font 属性(MSChart), Location 属性, TextLayout 属性, VtFont 属性。

## 方法

Select 方法。

请参阅

Backdrop 对象，Footnote 属性。

## 示例

下面的例子设置了图表的脚注位置、文本和颜色。

```
Private Sub Command1_Click()
```

```
    With MSChart1.Footnote
```

```
        ' Make Footnote Visible.
```

```
        .Location.Visible = True
```

```
        .Location.LocationType = _
```

```
        VtChLocationTypeBottomLeft
```

```
        ' Set Footnote properties.
```

```
        .text = "Chart Footnote"
```

```
        .VtFont.VtColor.Set 255, 0, 0
```

```
    End With
```

```
End Sub
```

## Footnote 属性

返回描述图表脚注文本的 **Footnote** 对象的引用。

应用于

**MSChart** 控件。

语法

*object*.Footnote

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

**Footnote** 对象。

## FootnoteActivated 事件

当用户双击图表脚注时产生该事件。

应用于

**MSChart** 控件。



# 语法

Private Sub *object\_* FootnoteActivated(*mouseFlags* As Integer, *cancel* As Integer)

FootnoteActivated 事件的语法有如下几个部分：

| 部分                | 描述                             |
|-------------------|--------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 该参数当前尚未使用                      |

# 设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 `mouseFlags` 设置为：

| 常量  | 描述                     |
|---|------------------------|
| <code>VtChMouseFlagsShiftKeyDown</code>   | 如果按下了 <b>SHIFT</b> 键   |
| <code>VtChMouseFlagsControlKeyDown</code> | 如果按下了 <b>CONTROL</b> 键 |

# 请参阅

FootnoteSelected 事件，FootnoteUpdated 事件。

# FootnoteSelected 事件

当用户双击图表脚注时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_FootnoteSelected(mouseFlags As Integer, cancel As Integer)
```

FootnoteSelected 事件的语法有如下几个部分：

| 部分                | 描述                               |
|-------------------|----------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象           |
| 部分                | 描述                               |
| <i>mouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”<br>中所示 |
| <i>cancel</i>     | 整数。现在尚未使用                        |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

请参阅

FootnoteActivated 事件，FootnoteUpdated 事件。

## FootnoteText 属性

返回或设置脚注文本。

应用于

MSChart 控件。

语法

*object*.FootnoteText [=*text*]

FootnoteText 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>text</i>   | 字符串。脚注文本               |

说明

使用 Footnote 对象的 Text 属性也可以得到同样的效果。

FootnoteUpdated 事件

当图表脚注改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_FootnoteUpdated(updateFlags As Integer)
```

FootnoteUpdated 事件的语法有以下几部分：

| 部分                 | 描述                     |
|--------------------|------------------------|
| <i>object</i>      | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>updateFlags</i> | 整数，提供脚注更新信息，如“设置”中所示   |

设置

下表列出了 *updateFlags* 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

请参阅

FootnoteActivated 事件，FootnoteSelected 事件。

### Format 属性（Label 对象）

返回或设置显示坐标轴标签格式的字符。

应用于

Label 对象（Item）。

语法

*object*.Format [=format]

Format 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>format</i> | 字符串。定义显示坐标轴文本的格式       |

## FormatLength 属性

返回格式字符串的长度。

应用于

Label 对象。

语法

*object*.FormatLength

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

返回值

整数。

## Frame 对象

保存了图表周围窗体元素外观信息。

语法

Frame

属性

Style 属性 (MSChart), Width 属性 (MSChart), FrameColor 属性, SpaceColor 属性。

请参阅

Frame 属性。

示例

下面的例子在图表背景上设置了一个蓝色双线窗体。

```
Private Sub Command1_Click()
```

```
    With MSChart1.backdrop.Frame
```

```
        .Style = VtFrameStyleDoubleLine
```

```
        .Width = 2
```

```
        .FrameColor.Set 0, 0, 255    ' Blue frame.
```

```
        .SpaceColor.Set 255, 0, 0    ' Red spacing.
```

End With

End Sub

## Frame 属性

返回描述图表元素周围窗体外观的 **Frame** 对象的引用。

应用于

**Backdrop** 对象。

语法

*object*.Frame

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

**Frame** 对象。

## FrameColor 属性

返回描述图表元素周围窗体外观所使用颜色的 **VtColor** 对象的引用。



应用于

Frame 对象。

语法

*object.FrameColor*

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

SpaceColor 属性。

## GetData 方法（MSChart）

返回图表数据格子中指定数据点当前存储的值。

应用于

DataGrid 对象。

语法

*object.GetData(row, column, dataPoint, nullFlag)*

GetData 方法的语法有如下几个部分：

| 部分               | 描述                     |
|------------------|------------------------|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>row</i>       | 整数。标识包含了数据点值的行         |
| <i>column</i>    | 整数。标识包含了数据点值的列         |
| <i>dataPoint</i> | 双精度数据。数据点的值            |
| <i>nullFlag</i>  | 整数。指示数据点的值是否是 null     |

## GetSelectedPart 方法

标识当前选择的图表元素。

应用于

MSChart 控件。

语法

*object*.GetSelectedPart(*part*, *index1*, *index2*, *index3*, *index4*)

GetSelectedPart 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>part</i>   | 整数。指明图表元素。有效常量是 <b>VtChPartType</b>   |
| <i>index1</i> | 整数。如果元素指一个系列或一个数据点，则该参数指明是哪个系列。系列的计数与它们出现在数据格子中相应的列次序相同，从左到右，开始是 1。如果元素是一个坐标轴或坐标轴标签，该参数就使用 <b>VtChAxisId</b> 标识坐标轴的类型 |
| <i>index2</i> | 整数。如果元素指一个数据点，该参数指明在 <i>index1</i> 系列中的哪个数据点  |
| <i>index3</i> | 整数。如果元素是坐标轴标签，该参数指定标签的级数。坐标轴标签的级数从坐标轴外算起，开始是 1。如果元素非坐标轴标签，不使用该参数  |
| <i>index4</i> | 整数。当前该参数尚未使用  |

请参阅

**AxisId** 常量，**PartType** 常量。

## Green 属性

返回或设置图表中 **RGB** 值中的绿（green）分量。

应用于  
VtColor 对象。

语法

*object*.Green [=g]

Green 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>g</i>      | 整数。绿值                  |

说明

RGB 指定了要显示颜色中红绿蓝的相对密度。标准 RGB 值的有效范围是 0 到 16777215。RGB 中任何超过 255 的参数都认为是 255。

## GuidelinePen 属性

返回一个描述用于显示标线的线条和颜色图案的 Pen 对象的引用。

应用于  
Series 对象。

## 语法

*object*.GuidelinePen

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## 说明

设置该属性将自动将 **ShowGuideLines** 属性设置为 **True**。

## 示例

下面的例子为二维 **xy** 图表系列设置了画笔属性。**GuidelinePen** 属性返回一个 **Pen** 对象的引用。

```
Private Sub Command1_Click()  
    ' Set Guide Lines for 2D XY chart Series 1.  
    MSChart1.ChartType = VtChChartType2dXY  
    MSChart1.Plot.SeriesCollection.Item(1) _  
        .ShowGuideLine(VtChAxisIdX) = True  
    With _  
        MSChart1.Plot.SeriesCollection.Item(1).GuideLinePen  
            ' Set Pen attributes.  
            .VtColor.Set 255, 255, 0  
            .Width = 10  
            .Style = VtPenStyleDashDot
```

```
.Join = VtPenJoinRound
.Cap = VtPenCapRound
End With
End Sub
```

## Hidden 属性（MSChart）

返回或设置一个值，该值确定是否在图表上显示系列。

应用于

SeriesPosition 对象。

语法

*object*.Hidden [=*boolean*]

Hidden 属性的语法有如下几个部分：

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>boolean</i> | 一个布尔表达式，控制是否在图表中显示系列，如“设置”中所示 |

# 设置

boolean 值的设置如下：

| 设置    | 描述                     |
|-------|------------------------|
| True  | 图表不显示系列。然而，为系列分配的空间仍存在 |
| False | (缺省)显示系列               |

## Hide 属性

返回或设置一个值，该值确定图表上的坐标轴是否隐藏。

应用于

AxisScale 对象。

### 语法

*object*.Hide [=*boolean*]

Hide 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定是否隐藏坐标轴，如“设置”中所示 |

## 设置

*boolean* 值的设置如下：

| 设置    | 描述               |
|-------|------------------|
| True  | 隐藏坐标轴、刻度、线、线宽、标题 |
| False | （缺省）不隐藏图表坐标轴     |

## HorzAlignment 属性

返回或设置文本水平对齐的方法。

应用于

TextLayout 对象。

语法

*object*.HorzAlignment [=*type*]

HorzAlignment 属性的语法有如下几个部分：



| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                                |
| <i>type</i>   | 整数。一个 <code>HorizontalAlignment</code> 常量，描述文本水平对齐的方法 |

请参阅

`HorizontalAlignment` 常量。

## Index 属性（Brush）

如果 `Style` 属性设置为 `VtBrushStylePattern` 或 `VtBrushStyleHatch`，则返回或设置刷子中使用的图案。

应用于

`Brush` 对象。

语法

*object*.`Index` [=num]

`Index` 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                     |
| <i>num</i>    | 一个 VtBrushPattern 或 VtBrushHatch 常量，描述刷子图案 |

请参阅

BrushPattern 常量，BrushHatch 常量，Index 属性（Intersection）。

## Index 属性（Intersection）

当有多个坐标轴的索引值相同时，返回与另一个坐标轴相交的坐标轴。

应用于

Intersection 对象。

语法

*object*.Index

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

返回值：

返回值是一个整数，该整数指明相交坐标轴的索引。目前，1 是该参量唯一有效的值。

请参阅

Index 属性（Brush）。

## InitializeLabels 方法

为数据格子的第一级标签中每一个标签分配一个唯一标识符。

应用于

DataGrid 对象。

语法

*object*.InitializeLabels

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## InsertColumnLabels 方法

在图表的一个数据格子中插入数据列的标签级。

应用于

DataGrid 对象。

# 语法

*object.InsertColumnLabels( labelIndex, count)*

InsertColumnLabels 方法的语法有如下几个部分：

| 部分                | 描述   |
|-------------------|--|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                           |
| <i>labelIndex</i> | 整数。标识你要插入的第一级标签数。列标签从下到上计算，开始是 1                 |
| <i>count</i>      | 整数。指定要插入的标签级数。要插入的列数从 <i>labelIndex</i> 指定的列向上算起 |

请参阅

InsertColumns 方法，InsertRows 方法，InsertRowLabels 方法。

## InsertColumns 方法

在图表的数据格子中插入一个或多个数据列。

应用于

DataGrid 对象。

语法

*object.InsertColumns( column, count)*

InsertColumns 方法的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>column</i> | 整数。标识一个指定数据列。列从左到右计数，开始是 1 |
| <i>count</i>  | 整数。指定要插入的列数                |

请参阅

InsertColumnLabels 方法，InsertRowLabels 方法，InsertRows 方法。

InsertRowLabels 方法

在图表的数据格子中插入数据行的标签级。

应用于

DataGrid 对象。

语法

*object.InsertRowLabels( labelIndex, count)*

InsertRowLabels 方法的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                        |
| <i>labelIndex</i> | 整数。标识你要输入的第一级标签数。行标签从右到左计数，开始是 1              |
| <i>count</i>      | 整数。指定要插入的标签级数。行标签从 <i>labelIndex</i> 指定的行向左插入 |

请参阅

InsertColumns 方法， InsertColumnLabels 方法， InsertRows 方法。

## InsertRows 方法

在图表的数据格子中插入一个或多个数据行。

应用于

DataGrid 对象。

语法

*object.InsertRows( row, count)*

InsertRows 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                    |
| <i>row</i>    | 整数。标识一个特定的数据行。行从上到下计数，开始是 1               |
| <i>count</i>  | 整数。指定要插入的行数。行包含 <b>null</b> 数据，除非你对其填充了数据 |

请参阅

InsertColumns 方法，InsertColumnLabels 方法，InsertRowLabels 方法。

## Intensity 属性

返回或设置来自光源的光线强度。

应用于

LightSource 对象。

语法

*object*.Intensity [=strength]

Intensity 属性的语法有如下几个部分：

| 部分              | 描述  |
|-----------------|---|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>strength</i> | 单精度数据。光线强度。如果强度设置为百分之百（1），则面对光源的图表表面将全部被照亮。如果设置为 50%，则表面从光源接受 50%的光线。有效值从 0 到 1 |

说明

Intensity 属性是 LightSource 对象的缺省属性。

Intersection 对象

坐标轴图表中一条相交的坐标轴与相交的交点。

语法

Intersection

属性：

Auto 属性（Intersection），AxisId 属性，Index 属性（Intersection），LabelsInsidePlot 属性，Point 属性。



请参阅

Intersection 属性。

示例：

下面的例子手工设置坐标轴交点的位置，并显示坐标轴标签。

```
Private Sub Command1_Click()  
    With MSChart1.Plot.Axis(VtChAxisIdX).Intersection  
        ' Set Intersection Properties.  
        .Auto = False ' Set positioning to manual.  
        .Point = 20    ' Set intersection with the Y  
                        ' Axis to 20.  
        .LabelsInsidePlot = True ' Display Labels  
                                ' with Axis not at  
                                ' the base.  
    End With  
End Sub
```

## Intersection 属性

返回描述坐标轴与图表中另外一坐标轴相交交点的 **Intersection** 对象的引

用。

应用于

Axis 对象。

语法

*object*.Intersection

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

Intersection 对象。

## Item 属性（MSChart）

返回描述图表元素集合中一个对象的引用。

应用于

DataPoints 集合，Labels 集合，LightSources 集合，SeriesCollection 集合。

语法

*object*.Item(*index*)

Item 属性的语法有以下几部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 指定集合数的唯一整数             |

## Join 属性

返回或设置一个值，该值确定如何组织线段。

应用于

Pen 对象。

语法

*object*.Join [=*type*]

Join 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>type</i>   | 整数。一个 VtPenJoin 常量，描述汇笔风格 |

请参阅

PenJoin 常量。

## Label 对象

在 Labels 集合中描述指定的图表坐标轴标签的部件。

语法

*axis*.Label

属性

Backdrop 属性, Font 属性 (MSChart), Auto 属性 (Label), Format 属性 (MSChart),

FormatLength 属性, Standing 属性, TextLayout 属性, VtFont 属性。

请参阅

Labels 属性, Labels 集合。

# LabelLevelCount 属性

返回给定坐标轴上标签级的数量。设置时间时无效。

应用于

Axis 对象。

语法

*object*.LabelLevelCount [=*count*]

LabelLevelCount 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>count</i>  | 整数。描述标签的数的整数           |

# Labels 集合

一组图表坐标轴标签。

语法

*axis*.Labels(*index*)

Label 集合的语法有如下几个部分：

| 部分           | 描述                       |
|--------------|--------------------------|
| <i>axis</i>  | 一个 Axis 对象               |
| <i>index</i> | 标识当前集合中某个坐标轴标签（Label 对象） |

属性：

Item 属性（MSChart），Count 属性(ActiveX 控件)。

请参阅

Labels 属性，Label 对象。

## Labels 属性

返回描述图表坐标轴标签的 Labels 集合的引用。

应用于

Axis 对象。

语法

*object*.Labels

`object` 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

Labels 集合，Label 对象。

## LabelsInsidePlot 属性

返回或设置一个值，该值确定是否将坐标轴标签留在标准的位置还是移动到新的交点位置。

应用于

Interseciton 对象。

语法

*object*.LabelsInsidePlot [=*boolean*]

LabelsInsidePlot 属性的语法有如下几个部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>boolean</i> | 一个布尔表达式，指定何处显示坐标轴标签，如“设置”中所示 |

# 设置

*boolean* 值的设置如下:

| 设置    | 描述               |
|-------|------------------|
| True  | (缺省) 坐标轴标签在标准的位置 |
| False | 坐标轴标签在新的交点位置     |

# 说明

如果设置了该属性, 则 **Intersection** 对象的 **Auto** 属性将自动设置为 **False**。

## LabelTick 属性

返回或设置一个值, 该值指明栏的坐标轴标签是否显示在坐标轴刻度标记的中间。

# 应用于

**CategoryScale** 对象。

# 语法

*object.LabelTicks* [=*boolean*]

**LabelTick** 属性的语法有如下几个部分:



| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，指定是否显示部件，如“设置”中所示 |

## 设置

*boolean* 值的设置如下：

| 设置    | 描述              |
|-------|-----------------|
| True  | 标签在刻度标记的中间      |
| False | (缺省)标签在两个刻度标记之间 |

## 说明

如果设置了该属性，则对象的 **Auto** 属性自动设置为 **False**。

请参阅

**Auto** 属性（**CategoryScale**）。

## Layout 方法

对图表进行布局，强制所有自动设置的值重新计算。

应用于

MSChart 控件。

语法

*object*.Layout

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

说明

图表在第一次绘制时进行布局安排。当改变了图表设置时，图表在下一次绘制时重新布局。有些值图表是自动计算的，如根据图表类型或其他一些设置计算坐标轴的最大和最小值等。这些值只有在图表布局时才能确定。如果你在图表布局之前获取这些自动设置的值将不能获得新值。

## LCoor 对象

描述长整数坐标点 x 和 y。

语法

**Lcoor**

属性

X 属性，Y 属性。

方法

Set 方法（Coor,LCoor）。

请参阅

Coor 对象。

## Legend 对象

表示描述图表系列的图形说明和相关的文本。

语法

Legend

属性

Backdrop 属性，Font 属性（MSChart），Location 属性，TextLayout 属性，VtFont 属性。

## 方法

Select 方法。

请参阅

Backdrop 对象， Legend 属性。

## 示例

下面的例子设置图表说明的文本和背景参数。

```
Private Sub Command1_Click()
```

```
    With MSChart1.Legend
```

```
        ' Make Legend Visible.
```

```
        .Location.Visible = True
```

```
        .Location.LocationType = VtChLocationTypeRight
```

```
        ' Set Legend properties.
```

```
        .TextLayout.HorzAlignment = _
```

```
        VtHorizontalAlignmentRight ' Right justify.
```

```
        ' Use Yellow text.
```

```
        .VtFont.VtColor.Set 255, 255, 0
```

```
        .Backdrop.Fill.Style = VtFillStyleBrush
```

```
        .Backdrop.Fill.Brush.Style = VtBrushStyleSolid
```

```
        .Backdrop.Fill.Brush.FillColor.Set 255, 0, 255
```

End With

End Sub

## Legend 属性

返回一个 **Legend** 对象引用，其包含了有关图表系列的说明和文本的外观和行为信息。

应用于

MSChart 控件。

语法

*object*.Legend

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## LegendActivated 事件

当用户双击图表说明时产生该事件。

应用于  
MSChart 控件。

语法

Private Sub *object*\_LegendActivated( *mouseFlags* As Integer, *cancel* As Integer)  
LegendActivated 事件的语法有如下几个部分：

| 部分                | 描述                             |
|-------------------|--------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 整数该参数当前尚未使用                    |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

# LegendSelected 事件

当用户单击图表说明时产生该事件。

应用于

MSChart 控件。

语法

Private Sub *object*\_LegendSelected(*mouseFlags* As Integer, *cancel* As Integer )

LegendSelected 事件的语法有如下几个部分：

| 部分                | 描述                           |
|-------------------|------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>mouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 整数。现在尚未使用                    |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

## LegendText 属性

返回或设置标识图表说明的文本。

应用于

Series 对象。

语法

*object*.LegendText [=*text*]

LegendText 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>text</i>   | 字符串。标识说明中当前系列的文本       |

说明

缺省的，该文本与 ColumnLabel 对象的 Text 属性相同。



# LegendUpdated 事件

当图表说明改变时产生该事件。

应用于

MSChart 控件。

语法

Private Sub *object*\_LegendUpdated(*updateFlags* As Integer)

LegendUpdated 事件的语法有以下几部分：

| 部分                | 描述                     |
|-------------------|------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>updateFlag</i> | 整数，提供说明更新信息，如“设置”中所示   |

设置

下表列出了 *updateFlags* 常量：

| 常量              | 描述                     |
|-----------------|------------------------|
| VtChNoDisplay   | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot | 更新将导致图表重新绘制            |
| VtChLayoutPlot  | 更新将导致图表重新布局            |

续表

|                    |                  |
|--------------------|------------------|
| VtChDisplayLegend  | 更新将导致说明部分重新绘制    |
| VtChLayoutLegend   | 更新将导致说明部分重新布局    |
| VtChLayoutSeries   | 更新将导致系列重新布局      |
| VtChPostionSection | 图表的一部分已被移动或改变了大小 |

Length 属性（MSChart）

返回或设置坐标轴刻度标记的长度，以点来计算。

应用于

Tick 对象。

语法

*object.Length* [=length]

Length 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>length</i> | 整数。坐标轴刻度标记的长度          |

## Light 对象

标识照亮一个三维图表的光源。

语法

Light

属性

AmbientIntensity 属性, EdgeIntensity 属性, EdgeVisible 属性, LightSources 属性。

方法

Set 方法 (Coor, LCoor)。

请参阅

LightSources 集合, LightSource 对象, Z 属性。

示例

下面的例子设置图表的周围光线密度和边框的光线密度。要使用该例子，需要在一个窗体上绘制一个 **MSChart** 控件，两个 **ComboBox** 控件。将下面的代码拷贝到 **Form** 对象的代码模块中，执行该程序。单击组合框控件来观看光线密度的改变。

```

Private Sub Command1_Click()
    ' Configure the chart.
    With MSChart1
        .Title = "Hold down Control and mousedown on chart"
        .chartType = VtChChartType3dBar
        .plot.Light.AmbientIntensity = 1 ' 100% Intensity.
        .Plot.Light.EdgeIntensity = 0.5 ' 50% Intensity.
        .Plot.Light.EdgeVisible = True
    End With
' Configure ComboBoxe Controls.
    ConfigCombo Combo1
    ConfigCombo Combo2
End Sub

Private Sub ConfigCombo(cmb As ComboBox)
    ' Populate a combobox with values.
    Dim i as single
    For i = 0 to 1 Setp 0.1
        cmb.AddItem i
    Next i
    cmb.ListIndex = 0
End Sub

```

```
Private Sub Combo1_Click()  
    MSChart1.Plot.Light.AmbientIntensity = Combo1.Text  
End Sub  
  
Private Sub Combo2_Click()  
    MSChart1.Plot.Light.EdgeIntensity = Combo2.Text  
End Sub
```

## Light 属性

返回一个 **Light** 对象的引用，该对象提供了三维图表光源的信息。

应用于

**Plot** 对象。

语法

*object*.Light

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## LightSource 对象

标识三维图表中照亮图表元素的光源。

语法

LightSource

属性

X 属性, Y 属性, Z 属性, Intensity 属性。

方法

Set 方法 (LightSource)。

## LightSources 集合

图表中一组 LightSource 对象。

语法

*object*.LightSources(*index*)

LightSources 集合的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 整数。标识集合中的一个成员的唯一数      |

属性:

Item 属性 (MSChart)，Count 属性 (VB 集合)，Count 属性 (ActiveX 控件)。

方法:

Add 方法 (LightSources 集合)，Remove 方法 (LightSources 集合)。

示例:

下面的例子设置了光源的坐标和密度，然后给图表增加了一个光源。要使用该例子，需要在一个窗体上绘制一个 MSChart 控件和两个 CommandButton 控件。将下面的代码拷贝到 Form 对象的代码模块中，并运行该程序。单击按钮，观看光源的改变。

Option Explicit

Private Sub Form\_Load()

    ' Configure the chart and its light source.

    With MSChart1

        .chartType = VtChChartType3dBar

        .Plot.Light.AmbientIntensity = 0.01   ' 1% Intensity.

```
.Plot.Light.EdgeIntensity = 0.5      ' 5% Intensity
```

```
.Plot.Light.EdgeVisible = True
```

```
End With
```

```
End Sub
```

```
Private Sub Command_Click()
```

```
    Dim LightSource As Object
```

```
    Set LightSource = _
```

```
MSChart1.Plot.Light.LightSources(1)
```

```
    ' Set coordinates for Light Source 1 as well as its
```

```
    ' intensity
```

```
    LightSource.X = 1
```

```
    LightSource.Y = 0.5
```

```
    LightSource.Z = 1
```

```
    Lightsource.Intensity = 1
```

```
End Sub
```

```
Private Sub Command2_Click ()
```

```
    ' Add a new light source.
```

```
    MSChart1.Plot.Light.LightSources.Add 0.5, 0.1, 1, 1
```

```
End Sub
```



## LightSources 属性

返回一个 **LightSources** 集合的引用，该集合描述了三维图表中所有的光源。

应用于

**Light** 对象。

语法

*object*.**LightSources**

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

**LightSources** 集合。

## Limit 属性

返回或设置线段汇点的限制。

应用于

**Pen** 对象。

语法

*object.Limit [=joint]*

Limit 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>joint</i>  | 单精度数据。汇点对多条线段的宽度有限制。如果两条线段相交成锐角，则斜接缝使得拐角在实际的交点之上。如果外接点和内接点的距离超过了该属性的值，汇点将自动改变为斜角 |

LineStyle 属性（DataPointLabel 对象）

返回或设置连接图表上数据点和标签的线条类型。

应用于

DataPointLabel 对象。

语法

*object.LineStyle [=type]*

LineStyle 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                        |
| <i>type</i>   | 整数。一个标识连接线 <code>VtChLabelLineStyle</code> 常量 |

请参阅

`LabelLineStyle` 常量。

## Location 对象

代表文本图表元素的当前位置，如标题、说明或脚注。

语法

`Location`

`object` 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

属性

`Visible` 属性（`MSChart`），`LocationType` 属性，`Rect` 属性。

请参阅

`Footnote` 对象，`Legend` 对象，`Title` 对象。

## 示例

下面的例子使用了 **Location** 对象设置了图表标题的位置。要使用该例子，需要在一个窗体上绘制一个 **MSChart** 控件和一个 **ComboBox** 控件。将下面的代码拷贝到 **Form** 对象的代码模块中，执行该程序。单击 **ComboBox** 控件观看标题位置的改变。

Option Explicit

Private Sub Combo1\_Click()

    MSChart1.Title.Location.LocationType = Combo1.ListIndex

End Sub

Private Sub Form\_Load()

    ' Set Title Text

    With MSChart1

        .TitleText = " Test Title Location"

        .Title.Location.Visible = True

    End With

    ' Add LocationType constants to ComboBox.

    With Combo1

        .AddItem "VtChLocationTypeTopLeft"       ' 0

        .AddItem "VtChLocationTypeTop"           ' 1

        .AddItem "VtChLocationTypeTopRight"       ' 2

        .AddItem "VtChLocationTypeLeft"           ' 3

```
.AddItem "VtChLocationTypeRight"      ' 4
.AddItem "VtChLocationTypeBottomLeft" ' 5
.AddItem "VtChLocationTypeBottom"     ' 6
.AddItem "VtChLocationTypeBottomRight" ' 7
.AddItem "VtChLocationTypeCustom"     ' 8
.ListIndex = 0
End With
End Sub
```

## Location 属性

返回一个 **Location** 对象的引用，该对象描述了文本图表元素的位置。

应用于

Footnote 对象，Legend 对象，Title 对象。

语法

*object*.Location

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

Location 对象。

## LocationRect 属性

返回一个 **Rect** 对象的引用，该对象使用  $x$  和  $y$  坐标描述了图表绘制的位置。

应用于

Plot 对象。

语法

*object*.LocationRect

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

说明

如果 **AutoLayout** 属性设置为 **False**，则使用该属性的值定位绘制位置。

如果设置了该属性，则 **AutoLayout** 属性自动设置为 **False**。

示例：

下面的例子使用了 **LocationRect** 属性和 **Rect** 对象的  $x$  和  $y$  属性来增加图表绘制区域的大小。

```
Private Sub Command1_Click()  
    ' Increase the size of the chart plot.  
    MSChart1.Plot.AutoLayout = False  
    With MSChart1.Plot.LocationRect  
        .Min.x = .Min.x * 1.2  
        .Min.y = .Min.y * 1.2  
        .Max.x = .Max.x * 1.2  
        .Max.y = .Max.y * 1.2  
    End With  
End Sub
```

## LocationType 属性

返回或设置显示图表元素的标准位置。

应用于

DataPointLabel 对象， Location 对象。

语法

*object*.LocationType [=type]

LocationType 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>type</i>   | 整数。对于 <code>DataPointLabel</code> 对象，一个标识标签位置的 <code>VtChLabelLocationType</code> 常量。对于 <code>Location</code> 对象，一个描述文本位置的 <code>VtChLocationType</code> 常量 |

请参阅

`LocationType` 常量。

## LogBase 属性

返回或设置在对数坐标轴上绘制图表数据所用对数的底数。

应用于

`AxisScale` 对象。

语法

*object*.LogBase [=base]

LogBase 属性的语法有如下几个部分：



| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象          |
| <i>base</i>   | 整数。描述对数底数的整数。缺省是 10，可以从 2 到 100 |

### 说明

仅当 `Type` 属性被设置为 `UtChScaleTypeLog.arithmic` 时本属性有效。

## MajorDivision 属性

返回或设置在坐标轴上显示的主要分度单位的数量。

应用于

`ValueScale` 对象。

### 语法

*object*.MajorDivision [=num]

MajorDivision 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>uum</i>    | 整数。分度单位的数量             |

说明

如果设置了该属性，则 **ValueScale** 对象的 **Auto** 属性将自动设置为 **False**。

请参阅

**MinorDivision** 属性。

## MajorPen 属性

返回一个 **Pen** 对象的引用，该对象描述了主要坐标轴格子线的外观。

应用于

**AxisGrid** 对象。

语法

*object*.MajorPen

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

**MinorPen** 属性。

## Marker 对象

标识图表数据点的标记。

语法

Marker

属性

Visible 属性（MSChart），FillColor 属性，Style 属性（MSChart），Size 属性（MSChart），Pen 属性。

请参阅

DataPoint 对象。

示例：

下面的例子为图表系列设置了蓝色的 X 标记风格。

Option Explicit

Option Basel

Private Sub Command1\_Click()

    ' Display Markers for Series 1.

```
    Dim i As Integer
For i=1 To MSChart1.Plot.SeriesCollection.Count
    With MSChart1.Plot.SeriesCollection _
        .Item(i).DataPoints.Item(-1).Marker
        .Visible = True
        .Size = 20
        .Style = VtMarkerStyleX
        .FillColor.Automatic = False
        .FillColor.Set 0, 0, 255
    End With
Next i
End Sub
```

## Marker 属性

返回一个 **Marker** 对象的引用，该对象描述了图表数据点上的一个图标。

应用于

**DataPoint** 对象。

语法

*object*.Marker

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Max 属性（MSChart）

返回一个 **Coor** 对象的引用，指明一个矩形结束的拐角。

应用于

**Rect** 对象。

语法

*object*.Max

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

**Min** 属性（MSChart）。

## Maximum 属性

返回或设置图表坐标轴上的最高值或结束值。

应用于

ValueScale 对象。

语法

*object*.Maximum [=value]

Maximum 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 双精度数据。坐标轴的最高值          |

说明

如果设置了该属性，则 ValueScale 对象的 Auto 属性自动设置为 False。

应在 Minimum 属性之前设置 Maximum 属性，以免产生图表显示错误。

请参阅

Minimum 属性。

## Min 属性（MSChart）

返回一个 **Coord** 对象的引用，该对象指定矩形开始的拐角。

应用于

**Rect** 对象。

语法

*object*.Min

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

**Max** 属性（MSChart）。

## Minumum 属性

返回或设置图表坐标轴最低或开始的值。

应用于

**ValueScale** 对象。

语法

```
object.Minumum[=value]
```

Maximum 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 双精度数据。坐标轴的最低值          |

说明

如果设置了该属性，则 ValueScale 对象的 Auto 属性自动设置为 False。  
应在 Minimum 属性之前设置 Maximum 属性，以免产生图表显示错误。

请参阅

MajorDivision 属性。

MinorDivision 属性

返回或设置在坐标轴上显示的次要分度单位的数量。



应用于

ValueScale 对象。

语法

*object.MinorDivision [=num]*

MinorDivision 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>num</i>    | 整数。分度单位的数量             |

说明

如果设置了该属性，则 ValueScale 对象的 Auto 属性将自动设置为 False。

## MinorPen 属性

返回一个 Pen 对象的引用，该对象描述了次要坐标轴格子线的外观。

应用于

AxisGrid 对象。

语法

*object.MinorPen*

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

*MajorPen* 属性。

## MousePointer 属性（MSChart）

返回或设置一个值，该值指定在运行时鼠标滑过图表的某个部分所显示的鼠标指针的类型。

应用于

*MSChart* 控件。

语法

*object.MousePointer* [=*value*]

*MousePointer* 属性的语法有如下几个部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                   |
| <i>value</i>  | 整数。VtMousePointerContants 类型常量，指定鼠标指针的类型 |

请参阅

MousePointer 常量。

## MoveData 方法

在图表的数据格子中移动数据的范围。

应用于

DataGrid 对象。

语法

*object.MoveData(top, left, bottom, right, overOffset, downOffset )*

MoveData 方法的属性有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>top</i>    | 整数。标识要移动的第一行           |

续表

|                   |                         |
|-------------------|-------------------------|
| <i>left</i>       | 整数。标识要移动的第一列            |
| <i>bottom</i>     | 整数。标识要移动的最后一行           |
| <i>right</i>      | 整数。标识要移动的最后一列           |
| <i>overOffset</i> | 整数。标识数据水平移动方向。正数右移，负数左移 |
| <i>downOffset</i> | 整数。标识数据垂直移动方向。正数下移，负数上移 |

## Name 属性（VtFont）

返回或设置字体的名字。该属性是 VtFont 对象的缺省属性。

应用于

MSChart 控件，VtFont 对象。

语法

*object*.Name [=text]

Name 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>text</i>   | 正数。包含字体名的文本            |

## Offset 属性

返回或设置图表元素从缺省位置偏移或移开的距离。

应用于

DataPoint 对象， DataPointLabel 对象， Shadow 对象。

语法

*object*.Offset [=*offset*]

Offset 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>offset</i> | 对于 DataPoint 对象，这是描述偏移距离的整数。偏移计量单位是英寸或厘米，视 Windows 的设置而定。对于 DataPointLable 和 Shadow 对象，这是一个 Coor 对象的引用，描述了偏移量的 <i>x</i> 和 <i>y</i> 值 |

说明

对于 DataPointLabel 对象，该属性描述了数据点标签从预定义（标准）标签位置偏移和移开的距离。该偏移量增加到根据 DataPointLabel 对象的 LocationType 设置而计算的点的位置上。

## Order 属性

返回或设置图表系列的位置。如果位置按次序与另一个系列匹配，则该系列被堆栈。

应用于

SeriesPosition 对象。

语法

*object*.Order [=*order*]

Order 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>order</i>  | 这是。位置的次序               |

## Orientation 属性（MSChart）

返回或设置文本定向的方法。

应用于  
TextLayout 对象。  
语法

*object.Orientation* [=*type*]  
Orientation 属性的语法有如下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象          |
| <i>type</i>   | 整数。一个用于描述定向方法的 VtOrientation 常量 |

请参阅  
Orientation 常量。

### PatternColor 属性

返回一个 VtColor 对象的引用，描述填充图表元素使用的图案颜色。  
应用于  
Brush 对象。

语法

*object*.PatternColor

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Pen 对象

描述图表线条或边的颜色和图案。

语法

Pen

属性

Style 属性（MSChart），Width 属性（MSChart），Cap 属性，Join 属性，Limit 属性，VtColor 属性。

请参阅

Axis 对象，Marker 对象，PlotBase 对象，Series 对象，Wall 对象。



## Pen 属性

返回或设置一个 Pen 对象的引用，该对象描述图表元素线条或边使用的颜色和图案。

应用于

Axis 对象， Marker 对象， PlotBase 对象， Series 对象， Wall 对象。

语法

*object*.Pen

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

Pen 对象。

## PercentBasis 属性

返回或设置图表中在百分比坐标轴上显示的百分比类型。

应用于

AxisScale 对象。

## 语法

*object*.PercentBasis [=*type*]

PercentBasis 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                       |
| <i>type</i>   | 一个用于描述百分比坐标轴值的百分比类型的<br>VtChPercentAxisBasis |

## 说明

仅当 Type 属性设置为 VtChScaleTypePercent 时，本属性有效。

## 请参阅

Type 属性（MSChart）， PercentAxisBasis 常量。

## PercentFormat 属性

返回或设置一个字符串，该字符串描述显示百分比标签的格式。

## 应用于

DataPointLabel 对象。

语法

`object.PercentFormat [=format]`

PercentFormat 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>format</i> | 字符串。描述显示百分比标签的格式       |

说明

使用 DataPointLabel 对象的 Component 属性改变标签类型。

下表给出了几个百分比格式字符串的例子。左边的值是有效的格式。

|       | 3       | -3       | .3     |
|-------|---------|----------|--------|
| 0%    | 300%    | -300%    | 30%    |
| 0.0%  | 300.0%  | -300.0%  | 30.0%  |
| 0.00% | 300.00% | -300.00% | 30.00% |

Plot 对象

显示图表的区域。

## 语法

### Plot

## 说明

Plot 对象允许你规划下列对象：

1. **Axis** 对象—代表图表的 x,y 和 z 轴。只有在 3D 图表中才能看见 z 轴。
2. **Backdrop** 对象—坐标轴后面的区域。
3. **Light** 对象—绘图区周围和边缘灯光。
4. **LocationRect** 对象—绘图区的位置。
5. **SeriesCollection** 对象—系列集的集合。
6. **View3D** 对象—3D 图像的正视图和旋转。
7. **Wall** 对象—绘图区后面的区域。

## 属性：

**SeriesCollection** 属性, **Axis** 属性, **Backdrop** 属性, **Light** 属性, **AngleUnit** 属性, **AutoLayout** 属性, **BarGap** 属性, **Clockwise** 属性, **DataSeriesInRow** 属性, **DefaultPercentBasis** 属性, **DepthToHeightRatio** 属性, **LocationRect** 属性, **PlotBase** 属性, **Projection** 属性, **Sort** 属性(MSChart), **StartingAngle** 属性, **SubPlotLabelPosition** 属性, **UniformAxis** 属性, **View3D** 属性, **Wall** 属性, **Weighting** 属性, **WidthToHeightRatio** 属性, **XGap** 属性, **ZGap** 属性。

请参阅

Light 对象。

示例

下面的例子设置了图表视图的间距和坐标轴分度间距。

```
Private Sub Command1_Click()  
    ' Change the chart type to 3D Bar.  
    MSChart1.ChartType = VtChChartType3dBar  
    With MSChart1.Plot  
        ' Changes 3d bar chart's viewing.  
        .DepthToHeightRatio = 2  
        .WidthToHeightRatio = 2  
        ' Changes the spacing between divisions on the  
        ' X-Axis.  
        .xGap = 0  
        ' Changes the spacing between divisions on the  
        ' Z-Axis.  
        .zGap = 0.8  
    End With  
End Sub
```

## Plot 属性

返回一个 **Plot** 对象的引用，该对象描述显示图表的区域。

应用于

**MSChart** 控件。

语法

*object*.Plot

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## PlotActivated 事件

当用户双击图表绘制区域时产生该事件。

应用于

**MSChart** 控件。

语法

Private Sub *object*\_PlotActivated(*mouseFlags* As Integer, *cancel* As Integer)

**PlotActivated** 事件的语法有如下几个部分：

| 部分                | 描述                             |
|-------------------|--------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 整数。该参数当前尚未使用                   |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

PlotBase 对象

图表下面的区域。

语法

PlotBase

## 属性

Brush 属性, Pen 属性, BaseHeight 属性。

## 请参阅

Backdrop 对象, Brush 对象, Plot 对象。

## 示例

下面的例子在一个三维棒图表中设置了图表基参数。

```
Private Sub Command1_Click()  
    ' Change the chart type to 3D.  
    MSChart1.ChartType = VtChChartType3dBar  
    With MSChart1.Plot.PlotBase  
        ' Change the base height.  
        .BaseHeight = 20  
        ' Use the pattern style for base.  
        .Brush.Style = VtBrushStylePattern  
        .Brush.Index = VtBrushPatternHorizontal  
        .Brush.FillColor.Set 255, 160, 160  
        .Brush.PatternColor.Set 180, 180, 255  
        .Pen.Style = VtPenStyleSolid  
        .Pen.VtColor.Set 72, 72, 255
```



```
End With  
End Sub
```

## PlotBase 属性

返回一个 **PlotBase** 对象的引用，该对象描述图表下面的区域外观。

应用于

**Plot** 对象。

语法

*object*.**PlotBase**

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## PlotSelected 事件

当用户单击图表绘制区域时产生该事件。

应用于

**MSChart** 控件。

语法

```
Private Sub object_PlotSelected(mouseFlags As Integer, cancel As Integer )
```

PlotSelected 事件的语法有如下几个部分：

| 部分                | 描述                           |
|-------------------|------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>mouseFlags</i> | 整数，表明在单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 整数。现在尚未使用                    |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

PlotUpdated 事件

当图表绘制区域改变时产生该事件。

应用于  
MSChart 控件。  
语法

Private Sub *object*\_PlotUpdated(*updateFlags* As Integer)  
PlotUpdated 事件的语法有以下几部分：

| 部分                | 描述                     |
|-------------------|------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>updateFlag</i> | 整数，提供说明更新信息，如“设置”中所示   |

设置

下表列出了 *updateFlags* 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

## Point 属性

返回或设置当前坐标轴与另一个坐标轴的交点。

应用于

Intersection 对象。

语法

*object*.Point [=*point*]

Point 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>point</i>  | 双精度数据。当前坐标轴交点          |

说明

如果设置了该属性，则 Intersection 对象的 Auto 属性就自动设置为 False。

## PointActivated 事件

当用户双击图表数据点时产生该事件。

应用于  
MSChart 控件。  
语法

```
Private Sub object_PointActivated(series As Integer, dataPoint As Integer ,  
    mouseFlags As Integer, cancel As Integer)
```

PointActivated 事件的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>series</i>     | 整数。标识包含数据点的系列。系列的计数根据在数据格子中所在列出现次序，开始是 1    |
| <i>dataPoint</i>  | 整数。标识数据点在系列中的位置。数据点的计数是根据在数据格子中所在行的次序，开始是 1 |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示              |
| <i>cancel</i>     | 整数。该参数当前尚未使用                                |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

## PointLabelActivated 事件

当用户双击图表数据点标签时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_PointLabelActivated(series As Integer, dataPoint As Integer ,  
    mouseFlags As Integer, cancel As Integer)
```

PointLabelActivated 事件的语法有如下几个部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                   |
| <i>series</i> | 整数。标识包含数据点的系列。系列的计数根据在数据格子中所在列出现次序，开始是 1 |

续表

|                   |   |
|-------------------|---|
| <i>dataPoint</i>  | 整数。标识数据点在系列中的位置。数据点的计数是根据在数据格子中所在行的次序，开始是 1 |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示              |
| <i>cancel</i>     | 整数。该参数当前尚未使用                                |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

PointLabelSelected 事件

当用户单击图表数据点标签时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_PointLabelSelected(series As Integer, dataPoint As Integer ,
```

*mouseFlags* As Integer, *cancel* As Integer)

PointLabelSelected 事件的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>series</i>     | 整数。标识包含数据点的系列。系列的计数根据在数据格子中所在列出现次序，开始是 1    |
| <i>dataPoint</i>  | 整数。标识数据点在系列中的位置。数据点的计数是根据在数据格子中所在行的次序，开始是 1 |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示              |
| <i>cancel</i>     | 整数。该参数当前尚未使用                                |

## 设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 *mouseFlags* 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |



# PointLabelUpdated 事件

当图表数据点标签改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_PointLabelUpdated(series As Integer, dataPoint As Integer ,  
    updateFlags As Integer)
```

PointLabelUpdated 事件的语法有以下几部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>series</i>     | 整数。标识包含数据点的系列。系列的计数根据在数据格子中所在列出现次序，开始是 1    |
| <i>dataPoint</i>  | 整数。标识数据点在系列中的位置。数据点的计数是根据在数据格子中所在行的次序，开始是 1 |
| <i>updateFlag</i> | 整数。提供说明更新信息，如“设置”中所示                        |

设置

下表列出了 *updateFlags* 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明部分重新绘制          |
| VtChLayoutLegend   | 更新将导致说明部分重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

## PointSelected 事件

当用户单击图表数据点时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_PointSelected(series As Integer, dataPoint As Integer ,  
    mouseFlags As Integer, cancel As Integer)
```

PointSelected 事件的语法有如下几个部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>series</i>     | 整数。标识包含数据点的系列。系列的计数根据在数据格子中所在列出现次序，开始是 1    |
| <i>dataPoint</i>  | 整数。标识数据点在系列中的位置。数据点的计数是根据在数据格子中所在行的次序，开始是 1 |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示              |
| <i>cancel</i>     | 整数。该参数当前尚未使用                                |

## 设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 `mouseFlags` 设置为：

| 常量  | 描述                     |
|---|------------------------|
| <code>VtChMouseFlagsShiftKeyDown</code>   | 如果按下了 <b>SHIFT</b> 键   |
| <code>VtChMouseFlagsControlKeyDown</code> | 如果按下了 <b>CONTROL</b> 键 |

## PointUpdated 事件

当图表数据点改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_PointUpdated(series As Integer, dataPoint As Integer ,  
updateFlags As Integer)
```

PointUpdated 事件的语法有下列各部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>series</i>     | 整数。标识包含数据点的系列。系列的计数根据其在数据格子中的出现次序,开始是 1     |
| <i>dataPoint</i>  | 整数。标识数据点在系列中的位置。数据点的计数是根据在数据格子中所在行的次序，开始是 1 |
| 部分                | 描述  |
| <i>updateFlag</i> | 整数。提供说明更新信息，如“设置”中所示                        |

设置

下表列出了 updateFlags 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明图注重新绘制          |
| VtChLayoutLegend   | 更新将导致说明图注重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

## Position 属性（MSChart）

返回一个 `SeriesPosition` 对象的引用，描述一个系列与另一个系列的相对位置。

应用于

`Series` 对象。

语法

*object*.Position

`object` 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

SeriesPosition 对象。

## Projection 属性

返回或设置显示图表的投影类型。

应用于

Plot 对象。

语法

*object*.Projection [=*type*]

Projection 属性的语法有如下几个部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象             |
| <i>type</i>   | 整数。一个 VtProjectionType 常量，描述图表投影类型 |

说明

Projection 属性确定 3 维图表的出现。

请参阅

**ProjectionType** 常量。

示例

下面的例子将使得你可以看到设置 **Projection** 属性后不同的效果。要使用该例子，需要在一个窗体上绘制一个 **MSChart** 控件和一个 **ComboBox** 控件。将下面的代码拷贝到 **Declarations** 一节并按 **F5** 键。单击 **ComboBox** 查看不同的设置。

Option Explicit

Private Sub Combo1\_Click()

    ' Change the projection when clicked.

    MSChart1.Plot.Projection = Combo1.ListIndex

End Sub

Private Sub Form\_Load()

    ' Set the chart to a 3D type.

    MSChart1.chartType = VtChChartType3dBar

    ' Configure the ComboBox by adding the valid settings.

    With Combo1

        .AddItem "VtProjectionTypePerspective" ' 0

        .AddItem "VtProjectionTypeOblique" ' 1

        .AddItem "VtProjectionTypeOrthogonal" ' 2

        .AddItem "VtProjectionTypeFrontal" ' 3

```
.AddItem "VtProjectionTypeOverhead" ' 4
.ListIndex = 0
End With
End Sub
```

## RandomDataFill 方法

使用随机生成的数据填充与指定图表关联的数据格子。

应用于

DataGridView 对象。

语法

*object*.RandomDataFill

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

MoveData 方法, RandomFillColumns 方法, RandomFillRows 方法, RandomFill 属性。



## RandomFill 属性

指定图表数据格子中的数据是否是随机产生的。

应用于

MSChart 控件。

语法

*object*.RandomFill [=boolean]

RandomFill 属性的语法有如下几个部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，控制数据如何产生，如“设置”中所示 |

设置

boolean 值的设置如下：

| 设置    | 描述                |
|-------|-------------------|
| True  | 使用随机数据绘制图表        |
| False | 不产生随机数据。用户为图表通过数据 |

请参阅

RandomDataFill 方法， RandomFillColumns 方法， RandomFillRows 方法。

## RandomFillColumns 方法

使用随机数据填充图表数据格子中的一些列。

应用于

DataGrid 控件。

语法

*object*.RandomFillColumns (*column*, *count*)

RandomFillColumns 方法的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>column</i> | 整数。代表要填充的第一列。列从左到右计数，开始是 1 |
| <i>count</i>  | 整数。指定你要用随机数据填充的列数          |

请参阅

RandomDataFill 方法， RandomFillRows 方法， RandomFill 属性。

# RandomFillRows 方法

使用随机数据填充图表数据格子中的一些行。

应用于

DataGrid 控件。

语法

*object*.RandomFillRows (*row*, *count*)

RandomFillRows 方法的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>row</i>    | 整数。代表要填充的第一行。行从上到下计数，开始是 1 |
| <i>count</i>  | 整数。指定你要用随机数据填充的行数          |

请参阅

MoveData 方法, RandomDataFill 方法, RandomFillColumns 方法, RandomFill 属性。

## Rect 对象

定义一个坐标点。

语法

**Rect**

属性

**Min** 属性（MSChart），**Max** 属性（MSChart）。

请参阅

**Coor** 对象。

## Rect 属性

返回或设置定义坐标位置的一个 **Rect** 对象。

应用于

**Location** 对象

语法

*object*.Rect

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

Rect 对象。

## Red 属性

返回或设置 RGB 值中的 R 分量。

应用于

VtColor 对象。

语法

*object*.Red [=*r*]

Red 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>r</i>      | 整数。R 分量                |

## 说明

RGB 指定了要显示颜色中红绿蓝的相对密度。标准 RGB 值的有效范围是 0 到 16777215。RGB 中任何超过 255 的参数都认为是 255。

## Remove 方法（LightSources 集合）

从 LightSources 集合中删除一个 LightSource。

应用于

LightSources 集合。

语法

*object.Remove(index)*

Remove 方法的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 整数。指定光源列表中的某个光源        |

# Repaint 属性

返回或设置一个值，该值确定图表修改时是否重新绘制 MSChart 控件。

应用于

MSChart 控件。

语法

*object.Repaint* [=*boolean*]

Repaint 属性的语法有如下几个部分：

| 部分             | 描述                          |
|----------------|-----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象      |
| <i>boolean</i> | 一个布尔表达式，控制图表是否重新绘制，如“设置”中所示 |

设置

boolean 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 刷新控件   |
| False | 对图表作修改时不允许重新绘制图表。当图表连续几次修改而你不想让图表连续刷新时该刷新很有用 |

## ResetCustom 方法

将数据点上自定义的属性值重新设置为系列的缺省值。

应用于

DataPoint 对象。

语法

*object*.ResetCustom

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## ResetCustomLabel 方法

将数据点标签上自定义的属性值重新设置为系列的缺省值。

应用于

DataPointLabel 对象。

语法

*object*.ResetCustomLabel

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。



## Rotation 属性

返回或设置一个值，该值确定三维图表对于视点的旋转角度。

应用于

View3D 对象。

语法

*object*.Rotation [=degree]

Rotation 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>degree</i> | 单精度数据。旋转的角度。有效范围是 0 到 360 度。缺省地，使用度来度量旋转的角度。然而，这些设置使用 <code>AngleUnits</code> 设置的当前设置值。可能的选项有弧度和梯度 |

## Row 属性（MSChart）

返回或设置数据格子中的当前行的数据。

应用于

MSChart 控件。

语法

*object*.Row [=num]

Row 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>num</i>    | 整数。当前列的行号。行从上到下计数，开始是 1 |

### RowCount 属性（MSChart）

返回或设置与图表相关的数据格子每列中有多少行。

应用于

MSChart 控件，DataGrid 对象。

语法

*object*.RowCount [=count]

RowCount 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>count</i>  | 整数，列中的行数               |

## RowLabel 属性（DataGrid）

返回或设置与图表相关的当前数据格子中指定行的标签。

应用于

DataGrid 对象。

语法

*object*.RowLabel( *row* , *labelIndex* ) [=*text*]

RowLabel 属性的语法有如下几个部分：

| 部分                | 描述                         |
|-------------------|----------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象     |
| <i>row</i>        | 整数。标识数据行。行的计算从上到下，从 1 开始计算 |
| <i>labelIndex</i> | 整数。指定标签的级数。行标签从左到右计数，开始是 1 |

续表

*text*

字符串。行的标签文本

请参阅

RowLabel 属性（MSChart 控件）。

## RowLabel 属性（MSChart 控件）

返回或设置标识图表数据格子中当前数据点的数据标签。

应用于

MSChart 控件。

语法

*object*.RowLabel [=*text*]

RowLabel 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>text</i>   | 字符串。行的标签文本。你指定的标签将设置由 <b>Row</b> 属性指定的数据点。对于大多数图表，该标签将与栏坐标一起显示，标识饼图中每个饼图块。如果标签文本太长，或许并不显示该标签 |

请参阅

RowLabel 属性（DataGrid）。

### RowLabelCount 属性

返回或设置图表数据格子中行标签的级数。

应用于

MSChart 控件，DataGrid 对象。

语法

*object*.RowLabelCount [=count]

RowLabelCount 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>count</i>  | 整数。行标签级数。设置该属性可以该指定的数据格子行增加或删除标签。列标签的级数从下向上算起，下面是 1。级从左边增加或删除 |

请参阅

SetSize 方法。

## RowLabelIndex 属性

返回或设置图表行标签的指定值。

应用于

MSChart 控件。

语法

*object*.RowLabelIndex [=*index*]

RowLabelIndex 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                            |
| <i>index</i>  | 整数。行标签级数。对于行的标签设置多级，你必须先指定要操作的标签级。行标签从右到左计数，开始是 1 |

## SecondaryAxis 属性

返回或设置一个值，该值确定系列是否在第二个坐标轴上绘制。

应用于

Series 对象。

语法

*object.SecondaryAxis* [=*boolean*]

SecondaryAxis 属性的语法有如下几个部分：

| 部分             | 描述                                |
|----------------|-----------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象            |
| <i>boolean</i> | 一个布尔表达式，控制系列是否绘制到第二个坐标轴上，如“设置”中所示 |

## 设置

boolean 值的设置如下：

| 设置    | 描述                |
|-------|-------------------|
| True  | 在第二个坐标轴上绘制系列      |
| False | （缺省）不在第二个坐标轴上绘制系列 |

## Select 方法

选择指定的图表元素。

### 应用于

DataPoint 对象，DataPointLabel 对象，Footnote 对象，Legend 对象，Series 对象，Title 对象。

### 语法

*object*.Select

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。



## SelectPart 方法

选择指定的图表部分。

应用于

MSChart 控件。

语法

*object*.SelectPart(*part*, *index1*, *index2*, *index3*, *index4*)

SelectPart 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>part</i>   | 整数。指明图表部分。有效常量是 <b>VtChPartType</b> 。   |
| <i>index1</i> | 整数。如果 <b>part</b> 指一个系列或一个数据点，则该参数指明是哪个系列。系列的计数与相应的列相同，从左到右，开始是 1。如果 <b>part</b> 是一个坐标轴或坐标轴标签，该参数就使用 <b>VtChAxisId</b> 标识坐标轴的类型 |

续表

|               |  |
|---------------|--|
| <i>index2</i> | 整数。如果 <b>part</b> 指一个数据点，该参数就指明 <b>indx1</b> 标明的是系列中的哪个点。数据点按数据栅格中相应行中出现的顺序计数，从上到下，开始是 1。如果 <b>part</b> 是指坐标轴、坐标轴标题或坐标轴标签，则该参数就指向当前尚未使用的坐标轴索引。在这种情况下，该参数的有效值就是 1 |
| <i>index3</i> | 整数。如果 <b>part</b> 是坐标轴标签，该参数指定标签的级数。坐标轴标签的级数从坐标轴外算起，开始是 1。如果 <b>part</b> 非坐标轴标签，不使用该参数   |
| <i>index4</i> | 整数。当前该参数尚未使用   |

请参阅

AxisId 常量，PartType 常量。

## Series 集合

一个图表系列集合。

语法

Series (*index*)

Series 集合的语法有如下几个部分：

| 部分           | 描述                                |
|--------------|-----------------------------------|
| <i>index</i> | 整数。标识图表的系列。系列以其在数据格子列中的次序标识，开始是 1 |

属性：

Item 属性（ActiveC 控件）。

方法：

Count 方法。

请参阅

Series 对象。

示例

下面的例子隐藏了图表中所有的系列。

```
Private Sub Command1_Click()
    Dim seriX As Series
    ' Hides All Series.
    For Each serX In MSChart1.Plot.SeriesCollection
        Series.Position.Hidden = True
    Next
```

End Sub

## Series 对象

**SeriesCollection** 集合中的一个成员，代表图表中的一组数据点。

语法

**Series**

属性

**DataPoints** 属性， **Pen** 属性， **GuidelinePen** 属性， **LegendText** 属性， **SecondaryAxis** 属性， **SeriesMarker** 属性， **SeriesType** 属性， **ShowGuideLine** 属性， **ShowLine** 属性， **StatLine** 属性， **TypeByChartType** 属性， **Position** 属性（MSChart）。

方法：

**Select** 方法。

请参阅

**SeriesCollection** 集合。

示例：

下面的例子设置了三维线图表中所有的系列平滑度。

```

Private Sub Command1_Click()
    Dim serX As Series
    'Change the chart type to 3D line and smoothing
    'each line
    MSChart1.chartType = VtChChartType3dLine
    MSChart1.columnCount = 4
    For Each serX In MSChart1.Plot.SeriesCollection
        serX.ShowGuideLine(VtChAxisIdY) = True
        serX.GuideLinePen.Style = VtPenStyleDitted
        serX.Pen.Style = 4
    Next
End Sub

```

## SeriesActivated 事件

当用户双击图表系列时产生该事件。你可以取消该事件，显示你自己的对话框，从而替换标准的用户界面。

应用于

**MSChart** 控件。

语法

```
Private Sub object_SeriesActivated(series As Integer , mouseFlags As Integer,  
cancel As  
Integer)
```

SeriesActivated 事件的语法有如下几个部分：

| 部分                | 描述                                    |
|-------------------|---------------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                |
| <i>series</i>     | 整数。标识包含了数据点的系列。系列以它们在数据格子中的列来标识，开始是 1 |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示。       |
| <i>cancel</i>     | 该参数当前尚未使用                             |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 mouseFlags 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

## SeriesCollection 集合

提供了组成图表系列的信息。

语法

SeriesCollection(*index*)

SeriesCollection 集合的语法有如下几个部分：

| 部分           | 描述           |
|--------------|--------------|
| <i>index</i> | 标识系列集合中指定的系列 |

属性

Item 属性（MSChart）。

方法

Count 属性（ActiveX 控件），Count 属性（VB 集合）。

## SeriesCollection 属性

返回 SeriesCollection 集合的引用，提供了组成图表的系列信息。

应用于

Plot 对象。

语法

*object*.SeriesCollection

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## SeriesColumn 属性

返回或设置当前系列数据所在列的位置。

应用于

MSChart 控件。

语法

*object*.SeriesColumn [=*pos*]

SeriesColumn 属性的语法有如下几个部分：



| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                                  |
| <i>pos</i>    | 整数。包含当前系列数据列的位置。你可以使用该属性对系列重新排序。如果两个系列分配了相同的位置，则它们就放入栈中 |

## SeriesMarker 对象

描述了标识图表一个系列中所有数据点的标记。

语法

SeriesMarker

属性

Auto 属性（SeriesMarkers）， Show 属性。

示例

下面的例子对图表中的所有系列设置了标记参数。

```
Private Sub Command1_Click()
```

```
    Dim serX As series
```

```
    ' Show markers and unshow the lines for all series.
```

```
MSChart1.ChartType = VtChChartType2dLine
For Each serX In MSChart1.Plot.SeriesCollection
    serX.SeriesMarker.Show = True
    serX.ShowLine = False
Next
End Sub
```

## SeriesMarker 属性

返回一个 **SeriesMarker** 对象的引用，描述标识图表一个系列中所有数据点的标记。

应用于

**Series** 对象。

语法

*object*.SeriesMarker

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## SeriesPosition 对象

一个系列与其他系列的相对绘制位置。如果所有的系列都有相同的次序，则它们都放入栈中。

语法

SeriesPosition

属性

Excluded 属性，Hidden 属性（MSChart），Order 属性，StackOrder 属性。

## SeriesSelected 事件

当用户单击图表系列时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_SeriesSelected(series As Integer , mouseFlags As Integer,  
cancel As
```

Integer)

SeriesSelected 事件的语法有如下几个部分：

| 部分                | 描述                                    |
|-------------------|---------------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                |
| <i>series</i>     | 整数。标识包含了数据点的系列。系列以它们在数据格子中的列来标识，开始是 1 |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示        |
| <i>cancel</i>     | 该参数当前尚未使用                             |

## 设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 `mouseFlags` 设置为：

| 常量  | 描述                           |
|---|------------------------------|
| <code>VtChMouseFlagsShiftKeyDown</code>   | 如果按下了 <code>SHIFT</code> 键   |
| <code>VtChMouseFlagsControlKeyDown</code> | 如果按下了 <code>CONTROL</code> 键 |

## SeriesType 属性

返回或设置显示当前系列的类型。

应用于

MSChart 控件，Series 对象。

语法

*object*.SeriesType [=type]

SeriesType 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>type</i>   | 整数。一个 VtChSeriesType 常量，描述显示系列的方法。<br>在使用 SeriesType 属性之前，你必须使用 Column 属性选择要修改的系列 |

请参阅

SeriesType 常量。

## SeriesUpdated 事件

当图表系列改变时产生该事件。

应用于

MSChart 控件。

语法

Private Sub *object\_SeriesUpdated*(*series* As Integer , *updateFlags* As Integer)

*SeriesUpdated* 事件的语法有如下几个部分：

| 部分                 | 描述                                    |
|--------------------|---------------------------------------|
| <i>object</i>      | 对象表达式，其值是“应用于”列表中的一个对象                |
| <i>series</i>      | 整数。标识包含了数据点的系列。系列以它们在数据格子中的列来标识，开始是 1 |
| <i>updateFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示        |

设置

下表列出了 *updateFlags* 常量：

| 常量              | 描述                     |
|-----------------|------------------------|
| VtChNoDisplay   | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot | 更新将导致图表重新绘制            |
| VtChLayoutPlot  | 更新将导致图表重新布局            |

续表

|                    |                  |
|--------------------|------------------|
| VtChDisplayLegend  | 更新将导致说明图例重新绘制    |
| VtChLayoutLegend   | 更新将导致说明图例重新布局    |
| VtChLayoutSeries   | 更新将导致系列重新布局      |
| VtChPostionSection | 图表的一部分已被移动或改变了大小 |

Set 方法（Coor, LCoor）

设置图表的 x,y 坐标位置。

应用于

Coor 对象，LCoor 对象。

语法

*object*.Set (x,y)

Set 方法的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象         |
| 部分            | 描述                             |
| <i>x</i>      | 单精度数据（对于 LCoor 是长整数）。指定 x 坐标位置 |
| <i>y</i>      | 单精度数据（对于 LCoor 是长整数）。指定 y 坐标位置 |

请参阅

X 属性，Y 属性。

## Set 方法 (LightSource)

设置 LightSource 对象的 x,y,z 坐标位置和光源密度。

应用于

LightSource 对象。

语法

*object.Set (x,y,z, intensity)*

Set 方法的语法有如下几个部分：

| 部分               | 描述                     |
|------------------|------------------------|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>x,y,z</i>     | 整数。指定光源的位置             |
| <i>intensity</i> | 单精度数据。指定光源密度           |



# Set 方法（View3D）

设置三维图表旋转和提升的角度。

应用于

View3D 对象。

语法

```
object.Set (rotation, elevation)
```

Set 方法的属性有如下几个部分：

| 部分               | 描述   |
|------------------|--|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>rotation</i>  | <p>单精度数据。旋转的度数</p> <p>有效范围是 0 到 360 度。缺省地，使用度来衡量旋转的角度。然而，这些设置使用 <b>AngleUnits</b> 属性的值。其他选项有弧度和梯度</p>  |
| <i>elevation</i> | <p>单精度数据。提升的角度。</p> <p>有效范围是 0 到 90 度。如果是 90 度，你就是从图表的数上方看；如果是 0 度，你就是从图表的正面看。缺省是 30 度。缺省地，使用度来衡量提升的角度。然而，这些设置使用 <b>AngleUnits</b> 属性的值。其他选项有弧度和梯度</p> |

## Set 方法（VtColor）

设置 VtColor 对象的 R,G,B 值。

应用于

VtColor 对象。

语法

*object.Set(red, green, blue)*

Set 方法的语法有如下几个部分：

| 部分                    | 描述                     |
|-----------------------|------------------------|
| <i>object</i>         | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>red,green,blue</i> | 整数。混合颜色的 RGB 值         |

说明

RGB 指定了要显示指定颜色中红绿蓝的相对密度。标准 RGB 值的有效范围是 0 到 16777215。RGB 中任何超过 255 的参数都认为是 255。

## Set 方法（Weighting）

设置 Weighting 对象的基和风格。

应用于

Weighting 对象。

语法

*object.Set (basis, style)*

Set 方法的语法有如下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象          |
| <i>basis</i>  | 一个 VtChPieWeightBasis 常量，标识权重类型 |
| <i>style</i>  | 一个 VtChPieWeightStyle 常量，标识权重因子 |

请参阅

PieWeightBasis 常量，PieWeightStyle 常量。

## SetData 方法（MSChart）

设置图表数据格子中指定数据点的数据。

应用于

MSChart 控件

语法

*object.SetData(row, column, dataPoint, nullFlag)*

SetData 方法的语法有如下几个部分：

| 部分               | 描述                        |
|------------------|---------------------------|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象    |
| 部分               | 描述                        |
| <i>row</i>       | 整数。标识包含了数据点的行             |
| <i>column</i>    | 整数。标识包含了数据点的列             |
| <i>dataPoint</i> | 双精度数据。数据点的值               |
| <i>nullFlag</i>  | 整数。指示数据点的值是否是 <b>null</b> |

# SetSize 方法

一次性地重新设置图表数据行数和列数，以及与图表数据格子关联的列标签级数和行标签级数。

应用于

DataGrid 对象。

语法

*object.SetSize(rowLabelCount, columnLabelCount, dataRowCount, columnLabelCount)*

SetSize 方法的语法有如下几个部分：

| 部分                      | 描述                     |
|-------------------------|------------------------|
| <i>object</i>           | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>rowLabelCount</i>    | 整数。返回或设置行标签的级数         |
| <i>dolumnLabelCount</i> | 整数。返回或设置列标签的级数         |
| <i>dataRowCount</i>     | 整数。返回或设置数据的行数          |
| <i>dataColumnCount</i>  | 整数。返回或设置数据的列数          |

说明

该方法可以替代 RowCount, ColumnCount, RowLabelCount 和

ColumnLabelCount。

## Shadow 对象

保存了有关图表元素阴影外观的信息。

语法

Shadow

属性

Brush 属性， Style 属性（MSChart）， Offset 属性。

请参阅

Brush 对象， Coor 对象。

示例

下面的例子设置了图表背景饼图的阴影部分。

```
Private Sub Command1_Click()
```

```
    ' Show shadow for title.
```

```
    With MSChart1.Title
```

```
        .Location.Visible = True
```

```
.Text = "Chart Title"  
.Backdrop.Frame.Width = 1  
.Backdrop.Frame.FrameColor.Set 255, 0, 0  
.Backdrop.Frame.Style = VtFrameStyleSingleLine  
.Backdrop.Shadow.Style = VtShadowStyleDrop  
.Backdrop.Shadow.Offset.x = 10  
.Backdrop.Shadow.Offset.y = 10
```

End With

End Sub

## Shadow 属性

返回一个 Shadow 对象的引用，描述图表元素阴影外观。

应用于

Backdrop 对象。

语法

*object*.Shadow

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

# Show 属性

返回或设置一个值，该值确定是否在图表中显示系列标记。

应用于

SeriesMarker 对象。

语法

*object*.Show [=*boolean*]

Show 属性的语法有如下几个部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>boolean</i> | 一个布尔表达式，控制是否显示系列的标记，如“设置”中所示 |

设置

boolean 值的设置如下：

| 设置    | 描述        |
|-------|-----------|
| True  | 显示图表系列标记  |
| False | 不显示图表系列标记 |



## ShowGuideLine 属性

返回或设置一个值，该值确定图表中是否显示连接系列中数据点的连线。

应用于

Series 对象。

语法

*object.ShowGuideLines (axisId, index) [=boolean]*

ShowGuideLines 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象                        |
| <i>axisId</i>  | 整数。一个 <b>VtChAxisId</b> 常量，指定系列的坐标轴           |
| <i>index</i>   | 整数。保留将来使用。对于当前版本的 <b>MSChart</b> 控件，1 是唯一有效的值 |
| <i>boolean</i> | 一个布尔表达式，控制是否在第二个坐标轴上显示系列，如“设置”中所示             |

设置

boolean 值的设置如下：

| 设置    | 描述        |
|-------|-----------|
| True  | 显示系列的基准线  |
| False | 不显示系列的基准线 |

请参阅

AxisId 常量

## ShowLegend 属性

返回或设置一个值，该值确定图表中的说明部分是否可视。

应用于

MSChart 控件。

语法

*object*.ShowLegend [=*boolean*]

ShowLegend 属性的语法有如下几个部分：

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>boolean</i> | 一个布尔表达式，控制在图表中是否显示基准线，如“设置”中所示 |

# 设置

boolean 值的设置如下：

| 设置    | 描述                               |
|-------|----------------------------------|
| True  | 说明图注显示在 <b>Location</b> 对象指定的位置上 |
| False | （缺省）图表中不显示说明图注。缺省的说明位置在图表的右边     |

# 说明

缺省的说明位置在图表的右边。

## ShowLine 属性

返回或设置一个值，该值确定图表中连接数据点的线条是否可视。

# 应用于

**Series** 对象。

# 语法

*object*.ShowLine [=*boolean*]

ShowLine 属性的语法有如下几个部分：

| 部分             | 描述                               |
|----------------|----------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象           |
| <i>boolean</i> | 一个布尔表达式，控制是否显示图表中数据点的连线，如“设置”中所示 |

## 设置

boolean 值的设置如下：

| 设置    | 描述           |
|-------|--------------|
| True  | 图表中显示数据点的连线  |
| False | 图表中不显示数据点的连线 |

## Size 属性（MSChart）

返回或设置图表元素的点数大小。

## 应用于

Marker 对象， VtFont 对象。

## 语法

*object*.Size [=size]

Size 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>size</i>   | 单精度数据。图表元素的点数大小        |

### Sort 属性（MSChart）

返回或设置饼图图表中的排序类型。

应用于

Plot 对象。

语法

*object*.Sort [=*type*]

Sort 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>type</i>   | 整数。一个 VtSortType 常量，描述绘制的排序次序 |

请参阅

SortType 常量。

## SpaceColor 属性

返回一个 VtColor 对象的引用，指定填充图表中两个窗体之间区域所使用的颜色。

应用于

Frame 对象。

语法

*object*.SpaceColor

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

FrameColor 属性。

# Stacking 属性

设置一个值，该值确定图表中所有的系列是否放入栈中。

应用于

MSChart 控件。

语法

*object*.Stacking [=*boolean*]

Stacking 属性的语法有如下几个部分：

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>boolean</i> | 一个布尔表达式，控制是否将图表系列放入栈中，如“设置”中所示 |

设置

boolean 值的设置如下：

| 设置    | 描述           |
|-------|--------------|
| True  | 所有的图表系列都放入栈中 |
| False | 图表系列不放入栈中    |

## StackOrder 属性

如果一个系列与其他系列放入栈中，返回或设置在何处绘制该系列。

应用于

SeriesPosition 对象。

语法

*object*.StackOrder [=position]

StackOrder 属性的语法有如下几个部分：

| 部分              | 描述                     |
|-----------------|------------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>position</i> | 整数。指定系列堆次序。低的次序排在栈的底部  |

## Standing 属性

返回或设置一个值，该值指定坐标轴标签是水平显示在 x 或 z 平面内还是垂直显示在 y 平面内。



应用于  
Label 对象。

语法

*object.Standing [=boolean]*

Standing 属性的语法有如下几个部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象       |
| <i>boolean</i> | 一个布尔表达式，指定如何显示坐标轴标签，如“设置”中所示 |

设置

boolean 值的设置如下：

| 设置    | 描述                     |
|-------|------------------------|
| True  | 在 y 平面的文本基准线上垂直显示坐标轴标签 |
| False | 在 x 或 z 平面内水平显示坐标轴标签   |

# StartingAngle 属性

返回或设置饼图块起始绘制位置。

应用于

Plot 对象。

语法

*object*.StartingAngle [=*angle*]

StartingAngle 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>angle</i>  | 单精度数据。角度可以是度、弧度或梯度，这得看 AngleUnits 属性的设置。0 值是 3 点钟位置。如果 ClockWise 设置为逆时针方向，90 度就是 12 点钟位置；如果 ClockWise 设置为顺时针方向，90 度就是 6 点钟位置。有效范围是-360 到 360 度 |

# StatLine 对象

描述如何在图表中显示统计线。

语法

StatLine

属性:

Width 属性 (MSChart), Style 属性 (StatLine), Flag 属性, VtColor 属性。

示例:

下面的例子给图表统计线设置了颜色和画笔参数。

```
Private Sub Command1_Click()  
    ' Show all statistic lines for series 2.  
    MSChart1.chartType = VtChChartType2dLine  
    With MSChart1.plot.SeriesCollection.(2).StatLine  
        .VtColor.Set 128, 128, 255  
        .Flag = VtChStatsMinimum Or VtChStatsMaximum _  
            Or VtChStatsMean Or VtChStatsStddev Or _  
            VtChStatsRegression  
        .Style(vtChStatsMinimum) = VtPenStyleDotted  
        .width = 2  
    End With  
End Sub
```

## StatLine 属性

返回一个 StatLine 对象的引用，描述如何在图表中显示统计线。

应用于

Series 对象。

语法

*object*.StatLine

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Style 属性（MSChart）

返回或设置绘制某个图表元素所使用的风格。

应用于

Brush 对象，Fill 对象，Frame 对象，Marker 对象，Pen 对象，Shadow 对象，Tick 对象，VtFont 对象，Weighting 对象。

语法

*object*.Style [=style]

Style 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>style</i>  | 对于 Brush 对象，是一个描述刷子图案的 VtBrushStyle 常量<br>对于 Fill 对象，是一个描述填充风格的 VtFillStyle 常量<br>填充使用的刷子可以是实心的或有图案的<br>对于 Frame 对象，是一个描述窗体类型的 VtFrameStyle 常量<br>对于 Marker 对象，是一个描述标记类型的 VtMarkerStyle 常量<br>对于 Pen 对象，是一个描述画笔风格的 VtPenStyle 常量<br>对于 Shadow 对象，是一个描述阴影类型的 VtShadowStyle 常量<br>对于 Tick 对象，是一个描述坐标轴分度标记的 VtChAxisTickStyle 常量<br>对于 VtFont 对象，是一个描述字体风格的 VtFontStyle 常量<br>对于 Weighting 对象。是一个描述权重因子方法的 VtChPieWeightStyle 常量 |

请参阅

AxisTickStyle 常量，BorderStyle 常量（MSChart），FillStyle 常量，FontStyle 常量，FrameStyle 常量，MarkerStyle 常量，PenStyle 常量，PieWeightStyle 常量，ShadowStyle 常量，Style 属性（StatLine），StatsType 常量。

## Style 属性 (StatLine)

返回或设置显示统计线的线条类型。

应用于

StatLine 对象。

语法

*object.Style (type) [=style]*

Style 属性的语法有如下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象           |
| <i>type</i>   | 整数。描述线条类型的 <b>VtChStats</b> 常量   |
| <i>style</i>  | 整数。描述统计线风格的 <b>VtPenStyle</b> 常量 |

请参阅

Style 属性 (MSChart)，PenStyle 常量。

# SubPlotLabelPosition 属性

返回或设置在图表饼图中显示标签的位置。

应用于

Plot 对象。

语法

*object*.SubPlotLabelPosition [=pos]

SubPlotLabelPosition 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                            |
| <i>pos</i>    | 整数。描述图表标签位置的一个<br>VtChSubPlotLabelLocationType 常量 |

请参阅

SubPlotLabelPositionType 常量。

## Text 属性（MSChart）

返回或设置显示图表元素的文本，如坐标轴标题，数据点标签，脚注或窗体标题等。

应用于

AxisTitle 对象， DataPointLabel 对象， Footnote 对象， Title 对象。

语法

*object*.Text [=*text*]

Text 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>text</i>   | 字符串表达式。包含了图表元素中显示的文本   |

说明

Text 属性是“应用于”中指定对象的缺省属性。



## extLayout 对象

表示文本的显示位置和方向。

语法

TextLayout

属性

WordWrap 属性（MSChart）， HorzAlignment 属性， Orientation 属性（MSChart），

VertAlignment 属性。

示例

下面的例子设置了图表的标题文本位置和显示方向。

```
Private Sub Command1_Click()  
    ' Sets the title text position and orientation.  
    With Form1.MSChart1.Title  
        .Location.Visible = True  
        .Location.LocationType = VtChLocationTypeLeft  
        .Text = "Title TextLayout"  
    End With  
    With Form1.MSChart1.Title.TextLayout
```

```
.Orientation = VtOrientationUp  
.HorzAlignment = VtHorizontalAlignmentCenter  
.VertAlignment = VtVerticalAlignmentCenter  
End With  
End Sub
```

## TextLayout 属性

返回一个 TextLayout 对象的引用，描述文本的位置和显示方向。

应用于

AxisTitle 对象，DataPointLabel 对象，Footnote 对象，Label 对象（Item），Legend 对象，Title 对象。

语法

*object*.TextLayout

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## TextLength 属性

返回或设置图表坐标轴标题、数据点标签、脚注或图表标题中文本的字符数。

应用于

AxisTitle 对象， DataPointLabel 对象， Footnote 对象， Title 对象。

语法

*object*.TextLength [=size]

TextLength 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>size</i>   | 整数。文本中的字符数             |

## TextLengthType 属性

返回或设置一个值，该值指定如何绘制文本以优化屏幕或打印页面的外观。

应用于  
MSChart 控件。

语法

*object.TextLengthType [=type]*

TextLengthType 属性的语法有如下几个部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象              |
| <i>type</i>   | 整数。一个 VtTextLengthType 常量，指定绘制文本的方法 |

请参阅

TextLengthType 常量。

## Tick 对象

指定图表坐标轴分度的标记。

语法

Tick

## 属性

**Style** 属性（MSChart）， **Length** 属性（MSChart）。

## 示例

下面的例子设置了图表 y 坐标轴分度的长度和风格。

```
Private Sub Command1_Click()  
    ' Set the tick for y axis.  
    With Form1.MSChart1.Plot.Axis(VtChAxisIdY, 1).Tick  
        .Length = 500  
        .Style = VtChAxisTickStyleOutside  
    End With  
End Sub
```

## Tick 属性

返回一个 **Tick** 对象的引用，描述图表坐标轴分度标记。

## 应用于

**Axis** 对象。

语法

*object.Tick*

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Title 对象

标识图表的文本。

语法

**Title**

属性

**Text** 属性(MSChart), **TextLength** 属性, **Backdrop** 属性, **Font** 属性(MSChart), **Location** 属性, **TextLayout** 属性, **VtFont** 属性。

方法

**Select** 方法。

请参阅

**Backdrop** 对象。

## Title 属性 (MSChart)

返回一个 Title 对象的引用，描述图表的标题文本。

应用于

MSChart 控件。

语法

*object*.Title

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## TitleActivated 事件

当用户双击图表标题时产生该事件。你可以取消该事件，显示你自己的对话框，从而替换标准的用户界面。

应用于

MSChart 控件。

语法

Private Sub *object*\_TitleActivated(*mouseFlags* As Integer, *cancel* As Integer)

TitleActivated 事件的语法有如下几个部分：

| 部分                | 描述                             |
|-------------------|--------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 该参数当前尚未使用                      |

## 设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 `mouseFlags` 设置为：

| 常量  | 描述                     |
|---|------------------------|
| <code>VtChMouseFlagsShiftKeyDown</code>   | 如果按下了 <b>SHIFT</b> 键   |
| <code>VtChMouseFlagsControlKeyDown</code> | 如果按下了 <b>CONTROL</b> 键 |

## TitleSelected 事件

当用户单击图表标题时产生该事件。

应用于

MSChart 控件。



语法

```
Private Sub object_TitleSelected(mouseFlags As Integer, cancel As Integer)
```

TitleSelected 事件的语法有如下几个部分：

| 部分                | 描述                             |
|-------------------|--------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>mouseFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示 |
| <i>cancel</i>     | 该参数当前尚未使用                      |

设置

事件处理器确定在单击鼠标按钮时是否按下了键，并将 mouseFlags 设置为：

| 常量                           | 描述              |
|------------------------------|-----------------|
| VtChMouseFlagsShiftKeyDown   | 如果按下了 SHIFT 键   |
| VtChMouseFlagsControlKeyDown | 如果按下了 CONTROL 键 |

TitleText 属性

返回或设置图表标题中显示的文本。

应用于

MSChart 控件。

语法

*object*.TitleText [=*text*]

TitleText 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>text</i>   | 显示图表标题的文本              |

说明

该属性提供了简单地设置或返回图表标题的方法。该属性与使用 MSChart.Title.Text 具有同样的功能。

## TitleUpdated 事件

当图表文本改变时产生该事件。

应用于

MSChart 控件。

语法

```
Private Sub object_TitleActivated(updateFlags As Integer)
```

TitleUpdated 事件的语法有如下几个部分：

| 部分                 | 描述                             |
|--------------------|--------------------------------|
| <i>object</i>      | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>updateFlags</i> | 整数。指明在用户单击鼠标按钮时是否有键按下，如“设置”中所示 |

设置

下表列出了 updateFlags 常量：

| 常量                 | 描述                     |
|--------------------|------------------------|
| VtChNoDisplay      | 没有更新标记；图表显示不受影响（缺省为 0） |
| VtChDisplayPlot    | 更新将导致图表重新绘制            |
| VtChLayoutPlot     | 更新将导致图表重新布局            |
| VtChDisplayLegend  | 更新将导致说明图例重新绘制          |
| VtChLayoutLegend   | 更新将导致说明图例重新布局          |
| VtChLayoutSeries   | 更新将导致系列重新布局            |
| VtChPostionSection | 图表的一部分已被移动或改变了大小       |

## ToDefault 方法

将图表参数设置为初始值。

应用于

MSChart 控件。

语法

*object*.ToDefault

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## TwipsToChartPart 方法

使用 x 和 y 坐标指定图表的一部分。

应用于

MSChart 控件。

语法

*object*.TwipsToChartPart(*xVal*, *yVal*, *part*, *index1*, *index2*, *index3*, *index4*)

TwipsToChartPart 方法的语法有如下几个部分：

| 部分               | 描述   |
|------------------|--|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>xVal,yVal</i> | 长整数。点的水平和垂直坐标  |
| <i>part</i>      | 整数。指明 <i>xVal</i> 和 <i>yVal</i> 位置的图表部分。是 <i>VtChPartType</i> 常量   |
| <i>index1</i>    | 整数。如果 <i>part</i> 指一个系列或一个数据点，则该参数指明是哪个系列。系列的计数与相应的列相同，从左到右，开始是 1。如果 <i>part</i> 是一个坐标轴或坐标轴标签，该参数就使用 <i>VtChAxisId</i> 标识坐标轴的类型                                    |
| <i>index2</i>    | 整数。如果 <i>part</i> 指一个数据点，该参数就指明 <i>index1</i> 标明的是系列中的哪个点。数据点，按数据栅格中相应行中出现的顺序计数，从上到下，开始是 1。如果 <i>part</i> 是指坐标轴、坐标轴标题或坐标轴标签，则该参数就指向当前尚未使用的坐标轴索引。在这种情况下，该参数的有效值就是 1 |
| <i>index3</i>    | 整数。如果 <i>part</i> 是坐标轴标签，该参数指定标签的级数。坐标轴标签的级数从坐标轴外算起，开始是 1。如果 <i>part</i> 非坐标轴标签，不使用该参数   |
| <i>index4</i>    | 整数。当前该参数尚未使用   |

请参阅

*AxisId* 常量，*PartType* 常量。

## Type 属性 (MSChart)

返回或设置坐标轴的刻度类型。

应用于

AxisScale 对象。

语法

*object*.Type [=type]

Type 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>type</i>   | 一个 VtChScaleType 常量，描述坐标轴刻度类型 |

请参阅

ScaleType 常量。

## TypeByChartType 方法

如果图表类型设置为 **chType**，则返回绘制系列使用的系列类型。该方法允

许你不必设置图表类型而获取指定图表的系列类型信息。

应用于  
Series 对象。

语法

*object.TypeByCharType (chtype)*

TypeByCharType 方法的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象       |
| 部分            | 描述                           |
| <i>chtype</i> | 整数。一个 VtChCharType 常量，描述图表类型 |

返回值

根据 chtype 参数所指定的图表类型，TypeByCharType 方法将返回一个 VtChSeriesType 值，如下：

| 常量                    | 值  | 描述    |
|-----------------------|----|-------|
| VtChSeriesTypeDefault | -1 | 缺省值   |
| VtChSeriesType3dBar   | 0  | 3D 棒图 |
| VtChSeriesType2dBar   | 1  | 2D 棒图 |

续表

|                      |    |       |
|----------------------|----|-------|
| VtChSeriesType3dLine | 5  | 3D 线段 |
| VtChSeriesType2dLine | 6  | 2D 线段 |
| VtChSeriesType3dArea | 7  | 3D 区域 |
| VtChSeriesType3dStep | 9  | 3D 台阶 |
| VtChSeriesType2dStep | 10 | 2D 台阶 |
| VtChSeriesType2dXY   | 11 | XY    |
| VtChSeriesType2dPie  | 24 | 2D 饼图 |

请参阅

CharType 常量， SeriesType 常量。

## UniformAxis 属性

返回或设置一个值，该值确定图表中的坐标轴单位是否都统一。

应用于

Plot 对象。

语法

*object*.UniformAxis [=boolean]



UniformAxis 属性的语法有如下几个部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，控制坐标轴单位，如“设置”中所示 |

## 设置

boolean 值的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | 图表中所有坐标轴的单位都统一  |
| False | 图表坐标轴单位不统一。单位的尺度由 AutoLayout 或 LocationRect 属性设置的绘制区域的大小和位置决定。如果 AutoLayout 设置为 True，绘制区域的大小或位置由其他自动布局的元素确定。如果 AutoLayout 设置为 False，使用 LocationRect 指定的坐标设置绘制区域的大小并确定坐标轴的尺度单位 |

## UpdateFlags 常量

下面所列出的 UpdateFlags 常量可用于各种事件以便确认图表的哪一部分发生了改变。

| 常量                  | 值  | 描述                |
|---------------------|----|-------------------|
| VtChNoDisplay       | 0  | 没有更新标志；图表显示效果不受影响 |
| VtChDisplayPlot     | 1  | 更新将使得图表的绘图区域重新绘制  |
| VtChLayoutPlot      | 2  | 更新将使得图表的绘图区域重新布局  |
| VtChDisplayLegend   | 4  | 更新将使得图表的说明区域重新绘制  |
| VtChLayoutLegend    | 8  | 更新将使得图表的说明区域重新布局  |
| VtChLayoutSeries    | 16 | 更新将使得序列重新布局       |
| VtChPositionSection | 32 | 图表的一部分被移动或改变了大小   |

### 说明

这些常量可以用于下面的事件：

|                     |                    |                      |
|---------------------|--------------------|----------------------|
| AxisLabelUpdated 事件 | DataUpdated 事件     | PointLabelUpdated 事件 |
| AxisTitleUpdated 事件 | FootnoteUpdated 事件 | PointUpdated 事件      |
| AxisUpdated 事件      | LegendUpdated 事件   | SeriesUpdated 事件     |
| ChartUpdated 事件     | PlotUpdated 事件     | TitleUpdated 事件      |

### ValueFormat 属性

返回或设置标签显示为值的格式。

应用于

DataPointLabel 对象。

语法

*object*.ValueFormat [=*format*]

ValueFormat 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>format</i> | 字符串。描述显示文本标签值的格式       |

说明

使用 DataPointLabel 对象的 Component 属性改变标签类型。

## ValueScale 对象

显示坐标轴数值的尺度。

语法

ValueScale

## 属性

Auto 属性 (ValueScale), MajorDivision 属性, Maximum 属性, Minimum 属性, MinorDivision 属性。

请参阅

CategoryScale 对象, ValueScale 属性。

## 示例

下面的例子使用 ValueScale 对象为二维棒图图表设置了主次分隔线的颜色。

```
Private Sub Command1_Click()  
    ' Set chart type to 2d bar.  
    MSChart1.ChartType = VtChChartType2dBar  
    ' Use manual scale to display y axis (value axis).  
    With MSChart1.Plot.Axis(VtChAxisIdY).ValueScale  
        .Auto = False  
        .MajorDivision = 2  
        .MinorDivision = 5  
    End With  
    ' Show major grid line in red and minor grid line
```

' in blue.

With MSChart1.Plot.Axis(VtChAxisIdY).AxisGrid

.MajorPen.VtColor.Set 255, 0, 0

.MajorPen.Width = 4

.MinorPen.VtColor.Set 0, 0, 255

.MinorPen.Width = 2

End With

End Sub

## ValueScale 属性

返回一个 ValueScale 对象的引用，描述显示坐标轴数值所使用的尺度。

应用于

Axis 对象。

语法

*object*.ValueScale

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

请参阅

CategoryScale 属性。

## VerticalAlignment 属性

返回或设置垂直对齐文本的方法。

应用于

TextLayout 对象。

语法

*object*.VerticalAlignment [=*type*]

VerticalAlignment 属性的语法有如下几个部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                   |
| <i>type</i>   | 整数。一个 VtVerticalAlignment 常量，描述文本垂直对齐的方法 |

请参阅

VerticalAlignment 常量。

## View3D 对象

表示一个三维图表的物理方位。

语法

View3D

属性

Elevation 属性， Rotation 属性。

方法

Set 方法（View3D）

示例

下面的例子使用 View3D 对象设置了三维棒图图表的提升和旋转角度。

```
Private Sub Command1_Click()
```

```
    ' Set the chart type to 3d bar.
```

```
    Form1.MSChart1.ChartType = VtChChartType3dBar
```

```
    With Form1.MSChart1.Plot.View3d
```

```
        .Elevation = 90      ' Look directly down onto the  
                             ' top of the chart.
```

```
        .Rotation = 90
```

End With  
End Sub

## View3D 属性

返回一个 View3D 对象的引用，描述三维图表的物理方位。

应用于

Plot 对象。

语法

*object*.View3D

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Visible 属性（MSChart）

返回或设置一个值，该值确定是否显示一个图表元素。

应用于

MSChart 控件，AxisTitle 对象，Location 对象，Marker 对象。



# 语 法

*object.Visible* [=*boolean*]

Visible 属性的语法有如下几个部分：

| 部分             | 描述                          |
|----------------|-----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象      |
| <i>boolean</i> | 一个布尔表达式，指定是否显示图表元素，如“设置”中所示 |

# 设置

boolean 值的设置如下：

| 设置    | 描述             |
|-------|----------------|
| True  | 显示图表、坐标轴、标签或标记 |
| False | 隐藏元素           |

# VtColor 对象

描述图表中绘制使用的颜色。

语法

VtColor

属性

Automatic 属性，Blue 属性，Green 属性，Red 属性。

方法

Set 方法（VtColor）。

## VtColor 属性

返回一个 VtColor 对象的引用，描述图表中的绘制颜色。

应用于

Pen 对象，StatLine 对象，VtFont 对象。

语法

*object*.VtColor

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## VtFont 对象

显示图表文本所使用的字体。

### 语法

**VtFont**

### 属性

**Style** 属性（MSChart），**Size** 属性（MSChart），**VtColor** 属性，**Effect** 属性，**Name** 属性（VtFont）。

### 示例

下面的例子设置了图表标题的字体参数。

```
Private Sub Command1_Click()  
'Ask user to supply a title  
MSChart1.Title.Text = Zinput Box("Title?")  
    ' Make Chart Title visible.  
    MSChart1.Title.Location.Visible = True  
    ' Set font for Chart Title.  
    With MSChart1.Title.VtFont  
        .Name = "Times New Roman"  
        .Size = 18
```

```
.Style = VtFontStyleBoldOr_VtFontStyleItalic  
' Use both StrikeThrough and Underline in the  
' text.  
.Effect = VtFontEffectStrikeThrough Or _  
VtFontEffectUnderline  
' Set text color to Blue.  
.VtColor.Set 0, 0, 255  
End With  
End Sub
```

## VtFont 属性

返回一个 VtFont 对象的引用，描述显示图表文本所使用的字体。

应用于

AxisTitle 对象，DataPointLabel 对象，Footnote 对象，Label 对象（Item），Legend 对象，Title 对象。

语法

*object*.VtFont

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Wall 对象

三维图表中 y 坐标轴所在的一个平面区域。

语法

**Wall**

属性

**Brush** 属性，**Width** 属性（MSChart），**Pen** 属性。

方法

**Set** 方法（Weighting）。

请参阅

**Brush** 对象。

示例

下面的例子显示了三维图表中一个有颜色的墙壁。

```
Private Sub Command1_Click()
```

```
    ' Displays a colored wall for a 3D chart.
```

```
    Form1.MSChart1.ChartType = VtChChartType3dBar
```

```
    With Form1.MSChart1.Plot.Wall
```

```
.Brush.Style = VtBrushStylePattern  
.Brush.Index = VtBrushPatternChecks  
.Brush.FillColor.Set 255, 120, 120  
.Brush.PatternColor.Set 120, 120, 0  
.Width = 20  
End With  
End Sub
```

## Wall 属性

返回一个 Wall 对象的引用，描述三维图标中 y 坐标轴所在的一个平面区域。

应用于

Plot 对象。

语法

*object*.Wall

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Weighting 对象

表示在同一个棒图中一个饼图块与其他饼图块的相对大小。

语法

Weighting

属性

Style 属性（MSChart），Basis 属性。

语法

Set 方法（Weighting）。

示例

下面的例子演示了饼图图表中的权重。

```
Private Sub Command1_Click()  
    ' Show the weighting of the pie.  
    Form1.MSChart1.ChartType = VtChChartType2dPie  
    With Form1.MSChart1.Plot.Weighting  
        .Basis = VtChPieWeightBasisTotal  
        .Style = VtChPieWeightStyleArea  
    End With
```

End Sub

## Weighting 属性

返回一个 **Weighting** 对象的引用，描述在同一个图表中一个饼图块与其他饼图块的相对大小。

应用于

Plot 对象。

语法

*object*.Weighting

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

## Width 属性（MSChart）

返回或设置图表元素以磅计的宽度。

应用于

Frame 对象，MSChart 控件，Pen 对象，StatLine 对象，Wall 对象。



语法

*object.Width* [=width]

Width 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>width</i>  | 单精度数据。图表元素的宽度          |

WidthToHeightRatio 属性

返回或设置图表高度对于图表宽度的百分比。

应用于

Plot 对象。

语法

*object.WidthToHeightRatio* [=pctg]

WidthToHeightRadio 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>pctg</i>   | 单精度数据。图表高度的百分比         |

## WordWrap 属性（MSChart）

返回或设置一个值，该值确定文本是否环绕。

应用于

TextLayout 对象。

语法

*object*.WordWrap [=*boolean*]

WordWrap 属性的语法有如下几个部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>boolean</i> | 一个布尔表达式，指定文本是否环绕，如“设置”中所示 |

# 设置

*boolean* 值的设置如下：

| 设置    | 描述    |
|-------|-------|
| True  | 文本环绕  |
| False | 文本不环绕 |

## X 属性

返回或设置图表浮动坐标的 *x* 值。

应用于

Coor 对象，LCoor 对象，LightSource 对象。

语法

*object.X* [=*x*]

X 属性的语法有如下几个部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                   |
| <i>x</i>      | 单精度数据（对于 Lcoor 对象是长整数）。标识坐标中的 <i>x</i> 值 |

请参阅

Y 属性，Set 方法（Coor,LCoor）。

## XGap 属性

返回或设置棒图在  $x$  坐标轴上的间距。该间距表示为棒图宽度的百分比。

应用于

Plot 对象。

语法

*object.xGap* [=spacing]

xGap 属性的语法有如下几个部分：

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>spacing</i> | 单精度数据。棒图宽度的百分比。0 值将使得棒图系列彼此接触 |

## Y 属性

返回或设置图表浮动坐标的 y 值。

应用于

Coor 对象， LCoor 对象， LightSource 对象。

语法

*object*.Y [=y]

Y 属性的语法有如下几个部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象            |
| <i>y</i>      | 单精度数据（对于 LCoor 对象是长整数）。标识坐标中的 y 值 |

请参阅

X 属性， Set 方法（Coor,LCoor）。

## Z 属性

返回或设置坐标位置的 z 值。

应用于  
LightSource 对象。

语法

*object.Z [=z]*

Z 属性的语法有如下几个部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                   |
| <i>z</i>      | 单精度数据（对于 Lcoor 对象是长整数）。标识坐标中的 <i>z</i> 值 |

请参阅

X 属性，Y 属性。

## ZGap 属性

返回或设置三维棒图在 *z* 坐标轴上的间距。该间距表示为棒图深度的百分比。

应用于  
Plot 对象。

## 语法

*object.zGap [=spacing]*

**zGap** 属性的语法有如下几个部分：

| 部分 | 描述 |
|----|----|
|----|----|

---

|                |                                     |
|----------------|-------------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象              |
| <i>spacing</i> | 单精度数据。棒图深度的百分比。0 值将使得棒图系列在 z 轴上彼此接触 |

# MSComm 控件

MSComm 控件为应用程序提供了串口通信功能，该应用程序允许通过串口发送和接收数据。

## 语法

## MSComm

## 说明

MSComm 控件提供了两种处理通信的方法：

- 事件驱动通信，是一种功能很强的处理串口活动的方法。在大多数情况下，用户需要获知事件发生的时间，例如，在 CD（Carrier Detect）线或 RTS（Request To Send）线上有字符到达或发生了改变等。在这种情况下，使用 MSComm 控件的 OnComm 事件捕获和处理这些通信事件。OnComm 也可以捕获和处理通信中的错误。要获取所有事件和通信错误的完整清单，请参阅 CommEvent 属性。
- 用户也可以在每个重要的程序功能之后检查 CommEvent 属性的值来检测事件和通信错误。这对小的自含程序可能比较常用。例如，如果编写一个简单的电话拨号程序，那么在接收了每个字符后都产生一个事件并没有



意义，因为你只打算从调制解调器中接收 OK 响应信息。

使用的每个 **MSComm** 控件都与一个串口对应。如果在应用程序中需要访问多个串口，必须使用多个 **MSComm** 控件。可以在 **Windows** 控制面板中修改串口地址的中断地址。

虽然 **MSComm** 控件有许多很重要的属性，但应先熟悉下面几个属性：

| 属性       | 描述                           |
|----------|------------------------------|
| CommPort | 设置或返回通信端口号                   |
| Settings | 以字符串的形式设置或返回波特率、奇偶校验、数据位和停止位 |
| PortOpen | 设置或返回通信端口的状态。也可以打开和关闭端口      |
| Input    | 返回和删除接收缓冲区中的字符               |
| Output   | 将字符串写入发送缓冲区                  |

## 示例

下面的例子演示了使用调制解调器进行基本的通信。

```
Private Sub Form_Load ()  
    ' Buffer to hold input string  
    Dim Instring As String  
    ' Use COM1.  
    MSComm1.CommPort = 1  
    ' 9600 baud, no parity, 8 data, and 1 stop bit.
```

```
MSComm1.Settings = "9600,N,8,1"
' Tell the control to read entire buffer when Input
' is used.
MSComm1.InputLen = 0
' Open the port.
MSComm1.PortOpen = True
' Send the attention command to the modem.
MSComm1.Output = "ATVtQ0"& + Chr$(13) 'Ensure that
' the modem responds with "OK"
' Wait for data to come back to the serial port.
Do
    DoEvents
Buffer$ = Buffer$ MSComm1.Input
Loop Until InStr (Buffer$, "OK" & vbCRLF)
' Read the "OK" response data in the serial port.
' Close the serial port.
MSComm1.PortOpen = False
End Sub
```

**注意：**MSComm 控件可以使用查询或事件驱动的方法从串口接收数据。本例使用了查询的方法。对于事件驱动方法的例子，请参阅 OnComm 事件的帮助。

# Break 属性

设置或清除断开信号状态。在设计时该属性不可用。

## 语法

```
object.Break [=value]
```

Break 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>value</i>  | 一个布尔表达式，指明是否设置断开信号状态，如“设置”中所示 |

## 设置

*vlaue* 值的设置如下：

| 设置    | 描述       |
|-------|----------|
| True  | 设置断开信号状态 |
| False | 清除断开信号状态 |

## 说明

当设置为 **True** 时，**Break** 属性就发送一个断开信号。断开信号将停止字符发送，将发送线置为断开状态，直到将 **Break** 属性置为 **False**。

通常，为短时间区以及只有当用于通信的设备需要设置断开状态时才设置断开状态。

## 数据类型

**Boolean。**

## 示例

下面的例子演示了如何发送十分之一秒的断开信号。

```
' Set the Break condition.
```

```
MSComm1.Break = True
```

```
' Set duration to 1/10 second.
```

```
Duration! = Timer + .1
```

```
' Wait for the duration to pass.
```

```
Do Until Timer > Duration!
```

```
    Dummy = DoEvents()
```

```
Loop
```

```
' Clear the Break condition.
```

```
MSComm1.Break = False
```

## CDHolding 属性

通过检查 CD 线的状态来查询是否有载波。CD（Carrier Detect，载波检测）是调制解调器发送给连接计算机的指示调制解调器在线的信号。该属性在设计时不可用，在运行时是只读的。

### 语法

*object*.CDHolding

CDHolding 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

### 设置

CDHolding 属性的设置如下：

| 设置    | 描述    |
|-------|-------|
| True  | CD 线高 |
| False | CD 线低 |

### 说明

当 CD 线高（CDHolding = True）并超时，MSComm 控件就将 CommEvent

属性设置为 commEventCDTO（Carrier Detect Timeout Error），并产生一个 OnComm 事件。CD 也称为 RLSD（Receive Line Singal Detect，接收线信号检测）。

注意：尤其要注意的是，在主机应用程序比如电子公告板中要侦测载波信号的丢失问题，因为呼叫者随时都有可能挂起。

数据类型：

Boolean。

### CommEvent 属性

返回最近的通信事件或错误。该属性在设计时不可用，在运行时是只读的。

语法

*object*.CommEvent

CommEvent 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

说明

尽管通信事件或错误都产生 OnComm 事件，但 CommEvent 属性中包含了

事件或错误的代码。要确定导致 OnComm 事件的实际错误或事件，必须引用 CommEvent 属性。

CommEvent 属性为通信事件或错误返回下列值之一。在该控件的对象库中也可以找到这些常量。

通信错误包含了下面的设置：

| 常量                   | 值    | 描述   |
|----------------------|------|--|
| ComEventBreak        | 1001 | 收到了断开信号  |
| ComEventCTST<br>O    | 1002 | Clear To Send Timeout。在发送字符时，在系统指定的事件内，CTS（Clear To Send）线是低电平   |
| ComEventDSRT<br>O    | 1003 | Data Set Ready Timeout。在发送字符时，在系统指定的事件内，DSR（Data Set Ready）线是低电平   |
| ComEventFrame        | 1004 | 数据帧错误。硬件检测到一个数据帧错误   |
| ComEventOverru<br>n  | 1006 | 端口溢出。硬件中的字符尚未读，下一个字符又到达，并且丢失   |
| ComEventCDTO         | 1007 | Carrier Detect Time。在发送字符时，在系统指定的事件内，CD（Carrier Detect）线是低电平。CD 也称为 RLSD（Receive Line Singal Detect，接收线信号检测） |
| ComEventRxOve<br>r   | 1008 | 接收缓冲区溢出。在接收缓冲区中没有空间  |
| ComEventRxPari<br>ty | 1009 | 奇偶校验错。硬件检测到奇偶校验错误  |

续表

|                |      |  |
|----------------|------|--|
| ComEventTxFull | 1010 | 发送缓冲区满。在对发送字符排队时，发送缓冲区满                    |
| ComEventDCB    | 1011 | 检取端口 DCB（Device Control BlicK）时发生了没有预料到的错误 |

通信事件包含了下面的设置：

| 常量               | 值 | 描述  |
|------------------|---|---|
| ComEvSend        | 1 | 发送缓冲区中的字符数比 Sthreshold 值低   |
| ComEvRecei<br>ve | 2 | 接收到了 Rthreshold 个字符。持续产生该事件，直到使用了<br>Input 属性删除了接收缓冲区中的数据                     |
| ComEvCTS         | 3 | CTS（Clear To Send）线改变   |
| ComEvDSR         | 4 | DSR（Data Set Ready）线改变。当 DSR 从 1 到 0 改变时，<br>该事件发生                            |
| ComEvCD          | 5 | CD（Carrier Detect）线改变   |
| ComEvRing        | 6 | 检测到响铃信号。一些 URAT（Universal Asynchronous<br>Reciver-Transmitters,通用异步收发器）不支持该事件 |
| ComEvEOF         | 7 | 收到了 EOF 字符（ASCII 字符 26）   |

数据类型

Integer。



# CommID 属性

返回标识通信设备的句柄。该属性在设计时不可用，在运行时是只读的。

## 语法

*object*.CommID

CommID 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

## 说明

该值与 Windows API 函数 CreateFile 返回值是一样的。在 Windows API 中调用任何通信例程都使用该值。

## 数据类型

Long

# CommPort 属性

返回或设置通信端口号。

# 语法

```
object.CommPort [=value]
```

CommPort 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 指定端口号的整数               |

# 说明

在设计时，可以将 *value* 置为 1 到 16 之间的整数（缺省为 1）。然而，当试图使用 **PortOpen** 属性打开一个不存在的端口号时，**MSComm** 控件就产生 68 错误（Device unavailable）。

**警告：** 必须在打开端口前设置 **CommPort** 属性。

# 数据类型

Integer。

# CTSHolding 属性

检查 CTS（Clear To Send）线的状态，确定是否可以发送数据。通常，调

制解调器给连接的计算机发送 CTS 信号，指明正在处理发送过程。该属性在设计时不可用，在运行时是只读的。

语法

*object*.CTSHolding

CTSHolding 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

CTSHolding 属性的设置如下表所示：

| 设置    | 描述                   |
|-------|----------------------|
| True  | CTS（Clear To Send）线高 |
| False | CTS（Clear To Send）线低 |

说明

当 CTS 线低(CTSHolding = False)并超时，MSComm 控件就设置 CommEvent 属性为 comEventCTSTO（Clear To Send Timeout）并激活 OnComm 事件。

在 RTS/CTS( Request To Send / Clear To Send)硬件握手中使用 CTS 线。如果需要确定 CTS 线的状态，可以使用 CTSHolding 属性手动查询。

要获取更多信息或握手协议，请参阅 Handshaking 属性。

# 数据类型

Boolean

## DSRHolding 属性

确定 DSR（Data Set Ready）线的状态。通常，调制解调器给连接的计算机发送 CTS 信号，指明准备就绪。该属性在设计时不可用，在运行时是只读的。

### 语法

*object*.DSRHolding

object 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

DSRHolding 属性返回如下值：

| 值     | 描述                  |
|-------|---------------------|
| True  | DSR（DataSetReady）线高 |
| False | DSR（DataSetReady）线低 |

### 说明

当 DSR 线高 (DSRHolding = False) 并超时，MSComm 控件设置 CommEvent 属性为 comEventDTSTO（Data Set Ready Timeout）并激活 OnComm 事件。

在为 DTE (Data Terminal Equipment) 计算机编写 DSR/DTR(Data Set Ready / Data Terminal Equipment )握手例程时，该属性很有用。

数据类型

Boolean

DRTEnable 属性

确定在通信过程中是否使用 DTR(Data Terminal Ready)线。通常，DTR 信号是计算机发送给调制解调器的，指明计算机可以接收输入数据了。

语法

```
object.DRTEnable [=value]
```

DRTEnable 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>value</i>  | 一个布尔表达式，指定是否使用 DTR 线，如“设置”中所示 |

设置

*value* 值的设置如下：

| 设置    | 描述           |
|-------|--------------|
| True  | 使用 DTR 线     |
| False | （缺省）禁止 DTR 线 |

## 说明

当 DTREnable 属性设置为 True 时,如果端口打开,DTR 线被设置为高(on),如果端口关闭,DTR 线被设置为低(off)。

**注意:** 在大多数情况下,将 DTR 线设置为低将挂起电话。

## 数据类型

Boolean

## EOFEnable 属性

EOFEnable 属性确定 MSComm 控件是否检查输入中的 EOF 字符。如果发现了 EOF 字符,停止输入,将把 CommEvent 属性置为 comEvEOF,激活 OnComm 事件。

## 语法

*object*.EOFEnable [=value]

EOFEnable 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>value</i>  | 一个布尔表达式，指定发现了 EOF 字符时是否触发 OnComm 事件，如“设置”所示 |

## 设置

value 设置如下：

| 设置    | 描述                             |
|-------|--------------------------------|
| True  | 当发现 EOF 字符时就触发 On Comm 事件      |
| False | (缺省值)当发现 EOF 字符时不触发 On Comm 事件 |

## 说明

当 EOFEnable 属性设置为 False 时，控件将不在输入流中扫描 EOF 字符。

## Error 消息（MSComm 控件）

下表列出了 MSComm 控件可捕获的错误消息：

| 常量                      | 值    | 描述             |
|-------------------------|------|----------------|
| ComInvalidPropertyValue | 380  | 无效的属性值         |
| ComSetNotSupported      | 383  | 属性只读           |
| ComGetNotSupported      | 394  | 属性只读           |
| ComPortOpen             | 8000 | 端口打开时该存在无效     |
|                         | 8001 | 超时设置必须比 0 值大   |
| ComPortInvalid          | 8002 | 无效的端口号         |
|                         | 8003 | 属性只在运行时有效      |
|                         | 8004 | 属性在运行时是只读的     |
| ComPortAlreadyOpen      | 8005 | 端口已经打开         |
|                         | 8006 | 设备标识符无效或不支持    |
|                         | 8007 | 不支持设备的波特率      |
|                         | 8008 | 指定的字节大小无效      |
|                         | 8009 | 缺省参数错误         |
|                         | 8010 | 硬件不可用（被其他设备锁住） |
|                         | 8011 | 函数不能分配队列       |
| ComNoOpen               | 8012 | 设备没有打开         |
|                         | 8013 | 设备已经打开         |
|                         | 8014 | 不能使用通信通知       |



续表

|                       |      |                  |
|-----------------------|------|------------------|
| ComSetCommStateFailed | 8015 | 不能设置通信状态         |
|                       | 8016 | 不能设置通信事件屏蔽       |
| ComPortNotOpen        | 8018 | 该存在只在端口打开是有效     |
|                       | 8019 | 设备忙              |
| ComReadError          | 8020 | 通信设备读错误          |
| ComDCBError           | 8021 | 检取端口设备控制块时出现内部错误 |

Handshaking 属性

设置或返回硬件握手协议。

语法

*object*.Handshaking [=*value*]

Handshaking 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>value</i>  | 一个整数表达式，指定握手协议，如“设置”中所示 |

# 设置

*value* 值的设置如下:

| 设置            | 值 | 描述   |
|---------------|---|--|
| comNone       | 0 | (缺省) 没有握手协议                                  |
| ComXOnXOff    | 1 | XON/XOFF 握手协议                                |
| ComRTS        | 2 | RTS/CTS(Request To Send / Clear To Send)握手协议 |
| ComRTSXOnXOff | 3 | RTS 和 XON/XOFF 协议                            |

# 说明

握手协议是指从硬件端口向接收缓冲区传输数据时使用的内部通信协议。当一个字符数据到达串口时，通信设备必须将其移动到接收缓冲区，使应用程序可以读取数据。如果没有接收缓冲区而应用程序希望直接从硬件读取每个字符，你将可能丢失数据，因为数据到达的速度很快。

握手协议保证缓冲区溢出时不丢失没有数据，到达串口的数据将被通信设备很快地移动到接收缓冲区中。

# 数据类型

Integer

## InBufferCount 属性

返回在接收缓冲区中等待的字符数。该属性在设计时不可用。

### 语法

*object*.InBufferCount [=*value*]

InBufferCount 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>value</i>  | 一个整数表达式，指定在接收缓冲区中等待的字符数 |

### 说明

InBufferCount 是指已被接收到接收缓冲区、等待应用程序读取的字符数。将 InBufferCount 设置为 0 将清除接收缓冲区。

**注意：**不要将该属性与 InBufferize 属性混淆。InBufferSize 属性反映的是接收缓冲区总的大小。

### 数据类型

Integer

## InBufferSize 属性

设置或返回接收缓冲区大小的字节数。

语法

*object*.InBufferSize[=*value*]

InBufferSize 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数表达式，指定接收缓冲区大小的字节数  |

说明

InBufferSize 是指整个接收缓冲区的大小。缺省是 1024 个字节。不要将该属性与反映等待应用程序读取字符数的 InBufferCount 属性混淆。

**注意：**对接收缓冲区设置的越大，应用程序可以使用的内存就越少。然而，如果接收缓冲区太小，缓冲区将会溢出，除非使用握手协议。通常，将缓冲区设置为 1024。如果发生了溢出，就增大该缓冲区，以满足应用程序传输速率。

# 数据类型

## Integer

### Input 属性

返回或删除接收缓冲区中的数据流。该属性在设计时不可用，在运行时是只读的。

#### 语法

*object*.Input

Input 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

#### 说明

InputLen 属性确定了 Input 属性读入的字符数。将 InputLen 属性设置为 0 将导致 Input 属性读入整个接收缓冲区的内容。

InputMode 属性确定了 Input 属性检取数据的类型。如果 InputMode 设置为 comInputModeText, 则 Input 属性将返回 Variant 类型的文本数据。如果 InputMode

属性设置为 `comInputModeBinary`，则 `Input` 属性返回一个 `Variant` 类型的二进制字节矩阵。

## 数据类型

`Variant`

## 示例

下面的例子演示了如何从接收缓冲区中检取数据。

```
Private Sub Command1_Click()  
    Dim InString as String  
    ' Retrieve all available data.  
    MSComm1.InputLen = 0  
  
    ' Check for data.  
    If MSComm1.InBufferCount Then  
        ' Read data.  
        InString = MSComm1.Input  
    End If  
End Sub
```

## InputLen 属性

设置和返回 **Input** 属性从接收缓冲区中读取的字符数。

应用于

MSComm 控件。

语法

*object*.InputLen [=value]

InputLen 属性的语法有如下几个部分：

| 部分            | 描述                                      |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                  |
| <i>value</i>  | 一个整数表达式，指定 <b>Input</b> 属性从接收缓冲区中读取的字符数 |

说明

InputLen 属性的缺省值是 0。将 InputLen 属性设置为 0 将导致 Input 属性读入整个接收缓冲区的内容。

如果接收缓冲区中没有可读的字符，就返回空字符串。在使用 **Input** 之前，

用户可以检查 **InBufferCount** 的值，以便确定是否读取了所要求的字符数。

从输出格式固定的计算机读取定长数据块时该属性很有用。

## 数据类型

### Integer

#### 示例

下面的例子演示了如何检取 10 个字符的数据。

```
Private Command1_Click()  
Dim CommData as String  
' Specify a 10 character block of data.  
MSComm1.InputLen = 10  
' Read data.  
CommData = MSComm1.Input  
End Sub
```

## InputMode 属性

设置和返回 **Input** 属性所检取数据的类型。



应用于  
MSComm 控件。

语法

*object*.InputMode [=value]

InputMode 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个值或常量，指定输入模式，如“设置”中所示 |

设置

value 值的设置如下：

| 常量                 | 值 | 描述                   |
|--------------------|---|----------------------|
| comInputModeText   | 0 | （缺省）Input 属性检取的数据是文本 |
| comInputModeBinary | 1 | Input 属性检取的数据是二进制数据  |

说明

InputMode 属性确定了如果通过 Input 属性检取数据。数据将被看作字符串

或 一个二进制字节矩阵。

对于使用 ANSI 字符集的数据应使用 `comInputModeText` 类型。所有其他数据如内嵌控制字符、Null 等都使用 `comInputModeBinary` 类型。

### 示例

下面的例子从通信端口中读取了 10 字节的二进制数据并分配给一个矩阵。

```
Private Sub Command1_Click()  
Dim Buffer as Variant  
Dim Arr() as Byte  
  
' Set and open port  
MSComm1.CommPort = 1  
MSComm1.PortOpen = True  
  
' Set InputMode to read binary data  
MSComm1.InputMode = comInputModeBinary  
  
' Wait until 10 bytes are in the input buffer  
Do Until MSComm1.InBufferCount < 10  
    DoEvents  
Loop  
  
' Store binary data in buffer  
Buffer = MSComm1.Input
```

```
' Assign to byte array for processing
```

```
Arr = Buffer
```

```
End Sub
```

## NullDiscard 属性

确定是否将 null 字符传输给接收缓冲区。

应用于

MSComm 控件。

语法

*object*.NullDiscard [=value]

NullDiscard 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                     |
| <i>value</i>  | 一个布尔表达式，确定是否将端口来的 Null 字符发送给接收缓冲区，如“设置”中所示 |

# 设置

value 值的设置如下：

| 设置    | 描述                        |
|-------|---------------------------|
| True  | 不把端口来的 Null 字符传输给接收缓冲区    |
| False | （缺省）将端口来的 Null 字符传输给接收缓冲区 |

# 说明

Null 字符定义为 ASCII 字符 0——Chr\$(0)。

# 数据类型

Boolean

## OnComm 事件

当 CommEvent 属性值改变时产生该事件，表明产生了通信事件或通信错误。

# 应用于

MSComm 控件。

语法

```
Private Sub object_OnComm()
```

OnComm 事件的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |

说明

CommEvent 属性捕获了 OnComm 产生事件或错误的代码。注意，将 RThreshold 或 Sthreshold 属性置为 0 将不捕获 comEvReceive 和 comEvSend 事件。

示例

下面的例子演示了如何处理通信错误和事件。你可以在 Case 语句的后面插入代码处理相应的错误或事件。

```
Private Sub MSComm_OnComm ()  
    Select Case MSComm1.CommEvent  
        ' Handle each event or error by placing  
        ' code below each case statement  
  
        ' Errors  
        Case comEventBreak    ' A Break was received.
```

Case comEventCDTO ' CD (RLSD) Timeout.  
Case comEventCTSTO ' CTS Timeout.  
Case comEventDSRTO ' DSR Timeout.  
Case comEventFrame ' Framing Error  
Case comEventOverrun ' Data Lost.  
Case comEventRxOver ' Receive buffer overflow.  
Case comEventRxParity ' Parity Error.  
Case comEventTxFull ' Transmit buffer full.  
Case comEventDCB ' Unexpected error retrieving DCB]

' Events

Case comEvCD ' Change in the CD line.  
Case comEvCTS ' Change in the CTS line.  
Case comEvDSR ' Change in the DSR line.  
Case comEvRing ' Change in the Ring Indicator.  
Case comEvReceive ' Received RThreshold # of  
' chars.  
Case comEvSend ' There are SThreshold number of  
' characters in the transmit  
' buffer.  
Case comEvEof ' An EOF charater was found in  
' the input stream

End Select

End Sub

## OutBufferCount 属性

返回在发送缓冲区中等待的字符数。你可以使用该属性清除发送缓冲区。该属性在设计时不可用。

应用于

MSComm 控件。

语法

*object*.OutBufferCount [=*value*]

OutBufferCount 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数表达式，指定发送缓冲区中等待的字符数 |

说明

将 OutBufferCount 属性置为 0 将清除发送缓冲区。

注意：不要将 OutBufferCount 属性与 OutBufferSize 属性混淆起来。

# 数据类型

## Integer

### OutBufferSize 属性

返回或设置发送缓冲区的字节大小。

应用于

MSComm 控件。

语法

*object*.OutBufferSize [=*value*]

OutBufferSize 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数表达式，指定发送缓冲区的字节大小   |

说明

OutBufferSize 指整个发送缓冲区的大小。缺省值是 512 字节。不要将该属性与 OutBufferCount 属性混淆起来,OutBufferCount 属性反映的是发送缓冲区当



前等待的字节数。

**注意：**对发送缓冲区设置的越大，应用程序可以使用的内存就越少。然而，如果你的发送缓冲区太小，缓冲区将会溢出，除非使用握手协议。通常，将缓冲区设置为 512 字节。如果发生了溢出，就增大该缓冲区，以满足你的应用程序传输速率。

## 数据类型

### Integer

## Output 属性

将数据写入发送缓冲区。该属性在设计时不可用，在运行时是只写的。

### 应用于

MSComm 控件。

### 语法

*object*.Output [=value]

Output 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串，是写入发送缓冲区中的字符     |

## 说明

**Output** 属性可以发送文本数据或二进制数据。要使用 **Output** 属性发送文本数据，你必须指明包含字符串的一个 **Variant** 变量。要发送二进制数据，必须将包含字节矩阵的 **Variant** 变量传递给 **Output** 属性。

通常，如果你给应用程序发送 **ANSI** 字符串，可以文本方式发送。如果数据包含了内嵌控制字符、**Null** 字符等，必须将其作为二进制传递过去。

## 数据类型

**Variant**

## 示例

下面的例子演示了如何将用户输入的字符发送给串口。

```
Private Sub Form_KeyPress (KeyAscii As Integer)
```

```
    Dim Buffer as Variant
```

```
    ' Set and open port
```

```
    MSComm1.CommPort = 1
```

```
    MSComm1.PortOpen = True
```

```
Buffer = Chr$(KeyAscii)
MSComm1.Output = Buffer
End Sub
```

## ParityReplace 属性

设置或返回一个字符，该字符在方式了奇偶校验错误时将替换数据流中的无效字符。

应用于

MSComm 控件。

语法

*object*.ParityReplace [=value]

ParityReplace 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 字符串表达式，表示替换字符，如“说明”中所示 |

## 说明

**parity bit** 是一个比特位，与指定的数据比特一起传送，可以提供一定的错误检测功能。

当你使用奇偶校验位时，**MSComm** 控件将数据中所有设置（即“1”）的比特位相加，来检测结果的奇偶性（根据端口的奇偶设置）。

缺省地，控件使用问号（“？”）替换无效字符。将 **ParityReplace** 属性设置为空字符串（""）将在奇偶校验错误时不替换无效字符，但仍将 **CommEvent** 属性设置为 **commEventRXParity** 并触发 **OnComm** 事件。

**ParityReplace** 字符是面向字节的操作，必须是单字节字符。你可以指定任意的 0 到 255 的 ANSI 字符作为替换字符。

## 数据类型

**String**

## PortOpen 属性

设置或返回通信端口的状态（打开或关闭）。在设计时该属性不可用。

## 应用于

**MSComm** 控件。

# 语法

*object.PortOpen [=value]*

PortOpen 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个布尔表达式，指定通信端口的状态      |

# 说明

将 PortOpen 属性设置为 True 将打开端口。设置为 False 将关闭端口并清除接收和发送缓冲区。当你的应用程序终止时，MSComm 控件将自动关闭串口。

在打开端口前，确定 CommPort 属性设置为正确的端口号。如果 CommPort 属性设置为无效的端口号，在你试图打开端口时，MSComm 控件将产生 error 68(Device unavailable)错误。

而且，你的串口设备必须支持 Settings 属性中的设置。如果你的硬件设备不支持 Settings 属性中的一些设置，则你的硬件工作或许不正确。

如果端口在打开之前将 DTREnable 或 RTSEnable 属性设置为 True，则当端口关闭时这些属性就自动设置为 False。另外，DTR 和 RTS 线保持它们原来的状态。

## 数据类型

Boolean

## 示例

下面的例子打开端口 1，波特率是 9600，没有奇偶校验，8 个数据位，一个停止位。

```
MSComm1.Settings = "9600, n, 8, 1"
```

```
MSComm1.CommPort = 1
```

```
MSComm1.PortOpen = True
```

## RThreshold 属性

设置或返回在 MSComm 控件将 CommEvent 属性设置为 comEvReceive 并在产生 OnComm 事件之前所接收的字符数。

## 应用于

MSComm 控件。

# 语法

*object.Rthreshold [=value]*

RThreshold 属性的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象         |
| <i>value</i>  | 一个整数表达式，指定产生 OnComm 事件之前接收的字符数 |

# 说明

将 RThreshold 属性设置为 0（缺省）将在接收字符时不产生 OnComm 事件。  
例如，将 RThreshold 属性设置为 1 将导致 MSComm 控件在每个字符放入缓冲区时就触发 OnComm 事件。

# 数据类型

Integer

# RTSEnable 属性

确定是否使用 RTS（Request To Send）线。通常 RTS 信号是计算机发送该连接的调制解调器，请求准许发送数据。

应用于

MSComm 控件。

语法

*object*. RTSEnable [=*value*]

RTSEnable 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>value</i>  | 一个布尔表达式，指定是否使用 RTS 线，如“设置”中所示 |

设置

value 值的设置如下：

| 设置    | 描述            |
|-------|---------------|
| True  | 使用 RTS 线      |
| False | （缺省）不使用 RTS 线 |

说明

当 RTSEnable 属性设置为 True 时，打开端口将把 RTS 线置为高，关闭端



口将把 RTS 线置为低。

在 RTS/CTS 握手协议中使用 RTS 线。如果你需要确定 RTS 线的状态，RTSEnable 属性允许你手工检测该线。

想了解有关握手协议中的更多信息，请参阅 HardShaking 属性。

## 数据类型

Boolean

## Settings 属性

设置或返回波特率、奇偶校验、数据位和停止位参数。

应用于

MSComm 控件。

语法

*object*.Settings [=value]

Settings 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，代表通信端口设置，如下所示 |

### 说明

当端口打开时 **value** 值设置不正确，**MSComm** 控件就产生 **error 380(Invalid property value)**错误。

**value** 由 4 部分组成，格式如下：

“BBBB, P, D, S”

**BBBB** 是波特率，**P** 是奇偶校验，**D** 是数据位，**S** 是停止位。**Value** 的缺省  
值如下：

" 9600, N, 8, 1"

下表列出了有效的波特率：

| 设置   |
|------|
| 110  |
| 300  |
| 600  |
| 1200 |
| 2400 |

9600（缺省）

14400

19200

28800

38400（保留）

56000（保留）

128000（保留）

256000（保留）

下表列出了有效的奇偶校验值：

| 设置 | 描述       |
|----|----------|
| E  | 偶校验      |
| M  | 屏蔽       |
| N  | （缺省）None |
| O  | 奇校验      |
| S  | 空格       |

下表列出了有效的数据位：

## 设置

---

4

5

6

7

8（缺省）

下表列出了有效的停止位：

## 设置

---

1（缺省）

1.5

2

数据类型

String

示例

下面的例子将端口设置为波特率 9600，没有奇偶校验，8 个数据位，一个停止位。

```
MSComm1.Settings = "9600, N, 8, 1"
```

## SThreshold 属性

设置或返回在 **MSComm** 控件将 **CommEvent** 属性设置为 **comEvSend** 并产生 **OnComm** 事件之前所发送缓冲区中允许的最少字符数。

应用于

**MSComm** 控件。

语法

*object*.Sthreshold [=value]

SThreshold 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象                      |
| <i>value</i>  | 一个整数表达式，指定产生 <b>OnComm</b> 事件之前发送缓冲区中的最少字符数 |

说明

将 **SThreshold** 属性设置为 0（缺省）将在发送字符时不产生 **OnComm** 事件。

例如，将 **SThreshold** 属性设置为 1 将导致发送缓冲区完全变空。

如果发送缓冲区中的字符比 **value** 值小，则将 **CommEvent** 属性设置为

comEvSend，并产生 OnComm 事件。comEvSend 事件只产生一次，在字符数低于 SThreshold 时。例如，如果 SThreshold 等于 5，当字符数从 5 降低到 4 时产生一次 comEvSend 事件。如果输出队列中的字符数总不多于 SThreshold 属性值，则永不产生该事件。

## 数据类型

Integer

## CoolBar 控件

CoolBar 控件包含一个 Band 对象集合，用于产生与窗体相关联的可配置工具条。

### 说明

CoolBar 控件是容器控件，它一般包含两个或多个可以由用户重新定义大小和重新调整的 Bands。每一个 Band 包含一个单一的 Child 控件。

### 属性

BandBorders 属性, Bands 属性, FixedBackground 属性 (Band 对象), Orientation 属性 (Coolbar 控件), Picture 属性 (Coolbar 控件), RowCount 属性 (Coolbar 控件), VariantHeight 属性, TabIndex 属性, Align 属性, DragIcon 属性, DragMode 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MouseIcon 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), ImageList 属性 (ActiveX 控件)。

## 方法

Drag 方法, Move 方法, OLEDrag 方法(ActiveX 控件), Refresh 方法(ActiveX 控件), Item 方法。

## 事件

HeightChanged 事件, DragDrop 事件, DragOver 事件, Resize 事件, OLECompleteDrag 事件(ActiveX 控件), OLEDragDrop 事件(ActiveX 控件), OLEDragOver 事件(ActiveX 控件), OLEGiveFeedback 事件(ActiveX 控件), OLESetData 事件(ActiveX 控件), OLEStartDrag 事件(ActiveX 控件), Click 事件(ActiveX 控件), DblClick 事件(ActiveX 控件), MouseDown, MouseUp 事件(ActiveX 控件), MouseMove 事件(ActiveX 控件)。

## 请参阅

Bands 集合, Band 对象。

## Add 方法 (Bands 集合)

添加一个新的 Band 对象到带区集合中。

## 应用于

Bands 集合



## 语法

*object*.Add ([*index* As Long], [*key* As String], [*caption* As String], [*image* As Variant], [*newrow* As Boolean], [*child* As Object], [*visible* As Boolean])

Add 方法语法含有这些部分:

| 部分             | 描述   |
|----------------|--|
| <i>object.</i> | 一个对象表达式，它给 CoolBar 控件的 Bands 集合赋值  |
| <i>index</i>   | 选项。一个长整数，它在 Bands 集合中唯一识别一个带区  |
| <i>key</i>     | 选项。一个识别 Band 对象的一个唯一字符串。用这个值检索一个指定的 Band 对象  |
| <i>caption</i> | 选项。一个含有显示在带区上标题的字符串  |
| <i>image</i>   | 选项。来指定与 <i>object</i> 一起使用的 ListImage 对象的一个整数或唯一字符串。这个整数是 Index 属性的值，字符串是 Key 属性的值 |
| <i>newrow</i>  | 选项。缺省 = False。指定新的带区是否将显示在它自己行上的布尔表达式  |
| <i>child</i>   | 选项。指定控件将成为带区子控件的一个对象引用   |
| <i>visible</i> | 选项。缺省 = False。指定在运行时新带区是否可见的一个布尔表达式  |

# AllowVertical 属性（Band 对象）

返回或设置一个值，表示当 CoolBar 控件的取向设置为竖直时是否显示带区。

应用于  
Band 对象。

语法

*object.AllowVertical* [= *boolean*]

AllowVertical 属性语法有以下部分:

| 部分             | 描述                               |
|----------------|----------------------------------|
| <i>object.</i> | 一个对象表达式，它给 CoolBar 控件的 Band 对象赋值 |
| <i>boolean</i> | 一个布尔表达式，它指定带区是可见的或隐藏的            |

设置值

boolean 设置值为:

| 设置值   | 描述        |
|-------|-----------|
| True  | (缺省值)带区可见 |
| False | 带区隐藏      |

## 说明

当 **Orientation** 设置为竖向时，要在启动时隐藏带区，应在设计时将 **AllowVertical** 属性设定为 **False**。在代码中设定该属性，使您在运行时能够响应特定的事件而隐藏或重新显示带区。

**注意：**当 **Orientation** 属性设置为竖向时，为了使带区可见，**AllowVertical** 和 **Visible** 二者必须是 **True**。如果二者任何一个为 **False**，则带区将不可见。

## 请参阅

**Orientation** 属性（**Coolbar** 控件）。

## Band 对象

一个 **Band** 对象，代表 **CoolBar** 控件 **Bands** 集合的单个带区。

## 说明

带区是 **CoolBar** 控件内部的一个区域，它可以包含单个子控件、标题和图像。每一个 **Band** 可以由用户在运行时单独移动和重新定义大小。

在设计时，使用 **CoolBar** 控件的“属性页”中“带区”制表符的“插入带区”和“删除带区”按钮，可以向 **Bands** 集合内插入和删除 **Band** 对象。在运行

时，可以用 Bands 集合的 Add 和 Remove 方法来添加和删除 Band 对象。

## 属性

AllowVertical 属性 (Band 对象)，Caption 属性 (Band 对象)，Child 属性 (Band 对象)，FixedBackground 属性 (Band 对象)，Image 属性 (Coolbar 控件)，MinHeight, MinWidth 属性 (Band 对象)，NewRow 属性，Picture 属性 (Coolbar 控件)，Position 属性 (Band 对象)，Style 属性 (Band 对象)，UseCoolbarColors 属性，UseCoolbarPicture 属性，Height, Width 属性 (ActiveX 控件)，Index 属性 (ActiveX 控件)，Tag 属性 (ActiveX 控件)，Visible 属性 (ActiveX 控件)，BackColor, ForeColor 属性 (ActiveX 控件)，Key 属性 (ActiveX 控件)。

请参阅

Coolbar 控件，Bands 集合。

## BandBorders 属性

返回或设置一个值，表示 CoolBar 控件是否显示分割带区的窄线。

应用于

Coolbar 控件。

# 语法

*object*.BandBorders [= *boolean*]

BandBorders 属性语法有以下部分:

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，它给 CoolBar 控件赋值  |
| 部分             | 描述                       |
| <i>boolean</i> | 一个布尔表达式，指定带区边框是可见的或者是隐藏的 |

# 设置

boolean 设置值为:

| 设置值   | 描述         |
|-------|------------|
| True  | (缺省值) 边框可见 |
| False | 边框隐藏       |

# 说明

为了删除带区之间的边框，应该在设计时将 BandBorders 属性设定为 False。在代码中设定该属性，可使您在运行时响应特定的事件而隐藏或显示带区之间的边框。设置该属性影响 CoolBar 控件内的所有带区。

**注意：**BandBorders 属性不影响 CoolBar 自身的外部边框。CoolBar

边框总是显示。

## Bands 集合

一个集合，其元素代表 CoolBar 控件上的带区。Bands 集合具有 Count 属性，它指定集合内的 Band 对象数目。

### 语法

*object*.Bands.Count

*object*.Bands (*index*)

Bands 集合语法有以下部分:

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，它给 CoolBar 控件赋值   |
| <i>index</i>  | 一个整数，其范围从 1 到 Bands.Count |

### 说明

Bands 集合列举 CoolBar 控件上的带区。例如，可以用它来改变一个控件所有带区的 BackColor 属性。

**注意：**Bands 集合不是 Visual Basic Collection 类的成员；无法创

造其实例。

## 属性

**Count** 属性（ActiveX 控件）。

## 方法

**Add** 方法（Bands 集合），**Remove** 方法（ActiveX 控件），**Clear** 方法（ActiveX 控件），**Item** 方法。

## 请参阅

Coolbar 控件，Band 对象。

## Bands 属性

返回一个对 CoolBar 控件的 Band 对象集合的引用。

## 应用于

Coolbar 控件。

## 语法

*object*.Bands

**object** 置换元是一个对象表达式，用于给 **CoolBar** 控件赋值。

## 说明

可以用标准的集合方法（如 **Add** 和 **Remove** 方法）处理 **Band** 对象。集合内的每一个元素都可以按其索引值（**Index** 属性值）或唯一的键值（**Key** 属性值）访问。

## Caption 属性 （Band 对象）

返回或设置出现在 **CoolBar** 控件中 **Band** 对象上的标题。

## 应用于

**Band** 对象。

## 语法

*object.Caption* [= *string*]

**Caption** 属性语法有以下部分：

| 部分            | 描述   |
|---------------|--|
| <b>object</b> | 一个对象表达式，它给 <b>CoolBar</b> 控件的 <b>Bands</b> 集合内的一个 <b>Band</b> 赋值 |
| <i>string</i> | 一个字符串表达式，内容是标题中显示的文本   |



# 说明

标题在移动柄和子控件之间按左对齐方式显示。当一个带区竖直显示时，带区的宽度是依赖于标题宽度的。

## Child 属性 （Band 对象）

设置或返回对 CoolBar 控件上的 Band 对象内的一个子控件的引用。

应用于

Band 对象。

语法

Set *object*.Child = control

*object*.Child

Child 属性语法有以下部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个对象表达式，它给 CoolBar 控件的 Bands 集合内的 Band 赋值 |
| <i>control</i> | 一个表达式，它给 CoolBar 控件 Band 对象内的一个控件赋值       |

## 说明

在设计时可以通过在 **CoolBar** 控件“属性页”内“带区”选项卡中，从“Child”列表选择一个控件来设置带区的“Child”属性。运行时可以通过读取 **Child** 属性来检索 **Band** 对象上控件的名称。

要在运行时给带区指定新的子控件，应使用 **Set** 关键字。这将替换 **Band** 对象上的现有控件。

## EmbossHighlight 属性

设置或返回 **CoolBar** 控件中所显示图像的增强亮度颜色。

应用于

**Coolbar** 控件。

## 语法

*object*.EmbossHighlight [= *color*]

**EmbossHighlight** 属性语法有以下部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 一个对象表达式，它给 CoolBar 控件赋值          |
| <i>color</i>  | 一个数值或者常量，它决定一个对象的背景或前景的颜色，如设置值所示 |

## 设置

Visual Basic 采用 Microsoft Windows 操作环境的红-绿-蓝（RGB）配色方案。颜色的设置值如下：

| 设置值       | 描述  |
|-----------|---|
| 正常 RGB 颜色 | 通过颜色调色板指定，或者在代码中使用 RGB 或 QBColor 函数来指定                                |
| 系统缺省颜色    | 由“对象浏览器”的 Visual Basic (VB) 对象库列出的系统颜色常量指定。Windows 操作环境将替换用户在控制面板中的设置 |

## 说明

EmbossHighlight 属性的缺省设置是由常量 vbButtonHighlight 指定的系统缺省颜色。

一个正常 RGB 颜色的有效范围是 0 到 16,777,215 (&HFFFFFF)。一个在该范围内的数的高字节等于 0；更低三位字节，从最低到最高有效字节，分别决定了红、绿和蓝的数量。红、绿和蓝的成分的每一个分别用 0 和 255 (&HFF) 之间的一个数字代表。如果高字节不是 0，Visual Basic 使用系统颜色，如是在

用户“控制面板”设置值定义的，它由 Visual Basic (VB) 对象浏览器的对象库列出的常量定义。

注意：除非 CoolBar 控件的 Picture 属性包含一个有效的图像，并且 EmbossPicture 属性设为 True，该属性不影响图像的显示。

请参阅

EmbossPicture 属性，EmbossShadow 属性。

## EmbossPicture 属性

返回或设置一个值，表示 CoolBar 控件上的背景图片是以双色还是以多色显示。

应用于

CoolBar 控件。

语法

*object*.EmbossPicture [= *boolean*]

EmbossPicture 属性语法有以下部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>Object</i>  | 一个对象表达式，它给 CoolBar 控件赋值    |
| <i>Boolean</i> | 一个布尔表达式，它指定背景图片是以双色还是以多色显示 |

## 设置

boolean 设置如下：

| 设置值   | 描述               |
|-------|------------------|
| False | （缺省值）背景图片是以多色显示的 |
| True  | 背景图片是以双色显示的      |

## 说明

缺省 CoolBar 控件上的背景颜色将用包含在图像文件中指定给 Picture 属性的颜色显示。当 EmbossPicture 属性为 True 时，背景图片将抖动到 EmbossHighlight 和 EmbossShadow 属性中定义的两颜色

注意：除非 CoolBar 控件的 Picture 属性包含一个有效图像，该属性没有影响。如果包含 Band 对象的 UseCoolbarPicture 属性设为 False，则该属性对该 Band 没有影响。

## 请参阅

EmbossHighlight 属性，EmbossPicture 属性。

# EmbossShadow 属性

设置或返回 CoolBar 控件中所显示图像的阴影颜色。

应用于

CoolBar 控件。

语法

*object*.EmbossShadow [= *color*]

EmbossShadow 属性的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>Object</i> | 一个对象表达式，其值等于一个 CoolBar 控件    |
| <i>Color</i>  | 一个值或常量，确定对象的背景或前景颜色，如“设置”中所示 |

设置

Visual Basic 使用 Microsoft 操作环境 RGB 颜色方案。Color 的设置如下：

| 设置        | 描述                                    |
|-----------|---------------------------------------|
| 标准 RGB 颜色 | 使用颜色调色板或在代码中使用 RGB 或 QBColor 函数所指定的颜色 |

续表

|        |                                   |
|--------|-----------------------------------|
| 系统缺省颜色 | 在对象浏览器 VB 对象库中所列的系统颜色常量所指定的颜色     |
|        | Windows 操作环境将用户的选择替换为控制面板设置中所指定的值 |

## 说明

EmbossShadow 属性的缺省设置是 vbButtonFace 常量所指定的系统缺省颜色。

标准 RGB 颜色的有效范围是 0 到 16777215 (&HFFFFFF)。该范围中的数值，其高位字节等于 0；低三位字节，从低到高，分别决定了红绿蓝颜色量。红绿蓝组分别以 0 到 255 (&255) 之间的一个数值来表示。如果高字节不是 0，Visual Basic 就使用系统颜色，如用户控制面板中的设置以及由对象浏览器 VB 对象库所列常量所定义的颜色。

**注意：**除非 CoolBar 控件包含一个有效的图像，而且 EmbossPicture 属性设置为 Ture，否则该属性对于图像的显示将没有什么影响。

## 请参阅

EmbossHighlight,EmbossPicture 属性。

# FixedBackground 属性（Band 对象）

设置或返回一个值，表示 CoolBar 控件上的背景图片当带区重新排列时是否保持固定。

应用于  
Band 对象。

语法

*object.FixedBackground* [= *boolean*]

FixedBackGround 属性语法有以下部分：

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 一个对象表达式，它给 Band 控件赋值    |
| <i>boolean</i> | 一个布尔表达式，它指定背景图片为固定还是可移动 |

设置

boolean 设置值如下：

| 设置值   | 描述          |
|-------|-------------|
| True  | （缺省值）背景图片固定 |
| False | 背景图片不固定     |



## 说明

CoolBar 控件的 **Picture** 属性的缺省行为是用 CoolBar 控件的 “Top” 和 “Left” 作为起点在控件上跨所有带区上平铺显示图片。如果带区重新排列或重新调整其大小，图片跨所有带区重绘。

当 **FixedBackground** 属性设置为 **False** 时，图片在控件上的每个带区内平铺显示。如果带区重新排列或重新调整其大小，图片在每个带区内保持固定。

**注意：**该属性只适用于 CoolBar 自身的 **Picture** 属性。如果图片指定给 Band 对象的 **Picture** 属性且该带区的 **UseCoolbarPicture** 属性设置为 **False**，则 **FixedBackground** 属性对该带区没有影响。

## 请参阅

**Picture** 属性（Coolbar 控件），**UseCoolbarPicture** 属性。

## FixedOrder 属性

设置或返回一个值，表示是否允许用户在运行时重新排列 CoolBar 控件上带区的顺序。

## 应用于

CoolBar 控件。

# 语法

`object.FixedOrder [= boolean]`

FixedOrder 属性语法有以下部分：

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 一个对象表达式，它给 CoolBar 控件赋值 |
| <i>boolean</i> | 一个布尔表达式，它指定带区可见还是隐藏     |

# 设置

boolean 设置值如下：

| 设置值   | 描述           |
|-------|--------------|
| False | （缺省值）带区可重新排列 |
| True  | 带区不能重新排列     |

# 说明

设计时，要固定 CoolBar 控件上带区的排列，将 FixedOrder 属性设为 True 。这将阻止用户利用拖放重新排列带区。以代码的形式设定该属性您可在运行时响应一个特定事件时禁用或启用重新排列带区的能力。设定该属性影响 CoolBar 控件包含的所有带区。

## HeightChanged 事件

当一个 CoolBar 控件的 Height 改变时发生。

应用于

CoolBar 控件。

语法

```
Private Sub object_HeightChanged([index As Integer], newheight As Single)
```

HeightChanged 事件语法有这些部分:

| 部分               | 描述                       |
|------------------|--------------------------|
| <i>object</i>    | 一个对象表达式, 它给 CoolBar 控件赋值 |
| <i>index</i>     | 一个整数, 其值唯一识别控件组中的一个控件    |
| <i>newheight</i> | 一个单精度浮点数来指定控件新高度         |

说明

当 CoolBar 高度需要改变时, HeightChanged 事件在 Resize 事件之后发生。当用户在运行时重新排列带区或当一个或多个带区的高度按程序改变时, 这个事件可以发生。

当 CoolBar 由一个能抑制变化的容器宿主时, 该事件有用。在这种情况下,

在 **Resize** 事件过程中读取 **Height** 属性可能不可靠或者 **Resize** 事件可能不能被抑制。 **HeightChanged** 事件允许您添加代码来使 **CoolBar** 控件正常显示。

## Image 属性 （Coolbar 控件）

返回或设置一个值，它指定在 **ImageList** 控件内的哪一个 **ListImage** 对象将出现在 **Coolbar** 控件的 **Band** 对象上。

应用于

**Band** 对象。

语法

*object*.Image [= *index*]

**Image** 属性语法有以下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，它给 <b>Band</b> 对象赋值  |
| <i>index</i>  | 指定与 <i>object</i> 一起使用的 <b>ListImage</b> 对象的整数或唯一字符串。整数为 <b>Index</b> 属性的值，而字符串为 <b>Key</b> 属性的值 |

## 说明

在设定 **Image** 属性之前，您必须通过将 **CoolBar** 的 **ImageList** 属性设定给一个 **ImageList** 控件将 **ImageList** 控件和 **CoolBar** 控件关联起来。

在运行时，包含在 **Image** 属性中所指定的 **ListImage** 对象的图像将在位于移动柄和标题之间的 **Band** 对象上显示。图像在设计时不显示出来。

## MinHeight, MinWidth 属性（Band 对象）

设置或返回一个值，表示 **Band** 对象可调整到那一个最小的高度和宽度。

应用于

**Band** 对象。

语法

*object.MinHeight [= single]*

*object.MinWidth [= single]*

**MinHeight** and **MinWidth** 属性语法有以下部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 一个对象表达式，它给 <b>Band</b> 对象赋值 |
| <i>single</i> | 一个单精度浮点数值，来表示高度或宽度          |

## 说明

缺省 **MinHeight** 的值等于 **Band** 对象的子控件的 **Height**。您可将 **MinHeight** 设置到一个很大的值，但您不能重新调整 **Band** 低于缺省值以下。

**注意：**当 **CoolBar** 的 **Orientation** 设为竖向时，**MinHeight** 则变为 **MinWidth** 而 **MinWidth** 则变为 **MinHeight**。

## NewRow 属性

设置或返回 **Band** 对象是否在 **CoolBar** 控件的新一行中显示。

应用于

**Band** 对象。

语法

*object.NewRow* [= *boolean*]

**NewRow** 属性语法有以下部分：

| 部分             | 描述                          |
|----------------|-----------------------------|
| <i>object</i>  | 一个对象表达式，它给 <b>Band</b> 对象赋值 |
| <i>boolean</i> | 一个布尔表达式，来指定是否在新一行显示带区       |

## 设置

布尔设置值为：

| 设置值   | 描述                |
|-------|-------------------|
| False | （缺省值）带区将加入到现存最后一行 |
| True  | 带区将从新的一行开始        |

**注意：** `NewRow` 属性不影响 `Bands` 集合内的第一个 `Band` 对象。第一个 `Band` 总是从新的一行开始。

## Orientation 属性 （Coolbar 控件）

设定一个值，表示 `CoolBar` 控件是横向还是竖向。

应用于

`Coolbar` 控件。

语法

`object.Orientation [= enum]`

`Orientation` 属性语法有这些部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 一个对象表达式，它给 <b>CoolBar</b> 控件赋值 |
| <i>enum</i>   | 一个常量或数值，用于指定方向，如设置值所示          |

## 设置

**enum** 的设置值是:

| 常量                              | 值 | 描述      |
|---------------------------------|---|---------|
| <b>cc3OrientationHorizontal</b> | 0 | (缺省) 横向 |
| <b>cc3OrientationVertical</b>   | 1 | 竖向      |

## Picture 属性（Coolbar 控件）

返回或设置一个在控件中显示的图形。

应用于

**CoolBar** 控件，**Band** 对象。

语法

*object*.Picture = LoadPicture(*pathname*)

*object*.Picture [= *picture*]



Picture 属性语法有这些部分:

| 部分              | 描述   |
|-----------------|--|
| <i>object</i>   | 一个对象表达式，其值是“应用于”列表中的一个对象                     |
| <i>pathname</i> | 一个字符串表达式来指定含有图形的文件的路径和文件名，<br>如设置值所示的        |
| <i>Picture</i>  | Form 对象的 Picture 属性，PictureBox 控件，或 Image 控件 |

## 设置

picture 设置是:

| 设置值                 | 描述   |
|---------------------|--|
| (none)              | (缺省)无图片  |
| (Bitmap, GIF, JPEG) | 指定一个图形。设计时您可从“属性页”加载一个图形。运行时您也可使用 LoadPicture 函数设置这个属性 |

## 说明

CoolBar 控件的 Picture 属性显示在任意一个子控件后面跨 CoolBar 控件上所有带区的背景图形。 每一个 CoolBar 控件上的 Band 对象也有一个 Picture 属性和一个可用来覆盖控件上 Picture 属性的 UseCoolBarPicture 属性。

## 请参阅

UseCoolbarPicture 属性。

## Position 属性（Band 对象）

返回 CoolBar 控件内 Band 对象的位置。

应用于

Band 对象。

语法

*object*.Position [= *integer*]

Position 属性语法有这些部分：

| 部分             | 描述                   |
|----------------|----------------------|
| <i>object</i>  | 一个对象表达式，它给 Band 对象赋值 |
| <i>integer</i> | 一个整数来表示当前位置          |

说明

Position 属性可用来决定 CoolBar 控件内一个 Band 的当前位置。然而 Band 的 Index 保持不变时，重新排列带区 Position 发生变化。

对于处在左上角位置上的 Band 其 Position 属性起始值为 1，并且当 CoolBar 为横向时它从左到右、从上到下增加。当它为竖向时其 Position 从上到下，从左到右增加。

## RowCount 属性（Coolbar 控件）

返回在 CoolBar 控件上显示的行数。

应用于

CoolBar 控件。

语法

*object.RowCount* [= *number*]

RowCount 属性语法有这些部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 一个对象表达式，它给 CoolBar 控件赋值 |
| <i>number</i> | 一个长整数来指示行数              |

说明

CoolBar 控件所显示的行数并不一定要与带区数相同，因为在一行中可以显示多个带区。

## Style 属性（Band 对象）

设置或返回一个值，指示 CoolBar 控件中 Band 对象的大小变化行为。

应用于

Band 对象。

语法

*object*.Style [= *enum*]

Style 属性语法有这些部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 一个对象表达式，它给 Band 对象赋值 |
| <i>enum</i>   | 一个常量或数值来指定样式，如设置值所示的 |

设置

enum 的设置值是：

| 常量               | 值 | 描述   |
|------------------|---|--|
| cc3BandNormal    | 0 | (缺省)带区大小可以调整。仅当 CoolBar 控件中含有两个或多个带区时显示调整块 |
| 常量               | 值 | 描述   |
| cc3BandFixedSize | 1 | 带区大小不能调整。不显示调整块                            |

## UseCoolbarColors 属性

设置或返回是否 Band 对象将使用 CoolBar 控件的 ForeColor 和 BackColor 属性。

应用于

Band 对象。

语法

*object*.UseCoolbarColors [= *boolean*]

UseCoolbarColors 属性语法有这些部分:

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个对象表达式, 它给 Band 对象赋值                                 |
| <i>boolean</i> | 一个布尔表达式, 来指定带区是否将覆盖 CoolBar 控件的 ForeColor 和 BackColor |

设置

**boolean** 的设置值是:

| 设置    | 描述  |
|-------|---|
| True  | (缺省)Band 将使用 CoolBar 控件的 ForeColor 和 BackColor 属性显示 |
| False | Band 将使用它自己的 ForeColor 和 BackColor 属性显示             |

## UseCoolbarPicture 属性

设置或返回是否 Band 对象将使用 CoolBar 控件的 Picture 属性。

应用于

Band 对象。

语法

*object*.UseCoolbarPicture [= *boolean*]

UseCoolbarPicture 属性语法有这些部分:

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个对象表达式, 它给 Band 对象赋值                               |
| <i>Boolean</i> | 一个布尔表达式, 它指定带区的 Picture 属性是否将覆盖 CoolBar 控件的 Picture |

## 设置

boolean 的设置值是:

| 设置    | 描述                               |
|-------|----------------------------------|
| True  | (缺省)Band 将显示 CoolBar 控件的 Picture |
| False | Band 将显示它自己的 Picture             |

请参阅

Picture 属性（Coolbar 控件）。

## VariantHeight 属性

返回或设置一个数值来指示 CoolBar 控件是否采用同一高度显示所有的带区。

应用于

CoolBar 控件。

语法

*object*.VariantHeight [= *boolean*]

VariantHeight 属性有这些部分:

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 一个对象表达式，它给 <b>CoolBar</b> 控件赋值 |
| <i>boolean</i> | 一个布尔表达式，指定所有带区是否是同一高度          |

## 设置

**boolean** 的设置值是:

| 设置           | 描述            |
|--------------|---------------|
| <b>True</b>  | (缺省) 带区高度可能变化 |
| <b>False</b> | 带区高度相等        |

## 说明

当该属性被设置成 **False** 时， **CoolBar** 控件将基于全部可见带区最大的 **MinHeight** 属性来显示所有带区。当该属性被设置成 **True** 时，如果每行只有一个带区，每个带区的高度就是基于它自己的 **MinHeight** 属性。如果在一行中有多个带区，这些带区将利用所有在这行中可见带区的最大的 **MinHeight** 属性来显示。

## 请参阅

**MinHeight**, **MinWidth** 属性（**Band** 对象）。



# Data 控件

使用三种类型的 **Recordset** 对象中的任何一种来提供对存储在数据库中数据的访问。**Data** 控件允许从一个记录移动到另一个记录，并显示和操纵来自被连结的控件的记录的数据。如果没有 **Data** 控件或等价的数据源控件，比如 **RemoteData** 控件，窗体上的被连结数据觉察控件不能自动访问数据。

## 语法

### Data

## 说明

可以使用 **Data** 控件来执行大部分数据访问操作，而根本不用编写代码。与 **Data** 控件相连结的数据觉察控件自动显示来自当前记录的一个或多个字段的数据，或者，在某些情况下，显示来自当前记录旁边的一个记录集合中的一个或者多个字段中的数据。**Data** 控件在当前记录上执行所有操作。

如果 **Data** 控件被指示移动到一个不同的记录，则所有被连结的控件自动把当前记录的任何改变传递给 **Data** 控件以保存在数据库中。**Data** 控件移动到被指定的记录，同时把当前记录中的数据传回被连结的控件，并在那里显示。

**Data** 控件自动处理一些意外事件包括空记录集，添加新记录，编辑和更新

现有记录,处理某些类型的错误。然而,在更复杂的应用程序里,则需要捕获 Data 控件不能处理的某些错误类型。例如,当 Microsoft Jet 数据库引擎访问数据文件有问题、没有权限或不能按照代码执行查询时,会导致可捕获的错误。如果错误发生在应用过程开始之前或由于某些内部错误,Error 事件被触发。

## 属性

DataBase 属性, DatabaseName 属性, Exclusive 属性, Options 属性, ReadOnly 属性 (Data Access), RecordSource 属性, Recordset 属性, BOFAction, EOFAction 属性, RecordsetType 属性, DefaultType 属性 (Data 控件), DefaultCursorType 属性 (Data 控件)。

## 方法

UpdateControls 方法, UpdateRecord 方法。

## BOFAction 属性、EOFAction 属性

返回或设置一个值指示在 BOF 或 EOF 属性为 True 时 Data 控件进行什么操作。

## 应用于

ADO Data 控件, Data 控件。

语法

*object*.BOFAction [= *integer*]

*object*.EOFAction [= *integer*]

BOFAction 和 EOFAction 属性的语法具有这些部分：

| 部分             | 描述                     |
|----------------|------------------------|
| <i>Object</i>  | 对象表达式，其值是“应用于”列表中的对象   |
| <i>Integer</i> | 如“设置值”中所示，一个指定某一操作的整数值 |

设置

对于 BOFAction 属性来说，integer 设置值为：

| 设置                   | 值 | 描述   |
|----------------------|---|--|
| VbBOFActionMoveFirst | 0 | MoveFirst（缺省设置）：将第一个记录为当前记录  |
| vbBOFActionBOF       | 1 | BOF：在 Recordset 的开头移动过去将在第一个记录上触发 Data 控件的 Validate 事件，紧接着是非法 (BOF) 记录上的 Reposition 事件。此刻禁止 Data 控件上的 Move Previous 按钮 |

对于 EOFAction 属性来说，integer 设置值为：

| 设置                  | 值 | 描述   |
|---------------------|---|--|
| vbEOFActionMoveLast | 0 | MoveLast（缺省设置）：保持最后一个记录为当前记录   |
| vbEOFActionEOF      | 1 | EOF：在 Recordset 的结尾移过去，这将在最后一个记录上触发 Data 控件的 Validate 事件，紧接着是在非法 (EOF) 记录上的 Reposition 事件。此刻禁止 Data 控件上的 MoveNext 按钮 |
| vbEOFActionAddNew   | 2 | AddNew：移过最后一个记录将在当前记录上触发 Data 控件的 Validate 事件，紧接着是自动的 AddNew，接下来是在新记录上的 Reposition 事件                                |

## 说明

这些常量列在“对象浏览器”中的 Visual Basic (VB) 对象库中。

如果将 EOFAction 属性设置为 vbEOFActionAddNew，一旦用户使用 Data 控件将当前记录指针移动到 EOF，当前记录就被定位于复制缓冲区中的新记录上。此刻可以编辑新添加的记录。如果对新记录做出改变，随后又使用 Data 控件移动当前记录的指针，则该记录被自动追加到 Recordset 中。如果未改变此新记录，并将当前记录定位到另一个记录上，则该新记录被放弃。当定位于这个新记录上时，如果使用 Data 控件定位到另一个记录上，则另一个新记录被创建。

当使用代码来操作 Data 控件创建的 Recordset 时，EOFAction 属性无效。仅当使用鼠标操作 Data 控件时它才生效。

在 Data 控件的 Recordset 不返回记录的情况下，或者在最后一个记录被删除之后，使用 EOFAction 属性的 vbEOFActionAddNew 选项可极大地简化代码，因为新记录作为当前记录时总是可编辑的。如果没有允许这个选项，则可能触发一个“无当前记录”错误。

## 数据类型

Integer

## Database 属性

返回对 Data 控件的基本的 Database 对象的一个引用。

应用于

Connection 对象，Data 控件。

语法

*object.Database*

Set *databaseobject* = *object.Database*（仅用于专业版和企业版）

Database 属性的语法具有这些部分：

| 部分                                     | 描述  |
|--|---|
| <i>databaseobject</i><br><i>object</i> | 计算 Data 控件所创建的合法的 Database 对象的对象表达式<br>对象表达式，其值是“应用于”列表中的对象 |

## 说明

由 Data 控件所创建的 Database 对象是以该控件的 DatabaseName、Exclusive、ReadOnly 和 Connect 属性为基础创建的。

Database 对象具有可以用来管理数据的属性和方法。通过 Data 控件的 Database 属性，可以使用 Database 对象的任何方法，比如 Close 和 Execute 方法。也可以通过使用 Database 的 TableDefs 集合，并依次使用单个 TableDefs 对象的 Fields 和 Indexes 集合的方法来检查 Database 的内部结构。

虽然可以创建一个 Recordset 对象并将它传递给 Data 控件的 Recordset 属性，但不能打开一个数据库并将新创建的 Database 对象传递给 Data 控件的 Database 属性。

## 数据类型

Database

## 示例

这个例子检查数据控件的 Database 属性并在 Debug 窗口中输出每个 Table 的名称。

```

Sub PrintTableNames ()
    Dim Td As TableDef
    ' 设置数据库文件。
    Data1.DatabaseName = "BIBLIO.MDB"
    Data1.Refresh    ' 打开数据库。
    ' 读入并输出数据库中每个表的名称。
    For Each Td in Data1.Database.TableDefs
        Debug.Print Td.Name
    Next
End Sub

```

## DatabaseName 属性

返回或设置 Data 控件的数据源的名称及位置。

应用于

Data 控件。

语法

*object*.DatabaseName [ = *pathname* ]

DatabaseName 属性的语法具有这些部分：

| 部分              | 描述                            |
|-----------------|-------------------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象          |
| <i>pathname</i> | 指示数据库文件的位置或 ODBC 数据源名称的字符串表达式 |

### 说明

如果网络系统支持，则 **pathname** 参数可以是一个完全限定的网络路径名，如 \\Myserver\\Myshare\\Database.mdb。

如下所示，数据库类型由 **pathname** 所指向的文件或目录加以指示：

| <b>pathname</b>              | 指向数据库类型              |
|------------------------------|----------------------|
| .mdb 文件                      | Microsoft Access 数据库 |
| 包含 dbf 文件的目录                 | dBASE 数据库            |
| 包含.xls 文件的目录                 | Microsoft Excel 数据库  |
| 包含.dbf 文件的目录                 | FoxPro 数据库           |
| 包含.wk1、.wk3、.wk4 或.wkS 文件的目录 | Lotus 数据库            |
| 包含 pdx 文件的目录                 | Paradox 数据库          |
| 包含文本格式的数据库文件的目录              | 文本格式数据库              |

对于 ODBC 数据库，比如 SQL Server 和 Oracle，如果控件的 **Connect** 属性标识了一个数据源名称 (DSN)，该数据源名称标识注册表中的某个 ODBC 数据源项目，则此属性可以为空白。



如果在控件的 **Database** 对象打开后改变了 **DatabaseName** 属性，则必须使用 **Refresh** 方法来打开新数据库。

**注意：**为了在访问外部数据库时获得更好的性能，建议将外部数据库表关联到 Microsoft Jet 引擎数据库 (.mdb) 上，并在 **DatabaseName** 属性中使用此 Jet .mdb 数据库的名称。

## 数据类型

**String**

## DataChanged 属性

返回或设置一个值，它指出被绑定的控件中的数据已被某进程改变，这个进程不是从当前记录中检索数据的进程。在设计时不可用。

## 应用于

**Masked** 编辑控件，**DataCombo** 控件，**DataList** 控件，**MonthView** 控件，**RichTextbox** 控件。

## 语法

*object*.DataChanged [= *value*]

**DataChanged** 属性的语法具有这些部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>value</i>  | 如“设置值”中所述，一个用于指示数据是否被改变的布尔表达式 |

## 设置

`value` 的设置值为：

| 设置    | 描述                              |
|-------|---------------------------------|
| True  | 当前在控件中的数据与当前记录中的数据不同            |
| False | （缺省设置）如果当前控件中有数据的话，则与当前记录中的数据相同 |

## 说明

当 `data` 控件从一个记录移动到另一个记录时，它将数据从当前记录中的字段传递到连结在特定字段上或整个记录上的控件。当数据在被绑定的控件中显示时，`DataChanged` 属性设置为 `False`。如果用户或其它操作改变了被绑定的控件中的值，则 `DataChanged` 属性被设置为 `True`。只移动到另一个记录并不影响 `DataChanged` 属性的值。

当 `data` 控件开始移动到一个不同记录时，`Validate` 事件将发生。如果任何被绑定的控件的 `DataChanged` 属性均为 `True`，则 `data` 控件自动地调用 `Edit` 和 `Update` 方法将改变内容发送到数据库中。

如果不希望被绑定的控件将改变的内容保存在数据库中，则可以在 `Validate`

事件中将 **DataChanged** 属性设置为 **False**。

检查控件的 **Change** 事件代码中 **DataChanged** 属性的值，以避免层叠事件发生。将这一点应用到被连结的和未被绑定的控件上。

## 数据类型

**Integer (Boolean)**

## DataField 属性

返回或设置值，将控件绑定到当前记录中的字段上。

应用于

**Masked** 编辑控件，**DataList** 控件，**FlatScrollBar** 控件，**MonthView** 控件，**ProgressBar** 控件，**RichTextBox** 控件。

语法

*object*.DataField [= *value*]

**DataField** 属性语法具有以下几部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的对象   |
| <i>value</i>  | 字符串表达式，其值为 <b>Recordset</b> 对象中字段的一个名称，而对象是由 <b>data</b> 控件的 <b>RecordSource</b> 和 <b>DatabaseName</b> 属性指定的 |

## 说明

被绑定的控件可在数据库中访问特定数据。通常，管理单一字段的被绑定的控件显示当前记录中的特定字段值。被绑定的控件的 **DataSource** 属性指定一个有效的 **data** 控件名称，而 **DataField** 属性则在 **data** 控件所创建的 **Recordset** 对象中指定有效的字段名称。这些属性指定出现在被绑定的控件中的数据。

在使用返回表达式的 **QueryDef** 对象或 **SQL** 语句时，字段名称自动由 **Microsoft Jet** 数据库引擎生成。例如，在 **SQL** 查询中编写 **SQL** 合计函数或表达式时，除非用 **AS** 子句给合计字段起别名，否则自动生成字段名称。通常，表达式字段名称为 **Expr1**，以三个字符数字 **000** 开头。返回的第一个表达式的名称是 **Expr1000**。

推荐您编写 **SQL** 查询别名表达式的列，如下所示：

```
Adodc1.RecordSource = "Select AVG(Sales)  " _
    & " AS AverageSales From SalesTable"
Text1.DataField = "AverageSales"
Adodc1.Refresh
```

**注意：**确保 DataField 属性设置对每个被绑定的控件都有效。如果改变 data 控件的 RecordSource 属性的设置并使用 Refresh，则 Recordset 识别新对象。这可能使被绑定的控件的 DataField 设置值无效，并产生可捕获的错误。

## 数据类型

String

请参阅

DataSource 属性（ActiveX 控件）。

## DataSource 属性

设置一个值，指明控件要通过哪个控件绑定到数据库上。在运行时不可用（DataGrid 控件除外）。

应用于

Masked 编辑控件, MSChart 控件, DataGrid 控件, DataList 控件, MSHFlexGrid 控件, MSFlexGrid 控件, ProgressBar 控件, DataRepeater 控件, RichTextBox 控件。

## 说明

要在运行时将控件绑定到数据库中的字段，你必须在设计时，使用属性窗口在 **DataSource** 属性中指定一个控件。

要完成与控件所管理的 **Recordset** 中的字段的连接，你还必须提供 **DataField** 属性中 **Field** 对象的名字。与 **DataField** 属性不同的是，在运行时 **DataSource** 属性设置不可用（**DataGrid** 控件除外）。

## 数据类型

**String**

## DefaultCursorType 属性（Data 控件）

控制什么类型的鼠标驱动程序被用于 **Data** 控件所创建的连接（仅用于 **ODBCDirect**）。

应用于

**Data** 控件。

语法

*object.DefaultCursorType* [ = *value* ]

DefaultCursorType 属性的语法具有这些部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>Value</i>  | 如“设置值”中所示，一个指定鼠标驱动程序类型的整数常量或值 |

设置

value 的设置值为：

| 设置                    | 值 | 描述   |
|-----------------------|---|--|
| VbUseDefaultCursor    | 0 | 让 ODBC 驱动程序来决定使用哪种光标                           |
| 设置                    | 值 | 描述   |
| VbUseODBCCursor       | 1 | 使用 ODBC 光标库。对于小结果集这个选项将提供更佳的性能，但对较大结果集性能将快速地降低 |
| VbUseServerSideCursor | 2 | 使用服务器端的光标。对于大多数大型操作，这将提供更佳的性能，但可能会导致更多的网络传递活动  |

说明

当在 Data 控件的 DefaultType 属性被设置为 dbUseODBC 时使用此属性。关于更多的信息，请参阅 Workspace 对象的 DefaultCursorDriver 属性。

# 数据类型

Integer

## DefaultType 属性（Data 控件）

返回或设置一个确定 Data 控件所使用的数据源（Jet 或 ODBCDirect）的类型的值。

应用于

Data 控件。

语法

*object.DefaultType* [= *value* ]

DefaultType 属性的语法具有这些部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象       |
| <i>Value</i>  | 如“设置值”中所示，一个指定数据源类型的整数常量或值 |



## 设置

value 的设置值为：

| 设置        | 值 | 描述                                |
|-----------|---|-----------------------------------|
| DbUseODBC | 1 | 使用 ODBCDirect 来访问数据               |
| DbUseJet  | 2 | （缺省设置）使用 Microsoft Jet 数据库引擎来访问数据 |

## 说明

设置 `DefaultType` 属性将告诉 `Data` 控件在创建 `Recordset` 时使用哪种类型的数据源（Jet 或 `ODBCDirect`）。`DefaultType` 属性也确定 `Data` 控件使用的基本的 `Workspace` 对象的类型。除非这个属性被设置为 `dbUseJet`，Jet 数据库引擎将不会被加载。

当将 `DefaultType` 属性设置为 `dbUseODBC` 时，Visual Basic 创建一个新的 `Workspace` 对象并将它添加到 `Workspaces` 集合中。`Data` 控件的 `DefaultType` 属性同 `CreateWorkspace` 方法的 `type` 参数相似。当使用 `dbUseJet` 时，将使用缺省的 `Workspace` 对象。

**注意：**当选择 `dbUseODBC` 作为 `DefaultType` 属性的值时，DAO 通过远程数据对象（RDO）DLL 来进行所有数据访问操作。如果在 `DefaultType` 属性中选定“1 - UseODBC”，则必须在 `Connect` 属性框

中指定 ODBC 连接字符串。为此，选定文本，然后输入 ODBC 连接字符串。有关 ODBC 连接字符串的更多信息，请参阅 Visual Basic 《数据访问指南》。

## 选择数据源

数据访问对象 (DAO) 可以被编程，并以两种方法之一来连接远程 ODBC 数据源：通过 Jet 数据库引擎或通过完全绕过 Jet 的远程数据对象 (RDO)。哪种方法对于特定的应用程序有意义，这要视所需的功能和性能而定。

使用 ODBCDirect：这种方法允许使用 Data 控件通过 RDO 接口而不是通过 ODBC 数据源进行所有的 DAO 操作。也就是说，当建立连接并使用 Data 控件来创建 Recordset 对象时，Jet 数据库引擎不被加载。这还意味着 Jet 引擎所提供的许多 DAO 功能在此 Workspace 中是不可用的。例如，不能执行异种连接，或在不使用附加的 ODBC 驱动程序就访问 mdb 数据库中的 ISAM。不过，当选择 ODBCDirect 时，许多不为 Jet 所普遍支持的 RDO 功能是有用的。

使用 Jet 数据库引擎：除非使 ODBCDirect 有效，否则 Jet 数据库引擎被装载并执行所有本地和远程的数据库操作。一旦创建 Jet Workspace，就可以用它向 ODBCDirect Worspace 传递数据。

## 数据类型

Integer

# Error 事件

仅当没有 Visual Basic 代码在执行时，数据存取错误这样的结果才会出现。

应用于

Data 控件。

语法

```
Private Sub object_Error ([index As Integer,] dataerr As Integer, response As Integer)
```

Error 事件的语法含有下列部分：

| 部分              | 描述                         |
|-----------------|----------------------------|
| <i>object</i>   | 是一个对象表达式，该对象一定能在“应用于”列表中找到 |
| <i>index</i>    | 如果它是在一个控件数组中，则用来标识该控件      |
| <i>dataerr</i>  | 错误编号                       |
| <i>response</i> | 按照设置中所描述的，是一个对应于所要采用响应的编号  |

设置

Response 的设置有：

| 常量                | 值 | 描述            |
|-------------------|---|---------------|
| VbDataErrContinue | 0 | 继续            |
| VbDataErrDisplay  | 1 | (默认值) 显示该错误信息 |

## 说明

这些常量列在对象浏览器的 **Visual Basic (VB)** 对象库中。

通常要在代码中为运行时错误提供错误处理功能。然而没有代码运行时，也会发生运行错误，如当：

- 用户单击 **Data** 控件按钮。
- **Data** 控件要自动打开一个数据库并在 **Form\_Load** 事件后装载一个 **Recordset** 对象。
- 为一个自定义控件执行一些诸如 **MoveNext** 方法，**AddNew** 方法或 **Delete** 方法之类的操作。

如果有一个错误出自上述操作，则产生 **Error** 事件。

如果未对 **Error** 事件编写事件过程，**Visual Basic** 将显示与该错误相关的信息。

出现在 **Form\_Load** 事件之前的错误是不可捕获的，也不会触发 **Error** 事件。例如，在设计时如果将数据控件的属性设置为指向一个不知名的数据库表，就会发生一个不可捕获的错误。

## 示例

本例中,在 `Form_Load` 事件完成后,如果由 `Data` 控件的 `DatabaseName` 属性所指定的数据库未找到,就显示一个“打开”对话框的示例。

```
Private Sub Data1_Error(DataError As Integer, Response As Integer)
    Select Case DataError
        ' 如果数据库文件未找到。
        Case 3024
            ' 显示一个“打开”对话框。
            CommonDialog1.ShowOpen
        ...
    End Select
End Sub
```

## Exclusive 属性

返回或设置一个值,指出 `Data` 控件的基本数据库是为单用户打开还是为多用户打开。

应用于

`Data` 控件。

# 语法

*object.Exclusive* [= *value*]

Exclusive 属性的语法具有这些部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象。    |
| <i>value</i>  | 如“设置值”中所示，确定用户访问方式的布尔表达式 |

# 设置

value 的设置值为：

| 设置    | 描述   |
|-------|--|
| True  | 数据库为单用户访问而打开。在它关闭前其它用户不能打开该数据库               |
| False | （缺省设置）数据库为多用户访问而打开。其它用户可以打开该数据库，并可以在它打开时访问数据 |

# 说明

此属性值连同 DatabaseName、ReadOnly 和 Connect 属性一起用于打开数据库。在专业版和企业版中，此属性同 OpenDatabase 方法中的排他性参数一致。

**Exclusive** 属性仅在打开 **Database** 时使用。如果在运行时改变此属性的值，则必须使用 **Refresh** 方法来使之生效。如果其它某个用户已经打开了此数据库，则将不能排它的打开数据库，并且会导致一个可捕获的错误。

如果为排它的使用而数据库打开，则数据库操作会快一些。

在排它打开数据库后，应用程序可以打开所需的多个实例。不过，正在系统上运行的其它应用程序将不得打开此数据库。

**Exclusive** 属性会因通过 **ODBC** 访问数据库而被忽视。

## 数据类型

**Boolean**

## IntegralHeight 属性

返回或设置一个值，指示控件是否显示部分项目。在运行时只读。

应用于

**DataList** 控件。

语法

*object*.IntegralHeight [= *value* ]

IntegralHeight 属性的语法具有这些部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>value</i>  | 如“设置值”中所示，一个确定列表是否改变过大小的布尔表达式 |

## 设置

value 的设置值为：

| 设置    | 描述                         |
|-------|----------------------------|
| True  | （缺省设置）该列表调整自身大小来显示完整的项目    |
| False | 即使项目太高以至无法完整显示，该列表也不调整自身大小 |

## 说明

如果列表中的项目数超出能显示的范围，控件中会自动添加一个滚动条。您可以通过设置 IntegralHeight 属性为 True 来避免显示部分行。

## 数据类型

Boolean

ODBCDirect

一种允许通过 RDO 并以使用绕过 Microsoft Jet 数据库引擎的 DAO 功能的方法来访问 ODBC 数据源的技术。



当 ODBC 数据源以此方式被引用时，它被称为 "ODBCDirect" 数据源。这是为了将其区别于称为“Jet 连接 ODBC 数据源”，它来自于通过 Jet 数据库引擎直接连接的 ODBC 数据源。在访问数据源中所使用的技术决定了哪些 DAO 对象、方法和属性可以被使用。

## Options 属性

返回或设置一个值，说明控件的 Recordset 属性中一个或多个 Recordset 对象特性。

应用于

Data 控件。

语法

*object.Options* [ = *value* ]

Options 属性的语法具有这些部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>value</i>  | 如“设置值”中所示，一个指定 Recordset 特点的常量或值 |

# 设置

使用下列值中的一或多个来设置 Options 属性。如果使用多个选项，则必须将它们的值相加。

| 常量                   | 值   | 描述  |
|----------------------|-----|---|
| dbDenyWrite          | 1   | 在多用户环境中，其它用户不能改变 Recordset 的记录  |
| dbDenyRead           | 2   | 在多用户环境中，其它用户不能读取记录（仅用于表类型 Recordset）  |
| dbReadOnly           | 4   | 不能改变 Recordset 中的记录   |
| dbAppendOnly         | 8   | 可以在 Recordset 中添加新记录，但不能读取现有的记录   |
| dbInconsistent       | 16  | 更新可以应用到 Recordset 的所有字段上，即使它们违反了联结条件  |
| dbConsistent         | 32  | （缺省设置）更新仅应用到那些不违反联结条件的字段上   |
| DbSQLPassThro<br>ugh | 64  | 当使用 RecordSource 属性中使用 SQL 语句的 Data 控件时，将此 SQL 语句发送给 ODBC 数据库，比如 SQL Server 或 Oracle 数据库来进行处理 |
| 常量                   | 值   | 描述  |
| dbForwardOnly        | 256 | Recordset 对象只支持向前的滚动。唯一允许的移动方法是 MoveNext。此选项不能用在以 Data 控件操作的 Recordset 对象上                    |

续表

|              |     |                                  |
|--------------|-----|----------------------------------|
| dbSeeChanges | 512 | 如果另一个用户正在改变您正在编辑的数据，则会产生一个可捕获的错误 |
|--------------|-----|----------------------------------|

## 说明

这些常量列在 **Object Browser** 中的 **Visual Basic (VB)** 对象库中。

如果在运行时改变 **Options** 属性，则必须使用 **Refresh** 方法来使之生效。在专业版和企业版中，此属性同 **OpenRecordset** 方法中的 **options** 参数一致。

要为此属性设置多个值，可以通过将值相加的方法来组合选项。例如，要设置 **dbAppendOnly** 和 **dbInconsistent**，可以使用下述代码：

```
Data1.Options = dbAppendOnly + dbInconsistent
```

要确定该属性是否包含某特定值，可以使用 **And** 运算符。例如，要查明 **Recordset** 是否为只读访问而打开，可以使用这条代码：

```
If Data1.Options And dbReadOnly Then...
```

使用 **dbInconsistent** 和 **dbConsistent** 会导致一致的更新，即 **Recordset** 对象的缺省方式。

**注意：** 仅在创建 **dynaset** 类型或快照类型的 **Recordset** 对象之时才使用 **dbSQLPassThrough** 选项，支持该选项是为了提供对先前版本的兼容性。为了获得更好的性能和功能，应使用以前创建的 **SQL**

PassThrough QueryDef 对象，并将 Data 控件的 Recordset 属性设置为一个用 QueryDef 创建的 Recordset 对象。

**注意：**如果试图访问一个包含相同列的 SQL Server 6.0 表，则会触发一个 3622 错误。要防止此问题发生，请使用 Options 属性的 dbSeeChanges 选项或 OpenRecordset 方法。

## 数据类型

### Integer

## ReadOnly 属性（数据访问）

返回或设置一个值，确定控件的 Database 是否为只读访问而打开。

应用于

Data 控件。

语法

*object.ReadOnly* [= *boolean*]

ReadOnly 属性的语法具有这些部分：

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象    |
| <i>boolean</i> | 如“设置值”中所示，确定读/写访问的布尔表达式 |

## 设置

value 的设置值是：

| 设置    | 描述                                 |
|-------|------------------------------------|
| True  | 控件的 Database 对象的打开方式为只读。不允许对数据进行修改 |
| False | （缺省设置）控件的 Database 对象的打开方式为读/写方式   |

## 说明

使用 Data 控件的 ReadOnly 属性来指定基本的 Database 中的数据是否可以修改。例如，可以创建一个仅显示数据的应用程序。使用只读结果集访问 Database 运行起来也会更快。

对 Data 控件来说，此属性仅在应用程序第一次打开数据库时才使用。如果应用程序随后又打开数据库的其它实例，则该属性被忽视。为了使此属性中的改变生效，必须关闭数据库的所有实例，然后再使用 Refresh 方法。

在专业版和企业版中，此属性同 OpenDatabase 方法中的 readonly 参数相一致。

# 数据类型

Boolean

## Recordset 属性

返回或设置由 Data 控件的属性或由现有的 Recordset 对象所定义的 Recordset 对象。

应用于

MSHFlexGrid 控件。

语法

Set *object*.Recordset [= *value* ]]

Recordset 属性的语法具有这些部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象   |
| <i>value</i>  | 一个包含 Recordset 对象的对象变量 |

说明

当应用程序在初始的 Form\_Load 过程之前启动时，Data 控件被自动地初

始化。如果 **Connect**、**DatabaseName**、**Options**、**RecordSource**、**Exclusive**、**ReadOnly** 和 **RecordsetType** 属性是合法的，或者在运行时设置这些 **Data** 控件属性并使用 **Refresh** 方法，则 Microsoft Jet 数据库引擎试图创建一个新的基于那些属性的 **Recordset** 对象。此 **Recordset** 对象可通过 **Data** 控件的 **Recordset** 属性访问。不过，如果在设计时错误地设置若干个这些属性，则当 Visual Basic 试图使用该属性来打开特定的数据库并创建 **Recordset** 对象时，将产生一个不可捕获的错误。

可以像使用其它 **Recordset** 对象一样使用 **Recordset** 属性。例如，可以使用任何 **Recordset** 方法或属性，并检查 **Recordset** 对象的基本模式

也可以通过设置 **Data** 控件的 **RecordsetType** 属性而设置创建的 **Recordset** 的类型。如果不请求特定类型，则创建一个 **dynaset** 类型的 **Recordset**。使用 **RecordsetType** 属性可以请求创建表类型、快照类型或者 **dynaset** 类型的 **Recordset**。但是如果 Jet 引擎不能创建所请求的类型，则将产生一个可捕获的错误。

在许多情况下，所创建的 **Recordset** 对象的缺省类型和配置是极为低效的。也就是说，可能不需要可以更新的、可以全部滚动的、**keyset** 类型的光标来访问数据。例如，一个只读的快照类型的 **Recordset** 可能会比缺省的 **Recordset** 创建起来快得多。要确保尽可能选择最有效的 **Type**、**Exclusive**、**Options** 和 **ReadOnly** 属性。

可以在运行时通过检查 **Recordset** 属性的 **Type** 属性或 **Data** 控件的 **RecordsetType** 属性的方法来检查创建的 **Recordset** 的类型。但是请注意，用于

创建的 Recordset 的类型的常量是不同的。例如：

```
If Data1.Recordset.Type = dbOpenDynaset Then ...
```

```
If Data1.RecordsetType = dbDynasetType Then ...
```

即使请求创建一个 dynaset 类型或表类型的 Recordset，Recordset 也可能是不可更新的。如果基本的数据库、表或字段是不可更新的，则所有或部分 Recordset 可能是只读的。检查 Database 和 Recordset 对象的 Updatable 属性或 Field 对象的 DataUpdatable 属性来确定程序代码是否可以改变记录。即使 DataUpdatable 属性返回 True，在某些情况下，基本的数据字段也不能更新，例如，没有足够的权限做出改变。其它因素也能阻止字段被更新。

Recordset 所返回的记录个数可以通过在 Recordset 中移动到最后一个记录并检查 Recordset 对象的 RecordCount 属性的方法来确定。在移动到最后一个记录之前，RecordCount 属性所返回的值只反映 Jet 引擎处理过的行数。下面的例子说明可以怎样将 Recordset 的 RecordCount 属性和 Recordset 属性结合起来显示 Data 控件的记录集中的记录的个数：

```
Data1.Recordset.MoveLast
```

```
MsgBox "Records: " & Data1.Recordset.RecordCount
```

专业版和企业版

如果使用代码或另一个 Data 控件创建 Recordset 对象，则可以将 Data 控件的 Recordset 属性设置为这个新的记录集。当新的 Recordset 被赋值给



Recordset 属性时,Data 控件中任何现有的 Recordset,以及与之相关的 Database 对象都被释放。

注意: 当 Recordset 属性被设置时, Data 控件不关闭当前的 Recordset 或 Database,但它的确释放了它。如果没有其它用户,则该数据库被自动关闭。可能希望在使用 Close 方法设置 Recordset 属性之前关闭同 Data 相关的 Recordset 和 Database。

要确保连接到 Data 控件上的被绑定控件的 DataField 属性的新的 Recordset 对象的字段名相匹配。

例如,用代码创建一个 Recordset 并将它传递到现有的 Data 控件:

```
Dim Db As Database, Rs As Recordset '定义为公用变量
```

```
Sub ApplyRecordset()
```

```
    Set Db = Workspaces(0).OpenDatabase("BIBLIO.MDB")
```

```
    Set Rs = Db.OpenRecordset("AUTHORS") 'Table 对象的缺省值
```

```
    Set Data1.Recordset = Rs '为 Recordset 赋值
```

```
    Data1.Recordset.Index = "PrimaryKey"
```

```
    Debug.print Rs.Type '显示创建的类型
```

```
End Sub
```

可以使用这种技术来创建一个在 MDI 父窗体上的单个的隐藏 Data 控件同另一个在 MDI 子窗体上的可见的 Data 控件的 MDI 父子数据连接。在 MDI 子窗体的 Form\_Load 事件中,将子 Data 控件的 Recordset 属性设置为父 Data 控件的 Recordset 属性。使用此技术使所有子窗体与其父窗体的被绑

定的控件同步。

**注意：**Data 控件不支持只向前的 Recordset 对象。如果试图将一个只能向前的 Recordset 对象赋值给 Data 控件的 Recordset 属性，则将产生一个可捕获的错误。

除 ODBCDirect (DefaultType = dbUseODBC)Recordset 对象外，Data 控件所创建的所有 Recordset 对象都是在 Workspaces(0) 中创建的。如果需要使 Data 控件来操作另一个 Workspace 中的数据库，请使用上述技术在所需的 Workspace 中打开数据库，创建新 Recordset 并将 Data 控件的 Recordset 属性设置为这个新 Recordset。

**重点：**总可以通过使用 Recordset 属性来引用 Data 控件的 Recordset 的属性。通过直接引用 Recordset，可以确定与 Table 对象一起使用的 Index, QueryDef 的 Parameters 集合或者 Recordset 的类型。

## 数据类型

### Recordset

## RecordsetType 属性

返回或设置一个值，指出要 Data 控件创建的 Recordset 对象的类型。

应用于  
Data 控件。

语法

*object.RecordsetType [= value ]*

RecordsetType 属性的语法具有这些部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>value</i>  | 如“设置值”中所示，一个指定 Recordset 类型的常量或值 |

设置

value 的设置值为：

| 设置               | 值 | 描述                            |
|------------------|---|-------------------------------|
| vbRSTypeTable    | 0 | 一个表类型 Recordset               |
| vbRSTypeDynaset  | 1 | （缺省设置）一个 dynaset 类型 Recordset |
| vbRSTypeSnapshot | 2 | 一个快照类型 Recordset.             |

说明

如果 Microsoft Jet 数据库引擎不能创建所请求的 Recordset 类型，则将产

生一个可捕获的错误。

如果在 Data 控件创建 Recordset 前没有指定 RecordsetType，则一个 dynaset 类型的 Recordset 将被创建（如果可能的话）。

如果不使用 Data 控件（即使是另一个 Data 控件）创建 Recordset，并且用这个新 Recordset 对象来设置 Recordset 属性，则 Data 控件的 RecordsetType 属性被设置为新 Recordset 的 RecordsetType 属性。

**重点：**RecordsetType 属性值同用于标识 Recordset 对象类型不一致。关于细节，请参阅 OpenRecordset 方法或 Type 属性。

在许多情况下，创建的 Recordset 的缺省类型和配置是极为低效的。也就是说，可能不需要可以更新的、可以全部滚动的、keyset 类型的光标来访问数据。例如，一个只读的、只能向前的、快照类型的 Recordset 可能会比缺省的光标创建起来快得多。请保证尽可能为 RecordsetType, Exclusive, Options 和 ReadOnly 属性选择最有效的设置值。请保证视状况可能选择最有效的 Type、Exclusive、Options 和 ReadOnly 属性。

## 数据类型

Integer

## 示例

这个例子使用 Data 控件来创建一个 Recordset 控件，并检查 Data 控件的 RecordsetType 属性以决定所创建的记录集的类型。

```
Sub DisplayRecordsetType()  
' 表示希望得到的 Recordset 类型。  
Data1.RecordsetType = vbRSTypeDynaset  
Data1.DatabaseName = "BIBLIO.MDB"  
Data1.RecordSource = "Authors"  
Data1.Refresh  
  
Select Case Data1.RecordsetType  
    Case vbRSTypeTable  
        Debug.print "Table-type Recordset created."  
    Case vbRSTypeDynaset  
        Debug.print "Dynaset-type Recordset created."  
    Case vbRSTypeSnapshot  
        Debug.print "Snapshot-type Recordset created."  
End Select  
End Sub
```

## RecordSource 属性

返回或设置 Data 控件的基本表、SQL 语句。

应用于  
Data 控件。

语法

```
object.RecordSource [= value ]
```

RecordSource 属性的语法具有这些部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象  |
| <i>value</i>  | 如“设置值”中所示，指定名称的字符串表达式 |

设置

value 的设置值为：

| 设置       | 描述   |
|----------|--|
| 表名称      | 在 Database 对象的 TableDefs 集合中定义的一张表的名称                        |
| SQL 查询   | 符合数据源的语法的合法 SQL 字符串。   |
| QueryDef | Database 对象的 QueryDefs 集合中一个 QueryDef 对象的名字—<br>当访问 Jet 数据库时 |

说明

RecordSource 属性指定通过窗体上的被绑定的控件访问的记录来源。

如果将 **RecordSource** 属性设置为数据库中的一个存在的表的名称，则那张表中的所有字段对附属于 **Data** 控件的被绑定的控件都是可见的。对于表类型的记录集 (**RecordsetType** = **vbRSTypeTable**)，被检索的记录顺序由 **Recordset** 的 **Index** 属性选择的 **Index** 对象来决定。对于 **dynaset** 类型和快照类型的 **Recordset** 对象，可以通过在 **Data** 控件的 **Recordset** 属性中使用带有 **Order By** 子句的 **SQL** 语句的方法来对记录排序。否则，不以特别的顺序返回数据。

如果将 **RecordSource** 顺序设置为数据库中一个存在的 **QueryDef** 的名称，则由 **QueryDef** 返回的所有字段对附属于 **Data** 控件的被绑定的控件都是可见的。被检索的记录顺序由 **QueryDef** 对象的查询设置。例如，**QueryDef** 可以包含一个 **ORDER BY** 子句来改变由 **Recordset** 返回的由 **Data** 控件创建的记录的顺序，或者包含一个 **WHERE** 子句来过滤记录。如果 **QueryDef** 不指明顺序，将不以特别的顺序返回数据。

注意：在设计时，显示在 **RecordSource** 属性的“属性”窗口中的 **QueryDef** 对象被过滤，从而只显示 **Data** 控件可使用的 **QueryDef** 对象。具有参数的 **QueryDef** 对象，以及具有如下类型的 **QueryDef** 对象不被显示：**dbQAction**、**dbQCrosstab**、**dbQSQLPassThrough** 和 **dbQSetOperation**。

如果将 **RecordSource** 属性设置为一个返回记录的 **SQL** 语句，则该 **SQL** 查询返回的所有字段对附属于 **Data** 控件的被绑定的控件都是可见的。此语句可以包含一个 **ORDER BY** 子句来改变由 **Data** 控件创建的 **Recordset** 返回的记录顺序，或者包含一个 **WHERE** 子句来过滤记录。如果在 **Database** 和

**Connect** 属性中指定的数据库不是 **Microsoft Jet** 引擎数据库，而且 **dbSQLPassThrough** 选项在 **Options** 属性中被设置，则 **SQL** 查询必须使用该数据库引擎所要求的语法。

**注意：**每当 **QueryDef** 或 **SQL** 语句从一个表达式返回一个值时，该表达式的字段名称由 **Microsoft Jet** 数据库引擎自动地创建。一般地，该名称为 **Expr1** 后跟一个从 000 开始的三个字符的数字。例如，第一个表达式将被命名为：**Expr1000**。

在大多数情况下，需要为表达式指定别名，从而知道绑定到被绑定的控件上的列的名称。关于更多的信息，请参阅 **SQL SELECT** 语句的 **AS** 子句。

当在运行时改变 **RecordSource** 属性的值后，必须使用 **Refresh** 方法使改变生效，并重建 **Recordset**。

在运行时，如果 **Recordset** 指定了一个非法的 **Table** 名称、**QueryDef** 名称，或者包含了非法的 **SQL** 语法，则产生一个可捕获的错误。如果此错误在初始的 **Form\_Load** 过程中发生，则该错误是不可捕获的。

**注意：**请保证每个被绑定的控件都对其 **DataField** 属性具有合法设置。如果改变了 **Data** 控件的 **RecordSource** 属性的设置并随后使用 **Refresh**，则 **Recordset** 标识新对象。这可能使被绑定的控件的 **DataField** 设置失效，并导致一个可捕获的错误。



# 数据类型

String

## Reposition 事件

在一条记录成为当前记录之后出现。

应用于

Data 控件。

语法

Private Sub *object*.Reposition ([*index* As Integer])

Reposition 事件有下列部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 是一个对象表达式，其值是“应用于”列表中的对象 |
| <i>index</i>  | 如果它在一个控件数组中，则可用于识别该控件   |

## 说明

当加载一个 Data 控件时，Recordset 对象中的第一条记录成为当前记录，会发生 Reposition 事件。无论何时只要用户单击 Data 控件上某个按钮，进行记录间的移动，或者使用了某个 Move 方法（如 MoveNext），Find 方法（如 FindFirst），或任何其它改变当前记录的属性或方法，在每条记录成为当前记录以后，均会发生 Reposition 事件。

相反，Validate 事件则是在移动到一条不同记录之前出现。

使用这种事件可以进行基于当前记录中数据的计算，或者改变窗体来响应当前记录中的数据。

## 示例

本例使用 Reposition 事件更新一个标题列表，它们来自 Biblio.mdb 示例数据库中的 Publisher。首先在一个窗体上放一个 DataGrid，一个 TextBox 和两个 Data 控件。将两个 Data 控件的 DatabaseName 属性设置为 Biblio.mdb 示例数据库。将 Data1 的 RecordSource 属性设置为 Publisher。将 Text1 的 DataSource 属性设置为 Data1，DataField 属性设置为 Name。将 DataGrid1 的 DataSource 属性设置为 Data2。添加下列代码：

```
Private Sub Data1_Reposition()  
    '选择由 Data1 中的  
    '当前记录所发布的全部标题  
    Data2.RecordSource = "Select * from Titles where PubID = " &
```

```
Data1.Recordset("PubID")
```

```
    Data2.Refresh ' Rebuild the recordset
```

```
End Sub
```

用 Data1 在 Publishers 记录集中移动时 DataGrid 控件中的标题列表自动更新。

## UpdateControls 方法

从一个 Data 控件的 Recordset 对象中取得当前记录，并且在连结某个 Data 控件的控件中显示适当的数据。不支持已命名参数。

应用于

Data 控件。

语法

*object*.UpdateControls

**object** 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

用这种方法将被连结控件的内容恢复为其原始值，等效于用户更改了数据之后决定取消更改。

这种方法等效于使当前记录又成为当前记录，除非无事件发生。

`UpdateControls` 方法终止任何挂起的 `Edit` 或 `AddNew` 操作。

## UpdateRecord 方法

保存被连结控件的当前值。不支持已命名的参数。

### 语法

*object*.UpdateRecord

**Object** 置换元代表对象表达式，其值是“应用于”列表中的对象。

### 说明

用这种方法在 **Validate** 事件期间将被连结的控件的当前内容保存到数据库中而不再次触发 **Validate** 事件。使用这种方法可避免创建一个层叠事件。

**UpdateRecord** 方法与执行 **Edit** 方法改变一个字段并随后执行 **Update** 方法产生同样效果，除非无事件发生。

可用这种方法以避免触发 **Validate** 事件。

不论何时想更新数据库中某条记录，所有有效性检查规则在记录被写到数据库之前都必须得到满足。这些规则可通过设置 **ValidationRule** 属性来建立；或者，在 Microsoft SQL Server 中，通过 Transact SQL 缺省值、规则和编写触

发器来确保参照完整性和数据完整性。

某些情况下，因为该操作违反了引用完整性的约束条件，包含记录的页被锁定数据库或 **Recordset** 对象不能更新或用户没有执行操作的许可权可能不发生更新。这些条件中任何一条都将产生一个可以捕获的错误。

## Validate 事件（Data 控件）

在一条不同的记录成为当前记录之前，Update 方法之前（用 UpdateRecord 方法保存数据时除外）；以及 Delete、Unload 或 Close 操作之前会发生该事件。

应用于

ImageCombo 控件。

语法

```
Private Sub object_Validate ([ index As Integer,] action As Integer, save As Integer)
```

Validate 事件的语法有下列部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 是一个对象表达式，其值是“应用于”列表中的对象          |
| <i>index</i>  | 如果它在一个控件数组中，则可用于识别该控件            |
| <i>action</i> | 如设置中所描述的，是一个整数，用来指示引发这种事件的操作     |
| <i>save</i>   | 如设置中所描述的，是一个布尔表达式，用来指定被连结的数据是否改变 |

## 设置

action 的设置为：

| 常量                       | 值 | 描述                         |
|--------------------------|---|----------------------------|
| vbDataActionCancel       | 0 | 当 Sub 退出时取消操作              |
| vbDataActionMoveFirst    | 1 | MoveFirst 方法               |
| vbDataActionMovePrevious | 2 | MovePrevious 方法            |
| vbDataActionMoveNext     | 3 | MoveNext 方法                |
| vbDataActionMoveLast     | 4 | MoveLast 方法                |
| 常量                       | 值 | 描述                         |
| vbDataActionAddNew       | 5 | AddNew 方法                  |
| vbDataActionUpdate       | 6 | Update 操作（不是 UpdateRecord） |
| vbDataActionDelete       | 7 | Delete 方法                  |
| vbDataActionFind         | 8 | Find 方法                    |

续表

|                      |    |                 |
|----------------------|----|-----------------|
| vbDataActionBookmark | 9  | Bookmark 属性已被设置 |
| vbDataActionClose    | 10 | Close 的方法       |
| vbDataActionUnload   | 11 | 窗体正在卸载          |

save 的设置为:

| 设置    | 描述        |
|-------|-----------|
| True  | 被连结数据已被改变 |
| False | 被连结数据未被改变 |

### 说明

这些常量列在对象浏览器的 Visual Basic (VB) 对象库中。

save 参数初始时指出被连结的数据是否已经改变。如果复制缓冲区中的数据改变,这个参数仍旧可为 False。如果该事件退出时 save 为 True,则激活 Edit 和 UpdateRecord 方法。UpdateRecord 方法只保存那些来自被绑定的控件或 DataChanged 属性设置为 True 的复制缓冲区的数据。

即使被连结的控件中的数据未改变或者不存在被连结的控件,这一事件也会发生。用这一事件可以改变值和更新数据。也可以选择保存数据或停止任何引起事件发生的操作并代之以一种不同的操作。

可以改变 action 的参数把一个操作转换为另一个。可以改变各种 Move 方法和 AddNew 方法,它们能自由地交换 (Move 到 AddNew, Move 到其它

Move, 或 AddNew 到 Move)。当使用 AddNew 时, 可以用 MoveNext 然后执行另一个 AddNew, 来检查 EditMode 属性, 以确定是一个 Edit 还是 AddNew 操作正在进行中。将 AddNew 或 Moves 之一改变为任何其它操作的试图或者忽略, 或者导致一个可以捕获的错误。任何操作都可通过将操作设为 0 而终止。

在这个事件的代码中, 可以检查每个 DataChanged 为 True 的被连接控件中的数据。然后将 DataChanged 设为 False 以避免将这个数据存入数据库。

在这个事件中不能使用带下划线的 Recordset 对象上的任何方法 (如 MoveNext)。

## 示例

这个例子说明了简单的数据合法性检查。在 Biblio.mdb 数据库的 Authors 表中有两个字段: Au\_ID 和 Author。由于 Au\_ID 中的值用于唯一地标识作者, 所以这个值不应被修改。这个例子不允许对与 Text1 连结的 Au\_ID 字段做任何修改。

```
Private Sub Data1_Validate (Action As Integer, Save As Integer)
```

```
    If Text1.DataChanged Then ' 检查数据是否被修改。
```

```
        MsgBox "You can't change the ID number."
```

```
        Text1.DataChanged = False ' 不保存修改过的数据。
```

```
    End If
```



...

End

# DataGrid 控件

显示并允许对 **Recordset** 对象中代表记录和字段的一系列行和列进行数据操纵。

## 语法

### DataGrid

## 说明

该数据识别的 **DataGrid** 控件看起来与 **Grid** 控件类似。但是，您可以设置 **DataGrid** 控件的 **DataSource** 属性为一个 **Data** 控件，以自动填充该控件并且从 **Data** 控件的 **Recordset** 对象自动设置其列标头。这个 **DataGrid** 控件实际上是一个固定的列集合，每一列的行数都是不确定的。

**DataGrid** 控件的每一个单元格都可以包含文本值，但不能链接或内嵌对象。可以在代码中指定当前单元格，或者用户可以使用鼠标或箭头键在运行时改变它。通过在单元格中键入或编程的方式，单元格可以交互地编辑。单元格能够被单独地选定或按照行来选定。

如果一个单元格的文本太长，以致于不能在单元格中全部显示，则文本将在同一单元格内折行到下一行。要显示折行的文本，必须增加单元格的 **Column**

对象的 **Width** 属性和 / 或 **DataGrid** 控件的 **RowHeight** 属性。在设计时，可以通过调节列来交互地改变列宽度，或在 **Column** 对象的属性页中改变列宽度。

使用 **DataGrid** 控件的 **Columns** 集合的 **Count** 属性和 **Recordset** 对象的 **RecordCount** 属性，可以决定控件中行和列的数目。**DataGrid** 控件的可包含的行数取决于系统的资源，而列数最多可达 32,767 列。

选择一个单元格，则 **ColIndex** 属性被设置，也就是选择了 **DataGrid** 对象的 **Columns** 集合中的一个 **Column** 对象。**Column** 对象的 **Text** 和 **Value** 属性引用当前单元格的内容。使用 **Bookmark** 属性能够访问当前行的数据，它能够对下一级 **Recordset** 对象中记录的访问。**DataGrid** 控件中的每一列都有自己的字体、边框、自动换行和另外一些与其他列无关的能够被设置的属性。在设计时，您可以设置列宽和行高，并且建立对用户不可见的列。您还能阻止用户在运行时改变格式。

**注意：**如果您在设计时设置了任何一个 **DataGrid** 列属性，就必须设置它的所有属性以保持当前的设置值。

**注意：**如果使用 **Move** 方法定位 **DataGrid** 控件，就必须使用 **Refresh** 方法强迫控件重画。

除了不能支持解除绑定模式以外，**DataGrid** 控件的功能与 **DBGrid** 控件类似。

**注意：**该控件支持 Unicode。当使用一个例如 Microsoft Windows NT 这样的支持 Unicode 的系统时，控件传递 Unicode 数据而无需转换。但

是，在其他系统中，数据则被从 ANSI 转换到 Unicode 并转换回来。详细信息请参阅 Programmer's Guide 中的 "ANSI, DBCS, and Unicode: Definitions"。

## 属性

DataGrid, Bookmark 属性 (DataGrid), AddNewMode 属性, AllowArrows 属性, ApproxCount 属性, CurrentCellModified 属性, CurrentCellVisible 属性, AllowAddNew 属性, AllowDelete 属性, AllowRowSizing 属性, AllowUpdate 属性, ColumnHeaders 属性, DataChanged 属性 (DBGrid), BoundText 属性, Column 属性, DataFormat 属性, DataBindings 属性, HelpContextID 属性, Name 属性, Parent 属性, Font 属性, Container 属性, Object 属性, ToolTipText 属性, Text 属性 (ActiveX 控件), RightToLeft 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Caption 属性 (ActiveX 控件)。

## 事件

Validate 事件, DblClick 事件。

## 方法

AboutBox 方法, Refresh 方法, SetFocus 方法。

请参阅

Data 窗体, Wizard-窗体, DataGrid 控件常量, Column 对象, Data 控件, 使用 ADO Data 控件, 使用 DataGrid 控件, 与数据类模块一起使用 DataGrid 控件, ANSI,DBCS 和 Unicode: 定义。

## Add 方法 (Columns、SelBookmarks、Splits 集合)

把一个新列添加到 Columns 集合, 或把一个新书签添加到 DataGrid 控件的 SelBookmarks 集合或把一个新拆分添加到 Splits 集合。不支持命名的参数。

应用于

Splits 集合, Columns 集合, SelBookmarks 集合, DataGrid 控件常量, DataGrid 控件。

语法

*object.Add colindex*

*object.Add bookmark*

Add 方法的语法包含下列部分：

| 部分              | 描述  |
|-----------------|---|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象  |
| <i>colindex</i> | 必需的。如“设置值”中所示。是一个整数，用于指定新的 Column 对象或 Split 对象被插入到 Column 集合或 Split 集合的什么地方 |
| <i>bookmark</i> | 将被添加到集合里的书签   |

## 设置

colindex 列索引的设置值为：

| 设置值   | 描述  |
|-------|---|
| 0     | 插入新列作为最左边的列                                 |
| Count | 如果 colindex 参数与 Count 属性设置相同，则新列作为最右边的列插入   |
| n     | 把新列插入到 Columns 集合中第 n 列的左边。第 n 列及所有后续的列相应增加 |

## 说明

Add 方法根据 colindex 参数把一个新 Column 插入到 Columns 集合中。添加的新列的 Visible 属性设置为 False，其它属性设置为缺省值。初始时，由于 DataField 属性被设置为一个“0 长度字符串”(""), 故新列未被连结。Columns 集合的 Count 属性增加以反映这一新的列。

**重点：**如果以前使用 Remove 方法删除了一个列，在增加了新的列后，就可能需要用 Rebind 和 Refresh 方法刷新显示。这就命令 DataGrid 控件重新生成其内部列的布局矩阵，以正确反映控件的真实状态。

使用 Add 方法把书签添加到 SelBookmarks 集合。当一个书签添加到 SelBookmarks 集合后，它即在 DataGrid 控件中被选中。

请参阅

Remove 方法（DBGrid），ColIndex 属性，Data 控件。

## AddNewMode 属性

返回一个值,描述当前单元关于网格的 AddNew 行的位置。运行时只读，设计时不可用。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.AddNewMode

AddNewMode 属性语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

值

AddNewMode 属性返回以下一个值：

| 常量               | 值 | 描述   |
|------------------|---|--|
| DbgNoAddNew      | 0 | 当前单元不在最后一行，没有 AddNew 操作，被挂起  |
| DbgAddNewCurrent | 1 | 当前单元在最后一行，但没有 AddNew 操作，被挂起  |
| DbgAddNewPending | 2 | 当前单元的下一行是最后一行，作为挂起 AddNew 操作的结果，通过网格的用户界面可以让用户进行初始化，或通过代码把 Value 或 Text 的列属性作为设置结果 |

说明

如果 AllowAddNew 属性为 True ，那么在网格中显示的最后一行是空的，允许用户输入新记录。如果 AllowAddNew 属性为 False, 空白行将不显示，并且 AddNewMode 总是返回 0。



请参阅

Value 属性（Column 对象）， AllowAddNew 属性， Text 属性。

## AfterColEdit 事件

在完成网格单元中的编辑之后出现。

应用于

DataGrid 控件常量， DataGrid 控件。

语法

```
Private Sub object_AfterColEdit([ index As Integer,] ByVal colindex As Integer)
```

AfterColEdit 事件的语法包含下面部分：

| 部分              | 描述                    |
|-----------------|-----------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象  |
| <i>index</i>    | Integer，它标识控件数组中的一个控件 |
| <i>colindex</i> | Integer，它标识已编辑过的列     |

说明

当完成网格单元中的编辑时，比如当按 Tab 键而切换到同一行的另外一列

时按了 ENTER 键,或者单击了另外一个单元时,都将执行 BeforeColUpdate 和 AfterColUpdate 事件,并且可将单元的数据移动到网格的复制缓冲区中。AfterColEdit 事件紧随 AfterColEdit 事件出现。

在网格单元中完成时,即使未对单元进行改动,或是已经取消了 BeforeColUpdate 事件,也会触发该事件。

如果取消 BeforeColEdit 事件,就不会启动 AfterColEdit 事件。

请参阅

BeforeColEdit 事件, AfterColUpdate 事件, BeforeColUpdate 事件。

## AfterColUpdate 事件

在数据从 DataGrid 控件中的一个单元格移动到该控件的复制缓冲区后被触发。

应用于

DataGrid 控件常量, DataGrid 控件。

语法

```
Private Sub object_AfterColUpdate ([index As Integer,] colindex As Integer)
```

AfterColUpdate 事件的语法包括下列部分：

| 部分              | 描述                     |
|-----------------|------------------------|
| <i>object</i>   | 对象表达式，其值为“应用于”列表中的一个对象 |
| <i>index</i>    | 整数，用来标识在控件数组中的一个控件     |
| <i>colindex</i> | 是用来标识控件中列的整数           |

## 说明

当用户完成了在一个 DataGrid 控件单元中的编辑，如在同一行中转到另一列，按 ENTER 键，或者控件失去了焦点时，BeforeColUpdate 事件将被执行，除非被取消，否则该单元中的数据将被移到控件的复制缓冲区中。一旦移动，就执行 AfterColUpdate 事件。

AfterColUpdate 事件发生在 BeforeColUpdate 事件之后，而且仅当 BeforeColUpdate 事件中的 cancel 变元未设置为 True 时。

一旦 AfterColUpdate 事件过程开始，单元数据就已经被移到控件的复制缓冲区，而且不能被取消，但在数据提交给 Recordset 之前，可以发生其它更新。

## 请参阅

BeforeUpdate 事件，ColIndex 属性。

## 示例

本例用于演示在一个列被更新时并把结果放到另一列去。

```
Private Sub DataGrid1_AfterColUpdate (ColIndex As Integer)
    If ColIndex = 1 Then
        Data1.Recordset.FindFirst "PubId = " _
            & DataGrid1.Columns(1).Value
        If Not Data1.Recordset.NoMatch Then
            DataGrid1.Columns(2).Value = _
                Data1.Recordset.Fields("Publisher")
        Else
            DataGrid1.Columns(2).Value = "No Match"
        End If
    End If
End Sub
```

## AfterDelete 事件

当用户在 **DataGrid** 控件中删除一条选定的记录后被触发。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

# 语法

```
Private Sub object_AfterDelete ([index As Integer,] colindex As Integer)
```

AfterDelete 事件语法包括下列部分

| 部分              | 描述                     |
|-----------------|------------------------|
| <i>Object</i>   | 对象表达式，其值为“应用于”列表中的一个对象 |
| <i>Index</i>    | 整数，用来标识一个在控件数组中的控件     |
| <i>Colindex</i> | 是用来标识列的整数              |

# 说明

当用户在 **DataGrid** 控件中选择一个记录选择器，并且按了 **DEL** 键或 **CTRL+X** 键时，所选行被删除。在记录被删除之前， **BeforeDelete** 事件被触发。该选择行被删除后， **AfterDelete** 事件被触发。被选择删除的行可在由 **SelBookmarks** 属性所指向的集合中获得。

请参阅

**BeforeColUpdate** 事件， **BeforeDelete** 事件， **SelBookmarks** 属性。

# 示例

本例显示一条信息，确认一条记录被成功地删除。

```
Private Sub DataGrid1_AfterDelete ()
```

```
MsgBox "Record has successfully been deleted!"
```

```
End Sub
```

## AfterInsert 事件

在用户往 **DataGrid** 控件中插入一条新记录后被触发。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

```
Private Sub object_AfterInsert (index As Integer)
```

**AfterInsert** 事件语法包括下列部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的一个对象 |
| <i>index</i>  | 整数，标识在控件数组中的控件         |

说明

当用户选择一条新记录（在控件的底部）并且在其中一个单元中输入一个字符时，触发 **BeforeInsert** 事件，接着 **BeforeUpdate**，**AfterUpdate** 和 **AfterInsert**

事件被触发。

当 **AfterInsert** 事件被触发时，该记录已被添加到数据库中。**Bookmark** 属性可用于存取新记录。

**AfterInsert event** 不能被取消。

**AfterInsert** 事件过程可用于更新其它的表，或进行其它控件事后更新的清理工作。

请参阅

**AfterUpdate** 事件，**BeforeInsert** 事件，**BeforeUpdate** 事件。

示例

如果用户在网格中向某列输入一个值，本例在一个相关表中创建一个条目。

```
Private Sub DataGrid1_AfterInsert ()  
    If DataGrid1.Columns(1).Value <> "" Then  
        Data2.Recordset.AddNew  
        Data2.Recordset.Fields("PubId") = DataGrid1.Columns(1).Value  
        Data2.Recordset.Update  
    End If  
End Sub
```

## AfterUpdate 事件

修改过的数据已经从 DataGrid 控件中被写到数据库后被触发。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

Sub *object*\_AfterUpdate (*index* As Integer)

AfterUpdate 事件语法包括下列部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>Object</i> | 对象表达式，其值为“应用于”列表中的一个对象 |
| <i>Index</i>  | 整数，用来标识在控件数组中的控件       |

说明

当用户移动到另一行，或执行 Recordset 对象的 Update 方法时，数据从控件的复制缓冲区移动到 Data 控件的复制缓冲区，并被写到数据库中。一旦完成写操作，就触发 AfterUpdate 事件。

通过使用 DataGrid 控件的 Bookmark 属性，被更新过的记录可以用。



**AfterUpdate** 事件发生在 **BeforeUpdate** 事件之后，但是在控件的 **LostFocus**（或者按 **Tab** 键次序的下一个控件的 **GotFocus**）事件之前。该事件以被连结和不被连结两种方式发生，并且不能被取消。

与 **Change** 事件不同的是，用代码在控件或记录中改变数据不会触发该事件。

请参阅

**BeforeColUpdate** 事件，**BeforeInsert** 事件，**BeforeUpdate** 事件。

示例

本例是在网格中发生了任何变化时更新一个标签。

```
Private Sub DataGrid1_AfterUpdate ()
```

```
    Label1.Caption = "Last modified: " & Format$(Now, "Long Date")
```

```
End Sub
```

## AllowAddNew 属性

返回或设置一个值，指出用户是否能够向与 **DataGrid** 控件连接的 **Recordset** 对象中添加新记录。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.AllowAddNew [= value]*

AllowAddNew 属性的语法具有这些部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>Value</i>  | 如“设置值”中所示，是一个确定用户是否能够添加新记录的布尔表达式 |

设置

value 的设置值为：

| 设置    | 描述                                      |
|-------|---|
| True  | 用户可以向与 DataGrid 控件连接的 Recordset 对象中添加记录 |
| False | 用户不能向与 DataGrid 控件连接的 Recordset 对象中添加记录 |

说明

如果 AllowAddnew 属性为 True，则在 DataGrid 控件中显示的最后一行

被留作空白以允许用户输入新记录。如果 **AllowAddNew** 属性为 **False**，则无空白行显示，用户无法定位进行输入。

即使 **AllowAddNew** 属性为 **True**，**Recordset** 也可能不允许插入。在此情况下，若用户试图添加记录就会产生错误提示。

请参阅

**AllowUpdate** 属性，**Data** 控件，**Recordset** 属性。

示例

下面的例子检查复选框的值。如果它是 **False**，用户就不能改变网格。

```
Private Sub Form_Load ()  
    If Check1.Value = 0 Then  
        DataGrid1.AllowDelete = False  
        DataGrid1.AllowAddNew = False  
        DataGrid1.AllowUpdate = False  
    End If  
End Sub
```

返回或设置一个值，指明用户是否可以给 **DataGrid** 控件底层的 **Recordset** 对象添加新的记录。

## AllowArrows 属性

设置或返回一个值，该值决定控件是否用箭头键对网格定位。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.AllowArrows [= value]*

AllowArrows 属性的语法包含下面部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象              |
| <i>Value</i>  | 布尔表达式，它决定箭头键是否用来对网格定位，参见“设置值”中的描述 |

设置

value 的设置值为：

| 设置    | 描述                                      |
|-------|---|
| True  | （缺省值）可用箭头键在同一行中从一个单元移动到另一个单元            |
| False | 左箭头键和右箭头键将把焦点从一个控件移动到另一个控件，但不能用来在单元之间移动 |

## 说明

当把该属性设置为 **True** 时，不能用箭头键移动到 **DataGrid** 控件之外。如果把 **WrapCellPointer** 属性也设置为 **True**，则箭头键会在行的边缘上自动换行，并且可用箭头键对整个网格定位。

## 请参阅

**WrapCellPointer** 属性。

## AllowDelete 属性

返回或设置一个值，指出用户能否从与 **DataGrid** 控件连接的 **Recordset** 对象中删除记录。

## 应用于

**DataGrid** 控件常量，**DataGrid** 控件。

# 语法

*object.AllowDelete [= value]*

AllowDelete 属性的语法具有这些部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象         |
| <i>value</i>  | 如“设置值”中所示，一个确定用户能否删除记录的布尔表达式 |

# 设置

value 的设置值为：

| 设置    | 描述                                      |
|-------|---|
| True  | 用户可以从与 DataGrid 控件连接的 Recordset 对象中删除记录 |
| False | 用户不能从与 DataGrid 控件连接的 Recordset 对象中删除记录 |

# 说明

使用 AllowDelete 属性防止用户在 Recordset 中通过与 DataGrid 控件的交互过程来删除记录。

对于 DataGrid 控件，即使 AllowDelete 属性为 True，Recordset 也可能不允许删除。在此情况下，若用户试图删除记录就会产生错误。

注意：从 DataGrid 控件中删除一个记录之后，您应该对其使用

Refresh 方法，强迫 DataGrid 更新。否则，尽管从下层记录集中删除了一个记录，记录将仍然在 DataGrid 中显示。

请参阅

AllowUpdate 属性，Data 控件，SpliE 对象。

## AllowFocus 属性

设置或返回一个值，该值判定拆分中的单元是否能接收焦点。

应用于

DataGrid 控件常量，DataGrid 控件，Split 对象。

语法

*object*.AllowFocus [= *value*]

AllowFocus 属性的语法包含下面部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>value</i>  | 布尔表达式，它决定单元是否接收焦点，参见“设置值”中的描述 |

## 设置

value 的设置值为：

| 设置    | 描述   |
|-------|--|
| True  | （缺省值）当拆分得到焦点，就能以交互作用方式来选定这个拆分                |
| False | 不能以交互作用方式选定拆分。单击时，拆分不接收焦点，而原先具有焦点的控件将继续保持有焦点 |

## 说明

将该属性与 **AllowSizing** 属性组合使用时可完全禁止对拆分进行修改（为此把两个属性都设置为 **False**）。当把 **TabAcrossSplits** 设置为 **True** 时将忽略那些不能选定的拆分。

## 请参阅

**TabAcrossSplits** 属性，**AllowSizing** 属性。

## AllowRowSizing 属性

返回或设置一个值，指示用户能否在运行时重置 **DataGrid** 控件的行或 **Split** 对象的大小。



应用于

Split 对象，DataGrid 控件常量，DataGrid 控件。

语法

*object.AllowRowSizing [= value]*

AllowRowSizing 属性的语法具有这些部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>value</i>  | 如“设置”中所示，一个确定用户能否重置行的大小的布尔表达式 |

设置

value 的设置值为：

| 设置    | 描述          |
|-------|-------------|
| True  | 行的大小可以被用户改变 |
| False | 行的大小不能被用户改变 |

说明

如果 AllowRowSizing 属性为 True，则当鼠标指针被定位在记录选定器之

间的分界线上时变为双向箭头（大小为 **N S**），且用户可以通过拖动重置行的大小。行的大小的任何改变都将引发一个 **RowResize** 事件。

**DataGrid** 控件的所有行总是具有相同的高度，这一高度是由 **RowHeight** 属性确定。

注意：即使 **AllowRowSizing** 属性为 **False**，行的高度也仍可用 **RowHeight** 属性可编程地改变。

请参阅

**RowResize** 事件，**DefColWidth** 属性，**VisibleRows** 属性，**Data** 控件。

示例

这个例子检查数据库是否具有备注字段，如果没有，则禁止重置行的大小。

```
Sub CheckForMemoField()
```

```
    Dim Fld As Field
```

```
    DataGrid1.AllowRowSizing = False
```

```
    For Each Fld in Data1.Recordset.Fields
```

```
        If Fld.Type = dbMemo Then
```

```
            DataGrid1.AllowRowSizing = True
```

```
            DataGrid1.RowHeight = DataGrid1.RowHeight * 2
```

```
            Exit For
```

```
        End If
```

Next

End Sub

## AllowSizing 属性

返回或设置一个值，指出用户是否能在运行期间将 **DataGrid** 控件中的列或拆分重置大小。

应用于

Split 对象，Column 对象，DataGrid 控件常量，DataGrid 控件。

语法

*object.AllowSizing* [= *value*]

AllowSizing 属性的语法具有这些部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象            |
| <i>value</i>  | 如“设置”所示，布尔表达式，它确定一个列或拆分是否能够重置大小 |

设置

*value* 的设置值为：

| 设置    | 描述                         |
|-------|----------------------------|
| True  | （对 Column 为缺省）用户可以重置列的大小   |
| False | （对 Split 为缺省）用户不能重置列或拆分的大小 |

### 说明

如果 AllowSizing 属性为 True，则当鼠标指针位于特定列的行的分界线时，它会变成双箭头（大小为 W E），用户可以通过拖曳重置列的大小。列大小的任何改变都会引发一个 ColResize 事件。

如果对最左边的拆分， AllowSizing 设置成 True，则当鼠标指针在拆分大小框中（左下角）时，它将转向一对箭头向下的竖线，用户可拖动鼠标创建新的拆分。创建新的拆分将会引起 SplitChange 事件。

如果对任何其它拆分， AllowSizing 都设置成 True，则当鼠标指针在拆分大小框中（左下角）时，它将转向一对具有双箭头的竖线，用户可拖动鼠标调整拆分的大小。在这种情况下不会引起任何事件（标准鼠标事件除外）。

### 请参阅

ColResize 事件， RowResize 事件， AllowRowSizing 属性， DefColWidth 属性， VisibleRows 属性， Data 控件。

### 示例

这个示例防止用户将网格的前三列重置大小或进行编辑。

```
Private Sub Form_Load ()  
    Dim I  
    For I = 0 to 2  
        DataGrid1.Columns(I).AllowSizing = False  
        DataGrid1.Columns(I).Locked = True  
    Next I  
End Sub
```

## AllowUpdate 属性

返回或设置一个值，指示用户能否修改 **DataGrid** 控件中的数据。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object.AllowUpdate [= value]*

**AllowUpdate** 属性的语法具有这些部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象       |
| <i>value</i>  | 如“设置值”中所示，确定用户能否改变数据的布尔表达式 |

## 设置

value 的设置值为：

| 设置    | 描述                     |
|-------|------------------------|
| True  | 用户可以修改 DataGrid 控件中的数据 |
| False | 用户不能修改 DataGrid 控件中的数据 |

## 说明

当 AllowUpdate 属性为 False 时，用户仍然可以通过 DataGrid 控件进行滚动并选择数据，但不能改变任何值；忽视任何改变网格中数据的操作。

也可以使用 Column 对象属性使 DataGrid 控件的单个列成为只读的，但 AllowUpdate 属性设置优先于列设置值（不改变列设置值）。

**注意：**即使 AllowUpdate 对 DataGrid 控件为 True，Recordset 对象也可能不允许更新；在此情况下，当用户试图改变记录时会发生一个可捕获的错误。

## 请参阅

AllowAddNew 属性， AllowDelete 属性， Data 控件。

## ApproxCount 属性

返回在网格中的行号的近似值。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.ApproxCount

Object 置换元表示对象表达式，其值是“应用于”列表中的对象。

说明

此属性返回网格用于校准垂直滚动条的近似行数。

一般来说，**ApproxCount** 属性用于改善垂直滚动条的精确性。这在行号已知的情况下非常有用，例如当网格与数组联合使用时。

**注意：**获得 ApproxCount 属性将查询下一级数据源。

## BeforeColEdit 事件

仅在键入字符而进入编辑模式之前出现该事件。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

```
Private Sub object_BeforeColEdit([index As Integer,] ByVal colindex As Integer,  
ByVal keyascii As Integer, cancel As Integer)
```

BeforeColEdit 事件的语法包含下面部分：

| 部分               | 描述  |
|------------------|---|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的对象  |
| 部分               | 描述  |
| <i>index</i>     | Integer，它标识控件数组中的一个控件   |
| <i>colindex</i>  | Integer，它标识已编辑过的那一列   |
| <i>keyasciik</i> | Integer，对于用户为启动编辑而键入的字符，它表示这些字符的 ANSI 键代码。如果用户通过单击鼠标来启动编辑，则它取值为 0。KeyAscii 按值而不是地址来传递；所以不能通过改变其值，使得可用不同字符启动编辑 |
| <i>cancel</i>    | Integer，可将其设置为 True，以防止用户编辑该单元，参见“设置值”中的描述  |



# 设置

cancel 的设置值为：

| 设置    | 描述   |
|-------|--|
| True  | 单元不进入编辑模式  |
| False | (缺省值) ColEdit 事件立即被引发，如果 KeyAscii 参数不为零，则 Change 和 KeyUp 事件将随后出现 |

# 说明

如果没有使用浮动编辑器字幕，那么，当用户单击当前单元或双击另一个单元时也会出现此事件。

可用该事件控制每个单元的可编辑性，或将最初的击键值转换成缺省值。

**注意：**如果没有使用浮动编辑器字幕，则 keyascii 参数只能为 0。

# 请参阅

AfterColEdit 事件，ColEdit 事件。

## BeforeColUpdate 事件

在一个单元内的编辑完成之后而数据从单元移到 DataGrid 控件的复制缓冲

区之前被触发。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

```
Private Sub object_BeforeColUpdate ([ index As Integer,] colindex As Integer,
oldvalue As Variant, cancel As Integer)
```

BeforeColUpdate 事件语法包含下列部分：

| 部分              | 描述                          |
|-----------------|-----------------------------|
| <i>object</i>   | 对象表达式 ， 其值为“应用于”列表中的一个对象    |
| <i>index</i>    | 整数，用来标识在控件数组中的控件            |
| <i>colindex</i> | 是用来标识列的整数                   |
| <i>oldvalue</i> | 包含着单元格中变化之前值的一个值            |
| <i>cancel</i>   | 如设置中所描述的，是一个指出是否有变化发生的布尔表达式 |

设置

对 *cancel* 的设置为

| 设置    | 描述  |
|-------|---|
| True  | 取消改变，单元恢复为 <code>oldvalue</code> ， 焦点也恢复到控件 |
| False | （缺省）持续变化并允许焦点的变化                            |

## 说明

当用户在一个单元内编辑完成后，如在一行中转到另一列，按 **ENTER** 键，或者把焦点从单元中改过来时，`oldvalue` 参数指定的数据也从单元移到控件的复制缓冲区。在数据从单元移到控件的复制缓冲区之前，触发 **BeforeColUpdate** 事件。这个事件提供一个把网格单元提交给控件的复制缓冲区之前有检查各个网格单元的机会。

如果事件过程将 `cancel` 参数设置为 **True** ， 则恢复单元中以前的值，把焦点保留在控件上，且不触发 **AfterColUpdate** 事件。

要恢复单元中 `oldvalue` 并且允许用户把焦点移出单元，将 `cancel` 设置为 **False** ， 并按如下方式将单元设置为原来的值：

```
Cancel = False
DataGrid1.Columns(ColIndex).Value = OldValue
```

**AfterColUpdate** 事件发生在 **BeforeColUpdate** 事件之后。

当把 `cancel` 参数设置为 **True** 时，，除非应用程序确定数据能够安全地移回控件的复制缓冲区，否则用户不可以从控件移出焦点。

请参阅

AfterColUpdate 事件, AfterUpdate 事件, BeforeInsert 事件, BeforeUpdate 事件, ColIndex 属性。

示例

本例检查并确保用户键入的值在一定范围之内, 否则禁止更新。

```
Private Sub DataGrid1.BeforeColUpdate (ColIndex As Long, OldValue As Variant, Cancel  
As Integer)  
    If ColIndex = 1 Then  
        If DataGrid1.Columns(1).Value < Now Then  
            Cancel = True  
            MsgBox "You must enter a date that is later than today."  
        End If  
    End If  
End Sub
```

**BeforeDelete 事件**

发生在 DataGrid 控件中选定的记录被删除之前。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

Private Sub *object*\_BeforeDelete ([*index* As Integer,] *cancel* As Integer)

BeforeDelete 事件语法含有下列部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 对象表达式 ， 其值为“应用于” 列表中的一个对象          |
| <i>index</i>  | 整数，用来标识一个在控件数组中的控件                 |
| <i>cancel</i> | 如设置值中所描述的是一个布尔表达式 ， 它指出有一条记录是否被删除了 |

设置

对 *cancel* 的设置为：

| 设置    | 描述                 |
|-------|--------------------|
| True  | 保持焦点在该控件，不删除记录     |
| False | （缺省）持续删除操作，使焦点能够改变 |

## 说明

当用户选择控件中的一个记录选择器并按了 **DEL** 键或 **CTL+X** 键时，在选定的行被删除之前，触发 **BeforeDelete** 事件。

一旦行被删除，就触发 **AfterDelete** 事件。被选定删除的行可由 **SelBookmarks** 属性提供的集合中获得。

如果事件过程将 **cancel** 参数设置为 **True**，则该行不被删除。

如果选择多于一行，则显示多行不能删除的错误信息。

## 请参阅

**AfterColUpdate** 事件，**AfterUpdate** 事件，**BeforeInsert** 事件，**BeforeUpdate** 事件。

## 示例

本例显示一条信息，要求用户确认从网格中的删除。

```
Private Sub DataGrid1_BeforeDelete (Cancel As Integer)
    Dim mResult As Integer
    mResult = MsgBox("Are you sure that you want to delete " & DataGrid1.SeletedRows &
" record?", _
        vbYesNo And vbQuestion, "Delete Confirmation")
    If mResult = vbNo Then Cancel = True
```

End Sub

## BeforeInsert 事件

出现在往一个 DataGrid 控件中插入一条记录之前。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

Private Sub *object*\_BeforeInsert ([ *index* As Integer,] *cancel* As Integer)

BeforeInsert 事件语法包括下列部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的一个对象       |
| <i>index</i>  | 整数，用来标识一个在控件数组中的控件           |
| <i>cancel</i> | 如设置值中所描述的是一个布尔表达式 指出是否添加一条记录 |

设置

对 *cancel* 的设置为：

| 设置    | 描述               |
|-------|------------------|
| True  | 保持焦点在该控件，不添加新记录  |
| False | （缺省）持续复制并使焦点能够改变 |

### 说明

当用户选择一条新记录（在 DataGrid 控件的底部）并且在其中一个单元中输入一个字符时，触发 BeforeInsert 事件，接着 BeforeUpdate, AfterUpdate 和 AfterInsert 事件被触发。

如果事件过程将 cancel 参数设置为 True ，该行不被插入，单元被清空。

当 BeforeInsert 事件被触发时，该记录还未添加到数据库中。新记录存在于 DataGrid 控件的复制缓冲区中，直到事件过程结束。

AfterInsert 事件完成后， DataGrid 控件中的新记录行被重新初始化，被编辑的记录成为 DataGrid 控件的最后一行。

### 请参阅

AfterDelete 事件， AfterInsert 事件， AfterUpdate 事件， BeforeUpdate 事件。

### 示例

本例显示一条信息，要求用户确认网格中一条新记录的添加。

```
Private Sub DataGrid1_BeforeInsert (Cancel As Integer)
    Dim mResult As Integer
```



```
mResult = MsgBox("Confirm: Add a new record?", _  
    vbYesNo And vbQuestion, "Confirmation")  
If mResult = vbNo Then Cancel = True  
End Sub
```

## BeforeUpdate 事件

发生在数据从 DataGrid 控件移动到控件的复制缓冲区之前。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

```
Private Sub object_BeforeUpdate ([index As Integer,] cancel As Integer)
```

BeforeUpdate 事件语法包括下列部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的一个对象     |
| <i>index</i>  | 整数，用来标识一个在控件数组中的控件         |
| <i>cancel</i> | 如设置值中所描述的是一个布尔表达式，指出是否复制数据 |

## 设置

cancel 的设置为：

| 设置    | 描述                  |
|-------|---------------------|
| True  | 保持焦点在该控件，不复制数据      |
| False | （缺省） 持续复制操作并使焦点能够改变 |

## 说明

当用户移动到另一行或者执行 **Recordset** 对象的 **Update** 方法时，数据将从 **DataGrid** 控件的复制缓冲区移动到 **Data** 控件的复制缓冲区，并且写到数据库里。

在数据从 **DataGrid** 控件的复制缓冲区移到 **Data** 控件的复制缓冲区之前，触发 **BeforeUpdate** 事件。除非复制操作被取消，在数据被移回至 **Data** 控件的复制缓冲区并写到数据库以后，**AfterUpdate** 事件被触发。更新的记录可通过使用 **DataGrid** 控件的 **Bookmark** 属性获取。

如果将 **BeforeUpdate** 事件的 **cancel** 参数设置为 **True**，焦点会保持在控件上，既不触发 **AfterUpdate** 事件也不触发 **LostFocus** 事件，记录也不存入数据库。

**BeforeUpdate** 事件发生在该控件的 **AfterUpdate** 和 **LostFocus** 事件之前，或发生在按 **Tab** 键次序的下一个控件的 **GotFocus** 事件之前。

即使该控件未被连结该事件也会发生。

与 **Change** 事件不同的是，使用代码改变控件或记录中的数据时不触发本事件。

在允许用户将变化提交 **Data** 控件的复制缓冲区之前，可以用本事件使一个被绑定的控件记录中的数据有效。通过将 **cancel** 参数设置为 **True**，用户不能将焦点移离控件，除非应用程序确定数据能够安全地移回 **Data** 的复制缓冲区。

请参阅

**AfterUpdate** 事件，**DBCombo** 控件，**DBList** 控件。

示例

本例显示一条信息，告诉用户在网格可以被更新之前，在第一列输入一个值。

```
Private Sub DataGrid1_BeforeUpdate (Cancel As Integer)
    If DataGrid1.Columns(1).Value = "" Then
        MsgBox "You must enter value in the first column!"
        Cancel = True
    End If
End Sub
```

## Bookmark 属性 (DataGrid)

返回或设置非绑定 DataGrid 控件中 RowBuffer 对象内部指定行的书签。

应用于

DataGrid 控件。

语法

*object*.Bookmark (*row*) [= *value*]

Bookmark 的属性语法由下列部分组成：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                 |
| <i>row</i>    | 指定放置书签的数据行的整数值。该值的范围为 0 到 RowCount-1 |
| <i>value</i>  | 表示指定 <i>row</i> 的书签的变体               |

说明

使用 Bookmark 属性返回的值来保存对当前行的引用，甚至在另一行变成当前行时，原当前行仍有效。

当您在代码中将 Bookmark 属性设置成有效值时，与该值相联结的行将变

成当前行，而网格则调整其显示，使得在必要时得以看到新的当前行。

将 **Bookmark** 属性定义成一个变体，以适合用户定义的未绑定模式下的书签。

在 **UnboundReadData** 事件中，可能有多个数据行，所以必须给每一行提供一个书签。

**UnboundWriteData** 事件传送一个书签，用来标识待更新的数据行。

**UnboundAddData** 事件传送一个书签，用来标识要加入的数据行。

注意：在未绑定模式下将 **Bookmark** 属性设置成自身，这样将强行通过 **UnboundWriteData** 事件更新当前行。

## 示例

这个例子中，当用户删除非绑定 **DataGrid** 控件的一行数据时，**UnboundDeleteRow** 事件被触发，允许用户从数据集中手工删除该行，在本例中数据库是一个简单数组。下面的代码片断说明，如何在 **UnboundDeleteRow** 事件中将书签作为参数来传递，标识要删除的行。

```
Private Sub DataGrid1_UnboundDeleteRow(Bookmark As Variant)
```

```
    For i% = Bookmark + 1 To RowCount - 1
```

```
        For j% = 0 to MAXCOLS - 1
```

```
            UserData(j%, i% - 1) = UserData(j%, i%)
```

```
        Next j%
```

Next I%

End Sub

**Bookmark** 属性赋值的例子，请参考 **UnboundReadData** 事件示例。

## Button 属性（Column 对象）

设置或返回一个值，该值决定是否在当前单元中显示按钮。

应用于

**DataGrid** 控件常量，**DataGrid** 控件，**Column** 对象。

语法

*object*.**Button** [= *value*]

**Button** 属性的语法包含下面部分：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                 |
| <i>value</i>  | 布尔表达式，该表达式决定是否在当前单元中显示按钮，参见“设置值”中的描述 |

# 设置

value 的设置值为：

| 设置值   | 描述                |
|-------|-------------------|
| True  | 运行时按钮将显示在当前单元的右上角 |
| 设置值   | 描述                |
| False | （缺省值）不显示按钮        |

# 说明

一般来说，当想下拉控件（如内置的组合框、绑定列表框，甚至另一个 DataGrid 控件）来进行编辑或数据录入时应使列按钮有效。在单击当前单元中的按钮时将触发 ButtonClick 事件。然后就可编写代码，从该单元下拉想要的那个控件。

请参阅

ButtonClick 事件（DataGrid 控件）。

## ButtonClick 事件（DataGrid 控件）

在单击当前单元的内置按钮时出现该事件。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

Private Sub *object*\_ButtonClick([ *index* As Integer,] ByVal *colindex* As Integer)

ButtonClick 事件的语法包含下面部分：

| 部分              | 描述                    |
|-----------------|-----------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象  |
| <i>index</i>    | Integer，它标识控件数组中的一个控件 |
| <i>colindex</i> | Integer，它标识其按钮被单击了的列  |

说明

当把内置按钮的 Button 属性设置为 True 时，内置按钮对列有效。

一般来说，在想放下一个 Visual Basic 控件（如内置的组合框、被绑定的列表框，甚至另一个 DataGrid 控件）来进行编辑或数据录入时，应启动该列按钮。单击当前单元中的按钮时将会引发 ButtonClick 事件。然后可编写代码，从该单元放下所需控件。

请参阅

Button 属性（Column 对象）。



## Caption 属性（DataGrid 控件、 Column 对象）

对于 DataGrid 控件，该属性决定网格顶部标题栏中显示的文本。

对于 Column 对象，该属性决定在该列的标头区中显示的文本。

应用于

DataGrid 控件常量，DataGrid 控件，Column 对象。

语法

*object.Caption* [= *value*]

Caption 属性的语法包含下面部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>value</i>  | 字符串表达式，决定要显示的内容，参见下面的描述 |

说明

对于 DataGrid 控件，把 Caption 属性设置为空字符串，可隐藏其标题栏。

对于 Column 对象，把 Caption 属性设置为空字符串，可清除该列头标区中的文本，但不会隐藏标头。如果 DataGrid 把控件的 ColumnHeaders 属性设置为 True，而未把 HeadLines 属性设置为 0，则仅显示列标题。

请参阅

ColumnHeaders 属性， HeadLines 属性。

## CaptureImage 方法

返回网格在当前状态下显示的被捕获图像。

应用于

DataGrid 控件常量， DataGrid 控件。

语法

*object*.CaptureImage

*object* 置换元表示对象表达式，其值是“应用于”列表中的对象。

说明

可用 CaptureImage 方法来获得网格的快照。

下列代码用 CaptureImage 方法将 DataGrid 控件的快照赋值给 PictureBox 控件。

```
Picture1.Picture = DataGrid1.CaptureImage
```

注意：CaptureImage 方法获得的是元文件类型的图像。所以应将图像的大小调整到其容器的大小。

请参阅

PictureBox 控件。

## CellText 方法

从一个 DataGridView 控件单元格返回一个格式化文本值。不支持命名的参数。

应用于

Column 对象，DataGridView 控件常量，DataGridView 控件。

语法

*object.CellText bookmark*

CellText 方法的语法包含下列部分：

| 部分              | 描述                                 |
|-----------------|------------------------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象               |
| <i>bookmark</i> | 必需的。字符串表达式，它代表 DataGridView 控件中的一行 |

## 说明

对 **bookmark** 值所指定的行，**CellText** 方法返回一个代表当前列中数据的格式化字符串。除了可以选择一个指定行并从中检索值外，使用 **CellText** 方法同访问 **Text** 属性很相似。

**CellText** 方法的返回值是通过按照 **Column** 对象的 **NumberFormat** 属性所指定的格式对 **Text** 属性进行格式化的结果。

使用 **CellText** 方法时，用 **Columns** 集合指定 **DataGrid** 控件的特定列，并设置 **bookmark** 参数为一个特定行。

利用 **CellText** 方法从一个单元提取信息不会影响当前选择。

## 请参阅

**Column** 对象，**CellValue** 方法，**Add** 方法（**Columns**，**SelBookmarks**，**Splits** 集合），**ColIndex** 属性，**NumberFormat** 属性，**Data** 控件。

## 示例

这个例子从顶行和底行取得信息，并将其显示在一个标签里。

```
Sub DataGrid1_Scroll (Cancel As Integer)
```

```
    Dim TopRow, BottomRow
```

```
    TopRow = DataGrid1.Columns(1).CellText(DataGrid1.FirstRow)
```

```
    BottomRow = DataGrid1.Columns(1).CellText(DataGrid1.RowBookmark _
```

```
(DataGrid1.VisibleRows - 1))  
Label1.Caption = "Records " & TopRow & " to " & _  
    BottomRow & " are currently displayed."  
End Sub
```

## CellValue 方法

对一个在 **DataGrid** 控件中指定的行，返回其中某列的原始数据。不支持命名的参数。

应用于

**Column** 对象，**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.CellValue *bookmark*

CellValue 方法的语法包含下列部分：

| 部分              | 描述  |
|-----------------|---|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象                                |
| <i>bookmark</i> | 必需的。字符串表达式，它包含存储在一个选定的 <b>DataGrid</b> 控件单元中的未格式化数据 |

## 说明

当使用 `CellValue` 方法时，使用 `Columns` 集合指定 `DataGrid` 控件的特定列，并将 `bookmark` 书签参数设置为一个特定行。

除非可以在 `DataGrid` 控件中指定一个特定行作为参照，使用 `CellValue` 方法返回当前 `Column` 对象的 `Value` 属性设定值。

利用 `CellValue` 方法从一个单元格提取信息不会影响当前选择。

## 请参阅

`ColIndex` 属性，`NumberFormat` 属性，`Data` 控件。

## 示例

这个例子从选定行的范围中，检索某个给定列中所有的值，并把它们装入一个数组中。

```
Sub Command1_Click ()
```

```
    Dim I
```

```
    ReDim CalcArray (0 to DataGrid1.SelBookmarks.Count - 1)
```

```
    For I = 0 to DataGrid1.SelBookmarks.Count - 1
```

```
        '在选定的行数组中，把当前行的值
```

```
        '放入相应的 CalcArray 单元格。
```

```
        CalcArray(I) = _
```

```
            DataGrid1.Columns(1).CellValue(DataGrid1.SelBookmarks(I))
```

Next I

End Sub

## ClearFields 方法

恢复缺省的网格布局。

应用于

DatGrid 控件常量，DataGrid 控件。

语法

*object*.ClearFields

**object** 置换元表示对象表达式，其值是“应用于”列表中的对象。

说明

ClearFields 方法将恢复缺省的网格布局（具有两个空白的列），使后面的 ReBind 操作自动从（可能已改变的）数据源导出新列的绑定。可以调用 HoldFields 方法来取消网格的自动布局行为。

请参阅

HoldFields 方法，Rebind 方法。

## ClearSelCols 方法

撤消对拆分中所有列所作的选择。如果未选择列，则不做任何事情。

应用于

DataGrid 控件常量，DataGrid 控件，Split 对象。

语法

*object*.ClearSelCols

**object** 置换元表示对象表达式，其值是“应用于”列表中的对象。

说明

如果网格包含多个拆分，则调用其 ClearSelCols 方法与调用当前拆分的 ClearSelCols 方法有相同效果。可由 DataGrid 控件的 Split 属性得到当前拆分的索引。

用 SelStartCol 和 SelEndCol 属性来决定拆分的当前列选定范围。

请参阅

Split 属性，SelEndCol, SelStartCol, SelEndRow, SelStartRow 属性。



## ColContaining 方法

返回包含指定的 (X) 坐标值的 DataGrid 控件的列的 ColIndex 值。不支持命名的参数。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.ColContaining coordinate*

ColContaining 方法的语法包含下列部分：

| 部分                | 描述                                    |
|-------------------|---------------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象                  |
| <i>coordinate</i> | 必需的。简单数值表达式，它定义一个基于容器的坐标系统的水平坐标 (X 值) |

说明

ColContaining 方法返回一个与 object 控件指定的列索引之一对应的数。这个数的范围是从 0 到 Columns 集合的 Count 属性减 1(0 到 Columns.Count - 1)。当用鼠标和拖动事件进行工作时，如果要确定在何处单击或在 DataGrid 控件的某列放置另一个控件时，本方法是很有用的。

如果 `coordinate` 到了容器的坐标系之外，就发生一个可以捕获的错误。

**注意：**`ColContaining` 方法返回的是指定列而不是可视列的 `ColIndex`。如果 `coordinate` 落在第一个可视的列中，但是两个列已滚出控件的左边，`ColContaining` 方法返回 2。

请参阅

`Column` 对象，`RowContaining` 方法，`ColIndex` 属性，`Data` 控件。

示例

本例是保存用户开始实施一个拖动方法时所处单元的值。

`Dim SaveValue`

```
Sub DataGrid1_MouseDown (Button As Integer, Shift As Integer, _  
    X As Single, Y As Single)
```

```
    Dim RowValue, ColValue
```

```
    '获得鼠标经过的行列值。
```

```
    RowValue = DataGrid1.RowContaining(Y)
```

```
    ColValue = DataGrid1.ColContaining(X)
```

```
    '如果两个值都有效，则保存单元的文本并
```

```
    '开始拖动。
```

```
    If RowValue > 0 And RowValue < DataGrid1.VisibleRows And _
```

```
        ColValue > 0 And ColValue < DataGrid1.VisibleCols Then
```

```
        SaveValue = DataGrid1.Columns(ColValue). _
```

```
        CellValue(DataGrid1.RowBookmark(RowValue))
        DataGrid1.Drag 1
    End If
End Sub
```

## ColEdit 事件

在键入字符，使单元第一次进入编辑模式时出现该事件。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

```
Private Sub object_ColEdit([ index As Integer,] ByVal colindex As Integer)
```

ColEdit 事件的语法包含下面部分：

| 部分              | 描述                    |
|-----------------|-----------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象  |
| <i>index</i>    | Integer，它标识控件数组中的一个控件 |
| <i>colindex</i> | Integer，它标识被编辑过的列     |

## 说明

如果没有使用浮动编辑器字幕，那么，当用户单击当前单元或双击另一单元时也会出现该事件。

只有在未取消 **BeforeColEdit** 事件时，**ColEdit** 事件才紧跟在 **BeforeColEdit** 事件后出现。

当在网格单元中完成编辑时，比如当 **Tab** 切换到同一行的另外一列时按 **ENTER** 键，或单击另外一单元，如果数据已被改变，则将执行 **BeforeColUpdate** 和 **AfterColUpdate** 事件。随后，引发 **AfterColEdit** 事件，表明已完成编辑。

## 请参阅

**AfterColEdit** 事件，**BeforeColEdit** 事件，**AfterColUpdate** 事件。

## ColIndex 属性

返回或设置一个值，指示列在 **DataGrid** 控件的 **Columns** 集合中列的位置以及列在 **DataGrid** 控件中的可见位置（从左到右）。此属性在运行时是只读的，而在设计时是不可用的。

## 应用于

**Column** 对象，**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.ColIndex

**object** 置换元表示对象表达式，其值是“应用于”列表中的对象。

说明

注意，给定位置的值是相同的，就好象所有的列均是可见的。如果隐藏一个或多个列，也不会发生变化。

此属性返回 **Columns** 集合内列的零级索引。注意，**For... Each** 并不能保证返回任何特殊顺序中的项目。

请参阅

**Column** 对象，**Data** 控件。

## ColResize 事件

当某个用户调整 **DataGrid** 控件的一个列的大小时该事件发生。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

# 语法

```
Private Sub object_ColResize ([ index As Integer,] colindex As Integer, cancel As Integer)
```

ColResize 事件语法包括下列部分：

| 部分              | 描述                              |
|-----------------|---------------------------------|
| <i>object</i>   | 对象表达式 ， 其值为 “应用于” 列表中的一个对象      |
| <i>index</i>    | 整数，用来标识一个在控件数组中的控件              |
| <i>colindex</i> | 用来标识列的一个整数                      |
| <i>cancel</i>   | 如设置值中所描述的， 是一个布尔表达式确定是否调整一个列的大小 |

# 设置值

cancel 的设置为：

| 设置    | 描述                 |
|-------|--------------------|
| True  | 取消所做的改变，将列恢复到原来的宽度 |
| False | （缺省）继续宽度改变         |

# 说明

当用户调整一个列的大小时，触发 ColResize 事件。 事件过程可以接收所

做的改变，修改改变程度，或完全取消所做的改变。

如果设置 `cancel` 变元为 `True`，列宽被恢复。要修改改变程度，可设置 `Column` 对象的 `Width` 属性为想要的值。

在过程中执行 `Refresh` 方法引起控件重画 (`repaint`)，不管 `cancel` 变元是否为 `True`。

请参阅

`RowResize` 事件，`ColIndex` 属性。

示例

如果用户调整第一列的尺寸的话，本例按第一列的大小调整所有列的尺寸。

```
Private Sub DataGrid1_ColResize (ColIndex As Integer, Cancel As Integer)
```

```
    Dim nCol As Column
```

```
    If ColIndex = 1 Then
```

```
        For Each nCol In DataGrid1.Columns
```

```
            nCol.Width = DataGrid1.Columns(1).Width
```

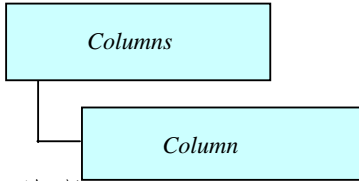
```
        Next
```

```
    End If
```

```
End Sub
```

## Column 对象

Column 对象代表在 DataGrid 控件中的一列。



说明

使用 Column 对象的方法和属性操作 DataGrid 控件中的一列。使用 Column 对象修改列头和列本身的属性。

**注意：**DataGrid 对象只能包含 32767 列，所以列索引以整数形式存储。为了使用 Column 对象，或者直接使用 DataGrid 控件的 Columns 属性，或者把每一列赋值给单独的变量，这些变量被作为 Column 对象指定了大小。下面示例说明后一种情况：

```
Dim Col1, Col2 as Column
Set Col1 = DataGrid1.Columns(0)
Set Col2 = DataGrid1.Columns(1)
Col1.Caption = "Column 1"
Col2.Caption = "Column 2"
```

如果在 DataGrid 控件中经常引用列，可以用上述方法，把值赋给列，而不



是用诸如下例中的 **Columns** 属性，从而提高性能。

```
DataGrid1.Columns(0).Caption = "Column 1"
```

## 属性

**Button** 属性（**Column** 对象），**WrapText** 属性（**Column** 对象），**AllowSizing** 属性，**ColIndex** 属性，**DividerStyle** 属性，**NumberFormat** 属性，**DefaultValue** 属性（**DBGrid** 控件），**DataFormat** 属性，**Height**，**Width** 属性，**Left**，**Top** 属性，**Alignment** 属性，**Locked** 属性，**Caption** 属性，**DataChanged** 属性，**DataField** 属性，**Text** 属性（**ActiveX** 控件），**Left**，**Top** 属性（**ActiveX** 控件），**Visible** 属性（**ActiveX** 控件），**Object** 属性（**ActiveX** 控件）。

## 方法

**CellText** 方法，**CellValue** 方法。

请参阅

**Columns** 集合。

## ColumnHeaders 属性

返回或设置一个值，指示是否在 **DataGrid** 控件中显示列标头。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.ColumnHeaders [= *value*]

ColumnHeaders 属性的语法具有这些部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象       |
| <i>value</i>  | 如“设置值”中所示，确定列标头是否被显示的布尔表达式 |

设置

value 的设置值为：

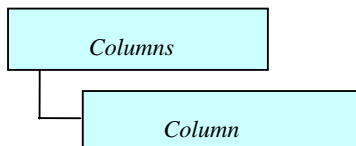
| 设置    | 描述                  |
|-------|---------------------|
| True  | 显示 DataGrid 控件的列标头  |
| False | 不显示 DataGrid 控件的列标头 |

请参阅

Column 对象，Column 属性（DBGrid），Data 控件。

## Columns 集合

Columns 集合包含 DataGrid 控件中所有保存的 Column 对象。



### 语法

Columns(*index*)

Columns.Item(*index*)

### 说明

能用 Columns 集合的属性和方法增加和删除 Column 对象，计算在 Columns 集合中列的数目，给 Columns 集合中的每个列定址。

可以通过 DataGrid 控件的 Columns 属性访问 Columns 集合。

### 属性

Count 属性。

## 方法

Item 属性, Item 方法, Add 方法(Columns, SelBookmarks, Split 集合), Remove 方法(DBGrid)。

请参阅

Column 对象。

## Columns 属性 (DataGrid)

返回一个 Column 对象的集合。

应用于

Split 对象, DataGrid 控件常量, DataGrid 控件。

语法

*object*.Columns

object 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

Columns 属性返回一个在 Variant 中的 Column 对象的集合。

可以通过改变 **Column** 对象的属性的方法来操作 **DataGrid** 控件的大多数属性。用 **Col** 属性选择一个特定的 **Column** 对象。

请参阅

**Column** 对象，**Add** 方法（**Columns**, **SelBookmarks**, **Splits** 集合），**Data** 控件。

## CurrentCellModified 属性

设置或返回当前单元的修改状态。在设计时不可用。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.CurrentCellModified [= *value*]

**CurrentCellModified** 属性的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>value</i>  | 布尔表达式，该表达式决定当前单元的修改状态，参见“设置”中的描述 |

# 设置

value 的设置值为：

| 设置值   | 描述                                       |
|-------|--|
| True  | 正在进行编辑，用户已修改当前单元（由 Bookmark 和 Col 属性所指定） |
| False | 尚未修改该单元，或未进行编辑                           |

# 说明

可用该属性取消用户对当前文本所做的任何改动。例如，对一个功能键编程，可放弃用户所做改动（就像 ESC 键那样），在网格的 KeyDown 事件中捕获键的编码，并把 CurrentCellModified 设置为 False。这会把当前单元改回列原来的内容。

# 请参阅

Bookmark 属性 (DataGrid) ,Col,Row 属性。

# CurrentCellVisible 属性

设置或返回当前单元的可见性。在设计时不可用。

应用于

DataGrid 控件常量，DataGrid 控件，Split 对象。

语法

*object*.CurrentCellVisible [= *value*]

CurrentCellVisible 属性的语法包含下面部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象            |
| <i>value</i>  | 布尔表达式，该表达式决定当前单元的可见性，参见“设置”中的描述 |

设置

value 的设置值为：

| 设置值   | 描述   |
|-------|--|
| True  | （由 Bookmark 和 Col 属性所指示的）当前单元在网格或拆分的显示区中可视 |
| False | 单元是不可视的                                    |

## 说明

对于 **DataGrid** 控件，把 **CurrentCellVisible** 属性设置为 **True** 将会引起网格滚动，使当前单元出现在视图中。如果网格包含多个拆分，则当前单元在每一个拆分中都是可视的。

对于 **Split** 对象，把 **CurrentCellVisible** 属性设置为 **True** 将使当前单元只在那个拆分中是可视的。

在所有的情况下，把该属性设置为 **False** 都是毫无意义的，而且都会被忽略。

## 请参阅

**Bookmark** 属性 (**DataGrid**) ,**Col**,**Row** 属性。

## **DataChanged** 属性 (**DataGrid**)

返回或设置一值，指出被绑定的控件中的数据已被某进程改变，而不获取当前记录的数据。在设计时不可用。

## 应用于

**DataGrid** 控件常量，**DataGrid** 控件。



# 语法

```
object.DataChanged [= value]
```

DataChanged 属性语法具有以下几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的对象     |
| <i>value</i>  | 布尔表达式，指出是否可改变数据，如“设置”中所述 |

# 设置

value 的设置值为：

| 设置    | 描述                              |
|-------|---------------------------------|
| True  | 当前控件中的数据不同于当前记录中的数据             |
| False | （缺省）如果当前控件中的数据存在的话，就与当前记录中的数据相同 |

# 说明

当 data 控件从记录移动到记录时，它将数据从当前记录中的字段移动到被绑定到特定字段或整个记录的控件。当在被绑定的控件中显示数据时，DataChanged 属性被设置成 False。如果用户或任何其它操作改变了被绑定的控件中的值，则 DataChanged 属性被设置成 True。直接移动到其它记录上并不

会影响 **DataChanged** 属性。

当 **data** 控件开始移动到不同的记录时，则出现有效性事件。如果 **DataChanged** 对任何被绑定的控件为 **True**，则 **data** 控件自动调用 **Edit** 和 **Update** 方法通知对数据库的改变。

如果不希望将变化从被绑定的控件保存到数据库，则可在有效性事件中将 **DataChanged** 属性设置成 **False**。

在控件的改变事件的代码中检查 **DataChanged** 属性的值，避免层叠事件。

## 数据类型

**Integer** (**Boolean**)

请参阅

**DataGrid** 控件常量。

## **DataField** 属性 (**DataGrid** 控件、 **Column** 对象)

返回或设置值，将控件绑定到当前记录中的字段上。

应用于

**DataGrid** 控件常量，**DataGrid** 控件，**Column** 对象。

## 语法

*object.DataField* [= *value*]

DataField 属性语法具有以下几部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的对象   |
| <i>value</i>  | 字符串表达式，其值为 Recordset 对象中字段的一个名称，而对象是由 data 控件的 RecordSource 和 DatabaseName 属性指定的 |

## 说明

被绑定的控件可在数据库中访问特定数据。通常，管理单一字段的被绑定的控件显示当前记录中的特定字段值。被绑定的控件的 DataSource 属性指定一个有效的 data 控件名称，而 DataField 属性则在 data 控件所创建的 Recordset 对象中指定有效的字段名称。这些属性指定出现在被绑定的控件中的数据。

在使用返回表达式的 QueryDef 对象或 SQL 语句时，字段名称自动由 Microsoft Jet 数据库引擎生成。例如，在 SQL 查询中编写 SQL 合计函数或表达式时，除非用 AS 子句给合计字段起别名，否则自动生成字段名称。通常，表达式字段名称为 Expr1，以三个字符数字 000 开头。返回的第一个表达式的名称 Expr1000。

推荐您编写 SQL 查询别名表达式的列，如下所示：

```
Adodc1.RecordSource = "Select AVG(Sales)  " _  
    & " AS AverageSales From SalesTable"  
Text1.DataField = "AverageSales"  
Adodc1.Refresh
```

**注意：**确保 DataField 属性设置对每个被绑定的控件都有效。如果改变 data 控件的 RecordSource 属性的设置并使用 Refresh，则 Recordset 识别新对象。这可能使被绑定的控件的 DataField 设置值无效，并产生可捕获的错误。

数据类型

String

请参阅

DataGrid 控件常量。

**DefaultValue 属性（DataGrid 控件）**

为新列数据设置缺省值。

应用于

DataGrid 控件常量，DataGrid 控件，Column 对象。

语法

*object.DefaultValue [= value]*

DefaultValue 属性的语法具有这些部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象    |
| <i>value</i>  | 一个包含指定列缺省值的 Variant 表达式 |

说明

DataGrid 控件本身不使用该属性，但将它作为定位符将缺省值与 DataGrid 控件中的列联结起来。可将该属性当作列的标记使用。可存储任意值并能在以后检索它们。

请参阅

Column 对象。

## DefColWidth 属性

返回或设置一个值，指示 **DataGrid** 控件中所有列的缺省宽度。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.DefColWidth [= *value*]

DefColWidth 属性的语法具有这些部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 一个基于控件的比例模型的整数       |

说明

如果将 **DefColWidth** 属性设置为 0，则该控件在列标头宽度或基本字段的 **Size** 属性设置的基础上自动调整所有列的大小。例如，将所有列的缺省列宽度设置为第一列的宽度：

```
DataGrid1.DefColWidth = DataGrid1.Columns(0).Width
```

请参阅

Column 对象，AllowRowSizing 属性，Data 控件。

## DividerStyle 属性

返回或设置一个值，指定画在 DataGrid 控件的选定列右边缘上的边框样式。

应用于

Column 对象，DataGrid 控件常量，DataGrid 控件。

语法

*object.DividerStyle* [= *value*]

DividerStyle 属性的语法具有这些部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 如“设置”中所示，一个指定边框样式的整数 |

设置

*value* 的设置值为：

| 常量               | 值 | 描述                          |
|------------------|---|-----------------------------|
| dbgNoDividers    | 0 | 无分界线                        |
| dbgBlackLine     | 1 | 黑色线                         |
| dbgDarkGrayLine  | 2 | （缺省）深灰色线                    |
| dbgRaised        | 3 | 突起                          |
| dbgInset         | 4 | 内陷                          |
| dbgUserForeColor | 5 | 分界线是用 ForeColor 属性所设置的颜色画出的 |
| dbgLightGrayLine | 6 | 浅灰色线                        |

### 说明

DividerStyle 属性不影响列是否能够通过拖动的方法来改变大小。当边框是突起的或内陷的时候，所用的颜色由 Microsoft Windows 来设置。

### 请参阅

RowDividerStyle 属性，Data 控件。

### 示例

这个例子在用户单击标头的时候改变列的分界线。

```
Private Sub DataGrid1_HeadClick (ColIndex As Long)
    If DataGrid1.Columns(ColIndex).DividerStyle <> 5 Then
        DataGrid1.Columns(ColIndex).DividerStyle = _
```



```
DataGrid1.Columns(ColIndex).DividerStyle + 1
Else
    DataGrid1.Columns(ColIndex).DividerStyle= 0
End If
End Sub
```

## EditActive 属性

设置或返回当前单元的编辑状态。在设计时不可用。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.EditActive [= *value*]

EditActive 属性的语法包含下面部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象        |
| <i>value</i>  | 布尔表达式，该表达式决定编辑状态，参见“设置”中的描述 |

# 设置

value 的设置值为：

| 设置值   | 描述       |
|-------|----------|
| True  | 正在编辑当前单元 |
| False | 未进行编辑    |

# 说明

如果网格不在编辑模式下，把 **EditActive** 设置为 **True** 将开始对当前单元进行编辑。将插入符定位到单元的末尾并触发 **ColEdit** 事件。

如果网格已在编辑模式下，则把 **EditActive** 设置为 **False**，此时将退出编辑模式。如果已对单元进行过修改，则将触发下列事件：**BeforeColUpdate**、**AfterColUpdate** 和 **AfterColEdit**。

注意：为完全取消编辑，可先把 **CurrentCellModified** 属性设置为 **False**，再把 **EditActive** 设置为 **False**。

# 请参阅

**AfterColEdit** 事件，**ColEdit** 事件，**CurrentCellModified** 属性，**AfterColUpdate** 事件，**BeforeColUpdate** 事件。

## Error 事件（DataGrid 控件）

该事件由于数据访问错误而出现，而在没有执行 Visual Basic 代码时就会产生这个错误。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

```
Private Sub object_Error([ index As Integer,] ByVal dataerror As Integer,  
response As Integer)
```

Error 事件的语法包含下面部分：

| 部分               | 描述                                   |
|------------------|--------------------------------------|
| <i>object</i>    | 对象表达式，其值是“应用于”列表中的一个对象               |
| <i>index</i>     | Integer，它标识控件数组中的一个控件                |
| <i>dataerror</i> | Integer，它标识已出现的错误                    |
| <i>response</i>  | Integer，将其设置为 0，就可不显示错误信息，参见“设置”中的描述 |

设置

*response* 的设置值为：

| 设置值 | 描述               |
|-----|------------------|
| 0   | 不显示错误消息          |
| 1   | (缺省值) 显示与错误关联的消息 |

### 说明

即使应用程序在代码中处理了运行时错误，但在未执行代码时，错误也仍将出现，比如在单击 **Data** 控件按钮时，或在通过与被绑定的控件交互作用，以改变当前记录的时候。如果这样的一个操作导致了数据访问错误，则将引发 **Error** 事件。

不为该事件添加代码，这等效于将 **response** 参数设置为 0。

**注意：**用 **ErrorText** 属性来检索要显示的错误字符串。

### 请参阅

**ErrorText** 属性。

## ErrorText 属性

返回来自下一级数据源的错误消息串。在设计时不可用。

### 应用于

**DataGrid** 控件常量，**DataGrid** 控件。

## 语法

*object*.ErrorText

ErrorText 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

## 说明

用户和网格交互作用会产生数据库错误，比如在一个数字型字段中输入文本，随后又试图移动到另一行以更新当前记录，此时将引发该网格的 **Error** 事件。但是，通过 **DataError** 参数传给该事件处理程序的错误代码不能标识所出现的特殊错误，甚至会因操作环境而不同。鉴于这些原因，应提供 **ErrorText** 属性，使应用程序可对实际的错误消息进行语法分析，以决定错误的本质。

**注意：**仅在 DataGrid 控件的 Error 事件处理程序中，ErrorText 属性才有效。如果试图在其它上下文中访问它，就会出现一个可捕获的错误。

## 请参阅

Error 事件（DataGrid 控件）。

## FirstRow 属性

返回或设置一个值，包含 **DataGrid** 控件或 **Split** 对象中第一个可见行的书签。在设计时不可用。

应用于

**Split** 对象，**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.FirstRow [= *value*]

**FirstRow** 属性的语法具有这些部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象  |
| <i>value</i>  | 字符串表达式，它包含相应于 <b>DataGrid</b> 控件或 <b>Split</b> 对象中第一个可见行的书签 |

说明

对 **DataGrid** 控件，设置 **FirstRow** 属性将导致网格滚动，使特定行成为最顶层的行。如果网格包含多个拆分，则最顶层的行在每个拆分中都改变，甚至在拆分具有不同的 **ScrollGroup** 属性设置值时也如此。

对 Split 对象，FirstRow 属性中的设置使指定的行只对该拆分变成最顶层的行。

请参阅

RowBookmark 方法。

### GetBookmark 方法

对与 DataGrid 控件中当前行相关的某一行，返回一个包含书签的值。 不支持命名参数。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.GetBookmark *value*

The GetBookmark 方法的语法包含下列部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                               |
| <i>value</i>  | 必需的。如“设置值”中所描述的长数值表达式，它确定与当前行的 DataGrid 控件相关的那些行位置 |

# 设置

value 的设置值为:

| 设置值 | 描述  |
|-----|---|
| 0   | 返回当前行的书签—与 <code>DataGrid1.Bookmark</code> 相同 |
| 1   | 返回当前行之后一行的书签                                  |
| -1  | 返回当前行之前一行的书签                                  |
| n   | 返回与基于 $(DataGrid1.Row + n)$ 的当前行相关的行的书签       |

# 说明

`GetBookmark` 方法返回的值可能同 `RowBookmark` 方法有很大差异, 因为当前行可能不是可见的。

请参阅

`RowBookmark` 方法, `Data` 控件。

# 示例

在这个例子中, 检查一个特定列的更新值, 以确保新值位于上一行和下一行的值之间。

```
Sub DataGrid1_BeforeColUpdate (ColIndex As Integer, _  
    OldValue as Variant, PrevVal, NextVal, CurVal, Cancel As Integer)
```



```

If ColIndex = 1 Then
    PrevVal = DataGrid1.Columns(1).CellValue(_
DataGrid1.GetBookmark(-1))
    NextVal = DataGrid1.Columns(1).CellValue(_
DataGrid1.GetBookmark(1))
    CurVal = DataGrid1.Columns(1).Value
    If CurVal > PrevVal Or CurVal < NextVal Then
        Cancel = True
        MsgBox "Value must be between" & PrevVal _
& " and " & NextVal
    End If
End If
End Sub

```

## HeadClick 事件

在用户单击一个 DataGrid 控件指定列的标题时发生。

应用于

DataGrid 控件常量，DataGrid 控件。

## 语法

Private Sub *object*\_HeadClick ([ *index* As Integer,] *colindex* As Integer)

HeadClick 事件语法包括下列部分：

| 部分              | 描述                       |
|-----------------|--------------------------|
| <i>object</i>   | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>index</i>    | 整数，用来标识在控件数组中的控件         |
| <i>colindex</i> | 用来标识列的一个整数               |

## 说明

这个事件可用于按照选定的列对 **Recordset** 对象进行重新排序。

## 请参阅

**ColIndex** 属性，**RecordSet** 属性。

## 示例

本例基于被单击的列对 **Data** 控件的记录源进行排序。

```
Private Sub DataGrid1_HeadClick (ColIndex As Integer)
    Data1.RecordSource = "Select * From Publishers Order By " & _
        DataGrid1.Columns(ColIndex).DataField
    Data1.Refresh
```

End Sub

## HeadFont 属性

返回或设置一个值，指示在 DataGrid 控件列标头中使用的字体。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.Type [= *value*]

Type 属性的语法具有这些部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                |
| <i>value</i>  | 对 Font 对象进行计算的表达式。缺省值是将该列的当前字体设置为粗体 |

说明

改变 HeadFont 属性可能会改变标头的对象以接纳新字体。

请参阅

Column 对象，DefColWidth 属性，Data 控件，Font 对象。

## HeadLines 属性

返回或设置一个值，指示显示在 DataGrid 控件标头的列标头中的文本行数。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.HeadLines [= *value*]

HeadLines 属性的语法具有这些部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象   |
| <i>value</i>  | 一个从 0 到 10 的 Single 数据类型缺省值为 1，它导致这样的控件，该控件对标头中的每一列显示基本字段的名称，设置 0 将删除该标头 |

## 说明

**HeadLines** 属性可以用来在 **DataGrid** 控件的列标头中显示多于一行的文本。

## 请参阅

**Column** 对象，**Columns** 属性（**DBGrid**），**DefColWidth** 属性，**Data** 控件。

## 示例

这个例子检查复选框的值来确定是否在网格中显示标头。

```
Private Sub Check1_Click ()  
    If Check1.Value = vbChecked Then  
        DataGrid1.HeadLines = 2    '如果已检查，  
                                   '列标头中的两行。  
    Else  
        DataGrid1.HeadLines = 0    '无标头。  
    End If  
End Sub
```

## HoldFields 方法

设置当前列/字段布局作为自定义布局。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.HoldFields

*object* 置换元表示对象表达式，其值是“应用于”列表中的对象。

说明

HoldFields 方法设置当前列/字段布局作为自定义布局，随后的 ReBind 操作将用当前布局进行显示。您可以通过调用 ClearFields 方法，使网格继续自动执行。

请参阅

ClearFields 方法，Rebind 方法。

## hWndEditor 属性

通过 Microsoft Windows 操作环境返回 DataGrid 控件编辑窗口的唯一窗口句柄。在运行时不可用。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.hWndEditor

hWndEditor 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

说明

有经验的用户可把属性值传递给需要有效窗口句柄的 Windows API 调用。

在尚未进行编辑的时候，该属性返回 0。

注意：由于该属性值在程序运行时会发生变化，所以绝不要把 hWndEditor 的值存储在变量中。也不要使用 hWndEditor 属性来检测是否正在进行编辑。正是为此才提供了 EditActive 属性。

请参阅

EditActive 属性。

## Index 属性（Split 对象）

返回对选定拆分的索引。

应用于

Split 对象。

语法

*object*.Index

object 置换元表示对象表达式 其值是“应用于”列表中的对象。

说明

如果需要标识一个给定的 Split 对象，此属性将对您很有用。

## LeftCol 属性

返回或设置一个整数，表示 DataGrid 控件最左端的可见列，该属性在设计时是只读的。

应用于

DataGrid 控件常量，DataGrid 控件。



语法

```
object.LeftCol [= value]
```

LeftCol 属性语法有以下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象   |
| <i>value</i>  | 数值表达式，指示最左端的可见列。缺省值是 0 |

MarqueeStyle 属性

设置或返回 DataGrid 控件或 Split 对象的字幕样式。

应用于

DataGrid 控件常量，DataGrid 控件，Split 对象。

语法

```
object.MarqueeStyle [= value]
```

MarqueeStyle 属性的语法包括以下部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象        |
| <i>value</i>  | 一个数字或常量，它指定字幕的样式，参见“设置”中的描述 |

设置

value 的设置值为：

| 常量                  | 值 | 描述   |
|---------------------|---|--|
| dbgDottedCellBorder | 0 | 用虚线边框将当前行中的当前单元框起来，因而可突出显示当前单元。用 Microsoft Windows 的术语来说，这是一个焦点方框                                    |
| dbgSolidCellBorder  | 1 | 用实线边框将当前行中的当前单元框起来，因而可突出显示当前单元。实线框比虚线框更为醒目，当网格使用三维分界属性时情况更是如此  |
| dbgHighlightCell    | 2 | 在当前单元中反转颜色，因而可突出显示整个当前单元。这就给当前单元提供了极为醒目的块状突出显示   |
| dbgHighlightRow     | 3 | 在包含当前单元的一整行中将颜色反转，因而可突出显示这一行。在该模式中不能从外观上判定哪个单元是当前单元，而只能判定当前行。当无法对网格或拆分进行编辑时，该设置值常常优先，因为单元位置在此时是无关紧要的 |

续表

|                   |   |   |
|-------------------|---|---|
| dbgHighlightRow   | 4 | 突出显示整个行。行中的当前单元会“凸出来”，看起来很醒目。在所有的背景色和分界设置的衬托下，该设置看起来不是很清楚。使用三维分界和浅灰色的背景时可获得最佳效果 |
| RaiseCell         |   |   |
| dbgNoMarquee      | 5 | 不显示字幕。如果当前行是无关紧要的，或者暂时还不想让用户的注意力集中在网格，则该设置极为有用                                  |
| dbgFloatingEditor | 6 | 通过一个带闪烁插入符的浮动编辑器（就像 Microsoft Access 中那种样式的编辑器）来突出显示当前单元。这是缺省设置                 |

说明

如果网格包含多个拆分，那么设置该网格的 **MarqueeStyle** 属性的效果与单独设置每个拆分的 **MarqueeStyle** 属性的效果相同。

**注意：**如果浮动编辑器的字幕设置值有效，而且当前单元包含了单选按钮或图形，则将显示一个虚线焦点框。

NumberFormat 属性

返回或设置一个值，指示 **DataGrid** 控件的 **Column** 对象的格式字符串。

应用于

Column 对象，DataGridView 控件常量，DataGridView 控件。

语法

*object.NumberFormat* [= *value*]

NumberFormat 属性的语法具有这些部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                         |
| <i>value</i>  | 字符串表达式定义 Value 属性中的表达式如何被格式化，缺省值为零长度的字符串("") |

说明

将此格式用于 Column 对象的 Value 属性上，从而影响 Column 对象的 Text 属性。如果 NumberFormat 被设置为一个非法的字符串，则单元中的数据被显示为 #ERR# 而且在 Value 属性中设置的值保持不变。有关合法的格式字符串的信息，请参阅 Format 函数。

请参阅

Column 对象，Data 控件，Text 属性，Value 属性，Format 函数，Format 事件，Unformat 事件。

## 示例

这个例子将 **DataGrid** 控件中的第二列格式化为 **long date**:

```
Private Sub Command1_Click ()  
    DataGrid1.Columns(1).NumberFormat = "long date"  
End Sub
```

## OnAddNew 事件

在用户操作调用 **AddNew** 操作时出现。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

```
Private Sub object_OnAddNew([ index As Integer])
```

**OnAddNew** 事件的语法包含下面部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>index</i>  | <b>Integer</b> ，它标识控件数组中的一个控件 |

## 说明

当使用下列几种方式启动 **AddNew** 操作时将出现 **OnAddNew** 事件：

- ◆ 在 **AddNew** 行中修改单元。一般来说，键入一个字符就会出现该事件，但也会由于选定一个内置的单选钮或组合框而使这个事件出现。
- ◆ 当 **AddNew** 行包含当前单元时，在代码中设置一系列的 **Value** 或 **Text** 属性。

只有当网格的 **AllowAddNew** 属性为 **True** 时才会引发该事件。

当 **OnAddNew** 事件点火时，**AddNewMode** 属性取值为 2（**AddNew** 挂起）。

## 请参阅

**AddNewMode** 属性，**Value** 属性（**Column** 对象），**AllowAddNew** 属性。

## Rebind 方法

重新生成 **DataGrid** 控件属性和列。不支持命名的参数。

## 应用于

**DataGrid** 控件常量，**DataGrid** 控件。

## 语法

*object*.Rebind

## 说明

**Rebind** 方法使 **DataGrid** 控件执行与设置 **DataSource** 属性时发生的操作相同的操作。**DataGrid** 控件对列标题以及基于当前 **Data** 控件属性的其它属性进行复位。

如果在设计时未修改网格列,则执行 **ReBind** 方法将复位列、标头以及其它基于当前数据源的属性。

但是,如果已在设计时以任何方式改变了列(甚至留下 **DataField** 属性空白),则网格将假定您想要保持已修改了的网格布局并不会自动复位列。

对未绑定的(具有设置为 1 的 **DataMode** 属性的)网格,该方法与 **Refresh** 方法相似,只是此处的网格要恢复当前的与最顶层的行。

**注意:** 为强制网格复位列绑定,甚至在设计时修改了列时也如此,请在 **ReBind** 之前直接调用 **ClearFields** 方法。反之,为取消网格自动布局响应并强制网格使用当前的列/字段布局,请在 **ReBind** 之前直接调用 **HoldFields** 方法。

## 请参阅

**Data** 控件。

## 示例

这个例子检查一个全局变量，查看用户是否改变了表的布局并用原始表信息重新配置。

```
Sub CheckForRebind_Click ()  
    If UserChangedLayout Then  
        DataGrid1.Rebind  
    End If  
End Sub
```

## RecordSelectors 属性

返回或设置一个值，指示记录选择器是否被显示在 **DataGrid** 控件或 **Split** 对象中。

应用于

**DataGrid** 控件常量，**DataGrid** 控件，**Split** 对象。

语法

*object*.RecordSelectors [= *value*]

**RecordSelectors** 属性的语法具有这些部分：



| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象         |
| <i>value</i>  | 如“设置值”中所示，确定记录选择器是否被显示的布尔表达式 |

## 设置

value 的设置值为：

| 设置    | 描述       |
|-------|----------|
| True  | 显示记录选择器  |
| False | 不显示记录选择器 |

## 说明

当显示时，记录选择器出现在网格或拆分中各行的左边。

如果网格包含多个拆分，则设置其 **RecordSelectors** 属性与分别单独设置每个拆分的 **RecordSelectors** 属性相同。

**注意：**当用户单击记录选择器选定行时，就把选定行的书签添加到 **SelBookmarks** 集合中。

## 请参阅

Column 对象，SelBookmarks 属性，Data 控件。

## Remove 方法 (DataGrid)

从 SelBookmarks 集合中删除指定的行，或从 DataGrid 控件的 Column 集合中删除指定的 Columns 对象。

应用于

Column 对象，SelBookmarks 集合，DataGrid 控件常量，DataGrid 控件。

语法

*object.Remove index*

Remove 方法的语法包含下列部分：

| 部分            | 描述                                    |
|---------------|---------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                  |
| <i>index</i>  | 必需的。一个整数，其值的范围为 0 到集合的 Count 属性设置值 -1 |

说明

对于 SelBookmarks 集合来说，Remove 方法删除由 index 参数指定的行，然后把 SelBookmarks.Count 属性减 1。如果从 SelBookmarks 集合中删除的行是可见的，它将从 DataGrid 控件中取消选中状态。

对于 Columns 集合，Remove 方法删除 index 参数指定的列，然后把 Columns.Count 属性减 1。

如果指定的行不在 SelBookmarks 集合，或者一个 Column 对象不在 Columns 集合中，将发生一个可以捕获的错误。

请参阅

Add 方法（Columns, SelBookmarks, Splits 集合），ColIndex 属性，Data 控件。

## RowBookmark 方法

对 DataGrid 控件中的可见行，返回一个包含书签的值。不支持命名的参数。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.RowBookmark value*

RowBookmark 方法的语法包含下列部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                                   |
| <i>value</i>  | 必需的。一个整数，其值的范围为 0 到 DataGrid 控件的 VisibleRows 属性的设置值减 1 |

## 说明

RowBookmark(0) 返回的书签与 DataGrid 控件的 FirstRow 属性相同。正像 DataGrid 控件的 Bookmark 属性所定义的，如果当前行不是可见的，那么当前行有可能不能通过这一方法返回。

**注意：**RowBookmark 返回的书签将不被保存，因为它们的值当行在 DataGrid 控件中成为可见时会发生变化。

## 请参阅

SelBookmarks 集合，GetBookmark 方法，FirstRow 属性，SelBookmarks 属性，VisibleRows 属性。

## 示例

本例选择网格上当前可见的所有行。

```
Sub SelectAllVisible_Click ()
```

```
    Dim I
```

```
    For I = 0 To DataGrid1.VisibleRows - 1
```

```
        DataGrid1.SelBookmarks.Add DataGrid1.RowBookmark(I)
```

Next I

End Sub

## RowContaining 方法

返回一个与 DataGrid 控件指定的纵坐标 (Y) 的行号相对应的值。不支持命名的参数。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.RowContaining coordinate*

RowContaining 方法的语法包含下列部分：

| 部分                | 描述                                |
|-------------------|-----------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象              |
| <i>coordinate</i> | 必需的。一个单数值表达式，它指定基于容器坐标系统的纵向值（Y 值） |

## 说明

**RowContaining** 方法返回一个值，它对应于由 **object** 指定的控件的列索引之一。这个值的范围是从 0 到 **VisibleRows** 属性的设置值减 -1。当用鼠标和拖事件进行工作时如果要确定用户在何处进行单击或以 **DataGrid** 控件的一个列的形式放置其它控件时这个方法是很有用的。

如果坐标位于容器的坐标系统之外，就发生一个可以捕获的错误。

请参阅

**Column** 对象，**ColContaining** 方法，**ColIndex** 属性，**Data** 控件。

## RowDividerStyle 属性

返回或设置一个值，指定画在选定的 **DataGrid** 控件的行之间的边框样式。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.RowDividerStyle [= *value*]

**RowDividerStyle** 属性的语法具有这些部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 如“设置”中所示，指定边框样式的整数   |

## 设置

value 的设置值为：

| 设置 | 描述                          |
|----|-----------------------------|
| 0  | 无分界线                        |
| 1  | 黑色线                         |
| 2  | （缺省）深灰色线                    |
| 3  | 突起的                         |
| 4  | 内陷的                         |
| 5  | 分界线是使用 ForeColor 属性设置的颜色画出的 |

## 说明

RowDividerStyle 属性不影响边框能否被拖动。当边框是突起的或内陷的时候，Microsoft Windows 设置颜色。

## 请参阅

Column 对象，DividerStyle 属性，Data 控件，BackColor, ForeColor 属性，

BackColor, ForeColor 属性（ActiveX 控件）。

## 示例

当单击一个命令按钮时，这个例在不同的行线样式间切换。

```
Private Sub ChangeStyle_Click ()  
    If DataGrid1.RowDividerStyle < 5 Then  
        DataGrid1.RowDividerStyle = DataGrid1.RowDividerStyle + 1  
    Else  
        DataGrid1.RowDividerStyle = 0  
    End If  
End Sub
```

## RowResize 事件

当某个用户调整一个 DataGrid 控件中的行尺寸时该事件发生。

## 应用于

DataGrid 控件常量，DataGrid 控件。

## 语法

```
Private Sub object_RowResize ([ index As Integer,] cancel As Integer)
```



RowResize 事件语法包括下列部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值为“应用于”列表中的一个对象        |
| <i>index</i>  | 整数，用来标识一个在控件数组中的控件            |
| <i>cancel</i> | 如设置值中所描述的，是一个布尔表达式，指出是否取消一个改变 |

## 设置

cancel 的设置为：

| 设置    | 描述                 |
|-------|--------------------|
| True  | 取消所做的改变，把行恢复为原来的高度 |
| False | （缺省）继续高度改变         |

## 说明

用户可以用鼠标调整 DataGrid 控件行的尺寸。当用户改变高度时，触发 RowResize 事件。事件过程可以接受所做的改变，修改改变的程度，或完全取消所做的改变。

DataGrid 控件的 RowHeight 属性确定了控件中所有行的高度。

如果将 cancel 参数设置为 True ，则恢复行高度。要修改改变的程度，可设置 RowHeight 属性为想要的值。

在过程中执行 **Refresh** 方法引起控件重画，不管 **cancel** 参数是否为 **True**。

请参阅

**Refresh** 方法，**RowHeight** 属性。

示例

本例确保网格上至少有五个可见的行。

```
Private Sub DataGrid1_RowResize (Cancel As Integer)
    If DataGrid1.VisibleRows < 5 Then Cancel = True
End Sub
```

## RowTop 方法

返回一个包含 **DataGrid** 控件的一个指定行顶部的纵坐标 (Y) 的值。不支持命名的参数。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.RowTop *value*

RowTop 方法的语法包含下列部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                           |
| <i>value</i>  | 必需的。整数，它指定一个介于 0 到 VisibleRows 属性的设置值减 1 范围内的行 |

## 说明

RowTop 方法返回一个值，对应于它指定行顶部的 Y 坐标。返回值基于容器的 ScaleMode 属性。

可以使用带 RowHeight, Left 和 Column 对象的 Width 属性的 RowTop 方法确定 DataGrid 控件中的一个选定单元的精确位置和维数。

## 请参阅

Column 对象, ColContaining 方法, VisibleRows 属性, Data 控件。

## 示例

本例开始在网格内的一个拖操作。利用网格单元位置和尺寸属性，一个 Label 控制单元大小被用作拖对象。

```
Sub DataGrid1_MouseDown (Button As Integer, _  
Shift As Integer, X As Single, Y As Single)  
    ' 声明变量。
```

```
Dim DY, DX, RowValue, ColValue, CellLeft, CellTop
ColValue = DataGrid1.ColContaining(X)
RowValue = DataGrid1.RowContaining(Y)
' 取得单元的高度。
DY = DataGrid1.RowHeight
' 取得单元的宽度。
DX = DBGrid1.DataGrid1.Columns(ColValue).Width
CellLeft = DataGrid1.Columns(ColValue).Left
CellTop = DataGrid1.RowTop(RowValue)
Label1.Caption = DataGrid1.Columns(ColValue). _
    CellValue(DataGrid1.RowBookmark(RowValue))
Label1.Move CellLeft, CellTop, DX, DY
Label1.Drag ' 拖动标签外框。
End Sub
```

## Scroll 方法

在一个简单操作中水平或垂直地滚动 DataGrid 控件。不支持命名参数。

应用于

DataGrid 控件常量，DataGrid 控件。

# 语法

*object.Scroll colvalue, rowvalue*

Scroll 方法的语法包含下列部分：

| 部分              | 描述                   |
|-----------------|----------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象 |
| <i>colvalue</i> | 必需的。长数值表达式，指定控件中的某列  |
| <i>rowvalue</i> | 必需的。长数值表达式，指定控件中的某行  |

# 说明

正值往右下方滚动。负值往左上方滚动。超出范围的值不会产生错误—DataGrid 控件可滚到最大级。通过设置 FirstRow 和 LeftCol 属性可以得到同样效果，但它们必须独立设置，且引起两个单独的 Paint 事件。

请参阅

FirstRow 属性。

# 示例

本例创建两个使用户能够沿对角线滚动的按钮，一是向右下移动，二是向左上移动。

```
Sub ScrollDownRight_Click
    ' 向右下滚动。
```

```
DataGrid1.Scroll DataGrid1.VisibleCols, DataGrid1.VisibleRows
End Sub

Sub ScrollUpLeft_Click
    ' 向左上滚动。
    DataGrid1.Scroll -DataGrid1.VisibleCols, -DataGrid1.VisibleRows
End Sub
```

## ScrollGroup 属性

用于使两个拆分之间的垂直滚动同步。

应用于

DataGrid 控件常量，DataGrid 控件，Split 对象。

语法

*object*.ScrollGroup [= *value*]

ScrollGroup 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 一个整数表达式，它决定拆分所属的滚动组  |

## 说明

用该属性使两个拆分之间的垂直滚动同步。对于具有相同 **ScrollGroup** 设置值的所有拆分，当在其中一个拆分中出现了垂直滚动时，所有拆分都将同步进行。属于不同组的拆分可独立滚动，从而允许不同的拆分显示数据库的不同部分。

如果将拆分的 **ScrollBars** 属性设置为 4（自动的），则组中只有最右拆分具有垂直滚动条。如果只有一个拆分，则设置该属性不会产生效果。

设置拆分的 **FirstRow** 属性会影响同一组中的所有其它拆分，以保持该组同步。

新创建的拆分其 **ScrollGroup** 属性值为 1。

请参阅

**FirstRow** 属性，**ScrollBars** 属性。

## SelBookmarks 集合

**SelBookmarks** 集合包含在 **DataGrid** 控件中选择的每一行的书签。

## 语法

**SelBookmarks**

## 说明

用 `DataGrid` 控件的 `SelBookmarks` 属性返回 `SelBookmarks` 集合。按照选中的顺序将书签加入 `SelBookmarks` 集合。可以通过把 `Bookmark` 属性设成在 `SelBookmarks` 集合中一个选定的书签来重新定位 `DataGrid` 控件的当前记录指针。

用 `Add` 方法向 `SelBookmarks` 集合中添加书签。一旦书签添加到 `SelBookmarks` 集合中，它在 `DataGrid` 控件中表现为被选中。

用 `Remove` 方法从 `SelBookmarks` 集合中删除书签。一旦书签从 `SelBookmarks` 集合中删除，它在 `DataGrid` 控件中不再表现为被选中。

`SelBookmarks` 集合支持 `Add` 和 `Remove` 方法以及 `Count` 属性。用这些方法和属性，可以操作在 `DataGrid` 控件中选中的项目列表。例如，能用 `Add` 方法通过编程选择附加项或用 `Count` 属性确定选中项目的数目。

## 属性

`Count` 属性（VB 集合）。

## 方法

`Add` 方法（`Columns`, `SelBookmarks`, `Spilts` 集合），`Remove` 方法（`DBGrid`），`Item` 属性，`Item` 方法。



请参阅

Column 对象，GetBookmarks 方法，RowBookmark 方法，SelBookmarks 属性。

## SelBookmarks 属性

返回一个在 DataGrid 控件中所有选定的记录的书签集合。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.SelBookmarks

object 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

当一个记录在 DataGrid 控件中被选定的时候，它的书签被追加到 SelBookmarks 属性所返回的集合中。例如，如果创建 DataGrid 控件所创建的一组 Recordset 对象，则可以用来自 SelBookmarks 集合的书签将分组的 Recordset 重新定位，以处理单个的数据记录。

请参阅

SelBookmarks 集合, Add 方法 (Columns, SelBookmarks, Splits 集合), Remove 方法 (DBGrid), Data 控件, RecordSet 属性。

示例

这个例子在用户已选定的行间循环并将它们从数据库中删除。

```
Sub DeleteRows()  
    Dim varBmk As Variant  
    For Each varBmk In DataGrid1.SelBookmarks  
        Data1.Recordset.Bookmark = varBmk  
        Data1.Recordset.Delete  
        Data1.Refresh  
    Next  
End Sub
```

## SelChange 事件 (DataGrid 控件)

当用户选定一个不同范围行或列时出现该事件。

应用于

DataGrid 控件常量, DataGrid 控件。

语法

```
Private Sub object_SelChange([ index As Integer,] cancel As Integer)
```

SelChange 事件的语法包含下面部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象        |
| <i>index</i>  | Integer，它标识控件数组中的一个控件       |
| <i>cancel</i> | 布尔表达式，它决定是否取消更改，参见“设置值”中的描述 |

设置

cancel 的设置值为：

| 设置值   | 描述   |
|-------|--|
| True  | 如果事件过程把 Cancel 参数设置为 True，则前面选定的多个行和列（如果有的话）将被保存起来，而 SelEndCol、SelStartCol 和 SelBookmarks 属性都恢复成以前的值 |
| 设置值   | 描述   |
| False | （缺省值）承认改动，继续   |

说明

在以下几种情况中将会触发：

- 单击单独一行的记录选择器来选定这一行。

- 按 **CTRL** 键并单击一行的记录选择器，将该行添加到已选定的行的列表中。
- 单击单独一列的标头来选定该列。
- 在标头行中将已选定的一组列拖动到相邻列来改变这组列。
- 按 **SHIFT** 键并单击一个尚未被选定的列的标头，以增大所选定的一组列。
- 单击一个单元来清除当前选定的行或列，该事件将在 **RowColChange** 事件出现之前引发。

当前所选列的范围是由 **SelStartCol** 和属性 **SelEndCol** 提供的。已选定行的书签都在由 **SelBookmarks** 属性所提供的集合中。在该事件过程中，这些属性反映了用户尚未确定的选择。

只有在用户和网格交互作用时才会触发该事件。而在代码中则不会触发该事件。

**注意：**当用户选定一列时，对行所作的选择都将被清除。同样，当用户选定一行时，对列所作的选择都将被清除。

请参阅

**SelBookmarks** 属性, **RowColChange** 事件, **SelEndCol**, **SelStartCol**, **SelEndRow**, **SelStartRow** 属性。

## Size 属性（Split 对象）

设置或返回拆分的大小。

应用于

DataGrid 控件常量，DataGrid 控件，Split 对象。

语法

*object*.Size [= *value*]

Size 的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 一个整数表达式，它指定拆分的大小     |

说明

该属性所返回值的含义由拆分的 **SizeMode** 属性的设置值来决定。

如果将 **SizeMode** 设置为缺省值 0（可放缩的），则 **Size** 属性返回的值是整数，它指明了该拆分对于另外几个可放缩拆分的相对大小。

如果将 **SizeMode** 设置为 1（大小确定的），则 **Size** 属性返回的值是浮点数，它指明该拆分按网格容器的坐标系来计量的确切大小。

如果将 **SizeMode** 设置为 2（列的个数），则 **Size** 属性返回的值是整数，它指明该拆分中显示的列数。

**注意：**当仅有一个拆分（网格的缺省性能）时，拆分的宽度将足以横跨整个网格，**SizeMode** 属性总是为 0 (dbgScalable)，**Size** 属性则总是为 1。当仅有一个拆分时，设置这两个属性中的任何一个都不会有效果。如果有多个拆分，则仅留一个而把其余拆分都删除掉，则 **SizeMode** 和 **Size** 属性会分别自动重新变成 0 和 1。

请参阅

**SizeMode** 属性（**Split** 对象）。

## SizeMode 属性（Split 对象）

设置或返回一个值，该值决定如何用 **Size** 属性来规定某个拆分的实际大小。

应用于

**DataGrid** 控件常量，**DataGrid** 控件，**Split** 对象。

语法

*object*.SizeMode [= value]

SizeMode 的语法包含下面部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                |
| <i>value</i>  | 一个数字或常量，它决定使用 Size 属性的方法，参见“设置”中的描述 |

## 设置

value 的设置值为：

| 常量          | 值 | 描述   |
|-------------|---|--|
| dbgScalable | 0 | （缺省值）Size 属性返回的值是整数，指明了这个拆分对于其它可缩放拆分的相对大小。例如，如果网格包含三个可缩放拆分，它们的 Size 属性依次取值为 1、2、3，则每个拆分的大小分别为整个网格宽度的 1/6、1/3 和 1/2 |
| dbgExact    | 1 | Size 属性返回的值是浮点数，它指明该拆分按网格容器的坐标系来计量的确切大小。有了这个设置值就可修改拆分的大小，使其总有相同宽度，即使添加新的拆分或删除现有的拆分，情况也如此                           |

续表

|                        |   |   |
|------------------------|---|---|
| dbgNumberOf<br>Columns | 2 | Size 属性返回的值是整数，它指明该拆分中显示的列数，而拆分将调整其宽度，以显示由 Size 属性所指定的列数。例如，如果把 Size 属性设置为 2，并水平滚动拆分，则拆分的宽度将会改变，以显示整整两列，而不管这两列有多宽 |
|------------------------|---|---|

说明

当只有一个拆分时（网格的缺省情况），拆分将伸展到整个网格，SizeMode 属性总是为 0 (dbgScalable)，Size 属性则总是为 1。当只有一个拆分时，设置这两个属性中的任何一个都不会有效果。如果有多个拆分，仅留一个而把其余拆分都删除掉，则 SizeMode 和 Size 属性会分别自动重新变成 0 和 1。

考虑一个网格，它既包含可放缩拆分，也包含具有固定列数的拆分。如果水平地滚动具有固定列数的拆分，则留给可放缩拆分的总宽度就可能改变，因为网格的列一般有不同宽度。但是，可放缩拆分的比例始终保持 Size 属性所规定的值。

请参阅

MarqueeStyle 属性，Size 属性（Split 对象），Split 属性。



## Split 对象

Split 对象表示 DataGrid 控件中的一个拆分。

### 说明

DataGrid 支持那些类似 Excel 的拆分，这些拆分把网格分成垂直窗格，以便为数据库提供不同视图。每个拆分都用 Split 对象来表示，而且还包含一组逐列滚动的彼此相邻的列。按照缺省规定，当创建 DataGrid 对象时，它将包含一个 Split 对象。

可以使用拆分在多个垂直窗格中表示数据。数据窗格（或拆分）可用不同颜色和字体来显示数据。它们可以一起（垂直）滚动或相互独立地滚动，也可以显示相同的或不同的列。还可用拆分来固定一列或多列，使之不能滚动。与其它网格产品不同，不必把已固定的列放在网格的左边，而是放在右边或中间的任何位置上。一网格中甚至还可以有多组固定的列。

每个 Split 对象都拥有自己的 Columns 集合。这些独立的拆分和列提供了功能很强，又极为灵活的数据表示能力。

如上所述，最初网格（DataGrid 对象）只包含一个拆分。如果创建了额外的拆分，则可像如下示例那样用 Split 属性来确定或设置当前拆分（就是已接收到焦点的那个拆分）：

'读当前拆分的索引，这个索引是从零开始算起的

```
Variable% = DataGrid1.Split
```

'把焦点设置到拆分处，拆分的索引等于

```
' Variable%
```

```
DataGrid1.Split = Variable%
```

网格中的每个拆分都是同一数据源的一个不同视图，每个拆分的性能都与一个独立的网格相似。如果没有定制拆分的任何属性而创建额外的 **Split** 对象，则所有的拆分将是等同的，而且每个拆分的性能都与原来那个具有拆分的网格相似。

**DataGrid** 控件的某些属性与 **Split** 对象的属性相同，所以可看作二者共有的。改变 **DataGrid** 控件的普通属性也会使当前 **Split** 对象的相同属性发生变化，反之亦然。例如，对具有两个拆分的网格，假定当前拆分索引为 1（也就是将网格的 **Split** 属性设置为 1）。如果想确定正在使用的字幕样式，则可注意到，下列语句是等价的：

```
marquee% = DataGrid1.MarqueeStyle
```

```
marquee% = DataGrid1.Splits(1).MarqueeStyle
```

如果将当前拆分索引设置为 1，则把 **MarqueeStyle** 属性设置为 **dbgSolidCellBorder** 下列代码是等价的：

```
DataGrid1.MarqueeStyle = dbgSolidCellBorder
```

```
DataGrid1.Splits(1).MarqueeStyle = dbgSolidCellBorder
```

**注意：**对 **DataGrid** 对象及与它关联的 **Split** 对象来说，公共属性

是唯一的。再没有一对对象具有如此相似的关系。

## 属性

AllowFocus 属性, CurrentCellVisible 属性, Index 属性 (Split 对象), MarqueeStyle 属性, ScrollGroup 属性, Size 属性 (Split 对象), SizeMode 属性 (Split 对象), AllowSizing 属性, AllowRowSizing 属性, Columns 属性 (DBGrid), FirstRow 属性, RecordSelectors 属性, Locked 属性, ScrollBars 属性, SelEndCol, SelStartCol, SelEndRow, SelStartRow 属性。

## 方法

ClearSelCols 方法。

请参阅

Splits 集合。

## Split 属性

设置或返回当前拆分的索引。在设计时不可用。

应用于

DataGrid 控件常量, DataGrid 控件。

## 语法

*object.Split* [= *value*]

**Split** 属性的语法包含下面部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                |
| <i>value</i>  | <b>Integer</b> ，它指定当前拆分的索引，参见说明中的描述 |

## 说明

**Split** 属性指定一个对当前拆分的基于零的索引。

请参阅

**Split** 对象。

## SplitChange 事件

当前单元变为另一个拆分中的不同单元时出现该事件。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

```
Private Sub object_SplitChange([ index As Integer])
```

SplitChange 事件的语法包含下面部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | Integer，它标识控件数组中的一个控件  |

说明

该事件将在下列情况中被触发：

- 第一次显示网格。
- 单击另一拆分中的单元（服从 AllowFocus 属性的设置值）。
- 按下定位键以跨越拆分边界（服从 TabAcrossSplits 属性的设置值）。
- 在程序代码中改变 Split 属性的值。
- 通过程序代码或用户交互作用，在当前拆分前插入新的拆分。
- 通过程序代码或用户交互作用来删除当前拆分。

如果编辑数据，接着又把当前单元位置移动到另一个拆分中的新行上，则在执行 SplitChange 事件之前就完成了原行的更新事件。

如果对拆分所做的改变导致当前行或列发生改变，那么，SplitChange 事件始终都在 RowColChange 事件之前出现。

请参阅

AllowFocus 属性，Split 属性，SplitChange 事件，TabActionSplits 属性，RowColChange 事件。

## SplitContaining 方法

返回包含一对指定坐标的拆分的 Index 值。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.SplitContaining* *x*, *y*

SplitContaining 方法的语法包含下面部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象      |
| <i>x</i>      | 必需的。一个单精度值，定义了网格容器坐标系上的水平坐标 |
| <i>y</i>      | 必需的。一个单精度值，定义了网格容器坐标系上的垂直坐标 |

## 说明

该值在 0 和 1 之间,小于 **Splits** 集合的 **Count** 属性设置值(0 到 **Splits.Count** - 1 之间)。

在使用鼠标和拖动事件工作并试图制定任何借助网格列来单击或拖动其它控件时,这一方法极为有用。

如果两个参数之一超出网格的数据区域,则该方法返回 -1。

## 请参阅

**Splits** 集合, **Splits** 属性。

## Splits 集合

**Splits** 集合包含所有存储在 **DataGrid** 控件中的 **Split** 对象。

## 语法

**Splits**(*index*)

**Splits.Item**(*index*)

## 说明

在设计时,可用网格的 **UI** 活动的上下文菜单来创建拆分。在运行时,则

可用 **Splits** 集合的 **Add** 和 **Remove** 方法来创建和删除拆分。每种方法都对拆分索引从零算起。下列代码对运行时添加和删除拆分进行演示：

'创建索引为 0 的拆分对象

**DataGrid1.Splits.Add 0**

'删除索引为 1 的拆分对象

**DataGrid1.Splits.Remove 1**

可用 **Splits** 集合的 **Count** 属性来确定网格中 **R** 拆分数目。

## 属性

**Count** 属性（VB 集合）。

## 方法

**Item** 属性，**Item** 方法，**Add** 方法（**Columns**, **SelBookmarks** , **Splits** 集合），**Remove** 方法（**DBGrid**）。

## Splits 属性

返回 **Split** 对象的集合。在设计时不可用。

## 应用于

**DataGrid** 控件常量，**DataGrid** 控件。



语法

*object.Splits*

Splits 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

请参阅

Split 对象。

TabAcrossSplits 属性

设置或返回 Tab 键和箭头键在拆分边缘处的性能。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object.TabAcrossSplits* [= *value*]

TabAcrossSplits 属性的语法包含下面部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                             |
| <i>value</i>  | 布尔表达式，该表达式决定 <b>Tab</b> 键和箭头键在拆分边缘的性能，参见“设置”中的描述 |

## 设置

**value** 的设置值为：

| 设置值   | 描述  |
|-------|---|
| True  | <b>Tab</b> 键和箭头键将穿越拆分边界移动当前单元。到了最右拆分的最后一列（或最左拆分的第一列）时，这些键或者会自动换到下一行并停下来，或者移动到别的控件上，这将取决于 <b>WrapCellPointer</b> 和 <b>TabAction</b> 属性的设置值 |
| False | （缺省值） <b>Tab</b> 键和箭头键不能穿越拆分边界移动当前单元。这些键或者会自动地换到下一行并停下来，或者移动到别的控件上，这将取决于 <b>WrapCellPointer</b> 和 <b>TabAction</b> 属性的设置值                 |

## 说明

**TabAcrossSplits** 属性不能判定 **Tab** 键和箭头键是否可以在单元之间或控件之间来回移动，或自动向下一行换行。可用 **AllowArrows**，**WrapCellPointer** 和 **TabAction** 属性来控制这种性能。如果 **Tab** 键和箭头键能够在单元之间来回移动，该属性就可判定这些键是否能够穿过拆分边界，移动到相邻的拆分。

请参阅

AllowArrows 属性, TabAction 属性, WrapCellPointer 属性。

## TabAction 属性

设置或返回一个值，该值定义了 Tab 键的行为。

应用于

DataGrid 控件常量, DataGrid 控件。

语法

*object*.TabAction [= *value*]

TabAction 属性的语法包含下面部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象              |
| <i>value</i>  | 一个数字或常量，它定义了 Tab 键的性能，参见“设置值”中的描述 |

设置

*value* 的设置值为：

| 常量                   | 值 | 描述  |
|----------------------|---|---|
| dbgControlNavigation | 0 | (缺省值) Tab 键移动到窗体中的下一个控件或前一个控件   |
| dbgColumnNavigation  | 1 | Tab 键把当前单元移动到下一列或前一列。但是，如果该操作使当前行发生改变，则窗体中的下一个或前一个控件将接收焦点                           |
| dbgGridNavigation    | 2 | Tab 键把当前单元移动到下一列或前一列。Tab 键在行边界上的性能由 WrapCellPointer 属性决定。使用该属性时，Tab 键始终都不会导致另一控件的移动 |

说明

TabAction 属性不能决定 Tab 键是否可穿过拆分的边界。可用 TabAcrossSplits 属性来控制这种性能。

TabAction 属性值覆盖 WrapCellPointer 属性行为。例如，如果 WrapCellPointer 为 True 并且 TabAction 被设为 dbgColumnNavigation，当前光标位于 DataGrid 最后一列处，按下 Tab 键会使光标移动到在 tab 索引顺序中的下一控件，而不是到下一行的第一列。

请参阅

TabActionSplits 属性，WrapCellPointer 属性。

# Value 属性（Column 对象）

设置或返回当前行的一列中的基本数据值。设计时不可用。

应用于

DataGrid 控件常量，DataGrid 控件，Column 对象。

语法

*object.Value* [= *value*]

Value 属性的语法包含下面部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象     |
| <i>value</i>  | 字符串表达式，它表示当前行的某一列中的基本数据值 |

说明

为了在单元中模拟数据输入项，Value 属性十分有用。设置该属性时，单元中显示的值会符合列的 NumberFormat 属性的设置值。

该属性总是返回一字符串变体，即使基本字段的数据类型是数字型，情况亦如此。

可用 Text 属性来访问当前行的列中的格式化数据值。

请参阅

NumberFormat 属性。

## VisibleCols 属性

返回或设置一个值，指示 DataGrid 控件中可见列数。在设计时不可用，且在运行时是只读的。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.VisibleCols

VisibleCols 属性的语法具有这些部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

说明

VisibleCols 属性是从 0 到全体可用列数的整数，包含全部或部分可见的 DBGrid 控件的列。如由 Columns 集合的 Count 属性所确定。

该属性返回当前拆分中可见的列数。返回的值包括全体或部分被显示的列。  
使用 **Split** 属性来确定当前拆分的索引。

请参阅

**Columns** 集合, **VisibleRows** 属性, **Data** 控件, **Count** 属性 (VB 集合), **Count** 属性 (ActiveX 控件)。

示例

这个例子定义按钮将网格向左或向右移动整个一个页面。

```
Private Sub PageRight_Click ()
```

```
    '页网格到右边。
```

```
    If DataGrid1.LeftCol + DataGrid1.VisibleCols < _
```

```
        DataGrid1.Columns.Count Then
```

```
        DataGrid1.LeftCol = DataGrid1.LeftCol + _
```

```
        DataGrid1.VisibleCols
```

```
    End If
```

```
End Sub
```

```
Private Sub PageLeft_Click ()
```

```
    '页网格到左边。
```

```
    If DataGrid1.LeftCol - DataGrid1.VisibleCols >= 0 Then
```

```
        DataGrid1.LeftCol = DataGrid1.LeftCol - _
```

```
DataGrid1.VisibleCols  
End If  
End Sub
```

## VisibleRows 属性

返回一个值，指示 **DataGrid** 控件中可见行数。该属性在运行时是只读的。

应用于

**DataGrid** 控件常量，**DataGrid** 控件。

语法

*object*.VisibleRows

**object** 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

**VisibleRows** 属性返回一个整数，其范围从 0 到 **DataGrid** 控件的 **Rows** 属性所设置的行数。**VisibleRows** 属性包含全部或部分可见的 **DataGrid** 控件行。

请参阅

**VisibleCols** 属性，**Data** 控件。



## 示例

这个例子选定在网格中当前可见的所有行。

```
Private Sub SelectVisible_Click ()  
    Dim I  
    For I = 0 To DataGrid1.VisibleRows - 1  
        DataGrid1.SelBookmarks.Add DataGrid1.RowBookmark(I)  
    Next I  
End Sub
```

## WrapCellPointer 属性

设置或返回一个值，决定箭头键的行为。

应用于

DataGrid 控件常量，DataGrid 控件。

语法

*object*.WrapCellPointer [= *value*]

WrapCellPointer 属性的语法包含下面部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>value</i>  | 布尔表达式，决定箭头键的行为，参见“设置”中的描述 |

## 设置

Value 的设置值为：

| 设置值   | 描述                                       |
|-------|--|
| True  | 单元指针将包围从最后一列到下一行的第一列（或从第一列到上一行的最后一列）     |
| False | （缺省）单元指针不会包围下一行（或上一行），但会停在当前行的最后一列（或第一列） |

## 说明

如果 TabAcrossSplits 为 False，单元指针将只会包围当前 split 。如果 TabAcrossSplits 为 True，单元指针将会在包围前，从一个 split 移动到下一个 split。

如果 TabAction 设置为 2 - Grid Navigation，tab 键将像箭头键一样，并且会自动包围下一个或上一个单元。

## 请参阅

AllowArrows 属性，TabAcrossSplits 属性，TabAction 属性。

## WrapText 属性 (Column 对象)

设置或返回一个值，指出对象是否在单元边界上使文本自动换行。

应用于

DataGrid 控件常量，DataGrid 控件，Column 对象。

语法

*object*.WrapText [= *value*]

WrapText 属性的语法包含下面部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象        |
| <i>value</i>  | 布尔表达式，决定对象是否自动换行，参见“设置”中的描述 |

设置

*value* 的设置值为：

| 设置值   | 描述                                 |
|-------|------------------------------------|
| True  | 一行在这样的单词之前折行，若不在此拆行，则这些单词不能被全部显示出来 |
| False | （缺省值）不折行，文本在单元右边缘被剪开               |

## 说明

可将该属性连同 **RowHeight** 属性一道使用，以形成多行显示。

请参阅

**AllowArrows** 属性，**TabAcrossSplits** 属性，**TabAction** 属性。

# DataCombo 控件

DataCombo 控件是一个数据绑定组合框，它自动地由一个附加数据源中的一个字段充填；并且可选择地更新另一个数据源的一个相关表中的一个字段。

## 语法

DataCombo

## 说明

DataCombo 控件与 DBCombo 控件代码兼容。但是，DataCombo 控件被优化来同 ActiveX Data Objects (ADO)一起工作。

**发行须知：**DataCombo 控件和 DataList 控件一起可以在文件 Msdatlst.ocx 中找到。要在应用程序中使用这一控件，必须把 .OCX 文件添加到工程中。当发布您的应用程序时，要把文件 Msdatlst.ocx 安装到用户的 Microsoft Windows System 或 System32 目录下。关于如何向一个 Visual Basic 工程添加 ActiveX 控件的详细信息，请参阅“标准 ActiveX 控件”。

## 属性

DataCombo, BoundColumn 属性, ListField 属性, RowSource 属性, SelectedItem 属性, VisibleItems 属性, VisibleColumn 属性, MatchedWithList 属性, MatchEntry 属性, Locked 属性 (DataCombo, DataList 控件), Style 属性 (DataCombo 控件), RowMember 属性, BoundText 属性, DataMember 属性, DataFormat 属性, DataBindings 属性, Height, Width 属性, Left, Top 属性, TabIndex 属性, Tag 属性, Visible 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, TabStop 属性, Enabled 属性, HelpContextID 属性, Index 属性 (控件矩阵), Name 属性, Parent 属性, Container 属性, Object 属性, ToolTip Text 属性, DataChanged 属性, DataField 属性, DataSource 属性, IntegralHeight 属性, WhatsThisHelpID 属性, OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), SelLength, SelStart, SelText 属性 (ActiveX 控件), Text 属性 (ActiveX 控件), RightToLeft 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), Font 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MouseIcon 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

Refill 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件)。

## 事件

DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), Change 事件 (ActiveX 控件), Click 事件 (ActiveX 控件), DblClick 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件), MouseDown, MouseUp 事件 (ActiveX 控件), MouseMove 事件 (ActiveX 控件)。

## 请参阅

DataList 控件, DBCombo 控件, 标准 ActiveX 控件, 使用 DataCombo 和 DataList 控件。

## DataList 控件

DataList 控件是一个数据绑定列表框, 它自动地由一个附加数据源中的一个字段充填, 并且可选择地更新另一个数据源中一个相关表的一个字段。

## 语法

DataList

## 说明

DataList 控件与 DBList 控件代码兼容，但是，DataList 控件被优化来同 ActiveX Data Objects (ADO) 一起工作。

**发行须知：**DataList 控件和 DataCombo 控件一起可以在文件 Msdatlst.ocx 中找到。要在应用程序中使用这一控件，必须把 .OCX 文件添加到工程中。当发布您的应用程序时，要把文件 Msdatlst.ocx 安装到用户的 Microsoft Windows System 或 System32 目录下。关于如何向一个 Visual Basic 工程添加 ActiveX 控件的详细信息，请参阅“标准 ActiveX 控件”。

## 属性

DataList, BoundColumn 属性, ListField 属性, RowSource 属性, SelectedItem 属性, VisibleItems 属性, VisibleCount 属性, MatchedWithList 属性, MatchEntry 属性, Locked 属性 (DataCombo, DataList 控件), RowMember 属性, BoundText 属性, DataMember 属性, DataFormat 属性, DataBindings 属性, Height, Width 属性, Left, Top 属性, TabIndex 属性, Tag 属性, Visible 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, TabStop 属性, Enabled 属性, HelpContextID 属性, Index 属性 (控件矩阵), Name 属性, Parent 属性, Container 属性, Object 属性, ToolTipText 属性, DataChanged 属性, DataField 属性, DataSource 属性, IntegralHeight 属性, WhatsThisHelpID 属性, OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), Text 属性 (ActiveX



控件），RightToLeft 属性（ActiveX 控件），Appearance 属性（ActiveX 控件），BackColor, ForeColor 属性（ActiveX 控件），Font 属性（ActiveX 控件），hWnd 属性（ActiveX 控件），MouseIcon 属性（ActiveX 控件），MousePointer 属性（ActiveX 控件）。

## 方法

Refill 方法，SetFocus 方法，Drag 方法，Move 方法，ZOrder 方法，OLEDrag 方法，ShowWhatsThis 方法，Refresh 方法（ActiveX 控件）。

## 事件

DragDrop 事件，DragOver 事件，GotFocus 事件，LostFocus 事件，Validate 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Click 事件（ActiveX 控件），DblClick 事件（ActiveX 控件），KeyDown, KeyUp 事件（ActiveX 控件），KeyPress 事件（ActiveX 控件），MouseDown, MouseUp 事件（ActiveX 控件），MouseMove 事件（ActiveX 控件）。

## 请参阅

DataCombo 控件，DBList 控件，标准 ActiveX 控件，使用 DataCombo 和 DataList 控件。

## DBCombo 控件

DBCombo 控件是带有下拉列表框的与数据相连的组合框，它能自动从与它相连的 Data 控件的字段中移居，也可以有选择地更新其它 Data 控件中相关表的字段。DBCombo 的文本框部分能用来编辑选定的字段。

### 语法

#### DBCombo

### 说明

DBCombo 控件和标准 ComboBox 控件不同。ComboBox 控件的列表用 AddItem 方法填加数据项，而 DBCombo 控件由和它相连的 Data 控件的 Recordset 对象中的字段中的数据自动填加数据项。标准 ComboBox 控件必须用 AddItem 方法手工移居。另外，DBCombo 控件有能力更新驻留在不同的 Data 控件中的相关的 Recordset 对象的字段。

DBCombo 控件支持自动查找模式，不用附加代码能迅速在列表中定位数据项。

下面是一系列属性，用于填充、管理 DBCombo 控件以及绑定选定数据和 Data 控件。

| 属性           | 说明  |
|--------------|---|
| DataSource   | 作出选择后更新的 Data 控件名   |
| DataField    | 由 DataSource 属性指定的在 Recordset 中更新的字段名   |
| RowSource    | 控件列表区字段的作为项目源使用的 Data 控件名   |
| ListField    | 由 RowSource 指定的在 Recordset 中的字段名以填充下拉列表。DBCombo 不支持 Listfield 属性的 LongBinary 型的字段                   |
| BoundColumn  | 由 RowSource 指定的在 Recordset 中的 Field 名，当选择确定后回传到 DataField。DBCombo 不支持 BoundColumn 的 LongBinary 型的字段 |
| BoundText    | BoundColumn 字段的文本值。当选择确定后，该值被回传以更新由 DataSource 和 DataField 属性指定的 Recordset 对象                       |
| Text         | 在列表框中选定项目的文本值   |
| MatchEntry   | 在运行时当用户键入字符时如何查找列表  |
| SelectedItem | 由 RowSource 属性指定的记录集中选定项目的书签  |
| VisibleCount | 在列表中可见的项目数目（全部或部分）  |
| VisibleItems | 一组书签，具有和 VisibleCount 属性相等的项目的最大数   |

用户能通过在控件的文本框部分键入值查找 DBCombo 控件。一旦键入，该值被放入列表，当前列表项目设置成该项。如果没有找到项目，BoundText 属性设成 null。

**注意：**如果未使控件的边界大到至少是下拉式列表的一行，在运行时列表不会显示出来。

## 属性

DBCombo, BoundColumn 属性, ListField 属性, RowSource 属性, SelectedItem 属性, VisibleItems 属性, VisibleCount 属性, MatchedWithList 属性, MatchEntry 属性, BoundText 属性, DataFormat 属性, RightToLeft 属性, DataBindings 属性, OLEDragMode 属性, OLDDropMode 属性, BackColor, ForeColor 属性, Height, Width 属性, Left, Top 属性, TabIndex 属性, DragIcon 属性, DragMode 属性, hWnd 属性, MouseIcon 属性, SelLength, SelStart, SelText 属性, Style 属性, TabStop 属性, Appearance 属性, Enabled 属性, HelpContextID 属性, Name 属性, Parent 属性, Font 属性, Container 属性, ToolTipText 属性, DataChanged 属性, DataField 属性, DataSource 属性, IntegralHeight 属性, WhatsThisHelpID 属性, SelLength, SelStart, SelText 属性 (ActiveX 控件), Text 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible (ActiveX 控件), Object 属性 (ActiveX 控件), RightToLeft 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

Refill 方法, Aboutbox 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, OLEDrag 方法, ShowWhatsThis 方法, Refresh 方法 (ActiveX 控件)。

## 事件

Click 事件 (ActiveX 控件), DblClick 事件 (ActiveX 控件), Change 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件), LostFocus 事件, MouseDown, MouseUp 事件 (ActiveX 控件), MouseMove 事件 (ActiveX 控件), Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件)。

请参阅

DataCombo 控件, DBList 控件, DataList 和 DataCombo 控件常量, Data 控件, UpdateRecord 方法, ComboBox 控件。

## DBList 控件

DBList 控件是和数据相连的列表框，它能自动从与之相连的 Data 控件的字段中移居，并有选择地更新其它 Data 控件中相关表的字段。

语法

DBList

# 说明

DBList 控件和标准 ListBox 控件不同。ListBox 控件的列表用 AddItem 方法填加数据项，而 DBList 控件由和它相连的 Data 控件的 Recordset 对象中的字段中的数据自动填加数据项。标准 ListBox 控件用 AddItem 方法手工移居。另外，DBList 控件有能力更新在不同的 Data 控件的相关的 Recordset 对象中的字段。

DBList 控件支持自动查找模式，不用附加代码能迅速在列表中定位数据项。

下面是一系列属性，用于填充、管理列表和把选定数据和 Data 控件绑定。

| 属性          | 说明  |
|-------------|---|
| DataSource  | 当选择确定时的被更新的 Data 控件名  |
| DataField   | 由 DataSource 属性指定的在 Recordset 中更新的字段名   |
| RowSource   | 作为控件列表区段的项目源使用的 Data 控件名  |
| ListField   | 由 RowSource 指定的在 Recordset 中的字段名，用于填充列表。<br>DBList 不支持 Listfield 属性的 LongBinary 型的字段            |
| BoundColumn | 由 RowSource 指定的在 Recordset 中的字段名，当选择确定后传回到 DataField。DBList 不支持 BoundColumn 属性的 LongBinary 型的字段 |
| BoundText   | BoundColumn 字段的文本值。当选择确定后，该值被回传以更新由 DataSource 和 DataField 属性指定的 Recordset 对象                   |

续表

|              |  |
|--------------|--|
| Text         | 在列表中选定项目的文本值                           |
| MatchEntry   | 在运行时当用户键入字符时如何查找列表                     |
| SelectedItem | 由 RowSource 属性指定的在 Recordset 中的选定项目的书签 |
| 属性           | 说明                                     |
| VisibleCount | 在列表中可见的项目数目（全部或部分）                     |
| VisibleItems | 一组书签，具有和 VisibleCount 属性相等的项目的最大数      |

和用 Data 控件改变当前记录一样，如果 BoundText 属性和由 DataSource 和 DataField 属性指定的字段的值相等，那么 DBList 控件会自动高亮显示列表中的某一项。

### 属性

DBList, BoundColumn 属性, ListField 属性, RowSource 属性（ActiveX 控件）, SelectedItem 属性, VisibleItems 属性, VisibleCount 属性, MatchedWithList 属性, MatchEntry 属性, BoundText 属性, DataFormat 属性, DataBindings 属性, Height, Width 属性（ActiveX 控件）, Left, Top 属性（ActiveX 控件）, TabIndex 属性, Tag 属性（ActiveX 控件）, Visible 属性（ActiveX 控件）, DragIcon 属性, DragMode 属性, TabStop 属性, Enabled 属性, HelpContextID 属性, Index 属性（ActiveX 控件）, Name 属性, Parent 属性, Container 属性, Object 属性（ActiveX 控件）, ToolTipText 属性, DataChanged 属性, DataField 属性, DataSource 属性, IntegralHeight 属性, WhatsThisHelpID 属性, OLEDragMode 属性, OLEDropMode 属性, Text 属性（ActiveX 控件）, RightToLeft 属性, Appearance

属性, BackColor, ForeColor 属性, Font 属性, hWnd 属性, MouseIcon 属性, MousePointer 属性 (ActiveX 控件)。

## 方法

Refill 方法, AboutBox 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, OLEDrag 方法(ActiveX 控件), ShowWhatsThis 方法, Refresh 方法(ActiveX 控件)。

## 事件

DragDrop 事件, DragOver 事件, GotFocus 事件, KeyDown,KeyUp 事件, KeyPress 事件, MouseDown,MouseUp 事件, MouseMove 事件, LostFocus 事件, KeyDown 事件, OLECompleteDrag 事件, OLEDragDrop 事件, OLEDragOver 事件, OLEGiveFeedback 事件, OLESetData 事件, OLEStartDrag 事件 (ActiveX 控件), Click 事件 (ActiveX 控件), DblClick 事件, KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件)。

## 请参阅

DataList 控件, DataCombo 控件, DataList 和 DataCombo 控件常量, Data 控件, UpdateRecord 方法。



## AboutBox 方法

显示控件的“关于”对话框。

应用于

MSChart 控件，DataGrid 控件。

语法

*object*.AboutBox

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

这与在属性窗口单击“关于”相同。

## BoundColumn 属性

返回或设置一个 Recordset 对象的源字段的名称，该 Recordset 对象用来为另一个 Recordset 提供数据值。

应用于

DataCombo 控件，DataList 控件，DBCombo 控件，DBList 控件。

## 语法

*object*.BoundColumn [= *value*]

BoundColumn 属性的语法包含以下部分：

| 部件            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象  |
| <i>value</i>  | 一个字符串表达式，它指定了由 RowSource 属性指定的 Data 控件创建的 Recordset 中的一个字段的名称 |

## 说明

通常，在使用 DataList 控件和 DataCombo 控件时，要用两个 Data 控件；一个用来填充由 Listfield 和 RowSource 属性指定的列表，另一个用来更新由 DataSource 和 DataField 属性指定的数据库中的字段。

ListField 属性指定用于填充列表的字段。由 DataSource 属性指定的第二个 Data 控件管理一个包含待更新字段的 Recordset。一旦用户选定了列表中的一项，由 BoundColumn 属性指定的字段就被送到由 DataSource 和 DataField 属性指定的第二个 Data 控件的字段中。这样，当选中一项时，就可以指定一个字段来填充列表，而使用另一个字段（同一 Recordset 中）来将数据传送到由 DataSource 和 DataField 属性指定的 Recordset 中。

如果在 Recordset 中找不到由 BoundColumn 属性指定的字段，就会产生一个可以捕获的错误。

# 数据类型

String

请参阅

Recordset 对象, FindDialog 对话框 (VisData), Table 窗体 (VisData), Zoom 对话框 (Vis Data), ListField 属性, Data 控件, DataField 属性, DataSource 属性, RecordSource 属性, DataSource 属性 (ActiveX 控件)。

## BoundText 属性

返回或设置由 BoundColumn 属性指定的字段的值。

应用于

DataCombo 控件, DataList 控件, DBCombo 控件, DBList 控件。

语法

*object*.BoundText [= *value*]

BoundText 属性的语法包含以下部分：

| 部件            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 一个指定数据值的字符串表达式       |

## 说明

在用 **DataList** 控件或 **DataCombo** 控件进行选择后，**BoundText** 属性包含 **BoundColumn** 属性的字段值。如果选定列表中的一项，则对于由 **DataSource** 属性所指定的 **Data** 控件，该项是可用的。被选定的内容也显示在 **DataCombo** 控件的文本框部分中，在那里可以对它进行编辑。如果在文本框部分输入一个值，列表部分将试图定位到某个匹配项目。如果匹配成功，则在 **BoundColumn** 属性的字段值的基础上设置 **BoundText** 属性。如果匹配失败，**BoundText** 则被设置为 **Null**。

可以使用 **BoundText** 属性值创建一个可以用来找到指定记录的查询：

```
Dim strQ As String
```

```
StrQ = "Select * From Products Where SupplierID = " & DataList1.BoundText
```

' 使用带有 ADO Data 控件的新查询返回一个新的记录集。

```
Adodc1.RecordSource = strQ
```

把 **DataSource** 属性所指定的 **Data** 控件定位到一个新的记录将会把 **BoundText** 属性设置为由 **DataField** 指定的值。然后，**DataList** 控件或 **DataCombo** 控件在列表中查找记录，查看 **BoundText** 的值是否和 **BoundColumn** 属性中的字段值相匹配。如果匹配成功，列表中的记录被强调表示或者被放入 **DataCombo** 控件的文本框部分。

## 数据类型

**String**

请参阅

BoundColumn 属性, ListField 属性, BoundText 属性, Data 控件, DataSource 属性, DataSource 属性 (ActiveX 控件)。

## Click 事件 (DataCombo 控件)

当用户在 DataCombo 控件上按下并释放鼠标键时发生。该事件也可以通过在键盘上按上下箭头键选择项目触发。

应用于

DBCombo 控件。

语法

```
Private Sub object_Click( [index As Integer ,] Area As Integer )
```

Click 事件的语法包括这些部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象    |
| <i>index</i>  | 在控件数组中唯一标识一个控件的整数       |
| <i>area</i>   | 整数表达式，如下所示，它指明了定在何处单击控件 |

Area 参数可包含如下值:

| 常量            | 值 | 描述                         |
|---------------|---|----------------------------|
| dbcAreaButton | 0 | 用户单击 DataCombo 控件上的按钮      |
| dbcAreaEdit   | 1 | 用户单击 DataCombo 控件上的文本框部分   |
| dbcAreaList   | 2 | 用户单击 DataCombo 控件上的下拉列表框部分 |

### 说明

DataCombo 控件不像标准组合框那样有 DropDown 事件, 在用户下拉 DataCombo 控件的下拉列表框部分时发出信号。用户能在 Click 事件的 Area 参数中得知它单击了 DataCombo 控件的哪个部分。

一般地, 把 Click 事件的过程和一个控件联系起来以执行命令或类似命令的动作。

单击控件产生除 Click 事件外的 MouseDown 和 MouseUp 事件。当把处理事件的过程和相关的事件联系起来时, 要确保这些动作不冲突。如果事件的顺序对应用程序重要, 测试控件以确定事件顺序。

**注意:** 使用 MouseDown 和 MouseUp 事件来区分用户按下的是鼠标的左, 右还是中键。

### 请参阅

DbClick 事件(DataCombo 控件), Data 控件, DropDown 事件, MouseDown,

MouseUp 事件。

## DblClick 事件（DataCombo 控件）

当用户在 DataCombo 控件上双击鼠标键时发生。

应用于

DBCombo 控件。

语法

Private Sub *object*\_DblClick ( [*index* As Integer,] *Area* As Integer)

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象  |
| <i>index</i>  | 在控件数组中唯一标识一个控件的整数     |
| <i>Area</i>   | 整数表达式，如下所示，它指明在何处双击控件 |

Area 参数可包含如下值：

| 常量            | 值 | 描述                       |
|---------------|---|--------------------------|
| dbcAreaButton | 0 | 用户双击 DataCombo 控件上的按钮    |
| dbcAreaEdit   | 1 | 用户双击 DataCombo 控件上的文本框部分 |

续表

|             |   |   |
|-------------|---|---|
| dbcAreaList | 2 | 用户双击 DataCombo 控件上的下拉列表部分<br>(只发生在 Style 属性设置成 1 时) |
|-------------|---|---|

## 说明

可以对一个隐含的操作使用 **DbClick** 事件过程或用它执行具有单一操作的一系列步骤。

如果 **DbClick** 没有发生在系统的双击时间限制内，对象将识别第二次 **Click** 事件。双击时间限制可能会有不同，因为用户能通过“控制面板”设置双击速度。

**注意：**为了区别鼠标的左，右和中键，用 **MouseDown** 和 **MouseUp** 事件。

## ListField 属性

返回或设置 **Recordset** 对象中的字段名，这个对象由 **RowSource** 属性指定，用于填充 **DataCombo** 控件或 **DataList** 控件的列表部分。

## 应用于

**DataCombo** 控件，**DataList** 控件，**DBCombo** 控件，**DBList** 控件。



# 语法

```
object.ListField [= value]
```

ListField 属性的语法包含以下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                              |
| <i>value</i>  | 一个字符串表达式，指定了由 RowSource 属性指定的 Recordset 中的一个字段的名称 |

# 说明

ListField 属性允许选择 Recordset 中的字段来填充控件的列表部分。这个属性和 RowSource 属性结合使用，后者指出用哪个 Data 控件创建用来填充列表的 Recordset。

一般要使用两个数据觉察的 Recordset 对象。一个 Recordset 包含合法选定内容的一个只读列表，而另一个 Recordset 则由列表中选中的内容更新。例如，DataList 控件可以由一个返回合法部件数列表及其描述的查询产生。ListField 属性指向 Recordset 的描述字段使用户看不到实际的部件数。BoundColumn 属性指向部件数字段，因为这是 Recordset 中需要更新的字段。

如果在 Recordset 中找不到由 ListField 属性指定的字段，就会产生一个可以捕获的错误。

## 数据类型

String

请参阅

Recordset 对象, Table 对话框 (VisData), Table 窗体 (VisData), Zoom 对话框 (VisData), BoundColumn 属性, RowSource 属性, Data 控件, RecordSource 属性。

## Locked 属性 (DataCombo、DataList 控件)

返回或设置一个值, 指示对象中的任何数据是否可以被修改。

应用于

DataCombo 控件, DataList 控件。

语法

*object*.Locked [= *boolean*]

Locked 属性的语法包含如下部分:

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象       |
| <i>boolean</i> | 可选的。一个布尔表达式，指示数据是否可以更改。如设置值中所述 |

## 设置

`boolean` 的设置值如下：

| 设置值   | 描述           |
|-------|--------------|
| False | （缺省值）数据可以被更改 |
| True  | 数据不能被更改      |

## MatchedWithList 属性

如果当前的 `BoundText` 属性的内容与控件列表部分中的一个记录匹配，则返回 `True`。

## 应用于

`DataCombo` 控件，`DataList` 控件，`DBCombo` 控件，`DBList` 控件。

# 语法

*object*.MatchedWithList

**object** 置换元代表一个对象表达式，其值是“应用于”列表中的对象。

# 返回值

**MatchedWithList** 属性的返回值如下：

| 值     | 描述                                 |
|-------|------------------------------------|
| True  | <b>BoundText</b> 属性中的内容与列表中的一个记录匹配 |
| False | <b>BoundText</b> 属性中的内容与列表中记录都不匹配  |

# 说明

在 **DataCombo** 控件的文本部分输入一个数值时，如果输入的数值是列表中显示的一项，那么 **MatchedWithList** 属性被设置为 **True**。在移动由 **DataCombo** 控件或 **DataList** 控件的 **DataSource** 属性指定的 **Data** 控件时，如果 **BoundText** 的值与列表中的一个记录匹配，那么 **MatchedWithList** 属性也会被设置为 **True**。在这种情况下，这个记录被强调表示。

# 数据类型

**Boolean**

请参阅

BoundText 属性，Data 控件，DataSource 属性，DataSource 属性（ActiveX 控件）。

## MatchEntry 属性

返回或设置一个值，它指示 DataCombo 控件或 DataList 控件如何在用户输入的基础上执行查找。

应用于

DataCombo 控件，DataList 控件，DBCombo 控件，DBList 控件。

语法

*object*.MatchEntry [= *value* ]

MatchEntry 属性的语法包含以下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                            |
| <i>value</i>  | 一个常量或值，当控件获得焦点且用户输入一个或多个字符时这个值定义控件的行为，这点正如设置中所述 |

# 设置

value 的设置情况如下：

| 设置                   | 数值 | 描述  |
|----------------------|----|---|
| dblBasic<br>Matching | 0  | 基本匹配：（缺省）对于使用列表输入项的第一个字母输入字符，该控件查找下一个匹配。重复键入相同的字母可以循环浏览列表中所有以那个字母开头的输入项 |
| dblExtendedMatching  | 1  | 扩展匹配：控件查找与输入的所有字符相匹配的项。键入字符后，查找就完成了，键入更多的字符可以使查找更准确                     |

# 说明

当 MatchEntry 属性被设置为 dblExtendedMatching 且用户输入了一个 backspace 键或等待数秒钟之后，匹配的字符串被重新设置。

# 数据类型

Integer

请参阅

MatchedWithList 属性，Data 控件。

## Refill 方法

再次创建 DataList 或 DataCombo 控件的列表并强制刷新。

应用于

DataList 控件，DBCombo 控件，DBList 控件。

语法

*object*.Refill

object 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

Refill 方法和标准的 Refresh 方法不同，它仅仅强制一个 Repaint 事件。

请参阅

Data 控件，Paint 事件，Refresh 方法，Requery。

## RowMember 属性

返回或设置用于显示列表文本的数据成员。

应用于

DataCombo 控件，DataList 控件。

语法

*object*.RowMember [= *string*]

RowMember 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>string</i> | 可选的。数据成员名                |

说明

RowMember 属性等价于 DataMember 属性，并且用于完全限定绑定到哪个数据集合。当一个数据源提供多于一个数据成员时，在设置 DataField 属性之前必须指定要使用哪一个数据成员。

请参阅

DataMember 属性（ActiveX 控件），DataSource 属性（ActiveX 控件）。



## RowSource 属性

设置一个指定 Data 控件的值，DataList 控件和 DataCombo 控件的列表由这个 Data 控件填充。运行时不可用。

### 应用于

DataCombo 控件，DataList 控件，DBCombo 控件，DBList 控件。

### 语法

*object*.RowSource [= *value*]

RowSource 属性的语法包含以下部分：

| 部件            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 指定 Data 控件名称的字符串表达式  |

### 说明

要填充 DataCombo 控件或 DataList 控件中的列表，必须在设计时使用 Properties 窗口在 RowSource 属性中指定一个 Data 控件。

要与 Data 控件管理的 Recordset 对象中的字段相结合，必须在 ListField 属性中提供 Field 对象的名称。

## 数据类型

String

请参阅

Field 对象, Add Field 对话框(VisData), Table 窗体(VisData), Table Structure 对话框 (VisData), Recordset 对象, Find 对话框 (VisData), Zoom 对话框 (VisData), BoundColumn 属性, ListField 属性, Data 控件, RecordSource 属性。

## SelectedItem 属性

返回一个值，包含 DataCombo 控件或 DataList 控件中选中的记录的书签。

应用于

DataCombo 控件, DataList 控件, DBCombo 控件, DBList 控件。

语法

*object*.SelectedItem

object 置换元代表一个对象表达式，其值是“应用于”列表中的对象。

## 说明

当选定控件列表部分中的一项时，**SelectedItem** 属性包含一个可以用来重新定位到由 **RowSource** 属性指定的 **Data** 控件的 **Recordset** 中的选定记录的书签。

## 数据类型

**Varaint**

## 请参阅

**Recordset** 对象, **Find** 对话框 (**VisData**), **Table** 窗体, **Zoom** 对话框 (**VisData**), **RowSource** 属性, **Data** 控件。

## Style 属性 (DataCombo 控件)

返回或设置一个值，指定控件的行为和/或外观。

## 应用于

**DataCombo** 控件。

## 语法

*object*.Style [= *integer*]

Style 属性的语法包含如下部分：

| 部分             | 描述                          |
|----------------|-----------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象    |
| <i>integer</i> | 可选的。一个数值表达式，决定控件的风格。如设置值中所示 |

设置

*integer* 的设置值如下：

| 常量                            | 值 | 描述   |
|-------------------------------|---|--|
| <code>dbcDropdownCombo</code> | 0 | （缺省的）下拉式组合框。包括一个下拉列表和一个文本框。用户可以从列表中选择或是在文本框中输入                             |
| <code>DbcSimpleCombo</code>   | 1 | 简单组合框。包括一个文本框和一个列表，列表不能下拉。用户可以从列表中选择或在文本框中输入。增加 <b>Height</b> 属性来显示列表的更多内容 |
| <code>DbcDropDownList</code>  | 2 | 下拉列表。这一风格只允许从下拉列表中选择   |

## VisibleCount 属性

返回一个值，它表示 `DataCombo` 控件或 `DataList` 控件的列表部分中可见项的数目。

应用于

DataCombo 控件，DataList 控件，DBCombo 控件，DBList 控件。

语法

*object.VisibleCount*

**object** 置换元代表一个对象表达式，其值是“应用于”列表中的对象。

说明

**VisibleCount** 属性返回一个范围在 0 到控件中可见项数之间的整数。当 **IntegralHeight** 属性被设置为 **False** 时，即使一项中的文本只是部分可见，这一项也被看作是可见的。

**注意：**在第一次显示 DataCombo 控件的列表部分之前，**VisibleCount** 属性可能被设置为 0。

数据类型

**Integer**

请参阅

**VisibleItems** 属性，Data 控件，**IntegralHeight** 属性。

## 示例

下面的示例代码使用 `VisibleCount` 和 `VisibleItems` 属性显示 `DataList` 控件中所有可见记录中的字段：

```
Private Sub Command1_Click()  
    Dim I As Integer, fld As Field, msg As Variant  
  
    For I = 0 To DataList1.VisibleCount - 1  
        Data1.Recordset.Bookmark = DataList1.VisibleItems(I)  
        msg = ""  
        For Each fld In Data1.Recordset.Fields  
            msg = msg & fld.Value & "-"  
        Next  
        MsgBox msg  
    Next I  
  
End Sub
```

## VisibleItems 属性

返回书签数组，每个书签对应 `DataCombo` 控件或 `DataList` 控件的列表中的一个可见项。

应用于

DataCombo 控件，DataList 控件，DBCombo 控件，DBList 控件。

语法

*object.VisibleItems(Index)*

VisibleItems 属性的语法包含以下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                      |
| <i>index</i>  | 指定数组元素的整型表达式。这个值在 0 到 VisibleCount - 1 之间 |

说明

这些书签可以用来在用于填充列表的 recordset 中获取个别记录。

数据类型

Varaint

请参阅

Bookmark 属性（DataGrid），VisibleCount 属性，Data 控件。

## DataRepeater 控件

**DataRepeater** 控件的功能是作为数据绑定用户控件的可滚动的容器。每一个控件都作为“重复的”控件出现在自己所在的行里，使用户能够一次浏览多个数据绑定用户控件。

### 语法

**DataRepeater**

### 说明

要使用 **DataRepeater** 控件，必须首先创建一个数据绑定用户控件，然后将其编译为 .ocx 文件。创建用户控件后，必须按照下面的基本步骤进行：

- 1、用 **Components** 对话框将用户控件添加到工程。这样就可以确保 **Visual Basic** 程序包和开发向导能够包含正确的文件。而且也允许对控件的公共属性、事件和方法进行访问。
- 2、在 **Properties** 窗口中单击 **RepeatedControlName**，然后在下拉列表中选择用户控件。
- 3、将数据源（例如 **ADO Data Control**）添加到窗体并同数据提供者连接。
- 4、将 **DataRepeater** 控件的 **DataSource** 属性设置为数据源。



5、在 DataRepeater 控件上单击右键，然后单击 DataRepeater Properties。

6、单击 RepeaterBindings 选项卡。

7、将 PropertyName 设置为适当的 DataField，然后单击 Add 按钮。

DataRepeater 控件一次只显示一个用户控件（活动控件），从而节约了计算机资源。所显示的其他控件只是简单的图像，并不保持同数据源的个别连接，如同一个窗体中包含多个用户控件所发生的情况一样。

**发布须知** 当使用 DataRepeater 控件创建并发布您的应用程序时，应当在用户的 Microsoft Windows System 或 System32 子目录中安装 MSDatRep.ocx 文件。Visual Basic 附带的安装工具提供了一些工具，能够帮助您编写安装程序，从而正确地安装应用程序。

## 属性

DataRepeater, RepeatedControl 属性, RepeatedControlName 属性, RepeaterBindings 属性, CaptionStyle 属性, RowDividerStyle 属性 (DataRepeater 控件), RowIndicator 属性, RecordCount 属性, ActiveRow 属性, IntegralHeight 属性 (DataRepeater 控件), VisibleRecords 属性, DataFields 属性, Scrollbars 属性 (DataRepeater 控件), VisibleRows 属性 (DataRepeater 控件), CurrentRecord 属性, PropertyNames 属性, DataBindings 属性, Height, Width 属性, Left, Top 属性, TabIndex 属性, Tag 属性, Visible 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, TabStop 属性, Enabled 属性, HelpContextID 属性, Index 属性 (控件矩阵), Name 属性, Parent 属性, Container 属性, Object 属

性, ToolTipText 属性, WhatsThisHelpID 属性, RightToLeft 属性 (ActiveX 控件), DataMember 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Caption 属性 (ActiveX 控件), DataSource 属性 (ActiveX 控件), Font 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MouseIcon 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, Refresh 方法 (ActiveX 控件), Clear 方法 (ActiveX 控件)。

## 事件

ActiveRowChanged 事件, VisibleRecordChanged 事件, DataUpdate 事件, CurrentRecordChanged 事件, RepeatedControlLoaded, RepeatedControlUnloaded 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, Validate 事件, Click 事件 (ActiveX 控件), DblClick 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件), MouseDown, MouseUp 事件 (ActiveX 控件), MouseMove 事件 (ActiveX 控件)。

## 请参阅

RepeaterBinding 对象, RepeaterBindings 集合, Add 方法 (RepeaterBindings 集合), 创建 ActiveX 控件, 构建 ActiveX 控件。

## ActiveRow 属性

返回或设置定位当前记录的行索引。当设置该属性时，ActiveRowChanged 事件将发生。

应用于

DataRepeater 控件。

语法

*object.ActiveRow* [=integer]

| 部分             | 描述                     |
|----------------|------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的对象 |
| <i>integer</i> | 一个数字表达式，指出当前记录所在的行     |

说明

该属性的值必须在 1 和 VisibleRows 属性值之间。

行索引是基于 1 的，且不允许将属性设置为 0。然而，当返回 0 时，表明当前的记录已滚动为不可见的了。

当 DataRepeater 控件显示一个记录集的开头或结尾时，有可能 ActiveRow 没有逻辑设置值。例如，如果当前的记录是记录集中的最后一条时，当

VisibleRows 是 3 时，将 ActiveRow 设置为 2 或 1 就是无效设置。

请参阅

ActiveRowChanged 事件，VisibleRows 属性（DataRepeater 控件）。

## ActiveRowChanged 事件

当设置 ActiveRow 属性时发生。

应用于

DataRepeater 控件。

语法

*object*\_ActiveRowChanged( )

object 置换元是一个对象表达式，其值是“应用于”列表中的对象。

请参阅

ActiveRow 属性。

## Add 方法（RepeaterBindings 集合）

添加一个 RepeaterBinding 对象到 RepeaterBindings 集合中，并返回对新对象的引用。

应用于

RepeaterBindings 集合。

语法

*object.Add(PropertyName, DataField, DataFormat, key)*

| 部分                  | 描述   |
|---------------------|--|
| <i>object</i>       | 一个对象表达式，其值是“应用于”列表中的对象                           |
| <i>propertyName</i> | 必需的。设置用户控件数据绑定属性的一个字符串                           |
| <i>DataField</i>    | 必需的。一个字符串，设置要绑定到 <b>PropertyName</b> 中指定属性的数据源字段 |
| <i>DataFormat</i>   | 可选的。设置要使用的 <b>DataFormat</b> 对象                  |
| <i>key</i>          | 可选的。一个唯一的字符串，用来标识对象。使用该值可以检索集合中的指定成员             |

请参阅

DataRepeater 控件。

## 示例

下列示例首先打印 `RepeaterBindings` 集合的当前属性名称，然后将 `DataBinding` 对象添加到集合中，最后改变对象 `DataBinding` 的格式。

```
Private Sub DataRepeater1_RepeatedControlLoaded()  
    Dim rb As RepeaterBinding  
  
    For Each rb In DataRepeater1.RepeaterBindings  
        Debug.Print rb.PropertyName ' 打印所有属性名。  
    Next  
  
    With DataRepeater1  
        ' 添加一个新的 RepeaterBinding 对象。  
        .RepeaterBindings.Add "cleared", "cleared"  
        ' 将格式改为全部大写。  
        .RepeaterBindings(2).DataFormat.Format = ">"  
    End With  
End Sub
```

## CaptionStyle 属性

返回或设置一个值，决定标题的样式，包括对齐和可见性。

应用于

DataRepeater 控件。

语法

*object.CaptionStyle [=integer]*

CaptionStyle 属性的语法包含下面几部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的对象       |
| <i>integer</i> | 一个数字表达式，决定标题的对齐形式，如下面的“设置”所示 |

设置

| 常量              | 值 | 描述        |
|-----------------|---|-----------|
| drpNoCaption    | 0 | 标题隐藏      |
| drpLeftAligned  | 1 | （缺省）标题左对齐 |
| 常量              | 值 | 描述        |
| drpRightAligned | 2 | 标题右对齐     |
| drpCentered     | 3 | 标题居中      |

请参阅

Caption 属性（ActiveX 控件）。

## CurrentRecord 属性

返回当前记录的书签。

应用于

DataRepeater 控件。

语法

*object*.CurrentRecord

object 置换元是一个对象表达式，其值是“应用于”列表中的一个对象。

返回类型

Variant

请参阅

VisibleRecords 属性， VisibleRows 属性 (DataRepeater 控件)，  
CurrentRecordChanged 事件。

示例

该示例使当前记录成为第一个可见记录。如果用户滚动 DataRepeater 控件，以使当前记录隐藏，则调用过程将会使当前记录重新出现。



```
Private Sub MakeFirstVisibleRecord()  
    DataRepeater1.VisibleRecords(1)=DataRepeater1.CurrentRecord  
End Sub
```

## CurrentRecordChanged 事件

当前记录更改为另一个不同的记录时发生。

应用于

DataRepeater 控件。

语法

```
Private Sub object_CurrentRecordChanged()
```

*object* 置换元是一个对象表达式，其值是“应用于”列表中的一个对象。

说明

用户能够通过单击控件中的任何记录、或在数据控件中使用向后、向前、移到第一和移到最后按钮来更改当前记录。

请参阅

CurrentRecord 属性。

### 示例

该示例使用 `CurrentRecord` 属性存储当前记录的书签。

## Option Explicit

### Option Base 1

### Private mRecords(10) As Variant ' 书签数组。

' 只储存后 10 条记录。

Private Sub DataRepeater1\_CurrentRecordChanged()

If intC = 10 Then intC = 1

```
mRecords(intC) = DataRepeater1.CurrentRecord
```

```
intC = intC + 1
```

End Sub

## DataField 属性 (RepeaterBinding 对象)

返回或设置要绑定到一个属性的 `DataField`。

应用于

RepeaterBindings 对象。

## 语法

*object.DataField* [=string]

DataField 属性的语法包含下面几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | 要绑定到的属性名                 |

请参阅

DataFields 属性。

## DataFields 属性

返回 DataSource 的 DataField 名的数组。

应用于

DataRepeater 控件。

## 语法

*object.DataFields(index)*

DataFields 属性的语法包含下面几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 数组中一个元素的索引。该数组是基于 0 的    |

返回类型

String

说明

DataFields 数组包含重复控件的所有可绑定属性名（请参阅 RepeatedControlName）。

使用 Ubound 方法来决定数组中有多少元素。因为数组是基于 0 的，所以使用下面的代码：

```
Debug.Print Ubound(DataRepeater1.DataFields) + 1
```

请参阅

DataField 属性（RepeaterBinding 对象）。

## DataFormat 属性（RepeaterBinding 对象）

返回或设置对 DataFormat 对象的一个引用。

应用于

RepeaterBinding 对象。

语法

*object*.DataFormat [=*dataformat*]

DataFormat 属性的语法包含下面几部分：

| 部分                | 描述                       |
|-------------------|--------------------------|
| <i>object</i>     | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>dataformat</i> | 可选的。一个对 DataFormat 对象的引用 |

请参阅

DataFormat 属性，DataFormats 属性。

## DataUpdate 事件

当通过程序编辑、或由用户编辑字段时发生。

应用于

DataRepeater 控件。

语法

Private Sub *object*\_DataUpdate(*cancel* As boolean, ByVal *Record* As Variant, ByVal *field* As String)

DataUpdate 事件的语法包含下面几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>cancel</i> | 一个布尔表达式，指定是否取消操作         |
| <i>record</i> | 一个变体型，包含当前记录的书签          |
| <i>field</i>  | 一个字符串，指示将要更改的字段名         |

说明

*field* 参数不一定是个绑定字段。如果记录集中的任何字段发生改变，则该事件发生。

IntegralHeight 属性 （DataRepeater 控件）

返回或设置一个值，决定控件是否显示部分行。

应用于

DataRepeater 控件。

语法

*object*.IntegralHeight [=*boolean*]

IntegralHeight 属性的语法包含下面几部分：

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象      |
| <i>boolean</i> | 一个布尔表达式，表明控件是否显示部分行，详见下面的“设置” |

设置

boolean 的设置值为：

| 常量    | 描述           |
|-------|--------------|
| True  | （缺省）控件只显示全部行 |
| False | 显示部分行        |

## PropertyName 属性（RepeaterBinding 对象）

返回或设置用户控件的一个可绑定属性的名称。

应用于

RepeaterBinding 对象。

语法

*object*.PropertyName [=string]

PropertyName 属性的语法包含下面几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | RepeatedControl 的一个可绑定属性 |

请参阅

PropertyNames 属性。

## PropertyNames 属性

返回重复控件属性名的数组。



应用于

DataRepeater 控件。

语法

*object.PropertyNames(index)*

PropertyNames 属性的语法包含下面几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 数组中一个元素的索引。该数组是基于 0 的    |

说明

使用 Ubound 方法来决定数组中有多少元素。因为数组是基于 0 的，故使用下列的代码：

```
Debug.Print Ubound(DataRepeater1.DataFields) + 1
```

请参阅

PropertyName 属性（RepeaterBinding 对象）。

## RecordCount 属性

返回记录集中的记录数。

应用于

DataRepeater 控件。

语法

*object*.RecordCount

*object* 置换元是一个对象表达式，其值是“应用于”列表中的对象。

返回类型

长整型。

## RepeatedControl 属性

返回对重复控件的一个引用。

应用于

DataRepeater 控件。

## 语法

*object*.RepeatedControl

**object** 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

## 说明

使用该引用设置公共属性或调用用户控件的公共函数。例如，下面的代码检索名称为 “Value” 的重复控件的公共属性。

```
MsgBox DataRepeater1.RepeatedControl.Value
```

## 请参阅

RepeatedControlName 属性, RepeatedControlLoaded, RepeatedControlUnloaded 事件。

## RepeatedControlLoaded, RepeatedControlUnloaded 事件

RepeatedControlLoaded — 在 RepeatedControl 完成创建和初始化后发生。

RepeatedControlUnLoaded — 在 RepeatedControl 将要卸载时发生。

## 应用于

DataRepeater 控件。

## 语法

```
Private Sub object_RepeatedControlLoaded()
```

```
Private Sub object_RepeatedControlUnloaded()
```

*object* 置换元是一个对象表达式，其值是“应用于”列表中的一个对象。

## 说明

运行时您可以通过将 `RepeatedControlName` 属性设置为一个新的程序 ID 来更改 `RepeatedControl`。在进行该操作时，使用 `RepeatedControlLoaded` 事件初始化用户控件的值。

## 请参阅

`RepeatedControl` 属性，`RepeatedControlName` 属性。

## RepeatedControlName 属性

返回或设置重复控件的程序 ID。

*object*.RepeatedControl [=string]

`RepeatedControl` 属性包含下面几部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的对象 |
| <i>string</i> | 用户控件的程序 ID             |

应用于

**DataRepeater** 控件。

说明

重复控件不是 **Visual Basic** 的内部控件。

控件的程序 ID 以 **projectname.classname** 的形式由其工程名和类名组成。**Visual Basic** 对象浏览器在工程/库对话框中显示工程名称，在类列表框中显示类名称。

请参阅

**RepeatedControl** 属性，**RepeatedControlLoaded**，**RepeatedControlUnloaded** 事件。

## RepeaterBinding 对象

**RepeaterBinding** 对象表示部件的一个可绑定属性。

## 语法

### RepeaterBinding

## 说明

运行时使用 **RepeaterBinding** 对象通过改变字段绑定方式改变控件 **DataRepeater** 的内容。

## 属性

**DataField** 属性（**RepeaterBinding** 对象），**DataFormat** 属性（**RepeaterBinding** 对象），**PropertyName** 属性（**RepeaterBinding** 对象），**Index** 属性（**ActiveX** 控件），**Tag** 属性（**ActiveX** 控件），**DataChanged** 属性，**Key** 属性。

## 请参阅

**RepeaterBindings** 属性，**RepeaterBindings** 集合。

## 示例

下列示例首先打印 **RepeaterBindings** 集合的当前属性名称，然后将 **DataBinding** 对象添加到集合中，最后改变对象 **DataBinding** 的格式。

```
Private Sub DataRepeater1_RepeatedControlLoaded()  
    Dim rb As RepeaterBinding  
  
    For Each rb In DataRepeater1.RepeaterBindings
```

Debug.Print rb.PropertyName ' 打印所有属性名。

Next

With DataRepeater1

' 添加一个新的 RepeaterBinding 对象。

.RepeaterBindings.Add "cleared", "cleared"

' 将格式改为全部大写。

.RepeaterBindings(2).DataFormat.Format = ">"

End With

End Sub

## RepeaterBindings 集合

RepeaterBinding 对象的集合。

语法

RepeaterBindings

说明

使用 RepeaterBindings 属性返回对集合的一个引用。

## 属性

Count 属性（ActiveX 控件）， Item 属性（ActiveX 控件）。

## 方法

Add 方法（RepeaterBindings 集合）， Remove 方法（ActiveX 控件）， Clear 方法（ActiveX 控件）。

## 请参阅

DataRepeater 控件， RepeaterBindings 属性， RepeaterBinding 对象。

## RepeaterBindings 属性

返回对 RepeaterBinding 对象的 RepeaterBindings 集合的引用。

## 应用于

DataRepeater 控件。

## 语法

*object*.RepeaterBindings

*object*.RepeaterBindings(*index*)



RepeaterBindings 属性的语法包含下面几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | Index 或 Key，指定集合中的一个成员   |

请参阅

RepeaterBinding 对象，RepeaterBindings 集合。

## RowDividerStyle 属性（DataRepeater 控件）

返回或设置一个值，决定行分隔线的外观。

应用于

DataRepeater 控件。

语法

*object*.RowDividerStyle [=integer]

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的对象       |
| <i>integer</i> | 一个数字表达式，决定行分隔线的外观，如下面的“设置”所示 |

## 设置

| 常量           | 值 | 描述         |
|--------------|---|------------|
| drpNoDivider | 0 | 行间无分隔线     |
| drpFlatLine  | 1 | （缺省）行间有分隔线 |
| drpInset     | 2 | 行间有插入线     |
| drpRaised    | 3 | 行间有凸起线     |

请参阅

RowIndicator 属性。

## RowIndicator 属性

返回或设置一个值，决定是否在控件上绘制行指示器（一个实心的黑三角）。指示器标明了活动记录。

应用于

DataRepeater 控件。

语法

*object*.RowIndicator [=*boolean*]

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的对象        |
| <i>boolean</i> | 一个布尔表达式，指出是否绘制行分隔线，如下面的“设置”所示 |

## 设置

`boolean` 的设置值为：

| 常量                 | 描述         |
|--------------------|------------|
| <code>False</code> | 不绘制行指示器    |
| <code>True</code>  | （缺省）绘制行指示器 |

## 请参阅

`RowDividerStyle` 属性（`DataRepeater` 控件）。

## Scrollbars 属性 （`DataRepeater` 控件）

返回或设置一个值，决定滚动条的样式。

## 应用于

`DataRepeater` 控件。

语法

*object.Scrollbars* [= *integer*]

Scrollbars 属性的语法包含下面几部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>integer</i> | 一个数字表达式，决定滚动条的外观，详见下面的“设置” |

设置

| 常量       | 值 | 描述          |
|----------|---|-------------|
| vbSBNone | 0 | 无           |
| vbHoriz  | 1 | 只有水平滚动条     |
| vbVert   | 2 | （缺省）只有垂直滚动条 |
| vbBoth   | 3 | 垂直和水平滚动条    |
| vbAuto   | 4 | 当需要时出现滚动条   |

VisibleRecords 属性

返回或设置要显示在指定行索引处记录的书签。

应用于

DataRepeater 控件。

语法

*object.VisibleRecords(rowindex) [=bookmark]*

VisibleRecords 属性的语法包含下面几部分：

| 部分              | 描述                                |
|-----------------|-----------------------------------|
| <i>object</i>   | 一个对象表达式，其值是“应用于”列表中的一个对象          |
| 部分              | 描述                                |
| <i>rowindex</i> | 必需的。一个长整型，其值指定在哪一行中显示记录           |
| <i>bookmark</i> | 可选的。设置将在 <i>rowindex</i> 行显示记录的标签 |

返回类型

Variant

说明

*rowindex* 的值不能超出记录集 EOF 或 BOF 的范围。

*rowindex* 值必须在 1 （顶端行）和 VisibleRows 属性值之间。

请参阅

`VisibleRows` 属性（`DataRepeater` 控件），`CurrentRecord` 属性。

## VisibleRecordChanged 事件

当 `VisibleRecords` 属性改变时发生。

应用于

`DataRepeater` 控件。

语法

*object*.`VisibleRecordChanged`()

*object* 替换元是一个对象表达式，其值是“应用于”列表中的一个对象。

## VisibleRows 属性（DataRepeater 控件）

返回控件中的完全可见行数。

应用于

`DataRepeater` 控件。

## 语法

*object.VisibleRows*

**object** 置换元是一个对象表达式，其值是“应用于”列表中的一个对象。

## 返回类型

长整型

## 说明

与 **VisibleRecords** 属性一起使用 **VisibleRow** 属性指定当前记录显示的位置。例如，要在最下行显示当前记录，使用下列代码：

```
DataRepeater1.VisibleRecords(DataRepeater1.VisibleRows) = _  
DataRepeater1.CurrentRecord
```

## 请参阅

**ActiveRow** 属性，**VisibleRecords** 属性，**CurrentRecord** 属性。

# DateTimePicker 控件

**DateTimePicker** 控件使您可以提供格式化的日期字段，使得进行日期选择很容易。另外，用户还可以从类似于 **MonthView** 控件的下拉式日历界面中选择日期。

## 语法

### DTPicker

## 说明

**DateTimePicker** 控件，有两种操作模式：

- 下拉式日历模式（缺省）— 允许用户显示一种能够用来选择日期的下拉式日历。
- 时间格式模式— 允许用户在日期显示中选择一个字段（例如：月、日、年等等），按下控件右边的上下箭头来设置它的值。

可以自定义控件的下拉式日历的外观。使用各种颜色属性，例如 **CalendarBackColor**, **CalendarForeColor**, **CalendarTitleBackColor**, **CalendarTitleForeColor** 和 **CalendarTrailingForeColor**，允许创建属于您自己的颜色方案。



可以使用键盘或鼠标对控件进行浏览。下拉式日历有两个按钮使您能够滚动月份数据出入视图。

**注意：**DateTimePicker 控件是 ActiveX 控件组的一部分，包含在 MSCOMCT2.OCX 文件中。要在应用程序中使用 DateTimePicker 控件，必须将 MSCOMCT2.OCX 文件加入到工程之中。发布您的应用程序时，要将 MSCOMCT2.OCX 文件装入到用户的 Microsoft Windows System 或 System32 目录下。有关如何将 ActiveX 控件添加到工程之中的更多信息，请参阅《Microsoft Visual Basic6.0 程序员指南》中的"Adding Controls to a Project"。

## 属性

CalendarBackColor , CalendarForeColor 属性 , CalendarTitleBackColor, CalendarTitleForeColor 属性, CalendarTrailingForeColor 属性, CheckBox 属性, CustomFormat 属性, Format 属性, Hour 属性, Minute 属性, UpDown 属性, Second 属性, Year 属性 (ActiveX 控件), Day 属性, DateOfWeek 属性, MaxDate, MinDate 属性, Month 属性, Value 属性 (MonthView, DatePicker 控件), DateMember 属性, DateFormat 属性, DateBindings 属性, Height, Width 属性, TabIndex 属性, Tag 属性, Visible 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, TabStop 属性, Enabled 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, Object 属性, ToolTipText 属性, DateChanged 属性, DataField 属性, DataSource 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Font

属性（ActiveX 控件），hWnd 属性（ActiveX 控件），MouseIcon 属性（ActiveX 控件），MousePointer 属性（ActiveX 控件）。

## 方法

Refresh 方法，SetFocus 方法，Drag 方法，Move 方法，ZOrder 方法，ShowWhatsThis 方法，OLEDrag 方法（ActiveX 控件）。

## 事件

CloseUp 事件，CallKeyDown 事件，Format 事件，FormatSize 事件，DropDown 事件（DateTimePicker 控件），DragDrop 事件，DragOver 事件，GotFocus 事件，LostFocus 事件，Validate 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Change 事件（ActiveX 控件），Click 事件（ActiveX 控件），DblClick 事件（ActiveX 控件），KeyDown, KeyUp 事件（ActiveX 控件），KeyPress 事件（ActiveX 控件），MouseDown, MouseUp 事件（ActiveX 控件），MouseMove 事件（ActiveX 控件）。

## 请参阅

使用 DateTimePicker 控件。

## CalendarBackColor, CalendarForeColor 属性

返回或设置一个值，指定控件中下拉式日历菜单部分的背景色或前景色。

应用于

DateTimePicker 控件。

语法

*object*.CalendarBackColor [= *color*]

*object*.CalendarForeColor [= *color*]

CalendarBackColor 和 CalendarForeColor 属性语法有以下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>color</i>  | 一个值或一个常量，决定所使用的颜色        |

设置

color 设置值是：

| 设置值                          | 描述  |
|------------------------------|---|
| <i>RGB colors</i>            | 用调色板或者在代码中用 <b>RGB</b> 或 <b>QBColor</b> 函数指定的颜色   |
| <i>System default colors</i> | 对象浏览器的 <b>Visual Basic (VB)</b> 对象库所列出的系统颜色常量所指定的颜色。 <b>Windows</b> 操作环境将替换用户在控件面板中的设置值 |

### 说明

**CalendarBackColor** 和 **CalendarForeColor** 属性可以与 **CalendarTitleBackColor**、**CalendarTitleForeColor** 和 **CalendarTrailingForeColor** 属性一起使用，来自定义控件的颜色。

正常 **RGB** 颜色的有效范围是从 0 到 16,777,215 (&HFFFFFF)。在这个范围内的数值的高位字节等于 0；更低的三个字节，由最低到最高有效字节，分别决定了红、绿、蓝三种颜色的数量。红、绿、蓝三种颜色的成分每一种分别由一个 0 到 255 (&HFF)之间的数表示。如果高字节不等于 0，**Visual Basic** 将使用用户在控件面板中设置的系统颜色，或使用对象浏览器 **Visual Basic (VB)** 对象库中所列出的常量代表的颜色。

### 示例

下例通过重新设置几个颜色属性来改变 **DateTimePicker** 控件日历的外观。要试验该示例，在窗体中添加一个 **DateTimePicker** 控件和一个 **CommandButton** 控件。把代码粘贴到代码模块的声明部分。启动工程并单击 **DateTimePicker** 控件，观察缺省的颜色。单击 **CommandButton** 并再次单击 **DateTimePicker** 控件观

察新颜色。

```
Private Sub Command1_Click()  
    With DTPicker1  
        .CalendarBackColor = vbYellow  
        .CalendarTitleBackColor = vbRed  
        .CalendarTitleForeColor = vbWhite  
        .CalendarTrailingForeColor = vbGreen  
    End With  
End Sub
```

请参阅

CalendarTitleBackColor, CalendarTitleForeColor 属性, CalendarTrailingForeColor 属性。

CalendarTitleBackColor, CalendarTitleForeColor 属性

返回或设置一个值，指定控件下拉式日历部分标题的前景色或背景色。

应用于

DateTimePicker 控件。

语法

*object*.CalendarTitleBackColor [= *color*]

*object*.CalendarTitleForeColor [= *color*]

CalendarTitleBackColor 和 CalendarTitleForeColor 属性语法有以下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>color</i>  | 一个值或一个常量，决定使用的颜色         |

设置

color 的设置值有：

| 设置值                          | 描述  |
|------------------------------|---|
| <i>RGB colors</i>            | 用调色板或者在代码中用 RGB 或 QBColor 函数指定的颜色                                       |
| <i>System default colors</i> | 对象浏览器的 Visual Basic (VB)对象库所列出的系统颜色常量所指定的颜色。Windows 操作环境将替换用户在控件面板中的设置值 |

说明

CalendarTitleBackColor 和 CalendarTitleForeColor 属性可以与 CalendarBackColor、CalendarForeColor 和 CalendarTrailingForeColor 属性一起使

用来自定义控件的颜色。

正常 RGB 颜色的有效范围是从 0 到 16,777,215 (&HFFFFFF)。在这个范围内的数值的高位字节等于 0；更低的三个字节，由最低到最高有效字节，分别决定了红、绿、蓝三种颜色的数量。红、绿、蓝三种颜色的成分每一种分别由一个 0 到 255 (&HFF)之间的数表示。如果高字节不等于 0，Visual Basic 将使用用户在控件面板中设置的系统颜色，或使用对象浏览器 Visual Basic (VB) 对象库中所列出的常量代表的颜色。

请参阅

CalendarBackColor, CalendarForeColor 属性, CalendarTrailingForeColor 属性。

## CalendarTrailingForeColor 属性

返回或设置一个值，定义下拉式日历中显示的后续日期的前景色。

应用于

DateTimePicker 控件。

语法

*object*.CalendarTrailingForeColor [= *color*]

CalendarTrailingForeColor 属性语法有下列部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象                    |
| <i>color</i>  | 一个值或一个常量，决定后续日期的颜色，后续日期是属于前面或后面的月份的。如设置值中所述 |

## 设置

color 的设置值为：

| 设置值                          | 描述   |
|------------------------------|--|
| <i>RGB colors</i>            | 用调色板或者在代码中用 RGB 或 QBColor 函数指定的颜色                                      |
| <i>System default colors</i> | 对象浏览器的 Visual Basic(VB)对象库所列出的系统颜色常量所指定的颜色。Windows 操作环境将替换用户在控件面板中的设置值 |

## 说明

后续日期是在下拉式日历中显示的当前选中的月份之前和后续的数字。缺省情况下，后续数据以 vbWhite 显示。

CalendarTrailingForeColor 属性可以与 CalendarBackColor、CalendarForeColor、CalendarTitleBackColor 和 CalendarTitleForeColor 属性一起使用来自定义控件的颜色。



正常 RGB 颜色的有效范围是从 0 到 16,777,215 (&HFFFFFF)。在这个范围内的数值的高位字节等于 0；更低的三个字节，由最低到最高有效字节，分别决定了红、绿、蓝三种颜色的数量。红、绿、蓝三种颜色的成分每一种分别由一个 0 到 255 (&HFF)之间的数表示。如果高字节不等于 0，Visual Basic 将使用用户在控件面板中设置的系统颜色，或使用对象浏览器 Visual Basic (VB) 对象库中所列出的常量代表的颜色。

请参阅

CalendarBackColor, CalendarForeColor 属性, CalendarTitleBackColor, CalendarTitleForeColor 属性。

## CallbackKeyDown 事件

在光标位于回调字段内的情况下，一个键被按下时发生。

应用于

DateTimePicker 控件。

语法

```
Private Sub object_CallbackKeyDown([index As Integer],KeyCode As Integer,Shift As Integer, CallbackField As String,Date As Date)
```

CallbackKeyDown 事件语法有如下部分：

| 部分                   | 描述                                       |
|----------------------|--|
| <i>object</i>        | 一个对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>index</i>         | 一个整数，它唯一标识控件数组中的一个控件                     |
| <i>keyCode</i>       | 一个数值表达式，指定被按下的键所对应的 ASCII 键码             |
| <i>shift</i>         | 一个数值表达式，指定事件发生时 SHIFT 键、CTRL 键和 ALT 键的状态 |
| <i>callbackField</i> | 一个字符串表达式，指定回调子字符串                        |
| <i>date</i>          | 一个日期表达式，指定控件的日期数值                        |

## 说明

CallbackKeyDown 事件是 KeyDown 事件的变种，可用于处理一个回调子字符串。

关于回调处理的更多信息，请参阅 CustomFormat 属性。

该事件在用户按下上下箭头键时也会发生。

请参阅

KeyDown, KeyUp 事件（ActiveX 控件）。

## CheckBox 属性

返回或设置一个值，决定是否在选中日期的左边显示一个复选框。

应用于

DateTimePicker 控件。

语法

*object.CheckBox* [= *boolean*]

The CheckBox 属性语法有如下部分:

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中一个对象 |
| <i>boolean</i> | 一个布尔表达式，定义是否显示复选框       |

设置

*boolean* 的设置值为:

| 设置值   | 描述                  |
|-------|---------------------|
| True  | 在选中日期的左边显示一个复选框     |
| False | (缺省) 在所选日期的左边不显示复选框 |

## 说明

复选框允许用户决定是否使用控件来设置日期。当复选框为空时，则没有日期被选中。

## CloseUp 事件

当下拉日历被关闭时发生。

应用于

DateTimePickerk 控件。

语法

Private Sub *object*\_CloseUp(*[index As Integer]*)

CloseUp 事件语法有如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 一个整数，它唯一标识控件数组中的一个控件     |

说明

CloseUp 事件可用于对用户关闭下拉日历的响应。

# CustomFormat 属性

返回或设置一个值，指定控件在显示日期/时间信息时所使用的格式。

应用于

DateTimePicker 控件。

语法

*object*.CustomFormat [= *string*]

CustomFormat 属性语法有以下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | 一个字符串表达式，指定显示的格式         |

设置

DateTimePicker 控件支持下列格式字符：

| 字符串分段      | 描述                      |
|------------|-------------------------|
| <i>d</i>   | 一位或两位数字的日期数值            |
| <i>dd</i>  | 两位数字的日期数值。一位数字的日期数值要前置零 |
| <i>ddd</i> | 三个字母的星期日期缩写             |

|              |  |
|--------------|--|
| <i>dddd</i>  | 星期日期全名   |
| <i>h</i>     | 12 小时格式的一位或两位数字的小时数值   |
| <i>hh</i>    | 12 小时格式的两位数字的小时数值，一位数字的小时数值要前置零  |
| <i>H</i>     | 24 小时格式的一位或两位数字的小时数值   |
| <i>HH</i>    | 24 小时格式的两位数字的小时数值。一位数字的小时数值要前置零  |
| <i>m</i>     | 一位或两位数字的分钟数值   |
| <i>mm</i>    | 两位数字的分钟数值。一位数的分钟数值要前置零   |
| <i>M</i>     | 一位或两位数字的月份数值   |
| <i>MM</i>    | 两位数字的月份数值。一位数字的月份数值要前置零  |
| <i>MMM</i>   | 三个字母的月份缩写  |
| <i>MMMM</i>  | 月份全名   |
| <i>s</i>     | 一位或两位数字的秒的数值   |
| <i>ss</i>    | 两位数字的秒的数值。一位数字的秒的数值要前置零  |
| <i>t</i>     | 一位字母缩写 AM/PM （例如： "AM" 由 "A" 来代表）  |
| <i>tt</i>    | 两位字母缩写 AM/PM （例如： "AM" 由 "AM" 来代表）   |
| <b>字符串分段</b> | <b>描述</b>  |
| <i>x</i>     | 一个回调字段。控件仍然使用其它的有效格式字符，并要求所有者填写 "X" 部分。所有者必须准备好去响应那些要求提供怎样填写这些字段信息的事件。可以使用连续的多个 'X' 字符来表示唯一的回调字段 |

续表

|     |                                    |
|-----|------------------------------------|
| y   | 一位数字的年份（就是说，1997 将显示为 "7" ）        |
| yy  | 年份的后两位数字来代表年份（这就是，1997 将显示为 "97" ） |
| yyy | 完整的年份表示（这就是，1997 将显示为 "1997" ）     |

## 说明

**CustomFormat** 属性的作用就像一个输入掩码。要显示一个自定义格式，必须把 **Format** 格式设置为 **dtbCustom**。

可以在格式字符串中加入正文。例如：如果希望控件以 "Today is:Friday July 25, 1997 Time: 8:34 AM" 的格式来显示当前日期，那么 **CustomFormat** 字符串应为 "Today is:' ddddMMMMdd, yyy ' Time: 'h:mtt"。正文必须以单引号括起。

上述描述的自定义格式字段中有一种称为回调字段。回调字段允许通过指定格式字符串的特定部分作为回调字段来自定义输出。要声明一个回调字段，必须在格式字符串中的任何位置包含一个或多个 **X** 字符（ASCII 码为 88）。回调字段按从左到右的顺序显示。

在包含一个或多个回调字段的格式中显示新的日期时，每个回调字段都将产生 **Format** 和 **FormatSize** 事件。可以使用 **Format** 事件自定义响应字符串，使用 **FormatSize** 事件决定显示该字符串所需要的空间。这就使您能够完全控制回调字段的显示。

每一个 **X** 的序列都有其特定的含义。例如：**X** 可能意味着 "st/nd/rd/th"（对 "1st" "2nd" "3rd" 或 "4th"）而 **XX** 可能意味着 "first" "second" "third" 或

"fourth"。这些字段不是格式化用户的文本。它们只是将日期格式化为可以显示的格式。

例如：希望用英语和西班牙语以下列格式显示月份：

**July (Julio) 29**

可以按下列格式创建字符串：

**MMMM XXXX d**

处理 **Format** 和 **FormatSize** 事件时，可以通过比较输入字符串与"XXXX"来检查哪一个回调字段正在被调用。如果字段字符串相互匹配，那么就能够创建输出字符串 "(Julio)"，并能够提供输出字符串的长度。

仅当应用程序决定向回调字段提供文本的内容时，才使用 **X** 的数量。在处理 **FormatSize** 事件时，文本的大小可以被编程计算。

可以通过使用多个"X"字符来产生唯一的回调字段。因此，格式字符串 "XXddddMMMd"、'yyyXXX' 包含两个回调字段。回调字段是有效字段，所以应用程序必须准备处理 **Key** 事件。

示例

在 **Load** 事件中，将 **CustomFormat** 属性设置为包含了 4 个“X”的字符串就可以设置回调字“XXXX”。**FormatSize** 事件先于 **Format** 事件发生，用于确定回调字段的大小。然后才发生 **Format** 事件，所返回的格式字符串将替换回调字段文本。要运行下面的例子，需要在窗体上放置一个 **DateTimePicker** 控件，并将下



面的代码拷贝到 Declarations 段。

```
Option Base 1
```

```
Private sSpanishMonthLong(12) As String
```

```
Private Sub DTPicker1_Format(ByVal CallbackField As String, FormattedString As String)
```

```
    If CallbackField = "XXXX" Then
```

```
        FormattedString = sSpanishMonthLong(DTPicker1.Month)
```

```
    End If
```

```
End Sub
```

```
Private Sub DTPicker1_FormatSize(ByVal CallbackField As String, Size As Integer)
```

```
    Dim iMaxMonthLen As Integer
```

```
    If CallbackField = "XXXX" Then
```

```
        iMaxMonthLen = 0
```

```
        For i = 1 To 12
```

```
            If iMaxMonthLen < Len(sSpanishMonthLong(i)) Then
```

```
                iMaxMonthLen = Len(sSpanishMonthLong(i))
```

```
            End If
```

```
        Next
```

```
    End If
```

```
    Size = iMaxMonthLen
```

```
End Sub
```

```
Private Sub Form_Load()  
    DTPicker1.CustomFormat = "MMMM(XXXX) dd, yyy"  
    DTPicker1.Format = dtpCustom  
  
    sSpanishMonthLong(1) = "Enero"  
    sSpanishMonthLong(2) = "Febrero"  
    sSpanishMonthLong(3) = "Marzo"  
    sSpanishMonthLong(4) = "Abril"  
    sSpanishMonthLong(5) = "Mayo"  
    sSpanishMonthLong(6) = "Junio"  
    sSpanishMonthLong(7) = "Julio"  
    sSpanishMonthLong(8) = "Agosto"  
    sSpanishMonthLong(9) = "Septiembre"  
    sSpanishMonthLong(10) = "Octubre"  
    sSpanishMonthLong(11) = "Noviembre"  
    sSpanishMonthLong(12) = "Diciembre"  
  
End Sub
```

请参阅

Format 事件, FormatSize 事件。

## DropDown 事件（DateTimePicker 控件）

当下拉日历被拉下时发生。

应用于

DateTimePicker 控件。

语法

```
Private Sub object_DropDown()
```

*object* 置换元表示一个对象表达式，其值是“应用于”列表中的一个对象。

## Format 事件

当控件请求要被显示在调用字段中的文本时发生。

应用于

DateTimePicker 控件。

语法

```
Private Sub object_Format([index As Integer], CallbackField As String,  
FormattedString As String)
```

Format 事件语法有如下部分：

| 部分                          | 描述                       |
|-----------------------------|--------------------------|
| <i>object</i>               | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>                | 一个整数，它唯一标识控件数组中的一个控件     |
| <i>callbackField</i>        | 一个字符串表达式，指定回调子字符串        |
| <i>formattedStringField</i> | 一个字符串表达式，指定所要显示的格式化字符串   |

## 说明

Format 事件用于设置在回调字段中显示的文本。

关于回调处理的更多信息，请参阅 CustomFormat 属性。

请参阅

FormatSize 事件。

## Format 属性

返回或设置一个值，决定在控件中显示文本所使用的格式类型。

应用于

DateTimePicker 控件。

# 语法

*object.Format* [= *integer*]

Format 属性语法有以下部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象     |
| <i>integer</i> | 一个值，指定控件中显示文本所使用的格式样式。如设置值所示 |

# 设置

integer 的设置值为：

| 常量                 | 值 | 描述                                  |
|--------------------|---|-------------------------------------|
| datetime.LongDate  | 0 | 长型日期格式（例如： "Friday, Nov 14, 1972" ） |
| datetime.ShortDate | 1 | 短型日期格式（例如： "11/14/72" ）             |
| datetime.Time      | 2 | 时间格式（例如 "5:31:47 PM" ）              |
| datetime.Custom    | 3 | 用户自定义格式                             |

# 说明

CustomFormat 属性可以覆盖 FormatStyle 属性。

请参阅

CustomFormat 属性。

## FormatSize 事件

在 CustomFormat 属性更改之后，并且在 Format 事件发生前发生。该事件允许您为格式化的字符串设置最大允许尺寸，这样就使控件可以在屏幕上以充足的空间描绘用户格式化的字符串。

应用于

DateTimePicker 控件。

语法

Private Sub *object*\_FormatSize(*[index* As Integer], *CallbackField* As String, *Size* As Long)

FormatSize 事件语法有如下部分：

| 部分                   | 描述                               |
|----------------------|----------------------------------|
| <i>object</i>        | 一个对象表达式，其值是“应用于”列表中的一个对象         |
| <i>index</i>         | 一个整数，唯一标识控件数组中的一个控件              |
| <i>callbackField</i> | 一个字符串表达式，指定回调子字符串                |
| <i>size</i>          | 一个长整数，指定在 Format 事件中将要被返回的字符串的大小 |

## 说明

**FormatSize** 事件用于设置回调字段中所显示文本的大小。

请参阅

**Format** 事件。

## Hour 属性

返回或设置一个值，指定当前所显示的小时的数值。

应用于

**DateTimePicker** 控件。

语法

*object*.Hour [= *value*]

**Hour** 属性语法有如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个数值表达式，指定当前所显示的小时的数值    |

## 说明

Hour 属性可以是 0 到 23 的任意整数值。

## Minute 属性

返回或设置一个值，指定当前所显示的分钟的数值。

应用于

DateTimePicker 控件。

## 语法

*object*.Minute [= *value*]

Minute 属性语法上有如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 一个数值表达式，指定当前所显示的分钟的数值    |

## 说明

Minute 属性可以是 0 到 59 的任意整数。



## Second 属性

设置或返回一个整数，代表当前所显示秒的数值。

应用于

DateTimePicker 控件。

语法

*object.Second* [= *integer*]

Second 属性语法有如下部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象     |
| <i>integer</i> | 一个数值表达式，其值在 0 和 59 之间，表示秒的数值 |

## UpDown 属性

返回或设置一个值，决定是否在 DateTimePicker 控件的右边显示一个 up-down 控件。

应用于

DateTimePicker 控件。

语法

`object.UpDown [= boolean]`

UpDown 属性语法有如下部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象                                       |
| <i>boolean</i> | 一个布尔表达式，指定是否在 <code>DateTimePicker</code> 控件的右边显示一个 up-down 控件 |

设置

`boolean` 的设置值为：

| 设置值   | 描述                   |
|-------|----------------------|
| True  | 显示一个 up-down 控件      |
| False | （缺省）不显示一个 up-down 控件 |

说明

上下按钮可以替代下拉日历来选择日期。

## Year 属性（ActiveX 控件）

返回或设置当前所显示的年份。

应用于

DateTimePicker 控件。

语法

*object*.Year [= *number*]

Second 属性语法有如下部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象          |
| <i>number</i> | 一个数值表达式，在 MinDate 和 MaxDate 属性值之间 |

# FlatScrollBar 控件

FlatScrollBar 控件是标准 Windows 滚动条的对鼠标敏感版本，它提供平面格式选项。它也可以代替标准 Windows 的立体滚动条。使用滚动箭头和滚动框时，FlatScrollBar 能提供更强的交互性能。

## 语法

FlatScrollBar

## 说明

FlatScrollBar 提供三种格式选项。

- 和 Internet Explorer 4.0 中的滚动条看起来一样的平面外观。滚动箭头和滚动条滑块都是对鼠标敏感的；当鼠标指针移动到它们上面时，它们就会改变颜色。
- 鼠标指针放在上面就变成立体的平面外观。当鼠标指针移动到滚动箭头和滚动条滑块上时，它们就会变成斜面，它仿照了 Microsoft Encarta Encyclopedia 中的滚动条外观。
- 和标准 Windows 的立体滚动条外观相同。立体模式没有鼠标敏感功能。

使用 FlatScrollBar 可以禁用两个滚动箭头之一,这就给用户提供了基于程序中其它因素的附加反馈信息,以作为在特定方向上的滚动指示。。

FlatScrollBar 按照对它的 Orientation 属性的设置,可以作为水平或垂直滚动条使用。

**发布须知:** FlatScrollBar 控件是 MSCOMCT2.OCX 文件中一组 ActiveX 控件的一部分。要在您的应用程序中使用 FlatScrollBar 控件,您必须将 MSCOMCT2.OCX 文件添加到工程。当发布您的应用程序时,将 MSCOMCT2.OCX 安装到用户的 Microsoft Windows System 或 System32 目录中。有关如何将一个 ActiveX 控件添加到工程的详细信息,请参阅《Microsoft Visual Basic 6.0 参考库》中《Microsoft Visual Basic 6.0 程序员指南》的“添加控件到工程”。

## 属性

Orientation 属性 (Slider 控件), Appearance 属性 (FlatScrollBar 控件), Arrows 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, Value 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), LargeChange 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MouseIcon 属性 (ActiveX 控件), MousPointer 属性 (ActiveX 控件), Max,Min 属性 (ActiveX 控件)。

## 方法

SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, Refresh 方法 (ActiveX 控件)。

## 事件

ScrollEvent 事件 (Slider 控件), DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), Change 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件)。

## 请参阅

使用 FlatScrollBars 控件。

## Appearance 属性 (FlatScrollBar 控件)

返回或设置 FlatScrollBar 控件的外观。

## 应用于

FlatScrollBar 控件。

# 语法

```
object.Appearance [= integer]
```

Appearance 属性的语法有这些部分：

| 部分             | 说明                             |
|----------------|--------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象         |
| <i>integer</i> | 一个数字表达式，指定滚动条的样式，如同“设置”中所描述的那样 |

# 设置

对 integer 的设置值为：

| 常量         | 值 | 说明                                |
|------------|---|-----------------------------------|
| fsb3D      | 0 | 滚动条有标准 Windows 滚动条的立体外观           |
| fsbFlat    | 1 | (缺省) 滚动条是平面的                      |
| fsbTrack3D | 2 | 滚动条是平面的，当鼠标移动到滑块和箭头按钮上时，它们就会变成立体的 |

# 说明

Appearance 属性设置滚动条的外观，并决定滚动条是平面的、立体的，还

是两者的组合。组合的滚动条当鼠标指针在其上时，将滚动箭头和滑块从平面的变成立体的，并允许创建一个动态界面。

## Arrows 属性

返回或设置一个值，决定允许使用哪一个滚动箭头。

应用于

FlatScrollBar 控件。

语法

*object*.Arrows [= *integer*]

Arrows 属性的语法有这些部分：

| 部分             | 说明                                 |
|----------------|------------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象             |
| <i>integer</i> | 一个数字表达式，指定允许使用的滚动按钮或按钮，如同“设置”中所描述的 |

设置

*integer* 的设置值为：



| 常量           | 值 | 说明   |
|--------------|---|--|
| cc2Both      | 0 | (缺省) 向左的和向右的(或向上的和向下的,由 <b>Orientation</b> 而定)滚动箭头允许使用 |
| cc2LeftUp    | 1 | 仅向左的(或向上的) 滚动按钮允许使用                                    |
| cc2RightDown | 2 | 仅向右的(或向下的) 滚动按钮允许使用                                    |

### 说明

当 **FlatScrollBar** 的 **Orientation** 从水平变为垂直时, 向左的滚动箭头变为向上的箭头, 向右的滚动箭头变为向下的箭头。

对 **Arrows** 属性最常见的使用是当控件达到最大或最小值时, 禁用某个适当的滚动箭头。

## Orientation 属性 (FlatScrollBar 控件)

返回或设置一个值。该值决定了对象的方向 (水平或垂直)。

### 语法

*object.Orientation* [= *integer*]

**Orientation** 属性的语法有如下部分:

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>integer</i> | 一个数值表达式，决定控件的方向，如“设置”中所示 |

## 设置

integer 的设置值为：

| 常量                       | 值 | 描述            |
|--------------------------|---|---------------|
| cc2OrientationHorizontal | 0 | （缺省）该控件是水平方向的 |
| cc2OrientationVertical   | 1 | 该控件是垂直方向的     |

# MSFlexGrid 控件

**MSFlexGrid** 控件显示和操作表格数据。其对包含字符串和图片的表格提供了灵活的排序、插入数据和格式编排功能。当与 **Data** 控件绑定时，**MSFlexGrid** 控件只显示只读数据。

## 语法

### MSFlexGrid

## 说明

你可以在 **MSFlexGrid** 中的任何单元放置文本、图片或这二者。**Row** 和 **Col** 属性指定 **MSFlexGrid** 控件的当前单元。你可以在代码中指定当前单元，也可以在运行时使用鼠标或键盘改变当前单元。**Text** 属性引用了当前单元中的文本。

如果单元中的文本太长以致不能在单元中一行显示，将 **WordWrap** 属性置为 **True** 将在单元中分多行显示文本。要在单元中显示多行文本，你需要增大单元的宽度（**ColWidth** 属性）或行高度（**RowHeight** 属性）。

使用 **Cols** 和 **Rows** 属性确定 **MSFlexGrid** 控件中的行数和列数。

**注意：**在你的应用程序可以使用 **MSFlexGrid** 控件之前，你必须将

MSFlxGrd.ocx 文件加入到你的工程中。要自动在工程中包含该文件，可以将其加入到 Autoload 文件中。当分布你的应用程序时，你应将 MSFlxGrd.ocx 文件安装到用户的 Microsoft Windows System 目录中。要获取如何给工程增加 ActiveX 控件，可以参阅《微软 Visual Basic 6.0 程序员指南》中的“标准 ActiveX 控件”一节。

## 属性

OLEDropMode 属性 (ActiveX 控件)，Height, Width 属性 (ActiveX 控件)，Index 属性 (ActiveX 控件)，Left, Top 属性 (ActiveX 控件)，Tag 属性 (ActiveX 控件)，Visible 属性 (ActiveX 控件)，Object 属性 (ActiveX 控件)，ColPos 属性，ColPositon, RowPosition 属性，Cols, Rows 属性 (MSHFlexGrid)，ColSel, RowSel 属性，ColWidth 属性 (MSHFlexGrid)，FillStyle 属性 (MSHFlexGrid)，FixedCols, FixedRows 属性 (MSHFlexGrid)，FocusRect 属性，FontWidth, FontWidthBand, FontWidthFixed, FontWidthHeader 属性 (MSHFlexGrid)，ForeColor, ForeColorBand, ForeColorFixed, ForeColorHeader, ForeColorSel 属性，FormatString 属性，GridColor, GirdColorBand, GridColorFixed, GridColorHeader, GridColorIndent, GridColorUnpopulated 属性，GridLines, GridLinesBand, GridLinesFixed, GridLinesHeader, GridLinesIndent, GridLineUnpopulated 属性 (MSHFlexGrid)，GridLineWidth, GridLineWidthBand, GridLineWidthFixed, GridLineWidthHeader, GridLineWidthIndent, GridLineWidthUnpopulated 属性 (MSHFlexGrid)，HighLight 属性 (MSHFlexGrid)，LeftCol 属性 (MSHFlexGrid)，MergeCells 属性，MergeCol, MergeRow 属性，MouseCol, MouseRow 属性，Name

属性 (MSHFlexGrid), Picture 属性 (MSHFlexGrid), PictureType 属性, Redraw 属性, RowHeight 属性 (MSHFlexGrid), RowHeightMin 属性, RowVisible 属性, RowPos 属性, ScrollBars 属性 (MSHFlexGrid), ScrollTrack 属性, SelectionMode 属性, Sort 属性 (MSHFlexGrid), Text 属性 (MSHFlexGrid), TextArray 属性, TextMatrix 属性, TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader 属性 (MSHFlexGrid), TopRow 属性 (MSHFlexGrid), Version 属性 (MSHFlexGrid), WordWrap 属性 (MSHFlexGrid), AllowBigSelection 属性, AllowUserResizing 属性, BackColor, BackColorBkg, BackColorFixed, BackColorSel 属性, CellAlignment 属性, CellBackColor, CellForeColor 属性, CellFontBold 属性, CellFontItalic 属性, CellFontName 属性, CellFontSize 属性, CellFontStrikeThrough 属性, CellFontUnderline 属性, CellFontWidth 属性, CellHeight, CellLeft, CellTop, CellWidth 属性 (MSHFlexGrid), CellPicture 属性, CellPictureAlignment 属性, CellTextStyle 属性, Clip 属性 (MSHFlexGrid), Col, Row 属性 (MSHFlexGrid), ColAlignment, ColAlignmentBand, ColAlignmentHeader 属性 (MSHFlexGrid), ColData, RowData, BandData 属性 (MSHFlexGrid), ColIsVisible 属性, DataBindings 属性, TabIndex 属性, DragIcon 属性, DragMode 属性, MouseIcon 属性, TabStop 属性, HelpContextID 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelp 属性, Appearance 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), DataSource 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件), RemoveItem 方法 (MSHFlexGrid), AddItem 方法 (MSHFlexGrid), Clear 方法 (MSHFlexGrid), SelfFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法。

## 事件

RowColChange 事件 (MSHFlexGrid), Scroll 事件 (MSHFlexGrid), SelChange 事件 (MSHFlexGrid), Compare 事件, EnterCell 事件, LeaveCell 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, MouseDown, MouseUp 事件, MouseMove 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), Click 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件)。

## 请参阅

MSHFlexGrid 控件, 标准 ActiveX 控件。

# MSHFlexGrid 控件

**MSHFlexGrid** 控件显示和操作表格数据。其对包含字符串和图片的表格提供了灵活的排序、插入数据和格式编排功能。当与 **Data** 控件绑定时, **MSHFlexGrid** 控件只显示只读数据。

## 语法

**MSHFlexGrid**

## 说明

你可以在 **MSHFlexGrid** 中的任何单元放置文本、图片或这二者。**Row** 和 **Col** 属性指定 **MSHFlexGrid** 控件的当前单元。你可以在代码中指定当前单元, 也可以在运行时使用鼠标或键盘改变当前单元。**Text** 属性引用了当前单元中的文本。

如果单元中的文本太长以致不能在单元中一行显示, 将 **WordWrap** 属性置为 **True** 将在单元中分多行显示文本。要在单元中显示多行文本, 你需要增大单元的宽度 (**ColWidth** 属性) 或行高度 (**RowHeight** 属性)。

使用 **Col** 和 **Row** 属性确定 **MSHFlexGrid** 控件中的行数和列数。使用 **Band** 属性确定 **MSHFlexGrid** 中的 **band** 风格。

注意：在你的应用程序可以使用 MSHFlexGrid 控件之前，你必须将 MSHFlxGrd.ocx 文件加入到你的工程中。要自动在工程中包含该文件，可以将其加入到 Autoload 文件中。当分布你的应用程序时，你应将 MSHFlxGd.ocx 文件安装到用户的 Microsoft Windows System 目录中。要获取如何给工程增加 ActiveX 控件，可以参阅《微软 Visual Basic 6.0 程序员指南》中的“标准 ActiveX 控件”一节。

## 属性

OLEDropMode 属性 (ActiveX 控件)，Height, Width 属性 (ActiveX 控件)，Index 属性 (ActiveX 控件)，Left, Top 属性 (ActiveX 控件)，Tag 属性 (ActiveX 控件)，Visible 属性 (ActiveX 控件)，Object 属性 (ActiveX 控件)，DataMember 属性 (ActiveX 控件)，Collapse 事件 (MSHFlexGrid)，CollapseAll 方法 (MSHFlexGrid)，ColPos 属性，ColPositon, RowPosition 属性，Cols, Rows 属性 (MSHFlexGrid)，ColSel, RowSel 属性，ColWidth 属性 (MSHFlexGrid)，DataField 属性 (MSHFlexGrid)，FillStyle 属性 (MSHFlexGrid)，FixedCols, FixedRows 属性 (MSHFlexGrid)，FocusRect 属性，Font, FontBand, FontFixed, FontHeader 属性 (MSHFlexGrid)，FontWidth, FontWidthBand, FontWidthFixed, FontWidthHeader 属性 (MSHFlexGrid)，ForeColor, ForeColorBand, ForeColorFixed, ForeColorHeader, ForeColorSel 属性，FormatString 属性，GridColor, GirdColorBand, GridColorFixed, GridColorHeader, GridColorIndent, GridColorUnpopulated 属性，GridLines, GridLinesBand, GridLinesFixed, GridLinesHeader, GridLinesIndent, GridLineUnpopulated 属性 (MSHFlexGrid)，GridLineWidth, GridLineWidthBand,



GridLineWidthFixed, GridLineWidthHeader, GridLineWidthIndent, GridLineWidthUnpopulated 属性 (MSHFlexGrid), Highlight 属性 (MSHFlexGrid), LeftCol 属性 (MSHFlexGrid), MergeCells 属性 (MSHFlexGrid), MergeCol, MergeRow 属性, MouseCol, MouseRow 属性, Name 属性 (MSHFlexGrid), Picture 属性 (MSHFlexGrid), PictureType 属性, Redraw 属性, RowExpandable, RowExpanded 属性, RowHeight 属性 (MSHFlexGrid), RowHeightMin 属性, RowIsVisible 属性, RowPos 属性, RowSizingmode 属性, ScrollBars 属性 (MSHFlexGrid), ScrollTrack 属性, SelectionMode 属性, Sort 属性 (MSHFlexGrid), Text 属性 (MSHFlexGrid), TextArray 属性, TextMatrix 属性, TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader 属性 (MSHFlexGrid), TopRow 属性 (MSHFlexGrid), Version 属性 (MSHFlexGrid), WordWrap 属性 (MSHFlexGrid), AllowBigSelection 属性, AllowUserResizing 属性, BackColorIndent, BackColor, BackColorBkg, BackColorFixed, BackColorSel 属性, BackColorBand, BackColorHeader, BackColorIndent, BackColorUnpopulated 属性, BandColIndex 属性 (MSHFlexGrid), BandPisplay 属性 (MSHFlexGrid), BandExpandable 属性 (MSHFlexGrid), BandIndert 属性 (MSHFlexGrid), BandLevel 属性 (MSHFlexGrid), BandLevel 属性 (MSHFlexGrid), Bands 属性 (MSHFlexGrid), CellAlignment 属性, CellBackColor, CellForeColor 属性, CellFontBold 属性, CellFontItalic 属性, CellFontName 属性, CellFontSize 属性, CellFontStrikeThrough 属性, CellFontUnderline 属性, CellFontWidth 属性, CellHeight, CellLeft, CellTop, CellWidth 属性 (MSHFlexGrid), CellPicture 属性, CellPictureAlignment 属性, CellTextStyle 属性, ColAlignmentFixed 属性, Clip 属性 (MSHFlexGrid), Col, Row

属性 (MSHFlexGrid), ColAlignment, ColAlignmentBand, ColAlignmentHeader 属性 (MSHFlexGrid), CellType 属性, ColData, RowData, BandData 属性 (MSHFlexGrid), ColHeader 属性(MSHFlexGrid), ColHeaderCaption 属性, ColIsVisible 属性, DataBindings 属性, TabIndex 属性, DragIcon 属性, DragMode 属性, MouseIcon (ActiveX 控件) 属性, TabStop 属性, HelpContextID 属性, Parent 属性, Container 属性, Recorset 属性, WhatsThisHelpID 属性, Appearance 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), DataSource 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), ToolTipText 属性 (ActiveX 控件)。

## 方法

OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件), CollapseAll 方法 (MSHFlexGrid), ExpandAll 方法 (MSHFlexGrid), RemoveItem 方法 (MSHFlexGrid), AddItem 方法 (MSHFlexGrid), Clear 方法 (MSHFlexGrid), ClearStructure 方法 (MSHFlexGrid), SelfFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法。

## 事件

RowColChange 事件 (MSHFlexGrid), Scroll 事件 (MSHFlexGrid), SelChange 事件 (MSHFlexGrid), Collapse 事件, Compare 事件, EnterCell 事件, Expand 事件 (MSHFlexGrid), LeaveCell 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, MouseDown, MouseUp 事件, MouseMove 事件,

OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Click 事件（ActiveX 控件），KeyDown, KeyUp 事件（ActiveX 控件），KeyPress 事件（ActiveX 控件）。

请参阅

MSFlexGrid 控件，标准 ActiveX 控件。

## AddItem 方法 (MSHFlexGrid)

该方法将一个行添加到 MSHFlexGrid 控件中。不支持命名参数。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.AddItem (string, index, number)*

AddItem 方法的语法包含以下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象  |
| <i>string</i> | 字符串必需的。字符串表达式，它在新增行中显示。可以用制表符 (vbTab) 来分隔每个字符串，从而将多个字符串（行中的多个列）添加进去   |
| <i>index</i>  | 可选的。Long 值，它代表了控件中放置新增行的位置。对于第一行来说，index = 0。如果省略 index，那么新增行将成为带区中的最后一行。注意 index 是 MSHFlexGrid 中的 BandColIndex |
| <i>number</i> | 可选的。Long 值，指出添加行的带区号  |

## 说明

如果 MSHFlexGrid 不包含带区（即它被绑定到一个非层次结构类型的 Recordset），那么 BandNumber 参数将是可选的。即使指定了它的值，指定值也必须为 0。如果 BandDisplay 属性被设置为水平的，而 MSHFlexGrid 被绑定到一个分层结构的 Recordset，则 BandNumber 参数将是必需的。如果 BandDisplay 属性被设置为垂直的，那么只有当带区有二义性的时候 BandNumber 参数才是必要的。index 参数永远是可选的。

## 示例

在该示例中，用 AddItem 方法将 100 项添加到 MSHFlexGrid 中。要试用此例，可以将代码粘贴到窗体（该窗体带有命名为 MSHFlexGrid1 的 MSHFlexGrid 控件）的声明部分，然后按下 F5 键，并单击该窗体。

注意：如果您使用 MSFlexGrid，请用“MSFlexGrid1”替换“MSHFlexGrid1”。

```
Private Sub Form_Click ()
```

```
    Dim Entry, i, Msg                '声明变量。
```

```
    Msg = _
```

```
    "Choose OK to add 100 items to your MSHFlexGrid."
```

```
    MsgBox Msg        '显示消息。
```

```
    MSHFlexGrid1.Cols = 2        '每行有两个字符串。
```

```
    For i = 1 To 100    '从 1 计数到 100。
```

```
        Entry = "Entry " & Chr(9) & I    '创建项。
```

```
        MSHFlexGrid1.AddItem Entry        '添加项。
```

```
    Next i
```

```
    Msg = "Choose OK to remove every other entry."
```

```
    MsgBox Msg        '显示消息。
```

```
    For i = 1 To 50        '决定怎样删除
```

```
        MSHFlexGrid1.RemoveItem I    '其它每一项。
```

```
    Next I
```

```
    Msg = "Choose OK to clear all items."
```

```
    MsgBox Msg        '显示消息。
```

```
    MSHFlexGrid1.Clear        '清除列表框。
```

```
End Sub
```

## AllowBigSelection 属性

该属性返回或者设置一个值，该值决定了在行头或者列头上单击时，是否可以使得整个行或者列都被选中。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.AllowBigSelection [=*boolean* ]

AllowBigSelection 属性的语法包含以下部分：

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象    |
| <i>boolean</i> | 布尔表达式，指出单击标头时，是否选择整行或整列 |

设置

Boolean 的设置值是：

| 设置值   | 描述                  |
|-------|---------------------|
| True  | 缺省。当用户单击标头时，选择整行或整列 |
| False | 当用户单击标头时，仅选择标头      |

请参阅

SelectionMode 属性。

注意：如果您使用 MSFlexGrid，请用 “MSFlexGrid1” 替换 “MSHFlexGrid1”。

示例

例如，下面的代码表示：当用户在标头上单击时，允许整个行或者列都被选中：

```
Sub Form1_Load ()  
    MSHFlexGrid1.AllowBigSelection = True  
End Sub
```

## AllowUserResizing 属性

该属性返回或者设置一个值，该值决定了是否可以用鼠标来对 MSHFlexGrid 控件中行和列的大小进行重新调整。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

# 语法

*object.AllowUserResizing* [= *value* ]

*AllowUserResizing* 属性的语法包含以下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                        |
| <i>value</i>  | 整数或者常量，它指定了用户是否可以对行和列的大小进行重新调整，就象下面的“设置”中所述 |

# 设置

*value* 的设置值是：

| 常量                       | 值 | 描述                      |
|--------------------------|---|-------------------------|
| <i>FlexResizeNone</i>    | 0 | （缺省的）不允许。用户不能用鼠标来重新调整大小 |
| <i>FlexResizeColumns</i> | 1 | 列。用户可以用鼠标来重新调整列的大小      |
| <i>FlexResizeRows</i>    | 2 | 行。用户可以用鼠标来重新调整行的大小      |
| <i>FlexResizeBoth</i>    | 3 | 行和列。用户可以用鼠标来重新调整行和列的大小  |

# 说明

为调整行和列的大小，鼠标应该在 **MSHFlexGrid** 控件的固定区域的上方，并接近于行和列之间的边界。鼠标指针将改变为适当的调整大小用的指针，用户可以拖动行或者列，以改变行的高度或者列的宽度。下面的代码就实现了这种功能：



请参阅

ColWidth 属性（MSHFlexGrid）。

示例

下列代码示例添加了用户调整大小的功能。

**注意：**如果您使用 MSHFlexGrid，请用 “MSFlexGrid1” 替换 “MSHFlexGrid1”。

```
Sub Form1_Load ()
```

```
    MSHFlexGrid1.AllowUserResizing = True
```

```
End Sub
```

## BackColor、BackColorBkg、BackColorFixed 以及 BackColorSel 属性

这些属性返回或设置 MSHFlexGrid 的各种不同元素的背景颜色。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.BackColor [=color]

*object*.BackColorBkg [=color]

*object*.BackColorFixed [=color]

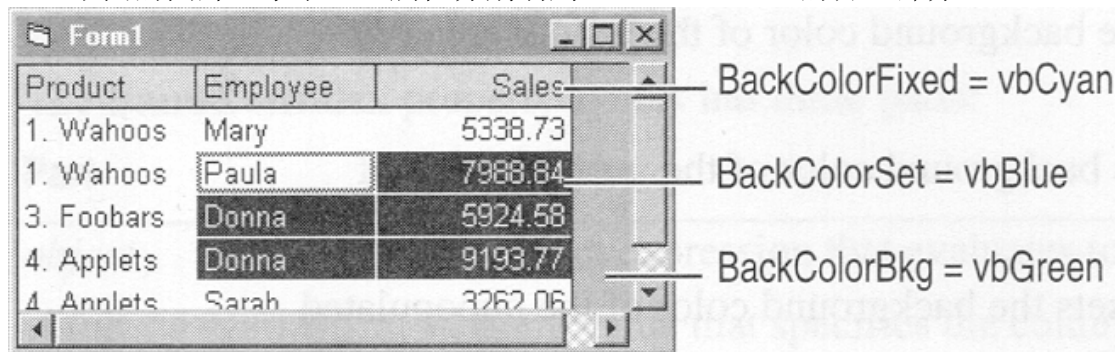
*object*.BackColorSel [=color ]

BackColor、BackColorBkg、BackColorFixed 以及 BackColorSel 属性的语法包含以下部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>color</i>  | 数值表达式，它指定了颜色         |

## 说明

下面的图片显示了这些属性分别引用 MSHFlexGrid 的哪一部分：



**BackColor** 影响所有未确定单元的颜色。可以用 **CellBackColor** 属性来对单个单元的背景颜色进行设置。

请参阅

**CellBackColor**, **CellForeColor** 属性。

示例

在该示例中，用了 **MSHFlexGrid** 中的 **BackColorBkg**, **BackColorFixed** 以及 **BackColorSel** 属性。它以每秒钟两次的速度，随机地对 **MSHFlexGrid** 控件的控件背景、已选定背景，以及确定单元背景的颜色进行重置。要试用此例，可以将下面的代码粘贴到窗体（该窗体带有命名为 **Timer1** 的 **Timer** 控件和命名为 **MSHFlexGrid1** 的 **MSHFlexGrid** 控件）的声明部分，然后加载该窗体。

**注意：**如果您使用 **MSFlexGrid**，请用 “**MSFlexGrid1**” 替换 “**MSHFlexGrid1**”。

```
Private Sub Form_Load ()
```

```
    Timer1.Interval =500
```

```
End Sub
```

```
Private Sub Timer1_Timer ()
```

```
    MSHFlexGrid1.BackColorBkg = QBColor(Rnd * 15)
```

```
    MSHFlexGrid1.BackColorFixed = QBColor(Rnd * 10)
```

```
    MSHFlexGrid1.BackColorSel = QBColor(Rnd * 10)
```

End Sub

## BackColorBand、BackColorHeader、BackColorIndent、 BackColorUnpopulated 属性

- **BackColorBand** — 返回或者设置 MSFlexGrid 带区的背景色。
- **BackColorHeader** — 返回或者设置 MSFlexGrid 标头区域的背景色。
- **BackColorIndent** — 返回或者设置 MSFlexGrid 缩进区域的背景色。
- **BackColorUnpopulated** — 返回或者设置 MSFlexGrid 未填数据区域的背景色。

应用于

MSFlexGrid 控件。

语法

*object*.BackColorBand [=color]

*object*.BackColorHeader [=color]

*object*.BackColorIndent [=color]

*object*.BackColorUnpopulated [=color]

BackColorBand 、 BackColorHeader 、 BackColorIndent 和  
BackColorUnpopulated 属性的语法包括以下各个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>color</i>  | 一个数值表达式，它指定了颜色           |

## BandColIndex 属性

返回相对于包含它的带区的当前单元格的列号。

应用于

MSHFlexGrid 控件。

语法

*object*.BandColIndex [=value]

BandColIndex 属性的语法包含以下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>value</i>  | 一个 Long 类型的数值，它指定了当前单元格的列号 |

# BandDisplay 属性 (MSHFlexGrid)

指定各个带区如何在 MSHFlexGrid 中显示。

应用于

MSHFlexGrid 控件。

语法

*object*.DisplayBandSettings [=*value*]

BandDisplay 属性的语法包含以下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象                   |
| <i>value</i>  | 一个数值表达式，它指定了带区如何显示在 MSHFlexGrid 中，如“设置”中所述 |

设置

*value* 的各种设置包括：

| 设置                        | 值 | 描述                                |
|---------------------------|---|-----------------------------------|
| FlexBandDisplayHorizontal | 0 | 缺省值。MSHFlexGrid 内的带区按水平方式显示（横跨排列） |
| FlexBandDisplayVertical   | 1 | MSHFlexGrid 内的带区按垂直方式显示（上下排列）     |

## BandExpandable 属性 (MSHFlexGrid)

返回和设置一个值，该值确定当前带区中的行是否能够被展开或收缩。当前带区是由 Col 和 Row 属性确定的。该属性在运行时是只读的，而在设计时是不可使用的。

应用于

MSHFlexGrid 控件。

语法

*object*.BandExpandable(*number*) [=boolean]

BandExpandable 属性的语法包含以下几个部分：

| 部分             | 描述                                  |
|----------------|-------------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象            |
| <i>number</i>  | 一个 Long 数值，它指示出带区在 MSHFlexGrid 中的编号 |
| <i>boolean</i> | 一个 Boolean 表达式，它确定所指定的带区是否能被展开或收缩   |

## 设置

Boolean 的设置为：

| 设置    | 描述                  |
|-------|---------------------|
| True  | 缺省值。被指定的带区是可以展开和收缩的 |
| False | 被指定的带区是不可以展开和收缩的    |

## 说明

如果带区是可展开的，那么带区内的第一个列的左边显示出一个标准的加号 (+) 和减号 (-) 位图。通过设置 **ExpandPicture** 和 **CollapsePicture** 属性，该位图可以被覆盖掉。当第一次显示该网格时，缺省情况下所有的带区都是收缩的。

通过设置该属性，用户将不能够展开或者收缩带区；它不会使指定带区内的各行被展开或者收缩。例如，在将该属性设置为 **False** 之前，如果子行已经被显示出来，那么子行将继续被显示。

要使一个带区成为可展开的，它至少需要有一个子带区。否则，**BandExpandable** 属性将被忽略。



## BandIndent 属性 (MSHFlexGrid)

指定带区缩进的列数。

应用于

MSHFlexGrid 控件。

语法

*object*.BandIndent [=*number*]

BandIndent 属性的语法包含以下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象                     |
| <i>number</i> | 一个 Long 数值，它指定了在 MSHFlexGrid 中要缩进的带区。缺省设置为 0 |

## BandLevel 属性 (MSHFlexGrid)

返回包含当前单元格的带区编号。带区编号从 0 开始。当前单元格是由 Col 和 Row 属性定义的。该属性不能在设计时使用。

应用于

MSHFlexGrid 控件。

语法

*object*.BandLevel [=*number*]

BandLevel 属性的语法包含以下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>number</i> | 一个整数或者常量，它指定了包含当前单元格的带区的编号 |

### Bands 属性 (MSHFlexGrid)

返回 MSHFlexGrid 中的带区个数。MSHFlexGrid 至少有一个带区。当 MSHFlexGrid 被绑定到一个标准 Recordset 时，整个 MSHFlexGrid 被作为一个带区处理。

应用于

MSHFlexGrid 控件。

## 语法

*object*.Bands [=value]

Bands 属性的语法包含以下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | MSHFlexGrid 中的带区个数       |

## 说明

该属性是只读的。其数值是由与控件绑定在一起的 Recordsets 的分层结构中的 Recordsets 的个数决定的。

设置 Col 和 Row 属性。

请参阅

Col 和 Row 属性 MSHFlexGrid。

## CellAlignment 属性

该属性返回或设置的数值确定了一个单元格或被选定的多个单元格所在区域的水平和垂直对齐方式。该属性在设计时是不可使用的。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellAlignment [=value]

CellAlignment 属性的语法包含以下几个部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象           |
| <i>value</i>  | 一个整数或者常量，它指定文本如何与单元格进行对齐，如“设置值”中所述 |

设置

value 的各种设置如下：

| 常量                    | 值 | 描述                      |
|-----------------------|---|-------------------------|
| flexAlignLeftTop      | 0 | 单元格的内容左、顶部对齐            |
| flexAlignLeftCenter   | 1 | 字符串的缺省对齐方式。单元格的内容左、居中对齐 |
| flexAlignLeftBottom   | 2 | 单元格的内容左、底部对齐            |
| flexAlignCenterTop    | 3 | 单元格的内容居中、顶部对齐           |
| flexAlignCenterCenter | 4 | 单元格的内容居中、居中对齐           |
| flexAlignCenterBottom | 5 | 单元格的内容居中、底部对齐           |

续表

|                                   |   |  |
|-----------------------------------|---|--|
| <code>flexAlignRightTop</code>    | 6 | 单元格的内容右、顶部对齐                           |
| <code>flexAlignRightCenter</code> | 7 | 数值的缺省对齐方式。单元格的内容右、居中对齐                 |
| <code>flexAlignRightBottom</code> | 8 | 单元格的内容右、底部对齐                           |
| <code>flexAlignGeneral</code>     | 9 | 单元格的内容按一般方式进行对齐。字符串按“左、居中”显示，数字按“右、居中” |

请参阅

`CellPicture` 属性，`ColAlignment`，`ColAlignmentBand`，`ColAlignmentHeader` 属性（`MSHFlexGrid`）。

示例

在下面的代码中，使用常量设置将每个单元格的文本对齐方式设置为左、居中。

注意：如果使用的是 `MSFlexGrid`，请将 `"MSHFlexGrid1"` 替换为 `"MSFlexGrid1"`。

```
Sub Form1_Load ()  
    MSHFlexGrid1.CellAlignment =flexAlignLeftCenter  
End Sub
```

## CellBackColor 和 CellForeColor 属性

- **CellBackColor** — 返回或者设置单独的单元格或者单元格区域的背景色。
- **CellForeColor** — 返回或者设置单独的单元格或者单元格区域的前景色。  
这些属性不能在设计时使用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellBackColor [=color]

*object*.CellForeColor [=color]

CellBackColor 和 CellForeColor 属性的语法包括以下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象   |
| <i>color</i>  | 整数（枚举）。数值表达式，它为当前选定单元指定了颜色。将这些属性中的任何一个设置为 0，都将用标准背景和前景颜色来画单元 |

## 说明

对这个属性进行修改会影响当前单元或者当前选定，这取决于 **FillStyle** 属性的设置值。

将这些属性中的任何一个设置为 0，**MSHFlexGrid** 都将用标准背景和前景颜色来画单元。如果想把这些属性中的任何一个设置为黑色，可以将它们设置为 1，而不是 0。

可以用 **BackColorBkg**, **BackColorFixed**, **BackColorSel**, **ForeColorFixed** 以及 **ForeColorSel** 属性对各种不同的 **MSHFlexGrid** 元素的颜色进行设置。可以用 **BackColor** 属性将所有未确定单元设置为同样的背景颜色。

## 请参阅

**FillStyle** 属性（**MSHFlexGrid**），**BackColor**, **BackColorBkg**, **BackColorFixed**, **BackColorSel** 属性。

## 示例

在以下示例中，用了 **MSHFlexGrid** 中的 **CellBackColor** 和 **CellForeColor** 属性。它以每秒钟两次的速度，随机地对 **MSHFlexGrid** 控件的焦点单元的背景颜色和文本颜色进行重置。要试用此例，可以将下面的代码粘贴到窗体（该窗体带有命名为 **Timer1** 的 **Timer** 控件和命名为 **MSHFlexGrid1** 的 **MSHFlexGrid** 控件）的声明部分，然后加载该窗体。

**注意：** 如果您使用 **MSFlexGrid**，请用 “**MSFlexGrid1**” 替换

“MSHFlexGrid1”。

```
Private Sub Form_Load ()  
    Timer1.Interval =500  
    MSHFlexGrid1.Text = "Focus Here"  
End Sub  
  
Private Sub Timer1_Timer ()  
    MSHFlexGrid1.CellBackColor = QBColor(Rnd * 15)  
    MSHFlexGrid1.CellForeColor = QBColor(Rnd * 10)  
End Sub
```

## CellFontBold 属性

该属性返回或设置当前单元文本的粗体样式。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellFontBold [=*boolean* ]

CellFontBold 属性的语法包含以下部分：



| 部分             | 描述                   |
|----------------|----------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象 |
| <i>boolean</i> | 布尔表达式决定了当前单元文本是否是粗体的 |

### 设置

Boolean 的设置值是:

| 设置值   | 描述                 |
|-------|--------------------|
| True  | 当前单元文本为粗体          |
| False | 缺省。当前单元文本为正常（不是粗体） |

### 说明

对该属性所做的更改会影响到当前单元或者当前选定，这取决于 **FillStyle** 属性的设置值。

### 请参阅

**FillStyle** 属性 (**MSHFlexGrid**)，**CellFontItalic** 属性，**CellFontName** 属性，**CellFontSize** 属性，**CellFontUnderline** 属性，**CellFontWidth** 属性。

### 示例

无论何时，当 **MSHFlexGrid** 控件在焦点中时，下面的代码将当前单元的

文本设置为粗体样式。

注意：如果您使用 MSFlexGrid，请用 “MSFlexGrid1” 替换 “MSHFlexGrid1”。

```
Sub MSHFlexGrid1_GotFocus()  
    MSHFlexGrid1.CellFontBold = 1  
End Sub
```

## CellFontItalic 属性

该属性返回或设置当前单元文本的斜体样式。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellFontItalic [=*boolean*]

CellFontItalic 属性的语法包含以下部分：

| 部分             | 描述                      |
|----------------|-------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象    |
| <i>boolean</i> | 布尔表达式，决定了单元中的文本样式是否是斜体的 |

## 设置

Boolean 的设置值是：

| 设置值   | 描述                 |
|-------|--------------------|
| True  | 当前单元文本为斜体          |
| False | 缺省。当前单元文本为正常（不是斜体） |

## 说明

对该属性所做的更改会影响到当前单元或者当前选定，这取决于 FillStyle 属性的设置值。

## 请参阅

FillStyle 属性（MSHFlexGrid），CellFontBold 属性，CellFontName 属性，CellFontSize 属性，CellFontUnderline 属性，CellFontWidth 属性。

## 示例

无论何时，当 MSHFlexGrid 控件在焦点中时，下面的代码将当前单元的文本设置为斜体样式：

注意：如果您使用 MSFlexGrid，请用 “MSFlexGrid1” 替换

“MSHFlexGrid1”。

```
Sub MSHFlexGrid1_GotFocus()
```

```
    MSHFlexGrid1.CellFontItalic = True
```

```
End Sub
```

## CellFontName 属性

该属性返回或设置当前单元文本的字体名。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellFontName [=string ]

CellFontName 属性的语法有以下部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>String</i> | 字符串表达式，命名一种可用的字面     |

说明

对该属性所做的更改会影响到当前单元或者当前选定，这取决于 FillStyle

属性的设置值。

请参阅

FillStyle 属性 (MSHFlexGrid) , CellFontBold 属性, CellFontItalic 属性, CellFontSize 属性, CellFontUnderline 属性, CellFontWidth 属性。

示例

无论何时，当 MSHFlexGrid 控件在焦点中时，下面的代码将当前单元的文本设置为一种特定的字体类型：

**注意：**如果您使用 MSFlexGrid，请用 “MSFlexGrid1” 替换 “MSHFlexGrid1”。

```
Sub MSHFlexGrid1_GotFocus()  
    MSHFlexGrid1.CellFontName = Screen.Fonts(3)  
    MSHFlexGrid1.Text = Screen.Fonts(3) '显示字体名。  
    ' name.  
End Sub
```

## CellFontSize 属性

该属性返回或设置以点数为单位的当前单元文本的尺寸。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellFontSize [= *value* ]

CellFontSize 属性的语法有以下部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>value</i>  | Single 类型。数值表达式，它指定了当前单元文本的尺寸 |

说明

对该属性所做的更改会影响到当前单元或者当前选定，这取决于 FillStyle 属性的设置值。

请参阅

FillStyle 属性（MSHFlexGrid），CellFontBold 属性，CellFontItalic 属性，CellFontName 属性，CellFontUnderline 属性，CellFontWidth 属性。

示例

无论何时，当 MSHFlexGrid 控件在焦点中时，下面的代码将当前单元的文本的尺寸设置为 12 个点：

注意：如果您使用 MSFlexGrid，请用 “MSFlexGrid1” 替换 “MSHFlexGrid1”。

```
Sub MSHFlexGrid1_GotFocus()  
    MSHFlexGrid1.CellFontSize = 12  
End Sub
```

## CellFontStrikeThrough 属性

该属性返回或者设置一个值，该值决定了是否将 **Strikethrough** 样式应用到当前单元文本中。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellFontStrikeThrough [= *boolean*]

CellFontStrikeThrough 属性语法有如下部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象                          |
| <i>boolean</i> | 布尔表达式，它决定了是否将 <b>StrikeThrough</b> 样式应用到单元文本中 |

# 设置

*Boolean* 的设置值是：

| 设置值   | 描述                                      |
|-------|---|
| True  | 将 <b>StrikeThrough</b> 样式应用到单元文本中       |
| False | （缺省的）不将 <b>StrikeThrough</b> 样式应用到单元文本中 |

# 说明

对 **CellFontStrikeThrough** 属性所做的更改会影响到当前单元或者当前选定，这取决于 **FillStyle** 属性的设置值。

请参阅

**CellFontUnderline** 属性。

# 示例

无论何时，当 **MSHFlexGrid** 控件在焦点中时，下面的代码将当前单元文本设置为 **StrikeThrough** 样式：

注意：如果您使用 **MSFlexGrid**，请用 “**MSFlexGrid1**” 替换 “**MSHFlexGrid1**”。

```
Sub MSHFlexGrid1_GotFocus
    MSHFlexGrid1.CellFontStrikeThrough = 1
End Sub
```



# CellFontUnderline 属性

该属性返回或者设置一个值，该值指定了是否将下划线样式应用到当前单元文本中。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellFontUnderline [= *boolean*]

CellFontUnderline 属性语法有如下部分：

| 部分             | 描述                                 |
|----------------|------------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象               |
| <i>boolean</i> | 布尔表达式，它决定了是否将 Underline 样式应用到单元文本中 |

设置

*Boolean* 的设置值是：

| 设置值   | 描述                           |
|-------|------------------------------|
| True  | 将 Underline 样式应用到单元文本中       |
| False | （缺省的）不将 Underline 样式应用到单元文本中 |

## 说明

对 **CellFontUnderline** 属性所做的更改会影响到当前单元或者当前选定，这取决于 **FillStyle** 属性的设置值。

## 示例

无论何时，当 **MSHFlexGrid** 控件在焦点中时，下面的代码将当前单元的文本设置为 **Underline** 样式：

注意：如果您使用 **MSFlexGrid**，请用 “**MSFlexGrid1**” 替换 “**MSHFlexGrid1**”。

```
Sub MSHFlexGrid1_GotFocus  
    MSHFlexGrid1.CellFontUnderline = 1  
End Sub
```

## CellFontWidth 属性

该属性返回或设置以点数表示的当前单元文本宽度。在设计时不可用。

## 应用于

**MSHFlexGrid** 控件，**MSFlexGrid** 控件。

## 语法

*object*.CellFontWidth [= *value* ]

CellFontWidth 属性的语法有以下部分：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                 |
| <i>value</i>  | Single 类型。数值表达式，它指定了当前单元文本字体所要求的点数宽度 |

## 说明

对 CellFontWidth 属性所做的更改会影响到当前单元或者当前选定，这取决于 FillStyle 属性的设置值。

## 请参阅

FillStyle 属性（MSHFlexGrid），CellFontBold 属性，CellFontItalic 属性，CellFontName 属性，CellFontUnderline 属性，CellFontSize 属性。

## 示例

当 MSHFlexGrid 控件在焦点中时，下面的代码对当前单元的文本宽度进行设置：

**注意：**如果您使用 MSFlexGrid，请用 “MSFlexGrid1” 替换

“MSHFlexGrid1”。

```
Sub MSHFlexGrid1_GotFocus()  
    MSHFlexGrid1.CellFontWidth = 5  
End Sub
```

## CellHeight,CellLeft,CellTop 以及 CellWidth 属性(MSHFlexGrid)

这些属性返回以缇(*twips*)为单位的当前单元的位置和大小。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellHeight [=value]

*object*.CellLeft [=value]

*object*.CellTop [=value]

*object*.CellWidth [=value]

CellHeight,CellLeft,CellTop 和 CellWidth 属性语法有如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个数值表达式，指出当前单元的返回位置或大小   |

### 说明

如果想对单元内编辑进行枚举的话，这些属性是很有用的。可以通过捕获 **MSHFlexGrid** 控件的 **KeyPress** 事件，把文本框或者其它控件放在当前单元的上方，并让用户对其内容进行编辑。

不管窗体的 **ScaleMode** 设置值怎样，这些属性的返回值总是以缇为单位。

请参阅

**FillStyle** 属性（**MSHFlexGrid**）。

## CellPicture 属性

该属性返回或设置在当前单元或者一群单元中显示的图像。在设计时不可用。

应用于

**MSHFlexGrid** 控件，**MSFlexGrid** 控件。

## 语法

*object*.CellPicture [= *picture* ]

CellPicture 属性的语法有以下部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象                          |
| <i>picture</i> | 一个位图、图标或元文件图像。也可以把它赋给另一个控件的 <b>Picture</b> 属性 |

## 说明

在运行时，可以通过在位图、图标或者元文件上使用 **LoadPicture** 函数，或者将之赋给另一个控件的 **Picture** 属性来对该属性进行设置。

对这个属性所做的更改会影响到当前单元或者当前选定，这取决于 **FillStyle** 属性的设置值。

每个单元都可能包含文本和图片。文本和图片的相对位置是由 **CellAlignment** 和 **CellPictureAlignment** 属性所决定的。

## 请参阅

**FillStyle** 属性（**MSHFlexGrid**），**CellAlignment** 属性，**CellPictureAlignment** 属性。

## 示例

在以下示例中，将 Visual Basic 图标库中的图标加载到 MSHFlexGrid 控件的两个单元中。可以使用任意两个图标。将下面的代码粘贴到某个窗体（该窗体带有 MSHFlexGrid 控件）的声明部分。按 F5 键来运行该程序，然后单击该窗体。

```
Private Sub Form_Click ()  
    '加载图标。  
    MSHFlexGrid1.Row = 1  
    MSHFlexGrid1.Col = 1  
    Set MSHFlexGrid1.CellPicture = _  
        LoadPicture("Icons\Computer\Trash02a.ico")  
    MSHFlexGrid1.Row = 1  
    MSHFlexGrid1.Col = 2  
    Set MSHFlexGrid1.CellPicture = _  
        LoadPicture("Icons\Computer\Trash02b.ico")  
End Sub
```

## CellPictureAlignment 属性

该属性返回或设置在单元或者一群选定单元中图片的对齐方式。在设计时

不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellPictureAlignment [= *value* ]

CellPictureAlignment 属性的语法有以下部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象            |
| <i>Value</i>  | 整数或者常量，它指定了单元中的图片如何对齐，如下“设置”中所述 |

设置

value 的设置值是：

| 常量                    | 值 | 描述     |
|-----------------------|---|--------|
| flexAlignLeftTop      | 0 | 左边顶端对齐 |
| flexAlignLeftCenter   | 1 | 左边中间对齐 |
| flexAlignLeftBottom   | 2 | 左边底端对齐 |
| flexAlignCenterTop    | 3 | 居中顶端对齐 |
| FlexAlignCenterCenter | 4 | 居中中间对齐 |



续表

|                                    |   |        |
|------------------------------------|---|--------|
| <code>flexAlignCenterBottom</code> | 5 | 居中底端对齐 |
| <code>flexAlignRightTop</code>     | 6 | 右边顶端对齐 |
| <code>flexAlignRightCenter</code>  | 7 | 右边中间对齐 |
| <code>flexAlignRightBottom</code>  | 8 | 右边底端对齐 |

说明

对该属性所做的更改会影响到当前单元或者当前选定，这取决于 `FillStyle` 属性的设置值。

`FillStyle` 属性对于 `CellAlignment` 必须设置为 1 (Repeat)，以对齐 `MSHFlexGrid` 中选定的单元范围。

请参阅

`FillStyle` 属性 (`MSHFlexGrid`)，`CellPicture` 属性。

示例

下面的代码用常量值来将当前单元的图片对齐方式设置为右边中间。

注意：如果您使用 `MSFlexGrid`，请用 “`MSFlexGrid1`” 替换 “`MSHFlexGrid1`”。

```
Sub Form1_Load
    MSHFlexGrid1.CellPictureAlignment = _
```

```
flexAlignRightCenter  
End Sub
```

## CellTextStyle 属性

该属性返回或设置指定单元或者一群单元上文本的三维样式。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.CellTextStyle [= *value* ]

CellTextStyle 属性的语法有以下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                         |
| <i>value</i>  | 整数或者常量，它指定了 CellTextStyle 属性中的一个常量，如下“设置”中所述 |

# 设置

value 的设置值为:

| 常量                  | 值 | 描述            |
|---------------------|---|---------------|
| FlexTextFlat        | 0 | 缺省的。平面的（普通文本） |
| FlexTextRaised      | 1 | 凸起的           |
| FlexTextInset       | 2 | 下陷的           |
| FlexTextRaisedLight | 3 | 轻微凸起的         |
| FlexTextInsetLight  | 4 | 轻微下陷的         |

# 说明

设置值 1 和 2 最好用于大字体和粗体字。设置值 3 和 4 最好用于小的常规字体。单元的外观同时还受到背景颜色的影响；许多背景颜色显不出凸起或者下陷的效果。

对这个属性所做的更改会影响到当前单元或者当前选定，这取决于 FillStyle 属性的设置值。

# 请参阅

FillStyle 属性（MSHFlexGrid），Text 属性（MSHFlexGrid），TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader 属性（MSHFlexGrid）。

## 示例

下面的代码用常量值来将当前单元或者当前选定的文本样式设置为 **Inset**。

**注意：**如果您使用 `MSFlexGrid`，请用“`MSFlexGrid1`”替换“`MSHFlexGrid1`”。

```
Sub MSHFlexGrid1_GotFocus  
    MSHFlexGrid1.CellTextStyle = flexTextInset  
End Sub
```

## CellType 属性 (MSHFlexGrid)

返回 `MSHFlexGrid` 当前活动单元格的类型。

应用于

`MSHFlexGrid` 控件。

语法

*object*.CellType [=value]

`CellType` 属性的语法包含以下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象         |
| <i>value</i>  | 一个 Long 数值，它指定了活动单元格的位置，如“设置”中所示 |

## 设置

*value* 的设置范围是：

| 常量                      | 值 | 描述                  |
|-------------------------|---|---------------------|
| FlexCellTypeStandard    | 0 | 缺省设置。单元格为标准单元格      |
| FlexCellTypeFixed       | 1 | 单元格被包含在固定的行或者列中     |
| FlexCellTypeHeader      | 2 | 单元格为一个数据带区的标头       |
| FlexCellTypeIndent      | 3 | 单元格被用于一列中，该列缩进了数据带区 |
| FlexCellTypeUnpopulated | 4 | 单元格中尚未被填入数据         |

## Clear 方法 (MSHFlexGrid)

该方法清除 MSHFlexGrid 的内容。这包括所有文本、图片和单元格式。Clear 方法并不影响 MSHFlexGrid 上的行数和列数。

## 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

## 语法

*object*.Clear

Clear 方法的语法有一部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

## 说明

为了删除单元，但并非仅仅清除它们，可以在每个要删除的行上使用 **RemoveItem** 方法。

## 请参阅

**RemoveItem** 方法（MSHFlexGrid）。

## 示例

在以下示例中，无论何时，当用户在单元上单击时，都将“Flex”放到当前单元中。无论何时，当用户双击时，都清除 **MSHFlexGrid** 控件。可以单击 F5 键来运行该程序。

**注意：** 如果您使用 **MSFlexGrid**，请用“**MSFlexGrid1**”替换“**MSHFlexGrid1**”。

```
Private Sub Form1_Load ()
```

```
MSHFlexGrid1.Rows = 8
MSHFlexGrid1.Cols = 5
End Sub
Private Sub MSHFlexGrid1_Click ()
    '将文本放到当前单元中。
    MSHFlexGrid1.Text = "Flex"
End Sub

Private Sub MSHFlexGrid1.DblClick ()
    MSHFlexGrid1.Clear
End Sub
```

## ClearStructure 方法 (MSHFlexGrid)

清除 **MSHFlexGrid** 中的全部映射信息，映射信息与被显示的列的显示顺序和名称有关。该信息可以在设计时进行设置，请使用“属性页”对话框的 **Band** 选项卡，或者使用快捷菜单上的“检索结构”命令。

该方法可以用来将 **MSHFlexGrid** 重新设置为一种已知的状态。如果用户在程序中改变了 **MSHFlexGrid** 的数据源（而且该数据源具有不同的结构），通过执行本方法即可正确地显示数据。

应用于

MSHFlexGrid 控件。

语法

*object*.ClearStructure

ClearStructure 方法的语法只有一个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>Object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |

说明

如果在改变结构的时候没有清除映射信息，那么 MSHFlexGrid 将试图使用现有的映射信息显示新的数据源。只有在映射信息中指定的列才能在 MSHFlexGrid 中显示数据。其它列也会显示出来，但是不会显示任何数据。

### Clip 属性 (MSHFlexGrid)

该属性返回或设置 MSHFlexGrid 控件的选定区域中单元的内容。在设计时不可用。



应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.Clip* [= *string* ]

Clip 属性的语法包含以下部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>Object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>String</i> | 字符串表达式，它带有已选定区域的内容   |

说明

*string* 可能包含多个行和列的内容。在 *string* 中，制表符，即 Chr (9)，或者说常量 vbTab 指示了某一行中的一个新单元，而回车换行符，即 Chr (13)，或者说常量 vbCR 则指示了一个新行的开始。可以用 Chr 函数或者 vb 常量来将这些字符嵌入到字符串中。

当把数据放入 MSHFlexGrid 控件时，只是选定的单元受影响。如果选定区域中的单元比 *string* 中描述的更多，那么余下的单元就被单独放到一边。如果 *string* 中描述的单元比选定区域中的更多，那么 *string* 的未使用部分就被忽略。

请参阅

ColPosition, RowPosition 属性。

示例

在该示例中，无论何时，当用户选定了一组单元时，将“James”、“Nancy”以及“Lisa”放到已选定单元中。无论何时，当用户双击时，都清除 MSHFlexGrid 控件。可以单击 F5 键来运行该程序。

注意：如果您使用 MSFlexGrid，请用“MSFlexGrid1”替换“MSHFlexGrid1”。

```
Private Sub Form1_Load ()  
    MSHFlexGrid1.Rows = 8  
    MSHFlexGrid1.Cols = 5  
End Sub
```

```
Private Sub MSHFlexGrid1_MouseUp (Button As Integer, _  
Shift as Integer, X As Single, Y As Single)  
    Dim myStr As String  
    myStr = "James" + Chr(9) + "Nancy" + Chr(9) + "Lisa"  
    MSHFlexGrid1.Clip = myStr  
End Sub
```

## Col 和 Row 属性 (MSHFlexGrid)

这两个属性返回或设置 MSHFlexGrid 中活动单元的坐标。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.Col* [= *number* ]

*object.Row* [= *number* ]

Col 和 Row 的语法包含以下部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>number</i> | Long 类型值。它指定了活动单元的位置 |

说明

可以用这些属性来指定 MSHFlexGrid 中的单元，或者找到包含当前单元的那个行或者列。行和列是从 0 开始计数的，对于行来说，以顶端为起始，而对于列来说，则以左边为起始。

对这些属性进行设置会自动对 **RowSel** 和 **ColSel** 进行重置，这样，所选定的就变成了当前单元。因此，为了指定一个块选定，必须首先对 **Row** 和 **Col** 进行设置，然后对 **RowSel** 和 **ColSel** 进行设置。

当前单元的值（由 **Col** 和 **Row** 的设置值所决定的），就是包含在那个单元中的文本。可以用 **TextMatrix** 属性在不改变已选定的 **Row** 和 **Col** 属性的情况下，对单元的值进行修改。

请参阅

**Cols, Rows** 属性（**MSHFlexGrid**），**ColSel**, **RowSel** 属性，**SelChange** 事件（**MSHFlexGrid**），**Sort** 属性（**MSHFlexGrid**），**Text** 属性（**MSHFlexGrid**），**TextMatrix** 属性，**Bands** 属性（**MSHFlexGrid**）。

示例

在该示例中，将“**Here**”放到当前单元中，然后将活动单元更改为第 3 行中的第 3 个单元，并将“**There**”放到那个单元中。可以单击 **F5** 键来运行该程序，然后单击网格。

**注意：**如果您使用 **MSFlexGrid**，请用“**MSFlexGrid1**”替换“**MSHFlexGrid1**”。

```
Private Sub Form1_Load ()  
    MSHFlexGrid1.Rows = 8  
    MSHFlexGrid1.Cols = 5
```

End Sub

Private Sub MSHFlexGrid1\_Click ()

  '将文本放到当前单元中。

  MSHFlexGrid1.Text = "Here"

  '  将文本放到第 3 行，第 3 列。

  MSHFlexGrid1.Col = 2

  MSHFlexGrid1.Row = 2

  MSHFlexGrid1.Text = "There"

End Sub

## ColAlignmentFixed 属性

返回或设置 MSHFlexGrid 的一列中固定单元格中的数据的对齐方式。

应用于

MSHFlexGrid 控件。

语法

*object*.ColAlignmentFixed(*index*) [=*value*]

ColAlignmentFixed 属性的语法包含以下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象        |
| <i>index</i>  | 一个 Long 数值，它指定了列                |
| <i>value</i>  | 一个整数，它确定固定单元格中的数据的对齐方式，如“设置”中所示 |

## 设置

*value* 的设置值包括：

| 常量                                 | 值 | 描述            |
|------------------------------------|---|---------------|
| <code>flexAlignLeftTop</code>      | 0 | 单元格的内容左、顶部对齐  |
| <code>flexAlignLeftCenter</code>   | 1 | 单元格的内容左、居中对齐  |
| <code>flexAlignLeftBottom</code>   | 2 | 单元格的内容左、底部对齐  |
| <code>flexAlignCenterTop</code>    | 3 | 单元格的内容居中、顶部对齐 |
| <code>flexAlignCenterCenter</code> | 4 | 单元格的内容居中、居中对齐 |
| <code>flexAlignCenterBottom</code> | 5 | 单元格的内容居中、底部对齐 |
| <code>flexAlignRightTop</code>     | 6 | 单元格的内容右、顶部对齐  |
| <code>flexAlignRightCenter</code>  | 7 | 单元格的内容右、居中对齐  |
| <code>flexAlignRightBottom</code>  | 8 | 单元格的内容右、底部对齐  |

## 说明

可以使用 `FixedCols` 和 `FixedRows` 属性来定义一列中的单元格。

请参阅

FixedCols, FixedRows 属性 (MSHFlexGrid)。

## ColAlignment、ColAlignmentBand、ColAlignmentHeader 属性(MSHFlexGrid)

返回或者设置列中数据的对齐方式。该列可以是一个标准列、带区中的一列或者标头中的一列。该属性在设计时是不可使用的（除非通过 **FormatString** 属性间接地使用）。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.ColAlignment(*number*) [=*value*]

*object*.ColAlignmentBand(*number*) [=*value*]

*object*.ColAlignmentHeader(*number*) [=*value*]

ColAlignment、ColAlignmentBand 和 ColAlignmentHeader 属性的语法包括：

| 部分            | 描述  |
|---------------|---|
| <i>Object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象                        |
| 部分            | 描述  |
| <i>Number</i> | 一个 Long 数值，它指定了列在 <code>MSHFlexGrid</code> 中的编号 |
| <i>value</i>  | 一个整数或者常量，它指定了列中的数据对齐方式，如“设置”中所示                 |

设置

*value* 的设置值包括：

| 常量                                 | 值 | 描述                                       |
|------------------------------------|---|--|
| <code>flexAlignLeftTop</code>      | 0 | 单元格的内容左、顶部对齐                             |
| <code>flexAlignLeftCenter</code>   | 1 | 字符串的缺省对齐方式。单元格的内容左、居中对齐                  |
| <code>flexAlignLeftBottom</code>   | 2 | 单元格的内容左、底部对齐                             |
| <code>flexAlignCenterTop</code>    | 3 | 单元格的内容居中、顶部对齐                            |
| <code>flexAlignCenterCenter</code> | 4 | 单元格的内容居中、居中对齐                            |
| <code>flexAlignCenterBottom</code> | 5 | 单元格的内容居中、底部对齐                            |
| <code>flexAlignRightTop</code>     | 6 | 单元格的内容右、顶部对齐                             |
| <code>flexAlignRightCenter</code>  | 7 | 数值的缺省对齐方式。单元格的内容右、居中对齐                   |
| <code>flexAlignRightBottom</code>  | 8 | 单元格的内容右、底部对齐                             |
| <code>flexAlignGeneral</code>      | 9 | 单元格的内容按一般方式进行对齐。字符串按“左、居中”显示，数字按“右、居中”显示 |



## 说明

任何一列都可以有与其他列不同的对齐方式。**ColAlignment** 属性将影响指定列的所有单元格，包括位于固定行中的那些单元格。

如果需要设置单个单元格的对齐方式，可以使用 **CellAlignment** 属性。如果需要在设计时设置列的对齐方式，可以使用 **FormatString** 属性。

如果 **MSHFlexGrid** 处于垂直模式，那么设置 **ColAlignment(3)** 可以影响到多个带区中的若干列。

## 请参阅

**FormatString** 属性，**CellAlignment** 属性。

## 示例

下面的代码用常量值将第 3 列的对齐方式设置为“右，中”对齐。

**注意：**如果您使用 **MSFlexGrid**，请用“**MSFlexGrid1**”替换“**MSHFlexGrid1**”。

```
Sub Form1_Load ()
```

```
    MSHFlexGrid1.ColAlignment(3) = flexAlignRightCenter
```

```
End Sub
```

## ColData 和 RowData、BandData 属性(MSHFlexGrid)

注意：如果您使用 MSFlexGrid，请用“MSFlexGrid1”替换“MSHFlexGrid1”。

```
Private Sub Form_Click  
    Set Picture1.Picture = MSHFlexGrid1.Picture  
End Sub
```

这些属性返回或设置跟每个行和列相关联的一个任意的 long 类型的值。在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
object.ColData(number) [=value]  
object.RowData(number) [=value]  
object.BandData(number) [=value]
```

ColData、RowData 和 BandData 属性的语法有如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                       |
| <i>number</i> | Long 类型值。在保存或者检索数据的 MSHFlexGrid 控件中的行号或者列号 |
| <i>value</i>  | Long 类型值。它指定 ColData 或者 RowData 数组的内容      |

## 说明

可以用 RowData 和 ColData 属性来使指定的数值跟 MSHFlexGrid 控件上的每一行或者列相关联,然后就可以在代码中使用这些数值来识别各项。

例如，可以将包含总数的行添加到 MSHFlexGrid 中，并通过将它们 RowData 属性设置为一个非 0 值来识别这些行。为了以后对这些总数进行更新，可以通过扫描 RowData 数组并删除相应的行来删除过时的总数。

对 RowData 属性的另一个典型应用是将一个索引放到数据结构的数组中，这些数据结构跟在每一行上所描述的各项相关联。

## 示例

下列代码显示了您如何使用 PictureType 属性捕获内存外错误并自动切换到单色模式。此外，还显示了如何在 MSHFlexGrid 内部创建一个图片，该图片仅包含当前选择。

```
Sub CopySelectedPictureToClipboard (myFlex As _
MSHFlexGrid)
```

```
Dim i As Integer, tr As Long, lc As Long, _  
hl As Integer  
' 准备操作。  
MyFlex.Redraw =False      ' 消除抖动。  
hl =MyFlex.HighLight      ' 保存当前设置。  
tr =MyFlex.TopRow  
lc =MyFlex.LeftCol  
MyFlex.HighLight =0      ' 在图片上没有突出显示。  
' 隐藏未选择行和列。  
' (在 RowData/ColData  
' 属性中保存原始大小。)  
For i =MyFlex.FixedRows To MyFlex.Rows - 1  
    If i < MyFlex.Row Or i > MyFlex.RowSel Then  
        MyFlex.RowData(i) =MyFlex.RowHeight(i)  
        MyFlex.RowHeight(i) =0  
    End If  
Next  
For i =MyFlex.FixedCols To MyFlex.Cols - 1  
    If i < MyFlex.Col Or i > MyFlex.ColSel Then  
        MyFlex.ColData(i) =MyFlex.ColWidth(i)  
        MyFlex.ColWidth(i) =0  
    End If
```

Next

' 滚动到左上角。

MyFlex.TopRow =MyFlex.FixedRows

MyFlex.LeftCol =MyFlex.FixedCols

' 复制图片。

clipboard.Clear

On Error Resume Next

MyFlex.PictureType =0 ' 彩色。

clipboard.SetData MyFlex.Picture

If Error <> 0 Then

    MyFlex.PictureType =1 ' 单色。

    clipboard.SetData MyFlex.Picture

Endif

' 恢复控件。

For i =MyFlex.FixedRows To MyFlex.Rows - 1

    If i < MyFlex.Row Or i > MyFlex.RowSel Then

        MyFlex.RowHeight(i) =MyFlex.RowData(i)

    End If

Next

For i =MyFlex.FixedCols To MyFlex.Cols - 1

    If i < MyFlex.Col Or i > MyFlex.ColSel Then

        MyFlex.ColWidth(i) =MyFlex.ColData(i)

```
End If
Next
MyFlex.TopRow =tr
MyFlex.LeftCol =lc
MyFlex.HighLight =hl
MyFlex.Redraw =True
End Sub
```

下列示例显示了如何将 `MSHFlexGrid` 的 `Picture` 属性设置为一个 `PictureBox` 控件：

注意：如果您使用 `MSFlexGrid`，请用“`MSFlexGrid1`”替换“`MSHFlexGrid1`”。

```
Private Sub Form_Click ()
    Set Picture1.Picture =MSHFlexGrid1.Picture
End Sub
```

## ColHeader 属性 (MSHFlexGrid)

指出是否为指定的带区显示标头。

应用于

MSHFlexGrid 控件。

语法

*object.ColHeader(value)*

ColHeader 属性语法有如下部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象               |
| <i>value</i>  | 可选的。一个 Long 值，指出是否显示包含标头的带区，如“设置”中所描述的 |

设置

*value* 的设置值是：

| 设置                | 值 | 描述                               |
|-------------------|---|----------------------------------|
| flexColHeaderOn   | 0 | 为每一个带区显示标头                       |
| flexColHeaderOff  | 1 | 不为带区显示标头                         |
| flexColHeaderOnce | 2 | 显示带区的标头。如果带区是折叠的，只显示一个标头。不显示重复标头 |

## ColHeaderCaption 属性 (MSHFlexGrid)

指定需要显示在指定的列和带区的标头中的标题。

应用于

MSHFlexGrid 控件。

语法

*object.ColHeaderCaption(number, index) [=string]*

ColHeaderCaption 属性的语法包含以下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>Object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象      |
| <i>Number</i> | 一个 Long 数值，它指明的带区中包含了需要设置标题的列 |
| <i>Index</i>  | 一个 Long 数值，它指明了特定的列           |
| <i>String</i> | 一个字符串表达式，它指定了一种可用的字体类型        |

## ColIsVisible 属性

该属性返回或设置一个值，该值指示了指定列在当前是否是可见的。



应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.CollIsVisible(index) [=boolean]*

CollIsVisible 属性的语法包含以下部分：

| 部分             | 描述                   |
|----------------|----------------------|
| <i>Object</i>  | 对象表达式，其值是“应用于”列表中的对象 |
| <i>Index</i>   | 一个 Long 值，它指定列       |
| <i>Boolean</i> | 一个布尔表达式，指出指定的列是否可见   |

设置

Boolean 的设置值是：

| 设置值   | 描述             |
|-------|----------------|
| True  | 缺省的。指定列在当前是可见的 |
| False | 指定列在当前是不可见的    |

## Collapse 事件 (MSHFlexGrid)

当用户在网格内折叠一行时发生。MSHFlexGrid 的 Col 和 Row 属性包含用于折叠带区的单元格。

应用于

MSHFlexGrid 控件。

语法

Private Sub *object\_* Collapse(*Boolean*)

Collapse 事件的语法包含如下部分：

| 部分             | 描述                                |
|----------------|-----------------------------------|
| <i>Object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象          |
| <i>Boolean</i> | 一个布尔表达式。如果把 Cancel 设置为 True，折叠被取消 |

## CollapseAll 方法 (MSHFlexGrid)

在 MSHFlexGrid 内折叠指定带区的所有行。

应用于

MSHFlexGrid 控件。

语法

*object.CollapseAll(number)*

CollapseAll 属性的语法包含如下部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象    |
| <i>number</i> | 可选的。一个 Long 值，它指定包含要折叠的行的带区 |

### ColPos 属性

以缇为单位，返回控件的左上角和一个指定列的左上角之间的距离。

应用于

MSHFlexGrid 控件。

语法

*object.ColPos(index)*

ColPos 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>index</i>  | 一个 Long 值，它指定列           |

## ColPosition,RowPosition 属性

- **ColPosition** — 设置一个 MSHFlexGrid 列的位置，允许移动列到指定的位置。
- **RowPosition** — 设置一个 MSHFlexGrid 行的位置，允许移动行到指定的位置。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.ColPosition(index, number) [= value]*

*object.RowPosition(number) [= value]*

**注意：**由于这一控件的限制，上面的 ColPosition 语法不能在 MSFlexGrid 中使用。如果正在使用 MSFlexGrid，采用以下语法

*object.ColPosition(number) [= value]*。

ColPosition 和 RowPosition 属性的语法包含如下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                                    |
| <i>index</i>  | 一个 Long 值，它指定要移动的列<br>注意不可应用于 MSFlexGrid                    |
| <i>number</i> | 一个 Long 值，它指定包含要移动的列的带区。可选的<br>注意在 MSFlexGrid 中，这是要移动的列或行的数 |
| <i>value</i>  | 整数。一个数值表达式，它指定列或行的新的位置                                      |

## 说明

在 MSFlexGrid 中，当 BandNumber 未指定时，缺省为 0。因此，当网格未绑定到一个分层结构的记录集时，BandNumber 为 0 和不指定 BandNumber 两者都会得到相同的结果。注意 BandNumber 是与 MSFlexGrid 向后兼容的一个可选的参数。

索引和设置必须对应有效的行或列数（在 0 到 Rows -1 或 Cols -1 范围），否则会产生一个错误。

当一个行或列被使用 RowPosition 和 ColPosition 属性移动时，所有的格式化的信息随着它一起移动。这包括宽、高、对齐、颜色和字体属性。如果想

只移动文本，可以使用 **Clip** 属性。

请参阅

**Cols, Rows** 属性（**MSHFlexGrid**），**Clip** 属性（**MSHFlexGrid**）。

示例

在实现时，当用户单击某列时，下列代码将引起这列移动到第一位置（最左边一列）。

**注意：**如果正在使用 **MSFlexGrid**，用 “**MSFlexGrid1**” 代替 “**MSHFlexGrid1**”。

```
Sub MSHFlexGrid1_Click ()  
    MSHFlexGrid1.ColPosition(MSFlexGrid1.MouseCol) = 0  
End Sub
```

## **Cols, Rows** 属性（**MSHFlexGrid**）

- **Cols** — 返回或设置在一个 **MSHFlexGrid** 中的总列数。
- **Rows** — 返回或设置在一个 **MSHFlexGrid** 中的总行数。**Rows** 属性也返回或设置在 **MSHFlexGrid** 中的每一个带区中的总列数。

**注意：****MSFlexGrid** 不包含带区。如果在 **MSFlexGrid** 中使用 **Rows** 属

性编程，只能返回或设置总列数或总行数。也由于这一控件的限制，下面的 Rows 语法不能在 MSFlexGrid 中使用。如果正在使用 MSFlexGrid，采用如下语法 *object.Rows [= value]*。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
object.Cols [= value]  
object.Rows(number) [= value]
```

Cols 和 Rows 属性的语法包含如下部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象      |
| <i>number</i> | 一个 Long 值，它指定获取或是设置总列数的带区。可选的 |
|               | 注意这不可应用于 MSHFlexGrid          |
| <i>value</i>  | 一个 Long 值，它指定列数或行数            |

说明

可以使用这些属性在运行时动态地扩充或收缩 MSHFlexGrid。  
行和列的最小数是 0。最大数受计算机的可用内存限制。

Cols 的值必须至少比 FixedCols 的值大一，除非它们两者都被设置为 0。  
Rows 的值必须至少比 FixedRows 的值大一，除非它们两者都被设置为 0。

当 *number* 未被指定时，缺省为 0。因此，当 MSHFlexGrid 未被绑定到一个分层结构的记录集时，使用 0 或不指定 *number* 两者都会得到相同的结果。注意 *number* 是与 MSFlexGrid 向后兼容的一个可选的参数。

请参阅

ColPosition, RowPosition 属性，FormatString 属性，Col, Row 属性 (MSHFlexGrid)。

示例

下面的示例把单词 “Here”放进当前单元，更改活动单元为第三行的第三单元格，然后把 “There”放进那个单元。要运行该示例，按 F5 键，然后单击 MSHFlexGrid。

注意：如果正在使用 MSFlexGrid，用 “MSFlexGrid1”代替 “MSHFlexGrid1”。

```
Private Sub Form_Load ()  
    MSHFlexGrid1.Rows = 8  
    MSHFlexGrid1.Cols = 5  
End Sub
```

```
Private Sub MsFlexGrid1_Click ()
```



```
' 把文本放在当前单元中。  
MSHFlexGrid1.Text = "Here"  
' 把文本放在第三行，第三列。  
MSHFlexGrid1.Col = 2  
MSHFlexGrid1.Row = 2  
MSHFlexGrid1.Text = "There"
```

```
End Sub
```

## ColSel、RowSel 属性

**ColSel** — 为一定范围的单元格返回或设置的起始列和或终止列。

**RowSel** — 为一定范围的单元格返回或设置的起始行和或终止行。  
这些属性在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.ColSel* [= *value*]

*object.RowSel* [= *value*]

ColSel 和 RowSel 语法包含如下部分：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象             |
| <i>value</i>  | 一个 Long 值，为一定范围的单元格指定起始行或列，或者指定终止行或列 |

## 说明

可以使用这些属性编程选择 **MSHFlexGrid** 的一个特定区域，或读出用户选择进入代码的区域的维数。

**MSHFlexGrid** 游标在 **Row**、**Col** 位置的单元格中。**MSHFlexGrid** 选择的是在行 **Row** 和 **RowSel** 之间以及列 **Col** 和 **ColSel** 之间的区域。注意 **RowSel** 可能在 **Row** 的上面或下面，而 **ColSel** 可能在 **Col** 的左边或右边。

无论什么时候设置 **Row** 和 **Col** 属性，**RowSel** 和 **ColSel** 都自动地重新设置，因此游标变为当前选择。要从代码中选择一块单元格，必须首先设置 **Row** 和 **Col** 属性，然后设置 **RowSel** 和 **ColSel**。

## 请参阅

**SelChange** 事件 (**MSHFlexGrid**)，**Sort** 属性 (**MSHFlexGrid**)，**Col,Row** 属性 (**MSHFlexGrid**)。

## 示例

下面的代码把 ColSel 属性的 *value* 返回到 MSHFlexGrid1 的第一单元格中。这个值随着用户单击不同的单元组的选择而更改。

**注意：**如果正在使用 MSFlexGrid，用 “MSFlexGrid1” 代替 “MSHFlexGrid1”。

```
Private Sub MSHFlexGrid1_MouseUp _  
(Button As Integer, Shift As Integer, x As Single, _  
y As Single)  
    MSHFlexGrid1.Text = MSHFlexGrid1.ColSel  
End Sub
```

## ColWidth 属性 (MSHFlexGrid)

以缇为单位，返回或设置指定带区中的列宽。这一属性在设计时不可用。

**注意：**当使用 MSFlexGrid 时，这一属性以缇为单位返回或设置指定列的宽度。由于这一控件的限制，下面的 ColWidth 语法也不能在 MSFlexGrid 中使用。如果正在使用 MSFlexGrid，采用如下语法 *object.ColWidth(number) [= value]*。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.ColWidth(index, number) [= value]*

ColWidth 属性的语法包含如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                                 |
| <i>index</i>  | 一个 Long 值，它指定要更改哪一列的宽度<br>注意这是不可应用于 MSFlexGrid           |
| <i>number</i> | 一个 Long 值，指定包含列的带区。可选的<br>注意在 MSFlexGrid 中，它是一个指定列的数值表达式 |
| <i>value</i>  | 一个数值表达式，它以缇为单位指定特定列的宽度                                   |

说明

可以使用这一属性在运行时设置任何列的宽度。对于在设计时设置列宽的指令，请参阅 FormatString 属性。

可以通过把 ColWidth 设置为 0 来创建不可见列，或设置为 -1 重新设置列宽为它的缺省值（它取决于当前字体的尺寸）。

当 *number* 没有指定时，它缺省为 0。因此，当 MSHFlexGrid 未被绑定

到分层结构的记录集时，使用 0 和未指定 *number* 两者都会得到相同的结果。注意 *number* 是与 MSFlexGrid 向后兼容的一个可选的参数。

请参阅

FormatString 属性，AllowUserResizing 属性。

示例

下面的代码设置第一列尺寸的 *value*。如果 AllowUserResizing 属性为 True，该值按照用户调整 Column 1 的大小而更改。

注意：如果正在使用 MSFlexGrid，用 “MSFlexGrid1” 代替 “MSHFlexGrid1”。

```
Sub Form1_Load()
```

```
    MSHFlexGrid1.AllowUserResizing = True
```

```
End Sub
```

```
Sub MSHFlexGrid1_MouseUp (Button As Integer, Shift As _
```

```
Integer, X As Single, Y As Single)
```

```
    MSHFlexGrid1.Text = MSHFlexGrid1.ColWidth(0)
```

```
End Sub
```

## Compare 事件

当 `MSHFlexGrid` 的 `Sort` 属性被设置为 `Custom Sort (9)` 时发生，因此用户可以自定义排序进程。

应用于

`MSHFlexGrid` 控件，`MSFlexGrid` 控件。

语法

```
Private Sub object_Compare(row1, row2, cmp)
```

`Compare` 事件的语法包含如下部分：

| 部分            | 描述                                      |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                |
| <i>row1</i>   | 一个 <code>Long</code> 整数，它指定正在比较的一对行的第一行 |
| <i>row2</i>   | 一个 <code>Long</code> 整数，它指定正在比较的一对行的第二行 |
| <i>cmp</i>    | 一个整数，它表示每一对的排序次序。如设置值中所述                |

设置

事件句柄必须比较 `row1` 和 `row2`，并把 `cmp` 设置为：

| 设置值 | 描述                                  |
|-----|-------------------------------------|
| - 1 | 如果 <i>row1</i> 应该显示在 <i>row2</i> 前面 |
| 设置值 | 描述                                  |
| 0   | 如果两行相等或任一行都可以显示在另一行之前               |
| 1   | 如果 <i>row1</i> 应该显示在 <i>row2</i> 之后 |

### 说明

当 **Sort** 属性被设置为 9（自定义排序）时，**Compare** 事件对 **MSHFlexGrid** 中的每一对行发生一次。因为 **Compare** 事件使用行号代替了文本值，可以对那一行比较任何属性值，包括 **RowData**。

**注意：**虽然自定义排序比内置的排序慢，但它们提供了使用任何列或使用任何单元属性排序一行的灵活性。

### 请参阅

**Sort** 属性（**MSHFlexGrid**）。

## DataField 属性 (MSHFlexGrid)

返回绑定到指定带区中指定列的字段名称。这一属性是只读的。

应用于

MSHFlexGrid 控件。

语法

*object.DataField(number, index) [= string]*

DataField 属性的语法包含如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                   |
| <i>number</i> | 一个 Long 值，它指定包含所影响的列的带区                    |
| <i>index</i>  | 一个 Long 值，它或者是指定获取的列或者是设置绑定字段              |
| <i>string</i> | 一个字符串表达式，其值为由数据提供商指定的 Recordset 对象中的一个字段名称 |

说明

当 *number* 未指定时，缺省值为 0。因此，当 MSHFlexGrid 未绑定到分层结构的记录集时，使用 0 和不指定 *number* 两者都会得到相同的结果。

如果 MSHFlexGrid 未绑定且 *number* 和 *index* 是有效的，DataField 返回一个空字符串。

如果在 DataProvider 属性被更改之后执行了一个 Refresh，Recordset 对象可能会有不同的字段。这可能使绑定列的 DataField 设置无效并产生一个可捕



获错误。

## EnterCell 事件

当前活动单元更改到一个不同单元时发生。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

Private Sub *object*\_EnterCell()

EnterCell 事件的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说明

在一个固定行上单击将引起这个事件在那一行的第一个不固定列上发生。拖动鼠标经过一个单元格不会引起这一事件的发生。

## Expand 事件 (MSHFlexGrid)

当用户在 MSHFlexGrid 里面扩充一行时发生。MSHFlexGrid 的 Col 和 Row 属性包含用于扩充带区的单元格。

应用于

MSHFlexGrid 控件。

语法

Private Sub *object*\_Expand(*Boolean*)

Expand 事件的语法包含如下部分：

| 部分             | 描述                                  |
|----------------|-------------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象            |
| <i>boolean</i> | 一个布尔表达式。如果开发者设置 Cancel 为 True，折叠被取消 |

## ExpandAll 方法 (MSHFlexGrid)

扩充在 MSHFlexGrid 里面指定带区的所有行。

应用于

MSHFlexGrid 控件。

语法

*object.ExpandAll(number)*

ExpandAll 方法的语法包含如下部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                 |
| <i>number</i> | 可选的。一个 Long 值，它指定包含要扩充的行的带区。如果未指定，缺省值为-1 |

### FillStyle 属性 (MSHFlexGrid)

返回或设置一个值，它决定不管是设置 Text 属性还是一个单元的格式属性，都把更改应用到 MSHFlexGrid 里的所有选定单元。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
object.FillStyle [= value]
```

FillStyle 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数或常量，它指定填充样式。如设置值中所述  |

设置

对 *value* 的设置值如下所示：

| 常量             | 值 | 描述                               |
|----------------|---|----------------------------------|
| flexFillSingle | 0 | 单个。更改 Text 或任何单元属性只影响到活动单元。这是缺省值 |
| flexFillRepeat | 1 | 重复。更改 Text 或任何单元属性影响到所有选定的单元     |

说明

无论什么时候想将一个单元的更改应用到 MSHFlexGrid 里面所有单元，这一属性必须设置为 1 (重复)。

请参阅

Text 属性(MSHFlexGrid) , CellAlignment 属性, CellBackColor, CellForeColor

属性，CellFontBold 属性，CellFontItalic 属性，CellFontName 属性，CellFontSize 属性，CellFontWidth 属性，CellHeight, CellLeft, CellTop, CellWidth 属性 (MSHFlexGrid)，CellPicture 属性，CellPictureAlignment 属性，CellTextStyle 属性。

## 示例

下面的代码允许不同地格式化各个单元。

**注意：**如果正在使用 MSFlexGrid，用 “MSFlexGrid1” 代替 “MSHFlexGrid1”。

```
Sub Form1_Load ()  
    MSHFlexGrid1.FillStyle = 0  
End Sub
```

## FixedCols、FixedRows 属性 (MSHFlexGrid)

**FixedCols** — 返回或设置在一个 MSHFlexGrid 里面的固定列的总数。  
**FixedRows** — 返回或设置在一个 MSHFlexGrid 里面的固定行的总数。  
按照缺省规定，MSHFlexGrid 有一个固定列和一个固定行。

## 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

# 语法

*object.FixedCols* [= *value*]

*object.FixedRows* [= *value*]

FixedCols 和 FixedRows 属性的语法包含如下内容：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个 Long 值，它指定固定列或固定行的总数  |

# 说明

当在 MSHFlexGrid 中滚动其它列或行时，固定的列和行是固定不变的。可以指定零个或多个固定的列行。另外，可以选择固定列和行的颜色、字体、网格线和文本样式。

如果 SelectionMode 属性是一个数值型值，当运行时选择一个固定列或固定行时，在那一行或列里面的所有单元被选定。

如果 AllowUserResizing 属性是一个数值型值，可以在运行时重新调整固定行或固定列的尺寸。

固定列和固定行在电子数据表应用程序中，用来显示行号以及列名或字母。

请参阅

ColAlignmentFixed 属性。

## 示例

下面的代码把第一行和第一、第二列设置为固定的。

**注意：**如果正在使用 `MSFlexGrid`，用 “`MSFlexGrid1`” 代替 “`MSHFlexGrid1`”。

```
Sub Form1_Load ()  
    MSHFlexGrid1.FixedCols = 2  
    MSHFlexGrid1.FixedRows = 1  
End Sub
```

## FocusRect 属性

返回或设置一个值，它决定 `MSHFlexGrid` 是否应该围绕着当前单元格绘制一个焦点矩形。

应用于

`MSHFlexGrid` 控件，`MSFlexGrid` 控件。

语法

*object*.FocusRect [= *value*]

`FocusRect` 属性的语法包含如下部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象   |
| <i>value</i>  | 一个整数或常量，它指定焦点矩形的样式。如设置值中所述 |

## 设置

对 `value` 的设置值如下所示：

| 常量                          | 值 | 描述                      |
|-----------------------------|---|-------------------------|
| <code>flexFocusNone</code>  | 0 | 当前单元格周围无焦点矩形            |
| <code>flexFocusLight</code> | 1 | 当前单元格周围有一个浅色的焦点矩形。此为缺省值 |
| <code>flexFocusHeavy</code> | 2 | 当前单元格周围有一个加重的焦点矩形       |

## 说明

如果绘制了一个焦点矩形，当前单元格涂成背景颜色，就象在多数电子数据表和网格中一样。否则，当前单元涂成选择的颜色，因此没有焦点矩形可以看到哪一个单元被选定。

## 示例

下面的代码使用常量值，对活动单元格设置焦点矩形为浅色的矩形。

注意：如果正在使用 `MSFlexGrid`，用“`MSFlexGrid1`”代替“`MSHFlexGrid1`”。



```
Sub Form1_Load ()  
    MSHFlexGrid1.FocusRect = flexFocusLight  
End Sub
```

## Font、FontBand、FontFixed、FontHeader 属性(MSHFlexGrid)

**Font** — 返回或设置缺省的字体或个别单元格的字体。

**FontBand** — 返回或设置带区中文本的字体。

**FontFixed** — 返回或设置固定单元格中文本的字体。

**FontHeader** — 返回或设置标头文本的字体。

应用于

MSHFlexGrid 控件。

语法

*object*.Font

*object*.FontBand

*object*.FontFixed

*object*.FontHeader

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

# FontWidth,FontWidthBand,FontWidthFixed,FontWidthHeader 属性 (MSHFlexGrid)

以磅为单位，返回或设置用于显示在一个 MSHFlexGrid 里面的文本字体的宽度，或者是用于网格的带区、固定或标头区域字体的宽度。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
object.FontWidth [= value]  
object.FontWidthBand [= value]  
object.FontWidthFixed [= value]  
object.FontWidthHeader [= value]
```

FontWidth,FontWidthBand,FontWidthFixed 和 FontWidthHeader 属性的语法包含如下部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>Object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>Value</i>  | 单精度。一个数值表达式，它为当前字体指定首选的磅宽 |

## 说明

字体宽度通常由 Windows 选取来匹配选定的字体宽度，并且提供一个标准的外观比率。但是，MSHFlexGrid 使得您能够指定比缺省字体更窄或宽的字体。这就使您或者是显示更多信息或者是强调一个单元成为可能。

当指定一个字体宽度时，Windows 选择或生成一种字体来匹配您的需求。为了得到最好的结果，使用 TrueType 字体。

要恢复缺省的字体宽度，请把这一属性设置为 0。

要设置个别单元或单元范围的字体，请使用 CellFontBold、CellFontItalic、CellFontName、CellFontSize 和 CellFontWidth 属性。

## ForeColor, ForeColorBand, ForeColorFixed, ForeColorHeader 和 ForeColorSel 属性

返回或设置用于在 MSHFlexGrid 的每一部分绘制文本的颜色。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.ForeColor [= *color*]

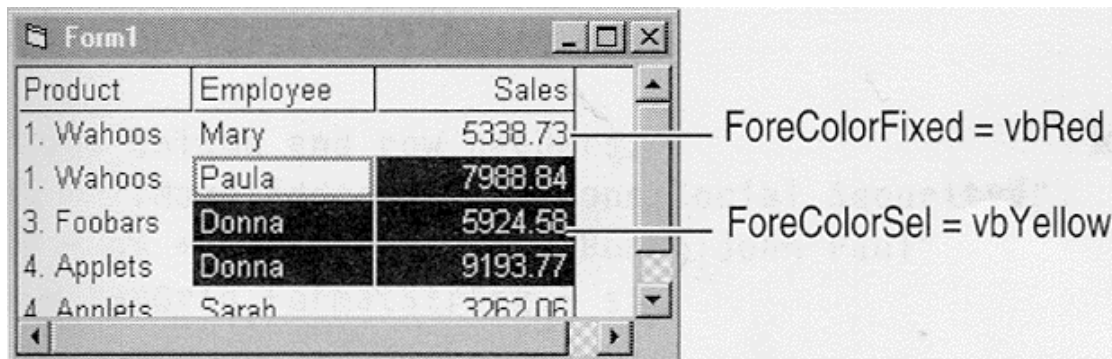
*object.ForeColorBand* [= *color*]  
*object.ForeColorFixed* [= *color*]  
*object.ForeColorHeader* [= *color*]  
*object.ForeColorSel* [= *color*]

ForeColor, ForeColorBand, ForeColorFixed, ForeColorHeader 和 ForeColorSel 属性的语法包含如下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                                  |
| <i>color</i>  | 一个值或常量，它决定用在 <b>MSHFlexGrid</b> 的可滚动的、固定的、带区或标头区域中涂绘文本的颜色 |

## 说明

下面的图画显示了这一属性所引用 **MSHFlexGrid** 的部分：



| Product    | Employee | Sales   |
|------------|----------|---------|
| 1. Wahoos  | Mary     | 5338.73 |
| 1. Wahoos  | Paula    | 7988.84 |
| 3. Foobars | Donna    | 5924.58 |
| 4. Applets | Donna    | 9193.77 |
| 4. Applets | Sarah    | 3262.06 |

ForeColorFixed = vbRed

ForeColorSel = vbYellow

使用 `ForeColor` 属性设置所有不固定单元格的文本颜色。

使用 `CellForeColor` 属性设置个别单元格的文本颜色。

## FormatString 属性

Form1

| Region | Product | Employee | Sales |
|--------|---------|----------|-------|
|--------|---------|----------|-------|

Form1

|                  |  |
|------------------|--|
| Name             |  |
| Address          |  |
| Telephone        |  |
| Social Security# |  |

Form1

|  | Name      | Address | Telephone | Social Security# |
|--|-----------|---------|-----------|------------------|
|  | Robert    |         |           |                  |
|  | Jimmy     |         |           |                  |
|  | Bonzo     |         |           |                  |
|  | John Paul |         |           |                  |

设置 `MSHFlexGrid` 的列宽、对齐方式、固定行文本和固定列文本。

应用于

`MSHFlexGrid` 控件，`MSFlexGrid` 控件。

语法

`object.FormatString [= string]`

`FormatString` 属性的语法包含如下部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象      |
| <i>string</i> | 一个字符串表达式，为格式化在行和列中的文本。如在说明中所述 |

说明

在设计时，`MSHFlexGrid` 语法分析和解释 `FormatString` 来获得如下信息：行和列的数目、行和列标头的文本、列宽和列对齐方式。

`FormatString` 属性包含由管道字符 (`|`) 分隔的段。管道字符之间的文本定义一列并且也可能包含特定的对齐字符。这些字符使整个列左对齐(<)、居中(^)或右对齐(>)。另外，根据缺省规定文本被指定给行 0，且文本宽度定义每一列的宽度。

**FormatString** 属性可能包含一个分号(;).这使得字符串的余下部分被解释为行标头和行宽度信息。另外，根据缺省规定文本被指定给列 0，且最长的字符串定义列 0 的宽度。

**MSHFlexGrid** 创建附加的行和列来驻留由 **FormatString** 定义的所有字段。如果只有几个字段被指定，附加的行和列不被删除。要删除附加的行和列，设置 **Rows** 和 **Cols** 属性。

请参阅

**Cols,Rows** 属性 (**MSHFlexGrid**)，**ColWidth** 属性 (**MSHFlexGrid**)，**ColAlignment**, **ColAlignmentBand**, **ColAlignmentHeader** 属性 (**MSHFlexGrid**)。

示例

下面的示例说明 **FormatString** 属性如何工作。

**注意：** 如果正在使用 **MSFlexGrid**，用 “**MSFlexGrid1**” 代替 “**MSHFlexGrid1**”。

' 设置列标头。

```
s$ = "<Region |<Product |<Employee |>Sales "
```

```
MSHFlexGrid1.FormatString = s$
```

' 设置行标头（注意开始的分号）。

```
s$ = ";Name|Address|Telephone|Social Security#"
```

```
MSHFlexGrid1.FormatString = s$
```



' 设置列和行标头。

```
s$ = "|Name|Address|Telephone|Social Security#"
```

```
s$ = s$ + ";;Robert|Jimmy|Bonzo|John Paul"
```

```
MSHFlexGrid.FormatString = s$
```

## GridColor, GridColorBand, GridColorFixed, GridColorHeader, GridColorIndent 和 GridColorUnpopulated 属性

返回或设置用在 MSHFlexGrid 的单元格、带区、标头、缩进或未填充区域之间的线的颜色。

注意：如果正在使用 MSFlexGrid，只有 GridColor 和 GridColorFixed 属性是可用的。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
object.GridColor [= color]
```

```
object.GridColorBand [= color]
```

```
object.GridColorFixed [= color]
```

```
object.GridColorHeader [= color]
```

*object*.GridColorIndent [= *color*]

*object*.GridColorUnpopulated [= *color*]

GridColor, GridColorBand, GridColorFixed, GridColorHeader, GridColorIndent 和 GridColorUnpopulated 属性的语法包含如下部分:

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 一个对象表达式, 其值为 “应用于” 列表中的一个对象              |
| <i>color</i>  | 一个值或常量, 决定用于涂绘 MSFlexGrid 滚动或固定区域中网格线的颜色 |

### 说明

GridColor 属性只有在 GridLines 属性被设置为 1 (线) 时才能使用。

GridColorFixed 属性只有在 GridLinesFixed 被设置为 1 (线) 时才能使用。凸起和凹入的网格线总是按黑和白颜色绘制。

### 请参阅

GridLines, GridLinesBand, GridLinesFixed, GridLinesHeader, GridLinesIndent, GridLinesUnpoluated 属性 (MSHFlexGrid)。

GridLines, GridLinesBand, GridLinesFixed, GridLinesHeader, GridLinesIndent 和 GridLinesUnpopulated 属性 (MSHFlexGrid)

返回或设置一个值，它决定是否在单元格、带区、标头、缩进或未填充区域之间绘制线。这些属性也决定在 MSHFlexGrid 中绘制线的类型。

注意：如果正在使用 MSFlexGrid，只有 GridLines 和 GridLinesFixed 属性是可用的。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.GridLines [= *value*]

*object*.GridLinesBand [= *value*]

*object*.GridLinesFixed [= *value*]

*object*.GridLinesHeader [= *value*]

*object*.GridLinesIndent [= *value*]

*object*.GridLinesUnpopulated [= *value*]

GridLines、GridLinesBand、GridLinesFixed、GridLinesHeader、GridLinesIndent 和 GridLinesUnpopulated 属性的语法包含如下部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象   |
| <i>value</i>  | 一个整数或常量，它指定绘制的线的类型，如设置值中所描 |

## 设置

对 `value` 的设置值如下所示：

| 常量                          | 值 | 描述   |
|-----------------------------|---|--|
| <code>flexGridNone</code>   | 0 | 在单元之间没有线。在 <code>MSFlexGrid</code> 中这是 <code>GridLines</code> 缺省值                |
| <code>flexGridFlat</code>   | 1 | 单元之间的线样式被设置为正常的、平面的线。在 <code>MSHFlexGrid</code> 中这是 <code>GridLines</code> 缺省值   |
| <code>flexGridInset</code>  | 2 | 单元格之间的线样式被设置为凹入线。在 <code>MSFlexGrid</code> 中这是 <code>GridLinesFixed</code> 的缺省值  |
| <code>flexGridRaised</code> | 3 | 单元格之间的线样式被设置为凸起线。在 <code>MSHFlexGrid</code> 中这是 <code>GridLinesFixed</code> 的缺省值 |

## 说明

当 `GridLines` 属性设置为 1（线）时，线的颜色由 `GridColor` 属性决定。凸起和凹入的网格线总是黑色和白色。

## GridLineWidth 属性(MSHFlexGrid)

以像素为单位，返回或设置显示在单元格、带区、标头、缩进或未填充区域之间的线的宽度。

注意：对于 MSFlexGrid 控件只有 GridLineWidth 属性是可用的。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.GridLineWidth [= *value*]

*object*.GridLineWidthBand [= *value*]

*object*.GridLineWidthFixed [= *value*]

*object*.GridLineWidthHeader [= *value*]

*object*.GridLineWidthIndent [= *value*]

*object*.GridLineWidthUnpopulated [= *value*]

GridLineWidth ， GridLineWidthBand ， GridLineWidthFixed ， GridLineWidthHeader, GridLineWidthIndent 和 GridLineWidthUnpopulated 属性的语法包含如下部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>value</i>  | 一个数值表达式，它以像素为单位为当前线指定首选宽度 |

## HighLight 属性 (MSHFlexGrid)

决定选定的单元格是否在 MSHFlexGrid 中突出显示。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.HighLight* [= *value*]

HighLight 属性的语法包含如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                         |
| <i>value</i>  | 一个整数或常量，它指定什么时候 MSHFlexGrid 应该突出显示选定的单元格。如设置值中所述 |

## 设置

对 *value* 的设置值如下所示：

| 常量                                  | 值 | 描述                  |
|-------------------------------------|---|---------------------|
| <code>flexHighlightNever</code>     | 0 | 选定的单元格上没有突出显示       |
| <code>flexHighlightAlways</code>    | 1 | 选定的单元格总是突出显示（缺省设置值） |
| <code>flexHighlightWithFocus</code> | 2 | 突出显示只在控件有焦点时有效      |

## 说明

当这一属性被设置为 0 并且一些单元被选定，没有可视的暗示或强调来指示选定的单元格。

## LeaveCell 事件

当前活动单元变更到一个不同的单元之前立即发生。

## 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

## 语法

Private Sub *object*\_LeaveCell()

LeaveCell 事件的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

## 说明

这一事件用于验证一个单元的内容。

当焦点移动到一个不同的控件时这一事件不发生。

## LeftCol 属性 (MSHFlexGrid)

返回或设置 MSHFlexGrid 中最左边可见的不固定列。这一属性在设计时不可用。

## 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。



# 语法

```
object.LeftCol [= value]
```

LeftCol 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数，它指定最左边的列            |

# 说明

这一属性可以在编程时使用，实现在 MSHFlexGrid 中的滚动。使用 TopRow 属性来决定 MSHFlexGrid 的最顶端可视行。

请参阅

Scroll 事件（MSHFlexGrid），TopRow 属性（MSHFlexGrid）。

# MergeCells 属性

返回或设置一个值，决定包含相同内容的单元是否应该跨越多行或多列分组在一个单个单元中。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.MergeCells [=value]*

*MergeCells* 属性的语法包含如下部分内容：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象    |
| <i>value</i>  | 一个整数或常量，指定单元分组（合并）。如设置值中所指定 |

设置

*value* 的设置值如下所示：

| 常量                           | 值 | 描述                              |
|------------------------------|---|---------------------------------|
| <i>flexMergeNever</i>        | 0 | 不显示。包含相同内容的单元不分组。这是缺省设置         |
| <i>flexMergeFree</i>         | 1 | 自由。包含相同内容的单元总是合并                |
| <i>flexMergeRestrictRows</i> | 2 | 限制行。只有行中包含相同内容的相邻单元（向当前单元左边）才合并 |

续表

|                                       |   |                                     |
|---------------------------------------|---|-------------------------------------|
| <code>flexMergeRestrictColumns</code> | 3 | 限制列。只有列中包含相同内容的相邻单元（向当前单元上方）才合并     |
| <code>flexMergeRestrictBoth</code>    | 4 | 限制行和列。只有在行中（向左）或在列中（向上）包含相同内容的单元才合并 |

说明

合并单元的能力使得您能够以一种清晰、简明的方式显示数据。可以连同排序和 `MSHFlexGrid` 的列序函数一起合并使用单元。

要使用 `MSHFlexGrid` 的单元合并能力：

把 `MergeCells` 设置为除 0 以外的一个值（设置值之间的区别在示例中解释）。

为要合并行和列把 `MergeRow` 和 `MergeCol` 的数组属性设置为 `True`。

当使用单元合并能力时，`MSHFlexGrid` 合并包含相同内容的单元。无论什么时候单元的内容更改，合并都自动的更新。

当 `MergeCells` 被设置为除 0 （不显示）以外的一个值时，突出显示的选择自动关闭。这样做是为加速重画，也是因为包含合并单元范围的选择可能导致不可预料的结果。

请参阅

`MergeCol`, `MergeRow` 属性。

## 示例

下面的示例显示基本的 MergeCells 属性。

不合并

MergeCells =0

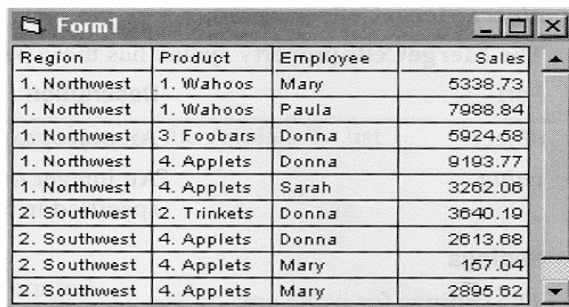
MergeRow(0) =True

MergeRow(1) =True

MergeRow(2) =True

MergeRow(3) =False

这是标准视图。



| Region       | Product     | Employee | Sales   |
|--------------|-------------|----------|---------|
| 1. Northwest | 1. Wahoos   | Mary     | 5338.73 |
| 1. Northwest | 1. Wahoos   | Paula    | 7988.84 |
| 1. Northwest | 3. Foobars  | Donna    | 5924.58 |
| 1. Northwest | 4. Applets  | Donna    | 9193.77 |
| 1. Northwest | 4. Applets  | Sarah    | 3262.06 |
| 2. Southwest | 2. Trinkets | Donna    | 3640.19 |
| 2. Southwest | 4. Applets  | Donna    | 2613.68 |
| 2. Southwest | 4. Applets  | Mary     | 157.04  |
| 2. Southwest | 4. Applets  | Mary     | 2895.62 |

自由合并

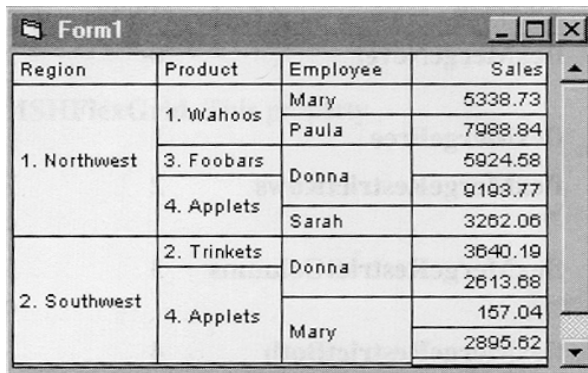
MergeCells =1

MergeRow(0) =True

MergeRow(1) =True

MergeRow(2) =True

MergeRow(3) =False



| Region       | Product    | Employee | Sales   |
|--------------|------------|----------|---------|
| 1. Northwest | 1. Wahoos  | Mary     | 5338.73 |
|              |            | Paula    | 7988.84 |
|              | 3. Foobars | Donna    | 5924.58 |
|              | 4. Applets |          | 9193.77 |
| 2. Southwest | 4. Applets | Sarah    | 3262.06 |
|              |            | Donna    | 3640.19 |
|              | 4. Applets | Donna    | 2613.68 |
|              |            | Mary     | 157.04  |
|              |            | Mary     | 2895.62 |

注意第三个 employee 单元(Donna)如何合并它左边的 products 和它右边的 sales。

限制合并

MergeCells =2

MergeRow(0) =True

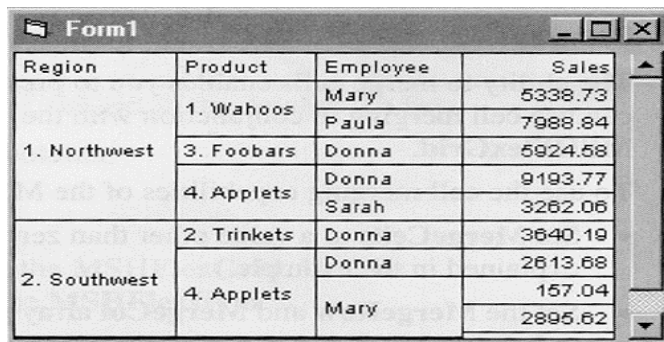
MergeRow(1) =True

MergeRow(2) =True

MergeRow(3) =False

注意第三个 employee

单元(Donna)不再合并 sales。



| Region       | Product     | Employee | Sales   |
|--------------|-------------|----------|---------|
| 1. Northwest | 1. Wahoos   | Mary     | 5338.73 |
|              |             | Paula    | 7988.84 |
| 1. Northwest | 3. Foobars  | Donna    | 5924.68 |
|              |             | Donna    | 9193.77 |
|              |             | Sarah    | 3262.06 |
| 2. Southwest | 2. Trinkets | Donna    | 3640.19 |
|              |             | Donna    | 2613.68 |
| 2. Southwest | 4. Applets  |          | 157.04  |
|              |             | Mary     | 2895.62 |

## MergeCol、MergeRow 属性

返回或设置一个值，决定哪些行和列可以把它们的内容合并。要使用 MergeCells 属性，这些属性必须为 True。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.MergeCol(number) [=Boolean]

*object*.MergeRow(number) [=Boolean]

MergeCol 和 MergeRows 属性的语法包含如下部分：

| 部分             | 描述                                |
|----------------|-----------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象          |
| <i>number</i>  | 一个 Long 值，指定 MSHFlexGrid 中的列或行    |
| <i>boolean</i> | 一个 Boolean 值，指定当相邻单元显示相同内容时合并是否发生 |

## 设置

*Boolean* 的设置值如下所示：

| 部分    | 描述  |
|-------|---|
| True  | 当相邻单元显示相同内容时，行向左合并或列向上合并                        |
| False | 当相邻单元显示相同内容时，单元不合并。这是 MergeCol 和 MergeRow 缺省设置值 |

## 说明

如果 MergeCells 属性被设置为非零值，具有相同值的相邻单元，只有它们都在一行并且 MergeRow 属性被设置为 True，或都在一列且 MergeCol 属性被设置为 True 时才合并。

关于 MSHFlexGrid 的合并功能的详细信息，请参阅 MergeCells 属性。

请参阅

MergeCells 属性。

## MouseCol、MouseRow 属性

按行和列坐标返回鼠标的当前位置。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.MouseCol [=value]

*object*.MouseRow [=value]

MouseCol 和 MouseRow 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 指定当前鼠标位置的行和列坐标           |

说明

使用这些属性编程来决定鼠标的位置。在对单独单元显示与上下文相关的帮助以及测试用户是否已经单击一个固定行或列时，这些属性是有用的。

## Name 属性 (MSHFlexGrid)

返回或设置合计字段或用来分组的字段的名称。

注意：这一属性不能应用于 MSFlexGrid。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.Name [=string]

Name 属性的语法包含如下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>string</i> | 一个字符串表达式，包含合计字段或用来分组的字段的名称。对于 GroupingField 对象，缺省值是当前 Command 对象中的第一个字段的名称。对于合计，缺省值是 Aggregate 加上一个唯一表示该字段的数字 |

说明

这一属性不能有与将创建合计的字段相同的名称。对于带分组的



AggregateOn，这是当前 Command 对象中任意字段的名称。对于带有关系的 AggregateOn，这是父 Command 对象中任意字段的名称。

GroupingField 对象的名称必须是存在当前 Command 对象中的一个字段名。

ADO Name 属性用于生成 SHAPE 命令。

## Picture 属性 (MSHFlexGrid)

返回 MSHFlexGrid 的一幅图片。这一图片适合打印、保存到磁盘、复制到剪贴板或给一个不同控件赋值。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.Picture [=picture]

Picture 属性的语法包含如下部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>picture</i> | 一个位图，显示 MSHFlexGrid      |

## 说明

位图图片是 Project Form 窗口上整个 MSHFlexGrid 的一个快照。因此，它可能是非常大的。有两种方法来减小位图图片的尺寸。一个选项是创建您的 MSHFlexGrid 的一节的图片。要完成这些，写一个例程隐藏所有不想显示的元素、获取图片，然后恢复 MSHFlexGrid。

另一种选择，可以设置 **PictureType** 属性为 1（单色）。但是，这不仅仅减少内存占用数量，同时也减低了图片的分辨率。

## 请参阅

**PictureType** 属性。

## PictureType 属性

返回或设置由 **Picture** 属性生成的图片的类型。

## 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

`object.PictureType [=type]`

PictureType 属性的语法包含如下部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象      |
| <i>type</i>   | 一个整数或常量，指定应该被生成的图片的类型。如设置值中所述 |

设置

*type* 的设置值如下所示：

| 常量                    | 值 | 描述                    |
|-----------------------|---|-----------------------|
| flexPictureColor      | 0 | 产生一个高品质的全色的图像         |
| flexPictureMonochrome | 1 | 产生一个较低品质、单色的占用少量内存的图像 |

请参阅

Picture 属性（MSHFlexGrid）。

## Redraw 属性

返回或设置一个值，决定 `MSHFlexGrid` 是否应该在每次更改后自动地重绘。

应用于

`MSHFlexGrid` 控件，`MSFlexGrid` 控件。

语法

*object.Redraw* [=*Boolean*]

`Redraw` 属性的语法包含如下部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象                            |
| <i>boolean</i> | 一个布尔表达式，指定 <code>MSHFlexGrid</code> 是否应该在每次更改后自动地重绘 |

设置

对 `Boolean` 的设置如下所示：

| 部分    | 描述                             |
|-------|--------------------------------|
| True  | MSHFlexGrid 在每次更改后自动地重绘。这是缺省设置 |
| False | MSHFlexGrid 在每次更改后不重绘          |

## 说明

可以在代码中使用这一属性，来减少当 MSHFlexGrid 的内容进行广泛更新时的闪烁。

## 示例

下面的代码关闭重绘，对 MSHFlexGrid 的内容做了几次更改，然后打开重绘来显示结果。

**注意：**如果正在使用 MSFlexGrid，用 “MSFlexGrid1” 代替 “MSHFlexGrid1”。

```
Dim i As Integer
```

```
' 冻结 MSHFlexGrid 避免闪烁。
```

```
MSHFlexGrid.Redraw =False
```

```
' 更新 MSHFlexGrid 内容。
```

```
For i =MSHFlexGrid1.FixedRows To MSHFlexGrid1.Rows - 1
```

```
MSHFlexGrid1.TextMatrix(i, 1) =GetName(i, 1)
```

```
MSHFlexGrid1.TextMatrix(i, 2) =GetName(i, 2)
```

```
Next
```

' 显示结果

MSHFlexGrid1.Redraw =True

## RemoveItem 方法 (MSHFlexGrid)

运行时从 MSHFlexGrid 中删除一行。这一属性不支持命名的参数。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object.RemoveItem(index, number)*

RemoveItem 方法的语法包含如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                             |
| <i>index</i>  | 一个整数，表示 MSHFlexGrid 中要删除的行。对于第一行，<br><i>index</i> =0 |
| <i>number</i> | 一个 Long 值，指定要从中删除行的带区                                |

## 说明

这一方法删除指定的整个行。要不删除行来清除数据，使用 **Clear** 方法。

如果 **BandDisplay** 属性被设置为水平并且 **MSHFlexGrid** 被绑定到一个分层的记录集，*number* 是必需的。如果 **BandDisplay** 属性被设置为竖直，*number* 只在带区意向不明时才需要。

在包含子记录的带区内删除一个行时，子记录被自动地删除。

## 请参阅

**Clear** 方法（**MSHFlexGrid**）。

## RowColChange 事件 (MSHFlexGrid)

当前活动单元更改到不同单元时发生。

## 应用于

**MSHFlexGrid** 控件，**MSFlexGrid** 控件。

## 语法

```
Private Sub object_RowColChange()
```

**RowColChange** 事件的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

## 说明

当活动单元更改时，下列事件顺序发生：首先是 `LeaveCell`、接着是 `EnterCell`、最后是 `RowColChange`。当用户单击一个新的单元时 `RowColChange` 事件发生。用户拖动一个选择经过 `MSHFlexGrid` 时不发生。

## RowExpandable, RowExpanded 属性 (MSHFlexGrid)

指定当前的行是否被扩充或被折叠。当前行由 `Col` 和 `Row` 列属性定义。  
`RowExpandable` 属性在设计时不可用。

## 应用于

`MSHFlexGrid` 控件，`MSFlexGrid` 控件。

## 语法

*object*.RowExpandable [=Boolean]

*object*.RowExpanded [=Boolean]

`RowExpandable` 和 `Row Expanded` 属性的语法包含如下部分：



| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，指定当前行是否可扩充或当前已经被扩充 |

设置

Boolean 的设置值如下所示：

| 部分    | 描述          |
|-------|-------------|
| True  | 当前行可扩充      |
| False | 当前行被折叠且不可扩充 |

### RowHeight 属性 (MSHFlexGrid)

以缬为单位，返回或设置指定行的高度。这一属性在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.RowHeight(*number*) [=*value*]

RowHeight 属性的语法包含如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象                       |
| <i>number</i> | 一个整数，设置 <b>MSHFlexGrid</b> 中的行号，或为 -1 来立即设置所有行 |
| <i>value</i>  | 一个单精度数值表达式，以缇为单位指定行的高度                         |

## 说明

可以设置 **RowHeight** 为 0 创建不可见的行，或为-1 来重新设置行的高度为它的缺省值。缺省的行的高度根据当前字体尺寸改变。

**RowHeight** 属性独立于窗体的缩放模式。

## 请参阅

**RowHeightMin** 属性。

## RowHeightMin 属性

以缇为单位，为整个控件返回或设置最小行高。

## 应用于

**MSHFlexGrid** 控件，**MSFlexGrid** 控件。

# 语法

`object.RowHeightMin [=value]`

RowHeightMin 属性的语法包含如下部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象        |
| <i>value</i>  | 一个单精度数值表达式，为 MSHFlexGrid 指定最小行高 |

# 说明

如果正在使用小字体却想要行变高，使用这一属性。设置这一属性比使用 RowHeight 属性设置个别的行高容易。

RowHeight 属性独立于窗体的缩放模式。

请参阅

RowHeight 属性（MSHFlexGrid）。

# RowIsVisible 属性

返回或设置一个值，决定一个指定行是否可见。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.RowIsVisible(*index*) [=Boolean]

RowIsVisible 属性的语法包含如下部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>index</i>   | 一个 Long 值，指定列            |
| <i>Boolean</i> | 一个布尔表达式，决定一个指定行当前是否可见    |

设置

对 Boolean 的设置如下所示：

| 设置值   | 描述               |
|-------|------------------|
| True  | 指定行当前可见。这是缺省的设置值 |
| False | 指定行当前不可见         |

## RowPos 属性

以缬为单位，返回 **MSHFlexGrid** 左上角和指定行的左上角之间的距离。

应用于

**MSHFlexGrid** 控件，**MSFlexGrid** 控件。

语法

*object*.RowPos(*index*)

RowPos 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>index</i>  | 一个 Long 值，指定列            |

## RowSizingMode 属性 (MSHFlexGrid)

返回或设置一个值，描述 **MSHFlexGrid** 中行的大小调整方式。

应用于

**MSHFlexGrid** 控件。

语法

`object.RowSizingMode [=value]`

RowSizing 属性的语法包含如下部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>value</i>  | 一个整数或常量，指定行尺寸的设置值。如设置值中所述 |

设置

*value* 的设置值如下所示：

| 常量                                 | 值 | 描述  |
|------------------------------------|---|---|
| <code>flexRowSizeIndividual</code> | 0 | 调整行的尺寸只更改那一行的高度。这是缺省的设置值                  |
| <code>flexRowSizeAll</code>        | 1 | 调整行的尺寸更改 <code>MSHFlexGrid</code> 中所有行的高度 |

说明

这一属性只在 `AllowUserResize` 被设置为 `Rows` 或 `Both` 时才使用。

Scroll 事件 (MSHFlexGrid)

当 `MSHFlexGrid` 的内容滚动时发生。这可以使用滚动条、键盘或通过编

程更改 TopRow 或 LeftCol 属性来实现。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

Private Sub *object*\_Scroll()

Scroll 事件的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说明

如果 ScrollTrack 属性为 True，这一事件在用户拖动或滚动鼠标时发生。  
如果 ScrollTrack 属性为 False 时，拖动一旦完成这一事件发生。

请参阅

ScrollBars 属性（MSHFlexGrid），ScrollTrack 属性。

## ScrollBars 属性 (MSHFlexGrid)

返回或设置一个值，决定 MSHFlexGrid 是否有水平和/或数值的滚动条。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.ScrollBars [=value]

ScrollBars 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数或常量，指定滚动条的类型。如设置值中所述 |

设置

*value* 的设置值如下所示：

| 常量                   | 值 | 描述                         |
|----------------------|---|----------------------------|
| flexScrollNone       | 0 | MSHFlexGrid 没有滚动条          |
| flexScrollHorizontal | 1 | MSHFlexGrid 有一个水平滚动条       |
| flexScrollVertical   | 2 | MSHFlexGrid 有一个竖直滚动条       |
| flexScrollBoth       | 3 | MSHFlexGrid 有水平和竖直滚动条（缺省值） |



## 说明

滚动条只在 `MSHFlexGrid` 的内容伸展到它的边框外边并且 `value` 指定滚动条时，才显示在其上。如果 `ScrollBars` 属性被设置为 `None`，`MSHFlexGrid` 就没有滚动条（不管它的内容）。

**注意：**如果 `MSHFlexGrid` 在两个方向上都没有滚动条，它就不允许在任一方向上的任何滚动，尽管用户使用键盘选择一个在控件的可视区域外面的一个单元。

## 请参阅

`Scroll` 事件（`MSHFlexGrid`），`ScrollTrack` 属性。

## ScrollTrack 属性

返回或设置一个值，决定在用户沿着滚动条移动滚动框时，`MSHFlexGrid` 是否应该滚动它的内容。

## 应用于

`MSHFlexGrid` 控件，`MSFlexGrid` 控件。

## 语法

*object*.ScrollTrack [=*Boolean*]

ScrollTrack 属性的语法包含如下部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象                        |
| <i>Boolean</i> | 一个布尔表达式，指定当用户沿着滚动条移动滚动框时，MSHFlexGrid 是否应该滚动它的内容 |

## 设置

*Boolean* 的设置值如下所示：

| 部分    | 描述                                  |
|-------|-------------------------------------|
| True  | 当用户沿着滚动条移动滚动框时，MSHFlexGrid 滚动它的内容   |
| False | MSHFlexGrid 内容只在滚动框释放时一次性更改。这是缺省设置值 |

## 说明

设置这一属性为 **False** 以避免过度的滚动和闪烁。只要把它设置为 **True** 来仿真其他有这一行为的控件，或查看滚动时的行和列。

## 请参阅

Scroll 事件（MSHFlexGrid），ScrollBars 属性（MSHFlexGrid）。

## SelChange 事件 (MSHFlexGrid)

当选定的范围更改到一个不同的单元或单元范围时发生。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
Private Sub object_SelChange()
```

SelChange 事件的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说明

当用户单击一个新的单元并拖动来选择一个新范围的单元或者是通过使用箭头键同时按下并保持 **SHIFT** 键来选择一定范围单元时，**SelChange** 事件发生。这一事件当用户拖动选择经过 **MSHFlexGrid** 时不发生。

这一事件可以通过使用 **Row**，**Col**，**RowSel** 或 **ColSel** 属性编程更改选择的区域发生。

请参阅

ColSel, RowSel 属性，Col, Row 属性（MSHFlexGrid）。

## SelectionMode 属性

返回或设置一个值，决定一个 MSHFlexGrid 是否应该允许正常的单元选择、按行选择或按列选择。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.SelectionMode [=value]

SelectionMode 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数或常量，指定选择方式。如设置值中所述   |

设置

*value* 的设置值如下所示：

| 常量                    | 值 | 描述   |
|-----------------------|---|--|
| flexSelectionFree     | 0 | 自由。允许选择 MSHFlexGrid 中的个别单元，电子数据表风格。这是缺省设置值 |
| flexSelectionByRow    | 1 | 按行。强制跨越整个行选择，如在一个多列列表框或基于记录的显示             |
| flexSelectionByColumn | 2 | 按列。强制跨越整个列选择，如同为图表选择范围或者是为排序选择字段           |

请参阅

AllowBigSelection 属性。

### Sort 属性 (MSHFlexGrid)

设置一个值，根据选定的条件排序选择的行。这一属性在设计时不可用。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.Sort [=value]

Sort 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数或常量，指定排序类型。如设置值中所述   |

## 设置

value 的设置值如下所示：

| 常量                            | 值 | 描述                          |
|-------------------------------|---|-----------------------------|
| flexSortNone                  | 0 | 无。不执行排序                     |
| flexSortGenericAscending      | 1 | 一般升序。执行估计文本不管是字符串或者是数字的升序排序 |
| flexSortGenericDescending     | 2 | 一般降序。执行估计文本不管是字符串或者是数字的降序排序 |
| flexSortNumericAscending      | 3 | 数值升序。执行将字符串转换为数值的升序排序       |
| flexSortNumericDescending     | 4 | 数值降序。执行将字符串转换为数值的降序排序       |
| flexSortStringNoCaseAscending | 5 | 字符串升序。执行不区分字符串大小写比较的升序排序    |
| flexSortNoCaseDescending      | 6 | 字符串降序。执行不区分字符串大小写比较的降序排序    |

续表

|                          |   |                         |
|--------------------------|---|-------------------------|
| flexSortStringAscending  | 7 | 字符串升序。执行区分字符串大小写比较的升序排序 |
| flexSortStringDescending | 8 | 字符串降序。执行区分字符串大小写比较的降序排序 |
| flexSortCustom           | 9 | 自定义。使用 Compare 事件比较行    |

### 说明

Sort 属性总是排序整个行。要指定排序的范围，设置 Row 和 RowSel 属性。如果 Row 和 RowSel 相同，MSHFlexGrid 将排序所有不固定行。

用于排序的关键字由 Col 和 ColSel 属性决定。排序总是在一个从左到右的方向上完成。例如，如果 Col =3 且 ColSel =1，排序根据列 1 的内容，然后是列 2 、列 3 的内容来进行。

用于行比较的方法由 value 决定，如设置值中的解释。设置值 9 （自定义）最灵活，但比其它设置慢，典型地大约是十分之一，使用这一设置的另一种可能是创建一个不可见列，用关键字填充，然后使用另一设置执行一个基于自定义的排序。这对基于日期的排序是一个好方法。

### 请参阅

ColSel,RowSel 属性，Compare 事件，Col,Row 属性（MSHFlexGrid）。

## 示例

下面的示例使用 `Sort` 和 `TextMatrix` 属性。它根据一个 `ComboBox` 控件的值执行一个 `MSHFlexGrid` 排序。要使用该示例，在窗体中放置一个 `MSHFlexGrid` 控件和一个 `ComboBox` 控件。把下面的代码粘贴到 `Declarations` 节，然后按 `F5` 键。

**注意：** 如果正在使用 `MSFlexGrid`，用 “`MSFlexGrid1`” 代替 “`MSHFlexGrid1`”。

```
Private Sub Combo1_Click()  
' 根据排序方法选择列。  
Select Case Combo1.ListIndex  
Case 0 To 2  
MSHFlexGrid1.Col = 1  
Case 3 To 4  
MSHFlexGrid1.Col = 2  
Case 4 To 8  
MSHFlexGrid1.Col = 1  
End Select  
' 根据 Combo1.ListIndex 排序。  
MSHFlexGrid1.Sort = Combo1.ListIndex  
End Sub  
Private Sub Form_Load()
```



```
Dim i As Integer
' 用随机数据填充 MSHFlexGrid。
MSHFlexGrid1.Cols =3 ' 创建三列。

For i =1 To 11 ' 添加十项。
MSHFlexGrid1.AddItem ""
MSHFlexGrid1.Col =2
MSHFlexGrid1.TextMatrix(i, 1) =SomeName(i)
MSHFlexGrid1.TextMatrix(i, 2) =Rnd()
Next i
' 用排序选择填充 combo 框。
With Combo1
.AddItem "flexSortNone" ' 0
.AddItem "flexSortGenericAscending" '1
.AddItem "flexSortGenericDescending" '2
.AddItem "flexSortNumericAscending" '3
.AddItem "flexSortNumericDescending" '4
.AddItem "flexSortStringNoCaseAsending" '5
.AddItem "flexSortNoCaseDescending" '6
.AddItem "flexSortStringAscending" '7
.AddItem "flexSortStringDescending" '8
.ListIndex =0
End With
```

```
End Sub
Private Function SomeName(i As Integer) As String
Select Case i
Case 1
SomeName ="Ann"
Case 2
SomeName ="Glenn"
Case 3
SomeName ="Sid"
Case 4
SomeName ="Anton"
Case 5
SomeName ="Hoagie"
Case 6
SomeName ="Traut "Trane"
Case 7
SomeName ="MereD Wah"
Case 8
SomeName ="Kemp"
Case 9
SomeName ="Sandy"
Case 10
```

```
SomeName ="Lien"  
Case 11  
SomeName ="Randy"  
End Select  
End Function
```

## Text 属性 (MSHFlexGrid)

返回或设置一个单元或一定范围单元的文本内容。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.Text [=string]

Text 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>string</i> | 一个字符串表达式，一个单元或单元范围包含的文本  |

## 说明

在获取时，Text 属性总是获取由 Row 和 Col 属性定义的当前单元的内容。

在设置时，Text 属性设置当前单元或者是依据 FillStyle 属性设置选择的内容。

## 请参阅

FillStyle 属性 (MSHFlexGrid)，TextArray 属性，TextMatrix 属性，TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader 属性 (MSHFlexGrid)，CellTextStyle 是，Col, Row 属性 (MSHFlexGrid)。

## TextArray 属性

返回或设置一个任意单元的文本内容。

## 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

## 语法

*object*.TextArray(*cellindex*) [=string]

TextArray 属性的语法包含如下部分：

| 部分               | 描述                           |
|------------------|------------------------------|
| <i>object</i>    | 一个对象表达式，其值为“应用于”列表中的一个对象     |
| <i>cellindex</i> | 整数。一个数值表达式，指定要读或写哪一个单元。请参阅说明 |
| <i>string</i>    | 一个字符串表达式，包含一个任意的单元的内容        |

## 说明

这一属性允许不更改 Row 和 Col 属性来设置或获取一个单元的内容。

*cellindex* 参数决定要用哪一个单元。它是由首选行乘 Cols 属性并加上首选列计算的。计算 *cellindex* 的最清晰和最方便的方法是定义一个函数来完成。如示例中显示。

## 请参阅

Text 属性（MSHFlexGrid），TextMatrix 属性。

## 示例

下面的示例显示如何通过定义一个函数计算 *cellindex*。

**注意：** 如果正在使用 MSFlexGrid，用 “MSFlexGrid1” 代替 “MSHFlexGrid1”。

' 为使用 TextArray 属性计算索引。

```

Function faIndex(row As Integer, col As Integer) As Long
faIndex =row * MSHFlexGrid1.Cols + col
End Function
Sub Form_Load()
Dim i as Integer
' 使用 TextArray 属性的数据填充 MSHFlexGrid。
For i =MSHFlexGrid1.FixedRows to MSFlexGrid1.Rows - 1
' ** 列 1
MSHFlexGrid1.TextArray(faIndex(i, 1)) =RandomName()
' 列 2.
MSHFlexGrid1.TextArray(faIndex(i, 2)) =RandomNumber()
Next

```

## TextMatrix 属性

返回或设置一个任意单元的文本内容。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

```
object.TextMatrix(rowindex, colindex) [=string]
```

TextMatrix 属性的语法包含如下部分：

| 部分                       | 描述                       |
|--------------------------|--------------------------|
| <i>object</i>            | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>rowindex colindex</i> | 整数。一个数值表达式，指定要读或写哪一个单元   |
| <i>string</i>            | 一个字符串表达式，包含一个任意的单元的内容    |

说明

这一属性允许不更改 Row 和 Col 属性来设置或获取一个单元的内容。

请参阅

Text 属性 (MSHFlexGrid)，TextArray 属性，Col,Row 属性 (MSHFlexGrid)。

TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader 属性 (MSHFlexGrid)

为在一个特定单元或单元范围中的文本返回或设置三维风格。

TextStyle — 决定常规的 MSHFlexGrid 单元的风格。

TextStyleBand — 决定带区的风格。  
TextStyleFixed — 决定固定行和列的风格。  
TextStyleHeader — 决定标头的风格。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.TextStyle [=*style*]  
*object*.TextStyleBand [=*style*]  
*object*.TextStyleFixed [=*style*]  
*object*.TextStyleHeader [=*style*]

TextStyle、TextStyleBand、TextStyleFixed 和 TextStyleHeader 属性的语法包含如下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>style</i>  | 一个整数或常量，指定文本风格。如设置值中所述   |

设置

*style* 的设置值如下所示：



| 常量                  | 值 | 描述                  |
|---------------------|---|---------------------|
| flexTextFlat        | 0 | 文本正常显示，平面文本。这是缺省设置值 |
| flexTextRaised      | 1 | 文本看起来凸起             |
| flexTextInset       | 2 | 文本看起来凹入             |
| flexTextRaisedLight | 3 | 文本看起来轻微凸起           |
| flexTextInsetLight  | 4 | 文本看起来轻微凹入           |

### 说明

设置值 1 和 2 对于大号 and 粗体字体最好。设置值 3 和 4 对于小号和正常体字体最好。

### 请参阅

Text 属性（MSHFlexGrid），CellStyle 属性。

## TopRow 属性 (MSHFlexGrid)

返回或设置 MSHFlexGrid 中最上面的可视行（不是固定行）。这一属性在设计时不可用。

### 应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

## 语法

*object*.TopRow [=*number*]

TopRow 属性的语法包含如下部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象        |
| <i>number</i> | 一个 Long 值，指定 MSHFlexGrid 中最上面的行 |

## 说明

可以使用这一属性编程来读或设置 MSHFlexGrid 的可视的顶行。使用 LeftCol 属性决定 MSHFlexGrid 中最左边的可视列。

当设置 TopRow 时可以使用的最大行号是总行数减去在 MSHFlexGrid 中可视的行数。如果这一属性被设置为一个更大行号，MSHFlexGrid 将重新把它设置为这个可能的最大值。

## 请参阅

LeftCol 属性（MSHFlexGrid），Scroll 事件（MSHFlexGrid）。

## Version 属性 (MSHFlexGrid)

返回当前加载到内存中的 MSHFlexGrid 的版本。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.Version [=integer]

Version 属性的语法包含如下部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>integer</i> | 一个整数，表示 MSHFlexGrid 的版本号 |

说明

版本号是一个三位数。第一位表示主版本号，后两位表示次版本号。例如，版本 3.5 返回一个整数 350。

## WordWrap 属性 (MSHFlexGrid)

返回或设置一个值，决定一个单元显示多行文本还是一长行文本。

**注意：**返回如 Chr (13) 的字符，也强制分行。

应用于

MSHFlexGrid 控件，MSFlexGrid 控件。

语法

*object*.WordWrap [=*Boolean*]

WordWrap 属性的语法包含如下部分内容：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>Boolean</i> | 一个布尔表达式，指定一个单元中的文本是否换行   |

设置

*Boolean* 的设置值如下所示：

| 部分    | 描述                   |
|-------|----------------------|
| True  | 单元文本显示为多行自动换行的文本     |
| False | 单元文本显示为一长行文本。这是缺省设置值 |

## 说明

当 WordWrap 被设置为 False 时 MSHFlexGrid 显示文本稍微快一些。



[返回总目录](#)

## 目 录

|                                      |     |
|--------------------------------------|-----|
| ImageCombo 控件 .....                  | 3   |
| Microsoft Internet Transfer 控件 ..... | 23  |
| ListView 控件 .....                    | 71  |
| MAPIMessages 控件 .....                | 193 |
| Multimedia MCI.....                  | 292 |
| MonthView 控件.....                    | 366 |
| Masked Edit（屏蔽编辑）控件.....             | 413 |

|                      |     |
|----------------------|-----|
| SSTab 控件 .....       | 439 |
| PictureClip 控件.....  | 469 |
| RichTextBox 控件 ..... | 494 |
| StatusBar 控件 .....   | 603 |
| SysInfo 控件 .....     | 647 |
| TabStrip 控件.....     | 718 |
| TreeView 控件.....     | 814 |
| Winsock 控件.....      | 959 |
| 附录 A.....            | 997 |

# ImageCombo 控件

**ImageCombo** 控件是标准 Windows 组合框的允许绘图版本。控件列表部分中的每一项都可以有一幅图片指定给它。

除了支持图片之外，**ImageCombo** 还提供了一个对象和基于集合的列表控件。控件列表部分的每一项是一个不同的 **ComboItem** 对象，而且列表中的所有项组合起来构成 **ComboItems** 集合。这就使它容易一项一项地指定诸如标记文本、ToolTip 文本、关键字值以及缩进等级等属性。

## 语法

**ImageCombo**

## 说明

使用 **ImageCombo** 控件可以显示一个包含图片的项目列表。每一项可以有自己的图片，也可以对多个列表项使用相同的图片。

**ImageCombo** 控件包括一个 **ComboItem** 对象的集合。一个 **ComboItem** 对象定义了出现在控件列表部分中的项目的各种特性。



除了用列表项目来显示图片外， **ImageCombo** 控件还使用集合和对象管理控件的列表部分。这使它很容易使用相似的对象和集合概念来对列表中的输入项进行操作，例如 **Add**， **Remove** 和 **Clear** 方法，以及 **For Each** 和 **With... End With** 结构。

**注意：** **ImageCombo** 控件是一组 **ActiveX** 控件的一部分，这组 **ActiveX** 控件能够在 **MSCOMCTL.ocx** 文件中找到。要在您的应用程序中使用 **ImageCombo** 控件，必须先将 **MSCOMCTL.ocx** 文件添加到工程中。当发布您的应用程序时，要把 **MSCOMCTL.ocx** 文件安装到用户的 **Microsoft Windows System** 或 **System32** 目录中。有关如何向一个工程中添加 **ActiveX** 控件的更多信息，请参阅《程序员指南》中的“添加控件到工程”。

## 属性

**Identation** 属性， **SelectedItem** 属性（**ActiveX** 控件）， **CaseSensitive** 属性， **UsePathSep** 属性， **Locked** 属性， **ComboItems** 属性， **DataFormat** 属性， **Left**， **Top** 属性， **TabIndex** 属性， **Tag** 属性， **DragIcon** 属性， **DragMode** 属性， **TabStop** 属性， **HelpContextID** 属性， **Index** 属性（控件矩阵）， **Name** 属性， **Parent** 属性， **Container** 属性， **Object** 属性， **ToolTipText** 属性， **WhatsThisHelpID** 属性， **OLEDragMode** 属性（**ActiveX** 控件）， **OLEDropMode** 属性（**ActiveX** 控件）， **SelLength**， **SelStart**， **SelText** 属性（**ActiveX** 控件）， **Text** 属性（**ActiveX** 控件），

Height, Width 属性（ActiveX 控件），Visible 属性（ActiveX 控件），BackColor, ForeColor 属性（ActiveX 控件），Enabled 属性（ActiveX 控件），Font 属性（ActiveX 控件），hWnd 属性（ActiveX 控件），MouseIcon 属性（ActiveX 控件），MousePointer 属性（ActiveX 控件），ImageList 属性（ActiveX 控件）。

## 方法

SetFirstVisible 方法，GetFirstVisible 方法，SetFocus 方法，Drag 方法，Move 方法，ZOrder 方法，ShowWhatsThis 方法，OLEDrag 方法（ActiveX 控件），Refresh 方法（ActiveX 控件）。

## 事件

DropDown 事件（ImageCombo 控件），DragDrop 事件，DragOver 事件，GotFocus 事件，LostFocus 事件，Validate 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedBack 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Change 事件（ActiveX 控件），Click 事件（ActiveX 控件），DbClick 事件（ActiveX 控件），KeyDown, KeyUp 事件（ActiveX 控件），KeyPress 事件（ActiveX 控件）。

请参阅

SellImage 属性, Add 方法(ComboItems 集合), ComboItem 对象, ComboImages 集合, ImageList 控件, Image 属性 (ActiveX 控件), 使用 ImageCombo 控件。

## Add 方法 (ComboItems 集合)

向集合中添加一个 ComboItem 对象，并返回对新创建对象的一个引用。

应用于

ComboItems 集合。

语法

*object.Add(Index As Variant, Key As Variant, Text As Variant, Image As Variant, SellImage OverLayImage As Variant, Indentation As Variant) As ComboItem*

Add 方法的语法有以下部分：

| 部分                  | 描述   |
|---------------------|--|
| <i>object</i>       | 一个对象表达式，其值是“应用于”列表中的一个对象                               |
| <i>index</i>        | 可选的。在集合内创建新对象的位置                                       |
| <i>key</i>          | 可选的。在集合内标识该项的唯一字符串。可用来代替 <i>Index</i> 指定对象             |
| <i>text</i>         | 可选的。该项的文本，同将在组合框的列表和文本部分中出现的一样                         |
| <i>image</i>        | 可选的。一个 <b>ImageList</b> 控件的索引或关键字，标识与列表项同时使用的图片        |
| <i>selImage</i>     | 可选的。一个 <b>ImageList</b> 控件的索引或关键字，标识当列表项被选中时与其同时使用的图片  |
| <i>overLayImage</i> | 可选的。一个 <b>ImageList</b> 控件的索引或关键字，标识列表项所使用的覆盖图片        |
| <i>indentation</i>  | 可选的。将应用于项的缩进等级。应用于每一个缩进等级的空格数由 <b>Indentation</b> 属性决定 |
| <i>comboItem</i>    | 新创建 <b>ComboItem</b> 对象的一个引用，作为此函数成功完成的一个结果返回          |

## 说明

如果没有参数传递到 **Add** 方法，则新对象将作为集合的最后一个对象来创建。没有指定缺省值，而且 *Indentation* 设置为 0。

**Add** 方法返回一个对新创建 **ComboItem** 对象的引用。可以利用它给一个对象变量赋予新的 **ComboItem** 值，然后就可以访问或更改它的任何一个属性。

## 示例

该实例表明了如何使用 **Add** 方法添加一个 **ComboItem**，以及如何利用返回的引用更改新对象的属性。

```
Dim ci As ComboItem  
  
Set ci = ImageCombo1.ComboItems._  
Add(1, "Signal1", "Signal", "RedLight", "GreenLight", 0, 0)  
  
ci.ToolTipText = "Traffic Light"  
ci.Indentation = 2
```

## CaseSensitive 属性

返回或设置在 **ImageCombo** 列表中进行搜索时字符串是否区分大小写。

应用于

ImageCombo 控件。

语法：

*object*.CaseSensitive [=*boolean*]

CaseSensitive 属性的语法有如下几个部分：

| 部分             | 描述                                   |
|----------------|--------------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象             |
| <i>boolean</i> | 一个布尔表达式，指明搜索字符串的大小写是否影响搜索结果，如“设置”中所示 |

设置

*boolean* 的设置如下：

| 设置    | 描述                            |
|-------|-------------------------------|
| True  | 列表框中的项按大小写排序；搜索时也是如此          |
| False | （缺省值）列表框中的项不按大小写排序，搜索时也不区分大小写 |

## 说明

**CaseSensitive** 属性决定了在搜索 **ImageCombo** 的列表框时如何处理其中的项。列表中所搜索内容就是用户在控件的文本框中输入的文本。例如，假设列表中有两项，一个“AAA”项和一个“Aaa”项。用户在组合框的编辑框中输入“Aa”。如果 **CaseSensitive** 设置为 **True**，则就选中“Aaa”。如果 **CaseSensitive** 设置为 **False**，就选中“AAA”。

## ComboItem 对象

**ComboItem** 对象是 **ImageCombo** 控件列表部分中的一项。**ComboItem** 对象可以显示文本和 / 或图片，而且这些文本和 / 或图片可以按列表中的其它项缩进样式显示。

## 语法

*object*.ComboItem

ComboItem 对象的语法有以下部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 一个对象表达式，其值是一个 <b>ImageCombo</b> 控件 |

## 说明

**ComboItem** 对象及其相应的 **ComboItems** 集合包括出现在 **ImageCombo** 控件列表部分中的所有项。因为列表项是存储在一个集合里的，所以每一项可以有多个属性与之相关联。这样就更容易分配和管理与列表项相关联的图像。

因为列表中的项都是一个集合中的对象，所以标准组合框（例如 **List**，**ListIndex**， 和 **ItemData** ）的某些属性将不再需要。

使用 **ComboItem** 对象可以：

- 指定一项的文本。
- 通过指定一个 **ImageList** 控件的索引来指定一幅图片显示在该项旁边。
- 当列表中的一项被选中时为其指定一幅不同的图片。
- 指定一项的一个缩进。
- 为某项指定一个唯一的 **Key** 值，可以用该值代替集合中的索引来引用该项。

**ImageCombo** 控件最初并不包括任何 **ComboItem** 对象。必须使用 **Add** 方法添加。



## 属性

Key 属性（ActiveX 控件），Index 属性（ActiveX 控件），SelItems 属性，Indentation 属性，Selected 属性（ActiveX 控件），Image 属性（ActiveX 控件）。

## 请参阅

ImageCombo 控件，ComboItems 集合，ComboItems 属性。

## ComboItems 集合

ComboItems 集合包括 ImageCombo 控件中所有的 ComboItem 对象。

## 语法

*object*.ComboItems(*index*)

ComboItems 集合的语法有以下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值为 <b>ImageCombo</b> 控件                                       |
| <i>index</i>  | 一个整数或字符串，用来唯一标识一个对象集合中的一个成员。整数是 <b>Index</b> 属性的值；字符串是 <b>Key</b> 属性的值 |

## 说明

**ComboItems** 集合是一个基于 1 的 **ComboItem** 对象的集合。

**ComboItem** 对象在 **ComboItems** 集合中出现的顺序与其在控件列表部分中的可见位置相同。一个 **ComboItem** 对象在集合中的位置由其 **Index** 属性指示。

## 属性

**Count** 属性（ActiveX 控件），**Item** 属性（ActiveX 控件）。

## 方法

**Clear** 方法（ActiveX 控件），**Remove** 方法（ActiveX 控件），**Add** 方法（**ComboItems** 集合）。

请参阅

ComboItem 对象，ComboItems 属性。

## ComboItems 属性

返回一个对 ComboItem 对象 ComboItems 集合的引用。

应用于

ImageCombo 控件。

语法

*object*.ComboItems(*index*)

ComboItems 属性的语法有以下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 集合中一个成员的索引或关键字           |

请参阅

ComboItem 对象，ComboItems 集合。

## DropDown 事件（ImageCombo 控件）

此事件在 ImageCombo 控件的列表部分即将被下拉时发生。

应用于

ImageCombo 控件。

语法

Private Sub *object*\_DropDown( )

DropDown 事件的语法有以下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |

## 说明

在用户作出选择之前，使用一个 **DropDown** 事件过程对 **ImageCombo** 的控件列表进行最后更新。这样做能够使您在列表中添加或删除一些项，改变列表项显示的图像，或者对列表、列表项或其属性做一些其它的修改。

## 示例

下面的代码检查一个菜单项的值来看这些项的状态是否应被显示。如果状态显示被激活，则代码检查每一列表项的 **Tag** 属性，并设置相应的 **OverlayImage** 属性。如果状态显示未被激活，则此项的 **OverlayImage** 属性设置为 0,从而删除它。

```
ImageCombo1_DropDown()  
    If mnuShowStatus.Checked = True Then  
        For Each CboItem in ImageCombo1.ComboItems  
            Select Case CboItem.Tag  
                Case "Locked"  
                    CboItem.OverlayImage = "Padlock"  
                Case "Deleted"  
                    CboItem.OverlayImage = "X-mark"  
                Case "Checked"
```

```
        CboItem. OverlayImage = "Checkmark"  
    Case Else  
        CboItem. OverlayImage = 0  
    End Select  
Next CboItem  
Else  
    For Each CboItem in ImageCombo1.ComboItems  
        CboItem. OverLayImage = 0  
    Next CboItem  
End If  
End Sub
```

在上面的代码中，“Padlock”、“X-mark”和“Checkmark”是关键字值，这些值指示在 **ImageList** 控件中与 **ImageCombo** 相关联的特定图像。

## Locked 属性

返回或设置一个值，指出一个控件能否被编辑。

应用于

ImageCombo 控件。

语法

*object.Locked* [= *boolean*]

Locked 属性的语法有以下部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指出该控件能否被编辑，如“设置”所描述的 |

设置

*boolean* 的设置值有：

| 常量    | 描述                |
|-------|-------------------|
| False | （缺省值）用户可以编辑控件或其内容 |
| True  | 用户不能编辑控件或其内容      |

# SelImage 属性

对 ImageList 控件设置索引或关键字，在这里能够找到该项选定的图像。

应用于

ComboItem 对象。

语法

*object.SelImage* [= *variant*]

SelImage 属性的语法有以下部分：

| 部分             | 描述                                    |
|----------------|---------------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象              |
| <i>variant</i> | 一个变量表达式，其值为 ImageList 控件中一幅图像的索引或关键字值 |

说明

SelImage 属性指定从列表选定一项时应显示的图片，它也决定组合框的文本框部分中下一项出现哪一幅图片。如果未给 SelImage 属性指定一个值，



该项的图片将不会在该项被选中时改变。

## SetFirstVisible 方法

设置控件内部区域中第一个可视项。

应用于

ImageCombo 控件。

语法

*object*.SetFirstVisible(*pIComboItem* as ComboItem)

SetFirstVisible 方法的语法有如下几个部分：

| 部分                 | 描述                      |
|--------------------|-------------------------|
| <i>object</i>      | 对象表达式，其值是“应用于”列表中的对象    |
| <i>pIComboItem</i> | 应设置为可见的 ComboItem 对象的引用 |

# UsePathSep 属性

只针对 Windows NT。返回或设置一个值，指明文本框是否使用斜线(“/”)、反斜线(“\”)和句号(“.”)做为单词分隔符。

应用于

ImageCombo 控件。

语法：

*object*.UsePathSep [= *boolean* ]

UsePathSep 属性的语法有如下几个部分：

| 部分             | 描述                                 |
|----------------|------------------------------------|
| <i>object</i>  | 一个对象表达式，与“应用于”中的对象等价               |
| <i>boolean</i> | 一个布尔表达式，指明路径分隔符是否解释为单词分隔符，如“设置”中所示 |

## 设置

boolean 的设置如下：

| 设置    | 描述                   |
|-------|----------------------|
| True  | 路径分隔符将充当单词分隔符        |
| False | （缺省值）路径分隔符将被当作单词的一部分 |

## 说明

单词分隔符决定了用户如何在文本中移动。通常，空格被认为是单词分隔符，用户在按下 **CTRL** 键并同时按下左方向键或右方向键时，就可以在文本字符串中一次移动一个单词。**CTRL+ARROW** 使得文本光标移到下一个分隔符（空格）。

如果 **UsePathSep** 设置为 **True**，则路径分隔符（“/”，“\”和“.”）也被当作单词分隔符。这就更加方便用户在长网络路径或 **Internet URL** 地址中使用 **CTRL+ARROW** 键进行移动。

# Microsoft Internet Transfer 控件

Internet Transfer 控件支持超文本传输协议 (HTTP) 和文件传输协议 (FTP)，它们是 Internet 网上使用最广泛的两种协议。

使用 HTTP 协议，可以连接全球信息网 (World Wide Web) 服务器，以检索 HTML 文档。使用 FTP 协议可以在 FTP 服务器上登录，以下载和加载文件。UserName 和 Password 属性可让您在要求验证身份的私有的服务器上登录。另外，也可以连接公用的 FTP 服务器，并下载文件。Execute 方法支持普通的 FTP 命令，如 CD 和 GET。

## 属性

RemoteHost 属性 (ActiveX 控件)，RemotePort 属性 (ActiveX 控件)，Index 属性 (ActiveX 控件)，Tag 属性 (ActiveX 控件)，Object 属性 (ActiveX 控件)，AccessType 属性，Document 属性，hInternet 属性，Password 属性 (Internet Transfer 控件)，Protocol 属性 (Internet Transfer 控件)，Proxy 属性，RequestTimeout 属性，ResponseCode 属性，ResponseInfo 属性，StillExecuting 属性，URL 属性，UserName 属性，Name 属性，Parent 属性。

## 方法

Cancel 方法, Execute 方法, GetChunk 方法(Internet Transfer 控件), GetHeader 方法, Open URL 方法。

## 事件

StateChanged 事件。

## 请参阅

Internet Transfer 控件的可捕获的错误, 使用 Internet Transfer 控件。

## AccessType 属性

设置或返回一个值, 决定该控件用来与 Internet 网进行通讯的访问类型(通过代理访问或直接访问)。正在处理异步请求时, 该值可以改变, 但直到创建了下一个连接时, 改变才会生效。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.AccessType = *type*

AccessType 属性的语法包含下面部分：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象               |
| <i>type</i>   | 整数（枚举型）。数值表达式，决定所使用的访问类型，参见“设置值”中的描述 |

设置

*type* 设置值如下：

| 常量           | 值 | 描述                                       |
|--------------|---|--|
| icUseDefault | 0 | 缺省。使用缺省值。控件使用在注册表中找到的缺省设置值来访问 Internet 网 |
| icDirect     | 1 | 直接连到 Internet 网。控件直接连到 Internet 网        |
| icNamedProxy | 2 | 命名代理。指示控件使用 Proxy 属性中指定的代理服务器            |

请参阅

Proxy 属性。

## Cancel 方法

取消当前请求，并关闭当前创建的所有连接。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.Cancel

*object* 置换元表示对象表达式，其值是“应用于”列表中的对象。

返回值

None

## Document 属性

返回或设置与 **Execute** 方法一起使用的文件或文档。如果未指定该属性，将返回服务器中的缺省文档；如果不指定文档写操作会发生错误。

应用于

Microsoft Internet Transfer 控件。



## 语法

*object.Document = string*

Document 属性的语法包含下面部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象      |
| <i>string</i> | 与 Execute 方法一起使用的文件或文档的名称 |

请参阅

Execute 方法。

## Execute 方法

执行对远程服务器的请求。只能发送对特定的协议有效的请求。

应用于

Microsoft Internet Transfer 控件。

# 语法

*object.Execute url, operation, data, requestHeaders*

Execute 属性的语法包含下面部分：

| 部分                    | 描述   |
|-----------------------|--|
| <i>object</i>         | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>url</i>            | 可选的。字符串，指定控件将要连接的 URL。如果这里未指定 URL，将使用 URL 属性中指定的 URL             |
| <i>operation</i>      | 可选的。字符串，指定将要执行的操作类型。所支持的操作的列表，参见下面的“设置”                          |
| <i>data</i>           | 可选的。字符串，指定用于操作的数据，参见下面的“设置”                                      |
| <i>requestHeaders</i> | 可选的。字符串，指定由远程服务器传来的附加的标头。它们的格式为：header name: header value vbCrLf |

# 设置

注意：operation 的有效设置值由所用的协议决定。下面的这张表格是按协议来组织的。

# 支持的 HTTP 命令

*operation* 的有效设置值:

| 运算   | 描述  |
|------|---|
| GET  | 检索由 URL 属性指定的 URL 中的数据  |
| HEAD | 发送请求的标头   |
| POST | 传递数据给服务器。该数据在 <i>data</i> 参数中。这是 GET 的替代方法，附加的指令在 <i>data</i> 参数中指定 |
| PUT  | Put 操作。被替代的页面名在 <i>data</i> 参数中                                     |

# 支持的 FTP 命令

**重点:** FTP 协议使用单个字符串，该字符串包含操作名以及操作所需的其它参数。换句话说，即不使用 *data* 和 *requestHeaders* 参数；所有的操作及操作的参数是在 *operation* 参数中作为单个字符串来传递的。各参数间由空格分隔。在下面的描述中，不要把 “file1”、 “file2” 与 *data* , *requestHeaders* 参数搞混。

FTP 操作的语法为:

*operationName file1 file2.*

例如，为获得一个文件，下面的代码调用 `Execute` 方法，该方法包含着操

作名（“GET”）以及此操作所需的两个文件名：

```
Inet1.Execute "FTP://ftp.microsoft.com", _  
"GET Disclaimer.txt C:\Temp\Disclaimer.txt"
```

注意：不支持嵌入空格的文件名。

*operation* 有效的 FTP 设置值：

| 运算                     | 描述  |
|------------------------|---|
| CD <i>file1</i>        | 改变目录。改变到 <i>file1</i> 中指定的目录  |
| CDUP                   | 改变到父目录。等效于 “CD”   |
| CLOSE                  | 关闭当前的 FTP 连接  |
| DELETE <i>file1</i>    | 删除 <i>file1</i> 中指定的文件  |
| DIR <i>file1</i>       | 目录。搜索 <i>file1</i> 中指定的目录（允许用通配符，但要使用远程主机的语法）。如果没有指定 <i>file1</i> ，将返回当前的整个工作目录 |
|                        | 使用 <b>GetChunk</b> 方法返回目录数据   |
| GET <i>file1 file2</i> | 检索 <i>file1</i> 中指定的远程文件，并创建 <i>file2</i> 中指定的新本地文件                             |
| LS <i>file1</i>        | 列表。搜索 <i>file1</i> 中指定的目录（允许用通配符，但要使用远程主机的语法）。使用 <b>GetChunk</b> 方法返回文件目录数据     |

续表

| 运算                        | 描述  |
|---------------------------|---|
| MKDIR <i>file1</i>        | 创建目录。创建 <i>file1</i> 中指定的目录。创建是否成功取决于用户在远程主机上的权限                    |
| PUT <i>file1 file2</i>    | 复制 <i>file1</i> 指定的本地文件到 <i>file2</i> 指定的远程主机上                      |
| PWD                       | 打印工作目录。返回当前目录名。使用 <b>GetChunk</b> 方法返回数据                            |
| QUIT                      | 终止当前用户  |
| RECV <i>file1 file2</i>   | 检索 <i>file1</i> 中指定的远程文件，并创建 <i>file2</i> 中指定的本地新文件。等效于 <b>GET</b>  |
| RENAME <i>file1 file2</i> | 将 <i>file1</i> 中命名的远程文件重命名为 <i>file2</i> 中指定的新名称。成功与否取决于用户在远程主机上的权限 |
| RMDIR <i>file1</i>        | 删除目录。删除 <i>file1</i> 中指定的远程目录。成功与否取决于用户在远程主机上的权限                    |
| SEND <i>file1 file2</i>   | 复制 <i>file1</i> 指定的本地文件到 <i>file2</i> 指定的远程主机上。等效于 <b>PUT</b>       |
| SIZE <i>file1</i>         | 返回 <i>file1</i> 指定的目录的大小  |

返回类型

None

## 说明

上面列出的许多命令都只有当用户在主机服务器上具有相应的权限时，才能够执行。例如，匿名的 **FTP** 站点不允许任何人删除文件或目录。

## 请参阅

**Document** 属性，**Protocol** 属性（**Internet Transfer** 控件），**URL** 属性。

## 示例

该示例列举了一系列使用 **Execute** 方法的 **FTP** 操作。该示例假定窗体中有三个 **TextBox** 控件。第一个控件 **txtURL** 包含 **FTP** 服务器的 **URL**。第二个控件 **txtRemotePath** 包含特殊命令所需的附加信息。第三个控件 **txtResponse** 包含服务器的响应。

```
Private Sub cmdChangeDirectory_Click()  
    '将目录改变到 txtRemotePath。  
    Inet1.Execute txtURL.Text, "CD " & _  
        txtRemotePath.Text  
End Sub  
  
Private Sub cmdDELETE_Click()
```

'删除 txtRemotePath 中的目录。

```
Inet1.Execute txtURL.Text, "DELETE " & _  
txtRemotePath.Text
```

End Sub

```
Private Sub cmdDIR_Click()
```

```
    Inet1.Execute txtURL.Text, "DIR FindThis.txt"
```

End Sub

```
Private Sub cmdGET_Click()
```

```
    Inet1.Execute txtURL.Text, _
```

```
    "GET GetThis.txt C:\MyDocuments\GotThis.txt"
```

End Sub

```
Private Sub cmdSEND_Click()
```

```
    Inet1.Execute txtURL.Text, _
```

```
    "SEND C:\MyDocuments\Send.txt SentDocs\Sent.txt"
```

End Sub

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
```

'State = 12 时，用 GetChunk 方法检索服务器的响应。

```
    Dim vtData As Variant ' Data variable.
```

```
    Select Case State
```

'...没有列举其它情况。

Case icError '11

'出现错误时，返回 ResponseCode 和 ResponseInfo。

vtData = Inet1.ResponseCode & ":" & \_

Inet1.ResponseInfo

Case icResponseCompleted ' 12

Dim vtData As Variant

Dim strData As String

Dim bDone As Boolean: bDone = False

'取得第一个块。

vtData = Inet1.GetChunk(1024, icString)

DoEvents

Do While Not bDone

strData = strData & vtData

'取得下一个块。

vtData = Inet1.GetChunk(1024, icString)

DoEvents

If Len(vtData) = 0 Then

bDone = True

End If



```
        Loop
        txtData.Text = strData
    End Select
End Sub
```

## GetChunk 方法（Internet Transfer 控件）

从 StateChanged 事件中检索数据。把 Execute 方法当作 GET 操作来调用之后使用该方法。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.GetChunk( *size* [,*datatype*] )

Get 属性的语法包含下面部分：

| 部分              | 描述                            |
|-----------------|-------------------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象          |
| <i>size</i>     | 必需的。长整型数值表达式，决定被检索的块的大小       |
| <i>datatype</i> | 可选的。整数，决定被检索块的数据类型，如下面的“设置”所示 |

## 设置

*datatype* 的设置值：

| 常量          | 值 | 描述              |
|-------------|---|-----------------|
| icString    | 0 | 缺省值。把数据作为字符串来检索 |
| IcByteArray | 1 | 把数据作为字节数组来检索    |

## 返回类型

## Variant

## 说明

在 `StateChanged` 事件中使用 `GetChunk` 方法。当 `State` 属性为 `icResponseCompleted (12)` 时，使用 `GetChunk` 方法检索缓冲区的内容。

请参阅

Execute 方法，StateChanged 事件。

示例

该示例在 StateChanged 事件中用 GetChunk 方法来检索一块数据。该示例使用 Select Case 语句来决定如何处理每种可能的状态。该示例假定窗体中有一个名为 txtData 的 TextBox 控件。

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
```

```
    'State = 12 时，使用 GetChunk 方法检索服务器的响应。
```

```
    '该示例假定数据为文本类型。
```

```
    Select Case State
```

```
    '...没有列举其它情况。
```

```
    Case icResponseReceived '12
```

```
        Dim vtData As Variant '数据变量。
```

```
        Dim strData As String: strData = ""
```

```
        Dim bDone As Boolean: bDone = False
```

```
        '取得第一块。
```

```
        vtData = Inet1.GetChunk(1024, icString)
```

```
DoEvents
Do While Not bDone
    strData = strData & vtData
    DoEvents
    '取得下一块。
    vtData = Inet1.GetChunk(1024, icString)
    If Len(vtData) = 0 Then
        bDone = True
    End If
Loop

txtData.Text = strData
End Select

End Sub
```

## GetHeader 方法

GetHeader 方法用于检索 HTTP 文件的标头文本。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.GetHeader (*hdrName*)

GetHeader 方法的语法包含下面部分：

| 部分             | 描述                   |
|----------------|----------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象 |
| <i>hdrName</i> | 可选的。字符串，指定将被检索的标头    |

返回类型

String

说明

如果没有已命名的标头，将返回所有的标头。

下面这张表格列举了一些典型的可用标头。

| 标头             | 描述  |
|----------------|---|
| Date           | 返回文档传输的日期和时间。返回的数据格式为：<br>Wednesday, 27-April-96 19:34:15 GMT |
| MIME-version   | 返回 MIME 协议的版本号，目前为 1.00                                       |
| Server         | 返回服务器的名称  |
| Content-length | 返回数据的字节长度   |
| Content-type   | 返回数据的 MIME 的当前类型  |
| Last-modified  | 返回最后一次修改文档的日期和时间。返回的数据格式为：Wednesday, 27-April-96 19:34:15 GMT |

## hInternet 属性

从下一级的 Wininet.dll API 返回 Internet 句柄。可直接使用该句柄来调用该 API。从 Visual Basic 访问控件时，不能使用该属性。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.hInternet

*object* 置换元表示对象表达式，其值是“应用于”列表中的对象。

数据类型

Long

## OpenURL 方法

打开并返回指定 URL 的文档。文档以变体型返回。该方法完成时，URL 的各种属性（以及该 URL 的一些部分，如协议）将被更新，以符合当前的 URL。

应用于

Microsoft Internet Transfer 控件。

# 语法

*object*.OpenUrl *url* [*,datatype*]

OpenURL 属性的语法包含下面部分：

| 部分              | 描述                    |
|-----------------|-----------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象  |
| <i>url</i>      | 必需的。被检索文档的 URL        |
| <i>datatype</i> | 可选的。整数，如“设置”所示，指定数据类型 |

# 设置

*datatype* 的设置值：

| 常量          | 值 | 描述              |
|-------------|---|-----------------|
| IcString    | 0 | 缺省值。把数据作为字符串来检索 |
| IcByteArray | 1 | 把数据作为字节数组来检索    |

# 返回类型

Variant



## 说明

**OpenURL** 方法的返回值取决于 **URL** 的目标。例如，如果 **URL** 的目标是某个 **FTP** 服务器的目录，将返回该目录。另一方面，如果目标是一个文件，则检索该文件。

**OpenURL** 方法等效于：调用带 **GET** 操作的 **Execute** 方法，然后在 **StateChanged** 事件中调用 **GetChunk** 方法。但是，**OpenURL** 方法会导致从站点返回同步数据流。

如下所示，如果正在检索一个二进制文件，在把它写到磁盘上之前，请务必使用一个字节数组作为临时变量：

```
Dim b() As Byte
Dim strURL As String
'设置 strURL 为一个有效的地址。
strURL = "FTP://ftp.GreatSite.com/China.exe"
b() = Inet1.OpenURL(strURL, icByteArray)

Open "C:\Temp\China.exe" For Binary Access _
Write As #1
Put #1, , b()
Close #1
```

**注意：**当使用 `OpenURL` 方法时，在设置 `Password` 和 `UserName` 属性之前，设置 `URL` 属性。如果最后设置 `URL` 属性，`UserName` 和 `Password` 属性将被置为 “”。

请参阅

`Protocol` 属性（`Internet Transfer` 控件）。

示例

该示例使用 `OpenURL` 方法来检索 `FTP` 服务器的目录。要运行该示例，在窗体中放置一个 `Internet Transfer` 控件和一个 `RichTextBox` 控件。然后，把这段代码粘贴到声明部分。按 `F5` 键运行此例，并双击此窗体。

```
Private Sub Form_DblClick()  
    Inet1.AccessType = icUseDefault  
    RichTextBox1.Text = Inet1.OpenURL _  
        (InputBox("URL", , "ftp://ftp.microsoft.com"))  
End Sub
```

该示例假定数据是二进制文件。用字节数组，并使用 `Open`、`Put` 和 `Close` 方法，就可以检索该文件并把它写到磁盘上。要运行此例，在窗体中放置一个 `Internet Transfer` 控件，并把这段代码粘贴到声明部分。再按 `F5` 键，并双击此

窗体。

```
Private Sub Form_DblClick()  
    Inet1.AccessType = icUseDefault  
    Dim b() As Byte  
    Dim strURL As String  
  
    '假定这仍然是一个有效的 URL。  
    strURL = "ftp://ftp.microsoft.com/" & _  
    "developr/drg/Win32/Autorun.zip"  
  
    '把该文件作为字节数组来检索。  
    b() = Inet1.OpenURL(strURL, icByteArray)  
  
    Open "C:\Temp\Autorun.zip" For Binary Access _  
    Write As #1  
    Put #1, , b()  
    Close #1  
    MsgBox "Done"  
End Sub
```

# Password 属性（Internet Transfer 控件）

设置或返回一个密码，该密码将和请求一道被发送，用以在远程计算机上登录。如果该属性为空，控件将发送一个缺省的密码。

## 应用于

Microsoft Internet Transfer 控件。

## 语法

*object.Password = string*

Password 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>string</i> | 当登录到远程计算机时，要发送的密码    |

## 说明

如下面的表格中所示，控件发送的缺省密码将取决于确切的方案：

| UserName 属性 | Password 属性 | 发送到 FTP 服务器的 UserName | 发送到 FTP 服务器的 Password |
|-------------|-------------|-----------------------|-----------------------|
| Null 或 “ ”  | Null 或 “ ”  | “anonymous”           | 用户的电子邮件名              |
| 非空字符串       | Null 或 “ ”  | UserName 属性           | “ ”                   |
| Null        | 非空字符串       | 错误                    | 错误                    |
| 非空字符串       | 非空字符串       | UserName 属性           | Password 属性           |

请参阅

UserName 属性。

## Protocol 属性（Internet Transfer 控件）

设置或返回一个值，指定和 Execute 方法一起使用的协议。

应用于

Microsoft Internet Transfer 控件。

# 语法

*object.Protocol = integer*

Protocol 属性的语法包含下面部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象       |
| <i>integer</i> | 整数。数值表达式，决定所用的协议，如“设置”中所描述 |

# 设置

Protocol 的有效设置值：

| 常量         | 值 | 描述           |
|------------|---|--------------|
| icUnknown  | 0 | 未知的          |
| icDefault  | 1 | 缺省协议         |
| icFTP      | 2 | FTP。文件传输协议   |
| icReserved | 3 | 为将来预留        |
| icHTTP     | 4 | HTTP。超文本传输协议 |
| icHTTPS    | 5 | 安全 HTTP      |

## 说明

指定该属性后，URL 属性被更新以显示新值。另外，如果此 URL 的协议部分被更新，Protocol 属性也将被更新以体现新值。OpenURL 和 Execute 方法都可能会修改该属性值。

直到调用下一个 Execute 或 OpenURL 方法时，该属性值的改变才会有效。

## 请参阅

Execute 方法，OpenURL 方法。

## Proxy 属性

设置或返回用以和 Internet 网进行通讯的代理服务器的名称。只有当 AccessType 属性设置为 icNamedProxy (3) 时，才使用该属性。

应用于

Microsoft Internet Transfer 控件。

语法

*object.Proxy = proxy*

Proxy 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>proxy</i>  | 所用的代理服务器的名称          |

数据类型

String

说明

直到调用下一个 Execute 或 OpenURL 方法时，该属性值的改变才会有效。



您可以为每一个协议设置单独代理。例如，如果您的网络有一个用于 **ftp** 协议的名为“**CorpFTP**”的代理，且使用了端口 123，则您可以将其设置为：

```
Inet1.Proxy = "ftp=CorpFTP:123"
```

您可以指定多个网关，用空格将其相互分开。例如，如果您的 **HTTP** 代理名称为 “**CorpHTTP**”，并且使用了端口 131，则为两个协议做如下设置：

```
Inet1.Proxy = "ftp=CorpFTP:123 HTTP=CorpHTTP:131"
```

如果您没有指定要设置哪一个协议，则为所有的协议使用相同的代理名称。例如，即使协议是 **http**，也使用以下的代理名称：

```
Inet1.Proxy = "CorpFTP:123"
```

请参阅

**AccessType** 属性。

## RequestTimeout 属性

设置或返回在超时截止之前，按秒计算的等待时间长度。如果请求在指定的时间内还没有响应，并且该请求使用 **OpenURL** 方法（同步地），就会产生

错误；如果请求使用 `Execute` 方法，将引发带错误码的 `StateChanged` 事件。  
把该属性设置为 `0`，则意味着不限定等待时间。

应用于

Microsoft Internet Transfer 控件。

语法

*object.RequestTimeout = time*

`RequestTimeout` 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>time</i>   | 在错误出现之前，按秒计算的等待时间长度  |

数据类型

Long

# ResponseCode 属性

StateChanged 事件中出现 icError (11) 状态时，从连接返回错误码。要获得此错误的描述，检查 ResponseInfo 属性。

应用于

Microsoft Internet Transfer 控件。

语法

```
object.ResponseCode= code
```

ResponseCode 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>code</i>   | 远程服务器返回的数字           |

数据类型

Long

## 说明

如下所示，可使用 **StateChanged** 事件来接收关于错误的通知：

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim strMess As String '消息变量。
    Select Case State
        '...Other cases not shown.
        Case icError '11
            '得到错误文本。
            strMess = "ErrorCode: " & Inet1.ResponseCode & _
                " : " & Inet1.ResponseInfo
    End Select

    Debug.Print strMess
End Sub
```

请参阅

**ReponseInfo** 属性，**StateChanged** 事件。

# ResponseInfo 属性

返回最后发生的错误的文本。想得到该错误码，检查 `ResponseCode` 属性。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.ResponseInfo

ResponseInfo 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>info</i>   | 连接返回的响应              |

返回类型

String

## 说明

如下所示，使用 **StateChanged** 事件来接收错误的通知：

```
Private Sub Inet1_StateChanged(ByVal State As Integer)
    Dim strMess As String '消息变量。
    Select Case State
        '...没有列举其它情况。
        Case icError ' 11
            '得到错误文本。
            strMess = "ErrorCode: " & Inet1.ResponseCode & _
                " : " & Inet1.ResponseInfo
    End Select

    Debug.Print strMess
End Sub
```

## 请参阅

**ResponseCode** 属性，**StateChanged** 事件。

# StateChanged 事件

连接中状态发生改变，就会引发该事件。

应用于

Microsoft Internet Transfer 控件。

语法

*object\_StateChanged*(ByVal *State* As Integer)

StateChanged 事件的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>state</i>  | 整数。如下面的“设置”所示，指定状态   |

设置

State 的设置值：

| 常量                   | 值  | 描述                     |
|----------------------|----|------------------------|
| icNone               | 0  | 无状态可报告                 |
| icHostResolvingHost  | 1  | 该控件正在查询所指定的主机的 IP 地址   |
| icHostResolved       | 2  | 该控件已成功地找到所指定的主机的 IP 地址 |
| icConnecting         | 3  | 该控件正在与主机连接             |
| icConnected          | 4  | 该控件已与主机连接成功            |
| icRequesting         | 5  | 该控件正在向主机发送请求           |
| icRequestSent        | 6  | 该控件发送请求已成功             |
| icReceivingResponse  | 7  | 该控件正在接收主机的响应           |
| icResponseReceived   | 8  | 该控件已成功地接收到主机的响应        |
| icDisconnecting      | 9  | 该控件正在解除与主机的连接          |
| icDisconnected       | 10 | 该控件已成功地与主机解除了连接        |
| icError              | 11 | 与主机通讯时出现了错误            |
| icResponseCompleted1 | 12 | 该请求已经完成，并且所有数据均已接收到    |

## 说明

一般来说，使用 `StateChanged` 事件决定何时使用 `GetChunk` 方法来检索数据。要这样做，须使用 `Select Case` 语句，并测试 `icResponseReceived` (8) 或 `icResponseCompleted` (12)。

注意，当该控件已完成一个操作时，且此操作在缓冲区中没有产生任何数据，此时 `icResponseReceived` 状态也可能出现。例如，当与某个 FTP 站点进



行连接时，该控件将与此 FTP 站点“握手”，但没有在缓冲区中产生任何数据，此时会出现 `icResponseReceived` 状态。

另一方面，一个操作完全完成后，会出现 `icResponseCompleted` 状态。例如，如果正在使用 `Execute` 方法和 `GET` 操作来检索某个文件，在此文件被完全检索之后，将出现 `icResponseCompleted` 事件，且仅出现一次。

实际上，使用 `icResponseReceived` 状态可以对数据做语法分析，直到检索到所需信息为止（例如，检索 `HTML` 文件时，只对标头进行检索）。获得该信息后，就可以取消这次检索。另一方面，如果想检索整个文件，`icResponseCompleted` 状态还会通知传输已经完成，可以继续。

请参阅

`Execute` 方法，`GetChunk` 方法（`Internet Transfer` 控件），`GetHeader` 方法，`OpenURL` 方法，`ReponseCode` 属性，`ResponseInfo` 属性。

## `StillExecuting` 属性

返回一个值，指明此 `Internet Transfer` 控件是否处于忙状态。如果该控件正在做诸如从 `Internet` 网上检索文件之类的操作，将返回 `True`。当该控件处于

忙的时候，不响应其它的请求。

应用于

Microsoft Internet Transfer 控件。

语 法

*object*.StillExecuting = *boolean*

StillExecuting 属性的语法包含下面部分：

| 部分             | 描述                   |
|----------------|----------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象 |
| <i>boolean</i> | 布尔表达式，指明该控件是否处于忙状态   |

数据类型

Boolean

设置

*boolean* 的设置值：

| 常量    | 值  | 描述    |
|-------|----|-------|
| True  | -1 | 该控件忙  |
| False | 0  | 该控件空闲 |

## Internet Transfer 控件的可捕获的错误

下面这张表格列出了 Internet Transfer 控件的可捕获的错误及常量：

| 常量                     | 值     | 描述               |
|------------------------|-------|------------------|
| icOutOfMemory          | 7     | “内存溢出”           |
| icTypeMismatch         | 13    | “类型不匹配”          |
| icInvalidPropertyValue | 380   | 无效属性值            |
| icInetOpenFailed       | 35750 | 不能打开 Internet 句柄 |
| icOpenFailed           | 35751 | “不能打开 URL”       |
| icBadUrlL              | 35752 | “URL 格式不正确”      |
| icProtMismatch         | 35753 | “该方法不支持的协议”      |
| icConnectFailed        | 35754 | “不能连接到远程主机”      |
| icNoRemoteHost         | 35755 | “没有指定远程计算机”      |
| icRequestFailed        | 35756 | “不能完成请求”         |

续表

| 常量                   | 值     | 描述                  |
|----------------------|-------|---------------------|
| icNoExecute          | 35757 | “检索数据前必须执行一个操作”     |
| icBlewChunk          | 35758 | “不能检索数据”            |
| icFtpCommandFailed   | 35759 | “FTP 命令失败”          |
| icUnsupportedType    | 35760 | “不能强制类型”            |
| icTimeOut            | 35761 | “请求超时”              |
| icUnsupportedCommand | 35762 | “无效的或不支持的命令”        |
| icInvalidOperation   | 35763 | “无效的操作参数”           |
| icExecuting          | 35764 | “还在执行上一个请求”         |
| icInvalidForFtp      | 35765 | “该调用对一个 FTP 连接是无效的” |
| icOutOfHandles       | 35767 | “句柄用尽”              |
| icinetTimeout        | 35768 | “超时”                |
| icInetTimeout        | 35768 | 超时                  |
| icExtendedError      | 35769 | 外部错误                |
| icIntervalError      | 35770 | 内部错误                |
| icInvalidURL         | 35771 | 无效的 URL.            |
| icUnrecognizedScheme | 35772 | 未知的时间表              |
| icNameNotResolved    | 35773 | 未保留的名称              |
| icProtocolNotFound   | 35774 | 协议未找到               |
| icInvalidOption      | 35775 | 无效的选项               |

续表

| 常量                      | 值     | 描述     |
|-------------------------|-------|--------|
| icBadOptionLength       | 35776 | 错误选项长度 |
| icOptionNotSettable     | 35777 | 不可设置选项 |
| icShutDown              | 35778 | 关闭     |
| icIncorrectUserName     | 35779 | 错误的用户名 |
| icLoginFailure          | 35781 | 登录失败   |
| icInetInvalidOpertation | 35782 | 无效操作   |
| icOperationCancelled    | 35783 | 操作取消   |
| icIncorrectHandleType   | 35784 | 错误句柄类型 |
| icIncorrectHandleState  | 35785 | 错误句柄状态 |
| icNotProxyRequest       | 35786 | 不是代理   |
| icRegistryValueNotFound | 35787 | 未找到注册值 |
| icbadRegistryParameteri | 35788 | 错误注册参数 |
| icNoDirectAccess        | 35789 | 无直接访问  |
| IcIncorrect Password    | 35780 | 错误密码   |
| icNoContext             | 35790 | 无上下文   |
| icNoCallback            | 35791 | 无回调    |
| icRequestPending        | 35792 | 请求挂起   |
| icIncorrectFormat       | 35793 | 错误格式   |
| icItemNotFound          | 35794 | 项目未找到  |

续表

| 常量                     | 值     | 描述             |
|------------------------|-------|----------------|
| icCannotConnect        | 35795 | 不能连接           |
| icConnectionAborted    | 35796 | 连接中止           |
| icConnectionReset      | 35797 | 连接复位           |
| icForceEntry           | 35798 | 强迫输入           |
| icInvalidProxyRequest  | 35799 | 无效代理请求         |
| icWouldBlock           | 35800 | 将阻塞            |
| icHandleExists         | 35802 | 句柄存在           |
| icSecCertDateInvalid   | 35803 | 安全身份验证日期无效     |
| icSecCertCnInvalid     | 35804 | 安全身份验证号无效      |
| icHttpsToHttpOnRedir   | 35806 | HTTPS 直接到 HTTP |
| icMixedSecurity        | 35807 | 复合安全性          |
| icChgPostIsNotSecure   | 35808 | 改变邮局不安全        |
| icHttpToHttpsOnRedir   | 35805 | HTTPS 直接到 HTTP |
| icPostIsNonSecure      | 35809 | 非安全邮局          |
| icClientAuthCertNeeded | 35810 | 需要客户身份验证       |
| icInvalidCa            | 35811 | 无效的客户身份        |
| icClientAuthNotSetup   | 35812 | 未设置客户身份验证      |
| icAsyncThreadFailed    | 35813 | 异步线程失败         |
| icRedirectSchemeChange | 35814 | 重定向时间表改变       |

续表

| 常量                           | 值     | 描述                     |
|------------------------------|-------|------------------------|
| icFtpTransferInProgress      | 35876 | FTP: 传输进行中             |
| icFtpDropped                 | 35877 | 连接放下                   |
| icGopherProtocolError        | 35896 | Gopher: 协议错误           |
| icGopherNotFile              | 35897 | Gopher: 不是一个文件         |
| icGopherDataError            | 35898 | Gopher: 数据错误           |
| icGopherEndOfData            | 35899 | Gopher: 数据结束           |
| icGopherInvalidLocator       | 35900 | Gopher: 无效的定位器         |
| icGopherIncorrectLocatorType | 35901 | Gopher: 错误的定位器类型       |
| e                            |       |                        |
| icGopherNotGopherPlus        | 35902 | Gopher: 不是 Gopher plus |
| icGopherAttributeNotFound    | 35903 | Gopher: 属性未找到          |
| icGopherUnknownLocator       | 35904 | Gopher: 未知的定位器         |
| icHeaderNotFound             | 35916 | HTTP: 头部未找到            |
| icHttpDownlevelServer        | 35917 | HTTP: 低级服务器            |
| icHttpInvalidServerResponse  | 35918 | HTTP: 无效的服务器响应         |
| icHttpInvalidHeader          | 35919 | HTTP: 无效头部             |
| icHttpInvalidQueryRequest    | 35920 | HTTP: 无效查询请求           |
| icHttpHeaderAlreadyExists    | 35921 | HTTP: 头已存在             |
| icHttpRedirectFailed         | 35922 | HTTP: 重定向失败            |

续表

| 常量                     | 值     | 描述     |
|------------------------|-------|--------|
| icSecurityChannelError | 35923 | 安全通道错误 |
| icUnableToCacheFile    | 35924 | 不能缓存文件 |

## URL 属性

设置或返回 `Execute` 或 `OpenURL` 方法使用的 URL。

应用于

Microsoft Internet Transfer 控件。

语法

*object*.URL [= *url*]

URL 属性的语法包含下面部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象                   |
| <i>url</i>    | 字符串，指定 <code>Execute</code> 方法中使用的 URL |



## 数据类型

### String

#### 说明

调用 `OpenURL` 或 `Execute` 方法会改变该属性的值。

直到下一次调用 `OpenURL` 或 `Execute` 方法时，对该属性所做的改变才有效。

`URL` 属性至少必须包含一个协议和一个远程主机名。

`URL` 属性可以是目录或文件。例如，下面这两个 `URLs` 都是有效的：

'设置该 `URL`，仅返回文件目录：

```
Inet1.URL = "HTTP://www.microsoft.com"
```

'然而，该 `URL` 将返回文件的文本：

```
Inet1.URL = "HTTP://www.microsoft.com/disclaimer.txt"
```

#### 请参阅

`Execute` 方法。

# UserName 属性

设置或返回与请求一起发送到远程计算机的名称。如果该属性为空，当提出请求时，该控件将把 “anonymous”作为用户名来发送。

应用于

Microsoft Internet Transfer 控件。

语法

```
object.UserName[= name]
```

UserName 属性的语法包含下面部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>name</i>   | 字符串，指定 Execute 方法使用的 UserName |

数据类型

String

## 说明

调用 `OpenURL` 或 `Execute` 方法会改变该属性的值。

直到下一次调用 `OpenURL` 或 `Execute` 方法时，对该属性所做的改变才有效。

## 请参阅

`Password` 属性（Internet Transfer 控件）。

# ListView 控件

**ListView** 控件可使用四种不同视图显示项目。通过此控件，可将项目组成带有或不带有列标头的列，并显示伴随的图标和文本。

## 语法

**ListView**

## 说明

可使用 **ListView** 控件将称作 **ListItem** 对象的列表条目组织成下列四种不同的视图之一：

- 大（标准）图标
- 小图标
- 列表
- 报表

**View** 属性决定在列表中控件使用何种视图显示项目。还可用 **LabelWrap** 属

性控制列表中与项目关联的标签是否可换行显示。另外，还可管理列表中项目的排序方法和选定项目的外观。

ListView 控件包括 ListItem 和 ColumnHeader 对象。ListItem 对象定义 ListView 控件中项目的各种特性，诸如：

- 项目的简要描述。
- 由 ImageList 控件提供的与项目一起出现的图标。
- 附加的文本片段，称作子项目，它们与显示在报表视图中的 ListItem 对象关联。

可以使用 HideColumnHeaders 属性决定是否在 ListView 控件中显示列标头。列标头可以在设计时添加，也可以在运行时添加。设计时，使用 ListView“控件属性”对话框的“列首”选项卡添加列标头。运行时，使用 Add 方法添加 ColumnHeader 对象到 ColumnHeaders 集合中。

**发布须知：**ListView 控件是 MSCOMCTL.OCX 文件中一组 ActiveX 控件的一部分。若要在应用程序中使用 ListView 控件，则必须将 MSCOMCTL.OCX 文件添加到工程中。当发布应用程序时，请将 MSCOMCTL.OCX 文件安装到用户的 Microsoft Windows System 或 System32 目录下。关于如何将 ActiveX 控件添加到 Visual Basic 工程的详细信息，请参阅《Microsoft Visual Basic 程序员指南 6.0》。

## 属性

DropHighlight 属性 (ListView, TreeView 控件), LabelEdit 属性, SelectedItem 属性 (ActiveX 控件), ColumnHeaderIcons 属性, Checkboxes 属性, FullRowSelect 属性, AllowColumnRecorder 属性, FlatScrollBar 属性, GridLines 属性 (ListView 控件), HoverSelection 属性, PictureAlignment 属性, HotTracking 属性, TextBackground 属性, Arrange 属性 (ListView 控件), ColumnHeaders 属性 (ListView 控件), HideColumnHeaders 属性 (ListView 控件), Icons, SmallIcons 属性, ListItems 属性 (ListView 控件), LabelWrap 属性 (ListView 控件), MultiSelect 属性 (ListView, TabStrip 控件), SorKey 属性 (ListView 控件), SortOrder 属性 (ListView 控件), View 属性 (ListView 控件), Sorted 属性 (ListView 控件), TabIndex 属性, DragIcon 属性, DragMode 属性, MouseIcon 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Font 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), Picture 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), HideSelection 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

HitTest 方法（ListView, TreeView 控件）， StartLabelEdit 方法， FindItem 方法（ListView 控件）， GetFirstVisible 方法， SetFocus 方法， Drag 方法， Move 方法， ZOrder 方法， ShowWhatsThis 方法， OLEDrag 方法（ActiveX 控件）， Refresh 方法（ActiveX 控件）。

## 请参阅

TreeView 控件， ImageList 控件， ColumnHeader 对象， ColumnHeaders 集合， Add 方法（ColumnHeaders 集合）， ListItem 对象， ListItems 集合， ListView 控件常量， 使用 ImageList 控件， 使用 ListView 控件。

## Add 方法（ColumnHeaders 集合）

将 ColumnHeader 对象添加到 ListView 控件的 ColumnHeaders 集合中。

## 应用于

ColumnHeader 对象， ColumnHeaders 集合。

# 语法

*object.Add(index, key, text, width, alignment, icon)*

Add 方法的语法包含下面部分：

| 部分               | 描述  |
|------------------|---|
| <i>object</i>    | 必需的。对象表达式，其值是 <b>ColumnHeaders</b> 集合   |
| <i>index</i>     | 可选的。唯一标识对象集合成员的整数   |
| <i>key</i>       | 可选的。唯一的字符串表达式，可以用来访问集合的成员   |
| <i>text</i>      | 可选的。出现在 <b>ColumnHeader</b> 对象中的字符串   |
| <i>width</i>     | 可选的。数值表达式，它使用控件容器的度量单位指定对象的宽度   |
| <i>alignment</i> | 可选的。决定 <b>ColumnHeader</b> 对象中文本对齐方式的整数。关于设置信息，请参阅“请参阅”列表中 <b>Alignment</b> 属性的信息 |
| <i>icon</i>      | 可选的。 <b>Smallicons</b> 图像列表中图像的关键字或索引   |

# 说明

Add 方法返回新插入的 **ColumnHeader** 对象的引用。

使用 *index* 参数在 **ColumnHeaders** 集合的特定位置插入列标头。



当 ColumnHeaders 集合成员可能动态变更时，应使用 Key 属性引用它们，因为任何 ColumnHeader 对象的 Index 属性都可以改变。

请参阅

Clear 方法（ActiveX 控件），Key 属性（ActiveX 控件），Index 属性（ActiveX 控件），Alignment 属性（ColumnHeader 对象），SubItemIndex 属性，SubItems 属性（ListView 控件）。

## Add 方法（ListItems 集合）

添加 ListItem 对象到 ListView 控件的 ListItems 集合中并返回新创建对象的引用。

应用于

ListItem 对象，ListItems 集合。

语法

*object.Add(index, key, text, icon, smallIcon)*

Add 方法的语法包含下面部分：

| 部分               | 描述   |
|------------------|--|
| <i>object</i>    | 必需的。对象表达式，其值是 <code>ListItems</code> 集合  |
| <i>index</i>     | 可选的。指定在何处插入 <code>ListItems</code> 的整数。若未指定索引，则将 <code>ListItems</code> 添加到 <code>ListItems</code> 集合的末尾 |
| <i>key</i>       | 可选的。唯一的字符串表达式，用来访问集合成员   |
| <i>text</i>      | 可选的。与 <code>ListItems</code> 对象控件关联的字符串  |
| <i>icon</i>      | 可选的。当 <code>ListView</code> 控件设为图标视图时，此整数设置从 <code>ImageList</code> 控件中选定的欲显示的图标                         |
| <i>smallIcon</i> | 可选的。当 <code>ListView</code> 控件设为小图标时，此整数设置从 <code>ImageList</code> 控件中选定的欲显示的图标                          |

## 说明

设置 `Icons` 或 `SmallIcons` 属性之前必须先初始化它们。有两种初始化方法在设计时，使用 `ListView` 控件属性对话框的“通用”选项卡指定 `ImageList` 对象；在运行时，使用下列代码初始化：

```
ListView1.Icons = ImageList1 '假设 Imagelist 为 ImageList1。
```

```
ListView1.SmallIcons = ImageList2
```

如果列表尚未排序，则可使用 *index* 参数将 `ListItems` 对象插入到任意位置。如果列表已排序，则将忽略 *index* 参数并根据排序顺序把 `ListItems` 对象插

入到适当的位置。

若未提供 *index*，则 `ListItem` 对象将被添加一个索引，此索引等于集合中 `ListItems` 对象的数目加 1。

当希望对象的 `Index` 属性可变更时，例如希望动态地从集合中添加和删除对象时，使用 `Key` 属性引用 `ListItems` 集合的成员。

请参阅

`Key` 属性（ActiveX 控件），`Index` 属性（ActiveX 控件），`ListView` 控件，`Ghosted` 属性，`Icons`，`SmallIcons` 属性，`SubItems` 属性（`ListView` 控件），`Sorted` 属性（`ListView` 控件），`Selected` 属性（ActiveX 控件）。

示例

下面的示例使用 `Biblio.mdb` 数据库作为资源，通过 `ListItems` 对象移居 `ListView` 控件。要试用此例，请将一个 `ListView` 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

还必须确保已将 `Biblio.mdb` 安装在机器上。在以下代码中检查 `OpenDatabase` 函数中的路径，并改变它，使之反映 `Biblio.mdb` 在机器上的实际路径。

注意：除非添加到 Microsoft DAO 3.5 对象库的“引用”，否则示例无法运行。为此，在 Project 菜单上单击 References。搜索 Microsoft DAO 3.5 对象库并单击复选框来选择。

Private Sub Form\_Load()

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

ListView1.ColumnHeaders. \_

Add , , "Author", ListView1.Width / 3)

ListView1.ColumnHeaders. \_

Add , , "Author ID", ListView1.Width / 3, \_

lvwColumnCenter

ListView1.ColumnHeaders. \_

Add , , "Birthdate", ListView1.Width / 3)

' Set View property to Report.

ListView1.View = lvwReport

'为数据访问对象声明对象变量。

Dim myDb As Database, myRs As Recordset

'设置 Database 为 BIBLIO.MDB 数据库。

' IMPORTANT: the Biblio.mdb must be on your

' machine, and you must set the correct path to

' the file in the OpenDatabase function below.

```
Set myDb = DBEngine.Workspaces(0) _  
    .OpenDatabase("c:\Program Files\VB\BIBLIO.MDB")
```

'设置 recordset 为 "Authors" 表。

```
Set myRs = _  
myDb.OpenRecordset("Authors", dbOpenDynaset)
```

'声明变量以添加 ListItem 对象。

```
Dim itmX As ListItem
```

'若当前记录不是最后一条记录，则添加一个 ListItem 对象。

'ListItem 对象的文本使用 author 字段。

'ListItem 对象的 SubItem(1) 使用 AuthorID 字段。

'ListItem 对象的 SubItem(2) 使用 "Year of Birth" 字段。

```
While Not myRs.EOF
```

```
    Set itmX = ListView1.ListItems. _
```

```
    Add(, , CStr(myRs!Author),1) 'Author 字段。
```

'若 AuthorID 字段不为空，则将 SubItem 1 设置为此字段。

```
If Not IsNull(myRs!Au_id) Then
```

```
    itmX.SubItems(1) = CStr(myRs!Au_id)
```

```
End If
```

```
'若 birth 字段不为空，则将 SubItem 2 设置为此字段。  
If Not IsNull(myRs![Year Born]) Then  
    itmX.SubItems(2) = myRs![Year Born]  
End If  
myRs.MoveNext    '移动到下一条记录。  
Wend  
End Sub
```

## Add 方法（ListSubItems 集合）

向 ListSubItems 集合中添加一个 ListSubItem 对象，并返回对新创建对象的引用。

应用于

ListSubItems 集合。

语法

*object.Add (index, key, text, ReportIcon, ToolTipText)*

| 部分                 | 描述   |
|--------------------|--|
| <i>object</i>      | 必需的。一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>index</i>       | 可选的。一个整数，指定对象插入的位置，如果没有指定此参数，对象将添加到集合的末尾   |
| <i>key</i>         | 可选的。一个字符串，唯一标识该对象，使用这个值在集合中检索指定对象  |
| <i>text</i>        | 可选的。在“报表”视图中 <b>ListView</b> 控件所显示的字符串   |
| <i>reportIcon</i>  | 可选的。 <b>Index</b> 或 <b>Key</b> 属性值，指定相关的 <b>ImageList</b> 控件中的 <b>ListImage</b> 对象 |
| <i>toolTipText</i> | 可选的。当鼠标指针停留在 <b>ListSubItem</b> 时显示的字符串  |

## 说明

**ListSubItems** 集合替换了 **SubItems** 字符串数组。**SubItems** 数组在 **ListView** 控件中仍然有效，但是建议新的应用程序使用 **ListSubItems** 集合，可以增强灵活性。

# Alignment 属性（ColumnHeader 对象）

返回或设置 ColumnHeader 对象中文本的对齐方法。

应用于

ListView 控件，ColumnHeader 对象，ColumnHeaders 集合。

语法

*object*.Alignment [= *integer*]

Alignment 属性的语法包含下面部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 对象表达式，其值是 ColumnHeader 对象 |
| <i>integer</i> | 决定对齐方式的整数，如“设置”中所描述       |

设置

*integer* 的设置值为：



| 常量                     | 值 | 描述             |
|------------------------|---|----------------|
| <i>lvwColumnLeft</i>   | 0 | （缺省）左对齐。文本向左对齐 |
| <i>lvwColumnRight</i>  | 1 | 右对齐。文本向右对齐     |
| <i>lvwColumnCenter</i> | 2 | 居中。文本居中对齐      |

请参阅

ColumnHeader 对象，ColumnHeaders 集合，Add 方法（ColumnHeaders 集合），Align 属性。

## AllowColumnReorder 属性

返回或设置一个值，确定用户是否可以用鼠标对列进行重新排列。

应用于

ListView 控件。

语法

*object*.AllowColumnReorder [= *boolean*]

AllowColumnReorder 属性语法有如下几部分:

| 部分             | 描述                              |
|----------------|---------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象        |
| <i>boolean</i> | 一个布尔表达式，指定用户是否能够对列重新排列，如设置值所描述的 |

### 设置

*boolean* 设置如下:

| 常量    | 描述             |
|-------|----------------|
| False | (缺省)用户不能对列重新排列 |
| True  | 用户能够对列重新排列     |

### Arrange 属性 (ListView 控件)

返回或设置一个值，此值决定如何排列 ListView 控件中的图标或小图标视图。

应用于

ListView 控件。

语法

*object*.Arrange [= *value*]

Arrange 属性的语法包含下面部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 对象表达式，其值是 ListView 控件        |
| <i>value</i>  | 决定如何排列图标或小图标的整数或常量，如“设置”中所描述 |

设置

*value* 的设置值为：

| 常量          | 值 | 描述              |
|-------------|---|-----------------|
| lvwNone     | 0 | （缺省）无           |
| lvwAutoLeft | 1 | 左对齐。项目自动沿控件左侧对齐 |
| lvwAutoTop  | 2 | 顶对齐。项目自动沿控件顶端对齐 |

请参阅

Icons, SmallIcons 属性, Sortkey 属性(ListView 控件), SortOrder 属性(ListView 控件), View 属性(ListView 控件), ListView 控件常量, Sorted 属性(ListView 控件)。

## 示例

本示例将若干 ListViewItem 对象和子项目添加到 ListView 控件中。单击 OptionButton 控件时将根据 OptionButton 的 Index 值设置 Arrange 属性。要试用此例，请将一个包含三个 OptionButton 控件的控件数组、一个 ListView 控件和两个 ImageList 控件放置在窗体上，并将代码粘贴到窗体的声明部分。运行示例，并单击 OptionButton 改变 Arrange 属性。

```
Private Sub Option1_Click(Index as Integer)
    '设置 Arrange 属性为 Option1.Index。
    ListView1.Arrange = Index
End Sub

Private Sub Form_Load()
    '用 Arrange 选项标注 OptionButton 控件。
    Option1(0).Caption = "No Arrange"
    Option1(1).Caption = "Align Auto Left"
```

Option1(2).Caption = "Align Auto Top"

'为创建 ListView 和 ImageList 对象声明变量。

Dim i As Integer

Dim itmX As ListItem '声明 ListItem 的对象变量。

Dim imgX As ListImage '声明 ListImages 的对象变量。

'添加 ListImage 对象到 ImageList 控件中。

Set imgX = ImageList1.ListImages. \_

Add(,LoadPicture("icons\mail\mail01a.ico"))

ListView1.ListItems.Add(imgX) '与 ImageList 控件关联。

'添加 10 个均包含图标的 ListItem 对象。

For i = 1 To 10

Set itmX = ListView1.ListItems.Add()

itmX.Icon = 1 '图标。

itmX.Text = "ListItem " & i

Next i

End Sub

# ColumnClick 事件

单击 ListView 控件中的 ColumnHeader 对象时，该事件发生。仅当控件的 View 属性为报表视图时可用。

应用于

ListView 控件。

语法

```
Private Sub object_ColumnClick(ByVal columnheader As ColumnHeader)
```

ColumnClick 事件的语法包含下面部分：

| 部分                  | 描述                      |
|---------------------|-------------------------|
| <i>object</i>       | 对象表达式，其值是 ListView 控件   |
| <i>columnheader</i> | 被单击的 ColumnHeader 对象的引用 |

说明

通常在代码中使用 Sorted、SortKey 和 SortOrder 属性把被单击列中的

ListItem 对象排序。

请参阅

ColumnHeader 对象，ColumnHeaders 集合，SortKey 属性（ListView 控件），SortOrder 属性（ListView 控件），Sorted 属性（ListView 控件）。

示例

本示例将三个 ColumnHeader 对象添加到 ListView 控件中，并将 Biblio.mdb 数据库的 Publishers 记录植入控件。一个由两个 OptionButton 控件组成的数组提供将记录排序的两个选项。单击 ColumnHeader 将根据由 OptionButtons 决定的 SortOrder 属性将 ListView 控件排序。要试用此例，请将一个 ListView 和一个由 OptionButton 控件组成的控件数组放置在窗体上，并将代码粘贴到窗体的声明部分。运行示例并单击 ColumnHeaders 进行排序，单击 OptionButton 来切换 SortOrder 属性。

**注意：**必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.5 对象库的引用，否则示例无法运行。

```
Private Sub Option1_Click(Index as Integer)
```

```
'这些 OptionButtons 提供两种选择：升序（索引 0），  
'和降序(索引 1)。请单击其一
```

'为 ListView 控件设置 SortOrder。

ListView1.SortOrder = Index

ListView1.Sorted = True '将列表排序。

End Sub

Private Sub Form\_Load()

'为 ColumnHeader 对象创建对象变量。

Dim clmX As ColumnHeader

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Company", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Address", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Phone", ListView1.Width / 3)

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表型。

'用 SortOrder 选项标注 OptionButton 控件。

Option1(0).Caption = "Ascending (A-Z)"

Option1(1).Caption = "Descending (Z-A)"



ListView1.SortOrder = lvwAscending ' Sort ascending.

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'为添加 ListItem 对象创建变量。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 subitem(1) 使用 Address 字段。

'ListItem 对象的 subitem(2) 使用 Phone 字段。

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

'若 Address 字段不为空，则设置 subitem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'若 Phone 字段不为空，则设置 subitem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

End Sub

Private Sub ListView1\_ColumnClick(ByVal ColumnHeader As ColumnHeader)

'单击 ColumnHeader 对象时，将根据

'那一列的子项目把 ListView 控件排序。

'设置 SortKey 为 ColumnHeader 的索引值减 1

ListView1.SortKey = ColumnHeader.Index - 1

'设置 Sorted 为 True 以将列表排序。

ListView1.Sorted = True

End Sub

## ColumnHeader 对象、ColumnHeaders 集合

ColumnHeader 对象是 ListView 控件中包含标头文字的项目。

ColumnHeaders 集合包含一个或多个 ColumnHeader 对象。

## 语法

- `listview.ColumnHeaders`
- `listview.ColumnHeaders(index)`

根据标准的集合语法，上面的语法行分别引用集合和集合中单个元素。

ColumnHeader 对象，ColumnHeaders 集合的语法包含下面部分：

| 部分              | 描述   |
|-----------------|--|
| <i>listview</i> | 对象表达式，其值是 ListView 控件  |
| <i>index</i>    | 一个唯一标识对象集合成员的整数或字符串。如果为整数，它将是 Index 属性的值；如果为字符串，它将是 Key 属性的值 |

## 说明

只能在报表视图中查看 ColumnHeader 对象。

可以在设计时或在运行时添加 ColumnHeader 对象到 ListView 控件中。

利用 ColumnHeader 对象，用户可以：

- 单击对象触发 **ColumnClick** 事件并根据数据项目将项目排序。
- 拖动对象的右边框来调整列宽度。
- 在报表视图中隐藏 **ColumnHeader** 对象。

**ListView** 控件中总是有一列，即列 1。这一列包含实际的 **ListItem** 对象而不是它们的子项目。第二列（列 2）包含子项目。因此，**ColumnHeader** 对象总是比子项目多一个，**ListItem** 对象的 **SubItems** 属性是大小为 **ColumnHeaders.Count - 1** 的、基于 1 的数组。

**ColumnHeader** 对象的数目决定每个 **ListItem** 对象可包含的子项目数目。删除 **ColumnHeader** 对象后所有与列关联的子项目也将被删除，并且每个 **ListItem** 对象的子项目数组将平移以更新 **ColumnHeader** 的索引，而这将导致剩余的列标头 **SubItemIndex** 属性的改变。

## 属性

**Count** 属性（**ActiveX** 控件），**Item** 属性（**ActiveX** 控件），**Key** 属性（**ActiveX** 控件），**Text** 属性（**ActiveX** 控件），**Index** 属性（**ActiveX** 控件），**Left**, **Top** 属性（**ActiveX** 控件），**Tag** 属性（**ActiveX** 控件），**Alignment** 属性（**ColumnHeader** 对象），**Position** 属性（**ColumnHeader** 对象），**SubItemIndex** 属性，**Icon** 属性（**Windows** 常见控件）。

## 方法

**Clear** 方法（ActiveX 控件），**Remove** 方法（ActiveX 控件），**Add** 方法（ColumnHeaders 集合）。

## 请参阅

**Clear** 方法（ActiveX 控件），**Add** 方法（ColumnHeaders 集合），**ListItem** 对象，**ListItems** 集合，**ColumnClick** 事件，**ColumnHeaders** 属性（ListView 控件），**HideColumnHeaders** 属性（ListView 控件），**SortKey** 属性（ListView 控件），**SortOrder** 属性（ListView 控件），**SubItemIndex** 属性，**SubItems** 属性（ListView 控件），**Sorted** 属性（ListView 控件）。

## ColumnHeaders 属性（ListView 控件）

返回 **ColumnHeader** 对象集合的引用。

## 应用于

**ListView** 控件，**ColumnHeader** 对象，**ColumnHeaders** 集合。

## 语法

*object*.ColumnHeaders

*object* 置换元代表一个对象表达式，其值是 ListView 控件。

## 说明

可以使用标准的集合方法（例如 Remove 方法）操作 ColumnHeader 对象。可通过 ColumnHeader 的索引或通过 Key 属性中存储的唯一关键字访问集合中的每个 ColumnHeader。

## 请参阅

Add 方法（ColumnHeaders 集合）。

## FindItem 方法（ListView 控件）

查找并返回 ListView 控件中 ListItem 对象的引用。

应用于

ListView 控件，ListItem 对象，ListItems 集合。

语法

*object.FindItem (string, value, index, match)*

FindItem 方法的语法包含下面部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 必需的。对象表达式，其值是 ListView 控件   |
| <i>string</i> | 必需的。指定欲查找的 ListItem 对象的字符串表达式   |
| <i>value</i>  | 可选的。整数或常量，它指定字符串是否与 ListItem 对象的 Text、Subitems 及 Tag 属性相匹配，如设置值中所描述               |
| <i>index</i>  | 可选的。唯一标识对象集合成员并指定搜索起始位置的整数或字符串。若为整数，其值为 Index 属性值；若为字符串，其值为 Key 属性值。未指定索引时缺省索引为 1 |
| <i>match</i>  | 可选的。指定项目的 Text 属性与字符串怎样匹配的整数或常量，如“设置”中所描述   |

# 设置

*value* 的设置值为:

| 常量         | 值 | 描述                                 |
|------------|---|------------------------------------|
| LvwText    | 0 | (缺省) 将字符串与 ListItem 对象的 Text 属性相匹配 |
| LvwSubitem | 1 | 将字符串与 ListItem 对象的 SubItems 属性相匹配  |
| LvwTag     | 2 | 将字符串与 ListItem 对象的 Tag 属性相匹配       |

*match* 的设置值为:

| 常量           | 值 | 描述  |
|--------------|---|---|
| LvwWholeWord | 0 | (缺省) 一个整数或常量, 它指定若项目的 Text 属性由所搜索的整字开始时匹配成功。搜索条件非文本时忽略此设置 |
| LvwPartial   | 1 | 一个整数或常量, 它指定若项目的 Text 属性由所搜索的字符串开始时匹配成功。搜索条件非字符串时忽略此设置    |

# 说明

如指定文本为搜索条件则可使用 *lvwPartial*, 这样若 ListItem 对象的 Text



属性由所搜索的字符串开始时匹配成功。例如，要查找文本为 “Autoexec.bat” 的 `ListItItem`，使用下列代码：

```
'创建 ListItItem 变量。  
Dim itmX As ListItItem  
'设置变量来查找项目。  
Set itmX = ListView1.FindItem("Auto",,,,lvwpartial)
```

请参阅

`key` 属性（ActiveX 控件），`Text` 属性（ActiveX 控件），`Index` 属性（ActiveX 控件），`SubItems` 属性（ListView 控件），`ListView` 控件常量，`Tag` 属性。

示例

本示例将 `Biblio.mdb` 数据库中 `Publishers` 表的内容植入 `ListView` 控件中。同时也将 `FindItem` 方法的三个选项植入 `ComboBox` 控件。一个 `CommandButton` 包含 `FindItem` 方法的代码；单击按钮后将提示输入欲搜索的字符串，然后使用 `FindItem` 方法在 `ListView` 控件中搜索此字符串。如找到字符串则使用 `EnsureVisible` 方法滚动控件以显示找到的 `ListItItem` 对象。要试用此例，请将 `ListView`、`ComboBox` 及 `CommandButton` 控件放置在窗体上，并将代码粘贴到窗体的声明部分。运行示例并单击命令按钮。

注意：必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0 对象库的引用，否则示例无法运行。

Private Sub Form\_Load()

'为 ColumnHeader 对象创建对象变量。

Dim clmX As ColumnHeader

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Company", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Address", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Phone", ListView1.Width / 3)

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表型。

Command1.Caption = "&FindItem"

'用 FindItem 选项标注 OptionButton 控件。

Option1(0).Caption = "Text"

Option1(1).Caption = "SubItem"

Option1(2).Caption = "Tag"

ListView1.FindItem = 0 '设置 ListView 控件的 FindItem 属性为文本。  
End With

'将数据库记录植入 ListView 控件。

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'使用新对象的引用设置属性。

'设置 Text 属性为 Name 字段 (myRS!Name)。

'设置 SubItem(1) 为 Address 字段 (myRS!Address)。

'设置 SubItem(7) 为 Phone 字段 (myRS!Telephone)。

While Not myRs.EOF

Dim itmX As ListItem 'ListItem 变量。

Dim intCount As Integer '计数器变量。

'使用 Add 方法添加新的 ListItem 并为新引用设置对象。

'使用引用设置

属性。

```
Set itmX = ListView1.ListItems.Add(, CStr(myRs!Name))  
intCount = intCount + 1 'Tag 属性计数器递增。  
itmX.Tag = "ListItem " & intCount '用计数器值设置 Tag。
```

```
'若 Address 字段不为空，则设置 subitem 1 为此字段。  
If Not IsNull(myRs!Address) Then  
    itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。  
End If
```

```
'若 Phone 字段不为空，则设置 subItem 2 为此字段。  
If Not IsNull(myRs!Telephone) Then  
    itmX.SubItems(2) = myRs!Telephone 'Phone 字段。  
End If
```

```
myRs.MoveNext '移动到下一条记录。
```

```
Wend
```

```
End Sub
```

```
Private Sub Command1_Click()
```

```
'FindItem 方法。
```

```
'创建名为 intSelectedOption 的整数类型变量
```

```
'来存储选定按钮的索引。
```

```
'创建名为 strFindMe 的字符串变量。使用 InputBox
```

```
'把查找的字符串存储在变量中。使用
```

' FindItem 方法查找字符串。使用 Option1

'切换决定在何处查找的 FindItem 参数。

```
Dim intSelectedOption as Integer
```

```
Dim strFindMe As String
```

```
If Option1(0).Value = True then
```

```
    strFindMe = InputBox("Find in " & Option1(0).Caption)
```

```
    intSelectedOption = lvwText
```

```
End If
```

```
If Option1(1).Value = True then
```

```
    strFindMe = InputBox("Find in " & Option1(1).Caption)
```

```
    intSelectedOption = lvwSubItem
```

```
End If
```

```
If Option1(2).Value = True then
```

```
    strFindMe = InputBox("Find in " & Option1(2).Caption)
```

```
    intSelectedOption = lvwTag
```

```
End If
```

'FindItem 方法返回找到的项目的引用，所以

'必须创建对象变量并将

'找到的项目设置给它。

```
Dim itmFound As ListItem    'FoundItem 变量。
```

```
Set itmFound = ListView1. _
```

```
FindItem(strFindMe, intSelectedOption, , lvwPartial)
```

'若未找到符合条件的 ListItem 则通知用户并退出。如果

'找到 ListItem, 则使用 EnsureVisible 方法滚动控件,

'并选定 ListItem。

If itmFound Is Nothing Then '若没有匹配成功, 则通知用户并退出。

```
    MsgBox "No match found"
```

```
    Exit Sub
```

Else

```
    itmFound.EnsureVisible '滚动 ListView 以显示找到的 ListItem。
```

```
    itmFound.Selected = True '选定 ListItem。
```

'将焦点返回给控件以查看选择。

```
    ListView1.SetFocus
```

End If

End Sub

Private Sub ListView1\_LostFocus()

'控件失去焦点后, 重新将每个

'ListItem 的 Selected 属性设置为 False。

```
Dim i As Integer
```

```
For i = 1 to ListView1.ListItems.Count
```

```
    ListView1.ListItems.Item(i).Selected = False
```

```
Next i
End Sub
```

## FlatScrollBar 属性

返回或设置一个值，确定对象中滚动条的外观。

应用于

ListView 控件。

语法

*object*.FlatScrollBar [= *boolean*]

FlatScrollBar 属性语法有如下几部分:

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>boolean</i> | 一个布尔表达式，指定滚动条的外观，如设置值所描述的 |

## 设置

*boolean* 设置如下:

| 常量    | 描述                    |
|-------|-----------------------|
| False | (缺省)显示标准滚动条           |
| True  | 对象显示 FlatScrollBar 样式 |

## GetFirstVisible 方法

返回控件内部区域中第一个可视对象的引用。

应用于

ImageCombo 控件，ListView 控件。

语法

*object*.GetFirstVisible()

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。



## 说明

ListView 控件可包含的 ListItem 对象数目比在其内部区域中可视的 ListItem 对象数目多。可以使用由 GetFirstVisible 方法返回的引用决定在列表或在报表视图中的第一个可视的 ListItem 对象。

## 请参阅

Key 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), EnsureVisible 方法。

## 示例

本示例将 Biblio.mdb 数据库 Publishers 表的内容植入 ListView 控件中。单击 CommandButton 控件将显示第一个可视项目的文本。单击列标头以变更 SortKey 属性并再次单击 CommandButton。要试用此例, 请将 ListView 和 CommandButton 控件放置在窗体上, 并将代码粘贴到窗体的声明部分。

**注意:** 必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0 对象库的引用, 否则示例无法运行。

```
Private Sub Command1_Click()
```

```
    '创建 ListItem 变量并将变量设置为
```

'由 GetFirstVisible 方法返回的对象。使用引用  
'显示 ListItem 的文本。

Dim itmX As ListItem

Set itmX = ListView1.GetFirstVisible

MsgBox itmX.Text

End Sub

Private Sub Form\_Load()

'为 ColumnHeader 对象创建对象变量。

Dim clmX As ColumnHeader

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

Set clmX = ListView1.ColumnHeaders. \_

Add(, , "Company", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, , "Address", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, , "Phone", ListView1.Width / 3)

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

```
Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
```

'设置 recordset 为 Publishers 表。

```
Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)
```

'创建变量以添加 ListItem 对象。

```
Dim itmX As ListItem
```

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 subitem(1) 使用 Address 字段。

'ListItem 对象的 subitem(2) 使用 Phone 字段。

```
While Not myRs.EOF
```

```
    Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))
```

'若 Address 字段不为空，则设置 subItem 1 为此字段。

```
    If Not IsNull(myRs!Address) Then
```

```
        itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。
```

```
    End If
```

'若 Phone 字段不为空，则设置 subItem 2 为此字段。

```
    If Not IsNull(myRs!Telephone) Then
```

```
        itmX.SubItems(2) = myRs!Telephone 'Phone 字段。
```

```
End If

myRs.MoveNext    '移动到下一条记录。
Wend
ListView1.View = lvwReport    '设置 view 属性为报表型。
End Sub

Private Sub ListView1_ColumnClick(ByVal ColumnHeader As ColumnHeader)
    ListView1.SortKey = ColumnHeader.Index - 1
    ListView1.Sorted = True
End Sub
```

## Ghosted 属性

返回或设置一个值，它决定是否 **ListView** 控件的 **ListItem** 对象不可用（此时控件是暗淡的）。

应用于

**ListView** 控件，**ListItem** 对象，**ListItems** 集合。

# 语法

*object*.Ghosed [= *boolean*]

Ghosed 属性的语法包含下面部分：

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 对象表达式，其值是 ListItem 对象         |
| <i>boolean</i> | 指定图标或小图标是否为幻影的布尔表达式，如“设置”中所描述 |

# 设置

*boolean* 的设置值为：

| 设置值   | 描述                  |
|-------|---------------------|
| True  | 无法使用 ListItem 对象    |
| False | （缺省）可使用 ListItem 对象 |

# 说明

通常使用 Ghosed 属性显示 ListItem 被剪切或由于某些原因被禁止使用。

选择幻影的 `ListItem` 时，其标签将突出显示，但不突出显示图像。

请参阅

`SelectedItem` 属性（ActiveX 控件）。

示例

本示例将 `Biblio.mdb` 数据库 `Authors` 表的内容植入 `ListView` 控件并允许使用 `OptionButton` 控件设置 `MultiSelect` 属性选项。可以只选择任何一个项目，也可按下 `SHIFT` 键选择多个项目。单击 `CommandButton` 设置选定项目的 `Ghosted` 属性为 `True`。要试用此例，请将一个包含两个 `OptionButton` 控件的控件数组、一个 `ListView` 控件、一个 `ImageList` 控件和一个 `CommandButton` 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

**注意：**必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0 对象库的引用，否则示例无法运行。运行此例，单击 `OptionButton` 选择 `MultiSelect` 选项，单击项目选定它们后，再次单击 `CommandButton` 将使它们成为幻影。

```
Private Sub Command1_Click()
```

```
    Dim x As Object
```

```
    Dim i As Integer
```

'使选定的项目成为幻影。

```
If ListView1.SelectedItem Is Nothing Then Exit Sub
    For i = 1 To ListView1.ListItems.Count
        If ListView1.ListItems(i).Selected = True Then
            ListView1.ListItems(i).Ghosted = True
        End If
    Next i
End Sub
```

Private Sub Form\_Load()

'为 ColumnHeader 对象创建对象变量。

```
Dim clmX As ColumnHeader
```

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

```
Set clmX = ListView1.ColumnHeaders. _
```

```
Add(, "Company", ListView1.Width / 3)
```

```
Set clmX = ListView1.ColumnHeaders. _
```

```
Add(, "Address", ListView1.Width / 3)
```

```
Set clmX = ListView1.ColumnHeaders. _
```

```
Add(, "Phone", ListView1.Width / 3)
```

'用 MultiSelect 选项标注 OptionButton 控件。

```
Option1(0).Caption = "No MultiSelect"
```

Option1(1).Caption = "MultiSelect"

ListView1.MultiSelect = 1 '设置 MultiSelect 为 True

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表。

'添加一个图像到 ImageList 控件中。

Dim imgX As ListImage

Set imgX = ImageList1.ListImages. \_

Add(, , LoadPicture("icons\mail\mail01a.ico"))

ListView1.Icons = ImageList1

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'创建变量以添加 ListItem 对象。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 SubItem(1) 使用 Address 字段。



'ListItem 对象的 SubItem(2) 使用 Phone 字段。

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

itmX.Icon = 1 '设置图标为 ImageList 图标。

'若 Address 字段不为空，设置 SubItem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'若 Phone 字段不为空，则设置 SubItem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

ListView1.View = lvwIcon '显示图标视图。

Command1.Caption = "Cut" '设置 CommandButton 的标题。

'为窗体添加标题。

Me.Caption = "Select any item(s) and click 'Cut'."

End Sub

```
Private Sub Option1_Click(Index as Integer)
    ListView1.MultiSelect = Index
End Sub
```

## GridLines 属性 （ListView 控件）

返回或设置一个值，确定在“报表”视图中 ListView 控件是否显示网格线。

应用于

ListView 控件。

语法

*object*.GridLines [= *boolean*]

GridLines 属性语法有如下几部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>boolean</i> | 一个布尔表达式，指定网格线的外观，如设置值所描述的 |

# 设置

*boolean* 设置如下:

| 常量    | 描述         |
|-------|------------|
| False | (缺省)不显示网格线 |
| True  | 控件显示网格线    |

## HideColumnHeaders 属性（ListView 控件）

返回或设置是否在报表视图中隐藏 ListView 控件的 ColumnHeader 对象。

应用于

ListView 控件，ColumnHeader 对象，ColumnHeaders 集合。

### 语法

*object.HideColumnHeaders* [= *boolean*]

HideColumnHeaders 属性的语法包含下面部分：

| 部分             | 描述                                 |
|----------------|------------------------------------|
| <i>object</i>  | 对象表达式，其值是 <code>ListView</code> 控件 |
| <i>boolean</i> | 指定是否在报表视图中隐藏列标头的布尔表达式，如“设置”中所描述    |

## 设置

`boolean` 的设置值为：

| 设置值                | 描述        |
|--------------------|-----------|
| <code>True</code>  | 列标头不可视    |
| <code>False</code> | （缺省）列标头可视 |

## 说明

即使 `HideColumnHeaders` 属性设置为 `True`，`ListItem` 对象和任何相关的子项目仍保持可视。

## 请参阅

`ColumnHeaders` 属性（`ListView` 控件）。

## 示例

本示例将若干包含子项目的 **ListItem** 对象添加到 **ListView** 控件中。单击 **CommandButton** 时，**HideColumnHeaders** 属性将在 **True (-1)** 和 **False (0)**之间切换。要试用此例，请将 **ListView** 和 **CommandButton** 控件放置在窗体上，并将代码粘贴到窗体的声明部分。运行此例并单击 **CommandButton** 以切换 **HideColumnHeaders** 属性。

```
Private Sub Command1_Click()
```

```
    '在开和关之间切换 HideColumnHeaders 属性。
```

```
    ListView1.HideColumnHeaders = Abs(ListView1.HideColumnHeaders) - 1
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    Dim clmX As ColumnHeader
```

```
    Dim itmX As ListItem
```

```
    Dim i As Integer
```

```
    Command1.Caption = "HideColumnHeaders"
```

```
    '为控件添加三个 ColumnHeader 对象。
```

```
    For i = 1 To 3
```

```
        Set clmX = ListView1.ColumnHeaders.Add()
```

```
        clmX.Text = "Col" & i
```

Next I

'设置视图为报表型。

ListView1.View = lvwReport

'为控件添加十个 ListItem。

For i = 1 To 10

Set itmX = ListView1.ListItems.Add()

itmX.Text = "ListItem " & i

itmX.SubItems(1) = "Subitem 1"

itmX.SubItems(2) = "Subitem 2"

Next i

End Sub

## HoverSelection 属性

返回或设置一个值，确定当鼠标指针在 **ListItem** 对象上停留时，对象是否被选中。

应用于

ListView 控件。

语法

*object*.HoverSelection [= *boolean*]

HoverSelection 属性语法有如下几部分:

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，指定对象是否被选中，如设置值所描述的 |

设置

*boolean* 设置如下:

| 常量    | 描述                  |
|-------|---------------------|
| False | (缺省)不选中             |
| True  | 鼠标指针在项目上停留片刻后，项目被选中 |

# Icon、SmallIcon 属性（ListItem 对象）

返回或设置与 ImageList 控件中 ListItem 对象关联的图标或小图标的索引值或关键字值。

应用于

ListItem 对象，ListItems 集合。

语法

*object*.Icon [= *index*]

*object*.SmallIcon [= *index*]

Icon, SmallIcon 属性的语法包括如下部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是 ListItem 对象  |
| <i>index</i>  | 标识关联 ImageList 控件中的图标或小图标的整数或唯一的字符串。若为整数，其值为 ListItem 对象的 Index 属性；若为字符串，其值为 Key 属性值 |



## 说明

使用 `ListItem` 对象中的图标之前必须将 `ImageList` 控件与包含对象的 `ListView` 控件相关联。关于详细的信息，请参阅 `Icons`，`SmallIcons` 属性（`ListView` 控件）的信息。下面的示例说明了正确的语法：

```
ListView1.ListItems(1).SmallIcons=1
```

`ListView` 控件设置为小图标视图时将显示图像。

## 请参阅

`ImageList` 控件，`ListItem` 对象，`ListItems` 集合，`Icons`，`SmallIcons` 属性。

## 示例

本示例将 `Biblio.mdb` 数据库中 `Publishers` 表的内容植入 `ListView` 控件。`View` 属性选项标注在四个 `OptionButton` 控件中。还必须将两个 `ImageList` 控件放置在窗体上：一个包含 `Icon` 属性的图像，另一个包含每个 `ListItem` 对象 `SmallIcon` 属性的图像。要试用此例，请将一个 `ListView`、一个包含四个 `OptionButton` 控件的控件数组和两个 `ImageList` 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

注意：必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0

对象库的引用，否则示例无法运行。运行示例并单击 ComboBox 控件来切换视图。

```
Private Sub Option1_Click(Index as Integer)
```

```
    '设置 ListView 控件的 View 属性为
```

```
    ' Option1 的索引
```

```
    ListView1.View = Index
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    '为 ColumnHeader 对象创建对象变量。
```

```
    Dim clmX As ColumnHeader
```

```
    '添加 ColumnHeaders。列宽度等于控件的宽度
```

```
    '除以 ColumnHeader 对象的数目。
```

```
    Set clmX = ListView1.ColumnHeaders. _
```

```
    Add(, "Company", ListView1.Width / 3)
```

```
    Set clmX = ListView1.ColumnHeaders. _
```

```
    Add(, "Address", ListView1.Width / 3)
```

```
    Set clmX = ListView1.ColumnHeaders. _
```

```
    Add(, "Phone", ListView1.Width / 3)
```

```
    ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。
```

```
    ListView1.View = lvwReport '设置 View 属性为报表型。
```

'添加一个图像到 ImageList1--图标 ImageList。

```
Dim imgX As ListImage
```

```
Set imgX = ImageList1.ListImages. _
```

```
Add(, , LoadPicture("icons\mail\mail01a.ico"))
```

'添加一个图像到 ImageList2--小图标 ImageList。

```
Set imgX = ImageList2.ListImages. _
```

```
Add(, , LoadPicture("bitmaps\assorted\w.bmp"))
```

'若要将 ImageList 控件和 ListView 控件一起使用，必须

'将 Icons 和 SmallIcons 属性

'与特定的 ImageList 控件关联。

```
ListView1.Icons = ImageList1
```

```
ListView1.SmallIcons = ImageList2
```

'用 View 选项标注 OptionButton 控件。

```
Option1(0).Caption = "Icon"
```

```
Option1(1).Caption = "SmallIcon"
```

```
Option1(2).Caption = "List"
```

```
Option1(3).Caption = "Report"
```

```
ListView1.View = lvwIcon '设置为图标视图
```

'为数据访问对象创建对象变量。

```
Dim myDb As Database, myRs As Recordset
```

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'为添加 ListItem 对象创建变量。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 SubItem(1) 使用 Address 字段。

'ListItem 对象的 SubItem(2) 使用 Phone 字段

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

itmX.Icon = 1 '设置 ImageList1 中的一个图标。

itmX.SmallIcon = 1 '设置 ImageList2 中的一个图标。

'若 Address 字段不为空，则设置 SubItem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'若 Phone 字段不为空，则设置 SubItem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

```
        itmX.SubItems(2) = myRs!Telephone 'Phone 字段。  
    End If  
  
    myRs.MoveNext '移动到下一条记录。  
Wend  
End Sub
```

## Icons, SmallIcons 属性

返回或设置与 **ListView** 控件中图标视图和小图标视图关联的 **ImageList** 控件。

应用于

**ListView** 控件。

语法

*object*.Icons [= *imagelist*]

*object*.SmallIcons [= *imagelist*]

Icons, SmallIcons 属性的语法包括下述部分：

| 部分               | 描述                     |
|------------------|------------------------|
| <i>object</i>    | 对象表达式，其值是 ListView 控件  |
| <i>imagelist</i> | 对象表达式，其值是 ImageList 控件 |

## 说明

如果要在运行时将 ImageList 控件与 ListView 控件关联，则请将这些属性设置为期望的 ImageList 控件。

ListView 控件中的每个 ListItem 对象也都具有 Icon 和 SmallIcon 属性，它们索引 ListImage 对象并决定将显示哪个图像。

将 ImageList 与 ListView 控件关联后，就可在过程中使用 Index 或 Key 属性值引用 ListImage 对象。

## 请参阅

ImageList 控件，ListImage 对象，ListImages 集合，ListItem 对象，ListItems 集合，Icon, SmallIcon 属性（ListItem 对象）。

## 示例

本示例将 Biblio.mdb 数据库中 Publishers 表的内容植入 ListView 控件。View 属性选项标注在四个 OptionButton 控件中。还必须将两个 ImageList 控件放置在窗体上：一个包含 Icon 属性的图像，另一个包含每个 ListItem 对象 SmallIcon 属性的图像。要试用此例，请将一个 ListView、一个包含四个 OptionButton 控件的控件数组和两个 ImageList 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

**注意：**必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0 对象库的引用，否则示例无法运行。运行示例并单击 ComboBox 控件来切换视图。

```
Private Sub Option1_Click(Index as Integer)
```

```
    '设置 ListView 控件的 View 属性为
```

```
    ' Option1 的索引
```

```
    ListView1.View = Index
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    '为 ColumnHeader 对象创建对象变量。
```

```
    Dim clmX As ColumnHeader
```

```
    '添加 ColumnHeaders。列宽度等于控件的宽度
```

'除以 ColumnHeader 对象的数目。

```
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Company", ListView1.Width / 3)  
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Address", ListView1.Width / 3)  
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Phone", ListView1.Width / 3)
```

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表型。

'添加一个图像到 ImageList1--图标 ImageList。

```
Dim imgX As ListImage  
Set imgX = ImageList1.ListImages. _  
Add(, LoadPicture("icons\mail\mail01a.ico"))  
'添加一个图像到 ImageList2--小图标 ImageList。  
Set imgX = ImageList2.ListImages. _  
Add(, LoadPicture("bitmaps\assorted\w.bmp"))
```

'若要将 ImageList 控件和 ListView 控件一起使用，必须

'将 Icons 和 SmallIcons 属性

'与特定的 ImageList 控件关联。

```
ListView1.Icons = ImageList1
```



ListView1.SmallIcons = ImageList2

'用 View 选项标注 OptionButton 控件。

Option1(0).Caption = "Icon"

Option1(1).Caption = "SmallIcon"

Option1(2).Caption = "List"

Option1(3).Caption = "Report"

ListView1.View = lvwIcon '设置为图标视图

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'为添加 ListItem 对象创建变量。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 SubItem(1) 使用 Address 字段。

'ListItem 对象的 SubItem(2) 使用 Phone 字段

While Not myRs.EOF

```
Set itmX = ListView1.ListItems.Add(, CStr(myRs!Name))
```

```
itmX.Icon = 1    '设置 ImageList1 中的一个图标。
```

```
itmX.SmallIcon = 1    '设置 ImageList2 中的一个图标。
```

```
'若 Address 字段不为空，则设置 SubItem 1 为此字段。
```

```
If Not IsNull(myRs!Address) Then
```

```
    itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。
```

```
End If
```

```
'若 Phone 字段不为空，则设置 SubItem 2 为此字段。
```

```
If Not IsNull(myRs!Telephone) Then
```

```
    itmX.SubItems(2) = myRs!Telephone 'Phone 字段。
```

```
End If
```

```
myRs.MoveNext    '移动到下一条记录。
```

```
Wend
```

```
End Sub
```

## ItemClick 事件（ListView 控件）

当用户复选一个项目时发生。

## 语法

```
Private Sub object_ItemCheck ([Index As Integer,] ByVal Item As  
ComctlLib.ListItem)
```

ItemCheck 事件语法有如下几部分:

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象        |
| <i>index</i>  | 一个整数，如果一个控件在控件数组中，则该整数唯一标识该控件   |
| <i>item</i>   | 返回一个对已单击的 <b>ListItem</b> 对象的引用 |

## 说明

只有当 **Checkboxes** 属性设置为 **True**，并且 **View** 属性设置为 **lvwReport**.时，才显示复选框。

# ItemClick 事件

单击 ListView 控件中 ListItem 对象时事件发生。

应用于

ListView 控件，ListItem 对象，ListItems 集合。

语法

Private Sub *object*\_ItemClick(ByVal *Item* As ListItem)

ItemClick 事件的语法包含下面部分：

| 部分              | 描述                    |
|-----------------|-----------------------|
| <i>object</i>   | 对象表达式，其值是 ListView 控件 |
| <i>listitem</i> | 被单击的 ListItem 对象      |

说明

使用此事件决定单击了哪个 ListItem。此事件在 Click 事件之前触发。鼠标单击 ListView 控件的任何部分时触发标准的 Click 事件。只有当鼠标单击

ListItem 对象的文本或图像时才触发 ItemClick 事件。

请参阅

Click 事件, Click 事件 (ActiveX 控件)。

示例

本示例将 Biblio.mdb 数据库 Publishers 表的内容植入 ListView 控件。单击 ListItem 对象时, 代码检查 Index 属性值。若此属性值小于 15 则不任何事情。若属性值大于 15, ListItem 对象将变成幻影。要试用此例, 请将 ListView 控件放置在窗体上, 并将代码粘贴到窗体的声明部分。运行示例并单击某个项目。

```
Private ListView1_ItemClick(ByVal Item As ListItem)
    Select Case Item.Index
        Case Is = <15
            Exit Sub
        Case Is => 15
            '切换 Ghosted 属性。
            Item.Ghosted = Abs(Item.Ghosted) - 1
        End Select
    End Sub
```

Private Sub Form\_Load()

'为 ColumnHeader 对象创建对象变量。

Dim clmX As ColumnHeader

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Company", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Address", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Phone", ListView1.Width / 3)

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'创建变量以添加 ListItem 对象。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 subitem(1) 使用 Address 字段。

'ListItem 对象的 subitem(2) 使用 Phone 字段。

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, CStr(myRs!Name))

'如 Address 字段不为空，则设置 subItem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'如 Phone 字段不为空，则设置 subItem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

ListView1.View = lvwReport '设置 View 属性为报表型。

End Sub

Private Sub ListView1\_ColumnClick(ByVal ColumnHeader As ColumnHeader)

```
ListView1.SortKey = ColumnHeader.Index - 1
```

```
ListView1.Sorted = True
```

```
End Sub
```

## LabelWrap 属性（ListView 控件）

返回或设置一个值，此值决定当 **ListView** 控件为图标视图时标签是否可换行。

应用于

**ListView** 控件。

语法

*object.LabelWrap* [= *boolean*]

**LabelWrap** 属性的语法包含下面部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 对象表达式，其值为 <b>ListView</b> 控件 |
| <i>boolean</i> | 指定标签是否可换行的布尔表达式，如“设置”中所描述    |



## 设置

*boolean* 的设置值为：

| 设置值   | 描述        |
|-------|-----------|
| True  | （缺省）标签可换行 |
| False | 标签不能换行    |

## 说明

在 Windows NT 中，桌面选项中控件面板设置的图标间距决定标签的长度。在 Windows 95 中，使用“显示器”控制面板中的“外观”选项卡设置标签的长度。

## 请参阅

View 属性（ListView 控件）。

## ListItem 对象、ListItems 集合

- ListItem 包含文本和相关图标（ListImage 对象）的索引，除此之外，当

它为报表视图时，**ListItem** 还包含代表子项目的字符串数组。

- **ListItems** 集合包含一个或多个 **ListItem** 对象。

语法

*listview.ListItems*

*listview.ListItems(index)*

根据标准的集合语法，上面的语法行分别引用集合和集合中的单个元素。

**ListItem** 对象，**ListItems** 集合的语法包含下面部分：

| 部分              | 描述  |
|-----------------|---|
| <i>listview</i> | 对象表达式，其值是 <b>ListView</b> 控件  |
| <i>index</i>    | 唯一标识 <b>ListItem</b> 集合成员的整数或字符串。如果为整数，它将是 <b>Index</b> 属性的值；如果为字符串，它将是 <b>Key</b> 属性的值 |

说明

**ListItem** 对象可包含文本和图片。然而，若要使用图片则必须通过 **Icons** 和 **SmallIcons** 属性引用 **ImageList** 控件。

也可使用 **Icon** 或 **SmallIcon** 属性变更图像。

下列示例说明了如何将 **ColumnHeaders** 对象和若干带有子项目的 **ListItem** 对象添加到 **ListView** 控件中。

```
Private Sub Form_Load()  
    Dim clmX As ColumnHeader  
    Dim itmX As ListItem  
    Dim i As Integer  
  
    For i = 1 To 3  
        Set clmX = ListView1.ColumnHeaders.Add()  
        clmX.Text = "Col" & i  
    Next I
```

添加 10 个具有相同图标的项目到列表中

```
    For i = 1 To 10  
        Set itmX = ListView1.ListItems.Add()  
        itmX.SmallIcon = 1  
        itmX.Text = "ListItem " & i  
        itmX.SubItems(1) = "Subitem 1"  
        itmX.SubItems(2) = "Subitem 2"  
    Next i  
End Sub
```

## 属性

Count 属性 (ActiveX 控件), Item 属性 (ActiveX 控件), Key 属性 (ActiveX 控件), Text 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Ghosted 属性, Icon, SmallIcon 属性 (ListView 对象), SubItems 属性 (ListView 控件), Selected 属性 (ActiveX 控件), ToolTipText 属性 (ActiveX 控件)。

## 方法

Clear 方法 (ActiveX 控件), Remove 方法 (ActiveX 控件), Add 方法 (ListItems 集合), CreateDragImage 方法, EnsureVisible 方法。

## 请参阅

Left, Top 属性 (ActiveX 控件), ImageList 控件, ListImage 对象, ListImages 集合, ColumnHeader 对象, ColumnHeaders 集合, Icons, SmallIcons 属性。

## 示例

下面的示例使用 Biblio.mdb 数据库作为资源, 通过 ListItem 对象移居 ListView 控件。要试用此例, 请将一个 ListView 控件放置在窗体上, 并将代

码粘贴到窗体的声明部分。

还必须确保已将 **Biblio.mdb** 安装在机器上。在以下代码中检查 **OpenDatabase** 函数中的路径，并改变它，使之反映 **Biblio.mdb** 在机器上的实际路径。

**注意：**除非添加到 Microsoft DAO 3.5 对象库的“引用”，否则示例无法运行。为此，在 Project 菜单上单击 References。搜索 Microsoft DAO 3.5 对象库并单击复选框来选择。

```
Private Sub Form_Load()
```

```
'添加 ColumnHeaders。列宽度等于控件宽度
```

```
'除以 ColumnHeader 对象的数目。
```

```
ListView1.ColumnHeaders._
```

```
Add , , "Author", ListView1.Width / 3)
```

```
ListView1.ColumnHeaders._
```

```
Add , , "Author ID", ListView1.Width / 3, _
```

```
lvwColumnCenter
```

```
ListView1.ColumnHeaders._
```

```
Add , , "Birthdate", ListView1.Width / 3)
```

```
' Set View property to Report.
```

```
ListView1.View = lvwReport
```

'为数据访问对象声明对象变量。

```
Dim myDb As Database, myRs As Recordset
```

'设置 Database 为 BIBLIO.MDB 数据库。

' IMPORTANT: the Biblio.mdb must be on your

' machine, and you must set the correct path to

' the file in the OpenDatabase function below.

```
Set myDb = DBEngine.Workspaces(0) _  
    .OpenDatabase("c:\Program Files\VB\BIBLIO.MDB")
```

'设置 recordset 为 "Authors" 表。

```
Set myRs = _
```

```
myDb.OpenRecordset("Authors", dbOpenDynaset)
```

'声明变量以添加 ListItem 对象。

```
Dim itmX As ListItem
```

'若当前记录不是最后一条记录，则添加一个 ListItem 对象。

'ListItem 对象的文本使用 author 字段。

'ListItem 对象的 SubItem(1) 使用 AuthorID 字段。

'ListItem 对象的 SubItem(2) 使用 "Year of Birth" 字段。

```
While Not myRs.EOF
```

```
    Set itmX = ListView1.ListItems.
```

```
    Add(, CStr(myRs!Author),1) 'Author 字段。
```

'若 AuthorID 字段不为空，则将 SubItem 1 设置为此字段。

```
If Not IsNull(myRs!Au_id) Then
```

```
    itmX.SubItems(1) = CStr(myRs!Au_id)
```

```
End If
```

'若 birth 字段不为空，则将 SubItem 2 设置为此字段。

```
If Not IsNull(myRs![Year Born]) Then
```

```
    itmX.SubItems(2) = myRs![Year Born]
```

```
End If
```

```
myRs.MoveNext    '移动到下一条记录。
```

```
Wend
```

```
End Sub
```

## ListItems 属性（ListView 控件）

返回 ListView 控件中 ListItem 对象集合的引用。

应用于

ListView 控件。

## 语法

*object*.ListItem

*object* 置换元代表对象表达式，其值是 ListView 控件。

## 说明

可使用标准集合方法操作 ListItem 对象。集合中每个 ListItem 可通过其唯一关键字访问，此关键字在 Key 属性中创建并存储。

也可使用 Index 属性根据 ListItem 对象的显示位置检索 ListItem 对象。

## 请参阅

ListItem 对象，ListItem 集合，Icons, SmallIcons 属性。

## 示例

下面的示例使用 Biblio.mdb 数据库作为资源，通过 ListItem 对象移居 ListView 控件。要试用此例，请将一个 ListView 控件放置在窗体上，并将代码粘贴到窗体的声明部分。



还必须确保已将 **Biblio.mdb** 安装在机器上。在以下代码中检查 **OpenDatabase** 函数中的路径，并改变它，使之反映 **Biblio.mdb** 在机器上的实际路径。

注意：除非添加到 Microsoft DAO 3.5 对象库的“引用”，否则示例无法运行。为此，在 **Project** 菜单上单击 **References**。搜索 Microsoft DAO 3.5 对象库并单击复选框来选择。

Private Sub Form\_Load()

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

ListView1.ColumnHeaders. \_

Add , , "Author", ListView1.Width / 3)

ListView1.ColumnHeaders. \_

Add , , "Author ID", ListView1.Width / 3, \_

lvwColumnCenter

ListView1.ColumnHeaders. \_

Add , , "Birthdate", ListView1.Width / 3)

' Set View property to Report.

ListView1.View = lvwReport

'为数据访问对象声明对象变量。

Dim myDb As Database, myRs As Recordset

'设置 Database 为 BIBLIO.MDB 数据库。

' IMPORTANT: the Biblio.mdb must be on your

' machine, and you must set the correct path to

' the file in the OpenDatabase function below.

```
Set myDb = DBEngine.Workspaces(0) _  
    .OpenDatabase("c:\Program Files\VB\BIBLIO.MDB")
```

'设置 recordset 为 "Authors" 表。

```
Set myRs = _  
myDb.OpenRecordset("Authors", dbOpenDynaset)
```

'声明变量以添加 ListItem 对象。

```
Dim itmX As ListItem
```

'若当前记录不是最后一条记录，则添加一个 ListItem 对象。

'ListItem 对象的文本使用 author 字段。

'ListItem 对象的 SubItem(1) 使用 AuthorID 字段。

'ListItem 对象的 SubItem(2) 使用 "Year of Birth" 字段。

```
While Not myRs.EOF
```

```
    Set itmX = ListView1.ListItems. _
```

```
    Add(, , CStr(myRs!Author),1) 'Author 字段。
```

'若 AuthorID 字段不为空，则将 SubItem 1 设置为此字段。

```
If Not IsNull(myRs!Au_id) Then
```

```
itmX.SubItems(1) = CStr(myRs!Au_id)
End If

'若 birth 字段不为空，则将 SubItem 2 设置为此字段。
If Not IsNull(myRs![Year Born]) Then
    itmX.SubItems(2) = myRs![Year Born]
End If
myRs.MoveNext    '移动到下一条记录。
Wend
End Sub
```

## ListSubItem 对象

ListSubItem 对象代表 ListView 控件的一个子项。

语法

ListSubItem

## 说明

**ListSubItem** 对象替代了以前版本的 **ListView** 控件中的 **SubItems** 字符串数组。

## 说明

**Bold** 属性, **ReportIcon** 属性, **ToolTipText** 属性 (ActiveX 控件), **Text** 属性 (ActiveX 控件), **BackColor**, **ForeColor** 属性 (ActiveX 控件)。

## 示例

该示例创建了二十个 **ListItem** 对象, 且每个 **ListItem** 对象又创建了四个 **ListSubItem** 对象。然而, 在创建 **ListSubItem** 对象前, 代码首先创建了五个 **ColumnHeader** 对象, 其中一个是为 **ListItem** 对象创建的, 另外四个是为 **ListSubItem** 对象创建的。为试验该示例, 在窗体中放置一个 **ListView** 控件, 然后将这段代码粘贴在代码的声明部分。

Option Explicit

Private Sub Form\_Load()

Dim i As Integer ' 计数器

Dim j As Integer ' ListSubItems 计数器

```
Dim sngWidth As Single
```

```
Dim si As ListSubItem
```

```
Dim li As ListItem
```

```
'看不见 ColumnHeaders 或 ListSubitems
```

```
'除非视图设置为 lvwReport。
```

```
ListView1.View = lvwReport
```

```
' 计算 ColumnHeader 对象的宽度。
```

```
sngWidth = ListView1.Width / 5
```

```
'创建五个 ColumnHeader 对象。
```

```
For i = 1 To 5
```

```
    ListView1.ColumnHeaders.Add Text:="Col " & i, Width:=sngWidth
```

```
Next i
```

```
'创建二十个 ListItem 对象，对每个 ListItem，创建四个
```

```
' ListSubItem 对象。设置每个对象的前景色为红色。
```

```
For i = 1 To 20
```

```
    Set li = ListView1.ListItems.Add(Text:="Item " & i)
```

```
    For j = 1 To 4
```

```
        Set si = li.ListSubItems.Add(Text:="Subitem " & j)
```

```
        si.ForeColor = vbRed
```

```
    Next j
```

```
Next i  
End Sub
```

## ListSubItems 集合

ListSubItem 对象的集合。

### 语法

ListSubItems

### 属性

**Index** 属性（ActiveX 控件），**Count** 属性（ActiveX 控件），**Item** 属性（ActiveX 控件），**Key** 属性（ActiveX 控件）。

### 方法

**Add** 方法（ListSubItems 集合），**Remove** 方法（ActiveX 控件），**Clear** 方法（ActiveX 控件）。

## 说明

和 `ListItem` 对象一样，`ListSubItem` 对象只能在运行时用 `Add` 方法创建。使用 `ListSubItems` 属性返回对该集合的引用。

## ListSubItems 属性

返回对 `ListSubItem` 对象的 `ListSubItems` 集合的一个引用。

## 应用于

`ListItem` 对象，`ListItems` 集合。

## 语法

*object*.ListSubItems

*object* 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

# MultiSelect 属性（ListView、TabStrip 控件）

返回或设置一个值，此值指示用户是否可以选择多个对象或项目。

应用于

ListView 控件，TabStrip 控件。

语法

*object*.MultiSelect [= *boolean*]

MultiSelect 属性的语法包含下面部分：

| 部分             | 描述                   |
|----------------|----------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一项 |
| <i>boolean</i> | 指定选择类型的值，如“设置”中所描述   |

设置

*boolean* 的设置值为：



| 常量    | 描述                |
|-------|-------------------|
| False | (缺省) 不允许选择多个对象或项目 |
| True  | 允许多个选择            |

## 说明

对于 **ListView** 控件：按下 **SHIFT** 键并单击鼠标，或按下 **SHIFT** 键和方向键（**UP** 键、**DOWN** 键、**LEFT** 键和 **RIGHT** 键）之一，可将选择从先前选定的 **Listitem** 延伸到当前的 **Listitem**。按 **CTRL** 键并单击鼠标将选定或者撤消选定列表中的一个 **Listitem**。

对于 **TabStrip** 控件：当 **Style** 属性被设为 **TabButton** 时，只能选择 **Tab** 对象。

## 请参阅

**DeselectAll** 方法，**ColumnHeader** 对象，**ColumnHeaders** 集合，**Listitem** 对象，**ListItems** 集合，**Ghosted** 属性。

## 示例

本示例将 **Biblio.mdb** 数据库 **Authors** 表的内容植入 **ListView** 控件并允许使用 **OptionButton** 控件设置 **MultiSelect** 属性选项。可以只选择任何一个项

目，也可按下 **SHIFT** 键选择多个项目。单击 **CommandButton** 设置选定项目的 **Ghosted** 属性为 **True**。要试用此例，请将一个包含两个 **OptionButton** 控件的控件数组、一个 **ListView** 控件、一个 **ImageList** 控件和一个 **CommandButton** 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

**注意：**必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0 对象库的引用，否则示例无法运行。运行此例，单击 **OptionButton** 选择 **MultiSelect** 选项，单击项目选定它们后，再次单击 **CommandButton** 将使它们成为幻影。

```
Private Sub Command1_Click()  
    Dim x As Object  
    Dim i As Integer  
    '使选定的项目成为幻影。  
    If ListView1.SelectedItem Is Nothing Then Exit Sub  
    For i = 1 To ListView1.ListItems.Count  
        If ListView1.ListItems(i).Selected = True Then  
            ListView1.ListItems(i).Ghosted = True  
        End If  
    Next i  
End Sub  
  
Private Sub Form_Load()
```

'为 ColumnHeader 对象创建对象变量。

```
Dim clmX As ColumnHeader
```

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

```
Set clmX = ListView1.ColumnHeaders. _
```

```
Add(, "Company", ListView1.Width / 3)
```

```
Set clmX = ListView1.ColumnHeaders. _
```

```
Add(, "Address", ListView1.Width / 3)
```

```
Set clmX = ListView1.ColumnHeaders. _
```

```
Add(, "Phone", ListView1.Width / 3)
```

'用 MultiSelect 选项标注 OptionButton 控件。

```
Option1(0).Caption = "No MultiSelect"
```

```
Option1(1).Caption = "MultiSelect"
```

```
ListView1.MultiSelect = 1 '设置 MultiSelect 为 True
```

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表。

'添加一个图像到 ImageList 控件中。

```
Dim imgX As ListImage
```

```
Set imgX = ImageList1.ListImages. _
```

```
Add(, LoadPicture("icons\mail\mail01a.ico"))
```

```
ListView1.Icons = ImageList1
```

'为数据访问对象创建对象变量。

```
Dim myDb As Database, myRs As Recordset
```

'设置数据库为 BIBLIO.MDB 数据库。

```
Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
```

'设置 recordset 为 Publishers 表。

```
Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)
```

'创建变量以添加 ListItem 对象。

```
Dim itmX As ListItem
```

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 SubItem(1) 使用 Address 字段。

'ListItem 对象的 SubItem(2) 使用 Phone 字段。

```
While Not myRs.EOF
```

```
    Set itmX = ListView1.ListItems.Add(, CStr(myRs!Name))
```

```
    itmX.Icon = 1 '设置图标为 ImageList 图标。
```

'若 Address 字段不为空，设置 SubItem 1 为此字段。

```
    If Not IsNull(myRs!Address) Then
```

```
        itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。
```

```
    End If
```

'若 Phone 字段不为空，则设置 SubItem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

ListView1.View = lvwIcon '显示图标视图。

Command1.Caption = "Cut" '设置 CommandButton 的标题。

'为窗体添加标题。

Me.Caption = "Select any item(s) and click 'Cut'."

End Sub

Private Sub Option1\_Click(Index as Integer)

ListView1.MultiSelect = Index

End Sub

## PictureAlignment 属性

返回或设置一个值，确定对象中图片的对齐方式。

应用于

ListView 控件。

语 法

*object*.PictureAlignment [= *integer*]

PictureAlignment 属性语法有如下几部分:

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>integer</i> | 一个数值表达式，确定图片的对齐方式，如设置值所描述的 |

设置

*integer* 设置如下:

| 常量             | 数值 | 描述    |
|----------------|----|-------|
| LvwTopLeft     | 0  | 左顶部对齐 |
| LvwTopRight    | 1  | 右顶部对齐 |
| LvwBottomLeft  | 2  | 左底部对齐 |
| LvwBottomRight | 3  | 右底部对齐 |

续表

| 常量        | 数值 | 描述      |
|-----------|----|---------|
| LvwCenter | 4  | 居中      |
| LvwTile   | 5  | (缺省) 平铺 |

Position 属性（ColumnHeader 对象）

返回或设置对象的位置。

应用于

ColumnHeader 对象，ColumnHeaders 集合。

语法

*object*.Position [= *integer*]

Position 属性语法有如下几部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |

续表

| 部分             | 描述  |
|----------------|---|
| <i>integer</i> | 一个数值表达式，用来指定 <b>ColumnHeader</b> 对象的位置，数值可以是 1 至 <i>n</i> 的任意整数，其中 <i>n</i> 为 <b>ColumnHeader</b> 对象的号数 |

说明

使用该属性重新排列列的顺序。

### ReportIcon 属性

返回或设置一个值，指明当 **View** 属性被设置为 **lvwReport** 时 **ColumnHeader** 对象使用的图像。

应用于

**ListSubItem** 对象。



## 语法

*object*.ReportIcon [= *index*]

ReportIcon 属性的语法有如下几部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象                         |
| <i>index</i>  | 一个相关 ImageList 控件中的 ListImage 对象的 Key 或 Index 属性 |

## 说明

ColumnHeader 对象与 ColumnHeaderIcons 属性所命名的 ListImage 控件相关联。

## Sorted 属性（ListView 控件）

返回或设置确定集合中的项目是否排序的值。

应用于

ListView 控件。

语法

*object*.Sorted [= *boolean*]

Sorted 属性的语法包含下面部分：

| 部分             | 描述                                 |
|----------------|------------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的对象               |
| <i>boolean</i> | 指定 ListItem 对象是否排序的布尔表达式，如“设置”中所描述 |

设置

*boolean* 的设置值为：

| 设置值   | 描述   |
|-------|--|
| True  | 项目按字母顺序排序。对于 ListView 控件，根据 SortOrder 属性，项目按字母顺序排序 |
| False | 列表项目不排序  |

## 说明

对于 `ListView` 控件，欲使 `SortOrder` 和 `SortKey` 属性的设置值生效，必须将 `Sorted` 属性设置为 `True`。

每当 `ListItem` 坐标变更时 `Sorted` 属性将设置为 `False`。

## 请参阅

`SortKey` 属性（`ListView` 控件），`SortOrder` 属性（`ListView` 控件）。

## `SortKey` 属性（`ListView` 控件）

返回或设置一个值，此值决定 `ListView` 控件中的 `ListItem` 对象如何排序。

## 应用于

`ListView` 控件，`ListItem` 对象，`ListItems` 集合。

# 语法

```
object.SortKey [= integer]
```

SortKey 属性的语法包含下面部分：

| 部分             | 描述                    |
|----------------|-----------------------|
| <i>object</i>  | 对象表达式，其值是 ListView 控件 |
| <i>integer</i> | 指定排序关键字的整数，如“设置”中所描述  |

# 设置

*integer* 的设置值为：

| 设置值 | 描述                        |
|-----|---------------------------|
| 0   | 使用 ListItem 对象的 Text 属性排序 |
| ≥1  | 使用子项目排序，子项目的集合索引在此指定      |

# 说明：

在改变发生前，Sorted 属性必须设置为 True。

通常希望单击列标头时将列表排序。所以，ColumnClick 事件通常包括

**SortKey** 属性并使用单击的列将列表排序，下面的示例演示了这一点：

```
Private Sub ListView1_ColumnClick (ByVal ColumnHeader as ColumnHeader)
    ListView1.SortKey=ColumnHeader.Index-1
End Sub
```

请参阅

**Text** 属性（**ActiveX** 控件），**ColumnClick** 事件，**SortOrder** 属性（**ListView** 控件），**Sorted** 属性（**ListView** 控件）。

示例

本示例将三个 **ColumnHeader** 对象添加到 **ListView** 控件中，并将 **Biblio.mdb** 数据库的 **Publishers** 记录植入控件。一个由两个 **OptionButton** 控件组成的数组提供将记录排序的两个选项。单击 **ColumnHeader** 将根据由 **OptionButtons** 决定的 **SortOrder** 属性将 **ListView** 控件排序。要试用此例，请将一个 **ListView** 和一个由 **OptionButton** 控件组成的控件数组放置在窗体上，并将代码粘贴到窗体的声明部分。运行示例并单击 **ColumnHeaders** 进行排序，单击 **OptionButton** 来切换 **SortOrder** 属性。

**注意：**必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.5 对象库的引用，否则示例无法运行。

Private Sub Option1\_Click(Index as Integer)

'这些 OptionButtons 提供两种选择：升序（索引 0），

'和降序(索引 1)。请单击其一

'为 ListView 控件设置 SortOrder。

ListView1.SortOrder = Index

ListView1.Sorted = True '将列表排序。

End Sub

Private Sub Form\_Load()

'为 ColumnHeader 对象创建对象变量。

Dim clmX As ColumnHeader

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Company", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Address", ListView1.Width / 3)

Set clmX = ListView1.ColumnHeaders. \_

Add(, "Phone", ListView1.Width / 3)

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表型。

'用 sortOrder 选项标注 OptionButton 控件。

Option1(0).Caption = "Ascending (A-Z)"

Option1(1).Caption = "Descending (Z-A)"

ListView1.SortOrder = lvwAscending ' Sort ascending.

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'为添加 ListItem 对象创建变量。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 subitem(1) 使用 Address 字段。

'ListItem 对象的 subitem(2) 使用 Phone 字段。

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

'若 Address 字段不为空，则设置 subitem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'若 Phone 字段不为空，则设置 subitem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

End Sub

Private Sub ListView1\_ColumnClick(ByVal ColumnHeader As ColumnHeader)

'单击 ColumnHeader 对象时，将根据

'那一列的子项目把 ListView 控件排序。

'设置 SortKey 为 ColumnHeader 的索引值减 1

ListView1.SortKey = ColumnHeader.Index - 1

'设置 Sorted 为 True 以将列表排序。

ListView1.Sorted = True

End Sub



# SortOrder 属性（ListView 控件）

返回或设置一个值，此值决定 ListView 控件中的 ListItem 对象以升序或降序排序。

应用于

ListView 控件，ListItem 对象，ListItems 集合。

语法

*object.SortOrder* [= *integer*]

SortOrder 属性的语法包含下面部分：

| 部分             | 描述                    |
|----------------|-----------------------|
| <i>object</i>  | 对象表达式，其值是 ListView 控件 |
| <i>integer</i> | 指定排序类型的整数，如“设置”中所描述   |

设置

*integer* 的设置值为：

| 常量            | 值 | 描述   |
|---------------|---|--|
| LvwAscending  | 0 | （缺省）升序。从字母表首部（A-Z）或最早的日期开始排序。数字以字符串的形式排序，第一个数位决定排序的起始位置，其后的数位决定子排序 |
| LvwDescending | 1 | 降序。从字母表尾部（Z-A）或最晚的日期开始排序。数字以字符串的形式排序，第一个数位决定排序的起始位置，其后的数位决定子排序     |

### 说明

根据 `SortOrder` 指定的顺序将列表排序之前，`Sorted` 属性必须设置为 `True`。

### 请参阅

`Sortkey` 属性（`ListView` 控件），`ListView` 控件常量，`Sorted` 属性（`ListView` 控件）。

### 示例

本示例将三个 `ColumnHeader` 对象添加到 `ListView` 控件中，并将

Biblio.mdb 数据库的 Publishers 记录植入控件。一个由两个 **OptionButton** 控件组成的数组提供将记录排序的两个选项。单击 **ColumnHeader** 将根据由 **OptionButtons** 决定的 **SortOrder** 属性将 **ListView** 控件排序。要试用此例，请将一个 **ListView** 和一个由 **OptionButton** 控件组成的控件数组放置在窗体上，并将代码粘贴到窗体的声明部分。运行示例并单击 **ColumnHeaders** 进行排序，单击 **OptionButton** 来切换 **SortOrder** 属性。

注意：必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.5 对象库的引用，否则示例无法运行。

```
Private Sub Option1_Click(Index as Integer)
```

```
'这些 OptionButtons 提供两种选择：升序（索引 0），
```

```
'和降序(索引 1)。请单击其一
```

```
'为 ListView 控件设置 SortOrder。
```

```
ListView1.SortOrder = Index
```

```
ListView1.Sorted = True '将列表排序。
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
'为 ColumnHeader 对象创建对象变量。
```

```
Dim clmX As ColumnHeader
```

```
'添加 ColumnHeaders。列宽度等于控件宽度
```

```
'除以 ColumnHeader 对象的数目。
```

```
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Company", ListView1.Width / 3)  
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Address", ListView1.Width / 3)  
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Phone", ListView1.Width / 3)
```

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表型。

'用 SortOrder 选项标注 OptionButton 控件。

```
Option1(0).Caption = "Ascending (A-Z)"
```

```
Option1(1).Caption = "Descending (Z-A)"
```

```
ListView1.SortOrder = lvwAscending ' Sort ascending.
```

'为数据访问对象创建对象变量。

```
Dim myDb As Database, myRs As Recordset
```

'设置数据库为 BIBLIO.MDB 数据库。

```
Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")
```

'设置 recordset 为 Publishers 表。

```
Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)
```

'为添加 ListItem 对象创建变量。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 subitem(1) 使用 Address 字段。

'ListItem 对象的 subitem(2) 使用 Phone 字段。

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, CStr(myRs!Name))

'若 Address 字段不为空，则设置 subitem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'若 Phone 字段不为空，则设置 subitem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

End Sub

Private Sub ListView1\_ColumnClick(ByVal ColumnHeader As ColumnHeader)

```
'单击 ColumnHeader 对象时，将根据  
'那一列的子项目把 ListView 控件排序。  
'设置 SortKey 为 ColumnHeader 的索引值减 1  
ListView1.SortKey = ColumnHeader.Index - 1  
'设置 Sorted 为 True 以将列表排序。  
ListView1.Sorted = True
```

```
End Sub
```

## SubItemIndex 属性

返回与 ListView 控件中 ColumnHeader 对象关联的子项目的索引。

应用于

ListView 控件，ColumnHeader 对象，ColumnHeaders 集合。

语法

*object*.SubItemIndex [= *integer*]

SubItemIndex 属性的语法包含下面部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 对象表达式，其值是 <code>ColumnHeader</code> 对象        |
| <i>integer</i> | 整数，它指定与 <code>ColumnHeader</code> 对象关联的子项目的索引 |

## 说明

子项目是字符串数组，代表显示在报表视图中的 `ListItem` 对象的数据。

第一列的列标头总有一个 `SubItemIndex` 属性设置为 0，这是因为小图标和 `ListItem` 对象的文字总出现在第一列中，而且它们被当作 `ListItem` 对象而不是子项目。

列标头数目取决于子项目数目。列标头数目总是比子项目数目多 1。

## 请参阅

`ListItem` 对象，`ListItems` 集合，`SubItems` 属性（`ListView` 控件）。

## 示例

本示例添加三个 `ColumnHeader` 对象到 `ListView` 控件中。然后代码添加若干 `ListItems` 并使用 `SubItemIndex` 将 `SubItems` 字符串与正确的 `ColumnHeader` 对象关联。要试用此例，请将 `ListView` 控件放置在窗体上，并

将代码粘贴到窗体的声明部分。运行示例。

'确保 ListView 控件的 view 属性为报表视图。

```
ListView1.View = lvwReport
```

'添加三列。

```
ListView1.ColumnHeaders.Add , "Name", "Name"
```

```
ListView1.ColumnHeaders.Add , "Address", "Address"
```

```
ListView1.ColumnHeaders.Add , "Phone", "Phone"
```

'向控件添加 ListItem 对象。

```
Dim itmX As ListItem
```

'添加 column 1 的名称。

```
Set itmX= ListView1.ListItems.Add(1, "Mary", "Mary")
```

'使用 SubItemIndex 将 SubItem 与正确的

' ColumnHeader 关联。使用关键字 ("Address") 指定正确的

' ColumnHeader。

```
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) _
```

```
= "212 Grunge Street"
```

'使用 ColumnHeader 关键字将 SubItems 字符串与

'正确的 ColumnHeader 关联。

```
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) _
```

```
= "555-1212"
```



```
Set itmX = ListView1.ListItems.Add(2, "Bill", "Bill")
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) _
= "101 Pacific Way"
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) _
= "555-7879"

Set itmX= ListView1.ListItems.Add(3, "Susan", "Susan")
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) = _
"800 Chicago Street"
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) = _
"555-4537"

Set itmX= ListView1.ListItems.Add(4, "Tom", "Tom")
itmX.SubItems(ListView1.ColumnHeaders("Address").SubItemIndex) _
= "200 Ocean City"
itmX.SubItems(ListView1.ColumnHeaders("Phone").SubItemIndex) = _
"555-0348"
```

## SubItems 属性（ListView 控件）

返回或设置一个字符串（子项目）数组，它代表 ListView 控件中 ListItem

对象的数据。

应用于

ListView 控件，ListItem 对象，ListItems 集合。

语法

*object*.SubItems(index) [= *string*]

SubItems 属性的语法包含下面部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 对象表达式，其值是 ListItem 对象 |
| <i>index</i>  | 标识指定的 ListItem 子项目的整数 |
| <i>string</i> | 描述子项目的文字              |

说明

子项目是字符串数组，代表显示在报表视图中 ListItem 对象的数据。例如，它可以显示文件大小及文件的最后修改日期。

ListItem 对象可包含任意数目的关联项目数据字符串（子项目），但每个 ListItem 对象子项目数目必须相同。

每个子项目都定义了相关的列标头。

无法直接向子项目数组添加元素。使用 `ColumnHeaders` 集合中的 `Add` 方法添加子项目。

请参阅

`ColumnHeader` 对象，`ColumnHeaders` 集合，`Add` 方法（`ColumnHeaders` 集合）。

示例

下面的示例使用 `Biblio.mdb` 数据库作为资源，通过 `Listitem` 对象移居 `ListView` 控件。要试用此例，请将一个 `ListView` 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

还必须确保已将 `Biblio.mdb` 安装在机器上。在以下代码中检查 `OpenDatabase` 函数中的路径，并改变它，使之反映 `Biblio.mdb` 在机器上的实际路径。

**注意：**除非添加到 Microsoft DAO 3.5 对象库的“引用”，否则示例无法运行。为此，在 `Project` 菜单上单击 `References`。搜索 Microsoft DAO 3.5 对象库并单击复选框来选择。

Private Sub Form\_Load()

'添加 ColumnHeaders。列宽度等于控件宽度

'除以 ColumnHeader 对象的数目。

ListView1.ColumnHeaders. \_

Add , , "Author", ListView1.Width / 3)

ListView1.ColumnHeaders. \_

Add , , "Author ID", ListView1.Width / 3, \_

lvwColumnCenter

ListView1.ColumnHeaders. \_

Add , , "Birthdate", ListView1.Width / 3)

' Set View property to Report.

ListView1.View = lvwReport

'为数据访问对象声明对象变量。

Dim myDb As Database, myRs As Recordset

'设置 Database 为 BIBLIO.MDB 数据库。

' IMPORTANT: the Biblio.mdb must be on your

' machine, and you must set the correct path to

' the file in the OpenDatabase function below.

Set myDb = DBEngine.Workspaces(0) \_

.OpenDatabase("c:\Program Files\VB\BIBLIO.MDB")

'设置 recordset 为 "Authors" 表。

```
Set myRs = _  
myDb.OpenRecordset("Authors", dbOpenDynaset)
```

'声明变量以添加 ListItem 对象。

```
Dim itmX As ListItem
```

'若当前记录不是最后一条记录，则添加一个 ListItem 对象。

'ListItem 对象的文本使用 author 字段。

'ListItem 对象的 SubItem(1) 使用 AuthorID 字段。

'ListItem 对象的 SubItem(2) 使用 "Year of Birth" 字段。

```
While Not myRs.EOF
```

```
    Set itmX = ListView1.ListItems. _
```

```
    Add(, , CStr(myRs!Author),1)    'Author 字段。
```

'若 AuthorID 字段不为空，则将 SubItem 1 设置为此字段。

```
If Not IsNull(myRs!Au_id) Then
```

```
    itmX.SubItems(1) = CStr(myRs!Au_id)    ' Author ID。
```

```
End If
```

'若 birth 字段不为空，则将 SubItem 2 设置为此字段。

```
If Not IsNull(myRs![Year Born]) Then
```

```
    itmX.SubItems(2) = myRs![Year Born]
```

```
End If
```

```
        myRs.MoveNext    '移动到下一条记录。  
    Wend  
End Sub
```

## TextBackground 属性

该属性返回或者设置的值决定 `Listitem` 对象的背景是透明的还是不透明的。

应用于

`ListView` 控件。

语法

*object*.TextBackground [= *integer*]

TextBackground 属性的语法包括这些部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象     |
| <i>integer</i> | 一个常量或数值，它确定了文本背景的样式，如“设置”中所示 |

## 设置

| 常量             | 值 | 描述                    |
|----------------|---|-----------------------|
| LvwTransparent | 0 | 文本的背景是透明的             |
| LvwOpaque      | 1 | 文本的背景与 BackColor 属性相同 |

## 说明

文本背景是环绕 **ListItem** 对象的文本的矩形区域。

**TextBackground** 属性被用于 **ListView** 控件显示一幅图像的时候（通过将一幅图片赋予 **Picture** 属性）。当一个 **ListItem** 对象被位于图片的上面，而属性被设置为不透明的时候，图片将不能透过文本背景显示出来。如果属性被设置为透明的，那么背景图片将透过文本显示出来。

## View 属性（ListView 控件）

返回或设置 **ListView** 控件中 **ListItem** 对象的外观。

应用于

ListView 控件，ListItem 对象，ListItems 集合。

语法

*object*.View [= *value*]

View 属性的语法包含下面部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 对象表达式，其值是 ListView 控件 |
| <i>value</i>  | 指定控件外观的整数或常量，如“设置中所描述 |

设置

*value* 的设置值为：

| 常量           | 值 | 描述                                       |
|--------------|---|--|
| LvwIcon      | 0 | （缺省）图标。每个 ListItem 对象由整幅（标准）的图标和文本标签代表   |
| LvwSmallIcon | 1 | 小图标。每个 ListItem 对象由小图标及其右侧的文本标签代表。项目水平排列 |



续表

| 常量        | 值 | 描述  |
|-----------|---|---|
| LvwList   | 2 | 列表。每个 ListItem 对象由小图标及其右侧的文本标签代表。ListItem 对象及其相关的信息在列中垂直排列  |
| LvwReport | 3 | 报表。每个 ListItem 对象显示为小图标和文本标签。可在子项目中提供关于每个 ListItem 对象的附加信息。图标、文本标签和信息显示在列中，其中最左侧一列包含小图标和文本标签。附加列显示每个项目的子项目的文本 |

说明

只有在图标视图中才可使用 LabelWrap 属性指定在显示 ListItem 对象的标签时是否可以换行。

在报表视图中，可通过设置 HideColumnHeaders 属性为 True 来隐藏列标头。也可使用 ColumnClick 事件和 Sorted、SortOrder、SortKey 属性实现当用户单击列标头时对 ListItem 对象或子项目排序的目的。还可通过拖动列标头的右边框到适当的位置来变更列的宽度。

请参阅

ColumnClick 事件，HideColumnHeaders 属性（ListView 控件），LabelWrap 属性（ListView 控件），SortKey 属性（ListView 控件），SortOrder 属性（ListView

控件），ListView 控件常量，Sorted 属性（ListView 控件）。

## 示例

本示例将 Biblio.mdb 数据库中 Publishers 表的内容植入 ListView 控件。View 属性选项标注在四个 OptionButton 控件中。还必须将两个 ImageList 控件放置在窗体上：一个包含 Icon 属性的图像，另一个包含每个 ListItem 对象 SmallIcon 属性的图像。要试用此例，请将一个 ListView、一个包含四个 OptionButton 控件的控件数组和两个 ImageList 控件放置在窗体上，并将代码粘贴到窗体的声明部分。

**注意：**必须使用“工具”菜单的“引用”命令添加 Microsoft DAO 3.0 对象库的引用，否则示例无法运行。运行示例并单击 ComboBox 控件来切换视图。

```
Private Sub Option1_Click(Index as Integer)
```

```
    '设置 ListView 控件的 View 属性为
```

```
    'Option1 的索引
```

```
    ListView1.View = Index
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    '为 ColumnHeader 对象创建对象变量。
```

```
    Dim clmX As ColumnHeader
```

'添加 ColumnHeaders。列宽度等于控件的宽度

'除以 ColumnHeader 对象的数目。

```
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Company", ListView1.Width / 3)  
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Address", ListView1.Width / 3)  
Set clmX = ListView1.ColumnHeaders. _  
Add(, "Phone", ListView1.Width / 3)
```

ListView1.BorderStyle = ccFixedSingle '设置 BorderStyle 属性。

ListView1.View = lvwReport '设置 View 属性为报表型。

'添加一个图像到 ImageList1--图标 ImageList。

```
Dim imgX As ListImage  
Set imgX = ImageList1.ListImages. _  
Add(, LoadPicture("icons\mail\mail01a.ico"))
```

'添加一个图像到 ImageList2--小图标 ImageList。

```
Set imgX = ImageList2.ListImages. _  
Add(, LoadPicture("bitmaps\assorted\w.bmp"))
```

'若要将 ImageList 控件和 ListView 控件一起使用，必须

'将 Icons 和 SmallIcons 属性

'与特定的 ImageList 控件关联。

ListView1.Icons = ImageList1

ListView1.SmallIcons = ImageList2

'用 View 选项标注 OptionButton 控件。

Option1(0).Caption = "Icon"

Option1(1).Caption = "SmallIcon"

Option1(2).Caption = "List"

Option1(3).Caption = "Report"

ListView1.View = lvwIcon '设置为图标视图

'为数据访问对象创建对象变量。

Dim myDb As Database, myRs As Recordset

'设置数据库为 BIBLIO.MDB 数据库。

Set myDb = DBEngine.Workspaces(0).OpenDatabase("BIBLIO.MDB")

'设置 recordset 为 Publishers 表。

Set myRs = myDb.OpenRecordset("Publishers", dbOpenDynaset)

'为添加 ListItem 对象创建变量。

Dim itmX As ListItem

'若当前记录不是最后一条记录，则添加 ListItem 对象。

'ListItem 对象的文本使用 Name 字段。

'ListItem 对象的 SubItem(1) 使用 Address 字段。

'ListItem 对象的 SubItem(2) 使用 Phone 字段

While Not myRs.EOF

Set itmX = ListView1.ListItems.Add(, , CStr(myRs!Name))

itmX.Icon = 1 '设置 ImageList1 中的一个图标。

itmX.SmallIcon = 1 '设置 ImageList2 中的一个图标。

'若 Address 字段不为空，则设置 SubItem 1 为此字段。

If Not IsNull(myRs!Address) Then

itmX.SubItems(1) = CStr(myRs!Address) 'Address 字段。

End If

'若 Phone 字段不为空，则设置 SubItem 2 为此字段。

If Not IsNull(myRs!Telephone) Then

itmX.SubItems(2) = myRs!Telephone 'Phone 字段。

End If

myRs.MoveNext '移动到下一条记录。

Wend

End Sub

## MAPIMessages 控件

MAPI（邮件应用程序编程接口，Messaging Application Program Interface）控件允许你创建 Visual Basic MAPI 邮件应用程序。有两种 MAPI 控件：

- MAPISession
- MAPIMessages

MAPISession 控件可以签到并建立一个 MAPI 会话，也可以从一个 MAPI 会话中签退。MAPIMessages 控件允许用户执行各种邮件系统函数。

MAPI 控件在运行时是不可见的。并且，这些控件没有事件。要使用它们，你必须指明相应的方法。

在 MAPISession 控件建立了一个会话连接之后，MAPIMessages 控件就执行各种邮件系统函数。

要使得这些控件工作，必须有 MAPI 服务。与 MAPI 兼容的电子邮件系统都提供 MAPI 服务。

**注意：**如果你想运行使用 MAPI 控件的应用程序，确保你也正确地安装了 32 位的 MAPI DLLs，否则，你连简单的 MAPI 函数如 SignOn 都不能

执行。例如，在 Windows95 上，要在 Visual Basic 中正确地使用 MAPI 函数或 MAPI 自定义控件，你必须在操作系统安装时安装 Mail 部分。

## 语法

### MAPIMessages

## 说明

使用 MAPIMessages 控件，你可以：

- 访问 Inbox 中当前的邮件。
- 编写新邮件。
- 增加和删除邮件接收者和附件。
- 发送邮件（使用或不使用用户界面）。
- 保存、拷贝和删除邮件。
- 显示 Address Book 对话框。
- 显示 Details 对话框。
- 访问附件，包括 OLE 附件。
- 在寻址时解析接收者姓名。
- 对邮件进行回函、全部回函和转发动作等。

MAPIMessages 控件的大部分属性可以分成 4 个功能区域：地址簿、文件附

件、邮件和接收者属性。文件附件、邮件和接收者属性分别由 `AttachmentIndex`, `MsgIndex` 和 `RecipIndex` 属性控制。

## 邮件缓冲区

当使用 `MAPIMessages` 控件时，你需要跟踪两个缓冲区，`compose` 缓冲区和 `read` 缓冲区。`read` 缓冲区是由从用户的 `Inbox` 取来的邮件组成的索引集合。`MsgIndex` 属性用来访问该集合中的每个邮件，索引值 0 表示第一个邮件，以后每个邮件的索引值加 1。

邮件集合是使用 `Fetch` 方法建立的。该集合包含了所有 `FetchMsgType` 类型的邮件，并根据 `FetchSorted` 属性排序。使用 `FetchUnreadOnly` 属性可以确定集合中是否包含以前已经读过的邮件。用户不能修改 `read` 缓冲区中的邮件，但可以将其拷贝到 `compose` 缓冲区进行修改。

在 `compose` 缓冲区中可以创建或编辑邮件。当 `MsgIndex` 属性设置为 -1 时，`compose` 缓冲区就是活动缓冲区。有许多邮件动作只有在 `compose` 缓冲区中才是有效的，如发送邮件，发送带对话框的邮件，保存邮件或删除接收者和附件等。

请参阅 `Object Browser` 中的对象库可以找到这些控件的属性和错误常量。



## 属性

Action 属性（MAPIMessages 控件）， AddressCaption 属性， AddressEditFieldCount 属性， AddressLabel 属性， AddressModifiable 属性， AddressResolveUI 属性， AttachmentCount 属性， AttachmentIndex 属性， AttachmentName 属性， AttachmentPathName 属性， AttachmentPosition 属性， AttachmentType 属性， FetchMsgType 属性， FetchSorted 属性， FetchUnreadOnly 属性， MsgConversationID 属性， MsgCount 属性， MsgDateReceived 属性， MsgID 属性， MsgIndex 属性， MsgNoteText 属性， MsgOrigAddress 属性， MsgOrigDisplayName 属性， MsgRead 属性， MsgReceiptRequested 属性， MsgSent 属性， MsgSubject 属性， MsgType 属性， RecipAddress 属性， RecipCount 属性， RecipDisplayName 属性， RecipIndex 属性， RecipType 属性， SessionID 属性（MAPIMessages 控件）， Name 属性， Parent 属性， Zndex 属性（ActiveX 控件）， Tag 属性（ActiveX 控件）， Object 属性（ActiveX 控件）。

## 方法

Compose 方法， Copy 方法（MAPIMessages 控件）， Delete 方法（MAPIMessages 控件）， Fetch 方法， Forward 方法， Reply 方法， ReplyAll 方法， ResolveName 方法， Save 方法， Send 方法， Show 方法（MAPIMessages 控件）。

请参阅

MAPISession 控件，MAPI 控件常量，Error 消息，MAPI 控件。

## MAPISession 控件

MAPI（邮件应用程序编程接口，Messaging Application Program Interface）控件允许你创建 Visual Basic MAPI 邮件应用程序。有两种 MAPI 控件：

- MAPISession
- MAPIMessages

MAPISession 控件可以签到并建立一个 MAPI 会话，也可以从一个 MAPI 会话中签退。MAPIMessages 控件允许用户执行各种邮件系统函数。

语法

MAPISession

## 说明

成功签到之后，**SessionID** 属性中就包含了 **MAPI** 会话句柄。当使用 **MAPISession** 控件时，必须将该句柄传递给 **MAPISession** 或错误结果。

**MAPISession** 控件在运行时是不可见的。并且，这些控件没有事件。要使用它们，你必须指明相应的方法。

要使得这些控件工作，必须有 **MAPI** 服务。与 **MAPI** 兼容的电子邮件系统都提供 **MAPI** 服务。

**注意：**如果你想运行使用 **MAPI** 控件的应用程序，确保你也正确地安装了 32 位的 **MAPI** DLL，否则，你连简单的 **MAPI** 函数如 **SignOn** 都不能执行。例如，在 Windows 95 中，要在 Visual Basic 中正确地使用 **MAPI** 函数或 **MAPI** 自定义控件，你必须在操作系统安装时安装 Mail 部分。

## 属性

**Action** 属性（**MAPISession** 控件），**DownloadMail** 属性，**LogonUI** 属性，**NewSession** 属性，**Password** 属性（**MAPISession** 控件），**SessionID** 属性（**MAPISession** 控件），**UserName** 属性，**Index** 属性（**ActiveX** 控件），**Tag** 属性（**ActiveX** 控件），**Object** 属性（**ActiveX** 控件）。

## 方法

SignOff 方法， SignOn 方法。

## 请参阅

MAPIMessages 控件， MAPI 控件常量， Error 消息， MAPI 控件。

## Action 属性（MAPIMessages 控件）

当 MAPIMessages 控件被激活时，确定完成什么动作。在设计时该属性不可用。在运行时设置该属性激活 MAPIMessages 控件。在设计时，该属性是只写的。

**注意：**Action 属性是为了与以前的 Visual Basic 版本兼容。要利用其他的功能，使用 MAPIMessages 控件中“方法”中列出的新方法。

## 应用于

MAPIMessages 控件。

# 语法

*object.Action* [=value]

Action 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个整数表达式，指定要执行的动作         |

# 说明

列表列出了 Action 属性的设置

| Action 属性设置     | 相应的方法      |
|-----------------|------------|
| MESSAGE_FETCH   | Fetch 方法   |
| MESSAGE_SENDDLG | Send 方法    |
| MESSAGE_SEND    | Send 方法    |
| MESSAGE_SAVEMSG | Save 方法    |
| MESSAGE_COPY    | Copy 方法    |
| MESSAGE_COMPOSE | Compose 方法 |
| MESSAGE_REPLY   | Reply 方法   |

续表

| Action 属性设置         | 相应的方法         |
|---------------------|---------------|
| MESSAGE_REPLYALL    | ReplyAll 方法   |
| MESSAGE_FORWARD     | Forward 方法    |
| MESSAGE_DELETE      | Delete 方法     |
| MESSAGE_SHOWADBOOK  | Show 方法       |
| MESSAGE_SHWODETAILS | Show 方法       |
| MESSAGE_RESOLVENAME | ResoveName 方法 |
| RECIPIENT_DELETE    | Delete 方法     |
| ATTACHMENT_DELETE   | Delete 方法     |

数据类型

Integer

请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

# Action 属性（MAPISession 控件）

当 MAPISession 控件被激活时，确定完成什么动作。在设计时该属性不可用。在运行时设置该属性激活 MAPIMessages 控件。在设计时，该属性是只写的。

注意：Action 属性是为了与以前的 Visual Basic 版本兼容。要利用其他的功能，使用 MAPISession 控件中“方法”中列出的新方法。

应用于

MAPISession 控件。

语法

*object*.Action [=value]

Action 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>value</i>  | 一个整数表达式，指定要执行的动作，如“设置”中所示 |

## 设置

*value* 值的设置如下:

| 常量         | 值 | 描述   |
|------------|---|--|
| MapSignOn  | 1 | 登录用户到 <b>UserName</b> 和 <b>Password</b> 属性指定的帐户中，并给底层的邮件系统提供一个会话句柄。会话句柄保存在 <b>SessionID</b> 属性中。根据 <b>NewSession</b> 属性的值，会话句柄有可能是一个新的会话或一个现存的会话 |
| MapSignOff | 2 | 结束会话，注销用户  |

## 说明

使用该属性从会话中选择签到或签退。当签到时，**SessionID** 属性中保存了会话句柄。

## 数据类型

**Integer**



请参阅

MAPISession 控件，NewSession 属性，Password 属性（MAPISession 控件），SessionID 属性（MAPISession 控件），UserName 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## AddressCaption 属性

当 Show 方法中没有提供参数 *value* 的值或该参数值设置为 False 时，该属性就指定在 Address Book 对话框顶部显示的标题。

应用于

MAPIMessages 控件。

语法

*object*.AddressCaption [=*value*]

AddressCaption 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 字符串表达式，指定地址簿对话框的标题       |

## 说明

如果该属性为 **null** 或空字符串，就使用地址簿的缺省值。

## 数据类型

String

请参阅

MAPISession 控件， MAPIMessages 控件， Show 方法（MAPIMessages 控件）， MAPI 控件常量， Error 消息， MAPI 控件。

## AddressEditFieldCount 属性

当 Show 方法中没有提供参数 *value* 的值或该参数值设置为 **False** 时，指定在地址簿对话框中给用户显示哪个编辑控件。

应用于

MAPIMessages 控件。

语法

*object*.AddressEditFieldCount [=*value*]

AddressEditFieldCount 属性的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象     |
| <i>value</i>  | 一个整数表达式，指定要显示哪个编辑控件，如“设置”中所示 |

设置

*value* 值的设置如下：

| 设置 | 描述                   |
|----|----------------------|
| 0  | 没有编辑控件。只允许浏览         |
| 1  | （缺省）在对话框中只显示 To 编辑控件 |

续表

| 设置 | 描述                        |
|----|---------------------------|
| 2  | 在对话框中显示 To 和 CC 编辑控件      |
| 3  | 在对话框中显示 To, CC 和 BCC 编辑控件 |
| 4  | 只在对话框中显示邮件系统支持的编辑控件       |

说明

例如，如果 AddressEditFieldCount 是 3，用户在地址簿中可以选择 To, CC 和 BCC 编辑控件。应调整 AddressEditFieldCount 属性的值，以便其至少等于接收收集所要求的最少编辑控件数量。

数据类型

Integer

请参阅

MAPISession 控件，MAPIMessages 控件，AddressLabel 属性，Show 方法（MAPIMessages 控件），MAPI 控件常量，Error 消息，MAPI 控件。

# AddressLabel 属性

当 Show 方法中没有提供参数 *value* 的值或该参数值设置为 False 时，指定在地址簿对话框中 To 编辑控件的外观。

应用于

MAPIMessages 控件。

语法

```
object.AddressLabel [=value]
```

AddressLabel 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指定一个地址标签        |

说明

通常忽略该属性，其包含了一个空字符串，使用缺省标签“To”。然而，当

AddressEditFieldCount 属性设置为 1 时，用户可以显式指定另一个标签（当 AddressEditFieldCount 属性设置为 1 时就可以）。

数据类型

String

请参阅

MAPISession 控件，MAPIMessages 控件，AddressEditFieldCount 属性，Show 方法（MAPIMessages 控件），MAPI 控件常量，Error 消息，MAPI 控件。

AddressModifiable 属性

指明地址簿是否可以被修改。

应用于

MAPIMessages 控件。

# 语法

*object*.AddressModifiable [=*value*]

AddressModifiable 属性的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象     |
| <i>value</i>  | 一个布尔表达式，指明地址簿是否可以修改，如“设置”中所示 |

# 设置

*value* 值的设置如下：

| 设置    | 描述                |
|-------|-------------------|
| True  | 允许用户修改他们的地址簿      |
| False | （缺省）不允许用户修改他们的地址簿 |

# 数据类型

Boolean

请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

## AddressResolveUI 属性

使用 **ResolveName** 方法时，指明在寻址时是否显示一个对话框来解析接收者姓名。

应用于

MAPIMessages 控件。

语法

*object*.AddressResolveUI [=value]

AddressResolveUI 属性的语法有如下几个部分：



| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象   |
| <i>value</i>  | 一个布尔表达式，指明是否显示对话框，如“设置”中所示 |

## 设置

*value* 值的设置如下：

| 设置    | 描述                             |
|-------|--------------------------------|
| True  | 显示一个对话框，并且显示与接收者最相匹配的名字        |
| False | （缺省）不显示对话框。如果找不到匹配的名字就显示一条错误消息 |

## 数据类型

Boolean

请参阅

MAPISession 控件，MAPIMessages 控件，ResolveName 方法，MAPI 控件常量，Error 消息，MAPI 控件。

## AttachmentCount 属性

返回与当前索引的邮件关联的所有附件。该属性在设计时是不可用的，在运行时是只读的。

应用于

MAPIMessages 控件。

语法

*object*. AttachmentCount

*object* 是在“应用于”中指定的对象表达式。

说明

缺省值是 0。AttachmentCount 的值依赖与当前索引邮件的附件数量。

数据类型

Long

请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

## AttachmentIndex 属性

设置当前索引的附件。在设计时该属性不可用。

应用于

MAPIMessages 控件。

语法

*object*.AttachmentIndex [=value]

AttachmentIndex 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个长整数表达式，指明当前的索引附件       |

## 说明

使用索引来标明某个邮件附件。该属性中的索引确定了 `AttachmentName`, `AttachmentPathName`, `AttachmentPosition` 和 `AttachmentType` 属性的值。由 `AttachmentIndex` 索引标识的附件称为当前索引附件。`AttachmentIndex` 的值从 0（缺省值）到 `AttachmentCount-1`。

要增加一个新的附件,将 `AttachmentIndex` 的值设置为大于或等于在 `composer` 缓冲区中(`MsgIndex = -1`)当前的附件计数。`AttachmentCount` 属性会自动更新以反映新增加的附件。

例如,如果当前的 `AttachmentCount` 值为 3,将 `AttachmentIndex` 的值设置为 4 将增加两个附件并将 `AttachmentCount` 属性值改为 5。

要删除一个现有的附件,在 `Delete` 方法中将 *value* 参数的值指定为 2。只有当 `MsgIndex` 属性值为 -1 时才可以增加或删除附件。

## 数据类型

Long

请参阅

MAPISession 控件， MAPIMessages 控件， AttachmentCount 属性， AttachmentName 属性， AttachmentPathName 属性， AttachmentPosition 属性， AttachmentType 属性， MAPI 控件常量， Error 消息， MAPI 控件。

## AttachmentName 属性

指定当前索引附件文件的名字。在设计时该属性不可用。该属性是只读的，除非 MsgIndex 属性值设置为-1。

应用于

MAPIMessages 控件。

语法

*object*.AttachmentName [=value]

AttachmentName 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明当前的索引附件文件名    |

## 说明

指定的文件名是当前索引邮件接收者所看到的文件名。如果 **AttachmentName** 是一个空字符串，就使用 **AttachmentPathName** 属性中的文件名。

如果附件是一个 OLE 对象，**AttachmentName** 就包含了该对象的类名，如“Microsoft Excel Worksheet”。

在 **read** 缓冲区中的附件在下一次取邮件时被删除。**AttachmentName** 的属性值由 **AttachmentIndex** 属性确定的当前索引邮件来确定。

## 数据类型

**String**

请参阅

**MAPISession** 控件，**MAPIMessages** 控件，**AttachmentIndex** 属性，**AttachmentPathName** 属性，**MsgIndex** 属性，**MAPI** 控件常量，**Error** 消息，**MAPI**

控件。

## AttachmentPathName 属性

指定当前索引附件的全路径。在设计时该属性不可用。该属性是只读的，除非 `MsgIndex` 属性值设置为-1。

应用于

MAPIMessages 控件。

语法

*object*.AttachmentPathName [=value]

AttachmentPathName 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明当前的索引附件的全路径名  |

## 说明

如果你试图发送路径名为空字符串的邮件，将产生一个错误。

在 **read** 缓冲区中的附件在下一次取邮件时被删除。在 **compose** 缓冲区中的附件需手工删除。**AttachmentName** 的属性值由 **AttachmentIndex** 属性确定的当前索引邮件来确定。

## 数据类型

**String**

## 请参阅

**MAPISession** 控件，**MAPIMessages** 控件，**AttachmentIndex** 属性，**MsgIndex** 属性，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## AttachmentPosition 属性

指定当前索引附件在邮件主体中的位置。在设计时该属性不可用。该属性是只读的，除非 **MsgIndex** 属性值设置为-1。



应用于

MAPIMessages 控件。

语法

*object*.AttachmentPosition [=value]

AttachmentPosition 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个长整数表达式，指明当前的索引附件的位置    |

说明

要确定何处安放一个附件，计算邮件主体中字符的个数，由此决定在哪个字符位置处放置附件。附件放置位置的字符计数就是 AttachmentPosition 属性值。

例如，在一个有 5 个字符的邮件主体中，将 AttachmentPosition 的值设置为 4 可以在邮件的末尾放置一个附件（邮件主体占据的位置是 0—4）。

在同一个邮件中的同一个位置上不能放置两个附件。并且，你不能在邮件

主体之外放置附件。

**AttachmentPosition** 属性值由 **AttachmentIndex** 属性确定的当前索引邮件来确定。

## 数据类型

**Long**

请参阅

**MAPISession** 控件，**MAPIMessages** 控件，**AttachmentIndex** 属性，**MsgIndex** 属性，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## AttachmentType 属性

指定当前索引附件文件类型。在设计时该属性不可用。该属性是只读的，除非 **MsgIndex** 属性值设置为-1。

应用于

MAPIMessages 控件。

语法

*object*.AttachmentType [=value]

AttachmentType 属性的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象       |
| <i>value</i>  | 一个长整数表达式，指明当前的索引附件的类型，如“设置”中所示 |

设置

*value* 值的设置如下：

| 常量      | 值 | 描述             |
|---------|---|----------------|
| MapData | 0 | 附件是一个数据文件      |
| MapEOLE | 1 | 附件是一个内嵌 OLE 对象 |
| MapSOLE | 2 | 附件是一个静态 OLE 对象 |

## 说明

`AttachmentType` 属性值由 `AttachmentIndex` 属性确定的当前索引邮件来确定。

## 数据类型

Integer

## 请参阅

`MAPISession` 控件，`MAPIMessages` 控件，`AttachmentIndex` 属性，`MsgIndex` 属性，`MAPI` 控件常量，`Error` 消息，`MAPI` 控件。

## Compose 方法

编写邮件。

应用于

MAPIMessages 控件。

语法

*object*.Compose

object 是在“应用于”中指定的对象表达式。

说明

该方法清除 compose 缓冲区中索引组件，并将 MsgIndex 属性设置为-1。

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## Copy 方法（MAPIMessages 控件）

将当前索引邮件拷贝到 compose 缓冲区。

应用于

MAPIMessages 控件。

语法

*object*.Copy

*object* 是在“应用于”中指定的对象表达式。

说明

该方法将 **MsgIndex** 属性设置为-1。

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

# Delete 方法（MAPIMessages 控件）

删除一个邮件、接收者或附件。

应用于

MAPIMessages 控件。

语法

*object.Delete [value]*

Delete 方法的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>value</i>  | 一个整数表达式，指定要删除的部件，如“设置”中所示 |

设置

value 值的设置如下：

| 常量                  | 值 | 描述  |
|---------------------|---|---|
| MapMessageDelete    | 0 | 删除当前索引邮件的所有组件，将 <b>MsgCount</b> 属性值减 1，该邮件后面的每个邮件索引值减 1。如果删除的邮件是最后一个邮件，则将 <b>MsgIndex</b> 属性值减 1          |
| MapRecipientDelete  | 1 | 删除当前索引的接收者。自动将 <b>RecipCount</b> 属性值减 1，该接收者之后的每个接收者索引值减 1。如果删除的是最后一个接收者，将 <b>RecipIndex</b> 属性值减 1       |
| MapAttachmentDelete | 2 | 删除当前索引的附件。自动将 <b>AttachmentCount</b> 属性值减 1，该附件之后的每个附件索引值减 1。如果删除的是最后一个附件，将 <b>AttachmentIndex</b> 属性值减 1 |

请参阅

MAPISession 控件， MAPIMessages 控件， AttachmentCount 属性， AttachmentIndex 属性， MsgCount 属性， MsgIndex 属性， RecipCount 属性， RecipIndex 属性， MAPI 控件常量， Error 消息， MAPI 控件。



# DownloadMail 属性

确定何时将邮件服务器上的新邮件下载给指定的用户。

应用于

MAPIMessages 控件。

语法

*object*.DownloadMail [=*value*]

DownloadMail 属性的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象   |
| <i>value</i>  | 一个布尔表达式，指定何时下载新邮件，如“设置”中所示 |

设置

*value* 值的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | （缺省）在签到过程中将所有新邮件下载给用户                       |
| False | 新邮件不立即下载到用户的 <b>Inbox</b> 中，而是由用户指定的时间间隔来下载 |

## 说明

如果想在用户签到时访问所有的邮件，可以将该属性值设置为 **True**。然而，这或许会增加登录时间。

## 数据类型

Boolean

## 请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

## Fetch 方法

从 **Inbox** 中选择的邮件创建一个邮件集合。

应用于

**MAPIMessages** 控件。

语法

*object*.Fetch

**object** 是在“应用于”中指定的对象表达式。

说明

邮件集合包含了 **Inbox** 中的所有由 **FetchMsgType** 属性指定类型的邮件。它们都按 **FetchSorted** 属性进行排序。如果 **FetchUnreadOnly** 属性设置为 **True**，则邮件集合中仅包含未读过的邮件。

**read** 缓冲区中的任何附件在后续的取邮件过程中都被删除。

请参阅

MAPISession 控件，MAPIMessages 控件，FetchMsgType 属性，FetchSorted 属性，FetchUnreadOnly 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## FetchMsgType 属性

指定邮件集合中邮件的类型。

应用于

MAPIMessages 控件。

语法

*object*.FetchMsgType [=value]

FetchMsgType 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指定邮件类型          |

## 说明

该属性决定在使用 **Fetch** 方法将哪些邮件加入到邮件集合中。如果该属性为 **null** 或空字符串，则指定为 **IPM**（**Interpersonal Message**）类型，这是缺省类型。

**注意：**除了 **IPM** 类型外，其他的邮件类型得看你安装的邮件系统。  
参考随机文档。

## 数据类型

**String**

## 请参阅

**MAPI**Session 控件，**MAPI**Messages 控件，**Fetch** 方法，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## FetchSorted 属性

指定邮件在邮件集合中的排列次序。

应用于

MAPIMessages 控件。

语法

*object*.FetchSorted [=*value*]

FetchSorted 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>value</i>  | 一个布尔表达式，指定邮件排序次序，如“设置”中所示 |

设置

*value* 值的设置如下：

| 设置    | 描述                        |
|-------|---------------------------|
| True  | 邮件按照它们收到的次序排序（先入先出）       |
| False | （缺省）邮件按用户的 Inbox 中指定的次序排序 |

## 数据类型

Boolean

请参阅

MAPISession 控件, MAPIMessages 控件, Fetch 方法, MAPI 控件常量, Error 消息, MAPI 控件。

## FetchUnreadOnly 属性

确定在邮件集合中是否包括已读过的邮件。

应用于

MAPIMessages 控件。

语法

*object*.FetchUnreadOnly [=value]

FetchUnreadOnly 属性的语法有如下几个部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象          |
| <i>value</i>  | 一个布尔表达式，指定邮件集合中是否包括读过的邮件，如“设置”中所示 |

## 设置

*value* 值的设置如下：

| 设置    | 描述                                     |
|-------|--|
| True  | （缺省）只有 FetchMsgType 指定的且未读过的邮件包含在邮件集合中 |
| False | 所有类型正确的邮件都加入到邮件集合中                     |

## 数据类型

Boolean



请参阅

MAPISession 控件，MAPIMessages 控件，FetchMsgType 属性，Fetch 方法，MAPI 控件常量，Error 消息，MAPI 控件。

## Forward 方法

转发邮件。

应用于

MAPIMessages 控件。

语法

*object*.Forward

object 是在“应用于”中指定的对象表达式。

## 说明

该方法将当前索引的邮件拷贝到 **compose** 缓冲区，在 **Subject** 一行的开头增加一个 **FW:** 编辑控件。也将 **MsgIndex** 的值设置为-1。

## 请参阅

**MsgIndex** 属性。

## LogonUI 属性

指定是否提供一个签到对话框。

## 应用于

**MAPISession** 控件。

## 语法

*object*.LogonUI [=value]

LogonUI 属性的语法有如下几个部分：

| 部分            | 描述                                 |
|---------------|------------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象           |
| <i>value</i>  | 一个布尔表达式，指明是否显示一个签到对话框，如“设置”<br>中所示 |

设置

| 设置    | 描述  |
|-------|---|
| True  | （缺省）一个对话框提示新用户输入用户名和口令（除非<br>已有一个有效的邮件会话。请参阅 NewSession 属性） |
| False | 不显示对话框  |

说明

如果你已经有了用户名和口令且不想在用户的参预下再开一个邮件会话，**False** 设置就很有用了。然而，如果没有提供足够的信息或信息无效，则产生一个错误。

## 数据类型

Boolean

请参阅

MAPISession 控件，NewSession 属性，MAPIMessages 控件，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgConversationID 属性

指明当前索引邮件的会话线程标识值。其是只读的，除非 **MsgIndex** 设置为 -1。

应用于

MAPIMessages 控件。

# 语法

*object*.MsgConversationID [=value]

MsgConversationID 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 字符串表达式，指明会话标识值           |

# 说明

使用会话线程标识从最初的邮件到后续的所有回函在内的一个邮件集合。相同的会话 ID 表明邮件都是同一个线程的。新邮件由邮件系统分配一个 ID。MsgConversationID 的值由 MsgIndex 属性指定的当前索引邮件来决定。

# 数据类型

String

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgCount 属性

返回当前邮件会话的邮件集合中的所有邮件总数。在设计时该属性不可用，在运行时是只读的。

应用于

MAPIMessages 控件。

语法

*object*.MsgCount

MsgCount 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

## 说明

使用该属性获取邮件集合中当前的邮件数量。缺省值是 0。每执行一次 **Fetch** 方法都重置该值。

## 数据类型

Long

## 请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgDateReceived 属性

返回当前索引邮件的接收日期。在设计时该属性不可用，在运行时是只读的。

应用于

MAPIMessages 控件。

语法

*object*.MsgDateReceived

MsgDateReceived 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说明

该属性的格式是 YYYY/MM/DD HH:MM。小时是按标准的 24 进制计算。  
MsgDateReceived 属性的值由邮件系统设置，由 MsgIndex 属性指定的当前索引邮件决定。

数据类型

String



请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgID 属性

返回当前索引邮件的字符串标识符。在设计时该属性不可用，在运行时是只读的。

应用于

MAPIMessages 控件。

语法

*object*.MsgID

MsgID 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

## 说明

邮件标识符字符串是由系统指定的、唯一标识一个邮件的 64 字符长的字符串。**MsgID** 属性的值由 **MsgIndex** 属性指定的当前索引邮件决定。

## 数据类型

**String**

## 请参阅

**MAPISession** 控件，**MAPIMessages** 控件，**MsgIndex** 属性，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## MsgIndex 属性

指明当前邮件的索引值。在设计时该属性不可用。

应用于

MAPIMessages 控件。

语法

*object*.MsgIndex [=value]

MsgIndex 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 长整数表达式，指定当前索引邮件的索引值      |

说明

MsgIndex 属性决定 MAPI Messages 控件的其他与邮件有关的属性。该属性值可以从-1 到 MsgCount-1。

注意：改变 MsgIndex 属性也将改变整个附件和接收者集合。

由 MsgIndex 标识的邮件称为当前索引邮件。当改变该属性时，所有其他属性也跟着改变。-1 表明该邮件正在 compose 缓冲区中——换句话说，该邮件正

要发送出去。

## 数据类型

Long

请参阅

MAPISession 控件，MAPIMessages 控件，MsgCount 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgNoteText 属性

指定邮件的文本主体。该属性在设计时是不可用的。其是只读的，除非 MsgIndex 设置为-1。

应用于

MAPIMessages 控件。

# 语法

*object*.MsgNoteText [=*value*]

MsgNoteText 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 字符串表达式，指定邮件文本            |

# 说明

该属性包含了整个邮件的文本主体部分（去除任何附件）。一个空字符串表明没有文本。

对于发来的邮件，每一段都以回车换行符（0x0d0a）结尾。对于发出的邮件，每一段都以回车键（0x0d），换行符（0x0a）或回车换行符结尾。MsgNoteText 属性的值由 MsgIndex 属性指定的当前索引邮件决定。

# 数据类型

String

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgOrigAddress 属性

返回当前索引邮件发送者的电子邮件地址。该属性在设计时是不可用的，在运行时是只读的。当你发送一个邮件时，电子邮件系统为你设置该属性值。

应用于

MAPIMessages 控件。

语法

*object*.MsgOrigAddress

MsgOrigAddress 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

## 说明

**MsgOrigAddress** 属性值由 **MsgIndex** 属性指定的当前索引邮件决定。在 **compose** 缓冲区中，该属性值是 **null**。

## 数据类型

**String**

## 请参阅

**MAPI**Session 控件，**MAPI**Messages 控件，**MsgIndex** 属性，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## MsgOrigDisplayName 属性

返回当前索引邮件发送者的名字。该属性在设计时是不可用的，在运行时是只读的。当你发送一个邮件时，电子邮件系统为你设置该属性值。

应用于

MAPIMessages 控件。

语 法

*object*.MsgOrigDisplayName

MsgOrigDisplayName 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说 明

该属性的中的名字是在邮件头中显示的发送者姓名。MsgOrigDisplayName 属性值由 MsgIndex 属性指定的当前索引邮件决定。在 compose 缓冲区中，该属性值是 null。

数据类型

String



请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgRead 属性

返回一个布尔表达式，表明邮件是否已被阅读。该属性在设计时是不可用的，在运行时是只读的。

应用于

MAPIMessages 控件。

语法

*object*.MsgRead

MsgRead 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

## 设置

MsgRead 属性的设置如下：

| 设置    | 描述           |
|-------|--------------|
| True  | 当前索引邮件已被用户阅读 |
| False | （缺省）当前邮件未被阅读 |

## 说明

MsgRead 属性值由 MsgIndex 属性指定的当前索引邮件决定。当邮件的文本或任何附件被阅读时该邮件都标记为已阅读。然而，访问邮件头并不标记邮件为已阅读。

## 数据类型

Boolean

请参阅

MAPISession 控件, MAPIMessages 控件, MsgIndex 属性, MAPI 控件常量, Error 消息, MAPI 控件。

## MsgReceiptRequested 属性

指明当前索引的邮件是否需要一个收据。该属性在设计时是不可用的。

应用于

MAPIMessages 控件。

语法

*object*.MsgReceiptRequested [=value]

MsgReceiptRequested 属性的语法有如下几个部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象    |
| <i>value</i>  | 一个布尔表达式，指明是否需要一个收据，如“设置”中所示 |

## 设置

*value* 的值设置如下：

| 设置    | 描述                  |
|-------|---------------------|
| True  | 当接收者打开邮件时给发送者返回一个收据 |
| False | （缺省）不产生收据           |

## 说明

MsgReceiptRequested 属性值由 MsgIndex 属性指定的当前索引邮件决定。

## 数据类型

Boolean

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，

Error 消息，MAPI 控件。

## MsgSent 属性

指明当前索引的邮件是否发送给了邮件服务器。该属性在设计时是不可用的，在运行时是只读的。当你发送一个邮件时，系统为你设置该属性的值。

应用于

MAPIMessages 控件。

语法

*object*.MsgSent

MsgSent 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

# 设置

MsgSent 属性的设置如下：

| 设置    | 描述               |
|-------|------------------|
| True  | 当前索引邮件已提交给邮件服务器  |
| False | 当前索引邮件尚未提交给邮件服务器 |

# 说明

MsgSent 属性值由 MsgIndex 属性指定的当前索引邮件决定。

# 数据类型

Boolean

# 请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

# MsgSubject 属性

指明当前索引邮件在邮件头中显示的主题。该属性在设计时是不可用的。其是只读的，除非 **MsgIndex** 属性值设置为-1。

应用于

MAPIMessages 控件。

语法

*object*.MsgSubject [=value]

MsgSubject 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明主题内容          |

说明

MsgSubject 属性值由 **MsgIndex** 属性指定的当前索引邮件决定。MsgSubject

属性值限制为 64 个字符长，包括 Null 字符。

## 数据类型

String

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## MsgType 属性

指明当前索引邮件的类型。该属性在设计时是不可用的。其是只读的，除非 MsgIndex 属性值设置为-1。

应用于

MAPIMessages 控件。



# 语法

*object*.MsgType [=value]

MsgType 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明邮件类型          |

# 说明

应用程序使用 **MsgType** 属性而不是 **IPM** 邮件类型。并不是所有的电子邮件系统都支持非 **IPM** 类型的邮件，因此有可能不提供该参数。

**null** 或空字符串表明是 **IPM** 邮件类型。**MsgType** 属性值由 **MsgIndex** 属性指定的当前索引邮件决定。**MsgType** 属性并非一个过滤器，不能用来隔离发送者、发送时间和满足其他条件的邮件。

# 数据类型

String

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## NewSession 属性

指明是否建立一个新的邮件会话，即使已经存在一个有效的邮件会话。

应用于

MAPIMessages 控件。

语法

*object*.NewSession [=value]

NewSession 属性的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象       |
| <i>value</i>  | 一个布尔表达式，指明是否要建立新的邮件会话，如“设置”中所示 |

## 设置

NewSession 属性的设置如下：

| 设置    | 描述                       |
|-------|--------------------------|
| True  | 不管当前是否有一个有效会话都建立一个新的邮件会话 |
| False | （缺省）使用当前用户已建立的邮件会话       |

## 数据类型

Boolean

请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

# Password 属性（MAPISession 控件）

指定与 UserName 属性关联的帐户口令。

应用于

MAPIMessages 控件。

语法

*object.Password* [=*value*]

Password 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明帐户口令          |

说明

该属性中的空字符串表明在签到对话框中应显示一个密码输入域。缺省是空字符串。

## 数据类型

**String**

请参阅

MAPISession 控件，UserName 属性，MAPIMessages 控件，MAPI 控件常量，Error 消息，MAPI 控件。

## RecipAddress 属性

指定当前索引邮件的电子邮件地址。该属性在设计时是不可用的。其是只读的，除非将 **MsgIndex** 属性设置为-1。

应用于

MAPIMessages 控件。

## 语法

*object*.RecipAddress [=*value*]

RecipAddress 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明电子邮件地址        |

## 说明

RecipAddress 属性值由 ReciptIndex 属性指定的当前索引接收者决定。

## 数据类型

String

## 请参阅

MAPISession 控件，UserName 属性，MAPIMessages 控件，MsgIndex 属性，RecipIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

# RecipCount 属性

返回当前索引邮件所有的接收者数量。该属性在设计时是不可用的，在运行时是只读的。

应用于

MAPIMessages 控件。

语法

*object*.RecipCount

RecipCount 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说明

缺省值是 0。RecipCount 属性值由 MsgIndex 属性指定的当前索引邮件决定。

## 数据类型

Long

请参阅

MAPISession 控件, MAPIMessages 控件, MsgIndex 属性, RecipIndex 属性, MAPI 控件常量, Error 消息, MAPI 控件。

## RecipDisplayName 属性

指明当前所有接收者的满足。该属性在设计时是不可用的。其是只读的，除非将 **MsgIndex** 属性设置为-1。

应用于

MAPIMessages 控件。



## 语法

*object*.RecipDisplayName [=*value*]

RecipDisplayName 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指明接收者的名字        |

## 说明

该属性中的名字是在邮件头中显示的接收者名字。**RecipDisplayName** 属性值由 **ReciptIndex** 属性指定的当前索引接收者决定。**ResolveName** 方法使用保存在该属性中的名字。

## 数据类型

String

## 请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，RecipIndex 属性，

MAPI 控件常量，Error 消息，MAPI 控件。

## RecipIndex 属性

设置当前索引接收者。该属性在设计时是不可用的。

应用于

MAPIMessages 控件。

语法

*object*.RecipIndex [=value]

RecipIndex 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个长整数表达式，指明当前接收者         |

## 说明

指定一个索引值来标识一个邮件接收者。该属性中的索引值决定 **RecipAddress**, **RecipCount**, **RecipDisplayName** 和 **RecipType** 属性的值。由 **RecipIndex** 索引标识的接收者称为当前索引接收者。**RecipIndex** 的值从 0（缺省值）到 **RecipCount**-1。当在 **read** 缓冲区中 **RecipIndex** 的属性值设置为-1 时，其他的接收者属性显示邮件发送者信息。缺省设置是 0。

要增加一个新的接收者，在 **composer** 缓冲区中将 **RecipIndex** 的值设置为大于或等于当前的接收者计数。**RecipCount** 属性会自动更新以反映新增加的接收者数量。

例如，如果当前的 **RecipCount** 值为 3，将 **RecipIndex** 的属性值设置为 4 将增加两个接收者并将 **RecipCount** 属性值改为 5。

要删除一个现有的接收者，在 **Delete** 方法中将 *value* 参数的值指定为 1。只有当 **MsgIndex** 属性值为-1 时才可以增加或删除接收者。

## 数据类型

**Long**

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，RecipAddress 属性，RecipCount 属性，RecipDisplayName 属性，RecipType 属性，Delete 方法（MAPIMessages 控件），MAPI 控件常量，Error 消息，MAPI 控件。

## RecipType 属性

指定当前索引接收者的类型。该属性在设计时是不可用的。其是只读的，除非将 MsgIndex 属性设置为-1。

应用于

MAPIMessages 控件。

语法

*object*.RecipType [=value]

RecipType 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象  |
| <i>value</i>  | 一个整数表达式，指明接收者的类型，如“设置”中所示 |

## 设置

*value* 值的设置如下：

| 常量          | 值 | 描述          |
|-------------|---|-------------|
| MapOrigList | 0 | 邮件发送者       |
| MapToList   | 1 | 接收者是主接收者    |
| MapCcList   | 2 | 接收者是拷贝接收者   |
| MapBccList  | 3 | 接收者是隐藏拷贝接收者 |

## 说明

**RecipType** 属性值依赖于 **ReciptIndex** 属性指定的当前索引接收者。你不能将接收者类型设置为 0（电子邮件系统使用 0 标识发送者）。

## 数据类型

String

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，RecipIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## Reply 方法

给消息回函。

应用于

MAPIMessages 控件。

语法

*object*.Reply

Object 置换元是一个对象表达式，其值等于“应用于”中指定的对象。

## 说明

该方法将当前索引邮件拷贝到 **compose** 缓冲区，并在 **Subject** 行的开头增加了一个 **RE:** 编辑控件。其也将 **MsgIndex** 属性设置为-1。

当前索引邮件的发送者称为所发邮件的接收者。

## 请参阅

**MAPI**Session 控件，**MAPI**Messages 控件，**MsgIndex** 属性，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## ReplyAll 方法

给所有邮件接收者回函。

## 应用于

**MAPI**Messages 控件。

## 语法

*object*.ReplyAll

Object 置换元是一个对象表达式，其值等于“应用于”中指定的对象。

## 说明

该方法将当前索引邮件拷贝到 compose 缓冲区，并在 Subject 行的开头增加了一个 RE: 编辑控件。其也将 MsgIndex 属性设置为-1。

该邮件发送给当前索引邮件的发送者以及所有 To: 和 CC: 中的接收者。

## 请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## ResolveName 方法

解析当前索引接收者的名字。



应用于

MAPIMessages 控件。

语法

*object*.ResolveName

Object 置换元是一个对象表达式，其值等于“应用于”中指定的对象。

说明

该方法搜索地址簿以便找到与当前索引接收者匹配的名字。如果没有找到，就返回一个错误信息。其不对邮件发送者名字或地址作额外的解析。

AddressResolveUI 属性确定是否显示一个对话框以便解析歧义名字。

该方法也将导致 RecipType 属性的更改。

请参阅

MAPISession 控件，MAPIMessages 控件，AddressResolveUI 属性，RecipType 属性，MAPI 控件常量，Error 消息，MAPI 控件。

## Save 方法

保存 compose 缓冲区中当前的邮件（MsgIndex = -1）。

应用于

MAPIMessages 控件。

语法

*object*.Save

Object 置换元是一个对象表达式，其值等于“应用于”中指定的对象。

请参阅

MAPISession 控件，MAPIMessages 控件，MsgIndex 属性，MAPI 控件常量，Error 消息，MAPI 控件。

# Send 方法

发送一个邮件。

应用于

MAPIMessages 控件。

语法

*object*.Save [*value*]

*Send* 方法的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象     |
| <i>value</i>  | 一个布尔表达式，指明是否显示一个对话框，如“设置”中所示 |

设置

value 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 在对话框中发送一个邮件。给用户提示各种邮件组件，并将邮件发送给邮件服务器等待转发出去。所有与 <b>compose</b> 缓冲区中创建的邮件相关联的属性组成了邮件对话框。然而，在对话框中的修改并不影响 <b>compose</b> 缓冲区中的信息 |
| False | （缺省）将邮件提交给邮件服务器，不显示对话框。如果你发送的邮件没有接收者或附件路径无效就产生一个错误   |

请参阅

**MAPI**Session 控件，**MAPI**Messages 控件，**MAPI** 控件常量，Error 消息，**MAPI** 控件。

### SessionID 属性（**MAPI**Messages 控件）

保存当前邮件会话句柄。该属性在设计时是不可用的。

应用于

**MAPI**Messages 控件。

# 语法

*object.SessionID[=value]*

SessionID 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个长整数表达式，指明当前的会话句柄       |

# 说明

该属性包含了 MAPISession 控件中 SessionID 属性返回的邮件会话句柄。要将一个 MAPISession 控件与一个有效的邮件会话句柄关联，将该属性设置为一个成功签到的 MAPISession 控件的 SessionID。

# 请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

# SessionID 属性（MAPISession 控件）

保存当前邮件会话句柄。该属性在设计时是不可用的，在运行时是只读的。

应用于

MAPISession 控件。

语法

*object.SessionID*

SessionID 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |

说明

当你使用 **SignOn** 方法时就设置该属性的值。**SessionID** 属性包含了一个唯一标识邮件会话的句柄。缺省是 0。

使用该属性设置 **MAPIMessages** 控件的 **SessionID** 属性。

## 数据类型

Long

请参阅

MAPISession 控件，SignOn 方法，MAPIMessages 控件，MAPI 控件常量，Error 消息，MAPI 控件。

## Show 方法（MAPIMessages 控件）

显示电子邮件地址簿或当前索引接收者的详细信息。

应用于

MAPIMessages 控件。

语法

*object*.Show[=*value*]

Show 方法的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象     |
| <i>value</i>  | 一个布尔表达式，指明要显示的对话框类型，如“设置”中所示 |

## 设置

*value* 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 显示当前索引接收者详细信息的对话框。在对话框中显示信息的数量由电子邮件系统决定。其至少包含接收者的名字和电子邮件地址 |
| False | （缺省）显示电子邮件地址簿。你可以使用地址簿创建或修改接收者。不保存在 compose 缓冲区之外对地址簿进行的修改 |

## 请参阅

MAPISession 控件, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。



## SignOff 方法

结束一个邮件会话，使用 **UserName** 和 **Password** 属性中的值签退。

应用于

**MAPISession** 控件。

语法

*object*. SignOff

**Object** 置换元是一个对象表达式，其值等于“应用于”中指定的对象。

请参阅

**MAPISession** 控件，**Password** 属性（**MAPISession** 控件），**UserName** 属性，**MAPIMessages** 控件，**MAPI** 控件常量，**Error** 消息，**MAPI** 控件。

## SignIn 方法

使用 **UserName** 和 **Password** 属性中的值签到，给底层的电子邮件系统提供一个邮件会话句柄。

应用于

**MAPISession** 控件。

语法

*object*. **SignIn**

**Object** 置换元是一个对象表达式，其值等于“应用于”中指定的对象。

说明

会话句柄保存在 **SessionID** 属性中。根据不同的 **NewSession** 属性值，该会话句柄指向一个新创建的会话或一个已有的会话。

请参阅

MAPISession 控件, NewSession 属性, Password 属性 (MAPISession 控件), SessionID 属性 (MAPISession 控件), UserName 属性, MAPIMessages 控件, MAPI 控件常量, Error 消息, MAPI 控件。

## UserName 属性

指定帐户的名字。

应用于

MAPISession 控件。

语法

*object*.UserName [=value]

UserName 属性的语法有如下几个部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>value</i>  | 一个字符串表达式，指定用户帐户          |

## 说明

该属性包含了签到和签退中需要的用户名。如果 **LogonUI** 属性值为 **True**，而 **UserName** 属性是一个空字符串，表明应显示一个签到对话框，并带有一个名字输入区域。缺省是空字符串。

## 数据类型

String

## 请参阅

MAPISession 控件，LogonUI 属性，MAPIMessages 控件，MAPI 控件常量，Error 消息，MAPI 控件。

# 错误消息，MAPI 控件

下表列出 MAPI 控件的可捕获的错误。

| 常量                 | 值     | 描述   |
|--------------------|-------|--|
| MapSuccessSuccess  | 32000 | 操作成功返回                                     |
| MapUserAbort       | 32001 | 处理被取消因为处理被取消，当前操作不能完成                      |
| MapFailure         | 32002 | 出现了不确定的故障当前操作出现了不确定的故障，例如，操作不能删除或不能正确对信件寻址 |
| MapLoginFail       | 32003 | 登录失败没有提供默认登录，出现登录故障                        |
| MapDiskFull        | 32004 | 磁盘已满磁盘已满，当前操作不能创建磁盘文件                      |
| MapInsufficientMem | 32005 | 内存不足没有足够内存继续当前操作                           |
| MapAccessDenied    | 32006 | 拒绝访问                                       |
| MapGeneralFailure  | 32007 | 一般故障有不确定的错误                                |
| MapTooManySessions | 32008 | 过程太多同时存在的打开过程太多                            |
| MapTooManyFiles    | 32009 | 文件太多消息包含的文件附文太多，该邮件不能发送或读取                 |

续表

| 常量                        | 值     | 描述   |
|---------------------------|-------|--|
| MapTooManyRecipients      | 32010 | 收件人太多指定的消息收件人太多，邮件不能发送或读取                            |
| MapAttachmentNotFound     | 32011 | 找不到附文找不到指定附文，邮件不能发送                                  |
| MapAttachmentOpenFailure  | 32012 | 打开附文故障附文无法定位，邮件不能发送。确定 AttachmentPathName 属性有效       |
| MapAttachmentWriteFailure | 32013 | 写附文故障附文不能写到临时文件，检查目录权限                               |
| MapUnknownRecipient       | 32014 | 不认识的收件人通讯录无此收件人。邮件不能发送                               |
| MapBadRecipType           | 32015 | 收件人类型无效收件人类型错误。有效类型值是 1（主要收件人）、2（抄送收件人）和 3（隐蔽的抄送收件人） |
| MapNoMessages             | 32016 | 无消息找不到下一条消息  |
| MapInvalidMessage         | 32017 | 无效消息使用了无效消息 ID，当前操作不能完成                              |
| MapTextTooLarge           | 32018 | 文本太长消息中的文本太长而不能发送。邮件不能发送。文本不能超过 32K                  |

续表

| 常量                   | 值     | 描述   |
|----------------------|-------|--|
| MapInvalidSession    | 32019 | 无效过程使用了无效过程 ID，要使 MAPI 信息控件与有效消息过程关联，要对 MAPI 过程控件的 SessionID 设置 SessionID 属性 |
| MapTypeNotSupported  | 32020 | 不支持的类型   |
| MapAmbiguousRecipie  | 32021 | 收件人不明确一个或多个收件人的地址无效。确定 RecipAddress 属性的地址有效                                  |
| MapMessageInUse      | 32022 | 消息在使用  |
| MapNetworkFailure    | 32023 | 网络故障   |
| MapInvalidEditFields | 32024 | 编辑字段无效 AddressEditFieldCount 属性值无效。有效值从 0 到 4                                |
| MapInvalidRecips     | 32025 | 收件人无效一个或多个收件人的地址无效。确定 RecipAddress 属性的地址有效                                   |
| MapNotSupported      | 32026 | 不支持基本邮件系统不支持当前操作   |
| mapSessionExist      | 32050 | 登录失败：有效过程 ID 已存在 MAPI 信息控件正在用有效过程 ID   |
| mapInvalidBuffer     | 32051 | 不用撰写缓冲区时，属性为只读。设置 MsgIndex = -1  |

续表

| 常量                                | 值     | 描述  |
|-----------------------------------|-------|---|
| mapInvalidReadBuffer<br>Action    | 32052 | 操作只对撰写缓冲区有效。设置<br>MsgIndex = -1                           |
| mapNoSession                      | 32053 | MAPI 失败：有效过程 ID 不存在 MAPI<br>消息控件没有来自 MAPI 过程控件的有效过<br>程句柄 |
| mapInvalidRecipient               | 32054 | 撰写缓冲区中无源点在撰写缓冲区中不<br>能查看消息源点的信息（MsgIndex 设置为 -<br>1）      |
| mapInvalidComposeBu<br>fferAction | 32055 | 操作对撰写缓冲区无效撰写缓冲区中欲<br>施操作无效（MsgIndex 设置为 -1）               |
| mapControlFailure                 | 32056 | 列表内无消息，操作不能执行   |
| mapNoRecipients                   | 32057 | 无收件人，操作不能执行   |
| mapNoAttachment                   | 32058 | 无附文，操作不能执行  |



# Multimedia MCI

Multimedia MCI 包含一组高层次的独立于设备的命令，可以控制音频和视频外设。首先发送的 MCI 命令就是 **Open**。这条命令打开规定的 MCI 设备，标识将要在设备上播放或记录的文件。有些设备，如 CDAudio、VCR 和视盘等，并不使用文件，所以无需提供文件名。

设备打开后，可以发送任何其它的 MCI 命令（如 **Prev**、**Next**、**Pause** 等）。**Close** 命令是向设备发送的最后一条 MCI 命令，它返回到可用的系统资源缓冲池，**Close** 命令还关闭与设备相关的数据文件。

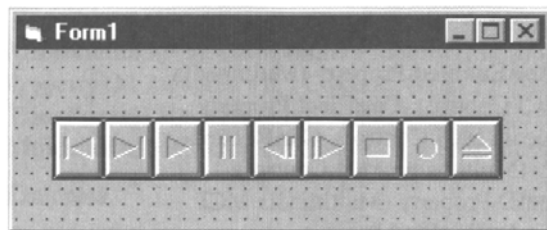
关于 Multimedia MCI 控件所支持的 MCI 命令清单，查看 **Command** 属性。关于 Multimedia MCI 的更多的信息，请参阅《Microsoft Win32 软件开发包多媒体程序员指南》。

## Multimedia MCI 控件

Multimedia MCI 控件管理 MCI（多媒体控制接口）设备上记录和播放多媒体文件。从概念上说，该控件应是一系列的按钮，给 MCI 设备发送命令，MCI 设备有声卡、MIDI 序列器、CD-ROM 驱动器、音频 CD 播放器、激光视盘、

放像机和录像机等。MCI 控件也支持播放 Video for Windows(\*.avi)格式的文件。

当你在设计时给窗体增加一个 Multimedia MCI 控件时，该控件在窗体上的外观如下：



这些按钮分别定义为 Prev, Next, Play, Pause, Back, Step, Stop, Record 和 Eject。

### 说明

在允许用户选择 Multimedia MCI 控件上的按钮时，你的应用程序应确保已经打开了 MCI 设备并且 Multimedia MCI 控件上的按钮是可用的。在 Visual Basic 中，在窗体 Form\_Load 事件中设置 MCI 的 Open 命令。

当你想使用 Multimedia MCI 控件记录声音时，要打开一个新文件。该动作可以确保所记录的声音数据文件与你的系统记录能力兼容。并且，在改变 MCI 设备之前使用 Save 命令保存记录的数据文件。

Multimedia MCI 控件有几种方式：

- 控件在运行时可视或不可视。

- 你可以给按钮定义参数或彻底重新定义按钮的功能。
- 你可以在一个窗体中控制多个设备。

如果你想使用 **Multimedia MCI** 控件中的按钮，要将 **Visible** 和 **Enabled** 属性设置为 **True**。如果你不想使用 **Multimedia MCI** 控件中的按钮，将 **Visible** 和 **Enabled** 属性设置为 **False**。应用程序可以控制 **MCI** 设备，有没有用户的参预均可。

**Multimedia MCI** 控件的事件是可编程控制的。你可以设置按钮的参数，或干脆开发事件代码重新定义按钮的功能。

**MCI** 扩展（**Extension**）在单个窗体中支持多个 **Multimedia MCI** 控件实例，可以并发控制多个 **MCI** 设备。你每次使用一个设备的 **Multimedia MCI** 控件。

**发布须知：**当你创建和发布使用 **Multimedia MCI** 控件的应用程序时，你应该在用户的 **Microsoft Windows System** 或 **System32** 目录下安装和注册相应的文件。**Visual Basic** 中提供的 **Setup Wizard** 工具可以正确地编写应用程序的安装程序。

## 属性

**AutoEnable** 属性（**Multimedia MCI** 控件），**ButtonEnabled** 属性（**Multimedia MCI** 控件），**ButtonVisible** 属性（**Multimedia MCI** 控件），**CanEject** 属性（**Multimedia MCI** 控件），**CanPlay** 属性（**Multimedia MCI** 控件），**CanRecord** 属性（**Multimedia MCI** 控件），**CanStep** 属性（**Multimedia MCI** 控件），**Command** 属性（**Multimedia MCI** 控件），**DeviceID** 属性（**Multimedia MCI** 控件），**DeviceType** 属性（**Multimedia**

MCI 控件), Enabled 属性 (Multimedia MCI 控件), Error 属性 (Multimedia MCI 控件), ErrorMessage 属性 (Multimedia MCI 控件), FileName 属性 (Multimedia MCI 控件), Frames 属性 (Multimedia MCI 控件), From 属性 (Multimedia MCI 控件), hWndDisplay 属性 (Multimedia MCI 控件), Index 属性, Length 属性 (Multimedia MCI 控件), Mode 属性 (Multimedia MCI 控件), Notify 属性 (Multimedia MCI 控件), NotifyMessage 属性 (Multimedia MCI 控件), NotifyValue 属性 (Multimedia MCI 控件), Orientation 属性 (Multimedia MCI 控件), Position 属性 (Multimedia MCI 控件), RecordMode 属性 (Multimedia MCI 控件), Shareable 属性 (Multimedia MCI 控件), Silent 属性 (Multimedia MCI 控件), Start 属性 (Multimedia MCI 控件), TimeFormat 属性 (Multimedia MCI 控件), To 属性 (Multimedia MCI 控件), Track 属性 (Multimedia MCI 控件), UpdateInterval 属性 (Multimedia MCI 控件), UsesWindows 属性 (Multimedia MCI 控件), Visible 属性 (Multimedia MCI 控件), Wait 属性 (Multimedia MCI 控件), hWnd 属性, MouseIcon 属性, TabStop 属性, ToolTipText 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方

法，OLEDrag 方法（ActiveX 控件），Refresh 方法（ActiveX 控件）。

## 事件

ButtonClick 事件（Multimedia MCI 控件），ButtonCompleted 事件（Multimedia MCI 控件），ButtonGotFocus 事件（Multimedia MCI 控件），ButtonLostFocus 事件（Multimedia MCI 控件），Done 事件（Multimedia MCI 控件），StatusUpdate（Multimedia MCI 控件），DragDrophVent，DragOver 事件，GotFocus 事件，LostFocus 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedBack 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件）。

## 请参阅

Multimedia MCI, AutoEnable 属性（Multimedia MCI 控件），ButtonEnabled 属性（Multimedia MCI 控件），Start 属性（Multimedia MCI 控件），TimeFormat 属性（Multimedia MCI 控件），使用 Multimedia MCI 控件，Multimedia MCI 控件常量，Error 消息（Multimedia MCI 控件）。

## 示例

下面的示例演示了打开一台使用兼容数据文件的 MCI 设备的过程。将这些代码放到 Form\_Load 过程，应用程序就可以使用 Multimedia MCI 控件来对文件 Gong.wav 进行 播放、记录和倒带。在试运行这个示例之前，首先应创

建一个包含 Multimedia MCI 控件的窗体。

```
Private Sub Form_Load ()  
    ' Set properties needed by MCI to open.  
    MMControl1.Notify = FALSE  
    MMControl1.Wait = TRUE  
    MMControl1.Shareable = FALSE  
    MMControl1.DeviceType = "WaveAudio"  
    MMControl1.FileName = "C:\WINDOWS\MMDATA\GONG.WAV"  
  
    '打开 MCI WaveAudio 设备。  
    MMControl1.Command = "Open"  
  
End Sub
```

为了正确管理多媒体资源，在退出应用程序之前，应该关闭那些已经打开的 MCI 设备。将下面的语句放到 Form\_Unload 过程，那么在退出包含 Multimedia MCI 控件的窗体之前，就可以关闭那些已经打开的 MCI 设备。

```
Private Sub Form_Unload (Cancel As Integer)  
    MMControl1.Command = "Close"  
  
End Sub
```

## AutoEnable 属性（Multimedia MCI 控件）

确定 Multimedia MCI 控件是否可以自动地禁止或允许使用控件中的某个按钮。如果 AutoEnable 属性设置为 True，Multimedia MCI 控件可以为指定 MCI 设备类型的当前模式允许这些按钮。该属性也将禁止 MCI 设备的当前模式不支持的按钮。

应用于

Multimedia MCI 控件。

语法

[*form.*] *MMControl*.AutoEnable [= { True | False } ]

说明

AutoEnable 属性的效果还取决于 Enable 属性。当 Multimedia MCI 控件允许使用时（Enabled 属性设置为 True），AutoEnable 属性可以自动允许使用或禁止单个的按钮。当 Enabled 属性设置为 False 时，不管 AutoEnable 属性的设置如何，总是关闭键盘和鼠标对 Multimedia MCI 控件的运行时访问。

下表列出了 Multimedia MCI 控件的 AutoEnable 属性的设置

| 设置    | 描述  |
|-------|---|
| False | 不禁止或允许使用按钮。程序通过 Enabled 和 ButtonEnabled 属性控制按钮的状态 |
| True  | （缺省）允许使用那些功能可以使用的按钮，禁止不能使用的按钮                     |

下表演示了如果反映 Multimedia MCI 控件的 MCI 模式：

Open 模式

Play 模式

Record 模式

Pause 模式

Stop 模式

Open 模式

Seek 或 Not Ready 模式

AutoEnable 属性值覆盖 *ButtonEnabled* 属性。当 Enabled 和 AutoEnabled 属性都设置为 True 时，不使用 *ButtonEnable* 属性。

数据类型

Integer(Boolean)



请参阅

Multimedia MCI, ButtonEnabled 属性（Multimedia MCI 控件），Enabled 属性（Multimedia MCI 控件）。

## Open 模式

| 按钮      | 状态 |
|---------|----|
| Back*   | 禁止 |
| Eject*  | 允许 |
| Next    | 禁止 |
| Pause   | 禁止 |
| Play*   | 禁止 |
| Prev    | 禁止 |
| Record* | 禁止 |
| Step*   | 禁止 |
| Stop    | 禁止 |

\*表示当打开的 MCI 设备相应的操作时按钮是允许使用的。

## Play 模式

| 按钮     | 状态 |
|--------|----|
| Back*  | 允许 |
| Eject* | 允许 |

续表

| 按钮      | 状态 |
|---------|----|
| Next    | 允许 |
| Pause   | 允许 |
| Play*   | 禁止 |
| Prev    | 允许 |
| Record* | 禁止 |
| Step*   | 允许 |
| Stop    | 允许 |

\*表示当打开的 MCI 设备相应的操作时按钮是允许使用的。

Record 模式

| 按钮      | 状态 |
|---------|----|
| Back*   | 允许 |
| Eject*  | 允许 |
| Next    | 允许 |
| Pause   | 允许 |
| Play*   | 禁止 |
| Prev    | 允许 |
| Record* | 禁止 |
| Step*   | 允许 |
| Stop    | 允许 |

\*表示当打开的 MCI 设备相应的操作时按钮是允许使用的。

## Pause 模式

| 按钮      | 状态 |
|---------|----|
| Back*   | 允许 |
| Eject*  | 允许 |
| Next    | 允许 |
| Pause   | 允许 |
| Play*   | 允许 |
| Prev    | 允许 |
| Record* | 允许 |
| Step*   | 允许 |
| Stop    | 允许 |

\*表示当打开的 MCI 设备相应的操作时按钮是允许使用的。

## Stop 模式

| 按钮     | 状态 |
|--------|----|
| Back*  | 允许 |
| Eject* | 允许 |
| Next   | 允许 |
| Pause  | 禁止 |

续表

| 按钮      | 状态 |
|---------|----|
| Play*   | 允许 |
| Prev    | 允许 |
| Record* | 允许 |
| Step*   | 允许 |
| Stop    | 禁止 |

\*表示当打开的 MCI 设备相应的操作时按钮是允许使用的。

Seek 或 Not Ready 模式

| 按钮      | 状态 |
|---------|----|
| Back*   | 禁止 |
| Eject*  | 禁止 |
| Next    | 禁止 |
| Pause   | 禁止 |
| Play*   | 禁止 |
| Prev    | 禁止 |
| Record* | 禁止 |
| Step*   | 禁止 |
| Stop    | 禁止 |

\*表示当打开的 MCI 设备相应的操作时按钮是允许使用的。

# ButtonClick 事件（Multimedia MCI 控件）

当用户在 Multimedia MCI 控件的按钮上按下和释放鼠标按钮时产生该事件。

应用于

Multimedia MCI 控件。

语法

Private Sub *MMControl\_ButtonClick*( *cancel* As Integer)

说明

*Button* 可以替换为下列按钮：Back, Eject, Next ,Pause, Play, Prev, Record, Step 或 Stop。

缺省地，用户选择一个按钮时每个 *ButtonClick* 事件就执行一个 MCI 命令。列表列出了控件中每个按钮执行的命令。

| 按钮    | 命令        |
|-------|-----------|
| Back  | MCI_STEP  |
| Step  | MCI_STEP  |
| Play  | MCI_PLAY  |
| Pause | MCI_PAUSE |

| 按钮     | 命令                                 |
|--------|------------------------------------|
| Prev   | MCI_SEEK                           |
| Next   | MCI_SEEK                           |
| Stop   | MCI_STOP                           |
| Record | MCI_RECORD                         |
| Eject  | 带 MCI_SET_DOOR_OPEN 参数的 MCI_SET 命令 |

将 *ButtonClick* 事件中的参数 *cancel* 设置为 **True** 将阻止执行缺省的 **MCI** 命令。*cancel* 参数的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | 阻止执行缺省的 <b>MCI</b> 命令                                   |
| False | 执行完相应的 <i>ButtonClick</i> 事件主体后执行与按钮相关的缺省 <b>MCI</b> 命令 |

在执行缺省的 **MCI** 命令之前执行事件的主体过程。给 *ButtonClick* 事件主体增加代码来提供相应的功能。如果在事件主体过程中将 *cancel* 参数或在参数传递时将参数 *cancel* 设置为 **True**，则不执行与该事件关联的缺省 **MCI** 命令。

**注意：**如果设备不支持 **MCI Resume** 命令，则给一个暂停的设备发送一个 **Pause** 命令将结束原来 **Play** 命令发送的信息。**Multimedia MCI** 控件使用 **MCI Play** 命令重新启动不支持 **MCI Resume** 命令的设备。重新启动暂停设备的 **Play** 命令将取消回调条件，并覆盖原来的 **Play** 命令信息。

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件)。

## ButtonCompleted 事件 (Multimedia MCI 控件)

当一个 Multimedia MCI 控件按钮结束时产生该 MCI 命令。

应用于

Multimedia MCI 控件。

语法

Private Sub *MMControl*\_ButtonCompleted (*Errorcode* As Long)

说明

*Button* 可以替换为下列按钮: Back, Eject, Next ,Pause, Play, Prev, Record, Step 或 Stop。

*Errorcode* 参数可以设置为:

| 设置    | 描述        |
|-------|-----------|
| 0     | 命令成功地执行完  |
| 任何其他值 | 命令没有成功执行完 |

如果在 *ButtonClick* 事件中的 *cancel* 参数设置为 **True**，则不触发 *ButtonCompleted* 事件。

请参阅

**MultimediaMCICommand** 属性（**Multimedia MCI** 控件）。

## **ButtonEnabled** 属性（**Multimedia MCI** 控件）

确定 **Multimedia MCI** 控件中的按钮是禁止还是允许使用的（禁止的按钮显示为灰色的）。

应用于

**Multimedia MCI** 控件。

语法

```
[form.]MMContro.ButtonEnabled [= {True|False}]
```

说明

*ButtonEnabled* 属性的效果还取决于 **Enabled** 和 **AutoEnable** 属性的设置。当 **Multimedia MCI** 控件是允许使用的（**Enabled** 属性设置为 **True**）以及 **AutoEnable** 属性关闭（设置为 **False**）时，单个 **ButtonEnabled** 属性将禁止或允许 **Multimedia MCI** 控件中的按钮。



对于该属性，*Button* 可以替换为 Back, Eject, Next, Pause, Play, Prev, Record, Step 或 Stop。例如，下面的代码禁止了 Play 按钮。

```
[form.]MMControl.PlayEnabled = False
```

要检查 Record 按钮是否允许使用，使用：

```
if [form.]MMControl.RecordEnabled Then ...
```

下表列出了 Multimedia MCI 控件中的 *ButtonEnabled* 属性的设置

| 设置    | 描述                                      |
|-------|---|
| False | （缺省）禁止 <i>Button</i> 指定的按钮。控件中该按钮的功能不可用 |
| True  | 允许指定的按钮。控件中该按钮的功能可用                     |

数据类型

Integer(Boolean)

请参阅

Multimedia MCI，AutoEnable 属性（Multimedia MCI 控件），Enabled 属性（Multimedia MCI 控件）。

## ButtonGotFocus 事件（Multimedia MCI 控件）

当 Multimedia MCI 控件中的按钮接收到输入角度时产生该事件。

应用于

Multimedia MCI 控件。

语法

```
Private Sub MMControl._ButtonGotFocus()
```

说明

Button 可以替换为 Back,Eject, Next, Pause, Play, Prev, Record, Step 或 Stop。

请参阅

Multimedia MCI， ButtonLostFocus 事件（Multimedia MCI 控件）。

## ButtonLostFocus 事件（Multimedia MCI 控件）

当 Multimedia MCI 控件中的按钮接失去输入角度时产生该事件。

应用于

Multimedia MCI 控件。

语法

Private Sub *MMControl\_Button*LostFocus()

说明

*Button* 可以替换为 Back,Eject, Next, Pause, Play, Prev, Record, Step 或 Stop。

请参阅

MultimediaMCI Multimedia MCI, ButtonGotFocus 事件（Multimedia MCI 控件）,CotFocus 事件，LostFocus 事件。

ButtonVisible 属性（Multimedia MCI 控件）

确定指定的按钮是否可视。

应用于

Multimedia MCI 控件。

# 语法

```
[form.]MMControl.ButtonVisible [= {True|False}]
```

# 说明

*ButtonVisible* 属性的效果还取决于 *Visible* 属性的设置。当 Multimedia MCI 控件是可视的（*Visible* 属性设置为 *True*）时，单个 *ButtonVisible* 属性将显示或隐藏 Multimedia MCI 控件中的按钮。

对于该属性，*Button* 可以替换为 *Back*, *Eject*, *Next*, *Pause*, *Play*, *Prev*, *Record*, *Step* 或 *Stop*。

下表列出了 Multimedia MCI 控件中的 *ButtonVisible* 属性的设置

| 设置    | 描述                                   |
|-------|--------------------------------------|
| False | 不显示 <i>Button</i> 指定的按钮。控件中该按钮的功能不可用 |
| True  | （缺省）显示指定的按钮                          |

# 数据类型

Integer(Boolean)

# 请参阅

Multimedia MCI, Multimedia MCI 控件常量, *Visible* 属性（ActiveX 控件）。

# CanEject 属性（Multimedia MCI 控件）

确定加电的 MCI 设备是否可以退出其中的介质。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

[form.]MMControl.CanEject

说明

下表列出了 Multimedia MCI 控件的 CanEject 属性的设置值：

| 设置    | 描述           |
|-------|--------------|
| False | （缺省）设备不能退出介质 |
| True  | 设备可以退出介质     |

在执行 Open 命令的过程中，使用 MCI\_GETDEVCAPS 检取 CanEject 属性的值。

数据类型

Integer (Boolean)

请参阅

Multimedia MCI, CanPlay 属性（Multimedia MCI 控件），CanRecord 属性（Multimedia MCI 控件），CanStep 属性（Multimedia MCI 控件），Command 属性（Multimedia MCI 控件）。

CanPlay 属性（Multimedia MCI 控件）

确定加电的 MCI 设备是否可以播放。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.CanPlay*

说明

下表列出了 Multimedia MCI 控件的 CanPlay 属性的设置值：

| 设置    | 描述         |
|-------|------------|
| False | （缺省）设备不能播放 |
| True  | 设备可以播放     |

在执行 **Open** 命令的过程中，使用 **MCI\_GETDEVCAPS** 检取 **CanPlay** 属性的值。

## 数据类型

**Integer (Boolean)**

请参阅

**Multimedia MCI**，**CanEject** 属性（**Multimedia MCI** 控件），**CanRecord** 属性（**Multimedia MCI** 控件），**CanStep** 属性（**Multimedia MCI** 控件），**Command** 属性（**Multimedia MCI** 控件）。

## CanRecord 属性（Multimedia MCI 控件）

确定加电的 **MCI** 设备是否可以记录。该属性在设计时不可用，在运行时是只读的。

应用于

**Multimedia MCI** 控件。

语法

[form.]MMControl.CanRecord

# 说明

下表列出了 Multimedia MCI 控件的 CanRecord 属性的设置值：

| 设置    | 描述         |
|-------|------------|
| False | （缺省）设备不能记录 |
| True  | 设备可以记录     |

在执行 Open 命令的过程中，使用 MCI\_GETDEVCAPS 检取 CanRecord 属性的值。

## 数据类型

Integer (Boolean)

请参阅

Multimedia MCI, CanEject 属性（Multimedia MCI 控件），CanPlay 属性（Multimedia MCI 控件），CanStep 属性（Multimedia MCI 控件），Command 属性（Multimedia MCI 控件）。

## CanStep 属性（Multimedia MCI 控件）

确定加电的 MCI 设备是否可以步进播放。该属性在设计时不可用，在运行时是只读的。



应用于

Multimedia MCI 控件。

语法

[form.]MMControl.CanStep

说明

下表列出了 Multimedia MCI 控件的 CanStep 属性的设置值：

| 设置    | 描述           |
|-------|--------------|
| False | （缺省）设备不能步进播放 |
| True  | 设备可以步进播放     |

目前，只有 MMMovie, Overlay 和 VCR MCI 设备可以步进播放（一次播放一帧）。因为没有办法确定设备是否可以步进播放，所以程序员需要在 Open 命令中检查该设备是否是 MMMovie, Overlay 和 VCR MCI 设备而设置该属性的值。

数据类型

Integer (Boolean)

请参阅

Multimedia MCI, CanEject 属性（Multimedia MCI 控件），CanRecord 属性

（Multimedia MCI 控件），CanPlay 属性（Multimedia MCI 控件），Command 属性（Multimedia MCI 控件）。

## Command 属性（Multimedia MCI 控件）

指定要运行的 MCI 命令。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.Command [=cmdstring$]`

说明

cmdstring\$参数给出要执行的 MCI 命令: Open, Close, Play, Pause, Stop, Back, Step, Prev, Next, Seek, Record, Eject, Sound 或 Save。立即执行该命令，错误码保存在 Error 属性中。

下表列出了每个命令以及相应的参数。如果没有设置该属性，则使用缺省值或根本不使用该属性。

| 命令   | 描述/使用属性              |
|------|----------------------|
| Open | 使用 MCI_OPEN 命令打开一个设备 |

续表

| 命令    | 描述/使用属性   |
|-------|---|
| Close | Notify (False)  |
|       | Wait(True)  |
|       | Sharable  |
|       | DeviceType  |
|       | FileName  |
| Play  | 使用 MCI_CLOSE 命令关闭一个设备   |
|       | Notify(False)   |
|       | Wait(True)  |
|       | 使用 MCI_PLAY 命令播放一个设备  |
|       | Notify(True)  |
| Pause | Wait(False)   |
|       | From  |
|       | To  |
|       | 使用 MCI_PAUSE 命令暂停一个设备。如果执行该命令时设备已经是暂停的，则使用 MCI_RESUME 命令恢复播放或记录 |
|       | Notify(False)   |
| Stop  | Wait(True)  |
|       | 使用 MCI_STOP 命令停止一个设备  |
|       | Notify(False)   |

续表

| 命令   | 描述/使用属性   |
|------|---|
| Back | Wait(True)  |
|      | 使用 MCI_STEP 命令回放一个设备  |
|      | Notify(False)   |
|      | Wait(True)  |
| Step | Frames  |
|      | 使用 MCI_STEP 命令步进播放一个设备  |
|      | Notify(False)   |
|      | Wait(True)  |
| Prev | Frames  |
|      | 使用 Seek 命令回到对齐磁道的开头。如果与上一个 Prev 命令之间的间隔少于 3 秒，则回到上一个磁道；如果是第一个磁道，则就返回到开头 |
|      | Notufy(False)   |
|      | Wait(True)  |
| Next | 使用 Seek 命令跳到下一个磁道的开头（如果是最后一个磁道，则返回到该磁道的开头）                              |
|      | Notify( False)  |
|      | Wait(True)  |

续表

| 命令     | 描述/使用属性  |
|--------|--|
| Seek   | 如果还没有播放，则使用 <b>MCI_SEEK</b> 命令进行定位。如果已经开始播放，使用 <b>MCI_PLAY</b> 命令在给定的位置接着播放<br>Notify( False)<br>Wait (True)<br>To |
| Record | 使用 <b>MCI_RECORD</b> 命令记录<br>Notify(False)<br>Wait(True)<br>From<br>To<br>RecoedMode ( 0 – Insert )                |
| Eject  | 使用 <b>MCI_SET</b> 命令退出介质<br>Notify(False)<br>Wait(True)  |
| Sound  | 使用 <b>MCI_SOUND</b> 命令播放声音<br>Notify(False)<br>Wait(True)<br>FileName  |
| Save   | 使用 <b>MCI_SAVE</b> 命令回放一个设备  |

续表

| 命令 | 描述/使用属性       |
|----|---------------|
|    | Notify(False) |
|    | Wait(True)    |
|    | FileName      |

数据类型

String

请参阅

Multimedia MCI 控件，CanEject 属性（Multimedia MCI 控件），CanPlay 属性（Multimedia MCI 控件），CanRecord 属性（Multimedia MCI 控件），CanStep 属性（Multimedia MCI 控件），Error 属性（Multimedia MCI 控件），ErrorMessage 属性（Multimedia MCI 控件），FileName 属性（Multimedia MCI 控件），Frames 属性（Multimedia MCI 控件），Length 属性（Multimedia MCI 控件），Notify 属性（Multimedia MCI 控件），RecordMode 属性（Multimedia MCI 控件），Shareable 属性（Multimedia MCI 控件），Silent 属性（Multimedia MCI 控件），Start 属性（Multimedia MCI 控件），To 属性（Multimedia MCI 控件），Track 属性（Multimedia MCI 控件），TrackLength 属性（Multimedia MCI 控件），TrackPosition 属性（Multimedia MCI 控件），Tracks 属性（Multimedia MCI 控件），UpdateInterval 属性（Multimedia MCI 控件），UsesWindows 属性（Multimedia MCI 控件），Wait 属性（Multimedia MCI 控件），ButtonClick 事件（Multimedia MCI 控件），

ButtonCompleted 事件（Multimedia MCI 控件），Done 事件（Multimedia MCI 控件），Error 消息（Multimedia MCI 控件）。

## DeviceID 属性（Multimedia MCI 控件）

指定当前打开 MCI 设备的设备 ID。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

```
[from.]MMControl.DeviceID [=id%]
```

说明

参数 *id*% 是当前打开 MCI 设备的设备 ID。该 ID 是 Open 命令执行 MCI\_OPEN 的结果。如果没有设备打开，则该参数是 0。

数据类型

Integer

请参阅

Multimedia MCI, Command 属性（Multimedia MCI 控件），DeviceType 属性（Multimedia MCI 控件），Error 消息（Multimedia MCI 控件）。

## DeviceType 属性（Multimedia MCI 控件）

指定要打开的 MCI 设备的类型。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.DeviceType [=device$]`

说明

参数 *device\$*是要打开的 MCI 设备的类型：AVIVideo, CDAudio, DAT, DigitalVideo, MMMovie, Other, Overlay, Scanner, Sequencer, VCR, Videodisc 或 WaveAudio。

打开一个简单的设备时（如不使用文件的 CD 唱盘）必须设置该属性。当文件名没有指定为设备所用时，打开组合 MCI 设备也需设置该属性。



数据类型

String

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), DeviceID 属性 (Multimedia MCI 控件)。

## Done 事件 (Multimedia MCI 控件)

当 Notify 属性为 True 的 MCI 命令执行完时产生该事件。

应用于

Multimedia MCI 控件。

语法

Private Sub *MMControl\_Done* (*NotifyCode* As Integer)

说明

*NotifyCode* 参数指明 MCI 命令是否成功。其可以取下面的设置

| 值 | 设置            | 结果          |
|---|---------------|-------------|
| 1 | mciSuccessful | 该命令成功完成     |
| 2 | mciSuperseded | 该命令被其他的命令替代 |
| 4 | mciAborted    | 用户的命令取消了    |
| 8 | mciFailure    | 该命令失败       |

请参阅

Multimedia MCI, Notify 属性（Multimedia MCI 控件），NotifyMessage 属性（Multimedia MCI 控件），NotifyValue 属性（Multimedia MCI 控件）。

## Enabled 属性（Multimedia MCI 控件）

确定该控件是否可以响应用户产生的事件，如 **KeyPress** 和鼠标事件等。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.Enabled [= {True|False} ]`

## 说明

该属性允许在运行时禁止或允许 **Multimedia MCI** 控件。**Enabled** 属性值将覆盖 **AutoEnable** 和 **ButtonEnabled** 属性。例如，如果 **Enabled** 属性是 **False**，则 **Multimedia MCI** 控件不允许访问按钮，不管 **AutoEnable** 和 **ButtonEnabled** 属性的设置如何。

下表列出了 **Multimedia MCI** 控件的 **Enabled** 属性的设置

| 设置    | 描述   |
|-------|--|
| True  | 控件上的所有按钮都禁止  |
| False | （缺省）控件允许使用。使用 <b>AutoEnable</b> 属性让 <b>Multimedia MCI</b> 控件自己管理控件中按钮的状态。或使用 <b>ButtonEnabled</b> 属性禁止或允许单个的按钮 |

## 数据类型

Integer (Boolean)

请参阅

**Multimedia MCI**,**AutoEnable** 属性（**Multimedia MCI** 控件），**ButtonEnabled** 属性（**Multimedia MCI** 控件）。

# Error 消息（Multimedia MCI 控件）

下表列出了 Multimedia MCI 控件可以捕获的错误：

| 常量                      | 值   | 描述   |
|-------------------------|-----|--|
| mciInvalidProcedureCall | 5   | 无效的过程调用  |
| mciInvalidPropertyValue | 380 | 属性值无效  |
| mciSetNotSupported      | 383 | 属性是只读的   |
| mciGetNotSupported      | 394 | 属性是只写的   |
| mciInvalidObjectUse     | 425 | 使用的对象无效  |
| mciWrongClipboardFormat | 461 | 指定的格式与数据格式不匹配  |
| mciObjectLocked         | 672 | DataObject 格式列表没有清除或扩展超出了 OLEStartDrag 事件                        |
| mciExpectedArgument     | 673 | 至少需要一个参数   |
| mciRecursiveOleDrag     | 674 | 重复调用 OLE 拖放无效  |
| mciFormatNotByteArray   | 675 | SetData 使用的非内部 OLE 拖放格式需要 Byte 矩阵数据。GetData 返回的字节数比 SetData 设置的多 |
| mciDataNotSetForFormat  | 676 | 在 OLESetData 事件中没有提供所要求的数据                                       |

续表

| 常量                     | 值     | 描述        |
|------------------------|-------|-----------|
| mciCantCreateButton    | 30001 | 不能创建按钮    |
| mciCantCreateTimer     | 30002 | 不能创建计数器资源 |
| mciUnsupportedFunction | 30004 | 不支持的功能    |

下表列出了 Multimedia MCI 控件的 MCI 错误字符串和错误码。

| MCI 错误字符串                      | MCI 错误码 |
|--------------------------------|---------|
| MCIERR_BASE                    | 256     |
| MCIERR_INVALID_DEVICE_ID       | 257     |
| MCIERR_UNRECOGNIZED_KEYWORD    | 259     |
| MCIERR_UNRECOGNIZED_COMMAND    | 261     |
| MCIERR_HARDWARE                | 262     |
| MCIERR_INVALID_DEVICE_NAME     | 263     |
| MCIERR_OUT_OF_MEMORY           | 264     |
| MCIERR_DEVICE_OPEN             | 265     |
| MCIERR_CANNOT_LOAD_DRIVER      | 266     |
| MCIERR_MISSING_COMMAND_STRING  | 267     |
| MCIERR_PARAM_OVERFLOW          | 268     |
| MCIERR_MISSING_STRING_ARGUMENT | 269     |

续表

| MCI 错误字符串                   | MCI 错误码 |
|-----------------------------|---------|
| MCIERR_BAD_INTEGER          | 270     |
| MCIERR_PARSER_INTERNAL      | 271     |
| MCIERR_DRIVER_INTERNAL      | 272     |
| MCIERR_MISSING_PARAMETER    | 273     |
| MCIERR_UNSUPPORTED_FUNCTION | 274     |
| MCIERR_FILE_NOT_FOUND       | 275     |
| MCIERR_DEVICE_NOT_READY     | 276     |
| MCIERR_INTERNAL             | 277     |
| MCIERR_DRIVER               | 278     |
| MCIERR_CANNOT_USE_ALL       | 279     |
| MCIERR_MULTIPLE             | 280     |
| MCIERR_EXTENSION_NOT_FOUND  | 281     |
| MCIERR_OUTOFRANGE           | 282     |
| MCIERR_FLAGS_NOT_COMPATIBLE | 283     |
| MCIERR_FILE_NOT_SAVED       | 286     |
| MCIERR_DEVICE_TYPE_REQUIRED | 287     |
| MCIERR_DEVICE_LOCKED        | 288     |
| MCIERR_DUPLICATE_ALIAS      | 289     |
| MCIERR_BAD_CONSTANT         | 290     |

续表

| MCI 错误字符串                     | MCI 错误码 |
|-------------------------------|---------|
| MCIERR_MUST_USE_SHAREABLE     | 291     |
| MCIERR_MISSING_DEVICE_NAME    | 292     |
| MCIERR_BAD_TIME_FORMAT        | 293     |
| MCIERR_NO_CLOSING_QUOTE       | 294     |
| MCIERR_DUPLICATE_FLAGS        | 295     |
| MCIERR_INVALID_FILE           | 296     |
| MCIERR_NULL_PARAMETER_BLOCK   | 297     |
| MCIERR_UNNAMED_RESOURCE       | 298     |
| MCIERR_NEW_REQUIRES_ALIAS     | 299     |
| MCIERR_NOTIFY_ON_AUTO_OPEN    | 300     |
| MCIERR_NO_ELEMENT_ALLOWED     | 301     |
| MCIERR_NONAPPLICABLE_FUNCTION | 302     |
| MCIERR_ILLEGAL_FOR_AUTO_OPEN  | 303     |
| MCIERR_FILENAME_REQUIRED      | 304     |
| MCIERR_EXTRA_CHARACTERS       | 305     |
| MCIERR_DEVICE_NOT_INSTALLED   | 306     |
| MCIERR_GET_CD                 | 307     |
| MCIERR_SET_CD                 | 308     |
| MCIERR_SET_DRIVE              | 309     |

续表

| MCI 错误字符串                       | MCI 错误码 |
|---------------------------------|---------|
| MCIERR_DEVICE_LENGTH            | 310     |
| MCIERR_DEVICE_ORD_LENGTH        | 311     |
| MCIERR_NO_INTEGER               | 312     |
| MCIERR_WAVE_OUTPUTSINUSE        | 320     |
| MCIERR_WAVE_SETOUTPUTINUSE      | 321     |
| MCIERR_WAVE_INPUTSINUSE         | 322     |
| MCIERR_WAVE_SETINPUTINUSE       | 323     |
| MCIERR_WAVE_OUTPUTUNSPECIFIED   | 324     |
| MCIERR_WAVE_INPUTUNSPECIFIED    | 325     |
| MCIERR_WAVE_OUTPUTSUNSUITABLE   | 326     |
| MCIERR_WAVE_SETOUTPUTUNSUITABLE | 327     |
| MCIERR_WAVE_INPUTSUNSUITABLE    | 328     |
| MCIERR_WAVE_SETINPUTUNSUITABLE  | 329     |
| MCIERR_SEQ_DIV_INCOMPATIBLE     | 336     |
| MCIERR_SEQ_PORT_INUSE           | 337     |
| MCIERR_SEQ_PORT_NONEXISTENT     | 338     |
| MCIERR_SEQ_PORT_MAPNODEVICE     | 339     |
| MCIERR_SEQ_PORT_MISCELLORROR    | 340     |
| MCIERR_SEQ_TIMER                | 341     |



续表

| MCI 错误字符串                  | MCI 错误码 |
|----------------------------|---------|
| MCIERR_SEQ_PORTUNSPECIFIED | 342     |
| MCIERR_SEQ_NOMIDIPRESENT   | 343     |
| MCIERR_NO_WINDOW           | 346     |
| MCIERR_CREATEWINDOW        | 347     |
| MCIERR_FILE_READ           | 348     |
| MCIERR_FILE_WRITE          | 349     |
| MCIERR_CUSTOM_DRIVER_BASE  | 512     |

请参阅

Multimedia MCI 控件，Error 属性（Multimedia MCI 控件），ErrorMessage 属性（Multimedia MCI 控件）。

### Error 属性（Multimedia MCI 控件）

指定上一个 MCI 命令返回的错误码。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.Error`

说明

如果上一个 MCI 命令没有错误，则该属性是 0。

数据类型

Integer

请参阅

Multimedia MCI, Command 属性（Multimedia MCI 控件）， ErrorMessage 属性（Multimedia MCI 控件）， Error 消息（Multimedia MCI 控件）。

ErrorMessage 属性（Multimedia MCI 控件）

描述保存在 Error 属性中的错误码。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.ErrorMessage`

数据类型

String

请参阅

Multimedia MCI, Command 属性（Multimedia MCI 控件），Error 属性（Multimedia MCI 控件），Error 消息（Multimedia MCI 控件）。

## FileName 属性（Multimedia MCI 控件）

指定 Open 命令打开的文件名或 Save 命令保存的文件名。要在运行时改变 FileName 属性，必须关闭和重新打开 Multimedia MCI 控件。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.FileName [=stringexpression$]`

## 说明

参数 *stringexpression*\$指定要打开或保存的文件名。

## 数据类型

String

请参阅

Multimedia MCI, Command 属性（Multimedia MCI 控件）。

## 示例

要在运行时改变 Multimedia MCI 控件的 FileName 属性，必须先关闭该控件再打开新文件。

下面的例子找到选择的项，关闭 Multimedia MCI 控件，重新设置 DeviceType 属性，重新设置 FileName 属性，然后重新打开 Multimedia MCI 控件。要使用本例，在一个窗体上放置一个 Multimedia MCI 控件和一个 FileListBox 控件，将下面的代码拷贝到窗体的 Declarations 段中。

```
Private Sub File1_Click()  
    Dim i As Integer  
    For i = 0 To File1.ListCount - 1  
        If File1.Selected(i) = True Then  
            MMControl1.Command = "close"
```

```
MMControl1.DeviceType = "WaveAudio"  
MMControl1.FileName = "c:\windows\" & File1.List(i)  
MMControl1.Command = "open"  
End If  
Next i  
End Sub
```

## Frames 属性（Multimedia MCI 控件）

找到 Step 命令前进或 Back 命令后倒的帧数。该属性在设计时不可用。

应用于

Multimedia MCI 控件。

语法

[*form.*]MMControl.Frames [=frames&]

说明

*frames&*参数指定前进或后倒的帧数。

数据类型

Long

请参阅

Multimedia MCI, Command 属性（Multimedia MCI 控件）。

## From 属性（Multimedia MCI 控件）

为 Play 或 Record 命令指定起点位置，该位置参数由 Multimedia MCI 控件 TimeFormat 属性定义。该属性在设计时不可用。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.From [=location&]*

说明

参数 *location&*指定播放或记录操作的起点。当前时间格式由 TimeFormat 属性指定。

给该属性分配的值只在下一个 MCI 命令使用。后续的 MCI 命令将忽略 From 属性，直到你分配了另一个值（相同或不同）。

## 数据类型

Long

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), TimeFormat 属性 (Multimedia MCI 控件), To 属性 (Multimedia MCI 控件)。

## hWndDisplay 属性 (Multimedia MCI 控件)

指定使用窗口显示输出的 MCI MIMovie 或 Overlay 设备的输出窗口。该属性在设计时不可用。

应用于

Multimedia MCI 控件。

语法

[*form.*]MMControl(hWndDisplay

说明

该属性是 MCI 设备应用于显示输出的窗口句柄。如果该句柄是 0，就使用缺省的窗口。

要确定设备是否使用该属性，查看 **UsesWindows** 属性的设置。

在 **Visual Basic** 中，要获取一个控件的句柄，首先使用 **SetFocus** 方法为所要求的控件设置输入焦点。然后调用 **Windows GetFocus** 函数。

要获得 **Visual Basic** 窗体的句柄，使用该窗体的 **hWnd** 属性。

## 数据类型

**Integer**

请参阅

**Multimedia MCI**，**UsesWindows** 属性（**Multimedia MCI** 控件），**SetFocus** 方法，**hWnd** 属性，**GotFocus** 事件。

## Length 属性（Multimedia MCI 控件）

指定 **MCI** 设备中介质的长度，该长度由 **Multimedia MCI** 控件中 **TimeFormat** 属性定义。该属性在设计时不可用。

应用于

**Multimedia MCI** 控件。



语法

`[form.]MMControl.Length`

数据类型

Long

请参阅

Multimedia MCI, Command 属性（Multimedia MCI 控件），From 属性（Multimedia MCI 控件），TimeFormat 属性（Multimedia MCI 控件）。

## Mode 属性（Multimedia MCI 控件）

返回当前打开的 MCI 设备的模式。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

`[form.]MMControl.Mode`

# 说明

下表列出了 Multimedia MCI 控件的 Mode 属性返回值。

| 值   | 设置/设备模式        | 模式     |
|-----|----------------|--------|
| 524 | mciModeNotOpen | 设备没有打开 |
| 525 | mciModeStop    | 设备停止   |
| 526 | mciModePlay    | 设备正在播放 |
| 527 | mciModeRecord  | 设备正在记录 |
| 528 | mciModeSeek    | 设备在寻道  |
| 529 | mciModePause   | 设备暂停   |
| 530 | mciModeReady   | 设备就绪   |

## 数据类型

Long

请参阅

Multimedia MCI。

## Notify 属性（Multimedia MCI 控件）

确定下一个 MCI 命令是否使用 MCI 通知服务。如果设置为 **True**，则在下

一个 MCI 命令完成时，Notify 属性产生回调事件（Done）。该属性在设计时不可用。

应用于

Multimedia MCI 控件。

语法

[form.]MMControl.Notify [= {True|False}]

说明

下面列出了 Multimedia MCI 控件的 Notify 属性的设置

| 设置    | 描述                   |
|-------|----------------------|
| False | （缺省）下一个命令不产生 Done 事件 |
| True  | 下一个命令产生 Done 事件      |

该属性的设置值只在下一个 MCI 命令中使用。后续的 MCI 命令将忽略该属性值，除非你又分配了一个新值（相同或不同）。

**注意：**如果你发送的新命令将阻止上一个命令满足回调条件，则就丢弃通知消息。例如，要启动不支持 MCI Resume 命令的处于暂停状态的设备，Multimedia MCI 控件就给该设备发送 Play 命令。然而，Play 命令重新设置回调条件，从而覆盖了上次命令的回调条件。

## 数据类型

Integer(boolean)

## 请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), NotifyMessage 属性 (Multimedia MCI 控件), NotifyValue 属性 (Multimedia MCI 控件), Done 事件 (Multimedia MCI 控件)。

## NotifyMessage 属性 (Multimedia MCI 控件)

描述 Notify 属性触发的 Done 事件中的通知码。NotifyMessage 属性在设计时不可用，在运行时是只读的。

## 应用于

Multimedia MCI 控件。

## 语法

[*form.*]MMControl.NotifyMessage

## 数据类型

String

请参阅

Multimedia MCI, Notify 属性（Multimedia MCI 控件），NotifyValue 属性（Multimedia MCI 控件），Done 事件（Multimedia MCI 控件）。

## NotifyValue 属性（Multimedia MCI 控件）

指定要求通知的上一个 MCI 命令的结果。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.NotifyValue*

说明

下表列出了 Multimedia MCI 控件的 NotifyValue 属性的返回值：

| 值 | 设置                  | 设备模式   |
|---|---------------------|--------|
| 1 | mciNotifySuccessful | 命令成功完成 |
| 2 | mciNotifySuperseded | 命令被替换  |

续表

| 值 | 设置               | 设备模式  |
|---|------------------|-------|
| 4 | MciNotifyAborted | 命令被取消 |
| 8 | mciNotifyFailure | 命令失败  |

程序中对于大多数 MCI 命令，可以检查 Done 事件中的 Notify 码确定该值。

数据类型

Integer（Enumerated）

请参阅

Multimedia MCI, Notify 属性（Multimedia MCI 控件），NotifyMessage 属性（Multimedia MCI 控件），Done 事件（Multimedia MCI 控件），Multimedia MCI 控件常量。

Orientation 属性（Multimedia MCI 控件）

确定控件中的按钮是水平排列还是垂直排列。

应用于

Multimedia MCI 控件。

# 语法

```
[form.]MMControl.Orientation [=orientation%]
```

下表列出了 Multimedia MCI 控件的 Orientation 属性的设置

| 常量            | 值 | 描述     |
|---------------|---|--------|
| mciOrientHorz | 0 | 按钮水平排列 |
| mciOrientVert | 1 | 按钮垂直排列 |

# 数据类型

Integer(Enumerated)

请参阅

Multimedia MCI, Visible 属性（Multimedia MCI 控件），Multimedia MCI 控件常量。

# Position 属性（Multimedia MCI 控件）

指定打开 MCI 设备的当前位置，该位置由 Multimedia MCI 控件的 TimeFormat 属性定义。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

[*form.*]MMControl.Position

数据类型

Long

请参阅

Multimedia MCI, TimeFormat 属性（Multimedia MCI 控件）。

## RecordMode 属性（Multimedia MCI 控件）

对于支持记录的 MCI 设备指定记录模式。

应用于

Multimedia MCI 控件。

语法

[*form.*]MMControl.RecordMode [=*mode%*]



# 说明

下表列出了 Multimedia MCI 控件的 RecordMode 属性的设置

| 常量                 | 值 | 描述 |
|--------------------|---|----|
| mciRecordInsert    | 0 | 插入 |
| mciRecordOverwrite | 1 | 覆盖 |

要确定 MCI 设备是否支持记录，可以检查 CanRecord 属性。

支持记录的设备或许支持一种或两种记录模式。没有办法提前检查设备的记录模式。如果一个记录模式失败，试着使用另一种。

WaveAudio 设备只支持插入模式。

## 数据类型

Integer( Enumerated)

请参阅

Multimedia MCI, CanRecord 属性（Multimedia MCI 控件），Command 属性（Multimedia MCI 控件）。

# Shareable 属性（Multimedia MCI 控件）

确定是否多个程序可以共享同一个 MCI 设备。

应用于

Multimedia MCI 控件。

语法

```
[form.]MMControl.Shareable [= {True|False}]
```

说明

下表列出了 Multimedia MCI 控件的 Shareable 属性的设置

| 设置    | 描述                 |
|-------|--------------------|
| False | 没有其他控件或应用程序可以服务该设备 |
| True  | 多个控件或应用程序可以共享该设备   |

数据类型

Integer(Boolean)

请参阅

Multimedia MCI 控件，Commmand 属性（Multimedia MCI 控件）。

# Silent 属性（Multimedia MCI 控件）

确定是否播放声音。

应用于

Multimedia MCI 控件。

语法

```
[form.]MMControl.Silent [= {True |False} ]
```

说明

下表列出了 Multimedia MCI 控件的 Silent 属性的设置

| 设置    | 描述   |
|-------|------|
| False | 播放声音 |
| True  | 关闭声音 |

数据类型

Integer(Boolean)

请参阅

Multimedia MCI 控件，Command 属性（Multimedia MCI 控件）。

## Start 属性（Multimedia MCI 控件）

指定当前介质的起始位置，该位置由 Multimedia MCI 控件的 TimeFormat 属性定义。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.Start*

数据类型

Long

请参阅

Multimedia MCI 控件，Command 属性（Multimedia MCI 控件），TimeFormat 属性（Multimedia MCI 控件）。

## StatusUpdate 事件（Multimedia MCI 控件）

在 UpdateInterval 属性给定的时间间隔内自动产生该事件。

应用于

Multimedia MCI 控件。

语法

Private Sub *MMControl*\_StatusUpdate( )

说明

该事件允许应用程序更新显示信息，以提示用户有关 MCI 设备的当前状态。应用程序可以从属性中获取状态信息，如 Position, Length 和 Mode 等。

请参阅

Multimedia MCI 控件，Length 属性（Multimedia MCI 控件），Mode 属性（Multimedia MCI 控件），Position 属性（Multimedia MCI 控件），UpdateInterval 属性（Multimedia MCI 控件）。

TimeFormat 属性（Multimedia MCI 控件）

指定汇报位置信息所使用的时间格式。

应用于

Multimedia MCI 控件。

语法

```
[form.]MMControl.TimeFormat [=format&]
```

说明

下表列出了 Multimedia MCI 控件的 TimeFormat 属性的设置

| 值 | 设置                    | 时间格式   |
|---|-----------------------|--|
| 0 | mciFormatMilliseconds | 毫秒格式表示为 4 字节的整数形式  |
| 1 | mciFormatHms          | 时，秒，分组成 4 字节整数从低到高依次是：小时（最低字节），分，秒，没有使用（最高字节）  |
| 2 | mciFormatMsf          | 从低到高依次是：分（最低字节），秒，帧，没有使用（最高字节）   |
| 3 | mciFormatFrames       | 帧格式表示为 4 字节的整数形式   |
| 4 | mciFormatSmppte24     | 24-frame SMPTE 格式，组成 4 字节整数。从低到高依次是：小时（最低字节），分，秒，帧（最高字节）。SMPTE（Society of Motion Picture and Television Engineers）时间是以小时、分、秒和帧表示的绝对时间格式。标准的 SMPTE 类型有每秒 24、25 和 30 帧 |

续表

| 值               | 设置                   | 时间格式   |
|-----------------|----------------------|--|
| 5               | mciFormatSmpte25     | 25-frame SMPTE 格式，组成 4 字节整数，与 24-frame 的 SMPTE 格式一样      |
| 6               | mciFormatSmpte30     | 30-frame SMPTE 格式，组成 4 字节整数，与 24-frame 的 SMPTE 格式一样      |
| 7               | mciFormatSmpte30Drop | 30-drop-frame SMPTE 格式，组成 4 字节整数，与 24-frame 的 SMPTE 格式一样 |
| 8               | mciFormatBytes       | 字节表示为 4 字节整数变量   |
| 9               | mciFormatSamples     | 样本表示为 4 字节整数格式   |
| 10              | mciFormatTmsf        | 磁道，分，秒，帧组成 4 字节整数格式，从低到高依次是：磁道（最低），分，秒，帧（最高）             |
| 注意：MCI 使用连续磁道记数 |                      |  |

注意：并不是每个设备都支持所有的格式。如果设置了一个无效的格式，则忽略所分配的值。

当前，时间信息总是表示为 4 字节整数。在某些格式中，返回的时间并不真是一个整数。当单个的信息字节都打包到一个长整数中。访问或发送当前时间格式信息的属性有：From, Length, Position, Start, To , TrackLength, TrackPosition。

## 数据类型

Long(Enumerated)

请参阅

Multimedia MCI 控件, From 属性 (Multimedia MCI 控件), Length 属性 (Multimedia MCI 控件), Position 属性 (Multimedia MCI 控件), Start 属性 (Multimedia MCI 控件), To 属性 (Multimedia MCI 控件), Track 属性 (Multimedia MCI 控件), TrackLength 属性 (Multimedia MCI 控件), TrackPosition 属性 (Multimedia MCI 控件)。

## To 属性 (Multimedia MCI 控件)

为 Play 或 Record 命令指定终点, 该终点位置由 Multimedia MCI 控件中 TimeFormat 属性定义。该属性在设计时不可用。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.To [=location&]*



## 说明

参数 *location* 指定播放或记录操作的终点。当前时间格式由 **TimeFormat** 属性给定。

分配给该属性的值只在下一个 **MCI** 命令使用。后续的 **MCI** 命令忽略该值，直到你重新分配了一个新值（相同或不同）。

## 数据类型

**Long**

请参阅

**Multimedia MCI, Command** 属性（**Multimedia MCI** 控件），**From** 属性（**Multimedia MCI** 控件），**TimeFormat** 属性（**Multimedia MCI** 控件）。

## Track 属性（**Multimedia MCI** 控件）

指定 **TrackLength** 和 **TrackPosition** 属性返回的磁道信息。该属性在设计时不可用。

应用于

**Multimedia MCI** 控件。

语法

[*form.*]MMControl.Track [=*track*&]

说明

参数 *track*&指定磁道号。

该属性只应用于获取磁道的信息。与当前的磁道无关。

数据类型

Long

请参阅

Command 属性（Multimedia MCI 控件），TrackLength 属性（Multimedia MCI 控件），TrackPosition 属性（Multimedia MCI 控件）。

TrackLength 属性（Multimedia MCI 控件）

指定 Track 属性给定的磁道长度，该长度由 Multimedia MCI 控件中的 TimeFormat 属性定义。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.TrackLength*

数据类型

Long

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), Track 属性 (Multimedia MCI 控件), TrackPosition 属性 (Multimedia MCI 控件)。

## TrackPosition 属性 (Multimedia MCI 控件)

指定 Track 属性的起始位置，该位置由 Multimedia MCI 控件的 TimeFormat 属性定义。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.TrackPosition*

数据类型

Long

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), TimeFormat 属性 (Multimedia MCI 控件), Track 属性 (Multimedia MCI 控件), TrackLength 属性 (Multimedia MCI 控件)。

Tracks 属性 (Multimedia MCI 控件)

指出当前 MCI 设备上可用的磁道数。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.Tracks*

数据类型

Long

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), Track 属性 (Multimedia MCI 控件), TrackLength 属性 (Multimedia MCI 控件), TrackPosition 属性 (Multimedia MCI 控件)。

## UpdateInterval 属性 (Multimedia MCI 控件)

指定连续的 StatusUpdate 事件间隔的毫秒数。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.UpdateInterval [=milliseconds%]*

说明

*milliseconds%* 参数指定事件间隔的毫秒数。如果指定为 0，则不产生 StatusUpdate 事件。

数据类型

Integer

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件), StatusUpdate 事件 (Multimedia MCI 控件)。

## UsesWindows 属性 (Multimedia MCI 控件)

确定当前的 MCI 设备是否使用窗口进行输出。该属性在设计时不可用，在运行时是只读的。

应用于

Multimedia MCI 控件。

语法

*[form.]MMControl.UsesWindows*

说明

下表列出了 Multimedia MCI 控件的 UsesWindows 属性返回值：

| 值     | 描述          |
|-------|-------------|
| False | 当前设备不使用窗口输出 |
| True  | 当前设备使用窗口输出  |

目前，只有 **MMMovie** 和 **Overlay** 设备使用窗口进行输出。因为没有办法确定设备是否使用窗口，所以程序员需要在 **Open** 命令中检查该设备类型来设置该属性的值。如果设备类型是 **MMMovie**, **Overlay** 和 **VCR**，则该设备使用窗口进行输出。

对于使用窗口的设备，你可以使用 **hWndDisplay** 属性设置显示输出的窗口。

## 数据类型

**Integer (Boolean)**

请参阅

**Multimedia MCI, Command** 属性（**Multimedia MCI** 控件），**hWndDisplay** 属性（**Multimedia MCI** 控件）。

## Visible 属性（**Multimedia MCI** 控件）

确定 **Multimedia MCI** 控件在运行时是否可视。

应用于

**Multimedia MCI** 控件。

语法

*[form.]MMControl.Visible* [= {True|False}]

# 说明

Visible 属性的设置将覆盖单个 *ButtonVisible* 属性的设置。如果 Multimedia MCI 控件是可视的，则 *ButtonVisible* 属性控制单个按钮的可视状态。如果 Visible 属性设置为 False，则整个控件都不可视，并使用 *ButtonVisible* 属性。

下表列出了 Multimedia MCI 控件的 Visible 属性的设置

| 设置    | 描述  |
|-------|---|
| False | 控件不可视   |
| True  | （缺省）每个按钮的状态由 <i>ButtonVisible</i> 属性控制。该按钮的功能在控件中可用 |

## 数据类型

Integer(Boolean)

## 请参阅

Multimedia MCI 控件，ButtonVisible 属性（Multimedia MCI 控件），Hide 方法，Show 方法。

## Wait 属性（Multimedia MCI 控件）

确定 Multimedia MCI 控件是否等待下一个 MCI 命令执行完才将控制传递给



应用程序。该属性在设计时不可用。

应用于

Multimedia MCI 控件。

语法

```
[form.]MMControl.Wait [= {True|False}]
```

说明

下表列出了 Multimedia MCI 控件的 Wait 属性的设置

| 设置    | 描述   |
|-------|--|
| False | Multimedia MCI 控件不等待下一个 MCI 命令执行完就将控制返回给应用程序 |
| True  | Multimedia MCI 控件等待下一个 MCI 命令执行完才将控制返回给应用程序  |

对该属性分配的值只在下一个 MCI 命令中使用。后续的 MCI 命令忽略该值，除非你重新分配了一个新值（相同或不同）。

数据类型

Integer(Boolean)

请参阅

Multimedia MCI, Command 属性 (Multimedia MCI 控件)。

# MonthView 控件

**MonthView** 控件可以用来创建一个能够让用户通过日历风格的界面查看和设置日期信息的应用程序。

## 语法

**MonthView**

## 说明

**MonthView** 控件的 **Value** 属性返回当前被选定的日期。

可以允许最终用户通过将 **MultiSelect** 属性设置为 **True**，并使用 **MaxSelProperty** 指定可选择的天数来选择一个连续的日期范围。**SelStart** 和 **SelEnd** 属性返回所选择的日期范围的第一个日期和最后一个日期。

可以用许多方法自定义一个 **MonthView** 控件的外观。可以使用各种颜色属性，例如 **MonthBackColor**、**TitleBackColor**、**TitleForeColor** 和 **TrailingForeColor** 为控件创建一个唯一的配色方案。

通过设置 **MonthRows** 和 **MonthColumns** 属性，可以在一个 **MonthView** 控件中一次显示多个月份（多至 12）。**MonthRows** 和 **MonthColumns** 属性的总数必须小于或等于 12。

**注意：** **MonthView** 控件是 **ActiveX** 控件组的一部分，位于 **Mscomctl2.ocx** 文件中。如果要在应用程序中使用 **MonthView** 控件，必须将 **Mscomctl2.ocx** 文件添加到工程中。在发布该应用程序时，需要在用户的 **Microsoft Windows** 的 **System** 或 **System32** 目录中安装这个 **Mscomctl2.ocx** 文件。有关如何将 **ActiveX** 控件添加到工程中的详细信息，请参阅《**Microsoft Visual Basic 6.0** 程序员指南》中的“添加控件到工程”。

## 属性

**Day** 属性，**DayOfWeek** 属性，**DayBold** 属性，**MaxDate**，**MinDate** 属性，**MaxSelCount** 属性，**Month** 属性，**MonthBackColor** 属性，**MultiSelect** 属性（**MonthView** 控件），**ShowToday** 属性，**StartOfWeek** 属性，**ScrollRate** 属性，**SelEnd**，**SelStart** 属性（**MonthView** 控件），**ShowWeekNumbers** 属性，**TitleBackColor**，**TitleForeColor** 属性，**TrailingForeColor** 属性，**Value** 属性（**MonthView**，**DatePicker** 控件），**VisibleDays** 属性，**Week** 属性，**Year** 属性，**MonthColumns**，**MonthRows** 属性，**DataMember** 属性，**DataFormat** 属性，**DataBindings** 属性，**Left**，**Top** 属性，**TabIndex** 属性，**Visible** 属性，**DragIcon** 属

性, DragMode 属性, CausesValidation 属性, TabStop 属性, Enabled 属性, HelpContextID 属性, Index 属性 (控件矩阵), Name 属性, Parent 属性, Container 属性, Object 属性, ToolTipText 属性, DataChanged 属性, DataField 属性, DataSource 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), Font 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MouseIcon 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

HitTest 方法 (MonthView 控件), ComputeControlSize 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, Refresh 方法 (ActiveX 控件), OLEDrag 方法 (ActiveX 控件) Refresh 方法, HitTest 方法 (MonthView 控件), ComputerControlSize 方法。

## 事件

DateClick 事件, DateDblClick 事件, GetDayBold 事件, SelChance 事件 (MonthView 控件), DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop

事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), Click 事件 (ActiveX 控件), DblClick 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件), MouseDown, MouseUp 事件 (ActiveX 控件), MouseMove 事件 (ActiveX 控件)。

请参阅

使用 MonthView 控件。

## ComputeControlSize 方法

返回对一个给定行数和列数的 MonthView 控件的宽度和高度。

应用于

MonthView 控件。

语法

*object*.ComputeControlSize(Rows, Columns, Width, Height)

ComputeControlSize 方法的语法包括以下部分：

| 部分             | 描述                     |
|----------------|------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>rows</i>    | 设置一个用来指定行数的整数          |
| <i>columns</i> | 设置一个用来指定列数的整数          |
| <i>width</i>   | 返回一个单精度数值，即预期控件的宽度     |
| <i>height</i>  | 返回一个单精度数值，即预期控件的高度     |

## 说明

当通过重新设置 `MonthColumns` 或 `MonthRows` 属性更改了 `MonthView` 控件的大小时，可以在更改前使用 `ComputeControlSize` 方法计算该控件的大小，并相应地重新调整窗体大小。

要使用 `ComputeControlSize` 方法，首先应声明两个 `Single` 类型的变量。然后将这两个变量作为 `Width` 和 `Height` 参数来调用该方法。

## 示例

该示例在使用 `MonthRows` 和 `MonthColumns` 属性增加 `MonthView` 控件的大小后，再使用 `ComputeControlSize` 方法来计算该控件的大小。要检验该示例，请将一个 `MonthView` 控件放置到一个窗体中，然后将下述代码粘贴到代

码模块的 **Declarations** 部分。启动该工程，并双击该窗体。

```
Private Sub Form_Load()  
    ' 设置该控件的 Top 和 Left 属性。  
    With MonthView1  
        .Left = 200      ' 假设 ScaleMode = Twip  
        .Top = 400  
    End With  
End Sub  
  
Private Sub Form_DblClick()  
    Dim sWidth As Single  
    Dim sHeight As Single  
  
    With MonthView1      ' 计算控件的大小，增加列和行。  
        .ComputeControlSize 3, 4, sWidth, sHeight  
        .MonthColumns = 4  
        .MonthRows = 3    '  
    End With  
  
    ' 使用从该方法获得的值，重新设置该控件的大小。  
    Me.Width = MonthView1.Left + sWidth + 500  
    Me.Height = MonthView1.Top + sHeight + 500  
End Sub
```



# DateClick 事件

单击该控件上的一个日期时发生。

应用于

MonthView 控件。

语法

Private Sub *object*\_DateClick(*[index* As Integer], *DateClicked* As Date)

DateClick 事件的语法包括以下部分：

| 部分                 | 描述                        |
|--------------------|---------------------------|
| <i>object</i>      | 一个对象表达式，其值为“应用于”列表中的对象    |
| <i>index</i>       | 一个唯一标识控件的整数，如果该控件在一个控件数组中 |
| <i>dateClicked</i> | 一个日期表达式，指定被单击的日期          |

说明

DateClick 事件可以应用于响应用户单击一个特定的日期。DateClicked 可

以用来决定单击的是哪一个日期。

## DateDbClick 事件

双击控件上的一个日期时发生。

应用于

MonthView 控件。

语法

```
Private Sub object_DateDbClick([index As Integer], DateDbClicked As Date)
```

DateDbClick 事件的语法包括以下部分：

| 部分                   | 描述                        |
|----------------------|---------------------------|
| <i>object</i>        | 一个对象表达式，其值为“应用于”列表中的对象    |
| <i>index</i>         | 一个唯一标识控件的整数，如果该控件在一个控件数组中 |
| <i>dateDbClicked</i> | 一个日期表达式，指出被单击的日期          |

## 说明

**DateDbClick** 事件可以应用于响应用户双击一个特定的日期。  
**DateDbClicked** 可以用来决定双击的是哪一个日期。

## Day 属性

返回或设置一个值，指出当前的日期数。

应用于

**MonthView** 控件。

语法

*object*.Day [= *number*]

Day 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>number</i> | 一个数值表达式，其值为一个表示日期数的整数  |

## 说明

Day 属性可以设置为从 1 到 31 的任意整数。

## DayBold 属性

返回或设置一个值，决定所显示的日份是否用粗体。

## 应用于

MonthView 控件。

## 语法

*object*.DayBold(date) [= *boolean*]

DayBold 属性的语法包括以下部分：

| 部分             | 描述                                     |
|----------------|--|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象                 |
| <i>date</i>    | 一个日期表达式，指定 <b>VisibleDays</b> 属性中的一个日期 |
| <i>boolean</i> | 一个布尔表达式，指定该日期是否用粗体显示                   |

## 设置

*boolean* 的设置值为：

| 设置值   | 描述             |
|-------|----------------|
| True  | 该日期用粗体显示       |
| False | （缺省值）该日期不用粗体显示 |

## 说明

**DayBold** 属性是一个对应于 **VisibleDays** 属性的数组。每个布尔元素表示是否以粗体显示所对应的日期。

只有当前显示的日期才是合法的。可以通过查看 **VisibleDays** 属性来得到合法的日期。

当从一个月份移动到另一个月份时，该属性的信息不会被保留。

## 示例

下述代码将第一个和最后一个显示日期设置为粗体。要检验该示例，请将一个 **MonthView** 控件放置到一个窗体上，将代码粘贴到 **Declarations** 部分。运行该工程，并双击该窗体。

```
Private Sub Form_DblClick()  
    With MonthView1  
        .DayBold(MonthView1.VisibleDays(1)) = True  
        .DayBold(MonthView1.VisibleDays(42)) = True  
    End With  
End Sub
```

## DayOfWeek 属性

返回或设置一个值，指出当前为星期几的数值。

应用于

**MonthView** 控件。

# 语法

*object*.DayOfWeek [= *number*]

DayOfWeek 属性的语法包括以下部分：

| 部分            | 说明                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象    |
| <i>number</i> | 一个数值表达式，指出是星期几，如下面的“设置”所示 |

# 设置

*number* 的设置值为：

| 常量           | 值 | 描述       |
|--------------|---|----------|
| mvwSunday    | 1 | （缺省值）星期日 |
| mvwMonday    | 2 | 星期一      |
| mvwTuesday   | 3 | 星期二      |
| mvwWednesday | 4 | 星期三      |
| mvwThursday  | 5 | 星期四      |
| mvwFriday    | 6 | 星期五      |
| mvwSaturday  | 7 | 星期六      |

## 说明

**DayOfWeek** 属性可以设置为从 0 到 6 的任意整数。

## GetDayBold 事件

当控件需要显示日期以获得粗体信息时发生。

应用于

**DateTimePicker** 控件，**MonthView** 控件。

## 语法

```
Private Sub object__GetDayBold([index As Integer], StartDate As Date, Count As Integer, State( ) As Array)
```

**DateDbClick** 事件的语法包括以下部分：



| 部分                | 描述                        |
|-------------------|---------------------------|
| <i>object</i>     | 一个对象表达式，其值为“应用于”列表中的对象    |
| <i>index</i>      | 一个唯一标识控件的整数，如果该控件在一个控件数组中 |
| <i>startDates</i> | 一个日期表达式，指出显示的第一个日期        |
| <i>count</i>      | 一个数值表达式，指出显示的天数           |
| <i>state( )</i>   | 一个布尔值数组，指出一个日期是否用粗体显示     |

## 说明

当要显示日期时，可以使用 **GetDayBold** 事件来设置日期用粗体显示。

## 示例

下列代码将所有的星期五设置为以粗体显示。要检验该示例，请将一个 **MonthView** 控件放置到一个窗体中，将代码粘贴到 **Declarations** 部分。然后运行该工程。

```
Private Sub MonthView1_GetDayBold(ByVal StartDate As Date, ByVal Count As Integer,
State() As Boolean)
    ' 假定一周的开始为星期日，将变量 intBold
    ' 设为第五个星期五。然后将每个星期五的状态设为真。
    Dim intBold As Integer
```

```
intBold = mvwFriday
While intBold < Count
    State(intBold - 1) = True

    intBold = intBold + 7
Wend
End Sub
```

## HitTest 方法（MonthView 控件）

返回位于特定的坐标集合的日期。绝大多数通常使用拖放操作来决定当前位置是否可以作为拖放的目标项。

应用于

MonthView 控件。

语法

*object*.HitTest(*x* as Single, *y* As Single, *Date* As Date)

HitTest 方法的语法包括以下部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象   |
| <i>x,y</i>    | 目标 <i>date</i> 的坐标       |
| <i>date</i>   | 当控件上的一个日期产生鼠标操作时返回的日期表达式 |

## 返回设置值

HitTest 方法可能返回下述值，指出鼠标指针目前所指向日历的哪个部分：

| 常量                               | 值 | 描述   |
|----------------------------------|---|--|
| <code>mvwCalendarBack</code>     | 0 | 日历背景   |
| <code>mvwCalendarDate</code>     | 1 | 日历的日期  |
| <code>mvwCalendarDateNext</code> | 2 | 如果单击该区域，则日历将显示下一月份   |
| <code>mvwCalendarDatePrev</code> | 3 | 如果单击该区域，则日历将显示前月份  |
| <code>mvwCalendarDay</code>      | 4 | 该日期上的日期标号  |
| <code>mvwCalendarWeekNum</code>  | 5 | 星期号，如果 <code>ShowWeekNumbers</code> 被设置为 <code>True</code> |

续表

| 常量              | 值  | 描述   |
|-----------------|----|--|
| mvwNoWhere      | 6  | 日历的底边  |
| mvwTitleBack    | 7  | 日历的背景  |
| mvwTitleBtnNext | 8  | 标题区域的 <b>Next</b> 按钮   |
| mvwTitleBtnPrev | 9  | 标题区域的 <b>Previous</b> 按钮   |
| mvwTitleMonth   | 10 | 标题的月份字符串   |
| mvwTitleYear    | 11 | 标题的年份字符串   |
| mvwTodayLink    | 12 | 如果单击该区域，则该日历将显示当前月份和日期。只有当 <b>ShowToday</b> 被设为 <b>True</b> 时才可用 |

说明

**HitTest** 方法可以应用于决定日历的什么区域正在被鼠标操作影响。

如果在指定的坐标中不存在日期，**HitTest** 方法将返回 **Null**。 如果该坐标处不存在日期，**Date** 也可以是 **Null**。

示例

该示例使用 **HitTest** 方法，当鼠标经过一个日期时，在一个标签控件中显

示该日期。要检验该示例，请将一个 **MonthView** 控件和一个 **Label** 控件放置到一个窗体中，然后将下述代码粘贴到 **Declarations** 部分。

```
Private Sub MonthView1_MouseMove(Button As Integer, Shift As Integer, x As Single, y  
As Single)  
    Dim iResult As Integer  
    Dim dtMyDate As Date  
    iResult = MonthView1.HitTest(x, y, dtMyDate)  
    Label1.Caption = dtMyDate  
End Sub
```

## MaxDate, MinDate 属性

返回或设置该日历所允许的第一天和最后一天。

应用于

**MonthView** 控件。

# 语法

*object*.MaxDate [= *date*]

*object*.MinDate [= *date*]

MaxDate 和 MinDate 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>date</i>   | 一个日期表达式，其值为一个合法的日期     |

# 说明

MaxDate 属性应用于设置日历的上限。

MinDate 属性应用于设置日历的下限。

# MaxSelCount 属性

返回或设置一次可以选择的连续日期的最大数目。

# 语法

*object*.MaxSelCount [= *number*]

MaxSelCount 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>number</i> | 一个数值表达式，其值为一次可以选择的日期数  |

# 说明

只有当 MultiSelect 属性为 True 时，MaxSelCount 属性才有效。此外，MaxSelCount 属性必须设置为一个大于 SelStart 和 SelEnd 属性之差的一个值。例如，给定从 9/15 到 9/18 的一个选择范围，则 MonthView.SelEnd - MonthView.SelStart = 3。然而，由于实际上选择了四天，因此 MaxSelCount 必须设为 4。

该属性的缺省值是一个星期（即 7 天）。

# Month 属性

返回或设置一个值，指出当前的月份。

应用于

MonthView 控件。

语法

*object*.Month [= *number*]

Month 属性的语法包括以下部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象          |
| <i>number</i> | 一个常量或数值表达式，其值为一个表示月份的整数，如“设置”所示 |



# 设置

*number* 的设置值为:

| 常量           | 值  | 描述        |
|--------------|----|-----------|
| mvwJanuary   | 1  | January   |
| mvwFebruary  | 2  | February  |
| mvwMarch     | 3  | March     |
| mvwApril     | 4  | April     |
| mvwMay       | 5  | May       |
| mvwJune      | 6  | June      |
| mvwJuly      | 7  | July      |
| mvwAugust    | 8  | August    |
| mvwSeptember | 9  | September |
| mvwOctober   | 10 | October   |
| mvwNovember  | 11 | November  |
| mvwDecember  | 12 | December  |

# 说明

**Month** 属性可以设置为从 1 到 12 的任意整数。

# MonthBackColor 属性

返回或设置一个值，指定在月份中所显示的背景色。

应用于

MonthView 控件。

语法

*object*.MonthBackColor [= *color*]

MonthBackColor 属性的语法包括以下部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象              |
| <i>color</i>  | 一个数值或常量，应用于确定在月份中所显示的背景色，如“设置”中所描述的 |

设置

Visual Basic 使用 Microsoft Windows 操作环境的红-绿-蓝 (RGB) 配色方

案。color 的设置值为：

| 设置值       | 描述   |
|-----------|--|
| 标准 RGB 颜色 | 通过使用“颜色”调色盘或在代码中使用 RGB 或 QBColor 函数指定的颜色                                     |
| 系统缺省颜色    | 由对象浏览器中的 Visual Basic (VBRUN) 对象库所列出的系统颜色常量指定的颜色。Windows 操作环境将替换用户在控制面板中的设置值 |

说明

MonthBackColor 属性可以与 TitleBackColor、TitleForeColor 以及 TrailingForeColor 属性一起使用来自定义控件的颜色。

一个标准的 RGB 颜色的合法范围是从 0 到 16,777,215 (&HFFFFFF)。这一范围中的值的高字节为 0，低三位字节，从最低到最高字节，即决定了红、绿、蓝三种颜色各自的量。红、绿、蓝每一种成分都可以由一个从 0 到 255 (&HFF) 之间的数值表示。如果高字节不为 0，Visual Basic 将使用用户在控制面板中设置的系统颜色，和对象浏览器的 Visual Basic (VB) 对象库中所列出的常量代表的颜色。

# MonthColumns, MonthRows 属性

返回或设置一个值，指出横向和纵向显示的月份数目。

应用于

MonthView 控件。

语法

*object*.MonthColumns [= *number*]

*object*.MonthRows [= *number*]

MonthColumns 和 MonthRows 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>number</i> | 一个数值表达式，指定月份的数目        |

说明

MonthColumns 和 MonthRows 属性可以一次显示多个月份。

**MonthColumns** 属性允许您指定横向显示的月份数目。**MonthRows** 属性允许您指定纵向显示的月份数目。

该控件可以显示多至十二个月。

## MultiSelect 属性 （MonthView 控件）

返回或设置一个值，决定是否可以一次选择多个日期。

应用于

MonthView 控件。

语法

*object*.MultiSelect [= *boolean*]

MultiSelect 属性的语法包括以下部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象    |
| <i>boolean</i> | 一个布尔表达式，应用于说明该控件所表现出的选择行为 |

## 设置

*boolean* 的设置值为：

| 设置值   | 描述                |
|-------|-------------------|
| True  | （缺省值）用户可以一次选择多个日期 |
| False | 用户不可以一次选择多个日期     |

## 说明

缺省情况下，该控件允许用户选择一个日期范围。缺省的最大范围是一个星期（即 7 天）。通过设置 **MaxSelCount** 属性，可以更改这个最大的可选择范围。**Value** 属性将在这一范围中，指出哪个日期有焦点。

## ScrollRate 属性

返回或设置一个值，指出当用户单击一个滚动按钮时滚动的月份数目。

## 应用于

**MonthView** 控件。

# 语法

*object.ScrollRate* [= *number*]

**ScrollRate** 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>number</i> | 一个数值表达式，指定一次滚动的月份数目    |

# 说明

**ScrollRate** 属性允许用户一次滚卷多个月份。

## SelChange 事件（MonthView 控件）

当用户选择一个新的日期或一个新的日期范围时发生该事件。

# 应用于

**MonthView** 控件。

# 语法

Private Sub *object\_SelChange*(*StartDate* As Date, *EndDate* As Date, *Cancel* As Boolean)

SelChange 事件的语法包括以下部分：

| 部分               | 描述                     |
|------------------|------------------------|
| <i>object</i>    | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>startDate</i> | 选择的第一个日期               |
| <i>endDate</i>   | 选择的最后一个日期              |
| <i>cancel</i>    | 决定是否取消这一选择             |

# 示例

该示例打印用户所选择的日期范围。要检验该示例，请将一个 **MonthView** 控件放置到一个窗体中，然后将下述代码粘贴到代码模块的 **Declarations** 部分。启动该工程，并选择各种日期范围（应将 **MultiSelect** 属性设为 **True**）。

```
Private Sub MonthView1_SelChange(ByVal StartDate As Date, ByVal EndDate As Date,
Cancel As Boolean)
    Dim d As Date
    d = StartDate
```



```
Debug.Print "Start"  
While d <= EndDate  
    Debug.Print d  
    d = d + 1  
Wend  
Debug.Print "End"  
End Sub
```

## SelEnd、SelStart 属性（MonthView 控件）

返回或设置所选择的日期范围的上限和下限。

应用于

MonthView 控件。

语法

*object*.SelEnd [= *date*]

*object*.SelStart [= *date*]

SelEnd 和 SelStart 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>date</i>   | 一个日期表达式，其值为一个合法的日期     |

## 说明

SelStart 属性定义了所选择的日期范围的下限。SelEnd 属性则定义了所选择的日期范围的上限。

所选择的日期范围可以跨越多个月份。它可以包括目前没有显示的日期。

如果要使多个日期选择能正确工作，则必须将 MaxSelCount 属性设置为一个大于 SelStart 和 SelEnd 属性之差的值。

只有当 MultiSelect 属性被设置为 TRUE 时，SelStart 和 SelEnd 的设置值才是合法的。

## 示例

假设 MaxSelCount 属性被设为 7，则下述代码将选择从 1998 年 11 月 8 日到 1998 年 11 月 14 日之间的日期：

```
With MonthView1  
    .MultiSelect = True
```

```
.Value = CDate("11/8/98")  
.SelStart = CDate("11/8/98")  
.SelEnd = CDate("11/14/98")
```

```
End With
```

设置 **Value** 属性时会自动将 **SelStart** 和 **SelEnd** 属性设为同一日期。如果不首先设置 **Value** 属性，则设置 **SelStart** 属性将选择从目前所选择的日期到 **SelStart** 日期之间的所有日期。

## ShowToday 属性

返回或设置一个值，决定是否在该控件的底部显示当前日期。

应用于

**MonthView** 控件。

语法

*object*.ShowToday [= *boolean*]

ShowToday 属性的语法包括以下部分：

| 部分             | 描述                     |
|----------------|------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>boolean</i> | 一个布尔表达式，指定当前日期指示器的状态   |

## 设置

*boolean* 的设置值为：

| 设置值   | 描述          |
|-------|-------------|
| True  | （缺省值）显示当前日期 |
| False | 不显示当前日期     |

## ShowWeekNumbers 属性

返回或设置一个值，决定是否在每个星期的相邻位置显示星期序号。

## 应用于

MonthView 控件。

# 语法

*object*.ShowWeekNumbers [= *boolean*]

ShowWeekNumbers 属性的语法包括以下部分：

| 部分             | 描述                     |
|----------------|------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>boolean</i> | 一个布尔表达式，指出当前日期指示器的状态   |

# 设置

*boolean* 的设置值为：

| 设置值   | 描述           |
|-------|--------------|
| True  | 显示星期序号       |
| False | （缺省值）不显示星期序号 |

# 说明

星期序号在星期的左边显示，且序号从日历年份的第一个星期开始。

# StartOfWeek 属性

返回或设置一个值，指定星期的开始日期。

应用于

MonthView 控件。

语法

*object*.StartOfWeek [= *integer*]

StartOfWeek 属性的语法包括以下部分：

| 部分             | 描述                                |
|----------------|-----------------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的对象            |
| <i>integer</i> | 一个数值表达式，指定一个星期的开始日期，如“设置值”<br>中所示 |

设置

*integer* 的设置值为：

| 常量           | 值 | 描述       |
|--------------|---|----------|
| mvwSunday    | 1 | （缺省值）星期日 |
| mvwMonday    | 2 | 星期一      |
| mvwTuesday   | 3 | 星期二      |
| mvwWednesday | 4 | 星期三      |
| mvwThursday  | 5 | 星期四      |
| mvwFriday    | 6 | 星期五      |
| mvwSaturday  | 7 | 星期六      |

### 说明

使用 **StartOfWeek** 属性可以指定将显示在日历最左边的日期。这一点在为多个国家/地区设计应用程序时可能会很有用。

### TitleBackColor,TitleForeColor 属性

返回或设置应用于指定该控件的标题域的背景色和前景色的数值。

应用于

MonthView 控件。

语法

*object*.TitleBackColor [= *color*]

*object*.TitleForeColor [= *color*]

TitleBackColor 和 TitleForeColor 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>color</i>  | 一个数值或常量，决定所用的颜色        |

设置

Visual Basic 使用 Microsoft Windows 操作系统的红-绿-蓝 (RGB) 配色方案。*color* 的设置值为：



| 设置值        | 描述   |
|------------|--|
| 标准的 RGB 颜色 | 通过使用 Color 调色盘或在代码中使用 RGB 或 QBColor 函数指定的颜色                                  |
| 系统缺省颜色     | 由对象浏览器的 Visual Basic (VBRUN) 对象库中列出的系统颜色常量所指定的颜色。Windows 操作环境将替换用户在控制面板中的设置值 |

### 说明

TitleBackColor 和 TitleForeColor 属性可以和 MonthBackColor 以及 TrailingForeColor 属性一起使用来自定义该控件的颜色。

一个标准的 RGB 颜色的合法范围是从 0 到 16,777,215 (&HFFFFFF)。这一范围中的值的高字节为 0，低三位字节，从最低到最高字节，即决定了红、绿、蓝三种颜色各自的量。红、绿、蓝每一种成分都可以由一个从 0 到 255 (&HFF) 之间的数值表示。如果高字节不为 0，Visual Basic 将使用用户在控制面板中设置的系统颜色，和对象浏览器的 Visual Basic (VBRUN) 对象库中所列出的常量代表的颜色。

### 示例

该示例通过重新设置 TitleBackColor,TitleForeColor 以及 TrailingForeColor

属性，来更改 **MonthView** 控件的外观。要检验该示例，请将一个 **MonthView** 控件放置到一个窗体中，然后将代码粘贴到代码模块的 **Declarations** 部分。运行该工程，并双击该窗体来察看日历的改动。

```
Private Sub Form_DblClick()  
    With MonthView1  
        .TitleBackColor = vbBlue  
        .TitleForeColor = vbWhite  
        .TrailingForeColor = vbRed  
    End With  
End Sub
```

## TrailingForeColor 属性

返回或设置一个值，指出当前显示的后续日期的前景色。

应用于

**MonthView** 控件。

语法

```
object.TrailingForeColor [= color]
```

TrailingForeColor 属性的语法包括以下部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象                        |
| <i>color</i>  | 一个数值或常量，决定 color of trailing dates，如“设置”中所描述的 |

设置

Visual Basic 使用 Microsoft Windows 操作系统的红-绿-蓝 (RGB) 配色方案。color 的设置值为：

| 设置值             | 描述   |
|-----------------|--|
| 标 准 的<br>RGB 颜色 | 通过使用“颜色”调色盘或在代码中使用 RGB 或 QBColor 函数指定的颜色                                     |
| 系 统 缺 省<br>颜色   | 由对象浏览器中的 Visual Basic (VBRUN) 对象库所列出的系统颜色常量指定的颜色。Windows 操作环境将替换用户在控制面板中的设置值 |

## 说明

后续日期是在当前所选择的月份之前或之后显示的天数。缺省情况下，后续日期用 `vbWhite` 颜色显示。

`TrailingForeColor` 属性可以与 `MonthBackColor`, `TitleBackColor` 以及 `TitleForeColor` 属性一起使用自定义控件的颜色。

一个标准的 RGB 颜色的合法范围是从 0 到 16,777,215 (&HFFFFFF)。这一范围中的值的高字节为 0，低三位字节，从最低到最高字节，即决定了红、绿、蓝三种颜色各自的量。红、绿、蓝每一种成分都可以由一个从 0 到 255 (&HFF) 之间的数值表示。如果高字节不为 0，Visual Basic 将使用用户在控制面板中设置的系统颜色，以及对象浏览器的 Visual Basic (VBRUN) 对象库中所列出的常量代表的颜色。

**Value 属性**（`MonthView`, `DatePicker` 控件）

返回或设置当前显示的日期。

应用于

DateTimePicker 控件， MonthView 控件。

语法

*object.Value* [= *date*]

Value 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>date</i>   | 一个日期表达式，指定由该控件所显示的日期   |

说明

Value 属性是该控件的缺省属性。

# VisibleDays 属性

返回一个包含当前可见日期的数组。

应用于

MonthView 控件。

语法

*object.VisibleDays(index)*

VisibleDays 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>index</i>  | 一个整数，指定在日历上显示的日期       |

说明

*Index* 可以是 1 到 42 的任意值。如果值为 1 则表示是目前所显示的第一天。

只有目前所显示的日期才能在 **VisibleDays** 属性中找到。此外，可以根据 **MonthColumns** 和 **MonthRows** 属性的设置值来更改可见日期的天数。

如果从一个月份移动到另一个月份，则该属性中的信息不会被保留。

## 示例

下列代码将打印在日历上所显示的所有日期。要检验该示例，请将一个 **MonthView** 控件放置到一个窗体中，然后将代码粘贴到 **Declarations** 部分。运行该工程，并双击该窗体。

```
Private Sub Form_DblClick()  
    Dim i As Integer  
    For i = 1 To 42  
        Debug.Print MonthView1.VisibleDays(i)  
    Next i  
End Sub
```

## Week 属性

返回或设置一个值，指出当前的星期序号。

应用于

MonthView 控件。

语法

*object*.Week [= *number*]

Week 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>number</i> | 一个数值表达式，其值为一个表示星期序号的整数 |

说明

Week 属性可以设置为从 1 到 52 的任意整数。

Year 属性

返回或设置应用于指定日历年份的数值。



应用于

MonthView 控件。

语 法

*object*.Year [= *number*]

Year 属性的语法包括以下部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个对象表达式，其值为“应用于”列表中的对象 |
| <i>number</i> | 一个数值表达式，其值为一个表示年份的整数   |

说 明

Year 属性可以设置为从 1601 到 9999 的任意整数。

## Masked Edit（屏蔽编辑）控件

**Masked Edit** 控件可以规定格式输出数据而且也可以限制输入的数据。该控件对输入或显示的数据类型都给出了提示。

文件名

MSMASK32.OCX。

类名

MaskedTextBox。

说明

**Masked Edit** 控件与标准的文本框控件相似，但其可以限定输出的格式和屏蔽输入数据。如果你没有使用输入屏蔽，则 **Masked Edit** 控件与标准的文本框一样，只是其还有 **DDE** 的能力。

如果你使用了 **Mask** 属性定义了一个屏蔽码，则 **Masked Edit** 控件的每个字符位置都映射为一个指定类型的置换元或提示字符。提示字符可以提示要使用的数据。例如，电话区号使用小括号括起来，如(206)。

如果你输入的字符与输入屏蔽码冲突，就产生一个 **ValidationError** 事件。输入屏蔽码将阻止你向控件中输入无效字符。

**Masked Edit** 控件有 3 个绑定属性：**DataChanged**, **DataField** 和 **DataSource**。这意味着其可以连接到数据控件，显示记录集中当前记录的数据域。**Masked Edit** 控件也可以将数据写入记录集。

当读取 **DataField** 属性引用的数据域时，只要有可能，其就转换为 **Text** 属性字符串。如果记录集是可以更新的，字符串就转换成数据域中的数据类型。

当你定义了一个屏蔽码而想清除 **Text** 属性时，必须先将 **Mask** 属性设置为空字符串，然后将 **Text** 属性设置为空字符串：

```
MaskedEdit1.Mask = ""
```

```
MaskedEdit1.Text = ""
```

定义了输入屏蔽码之后，**Masked Edit** 控件就与标准的文本框不一样了。当输入数据或移动光标时，光标将自动跳过提示字符。

当插入或删除一个字符时，如果有必要，光标右边的所有非提示字符都进行移位。如果移位将导致有效检验错误，就阻止移位并产生一个 **ValidationError** 事件。

假设 **Mask** 属性定义为 “?####”，**Text** 属性的当前值是 “A12”。如果你试图在字符 **A** 的左边插入一个字符 **B**，则字符 **A** 将移位到右边。因为输入的第二个字符应是数字，所以将产生一个 **ValidationError** 事件。

在运行时，**Masked Edit** 控件也检验 **Text** 属性的有效性。如果你设置的 **Text** 属性与输入屏蔽码有冲突，控件就产生一个运行时错误。

你可以像标准的文本编辑控件一样选择文本。当选择的文本被删除时，控件试图对剩余的字符进行移位。然而，如果移位将导致检验无效，就产生一个 **ValidationError** 事件。

通常，当 **Masked Edit** 控件中的选择内容拷贝到剪贴板上时，整个选择的内容，包括提示字符都传给了剪贴板。你可以使用 **ClipMode** 属性只将用户输入的事件传给剪贴板——不拷贝输入屏蔽码的提示字符。

## 属性:

**AllowPrompt** 属性, **AutoTab** 属性, **ClipMode** 属性, **ClipText** 属性, **Format** 属性, **FormattedText** 属性, **Mask** 属性, **MaxLength** 属性, **PromptChar** 属性, **PromptInclude** 属性, **SelText** 属性 (**Masked Edit** 控件) **Text** 属性 (**MaskedEdit** 控件), **DataFormat** 属性, **DataBinding** 对象, **TabIndex** 属性, **DragIcon** 属性, **DragMode** 属性, **MouseIcon** 属性, **SelLength**, **SelStart**, **SelText** 属性, **TabStop** 属性, **HelpContextID** 属性, **Name** 属性, **Parent** 属性, **Font** 属性, **Container** 属性,

ToolTipText 属性, DataChanged 属性, DataField 属性, WhatsThisHelpID 属性, OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), SelLength, SelStart, SelText 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), HideSelection 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

DataBindings 属性, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件)。

## 事件

ValidationError 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedBack 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件),

OLEStartDrag 事件（ActiveX 控件），Change 事件（ActiveX 控件），KeyDown, KeyUp 事件，KeyPress 事件。

请参阅

ValidationError 事件，Masked Edit 控件常量，TextBox 控件，ActiveX 控件常量，使用 MaskedEdit 控件。

## AllowPrompt 属性

确定提示符是否是有效字符。

应用于

Masked Edit 控件。

语法

*[form.]*MaskedEdit.AllowPrompt [= {True|False}]

## 说明

AllowPrompt 属性的设置如下：

| 设置    | 描述   |
|-------|--|
| False | （缺省）提示符不是有效输入字符。如果输入提示符将产生 <code>ValidationError</code> 事件 |
| True  | 提示符是有效输入字符   |

例如，假设定义了提示符是 0，而你想让 `Masked Edit` 控件接收 5 个 0—9 之间的数字。定义输入屏蔽码是#####。如果 `AllowPrompt` 属性设置为 `False` 而你输入了 0，就触发 `ValidationError` 事件。如果 `AllowPrompt` 设置为 `True`，输入的 0 就是有效输入字符。

## 事件类型

`Integer(Boolean)`

请参阅

`Masked Edit` 控件，`PromptChar` 属性，`ValidationError` 事件，`Masked Edit` 控件常量。

# AutoTab 属性

确定当 **Masked Edit** 控件的 **Text** 属性填满了有效字符后是否自动将输入焦点切换到下一个控件。**Mask** 属性决定 **Text** 属性中的值是否有效。

应用于

**Masked Edit** 控件。

语法

```
[form.]MaskedEdit.AutoTab [= {True|False}]
```

说明

只有输入了 **Mask** 属性定义的所有字符，字符有效并且 **AutoTab** 属性设置为 **True** 时才会自动切换焦点。**AutoTab** 属性的设置如下：

| 设置    | 描述   |
|-------|--|
| False | （缺省） <b>AutoTab</b> 关闭。当你输入的字符比输入屏蔽码定义的多时就触发 <b>ValidationError</b> 事件 |



续表

| 设置   | 描述   |
|------|--|
| True | AutoTab 打开。当输入了输入屏蔽码定义了所有字符后，输入焦点自动切换给后续的控件，后续输入的所有字符都由该控件处理 |

当输入 **Masked Edit** 控件中最后一个有效字符后，该控件就被填满了，不管你刚才输入的字符在什么位置。如果 **Mask** 属性设置为空字符串(""), 则该属性无效。

事件类型

Integer(Boolean)

请参阅

**Masked Edit** 控件，**Text** 属性，**ValidationError** 事件。

ClipMode 属性

确定在剪切或拷贝时是否包含输入屏蔽码中定义的提示字符。

应用于

Masked Edit 控件。

语法

[*form.*]MaskedEdit.ClipMode [=*setting* %]

说明

Masked Edit 控件中的 ClipMode 属性设置如下：

| 设置                    | 描述                         |
|-----------------------|----------------------------|
| mskIncludeLiterals(0) | （缺省）在剪切或拷贝时包含输入屏蔽码中定义的提示字符 |
| mskExcludeLiterals(1) | 在剪切或拷贝时不包含输入屏蔽码中定义的提示字符    |

如果 Mask 属性设置为空字符串，则该属性无效。

事件类型

Integer(Enumerated)

请参阅

**Masked Edit** 控件，**Masked Edit** 控件常量。

## ClipText 属性

返回 **Masked Edit** 控件中的文本，不包括输入屏蔽码中定义的提示字符。该属性在设计时是不可用的，在运行时是只读的。

应用于

**Masked Edit** 控件。

语法

[*form.*] **MaskedEdit.ClipText**

说明

当 **Mask** 属性设置为空字符串时，该属性与 **SelText** 属性的作用一样。

## 数据类型

String

请参阅

Masked Edit 控件，SelText 属性（Masked Edit 控件）。

## Format 属性（Masked Edit 控件）

指定显示和打印数字、日期、时间和文本的格式。

应用于

Masked Edit 控件。

语法

[*form.*]                      MaskedEdit.Format                      [=*posformat*%;                      *negformat*%;  
*zeroformat*%;*nullformat*%;

| 参数                  | 描述                |
|---------------------|-------------------|
| <i>posformat\$</i>  | 用于显示正数的表达式        |
| <i>negformat\$</i>  | 用于显示负数的表达式        |
| <i>zeroformat\$</i> | 用于显示 0 值的表达式      |
| <i>nullformat\$</i> | 用于显示 null 或空值的表达式 |

### 说明

Format 属性定义了用于显示控件内容的表达式格式。也可以使用 Visual Basic 定义的 Format 函数，但不能使用命名格式（“On/Off”）。

该属性可以有 1—4 个参数。如果没有指定其中的一个参数，就使用第一个参数指定的格式。如果有多个参数，必须使用相应的分割符。例如，要指定 *posformat\$* 和 *nullformat\$*，语法如下：

```
[form.] MaskedEdit.Format =posformat$;;;nullformat$
```

下表列出了常用的格式：

| 数据类型   | 值        | 描述              |
|--------|----------|-----------------|
| Number | （缺省）空字符串 | 一般的数字格式。按输入格式显示 |

续表

| 数据类型      | 值                       | 描述   |
|-----------|-------------------------|--|
| Number    | \$#,##0.00;(\$#,##0.00) | 货币格式。使用千分符；显示括号中的负数  |
| Number    | 0                       | 定点数格式。至少一个数字   |
| Number    | #,##0                   | 逗号格式。使用逗号作千分符  |
| Number    | 0%                      | 百分数格式。将设置乘以 100，添加一个百分号  |
| Number    | 0.00E+00                | 科学计数格式，使用标准的科学计数方法   |
| Date/Time | （缺省）c                   | 一般的日期和时间格式   |
| Date/Time | Dddddd                  | 长日期格式。与 Microsoft Windows Control Panel 中 International 中的格式一样。如 Tuesday, May 26, 1992 |
| Date/Time | dd-mmm-yy               | 中日期格式。如 26-May-92  |
| Date/Time | ddddd                   | 短日期格式。如 5/26/92  |
| Date/Time | ttttt                   | 长时间格式。如 05:36:17 A.M.  |
| Date/Time | hh:mm AM/PM             | 中时间格式。如 05:36 A.M.   |
| Date/Time | hh:mm                   | 短时间格式。如 05:36  |

数据类型

String

请参阅

Format 事件， Unformat 事件， Format 函数， MaskedEdit 控件。

## FormattedText 属性

当 Masked Edit 控件没有焦点时，FormattedText 属性返回控件中的字符串。该属性在设计时不可用，在运行时是只读的。

应用于

Masked Edit 控件。

语法

[*form.*] MaskedEdit.FormattedText

说明

如果 Format 属性是空字符串，则该属性与 Text 属性一样，除了该属性是只

读的之外。如果 **HideSelection** 属性设置为 **False**，在控件失去焦点时不显示格式化的文本。然而，仍可以通过该属性获取格式化文本。

数据类型

**String**

请参阅

**Masked Edit** 控件，**Text** 属性，**HideSelection** 属性。

## Mask 属性

确定控件的输入屏蔽码。

应用于

**Masked Edit** 控件。



# 语法

```
[form.]MaskedEdit.Mask [=string$]
```

# 说明

在设计时和运行时都可以定义输入屏蔽码。然而，在设计时通常使用下面的标准输入屏蔽码。控件可以区分数值和字母，但不能验证内容的正确与否，如当天的月份或时间等。

| 屏蔽码      | 描述                 |
|----------|--------------------|
| Null 字符串 | （缺省）没有屏蔽。与标准的文本框一样 |
| ##-??-## | 中日期格式。如 26-May-92  |
| ##-##-## | 短日期格式。如 5-26-92    |
| ##:## ?? | 中时间格式。如 05:36 AM   |
| ##:##    | 短时间格式。如 05:36      |

输入屏蔽码可以包含下列字符：

| 屏蔽码字符 | 描述   |
|-------|--|
| #     | 数字置换元  |
| .     | 小数点置换元。使用 <b>International</b> 设置中的实际字符。用于屏蔽提示 |
| ,     | 千分符。使用 <b>International</b> 设置中的实际字符。用于屏蔽提示    |

| 屏蔽码字符   | 描述  |
|---------|---|
| :       | 时间分隔符。使用 <b>International</b> 设置中的实际字符。用于屏蔽提示   |
| /       | 日期分隔符。使用 <b>International</b> 设置中的实际字符。用于屏蔽提示   |
| \       | 屏蔽码中的下一个字符是提示字符。允许在屏蔽码中包含 #,A,&和? 字符。<br>用于屏蔽提示 |
| &       | 字符置换元。该置换元的有效字符是 ANSI 字符集的 32—126 和 128—255     |
| >       | 将后续字符转换成大写                                      |
| <       | 将后续字符转换成小写                                      |
| A       | 字母或数字置换元。如 a-z, A-Z, 0-9                        |
| a       | 字母或数字置换元  |
| 9       | 数字置换元。如 0-9                                     |
| C       | 字母或空格置换元。与&置换元一样，与 Microsoft Access 兼容          |
| ?       | 字母置换元。如 a-z, A-Z                                |
| Literal | 所有其他符号都是提示符号                                    |

当 Mask 属性为空字符串时，控件的行为与标准的文本框控件一样。当定义了一个输入屏蔽码时，每个置换元的下面都带有下列线。你只能使用与屏蔽码中规定的相同类型的字符替换一个置换元。如果输入了无效字符，将触发 ValidationError 事件。

注意：当定义了 Masked Edit 控件的输入屏蔽码时，如果控件中有无效字符而你将输入焦点切换到其他控件，则将触发 ValidationError

事件。

数据类型

String

请参阅

Masked Edit 控件，Text 属性，ValidationError 事件。

## MaxLength 属性

返回或设置 Masked Edit 控件的最大长度。

应用于

Masked Edit 控件。

语法

[form.] MaskedEdit.MaxLength [=*setting*%]

## 说明

**Masked Edit** 控件最多可以有 64 字符长（该属性的有效范围是 1—64）。缺省值是 64，包括输入屏蔽码中的提示字符。

如果用户的输入超过了指定的最大长度，控件就发出警告声。

## 数据类型

Integer

请参阅

**Masked Edit** 控件。

## PromptChar 属性

设置或返回输入时提示用户的字符。

应用于

Masked Edit 控件。

语法

[*form.*] MaskedEdit.PromptChar [=*char*\$]

说明

该属性的缺省字符是下划线。**PromptChar** 属性只能设置一个字符。

使用 **PromptInclude** 属性指明在 **Text** 属性中是否包含提示字符。

使用 **AllowPrompt** 属性来检测是否允许用户输入提示字符。

数据类型

**String**

请参阅

Masked Edit 控件，**AllowPrompt** 属性，**PromptInclude** 属性。

# PromptInclude 属性

指明 Text 属性中是否包含提示字符。使用 PromptChar 属性检查提示字符值。

应用于

Masked Edit 控件。

语法

[*form.*] MaskedEdit.PromptInclude [= {True|False}]

说明

下面的表格列出了 Masked Edit 控件中 PromptInclude 属性的设置

| 设置    | 描述                   |
|-------|----------------------|
| False | （缺省）Text 属性值可以包含提示字符 |
| True  | Text 属性值不包含任何提示字符    |

如果 Masked Edit 控件与一个数据控件绑定，则 PromptInclude 属性影响到数据控件如何读取绑定的 Text 属性。如果 PromptInclude 设置为 False，则数据

控件忽略 **Text** 属性中的任何提示字符。这样，数据控件从 **Masked Edit** 控件中检取的数据就是 **ClipText** 属性值。

如果 **PromptInclude** 设置为 **Ture**，数据控件使用 **Text** 属性值作为数据的存储值。

## 数据类型

**Integer(Boolean)**

请参阅

**Masked Edit** 控件，**Text** 属性，**DataBinding** 对象，**DataField** 属性。

## SetText 属性（Masked Edit 控件）

设置或返回控件中包含的文本。

应用于

**Masked Edit** 控件。

## 语法

[*form.*] MaskedEdit.SelText [=*string*\$]

## 说明

如果 Masked Edit 控件没有定义输入屏蔽码，SelText 属性的行为与 Text Box 控件标准的 SelText 属性行为一样。

如果定义了输入屏蔽码，而且 Masked Edit 控件中有选择的文本，SelText 属性就返回文本字符串。根据 ClipMode 属性值的不同，并不是所选文本中的任何字符都返回。如果 ClipMode 是打开的，提示字符并不出现在返回字符串中。

如果设置了 SelText 属性，则 Masked Edit 控件的文本就像是从剪贴板粘贴过来的。

这意味着在 *string*\$ 中的每个字符都显示在控件中，就跟用户输入似的。

## 数据类型

String



请参阅

Masked Edit 控件，ClipMode 属性，SelLength, SelStart, SelText 属性，SefLenght,SelStart,SelText 属性（ActiveX 控件）。

## Text 属性（Masked Edit 控件）

返回或设置控件中包含的文本。该属性在设计时是不可用的。

应用于

Masked Edit 控件。

语法

[*form.*] MaskedEdit.Text [=*string*\$]

说明

该属性设置或检取 Masked Edit 控件中的文本，包括输入屏蔽码中的提示字

符和下划线。当设置 `Text` 属性时，`string$` 必须与输入屏蔽码中的字符严格匹配，包括提示字符和下划线。

**注意：**`ClipMode` 属性的设置对 `Text` 属性没有影响。

在 `Masked Edit` 控件中设置文本，`SetText` 属性提供的方法要容易些。

## 数据类型

`Variant`

请参阅

`Masked Edit` 控件，`ClipMode` 属性，`FormattedText` 属性，`SetText` 属性（`Masked Edit` 控件）。

## `ValidationError` 事件

当 `Masked Edit` 控件接受了无效字符时就触发该事件。

应用于

Masked Edit 控件。

语法

```
Private Sub ctlname_ValidationError(InvalidText As String ; StartPosition As Integer)
```

说明

*InvalidText* 是 Text 属性的值，包括无效字符。这意味着输入屏蔽码中的任何置换元和提示字符都包含在 *InvalidText* 中。

*StartPosition* 是 *InvalidText* 中发生错误的位置（第一个无效字符的位置）。

请参阅

Masked Edit 控件，AllowPrompt 属性，AutoTab 属性，Mask 属性，Text 属性。

# SSTab 控件

**SSTab** 控件提供了一组选项卡，每个都充当一个容器，包含了其他的控件。控件中每次只有一个选项卡是活动的，给用户提供了其所包含的控件，而其他选项卡都是隐藏的。

## 语法

## SSTab

## 说明

一个 **SSTab** 控件就有如一个笔记本中的分隔页或文件柜中的格子。使用 **SSTab** 控件，可以在应用程序中为窗体或对话框的同一个区域定义多个页面。使用该控件的属性可以：

- 确定选项卡的数量。
- 将选项卡组织成多行。
- 设置每个选项卡上的文本。
- 在每个选项卡上显示一个图像。
- 确定使用的选项卡风格。

- 设置每个选项卡的大小。

要使用该控件，必须先确定打算如何在选项卡中放置其他控件。设置 **Tabs** 和 **TabsPerRow** 属性来创建选项卡并将它们安排成多行显示。然后在设计时选择每个选项卡。对于每个选项卡，绘制每个你想显示的控件。如果必要，设置 **Caption**, **Picture**, **TabHeight** 和 **TabMaxWidth** 属性来安排选项卡的上部。

在运行时，用户可以使用 **CTRL+TAB** 或使用选项卡上定义的加速键浏览每个选项卡。

你也可以使用 **Style**, **ShowFocusRect**, **TabOrientation** 和 **WordWrap** 属性来自定义整个 **SSTab** 控件。

**发布须知：** **SSTab** 控件在文件 **TABCTL32.OCX** 中。要在应用程序中使用 **SSTab** 控件，必须将该控件的 **.OCX** 文件加入到工程中。当发布应用程序时，将 **.OCX** 文件安装到用户的 **Microsoft Windows System** 或 **System32** 目录下。要获取更多的信息，请参阅《**Microsoft Visual Basic 程序员指南**》。

## 属性

**Picture** 属性 (**SSTab** 控件)，**Rows** 属性 (**SSTab** 控件)，**ShowFocusRect** 属性 (**SSTab** 控件)，**Style** 属性 (**SSTab** 控件)，**Tab** 属性 (**SSTab** 控件)，**TabCaption** 属性 (**SSTab** 控件)，**TabEnabled** 属性 (**SSTab** 控件)，**TabHeight** 属性 (**SSTab** 控件)，**TabMaxWidth** 属性 (**SSTab** 控件)，**TabOrientation** 属性 (**SSTab** 控件)，

TabPicture 属性 (SSTab 控件), Tabs 属性 (SSTab 控件), TabsPerRow 属性 (SSTab 控件) TabVisible 属性 (SSTab 控件), WordWrap 属性 (SSTab 控件), DataBindings 属性, DragIcon 属性, DragMode 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, OLEDragMode 属性 (ActiveX 控件), Picture 属性 (ActiveX 控件), Height,Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), Caption 属性 (ActiveX 控件), Enable 属性 (ActiveX 控件), Font 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MouseIcon 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件)。

## 方法

SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件)。

## 事件

Click 事件 (SSTab 控件), DragDrop 事件, DragOver 事件, GotFocus 事件, KeyDown, KeyUp 事件, KeyPress 事件, LostFocus 事件, MouseDown, MouseUp 事件, MouseMove 事件, Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控

件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），DbClick 事件（ActiveX 控件），KeyDown, KeyUp 事件（ActiveX 控件），KeyPress 事件（ActiveX 控件）。

请参阅

TabStrip 控件，Picture 属性（SSTab 控件），ShowFocusRect 属性（SSTab 控件），Style 属性（SSTab 控件），TabHeight 属性（SSTab 控件），TabMaxWidth 属性（SSTab 控件），TabOrientation 属性（SSTab 控件），Tabs 属性（SSTab 控件），TabsPerRow 属性（SSTab 控件），SSTab 控件常量，WordWrap 属性（SSTab 控件），Caption 属性，Caption 属性（ActiveX 控件），使用 Tabbed Dialog 控件。

## Click 事件（SSTab 控件）

当用户选择 SSTab 控件上的一个选项卡时就产生一个 Click 事件。

应用于

SSTab 控件

语法

```
Private Sub object_Click( [index As Integer], previoustab As Integer)
```

Click 事件的语法有如下几个部分：

| 部分                 | 描述                      |
|--------------------|-------------------------|
| <i>object</i>      | 一个 SSTab 控件的对象表达式       |
| <i>index</i>       | 一个整数，当控件在一个矩阵中时，唯一标识该控件 |
| <i>previoustab</i> | 数字表达式，标识上一次活动的选项卡       |

## 说明

使用 Click 事件来确定用户什么时候点击激活选项卡。当选项卡收到 Click 事件时，该选项卡就成为活动选项卡，并且显示在设计时放置的控件。

在用户点击其他的选项卡时，可以使用 *previoustab* 参数来检查用户所作的修改。

使用 Tab 属性来确定当前的选项卡。

## 请参阅

SSTab 控件, Tab 属性 (SSTab 控件), DblClick 事件, DblClick 事件 (ActiveX 控件)。

## 示例

下面的例子中 SSTab 控件有两个选项卡，当用户点击不同的选项卡时，将保存所选择选项卡的信息。



```
Private Sub sstbPrefs_Click(PreviousTab As Integer)
    Dim ThisSetting As String
    Select Case PreviousTab
        Case 0
            If optLoanLen(0) = True Then
                ThisSetting = "Months"
            Else
                ThisSetting = "Years"
            End If
            SaveSetting("LoanSheet", "LoanLength", _
                "Period", ThisSetting)
        Case 1
            Dim X As Integer
            For X = 0 To 3
                If optPctsShown(X) = True Then
                    SaveSetting("LoanSheet", "InterestRate", _
                        "Precision", optPctsShown(X).Tag)
                Exit For
            End If
        Next X
    End Select
End Sub
```

End Select

End Sub

## Picture 属性（SSTab 控件）

返回或设置在 SSTab 控件的当前选项卡上显示的图像。

应用于

SSTab 控件。

语法

*object*.Picture [=*picture*]

Picture 属性的语法有如下几个部分：

| 部分             | 描述                                   |
|----------------|--------------------------------------|
| <i>object</i>  | 一个 SSTab 控件的对象表达式。                   |
| <i>picture</i> | 一个字符串表达式，设计在当前选项卡上显示的点位图或图标，如“设置”中所示 |

设置

*picture* 值的设置如下：

| 设置             | 描述                              |
|----------------|---------------------------------|
| (无)            | 一个 SSTab 控件的对象表达式               |
| (点位图, 图标, 元文件) | 一个字符串表达式, 设计当前选项卡上显示的<br>点位图或图标 |

## 说明

在设计时, 可以为每个选项卡设置 **Picture** 属性。在运行时, 可以使用 **LoadPicture** 函数或另外一个控件或窗体的 **Picture** 属性设置选项卡的 **Picture** 属性。通过设置 **Tag** 属性, 可以将任意一个选项卡设置为当前选项卡。

在设计时设置 **Picture** 属性, 该图像就保存下来, 并随包含 **SSTab** 控件的 **Form** 对象一起加载。如果创建了一个可执行文件, 文件就包含了该图像。在运行中加载图像时, 图像并不随应用程序一起保存。

设置 **Picture** 属性影响到当前选项卡的 **TabPicture** 属性, 也在当前的选项卡上显示该图像。

## 请参阅

**SSTab** 控件, **Tab** 属性 (**SSTab** 控件), **TabPicture** 属性 (**SSTab** 控件), **LoadPicture** 函数, **Form** 对象, **Forms** 集合, **Picture** 属性。

## 示例

下面的例子从一个文件中加载一个点位图并显示在当前活动的选项卡上。要使用本例，在一个窗体上放置一个 **SSTab** 控件和一个 **CommandButton** 控件。然后运行该例子。

```
Private Sub Command_Click()  
    SSTab1.Picture = LoadPicture("c:\Windows\Cars.bmp")  
End Sub
```

## Rows 属性（SSTab 控件）

返回 **SSTab** 控件中选项卡的总行数。

应用于

**SSTab** 控件。

语法

*object*.Rows

*object* 是一个 **SSTab** 控件的对象表达式。

## 说明

在设计时，设置 **Tabs** 和 **TabsPerRow** 属性可以指定一个 **SSTab** 控件上选项卡的行数。

## 请参阅

**SSTab** 控件，**Tabs** 属性（**SSTab** 控件），**TabsPerRow** 属性（**SSTab** 控件）。

## ShowFocusRect 属性

返回或设置一个值，该值确定 **SSTab** 控件中的选项卡获得了输入焦点时是否在选项卡上显示一个矩形边框。

## 应用于

**SSTab** 控件。

## 语法

*object*.ShowFocusRect [=*boolean*]

**ShowFocusRect** 属性的语法有如下几个部分：

| 部分             | 描述                               |
|----------------|----------------------------------|
| <i>object</i>  | 一个 <b>SSTab</b> 控件的对象表达式         |
| <i>boolean</i> | 一个布尔表达式，指定输入焦点边框的行为，如“设置”<br>中所示 |

### 设置

*boolean* 值的设置如下：

| 设置    | 描述                      |
|-------|-------------------------|
| True  | （缺省）控件在选项卡获取输入焦点时显示焦点矩阵 |
| False | 在选项卡获得输入焦点时不显示焦点矩阵      |

### 请参阅

**SSTab** 控件，**GotFocus** 事件，**LostFocus** 事件。

## Style 属性（**SSTab** 控件）

返回用户设置 **SSTab** 控件上选项卡的风格。

### 应用于

**SSTab** 控件。

语法

```
object.Style [=value]
```

Style 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个 SSTab 控件的对象表达式         |
| <i>value</i>  | 一个常量或整数，指明选项卡的风格，如“设置”中所示 |

设置

value 值的设置如下：

| 常量                  | 值 | 描述  |
|---------------------|---|---|
| ssStyleTabbedDialog | 0 | （缺省）选项卡的风格与 Microsoft Windows 3.1 中的 Microsoft Office 选项卡风格一样。在选择该风格时，活动选项卡采用黑体 |
| ssStylePropertyPage | 1 | 选项卡风格与 Windows 95 中一样。在选择该风格时，TabMaxWidth 属性被忽略，每个选项卡的宽度调整为标题中文本的宽度。字体不是黑体      |

请参阅

SSTab 控件，TabMaxWidth 属性（SSTab 控件），SSTab 控件常量。

# Tab 属性（SSTab 控件）

返回或设置一个 SSTab 控件当前的选项卡。

应用于

SSTab 控件。

语法

```
object.Tab [=tabnumber]
```

Tab 属性的语法有如下几个部分：

| 部分               | 描述                        |
|------------------|---------------------------|
| <i>object</i>    | 一个 SSTab 控件的对象表达式         |
| <i>tabnumber</i> | 一个整数表达式，指定某个选项卡。第一个选项卡是 0 |

说明

当前的选项卡移动到前面并成为活动选项卡。

通常，应用程序的使用者点击选项卡从而使其成为当前选项卡。然而你可能需要在代码中设置当前选项卡。例如，在应用程序中，或许需要每次打开某个对话框时都显示同一个选项卡。如果使用 Form 对象的 Hide 方法隐藏了对话框，则下次对话框出现时，上一次活动的选项卡就是当前选项卡。可以设置 SSTab



控件的 **Tab** 属性，这样，每次显示该对话框时总是出现同一个当前选项卡。

请参阅

**SSTab** 控件，**Form** 对象，**Forms** 集合，**Hide** 方法。

示例

下面的例子在显示窗体之前，总是将其中的第一个选项卡置为当前选项卡。要使用本例，创建两个 **Form** 对象，在 **Form1** 上放置一个 **CommandButton** 控件，在 **Form2** 上放置一个 **SSTab** 控件。将下面的代码拷贝到 **Form1** 上 **CommandButton** 控件的 **Click** 事件代码中。运行该例子。

```
Private Sub Command_Click()  
    Form2.SSTab1.Tab = 1  
    Form2.Show  
End Sub
```

## TabCaption 属性（SSTab 控件）

返回或设置 **SSTab** 控件中每个选项卡的标题。

应用于

SSTab 控件。

语法

*object.TabCaption (tab) [=text]*

TabCaption 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 一个 SSTab 控件的对象表达式       |
| <i>tyab</i>   | 一个数字表达式，指定要显示标题的选项卡     |
| <i>text</i>   | 一个字符串表达式，指定在选项卡中显示的标题文本 |

说明

在设计时，可以设置 TabCaption 属性，点击相应的选项卡，在 Properties 窗口中点击 Caption 属性即可进行设置。也可以在 Properties 窗口中选择（Custom），然后在 properties 对话框中的 General 选项卡中设置 TabCaption 属性。

在运行时，可以使用 TabCaption 属性读取或修改任何选项卡的标题。也可以使用 Caption 属性改变当前活动选项卡的 TabCaption 属性。

可以使用 TabCaption 属性给一个选项卡分配一个加速键。在 TabCaption 的

设置中，在加速键字符的前面放置一个&字符即可。加速键字符就带有了下划线。按下 ALT 和相应的加速键字符就可以将该选项卡激活。要在标题中放置一个&字符而不创建加速键，可以放置两个&字符。这样标题中就显示出一个&字符，而没有加速键。

请参阅

SSTab 控件，Tab 属性（SSTab 控件），Caption 属性，Caption 属性（ActiveX 控件）。

示例

下面的例子在 SSTab 控件的选项卡上增加或删除一个单词。点击 Form 对象的 CheckBox 控件，用户就可以切换标题。

```
Private Sub Check1_Click()  
    Dim X As Integer  
    For X = 0 To SSTab1.Tabs - 1  
        Select Case Check1.Value  
            Case 0 ' Toggle to short captions  
                SSTab1.TabCaption(X) = Left(SSTab1.TabCaption(X), 7)  
            Case 1 ' Toggle to long captions.  
                SSTab1.TabCaption(X) = SSTab1.TabCaption(X) & "Players"  
        End Select  
    End For  
End Sub
```

Next X

End Sub

## TabEnabled 属性（SSTab 控件）

返回或设置一个值，该值确定 SSTab 控件中的选项卡在点击时是否可用。

应用于

SSTab 控件。

语法

*object*.TabEnabled (*tab*) [=*boolean*]

TabEnabled 属性的语法有如下几个部分：

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 一个 SSTab 控件的对象表达式              |
| <i>tab</i>     | 一个数字表达式，指定选项卡                  |
| <i>boolean</i> | 一个布尔表达式，指定选项卡是否响应点击动作，如“设置”中所示 |

# 设置

*boolean* 值的设置如下：

| 设置    | 描述            |
|-------|---------------|
| True  | （缺省）选项卡响应点击动作 |
| False | 选项卡不响应点击动作    |

# 说明

当禁止一个选项卡时，选项卡上的文本变灰，用户不能选择该选项卡。

使用 **TabEnabled** 禁止或恢复单个选项卡。使用 **Enabled** 属性禁止或恢复整个 **SSTab** 控件。

# 请参阅

**SSTab** 控件，**Tab** 属性（**SSTab** 控件），**Enabled** 属性，**Enabled** 属性（**ActiveX** 控件）。

# TabHeight 属性（SSTab 控件）

返回或设置 **SSTab** 控件上所有选项卡的高度。

应用于  
SSTab 控件。

语法

*object.TabHeight [=height]*

TabHeight 属性的语法有如下几个部分：

| 部分            | 描述                |
|---------------|-------------------|
| <i>object</i> | 一个 SSTab 控件的对象表达式 |
| <i>height</i> | 一个数字表达式，指定选项卡的高度  |

请参阅

SSTab 控件，TabMaxWidth 属性（SSTab 控件）。

## TabMaxWidth 属性（SSTab 控件）

返回或设置 SSTab 控件上所有选项卡的最大宽度。

应用于  
SSTab 控件。

## 语法

*object*.TabMaxWidth [=width]

TabMaxWidth 属性的语法有如下几个部分：

| 部分            | 描述                |
|---------------|-------------------|
| <i>object</i> | 一个 SSTab 控件的对象表达式 |
| <i>width</i>  | 一个数字表达式，指定选项卡的宽度  |

## 说明

当 Style 属性设置为 ssStyleTabbedDialog,并且 TabMaxWidth 属性设置为 0, SSTab 控件将根据 TabsPerRow 属性自动调整选项卡的大小，使之充满整个控件。

如果 Style 属性设置为 ssStylePropertyPage，则忽略 TabMaxWidth 属性。每个选项卡的宽度将自动调整为 TabCaption 属性中的文本宽度。

## 请参阅

SSTab 控件，Style 属性（SSTab 控件），TabCaption 属性（SSTab 控件），TabHeight 属性（SSTab 控件），TabsPerRow 属性（SSTab 控件），SSTab 控件常量。

# TabOrientation 属性（SSTab 控件）

返回或设置 SSTab 控件中的选项卡位置。

应用于

SSTab 控件。

语法

*object*.TabOrientation [=*number*]

TabOrientation 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个 SSTab 控件的对象表达式         |
| <i>number</i> | 一个数字表达式，指定选项卡的位置，如“设置”中所示 |

设置

*number* 值的设置如下：

| 常量                     | 值 | 描述          |
|------------------------|---|-------------|
| ssTabOrientationTop    | 0 | 选项卡显示在控件的顶部 |
| ssTabOrientationBotton | 1 | 选项卡显示在控件的底部 |



续表

| 常量                     | 值 | 描述          |
|------------------------|---|-------------|
| ssTabOrientationLeft   | 2 | 选项卡显示在控件的左边 |
| ssTabOrientationRights | 3 | 选项卡显示在控件的右边 |

说明

如果使用的是 TureType 字符，当 TabOrientation 属性设置为 ssTabOrientationLeft 或 ssTabOrientationRight 时，文本将使用斜体。

请参阅

SSTab 控件，TabHeight 属性（SSTab 控件），TabMaxWidth 属性（SSTab 控件），SSTab 控件常量。

TabPicture 属性（SSTab 控件）

返回或设置 SSTab 控件中指定选项卡上显示的点位图或图标。

应用于

SSTab 控件。

# 语法

```
object.TabPicture (tab)[=picture]
```

TabPicture 属性的语法有如下几个部分：

| 部分             | 描述                                  |
|----------------|-------------------------------------|
| <i>object</i>  | 一个 SSTab 控件的对象表达式                   |
| <i>tab</i>     | 一个数字表达式，指定要显示图像的选项卡                 |
| <i>picture</i> | 一个字符串表达式，指定要在选项卡上显示的点位图或图标，如“设置”中所示 |

# 设置

*picture* 值的设置如下：

| 设置           | 描述   |
|--------------|--|
| (无)          | (缺省) 没有图像  |
| (点位图，图标，元文件) | 指定图像。在运行时，可以使用 LoadPicture 函数或其他控件或 Form 对象的 Picture 属性设置该属性的值 |

# 说明

在设计时，可以点击 Properties 窗口中的 Picture 属性来设置 TabPicture 属性值。或者，在 Properties 窗口中选择(Custom)，在 Properties 对话框中的 Picture

选项卡中设置 **Picture** 属性。

在运行时，可以使用 **TabPicture** 属性改变任意选项卡的图像，或使用 **Picture** 属性改变当前选项卡的图像。

请参阅

**SSTab** 控件，**Tab** 属性（**SSTab** 控件），**LoadPicture** 函数，**Form** 对象，**Forms** 集合，**Picture** 属性，**Picture** 属性（**ActiveX** 控件）。

## Tabs 属性（**SSTab** 控件）

返回或设置 **SSTab** 控件中所有选项卡的总数。

应用于

**SSTab** 控件。

语法

*object*.**Tabs**[=*tabnumber*]

**Tabs** 属性的语法有如下几个部分：

| 部分               | 描述   |
|------------------|--|
| <i>object</i>    | 一个 SSTab 控件的对象表达式                              |
| <i>tabnumber</i> | 一个数字表达式，指定控件上的选项卡。自动给选项卡分配标题 Tab x，x 指 1,2,3 等 |

## 说明

在运行时可以改变 Tabs 属性，给控件增加或删除选项卡。

在设计时，使用 Tabs 属性和 TabsPerRow 属性来确定控件中显示选项卡的行数。在运行时，使用 Rows 属性。

## 请参阅

SSTab 控件，Rows 属性（SSTab 控件），TabsPerRow 属性（SSTab 控件）。

## TabsPerRow 属性（SSTab 控件）

返回或设置一个 SSTab 控件中每行显示的选项卡数量。

## 应用于

SSTab 控件。

## 语法

*object*.TabsPerRow[=*tabnumber*]

TabsPerRow 属性的语法有如下几个部分：

| 部分               | 描述                      |
|------------------|-------------------------|
| <i>object</i>    | 一个 SSTab 控件的对象表达式       |
| <i>tabnumber</i> | 一个数字表达式，指定控件每行上显示的选项卡数量 |

## 说明

在设计时，使用 **Tabs** 属性来确定控件中显示选项卡的行数。在运行时，使用 **Rows** 属性。

## 请参阅

SSTab 控件，Rows 属性（SSTab 控件），Tabs 属性（SSTab 控件）。

## TabVisible 属性（SSTab 控件）

返回或设置一个值，该值指示 SSTab 控件是否可见。该属性在设计时不可用。

应用于  
SSTab 控件。

语法

```
object.TabVisible(tab)[=boolean]
```

TabVisible 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个 SSTab 控件的对象表达式          |
| <i>tab</i>     | 一个数字表达式，指定要隐藏或显示的选项卡       |
| <i>boolean</i> | 一个布尔表达式，指定选项卡是否可见，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                            |
|-------|-------------------------------|
| True  | （缺省）选项卡可见                     |
| False | 隐藏选项卡。其他的选项卡调整位置，这样在选项卡之间没有空隙 |

说明

TabVisible 属性隐藏或显示单个的选项卡。Visible 属性隐藏或显示整个的

SSTab 控件。

请参阅

SSTab 控件，Rows 属性（SSTab 控件），Tabs 属性（SSTab 控件），Visible 属性，Visible 属性（ActiveX 控件）。

## WordWrap 属性（SSTab 控件）

返回或设置一个值，该值指示在 SSTab 控件中选项卡文本太长时是否分行显示。

应用于

SSTab 控件。

语法

*object*.WordWrap[=*boolean*]

WordWrap 属性的语法有如下几个部分：

| 部分            | 描述                |
|---------------|-------------------|
| <i>object</i> | 一个 SSTab 控件的对象表达式 |

续表

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>boolean</i> | 一个布尔表达式，指定选项卡文本是否分行显示，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述                              |
|-------|---------------------------------|
| True  | 如果选项卡文本太长，超过了选项卡的宽度，则分行显示       |
| False | （缺省）文本不分行显示。如果文本超过了选项卡的宽度，就截断显示 |

说明

使用 **WordWrap** 属性来确定 **SSTab** 控件如何显示选项卡文本。例如，对话框中的选项卡动态变化时，选项卡文本的长度也跟着变化。如果文本太长而不想截断显示，可以将 **WordWrap** 属性设置为 **True**，**TabMaxWidth** 属性设置为 0 以及 **TabHeight** 属性设置为一个可以允许你看到整个文本的高度。



请参阅

SSTab 控件，TabCaption 属性（SSTab 控件），TabHeight 属性（SSTab 控件），TabMaxWidth 属性（SSTab 控件），Tabs 属性（SSTab 控件），Caption 属性，Caption 属性（ActiveX 控件）。

# PictureClip 控件

**PictureClip** 控件允许选择一个点位图的一部分，然后在窗体或图片框中使用它。**PictureClip** 控件在运行时是不可视的。

文件名

PICCLP32.OCX

类名

**PictureClip**

说明

**PictureClip** 控件为存储多个图片资源提供了有效的机制。可以在应用程序中创建一个包含了所有图标图像的源点位图，而不必使用多个点位图或图标。当需要显示单个图标时，使用 **PictureClip** 控件选择源点位图中相应的区域即可。

例如，可以使用该控件包含应用程序中显示工具箱所要求的所有图标。将工具箱图标保存在一个 **PictureClip** 控件中要比分开保存效率高得多。为此，需要先创建一个包含所有工具箱图标的源点位图。

**注意：**对于国际化的应用程序，资源文件有时是保存点位图的更有效方式。更多的信息请参阅《Microsoft Visual Basic 程序员指南》的第 16 章 “International Issues”。

使用下列两种方法之一指定源点位图中的剪裁区域：

- 选择源点位图的一部分作为剪裁区域。使用 **ClipX** 和 **ClipY** 属性指定剪裁区域的左上角。使用 **ClipHeight** 和 **ClipWidth** 属性指定剪裁区域的面积。当你想随机地选取剪裁区域时该方法是很有用的。
- 将源点位图分成指定的行和列。划分的结果是统一的图片单元分别标记为 0, 1, 2 等等。可以使用 **GraphicCell** 属性显示单个的单元。当元点位图包含了你想单个显示的图标模板时该方法就很有用，如工具箱图标。

使用 **Picture** 属性将源点位图加载到 **PictureClip** 控件中。只能给 **PictureClip** 控件加载点位图（.bmp）文件。

**发布须知：**当创建和发布使用了 **PictureClip** 控件的应用程序时，应该在用户的 Microsoft Windows System 或 System32 目录下安装 **PicClp32.ocx** 文件。Visual Basic 提供的 Setup Kit 工具可以帮助你编写正确的安装程序。

## 属性

**Clip** 属性 (**PictureClip** 控件)，**ClipHeight** 属性 (**PictureClip** 控件)，**ClipWidth** 属性 (**PictureClip** 控件)，**ClipX** 属性 (**PictureClip** 控件)，**ClipY** 属性 (**PictureClip**

控件），Cols, Rows 属性（PictureClip 控件），GraphicCell 属性（PictureClip 控件），Height, Width 属性（PictureClip 控件），Picture 属性（PictureClip 控件），StretchX, StretchY 属性（PictureClip 控件），CellHeight, CellWidth 属性（PictureClip 控件），hWnd 属性，Name 属性，Index 属性，Parent 属性，Picture 属性（ActiveX 控件），Height, Width 属性（ActiveX 控件），Index 属性（ActiveX 控件），Tag 属性（ActiveX 控件），Object 属性（ActiveX 控件），hWnd 属性（ActiveX 控件）。

请参阅

ClipHeight 属性（PictureClip 控件），ClipWidth 属性（PictureClip 控件），ClipX 属性（PictureClip 控件），ClipY 属性（PictureClip 控件），GraphicCell 属性（PictureClip 控件），Error 消息（PictureClip 控件），Picture 属性，Picture 属性（ActiveX 控件），PictureBox 控件，使用 PictureClip 控件。

示例

下面的例子在用户点击窗体指定 X 和 Y 坐标的位置时显示一个 Clip 图像。在窗体上放置一个 PictureBox 控件，一个 PictureClip 控件和两个 TextBox 控件。在设计时使用 Properties 窗口给 PictureClip 控件加载一个有效的点位图。

```
Private Sub Form_Click ()  
    Dim SaveMode As Integer  
    ' Save the current ScaleMode for the picture box.
```

```
SaveMode = Picture1.ScaleMode
' Get X and Y coordinates of the clipping region.
PicClip1.ClipX = Val(Text1.Text)
PicClip1.ClipY = Val(Text2.Text)
' Set the area of the clipping region (in pixels).
PicClip1.ClipHeight = 100
PicClip1.ClipWidth = 100
' Set the picture box ScaleMode to pixels.
Picture1.ScaleMode = 3
' Set the destination area to fill the picture box.
PicClip1.StretchX = Picture1.ScaleWidth
PicClip1.StretchY = Picture1.ScaleHeight
' Assign the clipped bitmap to the picture box.
Picture1.Picture = PicClip1.Clip
' Reset the ScaleMode of the picture box.
Picture1.ScaleMode = SaveMode
End Sub
```

## CellHeight,CellWidth 属性（PictureClip 控件）

返回 PictureClip 控件的 GraphicCell 属性的高度和宽度，以像素计算。

应用于

PictureClip 控件。

语法

*object*.CellHeight

*object*.CellWidth

*object* 是在“应用于”中指定的对象表达式。

说明

CellHeight 和 CellWidth 属性的值由 PictureClip 控件的 Cols 和 Rows 属性决定。例如，将图片分成 4×4 的单元，其 GraphicCell 的尺寸就是分成 8×8 单元的两倍。

请参阅

PicutreClip 控件，Cols, Rows 属性（PictureClip 控件），GraphicCell 属性（PictureClip 控件），Height, Width 属性（PictureClip 控件），Error 消息（PictureClip 控件）。

## Clip 属性（PictureClip 控件）

返回 PictureClip 控件中由 ClipX, ClipY, ClipWidth 和 ClipHeight 属性指定的点位图区域。该属性在运行时是只读的。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.Clip*

说明

使用该属性从选定的点位图中随机选取区域。

当在 Visual Basic 中给一个图片控件分配一个 Clip 图像时，确保该图片控件的 ScaleMode 属性设置为 3（像素）。必须设置为像素，因为定义剪裁区域的 ClipHeight 和 ClipWidth 属性度量单位是像素。

数据类型

Integer

请参阅

PictureClip 控件，ClipHeight 属性（PictureClip 控件），ClipWidth 属性（PictureClip 控件），ClipX 属性（PictureClip 控件），ClipY 属性（PictureClip 控件），GraphicCell 属性（PictureClip 控件），ScaleMode 属性。

## ClipHeight 属性（PictureClip 控件）

指定 Clip 属性所拷贝的点位图高度。该属性在设计时不可用。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.ClipHeight [=Height%]*

说明

该属性的单位是像素。

数据类型

Integer



请参阅

**ClipWidth** 属性（**PictureClip** 控件），**ClipX** 属性（**PictureClip** 控件），**ClipY** 属性（**PictureClip** 控件）。

## **ClipWidth** 属性（**PictureClip** 控件）

指定 **Clip** 属性所拷贝的点位图宽度。该属性在设计时不可用。

应用于

**PictureClip** 控件。

语法

*[form.]PictureClip.ClipWidth [=Width%]*

说明

该属性的单位是像素。

数据类型

**Integer**

请参阅

**ClipHeight** 属性（**PictureClip** 控件），**ClipX** 属性（**PictureClip** 控件），**ClipY** 属性（**PictureClip** 控件）。

## ClipX 属性（PictureClip 控件）

指定 **Clip** 属性所拷贝的点位图的左上角的 **x** 坐标。这个属性设计时不可用。

应用于

**PictureClip** 控件。

语法

*[form.]PictureClip.ClipX [=X%]*

说明

该属性的度量单位是像素。

数据类型

**Integer**

请参阅

Clip 属性 (PictureClip 控件), ClipHeight 属性 (PictureClip 控件), ClipWidth 属性 (PictureClip 控件), ClipY 属性 (PictureClip 控件)。

## ClipY 属性 (PictureClip 控件)

指定 Clip 属性所拷贝的点位图的左上角的 y 坐标。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.ClipY [=Y%]*

说明

该属性的度量单位是像素。

数据类型

Integer

请参阅

Clip 属性 (PictureClip 控件), ClipHeight 属性 (PictureClip 控件), ClipWidth 属性 (PictureClip 控件), ClipX 属性 (PictureClip 控件)。

## Cols, Rows 属性 (PictureClip 控件)

返回或设置图片中行或列的总数。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.Cols [=cols%]*

*[form.]PictureClip.Rows [=rows%]*

说明

使用这些属性将点位图分隔成统一的单元。使用 GraphicCell 属性指定单个的单元。

一个 PictureClip 控件必须至少有一行和一系列。

每个图片单元的高度由源点位图的高度除以行数，源点位图底部的余数则

剪裁掉。

宽度由源点位图的宽度除以列数，源点位图右边的余数则剪裁掉。

数据类型

Integer

请参阅

GraphicCell 属性 (PictureClip 控件)，Height,Width 属性 (PictureClip 控件)，Error 消息 (PictureClip 控件)。

Error 消息 (PictureClip 控件)

下表列出了 PictureClip 控件可以捕获的错误：

| 错误码 | 常量                 | 描述—消息解释           |
|-----|--------------------|-------------------|
| 7   | picOutOfMemory     | 内存用完              |
| 383 | picSetNotPermitted | 不能设置该控件的属性。属性是只读的 |
|     | picSetNotSupported |                   |
| 394 | picGetNotSupported | 属性是只写的            |

续表

| 错误码   | 常量                      | 描述—消息解释   |
|-------|-------------------------|---|
| 32000 | picInvalidPictureFormat | 不支持图片的格式。只能给 PictureClip 控件加载点位图文件（.bmp）  |
| 32001 | picDisplayContext       | 不能获取显示环境  |
| 32002 | picMemDevContext        | 不能获取内存设备环境  |
| 32003 | picBitmap               | 不能获取点位图   |
| 32004 | picSelBitmapObj         | 不能选择点位图对象   |
| 32005 | picIntPicStruct         | 不能分配内部图片结构  |
| 32006 | picBadIndex             | GraphicCell 索引无效<br>GraphicCell 属性的 <i>index</i> 参数超出了索引范围。该参数必须在 0 到 1(PicClip.Rows*PicClip.Cols)范围内 |
| 32007 | picNoPicSize            | 没有指定 GraphicCell 图片的尺寸  |
| 32008 | picBitmapsOnly          | GraphicCell 图片只允许点位图  |
| 32010 | picBadClip              | GraphicCell PictureClip 属性错   |
| 32012 | picGetObjFailed         | GetObject() Windows 函数失败<br>调用 Windows 的 GetObject()函数失败  |
| 32014 | pic                     | GlobalAlloc() Windows 函数失败<br>调用 Windows 的 GlobalAllic()函数失败  |

续表

| 错误码   | 常量                  | 描述—消息解释  |
|-------|---------------------|--|
| 32015 | picClipBounds       | 剪裁区域边界错误<br>ClipHeight 和 ClipWidth 属性指定的坐标在 PictureClip 控件包含的点位图边界之外 |
| 32016 | picCellTooSmall     | 单元尺寸太小（至少 1×1 个像素）   |
| 32017 | picRowNotPositive   | Rows 属性必须比 0 值大  |
| 32018 | picColNotPositive   | Cols 属性必须比 0 值大  |
| 32019 | picStretchXNegative | StretchX 属性不能是负数   |
| 32020 | picStretchYNegative | StretchY 属性不能是负数   |
| 32021 | picNoPicture        | 没有分配图片   |

请参阅

PictureClip 控件，ClipHeight 属性（PictureClip 控件），ClipWidth 属性（PictureClip 控件），ClipX 属性（PictureClip 控件），ClipY 属性（PictureClip 控件），Cols, Rows 属性（PictureClip 控件），GraphicCell 属性（PictureClip 控件），Picture 属性（PictureClip 控件），StretchX, StretchY 属性（PictureClip 控件），Miscellaneous 常量，Visual Basic 常量。

## GraphicCell 属性（PictureClip 控件）

一个一维矩阵，表示所有的图片单元。该属性在设计时不可用，在运行时  
是只读的。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.GraphicCell (index%)*

说明

- 使用 Rows 和 Cols 属性将一个图片分隔成统一的图片单元矩阵。
- GraphicCell 指定的单元有索引，开始是 0，从左到右，从上到下依次增加。
- 当读取该属性时，如果 Rows 或 Cols 属性设置为 0 或没有图片，则产生一个错误。

数据类型

Integer



请参阅

Cols, Rows 属性 (PictureClip 控件), Height, Width 属性 (PictureClip 控件), Error 消息 (PictureClip 控件)。

## Height, Width 属性 (PictureClip 控件)

返回控件中所包含的点位图的高度和宽度（以像素表示）。这些属性在设计时不可用。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.Height*

*[form.]PictureClip.Width*

说明

只有当控件包含了点位图时该属性才是有效的。

可在设计时使用 **Properties** 窗口给 **PictureClip** 控件加载点位图。在 **Visual Basic** 中，也可以在运行时使用 **LoadPicture** 函数给 **PictureClip** 控件加载点位图。

数据类型

Integer

请参阅

Cols, Rows 属性 (PictureClip 控件), GraphicCell 属性 (PictureClip 控件), Error 消息 (PictureClip 控件), LoadPicture 函数。

## Picture 属性 (PictureClip 控件)

该属性与标准的 Visual Basic Picture 属性一样，只是该属性只支持点位图 (.bmp) 文件。

应用于

PictureClip 控件。

请参阅

Error 消息 (PictureClip 控件), Picture 属性, Picture 属性 (ActiveX 控件)。

## StretchX, StretchY 属性（PictureClip 控件）

指定 Clip 属性创建的点位图目的地的尺寸。该属性在设计时不可用。

应用于

PictureClip 控件。

语法

*[form.]PictureClip.StretchX [=X%]*

*[form.]PictureClip.StretchY [=Y%]*

说明

使用这些属性定义剪裁的点位图所拷贝到的区域。当拷贝点位图时，该点位图放大或缩小以便适合 **StretchX** 和 **StretchY** 定义的区域。

**StretchX** 和 **StretchY** 都是以像素来度量的。

**注意：**在 Visual Basic 中，窗体和图片的缺省 ScaleMode 是 twip。对于显示来自 PictureClip 控件图片的所有控件，应将其 ScaleMode 设置为 3（像素）。

数据类型

Integer

请参阅

Clip 属性 (PictureClip 控件), Error 消息 (PictureClip 控件), PictureBox 控件, ScaleHeight, ScaleWidth 属性, ScaleMode 属性。

## ProgressBar 控件

ProgressBar 控件以块状填充一个矩形区域，表示操作的进度。

### 语法

ProgressBar

### 说明

- ProgressBar 控件监视一个操作结束的进度。

一个 ProgressBar 控件有一个范围以及当前的位置。范围代表了整个操作时间。当前位置表示了应用程序已完成该操作的进度。Max 和 Min 属性设置范围。Value 属性说明该范围中当前的位置。因为使用块来填充矩形框以表示进度，所以，填充的区域只是 Value 属性当前值的一个近似。根据控件的大小，Value 属性决定何时显示下一个填充块。

ProgressBar 控件的 Height 和 Width 属性确定填充控件所用的填充块的数量和大小。填充块越多，越能比较精确地表示操作的进度。要增加填充块的数量，应减少控件的 Height 或增大控件的 Width 属性。BorderStyle 属性的设置也影响到填充块的数量和大小。要带有边框，填充块就得小一些。

可以使用 **ProgressBar** 控件的 **Align** 属性自动将控件放置在窗体的上部或下部。

**提示：**要减少填充块的大小以便进度的增加近可能反映操作结果，应使得 **ProgressBar** 控件的宽度至少是高度的 12 倍。

下面的例子演示了如何使用 **ProgressBar** 控件显示对一个大矩阵的操作结果。在窗体上放置一个 **CommondButton** 控件和一个 **ProgressBar** 控件。**Align** 属性将控件放置在窗体的下部。**ProgressBar** 控件不显示文本。

```
Private Sub Command1_Click()  
    Dim Counter As Integer  
    Dim Workarea(250) As String  
    ProgressBar1.Min = LBound(Workarea)  
    ProgressBar1.Max = UBound(Workarea)  
    ProgressBar1.Visible = True
```

```
'Set the Progress's Value to Min.
```

```
    ProgressBar1.Value = ProgressBar1.Min
```

```
'Loop through the array.
```

```
    For Counter = LBound(Workarea) To UBound(Workarea)
```

```
        'Set initial values for each item in the array.
```

```
        Workarea(Counter) = "Initial value" & Counter
```

```
ProgressBar1.Value = Counter
Next Counter
ProgressBar1.Visible = False
ProgressBar1.Value = ProgressBar1.Min
End Sub
```

```
Private Sub Form_Load()
    ProgressBar1.Align = vbAlignBottom
    ProgressBar1.Visible = False
    Command1.Caption = "Initialize array"
End Sub
```

**发布须知：**ProgressBar 控件是 MSCOMCTL.OCX 文件中的 ActiveX 控件组的一部分。要在应用程序中使用 ProgressBar 控件，必须将其加入到你的工程中。在发布应用程序时，必须在用户的 Microsoft Windows System 或 System32 目录下安装 MSCOMCTL.OCX 文件。获取更多的信息，请参阅《Microsoft Visual Basic 6.0 程序员指南》。

属性：

Scrolling 属性, Orientation 属性 (ProgressBar 控件), Object 属性 (OLE Container), Negotiate 属性, TabIndex 属性, Align 属性, DragIcon 属性, DragMode 属性, hWnd 属性, MouseIcon 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件),

Value 属性（ActiveX 控件），Height, Width 属性（ActiveX 控件），Index 属性（ActiveX 控件），Left, Top 属性（ActiveX 控件），Tag 属性（ActiveX 控件），Object 属性（ActiveX 控件），Appearance 属性（ActiveX 控件），BorderStyle 属性（ActiveX 控件），Enabled 属性（ActiveX 控件），hWnd 属性（ActiveX 控件），MousePointer 属性（ActiveX 控件），Max, Min 属性（ActiveX 控件）。

## 方法

Drag 方法，Move 方法，ZOrder 方法，ShowWhatsThis 方法，OLEDrag 方法（ActiveX 控件）。

## 事件

DragDrop 事件，DragOver 事件，MouseDown, MouseUp 事件，MouseMove 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Click 事件（ActiveX 控件）。

## 请参阅

StatusBar 控件，HScrollBar, VScrollBar 控件，BorderStyle 属性，Height, Width 属性，Align 属性，Max, Min 属性（ScrollBar），Value 属性（ActiveX 控件），BorderStyle 属性（ActiveX 控件），加载 ActiveX 控件，使用 ProgressBar 控件。



# Orientation 属性（ProgressBar 控件）

返回或设置一个值，决定对象的方向（水平或垂直）。

应用于

ProgressBar 控件。

语法

*object.Orientation* [= *integer*]

Orientation 属性的语法有下面几部分：

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>integer</i> | 一个数值表达式，其值决定控件的方向，如设置值所示的 |

设置

“*integer*”的设置值为：

| 常量                      | 值 | 描述           |
|-------------------------|---|--------------|
| ccOrientationHorizontal | 0 | （缺省）控件是水平方向的 |
| ccOrientationVertical   | 1 | 控件是垂直方向的     |

# Scrolling 属性

返回或设置一个值，决定进度显示方式是连续的还是分段的。

应用于

ProgressBar 控件。

语法

*object.Scrolling* [= *integer*]

Scrolling 属性的语法有下面几部分：

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象       |
| <i>integer</i> | 一个数值表达式或常量，其值指定进度显示的外观，如设置值所示的 |

设置

*boolean* 的值为：

| 常量                  | 值 | 描述        |
|---------------------|---|-----------|
| ccScrollingStandard | 0 | 标准、分段的滚动条 |
| ccScrollingSmooth   | 1 | 连续的滚动条    |

# RichTextBox 控件

**RichTextBox** 控件允许用户输入和编辑文本的同时提供了比普通的 **TextBox** 控件更高级的格式特征。

## 语法

**RichTextBox**

## 说明

**RichTextBox** 控件提供了数个有用的特征，你可以在控件中安排文本的格式。要改变文本的格式，必须先选中该文本。只有选中的文本才可以编排字符和段落的格式。有了这些属性，就可以设置文本使用粗体，改变字体的颜色，创建超底稿和子底稿。也可以设置左右缩排或不缩排，从而调整段落的格式。

**RichTextBox** 控件可以打开和保存 **RTF** 文件或普通的 **ASCII** 文本文件。你可以使用控件的方法 (**LoadFile** 和 **SaveFile**) 直接读和写文件，或者在 **Visual Basic** 的文件输入/输出语句中使用控件的属性如 **SelRTF** 和 **TextRTF** 等。

**RichTextBox** 控件使用 **OLEObjects** 集合支持嵌入的对象。每个嵌入控件中的对象都表示为一个 **OLEObject** 对象。这允许文档中创建的控件可以包含其他

控件或文档。例如，可以创建一个包含 Microsoft Excel 报表、Microsoft Word 文档或任何在系统中注册的其他 OLE 对象的文档。要在 RichTextBox 控件中插入对象，可以简单地拖住一个文件（如使用 Windows 95 的 Explorer）或其他应用程序（如 Microsoft Word）中所用文件的加亮部分（选择部分），将其直接放到该 RichTextBox 控件上。

RichTextBox 控件支持剪贴板和 OLE 对象的 OLE 拖放功能。当从剪贴板粘贴对象时，就在当前的插入点插入该对象。如果对象是拖放到控件中，则插入点将跟随鼠标指针位置变动，直到释放开鼠标，然后在鼠标释放处插入对象。

要打印 RichTextBox 控件中的所有或部分文本，使用 SelPrint 方法。因为 RichTextBox 控件是数据绑定控件，可以将其与 Data 控件绑定到 Microsoft Access 数据库的 Binary 或 Memo 数据域，或其他数据库中类似的数据域（如 SQL Server 中的 TEXT 数据类型的数据域）。

RichTextBox 控件支持几乎所有的 TextBox 控件中的属性、事件和方法，如 MaxLength, MultiLine, ScrollBars, SelLength, SelStart 和 SelText。使用 TextBox 控件的应用程序很容易改为使用 RichTextBox 控件。然而，RichTextBox 控件并没有普通 TextBox 控件的 64K 字符能力的限制。

**注意：**要在应用程序中使用 RichTextBox 控件，必须给工程文件增加 Richtx32.OCX。当分布应用程序时，在用户的 Microsoft Windows SYSTEM 目录下安装 Richtx32.OCX 文件。其他信息请参阅《Microsoft Visual Basic 6.0 程序员指南》。

属性:

Appearance 属性 (ActiveX 控件), BackColor, ForeColor 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), HideSelection 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), SelLength, SelStart, SelText 属性 (ActiveX 控件), Text 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), DisableNoScroll 属性, SelHangingIndent, SelIndent, SelRightIndent 属性, BulletIndent 属性, SelAlignment 属性, SelBold, SelItalic, SelStrikethru, SelUnderline 属性, SelCharOffset 属性, SelColor 属性, SelFontName 属性, SelFontSize 属性, SelBullet 属性, SelTabCount, SelTabs 属性, SelRTF 属性, RightMargin 属性, AutoVerbMenu 属性 (RichTextBox 控件), SelProtected 属性, MaxLength 属性 (RichTextBox 控件), MultiLine 属性 (RichTextBox 控件), ScrollBars 属性 (RichTextBox 控件), OLEObjects 属性, DataChanged 属性, DataField 属性, FileName 属性 (RichTextBox 控件), TextRTF 属性, AutoVerbMenu 属性 (RichTextBox 控件), Top 属性, DataMember 属性, DataFormat 属性, DataBindings 属性, TabIndex 属性, DragIcon 属性, DragMode 属性, hWnd 属性, Locked 属性, Causes Validation 属性, MouseIcon 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Font 属性, Container 属性, ToolTipText 属性, DataSource 属性, WhatsThisHelpID 属性。

## 方法

OLEDrag 方法（ActiveX 控件），Refresh 方法（ActiveX 控件），Find 方法，GetLineFromChar 方法，LoadFile 方法，SaveFile 方法，SelPrint 方法，Span 方法，Upto 方法，SetFocus 方法，Drag 方法，Move 方法，ZOrder 方法，ShowWhatsThis 方法。

## 事件

SelChange 事件，DragDrop 事件，DragOver 事件，GotFocus 事件，KeyDown,KeyUp 事件，KeyPress 事件，LostFocus 事件，MouseDown, MouseUp 事件，MouseMove 事件，Validate 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Change 事件（ActiveX 控件），Click 事件（ActiveX 控件），DbtClick 事件（ActiveX 控件），KeyDown,KeyUp 事件（ActiveX 控件），KeyPress 事件（ActiveX 控件）。

## 请参阅

Error 消息（RichTextBox 控件），RichTextBox 控件常量，所支持的 RTF 代码（RichTextBox 控件），OLEObject 对象，OLEObjects 集合，OLEObjects 属性，TextBox 控件，使用 RichTextBox 控件。

# Add 方法（OLEObjects 集合）

给一个 OLEObjects 集合增加一个 OLEObject 对象。

应用于

RichTextBox 控件，OLEObjects 集合。

语法

*object.Add index, key, sourcedoc, class*

Add 方法的语法有如下几个部分：

| 部分               | 描述   |
|------------------|--|
| <i>object</i>    | 在“应用于”中指定的对象表达式  |
| <i>index</i>     | 可选参数。一个整数，标识对象集合中的一个成员。如果提供的话，新成员将加入到索引指定的成员后面   |
| <i>key</i>       | 可选参数。唯一字符串表达式，可用于访问集合中的成员。使用集合的 <b>Item</b> 方法访问 OLEObject 对象时， <b>index</b> 和 <b>key</b> 参数可以互换使用 |
| <i>sourcedoc</i> | 内嵌对象模板文档的文件名。 <b>RichTextBox</b> 控件不支持连接对象，因此文件的内容将被拷贝到 OLEObject 对象中。如果不指定源文档，必须使用一个 0 长度的字符串     |

续表

| 部分           | 描述  |
|--------------|---|
| <i>class</i> | 可选参数。要嵌入对象的 OLE 类名。该参数即是 OLE 在系统中注册的 ProgID。如果指定了源文档的文件名，则忽略该参数 |

说明

下面的代码给 RichTextBox 控件增加了一个 Microsoft Excel 表格，并将 Key 属性设置为 “SalesData”。

```
RichTextBox1.OLEObjects.Add , “SalesData” , , “Excel.Sheet.5”
```

当给集合增加对象时，该对象立即成为活动的，这样用户就可以给其增加数据了。

详见

OLEObject 对象，OLEObjects 集合，Item 方法。

请参阅

Key 属性（ActiveX 控件），OLEObject 对象，Index 属性（ActiveX 控件），RichTextBox 控件。

示例

下面的例子显示了一个对话框，可以找到一个 OLEObject 对象（一个点位



图），然后增加该 **RichTextBox** 控件。要使用本例，在窗体上放置一个 **CommonDialog** 控件和一个 **RichTextBox** 控件，将下面的代码拷贝到窗体的 **Declarations** 部分。运行本例，双击控件。

```
Private Sub RichTextBox1_DblClick()  
    With CommonDialog1  
        .Filter = ("bitmap (*.bmp)|*.bmp")  
        .ShowOpen  
    End With  
    RichTextBox1.OLEObjects. _  
        Add , , CommonDialog1.filename  
End Sub
```

## AutoVerbMenu 属性（RichTextBox 控件）

返回或设置一个值，该值确定在用户使用鼠标右键点击对象时是否在弹出式菜单中显示对象的动词。

应用于

**RichTextBox** 控件。

语法

```
object.AutoVerbMenu [=value]
```

AutoVerbMenu 属性的语法有如下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个 RichTextBox 控件的对象表达式   |
| <i>value</i>  | 一个布尔表达式，找到是否显示菜单，如“设置”中所示 |

设置

*value* 值的设置如下：

| 设置    | 描述                  |
|-------|---------------------|
| True  | 当用户点击鼠标右键时显示一个弹出式菜单 |
| False | 不显示弹出式菜单            |

说明

当该属性设置为 True 时，用鼠标右键点击 RichTextBox 控件并不产生 Click 和 MouseDown 事件。RichTextBox 控件的其他部分将产生正确的事件。

请参阅

OLEObject 对象, Click 事件, MouseDown, MouseUp 事件, Click 事件(ActiveX 控件)。

# BulletIndent 属性

当 SelBullet 设置为 True 时，返回或设置 RichTextBox 控件中缩排字符的数量。

应用于

RichTextBox 控件。

语法

*object*.BulletIndent [=integer]

BulletIndent 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 在“应用于”中指定的对象表达式   |
| <i>integer</i> | 一个整数，确定缩排字符的数量。这些属性使用的标尺模式单位是包含该 RichTextBox 控件的 Form 对象的标尺单位 |

说明

如果所选择的区域跨越了几个段而各段的边界设置又不相同，则 BulletIndent 属性的返回值是 Null。

请参阅

RichTextBox 控件，SelBullet 属性，Form 对象，Forms 集合。

## Class 属性（OLEObject 对象）

返回内嵌对象的类名。

应用于

RichTextBox 控件，OLEObject 对象。

语法

*object.Class*

**object** 是在“应用于”中指定的对象表达式。

说明

类名定义了对象的类型。支持 OLE 或 Automation 的应用程序使用下面的语法之一标识它们对象的类名：

*application.objecttype.version*

*objecttype.version*

OLE 类名的语法有如下几个部分：

| 部分                 | 描述                |
|--------------------|-------------------|
| <i>application</i> | 支持该对象的应用程序名       |
| <i>objecttype</i>  | 在对象库中定义的对象名       |
| <i>version</i>     | 支持该对象的应用程序或对象的版本号 |

例如，Microsoft Excel 5.0 支持多个对象，包括报表和图表，它们的类名分别是 Excel.Sheet.5 和 Excel.Chart.5。Microsoft WordArt 2.0 支持单个对象，类名是 MSWordArt.2。

注意：某些 OLE 编程文档使用 ProgID 应用类名。

使用 OLEObjects 集合的 Add 方法创建的对象将自动设置 OLEObject 对象的 Class 属性。

请参阅

RichTextBox 控件，OLEObjects 集合，Add 方法（OLEObjects 集合）。

## DisableNoScroll 属性

返回或设置一个值，该值确定在 RichTextBox 控件中是否禁止滚动杆。

应用于

RichTextBox 控件。

语法

*object*.DisableNoScroll [=*boolean*]

DisableNoScroll 属性的语法有如下几个部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 在“应用于”中指定的对象表达式            |
| <i>boolean</i> | 一个布尔表达式，找到是否禁止滚动杆，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述         |
|-------|------------|
| False | （缺省）显示滚动杆  |
| True  | 滚动杆被禁止（变灰） |

说明

当 ScrollBars 属性设置为 0（没有）时忽略 DisableNoScroll 属性。然而，当 ScrollBars 设置为 1（水平），2（垂直），3（二者）时，如果 RichTextBox 控件中只有很少几行或一行中有很少的字符，则会自动禁止滚动杆。

请参阅

RichTextBox 控件， ScrollBars 属性。

## DisplayType 属性（OLEObject 对象）

返回或设置一个值，指明对象显示其内容还是一个图标。

应用于

RichTextBox 控件， OLEObject 对象。

语法

*object*.DisplayType [=*value*]

DisplayType 属性的语法有如下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式                  |
| <i>value</i>  | 一个整数或常量，指定对象显示内容或图标，如“设置”<br>中所示 |

## 设置

*value* 值的设置如下：

| 设置                | 值 | 描述                                     |
|-------------------|---|--|
| rtfDisplayContent | 0 | 内容。当 RichTextBox 控件包含一个对象时，在控件中显示对象的数据 |
| rtfDisplayIcon    | 1 | 图标。当 RichTextBox 控件包含一个对象时，在控件中显示对象的图标 |

## 说明

给 OLEObjects 集合增加了一个对象后，使用 DisplayType 属性改变对象的显示方式。

## 请参阅

RichTextBox 控件，RichTextBox 控件常量，OLEObjects 集合。

## DoVerb 方法（OLEObject 对象）

为某个操作如编辑打开一个对象。



应用于

RichTextBox 控件，OLEObject 对象。

语法

*object.DoVerb (verb)*

DoVerb 方法的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 在“应用于”中指定的对象表达式  |
| <i>verb</i>   | 可选参数。在 RichTextBox 控件中执行 OLEObject 对象的动词。如果没有指定，则执行缺省动词。该参数的值可以是所有对象都支持的标准动词或者 ObjectVerbs 属性矩阵的一个索引 |

说明

DoVerb 方法执行指定 OLEObject 对象的一个动词。verb 参数是 ObjectVerbs 属性矩阵中某个动词的索引或下表列出的某个标准动词。

每个对象都可以支持自己的一套动词。下表列出了每个对象都应该支持的标准动词：

| 常量           | 描述      |
|--------------|---------|
| vbOLEPrimary | 对象的缺省动作 |

| 常量                    | 描述   |
|-----------------------|--|
| vbOLEShow             | 1 激活对象进行编辑。如果创建该对象的应用程序支持就地(in-place)激活，该对象就在 RichTextBox 控件中被激活 |
| vbOLEOpen             | 2 在单独的应用程序窗口中打开该对象。如果创建该对象的应用程序支持就地激活，就在该对象自己的窗口中激活它             |
| vbOLEHide             | 3 对于嵌入的对象，隐藏创建该对象的应用程序   |
| vbOLEUIActivate       | 4 如果对象支持就地激活，则就地激活该对象，显示用户界面工具。如果对象不支持就地激活，则不激活该对象，并产生一个错误       |
| vbOLEInPlaceActivate  | 5 如果用户将角度移动到嵌入对象，则为该对象创建一个窗口，准备进行编辑。如果对象不支持单鼠标点击激活，则产生一个错误       |
| VbOLEDiscardUndoState | 6 当激活对象进行编辑并扔掉对象应用程序可以恢复的所有修改记录时使用该常量                            |

注意：这些动词或许没有列在 ObjectVerbs 属性矩阵中。

请参阅

**RichTextBox** 控件常量，**OLEObjects** 集合，**ObjectVerbs** 属性（**OLEObject** 对象）。

## FileName 属性（RichTextBox 控件）

返回或设置在设计时加载到 **RichTextBox** 控件中的文件名。

应用于

**RichTextBox** 控件。

语法

*object*.FileName [=Path]

FileName 属性的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 需求。一个对象表达式，其值等于“应用于”中的一个对象 |
| <i>path</i>   | 字符串。有效文件的路径                |

## 设置

只能为该属性指定文本文件名或有效的.RTF 文件名。

## 请参阅

LoadFile 方法, SaveFile 方法, 支持的 RTF 代码 (RichTextBox 控件), PathChange 事件, PatternChange 事件, List 属性, ListIndex 属性, Drive 属性, Pattern 属性, Path 属性。

## Find 方法

在 RichTextBox 控件中搜索指定的字符串。

## 应用于

RichTextBox 控件。

## 语法

*object*.Find (*string*, *start*, *end* , *options*)

Find 方法的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个 RichTextBox 控件的对象表达式  |
| <i>string</i>  | 在控件中搜索的字符串表达式  |
| <i>start</i>   | 可选参数。一个整数，指定开始搜索的字符索引位置。<br>RichTextBox 控件中的每个字符都有一个整数索引位置，用于唯一标识该字符。RichTextBox 控件中的第一个字符的索引是 0 |
| <i>end</i>     | 可选参数。一个整数，指定结束搜索的字符索引位置  |
| <i>options</i> | 可选参数。一个或多个索引常量，指定选项特征，如“设置”中所示   |

## 设置

option 值的设置如下：

| 常量             | 值 | 描述                |
|----------------|---|-------------------|
| rtfWholeWord   | 2 | 确定匹配是整个单词还是单词的一部分 |
| rtfMatchCase   | 4 | 确定匹配是否使用大小写       |
| rtfNoHighlight | 8 | 确定匹配的字符是否加亮显示     |

你可以使用 Or 操作符组合这些条件。

# 说明

如果指定所搜索的文本，则 Find 方法加亮指定的文本，并返回加亮的第一个字符的索引。如果指定的文本没有找到，Find 方法返回-1。

如果使用 Find 方法时没有 rtfNoHightlight 选项，并且 HideSelection 属性设置为 True，RichTextBox 控件也没有输入焦点，则控件仍加亮显示找到的文本。后续的 Find 方法将只搜索加亮的文本，直到移动了插入点。

Find 方法的行为还取决于 start 和 end 参数的设置。下表描述了这些可能的行为：

| 起点 | 终点 | 搜索行为   |
|----|----|--|
| 指定 | 指定 | 从指定的起点和终点进行搜索                                |
| 指定 | 省略 | 从指定的起点到 RichTextBox 控件的结尾进行搜索                |
| 省略 | 指定 | 从当前插入点到指定的终点进行搜索                             |
| 省略 | 省略 | 如果有选择的文本，则搜索所选择的文本。否则搜索整个 RichTextBox 控件中的文本 |

# 请参阅

RichTextBox 控件常量。

# 示例

下面的例子根据在 TextBox 控件中输入的文本对 RichTextBox 控件进行搜

索。如果找到指定的字符串，则显示一个消息框，指出包含指定单词的行号。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件，一个 **CommandButton** 控件，一个 **TextBox** 控件。给 **RichTextBox** 控件加载一个文件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，在 **TextBox** 控件中输入文本，然后点击 **CommandButton** 控件。

```
Private Sub Command1_Click()  
    Dim FoundPos As Integer  
    Dim FoundLine As Integer  
    ' Find the text specified in the TextBox control.  
    FoundPos = RichTextBox1.Find(Text1.Text, , , rtfWholeWord)  
  
    ' Show message based on whether the text was found or not.  
  
    If FoundPos <> -1 Then  
        ' Returns number of line containing found text.  
        FoundLine = RichTextBox1.GetLineFromChar(FoundPos)  
        MsgBox "Word found on line " & CStr(FoundLine)  
    Else  
        MsgBox "Word not found."  
    End If  
End Sub
```

# GetLineFromChar 方法

返回 RichTextBox 控件中包含指定字符位置的行号。

应用于

RichTextBox 控件。

语法

*object*.GetLineFromChar ( *charpos*)

GetLineFromChar 方法的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 在“应用于”中指定的对象表达式                           |
| <i>charpos</i> | 长整数，指定要标识的字符索引。RichTextBox 控件中第一个字符的索引是 0 |

说明

可以使用 GetLineFromChar 方法在 RichTextBox 控件中找出在某个字符位置上的行。有时是需要这样做的，因为每行中的字符数可以变化，由文本的索引标识很难找到该字符所在行号。



请参阅

`SelCharOffset` 属性。

示例

下面的例子根据在 `TextBox` 控件中输入的文本对 `RichTextBox` 控件进行搜索。如果找到指定的字符串，则显示一个消息框，指出包含指定单词的行号。要使用本例，在一个窗体上放置一个 `RichTextBox` 控件，一个 `CommandButton` 控件，一个 `TextBox` 控件。给 `RichTextBox` 控件加载一个文件，将下面的代码拷贝到窗体的 `Declarations` 段中。运行本例，在 `TextBox` 控件中输入文本，然后点击 `CommandButton` 控件。

```
Private Sub Command1_Click()  
    Dim FoundPos As Integer  
    Dim FoundLine As Integer  
    ' Find the text specified in the TextBox control.  
    FoundPos = RichTextBox1.Find(Text1.Text, , , rtfWholeWord)  
    ' Show message based on whether the text was found or not.  
  
    If FoundPos <> -1 Then  
        ' Returns number of line containing found text.  
        FoundLine = RichTextBox1.GetLineFromChar(FoundPos)  
        MsgBox "Word found on line " & CStr(FoundLine)
```

```
Else
    MsgBox "Word not found."
End If
End Sub
```

## LoadFile 属性

给 RichTextBox 控件加载一个.RTF 文件或文本文件。

应用于

RichTextBox 控件。

语法

*object.LoadFile pathname, filetype*

LoadFile 方法的语法有如下几个部分：

| 部分              | 描述                              |
|-----------------|---------------------------------|
| <i>object</i>   | 在“应用于”中指定的对象表达式                 |
| <i>pathname</i> | 字符串表达式，指定要加载文件的路径名和文件名          |
| <i>filetype</i> | 可选参数。一个整数或常量，指定加载文件的类型，如“设置”中所示 |

## 设置

*filetype* 值的设置如下：

| 常量      | 值 | 描述                          |
|---------|---|-----------------------------|
| rtfRTF  | 0 | （缺省）RTF。要加载的文件必须是有效的.RTF 文件 |
| rtfText | 1 | 文本。RichTextBox 控件加载任何的文本文件  |

## 说明

当使用 LoadFile 方法加载一个文件时，所加载文件的内容替换 RichTextBox 控件中的文本内容。这将导致 Text 和 RTFText 属性值发生变化。

也可以使用 Visual Basic 中的 Input 函数和 RichTextBox 控件的 TextRTF 和 SelRTF 属性读取一个.RTF 文件。例如，可以使用下面的代码将一个.RTF 文件的内容加载到一个 RichTextBox 控件中：

```
Open "mytext.rtf" For Input As 1  
RichTextBox1.TextRTF = Input$(LOF(1), 1)
```

## 请参阅

FileName 属性（RichTextBox 控件），SaveFile 方法，SelRTF 属性，TextRTF 属性，RichTextBox 控件常量，支持的 RTF 码（RichTextBox 控件）。

## 示例

下面的例子显示了一个对话框选择一个 .RTF 文件，然后将该文件加载到 RichTextBox 控件中。要使用本例，在窗体上放置一个 RichTextBox 控件，一个 CommandButton 控件和一个 CommonDialog 控件。将下面的代码拷贝到窗体的 General Declarations 段中。运行本例。

```
Private Sub Command1_Click()  
    CommonDialog1.Filter = "Rich Text Format files|*.rtf"  
    CommonDialog1.ShowOpen  
    RichTextBox1.LoadFile CommonDialog1.FileName, rtfRTF  
End Sub
```

## Locked 属性（RichTextBox 控件）

返回或设置一个值，指明 RichTextBox 控件中的内容是否可以编辑。

应用于

RichTextBox 控件。

语法

*object*.Locked [=boolean]

Locked 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个 RichTextBox 控件的对象表达式                     |
| <i>boolean</i> | 一个布尔表达式，指定是否可以编辑 RichTextBox 控件的内容，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | 可以固定和选择控件中的文本，但不能进行编辑。程序仍可改变 Text 属性来修改文本 |
| False | （缺省）可以编辑 RichTextBox 控件中的文本               |

请参阅

Text 属性，ReadOnly 属性，Text 属性（ActiveX 控件）。

MaxLength 属性（RichTextBox 控件）

返回或设置一个值，指明 RichTextBox 控件中是否有最大字符数的限制，如果有指明该最大值。

应用于

RichTextBox 控件。

语法

*object*.MaxLength [=long]

MaxLength 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个 RichTextBox 控件的对象表达式  |
| <i>long</i>   | 长整数，指定用户可以在该 RichTextBox 控件中输入的最大字符数。MaxLength 属性的缺省值是 0，意味着没有最大值的限制。比 0 大的任何值都指明了最大的字符数 |

说明

使用 MaxLength 属性设置用户可以在一个 RichTextBox 控件中输入的最大字符数。

如果在程序代码中给 RichTextBox 控件分配的文本超过了 MaxLength 属性的设置，不产生错误。然而，Text 属性的值只是规定的最大字符数，其他字符都被截去了。改变该属性并不影响 RichTextBox 控件的当前内容，但会影响后续对内容的修改。

请参阅

MultiLine 属性，Text 属性，MultiLine 属性，Text 属性（ActiveX 控件）。

## MultiLine 属性（RichTextBox 控件）

返回或设置一个值，指明 RichTextBox 控件是否可以接受并显示多行文本。在运行时该属性是只读的。

应用于

RichTextBox 控件。

语法

*object*.MultiLine

*object* 是一个 RichTextBox 控件的对象表达式。

设置

MultiLine 属性的设置如下：

| 设置    | 描述                     |
|-------|------------------------|
| True  | 允许有多行文本                |
| False | （缺省）忽略回车键，将所有数据都限制在一行上 |

## 说明

在用户输入的文本超出了文本框时，多行的 **RichTextBox** 控件将把文本分行显示。

你也可以使用 **ScrollBars** 属性给大的 **RichTextBox** 控件添加滚动杆。如果没有指定 **HScrollBar** 控件（水平滚动杆），则多行 **RichTextBox** 控件将自动分行显示文本。

**注意：**在没有缺省按钮的窗体中，在多行 **RichTextBox** 控件中按回车键将把输入焦点移到下一行。如果有缺省按钮，必须使用 **CTRL+ENTER** 将输入焦点移到下一行。

## 请参阅

**ScrollBars** 属性，**Default** 属性。



# ObjectVerbFlags 属性（OLEObject 对象）

返回 ObjectVergs 矩阵中每个单词的菜单状态。

应用于

RichTextBox 控件，OLEObject 对象。

语法

*object*.ObjectVerbFlags (*value*)

ObjectVerbFlags 属性的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |
| <i>value</i>  | 数字表达式，指定矩阵中的元素  |

返回值

ObjectVerbFlags 属性的返回值如下：

| 常量                | 值      | 描述           |
|-------------------|--------|--------------|
| vbOLEFlagChecked  | &H0008 | 菜单项选中        |
| vbOLEFlagDisabled | &H0002 | 菜单项禁止（但没有变灰） |

续表

| 常量                 | 值      | 描述        |
|--------------------|--------|-----------|
| vbOLEFlagEnabled   | &H0000 | 菜单项使能     |
| vbOLEFlagGrayed    | &H0001 | 菜单项变灰     |
| vbOLEFlagSeparator | &H0800 | 菜单项是一个分隔条 |

注意：这些常量在 Visual Basic 的对象库中，在 Object Browser 可以找到。

说明

ObjectVerbs 矩阵中的第一个动词是缺省动词。该矩阵中的其他动词都可以显示在菜单上。ObjectVerbFlags 矩阵变化了 ObjectVerbs 矩阵中每个动词的菜单项状态（如变灰，检查等等）。

当显示变化了对象动词的菜单时，检查该属性的值以查看如何显示菜单项。

请参阅

OLEObjects 集合，ObjectVerbs 属性（OLEObject 对象），ObjectVerbsCount 属性（OLEObject 对象）。

# ObjectVerbs 属性（OLEObject 对象）

返回一个对象指出的动词列表。

应用于

RichTextBox 控件， OLEObject 对象。

语法

*object.ObjectVerbs(value)*

ObjectVerbs 属性的语法有如下几个部分：

| 部分            | 描述               |
|---------------|------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式  |
| <i>value</i>  | 一个数字表达式，指定矩阵中的元素 |

说明

ObjectVerbs 是基于 0 的字符串矩阵。使用该属性和 ObjectVerbsCount 属性可以获取该对象支持的动词。这些动词用于确定使用 DoVerb 方法激活对象时的这些动作。矩阵中的动词列表与对象有关，并且也受当前状态的影响。

每个对象都可以支持自己的一套动词。下表列出了每个对象都应该支持的

标准动词:

| 常量                    | 值  | 描述  |
|-----------------------|----|---|
| vbOLEPrimary          | 0  | 对象的缺省动作   |
| vbOLEShow             | -1 | 激活对象进行编辑。如果创建该对象的应用程序支持就地激活, 该对象就在 RichTextBox 控件中被激活       |
| vbOLEOpen             | -2 | 在单独的窗口中打开该对象。如果创建该对象的应用程序支持就地激活, 则该对象在自己的窗口中打开              |
| vbOLEHide             | -3 | 对于嵌入的对象, 隐藏创建该对象的应用程序                                       |
| vbOLEUIActivate       | -4 | 如果对象支持就地激活, 则就地激活该对象, 显示用户界面工具如果对象不支持就地激活, 则不激活该对象, 并产生一个错误 |
| vbOLEInPlaceActivate  | -5 | 如果用户将角度移动到嵌入对象, 则为该对象创建一个窗口, 准备进行编辑。如果对象不支持单鼠标点击激活, 则产生一个错误 |
| vbOLEDiscardUndoState | -6 | 当激活该对象扔弃所有的对象应用程序可以恢复的修改记录时使用                               |

注意: 在 ObjectVerbs 属性矩阵中或许不包含这些动词。

ObjectVerbs 矩阵中的第一个动词 ObjectVerbs(0)是缺省动词。除非特别指

定，使用该动词激活对象。

矩阵中的其他动词可以显示在菜单中。如果在菜单中可以正确地显示缺省动词，则缺省动词在 **ObjectVerbs** 矩阵中有两个入口。

显示对象的应用程序通常都在 **Edit** 菜单中包含一个 **Object** 命令。当用户选择 **Edit Object** 时，菜单就显示对象的动词。在运行时使用 **ObjectVerbs**, **ObjectVerbsCount** 和 **ObjectVerbFlags** 属性可以创建这样的菜单。

当用户在对象上点击鼠标右键时，如果想自动显示 **ObjectVerbs** 矩阵中的动词，可以将 **AutoVerbMenu** 属性设置为 **True**。

请参阅

**RichTextBox** 控件常量, **AutoVerbMenu** 属性(**RichTextBox** 控件), **OLEObjects** 集合, **ObjectVerbsCount** 属性(**OLEObject** 对象), **ObjectVerbFlags** 属性(**OLEObject** 对象), **DoVerb** 方法(**OLEObject** 对象)。

## **ObjectVerbsCount** 属性 (**OLEObject** 对象)

返回对象支持的动词数。

应用于

RichTextBox 控件，OLEObject 对象。

语法

*object*.ObjectVerbsCount

*object* 是在“应用于”中指定的对象表达式。

说明

使用该属性确定 ObjectVerbs 属性矩阵中的元素个数。

一个对象支持的动词列表可以变化，这取决于对象的状态。

请参阅

OLEObjects 集合，ObjectVerbs 属性（OLEObject 对象）。

## OLEObject 对象

一个 OLEObject 对象表示一个可插入 RichTextBox 控件中的对象。

说明

RichTextBox 控件使得你可以给 RTF 文件加入一个可插入对象。可插入对

象由 **OLEObject** 对象表示。

在 **RichTextBox** 控件中，每个插入对象都算作一个字符。即当从控件的开头计算字符个数时，对象将占据一个字符的位置。**OLE** 对象可以被用户加亮、剪切或拷贝，但不支持任何的 **Selxxx** 属性（如 **SelBold**, **SelItalic** 等等）。

每个对象都支持一个环境菜单，包含了标准的 **Cut**, **Copy**, **Paste** 和 **Delete** 命令，以及 **Open** 和 **Edit**。**Open** 命令导致对象的应用程序在自己的窗口中打开，这样就可以编辑对象了。如果对象支持就地激活，则 **Edit** 命令将导致对象被就地激活。

在运行时，可以使用 **Add** 方法手工给 **OLEObjects** 集合增加 **OLEObject** 对象，或者从 **Windows Explorer** 中拖住一个对象放到 **RichTextBox** 控件上。

## 属性

**Enabled** 属性（**ActiveX** 控件），**Key** 属性（**ActiveX** 控件），**Height**, **Width** 属性（**ActiveX** 控件），**Index** 属性（**ActiveX** 控件），**Object** 属性（**ActiveX** 控件），**Class** 属性（**OLEObject** 对象），**DisplayType** 属性（**OLEObject** 对象），**ObjectVerbs** 属性（**OLEObject** 对象），**ObjectVerbsCount** 属性（**OLEObject** 对象），**ObjectVerbFlags** 属性（**OLEObject** 对象）。

## 方法

**FetchVerbs** 方法（**ActiveX** 控件），**DoVerb** 方法（**OLEObject** 对象），**Clear**

方法（ActiveX 控件）。

请参阅

OLEObjects 集合，Add 方法（OLEObjects 集合），OLEObjects 属性。

## OLEObjects 集合

一个 OLEObjects 集合包含了 OLEObject 对象的集合。

语法

*object*.OLEObjects (*index* )

*object*.OLEObjects.Item(*index*)

OLEObjects 集合的语法有如下几个部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 在“应用于”中指定的对象表达式                        |
| <i>index</i>  | Index 属性或 Key 属性的值，唯一标识一个 OLEObject 对象 |

说明

在 RichTextBox 控件中创建的每个嵌入 OLE 对象都表示在 OLEObjects 集合中。在运行时，可以使用 Add 方法手工给 OLEObjects 集合增加 OLEObject



对象，或者从 Windows Explorer 中拖住一个对象放到 RichTextBox 控件上。

OLEObjects 集合是标准的对象集合，支持 Add, Item, Remove 方法以及 Count 属性。

## 属性

Count 属性（ActiveX 控件），Item 属性（ActiveX 控件）。

## 方法

Clear 方法（ActiveX 控件），Remove 方法（ActiveX 控件），Add 方法（OLEObjects 集合）。

## 请参阅

RichTextBox 控件常量，OLEObject 对象，OLEObjects 属性，Remove 方法。

## OLEObjects 属性

返回一个对 OLEObjects 集合的引用。

## 应用于

RichTextBox 控件。

语法

*object.OLEObjects*

*object* 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

说明

OLEObjects 集合包含可以插入到 RichTextBox 控件，包括文本、位图或其他文件的对象。

请参阅

OLEObject 对象，OLEObjects 集合。

## RightMargin 属性

返回或设置 RichTextBox 控件中文本的右边界。

应用于

RichTextBox 控件。

语法

*object.RightMargin* [=value]

**RightMargin** 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个 <b>RichTextBox</b> 控件的对象表达式                               |
| <i>value</i>  | 一个数字表达式，指定文本右端到 <b>RichTextBox</b> 控件右边框的缩排尺寸，以 twip 表示，如下所述 |

## 说明

**RightMargin** 属性用于设置文本分行、集中和缩排的右边界限制。集中是根据 **RichTextBox** 控件文本的最左边部分（不包含边框）和 **RightMargin** 属性进行的。并且，当设置 **SelRightIndent** 属性时，要看 **RightMargin** 属性的当前设置。

**RightMargin** 属性的缺省值是 0，控件将把文本分行设置为与 **RichTextBox** 控件的最右端部分相等，从而所有的文本都可视。

**注意：**在计算 **RightMargin** 属性的值时，必须考虑到 **RichTextBox** 控件的边框，如果其 **BorderStyle** 属性设置为 **rtfFixedSingle**。

## 请参阅

**SelHangingIndent**, **SelIndent**, **SelRightIndent** 属性，**RichTextBox** 控件常量。

# SaveFile 方法

将 RichTextBox 控件中的内容保存到一个文件中。

应用于

RichTextBox 控件。

语法

*object*.SaveFile ( *pathname*, *filetype* )

SaveFile 方法的语法有如下几个部分：

| 部分              | 描述                              |
|-----------------|---------------------------------|
| <i>object</i>   | 在“应用于”中指定的对象表达式                 |
| <i>pathname</i> | 字符串表达式，指定要保存文件的路径名和文件名          |
| <i>filetype</i> | 可选参数。一个整数或常量，指定保存文件的类型，如“设置”中所示 |

设置

*filetype* 值的设置如下：

| 常量      | 值 | 描述                          |
|---------|---|-----------------------------|
| rtfRTF  | 0 | （缺省）RTF。要保存的文件必须是.RTF 文件    |
| rtfText | 1 | 文本。RichTextBox 控件将内容保存为文本文件 |

## 说明

也可以使用 Visual Basic 中的 Write 函数和 RichTextBox 控件的 TextRTF 和 SelRTF 属性写一个.RTF 文件。例如，可以使用下面的代码将一个 RichTextBox 控件中选择的内容保存到一个.RTF 文件中：

```
Open "mytext.rtf" For Output As 1
Print #1, RichTextBox1.SelRTF
```

## 请参阅

LoadFile 方法，SelRTF 属性，TextRTF 属性，RichTextBox 控件常量，支持的 RTF 码（RichTextBox 控件）。

## 示例

下面的例子显示了一个对话框选择一个.RTF 文件，然后将 RichTextBox 控件中内容保存到该文件中。要使用本例，在窗体上放置一个 RichTextBox 控件，一个 CommandButton 控件和一个 CommonDialog 控件。将下面的代码拷贝到 CommandButton 控件的 Click 事件中。运行本例。

```
Private Sub Command1_Click()  
    CommonDialog1.ShowSave  
    RichTextBox1.SaveFile CommonDialog1.FileName, rtfRTF  
End Sub
```

## ScrollBars 属性（RichTextBox 控件）

分行或设置一个值，该值指定 RichTextBox 控件是否有水平或垂直滚动杆。该属性在运行时是只读的。

应用于

RichTextBox 控件。

语法

*object*.ScrollBars

*object* 是一个 RichTextBox 控件的对象表达式。

设置

ScrollBars 属性的设置如下：

| 常量            | 值 | 描述        |
|---------------|---|-----------|
| rtfNone       | 0 | （缺省）没有    |
| rtfHorizontal | 1 | 只有水平滚动杆   |
| rtfVertical   | 2 | 只有垂直滚动杆   |
| rtfBoth       | 3 | 有水平和垂直滚动杆 |

### 说明

对于 RichTextBox 控件设置为 1（水平）、2（垂直）、3（二者），必须将 MultiLine 属性设置为 True。

在运行时，Microsoft Windows 操作系统将自动给 RichTextBox 控件增加一个键盘接口，用户可以使用箭头键（UP ARROW，DOWN ARROW，LEFT ARROW，RIGHT ARROW），HOME 键和 END 键等对控件进行操作。

只有当 RichTextBox 控件的内容超出了边框范围时才显示滚动杆。如果 ScrollBars 设置为 False，则控件没有滚动杆，不管其内容如何。

### 请参阅

RichTextBox 控件常量，MultiLine 属性，WordWrap 属性。

# SelAlignment 属性

返回或设置一个值，该值确定 RichTextBox 控件中段落的对齐方式。

应用于

RichTextBox 控件。

语法

*object*.SelAlignment [=*value*]

SelAlignment 属性的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式            |
| <i>value</i>  | 一个整数或常量，指定段落的对齐方式，如“设置”中所示 |

设置

*value* 值的设置如下：

| 常量   | 值 | 描述                          |
|------|---|-----------------------------|
| null |   | 没有。目前的选择跨越了多个段，但各段的对齐格式都不相同 |



续表

| 常量        | 值 | 描述          |
|-----------|---|-------------|
| rtfLeft   | 0 | （缺省）左。段落左对齐 |
| rtfRight  | 1 | 右。段落右对齐     |
| rtfCenter | 2 | 中间。段落中间对齐   |

说明

**SelAlignment** 属性的值决定了当前所选择的文本中所有段落的当前方式，或者如果没有选择的文本，则是包含插入点的文本段。

在运行中读取该属性值时，如果要区分 0 和 Null 值，可以使用 **IsNull** 函数以及 **If...Then...Else** 语句。例如：

```
If IsNull(RichTextBox1.SelAlignment) = True Then
    ' Code to run when selection is mixed.
ElseIf RichTextBox1.SelAlignment = 0 Then
    ' Code to run when selection is left aligned.
```

End If

请参阅

**SelHangingIndent**, **SelIndent**, **SelRightIndent** 属性, **SelBullet** 属性, **SelTabCount**, **SelTabs** 属性, **RichTextBox** 控件常量。

## 示例

下面的例子使用了一个 **OptionButton** 控件矩阵改变 **RichTextBox** 控件中所选择文本段落的对齐格式，但只有当有选择的文本时才这样做。矩阵中控件的索引与 **SelAlignment** 属性的设置相对应。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件和 3 个 **OptionButton** 控件。**OptionButton** 控件的名字都设置为一样，但 **Index** 属性分别设置为 0,1, 2。将下面的代码拷贝到 **OptionButton** 控件的 **Click** 事件中。然后运行本例。

```
Private Sub Option1_Click(Index As Integer)
    If RichTextBox1.SelLength > 0 Then
        RichTextBox1.SelAlignment = Index
    End If
End Sub
```

## **SelBold, SelItalic, SelStrickethru, SelUnderline** 属性

返回或设置 **RichTextBox** 控件中所选择文本的字体风格。字体风格包含下面的格式：粗体，斜体，中间划横线以及带下划线。该属性在设计时不可用。

## 应用于

**RichTextBox** 控件。

# 语法

```
object.SelBole [=value]
object.SelItalic [=vlaue]
object.SelStrickethru [=value]
object.SelUnderline [=value]
```

SelBold, SelItalic, SelStrickethru, SelUnderline 属性的语法有如下几个部分：

| 部分            | 描述                         |
|---------------|----------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式            |
| <i>value</i>  | 一个布尔表达式或常量，指定字符风格，如“设置”中所示 |

# 设置

*value* 值的设置如下：

| 设置    | 描述                           |
|-------|------------------------------|
| Null  | 没有。当前选择的文本或插入点之后的字符的字体风格有多种  |
| True  | 选中的所有字符或插入点之后的字符设置为相应的风格     |
| False | （缺省）选择中的字符或插入点之后的字符不设置为指定的风格 |

# 说明

该属性与 Font 对象的 Bold, Italic ,Strickethru 和 Underline 属性差不多。RichTextBox 控件有一个 Font 属性，因此可以通过该控件的 Font 对象的属性设

置字体风格。使用这些属性设置选择文本中或插入点之后的字符风格。

通常，在应用程序的工具条中创建对应的按钮，用于对每种字体风格进行控制。

要在运行时读取并区分 `Null` 和 `False` 值，可以使用 `IsNull` 函数以及 `If...Then...Else` 语句。例如：

```
If IsNull(RichTextBox1.SelBold) = True Then
    ' Code to run when selection is mixed.
ElseIf RichTextBox1.SelBold = False Then
    ' Code to run when selection is not bold.
...
End If
```

请参阅

`SelCharOffset` 属性，`SelColor` 属性，`SelFontName` 属性，`SelFontSize` 属性。

## SelBullet 属性

返回或设置一个值，该值确定 `RichTextBox` 控件中包含当前选择文本或插入点的段落是否具有布告板风格。在设计时不可用。

应用于

RichTextBox 控件。

语法

*object.SelBullet [=value]*

SelBullet 属性的语法有如下几个部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式             |
| <i>value</i>  | 一个整数或常量，确定段落的布告板风格，如“设置”中所示 |

设置

value 值的设置如下：

| 设置    | 描述                        |
|-------|---------------------------|
| Null  | 没有。选择包含了多个段落，各段落的布告板风格不一致 |
| True  | 选择中的段落有布告板风格              |
| False | （缺省）选择中的段落没有布告板风格         |

说明

使用 SelBullet 属性建立 RichTextBox 控件中的布告板条目列表。

要在运行时读取并区分 Null 和 False 值，可以使用 IsNull 函数以及 If...Then...Else 语句。例如：

```
If IsNull(RichTextBox1.SelBullet) = True Then
    ' Code to run when selection is mixed.
ElseIf RichTextBox1.SelBullet = False Then
    ' Code to run when selection is doesn't have bullet style.
...
End If
```

请参阅

SelHangingIndent, SelIndent, SelRightIndent 属性，SelAlignment 属性，SelTabCount, SelTabs 属性。

示例

下面的例子改变了窗体上控件的状态以显示 RichTextBox 控件中所选择文本的布告板状态。要使用本例，在一个窗体上放置一个 RichTextBox 控件和一个 CheckBox 控件。将下面的代码拷贝到 RichTextBox 控件的 SelChange 事件中。然后运行本例。

```
Private Sub RichTextBox1_SelChange()
    If IsNull(RichTextBox1.SelBullet) = True Then
        Check1.Value = vbGrayed
    End If
End Sub
```

```
ElseIf RichTextBox1.SelBullet = True Then
    Check1.Value = vbChecked
ElseIf RichTextBox1.SelBullet = False Then
    Check1.Value = vbUnchecked
End If
End Sub
```

### SelChange 事件

当 RichTextBox 控件中当前选择的文本内容或插入点移动后产生该事件。

应用于

RichTextBox 控件。

语法

```
Private Sub object_SelChange(index As Integer )
```

SelChange 事件的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式     |
| <i>index</i>  | 一个整数，唯一标识控件矩阵中的某个控件 |

## 说明

可以使用 **SelChange** 事件检查各种属性，给出当前选择内容的状态，从而更新工具条中的按钮状态。

## 请参阅

**SelHangingIndent**, **SelIndent**, **SelRightIndent** 属性, **SelBold**, **SelItalic**, **SelStrikethru**, **SelUnderline** 属性, **SelFontName** 属性, **SelFontSize** 属性, **SelTabCount**, **SelTabs** 属性。

## 示例

下面的例子当前选择内容的大小，以便确定是否使能 **Cut** 或 **Copy** 等对剪贴板的操作。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件和 3 个 **Menu** 控件，创建一个带有 **Cut** 和 **Copy** 命令的菜单。将下面的代码拷贝到 **RichTextBox** 控件的 **SelChange** 事件中。然后运行本例：

```
Private Sub RichTextBox1_SelChange()  
    If RichTextBox1.SelLength > 0 Then  
        EditCutMenu.Enabled = True  
        EditCopyMenu.Enabled = True  
    Else
```



```
EditCutMenu.Enabled = False
EditCopyMenu.Enabled = False
End If
End Sub
```

## SelCharOffset 属性

返回或设置一个值，该值确定 **RichTextBox** 控件中的文本是显示在基准线上（普通），或作为超底稿显示在基准线之上，还是作为子底稿显示在基准线之下。该属性在设计时不可用。

应用于

**RichTextBox** 控件。

语法

*object*.SelCharOffset [=*offset*]

SelCharOffset 属性的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |

| 部分            | 描述                                      |
|---------------|---|
| <i>offset</i> | 一个整数，确定当前选择的文本或插入点之后的文本离基准线的距离，如“设置”中所示 |

设置

*offset* 值的设置如下：

| 设置   | 描述   |
|------|--|
| Null | 不是。所选择的文本有不同的偏移量                           |
| 0    | （缺省）标准。字符显示在基准线上                           |
| 正整数  | 超底稿。字符显示在基准线之上指定的 <i>offset</i> 个 twip 位置处 |
| 负整数  | 子底稿。字符显示在基准线之下指定的 <i>offset</i> 个 twip 位置处 |

说明

在运行时要读取并区分 Null 和 0 值，可以使用 `IsNull` 函数以及 `If...Then...Else` 语句。例如：

```
If IsNull(RichTextBox1.SelCharOffset) = True Then
    ' Code to run when selection is mixed.
ElseIf RichTextBox1.SelCharOffset = 0 Then
    ' Code to run when selection is all on the baseline
```

...

End If

请参阅

SetBold, SetItalic, SelStrikethru, SelUnderline 属性, SelColor 属性, SelFontName 属性, SelFontSize 属性。

示例

下面的例子使用滚动杆将选择的文本移动到基准线之上或之下。偏移量的最小和最大值由 RichTextBox 控件中文本的字体大小决定。要使用本例，在一个窗体上放置一个 RichTextBox 控件和一个 VScrollBar 控件。将下面的代码拷贝到 VScrollBar 控件的 Change 事件中。然后运行本例：

```
Private Sub VScroll1_Change ()  
    VScroll1.Max = RichTextBox1.SelFontSize  
    VScroll1.Min = -(VScroll1.Max)  
    RichTextBox1.SelCharOffset = VScroll1.Value  
End Sub
```

# SelColor 属性

返回或设置一个值，该值确定 RichTextBox 控件中文本的颜色。

应用于

RichTextBox 控件。

语法

```
object.SelColor [=color]
```

SelColor 属性的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |
| <i>color</i>  | 指定颜色的值，如“设置”中所示 |

设置

*color* 值的设置如下：

| 设置     | 描述                        |
|--------|---------------------------|
| Null   | 文本中包含了不同的颜色               |
| RGB 颜色 | 代码中使用 RGB 或 QColor 函数指定颜色 |

续表

| 设置   | 描述   |
|------|--|
| 系统颜色 | 使用 Object Browser 中 Visual Basic 对象库的系统颜色常量指定颜色。然后文本的颜色就与 Windows Control Panel 中指定的常量进行匹配 |

说明

如果 RichTextBox 控件中没有选择的文本，则设置该属性将确定当前插入点处所有输入的新字符的颜色。

请参阅

QBColor 函数，RGB 函数，SelBold，SelItalic，SelStrikethru，SelUnderline 属性，SelCharOffset 属性，SelFontName 属性，SelFontSize 属性，对象浏览器。

示例

下面的例子显示了一个 CommonDialog 控件中的对话框，指定 RichTextBox 控件中选择文本的颜色。要使用本例，在一个窗体上放置一个 RichTextBox 控件，一个 CommandButton 控件和一个 CommonDialog 控件。将下面的代码拷贝到 CommandButton 控件的 Click 事件中。然后运行本例：

```
Private Sub Command1_Click()  
    CommonDialog1.ShowColor
```

```
RichTextBox1.SelColor = CommonDialog1.Color  
End Sub
```

## SelFontName 属性

返回或设置 **RichTextBox** 控件中选择的文本或紧跟在插入点之后的字符所使用的字体。在设计时不可用。

应用于

**RichTextBox** 控件。

语法

*object*.SelFontName [=string]

SelFontName 属性的语法有如下几个部分：

| 部分            | 描述                |
|---------------|-------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式   |
| <i>string</i> | 一个字符串表达式，标识系统中的字体 |

说明

如果所选择的文本中包含了多种字体，则 **SelFontName** 属性的返回值是

Null。

请参阅

**SelBold**, **SelItalic**, **SelStrikethru**, **SelUnderline** 属性, **SelCharOffset** 属性, **SelColor** 属性, **SelFontSize** 属性。

示例

下面的例子显示了一个 **CommonDialog** 控件中的对话框, 指定 **RichTextBox** 控件中选择文本的字体属性。要使用本例, 在一个窗体上放置一个 **RichTextBox** 控件, 一个 **CommandButton** 控件和一个 **CommonDialog** 控件。将下面的代码拷贝到 **CommandButton** 控件的 **Click** 事件中。然后运行本例:

```
Private Sub Command1_Click ()  
    CommonDialog1.Flags = cdlCFBoth  
    CommonDialog1.ShowFont  
    With RichTextBox1  
        .SelFontName = CommonDialog1.FontName  
        .SelFontSize = CommonDialog1.FontSize  
        .SelBold = CommonDialog1.FontBold  
        .SelItalic = CommonDialog1.FontItalic  
        .SelStrikethru = CommonDialog1.FontStrikethru  
        .SelUnderline = CommonDialog1.FontUnderline
```

End With

End Sub

## SelFontSize 属性

返回或设置 **RichTextBox** 控件中选择的文本或紧跟在插入点之后的字符所使用的字体大小。在设计时不可用。

应用于

**RichTextBox** 控件。

语法

*object*.SelFontSize [=points]

SelFontSize 属性的语法有如下几个部分：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式                      |
| <i>points</i> | 一个整数，指定当前选择文本或紧跟在插入点之后的字符所使用的字体的点数大小 |



## 说明

`SelFontSize` 属性的最大值是 2160 点。

通常，在设置字体大小和字体属性之前应该改变 `SelFontName` 属性。然而，当设置的 `TrueType` 字体小于 8 点时，应该将 `SetFontSize` 属性设置为 3，然后设置 `SelFontName` 属性，再设置 `SelFontSize` 属性为想设置的值。

注意：可以使用的字体与系统配置、显示设备、打印设备有关，因此不同的系统其使用的字体也不相同。

如果所选择的文本中包含了多种字体大小，则 `SelFontName` 属性的返回值是 `Null`。

## 请参阅

`SelBold`, `SelItalic`, `SelStrikethru`, `SelUnderline` 属性, `SelCharOffset` 属性, `SelColor` 属性, `SelFontName` 属性。

## 示例

下面的例子显示了一个 `CommonDialog` 控件中的对话框，指定 `RichTextBox` 控件中选择文本的字体属性。要使用本例，在一个窗体上放置一个 `RichTextBox` 控件，一个 `CommandButton` 控件和一个 `CommonDialog` 控件。将下面的代码拷贝到 `CommandButton` 控件的 `Click` 事件中。然后运行本例：

```
Private Sub Command1_Click ()  
    CommonDialog1.Flags = Both  
    CommonDialog1.ShowFont  
    With RichTextBox1  
        .SelFontName = CommonDialog1.FontName  
        .SelFontSize = CommonDialog1.FontSize  
        .SelBold = CommonDialog1.FontBold  
        .SelItalic = CommonDialog1.FontItalic  
        .SelStrikethru = CommonDialog1.FontStrikethru  
        .SelUnderline = CommonDialog1.FontUnderline  
    End With  
End Sub
```

## SelHangingIndent, SelIndent, SelRightIndent 属性

返回或设置 **RichTextBox** 控件中文本段落的边界设置。该属性在设计时不可用。

应用于

**RichTextBox** 控件。

## 语法

*object.SelHangingIndent [=integer]*

*object.SelIndent [=integer]*

*object.SelRightIndent [=integer]*

**SelHangingIndent**, **SelIndent**, **SelRightIndent** 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 在“应用于”中指定的对象表达式   |
| <i>integer</i> | 一个整数，确定缩排的数量。这些属性使用包含 <b>RichTextBox</b> 控件的 <b>Form</b> 对象尺度单位 |

## 说明

对所受影响的段落来说，**SelIndent** 属性指定了 **RichTextBox** 控件的左边框与选择的文本左边之间的间距。同样，**SelRightIndent** 属性指定 **RichTextBox** 控件的右边框与选择文本的右边之间的间距。

**SelHangingIndent** 属性指定的间距是选择段落中第一行文本左边与同一段落中的后续行左边之间的距离。

如果选择中的行包含的边界设置不一致，则这些属性的返回值是 0。

请参阅

`SelAlignment` 属性, `SelBullet` 属性, `SelTabCount`, `SelTabs` 属性, `Form` 对象, `Forms` 集合。

示例

下面的例子选择了 `RichTextBox` 控件中的所有文本, 然后设置左缩排和右缩排的间距。要使用本例, 在一个窗体上放置一个 `RichTextBox` 控件, 一个 `CommandButton` 控件和一个 `TextBox` 控件。给 `RichTextBox` 控件加载一个文本文件, 将下面的代码拷贝到窗体的 `General Declarations` 段中。然后运行本例:

```
Private Sub Command1_Click()  
    Dim Margins As Integer  
    Margins = CInt(Text1.Text)  
    With RichTextBox1  
        .SelStart = 1  
        .SelLength = Len(RichTextBox1.Text)  
        .SelIndent = Margins  
        .SelRightIndent = Margins  
    End With  
End Sub
```

# SelPrint 方法

将 RichTextBox 控件中的格式化文本发送给打印设备。

应用于

RichTextBox 控件。

语法

*object.SelPrint (hdc)*

SelPrint 方法的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>hdc</i>    | 打印控件内容的设备环境            |

说明

如果 RichTextBox 控件中有选择的文本，则 SelPrint 方法只将选择的文本发送给打印设备。如果没有选择的文本，则将 RichTextBox 控件的所有文本内容发送给打印设备。

SelPrint 方法并不打印 RichTextBox 控件中的文本，而是将该文本的拷贝发送给可以打印文本的打印设备。例如，使用下面的代码将文本发送给一个 Printer

对象。

`RichTextBox1.SelPrint(Printer.hDC)`

Printer 对象的 `hDC` 属性用于指定 `SelPrint` 方法中的设备环境。

**注意：**如果使用 Printer 对象作为 RichTextBox 控件文本的打印目的地，必须打印某些非 0 长的字符串以便初始化 Printer 对象的设备环境。

请参阅

`SelBold`, `SelItalic`, `SelStrikethru`, `SelUnderline` 属性, `hDC` 属性, Printer 对象, Printers 集合。

示例

下面的例子打印了 RichTextBox 控件中的格式化文本。要使用本例，在一个窗体上放置一个 RichTextBox 控件，一个 CommandDialog 控件和一个 CommandButton 控件。将下面的代码拷贝到 CommandButton 控件的 Click 事件中。然后运行本例：

```
Private Sub Command1_Click()
```

```
    CommonDialog1.Flags = cdlIPDReturnDC + cdlPDNoPageNums
```

```
    If RichTextBox1.SelLength = 0 Then
```

```
        CommonDialog1.Flags = CommonDialog1.Flags + cdlPDAllPages
```

```
Else
    CommonDialog1.Flags = CommonDialog1.Flags + cdIPDSelection
End If
CommonDialog1.ShowPrinter
Printer.Print ""
RichTextBox1.SelPrint CommonDialog1.hDC
End Sub
```

## SelProtected 属性

返回或设置一个值，该值确定是否保护当前选择的内容。在设计时不可用。

应用于

RichTextBox 控件。

语法

*object*.SelProtected [=value]

SelProtected 属性的语法有如下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象           |
| <i>value</i>  | 可变量，确定当前的选择内容是否受到保护，如“设置”<br>中所示 |

## 设置

value 值的设置如下：

| 设置    | 描述              |
|-------|-----------------|
| Null  | 选择的内容包含了不同的保护类型 |
| True  | 选择中的内容受到保护      |
| False | 选择中的内容不保护       |

## 说明

保护的文本看起来与普通的文本一样，但不能被用户修改。即该内容在运行时不能被修改。这允许使用 **RichTextBox** 控件事件表格，在表格中创建用户不能修改的区域。

## 请参阅

**SelBold**, **SelItalic**, **SelStrikethru**, **SelUnderline** 属性。



# SelRTF 属性

返回或设置 RichTextBox 控件中当前选择的文本（以.RTF 格式）。该属性在设计时不可用。

应用于

RichTextBox 控件。

语法

*object.SelRTF [=string]*

SelRTF 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | 一个.RTF 格式的字符串表达式       |

说明

设置 SelRTF 属性替换 RichTextBox 控件中所选择的文本。如果控件中没有选择的文本，则返回空字符串。

可以与 Print 函数一起使用 SelRTF 属性写.RTF 文件。

请参阅

**TextRTF** 属性，支持的 RTF 码（**RichTextBox** 控件）。

示例

下面的例子将 **RichTextBox** 控件中加亮的内容保存到一个 .RTF 文件中。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件和一个 **CommandButton** 控件。将下面的代码拷贝到 **CommandButton** 控件的 **Click** 事件中。然后运行本例：

```
Private Sub Command1_Click ()  
    Open "mytext.rtf" For Output As 1  
    Print #1, RichTextBox1.SelRTF  
    Close 1  
  
End Sub
```

**SelTabCount**, **SelTabs** 属性

返回或设置 **RichTextBox** 控件中制表符的个数和文本的决定制表符位置。该属性在设计时不可用。

应用于

RichTextBox 控件。

语法

*object*.SelTabCount [=count]

*object*.SelTabs (index) [=location]

SelTabCount 和 SetTabs 属性的语法有如下几个部分：

| 部分              | 描述   |
|-----------------|--|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>count</i>    | 一个整数，确定所选择段落或插入点之后段落中制表符位置的个数                                    |
| <i>index</i>    | 一个整数，标识某个特定的制表符。第一个制表符索引是 0。最后一个制表符的索引等于 SelTabCount 减 1         |
| <i>location</i> | 一个整数，指定一个制表符的位置。表示制表符位置的单位由 Form 对象或其他包含 RichTextBox 控件的对象标尺模式决定 |

说明

缺省地，在 RichTextBox 控件中按下 TAB 键将把输入焦点按 TabIndex 属性给出的位置传递给下一个控件。在文本中输入制表符的一个方法是按下

CTRL+TAB 键。然而，那些熟悉字处理器应用程序的用户或许会发现 CTRL+TAB 键与他们的习惯不一样。可以使用 TAB 键在 RichTextBox 控件中输入制表符，方法是在 RichTextBox 控件拥有输入焦点时，将 Form 对象中所有其他控件的 TabStop 属性暂时设置为 False。例如：

```
Private Sub RichTextBox1_GotFocus()  
    ' Ignore errors for controls without the TabStop property.  
    On Error Resume Next  
    ' Switch off the change of focus when pressing TAB.  
    For Each Control In Controls  
        Control.TabStop = False  
    Next Control  
End Sub
```

当 RichTextBox 控件失去输入焦点时确保将 Form 的其他控件的 TabStop 属性恢复。

请参阅

SelHangingIndent, SelAlignment 属性, SelBullet 属性, Form 对象, Forms 集合, TabIndex 属性, TabStop 属性。

示例

下面的例子给一个 RichTextBox 控件设置了 5 个制表符，每个制表符的位

置均是序号乘 5。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件和一个 **CommandButton** 控件。将下面的代码拷贝到 **CommandButton** 控件的 **Click** 事件中。然后运行本例：

```
Private Sub Command1_Click()  
    With RichTextBox1  
        .SelTabCount = 5  
        For X = 0 To .SelTabCount - 1  
            .SelTabs(X) = 5 * X  
        Next X  
    End With  
End Sub
```

## Span 方法

根据指定的字符集选择 **RichTextBox** 控件中的文本。

应用于

**RichTextBox** 控件。

语法

*object.Span charset, forward, negate*

Span 方法的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>charset</i> | 当超出选择范围时，根据 <i>negate</i> 的值而查找的指定字符集的字符串表达式                     |
| <i>forward</i> | 可选参数。一个布尔表达式，指定插入点的移动方向，如“设置”中所示                                 |
| <i>negate</i>  | 可选参数。一个布尔表达式，指定 <i>charset</i> 中的字符是定义目的字符集还是排除在目的字符集之外，如“设置”中所示 |

设置

*forward* 值的设置如下：

| 设置    | 描述                              |
|-------|---------------------------------|
| True  | （缺省）从当前插入点或当前选择内容的开始向文本末尾方向选择文本 |
| False | 从当前插入点或选择内容的末尾向文本的开头方向选择文本      |

*negate* 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 选中的包含的字符是那些不在 <i>characteraset</i> 中出现的字符。<br>选择的停止位置是首次发现 <i>characteraset</i> 参数中的字符位置 |
| False | （缺省）选中的字符是包含在 <i>characteraset</i> 中的字符。选择的停止位置是首次发现不在 <i>characteraset</i> 参数中的字符位置     |

## 说明

**Span** 方法主要用于从 **RichTextBox** 控件中选择一个单词或语句。

如果 **Span** 方法根据提供的参数找不到指定的字符，则当前的插入点或选择内容不变。

**Span** 方法不返回任何值。

## 请参阅

**UpTo** 方法。

## 示例

下面的例子定义了一对键盘加速键，用于在 **RichTextBox** 控件中选择文本时范围是整个语句（**CTRL+S**）还是整个单词（**CTRL+W**）。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件，将下面的代码拷贝到窗体的 **KeyUp** 事件中。然后运行本例：

```
Private Sub RichTextBox1_KeyUp (KeyCode As Integer, Shift As Integer)
```

```

If Shift = vbCtrlMask Then
    Select Case KeyCode
        ' If Ctrl+S:
        Case vbKeyS
            ' Select to the end of the sentence.
            RichTextBox1.Span ".?!:", True, True
            ' Extend selection to include punctuation.
            RichTextBox1.SelLength = RichTextBox1.SelLength + 1
        ' If Ctrl+W:
        Case vbKeyW
            ' Select to the end of the word.
            RichTextBox1.Span " ,;:?!", True, True
    End Select
End If
End Sub

```

## 支持的 RTF 码

RichTextBox 控件支持下列 RTF 码。在加载文本时忽略所有其他的 RTF 码。



| RTF 码    | 描述                 | RTF 码      | 描述                  |
|----------|--------------------|------------|---------------------|
| -        | OptionalHyphen     | objcropl   | CropLeft            |
| \n       | EndParagraph       | objcropr   | CropRight           |
| \r       | EndParagraph       | objcropt   | CropTop             |
| -        | NonBreakingHyphen  | objdata    | ObjectData          |
|          | FormulaCharacter   | object     | Object              |
| ~        | NonBreakingSpace   | objemb     | ObjectEmbedded      |
| ansi     | CharSetAnsi        | objh       | Height              |
| b        | Bold               | objicemb   | ObjectMacICEbedder  |
| bin      | BinaryData         | objlink    | ObjectLink          |
| blue     | ColorBlue          | objname    | ObjectName          |
| bullet   | ANSI Character 149 | objpub     | ObjectMacPublisher  |
| cb       | ColorBackground    | objscalex  | ScaleX              |
| cell     | Cell               | objscaley  | ScaleY              |
| cf       | ColorForeground    | objsetsize | ObjectSetSize       |
| colortbl | ColorTable         | objsub     | ObjectMacSubscriber |
| cpg      | ColorPage          | objw       | Width               |
| deff     | DefaultFont        | par        | EndParagraph        |
| deflang  | DefaultLanguage    | pard       | ParagraphDefault    |
| deftab   | DefaultTabWidth    | pc         | CharSetPc           |

续表

| RTF 码    | 描述                   | RTF 码      | 描述                 |
|----------|----------------------|------------|--------------------|
| deleted  | Deleted              | pca        | CharSetPs2         |
| dibitmap | PictureWindowsDIB    | piccrob    | CropBottom         |
| dn       | Down                 | piccropl   | CropLeft           |
| dy       | TimeDay              | piccropr   | CropRight          |
| emdash   | ANSI Character 151   | pcccropt   | CropTop            |
| endash   | ANSI Character 150   | pich       | Height             |
| f        | FontSelect           | pichgoal   | DesiredHeight      |
| fbidi    | FontFamilyBidi       | picscalex  | ScaleX             |
| fchars   | FollowingPunct       | picscaley  | ScaleY             |
| fcharset | CharSet              | pict       | Picture            |
| fdecor   | FontFamilyDecorative | picw       | Width              |
| fi       | IndentFirst          | picwgoal   | DesiredWidth       |
| field    | Field                | plain      | CharacterDefault   |
| fldinst  | FieldInstruction     | pmmetafile | PictureOS2Metafile |
| fldrslt  | FieldResult          | pn         | ParaNum            |
| fmodern  | FontFamilyModern     | pnindent   | ParaNumIndent      |
| fname    | RealFontName         | pnlvlblt   | ParaNumBullet      |
| fnil     | FontFamilyDefault    | pntext     | ParaNumText        |

续表

| RTF 码    | 描述                                | RTF 码     | 描述                |
|----------|-----------------------------------|-----------|-------------------|
| fontemb  | FontEmbedded                      | pntxta    | ParaNumAfter      |
| fontfile | FontFile                          | pntxtb    | ParaNumBefore     |
| fonttbl  | FontTable                         | protect   | Protect           |
| footer   | NullDestination(Footer)           | qc        | AlignCenter       |
| footerf  | NullDestination(Footer,<br>first) | ql        | AlignLeft         |
| footerl  | NullDestination(Footer,<br>left)  | qr        | AlignRight        |
| footerr  | NullDestination(Footer,<br>right) | rdblquote | ANSI Character 34 |
| footnote | NullDestination(footnote)         | red       | ColorRed          |
| fprq     | Pitch                             | result    | ObjectResult      |
| froman   | FontFamilyRoman                   | revauth   | RevAuthor         |
| fs       | FontSize                          | revised   | Revision          |
| fscript  | FontFamilyScript                  | ri        | IndentRight       |
| fswiss   | FontFamilySwiss                   | row       | Row               |
| ftech    | FontFamilyTechnical               | rquote    | ANSI Character 39 |

续表

| RTF 码   | 描述                                       | RTF 码      | 描述                   |
|---------|--|------------|----------------------|
| ftncn   | NullDestination(Footnote<br>cont.)       | rtf        | Rtf                  |
| ftnsep  | NullDestination(Footnote<br>separ)       | rtfch      | RightToLeftChars     |
| ftnsepc | NullDestination(Footnote<br>cont, separ) | rtl doc    | RightToLeftDocument  |
| green   | ColorGreen                               | rtlmark    | DisplayRightToLeft   |
| header  | NullDestination(Header)                  | rtlpar     | RightToLeftParagraph |
| headerf | NullDestination(Header,fi<br>rst)        | sec        | TimeSecond           |
| headerl | NullDestination(Header,<br>left)         | sect       | EndSection           |
| headerr | NullDestination(Header,<br>right)        | sectd      | SectionDefault       |
| horzdoc | HorizontalRender                         | strike     | StrikeOut            |
| hr      | TimeHour                                 | stylesheet | StyleSheet           |
| I       | Italic                                   | sub        | Subscript            |

续表

| RTF 码     | 描述                        | RTF 码    | 描述                                  |
|-----------|---------------------------|----------|-------------------------------------|
| Info      | DocumentArea(Info fields) | super    | Superscript                         |
| intbl     | InTable                   | tb       | TabPosition                         |
| lang      | Language                  | tc       | NullDestination (Table of contents) |
| lchars    | LendingPunct              | tx       | TabPosition                         |
| ldblquote | ANSI Character 34         | ul       | Underline                           |
| li        | IndentLeft                | uld      | UnderlineDotted                     |
| line      | SoftBreak                 | uldash   | UnderlineDash                       |
| lquote    | ANSI Character 39         | uldashd  | UnderlineDashDotted                 |
| ltrch     | LeftToRightChars          | uldashdd | UnderlineDashDotDotted              |
| ltrdoc    | LeftToRightDocument       | uldb     | UnderlineDouble                     |
| ltrmark   | DisplayLeftToRight        | ulhair   | UnderlineHairline                   |
| ltrpar    | LeftToRightParagraph      | ulnone   | StopUnderline                       |
| mac       | CharSetMacintosh          | ulth     | UnderlineThick                      |
| macpict   | PictureQuickDraw          | ulw      | UnderlineWord                       |
| margl     | MarginLeft                | ulwave   | UnderlineWave                       |
| marglsxn  | SectionMarginLeft         | up       | Up                                  |

续表

| RTF 码      | 描述                 | RTF 码         | 描述                           |
|------------|--------------------|---------------|------------------------------|
| margr      | MarginRight        | v             | HiddenText                   |
| margrsxn   | SectionMarginRight | vertdoc       | VerticalRender               |
| min        | TimeMinute         | wbitmap       | PictureWindowsBitmap         |
| mo         | TimeMonth          | wbmbitspixel  | BitmapBitsPerPixel           |
| nocwrap    | NoWordBreak        | wbmplanes     | BitmapNumPlanes              |
| nooverflow | NoOverflow         | wbmwidthbytes | BitmapWidthBytes             |
| nosupersub | NoSuperSub         | wmetafile     | PictureWindowsMetafile       |
| nowwrap    | NoWordWrap         | xe            | NullDestination(index entry) |
| objautlink | ObjectAutoLink     | yr            | TimeYear                     |
| objclass   | ObjectClass        | zwj           | ZeroWidthJoiner              |
| objropb    | CropBottom         | zwnj          | ZeroWidthNonJoiner           |

请参阅

FileName 属性 (RichTextBox 控件), LoadFile 方法, SaveFile 方法, SelRTF 属性, TextRTF 属性。

# TextRTF 属性

返回或设置 RichTextBox 控件中的所有文本，包括.rtf 码。

应用于

RichTextBox 控件。

语法

*object*.TextRTF [=string]

TextRTF 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | .RTF 格式的字符串表达式         |

说明

设置 TextRTF 属性将替换 RichTextBox 控件的所有内容。

可以与 Print 函数一起使用 TextRTF 属性。结果文件可以被任何能读出 RTF 文件的字处理器应用程序读出。

请参阅

SaveFile 方法，SelRTF 属性，支持的 RTF 码（RichTextBox 控件）。

示例

下面的例子将 RichTextBox 控件的整个文本保存到一个.RTF 文件中。要使用本例，在一个窗体上放置一个 RichTextBox 控件和一个 CommandButton 控件。将下面的代码拷贝到 CommandButton 控件的 Click 事件中。然后运行本例：

```
Private Sub Command1_Click ()  
    Open "mytext.rtf" For Output As 1  
    Print #1, RichTextBox1.TextRTF  
    Close 1
```

End Sub

## Upto 方法

将插入点移动到首次找到的 RichTextBox 控件中指定的字符集中的字符位置，但不包括该字符。



应用于

RichTextBox 控件。

语法

*object.Upto (characterset, forward , negate)*

Upto 方法的语法有如下几个部分：

| 部分                  | 描述  |
|---------------------|---|
| <i>object</i>       | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>characterset</i> | 当移动插入点时，根据 <i>negate</i> 的值而查找的指定字符集的字符串表达式                           |
| <i>forward</i>      | 可选参数。一个布尔表达式，指定插入点的移动方向，如“设置”中所示                                      |
| <i>negate</i>       | 可选参数。一个布尔表达式，指定 <i>characterset</i> 中的字符是定义目的字符集还是排除在目的字符集之外，如“设置”中所示 |

设置

*forward* 值的设置如下：

| 设置    | 描述                 |
|-------|--------------------|
| True  | （缺省）插入点朝文本末尾方向向前移动 |
| False | 插入点朝文本开头方向向后移动     |

*negate* 值的设置如下:

| 设置    | 描述  |
|-------|---|
| True  | 使用没有在 <i>characteraset</i> 中找到的字符移动插入点    |
| False | (缺省) 使用在 <i>characteraset</i> 中找到的字符移动插入点 |

请参阅

Span 方法。

示例

下面的例子定义了一对键盘加速键，用于在 **RichTextBox** 控件中移动插入点时是整个语句 (**CTRL+S**) 还是整个单词 (**CTRL+W**)。要使用本例，在一个窗体上放置一个 **RichTextBox** 控件，将下面的代码拷贝到窗体的 **KeyUp** 事件中。然后运行本例：

```
Private Sub RichTextBox1_KeyUp (KeyCode As Integer, Shift As Integer)
    If Shift = vbAltMask Then
        Select Case KeyCode
            ' If Alt+S:
            Case vbKeyS
                ' Move insertion point to the end of the sentence.
                RichTextBox1.Upto ".?!:", True, False
```

```
' If Alt+W:
Case vbkeyW
    ' Move insertion point to the end of the word.
    RichTextBox1.Upto " .?!:", True, False
End Select
End If
End Sub
```

## Slider 控件

一个 **Slider** 控件是包含了滑块和可选的刻度标记的窗口。你可以拖动滑块或在滑块的两边点击鼠标或使用键盘来移动滑块。

### 语法

#### **Slider**

### 说明

当想选择一个离散值或一个范围中的连续值时，**Slider** 控件是很有用的。例如，可以使用一个 **Slider** 控件，通过拖动滑块来设置显示图像的大小，而不是手工输入一个值。要选择一个数值范围，将 **SelectRange** 属性设置为 **True**，当按下 **SHIFT** 键时，应用程序控制选择一个范围。

**Slider** 控件的方向是水平的或垂直的。

**发布须知：**要在你的应用程序中使用 **Slider** 控件，必须将 **MSCOMCTL32.OCX** 加入到工程中。当发布应用程序时，将 **MSCOMCTL32.OCX** 文件加入到用户的 Microsoft Windows System 或 System32 目录下。要获取其他信息，请参阅《Microsoft Visual Basic 6.0 程序员指南》。

## 属性

BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), Max, Min 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), Text 属性 (ActiveX 控件), Value 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), LargeChange, SmallChange 属性 (ActiveX 控件), Orientation 属性 (Slider 控件), SelectRange 属性, SelLength, SelStart 属性 (Slider 控件), TickFrequency 属性, TickStyle 属性, TextPosition 属性, DataBindings 属性, TabIndex 属性, DragIcon 属性, DragMode 属性, hWnd 属性, MouseIcon 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性。

## 方法

ClearSel 方法, GetNumTicks 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件)。

## 事件

Scroll 事件 (Slider 控件), DragDrop 事件, DragOver 事件, GotFocus 事

件,KeyDown,KeyUp 事件,KeyPress 事件,LostFocus 事件,MouseDown, MouseUp 事件,MouseMove 事件,OLECompleteDrag 事件(ActiveX 控件),OLEDragDrop 事件(ActiveX 控件),OLEDragOver 事件(ActiveX 控件),OLEGiveFeedback 事件(ActiveX 控件),OLESetData 事件(ActiveX 控件),OLEStartDrag 事件(ActiveX 控件),Change 事件(ActiveX 控件),KeyDown,KeyUp 事件(ActiveX 控件), KeyPress 事件(ActiveX 控件)。

请参阅

Slider 常量, 使用 Slider 控件。

## ClearSel 方法

清除 Slider 控件中当前的选择。

应用于

Slider 控件。

语法

*object*.ClearSel

object 是在“应用于”中指定的对象表达式。

## 说明

该方法将 SelStart 属性设置为 Value 属性，并将 SelLength 属性设置为 0。

## 请参阅

Slider 控件，SelectRange 属性，SelLength, SelStart 属性（Slider 控件），Value 属性，SelLength, SelStart 属性，SelText 属性，Value 属性（ActiveX 控件）。

## GetNumTicks 方法

返回 Slider 控件中 Min 和 Max 属性之间的刻度数。

## 应用于

Slider 控件。

## 语法

*object*.GetNumTicks

object 是在“应用于”中指定的对象表达式。

## 说明

要改变刻度数，需重新设置 Min 或 MaxTickFrequency 属性值。

请参阅

Slider 控件，TickFrequency 属性，Max ,Min 属性（Common Dialog）。

示例

下面的例子显示了 Slider 控件中的当前刻度数，然后将 Max 属性增加 10。要使用本例，在一个窗体上放置一个 Slider 控件，将下面的代码拷贝到窗体的 Declarations 段。运行本例，点击 Slider 控件获取刻度数。每次点击该控件都增加了刻度数。

```
Private Sub Slider1_Click()  
    MsgBox Slider1.GetNumTicks  
    Slider1.Max = Slider1.Max + 10  
End Sub
```

## Orientation 属性（Slider 控件）

设置一个值，该值确定 Slider 控件是水平的还是垂直的。

应用于

FlatScrollBar 控件，Slider 控件。



# 语法

```
object.Orientation = number
```

Orientation 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式         |
| <i>number</i> | 一个常量或值，指定控件的方位，如“设置”中所示 |

# 设置

*number* 值的设置如下：

| 常量            | 值 | 描述                               |
|---------------|---|----------------------------------|
| sldHorizontal | 0 | （缺省）水平。滑鼠水平移动，刻度显示在上面或下面或都显示或不显示 |
| sldVertical   | 1 | 垂直。滑块垂直移动，刻度显示在左边或右边或都显示或不显示     |

# 请参阅

Slider 控件常量。

# 示例

下面的例子在窗体上切换 Slider 控件的方位。要使用本例，在一个窗体上

放置一个 **Slider** 控件，将下面的代码拷贝到窗体的 **Declarations** 段，然后运行本例。点击窗体将切换 **Slider** 控件的方位。

```
Private Sub Form_Click()  
    If Slider1.Orientation = 0 Then  
        Slider1.Orientation = 1  
    Else  
        Slider1.Orientation = 0  
    End If  
End Sub
```

..

## Scroll 事件（Slider 控件）

当移动 **Slider** 控件中的滑块或使用鼠标或键盘点击控件时产生该事件。

应用于

**FlatScrollBar** 控件，**Slider** 控件。

语法

```
Private Sub object_Scroll ()
```

object 是在“应用于”中指定的对象表达式。

## 说明

Scroll 事件在 Click 事件之前产生。

当滑块移动时，Scroll 事件连续返回 Value 属性的值。可以使用该属性控制该 Slider 控件中要发生的事情。比较一个 Change 事件，在 Slider 控件的 Value 属性改变时，当只想更新一次时使用 Change 事件。

**注意：**在该事件中避免使用 MsgBox 语句或函数。

## 请参阅

MsgBox 函数，Slider 控件，Change 事件，Value 属性（ActiveX 控件），LargeChange, SmallChange 属性（ActiveX 控件），Change 事件（ActiveX 控件）。

## SelectRange 属性

设置一个值，该值确定 Slider 控件是否可以有一个数值范围。

## 应用于

Slider 控件。

# 语法

*object.SelectRange =boolean*

SelectRange 属性的语法有如下几个部分：

| 部分             | 描述                                       |
|----------------|--|
| <i>object</i>  | 在“应用于”中指定的对象表达式                          |
| <i>boolean</i> | 一个布尔表达式，确定 Slider 控件是否可以有一个数值范围，如“设置”中所示 |

# 设置

*boolean* 值的设置如下：

| 设置    | 描述                 |
|-------|--------------------|
| True  | Slider 控件可以有一个数值范围 |
| False | Slider 控件不能有数值范围   |

# 说明

如果 SelectRange 设置为 False，则 SelStart 属性的设置与 Value 属性一样。设置 SelStart 属性的值也改变 Value 属性值，反之一样，这将反映出滑块在 Slider 控件中的位置。当 SelectRange 属性设置为 False 时，设置 SelLength 属性不起作用。

请参阅

Slider 控件, SelectRange 属性, SelLength, SelStart 属性 (Slider 控件), ClearSel 方法, Value 属性, SelLength, SelStart, SelText 属性, Value 属性 (ActiveX 控件)。

示例

下面的例子允许用户在按下 **SHIFT** 键时选择一个数值范围。要使用本例，在一个窗体上放置一个 **Slider** 控件，将下面的代码拷贝到窗体的 **Declarations** 段。运行本例，按下 **SHIFT** 键并拖动鼠标或在 **Slider** 控件上点击鼠标选择一个范围。

```
Private Sub Form_Load()  
    'Set slider control settings  
    Slider1.Max = 20  
End Sub
```

```
Private Sub Slider1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As  
Single)
```

```
    If Shift = 1 Then                                ' If Shift button is down then  
        Slider1.SelectRange = True                  ' turn SelectRange on.  
        Slider1.SelStart = Slider1.Value            ' Set the SelStart value  
        Slider1.SelLength = 0                        ' Set previous SelLength (if any) to 0.  
    End If  
End Sub
```

```
Private Sub Slider1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)

    If Shift = 1 Then
        ' If user selects backwards from a point, an error will occur.
        On Error Resume Next
        ' Else set SelLength using SelStart and current value.
        Slider1.SelLength = Slider1.Value - Slider1.SelStart
    Else
        Slider1.SelectRange = False ' If user lifts SHIFT key.
    End If
End Sub
```

## SelLength, SelStart 属性（Slider 控件）

- SelLength 返回或设置 Slider 控件中选择范围的长度。
- SelStart 返回或设置 Slider 控件中选择范围的起点。

应用于

Slider 控件。

## 语法

*object.SelLength [=value]*

*object.SelStart [=value]*

SelLength, SelStart 属性的语法有如下几个部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式      |
| <i>value</i>  | Min 和 Max 属性值之间的一个数值 |

## 说明

使用 SelLength 和 SelStart 属性在 Slider 控件中选择一个连续的数值范围。

SelLength 属性不能设置得比 0 小，SelLength 和 SelStart 之和不能比 Max 属性值大。

## 请参阅

SelectRange 属性，ClearSel 方法，Max, Min 属性（Common 对话框）。

## 示例

下面的例子在 Slider 控件中选择一个范围。要使用本例，在一个窗体上放置一个 Slider 控件和 3 个 TextBox 控件，分别命名为 Text1, Text2 和 Text3。Slider 控件的 SelectRange 属性必须设置为 True。将下面的代码拷贝到窗体的

**Declarations** 段中，运行本例。在按下 **SHIFT** 键时可以选择一个连续的数值范围，这样将在文本框中显示各种数值。

```
Private Sub Form_Load()  
    ' Make sure SelectRange is True so selection can occur.  
    Slider1.SelectRange = True  
End Sub
```

```
Private Sub Slider1_MouseDown(Button As Integer, Shift As Integer, x As Single, y  
As Single)  
    If Shift = 1 Then ' If SHIFT is down, begin the range selection.  
        Slider1.ClearSel    ' Clear any previous selection.  
        Slider1.SelStart = Slider1.Value  
        Text2.Text = Slider1.SelStart    ' Show the beginning  
            ' of the range in the textbox.  
    Else  
        Slider1.ClearSel    ' Clear any previous selection.  
    End If  
End Sub
```

```
Private Sub Slider1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As  
Single)
```



```
' When SHIFT is down and SelectRange is True,  
' this event is triggered.  
If Shift = 1 And Slider1.SelectRange = True Then  
    ' Make sure the current value is larger than SelStart or  
    ' an error will occur--SelLength can't be negative.  
    If Slider1.Value >= Slider1.SelStart Then  
        Slider1.SelLength = Slider1.Value - Slider1.SelStart  
        Text1.Text = Slider1.Value    ' To see the end of the range.  
        ' Text3 is the difference between the end and start values.  
        Text3.Text = Slider1.SelLength  
    End If  
End If  
End Sub
```

## TextPosition 属性

返回或设置一个值，确定显示文本相对于对象的位置。

应用于

Slider 控件。

# 语法

*object*.TextPosition [= *integer*]

TextPosition 属性的语法有下面几部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象                    |
| <i>integer</i> | 一个数值表达式，其设置值决定了 ToolTipText 的位置，<br>如设置值所示的 |

# 设置

*integer* 的设置值为：

| 常量            | 值 | 描述            |
|---------------|---|---------------|
| sldAboveLeft  | 0 | 文本显示在控件的上边或左边 |
| sldBelowRight | 1 | 文本显示在控件的下边或右边 |

# 说明

Above 和 Below 只用于水平滑块，Left 和 Right 只用于垂直滑块。

# TickFrequency 属性

返回或设置 Slider 控件上与数值范围相关的刻度标记频率。例如，若范围是 100，而 TickFrequency 属性设置为 2，则范围每增加 2 就显示一个刻度。

应用于

Slider 控件。

语法

*object*.TickFrequency [=*number*]

TickFrequency 属性的语法有如下几个部分：

| 部分            | 描述               |
|---------------|------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式  |
| <i>number</i> | 一个数字表达式，指定刻度标记频率 |

请参阅

Slider 控件，GetNumTicks 方法。

示例

下面的例子将 TextBox 控件的宽度与 Slider 控件匹配。当 Slider 控件的 Value

值超过某个值时，TextBox 控件的宽度就与 Slider 控件的值匹配。TickFrequency 属性的值都取决于 Slider 控件的 Max 属性值。要使用本例，在一个窗体上放置一个 Slider 控件和一个 TextBox 控件，将下面的代码拷贝到窗体的 Declarations 段。运行本例，点击滑块几次。

```
Private Sub Form_Load()  
    Text1.Width = 4500          ' Set a minimum width for the TextBox.  
    Slider1.Left = Text1.Left   ' Align the Slider to the TextBox.  
    ' Match the width of the Slider to the TextBox.  
    Slider1.Max = Text1.Width  
    ' Place the Slider a little below the Textbox.  
    Slider1.Top = Text1.Top + Text1.Height + 50  
    ' Set TickFrequency to a fraction of the Max value.  
    Slider1.TickFrequency = Slider1.Max * 0.1  
    ' Set LargeChange and SmallChange value to a fraction of Max.  
    Slider1.LargeChange = Slider1.Max * 0.1  
    Slider1.SmallChange = Slider1.Max * 0.01  
End Sub
```

```
Private Sub Slider1_Change()  
    ' If the slider is under 1/3 the size of the textbox, no change.  
    ' Else, match the width of the textbox to the Slider's value.  
    If Slider1.Value > Slider1.Max / 3 Then
```

```
Text1.Width = Slider1.Value
End If
End Sub
```

## TickStyle 属性

返回或设置在 **Slider** 控件中显示的刻度标记风格（或定位）。

应用于

**Slider** 控件。

语法

*object*.TickStyle [=*number*]

TickStyle 属性的语法有如下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式                  |
| <i>number</i> | 一个常量或整数，指定 TickStyle 属性，如“设置”中所示 |

设置

**number** 值的设置如下：

| 常量             | 值 | 描述  |
|----------------|---|---|
| sldBottomRight | 0 | (缺省)底部/右边。如果 <b>Slider</b> 控件水平放置，则刻度显示在底部；如果 <b>Slider</b> 控件垂直放置，则刻度显示在右边 |
| sldTopLeft     | 1 | 顶部/左边。如果 <b>Slider</b> 控件水平放置，则刻度显示在顶部；如果 <b>Slider</b> 控件垂直放置，则刻度显示在左边     |
| sldBoth        | 2 | 二者。刻度显示在 <b>Slider</b> 控件两边或顶部和底部   |
| sldNoTicks     | 3 | 没有。不显示刻度  |

请参阅

**Slider** 控件，**GetNumTicks** 方法，**Slider** 常量。

示例

下面的例子允许在下拉框中查看各种刻度标记的风格。要使用本例，在一个窗体上放置一个 **Slider** 控件和一个 **ComboBox** 控件。将下面的代码拷贝到窗体的 **Declarations** 段中，运行本例。点击 **ComboBox** 改变 **TickStyle** 属性的值。

```
Private Sub Form_Load()  
    With combo1  
        .AddItem "Bottom/Right"  
        .AddItem "Top/Left"  
        .AddItem "Both"  
        .AddItem "None"
```

```
        .ListIndex = 0
```

```
    End With
```

```
End Sub
```

```
Private Sub combo1_Click()
```

```
    Slider1.TickStyle = combo1.ListIndex
```

```
End Sub
```

## StatusBar 控件

一个 **StatusBar** 控件通过了一个窗口，通常都在应用程序的底部，应用程序可以在其中显示各种状态数据。一个 **StatusBar** 控件最多可以分成 16 个 **Panel** 对象，都保存在一个 **Panels** 集合中。

### 语法

#### **StatusBar**

### 说明

一个 **StatusBar** 控件由 **Panel** 对象组成，每个 **Panel** 对象都可以包含文本和/或图片。控制每个 **Panel** 对象外观的属性有 **Width**, **Alignment** 和 **Bevel**。并且，可以使用 **Style** 属性的 7 个值之一自动显示普通的数据，如日期、时间和键盘状态。

在设计时，可以创建和自定义 **Panel** 对象的外观，在 **StatusBar** 控件的 **Properties Page** 中，选择 **Panel** 卡片就可以设置各种参数了。在运行时，可以根



据应用程序的状态对 **Panel** 对象重新配置以反映不同的功能。有关 **Panel** 对象的属性、事件和方法的信息，请参阅 **Panel** 对象和 **Panels** 集合。

一个 **StatusBar** 控件通常显示在窗体上所查看对象的信息、对象的组成或与对象操作有关的内容信息。**StatusBar** 控件与其他控件如 **ToolBar** 控件一起，可以使你创建经济而又富含信息的用户界面。

**发布须知：** **StatusBar** 控件是 COMCTL32.OCX 文件中自定义控件的一部分。要在应用程序中使用 **StatusBar** 控件，必须将 COMCTL32.OCX 加入到工程文件中。在发布应用程序时，将 COMCTL32.OCX 加入到用户的 Microsoft Windows SYSTEM 目录下。其他信息请参阅《Microsoft Visual Basic 6.0 程序员指南》。

## 属性

**Panels** 属性，**Simple Text** 属性，**Style** 属性（**StatusBar** 控件），**TabIndex** 属性，**Align** 属性，**DragIcon** 属性，**DragMode** 属性，**hWnd** 属性，**MouseIcon** 属性，**Name** 属性，**Parent** 属性，**Font** 属性，**Container** 属性，**ToolTipText** 属性，**WhatsThisHelpID** 属性，**OLEDropMode** 属性（**ActiveX** 控件），**ShowTips** 属性（**ActiveX** 控件），**Height**，**Width** 属性（**ActiveX** 控件），**Index** 属性（**ActiveX** 控件），**Tag** 属性（**ActiveX** 控件），**Visible** 属性（**ActiveX** 控件），**Object** 属性（**ActiveX** 控件），**Enabled** 属性（**ActiveX** 控件），**hWnd** 属性（**ActiveX** 控件），**MousePointer** 属性（**ActiveX** 控件）。

## 方法

Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件)。

## 事件

Panelclick 事件, PanelDbclick 事件, DragDrop 事件, DragOver 事件, MouseDown, MouseUp 事件, MouseMove 事件, Dbclick 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), Click 事件 (ActiveX 控件), Dbclick 事件 (ActiveX 控件)。

## 请参阅

Panels 集合, Panel 对象, Alignment 属性 (HeaderItem, Panel 对象), Bevel 属性 (Panel 对象), Width 属性 (Panel 对象), 使用 StatusBar 控件, Toolbar 控件。

# Add 方法（Panels 集合）

给 Panels 集合增加一个 Panel 对象，并返回对新创建的 Panel 对象的引用。

应用于

StatusBar 控件，Panels 集合

语法

*object.Add(index, key, text, style, picture)*

Add 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 Panels 集合对象的表达式  |
| <i>index</i>  | 可选参数。一个整数，指定对象插入的位置。如果没有指定 <i>index</i> 参数，则 Panel 对象就插入到 Panels 集合的末尾            |
| <b>Key</b>    | 可选参数。标识 Panel 的唯一字符串。使用 <b>key</b> 可以检取某个特定的 Panel 对象，这与新加入对象的 <b>Key</b> 属性值设置相同 |

续表

| 部分             | 描述  |
|----------------|---|
| <i>text</i>    | 可选参数。在 <b>Panel</b> 中出现的字符串。这与新加入对象的 <b>Text</b> 属性值设置相同  |
| <i>style</i>   | 可选参数。 <b>Panel</b> 对象的风格。可用风格列在 <b>Style</b> 属性（ <b>Panel</b> 对象）中。这与新加入对象的 <b>Style</b> 属性设置相同 |
| <i>picture</i> | 可选参数。指定在活动 <b>Panel</b> 对象中显示的点位图。其他信息请参阅 <b>LoadPicture</b> 函数。这与新加入对象的 <b>Picture</b> 属性值设置相同 |

## 说明

在运行时，**Add** 方法返回新增加的 **Panel** 对象的引用，有了该引用，就可以如下方式设置每个新 **Panel** 对象的属性：

```
Dim pnlX As Panel
Dim i As Integer
For i = 1 To 6 ' Add six Panel objects.
    ' Create a panel and get a reference to it simultaneously.
    Set pnlX = StatusBar1.Panels.Add(, "Panel" & i) ' Set Key property.
    pnlX.Style = i ' Set Style property.
    pnlX.AutoSize = sbrContents ' Set AutoSize property.
Next i
```

当 Panel 对象的 Style 属性设置为 sbrText 时，Text 属性值就显示在 Panel 对象中。

Panels 集合是基于 1 的集合。为了获得集合中第一个（缺省）Panel 对象的引用，可以使用 Index 或 Key 属性或使用 Item 方法。下面演示了这些方法

```
Dim pnlX As Panel
```

```
' Get a reference to first Panel.
```

```
Set pnlX = StatusBar1.Panels(1) ' Use the index
```

```
pnlX.Text = "Changed text" ' Alter the Panel object's text.
```

缺省地，控件中已存在一个 Panle。因此，在给集合增加 Panel 对象时，Count 属性值比用户增加的数量大 1。例如：

```
Dim i as Integer
```

```
For i = 1 to 4 ' Add four panels.
```

```
StatusBar1.Panels.Add ' Add panels without any properties.
```

```
Next i
```

```
MsgBox StatusBar1.Panels.Count ' Returns 5 panels.
```

请参阅

StatusBar 控件, Panel 对象, Alignment 属性(HeaderItem, Panel 对象), AutoSize 属性 (Panle 对象), LoadPicture 函数, Picture 属性, Style 属性, Count 属性 (VB 集合), Picture 属性 (ActiveX 控件), Text 属性 (ActiveX 控件), Index

属性（ActiveX 控件），Count 属性（ActiveX 控件），Key 属性（ActiveX 控件）。

## 示例

下面的例子使用 Add 方法给一个 StatusBar 控件增加了 3 个新的 Panel 对象。要使用本例，在一个窗体上放置一个 StatusBar 控件，并将下面的代码拷贝到窗体的 Declarations 段中。运行本例。

```
Private Sub Form_Load()  
Dim pnlX As Panel  
    ' Add blank panel as a spacer  
    Set pnlX = StatusBar1.Panels.Add()  
    pnlX.AutoSize = sbrSpring  
    pnlX.MinWidth = 1  
    ' Add a panel with a clock icon and time style.  
    Set pnlX = StatusBar1.Panels.Add _  
    (, , sbrTime, LoadPicture("Graphics\icons\misc\clock03.ico"))  
    ' Add second panel, with bitmap and Date style.  
    Set pnlX = StatusBar1.Panels.Add _  
    (, , sbrDate, LoadPicture("Graphics\bitmaps\assorted\calendar.bmp"))  
    ' Set Bevel property for last Panel object.  
    pnlX.Bevel = sbrInset    ' Inset bevel.  
    pnlX.Alignment = sbrRight    ' Set Alignment property for last object.
```

```
' Set Text and AutoSize properties for first (default )Panel object.
StatusBar1.Panels(1).Text = "Add Panel Example"
StatusBar1.Panels(1).AutoSize = sbrContents
End Sub
..
```

## Alignment 属性（HeaderItem,Panel 对象）

返回或设置 StatusBar 控件中 Panel 对象标题文本的对齐方式。

应用于

StatusBar 控件， Panel 对象

语法

*object*.Alignment [=*number*]

Alignment 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式         |
| <i>number</i> | 一个常量或值，指定动作的类型，如“设置”中所示 |

# 设置

number 值的设置如下：

| 常量                     | 值 | 描述                   |
|------------------------|---|----------------------|
| sbrLeft<br>hdrLeft     | 0 | （缺省）。文本显示在点位图的右边，左对齐 |
| sbrCenter<br>hdrCenter | 1 | 文本显示在点位图的右边，中间对齐     |
| sbrRight<br>herRight   | 2 | 文本显示在点位图的左边，右对齐      |

# 说明

在定位文本的同时，Alignment 属性也指定点位图的位置，如“设置”中所示。在 Panel 对象中不能单独指定点位图的位置。



请参阅

StatusBar 控件，Add 方法（Panels 集合），Clear 方法（ActiveX 控件）。

## 示例

下面的例子给 StatusBar 控件增加了 2 个新的 Panel 对象，使用 3 种风格之一对齐文本。要使用本例，在一个窗体上放置一个 StatusBar 控件对象，将下面的代码拷贝到窗体的 Declarations 段中。运行本例：

```
Private Sub Form_Load()  
    ' Declare variables.  
    Dim pnlX As Panel  
    Dim I As Integer  
  
    For I = 1 To 2 ' Add two panels.  
        StatusBar1.Panels.Add  
    Next I  
  
    For I = 1 To 3 ' Add pictures to each Panel.  
        Set pnlX = StatusBar1.Panels(I)  
        Set pnlX.Picture = LoadPicture("Graphics\icons\comm\net12.ico")  
        ' Set AutoSize and MinWidth so that panels  
        ' are always in view.
```

```
        pnlX.AutoSize = sbrSpring
        pnlX.MinWidth = 1
    Next I

' Set styles and alignment.
    With StatusBar1.Panels
        .Item(1).Text = "Left"
        .Item(1).Alignment = sbrLeft           ' Left alignment.
        .Item(2).Text = "Center"
        .Item(2).Alignment = sbrCenter        ' Centered alignment.
        .Item(3).Text = "Right"
        .Item(3).Alignment = sbrRight        ' Right alignment.
    End With
End Sub
```

## AutoSize 属性（Panel 对象）

返回或设置一个值，该值确定在 **StatusBar** 控件改变大小之后 **Panel** 对象的宽度。

应用于

StatusBar 控件，Panel 对象

语 法

*object*.AutoSize [=*number*]

AutoSize 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个 Panel 对象的表达式        |
| <i>number</i> | 一个常量或值，指定动作类型，如“设置”中所示 |

设置

*number* 值的设置如下：

| 常量            | 值 | 描述  |
|---------------|---|---|
| sbrNoAutoSize | 0 | （缺省）没有。不自动设置大小。Panel 对象的宽度总是 Width 属性指定的值 |

续表

| 常量          | 值 | 描述  |
|-------------|---|---|
| sbrSpring   | 1 | Spring。当父窗体的大小改变有了更多的空间时，各 Panel 对象将该空间瓜分并相应地调整自己的大小。然而，Panle 对象的宽度总是不低于 MinWidth 属性值 |
| sbrContents | 2 | Content。Panel 重新调整以便显示内容。然而，Panle 对象的宽度总是不低于 MinWidth 属性值                             |

说明

具有 Contents 风格的 Panel 对象比具有 Spring 风格的对象优先级高。这意味着如果具有 Contents 风格的 Panel 对象需要空间的话，Spring 风格的 Panel 对象将缩短自己使用的空间。

请参阅

StatusBar 控件，MinWidth 属性，Width 属性（Panel 对象）。

示例

下面的例子给一个 StatusBar 控件，增加了两个 Panel 对象并将所有 Panel 对象的 AutoSize 属性设置为 Content 风格。当鼠标在窗体上的这些对象上移动

时，就显示 x 和 y 坐标以及控件的 Tag 属性值。要使用本例，在一个窗体上放置一个 StatusBar 控件，一个 PictureBox 控件和一个 CommandButton 控件。将下面的代码拷贝到窗体的 Declarations 段中。运行本例并在各控件上移动鼠标。

```
Private Sub Form_Load()  
    Dim pnlX As Panel  
    ' Set long tags for each object.  
    Form1.Tag = "Project 1 Form"  
    Command1.Tag = "A command button"  
    Picture1.Tag = "Picture Box Caption"  
    StatusBar1.Tag = "Application StatusBar1"  
    ' Set the AutoSize style of the first panel to Contents.  
    StatusBar1.Panels(1).AutoSize = sbrContents  
    ' Add 2 more panels, and set them to Contents.  
    Set pnlX = StatusBar1.Panels.Add  
    pnlX.AutoSize = sbrContents  
    Set pnlX = StatusBar1.Panels.Add  
    pnlX.AutoSize = sbrContents  
End Sub
```

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, x As Single, y As  
Single)  
    ' Display the control's tag in panel 1, and x and y
```

' coordinates in panels 2 and 3. Because AutoSize = Contents,  
' the first panel stretches to accommodate the varying text.

StatusBar1.Panels(1).Text = Form1.Tag

StatusBar1.Panels(2).Text = "X = " & x

StatusBar1.Panels(3).Text = "Y = " & y

End Sub

Private Sub Command1\_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)

StatusBar1.Panels(1).Text = Command1.Tag

StatusBar1.Panels(2).Text = "X = " & x

StatusBar1.Panels(3).Text = "Y = " & y

End Sub

Private Sub Picture1\_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)

StatusBar1.Panels(1).Text = Picture1.Tag

StatusBar1.Panels(2).Text = "X = " & x

StatusBar1.Panels(3).Text = "Y = " & y

End Sub

Private Sub StatusBar1\_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)

```
StatusBar1.Panels(1).Text = StatusBar1.Tag
StatusBar1.Panels(2).Text = "X = " & x
StatusBar1.Panels(3).Text = "Y = " & y
End Sub
```

## Bevel 属性（Panel 对象）

返回或设置 StatusBar 控件中 Panel 对象的斜面风格。

应用于

StatusBar 控件，Panel 对象

语法

*object*.Bevel [=value]

Bevel 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个 Panel 对象的对象表达式      |
| <i>value</i>  | 一个常量或值，指定斜面风格，如“设置”中所示 |

## 设置

*value* 值的设置如下：

| 常量         | 值 | 描述                            |
|------------|---|-------------------------------|
| sbrNoBevel | 0 | 没有。Panel 不显示斜面。文本看起来就象显示在状态条上 |
| sbrInset   | 1 | （缺省）Inset。Panel 看起来象嵌入到了状态条中  |
| sbrRaised  | 2 | Raised。Panel 看起来象在状态条之上       |

请参阅

StatusBar 控件，Add 方法（Panels 集合）。

## 示例

下面的例子给一个 StatusBar 控件增加了两个 Panel 对象，并将每个 Panel 对象分配不同的斜面风格。要使用本例，在一个窗体上放置一个 StatusBar 控件，将斜面的代码拷贝到窗体的 Declarations 段中。

```
Private Sub Form_Load()  
    Dim pnlX As Panel  
    Dim I as Integer
```



```
For I = 1 to 2
    Set pnlX = StatusBar1.Panels.Add()    ' Add 2 panels.
Next I

With StatusBar1.Panels
    .Item(1).Style = sbrCaps ' Caps Lock
    .Item(1).Bevel = sbrInset ' Inset
    .Item(2).Style = sbrNum ' NumLock
    .Item(2).Bevel = sbrNoBevel ' No bevel
    .Item(3).Style = sbrDate ' Date
    .Item(3).Bevel = sbrRaised    ' Raised bevel
End With
End Sub
```

## MinWidth 属性

返回或设置 StatusBar 控件中 Panel 对象的最小宽度。

应用于

StatusBar 控件， Panel 对象

## 语法

*object.MinWidth [=value]*

MinWidth 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个 Panel 对象的对象表达式                            |
| <i>value</i>  | 一个整数，指定 Panel 对象的最小宽度。该值的度量模式取决于包含该控件的容器度量模式 |

## 说明

当 **AutoSize** 属性设置为 **Contents** 或 **Spring** 时使用 **MinWidth** 属性，以便防止 **Panel** 对象将增加的宽度设置得太小。当 **AutoSize** 属性设置为 **None**，则 **MinWidth** 属性值总是设置为 **Width** 属性的值。

该属性的缺省值与 **Width** 属性值一样。该参数使用的度量模式与父窗体或容器的度量模式一样。

请参阅

**StatusBar** 控件，**AutoSize** 属性（**Panel** 对象），**Width** 属性（**Panel** 对象）。

示例

下面的例子使用 **StatusBar** 控件的缺省 **Panel** 对象显示当前的日期。设置了 **MinWidth** 属性值，以便在点击 **Panel** 对象清除日期时仍保持 **Panel** 对象的宽度。要使用本例，在一个窗体上放置一个 **StatusBar** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，用鼠标点击 **Panel** 对象以清除日期。

```
Private Sub Form_Load()  
    StatusBar1.Panels(1).AutoSize = sbrContents  
    StatusBar1.Panels(1).Text = "Today's Date is: " & Str(Now)  
    ' Set minimum width to the current size of panel  
    StatusBar1.Panels(1).MinWidth = StatusBar1.Panels(1).Width  
End Sub
```

```
Private Sub StatusBar1_PanelClick(ByVal Panel As ComctlLib.Panel)  
    ' Clear today's date but keep size at minimum width.  
    Panel.Text = "Today's Date is: "  
End Sub
```

## Panel 对象

一个 Panel 对象代表了 StatusBar 控件中 Panels 集合中的单个格子对象。

### 说明

一个 Panel 对象可以包含文本和点位图，以便显示应用程序的状态。

使用 Panels 集合来检取、增加或删除一个 Panel 对象。

要改变格子的外观，就要改变 Panel 对象的属性值。要在设计时改变 Panel 对象的属性值，可以通过 Properties Page 中的 Panels 卡片来设置 Panel 对象的属性值。在运行时，可以在程序代码中改变 Panel 对象的属性值。

### 属性

Enabled 属性 (ActiveX 控件)，Key 属性 (ActiveX 控件)，Picture 属性 (ActiveX 控件)，Index 属性 (ActiveX 控件)，Left,Top 属性 (ActiveX 控件)，Tag 属性 (ActiveX 控件)，Visible 属性 (ActiveX 控件)，Alignment 属性 (HeaderItem, Panel 对象)，AutoSize 属性 (Panel 对象)，Bevel 属性 (Panel 对象)，MinWidth 属性，Style 属性 (Panel 对象)，Width 属性 (Panel 对象)。

## 方法

**Clear** 方法（ActiveX 控件）。

## 请参阅

**StatusBar** 控件, **Add** 方法(Panels 集合), **Panels** 属性, **Properties** 窗口, **Property Page** 对话框, **Property Pages** 对话框（ActiveX 控件）。

## PanelClick 事件

**PanelClick** 事件与标准的 **Click** 事件一样，只是当用户在 **StatusBar** 控件中的 **Panel** 对象上点击并释放鼠标时产生该事件。

## 应用于

**StatusBar** 控件

# 语法

Private Sub *object*\_PanelClick (ByVal *panel* As Panel)

PanelClick 事件的语法有如下几个部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 一个 StatusBar 控件的对象表达式 |
| <i>panel</i>  | 一个 Panel 对象的引用        |

# 说明

当点击 Panel 对象时也产生标准的 Click 事件。

只有在 Panel 对象上点击鼠标时才产生 PanelClick 事件。当 StatusBar 控件的 Style 属性设置为 Simple 风格时，Panel 对象是隐藏的，因此不触发 PanelClick 事件。

可以使用该 Panel 对象的引用设置该 Panel 对象的属性值。例如，下面的代码设置了所点击 Panel 对象的 Bevel 属性值：

```
Private Sub StatusBar1_PanelClick(ByVal Panel As Panel)
    Select Case Panel.Key
    Case "DisplayFileName" ' Key="DisplayFileName"
        Panel.Bevel = sbrRaised ' Reset Bevel property
```

```
' Add other case statements for other panels
End Select
End Sub
```

请参阅

Panel 对象, Bevel 属性 (Panel 对象), PanelDbClick 事件, Style 属性 (Panel 对象)。

示例

下面的例子给一个 StatusBar 控件增加了两个 Panel 对象；当点击每个 Panel 对象时，在 Panel 中显示 Key 和 Width 属性。要使用本例，在一个窗体上放置一个 StatusBar 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例：

```
Private Sub Form_Load()
    Dim I as Integer
    For I = 1 to 2
        StatusBar1.Panels.Add
    Next I

    With StatusBar1.Panels
        .Item(1).Style = sbrDate
```

```
.Item(1).Key = "Date panel"  
.Item(1).AutoSize = sbrContents  
.Item(1).MinWidth = 2000  
.Item(2).Style = sbrTime  
.Item(2).Key = "Time panel"  
.Item(3).AutoSize = sbrContents ' Content  
.Item(3).Text = "Panel 3"  
.Item(3).Key = "Panel 3"
```

```
End With
```

```
End Sub
```

```
Private Sub StatusBar1_PanelClick(ByVal Panel As Panel)
```

```
    ' Show clicked panel's key and width in Panel 3.
```

```
    StatusBar1.Panels("Panel 3").Text = Panel.Key & " Width = " & Panel.Width
```

```
End Sub
```

## PanelDblClick 事件

PanelDblClick 事件与标准的 DblClick 事件一样，只是在 StatusBar 控件中的 Panel 对象中双击鼠标时才产生该事件。



应用于

StatusBar 控件。

语法

Sub *object*\_PanelDbClick (ByVal *panel* As Panel )

PanelDbClick 事件的语法有如下几个部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 一个 StatusBar 控件的对象表达式 |
| <i>panel</i>  | 一个 Panel 对象的引用        |

说明

当双击 Panel 对象时也产生一个 DbClick 事件。

只有在 Panel 对象上双击鼠标时才产生 PanelDbClick 事件。当 StatusBar 控件的 Style 属性设置为 Simple 时, Panel 对象是隐藏的, 因此不产生 PanelDbClick 事件。

请参阅

Panel 对象，PanelDbClick 事件，Style 属性（Panel 对象）

示例

下面的例子给一个 **StatusBar** 控件增加了两个 **Panel** 对象。当用户双击控件时，就显示所点击 **Panel** 对象的文本。要使用本例，在一个窗体上放置一个 **StatusBar** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，双击控件。

```
Private Sub Form_Load()  
Dim I as Integer  
    For I = 1 to 2  
        StatusBar1.Panels.Add  
    Next I  
  
    With StatusBar1.Panels  
        .Item(1).Text = "A long piece of information."  
        .Item(1).AutoSize = sbrContents ' Content  
        .Item(2).Style = sbrDate ' Date style  
        .Item(2).AutoSize = sbrContents ' Content  
        .Item(3).Style = sbrTime ' Time style
```

```
End With
End Sub

Private Sub StatusBar1_PanelDbClick(ByVal Panel As Panel)
    MsgBox "Panel.Style = " & Panel.Style
End Sub
```

## Panels 集合

Panels 集合包含了 Panel 对象的集合。

### 语法

*statusbar.Panels (index)*

Panels 集合的语法有如下几个部分：

| 部分               | 描述   |
|------------------|--|
| <i>statusbar</i> | 一个 StatusBar 控件的对象表达式                                  |
| <i>index</i>     | 一个整数或字符串，唯一标识集合中的一个对象。整数是对象的 Index 属性值；字符串是对象的 Key 属性值 |

## 说明

**Panels** 集合是基于 1 的 **Panel** 对象矩阵。缺省地，在 **StatusBar** 控件中已经存在一个 **Panel** 对象。因此，如果想创建有 3 个格子的状态条，只需给 **Panels** 集合增加两个 **Panel** 对象。

**Panels** 属性返回对 **Panels** 集合的引用。

要给一个 **Panels** 集合增加对象，在运行时使用 **Panel** 对象的 **Add** 方法；或在设计时使用 **StatusBar** 控件的 **Properties Page** 对话框中的 **Panels** 卡片。

集合中的每个成员可以通过 **Index** 属性或 **Key** 属性来访问。例如，要获取集合中第三个 **Panel** 对象的引用，使用下面的代码：

```
Dim pnlX As Panel
Set pnlX = StatusBar1.Panels(3)    ' Reference by index number.
' or
Set pnlX = StatusBar1.Panels("Third") ' Reference by unique key.
' or
Set pnlX = StatusBar1.Panels.Item(3) ' Use Item method.
```

## 属性

**Count** 属性（**ActiveX** 控件），**Item** 属性（**ActiveX** 控件）。

## 语法

**Clear** 方法 (ActiveX 控件), **Remove** 方法 (ActiveX 控件), **Add** 方法 (Panels 集合)。

## 请参阅

**Panel** 对象, **Add** 方法 (Panels 集合), **Panels** 属性, **Property Pages** 对话框, **Index** 属性 (ActiveX 控件), **Property Pages** 对话框 (ActiveX 控件), **Key** 属性 (ActiveX 控件)

## Panels 属性

返回一个 **Panel** 集合的引用。

## 应用于

**StatusBar** 控件

语法

*object*.Panels

*object* 是一个 StatusBar 控件的对象表达式。

请参阅

Panels 集合，Panel 对象，AutoSize 属性（Panel 对象），Width 属性（Panel 对象）

## SimpleText 属性

当 StatusBar 控件的 Style 属性设置为 Simple 时，返回或设置所显示的文本。

应用于

StatusBar 控件

## 语法

*object*.SimpleText [=string]

SimpleText 属性的语法有如下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个 StatusBar 控件的对象表达式           |
| <i>string</i> | 当 Style 属性设置为 Simple 时所显示的文本字符串 |

## 说明

StatusBar 控件有一个 Style 属性，可以在 Simple 和 Normal 风格之间切换。当使用 Simple 风格时，状态条只显示一个格子。在 Simple 风格中显示的文本也与 Normal 风格中的不一样。该文本使用 SimpleText 属性设置。

在应用程序的操作模式临时切换时使用 SimpleText 属性。例如，当下拉一个菜单时，SimpleText 可以描述菜单的目的。

## 请参阅

Style 属性（StatusBar 控件）

## 示例

下面的例子给一个 **StatusBar** 控件增加了两个 **Panel** 对象，设置为 **Normal** 风格，然后使用 **SimpleText** 属性增加了字符串，当 **Style** 属性设置为 **Simple** 时就显示该字符串。控件可以在 **Simple** 和 **Normal** 风格之间切换。要使用本例，在一个窗体上放置一个 **StatusBar** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击 **StatusBar** 控件。

```
Private Sub Form_Load()  
    Dim I As Integer  
    For I = 1 to 2  
        StatusBar1.Panels.Add ' Add 2 Panel objects.  
    Next I  
  
    With StatusBar1.Panels  
        .Item(1).Style = sbrNum ' Number lock  
        .Item(2).Style = sbrCaps ' Caps lock  
        .Item(3).Style = sbrScrl ' Scroll lock  
    End With  
End Sub  
  
Private Sub StatusBar1_Click()  
    ' Toggle between simple and normal style.
```



```
With StatusBar1
    If .Style = 0 Then
        ' This text will be displayed when the StatusBar is in Simple style.
        .SimpleText = "Date and Time: " & Now
        .Style = sbrSimple ' Simple style.
    Else
        .Style = sbrNormal ' Normal style.
    End If
End With
End Sub
```

## Style 属性（Panel 对象）

返回或设置 StatusBar 控件的 Panel 对象的风格。

应用于

StatusBar 控件，Panel 对象

# 语法

*object.Style* [=*number*]

Style 属性的语法有如下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个 Panel 对象的对象表达式               |
| <i>number</i> | 一个整数或常量，指定 Panel 对象的风格，如“设置”中所示 |

# 设置

*number* 值的设置如下：

| 常量      | 值 | 描述   |
|---------|---|--|
| sbrText | 0 | （缺省）显示文本和/或点位图。使用 Text 属性设置文本                            |
| sbrCaps | 1 | 显示 Caps 键的状态。当 Caps Lock 允许使用时，以黑体显示 CAPS 字符，当禁止时，则以灰体显示 |
| sbrNum  | 2 | 显示 Num 键的状态。当数字键允许使用时，以黑体显示 NUM 字符，当禁止时，以灰体显示            |
| sbrIns  | 3 | 显示 Insert 键的状态。当插入键允许使用时，以黑体显示 INS 字符；当禁止时，以灰体显示         |

续表

| 常量       | 值 | 描述   |
|----------|---|--|
| sbrScrl  | 4 | 显示 Scroll Lock 键的状态。当 Scroll Lock 允许使用时，以黑体显示 SCRL 字符；当禁止时，以灰体显示 |
| sbrTime  | 5 | 时间，以系统格式显示当前时间   |
| sbrDates | 6 | 日期，以系统格式显示当前日期   |
| sbrKana  | 7 | Kana。当 Scroll Lock 键使能时以黑体显示 KANA；该键禁止时将 KANA 变灰                 |

说明

如果将 Style 属性设置为除 0（文本和点位图）之外的其他风格，则不显示 Text 属性中的文本，除非将 Style 属性设置为 0。

在给集合增加 Panel 对象时也可以设置 Style 属性。请参阅 Add 方法。

注意：StatusBar 控件也有 Style 属性。当 StatusBar 控件的 Style 属性设置为 Simple 时，控件只显示一个大的格子及其字符串(SimpleText 属性值)。

请参阅

Panels 集合, Add 方法(Panels 集合), SimpleText 属性, Style 属性(StatusBar

控件），Text 属性（ActiveX 控件），StatusBar 控件

## 示例

下面的例子在 **StatusBar** 控件中以不同的风格显示数据。要使用本例，在一个窗体上放置一个 **StatusBar** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例。

```
Private Sub Form_Load()  
    ' Dim variables.  
    Dim I as Integer  
    Dim pnlX as Panel  
  
    For I = 1 to 5 ' Add 5 panels.  
        Set pnlX = StatusBar1.Panels.Add( )  
    Next I  
  
    ' Set the style of each panel.  
    With StatusBar1.Panels  
        .Item(1).Style = sbrDate ' Date  
        .Item(2).Style = sbrTime ' Time  
        .Item(3).Style = sbrCaps ' Caps lock  
        .Item(4).Style = sbrNum ' Number lock  
        .Item(5).Style = sbrIns ' Insert key
```

```
.Item(6).Style = sbrScrl ' Scroll lock
End With
Form1.Width = 9140 ' Widen form to show all panels.
End Sub
```

## Style 属性（StatusBar 控件）

返回或设置 StatusBar 控件的风格。

应用于

StatusBar 控件

语法

*object*.Style [=*number*]

Style 属性的语法有如下几个部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 一个 StatusBar 控件的对象表达式 |

续表

| 部分            | 描述   |
|---------------|--|
| <i>number</i> | 一个整数或常量，指定 <b>StatusBar</b> 控件的风格，如“设置”中所示 |

设置

*number* 值的设置如下：

| 常量        | 值 | 描述   |
|-----------|---|--|
| sbrNormal | 0 | （缺省）Normal。 <b>StatusBar</b> 控件显示所有的 <b>Panel</b> 对象 |
| sbrSimple | 1 | Simple。控件只显示一个大的格子                                   |

说明

**StatusBar** 控件可以在两种模式之间切换：**Normal** 和 **Simple**。在 **Simple** 风格中，**StatusBar** 控件只显示一个大的格子。外观也发生了改变：斜面风格是 **Raised**，没有边框。这允许控件有两种外观，每个都与另一个不同。

根据控件的风格，可以显示不同的字符串。当 **Style** 属性设置为 **Simple** 时，使用 **SimpleText** 属性设置要显示的文本字符串。

注意：**Style** 属性设置为 **Simple** 时，**StatusBar** 控件显示一个大的格子（控件宽度），其不能通过 **Panels** 集合进行控制。

请参阅

Panel 对象, SimpleText 属性, Style 属性 (Panel 对象)

示例

下面的例子给一个 StatusBar 控件增加了两个 Panel 对象, 设置为 Normal 风格, 然后使用 SimpleText 属性设置了字符串, 当 Style 属性设置为 Simple 时显示该字符串。控件在 Normal 和 Simple 风格之间切换以显示 SimpleText 属性字符串。要使用本例, 在一个窗体上放置一个 StatusBar 控件, 将下面的代码拷贝到窗体的 Declarations 段中。允许本例, 点击 StatusBar 控件。

```
Private Sub Form_Load()  
    Dim I As Integer  
    For I = 1 to 2  
        StatusBar1.Panels.Add  
    Next I  
    With StatusBar1.Panels  
        .Item(1).Style = sbrDate ' Date  
        .Item(2).Style = sbrCaps ' Caps lock  
        .Item(3).Style = sbrScrl ' Scroll lock  
    End With
```

End Sub

Private Sub StatusBar1\_Click()

With StatusBar1

If .Style = sbrNormal Then

.SimpleText = Time' Show the time.

.Style = sbrSimple ' Simple style

Else

.Style = sbrNormal ' Normal style

End If

End With

End Sub

## Width 属性（Panel 对象）

返回或设置 StatusBar 控件中 Panel 对象的当前宽度。

应用于

StatusBar 控件，Panle 对象



## 语法

*object*.Width [=*number*]

Width 属性的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 一个 Panel 对象的对象表达式   |
| <i>number</i> | 一个整数，确定 Panel 对象的宽度 |

## 说明

Width 属性总是反映一个 Panel 对象的实际宽度，不能比 MinWidth 属性值小。

## 请参阅

StatusBar 控件，MinWidth 属性

## 示例

下面的例子创建了 3 个 Panel 对象，并将 Width 属性设置为不同的值。当点击窗体时，重新设置第一个 Panel 对象的 Width 属性。要使用本例，在一个窗

体上放置一个 **StatusBar** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。  
运行本例，点击每个 **Panel** 对象以查看其宽度。

```
Private Sub Form_Load()  
    Dim X As Panel  
    Dim I as Integer  
    For I = 1 to 2 ' Add 2 panels.  
        Set X = StatusBar1.Panels.Add()  
    Next I  
    With StatusBar1.Panels  
        .Item(1).Text = "Path = " & App.Path  
        .Item(1).AutoSize = sbrContents ' Contents  
        .Item(1).Width = 2000           ' A long panel  
        .Item(2).Text = "Record Field"  
        .Item(2).AutoSize = sbrSpring   ' Spring  
        .Item(2).Width = 1000           ' A medium panel  
        .Item(3).Style = sbrTime        ' Time  
        .Item(3).AutoSize = sbrSpring   ' Spring  
        .Item(3).Width = 500            ' A medium panel  
    End With  
End Sub  
  
Private Sub Form_Click()
```

```
' Change Width.
```

```
StatusBar1.Panels(1). Width = 800
```

```
End Sub
```

# SysInfo 控件

**SysInfo** 控件用来响应操作系统发送给所有应用程序的系统消息。于是应用程序在必要时就可适应操作系统的变化。

## 语法

**SysInfo**

## 说明

用 **SysInfo** 控件可监视操作系统信息，还可响应系统事件。该控件的特点如下所列：

- 与系统变化相关的事件（例如，**DisplayChanged**、**TimeChanged** 以及 **SettingChanged** 事件）。
- 电源状态事件及属性（例如，**PowerSuspend**、**PowerResume** 事件及 **ACStatus** 和 **BatteryStatus** 属性）。

- Plug 和 Play 事件（例如，DeviceArrival 和 DeviceRemoveComplete 事件）。
- 操作系统属性（例如，OSVersion 和 WorkAreaHeight 属性）。

## 属性

ACStatus 属性, BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLiteTime 属性, BatteryStatus 属性, Name 属性, OSBuild 属性, OSPlatform 属性, OSVersion 属性, ScrollBarSize 属性, Tag 属性, WorkAreaHeight 属性, WorkAreaLeft 属性, WorkAreaTop 属性, WorkAreaWidth 属性, Parent 属性, Index 属性（ActiveX 控件）, Tag 属性（ActiveX 控件）, Object 属性（ActiveX 控件）

## 事件

ConfigChangeCancelled 事件, ConfigChanged 事件, DeviceArrival 事件, DeviceOtherEvent 事件, DeviceQueryRemoveFailed 事件, DeviceRemoveComplete 事件, DeviceRemovePending 事件, DevModeChange 事件, DisplayChanged 事件, PowerQuerySuspend 事件, PowerResume 事件, PowerStatusChanged 事件, PowerSuspend 事件, QueryChangeConfig 事件, SettingChanged 事件, SysColorsChanged 事件, TimeChanged 事件

请参阅

使用 SysInfo 控件

## ACStatus 属性

返回一值，指示系统是否使用 AC 电源。

应用于

SysInfo 控件

语法

*object*.ACStatus

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

设置

ACStatus 属性的设置值如下：

| 设置值 | 描述           |
|-----|--------------|
| 0   | 系统未使用 AC 电源  |
| 1   | 系统正在使用 AC 电源 |
| 255 | AC 电源状态未知    |

请参阅

BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLifeTime 属性, BatteryStatus 属性, SysInfo 控件

示例

该示例用窗体中的 Label 控件显示系统的 AC 电源状态。要运行此例, 请在窗体中放置 SysInfo 控件、Label 控件及 Timer 控件, 然后将下列代码粘贴到 Timer 控件的 Timer 事件中, 再将 Timer 控件的 Interval 属性值设置成 5000, 然后运行此例。

```
Private Sub Timer1_Timer()  
    Select Case SysInfo1.ACStatus  
        Case 0  
            Label1.Caption = "AC Power: Off"  
        Case 1
```

```
        Label1.Caption = "AC Power: On"  
    Case 255  
        Label1.Caption = "AC Power: Unknown"  
    End Select  
End Sub
```

## BatteryFullTime 属性

返回一值，指示电池在完全充电状态下的使用寿命。

应用于

SysInfo 控件

语法

*object*.BatteryFullTime

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。



## 说明

**BatteryFullTime** 属性返回电池在完全充电状态下以秒为单位的使用寿命。如果不知道寿命值，则返回值 **&HFFFFFFF**。

## 请参阅

**ACStatus** 属性, **BatteryLifePercent** 属性, **BatteryLifeTime** 属性, **BatteryStatus** 属性, **SysInfo** 控件

## 示例

该示例以小时和分的形式显示电池的使用寿命。要运行此例，请在窗体中放置 **SysInfo** 控件及 **CommandButton** 控件，将下列代码粘贴到 **CommandButton** 控件的 **Click** 事件中，然后运行此例。

```
Private Sub Command1_Click()  
    ' BatteryLifeTime 属性示例  
    If SysInfo1.BatteryLifeTime <> &HFFFFFFF Then  
        Dim TimeLeft As String  
        Dim TimeTotal As String  
        Dim temp
```

```
temp = TimeSerial(0, 0, SysInfo1.BatteryLifeTime)
TimeLeft = Format(temp, "h:mm")
temp = TimeSerial(0, 0, SysInfo1.BatteryFullTime)
TimeTotal = Format(temp, "h:mm")
MsgBox TimeLeft & " time left from " & TimeTotal & " total."
Else
    MsgBox "Cannot determine remaining battery time."
End If
End Sub
```

## BatteryLifePercent 属性

返回电池所剩能量占完全充电时能量的百分比。

应用于

SysInfo 控件

## 语法

*object*.BatteryLifePercent

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

## 设置值

BatteryLifePercent 属性的设置值如下：

| 设置值     | 描述         |
|---------|------------|
| 0 – 100 | 电池所剩能量的百分比 |
| 255     | 电池充电状态未知   |

## 请参阅

ACStatus 属性，BatteryFullTime 属性，BatteryLifeTime 属性，BatteryStatus 属性，SysInfo 控件

## 示例

该示例用窗体中的 ProgressBar 控件显示系统电池剩余能量的状态。要运行此例，请在窗体中放置 SysInfo 控件、ProgressBar 控件以及 Timer 控件将

下列代码粘贴到 **Timer** 控件的 **Timer** 事件中，再将 **Timer** 控件的 **Interval** 属性值置为 **5000**，然后运行此例。

```
Private Sub Timer1_Timer()  
    If SysInfo1.BatteryLifePercent <> 255 Then  
        ProgressBar1.Value = SysInfo1.BatteryLifePercent  
        ProgressBar1.Enabled = True  
    Else  
        ProgressBar1.Value = 0  
        ProgressBar1.Enabled = False  
    End If  
End Sub
```

## BatteryLifeTime 属性

返回一值，指示电池的剩余寿命。

应用于

**SysInfo** 控件

## 语法

*object*.BatteryLifeTime

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

## 说明

**BatteryLifeTime** 属性返回电池以秒为单位的剩余寿命值。如果电池的剩余寿命未知，则返回值 **&HFFFFFFF**。

## 请参阅

**ACStatus** 属性，**BatteryFullTime** 属性，**BatteryLifePercent** 属性，**BatteryStatus** 属性，**SysInfo** 控件

## 示例

该示例以小时和分的形式显示电池的剩余寿命值。要运行此例，请在窗体中放置 **SysInfo** 控件及 **CommandButton** 控件将下列代码粘贴到 **CommandButton** 控件的 **Click** 事件中，然后运行此例。

```
Private Sub Command1_Click()
```

' BatteryLifeTime 属性示例

If SysInfo1.BatteryLifeTime <> &HFFFFFFF Then

Dim TimeLeft As String

Dim TimeTotal As String

Dim temp

temp = TimeSerial(0, 0, SysInfo1.BatteryLifeTime)

TimeLeft = Format(temp, "h:mm")

temp = TimeSerial(0, 0, SysInfo1.BatteryFullTime)

TimeTotal = Format(temp, "h:mm")

MsgBox TimeLeft & " time left from " & TimeTotal & " total."

Else

MsgBox "Cannot determine remaining battery time."

End If

End Sub

## BatteryStatus 属性

返回一值，指示电池的充电状态。

应用于

SysInfo 控件

语法

*object*.BatteryStatus

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

设置

BatteryStatus 属性的设置值如下：

| 设置值 | 描述         |
|-----|------------|
| 1   | 电池电量充足     |
| 2   | 电池电量不足     |
| 4   | 电池电量处于临界状态 |
| 8   | 电池正在充电     |
| 128 | 系统无电池      |
| 255 | 电池充电状态未知   |

请参阅

ACStatus 属性, BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLifeTime 属性, SysInfo 控件。

## 示例

该示例在电源状态发生变化时更新 Label 控件，使应用程序及时显示当前的电池状态信息。要运行此例，请在窗体中放置 SysInfo 控件和 Label 控件将此代码粘贴到 SysInfo 控件的 PowerStatusChanged 事件中，然后运行此例。

```
Private Sub SysInfo1_PowerStatusChanged()  
    Select Case SysInfo1.BatteryStatus  
        Case 1  
            Label1.Caption = "Battery OK"  
        Case 2  
            Label1.Caption = "Battery Low"  
        Case 4  
            Label1.Caption = "Battery Critical"  
        Case 8  
            Label1.Caption = "Battery Charging"  
        Case 128, 255
```



```
        Label1.Caption = "No Battery Status"  
    End Select  
End Sub
```

## ConfigChangeCancelled 事件

当操作系统向所有应用程序发送一个取消硬件配置文件变化的消息时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_ConfigChangeCancelled([index As Integer])
```

ConfigChangeCancelled 事件的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |

续表。

| 部分           | 描述                               |
|--------------|----------------------------------|
| <i>index</i> | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

说明

应该用此事件清除在 QueryChangeConfig 事件过程中开始的任何处理。

请参阅

ConfigChanged 事件，QueryChangeConfig 事件，SysInfo 控件

ConfigChanged 事件

当系统硬件配置文件发生改变时，该事件发生。

应用于

SysInfo 控件

# 语法

Private Sub *object*\_ConfigChanged(*[index* As Integer,] ByVal *oldconfignum* As Long, ByVal *newconfignum* As Long)

ConfigChanged 事件包含下面部分：

| 部分                  | 描述                               |
|---------------------|----------------------------------|
| <i>object</i>       | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>        | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>oldconfignum</i> | 对旧系统配置，Windows 注册表中的关键字名称        |
| <i>newconfignum</i> | 对新系统配置，Windows 注册表中的关键字名称        |

# 说明

为连接或断开便携机，当系统恢复操作后，该事件发生。

# 请参阅

ConfigChangeCancelled 事件，QueryChangeConfig 事件，SysInfo 控件

# DeviceArrival 事件

当添加新设备到系统中时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_DeviceArrival([index As Integer,] ByVal devicetype As Long,
ByVal deviceid As Long, ByVal devicename As String, ByVal devicedata As Long)
```

DeviceArrival 事件的语法包含下面部分：

| 部分                | 描述                               |
|-------------------|----------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>      | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>devicetype</i> | 一个值，指示已添加设备的类型，详见“设置”            |
| <i>deviceid</i>   | 识别设备的值，详见“设置”                    |

续表

| 部分                | 描述   |
|-------------------|--|
| <i>devicename</i> | 对 DeviceTypePort 之外的所有 <i>devicetype</i> 设置值均为 Null。当 <i>devicetype</i> 值为 DeviceTypePort 时, <i>devicename</i> 为 dbcp_name |
| <i>devicedata</i> | “设置”中描述的值  |

设置

*devicetype* 的设置值如下:

| 常量                | 值 | 描述                     |
|-------------------|---|------------------------|
| DeviceTypeOEM     | 0 | OEM 定义的设备类型            |
| DeviceTypeDevNode | 1 | Devnode 号 (Windows 95) |
| DeviceTypeVolume  | 2 | 逻辑卷标 (磁盘驱动器)           |
| DeviceTypePort    | 3 | 串行或并行端口                |
| DeviceTypeNet     | 4 | 网络资源                   |

*deviceid* 的设置值如下:

| <i>devicetype</i> 设置值 | <i>deviceid</i> 设置值 |
|-----------------------|---------------------|
| DeviceTypeOEM         | dbco_identifier     |
| DeviceTypeDevNode     | dbcd_devnode        |
| DeviceTypeVolume      | dbcv_unitmask       |

续表

| <i>devicetype</i> 设置值 | <i>deviceid</i> 设置值 |
|-----------------------|---------------------|
| DeviceTypePort        | Null                |
| DeviceTypeNet         | dbcn_resource       |

*devicedata* 的设置值如下：

| <i>devicetype</i> 设置值 | <i>devicedata</i> 设置值 |
|-----------------------|-----------------------|
| DeviceTypeOEM         | dbco_supfunc          |
| DeviceTypeDevNode     | Null                  |
| DeviceTypeVolume      | dbcv_flags            |
| DeviceTypePort        | Null                  |
| DeviceTypeNet         | dbcn_flags            |

说明

如果应用程序可以动态地使用新硬件，则此事件非常有用。

请参阅

DeviceOtherEvent 事件, DeviceQueryRemove 事件, DeviceQueryRemoveFailed 事件, DeviceRemoveComplete 事件, DeviceRemovePending 事件, SysInfo 控件

# DeviceOtherEvent 事件

该事件是一个通知事件，不会映射到通用事件中。

应用于

SysInfo 控件

语法

```
Private Sub object_DeviceOtherEvent([index As Integer,] ByVal devicetype As Long, ByVal eventname As String, ByVal datapointer As Long)
```

DeviceOtherEvent 事件的语法包含下面部分：

| 部分                  | 描述                               |
|---------------------|----------------------------------|
| <i>object</i>       | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>        | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>devicetype</i>   | 一个值，指示处理事件的设备类型，详见“设置”           |
| <i>eventname</i>    | 字符串表达式，其值是事件的名称                  |
| <i>datapointerd</i> | 指向设备专用数据的长值                      |

# 设置

*devicetype* 的设置值如下：

| 常量                | 值 | 描述                     |
|-------------------|---|------------------------|
| DeviceTypeOEM     | 0 | OEM 定义的设备类型            |
| DeviceTypeDevNode | 1 | Devnode 号 (Windows 95) |
| DeviceTypeVolume  | 2 | 逻辑卷标 (磁盘驱动器)           |
| DeviceTypePort    | 3 | 串行或并行端口                |
| DeviceTypeNet     | 4 | 网络资源                   |

请参阅

DeviceArrival 事件，DeviceQueryRemove 事件，DeviceQueryRemoveFailed 事件，DeviceRemoveComplete 事件，DeviceRemovePending 事件，SysInfo 控件

## DeviceQueryRemove 事件

从系统中删除设备之前，该事件发生。



应用于

SysInfo 控件

语法

```
Private Sub object_DeviceQueryRemove([index As Integer,] ByVal devicetype
As Long, ByVal deviceid As Long, ByVal devicename As String, ByVal devicedata
As Long, cancel As Boolean)
```

DeviceQueryRemove 事件的语法包含下面部分：

| 部分                | 描述  |
|-------------------|---|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象  |
| <i>index</i>      | 数字表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引  |
| <i>devicetype</i> | 一个值，指示已添加设备的类型，详见“设置”   |
| <i>deviceid</i>   | 识别设备的值，详见“设置”   |
| <i>devicename</i> | 对 DeviceTypePort 之外的所有 <i>devicetype</i> 设置值均为 Null。当 <i>devicetype</i> 值为 DeviceTypePort 时， <i>devicename</i> 为 <i>dbcp_name</i> |
| <i>devicedata</i> | “设置”中描述的值   |
| <i>cancel</i>     | 布尔表达式，详见“设置”  |

# 设置

*devicetype* 的设置值如下:

| 常量                | 值 | 描述                    |
|-------------------|---|-----------------------|
| DeviceTypeOEM     | 0 | OEM 定义的设备类型           |
| DeviceTypeDevNode | 1 | Devnode 号(Windows 95) |
| DeviceTypeVolume  | 2 | 逻辑卷标 (磁盘驱动器)          |
| DeviceTypePort    | 3 | 串行或并行端口               |
| DeviceTypeNet     | 4 | 网络资源                  |

*deviceid* 的设置值如下:

| <i>devicetype</i> 设置值 | <i>deviceid</i> 设置值 |
|-----------------------|---------------------|
| DeviceTypeOEM         | dbco_identifier     |
| DeviceTypeDevNode     | dbcd_devnode        |
| DeviceTypeVolume      | dbcv_unitmask       |
| DeviceTypePort        | Null                |
| DeviceTypeNet         | dbcn_resource       |

*devicedata* 的设置值如下:

| <i>devicetype</i> 设置值 | <i>devicedata</i> 设置值 |
|-----------------------|-----------------------|
| DeviceTypeOEM         | dbco_suppfunc         |

续表

| <i>devicetype</i> 设置值 | <i>devicedata</i> 设置值 |
|-----------------------|-----------------------|
| DeviceTypeDevNode     | Null                  |
| DeviceTypeVolume      | dbcv_flags            |
| DeviceTypePort        | Null                  |
| DeviceTypeNet         | dbcn_flags            |

*cancel* 的设置值如下：

| 设置值   | 描述       |
|-------|----------|
| True  | 系统禁止删除设备 |
| False | 系统允许删除设备 |

说明

在删除设备之前，对删除设备作出响应的进程向操作系统发送必要的信息，该事件仅在此时发生。

请参阅

DeviceArrival 事件，DeviceOtherEvent 事件，DeviceQueryRemoveFailed 事件，DeviceRemoveComplete 事件，DeviceRemovePending 事件，SysInfo 控件

# DeviceQueryRemoveFailed 事件

当 DeviceQueryRemove 事件代码取消了删除设备的操作时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_DeviceQueryRemoveFailed([index As Integer,] ByVal devicetype As Long, ByVal deviceid As Long, ByVal devicename As String, ByVal devicedata As Long, cancel As Boolean)
```

DeviceQueryRemoveFailed 事件的语法包含下面部分：

| 部分                | 描述                               |
|-------------------|----------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>      | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>devicetype</i> | 一个值，指示已添加设备的类型，详见“设置”            |

续表

| 部分                | 描述   |
|-------------------|--|
| <i>deviceid</i>   | 识别设备的值，详见“设置”  |
| <i>devicename</i> | 对 DeviceTypePort 之外的所有 <i>devicetype</i> 设置值均为 Null。<br>当 <i>devicetype</i> 值为 DeviceTypePort 时， <i>devicename</i> 为 dbcp_name |
| <i>devicedata</i> | “设置”中描述的值  |
| <i>cancel</i>     | 布尔表达式，详见“设置”   |

设置

*devicetype* 的设置值如下：

| 常量                | 值 | 描述                     |
|-------------------|---|------------------------|
| DeviceTypeOEM     | 0 | OEM 定义的设备类型            |
| DeviceTypeDevNode | 1 | Devnode 号 (Windows 95) |
| DeviceTypeVolume  | 2 | 逻辑卷标（磁盘驱动器）            |
| DeviceTypePort    | 3 | 串行或并行端口                |
| DeviceTypeNet     | 4 | 网络资源                   |

*deviceid* 的设置值如下：

| <i>devicetype</i> 设置值 | <i>deviceid</i> 设置值 |
|-----------------------|---------------------|
| DeviceTypeOEM         | dbco_identifier     |
| DeviceTypeDevNode     | dbcd_devnode        |
| DeviceTypeVolume      | dbcv_unitmask       |
| DeviceTypePort        | Null                |
| DeviceTypeNet         | dbcn_resource       |

*devicedata* 的设置值如下：

| <i>devicetype</i> 设置值 | <i>devicedata</i> 设置值 |
|-----------------------|-----------------------|
| DeviceTypeOEM         | dbco_suppfunc         |
| DeviceTypeDevNode     | Null                  |
| DeviceTypeVolume      | dbcv_flags            |
| DeviceTypePort        | Null                  |
| DeviceTypeNet         | dbcn_flags            |

*cancel* 的设置值如下：

| 设置值   | 描述       |
|-------|----------|
| True  | 系统禁止删除设备 |
| False | 系统允许删除设备 |

请参阅

DeviceArrival 事件, DeviceOtherEvent 事件, DeviceQueryRemove 事件, DeviceRemoveComplete 事件, DeviceRemovePending 事件, SysInfo 控件

## DeviceRemoveComplete 事件

该事件在删除设备后发生。

应用于

SysInfo 控件

语法

```
Private Sub object_DeviceQueryRemove([index As Integer,] ByVal devicetype As Long, ByVal deviceid As Long, ByVal devicename As String, ByVal devicedata As Long, cancel As Boolean)
```

DeviceRemoveComplete 事件的语法包含下面部分：

| 部分                | 描述   |
|-------------------|--|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象   |
| <i>index</i>      | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引   |
| <i>devicetype</i> | 一个值，指示已添加设备的类型，详见“设置”  |
| <i>deviceid</i>   | 识别设备的值，详见“设置”  |
| <i>devicename</i> | 对 DeviceTypePort 之外的所有 <i>devicetype</i> 设置值均为 Null。<br>当 <i>devicetype</i> 值为 DeviceTypePort 时， <i>devicename</i> 为 dbcp_name |
| <i>devicedata</i> | “设置”中描述的值  |
| <i>cancel</i>     | 布尔表达式值，详见“设置”  |

## 设置

*devicetype* 的设置值如下：

| 常量                | 值 | 描述                     |
|-------------------|---|------------------------|
| DeviceTypeOEM     | 0 | OEM 定义的设备类型            |
| DeviceTypeDevNode | 1 | Devnode 号 (Windows 95) |
| DeviceTypeVolume  | 2 | 逻辑卷标 (磁盘驱动器)           |
| DeviceTypePort    | 3 | 串行或并行端口                |
| DeviceTypeNet     | 4 | 网络资源                   |

*deviceid* 的设置值如下：



| <i>devicetype</i> 设置值 | <i>deviceid</i> 设置值 |
|-----------------------|---------------------|
| DeviceTypeOEM         | dbco_identifier     |
| DeviceTypeDevNode     | dbcd_devnode        |
| DeviceTypeVolume      | dbcv_unitmask       |
| DeviceTypePort        | Null                |
| DeviceTypeNet         | dbcn_resource       |

*devicedata* 的设置值如下：

| <i>devicetype</i> 设置值 | <i>devicedata</i> 设置值 |
|-----------------------|-----------------------|
| DeviceTypeOEM         | dbco_suppfunc         |
| DeviceTypeDevNode     | Null                  |
| DeviceTypeVolume      | dbcv_flags            |
| DeviceTypePort        | Null                  |
| DeviceTypeNet         | dbcn_flags            |

*cancel* 的设置值如下：

| 设置值   | 描述       |
|-------|----------|
| True  | 系统禁止删除设备 |
| False | 系统允许删除设备 |

## 说明

在某些情况下，即使没有其它的设备删除事件发生，也可能产生该事件。

请参阅

DeviceArrival 事件, DeviceOtherEvent 事件, DeviceQueryRemove 事件, DeviceQueryRemoveFailed 事件, DeviceRemovePending 事件, SysInfo 控件

## DeviceRemovePending 事件

当所有应用程序都同意删除某设备且准备将设备删除之时, 该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_DeviceRemovePending([index As Integer,] ByVal devicetype As Long, ByVal deviceid As Long, ByVal devicename As String, ByVal devicedata As Long, cancel As Boolean)
```

DeviceRemovePending 事件的语法包含下面部分:

| 部分                | 描述   |
|-------------------|--|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象   |
| <i>index</i>      | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引   |
| <i>devicetype</i> | 一个值，指示已添加设备的类型，详见“设置”  |
| <i>deviceid</i>   | 识别设备的值，详见“设置”  |
| <i>devicename</i> | 对 DeviceTypePort 之外的所有 <i>devicetype</i> 设置值均为 Null。当 <i>devicetype</i> 值为 DeviceTypePort 时， <i>devicename</i> 为 dbcp_name |
| <i>devicedata</i> | “设置”中描述的值  |
| <i>cancel</i>     | 布尔表达式值，详见“设置”  |

## 设置

*devicetype* 的设置值如下：

| 常量                | 值 | 描述                     |
|-------------------|---|------------------------|
| DeviceTypeOEM     | 0 | OEM 定义的设备类型            |
| DeviceTypeDevNode | 1 | Devnode 号 (Windows 95) |
| DeviceTypeVolume  | 2 | 逻辑卷标（磁盘驱动器）            |
| DeviceTypePort    | 3 | 串行或并行端口                |
| DeviceTypeNet     | 4 | 网络资源                   |

*deviceid* 的设置值如下:

| <i>devicetype</i> 设置值 | <i>deviceid</i> 设置值 |
|-----------------------|---------------------|
| DeviceTypeOEM         | dbco_identifier     |
| DeviceTypeDevNode     | dbcd_devnode        |
| DeviceTypeVolume      | dbcv_unitmask       |
| DeviceTypePort        | Null                |
| DeviceTypeNet         | dbcn_resource       |

*devicedata* 的设置值如下:

| <i>devicetype</i> 设置值 | <i>devicedata</i> 设置值 |
|-----------------------|-----------------------|
| DeviceTypeOEM         | dbco_supfunc          |
| DeviceTypeDevNode     | Null                  |
| DeviceTypeVolume      | dbcv_flags            |
| DeviceTypePort        | Null                  |
| DeviceTypeNet         | dbcn_flags            |

*cancel* 的设置值如下:

| 设置值   | 描述       |
|-------|----------|
| True  | 系统禁止删除设备 |
| False | 系统允许删除设备 |

请参阅

DeviceArrival 事件， DeviceOtherEvent 事件， DeviceQueryRemove 事件， DeviceQueryRemoveFailed 事件， DeviceRemoveComplete 事件， SysInfo 控件

## DevModeChange 事件

当修改设备模式设置值时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_DevModeChange(index As Integer,] ByVal devicename As String)
```

DevModeChange 事件语法包含下面部分：

| 部分                | 描述                               |
|-------------------|----------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>      | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>devicename</i> | 字符串表达式，用来识别 Windows 注册表中指定的设备名   |

请参阅

SysInfo 控件

## DisplayChanged 事件

当屏幕分辨率改变时，该事件发生。

应用于

SysInfo 控件

# 语法

Private Sub *object*\_DisplayChanged(*[index* As Integer])

DisplayChanged 事件的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

# 说明

利用此事件调整应用程序界面，以适应屏幕分辨率的变化。

# 请参阅

SysInfo 控件，WorkAreaHeight 属性，WorkAreaLeft 属性，WorkAreaTop 属性，WorkAreaWidth 属性

# 示例

在屏幕分辨率改变后，该示例测试活动窗体的大小，而且，在窗体超出屏

幕可见区域时还要调整其尺寸。要运行此例，请在窗体中放置 **SysInfo** 控件，并将下列代码粘贴到 **SysInfo** 控件的 **DisplayChanged** 事件中，然后运行下例，改变屏幕的分辨率。

```
Private Sub SysInfo1_DisplayChanged()  
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then  
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft  
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth  
    End If  
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then  
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop  
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight  
    End If  
End Sub
```

## Index 属性

返回或设置唯一标识控件数组中某一控件的数。只有当控件是控件数组的一部分时，才可用该属性。



应用于

SysInfo 控件

语法

*object*[(*number*)].Index

Index 属性的语法包含下面部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象      |
| <i>number</i> | 数值表达式，其值为一整数，识别控件数组中的个别控件 |

设置

*number* 的设置值如下：

| 设置值        | 描述   |
|------------|--|
| 没有值        | （缺省值）不是控件数组的部分   |
| 0 到 32,767 | 控件数组的部分。指定一个大于或等于 0 的整数，识别控件数组中的控件控件数组中的所有控件都有相同的 <b>Name</b> 属性。<br>Visual Basic 将自动分配控件数组中的下一个可用整数 |

## 说明

由于控件数组中的元素共享同一个 **Name** 属性设置值，所以在代码中必须用 **Index** 属性指定数组中的特定控件 **Index** 必须是一个整数（或值为整数的数值表达式），它在控件数组名称后面的括号中，例如，**MyButtons(3)**。也可用 **Tag** 属性值区分控件数组中的控件

当数组中的控件识别出发生的事件时，**Visual Basic** 将调用控件数组的事件过程，并将 **Index** 设置值作为附加参数传递过去。当用 **Load** 语句在运行时动态创建控件，或用 **Unload** 语句删除控件时，也会用到该属性。

尽管在缺省方式下，**Visual Basic** 分配下一个可用的整数作为控件数组中新控件的 **Index** 值，还是可不用这个分配的值而跳过整数。此外，还可对数组中的第一个控件，将 **Index** 的值设置为非 0 的整数。如果代码中引用的 **Index** 值不识别控件数组中的控件，则将引起 **Visual Basic** 运行时错误。

**注意：**为从控件数组中删除控件，需改变控件的 **Name** 属性值，并删除其 **Index** 属性设置值。

## 请参阅

**SysInfo** 控件

# Left 属性

返回或设置对象的内部左边界及其容器的左边界之间的距离。

应用于

SysInfo 控件

语法

*object*.Left [= *value*]

Left 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 指定距离的数值表达式           |

说明

对控件来说，其 Left 属性的表示单位总是依赖于容器的坐标系统。当用户或代码移动对象时，则这些属性值将改变。

请参阅

**SysInfo** 控件，**Top** 属性

**Name** 属性

返回代码中用来标识 **SysInfo** 控件的名称。

应用于

**SysInfo** 控件

语法

*object*.Name

**object** 置换元代表对象表达式，其值是“应用于”列表中的对象。如果省略 **object**，则假定与活动窗体模块相关联的那个窗体为 **object**。

## 说明

新对象的缺省名称是对象类型加上一个特定的整数。例如，第一个新窗体对象是 `Form1`，一个新 `SysInfo` 控件是 `SysInfo1`，而在窗体中创建的第三个 `TextBox` 控件是 `Text3`。

对象的 `Name` 属性必须以字母开头，最多可有 40 个字符。其中可包括数字和下划线 (`_`) 字符，但不能有标点符号及空格。窗体不能与其它公用对象同名，如 `Clipboard`、`Screen` 或 `App`。尽管 `Name` 属性值可以是关键字、属性名或其它对象的名字，但这样做会在代码中产生冲突。

将 `Name` 属性设置成相同值就可创建相同类型的控件组成的数组。例如，在将一个组中的所有选项按钮名设置成 `MyOpt` 时，`Visual Basic` 对每个控件的 `Index` 属性分配一个唯一的整数，以区分数组中的其它元素。两个不同类型的控件不能用同一个名字。

## 请参阅

`SysInfo` 控件

## OSBuild 属性

该属性返回值提供运行当前应用程序的操作系统信息。它在运行时不可使用。

应用于

SysInfo 控件

语法

*object*.OSBuild

*object* 置换元代表对象表达式，其值是“应用于”列表中的一个对象。

请参阅

OSPlatform 属性，OSVersion 属性，SysInfo 控件

# OSPlatform 属性

返回一标识操作系统的值，应用程序当前在该操作系统下运行。在设计时不可使用。

应用于

SysInfo 控件

语法

*object*.OSPlatform

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

返回值

OSPlatform 属性返回以下值：

| 值 | 描述     |
|---|--------|
| 0 | Win32s |

续表

| 值 | 描述         |
|---|------------|
| 1 | Windows 95 |
| 2 | Windows NT |

请参阅

OSBuild 属性, OSVersion 属性, SysInfo 控件

示例

该示例显示当前操作系统类型及其版本号。要运行此例，请在窗体中放置 SysInfo 控件及 CommandButton 控件将此例粘贴到窗体的声明部分，运行工程并单击 CommandButton。

```
Private Sub Command1_Click()  
    Dim MsgEnd As String  
    Select Case SysInfo1.OSPlatform  
        Case 0  
            MsgEnd = "Unidentified"  
        Case 1  
            MsgEnd = "Windows 95, ver. " & CStr(SysInfo1.OSVersion)
```



Case 2

```
MsgEnd = "Windows NT, ver. " & CStr(SysInfo1.OSVersion)
```

```
End Select
```

```
MsgBox "System: " & MsgEnd
```

```
End Sub
```

## OSVersion 属性

返回识别操作系统版本的值，应用程序当前在该操作系统下运行。在设计时不可使用。

应用于

SysInfo 控件

语法

*object*.OSVersion

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

请参阅

OSBuild 属性, OSPlatform 属性, SysInfo 控件

示例

该示例显示当前操作系统类型及其版本号。要运行此例,请在窗体中放置 **SysInfo** 控件及 **CommandButton** 控件,然后将此例粘贴到窗体的声明部分,运行工程并单击 **CommandButton**。

```
Private Sub Command1_Click()  
    Dim MsgEnd As String  
    Select Case SysInfo1.OSPlatform  
        Case 0  
            MsgEnd = "Unidentified"  
        Case 1  
            MsgEnd = "Windows 95, ver. " & CStr(SysInfo1.OSVersion)  
        Case 2  
            MsgEnd = "Windows NT, ver. " & CStr(SysInfo1.OSVersion)  
    End Select  
    MsgBox "System: " & MsgEnd  
End Sub
```

# PowerQuerySuspend 事件

当系统电源将被挂起时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_PowerQuerySuspend([index As Integer,] cancel As Boolean)
```

PowerQuerySuspend 事件的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>cancel</i> | 布尔表达式，它指示是否禁止系统进入挂起模式，详见“设置”     |

# 设置

*cancel* 的设置值如下:

| 设置值   | 描述               |
|-------|------------------|
| True  | 禁止系统进入挂起模式       |
| False | (缺省值) 允许系统进入挂起模式 |

请参阅

ACStatus 属性, BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLifeTime 属性, BatteryStatus 属性, PowerResume 事件, PowerStatusChanged 事件, PowerSuspend 事件, SysInfo 控件

## PowerResume 事件

当系统脱离挂起模式, 而应用程序恢复正常操作时, 该事件发生。

应用于

SysInfo 控件

# 语法

Private Sub *object*.PowerResume(*[index As Integer]*)

PowerResume 事件的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

# 请参阅

ACStatus 属性, BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLifeTime 属性, BatteryStatus 属性, PowerQuerySuspend 事件, PowerStatusChanged 事件, PowerSuspend 事件, SysInfo 控件

# PowerStatusChanged 事件

当系统电源状态改变时，该事件发生。

应用于

SysInfo 控件

语法

Private Sub *object*.PowerStatusChanged(*[index As Integer]*)

PowerStatusChanged 事件的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

说明

系统电源的下列变化状态将触发 PowerStatusChanged 事件：

- 电池能源不足
- 电源从 AC 切换到电池或从电池切换到 AC
- 电池充电结束

请参阅

ACStatus 属性, BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLifeTime 属性, BatteryStatus 属性, PowerQuerySuspend 事件, PowerResume 事件, PowerSuspend 事件, SysInfo 控件

示例

该示例在每次电源状态变化时更新 Label 控件, 使应用程序及时显示当前电池状态信息。要运行此例, 请在窗体中放置 SysInfo 控件和 Label 控件将代码粘贴到 SysInfo 控件的 PowerStatusChanged 事件中。然后运行此例。

```
Private Sub SysInfo1_PowerStatusChanged()  
    Select Case SysInfo1.BatteryStatus  
        Case 1  
            Label1.Caption = "Battery OK"  
        Case 2  
            Label1.Caption = "Battery Low"  
        Case 4  
            Label1.Caption = "Battery Critical"  
        Case 8  
            Label1.Caption = "Battery Charging"
```

Case 128, 255

Label1.Caption = "No Battery Status"

End Select

End Sub

## PowerSuspend 事件

当系统即将进入挂起模式之前，该事件直接发生。

应用于

SysInfo 控件

语法

Private Sub *object*\_PowerSuspend(*[index* As Integer])

PowerSuspend 事件的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |



续表

| 部分           | 描述                               |
|--------------|----------------------------------|
| <i>index</i> | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

请参阅

ACStatus 属性, BatteryFullTime 属性, BatteryLifePercent 属性, BatteryLifeTime 属性, BatteryStatus 属性, PowerQuerySuspend 事件, PowerResume 事件, PowerStatusChanged 事件, SysInfo 控件

QueryChangeConfig 事件

在连接或断开系统之前请求改变当前硬件配置文件时，改变硬件配置文件可通过操作系统用户界面进行，也可通过请求。

应用于

SysInfo 控件

# 语法

Private Sub *object*\_QueryChangeConfig(*[index* As Integer,] *cancel* As Boolean)

QueryChangeConfig 事件语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |
| <i>cancel</i> | 布尔表达式，详见“设置”                     |

# 设置

*cancel* 的设置值如下：

| 设置值   | 描述                |
|-------|-------------------|
| True  | 禁止系统改变硬件配置文件      |
| False | （缺省值）允许系统改变硬件配置文件 |

请参阅

ConfigChangeCancelled 事件, ConfigChanged 事件, SysInfo 控件

## ScrollBarSize 属性

以缇（twip）为单位返回系统滚动条的宽度的度量值。

应用于

SysInfo 控件

语法

*object*.ScrollBarSize

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

请参阅

SysInfo 控件

# SettingChanged 事件

当应用程序修改系统范围的参数时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_SettingChanged([index As Integer,] ByVal item As Integer,
ByVal section As Integer)
```

SettingChanged 事件语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

| 部分             | 描述   |
|----------------|--|
| <i>item</i>    | 一个长整数，包含已改变的系统参数信息。如果消息不是从 USER32 中发出的，则系统参数（它是来自 WM_SETTINGCHANGE 的 WPARAM，对应 win32 API 例程 SystemParameterInfo 的参数）总是为零。控制面板中的许多小应用程序都可使消息得以发送出来，但它们自己完成此操作，而不必通过 USER32 来做。在这些条件下，该事件中的值可能是零 |
| <i>section</i> | 字符串表达式，其值是一个部分的名称，该部分包含已改变的系统参数  |

说明

可用 SettingChanged 事件检测许多有用的系统变化。其中可检测出来的最有用的变化是在移动或改变 Windows 95 任务栏的大小时所引起的变化。

请参阅

SysInfo 控件

# SysColorsChanged 事件

当应用程序或“控制面板”改变系统颜色设置时，该事件发生。

应用于

SysInfo 控件

语法

```
Private Sub object_SysColorsChanged([index As Integer])
```

SysColorsChanged 事件的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

请参阅

SettingChanged 事件，SysInfo 控件

# Tag 属性

返回或设置一个表达式，它存储程序需要的额外数据。与其它属性不同，Visual Basic 不使用 Tag 属性的值；可用该属性识别对象。

应用于

SysInfo 控件

语法

*object*.Tag

*object*.Tag [= *expression*]

Tag 属性的语法包含下面部分：

| 部分                | 描述                            |
|-------------------|-------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的对象          |
| <i>expression</i> | 识别对象的字符串表达式。缺省值是一个零长度的字符串("") |

## 说明

可用该属性为对象分配一个标识字符串，而不会影响此对象的任何其它属性设置值也不会产生副作用。

## 请参阅

**SysInfo** 控件

## TimeChanged 事件

当应用程序或“控制面板”改变系统时间时，该事件发生。

## 应用于

**SysInfo** 控件

## 语法

```
Private Sub object_TimeChanged([index As Integer])
```



TimeChanged 事件的语法包含下面部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象             |
| <i>index</i>  | 数值表达式，如果控件在控件数组中，则该数值表达式的值是控件的索引 |

请参阅

SettingChanged 事件，SysColorsChanged 事件，SysInfo 控件

## Top 属性

返回或设置对象的内部上边界与其容器的上边界之间的距离。

应用于

SysInfo 控件，RichTextBox 控件

## 语法

*object*.Top [= *value*]

Top 属性的语法包含下面部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>value</i>  | 指定距离的数值表达式           |

请参阅

SysInfo 控件，Left, Top 属性。

## WorkAreaHeight 属性

返回对 Windows 95 的任务栏进行调整所得的可见桌面的高度。在设计时不可使用。

应用于

SysInfo 控件

语法

*object*.WorkAreaHeight

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

说明

当任务栏出现在屏幕的顶部或底部时，WorkAreaHeight 属性将指出可见桌面减去任务栏后的高度。

请参阅

SysInfo 控件，WorkAreaLeft 属性，WorkAreaTop 属性，WorkAreaWidth 属性

示例

该示例测试活动窗体在屏幕分辨率改变后的尺寸，如果窗体超出屏幕可见

区域，此例还将调整窗体的大小。要运行此例，请将 **SysInfo** 控件放置到窗体中。将下列代码粘贴到 **SysInfo** 控件的 **DisplayChanged** 事件中。运行此例，改变屏幕的分辨率。

```
Private Sub SysInfo1_DisplayChanged()  
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then  
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft  
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth  
    End If  
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then  
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop  
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight  
    End If  
End Sub
```

## WorkAreaLeft 属性

返回对 Windows 95 任务栏进行调整所得可见桌面的左边界坐标。在设计时不可使用。

应用于

**SysInfo** 控件

语法

*object*.WorkAreaLeft

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

请参阅

SysInfo 控件，WorkAreaHight 属性，WorkAreaTop 属性，WorkAreaWidth 属性。

示例

该示例测试活动窗体在屏幕分辨率改变后的尺寸，如果窗体超出屏幕可见区域，则此例还将调整窗体的大小。要运行此例，请将 **SysInfo** 控件放置到窗体中。将下列代码粘贴到 **SysInfo** 控件的 **DisplayChanged** 事件中。运行此例，改变屏幕的分辨率。

```
Private Sub SysInfo1_DisplayChanged()
```

```
If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then
    Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft
    Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth
End If
If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
    Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
    Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
End If
End Sub
```

## WorkAreaTop 属性

返回对 Windows 95 任务栏进行调整所得可见桌面的上边界坐标。在设计时不可使用。

应用于

SysInfo 控件

## 语法

*object*.WorkAreaTop

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

## 请参阅

SysInfo 控件，WorkAreaHeight 属性，WorkAreaLeft 属性， WorkAreaWidth 属性。

## 示例

该示例测试活动窗体在屏幕分辨率改变后的尺寸，如果窗体超出屏幕可见区域，则此例还将调整窗体的大小。要运行此例，请将 **SysInfo** 控件放置到窗体中。将下列代码粘贴到 **SysInfo** 控件的 **DisplayChanged** 事件中。运行此例，改变屏幕的分辨率。

```
Private Sub SysInfo1_DisplayChanged()  
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then  
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft  
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth  
    End If
```

```
If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then
    Screen.ActiveForm.Top = SysInfo1.WorkAreaTop
    Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight
End If
End Sub
```

## WorkAreaWidth 属性

### 语法

*object*.WorkAreaWidth

*object* 置换元代表对象表达式，其值是“应用于”列表中的对象。

### 说明

当任务条出现在屏幕的左或右边沿时，**WorkAreaWidth** 属性返回可见桌面的宽度减去任务条宽度之后的宽度。



请参阅

**SysInfo** 控件，**WorkAreaHeight** 属性，**WorkAreaLeft** 属性， **WorkAreaTop** 属性

## 示例

该示例测试活动窗体在屏幕分辨率改变后的尺寸，如果窗体超出屏幕可见区域，则此例还将调整窗体的大小。要运行此例，请将 **SysInfo** 控件放置到窗体中。将下列代码粘贴到 **SysInfo** 控件的 **DisplayChanged** 事件中。运行此例，改变屏幕的分辨率。

```
Private Sub SysInfo1_DisplayChanged()  
    If Screen.ActiveForm.Width > SysInfo1.WorkAreaWidth Then  
        Screen.ActiveForm.Left = SysInfo1.WorkAreaLeft  
        Screen.ActiveForm.Width = SysInfo1.WorkAreaWidth  
    End If  
    If Screen.ActiveForm.Height > SysInfo1.WorkAreaHeight Then  
        Screen.ActiveForm.Top = SysInfo1.WorkAreaTop  
        Screen.ActiveForm.Height = SysInfo1.WorkAreaHeight  
    End If  
End Sub
```

返回对 Windows 95 任务栏进行调整所得到的可见桌面宽度。在设计时不可用。

# TabStrip 控件

一个 **TabStrip** 控件就象笔记本中的分隔标签或一组文件夹上的标签。使用 **TabStrip** 控件，可以在应用程序中在窗口或对话框中的同一区域定义多个数据页面。

## 语法

### **TabStrip**

## 说明

该控件包含了一个或多个 **Tabs** 集合中的 **Tab** 对象。在设计时和运行时，都可以设置其属性来控制 **Tab** 对象的外观。在设计时，可以通过 **TabStrip** 控件的 **Properties Page** 对话框增加或删除 **Tab** 对象，或在运行时使用方法增加或删除 **Tab** 对象。

**Style** 属性决定 **TabStrip** 控件看起来象一个按钮 (**Button**) 还是象笔记本的标签 (**Tab**)。在设计时，当在窗体上放置一个 **TabStrip** 控件时，其有一个笔记

本标签。如果 **Style** 属性设置为 **tabTabs**，则在 **TabStrip** 控件的内部区域将有一个边框。当 **Style** 属性设置为 **tabButtons** 时，不在内部区域显示边框，然而，该区域仍存在。

要设置 **TabStrip** 控件的总体大小，使用拖动或/和设置 **Top**, **Left**, **Height** 和 **Width** 属性。在运行时，根据 **TabStrip** 控件的总体大小，**Visual Basic** 将自动决定内部区域的大小和位置，并返回 **Client** 坐标属性——**ClientLeft**, **ClientTop**, **ClientHeight** 和 **ClientWidth**。**MultiRow** 属性决定 **TabStrip** 控件是否可以有多行标签，**TabWidthStyle** 属性决定每行的外观，如果 **TabWidthStyle** 属性设置为 **True**，可以使用 **TabFixedHeight** 和 **TabFixedWidth** 属性将 **TabStrip** 控件中的所有选项卡的高和宽都设置为相同。

**TabStrip** 控件不是一个容器。要包含实际的页面和大小，必须使用 **Frame** 控件或其他容器，使之与所有选项卡所共享的内部区域匹配。如果为容器使用一个控件矩阵，可以将矩阵中的每个成员与一个 **Tab** 对象关联，如下面的例子：

Option Explicit

Private mintCurFrame As Integer ' Current Frame visible

Private Sub Tabstrip1\_Click()

If Tabstrip1.SelectedItem.Index = mintCurFrame \_

Then Exit Sub ' No need to change frame.

' Otherwise, hide old frame, show new.

Frame1(Tabstrip1.SelectedItem.Index).Visible = True

```
Frame1(mintCurFrame).Visible = False
' Set mintCurFrame to new value.
mintCurFrame = Tabstrip1.SelectedItem.Index
End Sub
```

**注意:** 当将容器中的控件组成一组时, 必须使用上面显示的 show/hide 策略而不是 Zorder 方法将窗体换到前台。否则, 实施了加速键的控件仍可以响应键盘命令, 即使该控件不是容器中最上面的控件。注意, 对多个 OptionButton 控件, 必须放置在各自的容器中, 否则窗体中的所有 OptionButtons 控件就是一组。

**提示:** 在使用容器时, 使用 BorderStyle 属性设置为 None 的 Frame 控件代替 PictureBox 控件因为 Frame 控件比 PictureBox 控件的开销小。

TabStrip 控件的 Tabs 属性是所有 Tab 对象的集合。每个 Tab 对象都有与当前状态和外观关联的属性。例如, 可以将一个 ImageList 控件与 TabStrip 控件关联, 然后在每个 Tab 上使用这些图片。也可以将一个 ToolTip 与每个 Tab 对象关联。

**发布须知:** TabStrip 控件是在 MSCOMCTL32.OCX 文件中自定义控件的一部分。要在应用程序中使用 TabStrip 控件, 必须给工程文件增加 MSCOMCTL.OCX。当发布应用程序时, 在用户的 Microsoft Windows SYSTEM 目录下安装 RICHTX32.OCX 文件。有关如何给工程增加自定义控件, 请参阅《Microsoft Visual Basic 6.0 程序员指南》。

## 属性

Enabled 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), OLEDragMode 属性 (ActiveX 控件), ShowTips 属性 (ActiveX 控件), Height,Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left,Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), MultiSelect 属性 (ListView, TabStrip 控件), Tab 对象, Tabs 集合, ClientHeight, ClientWidth, ClientLeft, ClientTop 属性, MultiRow 属性, Style 属性 (TabStrip 控件), Tabs 属性 (TabStrip 控件), TabFixedHeight, TabFixedWidth 属性, TabWidthStyle 属性, SelectedItem 属性 (ActiveX 控件)

## 方法

OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件)

请参阅

ImageList 控件

# Add 方法（Tabs 集合）

给 TabStrip 控件的 Tabs 集合增加一个 Tab 对象。

应用于

TabStrip 控件

语法

*object.Add (index ,key, caption, image)*

Add 方法的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | Tabs 集合的对象表达式  |
| <i>index</i>  | 可选参数。一个整数，指定插入 Tab 对象的位置。如果没有指定该参数，则 Tab 插入到 Tabs 集合的末尾    |
| <i>key</i>    | 可选参数。标识 Tab 的唯一字符串。使用键值检取一个 Tab 对象。这与新加入的 Tab 对象的 Key 属性相同 |

续表

| 部分             | 描述  |
|----------------|---|
| <i>caption</i> | 可选参数。在 Tab 上显示的字符串。这与新加入 Tab 对象的 Caption 属性值相同                  |
| <i>image</i>   | 可选参数。关联的 ImageList 控件中图像的索引。在选项卡上显示该图片。这与新加入 Tab 对象的 Image 属性相同 |

## 说明

要在设计时给 TabStrip 控件增加 Tab 对象，点击 TabStrip 控件的 Properties Page 对话框中 Tab 选项卡上的 Insert Tab 按钮，输入相应的参数。

在运行时给 TabStrip 控件增加 Tab 对象，可以使用 Add 方法，其返回新加入对象的引用。例如，下面的代码增加了一个标题为 “Howdy!”，键值为 “MyTab” 的一个 Tab 对象：

```
Set X =TabStrip1.Tabs.Add(2, “MyTab”, “Howdy!”)
```

## 请参阅

Clear 方法（ActiveX 控件），Property 页面对话框（ActiveX 控件），Key 属性（ActiveX 控件），Index 属性（ActiveX 控件），ImageList 控件



## 示例

下面的例子给 TabStrip 控件增加了 3 个 Tab 对象，其标题和图片都由 ImageList 获得。要使用本例，在一个窗体上放置一个 ImageList 控件和一个 TabStrip 控件 ImageList 控件提供了 Tab 对象的图像，因此给 ImageList 控件增加 3 个图像。将下面的代码拷贝到窗体的 Load 事件中。运行本例：

```
Private Sub Form_Load()  
    Dim X As Integer  
    Set TabStrip1.ImageList = ImageList1  
    TabStrip1.Tabs(1).Caption = "Time"  
    TabStrip1.Tabs.Add 2, "Date"  
    TabStrip1.Tabs.Add 3, "Mail"  
    For X = 1 To TabStrip1.Tabs.Count  
        TabStrip1.Tabs(X).Image = X  
    Next X  
End Sub
```

# BeforeClick 事件

当点击 TabStrip 控件中的 Tab 对象或 Tab 对象的 Selected 设置发生改变时产生该事件。

应用于

TabStrip 控件

语法

Private Sub *object*\_BeforeClick ( *cancel* As Integer)

BeforeClick 事件的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 一个 TabStrip 控件的对象表达式           |
| <i>cancel</i> | 整数枚举值 0(False)和-1(True)，初始值时 0 |

说明

在所选择的新 Tab 对象产生实际的 Click 事件之前使用 BeforeClick 事件验

证旧的 Tab 对象信息。将 Cancel 参数设置为 True 将允许你停止对新选择内容的修改操作。

注意：将 *cancel* 参数设置为 True 将阻止窗体切换到另一个 Tab 对象，但不阻止产生 Click 事件。

注意：如果在 BeforeClick 事件过程中使用 MsgBox 或 InputBox 函数，则 TabStrip 控件接收不到 Click 事件，不管 cancel 参数的设置如何。

## 示例

下面的例子使用了 BeforeClick 事件，演示了如何阻止用户切换到另一个 Tab 对象。当想在显示新的 Tab 对象之前验证当前 Tab 对象上的信息时是很有用的。

要使用本例，在一个窗体上(BorderStyle 属性设置为 None)放置一个 TabStrip 控件和一个 2 元素 Frame 控件矩阵。在第一个 Frame 控件中增加一个 CheckBox 控件，在第二个 Frame 控件中增加一个 TextBox 控件将下面的代码拷贝到 Form 对象的 Load 事件中。在你选择了标签为 Check 的选项卡上的 CheckBox 之后再选择标签为 Text 的选项卡。

```
Private Sub Form_Load()  
Dim i As Integer  
Dim Tabx As Object
```

```

' Sets the caption of the first tab to "Check."
TabStrip1.Tabs(1).Caption = "Check"
' Adds a second tab with "Text" as its caption.
Set Tabx = TabStrip1.Tabs.Add(2, , "Text")
' Labels the checkbox.
Check1.Caption = "Cancel tab switch"
    ' Aligns the Frames with the internal area
    ' of the Tabstrip Control.
    For i = 0 To 1
        Frame1(i).Left = TabStrip1.ClientLeft
        Frame1(i).Top = TabStrip1.ClientTop
        Frame1(i).Height = TabStrip1.ClientHeight
        Frame1(i).Width = TabStrip1.ClientWidth
    Next
    ' Puts the first tab's Frame container on top.
    Frame1(0).ZOrder 0
End Sub

' The BeforeClick event verifies the check box value
' to determine whether to proceed with the Click event.
Private Sub TabStrip1_BeforeClick(Cancel As Integer)
    If TabStrip1.Tabs(1).Selected Then

```

```
        If Check1.Value = 1 Then Cancel = True
    End If
End Sub

Private Sub TabStrip1_Click()
    Frame1(TabStrip1.SelectedItem.Index-1).ZOrder 0
End Sub
```

## Caption 属性（Tab 对象）

返回或设置 TabStrip 控件中在 Tab 对象的按钮或标签上显示的标题。

应用于

TabStrip 控件

语法

*object*.Caption [=string]

Caption 属性的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 一个 Tab 对象的对象表达式 |
| <i>string</i> | 字符串表达式，所显示的文本标题 |

## 说明

可以在设计时或运行时设置 TabStrip 控件中 Tab 对象的 Caption 属性。

- 设计时——在 TabStrip 控件的 Properties Page 对话框中的 Tab 选项卡上，在 Caption 文本框中输入标题内容。
- 运行时——使用下面的代码设置标题：

```
TabStrip1.Tabs(1).Caption = "First Tab"
```

或

```
TabStrip1.Tabs.Add 2, , "Second Tab"
```

## 示例

下面的例子设置了 TabStrip 控件中 3 个新增 Tab 对象的 Caption 属性。标题字符串分别是 “Time”，“Data”，“Mail”。每个 Tab 对象也显示 ImageList 控件中的一个图像。要使用本例，在一个窗体上放置一个 ImageList 控件和一个 TabStrip 控件在 ImageList 中加载 3 个点位图。该 ImageList 控件为 Tab 对象提供图像。将下面的代码拷贝到 Form 对象的 Load 事件中，运行本例：

```
Private Sub Form_Load()  
    Dim X As Integer  
    ' Associate an ImageList with the TabStrip control.  
    Set TabStrip1.ImageList = ImageList1  
    ' Set the captions.  
    TabStrip1.Tabs(1).Caption = "Time"  
    TabStrip1.Tabs.Add 2, "Date"  
    TabStrip1.Tabs.Add 3, "Mail"  
    For X = 1 To TabStrip1.Tabs.Count  
        ' Associate an image with a tab.  
        TabStrip1.Tabs(X).Image = X  
    Next X  
End Sub
```

## ClientHeight, ClientWidth, ClientLeft, ClientTop 属性

返回 **TabStrip** 控件中内部区域（显示区域）的坐标。在运行时是只读的；在设计时不可用。

应用于

TabStrip 控件

语法

*object*.ClientHeight

*object*.ClientWidth

*object*.ClientLeft

*object*.ClientTop

*object* 是 TabStrip 控件的对象表达式。

说明

在运行时，ClientHeight, ClientWidth, ClientLeft, ClientTop 属性自动保存了 TabStrip 控件中内部区域的坐标，其被所有的 Tab 对象共享。这样，当选择一个 Tab 对象时，就出现与 Tab 关联的控件，将 Tab 对象控件放置到一个容器如 Frame 控件中，其大小和位置与客户区域坐标匹配。要将一个容器与 Tab 对象关联，创建一个控件矩阵，如 Frame 控件矩阵。所有的客户区域坐标都使用父窗体的标尺度量模式。要在内部区域放置一个 Frame 控件，使用下面的代码：

```
Frame1.Left = TabStrip1.ClientLeft
```



```
Frame1.Top = TabStrip1.ClientTop  
Frame1.Width = TabStrip1.ClientWidth  
Frame1.Height = TabStrip1.ClientHeight
```

当选择一个选项卡时，要将新的选项卡和关联的容器放置在最上面：

- 将 **TabStrip** 控件中内部区域中容器大小和位置设置为客户区域属性
- 使用 **ZOrder** 方法将选择的选项卡容器控件放置在前台或 **z-order** 的后面

## 示例

下面的例子演示了使用 **ClientHeight**, **ClientWidth**, **ClientLeft**, **ClientTop** 属性和 **Frame** 控件矩阵显示选项卡——当切换选项卡时 **TabStrip** 控件内部区域中的特定对象。本例使用了 **ZOrder** 方法显示相应的 **Frame** 控件以及其包含的控件

要使用本例，在一个窗体上放置一个 **TabStrip** 控件和 3 元素的 **Frame** 控件矩阵。在一个 **Frame** 控件中放置一个 **CheckBox** 控件，在另一个中放置一个 **CommandButton** 控件，在第三个 **Frame** 控件中放置一个 **TextBox** 控件将下面的代码拷贝到 **Form** 对象的 **Load** 事件中，运行本例。点击各选项卡选择内容。

```
Private Sub Form_Load()  
Dim Tabx As Object  
Dim i As Integer  
' Sets the caption of the first tab to "Check."  
TabStrip1.Tabs(1).Caption = "Check"
```

' Adds a second tab with "Command" as its caption.

Set Tabx = TabStrip1.Tabs.Add(2, , "Command")

' Adds a third tab with "Text" as its caption.

Set Tabx = TabStrip1.Tabs.Add(3, , "Text")

' Aligns the frame containers with the internal

' area of the TabStrip control.

For i = 0 To 2

With TabStrip1

Frame1(i).Move .ClientLeft, .ClientTop, \_

ClientWidth, .ClientHeight

End With

Next

' Puts the first tab's picture box container on top

' at startup.

Frame1(0).ZOrder 0

End Sub

Private Sub TabStrip1\_Click()

Frame1(TabStrip1.SelectedItem.Index - 1).ZOrder 0

End Sub

# HighLighted 属性

返回或设置一个值，决定对象是否突出显示。

## 语 法

*object*.HighLighted [= *boolean*]

HighLighted 属性的语法有如下部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，指定对象是否突出显示，如设置中所示的 |

## 设置

*boolean* 设置值的为：

| 常量    | 描述        |
|-------|-----------|
| False | (缺省)不突出显示 |
| True  | 该项被突出显示   |

# MultiRow 属性

返回或设置一个值，指明 TabStrip 控件中是否可以显示多行选项卡。

应用于

TabStrip 控件

语法

*object*.MulitRow [=*boolean*]

MultiRow 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个 TabStrip 控件的对象表达式                        |
| <i>boolean</i> | 一个布尔表达式，指定 TabStrip 控件中是否可以显示多行选项卡，如“设置”中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述         |
|-------|------------|
| True  | 允许显示多行选项卡  |
| False | 不允许显示多行选项卡 |

## 说明

行数将由根据选项卡的宽度和数量自动设置。如果控件的大小改变了，则选项卡的行数也会发生变化，以确保选项卡可以在多行显示。如果 **MultiRow** 属性设置为 **False**，并且最后一个选项卡超出了 **TabStrip** 控件的宽度，则在 **TabStrip** 控件的右边增加一个水平滚动杆。

在设计时，在 **TabStrip** 控件的 **Properties Page** 对话框的 **General** 选项卡上设置 **MultiRow** 属性。在运行时，使用下面的代码设置 **MultiRow** 属性：

```
' Allow more than one row of tabs in the TabStrip control.
```

```
TabStrip1.MultiRow =TRUE
```

## 请参阅

**TabStrip** 控件常量, **ClientHeight**, **ClientWidth**, **ClientLeft**, **ClientTop** 属性, **TabFixedHeight**, **TabFixedWidth** 属性, **TabWidthStyle** 属性

# Placement 属性

返回或设置一个值，指定选项卡的位置—上、下、左、右。

## 语法

*object.Placement* [= *integer*]

Placement 属性的语法有如下几部分:

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>integer</i> | 一个数值表达式，指定选项卡的位置，如设置所描述的 |

## 设置

*integer* 设置如下:

| 常量                 | 数值 | 描述               |
|--------------------|----|------------------|
| tabPlacementTop    | 0  | (缺省) 选项卡出现在控件的上部 |
| tabPlacementBottom | 1  | 选项卡出现在控件的下部      |
| tabPlacementLeft   | 2  | 选项卡出现在控件的左边      |
| tabPlacementRight  | 3  | 选项卡出现在控件的右边      |

## 说明

Placement 属性对 TabStyle 属性的行为也会产生影响。例如，如果 Placement 设置为 tabPlacementLeft，并且 TabStyle 设置为 tabTabOpposite，则当用户单击一个选项卡时，其余的选项卡将被重新排列在控件的右边。

## Separators 属性

返回或设置一个值，确定在 TabStrip 控件上，在具有 tabButton 或 tabFlatButton 样式的按钮之间是否绘制分隔符。

## 语法

*object*.Separators [= *boolean*]

Separators 属性语法有如下几部分:

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>boolean</i> | 一个布尔表达式，指定是否绘制分隔符，如设置所描述的 |

## 设置

*boolean* 设置如下:

| 常量    | 描述         |
|-------|------------|
| False | (缺省)不绘制分隔符 |
| True  | 绘制分隔符      |

## 说明

要显示分隔符，TabStrip 控件的 **Style** 属性必须设置为 **tabButton** 或 **tabFlatButton**。

## Style 属性（TabStrip 控件）

返回或设置 TabStrip 控件的外观——选项卡或按钮。

## 应用于

TabStrip 控件



# 语法

*object*.Style [=value]

Style 属性的语法有如下几个部分：

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 一个 TabStrip 控件的对象表达式         |
| <i>value</i>  | 一个常量或整数，确定选项卡对话框的外观，如“设置”中所示 |

# 设置

*value* 值的设置如下：

| 常量             | 值 | 描述                                    |
|----------------|---|---------------------------------------|
| tabTabs        | 0 | （缺省）Tabs。选项卡显示为笔记本选项卡格式，内部区域有三维的边框    |
| tabButtons     | 1 | Button。选项卡显示为普通的按钮，内部区域没有边框           |
| tabFlatButtons | 2 | 扁平按钮。所选择的选项卡看起来象按到了背景中。未选择的选项卡看起来是扁平的 |

## 说明

在设计时，从 TabStrip 控件中的 Properties Page 对话框中 General 选项卡上的 Style 列表中选择你想要的 Style 属性。

在运行时，使用下面的代码设置 Style 属性：

```
' Style property set to the Tabs style.
```

```
TabStrip1.Style = tabTabs
```

```
' Style property set to the Buttons style:
```

```
TabStrip1.Style = tabButtons
```

## 请参阅

Property 页面对话框（ActiveX 控件）

## Tab 对象

一个 Tab 对象代表了 TabStrip 控件中 Tabs 集合中的一个成员。

## 说明

对于每个 Tab 对象，可以使用各种属性设置其外观，可以使用 **Selected** 属性指定其状态。

在设计时，在 TabStrip 控件的 **Properties Page** 对话框中的 **Tabs** 选项卡上，使用 **Insert Tab** 和 **Remove Tab** 按钮插入和删除选项卡，使用文本框指定 Tab 对象的属性：**Caption**, **Image**, **ToolTipText**, **Tag**, **Index** 和/或 **Key**。也可以在运行时指定这些属性。

使用 **Caption** 和 **Image** 属性标签一个选项卡或在选项卡上显示一个图像。

- 要使用 **Caption** 属性，在 TabStrip 控件的 **Properties Page** 对话框的 **Tabs** 选项卡上的 **Caption** 文本框中输入在运行时要显示的文本内容。
- 要使用 **Image** 属性，在窗体上放置一个 **ImageList** 控件，使用 **ListImage** 对象填充 **ListImages** 集合，每个对象都有一个索引和可选的键值。在 TabStrip 控件的 **Properties Page** 对话框的 **General** 选项卡上，选择与 TabStrip 控件关联的 **ImageList** 对象。在 **Tabs** 选项卡的 **Image** 文本框中，输入在 Tab 对象上显示的 **ImageList** 对象的索引值或键值。

在运行时，当用户的鼠标在选项卡上停留约 1 秒时，使用 **ToolTipText** 属性在该对象下面的小矩形框中显示字符串文本。要在设计时设置 **ToolTipText** 属性，选择 **General** 选项卡上的 **ShowTips** 复选框，然后在 **Tabs** 选项卡上的

ToolTipText 文本框中输入 ToolTip 字符串。

要返回用户选择的 Tab 对象的引用，使用 SelectedItem 属性；要确定选择了哪个特定的 Tab 对象，使用 Selected 属性。这些属性与 BeforeClick 一起使用，对于在显示新的 Tab 之前验证当前活动 Tab 内容是很有用的。

每个 Tab 对象都有只读属性，可以用来引用 Tabs 集合中的单个 Tab 对象：Left, Top, Height 和 Width。

### 属性：

Key 属性（ActiveX 控件），Height, Width 属性（ActiveX 控件），Index 属性（ActiveX 控件），Left, Top 属性（ActiveX 控件），Tag 属性（ActiveX 控件），Caption 属性（Tab 对象），Selected 属性（ActiveX 控件）

### 方法

Clear 方法（ActiveX 控件），Add 方法（Tabs 集合）。

### 请参阅

Caption 属性（ActiveX 控件），Property 页面对话框（ActiveX 控件），ImageList 控件，ListItem 对象，ListImages 集合。

# TabFixedHeight, TabFixedWidth 属性

返回或设置一个 TabStrip 控件中所有 Tab 对象的固定高度和宽度，但只有在 TabWdithStyle 设置为 tabFixed 时才这样做。

应用于

TabStrip 控件，Tab 对象。

语法

```
object.TabFixedHeight [=integer ]  
object.TabFixedWidth [=integer]
```

TabFixedHeight, TabFixedWidth 属性的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个 TabStrip 控件的对象表达式   |
| <i>integer</i> | TabStrip 控件高和宽的像素点数或 twip 数值。 <i>integer</i> 参数的尺度模式取决于容器的 ScaleMode 属性的设置 |

## 说明

**TabFixedHeight** 属性适用于 **TabStrip** 控件中所有的 **Tab** 对象。其缺省为 **Font** 属性中指定的总体高度或 **Image** 属性指定的 **ListImage** 对象的高度, 后者高一点, 其加入了额外的像素做边框。如果 **TabWidthStyle** 属性设置为 **tabFixed**, 并且设置了 **TabFixedWidth** 属性, 则不管往 **TabStrip** 控件中增加还是删除 **Tab** 对象, 每个 **Tab** 对象的宽度都保持不变。

## 请参阅

**ListImage** 对象, **ListImages** 集合

## TabMinWidth 属性

返回或设置允许的选项卡最小宽度。

## 语法

*object*.**TabMinWidth** [= *number*]

TabMinWidth 属性语法有如下几部分:

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式, 其值是“应用于”列表中的一个对象                            |
| <i>number</i> | Tab 对象的最小宽度。 <i>number</i> 所使用的刻度由容器的 ScaleMode 属性确定 |

## 说明

如果 TabWidthStyle 属性设置为 tabFixed, 则 TabMinWidth 属性无效。

## Tabs 集合

一个 Tabs 集合包含了 Tab 对象的集合。

## 语法

*tabstrip*.Tabs(*index*)

*tabstrip*.Tabs.Item (*index*)

Tabs 集合的语法有如下几个部分:

| 部分              | 描述   |
|-----------------|--|
| <i>tabstrip</i> | 一个 TabStrip 控件的对象表达式   |
| <i>index</i>    | 一个整数或字符串，唯一标识对象集合中的一个成员。整数是 Tab 对象的 Index 属性值；字符串是 Tab 对象的 Key 属性值 |

在设计时，在 TabStrip 控件的 Properties Page 对话框的 Tabs 选项卡上，使用 Insert Tab 和 Remove Tab 命令增加或删除 Tabs 集合中的 Tab 对象。

Tabs 集合使用 Count 属性返回集合中对象的数量。要操纵 Tabs 集合中的 Tab 对象，使用下面的 4 种方法之一：

- Add——给 TabStrip 控件增加 Tab 对象。
- Item——使用 Key 或 Index 检取 Tab 对象。
- Clear——删除集合中所有的 Tab 对象。
- Remove——从集合中删除 Key 或 Index 指定的 Tab 对象。

## 属性

Count 属性（ActiveX 控件），Item 属性（ActiveX 控件）

## 方法

Clear 方法（ActiveX 控件），Remove 方法（ActiveX 控件），Add 方法（Tabs



集合)

请参阅

Count 属性 (ActiveX 控件), Key 属性 (ActiveX 控件), Index 属性 (ActiveX 控件)

**Tabs** 属性 (TabStrip 控件)

返回 TabStrip 控件中 Tab 对象集合的引用。

应用于

TabStrip 控件

语法

*object*.Tabs (*index*)

Tabs 属性的语法有如下几个部分:

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 TabStrip 控件的对象表达式                                  |
| <i>index</i>  | 一个值，标识 Tabs 集合中的 Tab 对象。这可以是 Tab 对象的 Index 属性或 Key 属性 |

## 说明

使用标准的集合方法如 **Item** 方法可以访问 Tabs 集合。

## TabStyle 属性

返回或设置一个值，确定位于被选中选项卡前面的选项卡行如何放置。

## 应用于

TabStyle 属性。

## 语法

*object*.TabStyle [= *integer*]

TabStyle 属性的语法有如下几部分:

| 部分             | 描述                             |
|----------------|--------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象       |
| <i>integer</i> | 一个数值表达式，确定如何重新放置其它的选项卡，如设置所描述的 |

## 设置

*integer* 设置如下:

| 常量             | 数值 | 描述                           |
|----------------|----|------------------------------|
| tabTabStandard | 0  | (缺省)其它的选项卡保留在控件的同一侧          |
| tabTabOpposite | 1  | 位于被选中选项卡前面的选项卡行将重新放置在控件的相反一侧 |

## TabWidthStyle 属性

返回或设置一个值，确定 TabStrip 控件中 Tab 对象的宽度或调整方法。

应用于

TabStrip 控件

语法

*object*.TabWidthStyle [=value]

TabWidthStyle 属性的语法有如下几个部分：

| 部分            | 描述                             |
|---------------|--------------------------------|
| <i>object</i> | 在 TabStrip 控件中的对象表达式           |
| <i>value</i>  | 一个整数或常量，确定 Tab 宽度如何调整，如“设置”中所示 |

设置

value 值的设置如下：

| 常量           | 值 | 描述   |
|--------------|---|--|
| tabJustified | 0 | 每个选项卡都足够宽，可容纳下其内容，如果有必要，每个选项卡的宽度都增加以便覆盖整个控件宽度。如果只有单行选项卡，则该风格不起作用 |

续表

| 常量              | 值 | 描述   |
|-----------------|---|--|
| tabNonJustified | 1 | 每个选项卡刚能容纳其内容。行不作调整，因此选项卡会参差不齐                          |
| tabFixed        | 2 | 所有 Tab 对象的高和宽都相同，由 TabFixedHeight 和 TabFixedWidth 属性设置 |

说明

在设计时，在 TabStrip 控件的 Properties Page 对话框的 General 选项卡上设置 TabWidthStyle 属性。TabWidthStyle 属性影响每个 Tab 对象在运行时的外观。在运行时，可以使用下面的代码设置 TabWidthStyle 属性：

```
' Justifies all the tabs in a row to fit the width of the control.
TabStrip1.MultiRow = True
TabStrip1.TabWidthStyle = tabJustified

' Creates ragged rows of tabs.
TabStrip1.MultiRow = True
TabStrip1.TabWidthStyle = tabNonJustified

' Sets the same width for all tabs.
TabStrip1.TabFixedWidth = 500
```

```
TabStrip1.TabWidthStyle = tabFixed
```

## Toolbar 控件

一个 **Toolbar** 控件包含了一个 **Button** 对象集合，用于创建与应用程序关联的工具条。

### 语法

#### **Toolbar**

### 说明

通常，一个工具条中的按钮都与应用程序菜单中的命令相对应，为用户提供了常用命令和功能的图形化界面。

**Toolbar** 控件允许你给一个 **Buttons** 集合增加 **Button** 对象从而创建一个工具条。每个 **Button** 对象都有可选的文本或图像，图像由关联的 **ImageList** 控件提供。可以使用 **Image** 属性在按钮上显示图像或使用 **Caption** 属性在按钮上显示文本，或者显示二者。在设计时，你可以使用 **Toolbar** 控件的 **Properties Page** 对话框给该控件增加 **Button** 对象。在运行时，可以使用 **Add** 和 **Remove** 方法增加或删除 **Buttons** 集合中的按钮。

在应用程序中，使用 **ButtonClick** 事件响应所选择的按钮。也可以使用 **Style** 属性确定每个 **Button** 对象的外观和行为。例如，如果给 4 个按钮指定了 **ButtonGroup** 风格，则一次只能按下其中的一个按钮并且总有一个按钮是按下的。

给一个 **Button** 对象指定 **PlaceHolder** 风格，你可以在工具条上为其他控件预留空间，然后在预留的位置上放置其他空间。使用 **AllowCustomize** 属性使能或禁止对话框。也可以使用 **Customize** 方法激活 **Customize Toolbar** 对话框。如果你希望保存或恢复工具条的状态或允许用户这样做，有两种方法：**SaveToolbar** 方法和 **RestoreToolbar** 方法。在工具条改变时触发 **Change** 事件，通常用于激活 **SaveToolbar** 方法。

**注意** **Customize** 对话框包含了一个 **Help** 按钮。当用户点击 **Help** 按钮时，使用 **HelpFile** 和 **HelpContextID** 属性确定显示了哪个帮助文件。

给每个 **Button** 对象增加 **ToolTipText** 字符串可以增强工具条的使用性。要显示 **ToolTip**，必须将 **ShowTipsProperty** 属性设置为 **True**。当用户激活 **Customize Toolbar** 对话框时，点击一个按钮将导致在对话框中显示文本的描述信息；该描述信息可以通过设置 **Description** 属性来控制。

**发布须知：****Toolbar** 控件是在 **COMCTL32.OCX** 文件中定义的 **ActiveX** 控件的一部分。要在应用程序中使用 **Toolbar** 控件，你必须给工程文件增加 **RICHTX32.OCX**。当发布你的应用程序时，在用户的 **Microsoft Windows SYSTEM** 目录下安装 **RICHTX32.OCX** 文件。其他信息请参阅《Microsoft



Visual Basic 6.0 程序员指南》。

## 属性

Style 属性 (Toolbar 控件), ButtonMenus 属性, DisabledImageList 属性, HotImageList 属性, TextAlignment 属性, AllowCustomize 属性, ButtonHeight, ButtonWidth 属性, Buttons 属性, Wrappable 属性, Controls 属性 (Toolbar 控件), DataBinding 对象, DataBindings 属性, HelpFile 属性 (App, CommonDialog, MenuLife), TabIndex 属性, Align 属性, DragIcon 属性, DragMode 属性, hWnd 属性, MouseIcon 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件), ShowTips 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enabled 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), ImageList 属性 (ActiveX 控件)。

## 方法

Customize 方法, RestoreToolbar 方法, SaveToolbar 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件), Refresh

方法（ActiveX 控件）。

## 事件

ButtonMenuClick 事件，ButtonClick 事件，Change 事件（Toolbar, Slider 控件），DragDrop 事件，DragOver 事件，MouseDown, MouseUp 事件，MosueMove 事件，OLECompleteDrag 事件（ActiveX 控件），OLEDragDrop 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件），Change 事件（ActiveX 控件），Click 事件（ActiveX 控件），DbClick 事件（ActiveX 控件）。

## 请参阅

ButtonMenu 对象，ButtonMenus 集合，ImageList 控件，Button 对象，Buttons 集合，Description 属性（Button 对象），Style 属性（Button 对象），Toolbar 控件常量，Customize 对话框，使用 ImageList 控件，使用 ToolBar 控件

## 示例

下面的例子使用 Add 方法给一个 Toolbar 控件增加 Button 对象，并从 ImageList 控件中分配图像。每个按钮的行为由 Style 属性。创建的按钮用于打

开和保存文件，包括用于改变窗体背景色的 **ComboBox** 控件要使用本例，在一个窗体上放置一个 **Toolbar** 控件，一个 **ImageList** 控件和一个 **ComboBox** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。确定你直接将 **ComboBox** 控件插入到了 **Toolbar** 控件中。运行本例，点击各按钮，并从组合框中选择。

```
Private Sub Form_Load()  
    ' Create object variable for the ImageList.  
    Dim imgX As ListImage  
  
    ' Load pictures into the ImageList control.  
    Set imgX = ImageList1.ListImages. _  
    Add(, "open", LoadPicture("Graphics\bitmaps\tlbr_w95\open.bmp"))  
    Set imgX = ImageList1.ListImages. _  
    Add(, "save", LoadPicture("Graphics\bitmaps\tlbr_w95\save.bmp"))  
    Toolbar1.ImageList = ImageList1  
  
    ' Create object variable for the Toolbar.  
    Dim btnX As Button  
    ' Add button objects to Buttons collection using  
    ' the  
    ' Add method. After creating each button, set both  
    ' Description and ToolTipText properties.  
    Toolbar1.Buttons.Add , , tbrSeparator
```

```
Set btnX = Toolbar1.Buttons.Add(, "open", , tbrDefault, "open")
btnX.ToolTipText = "Open File"
btnX.Description = btnX.ToolTipText
Set btnX = Toolbar1.Buttons.Add(, "save", , tbrDefault, "save")
btnX.ToolTipText = "Save File"
btnX.Description = btnX.ToolTipText
Set btnX = Toolbar1.Buttons.Add(, , tbrSeparator)
```

' The next button has the Placeholder style. A

' ComboBox control will be placed on top of this

' button.

```
Set btnX = Toolbar1.Buttons.Add(, "combo1", , tbrPlaceholder)
btnX.Width = 1500 ' Placeholder width to accommodate a combobox.
```

Show ' Show form to continue configuring ComboBox.

' Configure ComboBox control to be at same location

' as the

' Button object with the Placeholder style (key =

' "combo1").

With Combo1

```
.Width = Toolbar1.Buttons("combo1").Width
```

```
.Top = Toolbar1.Buttons("combo1").Top
```

```
.Left = Toolbar1.Buttons("combo1").Left
.AddItem "Black" ' Add colors for text.
.AddItem "Blue"
.AddItem "Red"
.ListIndex = 0
```

```
End With
```

```
End Sub
```

```
Private Sub Form_Resize()
```

```
    ' Configure ComboBox control.
```

```
    With Combo1
```

```
        .Width = Toolbar1.Buttons("combo1").Width
```

```
        .Top = Toolbar1.Buttons("combo1").Top
```

```
        .Left = Toolbar1.Buttons("combo1").Left
```

```
    End With
```

```
End Sub
```

```
Private Sub toolbar1_ButtonClick(ByVal Button As Button)
```

```
    ' Use the Key property with the SelectCase statement to specify  
    ' an action.
```

```
    Select Case Button.Key
```

```
        Case Is = "open"           ' Open file.
```

```
        MsgBox "Add code to open file here!"
    Case Is = "save"           ' Save file.
        MsgBox "Add code to save file here!"
    End Select
End Sub
```

```
Private Sub Combo1_Click()
    ' Change backcolor of form using the ComboBox.
    Select Case Combo1.ListIndex
    Case 0
        Form1.BackColor = vbBlack
    Case 1
        Form1.BackColor = vbBlue
    Case 2
        Form1.BackColor = vbRed
    End Select
End Sub
```

# Add 方法（ButtonMenus 集合）

向 ButtonMenus 集合中添加一个 ButtonMenu 对象，并返回对新创建对象的引用。

## 语法

```
object.Add (index, key, text)
```

Add 方法的语法有下面几部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 必需的。一个对象表达式，其值是“应用于”列表中的一个对象             |
| <i>Index</i>  | 可选的。一个整数，指定对象插入的位置，如果没有指定此参数，对象将添加到集合的末尾 |
| <i>key</i>    | 可选的。一个字符串，唯一标识该对象，使用该值在集合中检索指定的对象        |
| <i>text</i>   | 必需的。出现在菜单中的字符串                           |

## 说明

要查看 **ButtonMenu** 对象，您必须将 **Button** 对象的 **Style** 属性设置为 **tbrDropDown**。

要检测用户单击 **ButtonMenu** 对象的时间，使用 **Toolbar** 控件的 **ButtonMenuClick** 事件。

## 请参阅

**ButtonMenu** 对象，**ButtonMenuClick** 事件，**Style** 属性（**Button** 对象）。

## Add 方法（Buttons 集合）

给 **Buttons** 集合增加一个 **Button** 对象，返回新增对象的引用。

## 应用于

**Toolbar** 控件，**Buttons** 集合。



# 语法

*object.Add (index, key , caption, style, image)*

Add 方法的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个 Buttons 集合的对象表达式  |
| <i>index</i>   | 可选参数。一个整数，指定插入 Button 对象的位置。如果没有指定 index 参数，则将 Button 对象插入的矩阵的末尾 |
| <i>Key</i>     | 可选参数。一个字符串表达式，唯一标识一个 Button 对象。<br>使用该属性检取 Button 对象             |
| <i>caption</i> | 可选参数。字符串表达式，指定 Button 对象上显示的文本                                   |
| <i>style</i>   | 可选参数。Button 对象的风格。可用风格列在 Style 属性（Button 对象）中                    |
| <i>image</i>   | 可选参数。一个整数或唯一禁止，指定关联的 ImageList 控件中的 ListImage 对象                 |

# 说明

在设计时，在 Toolbar 控件的 Properties Page 对话框的 Buttons 选项卡上增加 Button 对象。在运行时，使用 Add 方法增加 Button 对象，代码如下：

Dim btnButton As Button

```
Set btnButton = Toolbar1.Buttons.Add(“open”, , tbrDefault, “open”)
```

通过 **Toolbar** 控件的 **ImageList** 属性，可以将一个 **ImageList** 控件与一个 **Toolbar** 控件关联。

请参阅

**ImageList** 控件，**ListImage** 对象，**ListImages** 集合，**Style** 属性（**Button** 对象），**Index** 属性（**ActiveX** 控件），**Key** 属性（**ActiveX** 控件），**ImageList** 属性（**ActiveX** 控件）。

## AllowCustomize 属性

返回或设置一个值，确定用户是否可以使用 **Customize Toolbar** 对话框自定义一个 **Toolbar** 控件

应用于

**Toolbar** 控件

# 语法

*object.AllowCustomize [=boolean]*

AllowCustomize 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 一个 Toolbar 控件的对象表达式                       |
| <i>boolean</i> | 一个常量或整数，确定用户是否可以自定义一个 Toolbar 控件，如“设置”中所示 |

# 设置

boolean 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 允许用户双击 Toolbar 控件激活 Customize Toolbar 对话框    |
| False | 不允许激活 Customize Toolbar 对话框对 Toolbar 控件进行自定义 |

# 说明

如果 AllowCustomize 属性设置为 True，在运行时双击 Toolbar 控件将激活 Customize Toolbar 对话框。

也可以使用 `Customize` 方法激活 `Customize Toolbar` 对话框。

请参阅

`Customize` 方法，`RestoreToolbar` 方法，`SaveToolbar` 方法。

## Button 对象

一个 `Button` 对象代表了 `Toolbar` 控件中 `Buttons` 集合中的单个按钮。

说明

对于每个 `Button` 对象，你可以增加文本或从 `ImageList` 控件中增加点位图，或者二者；并设置属性改变其状态和风格。

在设计时，在 `Toolbar` 控件的 `Properties Page` 对话框的 `Buttons` 选项卡上，使用 `Insert Button` 和 `Remove Button` 按钮给 `Buttons` 集合增加或删除 `Button` 对象。在运行时，你可以使用 `Buttons` 集合的 `Add` 方法增加 `Button` 对象。

在设计时和运行时，你可以设置 `Caption`，`Image`，`Value`，`MixedState`，`ToolTipText` 属性改变每个 `Button` 对象的外观。

当点击 **Toolbar** 控件中的按钮时，调用 **ButtonClick** 事件，以所点击的按钮作为参数。在点击按钮时，如果要采取某些动作，在 **Select Case** 语句中使用 **Index** 或 **Key** 属性，如下所示：

```
Select Case Button.Key
    Case Is = "open" ' Open file.
        ' Add code to Open a file here
    Case Is = "save" ' Save file.
        ' Add code to Save a file here
    Case Else
        ' If any other button is pressed
End Select
```

## 属性

**ButtonMenus** 属性, **ToolTipText** 属性(**ActiveX** 控件), **Description** 属性(**Button** 对象), **MixedState** 属性, **Style** 属性 (**Button** 对象), **Value** 属性 (**ActiveX** 控件), **Height, Width** 属性 (**ActiveX** 控件), **Index** 属性 (**ActiveX** 控件), **Left, Top** 属性 (**ActiveX** 控件), **Tag** 属性 (**ActiveX** 控件), **Visible** 属性 (**ActiveX** 控件), **Caption** 属性 (**ActiveX** 控件), **Enabled** 属性 (**ActiveX** 控件), **Key** 属性 (**ActiveX** 控件), **Image** 属性 (**ActiveX** 控件)。

## 方法

**Clear** 方法（ActiveX 控件）。

## 请参阅

**ImageList** 控件，**ListImage** 对象，**ListImages** 集合，**Button** 对象，**Style** 属性（**Button** 对象），**Index** 属性（ActiveX 控件），**Key** 属性（ActiveX 控件），**ImageList** 属性（ActiveX 控件）。

## ButtonClick 事件

当用户点击 **Toolbar** 控件中的 **Button** 对象时产生该事件。

## 应用于

**Toolbar** 控件

## 语法

Private Sub *object*\_ButtonClick (ByVal *button* As Button )

ButtonClick 事件的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 一个 Toolbar 控件的对象表达式 |
| <i>button</i> | 一个 Button 对象的引用     |

## 说明

要响应单个 Button 对象的 ButtonClick 事件，使用 **button** 参数值。例如，下面的代码使用 Button 对象的 Key 属性值确定相应的动作：

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
    Select Case Button.Key
        Case "Open"
            CommonDialog1.ShowOpen
        Case "Save"
            CommonDialog1.ShowSave
    End Select
End Sub
```

**注意：**因为用户可以使用 Customize Toolbar 对话框重新安排 Button 按钮，Index 属性可能不总是指定按钮的位置。因此，最好使用 key 属性检取 Button 对象。

请参阅

Button 对象，Index 属性（控件矩阵），Value 属性（ActiveX 控件）。

## ButtonHeight, ButtonWidth 属性

返回或设置 Toolbar 控件中按钮的高和宽。

应用于

Toolbar 控件

语法

*object*.ButtonHeight [=number]

*object*.ButtonWidth [=number]



ButtonHeight, ButtonWidth 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 Toolbar 控件的对象表达式   |
| <i>number</i> | 一个数字表达式，指定 Toolbar 控件中所有具有 Button, Check 或 ButtonGroup 风格按钮尺寸 |

## 说明

ButtonHeight 和 ButtonWidth 使用 Toolbar 控件容器的度量单位。该单位由容器的 ScaleMode 属性决定。

缺省地，ButtonWidth 和 ButtonHeight 属性将自动更新以便容纳 Button 对象 Caption 属性中的字符串或 Image 属性中的图像。

## 请参阅

Button 对象，Image 属性，Caption 属性， ScaleMode 属性，Caption 属性（ActiveX 控件）。

## ButtonMenu 对象

ButtonMenu 对象代表一个由 Toolbar 控件的 Button 对象下拉的菜单。

### 语法

ButtonMenu

### 说明

只有当 Button 对象的 Style 属性设置为 tbrDropdown.时按钮菜单才出现。

### 属性

Parent 属性（ButtonMenu 对象），ToolTipText 属性，Text 属性（ActiveX 控件），Index 属性（ActiveX 控件），Tag 属性（ActiveX 控件），Enabled 属性（ActiveX 控件），Key 属性（ActiveX 控件）。

请参阅

**ButtonMenus** 集合, **ButtonMenus** 属性, **ButtonMenuClick** 事件, **Style** 属性 (**Button** 对象)。

示例

本例在一个 **Toolbar** 控件中添加五个 **Button** 对象, 并且向每个 **Button** 对象添加二个 **ButtonMenu** 对象。单击 **ButtonMenu** 对象时, 其行为由 **ButtonMenuClick** 事件来决定。为了试验本例, 在窗体中放置一个 **Toolbar** 控件, 将代码粘贴到代码模块的声明部分。

Option Explicit

Private Sub Form\_Load()

Dim i As Integer

Dim btn As Button

' 添加五个 Button 对象到 Toolbar 控件

For i = 1 To 5

Set btn = Toolbar1.Buttons.Add(Caption:= i, Style:= tbrDropDown)

' 添加两个 ButtonMenu 对象到每一个 Button。

btn.ButtonMenus.Add Text:="Help"

```
        btn.ButtonMenus.Add Text:="Options"
    Next i
End Sub

Private Sub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As ComctlLib.ButtonMenu)
    Select Case ButtonMenu.Index
        Case 1
            MsgBox "Press the button."
        Case 2
            MsgBox "Offer some option"
    End Select
End Sub
```

## ButtonMenuClick 事件

当用户单击一个 **ButtonMenu** 对象时发生。

应用于

**Toolbar** 控件

## 语法

```
Private Sub object_ButtonMenuClick([index As Integer,]ByVal ButtonMenu As ComctlLib.ButtonMenu)
```

ButtonMenuClick 事件语法有下面几部分：

| 部分                | 描述                              |
|-------------------|---------------------------------|
| <i>object</i>     | 一个对象表达式，其值是“应用于”列表中的一个对象        |
| <i>index</i>      | 一个整数，它唯一标识控件数组中的一个控件            |
| <i>buttonMenu</i> | 对被单击的 <b>ButtonMenu</b> 对象的一个引用 |

## 说明

使用 **ButtonMenuClick** 事件时，可以使用 **ButtonMenu** 对象的 **Parent** 属性决定哪一个按钮被单击。

## 请参阅

**ButtonMenu** 对象，**ButtonMenus** 集合，**ButtonMenus** 属性。

## ButtonMenus 集合

**ButtonMenu** 对象的集合。

语法

**ButtonMenus**

说明

使用 **ButtonMenus** 属性返回对该集合的一个引用。

属性

**Count** 属性（ActiveX 控件），**Item** 属性（ActiveX 控件）。

方法

**Add** 方法（**ButtonMenus** 集合），**Remove** 方法（ActiveX 控件），**Clear** 方法（ActiveX 控件）。

请参阅

ButtonMenu 对象，ButtonMenus 属性

## ButtonMenus 属性

返回一个对 ButtonMenus 集合的引用，仅在运行时有效。

应用于

Button 对象。

语法

*object*.ButtonMenus

*object* 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

请参阅

ButtonMenu 对象，ButtonMenus 集合。

# Buttons 集合

Buttons 集合是 Toolbar 控件中 Button 对象的集合。

## 语法

*toolbar.Buttons (index)*

*toolbar.Buttons.Item (index)*

Buttons 集合的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>toolbar</i> | 一个 Toolbar 控件的对象表达式                              |
| <i>index</i>   | 一个整数或字符串，唯一标识集合中的一个对象。整数是 Index 属性值；字符串是 Key 属性值 |

## 说明

Buttons 集合是基于 1 的集合，这意味着该集合的 Index 属性从 1 开始。

集合中的每个元素都可以通过索引或键值来访问。例如，要获取集合中第三个 Button 对象的引用，使用下面的代码：

```
Dim btnX As Button
```



' Reference by index number.

Set btnX = Toolbar1.Buttons(3)

' Or reference by unique key.

Set btnX = Toolbar1.Buttons("third") ' Assuming Key is "third."

' Or use Item method.

Set btnX = Toolbar1.Buttons.Item(3)

## 属性

Count 属性（ActiveX 控件），Item 属性（ActiveX 控件）。

## 方法

Clear 方法（ActiveX 控件），Remove 方法（ActiveX 控件），Add 方法（Buttons 集合）。

## 请参阅

Key 属性（ActiveX 控件），Index 属性（ActiveX 控件）。

## Buttons 属性

返回 **Toolbar** 控件中 **Button** 对象集合的引用。

应用于

**Toolbar** 控件

语法

*object*.Buttons

*object* 是 **Toolbar** 控件的对象表达式。

说明

可以使用标准的集合方法（如 **Add** 和 **Remove** 方法）对 **Button** 对象进行操作。集合中的每个元素都可以通过 **Index** 属性或 **Key** 属性来访问。

请参阅

**Button** 集合，**Add** 方法（**Buttons** 集合），**Index** 属性（控件矩阵），**Clear**

方法（ActiveX 控件）。

## Change 事件（Toolbar, Slider 控件）

表明控件中的内容发生了变化。该事件如何以及何时产生与不同的控件有关。

应用于

DBCombo 控件，ComboBox 控件，DirListBox 控件，DriveListBox 控件，HScrollBar, VScrollBar 控件，PictureBox 控件，TextBox 控件

语法

Private Sub *object*\_Change( [*index* As Integer ])

Change 事件的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 在“应用于”中指定的控件对象表达式   |
| <i>index</i>  | 一个整数，唯一标识控件矩阵中的某个控件 |

## 说明

- **Slider**——当 **Value** 属性值发生改变时产生该事件，不管是通过程序代码修改还是用户移动了 控件的滑鼠。
- **Toolbar**——当用户使用 **Customize Toolbar** 对话框定制一个 **ToolBar** 控件时产生该事件。

**Change** 事件过程可以同步控件中的数据显示。例如，你可以使用 **Slider** 控件的 **Change** 事件过程更新 **TextBox** 控件中的 **Value** 属性值。或者，可以使用 **Change** 事件过程在工作区域显示数据和公式，而在结果区域显示数据结果。

**注意：****Change** 事件过程有时会导致连锁事件。当你在程序代码中使用控件的 **Change** 事件修改控件的内容，而该内容又决定控件的值如 **TextBox** 控件中的 **Text** 属性，此时就发生了连锁事件。要阻止连锁事件：

- 如果有可能，避免编程修改控件内容的 **Change** 事件过程。如果你一定要编写这样的过程，确保设置一个标记阻止进一步产生事件。
- 避免创建两个在 **Change** 事件中彼此影响内容的控件例如，两个 **TextBox** 控件在 **Change** 事件过程中彼此更新。

## 请参阅

**AllowCustomize** 属性，**Customize** 方法，**RestoreToolbar** 方法，**SaveToolbar**

方法，Text 属性，KeyDown, KeyUp 事件，KeyPress 事件，LostFocus 事件，PathChange 事件，PatternChange 事件，Picture 属性，Text 属性，Value 属性，Drive 属性，LinkTopic 属性，Style 属性，Caption 属性，Path 属性，Picture 属性（ActiveX 控件），Text 属性（ActiveX 控件），Value 属性（ActiveX 控件），Caption 属性（ActiveX 控件）。

## 示例

本例在 TextBox 控件中显示水平滚动条的 Value 属性的数值。要尝试这个例子，需创建一个带有 TextBox 控件及 HScrollBar 控件的窗体，然后将码粘贴到一个带有水平滚动条 (HScrollBar 控件) 和 TextBox 控件的窗体的声明部分。按 F5 键并单击水平滚动条。

```
Private Sub Form_Load ()
    HScroll1.Min = 0           ' 设置最小值。
    HScroll1.Max = 1000       ' 设置最大值。
    HScroll1.LargeChange = 100 ' 设置 LargeChange.
    HScroll1.SmallChange = 1  ' 设置 SmallChange.
End Sub

Private Sub HScroll1_Change ()
    Text1.Text = HScroll1.Value
End Sub
```

# Controls 属性（Toolbar 控件）

返回对象中包含的控件集合的引用。

应用于

Toolbar 控件

语法

*object.Controls (index)*

*object.Controls.Item (index)*

Controls 属性的语法有如下几个部分：

| 部分            | 描述                 |
|---------------|--------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式    |
| <i>index</i>  | 标识 Controls 集合成员的值 |

说明

Controls 属性与 Form 对象中的 Cotrols 集合相似，访问方法也相同。例如，使用下面的代码获取 Toolbar 控件中第二个控件的 Top 属性值。

MsgBox Toolbar1.Controls (2) .Top

使用 **Controls** 属性，你可以遍历 **Tab** 对象中或 **Toolbar** 控件中所有的控件，并使用如下的代码改变每个控件的属性：

```
Dim ctlX As Control
For Each ctlX In Toolbar1.Controls
    ctlX.Width =
        Toolbar1.Width / Toolbar1.Controls.Count
Next
```

**注意：**Controls 集合是指 Toolbar 控件中包含的控件，如 Combox 控件，而不是 Button 对象，Button 对象是 Toolbar 控件的一部分。

请参阅

Tab 对象，Button 对象，ComboBox 控件，Controls 集合，Form 对象，Forms 集合，Controls 集合，Left, Top 属性，Index 属性（控件矩阵）。

## Customize 方法

激活 Customize Toolbar 对话框,允许用户重新安排或隐藏 Toolbar 控件中的

Button 对象。

应用于

Toolbar 控件

语法

*object*.Customize

*object* 是一个 Toolbar 控件的对象表达式。

说明

Toolbar 控件包含了一个内置的对话框，允许用户隐藏、显示和重新安排工具条上的按钮。双击工具条将调用 Customize 方法，激活该对话框。

如果你想限制对工具条的修改，可以使用 Customize 方法。例如，下面的代码只允许通过了正确口令的用户修改工具条：

```
Private Sub Command1_Click()  
    If InputBox("Password:") = "Chorus&Line9" Then  
        Toolbar1.Customize    ' Invoke Customize method.  
    End If  
End Sub
```



要保存 Toolbar 控件的状态，使用 SaveToolbar 方法将状态写入 Windows 登记项数据库中（Registry）。可以使用 RestoreToolbar 方法读取以前保存在登记项数据库中 Toolbar 控件的状态。

请参阅

Button 对象，AllowCustomize 属性，Description 属性（Button 对象），RestoreToolbar 方法，SaveToolbar 方法。

## Description 属性（Button 对象）

返回或设置 Button 对象的描述性文本，该文本显示在 Customize Toolbar 对话框中。

应用于

Toolbar 控件，Button 对象。

语法

object.Description [=string]

Description 属性的语法有如下几个部分：

| 部分            | 描述                                       |
|---------------|--|
| <i>object</i> | 一个 Button 对象的表达式                         |
| <i>string</i> | 当选择该按钮时，在 Customize Toolbar 对话框中显示的描述性文本 |

## 说明

在运行时，用户双击 Toolbar 控件或使用 Customize 方法可以激活 Customize Toolbar 对话框。无论哪种方法，当用户在对话框中选择一个按钮时，按钮的描述性文本就显示在对话框的左下角。使用 Description 属性设置描述性文本。

当你增加 Button 对象时，可以使用下面的代码设置 Description 属性文本：

```
Dim btnX As Button
' Add a button with the Key "save."
Set btnX = Toolbar1.Buttons.Add("save")
btnX.Description = "Save a file."
```

## 请参阅

AllowCustomize 属性，Customize 方法。

## DeselectAll 方法

清除 TabStrip 控件上所有选中的 Tab 对象。

应用于

TabStrip 控件

语法

*object*.DeselectAll

*object* 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

说明

仅当用户选中了多个选项卡时使用该方法，而只有 **MultiSelect** 属性设置为 **True** 时用户才能选择多个选项卡。

请参阅

**Multiselect** 属性（**ListView**, **TabStrip** 控件）。

# DisabledImageList 属性

返回或设置用于使图像无效的 ImageList 控件

## 语法

```
object.DisabledImageList [= imageList]
```

| 部分               | 描述                               |
|------------------|----------------------------------|
| <i>object</i>    | 一个对象表达式，其值是“应用于”列表中的一个对象         |
| <i>imageList</i> | 一个对象引用，指定使用哪一个 ImageList 控件使图像无效 |

## 说明

Button 对象只能为每一个按钮显示一个图像。运行时，它首先决定应该如何绘制按钮（例如：正常、“热点”或无效），然后使用唯一的 Image 属性作为关键字，使用相应的图像列表( ImageList、DisabledImageList 或 HotImageList) 的图像。了解三个图像列表中相关图像的命名必须一致是很重要的，只有这样 Toolbar 控件才能找出正确的图像。例如，如果一个特别的按钮使用了所有三类图像，则三个图像的每一个在其各自的图像列表中必须定义与另外二个图像相同的 Index，或相同的 Key。

将 `Toolbar` 控件的 `Enabled` 属性设置为 `False` 将不会显示 `DisabledImageList` 所包含的图像。而将 `Button` 对象的 `Enabled` 属性设置为 `False` 将从 `DisabledImageList` 显示一个图像。要禁用所有的按钮，请使用下列代码：

```
Private Sub DisableAllButtons
    Dim b As Button
    For Each b In Toolbar1.Buttons
        b.Enabled = False
    Next
End Sub
```

**注意：**赋值给 `DisabledImageList` 属性的 `ImageList` 控件中的图像必须与赋值给 `ImageList` 属性的 `ImageList` 控件中的图像大小相同。

请参阅

`HotImageList` 属性，`Image` 属性（`ActiveX` 控件）。

示例

下面的示例演示了 `DisableImageList`、`HotImageList` 和 `ImageList` 属性的用法，请注意，所有三个图像列表使用相同的 `Key` 属性。示例中假定命名为”

imgNormal”、” imgHot”和” imgDisabled”的三个 ImageList 控件放置在一个窗体中。

```
Private Sub Form_Load()
```

```
    Dim str As String
```

```
    str = "D:\VB\Icons\Misc\" '改变其值以匹配您的图形位置.
```

```
    imgNormal.ListImages.Add Key:="face", _
```

```
    Picture:=LoadPicture(str & "Face02.ico")
```

```
    imgHot.ListImages.Add Key:="face", _
```

```
    Picture:=LoadPicture(str & "Face03.ico")
```

```
    imgDisabled.ListImages.Add Key:="face", _
```

```
    Picture:=LoadPicture(str & "Face01.ico")
```

```
    ' 设置 ImageList、DisableImageList、和 HotImageList 属性。
```

```
With Toolbar1
```

```
    .ImageList = imgNormal
```

```
    .DisabledImageList = imgDisabled
```

```
    .HotImageList = imgHot
```

```
    .Buttons.Add Caption:="Sad", Image:="face", Style:=tbrCheck
```

```
    .Buttons.Add Caption:="Happy", Image:="face", Style:=tbrCheck
```

```
    .Buttons.Add Caption:="Face", Image:="face", Style:=tbrCheck
```

```
    .Buttons(1).Enabled = False
```

```
    .Buttons(3).Value = tbrPressed
```

End With

End Sub

## HelpContextID 属性（Toolbar 控件）

返回或设置与对象关联的文本数字。用于为应用程序提供与环境有关的帮助信息。

**注意：**Toolbar 控件的 HelpContextID 属性从 Customize Toolbar 对话框中而不是控件自己建立一个 Help 连接。这与其他包含 HelpContextID 属性的 Visual Basic 控件不一样。

应用于

Toolbar 控件

语法

*object*.HelpContextID [=*number*]

HelpContextID 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 在“应用于”中指定的对象表达式。如果省略 <i>object</i> 对象，与当前活动窗体模块关联的 <b>Form</b> 就假设为 <i>object</i> |
| <i>number</i> | 数字表达式，指定与 <i>object</i> 关联的 <b>Help</b> 主体的内容数字                                   |

## 设置

*number* 值的设置如下：

| 设置 | 描述             |
|----|----------------|
| 0  | （缺省）不指定环境数字    |
| >0 | 一个整数，指定有效的环境数字 |

## 说明

对于应用程序中的与环境有关的 **Help**，必须在编译你的 **Help** 文件时分配给 *object* 和关联的 **Help** 主体相同的环境数字。

如果创建了 Microsoft Windows 操作系统 **Help** 文件，并设置了应用程序的 **HelpFile** 属性，则当用户按下 F1 键时，Visual Basic 将自动调用 **Help** 并搜索当前环境数字标识的主体。

当前的环境数字是有输入焦点对象的 **HelpContextID** 属性值。如果 **HelpContextID** 设置为 0，则 Visual Basic 查找对象容器中的 **HelpContextID**，依



次向上。如果没有找到当前环境数字，则忽略 F1 按键。

对于一个 **Menu** 控件，通常 **HelpContextId** 在运行时是可以读写的。在如果菜单项是有 **Visual Basic** 提供的 **Add-in** 或其他控件提供的，则 **HelpContextID** 属性是只读的。

请参阅

**HelpFile** 属性（**Toolbar** 控件），**Menu** 控件，**Customize** 对话框。

## **HelpFile** 属性（**Toolbar** 控件）

找到你的应用程序中使用的 **Microsoft Windows Help** 文件或显示 **Help** 或在线文档的路径和文件名。

**注意：****Toolbar** 控件的 **HelpFile** 属性可以从 **Customize Toolbar** 对话框而不是控件本身建立 **Help** 连接。这与 **Visual Basic** 中包含 **HelpFile** 属性的其他控件不同。

应用于

Toolbar 控件

语法

*object.HelpFile [=filename]*

HelpFile 属性的语法有如下几个部分：

| 部分              | 描述                                      |
|-----------------|---|
| <i>object</i>   | 在“应用于”中指定的对象表达式                         |
| <i>filename</i> | 字符串表达式，找到你的应用程序中 Windows Help 文件的路径和文件名 |

说明

如果创建了 Microsoft Windows 操作系统 Help 文件，并设置了应用程序的 HelpFile 属性，则当用户按下 F1 键时，Visual Basic 将自动调用 Help 文件。如果当前活动控件和活动窗体都没有设置 HelpContextID 属性，Help 就显示与当前 Help 环境相对应的主体；否则显示主要的内容。

也可以使用 HelpFile 属性找到当用户从 ActiveX 组件的 Object Browser 中调

用 **Help** 文件时显示哪个 **Help** 文件。

请参阅

**HelpContextID** 属性，**Customize** 对话框。

## HotImageList 属性

返回或设置 **ImageList** 控件，用于“热点”图像——当光标停留在一个可单击的位置时显示，并且 **Style** 属性设置为 **tbrTransparent**。

应用于

**Toolbar** 控件

语法

*object*.HotImageList [= *imageList*]

| 部分               | 描述                                       |
|------------------|--|
| <i>object</i>    | 一个对象表达式，其值是“应用于”列表中的一个对象                 |
| <i>imageList</i> | 一个对象引用，指定哪一个 <b>ImageList</b> 控件用于“热点”图像 |

## 说明

**Button** 对象只能为每一个按钮显示一个图像。运行时，它首先决定应该如何绘制按钮（如：正常、“热点”或无效），然后使用唯一的 **Image** 属性作为关键字，使用相应的图像列表( **ImageList**、**DisabledImageList** 或 **HotImageList**) 中的图像。了解三个图像列表中相关图像的命名必须一致是很重要的，只有这样 **Toolbar** 控件才能找出正确的图像。例如，如果一个特别的按钮使用了所有三类图像，则三个图像的每一个在其各自的图像列表中必须定义与另外二个图像相同的 **Index**，或相同的 **Key**。

## 请参阅

**DisableImageList** 属性，**Image** 属性（**ActiveX** 控件）。

# MixedState 属性

返回或设置一个值，该值确定 **Toolbar** 控件中的 **Button** 对象是否显示中介状态。

应用于

**Toolbar** 控件，**Button** 对象。

语法

*object*.MixedState [=*boolean*]

MixedState 属性的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个 <b>Button</b> 对象的表达式                      |
| <i>boolean</i> | 一个布尔表达式，确定 <b>Button</b> 对象是否显示中介状态，如“设置”中所示 |

## 设置

boolean 值的设置如下：

| 设置    | 描述                     |
|-------|------------------------|
| True  | Button 控件显示中介状态，变灰     |
| False | Button 控件不显示中介状态，看起来正常 |

## 说明

当选择的内容包含了多种属性时通常使用 **MixedState** 属性。例如，如果你选择的文本包含了普通字符和黑体字符，则使用 **MixedState** 属性。然后可以改变 Button 对象所使用的图像，以表明其状态，与 Value 属性返回的 **Checked** 和 **Unchecked** 值不同。

## 请参阅

Value 属性（ActiveX 控件）。

# Parent 属性（ButtonMenu 对象）

返回或设置一个对 ButtonMenu 对象的父对象的引用。

语 法

```
object.Parent [= button]
```

Parent 属性的语法有下面几部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>button</i> | ButtonMenu 对象的父 Button 对象 |

# RestoreToolbar 方法

将 Toolbar 控件创建的工具条恢复到自定义以前的状态。

应用于

Toolbar 控件

语法

*object*.RestoreToolbar ( *key* As String, *subkey* As String, *value* As String)

RestoreToolbar 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 Toolbar 控件的对象表达式                         |
| <i>key</i>    | 字符串表达式，指定检取 Toolbar 信息的 Windows 登记项数据库的键值   |
| <i>subkey</i> | 字符串表达式，指定登记项数据库中 <i>key</i> 参数的子键           |
| <i>vlaue</i>  | 字符串表达式，标识在 <i>subkey</i> 下存放的 Toolbar 控件的信息 |

说明

**警告：**当使用 RestoreToolbar 方法时，任何不包含 ImageList ListImage 对象的工具条按钮都将消失。用户可以使用 Customize Toolbar 对话框中的 Reset 按钮使之重新可视。你可以在程序代码中为用户激活该对话框。

要在运行时自定义 Toolbar 控件，在程序代码中调用 Customize 方法或如果



**AllowCustomize** 属性设置为 **Ture**，用户可以双击该控件进行自定义。

工具条的状态可以使用 **SaveToolbar** 方法保存到登记项数据库中。  
**RestoreToolbar** 方法读取登记项数据库中的信息恢复工具条的状态。

下面的代码恢复了 **Toolbar** 控件的状态，假设它们已使用 **SaveToolbar** 方法保存在登记项数据库中：

```
Toolbar1.RestoreToolbar “AppName”, “User1”, “Toolbar1”
```

请参阅

**AllowCustomize** 属性，**Customize** 方法，**SaveToolbar** 方法。

## SaveToolbar 方法

在运行时，将 **Toolbar** 控件创建的工具条状态保存到登记项数据库中。

应用于

**Toolbar** 控件

# 语法

*object*.SaveToolbar (*key* As String, *subkey* As String, *value* As String )

SaveToolbar 方法的语法有如下几个部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 一个 Toolbar 控件的对象表达式                 |
| <i>key</i>    | 一个字符串表达式，指定保存 Toolbar 控件信息的登记项数据库键值 |
| <i>subkey</i> | 一个字符串表达式，指定 <b>key</b> 键值下面的子键      |
| <i>value</i>  | 在 <i>subkey</i> 中保存的 Toolbar 控件信息   |

# 说明

要在运行时自定义 Toolbar 控件，在代码中使用 **Customize** 方法或如果 **AllowCustomize** 属性设置为 **True**，用户可以双击该控件进行自定义。

如果指定的 *key*, *subkey* 或 *value* 不存在，则就在登记项数据库中创建它。

要保存多个版本的工具条，你可以改变 **subkey** 和 **value** 参数。这导致工具条写登记项数据库中的不同部分。下面的代码保存了工具条的两个版本：

```
' Save settings for User1
Toolbar1.SaveToolbar "AppName", "User1", "Toolbar1"
```

```
' Save settings for User2
```

```
Toolbar1.SaveToolbar "AppName", "User2", "Toolbar1"
```

因为在工具条自定义之后产生 **Change** 事件，所以可以将上面的代码放入工具条的 **Change** 事件中。

请参阅

**AllowCustomize** 属性，**Customize** 方法，**RestoreToolbar** 方法。

## Style 属性（Button 对象）

返回或设置一个常量或值，确定 **Toolbar** 控件中 **Button** 对象的外观和行为。

应用于

**Toolbar** 控件，**Button** 对象。

语法

*object*.Style [=value]

**Style** 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 <b>Button</b> 对象表达式                    |
| <i>value</i>  | 一个常量或值，确定 <b>Button</b> 对象的外观和行为，如“设置”中所示 |

## 设置

value 值的设置如下：

| 常量             | 值 | 描述  |
|----------------|---|---|
| TbrDefault     | 0 | （缺省） <b>Buttton</b> 。是普通的按钮                             |
| TbrCheck       | 1 | <b>Check</b> 。检查框按钮                                     |
| TbrButtonGroup | 2 | <b>ButtonGroup</b> 。该按钮保持为按下状态，直到另一个按钮按下。任何时刻组中只有一个按钮按下 |
| TbrSeparator   | 3 | <b>Separator</b> 。按钮充当分隔条，8 个像素的固定宽度                    |
| TbrPlaceholder | 4 | <b>Placeholder</b> 。该按钮与分隔条相似，但宽度可以设置                   |
| TbrDropDown    | 5 | <b>MenuButton</b> 下拉列表。使用该风格查看 <b>MenuButton</b> 对象     |

## 说明

具有 **ButtonGroup** 风格的按钮必须成组。要区分不同的组，将所有相同风格（**ButtonGroup**）的 **Button** 对象放置在两个 **Separator** 风格的按钮之间。

你可以为 **Button** 对象分配 **Placeholder** 风格从而在工具条上放置另一个控件，然后在工具条上绘制该控件例如，增加一个 **Button** 对象，其为 **Placeholder** 风格，大小设置为一个 **ComboBox** 控件的大小。然后在工具条上放置一个 **ComboBox** 控件

当一个 **Button** 对象指定为 **Placeholder** 风格时，你可以设置其 **Width** 属性以便容纳另一个控件如果 **Button** 对象有 **Button**, **Check** 或 **ButtonGroup** 风格，则高度和宽度由 **ButtonHeight** 和 **ButtonWidth** 属性决定。

如果你在控件上放置一个 **Placeholder** 风格的按钮，如果窗体改变了大小，你必须使用程序代码重新设置控件的大小和对齐控件，如下所示：

```
Private Sub Form_Resize()  
    ' Track a ComboBox by setting its Top, Left, and  
    ' Width properties  
    ' to the Top, Left, and Width properties of a  
    ' Button object  
    With Toolbar1.Buttons("Combo1")  
        Combo1.Move .Left,.Top,.Width  
    End With  
End Sub
```

请参阅

ButtonMenu 对象， ButtonHeight, ButtonWidth 属性， Toolbar 控件常量， ComboBox 控件， Height, width 属性。

## Style 属性（Toolbar 控件）

返回或设置一个值，决定控件的外观。

应用于

Toolbar 控件

语法

*object*.Style [= *integer*]

Style 属性的语法有下面几部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>integer</i> | 一个数值表达式，其值决定控件的外观，如设置值所示 |

## 设置

*integer* 的设置值为：

| 常量             | 描述  |
|----------------|---|
| tbrStandard    | （缺省）标准工具栏                                     |
| tbrTransparent | 按钮和工具栏都是透明的，按钮文本出现在按钮位图下面并启动热跟踪               |
| tbrRight       | 如果有按钮文本，则它出现在图像的右边，除此之外，与“tbrTransparent”样式相同 |

## TextAlignment 属性

该属性返回或设置的数值确定了文本相对于按钮的位置。

# 语法

*object*.TextAlignment [= *integer*]

TextAlignment 属性的语法包括这些部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 一个对象表达式，其值为“应用于”列表中的一个对象 |
| <i>integer</i> | 文本的位置，如“设置”所示            |

# 设置

*integer* 的设置值包括：

| 常量                 | 值 | 描述         |
|--------------------|---|------------|
| TbrTextAlignBottom | 0 | 文本与按钮的底部对齐 |
| TbrTextAlignRight  | 1 | 文本与按钮的右侧对齐 |

# Wrappable 属性

返回或设置一个值，确定当窗口改变大小时是否自动对 **Toolbar** 控件分行显示。



应用于

Toolbar 控件

语法

*object*.Wrappable [=boolean]

Wrappable 属性的语法有如下几个部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个 Toolbar 控件的对象表达式                                |
| <i>boolean</i> | 一个布尔表达式，确定 Toolbar 控件中的 Button 大小对象是否分行显示，如“设置”中所示 |

设置

boolean 值的设置如下：

| 值     | 描述                          |
|-------|-----------------------------|
| True  | 如果窗体改变了大小，则 Toolbar 控件分行显示  |
| False | 如果窗体改变了大小，则 Toolbar 控件不分行显示 |

请参阅

**Button** 对象。

# TreeView 控件

一个 **TreeView** 控件显示 **Node** 对象的等级体系结构，每个 **Node** 对象包含了一个标签和可选的点位图。**TreeView** 控件通常用于显示文档头、索引中的条目、磁盘上的文件和目录或者可以显示为等级结构的各种其他信息。

## 语法

**TreeView**

## 说明

在创建了一个 **TreeView** 控件之后，你可以设置 **Node** 对象的属性和调用其方法增加、删除或者操纵 **Node** 对象。你可以编程展开或收缩 **Node** 节点以便显示或隐藏所有的子节点。3 个事件 **Collapse**, **Expand** 和 **NodeClick** 提供了在程序中使用功能。

使用 **Root**, **Parent**, **Child**, **FirstSibling**, **Next**, **Previous** 和 **LastSibling** 属性可以检取 **Node** 对象的引用，从而在程序代码中浏览节点树。用户也可以使用键盘

来浏览节点。**UP ARROW** 和 **DOWN ARROW** 键循环展开所有的 Node 对象。Node 对象的选择从左到右，从上到下。在树的根部，选择则跳到树头，如果有必要就滚动窗口。**RIGHT ARROW** 和 **LEFT ARROW** 键也可以跳过展开的 Node 节点，但如果按下 **RIGHT ARROW** 键时选择了一个未展开的 Node 对象，则该对象就展开。第二次按键将选择移到下一个 Node 对象。相反，如果按下 **LEFT ARROW** 键时展开的 Node 节点有焦点，则收缩该 Node 对象。如果用户按下了 **ANSI** 键，则焦点将跳到最近的以该 ANSI 字符开头的 Node 对象节点。后续的按键将导致选择轮回所有以该字符开头的展开节点。

**TreeView** 控件的外观有几种选择。Node 对象可以表现为文本，点位图，线条和加减号的 8 种组合之一。

**TreeView** 控件使用 **ImageList** 属性指定的 **ImageList** 控件保存在 Node 对象中使用的点位图和图标。一个 **TreeView** 控件一次只能使用一个 **ImageList**。这意味着当 **TreeView** 控件的 **Style** 属性设置为显示图像的风格时，**TreeView** 控件中的每个成员旁边都显示一个等大的图像。

**发布须知：**TreeView 控件是 **MSCOMCTL32.OCX** 文件中 **ActiveX** 控件的一部分。要在你的应用程序中使用 **TreeView** 控件，你必须将 **MSCOMCTL32.OCX** 文件加入到你的工程文件中。在发布你的应用程序时，应在用户的 **Microsoft Windows System** 或 **System32** 目录下安装 **MSCOMCTL32.OCX** 文件。

## 属性

DropHighlight 属性 (ListView, TreeView 控件), Indentation 属性, LabelEdit 属性, LineStyle 属性, Nodes 属性, PathSeparator 属性 (TreeView 控件), SelectedItem 属性 (ActiveX 控件), Sorted 属性 (TreeView 控件), Style 属性 (TreeView 控件), Checkboxes 属性, FullRowSelect 属性, Scroll 属性, SingleSel 属性, HotTracking 属性, TabIndex 属性, Tag 属性, Visible 属性, DragIcon 属性, DragMode 属性, CausesValidation 属性, MouseIcon 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDragMode 属性 (ActiveX 控件), OLEDropMode 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), Appearance 属性 (ActiveX 控件), BorderStyle 属性 (ActiveX 控件), Enable 属性 (ActiveX 控件), Font 属性 (ActiveX 控件), HideSelection 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件), MousePointer 属性 (ActiveX 控件), ImageList 属性 (ActiveX 控件)。

## 方法

GetVisibleCount 方法, HitTest 方法 (ListView, TreeView 控件), StartLabelEdit 方法, SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis

方法, OLEDrag 方法 (ActiveX 控件), Refresh 方法 (ActiveX 控件)。

## 事件

Collapse 事件 (TreeView 控件), AfterLabelEdit 事件 (ListView,TreeView 控件), BeforeLabelEdit 事件 (ListView,TreeView 控件), Expand 事件 (TreeView 控件), NodeClick 事件, NodeCheck 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, MouseDown, MouseUp 事件, MouseMove 方法, Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 事件 (ActiveX 控件), OLEDragOver 事件 (ActiveX 控件), OLEGiveFeedback 事件 (ActiveX 控件), OLESetData 事件 (ActiveX 控件), OLEStartDrag 事件 (ActiveX 控件), Click 事件 (ActiveX 控件), DblClick 事件 (ActiveX 控件), KeyDown, KeyUp 事件 (ActiveX 控件), KeyPress 事件 (ActiveX 控件)。

## 请参阅

TreeView 控件常量,Node 对象,Nodes 集合,Child 属性(Node 对象),FirstSibling 属性,LastSibling 属性,Next 属性,NodeClick 事件,ImageList 控件,使用 TreeView 控件。

# Add 方法（Nodes 集合）

给 TreeView 控件中的 Nodes 集合增加一个 Node 对象。

应用于

TreeView 控件。

语法

*object.Add (relative, relationship, key, text, image, selectedimage)*

Add 方法的语法有如下几个部分：

| 部分                  | 描述   |
|---------------------|--|
| <i>object</i>       | 一个 TreeView 控件的对象表达式   |
| <i>relative</i>     | 可选参数。已存在的 Node 对象的索引或键值。新节点与该已存在的节点之间的关系在下一个参数 <b>relationship</b> 中 |
| <i>relationship</i> | 可选参数。指定 Node 对象的相对位置关系，如“设置”中所示                                      |
| <i>key</i>          | 可选参数。使用 <b>Item</b> 方法检取 Node 对象的唯一字符串                               |

| 部分                   | 描述  |
|----------------------|---|
| <i>text</i>          | 在 Node 中显示的字符串                              |
| <i>image</i>         | 可选参数。与 ImageList 控件关联的图像索引                  |
| <i>selectedimage</i> | 可选参数。当 Node 对象选中时，所显示的与 ImageList 控件关联的图像索引 |

设置

relationship 值的设置如下：

| 常量          | 值 | 描述   |
|-------------|---|--|
| TvwFirst    | 0 | 第一个。该 Node 节点放在 relative 命名的所有同级节点的前面                  |
| TvwLast     | 1 | 最后一个。该 Node 节点放在 relative 命名的所有同级节点的最后。后续增加的节点可以在该节点之后 |
| TvwNext     | 2 | （缺省）下一个。该 Node 节点放置在 relative 命名的节点之后                  |
| TvwPrevious | 3 | 前一个。该 Node 节点放置在 relative 命名的节点之前                      |
| TvwChild    | 4 | 子节点。该 Node 节点是 relative 命名节点的子节点                       |

注意：如果在 relative 参数中没有指定 Node 对象，则该对象放在顶级节点中的最后位置。



## 说明

Nodes 集合是基于 1 的集合。

增加一个 Node 对象时就给其分配了索引值，保存在该 Node 对象的 Index 属性中。新成员的该属性值也是 Nodes 集合的 Count 属性值。

因为 Add 方法返回新增 Node 对象的引用，所以设置新 Node 对象属性最方便的方法就是使用该引用值。下面的例子增加了几个属性相同的 Node 对象：

```
Dim nodX As Node      ' Declare the object variable.
Dim I as Integer      ' Declare a counter variable.
For I = 1 to 4
    Set nodX = TreeView1.Nodes.Add(,, "Node " & Cstr(i))
    ' Use the reference to set other properties, such as Enabled.
    nodX.Enabled = True
    ' Set image property to image 3 in an associated ImageList.
    nodX.ExpandedImage = 3
Next I
```

请参阅

Item 属性，Item 方法，ImageList 控件，Count 属性（VB 集合），Index 属

性（ActiveX 控件），Key 属性（ActiveX 控件）。

## 示例

下面的例子给一个 TreeView 控件增加了两个 Node 对象。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，点击 Node 对象展开它。

```
Private Sub Form_Load()  
    ' Set Treeview control properties.  
    TreeView1.LineStyle = tvwRootLines ' Linstyle 1  
  
    ' Add Node objects.  
    Dim nodX As Node ' Declare Node variable.  
    ' First node with 'Root' as text.  
    Set nodX = TreeView1.Nodes.Add(, "r", "Root")  
  
    ' This next node is a child of Node 1 ("Root").  
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "child1", "Child")  
  
End Sub
```

# AfterLabelEdit 事件（ListView，TreeView 控件）

当用户编辑当前选择的 Node 对象或 ListItem 对象的标签时产生该事件。

应用于

TreeView 控件，Listview 控件。

语法

Private Sub *object*\_AfterLabelEdit (*cancel* As Integer, *newstring* As String)

AfterLabelEdit 事件的语法有如下几个部分：

| 部分               | 描述                                     |
|------------------|--|
| <i>object</i>    | 在“应用于”中指定的对象表达式                        |
| <i>cancel</i>    | 一个整数，确定标签操作是否被取消。任何非 0 整数将取消该操作。也接受布尔值 |
| <i>newstring</i> | 用户输入的字符串；如果用户取消了操作，则是 Null             |

## 说明

只有当 `LabelEdit` 属性设置为 0 或调用了 `StartLabelEdit` 方法时才产生 `AfterLabelEdit` 和 `BeforeLabelEdit` 事件。

当用户编辑完标签点击另一个 `Node` 或 `ListItem` 对象或按下 `ENTER` 键时产生 `AfterLabelEdit` 事件。

要取消标签编辑操作，将 `cancel` 设置为非 0 数或 `True`。如果取消了标签编辑操作，则恢复前一个标签。

在取消标签编辑操作之前，可以使用 `newstring` 参数测试取消条件。例如，下面的代码在 `newstring` 是数字时取消标签编辑操作：

```
Private Sub TreeView1_AfterLabelEdit(Cancel As Integer, NewString As String)
    If IsNumeric(NewString) Then
        MsgBox "No numbers allowed"
        Cancel = True
    End If
End Sub
```

## 请参阅

`Node` 对象，`Nodes` 集合，`BeforeLabelEdit` 事件（`ListView`，`TreeView` 控件），

LabelEdit 属性, NodeClick 事件, StartLabelEdit 方法, ListItem 对象, ListItems 集合。

## 示例

下面的例子给一个 **TreeView** 控件增加了 3 个 **Node** 对象。当你试图编辑 **Node** 对象的标签时, 就检查该对象的索引。如果是 1, 则取消该编辑操作。要使用本例, 在一个窗体上放置一个 **TreeView** 控件, 将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例, 点击顶级的 **Node** 对象的标签两次进行编辑, 输入一些文本, 按下 **ENTER**。

```
Private Sub Form_Load()  
    TreeView1.Style = tvwTreelinesText ' Lines and text.  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(,, "Parent")  
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child1")  
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child2")  
    nodX.EnsureVisible      ' Make sure all nodes are visible.  
End Sub
```

```
Private Sub TreeView1_AfterLabelEdit _  
    (Cancel As Integer, NewString As String)  
    ' If current node's index is 1, edit is canceled.
```

```

If TreeView1.SelectedItem.Index = 1 Then
    Cancel = True
    MsgBox "Can't replace " & TreeView1.SelectedItem.Text & _
        " with " & NewString
End If
End Sub

```

下面的例子给一个 **ListView** 控件增加了 3 个 **ListItem** 对象。当你试图编辑 **ListItem** 对象的标签时，就检查对象的索引。如果是 1，则取消该操作。要使用本例，在一个窗体上放置一个 **ListView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击任一 **ListItem** 对象的标签两次进行编辑，输入一些文本，按下 **ENTER**。

```

Private Sub Form_Load()
    Dim itmX As ListItem
    Set itmX = ListView1.ListItems.Add(, "Item1")
    Set itmX = ListView1.ListItems.Add(, "Item 2")
    Set itmX = ListView1.ListItems.Add(, "Item 3")
End Sub

```

```

Private Sub ListView1_AfterLabelEdit _
(Cancel As Integer, NewString As String)
    ' If current ListItem's index is 1, edit is canceled.

```

```
    If ListView1.SelectedItem.Index = 1 Then
        Cancel = True
        MsgBox "Can't replace " & ListView1.SelectedItem.Text & _
            " with " & NewString
    End If
End Sub
```

## BeforeLabelEdit 事件（ListView，TreeView 控件）

当用户试图编辑当前选择的 **Node** 对象或 **ListItem** 对象的标签时产生该事件。

应用于

**TreeView** 控件，**Listview** 控件。

语法

```
Private Sub object_BeforeLabelEdit (cancel As Integer)
```

BeforeLabelEdit 事件的语法有如下几个部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 在“应用于”中指定的对象表达式                        |
| <i>cancel</i> | 一个整数，确定标签操作是否被取消。任何非 0 整数将取消该操作。也接受布尔值 |

## 说明

只有当 `LabelEdit` 属性设置为 0（Automatic）或调用了 `StartLabelEdit` 方法时才产生 `AfterLabelEdit` 和 `BeforeLabelEdit` 事件。

`BeforeLabelEdit` 事件在标准的 `Click` 事件之后产生。

要开始编辑一个标签，用户必须先点击该对象，然后再点击一次进行编辑。`BeforeLabelEdit` 事件在第二次点击之后产生。

要确定正在编辑哪个对象的标签，使用 `SelectedItem` 属性。下面的例子在允许编辑之前检查所选择的 `Node` 对象的索引。如果索引为 1，则取消标签动作。

```
Private Sub TreeView1_BeforeLabelEdit(Cancel As Integer)
    If TreeView1.SelectedItem.Index = 1 Then
        Cancel = True ' Cancel the operation
    End If
End Sub
```



请参阅

Node 对象, Nodes 集合, AfterLabelEdit 事件 (ListView, TreeView 控件), LabelEdit 属性, SelectedItem 属性 (ActiveX 控件), StartLabelEdit 方法, ListItem 对象, ListItems 集合, Click 事件, Click 事件 (ActiveX 控件)。

示例

下面的例子给一个 TreeView 控件增加了 3 个 Node 对象。当你试图编辑 Node 对象的标签时, 就检查该对象的索引。如果是 1, 则取消该编辑操作。要使用本例, 在一个窗体上放置一个 TreeView 控件, 将下面的代码拷贝到窗体的 Declarations 段中。运行本例, 试着编辑对象的标签。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(., "P1", "parent 1")  
    Set nodX = TreeView1.Nodes.Add( "P1,tvwChild., "Child 1")  
    Set nodX = TreeView1.Nodes.Add("P1",twvChild,"Child 2")  
    nodX.EnsureVisible      ' Make sure all nodes are visible.  
End Sub  
  
Private Sub TreeView1_BeforeLabelEdit (Cancel As Integer)  
'check Selected mode's index,if it is 1,then Cancel the editing operation
```

```

If TreeView1.SelectedItem.Index = 1 Then
    MsgBox "Can't edit " + TreeView1.SelectedItem.Text
    Cancel = True
End If
End Sub

```

下面的例子给一个 **ListView** 控件增加了 3 个 **ListItem** 对象。当你试图编辑 **ListItem** 对象的标签时，就检查对象的索引。如果是 1，则取消该操作。要使用本例，在一个窗体上放置一个 **ListView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，试着编辑标签。

```

Private Sub Form_Load()
    Dim itmX As ListItem
    Set nodX = ListView1.ListItems.Add(,,"Item1")
    Set nodX = ListView1.ListItems.Add(,,"Item 2")
    Set nodX = ListView1.ListItems.Add(,,"Item 3")
End Sub

Private Sub ListView1_BeforeLabelEdit (Cancel As Integer)
    'check Selected item's index.if it is 1,then Cancel the editing operation
    If ListView1.SelectedItem.Index = 1 Then
        Cancel = True
        MsgBox "Can't edit " + ListView1.SelectedItem.Text
    End If
End Sub

```

```
Cancel=true  
End If  
End Sub
```

## Child 属性（Node 对象）

返回 TreeView 控件中一个 Node 对象的第一个子节点。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object*.Child

*object* 是在“应用于”中指定的对象表达式。

## 说明

Child, FirstSibling, LastSibling, Previous, Parent, Next 和 Root 属性返回另一个 Node 对象的索引。因此，你可以同时引用和操作一个 Node 对象，如下：

```
With TreeView1.Nodes(TreeView1.SelectedItem.Index).Child
    .Text = "New text"
    .Key = "New key"
    .SelectedImage = 3
End With
```

也可以设置一个对象变量引用 Node 对象，如下：

```
Dim NodChild As Node
' Get a reference to the child of the selected node.
Set NodChild = TreeView1.Nodes(TreeView1.SelectedItem.Index).Child
' Use this reference to perform operations on the child Node.
With nodChild
    .Text = "New text" ' Change the text.
    .Key = "New key" ' Change key.
    .SelectedImage = 3 ' Change SelectedImage.
End with
```

请参阅

**Children** 属性，**FirstSibling** 属性，**LastSibling** 属性，**Next** 属性，**Parent** 属性（Node 对象），**Previous** 属性（Node 对象），**Root** 属性（Node 对象）。

## 示例

下面的例子创建了几个 **Node** 对象。当点击一个 **Node** 对象时，代码首先使用 **Children** 属性确定是否是子节点。如果是，则窗体的标题显示该 **Child** 节点的文本。

Option Explicit

Private Sub Form\_Load()

' This code creates a tree with 3 Node objects.

TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.

TreeView1.LineStyle = tvwRootLines ' Linestyle 1.

' Add several Node objects.

Dim nodX As Node ' Create variable.

Set nodX = TreeView1.Nodes.Add(, "r", "Root")

Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c1", "Child 1")

```
nodX.EnsureVisible ' Show all nodes.
```

```
Set nodX = TreeView1.Nodes.Add("c1", tvwChild, "c2", "Child 2")
```

```
Set nodX = TreeView1.Nodes.Add("c1", tvwChild, "c3", "Child 3")
```

```
nodX.EnsureVisible ' Show all nodes.
```

```
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
```

```
    ' If the Node does have children, then display the text of
```

```
    ' the child Node.
```

```
    If Node.Children Then
```

```
        Caption = Node.Child.Text
```

```
    End If
```

```
End Sub
```

## Children 属性

返回 Node 对象中子节点的数量。

应用于

TreeView 控件，Node 对象，Nodes 集合。

## 语法

*object.Children*

*object* 是在“应用于”中指定的对象表达式。

## 说明

在执行影响子节点的操作之前使用 **Children** 属性确定 **Node** 对象是否有子节点。例如，下面的代码在使用 **Child** 属性检取第一个 **Node** 对象的 **Text** 属性之前检查其是否有子节点：

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
    If Node.Children > 0 Then
        MsgBox Node.Child.Text
    End If
End Sub
```

## 请参阅

**Child** 属性（**Node** 对象），**Text** 属性。

## 示例

下面的例子在一个 **TreeView** 控件中放置了几个 **Node** 对象。代码检查 **Node** 是否有子节点。如果有，则显示子节点的文本。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击 **Node** 对象，然后点击窗体查看 **Node** 对象子节点的文本。

```
Option Explicit
```

```
Private Sub Form_Click()
```

```
    Dim strC As String
```

```
    Dim N As Integer
```

```
    If TreeView1.SelectedItem.Children > 0 Then ' There are children.
```

```
        ' Get first child's text, and set N to its index value.
```

```
        strC = TreeView1.SelectedItem.Child.Text & vbCrLf
```

```
        N = TreeView1.SelectedItem.Child.Index
```

```
        ' While N is not the index of the child node's
```

```
        ' last sibling, get next sibling's text.
```

```
        While N <> TreeView1.SelectedItem.Child.LastSibling.Index
```

```
            strC = strC & TreeView1.Nodes(N).Next.Text & vbCrLf
```

```
            ' Reset N to next sibling's index.
```

```
            N = TreeView1.Nodes(N).Next.Index
```



```
Wend
' Show results.
MsgBox "Children of " & TreeView1.SelectedItem.Text & _
" are: " & vbCrLf & strC
Else ' There are no children.
MsgBox TreeView1.SelectedItem.Text & " has no children"
End If
End Sub
```

```
Private Sub Form_Load()
```

```
TreeView1.BorderStyle = 1 ' Ensure border is visible
Dim nodX As Node
Set nodX = TreeView1.Nodes.Add(,"d","Dates")
Set nodX = TreeView1.Nodes.Add("d",tvwChild,"d89","1989")
Set nodX = TreeView1.Nodes.Add("d",tvwChild,"d90","1990")

' Create children of 1989 node.
Set nodX = TreeView1.Nodes.Add("d89",tvwChild,,"John")
Set nodX = TreeView1.Nodes.Add("d89",tvwChild,,"Brent")
Set nodX = TreeView1.Nodes.Add("d89",tvwChild,,"Eric")
Set nodX = TreeView1.Nodes.Add("d89",tvwChild,,"Ian")
nodX.EnsureVisible ' Show all nodes.
```

```
' Create children of 1990 node.  
Set nodX = TreeView1.Nodes.Add("d90",tvwChild,,"Randy")  
Set nodX = TreeView1.Nodes.Add("d90",tvwChild,,"Ron")  
nodX.EnsureVisible ' Show all nodes.  
End Sub
```

## Collapse 事件（TreeView 控件）

当 TreeView 控件中的 Node 对象收缩时产生该事件。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

```
Private Sub object_Collapse (ByVal node As Node)
```

Collapse 事件的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |
| <i>node</i>   | 所点击的 Node 对象的引用 |

## 说明

Collapse 事件在标准的 Click 事件之前产生。

有 3 种方法收缩一个 Node 对象: 将 Node 对象的 Expanded 属性设置为 False, 双击 Node 对象或者当 TreeView 控件的 Style 属性设置为包含加减号风格时点击加/减符号。所有这些方法都产生 Collapse 事件。

该事件传递了收缩的 Node 对象引用。该引用可用于验证动作, 如下所示:

```
Private Sub TreeView1_Collapse(ByVal Node As Node)
    If Node.Index = 1 Then
        Node.Expanded = True ' Expand the node again.
    End If
End Sub
```

## 请参阅

Expand 事件 (TreeView 控件), Expanded 属性, NodeClick 事件, SelectedItem

属性（ActiveX 控件），Style 属性（TreeView 控件）。

## 示例

下面的例子给 TreeView 控件增加了一个有几个子节点的 Node 对象。当用户收缩 Node 对象时，代码检查其有几个子节点。如果多于一个，则 Node 重新打开。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，双击 Node 节点收缩之，以产生该事件。

```
Private Sub Form_Load()  
    TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(., "DV", "Da Vinci")  
    Set nodX = TreeView1.Nodes.Add("DV", tvwChild, "T", "Titian")  
    Set nodX = TreeView1.Nodes.Add("T", tvwChild, "R", "Rembrandt")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "Goya")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "David")  
    nodX.EnsureVisible ' Show all nodes.  
End Sub
```

```
Private Sub TreeView1_Collapse(ByVal Node As Node)  
    ' If the Node has more than one child node,  
    ' keep the node expanded.
```

```
Select Case Node.Children
    Case Is > 1
        Node.Expanded = True
End Select
End Sub
```

## ColumnHeaderIcons 属性

返回或者设置 **Imagelist** 控件，该控件为 **ColumnHeaders** 集合提供图标。

应用于

**ListView** 控件。

语法

*object*.ColumnHeaderIcons [= *imagelist*]

**ColumnHeaderIcons** 的语法包括下列部分：

| 部分               | 描述                                       |
|------------------|--|
| <i>object</i>    | 一个对象表达式，其值等于“应用于”列表中的一个对象                |
| <i>imagelist</i> | 一个 ImageList 控件，或者指向 Imagelist 控件的一个对象引用 |

## 说明

要为 ColumnHeader 对象设置图标，请将它的 Icon 属性设置成为一个索引值、关键字或者对象引用，以指向 ColumnHeaderIcons 属性指定的 ImageList 控件中的一个 ListImage 对象。

## 请参阅

Icon 属性（Windows 常见控件）。

## CreateDragImage 方法

使用对象关联的图像产生一个抖动的拖动图像。通常在拖放操作中使用该图像。

应用于

Node 对象，Nodes 集合，ListItems 对象，ListItems 集合。

语法

*object*.CreateDragImage

*object* 是在“应用于”中指定的对象表达式。

说明

通常在启动拖放操作时使用 CreateDragImage 方法给 DragIcon 属性分配一个图像。

请参阅

DropHighlight 属性（ListView，TreeView 控件），HitTest 方法（ListView，TreeView 控件），DragIcon 属性。

示例

下面的例子给 TreeView 控件增加了有几个子节点的 Node 对象。在选择了

一个 Node 后，可以将其拖放到另一个 Node 对象上。要使用本例，在一个窗体上放置一个 TreeView 控件和一个 ImageList 控件，将下面的代码粘贴到窗体的 Declarations 段中。运行本例，拖放 Node 对象，查看效果。

```
' Declare global variables.
```

```
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
```

```
Dim nodX As Object ' Item that is being dragged.
```

```
Private Sub Form_Load()
```

```
    ' Load a bitmap into an Imagelist control.
```

```
    Dim imgX As ListImage
```

```
    Dim BitmapPath As String
```

```
    BitmapPath = "icons\mail\mail01a.ico"
```

```
    Set imgX = imagelist1.ListImages.Add(, , LoadPicture(BitmapPath))
```

```
    ' Initialize TreeView control and create several nodes.
```

```
    TreeView1.ImageList = imagelist1
```

```
    Dim nodX As Node ' Create a tree.
```

```
    Set nodX = TreeView1.Nodes.Add(, , "Parent1", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(, , "Parent2", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 1", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 2", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(2, tvwChild, "Child 3", 1)
```



```
Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
```

```
Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
```

```
nodX.EnsureVisible ' Expand tree to show all nodes.
```

```
End Sub
```

```
Private Sub TreeView1_MouseDown_
```

```
(Button As Integer, Shift As Integer, x As Single, y As Single)
```

```
Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
```

```
End Sub
```

```
Private Sub TreeView1_MouseMove _
```

```
(Button As Integer, Shift As Integer, x As Single, y As Single)
```

```
If Button = vbLeftButton Then ' Signal a Drag operation.
```

```
indrag = True ' Set the flag to true.
```

```
' Set the drag icon with the CreateDragImage method.
```

```
TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
```

```
TreeView1.Drag vbBeginDrag ' Drag operation.
```

```
End If
```

```
End Sub
```

```
Private Sub TreeView1_DragDrop_
```

```
(Source As Control, x As Single, y As Single)
```

```
If TreeView1.DropHighlight Is Nothing Then
```

```
Set TreeView1.DropHighlight = Nothing
```

```
indrag = False
```

```
Exit Sub
```

```
Else
```

```
If nodX = TreeView1.DropHighlight Then Exit Sub
```

```
Cls
```

```
Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
```

```
Set TreeView1.DropHighlight = Nothing
```

```
indrag = False
```

```
End If
```

```
End Sub
```

```
Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, State As Integer)
```

```
If indrag = True Then
```

```
' Set DropHighlight to the mouse's coordinates.
```

```
Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
```

```
End If
```

```
End Sub
```

# DropHighlight 属性（ListView，TreeView 控件）

返回或设置一个 Node 对象或 ListItem 对象的引用。该对象在鼠标移到其上使用系统加亮颜色加亮。

应用于

ListView 控件，TreeView 控件。

语法

*object*.DropHighlight [=value]

DropHighlight 属性的语法有如下几个部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式       |
| <i>value</i>  | 一个 Node 或 ListItem 对象 |

说明

通常在拖放操作中与 HitTest 方法一起使用 DropHighlight 属性。当鼠标拖

动到 `ListItem` 或 `Node` 对象之上时，`HitTest` 方法返回该对象的引用。接着，`DropHighlight` 属性设置为该命中对象，立即返回该对象的引用。`DropHighlight` 属性然后使用系统加亮颜色加亮显示该对象。下面的方法将 `DropHighlight` 属性设置为 `HitTest` 方法命中的对象：

```
Private Sub TreeView1_DragOver _  
(Source As Control, X As Single, Y As Single, State As Integer)  
    Set TreeView1.DropHighlight = TreeView1.HitTest(X,Y)  
End Sub
```

接着，你可以在 `DragDrop` 事件中使用 `DropHighlight` 属性返回源控件拖放的上一个对象的引用，如下所示：

```
Private Sub TreeView1_DragDrop _  
(Source As Control, x As Single, y As Single)  
    ' DropHighlight returns a reference to object drop occurred over.  
    Me.Caption = TreeView1.DropHighlight.Text  
    ' To release the DropHighlight reference, set it to Nothing.  
    Set TreeView1.DropHighlight = Nothing  
End Sub
```

注意，在上面的例子中，该工程完成后，将 `DropHighlight` 属性设置为 `Nothing`。要释放加亮效果必须这样做。

请参阅

Node 对象，Nodes 集合，HitTest 方法（ListView, TreeView 控件），ListItem 对象，ListItems 集合，DragDrop 事件。

## 示例

下面的例子给 TreeView 控件增加了有几个子节点的 Node 对象。在选择了一个 Node 后，可以将其拖放到另一个 Node 对象上。要使用本例，在一个窗体上放置一个 TreeView 控件和一个 ImageList 控件，将下面的代码粘贴到窗体的 Declarations 段中。运行本例，拖放 Node 对象，查看效果。

```
' Declare global variables.  
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.  
Dim nodX As Object ' Item that is being dragged.  
  
Private Sub Form_Load()  
    ' Load a bitmap into an Imagelist control.  
    Dim imgX As ListImage  
    Dim BitmapPath As String  
    BitmapPath = "icons\mail\mail01a.ico"  
    Set imgX = imagelist1.ListImages.Add(, LoadPicture(BitmapPath))
```

' Initialize TreeView control and create several nodes.

TreeView1.ImageList = imagelist1

Dim nodX As Node' Create a tree.

Set nodX = TreeView1.Nodes.Add(, , "Parent1", 1)

Set nodX = TreeView1.Nodes.Add(, , "Parent2", 1)

Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 1", 1)

Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 2", 1)

Set nodX = TreeView1.Nodes.Add(2, tvwChild, "Child 3", 1)

Set nodX = TreeView1.Nodes.Add(2, tvwChild, "Child 4", 1)

Set nodX = TreeView1.Nodes.Add(3, tvwChild, "Child 5", 1)

nodX.EnsureVisible ' Expand tree to show all nodes.

End Sub

Private Sub TreeView1\_MouseDown\_

(Button As Integer, Shift As Integer, x As Single, y As Single)

Set nodX = TreeView1.SelectedItem ' Set the item being dragged.

End Sub

Private Sub TreeView1\_MouseMove \_

(Button As Integer, Shift As Integer, x As Single, y As Single)

If Button = vbLeftButton Then ' Signal a Drag operation.

indrag = True ' Set the flag to true.

' Set the drag icon with the CreateDragImage method.

```
TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
```

```
TreeView1.Drag vbBeginDrag      ' Drag operation.
```

```
End If
```

```
End Sub
```

```
Private Sub TreeView1_DragDrop_
```

```
(Source As Control, x As Single, y As Single)
```

```
    If TreeView1.DropHighlight Is Nothing Then
```

```
        Set TreeView1.DropHighlight = Nothing
```

```
        indrag = False
```

```
        Exit Sub
```

```
    Else
```

```
        If nodX = TreeView1.DropHighlight Then Exit Sub
```

```
        Cls
```

```
        Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
```

```
        Set TreeView1.DropHighlight = Nothing
```

```
        indrag = False
```

```
    End If
```

```
End Sub
```

```
Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, State As
```

```
Integer)
```

```
    If indrag = True Then
```

```
' Set DropHighlight to the mouse's coordinates.  
Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)  
End If  
End Sub
```

## EnsureVisible 方法

确保指定的 **ListItem** 或 **Node** 对象是可视的。如果必要的话,该方法展开 **Node** 对象并滚动 **TreeView** 控件窗口; 对于 **ListView** 控件, 只滚动。

### 应用于

**View** 控件, **ListItem** 对象, **ListItems** 集合, **TreeView** 控件, **Node** 对象, **Nodes** 集合。

### 语法

*object*.EnsureVisible

*object* 是在“应用于”中指定的对象表达式。



## 返回值

| 值     | 描述  |
|-------|---|
| True  | 如果 ListView 或 TreeView 控件必须滚动窗口才能显示对象时返回 True |
| False | 如果不必滚动窗口，则返回 False                            |

## 说明

如果你想让一个 TreeView 控件或 ListView 控件中隐藏较深的对象显示出来，可以使用 EnsureVisible 方法确保其可视。

## 请参阅

Node 对象，Nodes 集合，Expand 事件（TreeView 控件），Expanded 属性，ListItem 对象，ListItems 集合。

## 示例

下面的例子给 TreeView 控件增加了许多节点，使用 EnsureVisible 方法滚动并展开节点树。要使用本例，在一个窗体上放置一个 TreeView 控件，并将下面的代码拷贝到窗体的 Declarations 段中。运行本例，点击窗体查看节点树的扩展。

```

Private Sub Form_Load()
    Dim nodX As Node
    Dim i as Integer
    TreeView1.BorderStyle = FixedSingle          ' Show borders.

    Set nodX = TreeView1.Nodes.Add(,,,"Root")    ' Add first node.
    For i = 1 to 15 ' Add 15 nodes
        Set nodX = TreeView1.Nodes.Add(i,,,"Node " & CStr(i))
    Next i

    Set nodX = TreeView1.Nodes.Add(,,,"Bottom")  ' Add one with text.
    Set nodX = TreeView1.Nodes.Add(i,,,"Expanded") ' Add child to node.
    Set nodX = TreeView1.Nodes.Add(i+1,,,"Show me") ' Add a final child.
End Sub

```

```

Private Sub Form_Click()
    ' Tree will scroll and expand when you click the form.
    TreeView1.Nodes(TreeView1.Nodes.Count).EnsureVisible
End Sub

```

# Expand 事件（TreeView 控件）

当 TreeView 控件中展开 Node 对象即子节点可视时产生该事件。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

Private Sub *object*\_Expand (ByVal *node* As Node)

Expand 事件的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |
| <i>node</i>   | 展开的 Node 对象的引用  |

说明

在 Click 和 DbClick 事件之后产生 Expand 事件。

有 3 种方法产生 Expand 事件用户事件有子节点的 Node 对象；当 Expanded

属性设置为 **True**；当点击有加减号的图像时。使用 **Expand** 事件验证对象，如下所示：

```
Private Sub TreeView1_Expand(ByVal Node As Node)
    If Node.Index <> 1 Then
        Node.Expanded = False ' Prevent expand.
    End If
End Sub
```

请参阅

**Expanded** 属性，**Click** 事件，**DbClick** 事件，**Click** 事件（ActiveX 控件），**DbClick** 事件（ActiveX 控件）。

示例

下面的例子给 **TreeView** 控件增加了一个 **Node** 对象。当展开一个 **Node** 对象时，产生 **Expand** 事件，显示与该对象有关的信息。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，扩展节点。

```
Private Sub Form_Load()
    Dim nodX As Node
```

```

Set nodX = TreeView1.Nodes.Add(, "RP", "Root Parent")
Set nodX = TreeView1.Nodes.Add("RP", tvwChild, "C1", "Child1")
Set nodX = TreeView1.Nodes.Add("C1", tvwChild, "C2", "Child2")
Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C3", "Child3")
Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C4", "Child4")
TreeView1.Style = tvwTreelinesPlusMinusText ' Style 6.
TreeView1.LineStyle = tvwRootLines ' Style 1
End Sub

```

```

Private Sub TreeView1_Expand(ByVal Node As Node)
    Select Case Node.Key Like "C*"
        Case Is = True
            MsgBox Node.Text & " is a child node."
    End Select
End Sub

```

## Expanded 属性

返回或设置一个值，确定 **TreeView** 控件中的一个 **Node** 对象是展开的还是收缩的。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object.Expanded* [=boolean]

Expanded 属性的语法有如下几个部分：

| 部分             | 描述                    |
|----------------|-----------------------|
| <i>object</i>  | 在“应用于”中指定的对象表达式       |
| <i>boolean</i> | 一个布尔表达式，指定节点是展开的还是收缩的 |

*boolean* 值的设置如下：

| 设置    | 描述          |
|-------|-------------|
| True  | Node 当前是展开的 |
| False | Node 当前是收缩的 |

说明

在程序中，可以使用 Expanded 属性展开一个 Node 对象。下面的代码在双

击第一个 Node 对象时展开它：

```
TreeView1.Nodes(1).Expanded = True
```

当一个 Node 对象展开时，产生 **Expand** 事件。

如果一个 Node 对象没有子节点，则忽略该属性。

请参阅

**EnsureVisible** 方法，**Expand** 事件（**TreeView** 控件）。

示例

下面的例子给一个 **TreeView** 控件增加了几个 **Node** 对象。当点击窗体时，每个 **Node** 对象的 **Expanded** 属性都设置为 **True**。要使用本例， 在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击窗体展开每个 **Node** 对象。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Dim i as Integer  
    TreeView1.BorderStyle = vbFixedSingle ' Show border.  
  
    ' Create a root node.
```

```

Set nodX = TreeView1.Nodes.Add(, "root", "Root")

For i = 1 to 5 ' Add 5 child nodes.
    Set nodX = TreeView1.Nodes.Add(i, tvwChild, "Node " & CStr(i))
Next i
End Sub

Private Sub Form_Click()
    Dim I as Integer
    For I = 1 to TreeView1.Nodes.Count
        ' Expand all nodes.
        TreeView1.Nodes(i).Expanded = True
    Next I
End Sub

```

## ExpandedImage 属性

返回或设置与 **ImageList** 控件关联的 **ListImage** 对象的索引或键值；当 **Node** 对象展开时显示该 **ListImage** 对象。



应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object.ExpandedImage [=number]*

ExpandedImage 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式               |
| <i>number</i> | 一个数字表达式或字符串表达式，分别对应于图像的索引值和键值 |

说明

当用户双击节点或 Node 对象的 Expanded 属性设置为 True 时，该属性允许你改变与 Node 对象关联的图像。

请参阅

Expand 事件(TreeView 控件)，Expanded 属性，SelectedImage 属性，ImageList

控件，ListImage 对象，ListImages 集合， Image 属性（ActiveX 控件），ImageList 属性（ActiveX 控件）。

## FirstSibling 属性

返回 TreeView 控件中 Node 对象的第一个兄弟对象。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object*.FirstSibling

*object* 是在“应用于”中指定的对象表达式。

说明

Node 对象的第一个兄弟对象显示在一个体系级中的第一个位置。哪个对象显示在第一个位置取决于对象保存在哪个体系级，其由 Sorted 属性决定。

Child, FirstSibling, LastSibling, Previous, Parent, Next 和 Root 属性返回另一个 Node 对象的引用。因此，可以同时引用和操作一个 Node 对象，如下：

```
With TreeView1.Nodes(x).FirstSibling
.Text = "New text"
.Key = "New key"
.SelectedImage = 3
End With
```

也可以设置一个对象本例引用 Node 对象，如下：

```
Dim NodFirstSib As Node
' Get a reference to the first sibling of Node x.
Set NodFirstSib = TreeView1.Nodes(x).FirstSibling
' Use this reference to perform operations on the first sibling Node.
With nodFirstSib
    .Text = "New text" ' Change the text.
    .Key = "New key" ' Change key.
    .SelectedImage = 3 ' Change SelectedImage.
End With
```

请参阅

Child 属性 (TreeView 控件), LastSibling 属性, Next 属性, Parent 属性 (Node 对象), Previous 属性 (Node 对象), Root 属性 (Node 对象), Sorted 属性 (Node 对象)。

示例

下面的例子给 TreeView 控件增加了一个 Node 对象。使用 FirstSibling, Next 和 LastSibling 属性浏览 Node 对象的等级结构。要使用本例, 在一个窗体上放置一个 TreeView 控件, 将下面的代码拷贝到窗体的 Declarations 段中。运行本例, 点击各节点查看返回值。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(., "dad", "Mike") ' A first sibling.  
    Set nodX = TreeView1.Nodes.Add(., "mom", "Carol")  
    Set nodX = TreeView1.Nodes.Add(., "Alice")  
  
    ' Marsha is the FirstSibling.  
    Set nodX = TreeView1.Nodes.Add("mom", tvwChild, "Marsha")  
    Set nodX = TreeView1.Nodes.Add("mom", tvwChild, "Jan")  
    Set nodX = TreeView1.Nodes.Add("mom", tvwChild, "Cindy")
```

```
nodX.EnsureVisible ' Show all nodes.
```

```
' Greg is the FirstSibling.
```

```
Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Greg")
```

```
Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Peter")
```

```
Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Bobby")
```

```
nodX.EnsureVisible ' Show all nodes.
```

```
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
```

```
    Dim strText As String
```

```
    Dim n As Integer
```

```
    ' Set n to FirstSibling's index.
```

```
    n = Node.FirstSibling.Index
```

```
    ' Place FirstSibling's text & linefeed in string variable.
```

```
    strText = Node.FirstSibling.Text & vbLF
```

```
    While n <> Node.LastSibling.Index
```

```
        ' While n is not the index of the last sibling, go to the
```

```
        ' next sibling and place its text into the string variable.
```

```
            strText = strText & TreeView1.Nodes(n).Next.Text & vbLF
```

```
        ' Set n to the next node's index.
```

```
            n = TreeView1.Nodes(n).Next.Index
```

```
    Wend
```

```
MsgBox strText    ' Display results.  
End Sub
```

## FullPath 属性

返回 **TreeView** 控件中所引用 **Node** 对象的全路径。当为该属性分配一个字符串变量时，字符串就设置为该对象的 **FullPath**。

应用于

**Node** 对象，**Nodes** 集合。

语法

*object*.FullPath

*object* 是在“应用于”中指定的对象表达式。

说明

全路径是将所引用 **Node** 对象 **Text** 属性中的文本与其所有祖先节点的 **Text**

属性值连接起来的结果。**PathSeparator** 属性决定了分隔符。

请参阅

**Node** 对象，**Nodes** 集合，**PathSeparator** 属性（**TreeView** 控件），**Text** 属性。

示例

下面的例子给 **TreeView** 控件增加了几个 **Node** 对象，并显示所选择的对象的全路径。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，选择一个节点，然后点击窗体查看该节点的全路径。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(,, "Root")  
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Dir1")  
    Set nodX = TreeView1.Nodes.Add(2, tvwChild, "Dir2")  
    Set nodX = TreeView1.Nodes.Add(3, tvwChild, "Dir3")  
    Set nodX = TreeView1.Nodes.Add(4, tvwChild, "Dir4")  
    nodX.EnsureVisible ' Show all nodes.  
    TreeView1.Style = tvwTreelinesText ' Style 4.  
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
    MsgBox Node.FullPath
End Sub
```

## GetVisibleCount 方法

返回 **TreeView** 控件中可视区域内 **Node** 对象的数量。

应用于

**TreeView** 控件，**Node** 对象，**Nodes** 集合。

语法

*object*.GetVisibleCount

*object* 是在“应用于”中指定的对象表达式。

说明

**Node** 对象的数量有窗口中可以显示的行数决定。总行数由控件的高度和



Font 对象的 Size 属性决定。该数值包括列表中部分可视的对象。

你可以使用 **GetVisibleCount** 属性确定可视行的最小数量，以便用户可以正确地浏览整个等级体系结构。如果最小行数不可视，可以使用 **Height** 属性重新设置 **TreeView** 控件的大小。

如果某个 **Node** 对象必须可视，使用 **EnsureVisible** 方法滚动并展开 **TreeView** 控件。

请参阅

**EnsureVisible** 方法，**Font** 对象，**Height**, **Width** 属性，**Size** 属性（**Font**）。

示例

下面的例子给一个 **TreeView** 控件增加了一个 **Node** 对象。当你点击窗体时，代码使用 **GetVisibleCount** 方法检查有多少行可视，然后增大控件以显示所有的对象。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击窗体以增大控件。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Dim i as Integer  
    TreeView1.BorderStyle = 1 ' Show border.
```

```
For i = 1 to 20
```

```
    Set nodX = TreeView1.Nodes.Add(,, "Node " & CStr(i))
```

```
Next I
```

```
TreeView1.Height = 1500 ' TreeView is short, for comparison's sake.
```

```
End Sub
```

```
Private Sub Form_Click()
```

```
    While Treeview1.GetVisibleCount < 20
```

```
        ' Make the treeview larger.
```

```
        TreeView1.Height = TreeView1.Height + TreeView1.Font.Size
```

```
    Wend
```

```
End Sub
```

## HitTest 方法（ListView，TreeView 控件）

返回 x 和 y 坐标处 ListItem 对象或 Node 对象的引用。常在拖放操作中使用该方法确定指定的位置是否有所要求的拖放目的对象。

应用于

ListView 控件, ListItem 对象, ListItemsih, TreeView 控件, Node 对象, Nodes

集合。

语法

*object*.HitTest ( *x* As Single, *y* As Single)

HitTest 方法的语法有如下几个部分：

| 部分                  | 描述                                      |
|---------------------|---|
| <i>object</i>       | 在“应用于”中指定的对象表达式                         |
| <i>x</i> , <i>y</i> | 目的对象的坐标，要么是一个 Node 对象，要么是一个 ListItem 对象 |

说明

如果指定的坐标处没有对象，则 HitTest 返回 Nothing。

HitTest 方法常和 DropHighlight 属性一起使用，在鼠标移到某个对象时将该对象加亮显示。DropHighlight 属性需要加亮对象的引用。为了确定该对象，使用 HitTest 方法和某个事件如 DragOver 事件返回对象的 x 和 y 坐标，如下所示：

```
Private Sub TreeView1_DragOver _  
(Source As Control, X As Single, Y As Single, State As Integer)  
    Set TreeView1.DropHighlight = TreeView1.HitTest(X,Y)  
End Sub
```

然后，在 DragDrop 事件中使用 DropHighlight 属性返回源控件拖放的上一个对象的引用，如下所示：

```
Private Sub TreeView1_DragDrop _  
(Source As Control, x As Single, y As Single)  
    ' DropHighlight returns a reference to object drop occurred over.  
    Me.Caption = TreeView1.DropHighlight.Text  
    ' To release the DropHighlight reference, set it to Nothing.  
    Set TreeView1.DropHighlight = Nothing  
End Sub
```

注意，上面的例子中，在该过程的末尾将 DropHighlight 属性设置为 Nothing。要释放加亮效果，必须如此。

请参阅

DropHighlight 属性（ListView，TreeView 控件），DragDrop 事件，DragOver 事件。

示例

下面的例子给一个 TreeView 控件增加了一个 Node 对象。当你选择了一个 Node 对象后，可以将其拖放到另一个 Node 对象中。要使用本例，在一个窗体

上放置一个 **TreeView** 控件和一个 **ImageList** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，任意拖放 **Node** 对象，查看结果。

```
' Declare global variables.
```

```
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
```

```
Dim nodX As Object ' Item that is being dragged.
```

```
Private Sub Form_Load()
```

```
    ' Load a bitmap into an Imagelist control.
```

```
    Dim imgX As ListImage
```

```
    Dim BitmapPath As String
```

```
    BitmapPath = "icons\mail\mail01a.ico"
```

```
    Set imgX = imagelist1.ListImages.Add(, LoadPicture(BitmapPath))
```

```
    ' Initialize TreeView control and create several nodes.
```

```
    TreeView1.ImageList = imagelist1
```

```
    Dim nodX As Node ' Create a tree.
```

```
    Set nodX = TreeView1.Nodes.Add(, , "Parent1", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(, , "Parent2", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 1", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 2", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(2, tvwChild, "Child 3", 1)
```

```
    Set nodX = TreeView1.Nodes.Add(2, tvwChild, "Child 4", 1)
```

```
Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
nodX.EnsureVisible ' Expand tree to show all nodes.
```

```
End Sub
```

```
Private Sub TreeView1_MouseDown_
```

```
(Button As Integer, Shift As Integer, x As Single, y As Single)
```

```
Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
```

```
End Sub
```

```
Private Sub TreeView1_MouseMove _
```

```
(Button As Integer, Shift As Integer, x As Single, y As Single)
```

```
If Button = vbLeftButton Then ' Signal a Drag operation.
```

```
indrag = True ' Set the flag to true.
```

```
' Set the drag icon with the CreateDragImage method.
```

```
TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
```

```
TreeView1.Drag vbBeginDrag ' Drag operation.
```

```
End If
```

```
End Sub
```

```
Private Sub TreeView1_DragDrop_
```

```
(Source As Control, x As Single, y As Single)
```

```
If TreeView1.DropHighlight Is Nothing Then
```

```
Set TreeView1.DropHighlight = Nothing
```

```
        indrag = False
    Exit Sub
Else
    If nodX = TreeView1.DropHighlight Then Exit Sub
    Cls
    Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
    Set TreeView1.DropHighlight = Nothing
    indrag = False
End If
End Sub
```

```
Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, State As Integer)
    If indrag = True Then
        ' Set DropHighlight to the mouse's coordinates.
        Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
    End If
End Sub
```

# Icon 属性（Windows 通用控件）

返回或者设置被对象显示的图标。

应用于

ColumnHeader 对象，ColumnHeaders 集合。

语法

*object*.Icon [= *icon*]

Icon 属性的语法包括以下几个部分：

| 部分            | 描述                               |
|---------------|----------------------------------|
| <i>object</i> | 一个对象表达式，其值等于“应用于”列表中的一个对象        |
| <i>icon</i>   | 一个 ListImage 对象的 Key 或者 Index 属性 |

说明

要设置一个对象的图标，ImageList 控件中必须被填入若干 ListImage 对象（每个代表了一个图像），而且必须为 ImageList 控件设置适当的属性。然后



可以将 **Icon** 属性设置为 **ListImage** 对象的 **Key** 或 **Index** 属性。

请参阅

**ColumnHeaderIcons** 属性。

示例

下列的例子将 **ColumnHeaderIcons** 属性置设为 **ImageList** 控件，然后将 **Icon** 属性设置为 **ListImage** 对象的 **Key**。

Option Explicit

Private Sub Form\_Load()

'Assumes an ImageList control populated with at least one image.

Dim c As ColumnHeader

Dim i As Integer

For i = To 4 'create four ColumnHeader Objects

Listview1.ColumnHeaders.Add,, "Col" & i

Next I

Listview1.View = lvwReport

ImageList1.ListImages(1).key= "key1" ' Setkey Property of ListImage,

Listview1.ColumnHeaderIcons = ImageList 1

For Each c In Listview1.columnHeaders

```
c.Icon = "key1"
```

```
Next
```

```
End Sub
```

# Indentation 属性

返回或设置 **TreeView** 控件缩排的宽度。每个新增加的子节点使用该宽度缩排。

应用于

**TreeView** 控件， **ImageCombo** 控件。

语法

*object*.Indentation [= *number*]

Indentation 属性的语法有如下几个部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 一个 <b>TreeView</b> 控件的对象表达式 |
| <i>number</i> | 一个整数，指定子节点缩排的宽度             |

## 说明

如果你在运行时改变了 **Indentation** 属性值，则重新绘制 **TreeView** 控件以反映新的宽度。该属性值不能是负数。

## 请参阅

**Node** 对象，**Nodes** 集合。

## 示例

下面的例子给一个 **TreeView** 控件增加了几个 **Node** 对象，**Indentation** 属性值显示在窗体的标题栏中。一个 **OptionButton** 控件矩阵提供了可选的 **Indentation** 宽度。要使用本例，在一个窗体上放置一个 **TreeView** 控件和 3 个 **OptionButton** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击 **OptionButton** 改变 **Indentation** 属性值。

```
Private Sub Form_Load()  
    ' Label OptionButton controls with Indentation choices.  
    Option1(0).Caption = "250"  
    Option1(1).Caption = "500"  
    Option1(2).Caption = "1000"
```

' Select the first option, and set the indentation to 250 initially

Option1(0).Value = True

TreeView1.Indentation = 250

Dim nodX As Node

Dim i As Integer

Set nodX = TreeView1.Nodes.Add(,,CStr(1)) ' Add first node.

For i = 1 To 6 ' Add 6 nodes.

Set nodX = TreeView1.Nodes.Add(i,tvwChild,,CStr(i + 1))

Next i

nodX.EnsureVisible ' Makes sure all nodes are visible.

Form1.Caption = "Indentation = " & TreeView1.Indentation

End Sub

Private Sub Option1\_Click(Index as Integer)' Change Indentation with OptionButton value.

TreeView1.Indentation = Val(Option1(Index).Caption)

Form1.Caption = "Indentation = " & TreeView1.Indentation

End Sub

# LabelEdit 属性

返回或设置一个值，确定用户是否可以编辑 TreeView 控件或 ListView 控件中的 Node 对象或 ListItem 对象。

应用于

TreeView 控件， ListView 控件。

语法

*object*.LabelEdit [=integer]

LabelEdit 属性的语法有如下几个部分：

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 在“应用于”中指定的对象表达式                             |
| <i>integer</i> | 一个整数，指定用户是否可以编辑 Node 或 ListItem 对象，如“设置”中所示 |

# 设置

*integer* 值的设置如下:

| 常量                     | 值 | 描述                            |
|------------------------|---|-------------------------------|
| ListView: lvwAutomatic | 0 | (缺省) 自动。当用户点击选择对象的标签时产生       |
| TreeView: tvwAutomatic |   | BeforeLabelEdit 事件            |
| ListView: lvwManual    | 1 | 手工。只有激活 StartLabelEdit 方法时才产生 |
| TreeView: tvwManual    |   | BeforeLabelEdit 事件            |

# 说明

当点击选择的对象时 (如果 LabelEdit 属性设置为 Automatic), 则初始化对象的标签编辑。即第一次点击对象选中它; 第二次点击对象则初始化标签编辑操作。

与 StartLabelEdit 方法一起, LabelEdit 属性允许你在程序中确定何时和哪些标签可以编辑。当 LabelEdit 属性设置为 1 时, 不能编辑标签, 除非调用了 StartLabelEdit 方法。例如, 下面的代码允许用户点击一个命令按钮编辑 Node 对象的标签。

```
Private Sub Command1_Click()  
    ' Determine if the right Node is selected.  
    If TreeView1.SelectedItem.Index = 1 Then  
        TreeView1.StartLabelEdit ' Let user begin editing.  
    End If  
End Sub
```

## 请参阅

TreeView 控件常量, Node 对象, Nodes 集合, AfterLabelEdit 事件 (ListView 控件, TreeView 控件), BeforeLabelEdit 事件 (ListView 控件, TreeView 控件), StartLabelEdit 方法, ListItem 对象, ListItems 集合, ListView 控件常量。

## 示例

下面的例子在点击命令按钮时初始化标签编辑操作。这允许编辑 Node 对象的标签, 除非该对象是一个根节点对象。LabelEdit 属性必须设置为 Manual。要使用本例, 在一个窗体上放置一个 TreeView 控件和一个 CommandButton 控件。将下面的代码拷贝到窗体的 Declarations 段中。运行本例, 选择要编辑的节点, 按下命令按钮。

```
Private Sub Form_Load()  
    Dim nodX As Node
```

```
Dim i As Integer
```

```
TreeView1.LabelEdit = tvwManual      ' Set property to manual.
```

```
Set nodX = TreeView1.Nodes.Add(,, " Node 1") ' Add first node.
```

```
For i = 1 to 5 ' Add 5 nodes.
```

```
    Set nodX = TreeView1.Nodes.Add(i,tvwChild,"Node " & CStr(i + 1))
```

```
Next I
```

```
nodX.EnsureVisible      ' Show all nodes.
```

```
End Sub
```

```
Private Sub Command1_Click()
```

```
    ' Invoke the StartLabelEdit method on the selected node,
```

```
    ' which triggers the BeforeLabelEdit event.
```

```
    TreeView1.StartLabelEdit
```

```
End Sub
```

```
Private Sub TreeView1_BeforeLabelEdit (Cancel as Integer)
```

```
    ' If the selected item is the root, then cancel the edit.
```

```
    If TreeView1.SelectedItem Is TreeView1.SelectedItem.Root Then
```

```
        Cancel = True
```

```
    End If
```

```
End Sub
```

```
Private Sub TreeView1_AfterLabelEdit _
```



```
(Cancel As Integer, NewString As String)
' Assume user has entered some text and pressed the ENTER key.
' Any nonempty string will be valid.
If Len(NewString) = 0 Then
    Cancel = True
End If
End Sub
```

## LastSibling 属性

返回 TreeView 控件中 Node 对象的最后一个兄弟对象。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object*.LastSibling

*object* 是在“应用于”中指定的对象表达式。

## 说明

**Node** 对象的最后一个兄弟对象显示在一个体系级中的最后位置上。哪个对象显示在最后位置取决于对象保存在哪个体系级，其由 **Sorted** 属性决定。要在一级上排序 **Node** 对象，将 **Parent** 节点的 **Sorted** 属性设置为 **True**，如下面的代码所示：

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
    Node.Parent.Sorted = True
End Sub
```

**Child**, **FirstSibling**, **LastSibling**, **Previous**, **Parent**, **Next** 和 **Root** 属性返回另一个 **Node** 对象的引用。因此，可以同时引用和操作一个 **Node** 对象，如下：

```
With TreeView1.Nodes(x).FirstSibling
    .Text = "New text"
    .Key = "New key"
    .SelectedImage = 3
End With
```

也可以设置一个对象本例引用 **Node** 对象，如下：

```
Dim NodFirstSib As Node
' Get a reference to the first sibling of Node x.
```

```
Set nodFirstSib = TreeView1.Nodes(x).FirstSibling
' Use this reference to perform operations on the first sibling Node.
With nodFirstSib
    .Text = "New text" ' Change the text.
    .Key = "New key" ' Change key.
    .SelectedImage = 3 ' Change SelectedImage.
End With
```

## 请参阅

Node 对象，Nodes 集合，Child 属性（TreeView 控件），FirstSibling 属性，Next 属性，Parent 属性（Node 对象），Previous 属性（Node 对象），Root 属性（Node 对象），Sorted 属性（TreeView 控件）。

## 示例

下面的例子给 TreeView 控件增加了一个 Node 对象。使用 FirstSibling, Next 和 LastSibling 属性浏览 Node 对象的等级结构。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，点击各节点查看返回值。

```
Private Sub Form_Load()
    Dim nodX As Node
```

```
Set nodX = TreeView1.Nodes.Add(,,"dad","Mike") ' A first sibling.
```

```
Set nodX = TreeView1.Nodes.Add(,,"mom","Carol")
```

Alice is the LastSibling

```
Set nodX = TreeView1.Nodes.Add(,,"Alice")
```

```
Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Marsha")
```

```
Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Jan")
```

Cindy is the lastSibling

```
Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Cindy")
```

```
nodX.EnsureVisible ' Show all nodes.
```

```
Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Greg")
```

```
Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Peter")
```

'Bobby is the lastSibling

```
Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Bobby")
```

```
nodX.EnsureVisible ' Show all nodes.
```

End Sub

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
```

```
Dim strText As String
```

```
Dim n As Integer
```

```
' Set n to FirstSibling's index.
```

```
n = Node.FirstSibling.Index
```

```
' Place FirstSibling's text & linefeed in string variable.
```

```

strText = Node.FirstSibling.Text & vbLF
While n <> Node.LastSibling.Index
' While n is not the index of the last sibling, go to the
' next sibling and place its text into the string variable.
    strText = strText & TreeView1.Nodes(n).Next.Text & vbLF
' Set n to the next node's index.
    n = TreeView1.Nodes(n).Next.Index
Wend
MsgBox strText    ' Display results.
End Sub

```

## LineStyle 属性（TreeView 控件）

返回或设置显示在两个 Node 对象之间的线风格。

应用于

TreeView 控件。

# 语法

*object*.LineStyle [=*number*]

LineStyle 属性的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个 TreeView 控件的对象表达式   |
| <i>number</i> | 一个值或常量，指定线的风格，如“设置”中所示 |

# 设置

*number* 值的设置如下：

| 常量           | 值 | 描述                                |
|--------------|---|-----------------------------------|
| TvwTreeLines | 0 | （缺省）树线。在兄弟节点和父节点之间显示线             |
| TvwRootLines | 1 | 根线。除了在兄弟节点和父节点之间显示线之外，也显示与根节点之间的线 |

# 说明

必须将 Style 属性设置为包含树线的一种风格。

请参阅

TreeView 控件，TreeView 控件常量，Style 属性（TreeView 控件）。

示例

下面的例子给一个 TreeView 控件增加了几个带图像的 Node 对象。你可以选择 OptionButton 控件矩阵改变 LineStyle 和 Style 属性。要使用本例，在一个窗体上放置一个 TreeView 控件，一个 ImageList 控件和两个 OptionButton 控件矩阵（一个有 2 个按钮，一个有 8 个按钮）。将下面的代码拷贝到窗体的 Declarations 段中。运行本例，点击 OptionButton 改变 LineStyle 和 Style 属性。

```
Private Sub Form_Load()  
    ' Add an image to the ImageList control.  
    Dim imgX As ListImage  
    Set imgX = ImageList1.ListImages. _  
        Add(,LoadPicture("bitmaps\outline\leaf.bmp"))  
  
    TreeView1.ImageList = ImageList1 ' Initialize ImageList.  
  
    ' Label OptionButton controls with line styles choices.  
    Option1(0).Caption = "TreeLines"  
    Option1(1).Caption = "RootLines"
```

' Select the first option, and set the LineStyle to TreeLines initially

Option1(0).Value = True

Treeview1.LineStyle = tvwTreeLines

' Label OptionButton controls with Style choices.

Option2(0).Caption = "Text only"

Option2(1).Caption = "Image & text"

Option2(2).Caption = "Plus/minus & text"

Option2(3).Caption = "Plus/minus, image & text"

Option2(4).Caption = "Lines & text"

Option2(5).Caption = "Lines, image & Text"

Option2(6).Caption = "Lines, plus/minus & Text"

Option2(7).Caption = "Lines, plus/minus, image & text"

' Select the last option, and set the initial Style

Option2(7).Value = True

Treeview1.Style = tvwTreelinesPlusMinusPictureText

Dim nodX As Node

Dim i as Integer

' Create root node.

Set nodX = TreeView1.Nodes.Add(,,"Node " & "1",1)

For i = 1 to 5 ' Add 5 nodes.



```
Set nodX = TreeView1.Nodes. _  
Add(i,tvwChild,,"Node " & CStr(i + 1),1)  
Next I  
nodX.EnsureVisible      ' Show all nodes.  
End Sub
```

```
Private Sub Option1_Click(Index as Integer)  
    ' Change line style from OptionButton.  
    TreeView1.LineStyle = Index  
End Sub
```

```
Private Sub Option2_Click(Index as Integer)  
    ' Change Style with OptionButton.  
    TreeView1.Style = Index  
    Form1.Caption = "TreeView Style = " & Option2(Index).Caption  
End Sub
```

## Next 属性

返回 TreeView 控件中 Node 节点的下一个兄弟节点的引用。

应用于

Node 对象，Nodes 集合。

语法

*object*.Next

*object* 是在“应用于”中指定的对象表达式。

说明

Child, FirstSibling, LastSibling, Previous, Parent, Next 和 Root 属性返回另一个 Node 对象的引用。因此，可以同时引用和操作一个 Node 对象，如下：

```
With TreeView1.Nodes(x).child
```

```
.Text = "New text"
```

```
.Key = "New key"
```

```
.SelectedImage = 3
```

```
End With
```

也可以设置一个对象本例引用 Node 对象，如下：

```
Dim Nodchild As Node
```

```
' Get a reference to the first child of Node x.
Set Nodchild = TreeView1.Nodes(x).FirstSibling
' Use this reference to perform operations on the child Node.
With nodchild
    .Text = "New text" ' Change the text.
    .Key = "New key" ' Change key.
    .SelectedImage = 3 ' Change SelectedImage.
End With
```

请参阅

TreeView 控件，Child 属性（Node 对象），FirstSibling 属性，Parent 属性（Node 对象），Previous 属性（Node 对象），Root 属性（Node 对象）。

示例

下面的例子给 TreeView 控件增加了一个 Node 对象。使用 FirstSibling, Next 和 LastSibling 属性浏览 Node 对象的等级结构。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，点击各节点查看返回值。

```
Private Sub Form_Load()
    Dim nodX As Node
```

Set nodX = TreeView1.Nodes.Add(,,"dad","Mike") ' A first sibling.

Set nodX = TreeView1.Nodes.Add(,,"mom","Carol")

Cindy is the Lastsibling

Set nodX = TreeView1.Nodes.Add(,,"Alice")

Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Marsha")

Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Jan")

Cindy is the lastsibling

Set nodX = TreeView1.Nodes.Add("mom",tvwChild,,"Cindy")

nodX.EnsureVisible ' Show all nodes.

Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Greg")

Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Peter")

Bobby is the lastsibling

Set nodX = TreeView1.Nodes.Add("dad",tvwChild,,"Bobby")

nodX.EnsureVisible ' Show all nodes.

End Sub

Private Sub TreeView1\_NodeClick(ByVal Node As Node)

Dim strText As String

Dim n As Integer

' Set n to FirstSibling's index.

n = Node.FirstSibling.Index

' Place FirstSibling's text & linefeed in string variable.

```
strText = Node.FirstSibling.Text & vbCrLf
While n <> Node.LastSibling.Index
' next sibling and place its text into the string variable.
' While n is not the index of the last sibling, go to the
    strText = strText & TreeView1.Nodes(n).Next.Text & vbCrLf
' Set n to the next node's index.
    n = TreeView1.Nodes(n).Next.Index
Wend
MsgBox strText    ' Display results.
End Sub
```

# Node 对象，Nodes 集合

- 一个 Node 对象是 TreeView 控件中可以包含文本和图像的成员。
- 一个 Nodes 集合包含了一个或多个 Node 对象。

## 语法

```
treeview.Nodes  
treeview.Nodes.Item (index)
```

根据标准的集合语法，上面的语句分别指集合和集合中的成员。

Node 对象和 Nodes 集合的语法有如下几个部分：

| 部分              | 描述  |
|-----------------|---|
| <i>treeview</i> | 一个 TreeView 控件的对象表达式                                  |
| <i>index</i>    | 一个整数或字符串，唯一标识 Nodes 集合中的成员。整数是 Index 属性值；字符串是 Key 属性值 |

## 说明

节点可以包含文本和图片。然而，必须使用 ImageList 属性关联一个 ImageList 控件。可以根据节点的状态改变图片。例如，如果你将 SelectedImage 属性设置为关联的 ImageList 中的图像，选中的节点与没有选中的节点图像就不一样。

## 属性

Child 属性 (Node 对象), Expanded 属性, ExpandedImage 属性, FirstSibling 属性, FullPath 属性, LastSibling 属性, Next 属性, Parent 属性 (Node 对象), Previous 属性 (Node 对象), Root 属性 (Node 对象), Selected 属性 (ActiveX 控件), SelectedImage 属性, Sorted 属性 (TreeView 控件), Bold 属性, Checked 属性, Text 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Count 属性 (ActiveX 控件), Item 属性 (ActiveX 控件), Key 属性 (ActiveX 控件), Image 属性 (ActiveX 控件)。

## 方法

Add 方法 (Nodes 集合), CreateDragImage 方法, EnsureVisible 方法, Remove 方法 (ActiveX 控件), Clear 方法 (ActiveX 控件)。

## 请参阅

TreeView 控件, Nodes 属性, SelectedImage 属性, Index 属性 (控件矩阵), ImageList 控件, ImageList 属性 (ActiveX 控件)。

## NodeCheck 事件

如果 CheckBoxes 属性等于 True, 那么当 Node 对象被选择或者取消选择时将发生此事件。

应用于

TreeView 控件。

语法

*Event* NodeCheck(ByVal *Node* As ComctlLib.Node)

NodeCheck 事件的语法包括以下几个部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 一个对象表达式，其值等于“应用于”列表中的一个对象 |
| <i>node</i>   | 返回被选择的 Node 对象的对象引用       |

说明

当 Node 对象的 Checked 属性被设置为 True 或 False 时不会发生此事件。

请参阅

Checkboxes 属性。

## NodeClick 事件

当点击 Node 对象时产生该事件。



应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

```
Private Sub object_NodeClick ( ByVal node As Node)
```

NodeClick 事件的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |
| <i>node</i>   | 所选择的 Node 对象的引用 |

说明

当用户点击 Node 对象之外的 TreeView 控件中任何部位时都产生标准的 Click 事件。当用户点击某个 Node 对象时产生 NodeClick 事件；NodeClick 事件也返回某个特定 Node 对象的引用，以便在进一步采取动作之前验证 Node 节点。

NodeClick 事件在标准的 Click 事件之前产生。

请参阅

SelectedItem 属性 (ActiveX 控件)，Click 事件，Click 事件 (ActiveX 控件)。

示例

下面的例子给一个 TreeView 控件增加了几个 Node 对象。当点击一个 Node

对象时，触发 **NodeClick** 事件获取该 **Node** 对象的索引和文本。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击任一 **Node** 节点。

```
.Private Sub Form_Load()  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(., "R", "Root")  
    nodX.Expanded = True  
    Set nodX = TreeView1.Nodes.Add(., "P", "Parent")  
    nodX.Expanded = True  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "Child 1")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "Child 2")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "Child 3")  
    Set nodX = TreeView1.Nodes.Add("P", tvwChild, "Child 4")  
    Set nodX = TreeView1.Nodes.Add("P", tvwChild, "Child 5")  
    Set nodX = TreeView1.Nodes.Add("P", tvwChild, "Child 6")  
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)  
    Form1.Caption = "Index = " & Node.Index & " Text:" & Node.Text  
End Sub
```

## Nodes 属性

返回 **TreeView** 控件中 **Node** 对象集合的引用。

应用于

**TreeView** 控件，**Node** 对象，**Nodes** 集合。

语法

*object*.Nodes

*object* 是在“应用于”中指定的对象表达式。

说明

可以使用标准的集合方法（如 **Add** 和 **Remove** 方法）操纵 **Node** 对象。可以使用索引或保存在 **Key** 属性中的键值访问每个对象。

请参阅

**Add** 方法（**Nodes** 集合）。

示例

下面的例子给一个 **TreeView** 控件增加了几个 **Node** 对象。当点击窗体时，使用每个 **Node** 对象的引用显示其文本。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点

击窗体。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(, "R", "Root")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "C1", "Child 1")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "C2", "Child 2")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "C3", "Child 3")  
    Set nodX = TreeView1.Nodes.Add("R", tvwChild, "C4", "Child 4")  
    nodX.EnsureVisible  
    TreeView1.Style = tvwTreelinesText ' Style 4.  
    TreeView1.BorderStyle = vbFixedSingle  
End Sub
```

```
Private Sub Form_Click()  
    Dim i As Integer  
    Dim strNodes As String  
    For i = 1 To TreeView1.Nodes.Count  
        strNodes = strNodes & TreeView1.Nodes(i).Index & " " & _  
        "Key: " & TreeView1.Nodes(i).Key & " " & _  
        "Text: " & TreeView1.Nodes(i).Text & vbLF  
    Next i  
    MsgBox strNodes
```

End Sub

## Parent 属性（Node 对象）

返回或设置一个 Node 对象的父节点对象。只在运行时可用。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object.Parent* [=*node*]

Parent 属性的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在“应用于”中指定的对象表达式 |
| <i>node</i>   | 对象父节点的 Node 对象  |

说明

在运行时，如果将该属性设置成为一个循环，则产生错误。例如，不能将一个 Node 对象设置为其后代的子节点。

Child, FirstSibling, LastSibling, Previous, Parent, Next 和 Root 属性返回另一

个 Node 对象的引用。因此，可以同时引用和操作一个 Node 对象，如下：

```
With TreeView1.Nodes(x).parent  
    .Text = "New text"  
    .Key = "New key"  
    .SelectedImage = 3  
End With
```

也可以设置一个对象本例引用 Node 对象，如下：

```
Dim Nodparent As Node  
' Get a reference to the firstparent of Node x.  
Set Nodparent = TreeView1.Nodes(x).parent  
' Use this reference to perform operations on the parent Node.  
With nodparent  
    .Text = "New text" ' Change the text.  
    .Key = "New key" ' Change key.  
    .SelectedImage = 3 ' Change SelectedImage.  
End With
```

请参阅

Node 对象, Nodes 集合, Child 属性(Node 对象), FirstSibling 属性, LastSibling 属性, Next 属性, Previous 属性 (Node 对象), Root 属性 (Node 属性), Form 对象, Forms 集合, MDIForm 对象, MDIChild 属性。

## 示例

下面的例子给一个 **TreeView** 控件增加了几个 **Node** 对象。在选择了一个 **Node** 对象后，可以将其拖放到其他 **Node** 对象上，使之成为目的对象的子节点。要使用本例，在一个窗体上放置一个 **TreeView** 控件和一个 **ImagList** 控件，将下面的代码拷贝到窗体的 **Declaratons** 段中。运行本例，将 **Node** 对象拖放到其他的 **Node** 对象上，查看结果。

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
    ' Load a bitmap into an Imagelist control.
    Dim imgX As ListImage
    Dim BitmapPath As String
    BitmapPath = "icons\mail\mail01a.ico"
    Set imgX = ImageList1.ListImages.Add(, LoadPicture(BitmapPath))

    ' Initialize TreeView control and create several nodes.
    TreeView1.ImageList = ImageList1
    Dim nodX As Node      ' Create a tree.
    Set nodX = TreeView1.Nodes.Add(, , "Parent1", 1)
    Set nodX = TreeView1.Nodes.Add(, , "Parent2", 1)
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Child 1", 1)
```

```
Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
nodX.EnsureVisible ' Expand tree to show all nodes.
```

```
End Sub
```

```
Private Sub TreeView1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
```

```
    Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
    Set TreeView1.DropHighlight = Nothing
```

```
End Sub
```

```
Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
```

```
    If Button = vbLeftButton Then ' Signal a Drag operation.
        indrag = True ' Set the flag to true.
        ' Set the drag icon with the CreateDragImage method.
        TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
        TreeView1.Drag vbBeginDrag ' Drag operation.
```

```
    End If
```

```
End Sub
```

```
Private Sub TreeView1_DragDrop(Source As Control, x As Single, y As Single)
```



' If user didn't move mouse or released it over an invalid area.

If TreeView1.DropHighlight Is Nothing Then

    indrag = False

    Exit Sub

Else

    ' Set dragged node's parent property to the target node.

    On Error GoTo checkerror ' To prevent circular errors.

    Set nodX.Parent = TreeView1.DropHighlight

    Cls

    Print TreeView1.DropHighlight.Text & \_

    " is parent of " & nodX.Text

    ' Release the DropHighlight reference.

    Set TreeView1.DropHighlight = Nothing

    indrag = False

    Exit Sub ' Exit if no errors occurred.

End If

checkerror:

    ' Define constants to represent Visual Basic errors code.

    Const CircularError = 35614

    If Err.Number = CircularError Then

        Dim msg As String

        msg = "A node can't be made a child of its own children."

```

        ' Display the message box with an exclamation mark icon
        ' and with OK and Cancel buttons.
        If MsgBox(msg, vbExclamation & vbOKCancel) = vbOK Then
            ' Release the DropHighlight reference.
            indrag = False
            Set TreeView1.DropHighlight = Nothing
            Exit Sub
        End If
    End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, State As
Integer)
    Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
End Sub

```

## PathSeparator 属性（TreeView 控件）

返回或设置 FullPath 属性返回的全路径中的分隔符。

应用于

TreeView 控件。

语法

*object*.PathSeparator [=string]

PathSeparator 属性的语法有如下几个部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 一个 TreeView 控件的对象表达式            |
| <i>string</i> | 一个字符串，确定 PathSeparator，通常都是单个字符 |

说明

缺省的分隔符是 “\”。

请参阅

FullPath 属性。

示例

下面的例子给一个 TreeView 控件增加了几个 Node 对象。用户可以使用 OptionButton 控件矩阵改变 PathSeparator 属性值。要使用本例，在一个窗体上放置一个 TreeView 控件和一个 OptionButton 控件矩阵，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，选择一个 Node 对象，然后点击窗体。使用 OptionButton 改变 PathSeparator 属性值。

```
Private Sub Form_Load
```

```
    TreeView1.BorderStyle = vbFixedSingle    ' Show border.
```

' Label OptionButton controls with Style choices.

Option1(0).Caption = "/"

Option1(1).Caption = "-"

Option1(2).Caption = ":"

' Select the last option, and set the initial Style

Option2(1).Value = True

TreeView1.PathSeparator = Option1(1).Caption

Dim nodX As Node

Dim i As Integer

Set nodX = TreeView1.Nodes.Add(,,CStr(1))     ' Add first node.

For i = 1 to 5     ' Add other nodes.

    Set nodX = TreeView1.Nodes.Add(i,tvwChild,,CStr(i + 1))

Next i

nodX.EnsureVisible     ' Ensure all are visible.

End Sub

Private Sub Option1\_Click(Index as Integer)

    ' Change the delimiter character.

    TreeView1.PathSeparator = Option1(Index).Caption

End Sub

Private Sub TreeView1\_NodeClick(ByVal Node As Node)

```
' Show path in form's caption.  
Me.Caption = Node.FullPath  
End Sub
```

## Previous 属性（Node 对象）

返回 TreeView 控件中 Node 节点的前一个兄弟节点的引用。

应用于

Node 对象，Nodes 集合，TreeView 控件。

语法

*object*.Previous

*object* 是在“应用于”中指定的对象表达式。

说明

Child, FirstSibling, LastSibling, Previous, Parent, Next 和 Root 属性返回另一个 Node 对象的引用。因此，可以同时引用和操作一个 Node 对象，如下：

```
With TreeView1.Nodes(x).FirstSibling  
.Text = "New text"  
.Key = "New key"
```

```
.SelectedImage = 3
```

```
End With
```

也可以设置一个对象本例引用 Node 对象，如下：

```
Dim Nodprevious As Node
```

```
' Get a reference to the first sibling of Node x.
```

```
Set Nodchild = TreeView1.Nodes(x).FirstSibling
```

```
' Use this reference to perform operations on the previous Node.
```

```
With nodprevious
```

```
    .Text = "New text" ' Change the text.
```

```
    .Key = "New key" ' Change key.
```

```
    .SelectedImage = 3 ' Change SelectedImage.
```

```
End With
```

请参阅

Child 属性（Node 对象），FirstSibling 属性，LastSibling 属性，Next 属性，Previous 属性（Node 对象），Root 属性（Node 对象）。

示例

下面的例子给 TreeView 控件增加了一个 Node 对象。使用 FirstSibling, Previous 和 LastSibling 属性浏览 Node 对象的等级结构。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，点击各节点查看返回值。

```
Private Sub Form_Load()
```

```
    Dim nodX As Node
```

```
    Set nodX = TreeView1.Nodes.Add(, "r", "Root")
```

```
    Set nodX = TreeView1.Nodes.Add(, "p", "parent")
```

```
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "Child 1")
```

```
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "Child 2")
```

```
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "Child 3")
```

```
    nodX.EnsureVisible ' Show all nodes.
```

```
    Set nodX = TreeView1.Nodes.Add("p", tvwChild, "Child 4")
```

```
    Set nodX = TreeView1.Nodes.Add("p", tvwChild, "Child 5")
```

```
    Set nodX = TreeView1.Nodes.Add("p", tvwChild, "Child 6")
```

```
    nodX.EnsureVisible ' Show all nodes.
```

```
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
```

```
    Dim strText As String
```

```
    Dim n As Integer
```

```
    ' Set n to LastSibling's index.
```

```
    n = Node.LastSibling.Index
```

```
    ' Place LastSibling's text & linefeed in string variable.
```

```
    strText = Node.LastSibling.Text & vbCrLf
```

```
    While n <> Node.FirstSibling.Index
```

```
' While n is not the index of the FirstSibling, go to the  
' previous sibling and place its text into the string variable.  
strText = strText & TreeView1.Nodes(n).Previous.Text & vbCrLf  
' Set n to the previous node's index.  
n = TreeView1.Nodes(n).Previous.Index  
Wend  
MsgBox strText ' Display results.  
End Sub
```

## Root 属性（Node 对象）

返回选择的 Node 的根节点的引用。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object*.Root

*object* 是在“应用于”中指定的对象表达式。

说明

Child, FirstSibling, LastSibling, Previous, Parent, Next 和 Root 属性返回另一



个 Node 对象的引用。因此，可以同时引用和操作一个 Node 对象，如下：

```
With TreeView1.Nodes(x).Root
    .Text = "New text"
    .Key = "New key"
    .SelectedImage = 3
End With
```

也可以设置一个对象本例引用 Node 对象，如下：

```
Dim NodRoot As Node
' Get a reference to the first sibling of Node x.
Set NodRoot= TreeView1.Nodes(x).FirstSibling
' Use this reference to perform operations on the Root Node.
With nodRoot
    .Text = "New text" ' Change the text.
    .Key = "New key" ' Change key.
    .SelectedImage = 3 ' Change SelectedImage.
End With
```

请参阅

Child 属性 (Node 对象)，FirstSibling 属性，LastSibling 属性，Next 属性，Previous 属性 (Node 对象)，Parent 属性 (Node 对象)，SelectedItem 属性 (ActiveX 控件)。

## 示例

下面的例子给 **TreeView** 控件增加了一个 **Node** 对象。当点击一个 **Node** 对象时，程序将追溯到根节点，并显示每个父节点的文本。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击一个节点。

```
Private Sub Form_Load()  
    Dim nodX As Node ' Create a tree.  
    Set nodX = TreeView1.Nodes.Add(, "r", "Root")  
    Set nodX = TreeView1.Nodes.Add(, "p", "Parent")  
    Set nodX = TreeView1.Nodes.Add("p", tvwChild, "Child 1")  
    nodX.EnsureVisible      ' Show all nodes.  
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "C2", "Child 2")  
    Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C3", "Child 3")  
    Set nodX = TreeView1.Nodes.Add("C3", tvwChild, "Child 4")  
    Set nodX = TreeView1.Nodes.Add("C3", tvwChild, "Child 5")  
    nodX.EnsureVisible      ' Show all nodes.  
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)  
    Dim n As Integer  
    Dim strParents As String      ' Variable for information.  
    n = Node.Index      ' Set n to index of clicked node.
```

```

strParents = Node.Text & vbCrLf
While n <> Node.Root.Index
    strParents = strParents & _
    TreeView1.Nodes(n).Parent.Text & vbCrLf
    ' Set n to index of next parent Node.
    n = TreeView1.Nodes(n).Parent.Index
Wend
MsgBox strParents
End Sub

```

## Scroll 属性

返回或设置一个值，确定是否显示滚动条。

应用于

TreeView 控件。

语法

*object*.Scroll [= *boolean*]

Scroll 属性语法有如下几部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定对象是否含有滚动条，如设置值所描述的 |

## 设置

*boolean* 设置如下:

| 常量    | 描述          |
|-------|-------------|
| False | (缺省)对象没有滚动条 |
| True  | 对象有滚动条      |

## Selected 属性（ActiveX 控件）

返回或设置一个值，该值确定是否选择了 Node 对象或 Tab 对象。对于 ListItem 对象，Selected 属性没有设置 SelectedItem 属性，因此没有选择的对象。其仅返回一个值表明 ListItem 对象是否已被其他方法选中。

## 应用于

Node 对象，Nodes 集合，ComboItem 对象，ListItem 对象， ListItem 集合，Tab 对象。

## 语法

*object.Selected* [= *boolean*]

Selected 属性的语法有如下几个部分：

| 部分             | 描述               |
|----------------|------------------|
| <i>object</i>  | 在“应用于”中指定的对象表达式  |
| <i>boolean</i> | 一个布尔表达式，指定对象是否选中 |

## 说明

使用 Selected 属性选择某个特定的 Node 或 Tab 对象。一旦以这种方式选择了一个对象，可以对该对象进行各种操作，如设置属性和调用方法等。

要选择某特定对象，必须使用 Index 属性或 Key 属性引用它。下面的代码选择了 TreeView 控件中的一个 Node 对象：

```
Private Sub Command1_Click()  
    TreeView1.Nodes(3).Selected = True ' Selects an object.  
    ' Use the SelectedItem property to get a reference to the object.  
    TreeView1.SelectedItem.Text = "Changed Text"  
End Sub
```

在 ListView 控件中，SelectedItem 属性总是指向第一个选中的成员。因此，如果选中了多个对象，必须轮回检查每个对象的 Selected 属性。

**注意：**要使用 Selected 属性选择一个 ListItem 对象，可以使用 Set 语句和 SelectedItem 属性，如下：

```
Set ListView1.SelectedItem = ListView1.ListItem(1)
```

请参阅

Set 语句，SelectedItem 属性（ActiveX 控件）。

示例

下面的例子给一个 TreeView 控件增加了几个 Node 对象。当选中一个 Node 对象时，使用该对象的引用显示键值。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，选择一个 Node 对象，点击窗体。

```
Private Sub Form_Load()  
    Dim nodX As Node ' Create a tree.  
    Set nodX = TreeView1.Nodes.Add(., "r", "Root")  
    Set nodX = TreeView1.Nodes.Add(., "p", "Parent")  
    Set nodX = TreeView1.Nodes.Add("p", tvwChild, "Child 1")  
    nodX.EnsureVisible      ' Show all nodes.  
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "C2", "Child 2")  
    Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C3", "Child 3")  
    Set nodX = TreeView1.Nodes.Add("C3", tvwChild, "Child 4")  
    Set nodX = TreeView1.Nodes.Add("C3", tvwChild, "Child 5")
```

```

        nodX.EnsureVisible      ' Show all nodes.
End Sub

Private Sub Form_Click()
    Dim intX As Integer
    On Error Resume Next      ' If an integer isn't entered.
    intX = InputBox("Check Node",TreeView1.SelectedItem.Index)
    If IsNumeric(intX) Then ' Ensure an integer was entered.
        If TreeView1.Nodes(intX).Selected = True Then
            MsgBox TreeView1.Nodes(intX).Text & " is selected."
        Else
            MsgBox "Not selected"
        End If
    End If
End Sub

```

下面的例子给一个 **ListView** 控件增加了 3 个 **ListItem** 对象。当点击窗体时，程序代码使用 **Selected** 属性确定是否选中了某个特定的 **ListItem** 对象。要使用本例，在一个窗体上放置一个 **ListView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，选择一个 **ListItem** 对象，点击窗体。

```

Private Sub Form_Load()
    Listview1.BorderStyle = vbFixedSingle  ' Show the border.
    Dim itmX As ListViewItem

```

```
Set itmX = ListView1.ListItems.Add(, "Item 1")
Set itmX = ListView1.ListItems.Add(, "Item 2")
Set itmX = ListView1.ListItems.Add(, "Item 3")
End Sub
```

```
Private Sub Form_Click()
    Dim intX As Integer
    On Error Resume Next ' If an integer isn't entered.
    intX = InputBox("Check Item", , ListView1.SelectedItem.Index)
    If IsNumeric(intX) Then ' Ensure an integer was entered.
        If ListView1.ListItems(intX).Selected = True Then
            MsgBox ListView1.ListItems(intX).Text & " is selected."
        Else
            MsgBox "Not selected"
        End If
    End If
End Sub
```

## SelectedImage 属性

返回或设置与 ImageList 控件关联的 ListImage 对象的索引或键值。当选择一个 Node 对象时，显示该 ListImage 对象。



应用于

TreeView 控件，Node 对象，Nodes 集合，ImageList 控件，ListImage 对象，ListImages 集合。

语法

*object*.SelectedImage [=*index*]

SelectedImage 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 在“应用于”中指定的对象表达式   |
| <i>index</i>  | 一个整数或唯一字符串，标识关联的 ImageList 控件中的一个 ListImage 对象。整数是 ListImage 对象的 Index 属性值；字符串是 ListImage 对象的 Key 属性值 |

说明

如果该属性设置为 Null， 则使用 Image 属性指定的缺省图像的掩码。

请参阅

Image 属性（ActiveX 控件）。

## SelectedItem 属性（ActiveX 控件）

返回所选中的 ListItem, Node 或 Tab 对象的引用。

应用于

TreeView 控件, Node 对象, Nodes 集合, ListView 控件, ListItem 对象, ListItems 集合, TabStrip 控件, Tab 对象。

语法

*object*.SelectedItem

*object* 是在“应用于”中指定的对象表达式。

说明

SelectedItem 属性返回一个对象的引用，可以用来设置对象的属性或调用对象的方法。该属性通常用于返回用户选中的 ListItem, Node 或 Tab 对象的引用。有了该引用，可以在采取进一步的动作之前验证对象，如下所示：

```
Command1_Click()  
    ' If the selected object is not the root, then remove the Node.  
    If TreeView1.SelectedItem.Index <> 1 Then  
        Treeview1.Nodes.Remove TreeView1.SelectedItem.Index  
    End If  
End Sub
```

要在程序代码中选择一个 **ListItem** 对象，使用 **Set** 语句和 **SelectedItem** 属性，如下所示：

```
Set ListView1.SelectedItem = ListView1.ListItems(1)
```

请参阅

**Set** 语句，**SelectedItem** 属性（**ActiveX** 控件）。

示例

下面的例子给一个 **TreeView** 控件增加了几个 **Node** 对象。在选中了一个 **Node** 对象后，点击窗体查看该 **Node** 对象的各种属性。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，选择一个 **Node** 对象，点击窗体。

```
Private Sub Form_Load()  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(, "r", "Root")  
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c1", "Child 1")  
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c2", "Child 2")  
    Set nodX = TreeView1.Nodes.Add("r", tvwChild, "c3", "Child 3")  
    Set nodX = TreeView1.Nodes.Add("c3", tvwChild, "c4", "Child 4")  
    Set nodX = TreeView1.Nodes.Add("c3", tvwChild, "c5", "Child 5")  
    Set nodX = TreeView1.Nodes.Add("c5", tvwChild, "c6", "Child 6")  
    Set nodX = TreeView1.Nodes.Add("c5", tvwChild, "c7", "Child 7")
```

```
nodX.EnsureVisible
TreeView1.BorderStyle = vbFixedSingle
End Sub
```

```
Private Sub Form_Click()
    Dim nodX As Node
    ' Set the variable to the SelectedItem.
    Set nodX = TreeView1.SelectedItem
    Dim strProps As String
    ' Retrieve properties of the node.
    strProps = "Text: " & nodX.Text & vbLF
    strProps = strProps & "Key: " & nodX.Key & vbLF
    On Error Resume Next ' Root node doesn't have a parent.
    strProps = strProps & "Parent: " & nodX.Parent.Text & vbLF
    strProps = strProps & "FirstSibling: " & _
    nodX.FirstSibling.Text & vbLF
    strProps = strProps & "LastSibling: " & _
    nodX.LastSibling.Text & vbLF
    strProps = strProps & "Next: " & nodX.Next.Text & vbLF

    MsgBox strProps
End Sub
```

# SingleSel 属性

返回或设置一个值，指定项目被选中时是否展开。

应用于

TreeView 控件。

语法

*object*.SingleSel [= *boolean*]

SingleSel 属性语法有如下几部分:

| 部分             | 描述                            |
|----------------|-------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象      |
| <i>boolean</i> | 一个布尔表达式，指定项目被选中时是否展开，如设置值所描述的 |

设置

*boolean* 设置如下:

| 常量    | 描述            |
|-------|---------------|
| False | (缺省)项目被选中时不展开 |
| True  | 项目被选中时展开      |

# Sorted 属性（TreeView 控件）

- 返回或设置一个值，该值确定 Node 对象的子节点是否按字符表次序排列。
- 返回或设置一个值，该值确定 TreeView 控件中根一级的 Node 对象是否按字符表次序排列。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object*.Sorted [= *boolean*]

Sorted 属性的语法有如下几个部分：

| 部分             | 描述                                  |
|----------------|-------------------------------------|
| <i>object</i>  | 在“应用于”中指定的对象表达式                     |
| <i>boolean</i> | 一个布尔表达式，指定 Node 对象是否排序，如“设置”<br>中所示 |

设置

*boolean* 值的设置如下：

| 设置    | 描述  |
|-------|---|
| True  | Node 对象根据 Text 属性按字符表次序排列。Node 对象中，Text 属性是以数字开头的按字符串排序，第一个数字决定了排序的初始位置，后续数字决定子排序 |
| False | Node 对象不排序  |

## 说明

可以两种方式使用 Sorted 属性第一，排序 TreeView 控件中根节点 Node 对象；第二，排序单个 Node 对象子节点。例如，下面的代码对 TreeView 控件的根节点进行了排序：

```
Private Sub Command1_Click()
    TreeView1.Sorted = True    ' Top level Node objects are sorted.
End Sub
```

下面的代码显示了在创建 Node 对象时如果设置 Sorted 属性

```
Private Sub Form_Load()
    Dim nodX As Node
    Set nodX = TreeView1.Nodes.Add(,, "Parent Node")
    nodX.Sorted = True
End Sub
```

将 Sorted 属性设置为 True 将只排序当前的 Nodes 集合。当给 TreeView 控

件增加了新的 **Node** 对象时，必须再次将 **Sorted** 属性设置为 **True** 以便排序新增加的 **Node** 对象。

请参阅

**Text** 属性（**ActiveX** 控件）。

示例

下面的例子给一个 **TreeView** 控件增加了几个 **Node** 对象。当点击一个 **Node** 对象时，询问你是否进行排序。要使用本例，在一个窗体上放置一个 **TreeView** 控件，将下面的代码拷贝到窗体的 **Declarations** 段中。运行本例，点击 **Node** 进行排序。

```
Private Sub Form_Load()  
    ' Create a tree with several unsorted Node objects.  
    Dim nodX As Node  
    Set nodX = TreeView1.Nodes.Add(, , "Adam")  
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "z", "Zachariah")  
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Noah")  
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Abraham")  
    Set nodX = TreeView1.Nodes.Add("z", tvwChild, , "Stan")  
    Set nodX = TreeView1.Nodes.Add("z", tvwChild, , "Paul")  
    Set nodX = TreeView1.Nodes.Add("z", tvwChild, "f", "Frances")  
    Set nodX = TreeView1.Nodes.Add("f", tvwChild, , "Julie")
```



```

Set nodX = TreeView1.Nodes.Add("f", tvwChild, "c", "Carol")
Set nodX = TreeView1.Nodes.Add("f", tvwChild, , "Barry")
Set nodX = TreeView1.Nodes.Add("c", tvwChild, , "Yale")
Set nodX = TreeView1.Nodes.Add("c", tvwChild, , "Harvard")
nodX.EnsureVisible

End Sub

Private Sub TreeView1_NodeClick(ByVal Node As Node)
    Dim answer As Integer
    ' Check if there are children nodes.
    If Node.Children > 1 Then ' There are more than one children nodes.
        answer = MsgBox("Sort this node?", vbYesNo) ' Prompt user.
        If answer = vbYes Then ' User wants to sort.
            Node.Sorted = True
        End If
    End If
End Sub

```

## StartLabelEdit 方法

使用户可以编辑标签。

应用于

TreeView 控件，ListView 控件。

语法

*object*.StartLabelEdit

object 是一个 TreeView 控件的对象表达式。

说明

当 LabelEdit 属性设置为 1（Automatic）时，必须使用 StartLabelEdit 方法初始化标签编辑操作。

当调用一个对象的 StartLabelEdit 方法时，也产生 BeforeLabelEdit 事件。

请参阅

Node 对象，Nodes 集合，AfterLabelEdit 事件（ListView 控件，TreeView 控件），BeforeLabelEdit 事件（ListView 控件，TreeView 控件），LabelEdit 属性，ListItem 对象，ListItems 集合。

示例

下面的例子给一个 TreeView 控件增加了几个 Node 对象。选中了一个 Node 对象之后，点击窗体开始进行编辑。要使用本例，在一个窗体上放置一个 TreeView 控件，将下面的代码拷贝到窗体的 Declarations 段中。运行本例，选择 Node 对象，点击窗体。

```
.Private Sub Form_Load
    Dim nodX As Node

    Set nodX = TreeView1.Nodes.Add(,, "Da Vinci") ' Root
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Titian")
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Rembrandt")
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "Goya")
    Set nodX = TreeView1.Nodes.Add(1, tvwChild, "David")
    nodX.EnsureVisible ' Expand tree to see all nodes.
End Sub
```

```
Private Sub Form_Click()
    ' If selected Node isn't the Root node then allow edits.
    If TreeView1.SelectedItem.Index <> 1 Then
        TreeView1.StartLabelEdit
    End If
End Sub
```

## Style 属性（TreeView 控件）

返回或设置 TreeView 控件中每个 Node 对象上显示的图形和文本类型。

应用于

TreeView 控件，Node 对象，Nodes 集合。

语法

*object*.Style [=*number*]

Style 属性的语法有如下几个部分：

| 部分            | 描述                    |
|---------------|-----------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式       |
| <i>number</i> | 一个整数，决定图形的类型，如“设置”中所示 |

设置

*number* 值的设置如下：

| 设置 | 描述              |
|----|-----------------|
| 0  | 只有文本            |
| 1  | 图像和文本           |
| 2  | 加减号和文本          |
| 3  | 加减号，图像和文本       |
| 4  | 线和文本            |
| 5  | 线，图像和文本         |
| 6  | 线，加减号和文本        |
| 7  | （缺省）线，加减号，图像和文本 |

## 说明

如果 **Style** 属性设置的值中包含线，则 **LineStyle** 属性决定了线的外观。如果 **Style** 属性设置值中不包含线，则忽略 **LineStyle** 属性。

## 请参阅

**LineStyle** 属性

# UpDown 控件

一个 **UpDown** 控件包含了一对箭头按钮，点击按钮可以增大或减少一个值，如滚动杆的位置或关联的伴随控件的值。

## 语法

**UpDown**

## 说明

对于用户而言，**UpDown** 控件及其伴随控件看起来就象一个控件。伴随控件是任何可以通过 **BuddyControl** 属性与 **UpDown** 控件连接的控件，通常用于显示数据，如一个 **TextBox** 控件或一个 **CommandButton** 控件。

**注意：**小的无窗口的控件如内部标签控件不能成为伴随控件。

设置 **AutoBuddy** 属性，**UpDown** 控件可以自动根据跳跃次序将上一个控件作为自己的伴随控件。如果没有上一个控件，则 **UpDown** 控件使用下一个控件作为自己的伴随控件。设置伴随控件的另一种方法是使用 **BuddyControl** 属性。在设计时，无论是设置了 **AutoBuddy** 还是 **BuddyControl** 属性，伴随控件将自动与 **UpDown** 控件结成伙伴，在 **UpDown** 控件的旁边定位并相应地改变大小。使用 **Alignment** 属性，**UpDown** 控件可以在伴随控件的右边或左边。

**Increment**, **Min**, **Max** 和 **Wrap** 属性决定在用户点击 **UpDown** 控件时，该控件如何改变 **Value** 属性的值。例如，如果你的值要乘 10，范围从 20 到 80，你可以将 **Increment**, **Min** 和 **Max** 分别设置 10, 20 和 80。**Wrap** 属性允许 **Value** 值增大超过 **Max** 属性并从 **Min** 属性重新开始，或反之。

没有伴随控件的 **UpDown** 控件就象简化的滚动杆。

**注意：****UpDown** 控件应作为 Visual Basic 4.0 中的 **Spin Button** 控件的替代品。

**发布须知：****UpDown** 控件是 **COMCTL32.OCX** 文件中 **ActiveX** 控件的一部分。要在你的应用程序中使用 **UpDown** 控件，必须将 **COMCTL32.OCX** 文件加入到你的工程文件中。当发布你的应用程序时，在用户的 **Microsoft Windows System** 或 **System32** 目录下安装 **COMCTL32.OCX** 文件。有关其他信息，请参阅《**Microsoft Visual Basic 6.0 程序员指南**》。

请参阅

**Error** 常量 (**ComCtl32**)。

## 属性

AutoBuddy 属性, Increment 属性, SyncBuddy 属性, Alignment 属性 (UpDown 控件), BuddyControl 属性, Max 属性, Min 属性, Orientation 属性, Value 属性 (UpDown 控件), Wrap 属性, BuddyProperty 属性, TabIndex 属性, DragIcon 属性, DragMode 属性, hWnd 属性, TabStop 属性, HelpContextID 属性, Name 属性, Parent 属性, Container 属性, ToolTipText 属性, WhatsThisHelpID 属性, OLEDropMode 属性 (ActiveX 控件), Value 属性 (ActiveX 控件), Height, Width 属性 (ActiveX 控件), Index 属性 (ActiveX 控件), Left, Top 属性 (ActiveX 控件), Tag 属性 (ActiveX 控件), Visible 属性 (ActiveX 控件), Object 属性 (ActiveX 控件), Enable 属性 (ActiveX 控件), hWnd 属性 (ActiveX 控件)。

## 方法

SetFocus 方法, Drag 方法, Move 方法, ZOrder 方法, ShowWhatsThis 方法, OLEDrag 方法 (ActiveX 控件)。

## 事件

Change 事件 (UpDown 控件), DownClick 事件, UpClick 事件, DragDrop 事件, DragOver 事件, GotFocus 事件, LostFocus 事件, MouseDown, MouseUp 事件, MouseMove 事件, Validate 事件, OLECompleteDrag 事件 (ActiveX 控件), OLEDragDrop 属性 (ActiveX 控件), OLEDragOver 属性 (ActiveX 控件), OLEGiveFeedback 属性 (ActiveX 控件), OLESetData 属性 (ActiveX 控

件），OLEStartDrag 属性（ActiveX 控件）。

请参阅

AutoBuddy 属性，Increment 属性，BuddyControl 属性，Max 属性，Min 属性，Value 属性(UpDown 控件)，Wrap 属性，BuddyProperty 属性，CommandButton 控件，TextBox 控件，使用 UpDown 控件。

## Alignment 属性（UpDown 控件）

返回或设置一个值，该值确定 UpDown 控件与伴随控件的对齐方式。

应用于

UpDown 控件。

语法

*object*.Alignment [=value]

Alignment 属性的语法有如下几个部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式                  |
| <i>value</i>  | 一个值，指定 UpDown 控件与伴随控件的对齐方式，如“设置”中所示 |



# 设置

value 值的设置如下:

| 常量                | 值 | 描述                |
|-------------------|---|-------------------|
| cc2alignmentLeft  | 0 | UpDown 控件在伴随控件的左边 |
| cc2alignmentRight | 1 | UpDown 控件在伴随控件的右边 |

# 说明

使用 Alignment 属性指定 UpDown 控件与伴随控件的位置。缺省地, UpDown 控件显示在伴随控件的右边。

设置 Alignment 属性将自动重新对齐 UpDown 控件和伴随控件。伴随控件的宽度将减去 UpDown 控件的宽度, 从而二者宽度之和与单个的伴随控件宽度一样。

注意: 当对齐 UpDown 控件时, Alignment 属性将忽略 Orientation 属性的值。

# 请参阅

BuddyControl 属性, Orientation 属性。

# AutoBuddy 属性

返回或设置一个值，该值确定 UpDown 控件是否根据跳跃次序自动设置一个伴随控件。

应用于  
UpDown 控件。

语法

*object*.AutoBuddy [=*value*]

AutoBuddy 属性的语法有如下几个部分：

| 部分            | 描述                      |
|---------------|-------------------------|
| <i>object</i> | 在“应用于”中指定的对象表达式         |
| <i>value</i>  | 一个布尔表达式，指定伴随控件，如“设置”中所示 |

设置

*value* 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | <b>UpDown</b> 控件使用跳跃次序中上一个控件作为伴随控件。<br>如果没有上一个控件，则使用索引值高的第一个可用控件作为伴随控件 |
| False | （缺省） <b>UpDown</b> 控件使用 <b>BuddyControl</b> 属性中设置的控件作为伴随控件             |

### 说明

将 **AutoBuddy** 属性设置为 **True** 也将设置 **BuddyControl** 属性。将 **AutoBuddy** 设置为 **False** 将清除 **BuddyControl** 属性。

### 请参阅

**SyncBuddy** 属性，**BuddyControl** 属性，**BuddyProperty** 属性。

## BuddyControl 属性

设置或返回伴随控件。

### 应用于

**UpDown** 控件。

## 语法

*object*.BuddyControl [=*value*]

BuddyControl 属性的语法有如下几个部分：

| 部分            | 描述              |
|---------------|-----------------|
| <i>object</i> | 在”应用于”中指定的对象表达式 |
| <i>value</i>  | 可变量，指定伴随控件，如下所述 |

## 说明

可以在设计时或运行时设置 **BuddyControl** 属性。在设计时，**BuddyControl** 属性在 **Properties** 窗口中。在运行时，使用下面的语句设置 **BuddyControl** 属性

```
UpDown1.BuddyControl = Text1
```

将 **AutoBuddy** 属性设置为 **True** 将隐式设置 **BuddyControl** 属性。在这种情况下，**BuddyControl** 属性将自动设置为跳跃次序中上一个控件。

将 **BuddyControl** 属性设置为 **Nothing** 也将自动设置 **AutoBuddy** 属性为 **False**。

**注意：**如果将 **BuddyControl** 属性设置为不能成为伴随控件的控件，则产生一个错误，如内部标签控件。

## 请参阅

**AutoBuddy** 属性，**SyncBuddy** 属性，**BuddyProperty** 属性。

# BuddyProperty 属性

设置或返回一个属性，用于同步 UpDown 控件与伴随控件。

应用于

UpDown 控件。

语法

```
object.BuddyProperty [=value]
```

BuddyProperty 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个 UpDown 控件的对象表达式   |
| <i>value</i>  | 可变量，指定由 <b>BuddyControl</b> 属性指定的控件的一个属性。如果该属性没有值，则使用控件的缺省属性 |

说明

如果 SyncBuddy 属性是 True，则控件将同步 Value 属性与 BuddyPerperty 属性指定的属性。

在设计时，在 Properties 窗口中可以设置 BuddyProperty 属性。

在设置 BuddyProperty 属性之前，必须先设置 BuddyControl，否则将产生一个错误。

请参阅

AutoBuddy 属性, SyncBuddy 属性, BuddyControl 属性, Value 属性 (UpDown 控件)。

## Change 事件 (UpDown 控件)

当 Value 属性改变时产生该事件。

应用于

UpDown 控件。

语法

Private Sub *object*\_Change ( [*index* As Integer] )

Change 事件的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式  |
| <i>index</i>  | 一个整数，唯一标识控件矩阵中的一个控件 |

说明

当 Value 属性值改变时总是产生 Change 事件。可以通过重新代码或点击箭

头按钮或当 SyncBuddy 属性设置为 True 时通过改变伴随控件的值而改变 Value 属性的值。

请参阅

SyncBuddy 属性， Value 属性（UpDown 控件）， DownClick 事件， UpClick 事件。

## DownClick 事件

当点击下或左箭头按钮时产生该事件。

应用于

UpDown 控件。

语法

Private Sub *object*\_DownClick ( [*index* As Integer] )

DownClick 事件的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式  |
| <i>index</i>  | 一个整数，唯一标识控件矩阵中的一个控件 |

## 说明

在 **Change** 事件之后产生 **DownClick** 事件。

请参阅

**Change** 事件（**UpDown** 控件），**UpClick** 事件。

## Increment 属性

设置或返回一个值，该值确定在点击 **UpDown** 控件的按钮时 **Value** 属性值改变的大小。

应用于

**UpDown** 控件。

语法

*object*.Increment [=value]

**Increment** 属性的语法有如下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个 <b>UpDown</b> 控件的对象表达式                                  |
| <i>value</i>  | 一个长整数，指定 <b>Value</b> 属性值改变的大小。 <b>Value</b> 值不能是负数。缺省值是 1 |



## 说明

**Increment** 属性确定了当你点击 **UpDown** 控件的箭头按钮时 **Value** 属性值改变的大小。点击右或上箭头将使得 **Value** 增加，朝 **Max** 值方向变化；点击下或左箭头将使得 **Value** 值减少，朝 **Min** 值方向变化。

## 请参阅

**Max** 属性，**Min** 属性，**Value** 属性（**UpDown** 控件），**Wrap** 属性，**DownClick** 事件，**UpClick** 事件。

## Max 属性

返回或设置 **UpDown** 控件中滚动范围的最大值。

## 应用于

**UpDown** 控件。

## 语法

*object*.Max [=value]

**Max** 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式            |
| <i>value</i>  | 一个长整数，指定最大值，如下所述。Value 值可以是负数 |

## 说明

按下上或右箭头将导致 UpDown 控件增大 Value 属性值。然而，如果 Max 属性的设置比 Min 属性小，则 UpDown 控件按相反的方向操作。

按下上或右箭头总是导致 Value 属性朝 Max 属性值方向改变。按下下或左箭头总是导致 Value 属性朝 Min 属性值方向改变。

## 请参阅

Increment 属性, Min 属性, Value 属性(UpDown 控件), Wrap 属性, DownClick 事件, UpClick 事件。

## Min 属性

返回或设置 UpDown 控件中滚动范围的最小值。

## 应用于

UpDown 控件。

## 语法

*object.Min [=value]*

Max 属性的语法有如下几个部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式            |
| <i>value</i>  | 一个长整数，指定最小值，如下所述。value 值可以是负数 |

## 说明

按下下或左箭头将导致 UpDown 控件减少 Value 属性值。然而，如果 Min 属性的设置比 Max 属性大，则 UpDown 控件按相反的方向操作。

按下上或右箭头总是导致 Value 属性朝 Max 属性值方向改变。按下下或左箭头总是导致 Value 属性朝 Min 属性值方向改变。

## 请参阅

Increment 属性，Max 属性，Value 属性（UpDown 控件），Wrap 属性，Change 事件（UpDown 控件），DownClick 事件，UpClick 事件。

## Orientation 属性

返回或设置 UpDown 控件中箭头按钮的位置。该属性在运行时是只读的。

应用于  
UpDown 控件。  
语法

*object.Orientation*

Orientation 属性的语法有如下几个部分：

| 部分            | 描述                 |
|---------------|--------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式 |

设置

Orientation 属性的设置如下：

| 常量                       | 值 | 描述           |
|--------------------------|---|--------------|
| cc2orientationVertical   | 0 | （缺省）箭头按钮垂直放置 |
| cc2orientationHorizontal | 1 | 箭头按钮水平放置     |

说明

在设计时，可以在 Properties 窗口或控件的属性对话框中设置该属性值。  
当设置 Orientation 属性时，UpDown 控件将自动定位到伴随控件的旁边。

请参阅

BuddyControl 属性，BuddyProperty 属性。

# SyncBuddy 属性

设置或返回一个值，该值确定 UpDown 控件中的 Value 属性是否与伴随控件中的一个属性同步。

应用于

UpDown 控件。

语法

*object.SyncBuddy [=value]*

SyncBuddy 属性的语法有如下几个部分：

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 一个 UpDown 控件的对象表达式                     |
| <i>value</i>  | 一个布尔表达式，指定伴随控件是否与 UpDown 控件同步，如“设置”中所示 |

设置

value 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | UpDown 控件将 Value 属性值与 BuddyProperty 属性指定的伴随控件中的属性同步。如果 BuddyProperty 属性中没有指定伴随控件的属性，则使用伴随控件的缺省属性 |
| False | （缺省）UpDown 控件不同步 Value 属性与伴随控件中的属性   |

### 说明

使用 SyncBuddy 属性自动同步伴随控件的属性与 UpDown 控件的 Value 属性。例如，你可以同步 UpDown 控件的 Value 属性与 TextBox 控件中的 Text 属性。当 Value 属性改变时，其将更新 TextBox 控件中的 Text 属性，反之亦然。你可以将 UpDown 控件的 Value 属性与其他属性如 Top, Left, BackColor, ForeColor 等属性同步。

### 请参阅

Increment 属性, BuddyControl 属性, Max 属性, Min 属性, Value 属性(UpDown 控件), DownClick 事件, UpClick 事件, TextBox 控件, BuddyProperty 属性, , Left, Top 属性, Text 属性, BackColor, ForeColor 属性 (ActiveX 控件)。

### UpClick 事件

当点击上或右箭头按钮时产生该事件。

应用于

UpDown 控件。

语法

Private Sub *object*\_UpClick ( [*index* As Integer])

UpClick 事件的语法有如下几

个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式  |
| <i>index</i>  | 一个整数，唯一标识控件矩阵中的一个控件 |

说明

在 Change 事件之后产生 UpClick 事件。

请参阅

Change 事件（UpDown 控件）， DownClick 事件。

## Value 属性（UpDown 控件）

设置或返回滚动值的当前位置。

应用于

UpDown 控件。

语法

*object.Value* [=long]

Value 属性的语法有如下几个部分：

| 部分            | 描述                 |
|---------------|--------------------|
| <i>object</i> | 一个 UpDown 控件的对象表达式 |
| <i>long</i>   | 一个长整数，指定当前值，如下所述   |

说明

Value 属性指定了 Min 和 Max 之间的当前值。该属性在点击箭头按钮时增大或减少。Min 和 Max 属性的设置决定了该值是增大还是减少。

如果 SyncBuddy 属性设置为 True，则当 Value 属性这发生改变或 BuddyProperty 属性值发生改变都将进行同步。

请参阅

Increment 属性，SyncBuddy 属性，Max 属性，Min 属性，BuddyProperty 属性。



# Wrap 属性

返回或设置一个值，该值确定在控件的 Value 属性超过 Max 或 Min 值时是否截断数据从头或尾重新开始。

应用于  
UpDown 控件。

## 语法

*object*.Wrap [=*value*]

Wrap 属性的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 UpDown 控件的对象表达式                          |
| <i>value</i>  | 一个布尔表达式，指定 UpDown 控件是否截断 Value 属性值，如“设置”中所示 |

## 设置

value 值的设置如下：

| 设置    | 描述   |
|-------|--|
| True  | 如果用户使用箭头按钮使得 <b>Value</b> 属性值超出了 <b>Max</b> 或 <b>Min</b> 属性值，则截断数据从头或尾重新开始 |
| False | （缺省） <b>UpDown</b> 控件不截断 <b>Value</b> 属性值                                  |

### 说明

如果 **Wrap** 属性设置为 **True**，则 **Value** 属性值将根据 **Min**, **Max** 和 **Increment** 属性值截断。

当 **Value** 属性值截断时，其首先跳到 **Min** 值。例如，如果 **Min** 是 20，**Max** 是 70，**Increment** 是 10，**Value** 是 70，当用户点击上或右箭头时，**Value** 属性值变为 20。

### 请参阅

**Increment** 属性，**Max** 属性，**Min** 属性，**Value** 属性（**UpDown** 控件）。

### 示例

下面的例子使用了 **UpDown** 控件中的 **Wrap** 属性创建了一个可截断的滚动控件，范围从 10 到 70，增量 10。要使用本例，在一个窗体上放置一个 **TextBox** 控件和一个 **UpDown** 控件，增加下面的代码：

```
Private Sub Form_Load()  
    UpDown1.BuddyControl = Text1  
    With UpDown1
```

```
.Min = 10
.Max = 70
.Increment = 10
.Wrap = True
.SyncBuddy = True
End With

' So the TextBox reflects the starting value
Text1.Text = UpDown1.Value
End Sub
```

# Winsock 控件

Winsock 控件对用户是不可视的,可以很容易地访问 TCP 和 UDP 网络服务。其可以被 Microsoft Access, Visual Basic ,Visual C++或 Visual FoxPro 开发人员使用。要编写客户和服务端应用程序,不需要了解 TCP 或调用底层 Winsock API 的具体细节。通过设置 Winsock 控件的属性和调用该控件的方法,可以很容易地连接到远程计算机并进行双向的数据交换。

## TCP 基本知识

TCP (Transfer Control Protocol) 允许你创建并维护一个与远程计算机的连接。使用该连接,两台计算机之间就可以交换数据了。

如果你在创建一个客户应用程序,你必须知道服务器计算机的名字或 IP 地址 (RemoteHost 属性), 以及要监听的端口号 (RemotePort 属性)。然后调用 Connect 方法。

如果你在创建一个服务器应用程序,设置要监听的端口号 (LocalPort 属性), 调用 Listen 方法。当客户建立连接请求时,产生 ConnectionRequest 事件。要完

成该连接，在 `ConnectionRequest` 事件中调用 `Accept` 方法。

一旦建立了连接，两台计算机之间就可以发送和接受数据了。要发送数据，调用 `SendData` 方法。当接受数据时，产生 `DataArrival` 事件。在 `DataArrival` 事件中调用 `GetData` 方法来检取数据。

## UDP 基本知识

UDP (`User Datagram Protocol`) 是无连接的协议。与 TCP 操作不同，计算机并不建立一个连接。并且，UDP 应用程序可以是客户也可以是服务器。

要传输数据，首先设置客户计算机的 `LocalPort` 属性。服务器计算机只需要将 `RemoteHost` 属性设置为客户计算机的 IP 地址，将 `RemotePort` 属性设置为客户计算机上的 `LocalPort` 属性值，然后调用 `SendData` 方法开始发送数据。客户计算机在 `DataArrival` 事件中使用 `GetData` 方法检取数据。

## 属性

`BytesReceived` 属性，`LocalHostName` 属性，`LocalIP` 属性，`LocalPort` 属性，`RemoteHost` 属性 (`ActiveX` 控件)，`SocketHandle` 属性，`State` 属性 (`Winsock` 控件)，`Protocol` 属性 (`Winsock` 控件)，`Name` 属性，`Parent` 属性，`RemoteHost` 属性 (`ActiveX` 控件)，`RemotePort` 属性 (`ActiveX` 控件)，`Index` 属性 (`ActiveX` 控件)，`Tag` 属性 (`ActiveX` 控件)，`Object` 属性 (`ActiveX` 控件)。

## 方法

Accept 方法, Bind 方法, Close 方法 (Winsock 控件), Listen 方法, PeerData 方法, SendData 方法, GetData 方法 (WinSock 控件), GetData 方法 (ActiveX 控件)。

## 事件

Close 事件, ConnectionRequest 事件, DataArrival 事件, SendComplete 事件, SendProgress 事件, Error 事件, Connect 事件 (Winsock 控件), Connect 事件。

## 请参阅

可捕获的 Internet Transfer 控件错误, 使用 Winsock 控件。

## Accept 方法

只对于 TCP 服务器应用程序适用。该方法用于在处理 ConnectionRequest 事件时接受连入请求。

应用于

Winsock 控件。

语法

*object.Accept requestID*

object 是 Winsock 控件的对象表达式。

数据类型

Long

返回值

Void

说明

在 ConnectionRequest 事件中使用 Accept 方法。ConnectionRequest 事件相应的参数 RequestID 也应传递给 Accept 方法。下面是一个例子：

```

Private Sub Winsock1_ConnectionRequest _
    (ByVal requestID As Long)
    ' Close the connection if it is currently open
    ' by testing the State property.
    If Winsock1.State <> sockClosed Then Winsock1.Close

    ' Pass the value of the requestID parameter to the
    ' Accept method.
    Winsock1.Accept requestID
End Sub

```

应在新的控件实例中使用该方法而不是处于监听状态的控件。

请参阅

Connect 方法，ConnnectionRequest 事件。

示例

下面的例子列出了使用 TCP 连接 Winsock 控件必须的代码。使用 RequestID 标识请求。该参数传递给接受请求的 Accept 方法。

```

Private Sub WinsockTCP_ConnectionRequest _
    (requestID As Long)

```



```
If Winsock1.State <> sckClosed Then Winsock1.Close  
Winsock1.Accept requestID  
End Sub
```

## Bind 方法

指定 TCP 连接中使用的 LocalPort 和 LocalIP。如果你有多个协议适配器，使用该方法。

应用于

Winsock 控件。

语法

*object.Bind LocalPort, LocalIP*

Bind 方法的语法有如下几个部分：

| 部分            | 描述                  |
|---------------|---------------------|
| <i>object</i> | 一个 Winsock 控件的对象表达式 |

续表

| 部分               | 描述               |
|------------------|------------------|
| <i>localPort</i> | 建立连接所使用的端口号      |
| <i>localIP</i>   | 建立连接所使用的本地 IP 地址 |

## 说明

在调用 **Listen** 方法之前你必须调用 **Bind** 方法。

## 请参阅

**RemoteHost** 属性（ActiveX 控件），**RemotePort** 属性（ActiveX 控件）。

## BytesReceived 属性

返回接受数据的数量（当前接受缓冲区中的数据）。使用 **GetData** 方法检索数据。

在设计时不可用， 在运行时是只读的。

应用于

Winsock 控件。

语法

*object*.BytesReceived

*object* 是 Winsock 控件的对象表达式。

返回值

Long

请参阅

DataArrival 事件。

## Close 事件

当远程计算机关闭连接时产生该事件。应用程序应使用 **Close** 方法正确地

关闭一个 TCP 连接。

应用于

Winsock 控件。

语法

*object*.Close()

*object* 是 Winsock 控件的对象表达式。

参数

没有

请参阅

Close 方法（Winsock 控件）。

## Close 方法（Winsock 控件）

关闭客户或服务器应用程序的 **TCP** 连接或监听插槽。

应用于

Winsock 控件。

语法

*object*.Close

*object* 是一个 Winsock 控件的对象表达式。

参数

没有

返回值

Void

请参阅

Close 事件。

## Connect 事件（Winsock 控件）

当一个 Connect 操作完成时发生。

应用于

Winsock 控件。

语法

*object*.Connect()

*object* 置换元代表一个对象表达式，其值是一个 Winsock 控件。

说明

使用 Connect 事件确认已经成功建立了。

# Connect 方法

返回与远程计算机的连接。

应用于

Winsock 控件。

语法

*object.Connect remoteHost, remotePort*

Connect 方法的语法有如下几个部分：

| 部分                | 描述                  |
|-------------------|---------------------|
| <i>object</i>     | 一个 Winsock 控件的对象表达式 |
| <i>remoteHos</i>  | 要连接的远程计算机的名字        |
| <i>remotePort</i> | 要连接的远程计算机的端口号       |

返回值

没有

## 说明

当试图建立一个 **TCP** 连接时，你必须调用 **Connect** 方法。

## 请参阅

**Accept** 方法，**ConnectionRequest** 事件，**RemoteHost** 属性（**ActiveX** 控件），**RemotePort** 属性（**ActiveX** 控件）。

## ConnectionRequest 事件

当远程计算机请求一个连接时产生该事件。

一只对于 **TCP** 服务器应用程序适用。当有一个连入请求时就触发该事件。该事件触发之后，**RemoteHostIP** 和 **RemotePort** 属性中保存了客户机的信息。

## 应用于

**Winsock** 控件。



## 语法

*object*\_ConnectionRequest ( *requestID* As Long)

ConnectionRequest 事件的语法有如下几个部分：

| 部分               | 描述                                |
|------------------|-----------------------------------|
| <i>object</i>    | 一个 Winsock 控件的对象表达式               |
| <i>requestID</i> | 连入请求标识符。该参数应传递给第二个控件实例的 Accept 方法 |

## 说明

服务器可以确定是否接受一个连入请求。如果没有接受连入请求，在客户将得到 Close 事件。适用 Accept 方法（在新的控件实例中）接受连入请求。

## 请参阅

Accept 方法，Connect 方法。

## DataArrival 事件

当新数据到达时产生该事件。

应用于

Winsock 控件。

语 法

*object\_DataArrival (bytesTotal As Long)*

DataArrival 事件的语法有如下几个部分：

| 部分                | 描述                  |
|-------------------|---------------------|
| <i>object</i>     | 一个 Winsock 控件的对象表达式 |
| <i>bytesTotal</i> | 长整数。可以检取数据的总数       |

说 明

如果你不在一次 **GetData** 调用中检取所有的数据则不产生该事件。只有当新数据到来时才触发该事件。可以使用 **BytesReceived** 属性检查可检取数据的数量。

请 参 阅

**BytesReceived** 属性，**SendData** 方法，**SendComplete** 事件，**SendProgress** 事

件。

## 示例

下面的例子在 **Winsock** 控件的 **DataArrival** 事件中使用了 **GetData** 方法。当产生该事件时，代码调用 **GetData** 方法检取数据并将其保存在一个字符串中。然后将数据写入一个 **TextBox** 控件。

```
Private Sub Winsock1_DataArrival _  
    (ByVal bytesTotal As Long)  
    Dim strData As String  
    Winsock1.GetData strData, vbString  
    Text1.Text = Text1.Text & strData  
End Sub
```

## Error 事件

后台进程发生错误时产生该事件（如连接失败，后台发送或检取数据失败等）。

应用于

Winsock 控件。

语法

```
object_Error(number As Integer, Description As String, Scode As Long,  
Source As String, HelpFile as String, HelpContext As Long, CancelDisplay As  
Boolean)
```

Error 事件的语法有如下几个部分：

| 部分                   | 描述  |
|----------------------|---|
| <i>object</i>        | 一个 Winsock 控件的对象表达式   |
| <i>number</i>        | 一个整数，指定错误码。请参阅“设置”中的常量                                      |
| <i>description</i>   | 包含错误消息的字符串  |
| <i>scode</i>         | 长 SCODE   |
| <i>source</i>        | 描述错误源的字符串   |
| <i>helpFile</i>      | 包含帮助文件名的字符串   |
| <i>helpContext</i>   | Help 文件环境   |
| <i>cancelDisplay</i> | 指明是否取消该显示动作。缺省是 False，显示缺省的错误消息。如果你不想使用缺省的错误消息，可以将其设置为 True |

## 设置

number 值的设置如下：

| 常量                      | 值     | 描述                     |
|-------------------------|-------|------------------------|
| SckOutOfMemory          | 7     | 内存不足                   |
| SckInvalidPropertyValue | 380   | 属性值无效                  |
| SckGetNotSupported      | 394   | 不能读取属性值                |
| SckSetNotSupported      | 383   | 属性是只读的                 |
| SckBadState             | 40006 | 连接事务或请求的协议或连接状态不正确     |
| SckInvalidArg           | 40014 | 传递给函数的参数格式不正确或范围不对     |
| SckSuccess              | 40017 | 成功                     |
| SckUnsupported          | 40018 | 不支持的变量类型               |
| SckInvalidOp            | 40020 | 对于当前的状态，该操作不正确         |
| SckOutOfRange           | 40021 | 参数超出了范围                |
| SckWrongProtocol        | 40026 | 连接事务或请求的协议不正确          |
| SckOpCanceled           | 1004  | 操作被取消                  |
| SckInvalidArgument      | 10014 | 请求的地址是广播地址，但没有设置标记     |
| SckWouldBlock           | 10035 | 插槽是非阻塞的，指定的操作将被阻塞      |
| SckInProgress           | 10036 | 过程中有阻塞的Winsock操作       |
| SckAlreadyComplete      | 10037 | 操作完成。过程中没有阻塞的Winsock操作 |

续表

| 常量                        | 值     | 描述                    |
|---------------------------|-------|-----------------------|
| SckNotSocket              | 10038 | 描述符不是插槽               |
| SckMsgTooBig              | 10040 | 数据报太长，被截断放在缓冲区中       |
| SckPortNotSupported       | 10043 | 不支持指定的端口              |
| SckAddressInUse           | 10048 | 地址已被使用                |
| SckAddressNotAvailable    | 10049 | 本地计算机上的地址不可用          |
| SckNetworkSubsystemFailed | 10050 | 网络系统故障                |
| SckNetworkUnreachable     | 10051 | 本次该主机访问不到网络           |
| SckNetReset               | 10052 | 当设置了 SO_KEEPAIVE时连接超时 |
| SckConnectAborted         | 11053 | 由于超时或其他失败取消了连接        |
| SckConnectionReset        | 10054 | 远程端重新设置了连接            |
| SckNoBufferSpace          | 10055 | 没有缓冲区空间了              |
| SckAlreadyConnected       | 10056 | 插槽已被连接                |
| SckNotConnected           | 10057 | 插槽没有连接                |
| SckSocketShutdown         | 10058 | 插槽已关闭                 |
| SckTimedout               | 10060 | 插槽已关闭                 |
| 10061                     |       | 连接被强制退出               |
| SckNotInitialized         | 10093 | 应先调用WinsockInit       |
| SckHostNotFound           | 11001 | 验证回答：没有找到主机           |
| SckHostNotFoundTryAgain   | 11002 | 非验证回答：没有找到主机          |

续表

| 常量                     | 值     | 描述                |
|------------------------|-------|-------------------|
| SckNonRecoverableError | 11003 | 不可恢复性错误           |
| SckNoData              | 11004 | 名字有效，所请求的类型没有数据记录 |

## GetData 方法（Winsock 控件）

检取当前的数据块，将其保存在一个 Variant 类型的变量中。

应用于

Winsock 控件。

返回值

Void

语法

*object.GetData data [,type] [,maxLen]*

GetData 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 Winsock 控件的对象表达式   |
| <i>data</i>   | 方法成功返回后保存数据的地方。如果没有足够的空间保存数据，则 <b>data</b> 设置为 <b>Empty</b>                 |
| <i>type</i>   | 可选参数。要检取的数据类型，如“设置”中所示  |
| <i>maxLen</i> | 可选参数。在检取字节矩阵或字符串时指定检取长度。如果没有指定该参数，则检取字节矩阵或字符串中的所有数据。如果数据类型不是字节矩阵或字符串，则忽略该参数 |

## 设置

type 值的设置如下：

| 描述       | 常量         |
|----------|------------|
| Byte     | vbByte     |
| Integer  | vbInteger  |
| Long     | vbLong     |
| Single   | vbSingle   |
| Double   | vbDouble   |
| Currency | vbCurrency |



续表

| 描述      | 常量               |
|---------|------------------|
| Date    | vbDate           |
| Boolean | vbBoolean        |
| SCODE   | vbError          |
| String  | vbString         |
| Byte 矩阵 | vbArray + vbByte |

说明

通常在 `DataArrival` 事件中使用 `GetData` 方法，其包含了一个 `totalBytes` 参数。如果你指定的 `maxLen` 比 `totalBytes` 参数小，则将得到一个 10040 警告，表明其余数据将丢失。

Listen 方法

创建一个插槽，并设置为监听模式。该方法只适用于 `TCP` 连接。

应用于

Winsock 控件。

语法

*object*.Listen

*object* 是一个 Winsock 控件的对象表达式。

参数

没有

返回值

Void

说明

当有连入请求时产生 **ConnectionRequest** 事件。在处理 **ConnectionRequest** 事件时，应用程序应使用 **Accept**（在新的控件实例）方法接受连接。

请参阅

Connect 方法，Close 方法（Winsock 控件）。

## LocalHostName 属性

返回本地计算机的名字。只读，在设计时不可用。

应用于

Winsock 控件。

语法

*object*.LocalHostName

*object* 是一个 Winsock 控件的对象表达式。

返回值

String

## LocalIP 属性

返回本地计算机的 IP 地址。只读属性，在设计时不可用。

应用于

Winsock 控件。

语法

*object*.LocalIP

*object* 是 Winsock 控件的对象表达式。

数据类型

String

## LocalPort 属性

返回或设置本地使用的端口。可读写，在设计时可用。

- 对于客户，这将指定发送数据的端口。如果应用程序不需要特定的端口，指定为 0。在这种情况下，控件将随机选择一个端口。连接建立后，该本地端口就用于 TCP 连接。
- 对于服务器，这是监听的本地端口。如果指定为 0，则随机选用一个端口。在调用了 Listen 方法之后，属性包含了实际选中的端口。

应用于

Winsock 控件。

语法

*object*.LocalPort=*long*

*object* 是一个 Winsock 控件的对象表达式。

数据类型

Long

## 说明

通常使用端口 0 在两台计算机之间动态建立连接。例如，希望服务器回调的客户可以使用端口 0 随机选中一个端口号，该端口号将传递给远程的服务器。

## PeekData 方法

与 **GetData** 方法类似，只是其不从输入队列中删除数据。

应用于

Winsock 控件。

语法

*object.PeekData data [,type] [,maxLen]*

**PeekData** 方法的语法有如下几个部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 一个 Winsock 控件的对象表达式   |
| <i>data</i>   | 方法成功返回后保存数据的地方。如果没有足够的空间保存数据，则 <i>data</i> 设置为 <code>Empty</code>           |
| <i>type</i>   | 可选参数。要检取的数据类型，如“设置”中所示。缺省是 <code>vbArray + vbByte</code>                    |
| <i>maxLen</i> | 可选参数。在检取字节矩阵或字符串时指定检取长度。如果没有指定该参数，则检取字节矩阵或字符串中的所有数据。如果数据类型不是字节矩阵或字符串，则忽略该参数 |

## 设置

type 值的设置如下：

| 描述       | 常量                      |
|----------|-------------------------|
| Byte     | <code>vbByte</code>     |
| Integer  | <code>vbInteger</code>  |
| Long     | <code>vbLong</code>     |
| Single   | <code>vbSingle</code>   |
| Double   | <code>vbDouble</code>   |
| Currency | <code>vbCurrency</code> |

续表

| 描述      | 常量               |
|---------|------------------|
| Date    | vbDate           |
| Boolean | vbBoolean        |
| SCODE   | vbError          |
| String  | vbString         |
| Byte 矩阵 | vbArray + vbByte |

返回值

Void

说明

如果类型指定为 vbString，则数据返回给用户之前转换为 UNICODE。

请参阅

GetData 方法（Winsock 控件）。



# Protocol 属性（Winsock 控件）

返回或设置 Winsock 控件使用的协议，TCP 或 UDP。

应用于

Winsock 控件。

语法

*object*.Protocol [=*protocol*]

*object* 是 Winsock 控件的对象表达式。

设置

protocol 值的设置如下：

| 常量             | 值 | 描述           |
|----------------|---|--------------|
| sckTCPProtocol | 0 | 缺省。使用 TCP 协议 |
| sckUDPProtocol | 1 | 使用 UDP 协议    |

返回值

Void

说明

在重新设置该属性之前必须关闭控件（使用 **Close** 方法）。

## RemoteHostIP 属性

返回远程计算机的 IP 地址。

- 对于客户应用程序，使用 **Connect** 方法建立连接之后，该属性包含了远程计算机 IP 字符串。
- 对于服务器应用程序，进来一个连接请求时，该属性包含了初始化请求的远程计算机的 IP 字符串。
- 在使用 **UDP** 协议时，在 **DataArrival** 事件产生后，该属性包含了发送 **UDP** 数据的远程计算机的 IP 地址。

应用于

Winsock 控件。

语法

*object*.RemoteHostIP

*object* 是 Winsock 控件的对象表达式。

数据类型

String

## SendComplete 事件

当发送操作完成时产生该事件。

应用于

Winsock 控件。

语法

*object*\_SendComplete

*object* 是一个 Winsock 控件的对象表达式。

参数

没有。

请参阅

DataArrival 事件， SendProgress 事件。

## SendData 方法

给远程计算机发送数据。

返回值

Void

应用于

Winsock 控件。

语法

*object.SendData data*

SendData 方法的语法有如下几个部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 一个 Winsock 控件的对象表达式    |
| <i>data</i>   | 要发送的数据。对于二进制数据，应使用字节矩阵 |

说明

当传递 UNICODE 字符串时，在发送之前转换为 ANSI 字符串。

## SendProgress 事件

在发送数据时产生该事件。

应用于

Winsock 控件。

语法

*object*.SendProgress (*bytesSent* As Long, *bytesRemaining* As Long)

SendProgress 事件的语法有如下几个部分：

| 部分                    | 描述                   |
|-----------------------|----------------------|
| <i>object</i>         | 一个 Winsock 控件的对象表达式  |
| <i>bytesSent</i>      | 从上次触发该事件到现在已发送数据的字节数 |
| <i>bytesRemaining</i> | 在发送缓冲区中等待发送数据的字节数    |

请参阅

DataArrival 事件， SendComplete 事件。

## SocketHandle 属性

返回一个值，该值与 **Winsock** 控件用来与 **Winsock** 层通信所用的插槽句柄相对应。该属性是只读的，在设计时不可用。

应用于

**Winsock** 控件。

语法

*object*.SocketHandle

*object* 是一个 **Winsock** 控件的对象表达式。

数据类型

**Long**

说明

该参数将传递给 **Winsock** API。

## State 属性（Winsock 控件）

返回控件的状态，以枚举类型表示。该属性是只读的，在设计时不可用。

应用于

Winsock 控件。

语法

*object.State*

*object* 是一个 Winsock 控件的对象表达式。

数据类型

Integer

设置

State 属性的设置如下：



| 常量                   | 值 | 描述      |
|----------------------|---|---------|
| SckClosed            | 0 | 缺省。关闭   |
| SckOpen              | 1 | 打开      |
| SckListening         | 2 | 监听      |
| SckConnectionPending | 3 | 连接未决    |
| SckResolvingHost     | 4 | 解析主机    |
| SckHostResolved      | 5 | 主机被解析   |
| SckConnecting        | 6 | 连接      |
| SckConnected         | 7 | 已连接     |
| SckClosing           | 8 | 对方在关闭连接 |
| SckError             | 9 | 错误      |

## 附录 A

### Add 方法（Files 集合）

将一个文件名添加到 `DataObject` 对象的 `Files` 集合中。

应用于

`DataObjectFiles` 集合（ActiveX 控件）。

语法

*object.Add (filename, index)*

Add 方法的语法包括以下各个部分：

| 部分              | 描述   |
|-----------------|--|
| <i>object</i>   | 一个对象表达式，其值为“应用于”列表中的一个对象                             |
| <i>filename</i> | 必需的。该字符串设置了文件的名称                                     |
| <i>index</i>    | 可选的。该参数为整数类型，指定将新的成员插到什么位置上。如果没有指定任何索引，该成员将被添加到集合的尾部 |

## 说明

只有当 **DataObject** 对象中包含 **vbCFFiles** 的数据时，**Files** 集合中才能加入文件名（**DataObject** 对象能够包含几种不同类型的数据）。通过对集合进行遍历即可检索文件名的列表。

## Alignment 属性（ActiveX 控件）

返回或设置控件的文本对齐格式。

## 语法

*object*.Alignment [= *integer*]

**Alignment** 属性语法具有下列组成部分：

| 部分             | 描述                       |
|----------------|--------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>integer</i> | 常量或数值，指定对齐类型，“设置值”中有详细描述 |

## 设置

对 **CheckBox** 和 **OptionButton** 控件，**integer** 的设置值如下：

| 常量     | 值 | 描述         |
|--------|---|------------|
| WLeft  | 0 | （缺省值）文本左对齐 |
| WRight | 1 | 文本右对齐      |

对 **TextBox** 控件，**integer** 设置值为：

| 常量            | 值 | 描述         |
|---------------|---|------------|
| WLeftJustify  | 0 | （缺省值）文本左对齐 |
| WRightJustify | 1 | 文本右对齐      |
| WLCenter      | 2 | 文本居中       |

## Appearance 属性（ActiveX 控件）

返回或设置 **MDIForm** 或 **Form** 对象上的控件在设计时的绘图风格。在运行

时是只读的。

应用于

Masked 编辑控件, DataCombo 控件, DataList 控件, MSHFlexGrid 控件, MSFlexGrid 控件, FlatScrollBar 控件, ListView 控件, MonthView 控件, ProgressBar 控件, DataRepeater 控件, RichTextBox 控件, TreeView 控件, Toolbar 控件, DBCombo 控件, DBList 控件, TextBox 控件 (Lightweight)。

语法

*object*.Appearance

object 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

设置

Appearance 属性的设置值是：

| 设置值 | 描述                  |
|-----|---------------------|
| 0   | 平面，绘制控件和窗体没有可视化效果   |
| 1   | （缺省值）3D。带有三维效果的绘制控件 |

## 说明

如果在设计时将其设置为 1，那么 **Appearance** 属性在画出控件时带有三维效果。如果窗体的 **BorderStyle** 属性被设置为固定双边框（**vbFixedDouble**，或 3），窗体的标题和边框也是以有三维效果的方式绘画的。将 **Appearance** 属性设置为 1，也导致窗体及其控件的 **BackColor** 属性被设置为这样的颜色，该颜色是为操作系统控制面板“颜色选项”中的按钮表面颜色选定的。将 **MDIForm** 对象的 **Appearance** 属性设置为 1，只对 MDI 父窗体产生影响。想要在 MDI 子窗体上具有三维效果，必须将每个子窗体的 **Appearance** 属性设置为 1。

## 请参阅

**Masked** 编辑控件，**ListView** 控件，**ProgressBar** 控件，**RichTextBox** 控件，**TreeView** 控件，**ToolBar** 控件，**RemoteData** 控件。

## BackColor、ForeColor 属性（ActiveX 控件）

**BackColor**—返回或设置对象的背景颜色。

**ForeColor**—返回或设置在对象里显示图片和文本的前景颜色。

## 应用于

Masked 编辑控件，Animation 控件，CoolBar 控件，Band 对象，DataCombo 控件，DataList 控件，DateTimePicker 控件，ImageCombo 控件，ImageList 控件，ListView 控件，DataRepeater 控件，RichTextBox 控件，SSTab 控件，TextBox 控件（Lightweight），ListSubItem 对象。

## 语法

*object*.BackColor [= *color*]

*object*.ForeColor [= *color*]

BackColor 和 ForeColor 属性语法包含下面部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>color</i>  | 值或常量，确定对象前景或背景的颜色，“设置值”中有详细说明 |

## 设置

Visual Basic 用 Microsoft Windows 运行环境的红-绿-蓝 (RGB) 颜色方案。*color* 的设置值如下：

| 设置值       | 描述  |
|-----------|---|
| 标准 RGB 颜色 | 使用调色板或在代码中使用 RGB 或 QBColor 函数指定的颜色  |
| 系统缺省颜色    | 由对象浏览器中的 Visual Basic (VB) 对象库所列的系统颜色常量指定的颜色。Windows 运行环境替换使用用户在控制面板设置值中的选择 |

对所有的窗体和控件，在设计时的缺省设置值如下：

**BackColor**—设置为由常量 `vbWindowBackground` 定义的系统缺省颜色。

**ForeColor**—设置为由常量 `vbWindowText` 定义的系统缺省颜色。

## 说明

在 **Label** 和 **Shape**，控件中，如果 **BackStyle** 属性的设置值为 0（透明），则忽略 **BackColor** 属性。

如果在 **Form** 对象或 **Picturebox** 控件中设置 **BackColor** 属性，则所有的文本和图片，包括指定的图片，都被擦除。设置 **ForeColor** 属性值不会影响已经绘出的图片或打印输出。在其它的所有控件中，屏幕的颜色会立即改变。

标准 RGB 颜色的有效取值范围是 0 到 16,777,215 (&HFFFFFF)。该范围内数的高字节为 0；较低的 3 个字节，从最低字节到最高字节依次决定红、绿和蓝的量。红、绿和蓝的成分，分别由一个介于 0 与 255 (&HFF) 之间的数来表示。如果最高字节不为 0，Visual Basic 将使用系统颜色，这些颜色由用户的



控制面板设置值和由对象浏览器中的 **Visual Basic (VB)** 对象库所列出的常量来确定。

在 **Windows** 操作环境中显示文本，文本和背景的颜色必须都是原色。如果所选择的文本或背景颜色没有显示出来，则选择颜色中可能有抖动色——也就是说，最多由三种不同颜色的像素组成的颜色。如果对文本或背景选择了抖动色，则会用最接近的原色来代替。

**注意：** **Animation** 控件只显示两种类型的 **AVI** 文件，压缩或未压缩的 **RLE8** 格式文件。以 **RLE8** 压缩的 **AVI** 文件只显示 8 位颜色。**Animation** 控件的 **BackColor** 属性将“近似”为标准调色板中最近的 8 位颜色。

请参阅

**Masked** 编辑控件，**Animation** 控件，**ListView** 控件，**RichTextBox** 控件，**RemoteData** 控件。

## **Bold** 属性（**Windows** 通用控件）

返回或设置一个值，确定对象文本是否为粗体。

应用于

Font 对象。

语法

*object*.Bold [= *boolean*]

Bold 属性语法有如下几部分：

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，指定文本是否为粗体，如设置值所描述的 |

设置

*boolean* 设置如下：

| 常量    | 描述         |
|-------|------------|
| False | (缺省)文本不是粗体 |
| True  | 文本是粗体      |

请参阅

Name 属性, FontTransparent 属性, Weight 属性, Italic 属性, Size 属性 (Font), StrikeThrough 属性, Underline 属性。

## BorderStyle 属性 (ActiveX 控件)

返回或设置对象的边框样式。Form 对象和 Textbox 控件在运行时是只读的。

应用于

Masked 编辑控件, Multimedia MCI 控件, MSChart 控件, MSHFlexGrid 控件, MSFlexGrid 控件, ListView 控件, MonthView 控件, ProgressBar 控件, DataRepeater 控件, RichTextBox 控件, Slider 控件, TreeView 控件, Toolbar 控件, TextBox 控件 (Lightweight)。

语法

*object*.BorderStyle = [value]

BorderStyle 属性语法有这些组成部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>value</i>  | 值或常量，用于决定边框样式，“设置值”中有详细说明 |

## 设置

Form 对象的 BorderStyle 属性设置值如下：

| 常量                | 设置值 | 描述   |
|-------------------|-----|--|
| VbBSNone          | 0   | 无（没有边框或与边框相关的元素）   |
| VbFixedSingle     | 1   | 固定单边框。可以包含控制菜单框，标题栏，“最大化”按钮，和“最小化”按钮。只有使用最大化和最小化按钮才能改变大小 |
| VbSizable         | 2   | （缺省值）可调整的边框。可以使用设置值 1 列出的任何可选边框元素重新改变尺寸                  |
| VbFixedDouble     | 3   | 固定对话框。可以包含控制菜单框和标题栏，不能包含最大化和最小化按钮，不能改变尺寸                 |
| VbFixedToolWindow | 4   | 固定工具窗口。不能改变尺寸。显示关闭按钮并用缩小的字体显示标题栏。窗体在 Windows 95 的任务条中不显示 |

续表

| 常量                  | 设置值 | 描述   |
|---------------------|-----|--|
| VbSizableToolWindow | 5   | 可变尺寸工具窗口。可变大小。显示关闭按钮并用缩小的字体显示标题栏。窗体在 Windows 95 的任务条中不显示 |

MS Flex Grid、Image、Label、OLE 容器、PictureBox、Frame 和 TextBox 控件的 BorderStyle 属性设置值如下：

| 设置值 | 描述  |
|-----|---|
| 0   | （Image 和 Label 控件的缺省值）无                               |
| 1   | （MS Flex Grid、PictureBox、TextBox 和 OLE 容器控件的缺省值）固定单边框 |

Line 和 Shape 控件的 BorderStyle 属性设置值如下：

| 常量            | 设置值 | 描述                  |
|---------------|-----|---------------------|
| VbTransparent | 0   | 透明                  |
| VbBSSolid     | 1   | （缺省值）实线。边框处于形状边缘的中心 |
| VbBSDash      | 2   | 虚线                  |
| VbBSDot       | 3   | 点线                  |
| VbBSDashDot   | 4   | 点划线                 |

续表

| 常量              | 设置值 | 描述                  |
|-----------------|-----|---------------------|
| VbBSDashDotDot  | 5   | 双点划线                |
| VbBSInsideSolid | 6   | 内收实线。边框的外边界就是形状的外边缘 |

说明

对于窗体，**BorderStyle** 属性决定了其主要特征，这些特征从外观上就能确定窗体是通用窗口或对话框。设置值 3（固定对话框）用于标准对话框。设置值 4（固定工具窗口）和 5（可变工具窗口）用于创建工具箱样式的窗口。

设置值为 2（可变尺寸）的 MDI 子窗体，以 Windows 运行环境运行时定义的缺省尺寸，在 MDI 窗体内显示。对于任何其它设置值，窗体按设计时指定的大小显示。

改变 Form 对象的 **BorderStyle** 属性设置值，可能会改变 **MinButton**、**MaxButton** 和 **ShowInTaskbar** 属性的设置值。当 **BorderStyle** 设置为 1（固定单边框）或 2（可变尺寸）时，**MinButton**、**MaxButton** 和 **ShowInTaskbar** 属性自动设置为 **True**。当 **BorderStyle** 设置为 0（无）、3（固定对话框）、4（固定工具窗口）或 5（可变工具窗口）、**MinButton**、**MaxButton** 和 **ShowInTaskbar** 属性自动设置为 **False**。

注意：如果带有菜单的窗体设置为 3（固定对话框），该窗体将按设置值 1（固定单边框）显示。

运行时，窗体或者是模式或者是无模式，都可以用 Show 方法指定。

## Caption 属性（ActiveX 控件）

- 窗体——确定显示在 Form 或 MDIForm 对象的标题栏中的文本。当窗体为最小化时，该文本被显示在窗体图标下面。
- 控件——确定显示在控件中还是附在控件之后的文本。
- MenuLine 对象——确定为 Menu 控件还是为 MenuItems 集合中的对象显示的文本。

对于 Menu 控件，Caption 在运行时通常是可读/写的。但是对于被 Visual Basic 的加载宏遗弃或提供的菜单项来说，Caption 是只读的，例如 MenuLine 对象。

应用于

DataRepeater 控件，Buttond 对象，SSTab 控件。

## 语法

*object*.Caption [= *string*]

Caption 属性的语法包含下面部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象。如果 <i>object</i> 被省略，那么与活动窗体模块相联系的窗体被认为是 <i>object</i> |
| <i>string</i> | 字符串表达式，其值是被显示为标题的文本   |

## 说明

当创建一个新的对象时，其缺省标题为缺省的 **Name** 属性设置。该缺省标题包括对象名和一个整数，如 **Command1** 或 **Form1**。为了获得一个描述更清楚的标签，应对 **Caption** 属性进行设置。

可以使用 **Caption** 属性赋予控件一个访问键。在标题中，在想要指定为访问键的字符前加一个 (&) 符号。该字符就带有一个下划线。同时按下 **ALT** 键和带下划线的字符就可把焦点移动到那个控件上。为了在标题中加入一个 (&) 符号而不是创建访问键，需要在标题中加入两个即 (&&) 符号。这样，在标题中只有单个 (&) 符号被显示而没有带下划线的字符。

**Label** 控件标题的大小没有限制。对于窗体和所有别的有标题的控件，标



题大小的限制是 255 个字符。

要为窗体显示标题，可以将 **BorderStyle** 属性设为定长单线（1 或 **vbFixedSingle** & ©、复长（2 或 **vbSizable**）、或定长对话框（3 或 **vbFixedDialog**）。超出窗体标题栏的标题将被裁切。当一个 **MDI** 子窗体在一个 **MDIForm** 对象中被最大化时，子窗体的标题将被包括在父窗体的标题中。

**提示：**对于标签来说，将 **AutoSize** 属性设为 **True** 自动调整控件的大小以与其标题相适合。

请参阅

**CaptionStyle** 属性。

## Change 事件（ActiveX 控件）

指示一个控件的内容已经改变的。此事件如何和何时发生则随控件的不同而不同：

- **ComboBox** — 改变控件的文本框部分的正文。该事件仅在 **Style** 属性设置为 0（下拉 **Combo**）或 1（简单 **Combo**）和正文被改变或者通过代码改变了 **Text** 属性的设置时才会发生。

- **DirListBox** — 改变所选择的目录。该事件在双击一个新的目录或通过代码改变 **Path** 属性的设置时发生。
- **DriveListBox** — 改变所选择的驱动器。该事件当选择一个新的驱动器或通过代码改变 **Drive** 属性的设置时发生。
- **HScrollBar** 和 **VScrollBar**（水平和垂直滚动条） — 移动滚动条的滚动框部分。该事件在进行滚动或通过代码改变 **Value** 属性的设置时发生。
- **Label** — 改变 **Label** 的内容。该事件在一个 **DDE** 链接更新数据或通过代码改变 **Caption** 属性的设置时发生。
- **PictureBox** — 改变 **PictureBox** 的内容。该事件当一个 **DDE** 链接更新数据或通过代码改变 **Picture** 属性的设置时发生。
- **TextBox** — 改变文本框的内容。该事件当一个 **DDE** 链接更新数据、用户改变正文或通过代码改变 **Text** 属性的设置时发生。

应用于

**Masked** 编辑控件，**DateTimePicker** 控件，**FlatScrollBar** 控件，**ImageCombo** 控件，**RichTextBox** 控件，**Slider** 控件，**Toolbar** 控件，**UpDown** 控件，**TextBox** 控件（**Lightweight**）。

## 语法

Private Sub *object\_Change*(*index* As Integer)

Change 事件语法包括下列部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 一个整数，用来唯一地标识一个在控件数组中控件   |

## 说明

Change 事件过程可协调在各控件间显示的数据或使它们同步。例如，可用一个滚动条的 Change 事件过程更新一个 TextBox 控件中滚动条的 Value 属性的设置。或者可以利用 Change 事件过程在一个工作区里显示数据和公式，在另一个区域里显示结果。

Change 事件过程在更新系统控件（DirListBox，DriveListBox 和 FileListBox）中的各属性时也是有用的。例如，可更新一个 DirListBox 控件的 Path 属性设置来反映一个 DriveListBox 控件的 Drive 属性设置的改变。

**注意：**一个 Change 事件过程有时会导致一个层叠事件。这种情况在控件的 Change 事件过程改变该控件的内容时会发生，例如，通过用代码设置一个决定该控件的值的属性，如对一个 TextBox 控件的 Text 属

性之类的设置。为了避免层叠事件：

- 如果可能，应避免为控件编写能改变该控件内容的 Change 事件过程。如果编写了那样的过程，应确保设置一个标志用来防止在当前变化进行中更进一步的变化。
- 避免创建两个或两个以上其 Change 事件过程互相影响的控件。例如，两个 TextBox 控件在它们的 Change 事件期间互相更新。
- 避免对 HScrollBar 和 VScrollBar 控件在 Change 事件中使用 MsgBox 函数或语句。

## Checkboxes 属性

返回或设置一个值，确定是否显示复选框。

应用于

TreeView 控件，ListView 控件。

## 语法

*object*.Checkboxes [= *boolean*]

Checkboxes 属性语法有如下几部分:

| 部分             | 描述                         |
|----------------|----------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>boolean</i> | 一个布尔表达式，指定复选框是否显示，如设置值所描述的 |

## 设置

*boolean* 设置如下:

| 常量         | 描述     |
|------------|--------|
| False (缺省) | 复选框不显示 |
| True       | 复选框显示  |

## 请参阅

NodeCheck 事件。

# Checked 属性（Windows 通用控件）

返回或设置一个值，确定某个项目是否被复选（在旁边有一个复选标志）。

应用于

Menu 控件。

语法

object.Checked [= *boolean*]

Checked 属性语法有如下几部分：

| 部分             | 描述                           |
|----------------|------------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象     |
| <i>boolean</i> | 一个布尔表达式，指定一个项目是否被复选，如设置值所描述的 |

设置

boolean 设置如下：

| 常量    | 描述         |
|-------|------------|
| False | (缺省)项目未被选择 |
| True  | 项目被选择      |

## 说明

在设计时，你可以使用菜单编辑器将 **checked** 设置为 **True**。在运行时，你可以在 **Menu** 控件附带的 **Click** 事件过程中切换 **checked** 的开与关。

对于 **Menu** 控件，在运行时 **checked** 通常都是可读/可写的。但对于 **Visual Basic** 做为附加项所支持或输出的菜单项，如 **Add-Ins** 菜单上的 **Add-In Manager** 命令，**Checked** 是只读的。

## 请参阅

**Enabled** 属性，**Load** 事件，**Visible** 属性。

## Clear 方法（ActiveX 控件）

删除集合中的全部对象。

应用于

Bands 集合, DataObject 对象 (ActiveX 控件), DataObjectFiles 集合 (ActiveX 控件), ComboItems 集合, ListItem 对象, ListItems 集合, DataRepeater 控件, RepeaterBindings 集合, OLEObjects 集合, OLEObject 对象, Slider 控件, Panels 集合, Node 对象, Nodes 集合, ButtonMenus 集合, ColumnHeader 对象, ColumnHeaders 集合, Panel 对象, Tab 对象, Tabs 集合, Button 对象, WebItemProperties 对象。

语法

*object*.Clear

*Object* 置换元表示对象表达式, 其值是“应用于”列表中的一个对象。

说明

为了从集合中只删除一个对象, 使用 **Remove** 方法。

请参阅

**Remove** 方法 (ActiveX 控件)。



## Click 事件（ActiveX 控件）

此事件是在一个对象上按下然后释放一个鼠标按钮时发生。它也会发生在一个控件的值改变时。

对一个 Form 对象来说，该事件是在单击一个空白区或一个无效控件时发生。对一个控件来说，这类事件的发生是当：

- 用鼠标的左键或右键单击一个控件。对 CheckBox, CommandButton, Listbox 或 OptionButton 控件来说，Click 事件仅当单击鼠标左键时发生。
- 通过按下箭头键或者单击鼠标按钮，对 ComboBox 或 ListBox 控件中的项目进行选择。
- 当 CommandButton, OptionButton 或 CheckBox 控件具有焦点时，按下 SPACEBAR 键。
- 当窗体带有其 Default 属性设置为 True 的 CommandButton 控件时，按下 ENTER 键。
- 当窗体带有一个 Cancel 按钮 — 其 Cancel 属性设置为 True 的 CommandButton 控件时，按下 ESC 键。
- 对控件按下一个访问键。例如，如果一个 CommandButton 控件的标题是 "&Go"，则按下 ALT+G 键可触发该事件。

也可在代码中触发 Click 事件，通过：

- 将一个 CommandButton 控件的 Value 属性设置为 True。
- 将一个 OptionButton 控件的 Value 属性设置为 True。
- 改变一个 CheckBox 控件的 Value 属性的设置。

应用于

ImageCombo 控件，ListView 控件，ProgressBar 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，DateTimePicker 控件，Animation 控件，CoolBar 控件，DataRepeater 控件，DataList 控件，DBCombo 控件，DBList 控件，TextBox 控件（Lightweight），MSChart 对象，MSHFlexGrid 控件，MSFlexGrid 控件，RichTextBox 控件。

语法

```
Private Sub Form_Click( )
```

```
Private Sub object_Click([index As Integer])
```

Click 事件的语法包括下列部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>index</i>  | 一个整数，用来唯一地标识一个在控件数组中的控件  |

## 说明

通常，将一个 Click 事件过程附加到一个 CommandButton 控件，Menu 对象或 PictureBox 控件上用来执行命令或类似命令的操作。对其它可应用的控件来说，使用这个事件来触发一个动作以响应控件中的变化。

可用一个控件的 Value 属性从代码中测试该控件的状态。单击一个控件除了产生 Click 事件以外还可产生 MouseDown 和 MouseUp 事件。这三种事件发生的顺序根据控件的不同而不同。例如，对 ListBox 和 CommandButton 控件来说，这些事件按下列顺序发生：MouseDown、Click、MouseUp。但对 FileListBox、Label 或 PictureBox 控件来说，这些事件按下列顺序发生：MouseDown、MouseUp 和 Click。当给这些相关的事件附加事件过程时，要确保它们的操作不互相冲突。如果在应用程序中事件发生的顺序是重要的，则应对控件进行测试以确定事件的顺序。

**注意：**为区别鼠标的左、中、右按钮，应使用 MouseDown 和 MouseUp 事件。

如果在 Click 事件中有代码，则 DblClick 事件将永远不会被触发，因为

Click 事件是两个事件中首先被触发的事件。其结果是鼠标单击被 Click 事件截断，从而使 DblClick 事件不会发生。

## Count 属性（ActiveX 控件）

返回集合中对象的数目。

应用于

Node 对象，Nodes 集合，ButtonMenus 集合，ComboItems 集合，ListImage 对象，ListImages 集合，ColumnHeader 对象，ColumnHeaders 集合，ListItem 对象，ListItems 集合，Panels 集合，Tabs 集合，Buttons 集合，Bands 集合，RepeaterBindings 集合，DataPoints 集合，Labels 集合，LightSources 集合。

语法

*object.Count*

object 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

## 说明

能够与 **For...Next** 语句一起使用该属性对集合中的窗体或控件上执行操作。例如，下面的代码将一个窗体上的所有控件向右移动 0.5 英寸（**ScaleMode** 属性设置为 1 或 **vbTwips**）：

```
For I = 0 To Form1. Controls.Count - 1
    Form1. Controls (I).Left=Form1. Controls(I).Left + 720
Next I
```

也可以使用这种结构快速地将窗体中的所有控件有效或失效。

当与 **If TypeOf** 语句一起使用时，可以经过所有的控件进行循环并进行改变，例如，仅对文本框的 **Enabled** 属性设置或仅对选项按钮的 **BackColor** 属性的设置进行处理。

## **DataChanged** 属性（ActiveX 控件）

返回或设置一个值，它指出被绑定的控件中的数据已被某进程改变，这个进程不是从当前记录中检索数据的进程。该属性在设计时不可用。

应用于

RepeaterBinding 对象。

语法

*object.DataChanged [= value]*

DataChanged 属性的语法具有这些部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象          |
| <i>value</i>  | 如“设置值”中所述，一个用于指示数据是否被改变的布尔表达式 |

设置

value 的设置值为：

| 设置    | 描述                            |
|-------|-------------------------------|
| True  | 当前控件中的数据与当前记录中的数据不同           |
| False | （缺省设置）如果当前控件中有数据，则与当前记录中的数据相同 |

## 说明

当 **data** 控件从一个记录移动到另一个记录时，它将数据从当前记录中的字段传递到连接在特定字段上或整个记录上的控件。当数据在被绑定的控件中显示时，**DataChanged** 属性设置为 **False**。如果用户或其它操作改变了被绑定的控件中的值，则 **DataChanged** 属性被设置为 **True**。只移动到另一个记录并不影响 **DataChanged** 属性的值。

当 **data** 控件开始移动到一个不同记录时，**Validate** 事件将发生。如果任何被绑定的控件的 **DataChanged** 属性均为 **True**，则 **data** 控件自动地调用 **Edit** 和 **Update** 方法将改变内容发送到数据库中。

如果不希望被绑定的控件将改变的内容保存在数据库中，则可以在 **Validate** 事件中将 **DataChanged** 属性设置为 **False**。

检查控件的 **Change** 事件代码中 **DataChanged** 属性的值，以避免层叠事件发生。将这一点应用到被连接的和未被绑定的控件上。

## 数据类型

Integer (Boolean)

# DataMember 属性（ActiveX 控件）

在数据提供程序提供的几个数据成员中，返回或者设置特定的一个。

应用于

MSChart 控件，MSHFlexGrid 控件，DataRepeater 控件。

语法

*object.DataMember* [= *string*]

| 部分            | 描述                           |
|---------------|------------------------------|
| <i>object</i> | 必需的。一个对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | 数据成员的名称                      |

说明

一个数据提供程序可以拥有多个数据集合，数据使用者可以选择其中的一个用于绑定。每个数据集合被称为一个“数据成员”，并且用一个唯一的字符串予以标识。



例如，如果所使用的 **Data Environment** 中包含了若干个作为 **DataSource** 的 **Command** 对象，**DataMember** 可以指定应该使用哪一个 **Command** 对象。

如果使用类模块或者用户控件作为数据源，程序中应该保证 **GetDataMember** 能够返回适当的数据成员。该事件的 **DataMember** 参数包含了 **DataMember** 属性的值。通过查询该参数，就可以确定被请求的是哪一个数据成员，并且通过 **Data** 参数传递回合适的数据。

## **DataObject** 对象 （ActiveX 控件）

**DataObject** 对象是数据的容器，这些数据从部件源传送到部件目标，它们存储的格式由使用 **DataObject** 对象的方法定义。

### 语法

**DataObject**

### 说明

映射 **IDataObject** 接口的 **DataObject** 对象，允许实现 OLE 拖放及剪贴板操作。

大部分部件支持人工 OLE 拖放事件，也有一些支持自动 OLE 拖放事件。

## 属性

Files 方法（ActiveX 控件）。

## 方法

Clear 方法（ActiveX 控件），GetData 方法（ActiveX 控件），GetFormat 方法（ActiveX 控件），SetData 方法（ActiveX 控件）。

## 请参阅

DataObjectFiles 集合（ActiveX 控件），DataSource 属性（ActiveX 控件）。

## DataObjectFiles 集合 （ActiveX 控件）

一个集合，其元素代表由 DataObject 对象使用的所有文件名的列表（例如用户在文件资源管理器上拖动来、拖动去的文件名。）

# 语法

*object.DataObjectFiles(index)*

DataObjectFiles 集合的语法具有以下几个部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是一个 DataObject 对象                  |
| <i>index</i>  | 在 0 到 DataObjectFiles.Count - 1 范围内取值的一个整数 |

# 说明

注意：仅当 DataObject 对象中的数据具有 vbCFFiles 格式时，Files 属性才会使用该集合。

Files 属性使用 DataObjectFiles 集合，以存储 DataObject 对象中的文件名。该集合包括 Remove、Add 和 Clear 方法，有了这些方法，就可操作集合中的内容。

# 属性

Count 属性（ActiveX 控件），Item 属性（ActiveX 控件）。

## 方法

Remove 方法（ActiveX 控件），Add 方法（Files 集合），Clear 方法（ActiveX 控件）。

## DataSource 属性 （ActiveX 控件）

返回或设置一个数据源，通过该数据源将一个数据使用者绑定到一个数据库。

## 应用于

MSChart 控件，MSHFlexGrid 控件，MSFlexGrid 控件，DataRepeater 控件。

## 说明

要在运行时将控件绑定到数据库的字段上，必须在设计时使用属性窗口在 DataSource 属性中指定一个 Data 控件。

要完成与 Data 控件所管理的 Recordset 字段的连接，也必须在 DataField 属

性中提供 **Field** 对象的名字。与 **DataField** 属性不同，**DataSource** 属性的设置在运行时不可用。

## 数据类型

**String**

## **DbClick** 事件 （ActiveX 控件）

当在一个对象上按下和释放鼠标按钮并再次按下和释放鼠标按钮时，该事件发生。

对于窗体而言，当双击被禁用的控件或窗体的空白区域时，**DbClick** 事件发生。对于控件而言，**DbClick** 事件在以下情形下发生：

- 用鼠标左键双击控件。
- 双击 **Style** 属性设置为 **1(Simple)** 的 **ComboBox** 控件中的项目，或者在 **FileListBox** 、 **ListBox** 、 **DBCombo** 或 **DBList** 控件中的项目。

## 应用于

TreeView 控件, ImageCombo 控件, ListView 控件, StatusBar 控件, Toolbar 控件, DateTimePicker 控件, Animation 控件, CoolBar 控件, TextBox 控件 (Lightweight), MSChart 对象, SSTab 控件, RichTextBox 控件, DataRepeater 控件, DataList 控件。

## 语法

```
Private Sub Form_DblClick ( )
```

```
Private Sub object_DblClick (index As Integer)
```

| 部分            | 描述                                     |
|---------------|--|
| <i>object</i> | 对象表达式, 其值是“应用于”列表中的一个对象                |
| <i>index</i>  | 如果控件在控件数组内, 则这个 <i>index</i> 值就用来标识该控件 |

## 说明

如果控件在一个控件数组内, 则这个 **Index** 参数唯一地标识这个控件。可以使用 **DblClick** 事件过程执行一个隐式操作, 如双击图标来打开一个窗口或文档。也可用这类过程执行单一操作的多个步骤, 例如用双击在列表框中选定一项并关闭对话框。

要在 Visual Basic 中产生这类快捷效果，可以使用带有缺省按钮的列表框的或叠层排列文件列表框的 **DblClick** 事件过程；所谓缺省按钮就是 **Default** 属性设置为 **True** 的 **CommandButton** 控件。作为列表框 **DblClick** 事件过程的一部分，只需简单调用缺省按钮的 **Click** 事件。

对于那些接收 **Mouse** 事件的对象，事件按这样的次序发生：**MouseDown**，**MouseUp**，**Click**，**DblClick** 和 **MouseUp**。

如果 **DblClick** 在系统双击时间限制内没有出现，则对象识别另一个 **Click** 事件。双击时间限制可以变化，因为用户可在控制面板设置双击速度。当与这些相关事件过程打交道时，必须确保它们的活动不发生冲突。不接受 **DblClick** 事件的控件可能接受两次单击而不是 **DblClick** 事件。

**注意：**要想区别鼠标的左、右、中按钮，使用 **MouseDown** 和 **MouseUp** 事件。

如果在 **Click** 事件中有编码，**DblClick** 事件将永远不会触发。

请参阅

**Click** 事件，**MouseDown**，**MouseUp** 事件，**Click** 事件（**ActiveX** 控件）。

## 示例

无论是通过单击 **CommandButton** 控件还是双击列表项，本范例将显示 **TextBox** 控件中选定的列表项。要试用此例，将代码粘贴包含 **ListBox** 控件、**TextBox** 控件和 **CommandButton** 控件的 **Form** 对象的声明部分。然后运行此例并单击 **CommandButton** 控件或双击在 **ListBox** 控件中的一项。

```
Private Sub Form_Load ()  
    List1.AddItem "John"    ' 添加列表框项。  
    List1.AddItem "Paul"  
    List1.AddItem "George"  
    List1.AddItem "Ringo"  
End Sub  
  
Private Sub List1_DblClick ()  
    Command1.Value = True    ' 触发 Click 事件。  
End Sub  
  
Private Sub Command1_Click ()  
    Text1.Text = List1.Text    ' 显示选定。  
End Sub
```



## Enabled 属性（ActiveX 控件）

返回或设置一个值，该值用来确定一个窗体或控件是否能够对用户产生的事件作出反应。

### 应用于

TreeView 控件，ListView 控件，ProgressBar 控件，Slider 控件，StatusBar 控件，Panel 对象，TabStrip 控件，ToolBar 控件，Button 对象，DateTimePicker 控件，Animation 控件，FlatScrollBar 控件，UpDown 控件，CoolBar 控件，DBCombo 控件，DBList 控件，TextBox 控件（Lightweight），MaskedEdit 控件，Multimedia MCI 控件，MSChart 对象，MSHFlexGrid 控件，MSFlexGrid 控件，SSTab 控件，RichTextBox 控件。

### 语法

`object.Enabled [= boolean]`

Enabled 属性的语法包含下面部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象。如果 <i>object</i> 被省略，则与活动窗体模块相联系的窗体被认为是 <i>object</i> |
| <i>boolean</i> | 一个用来指定 <i>object</i> 是否能够对用户产生的事件作出反应的布尔表达式                                  |

## 设置

*boolean* 的设置为

| 设置    | 描述                           |
|-------|------------------------------|
| True  | （缺省）允许 <i>object</i> 对事件作出反应 |
| False | 阻止 <i>object</i> 对事件作出反应     |

## 说明

**Enabled** 属性允许在运行时使窗体和控件成为有效或无效。例如，可以使对象成为不能用于应用程序的当前状态的无效状态。也可以使之纯粹用来显示，比如一个提供只读信息的文本框的控件无效。

通过把 **Enabled** 设置为 **False** 来使 **Timer** 控件成为无效，将取消由控件的 **Interval** 属性所建立的倒计时。

对于 **Menu** 控件，**Enabled** 在运行时可正常地读/写。但是对于那些被 **Visual**

Basic 的加载宏遗弃或提供的菜单项来说，Enabled 是只读的，例如在“外接程序”菜单中的“外接程序管理器”命令。

## FetchVerbs 方法（ActiveX 控件）

更新对象支持的谓词列表。

应用于

OLEObject 对象，OLE Container 控件。

语法

*object*.FetchVerbs

object 是一个对象表达式，其值是“应用于”列表中的一个对象。

说明

可用 ObjectVerbs 属性读取更新的谓词列表。

# Files 方法 （ActiveX 控件）

返回一个文件名的集合，这些文件名是由 `vbCFFiles` 格式（一个 `DataObjectFiles` 集合）来使用的，该格式反过来又包含 `DataObject` 对象使用的所有文件名的列表；例如，被用户拖动到 Windows“文件管理器”的文件名或从该处拖出来的文件名。

应用于

`DataObject` 对象（ActiveX 控件）。

语法

*object.Files(index)*

Files 集合的语法有下面这些部分：

| 部分            | 描述                                   |
|---------------|--------------------------------------|
| <i>object</i> | 对象表达式，其值是 <code>DataObject</code> 对象 |
| <i>index</i>  | 整数，该整数是某个文件名数组的索引                    |

## 说明

仅当 **DataObject** 对象包含类型为 **vbCFFiles** 的数据时，才可用文件名来对 **Files** 集合进行填充。**DataObject** 对象可以包含几个不同的数据类型。通过遍历该集合就可以获取文件名列表。

可对 **Files** 集合进行填充，所以 **Visual Basic** 应用程序可作为文件列表的拖放源。

## Font 属性（ActiveX 控件）

返回一个 **Font** 对象。

## 应用于

**TreeView** 控件，**ListView** 控件，**StatusBar** 控件，**TabStrip** 控件，**DateTimePicker** 控件，**DataRepeater** 控件，**DataCombo** 控件，**DataList** 控件，**TextBox** 控件（**Lightweight**），**Masked Edit** 控件，**AxisTitle** 对象，**DataPointLabel** 对象，**Footnote** 对象，**Label** 对象，**Legend** 对象，**Title** 对象，**SSTab** 控件，**RichTextBox** 控件。

## 语法

*object*.Font

**object** 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

## 说明

为了标识一个具体的要使用其属性的 **Font** 对象应使用一个对象的 **Font** 属性。例如，下面的代码将改变一个 **Font** 对象的 **Bold** 属性设置，该 **Font** 对象被 **TextBox** 对象的 **Font** 属性所标识：

```
txtFirstName.Font.Bold = True
```

## FontName 属性（ActiveX 控件）

返回或设置在控件中或在运行时画图或打印操作中，显示文本所用的字体。

**注意：**包含 **FontName** 属性是为了和 **CommonDialog** 控件一起使用，以及与先前的 **Visual Basic** 版本兼容。对于其它功能，请使用新的 **Font** 对象属性（对 **CommonDialog** 控件不可用）。

应用于

CommonDialog 控件。

语法

*object*.FontName [= *font*]

FontName 属性语法包括下列组成部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>font</i>   | 字符串表达式，指定所用的字体名        |

说明

该属性的缺省值取决于系统，Visual Basic 中可用的字体取决于系统的配置、显示设备和打印设备。与字体相关的属性只能设置为真正存在的字体的值。

一般来说，用 FontSize、FontBold、FontItalic、FontStrikethru 和 FontUnderline 属性来设置大小和样式属性前，要先改变 FontName 属性。

**注意：**在运行时，可以用 FontCount 和 Fonts 属性获得系统可用字体的信息。

# FontSize 属性 （ActiveX 控件）

返回或设置在控件中或在运行画图或打印操作中，显示文本所用的字体的大小。

注意：包含 FontSize 属性是为了和 CommonDialog 控件一起使用，以及与以前的 Visual Basic 版本兼容。对于其它功能，请使用新的 Font 对象属性（对 CommonDialog 控件不可用）。

应用于

CommonDialog 控件。

语法

*object*.FontSize [= *points*]

FontSize 属性语法包含下面部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>Points</i> | 数值表达式，用磅为单位指定所用字体的大小   |



## 说明

用该属性以所要的字体格式化文本。缺省值由系统决定。要改变缺省值，以磅为单位指定字体尺寸。

**FontSize** 的最大值为 2160 磅。

**注意：**Visual Basic 中可用的字体取决于系统的配置、显示设备和打印设备。与字体相关的属性只能设置为真正存在的字体值。

一般来说，用 **FontSize**、**FontBold**、**FontItalic**、**FontStrikethru** 和 **FontUnderline** 属性来设置大小和样式属性前，应该先改变 **FontName** 属性。然而，在设置 **TrueType** 字体尺寸小于 8 磅时，应用 **FontSize** 属性来设置字体大小，然后设置 **FontName** 属性，用 **FontSize** 属性再一次设置字体大小。Microsoft Windows 运行环境对于小于 8 磅的 **TrueType** 字体使用不同的字体。

## FullRowSelect 属性

返回或设置一个值，指定是否选择整行。

应用于

TreeView 控件，ListView 控件。

语法

*object*.FullRowSelect [= *boolean*]

FullRowSelect 属性语法有如下几部分:

| 部分             | 描述                        |
|----------------|---------------------------|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>boolean</i> | 一个布尔表达式，指定是否选择整行，如设置值所描述的 |

设置

*boolean* 设置如下:

| 常量    | 描述        |
|-------|-----------|
| False | (缺省)不选择整行 |
| True  | 选择整行      |

## 说明

应用于 ListView 控件时, 仅当 View 属性设置为 lvwReport 时, FullRowSelect 属性才有效。

## 请参阅

View 属性 (ListView 控件)。

## GetData 方法 (ActiveX 控件)

用于从 Clipboard 对象返回一个图形。不支持命名参数。

## 应用于

DataObject 对象 (ActiveX 控件), Winsock 控件。

## 语法

*object.GetData (format)*

GetData 方法的语法包含下列部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 必需的。一个对象表达式，该对象一定能在“应用于”列表中找到  |
| <i>format</i> | 可选的。一个常量或数值，如设置中所述，它指定 Clipboard 图形的格式。必须用括号将该常量或数值括起来。如果 format 为 0 或省略，GetData 自动使用适当的格式 |

## 设置

format 的设置有：

| 常量           | 数值 | 描述           |
|--------------|----|--------------|
| vbCFBitmap   | 2  | 位图（.bmp 文件）  |
| vbCFMetafile | 3  | 元文件（.wmf 文件） |
| vbCFDIB      | 8  | 设备无关位图（DIB）  |
| vbCFPalette  | 9  | 调色板          |

## 说明

上述常量在 Visual Basic (VB) 对象浏览器中的对象库里列出。

如果 Clipboard 对象没有与期望的格式相匹配的图形，则返回空。如果 Clipboard 对象中只有一个调色版，则创建最小尺寸 (1x1) 的 DIB。

# GetFormat 方法 （ActiveX 控件）

返回一个整数，指出 Clipboard 对象中的项目是否匹配期望的格式。不支持命名参数。

应用于

DataObject 对象（ActiveX 控件）。

语法

*object*.GetFormat (*format*)

GetFormat 方法的语法包含下列部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 必需的。一个对象表达式，该对象一定能在“应用于”列表中找到                        |
| <i>format</i> | 必需的。一个数值或常量，如设置中所述，它指定 Clipboard 对象的格式。必须用括号包括该常量或数值 |

## 设置

用于 format 的设置有：

| 常量           | 数值     | 描述           |
|--------------|--------|--------------|
| vbCFLink     | &HBF00 | DDE 对话信息     |
| vbCFText     | 1      | 文本           |
| vbCFBitmap   | 2      | 位图（.bmp 文件）  |
| vbCFMetafile | 3      | 元文件（.wmf 文件） |
| vbCFDIB      | 8      | 设备无关位图（DIB）  |
| vbCFPalette  | 9      | 调色板          |

## 说明

上述常量在 Visual Basic (VB) 对象浏览器中的对象库里列出。

如果 Clipboard 对象中一个项目匹配指定的格式，则 GetFormat 方法返回 True。否则，返回 False。

对于 vbCFDIB 和 vbCFBitmap 两种格式，显示图形时不管 Clipboard 中是什么样的调色板都要使用。

## hDC 属性（ActiveX 控件）

返回一个句柄，该句柄是由 Microsoft Windows 运行环境提供给一个对象的设备描述体。

应用于

CommonDialog 控件（Print 对话框）。

语法

*object*.hDC

*object* 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

说明

该属性是 Windows 运行环境的设备描述体句柄。Windows 运行环境，通过给 Grid 对象和应用程序中每个 Grid 和 Grid 控件分配一个设备描述体，管理系统显示。可以用 Print( ) 属性引用对象的设备描述体句柄。这提供了一个传递给 Windows API 调用的值。

对于 Printer 窗体 Picture 控件，在设置 cdlReturnDC 标志时，该属性为打印机对话框中选择的打印机，返回一设备描述体，或设置 cdlReturnIC 标志时，返回一信息关联。

**注意：**hDC 属性值可以在程序运行中改变，因此不要将该值存储在变量中，应在每次需要时使用 hDC 属性。

AutoRedraw 属性能引起 hDC 属性改变。如果窗体或 窗体 PictureBox 容器的 AutoRedraw 属性设置为 True，hDC 将作为持久图形 的设备描述体句柄（等价于 Image 属性）。当 AutoRedraw 属性为 False 时，hDC 属性是窗体窗口或 PictureBox 容器的真正的 hDC 值。程序运行中，不论 AutoRedraw 设置为何值，hDC 属性都可以改变。

请参阅

hWnd 属性（ActiveX 控件），Autodraw 属性，Icon 属性。

示例

这个例子画一个三角形，然后使用 Microsoft Windows 的函数用颜色填充该三角形。要试用此例，先用“工程”菜单中的“添加模块”命令创建一个新的模块。把 Declare 语句粘贴到新模块的声明部分，确保该语句在一行，并且没有断点或隐藏字。然后把 Sub 过程粘贴到窗体的声明部分。按 F5 键，并单



击窗体。

声明 Windows 例程。该语句是模块的。

```
Declare Sub FloodFill Lib "GDI32" Alias "FloodFill" _  
    (ByVal hDC As Long, ByVal X As Long, ByVal Y As _  
    Long, ByVal crColor As Long) As Long
```

' 将下列代码放入窗体。

```
Private Sub Form_Click ()
```

```
    ScaleMode = vbPixels          ' Windows 用像素画。
```

```
    ForeColor = vbBlack           ' 设置画的线为黑色。
```

```
    Line (100, 50)-(300, 50) ' 画一个三角形。
```

```
    Line -(200, 200)
```

```
    Line -(100, 50)
```

```
    FillStyle = vbFSSolid         ' 设置 FillStyle 为实线。
```

```
    FillColor = RGB(128, 128, 255) ' 设置 FillColor。
```

```
    ' 调用 Windows API 填充。
```

```
    FloodFill hDC, 200, 100, ForeColor
```

```
End Sub
```

## Height、Width 属性（ActiveX 控件）

返回或设置对象的维数、或 DataGrid 控件 Columns 对象的宽度。对于 Printer 和 Screen 对象，在设计时不可用。

应用于

TreeView 控件，ImageCombo 控件，ListView 控件，ColumnHeader 对象，ColumnHeaders 集合，ListItem 对象，ListItems 集合，ProgressBar 控件，Slider 控件，StatusBar 控件，Panel 对象，TabStrip 控件，Tab 对象，Toolbar 控件，Button 对象，DateTimePicker 控件，Animation 控件，FlatScrollBar 对象，UpDown 控件，Band 对象，MaskedEdit 控件，Multimedia MCI 控件，Frame 对象，MSChart 对象，Pen 对象，StatLine 对象，Wall 对象，MSHFlexGrid 控件，MSFlexGrid 控件，SSTab 控件，PictureClip 控件，RichTextBox 控件。

语法

*object.Height* [= *number*]

*object.Width* [= *number*]

Height 和 Width 属性语法包含下面部分：

| 部分            | 描述                        |
|---------------|---------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象    |
| <i>number</i> | 数值表达式，指定对象的维数，“设置值”中有详细说明 |

## 设置

大小如下计算：

- **Form** 一窗体的外部高度和宽度，包括边框和标题栏。
- **Control** 一从控件边框的中心度量，以使边框宽度不同的控件能够正确对齐。这些属性使用控件容器的度量单位。
- **Printer** 对象一为打印设备设置的纸张物理尺寸，在设计时无效。如果在运行时设置该属性，则使用这些属性的值而不用 **PaperSize** 属性的设置。
- **Screen** 对象一屏幕的高度和宽度；在设计时无效，在运行时为只读。
- **Picture** 对象一以 **HiMetric** 为单位的图片的高度和宽度。

## 说明

对于 **Form**、**Printer** 和 **Screen** 对象，这些属性值是以缇来度量的。对于窗体或控件，这些属性值随着用户或调整对象大小而改变。所有对象的这些属性的最大值与系统有关。

对不允许设置这些属性的打印机驱动程序，若设置 **Height** 和 **Width** 属性，不会发生错误但纸张的大小保持不变。对只允许某些值的打印机驱动程序，设置 **Height** 和 **Width** 属性时，不会发生错误，且该属性被设置为驱动程序所允许的值。例如，可以将 **Height** 设置成 150 但驱动程序可能会把它设置成 144。

对基于对象全区代作或计算，如改变对象大小或移动对象，要使用 **Height**、**Width**、**Left** 和 **Top** 属性。对基于对象内部区域的操作或计算，如在一对象内绘制或移动对象，要使用 **ScaleLeft**、**ScaleTop**、**ScaleHeight** 和 **ScaleWidth** 属性。

**注意：**DriveListBox 控件或 ComboBox 控件的 **Height** 属性不能改变，这两控件的 **Style** 属性设置为 0（下拉组合框）或 2（下拉列表框）。

对于 DataGrid 控件的 Columns 对象，**Width** 按包含 DataGrid 的对象的度量单位来指定。**Width** 的缺省值为 DataGrid 的 **DefColWidth** 属性值。

对于 Picture 对象，用 **ScaleX** 和 **ScaleY** 方法将 HiMetric 单位转换为所需的单位。

请参阅

**View** 属性（ListView 控件），**ClientHeight**、**ClientWidth**、**ClientLeft**、**ClientTop** 属性，**Move** 方法，**ScaleX**、**ScaleY** 方法，**PaperSize** 属性，**ScaleHeight**、**ScaleWidth**

属性，ScaleLeft, ScaleTop 属性，Left,Top 属性（ActiveX 控件）。

## 示例

这个例子在窗体被加载时，将窗体的大小设置为屏幕大小的 75%并使窗体居中显示。要尝试这个例子，请将代码粘贴到窗体的声明部分。然后按 F5 键并单击窗体。

```
Private Sub Form_Click ()  
    Width = Screen.Width * .75    ' 设置窗体的宽度。  
    Height = Screen.Height * .75    ' 设置窗体的高度。  
    Left = (Screen.Width - Width) / 2    ' 在水平方向上居中显示。  
    Top = (Screen.Height - Height) / 2    ' 在垂直方向上居中显示。  
End Sub
```

## HideSelection 属性 （ActiveX 控件）

返回一个值，以决定当控件失去焦点时选择文本是否加亮显示。

应用于

Masked Edit 控件, ListView 控件, RichTextBox 控件, TreeView 控件, TextBox 控件 (Lightweight)。

语法

*object*.HideSelection

object 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

返回值

HideSelection 返回值为:

| 值     | 描述                       |
|-------|--------------------------|
| True  | (缺省值) 当控件失去焦点时，选择文本不加亮显示 |
| False | 当控件失去焦点时，选择文本加亮显示        |

说明

该属性用于指示在另一个窗体或对话框拥有焦点时，哪些文本要加亮显示

——例如，可用在拼写检查程序中。

## HotTracking 属性

该属性的返回值确定是否使用灵敏的鼠标高亮度显示功能。

### 语法

*object*.HotTracking

*object* 置换元是一个对象表达式，其值为“应用于”列表中的一个对象。

### 返回值

返回的设置值包括：

| 设置    | 描述  |
|-------|---|
| True  | 热跟踪功能被开启。当鼠标指针经过标头标题的时候，标题将变为高亮度显示              |
| False | （缺省设置）除非单击了鼠标按钮，否则 <b>HeaderItem</b> 对象将不响应鼠标移动 |

## 说明

热跟踪功能的作用是：在鼠标指针经过控件的时候为用户提供反馈信息。如果将 **HotTracking** 设置为 **True**，那么控件将对鼠标的移动作出反映：将鼠标指针下面的标头变为高亮度的。

## 请参阅

**TreeView** 控件, **ListView** 控件, **TabStrip** 控件。

## hWnd 属性（ActiveX 控件）

返回窗体或控件的句柄。

**注意：**OLE 容器控件不支持该属性。

## 应用于

**TreeView** 控件, **ImageCombo** 控件, **ListView** 控件, **ProgressBar** 控件, **Slider** 控件, **StatusBar** 控件, **TabStrip** 控件, **Toolbar** 控件, **DateTimePicker** 控件, **Animation**



控件, FlatScrollBar 控件, UpDown 控件, CoolBar 控件, DataRepeater 控件, DataCombo 控件, DataList 控件, DBCombo 控件, DBList 控件, Masked Edit 控件, MSChart 对象, MSFlexGrid 控件, MSFlexGrid 控件, SSTab 控件, PictureClip 控件, RichTextBox 控件。

## 语法

*object*.hWnd

*object* 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

## 说明

Microsoft Windows 运行环境，通过给应用程序中的每个窗体和控件分配一个句柄（或 hWnd）来标识它们。hWnd 属性用于 Windows API 调用。许多 Windows 运行环境函数需要活动窗口的 hWnd 作为参数。

**注意：**由于该属性值在程序运行时可以改变，绝对不要将 hWnd 存储在变量中。

## Index 属性（ActiveX 控件）

返回或设置数，它唯一指定集合中的对象。

应用于

TreeView 控件，Node 对象，Nodes 集合，ImageCombo 控件，ImageList 控件，ListImage 对象，ListImages 集合，ListView 控件，ColumnHeader 对象，ColumnHeaders 集合，ListItem 对象，ListItems 集合，ProgressBar 控件，Slider 控件，StatusBar 控件，Panel 对象，TabStrip 控件，Tab 对象，Toolbar 控件，Button 对象，DateTimePicker 控件，FlatScrollBar 控件，UpDown 控件，Band 对象，MSComm 控件，RepeaterBinding 对象，Microsoft Internet Transfer 控件，MAPISession 控件，Masked Edit 控件，Multimedia MCI 控件，Brush 对象，Intersection 对象，MSChart 对象，MSHFlexGrid 控件，MSFlexGrid 控件，SSTab 控件，Winsock 控件，PictureClip 控件，RichTextBox 控件，OLEObject 对象，SysInfo 控件。

语法

*object*.Index

对象置换元是对象表达式，其值是“应用于”列表中的对象。

## 说明

按照缺省，**Index** 属性被设置成在集合中创建对象的次序。集合中的第一个对象的索引总是 (1)。

当集合中的对象重编次序时，比如将 **Sorted** 属性 设置成 **True** 时，对象的 **Index** 属性值可以改变希望 **Index** 属性动态地变化，则通过使用 **Key** 属性来引用集合中的对象将会更有益。

## 请参阅

**Index** 属性 (**Brush**)，**Index** 属性 (**Intersection**)，**Load** 语句，**Unload** 语句，**Name** 属性，**Key** 属性 (**ActiveX** 控件)，**Tag** 属性 (**ActiveX** 控件)。

## Item 属性 (ActiveX 控件)

利用位置或键返回 **Collection** 对象的指定成员。

# 应用于

Node 对象，Nodes 集合，ButtonMenus 集合，ComboItems 集合，ListImage 对象，ListImages 集合，ColumnHeader 对象，ColumnHeaders 集合，ListItems 属性（ListView 控件），Panels 集合，Tabs 集合，RepeaterBindings 集合，Splits 集合，SelBookmarks 集合，Series 集合，OLEObject 对象，OLEObjects 集合，DataObjectFiles 集合（ActiveX 控件）。

# 语法

*object.Item(index)*

Item 属性的语法具有下列对象限定符和部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 必要。对象表达式，其值为“应用于”列表中对象  |
| Index         | 必要。为一表达式，指定集合中成员的位置。如果是数值表达式，则 index 必须是从 1 到集合 Count 属性值之间的数值。如果是字符串表达式，则当加入一被引用的成员到集合时，index 必须和 key 参数对应 |

## 说明

如果 **index** 值无法与集合的任何现有成员匹配，就会导致错误发生。

**Item** 是集合的缺省属性。因此，以下两行程序代码等价：

```
Print MyCollection(1)
```

```
Print MyCollection.Item(1)
```

## 请参阅

**Caption** 属性（ActiveX 控件）。

## Key 属性（ActiveX 控件）

返回或设置字符串，在集合中唯一地标识一个成员。

## 应用于

**Node** 对象，**Nodes** 集合，**ListImage** 对象，**ListImages** 集合，**ColumnHeader** 对象，**ColumnHeaders** 集合，**ListItems** 集合，**Panel** 对象，**Tab** 对象，**Button** 对

象，Band 对象，RepeaterBinding 对象，OLEObject 对象。

语法

```
object.Key [= string]
```

Key 属性语法包含下面部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>string</i> | 唯一的字符串，在集合中标识一个成员      |

说明

如果字符串不唯一，将发生错误。

用 Add 方法将对象添加到集合中时，可设置 Key 属性。

当集合中的对象被重新排序时，如在设置 Sorted 属性为 True 时，对象的 Index 属性值会改变。如果想要 Index 属性动态地改变，则使用 Key 属性引用集合中的对象。

另外，通过给集合中的对象赋予有意义的名字，可用 Key 属性使 Visual Basic 工程成为“自编文件”。

## KeyDown、KeyUp 事件（ActiveX 控件）

这些事件是当一个对象具有焦点时按下（**KeyDown**）或松开（**KeyUp**）一个键时发生的（要解释 ANSI 字符集中的字符，应使用 **KeyPress** 事件）。

### 应用于

TreeView 控件，ImageCombo 控件，ListView 控件，Slider 控件，TabStrip 控件，DateTimePicker 控件，FlatScrollBar 控件，DataRepeater 控件，DataList 控件，DBCombo 控件，DBList 控件，TextBox 控件（Lightweight），Masked Edit 控件，MSChart 控件，MSHFlexGrid 控件，MSFlexGrid 控件，SSTab 控件，RichTextBox 控件。

### 语法

```
Private Sub Form_KeyDown(keycode As Integer, shift As Integer)
```

```
Private Sub object_KeyDown([index As Integer,]keycode As Integer, shift As Integer)
```

```
Private Sub Form_KeyUp(keycode As Integer, shift As Integer)
```

```
Private Sub object_KeyUp([index As Integer,]keycode As Integer, shift As Integer)
```

KeyDown 和 KeyUp 事件包括下列部分：

| 部分             | 描述   |
|----------------|--|
| <i>object</i>  | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>index</i>   | 是一个整数，它用来唯一标识一个在控件数组中的控件   |
| <i>keycode</i> | 是一个键代码，诸如 <code>vbKeyF1</code> （F1 键）或 <code>vbKeyHome</code> （HOME 键）。要指定键代码，可使用对象浏览器中的 Visual Basic (VB) 对象库中的常量   |
| <i>shift</i>   | 是在该事件发生时响应 SHIFT、CTRL 和 ALT 键的状态的一个整数。 <code>shift</code> 参数是一个位域，它用最少的位响应 SHIFT 键（位 0）、CTRL 键（位 1）和 ALT 键（位 2）。这些位分别对应于值 1、2 和 4。可通过对一些、所有或无位的设置来指明有一些、所有或零个键被按下。例如，如果 CTRL 和 ALT 这两个键都被按下，则 <code>shift</code> 的值为 6 |

## 说明

对于这两个事件来说，带焦点的对象都接收所有击键。一个窗体只有在不具有可视的和有效的控件时才可以获得焦点。虽然 KeyDown 和 KeyUp 事件可应用于大多数键，它们最经常地还是应用于

- 扩展的字符键如功能键等。
- 定位键。



- 键盘修饰键和按键的组合。
- 区别数字小键盘和常规数字键。

在需要对按下和松开一个键都响应时，可使用 **KeyDown** 和 **KeyUp** 事件过程。

下列情况不能引用 **KeyDown** 和 **KeyUp** 事件：

- 窗体有一个 **CommandButton** 控件，并且 **Default** 属性设置为 **True** 时的 **ENTER** 键。
- 窗体有一个 **CommandButton** 控件，并且 **Cancel** 属性设置为 **True** 时的 **ESC** 键。
- **TAB** 键。

**KeyDown** 和 **KeyUp** 用两种参数解释每个字符的大写形式和小写形式：  
**keycode** —显示物理的键(将 **A** 和 **a** 作为同一个键返回)和 **shift** —显示 **shift** + **key** 键的状态而且返回 **A** 或 **a** 其中之一。

如果需要测试 **shift** 参数，可使用该参数中定义各位的 **shift** 常量。该常量有下列值：

| 常量          | 值 | 描述          |
|-------------|---|-------------|
| VbShiftMask | 1 | SHIFT 键的位屏蔽 |

续表

| 常量         | 值 | 描述         |
|------------|---|------------|
| VbCtrlMask | 2 | CTRL 键的位屏蔽 |
| VbAltMask  | 4 | ALT 键的位屏蔽  |

该常量用作位屏蔽。它可被用来测试任何键组合。

测试一个条件时，首先将每个结果分配给一个临时整数变量，然后将 shift 与一个位屏蔽进行对比。如下例，可用 And 运算符和 shift 参数一起来测试条件是否大于 0。该条件说明该修正键被按下：

```
ShiftDown = (Shift And vbShiftMask) > 0
```

可按此例在一个过程中测试任何条件的组合：

```
If ShiftDown And CtrlDown Then
```

**注意：**如果 KeyPreview 属性被设置为 True，则一个窗体先于该窗体上的控件接收到此事件。可用 KeyPreview 属性来创建全局键盘处理例程。

# KeyPress 事件 （ActiveX 控件）

此事件当用户按下和松开一个 ANSI 字符集中的键时发生。

应用于

TreeView 控件， ImageCombo 控件， ListView 控件， Slider 控件， TabStrip 控件， DateTimePicker 控件， FlatScrollBar 控件， DataRepeater 控件， DataList 控件， DBCombo 控件， DBList 控件， TextBox 控件（Lightweight）， Masked Edit 控件， MSChart 控件， MSHFlexGrid 控件， MSFlexGrid 控件， SSTab 控件， RichTextBox 控件。

语法

Private Sub Form\_KeyPress(*keyascii* As Integer)

Private Sub *object*\_KeyPress(*[index* As Integer,*]keyascii* As Integer)

KeyPress 事件语法包含下列部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象 |

续表

| 部分              | 描述   |
|-----------------|--|
| <i>index</i>    | 一个整数，它用来唯一标识一个在控件数组中的控件  |
| <i>keyascii</i> | 是返回一个标准数字 ANSI 键代码的整数。 <b>Keyascii</b> 通过引用传递,对它进行改变可给对象发送一个不同的字符。将 <b>keyascii</b> 改变为 0 时可取消击键，这样一来对象便接收不到字符 |

说明

具有焦点的对象接收该事件。一个窗体仅在它没有可视和有效的控件或 **KeyPreview** 属性被设置为 **True** 时才能接收该事件。一个 **KeyPress** 事件可以引用任何可打印的键盘字符，一个来自标准字母表的字符或少数几个特殊字符之一的字符与 **CTRL** 键的组合，以及 **ENTER** 或 **BACKSPACE** 键。**KeyPress** 事件过程在截取 **TextBox** 或 **ComboBox** 控件所输入的击键时是非常有用的。它可立即测试击键的有效性或在字符输入时对其进行格式处理。改变 **keyascii** 参数的值会改变所显示的字符。

可使用下列表达式将 **keyascii** 参数转变为一个字符：

**Chr(KeyAscii)**

然后执行字符串操作，并将该字符翻译成一个控件可通过该表达式解释的 ANSI 数字：

`KeyAscii = Asc(char)`

应当使用 `KeyDown` 和 `KeyUp` 事件过程来处理任何不被 `KeyPress` 识别的击键，诸如：功能键、编辑键、定位键以及任何这些键和键盘换档键的组合等。与 `KeyDown` 和 `KeyUp` 事件不同的是，`KeyPress` 不显示键盘的物理状态，而只是传递一个字符。

`KeyPress` 将每个字符的大、小写形式作为不同的键代码解释，即作为两种不同的字符。而 `KeyDown` 和 `KeyUp` 用两种参数解释每个字符的大写形式和小写形式：`keycode` — 显示物理的键（将 `A` 和 `a` 作为同一个键返回）和 `shift` — 指示 `shift + key` 键的状态而且返回 `A` 或 `a` 其中之一。

如果 `KeyPreview` 属性被设置为 `True`，窗体将先于该窗体上的控件接收此事件。可用 `KeyPreview` 属性来创建全局键盘处理例程。

注意：CTRL+@ 组合的 ANSI 编号是 0。因为 Visual Basic 将一个零值的 `keyascii` 识别为一个长度为零的字符串 ("")，在应用程序中应避免使用 CTRL+@ 的组合。

## LargeChange、SmallChange 属性（ActiveX 控件）

- `LargeChange` 属性指定了在按 `PAGEUP` 或 `PAGEDOWN` 键时，或者使

用鼠标单击滚动条的左部或右部时，滚动条移动的刻度数。

- **SmallChange** 属性指定了在按左右箭头键的时候，滚动条移动的刻度数。

应用于

FlatScrollBar 控件，Slider 控件。

语法

*object.LargeChange = number*

*object.SmallChange = number*

LargeChange 和 SmallChange 属性的语法包括下列各个部分：

| 部分            | 描述                          |
|---------------|-----------------------------|
| <i>object</i> | 一个对象表达式，其值是一个 Slider 控件     |
| <i>number</i> | 一个 Long 类型的整数，指定滚动条要移动多少个刻度 |

说明

LargeChange 属性的缺省值为 5。SmallChange 属性的缺省值为 1。

请参阅

Max, Min 属性（ScrollBar），Change 事件，Change 事件（ActiveX 控件），Value 属性（ActiveX 控件），Slider 控件。

## 示例

本示例使 `TextBox` 控件的宽度与 `Slider` 控件的宽度保持一致。当 `Slider` 控件的 `Value` 属性大于特定数值时，`TextBox` 控件的宽度与 `Slider` 控件的值相匹配。`SmallChange` 和 `LargeChange` 属性依赖于 `Slider` 控件的 `Max` 属性。如果要试运行该例，请将一个 `Slider` 控件和一个 `TextBox` 控件放到一个窗体上，然后将下面的代码粘贴到窗体的“声明”部分。请运行该示例并按 `PAGEDOWN`、`PAGEUP`、`LEFT` 和 `RIGHT ARROW` 键。

```
Private Sub Form_Load()  
    Text1.Width = 4500 ' 设置 TextBox 的最小宽度。  
    Slider1.Left = Text1.Left ' 使 Slider 与 TextBox 对齐。  
    ' 使 Slider 的宽度与 TextBox 一致。  
    Slider1.Max = Text1.Width  
    ' 将 Slider 放在比文本框稍微靠下的位置。  
    Slider1.Top = Text1.Top + Text1.Height + 50  
    ' 将 TickFrequency 设置为 Max 数值的十分之一。  
    Slider1.TickFrequency = Slider1.Max * 0.1  
    ' 将 LargeChange 和 SmallChange 的值分别设置为 Max 的几分之一。  
    Slider1.LargeChange = Slider1.Max * 0.1  
    Slider1.SmallChange = Slider1.Max * 0.01  
  
End Sub  
Private Sub Slider1_Change()
```

```
' 如果滚动条小于文本框大小的 1/3，不进行任何修改。  
' 否则，使文本框的宽度与 Slider 的数值保持一致。  
If Slider1.Value > Slider1.Max / 3 Then  
    Text1.Width = Slider1.Value  
End If  
End Sub
```

## Left、Top 属性（ActiveX 控件）

- Left—返回或设置对象内部的左边与它的容器的左边之间的距离。
- Top—返回或设置对象的内顶部和它的容器的顶边之间的距离。
- 仅对于 Panel 对象来说，这是一个只读属性。

### 应用于

TreeView 控件，ImageCombo 控件，ListView 控件，Masked Edit 控件，Multimedia MCI 控件，Animation 控件，MSChart 对象，MSChart 控件，CoolBar 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ColumnHeader 对象，ColumnHeaders 集合，ListItem 对象，ListItems 集合，ProgressBar 控件，RichTextBox 控件，Slider



控件, StatusBar 控件, Panel 对象, TabStrip 控件, Tab 对象, Toolbar 控件, Button 对象, TreeView 控件, UpDown 控件, SSTab 控件。

语法

*object*.Left [= *value*]

*object*.Top [= *value*]

Left 和 Top 属性语法包含下面部分：

| 部分            | 描述                     |
|---------------|------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>value</i>  | 数值表达式，用于指定距离           |

说明

对于窗体，Left 和 Top 属性总以缇为单位来表达；对于控件，它们的度量单位决定于它的容器的坐标系统。这些属性值随着用户或程序中移动该对象而改变。对于公共对话框和 Timer 控件，这些属性在运行时无效。

对任一个属性，都可以指定单精度数值。

用 Left、Top、Height 和 Width 属性来完成基于对象外部维数的操作，如

移动或改变尺寸。用 `ScaleLeft`、`ScaleTop`、`ScaleHeight` 和 `ScaleWidth` 来完成基于对象内部尺寸的操作，如绘出或移动包含在该对象中的对象。与比例相关的属性只适用于 `PictureBox` 控件和 `Form` 以及 `Printer` 对象。

请参阅

`Move` 方法，`ScaleHeight`、`ScaleWidth` 属性，`ScaleLeft`、`ScaleTop` 属性，`ScaleMode` 属性，`Height`、`Width` 属性（`ActiveX` 控件），`View` 属性（`ListView` 控件）。

示例

这个例子在窗体被加载时将窗体的大小设置为屏幕大小的 75% 并使窗体居中。要尝试这个例子，请将代码粘贴到窗体的声明部分，然后按 `F5` 键并单击窗体。

```
Private Sub Form_Click ()  
    Width = Screen.Width * .75    ' 设置窗体的宽度。  
    Height = Screen.Height * .75   ' 设置窗体的高度。  
    Left = (Screen.Width - Width) / 2    ' 在水平方向上居中。  
    Top = (Screen.Height - Height) / 2    ' 在垂直方向上居中。  
End Sub
```

## Max、Min 属性（ActiveX 控件）

- ♦ **Max**—返回或设置当滚动框处于底部或最右位置时，一个滚动条位置的 **Value** 属性最大设置值。对于 **ProgressBar** 控件，它返回或设置其最大值。
- ♦ **Min**—返回或设置当滚动框处于顶部或最右位置时，一个滚动条位置的 **Value** 属性最小设置值。对于 **ProgressBar** 控件，它返回或设置其最小值。

### 应用于

UpDown 控件, CommonDialog 控件, Rect 对象, FlatScrollBar 控件, ProgressBar 控件, Slider 控件。

### 语法

*object*.Max [= *value*]

*object*.Min [= *value*]

**Max** 和 **Min** 属性的语法包含下面部分：

| 部分            | 描述                                  |
|---------------|-------------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象              |
| <i>value</i>  | 数值表达式，按照设置值中所述，指定最大或最小的 Value 属性设置值 |

## 设置值

对于每个属性，可指定在 -32,768 和 32,767 范围之间的一个整数，包括 -32,768 和 32,767。缺省设置值为：

Max—32,767

Min—0

## 说明

Microsoft Windows 操作环境自动地将滚动条范围设置为与窗体、ComboBox 控件、和 ListBox 控件的内容成比例。对于一个滚动条（HScrollBar 或 VScrollBar）控件，无论如何，必须指定这些范围。根据滚动条的具体使用情况，例如，作为输入设备或作为一个速度或数量的指示器，使用 Max 和 Min 设置一个合适的范围。

典型地，你在设计时设置 Max 和 Min。如果滚动范围必须动态地改变，

你也能够在运行时在代码中设置它们—例如，当向能够被滚动的数据库增加记录时。使用属性 `LargeChange` 和 `SmallChange`，为一个滚动条控件设置最大和最小滚动增量。

**注意：**如果 `Max` 被设为比 `Min` 小的值，那么最大值将被分别设为水平或垂直滚动条的最左或最上位置处。`ProgressBar` 控件的 `Max` 属性必须总是比其 `Min` 属性大一些，并且其 `Min` 属性必须总是大于或等于 0。

`Max` 和 `Min` 属性定义了控件的范围。缺省 `ProgressBar` 控件的 `Min` 属性是 0 并且其 `Max` 属性设置值为 100，表示操作的百分比持续时间。

请参阅

`LargeChange`, `SmallChange` 属性（ActiveX 控件）。

`MouseDown`、`MouseUp` 事件（ActiveX 控件）

这些事件是当按下（`MouseDown`）或者释放（`MouseUp`）鼠标按钮时发生。

应用于

Animation 控件, MSChart 控件, CoolBar 控件, DataList 控件, DateTimePicker 控件, ListView 控件, ProgressBar 控件, DataRepeater 控件, RichTextBox 控件, Slider 控件, StatusBar 控件, TabStrip 控件。

语法

```
Private Sub Form_MouseDown(button As Integer, shift As Integer, x As Single, y As Single)
```

```
Private Sub MDIForm_MouseDown(button As Integer, shift As Integer, x As Single, y As Single)
```

```
Private Sub object_MouseDown([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)
```

```
Private Sub Form_MouseUp(button As Integer, shift As Integer, x As Single, y As Single)
```

```
Private Sub MDIForm_MouseUp(button As Integer, shift As Integer, x As Single, y As Single)
```

```
Private Sub object _MouseUp([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)
```

MouseDown 和 MouseUp 事件各种语法包含下列部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 返回一个对象表达式，其值是“应用于”列表中的一个对象  |
| <i>index</i>  | 返回一个整数，用来唯一地标识一个在控件数组中的控件   |
| <i>button</i> | 返回一个整数，用来标识该事件的产生是按下（MouseDown）或者释放（MouseUp）按钮引起的。button 参数是具有相应于左按钮（位 0），右按钮（位 1），以及中间按钮（位 2）的一个位字段。这些位的值分别等于 1、2、和 4。其中仅有一位被设置，指示出引起该事件的那个按钮  |
| <i>shift</i>  | 返回一个整数，在 button 参数指定的按钮被按下或者被释放的情况下，该整数相应于 SHIFT、CTRL、和 ALT 键的状态。某键被按下使得一个二进制位被设置。Shift 参数是具有相应于 SHIFT 键（位 0），CTRL 键（位 1），以及 ALT 键（位 2）最少二进制位的一个位字段。这些位的值分别等于 1、2、和 4。shift 参数指示这些键的状态。这些位中可能有一些，全部，或者一个也没有被设置，指示这些键中的一些、全部，或者一个也没有被按下。例如，CTRL 和 ALT 键都被按下，则 shift 的值就是 6 |
| <i>x, y</i>   | 返回一个指定鼠标指针当前位置的数。x 和 y 的值所表示的总是通过该对象 ScaleHeight, ScaleWidth, ScaleLeft, 和 ScaleTop 属性所建立的坐标系统的方式   |

# 说明

为了在给定的一个鼠标按钮按下或释放时指定将引起的一些操作，应当使用 `MouseDown` 或者 `MouseUp` 事件过程。不同于 `Click` 和 `DbClick` 事件的是，`MouseDown` 和 `MouseUp` 事件能够区分出鼠标的左、右、和中间按钮。也可以为使用 `SHIFT`、`CTRL`、和 `ALT` 等键盘换挡键编写用于鼠标—键盘组合操作的代码。

下列情况对 `Click` 和 `DbClick` 事件都适用：

- 如果鼠标按钮是当其指针在窗体或控件之上时被按下，则该对象将“捕获”鼠标并接收包括最后 `MouseUp` 事件在内的全部鼠标事件。这暗示了通过鼠标事件所返回的 `x`，`y` 鼠标指针坐标值，可以不总是在接收它们的对象的内部区域之内。
- 如果鼠标被持续地按下，则第一次按下之后捕获鼠标的对象将接收全部鼠标事件直至所有按钮被释放。

如果要测试 `button` 或 `shift` 参数，可以使用对象浏览器中的 `Visual Basic (VB)` 对象库中所列出的常量，用来定义该参数中的各个二进制位：

| 常量（按钮）                    | 值 | 描述     |
|---------------------------|---|--------|
| <code>vbLeftButton</code> | 1 | 左按钮被按下 |



续表

| 常量（按钮）         | 值 | 描述         |
|----------------|---|------------|
| vbRightButton  | 2 | 右按钮被按下     |
| vbMiddleButton | 4 | 中间按钮被按下    |
| vbShiftMask    | 1 | SHIFT 键被按下 |
| vbCtrlMask     | 2 | CTRL 键被按下  |
| vbAltMask      | 4 | ALT 键被按下   |

随后这些常量作为位屏蔽，对于按钮的各种组合，无须计算各个组合的唯一的位字段即可进行测试。

**注意：**可使用 MouseMove 事件过程对由于鼠标移动而引起的事件进行响应。MouseDown 和 MouseUp 所使用的 button 参数与 MouseMove 所使用的 button 参数是不同的。对于 MouseDown 和 MouseUp 来说，button 参数要精确地指出每个事件的一个按钮，而对于 MouseMove 来说，它指示的是所有按钮的当前状态。

## MouseDown 属性 （ActiveX 控件）

返回或设置自定义的鼠标图标。

# 应用于

Masked 编辑控件，CoolBar 控件，DataCombo 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，DataRepeater 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件。

# 语法

*object*.MouseIcon = LoadPicture(*pathname*)

*object*.MouseIcon [= *picture*]

MouseIcon 属性的语法包含下面部分：

| 部分              | 描述                                      |
|-----------------|---|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的一个对象                  |
| <i>pathname</i> | 字符串表达式，指定包含自定义图标文件的路径和文件名               |
| <i>picture</i>  | 对象、PictureBox 控件、或 Image 控件的 Picture 属性 |

# 说明

MouseIcon 属性提供一个自定义图标，它在 MousePointer 属性设为 99 时

使用。

可以使用 **MouseIcon** 属性来装入光标或者图标文件。**MouseIcon** 属性使程序能够很容易访问自定义光标，它可以是任意大小并具有任何热点位置的光标。**Visual Basic** 不能装入动画光标 (.ani) 文件，即使 32-位版的 **Windows** 支持这些光标。

## MouseMove 事件 （ActiveX 控件）

此事件在移动鼠标时发生。

应用于

**MSChart** 控件，**CoolBar** 控件，**DataList** 控件，**DateTimePicker** 控件，**DataRepeater** 控件，**StatusBar** 控件，**TabStrip** 控件。

语法

```
Private Sub Form_MouseMove(button As Integer, shift As Integer, x As Single, y As Single)
```

```
Private Sub MDIForm_MouseMove(button As Integer, shift As Integer, x As
```

Single, y As Single)

Private Sub *object\_MouseMove*([*index* As Integer,] *button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)

MouseMove 事件语法包含下列部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 一个对象表达式，其值是“应用于”列表中的一个对象   |
| <i>index</i>  | 一个整数，用来唯一地标识一个在控件数组中的控件  |
| <i>button</i> | 一个整数，它对应鼠标各个按钮的状态，如果某个按钮按下，其中就有一个二进制位被设置。 <b>button</b> 参数是具有相应于左按钮（位 0），右按钮（位 1），以及中间按钮（位 2）的一个位字段。这些位的值分别等于 1，2，和 4。它指示这些鼠标按钮的整体状态；三个二进制位中的一些、全部、或一个也没有被设置，指示这些按钮中的一些、全部、或一个也没有被按下   |
| <i>shift</i>  | 一个整数，该整数相应于 <b>SHIFT</b> 、 <b>CTRL</b> 、和 <b>ALT</b> 键的状态。某键被按下使得一个二进制位被设置。 <b>shift</b> 参数是具有相应于 <b>SHIFT</b> 键（位 0）， <b>CTRL</b> 键（位 1），以及 <b>ALT</b> 键（位 2）最少二进制位的一个位字段。这些位的值分别等于 1，2，和 4。 <b>shift</b> 参数指示这些键的状态。这些位中可能有一些、全部、或者一个也没有被设置，指示这些键中的一些、全部、或者一个也没有被按下。例如， <b>CTRL</b> 和 <b>ALT</b> 键都被按下，则 <b>shift</b> 的值就是 6 |

续表

| 部分   | 描述   |
|------|--|
| x, y | 一个指定鼠标指针当前位置的数。x 和 y 的值所表示的总是通过该对象 ScaleHeight, ScaleWidth, ScaleLeft,和 ScaleTop 属性所建立的坐标系统的方式 |

说明

MouseMove 事件伴随鼠标指针在对象间移动时连续不断地产生。除非有另一个对象捕获了鼠标，否则，当鼠标位置在对象的边界范围内时该对象就能接收 MouseMove 事件。

要测试 button 或 shift 参数，可使用对象浏览器中的 Visual Basic (VB) 对象库中所列出的常量，用来定义该参数中的各个位：

| 常量（按钮）         | 值 | 描述        |
|----------------|---|-----------|
| VbLeftButton   | 1 | 左按钮按下     |
| VbRightButton  | 2 | 右按钮按下     |
| VbMiddleButton | 4 | 中间按钮按下    |
| VbShiftMask    | 1 | SHIFT 键按下 |
| VbCtrlMask     | 2 | CTRL 键按下  |
| VbAltMask      | 4 | ALT 键按下   |

然后这些常量用作位屏蔽，对于按钮的各种组合，无须计算出各个组合的

唯一的位字段值即可进行检测。

要测试某一条件，首先将各个结果赋给一个临时整型变量然后再与一个位屏蔽的 `button` 或 `shift` 参数进行比较。测试应当用各个参数进行 `And` 运算，若结果大于零，则说明该键或按钮被按下。其操作如下：

```
LeftDown = (Button And vbLeftButton) > 0
```

```
CtrlDown = (Shift And vbCtrlMask) > 0
```

然后，接下去可对结果的各种组合进行检测，其操作如下：

```
If LeftDown And CtrlDown Then
```

注意：为了对鼠标按钮按下和释放所引起的事件进行处理，可使用 `MouseDown` 和 `MouseUp` 事件过程。

`MouseMove` 事件的 `button` 参数与 `MouseDown` 和 `MouseUp` 事件的 `button` 参数是不同的。对于 `MouseMove` 事件来说，`button` 参数指示的是所有按钮当前的状态；一个 `MouseMove` 事件可指示某些、全部或没有一个按钮被按下。对于 `MouseDown` 和 `MouseUp` 事件来说，`button` 参数在每个事件精确地指示一个按钮。

在 `MouseMove` 事件中任何时候移动窗口，都能引起层叠事件。当该窗口移动到指针下面时 `MouseMove` 事件将产生。即使是鼠标完全不动 `MouseMove` 事件也能产生。

# MousePointer 属性 （ActiveX 控件）

返回或设置一个值，该值指示在运行时当鼠标移动到对象的一个特定部分时，被显示的鼠标指针的类型。

应用于

Masked 编辑控件，Multimedia MCI 控件，CoolBar 控件，DataCombo 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，DataRepeater 控件，RichTextBox 控件，Slider 控件，StatusBar 控件。

语法

*object*.MousePointer [= *value*]

MousePointer 属性语法包含下面部分：

| 部分            | 描述                       |
|---------------|--------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象   |
| <i>value</i>  | 整数，按照设置值中的描述指定被显示的鼠标指针类型 |

# 设置

value 的设置值为:

| 常量            | 值  | 描述                       |
|---------------|----|--------------------------|
| VbDefault     | 0  | (缺省值) 形状由对象决定            |
| VbArrow       | 1  | 箭头                       |
| VbCrosshair   | 2  | 十字线 (crosshair 指针)       |
| VbIbeam       | 3  | I 型                      |
| VbIconPointer | 4  | 图标 (矩形内的小矩形)             |
| VbSizePointer | 5  | 尺寸线 (指向东、南、西和北四方向的箭头)    |
| VbSizeNESW    | 6  | 右上-左下尺寸线 (指向东北和西南方向的双箭头) |
| VbSizeNS      | 7  | 垂-直尺寸线 (指向南和北的双箭头)       |
| VbSizeNWSE    | 8  | 左上-右下尺寸线 (指向东南和西北方向的双箭头) |
| VbSizeWE      | 9  | 水-平尺寸线 (指向东和西两个方向的双箭头)   |
| VbUpArrow     | 10 | 向上的箭头                    |
| VbHourglass   | 11 | 沙漏 (表示等待状态)              |
| VbNoDrop      | 12 | 不允许放下                    |



续表

| 常量               | 值  | 描述                                    |
|------------------|----|---------------------------------------|
| VbArrowHourglass | 13 | 箭头和沙漏                                 |
| VbArrowQuestion  | 14 | 箭头和问号                                 |
| VbSizeAll        | 15 | 四向尺寸线                                 |
| VbCustom         | 99 | 通过 <code>MouseIcon</code> 属性所指定的自定义图标 |

## 说明

在鼠标指针越过窗体或对话框上的控件时，为了指出功能上的改变，可以使用该属性。沙漏标形状设置值 (11) 是很有用的，用来指示用户需要等待过程或操作的完成。

**注意：**如果应用程序调用 `DoEvents`，那么 `MousePointer` 属性在经过 `ActiveX` 部件时可能暂时改变。

## Object 属性（ActiveX 控件）

返回对象与/或对象的方法或属性的设置。

# 应用于

MAPISession 控件, MAPIMessages 控件, Masked 编辑控件, Multimedia MCI 控件, PictureClip 控件, Animation 控件, MSChart 控件, CoolBar 控件, DateTimePicker 控件, MSHFlexGrid 控件, MSFlexGrid 控件, FlatScrollBar 控件, ImageCombo 控件, ImageList 控件, Microsoft Internet Transfer 控件, ListView 控件, Winsock 控件, ProgressBar 控件, RichTextBox 控件, Slider 控件, StatusBar 控件, SysInfo 控件, TabStrip 控件, Toolbar 控件, TreeView 控件, UpDown 控件, DBCombo 控件, DBList 控件。

# 语法

*object*.Object[*.Property* | *.Method*]

Object 属性的语法具有以下几部分：

| 部分              | 描述                   |
|-----------------|----------------------|
| <i>object</i>   | 对象表达式，其值是“应用于”列表中的对象 |
| <i>property</i> | 对象支持的属性              |
| <i>method</i>   | 对象支持的方法              |

## 说明

用该属性指定要在 **Automation** 任务中使用的对象。

在自动化任务中通过使用对象的属性和方法来使用 **Object** 属性返回的对象。对属性和方法支持的信息，请参阅有关创建对象的应用程序的文档。

## OLECompleteDrag 事件（ActiveX 控件）

当源部件被放到目标部件时发生，并通知源部件拖放操作被执行或取消。

## 应用于

**Masked** 编辑控件，**Multimedia MCI** 控件，**Animation** 控件，**MSChart** 控件，**CoolBar** 控件，**DataList** 控件，**DateTimePicker** 控件，**MSHFlexGrid** 控件，**MSFlexGrid** 控件，**FlatScrollBar** 控件，**ImageCombo** 控件，**ListView** 控件，**ProgressBar** 控件，**RichTextBox** 控件，**Slider** 控件，**StatusBar** 控件，**TabStrip** 控件，**ToolBar** 控件，**TreeView** 控件，**UpDown** 控件。

# 语法

Private Sub *object*\_OLECompleteDrag([*effect* As Long])

OLECompleteDrag 事件语法包含下面部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>effect</i> | 源对象设置的长整型数，用来识别执行的动作，这样当部件被移动后允许源采取适当的动作（例如，如果源被从一个部件移到另一个部件，则执行删除数据操作）。可能的取值列于“设置值”中 |

# 设置

effect 设置如下：

| 常量               | 值 | 描述                                  |
|------------------|---|-------------------------------------|
| VbDropEffectNone | 0 | 放目标不接受数据，或者放操作被取消。                  |
| VbDropEffectCopy | 1 | 放结果保存于从源到目标的数据副本中。<br>初始数据没有被拖放操作改变 |
| VbDropEffectMove | 2 | 放下导致数据从拖动源移动到放下源。拖动源应该在移动后从自身将数据删除  |

## 说明

**OLECompleteDrag** 事件是 OLE 拖放操作最后调用的事件。当对象被放到目标部件时，此事件通知源部件所执行的动作。目标通过 **OLEDragDrop** 事件的 **effect** 参数设置此值。基于此，源可决定需采取的适当动作。例如，若对象被移到目标 (**vbDropEffectMove**)，移动后源需要在自身删除该对象。

如果 **OLEDragMode** 设置为 **Automatic**，则 **Visual Basic** 执行缺省的动作。此事件仍然发生，允许添加或改变动作。

大部分部件支持人工 OLE 拖放事件，也有一些支持自动化 OLE 拖放事件。

## OLEDrag 方法 (ActiveX 控件)

引起部件初始化 OLE 拖放操作。

应用于

**Masked** 编辑控件，**Multimedia MCI** 控件，**Animation** 控件，**MSChart** 控件，**CoolBar** 控件，**DataList** 控件，**DateTimePicker** 控件，**MSHFlexGrid** 控件，

MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

## 语法

*object*.OLEDrag

*object* 置换元代表对象表达式，其值是“应用于”列表中的一个对象。

## 说明

当调用 OLEDrag 方法时，部件的 OLEStartDrag 事件发生，允许向目标部件提供数据。

## OLEDragDrop 事件（ActiveX 控件）

当源部件决定放操作能发生，且源部件被放到目标部件时，此事件发生。

**注意：**仅当 OLEDropMode 被设置为 1 (Manual) 时，此事件才发生。

# 应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

# 语法

Private Sub *object*\_OLEDragDrop(*data* As DataObject, *effect* As Long, *button* As Integer, *shift* As Integer, *x* As Single, *y* As Single)

OLEDragDrop 事件语法包含下面部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>data</i>   | 对象，包含源提供的格式，另外也可能包含这些格式的数据。若 DataObject 不包含数据，则当控件调用 GetData 方法时提供数据。SetData 和 Clear 方法不能用在这里 |

续表

| 部分            | 描述  |
|---------------|---|
| <i>effect</i> | 源对象设置的长整型数，用来识别执行的动作，这样当部件被移动后允许源采取适当的动作（例如，如果源被从一个部件移到另一个部件，则执行删除数据操作）。可能的取值列于“设置值”中   |
| <i>button</i> | 整数，当按下鼠标键时，与鼠标状态相对应。左键为位 0，右键为位 1，中键为位 2。这些位相应的值分别为 1, 2 和 4，它代表了鼠标键的状态。可设置三个位中的部分、全部或根本不设置，相应地表明部分、全部按键被按下或没有按键按下  |
| <i>shift</i>  | 整数，当按下 SHIFT、CTRL 和 ALT 键时，与这些键状态相对应。SHIFT 键为位 0，CTRL 键为位 1，ALT 键为位 2。这些位相应的值分别为 1, 2 和 4，shift 参数代表了这些键的状态。可设置三个位中的部分、全部或根本不设置，相应地表明部分、全部按键被按下或没有按键按下。例如，同时按下 CTRL 和 ALT 键，shift 值为 6。CTRL |
| <i>x,y</i>    | 确定鼠标指针当前位置的数值。x 和 y 值由对象的 ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop 属性设置的坐标系统的格式来表示   |



# 设置

effect 设置如下：

| 常量               | 值 | 描述                               |
|------------------|---|----------------------------------|
| VbDropEffectNone | 0 | 放目标不接受数据                         |
| VbDropEffectCopy | 1 | 放结果保存于从源到目标的数据拷贝中。初始数据没有被拖放操作改变  |
| VbDropEffectMove | 2 | 放结果保存于要从拖放源移到放源的数据中。移动后，拖放源要删除数据 |

# 说明

源 ActiveX 部件应总是屏蔽 effect 参数值，以确保同将来实现的 ActiveX 部件兼容。目前，仅使用了 effect 参数 32 位中的 3 位，然而将来的 Visual Basic 版本就可能用到其它位。所以考虑到将来的问题， 拖源和放目标在进行任何比较之前应屏蔽这些值。

例如，源部件不能把 effect 同 vbDropEffectCopy 相比，如：

If Effect = vbDropEffectCopy...

而是应该屏蔽该值或被搜寻的值，如：

If Effect And vbDropEffectCopy = vbDropEffectCopy...

-或-

If (Effect And vbDropEffectCopy)...

这样，允许在 Visual Basic 新版本中定义新的放效果，并与现存的代码保持向后兼容性。

大部分部件支持人工 OLE 拖放事件，也有一些支持自动化 OLE 拖放事件。

请参阅

OLECompleteDrag 事件（ActiveX 控件），OLEDragOver 事件（ActiveX 控件），OLEGiveFeedback 事件（ActiveX 控件），OLESetData 事件（ActiveX 控件），OLEStartDrag 事件（ActiveX 控件）。

OLEDragMode 属性（ActiveX 控件）

返回或设置是由部件还是由程序员来处理 OLE 拖放操作。

# 应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

# 语法

*object.OLEDragMode = mode*

OLEDragMode 属性语法包含下面部分：

| 部分            | 描述                              |
|---------------|---------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象          |
| <i>mode</i>   | 整数，规定部件处理 OLE 拖放操作的方法，如“设置值”中所述 |

# 设置

mode 设置如下：

| 常量                 | 值 | 描述                       |
|--------------------|---|--------------------------|
| VbOLEDragManual    | 0 | （缺省）人工。程序员处理所有的 OLE 拖放操作 |
| VbOLEDragAutomatic | 1 | 自动。部件处理所有的 OLE 拖放操作      |

## 说明

当把 **Manual** 设置为 **OLEDragmode** 时，必须调用 **OLEDrag** 方法启动拖放，进而触发 **OLEStartDrag** 事件。

当把 **Automatic** 设置为 **OLEDragmode** 时，若想拖出控件，源部件用其包含的数据填充 **DataObject** 对象，并在初始化 **OLEStartDrag** 事件（也包括 **OLESetData** 和其它源级的 OLE 拖放事件）之前设置 **effects** 参数。这样可以控制拖放操作，允许通过添加其它格式，或者用 **Clear** 或 **SetData** 方法忽略或禁用自动数据来进行调整。

若源的 **OLEDragMode** 属性被设为 **Automatic**，并且也没有数据被加载到 **OLEStartDrag** 事件中，或 **aftereffects** 设置为 0，则 OLE 拖放操作不发生。

**注意:** 如果控件的 **DragMode** 属性被设为 **Automatic**，则 **OLEDragMode** 的设置被忽略，因为常规的 Visual Basic 拖放事件优先发生。

## OLEDragOver 事件（ActiveX 控件）

当一个部件在另一个部件上拖动时发生。

应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

语法

```
Private Sub object_OLEDragOver(data As DataObject, effect As Long, button As Integer, shift As Integer, x As Single, y As Single, state As Integer)
```

OLEDragOver 事件语法包含下面部分：

| 部分                 | 描述   |
|--------------------|--|
| <i>object data</i> | 对象表达式，其值是“应用于”列表中的一个对象对象，包含源提供的格式，另外也可能包含这些格式的数据。若 <code>DataObject</code> 不包含数据，则当控件调用 <code>GetData</code> 方法时提供数据。 <code>SetData</code> 和 <code>Clear</code> 方法不能用在这里   |
| <i>effect</i>      | 源对象最初设置的长整型数，用来识别它支持的效果。在事件过程中，此参数必须被目标部件正确地设置。 <code>effect</code> 值由所有活动的效果（如设置值表）逻辑 <b>Or</b> 确定。目标部件应检查这些效果以及其它参数以确定哪个动作适合于它，然后把此参数设置为允许的效果之一（如源所规定），以便确定放置选项到部件该执行哪个动作。可能的取值列于“设置值”中  |
| <i>button</i>      | 整数，当按下鼠标键时，与鼠标状态相对应。左键为位 0，右键为位 1，中键为位 2。这些位相应的值分别为 1，2 和 4，它代表了鼠标键的状态。可设置三个位中的部分、全部或根本不设置，相应地表明部分、全部按键被按下或没有按键按下  |
| <i>shift</i>       | 整数，当按下 <b>SHIFT</b> 、 <b>CTRL</b> 和 <b>ALT</b> 键时，与这些键状态相对应。 <b>SHIFT</b> 键为位 0， <b>CTRL</b> 键为位 1， <b>ALT</b> 键为位 2。这些位相应的值分别为 1，2 和 4， <code>shift</code> 参数代表了这些键的状态。可设置三个位中的部分、全部或根本不设置，相应地表明部分、全部按键被按下或没有按键按下。例如，同时按下 <b>CTRL</b> 和 <b>ALT</b> 键， <code>shift</code> 值为 6 |

续表

| 部分           | 描述  |
|--------------|---|
| <i>x,y</i>   | 在目标窗体或控件中， 规定当前鼠标指针水平 (x) 和垂直 (y) 位置的数。x 和 y 值由对象的 <b>ScaleHeight</b> 、 <b>ScaleWidth</b> 、 <b>ScaleLeft</b> 和 <b>ScaleTop</b> 属性设置的坐标系统格式来表示 |
| <i>state</i> | 整数，相应于控件的转换状态，此控件将被拖放到与其相关的目标窗体或控件中。可能的取值列于“设置值”中   |

设置

effect 设置如下：

| 常量                        | 值          | 描述                               |
|---------------------------|------------|----------------------------------|
| <b>VbDropEffectNone</b>   | 0          | 放目标不接受数据                         |
| <b>VbDropEffectCopy</b>   | 1          | 放结果保存于从源到目标的数据拷贝中。初始数据没有被拖放操作改变  |
| <b>VbDropEffectMove</b>   | 2          | 放结果保存于要从拖放源移到放源的数据中。移动后，拖放源要删除数据 |
| <b>VbDropEffectScroll</b> | -          | 在目标部件中，滚动正在或将要发生。                |
|                           | 2147483648 | 此值与其它值共同使用。注意 仅当在部件              |
|                           | (&H80000   | 中执行自己的滚动时才能应用                    |
|                           | 000)       |                                  |

state 设置如下:

| 常量      | 值 | 描述                    |
|---------|---|-----------------------|
| VbEnter | 0 | 在目标范围内源部件正被拖动         |
| VbLeave | 1 | 在目标范围之外源部件正被拖动        |
| VbOver  | 2 | 在目标内源部件已经从一个位置移到另一个位置 |

## 说明

注意: 如果 state 参数是 vbLeave, 表明鼠标指针已离开目标, 则 x 和 y 参数保持为零值。

源 ActiveX 部件应总是屏蔽 effect 参数值, 以确保同将来实现的 ActiveX 部件兼容。现在仅使用了 effect 参数 32 位中的 3 位, 然而将来的 Visual Basic 版本就可能用到其它位。所以考虑到将来的问题, 拖源和放目标在进行任何比较之前应屏蔽这些值。

例如, 源部件不能把 effect 同 vbDropEffectCopy 相比, 如:

```
If Effect = vbDropEffectCopy...
```

而应屏蔽该值或被搜寻的值, 如:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```



-或-

If (Effect And vbDropEffectCopy)...

允许在 Visual Basic 新版本中定义新的放效果，并与现存的代码保持向后兼容性。

大部分部件支持人工 OLE 拖放事件，也有一些支持自动化 OLE 拖放事件。

## OLEDropMode 属性（ActiveX 控件）

返回或设置目标部件如何处理放操作。

应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

语法

*object*.OLEDropMode [= *mode*]

OLEDropMode 属性语法包含下面部分：

| 部分            | 描述                                |
|---------------|-----------------------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象            |
| <i>mode</i>   | 枚举整数，规定部件处理 OLE 拖放操作的方法，如“设置值”中所述 |

设置

*mode* 设置如下：

| 常量                 | 值 | 描述   |
|--------------------|---|--|
| VbOLEDropNone      | 0 | 无。目标部件不接受 OLE 放操作，并且显示 No Drop 图标。   |
| VbOLEDropManual    | 1 | 人工。目标部件触发 OLE 放事件，允许程序员用代码处理 OLE 放操作   |
| VbOLEDropAutomatic | 2 | 自动。如果 DataObject 对象包含目标部件能识别的格式的数据，则自动接受 OLE 放操作。当 OLEDropMode 设为 vbOLEDropAutomatic 时，在目标上鼠标事件和 OLE 拖放事件都不会发生 |

## 说明

注意：目标部件检查将在其上拖动的内容，以便确定触发何种事件，是 OLE 拖放事件还是 Visual Basic 拖放事件。在同一时间内仅能拖动一种对象，所以没有部件的冲突，没有触发任何事件的疑惑。

## OLEGiveFeedback 事件（ActiveX 控件）

在每个 OLEDragOver 事件后发生。OLEGiveFeedback 允许源部件提供可视的反馈，例如通过改变鼠标的图标来表明放目标后将发生什么，或者在选项上提供可视的反馈（在源部件中）以指出将发生什么。

## 应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

# 语法

Private Sub *object*.OLEGiveFeedback(*effect* As Long, *defaultcursors* As Boolean)

OLEGiveFeedback 事件语法包含下面部分：

| 部分                    | 描述  |
|-----------------------|---|
| <i>object</i>         | 对象表达式，其值是“应用于”列表中的一个对象。   |
| <i>effect</i>         | 在 OLEDragOver 事件中目标部件设置的长整型数， 若放选定项，则由此数确定要执行的动作。允许源采取合适的动作（例如给出可视的反馈）。可能的取值列于“设置值”中。                               |
| <i>defaultcursors</i> | 布尔值，决定 Visual Basic 使用部件缺省鼠标图标，还是自定义鼠标图标 True（缺省） = 使用缺省鼠标图标<br>False = 不用缺省图标。必须用 Screen 对象的 MousePointer 属性设置鼠标图标 |

# 设置

*effect* 设置如下：

| 常量                 | 值          | 描述                               |
|--------------------|------------|----------------------------------|
| VbDropEffectNone   | 0          | 放目标不接受数据                         |
| VbDropEffectCopy   | 1          | 放结果保存于从源到目标的数据拷贝中。初始数据没有被拖放操作改变  |
| VbDropEffectMove   | 2          | 放结果保存于要从拖放源移到放源的数据中。移动后，拖放源要删除数据 |
| VbDropEffectScroll | -          | 在目标部件中，滚动正在或将要发生。                |
|                    | 2147483648 | 此值与其它值共同使用。注意仅当在部件中              |
|                    | (&H800     | 执行自己的滚动时才能应用                     |
|                    | 00000)     |                                  |

### 说明

如果 OLEGiveFeedback 事件中没有代码，或者如果 defaultcursors 参数设为 True，则 Visual Basic 自动地把鼠标图标设为部件提供的缺省图标。

源 ActiveX 部件应总是屏蔽 effect 参数值，以确保同将来实现的 ActiveX 部件兼容。现在仅使用了 effect 参数 32 位中的 3 位，然而将来的 Visual Basic 版本就可能用到其它位。所以考虑到将来的问题，拖源和放目标在进行任何比较之前应屏蔽这些值。

例如，源部件不能把 effect 同 vbDropEffectCopy 相比，如：

If Effect = vbDropEffectCopy...

而应屏蔽该值或被搜寻的值，如：

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-或-

```
If (Effect And vbDropEffectCopy)...
```

允许在 **Visual Basic** 新版本中定义新的放效果，并与现存的代码保持向后兼容性。

大部分部件支持人工 **OLE** 拖放事件，也有一些支持自动化 **OLE** 拖放事件。

## OLESetData 事件（ActiveX 控件）

当目标部件在源的 **DataObject** 对象上执行 **GetData** 方法，但是还没有加载规定格式的数据之前，在源部件上发生。

应用于

**Masked** 编辑控件，**Multimedia MCI** 控件，**Animation** 控件，**MSChart** 控件，**CoolBar** 控件，**DataList** 控件，**DateTimePicker** 控件，**MSHFlexGrid** 控件，

MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

语法

```
Private Sub object_OLESetData(data As DataObject, dataformat As Integer)
```

OLESetData 事件语法包含下面部分：

| 部分                | 描述   |
|-------------------|--|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象                         |
| <i>data</i>       | 放置所需数据的 DataObject 对象。部件调用 SetData 方法加载所需的格式   |
| <i>dataformat</i> | 整数，确定目标部件所需要的数据格式。源部件用此值来定向 DataObject 对象加载的内容 |

说明

在某些情况下，特别是在源部件支持多种格式时，可能希望延迟向源部件 DataObject 对象加载数据以节省时间。此事件允许源对给定的数据格式仅响应一个请求。调用此事件时，源应该检查 format 参数以确定需加载的内容，然后在 DataObject 上执行 SetData 方法加载数据，这些数据随后被送回到目标

部件。

## OLEStartDrag 事件（ActiveX 控件）

当部件的 `OLEDrag` 方法被执行时，或者在 `OLEDragMode` 属性被设为 `Automatic`，部件初始化 OLE 拖放操作时发生。

此事件指定源部件支持的数据格式和放效果。也可用于向 `DataObject` 对象中插入数据。

### 应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件，UpDown 控件。

### 语法

```
Private Sub object_OLEStartDrag(data As DataObject, allowedeffects As Long)
```



OLEStartDrag 事件语法包含下面部分：

| 部分                    | 描述  |
|-----------------------|---|
| <i>object</i>         | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>data</i>           | <b>DataObject</b> 对象，含源所提供的格式以及（可选）适合那些格式的数据。若 <b>DataObject</b> 不包含数据，则当控件调用 <b>GetData</b> 方法时会提供。您应提供此数据参数的值。 <b>SetData</b> 和 <b>Clear</b> 方法不能用于此处 |
| <i>allowedeffects</i> | 长整型数，包含源部件支持的效果。其可能的取值列于“设置值”中  |

设置

*allowedeffects* 设置如下：

| 常量                      | 值 | 描述                              |
|-------------------------|---|---------------------------------|
| <b>VbDropEffectNone</b> | 0 | 放目标不接受数据                        |
| <b>VbDropEffectCopy</b> | 1 | 放结果保存于从源到目标的数据拷贝中。初始数据没有被拖放操作改变 |
| <b>VbDropEffectMove</b> | 2 | 放结果保存于要从拖源移到放源的数据中。移动后，拖源要删除数据  |

## 说明

源部件应该逻辑 Or 所支持的数据并把结果放到 `allowedeffects` 参数中。目标部件使用此值确定合适的动作（以及合适的用户反馈）。

如果部件的 `OLEDragMode` 属性被设为 `Automatic`，`StartDrag` 事件也会发生。这样就允许在部件执行该操作后，再次向 `DataObject` 对象加入格式及数据。也可以通过清除 `DataObject` 对象（用 `Clear` 方法）忽略部件的缺省行为，然后加入自己的数据和格式。

除非目标部件需要，可能希望延迟向 `DataObject` 对象中加载数据。这样，由于没有加载多余的数据格式，使源部件节省了时间。当目标在 `DataObject` 上执行 `GetData` 方法时，若 `DataObject` 中未包含所需的数据，则源的 `OLESetData` 事件会发生。此时，数据可被加载到 `DataObject` 中，这将依次地向目标提供数据。

如果用户没有给 `DataObject` 加载任何格式，则就取消拖/放操作。

## Picture 属性（ActiveX 控件）

返回或设置控件中要显示的图片。对于 `OLE` 容器控件，在设计时不可用，

在运行时为只读。

应用于

PictureClip 控件， ListImage 对象， ListImages 集合， Panel 对象。

语法

*object*.Picture [= *picture*]

Picture 属性有下列组成部分：

| 部分             | 描述                                 |
|----------------|------------------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象             |
| <i>picture</i> | 字符串表达式，指定一个包含图片的文件，“设置值”中有<br>详细说明 |

设置

picture 的设置值如下：

| 设置值    | 描述       |
|--------|----------|
| (None) | （缺省值）无图片 |

续表

| 设置值                      | 描述  |
|--------------------------|---|
| (Bitmap, icon, metafile) | 指定一个图片。设计时可以从属性窗口中加载图片。在运行时，也可以在位图，图标，或元文件上使用 <b>LoadPicture</b> 函数来设置该属性 |

## 说明

在设计时，利用“编辑”菜单中的“复制”、“剪切”和“粘贴”命令通过剪贴板来传递图片，运行时，可以使用剪贴板方法，诸如具有非文本剪贴板常量 **vbCFBitmap**、**vbCFMetafile** 和 **vbCFDIB** 的 **GetData**、**SetData** 和 **GetFormat**，它们列在对象浏览器中的 **Visual Basic (VB)** 对象库中。

在设计时设置 **Picture** 属性，图片被保存起来并与窗体同时加载。如果创建可执行文件，该文件中包含该图像。如果在运行时加载图片，该图片不和应用程序一起保存。用 **SavePicture** 语句可以从窗体或图片框的图片存储到文件中。

**注意：**运行时，**Picture** 属性可以被设置为任何其它对象的 **DragIcon**、**Icon**、**Image** 或 **Picture** 属性，或者可将 **LoadPicture** 函数返回的图片分配给它。

请参阅

ListImage 对象，ListImages 集合，Panel 对象，Add 方法（Panels 集合）。

## Refresh 方法（ActiveX 控件）

强制全部重绘一个窗体或控件。

请参阅

Masked 编辑控件，Multimedia MCI 控件，MSChart 控件，CoolBar 控件，DataList 控件，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，DataRepeater 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，TabStrip 控件，ToolBar 控件，TreeView 控件。

语法

*object*.Refresh

*object* 置换元代表一个对象表达式，其值是“应用于”列表中的一个对象。

## 说明

在下列情况下使用 **Refresh** 方法

- 在另一个窗体被加载时显示一个窗体的全部。
- 更新诸如 **FileListBox** 控件之类的文件系统列表框的内容。
- 更新 **Data** 控件的数据结构。

**Refresh** 方法不能用于 **MDI** 窗体，但能用于 **MDI** 子窗体。不能在 **Menu** 或 **Timer** 控件上使用 **Refresh** 方法。

通常，如果没有事件发生，窗体或控件的绘制是自动处理的。但是，有些情况下希望窗体或控件立即更新。例如，如果使用文件列表框、目录列表框或者驱动器列表框显示当前的目录结构状态，当目录结构发生变化时可以使用 **Refresh** 更新列表。

可以在 **Data** 控件上使用 **Refresh** 方法来打开或重新打开数据库（如果 **DatabaseName**、**ReadOnly**、**Exclusive** 或 **Connect** 属性的设置值发生改变），并能重建控件的 **Recordset** 属性内的 **dynaset**。

# RemoteHost 属性（ActiveX 控件）

返回或设置远程计算机，控件向它发送数据或从它那里接收数据。既可提供主机名，比如 "FTP://ftp.microsoft.com"，也可提供点格式下的 IP 地址字符串，比如 "100.0.1.1"。

应用于

Microsoft Internet Transfer 控件，Winsock 控件。

语法

*object.RemoteHost = string*

RemoteHost 属性的语法具有以下几部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>string</i> | 远程计算机的名称或地址          |

## 说明

在指定该属性时，应更新 **URL** 属性来显示新值。如果更新 **URL** 的主机部分，则也要更新该属性来反映新值。

在调用 **OpenURL** 或 **Execute** 方法时也可改变 **RemoteHost** 属性。

在运行时直到下一个连接之前，改变该值都不会造成什么影响。

## 请参阅

**Bind** 方法。

## RemotePort 属性（ActiveX 控件）

返回或设置要连接的远程端口号

## 应用于

**Microsoft Internet Transfer** 控件，**Winsock** 控件。



# 语法

```
object.RemotePort = port
```

RemotePort 属性的语法具有以下几部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>port</i>   | 要连接的端口。该属性的缺省值是 80   |

# 数据类型

Long

# 说明

在设置 Protocol 属性时，将对每个协议自动把 RemotePort 属性设置成适当的缺省端口。下表列出缺省端口号：

| 端口 | 描述                          |
|----|-----------------------------|
| 80 | HTTP，通常用于 World Wide Web 连接 |
| 21 | FTP                         |

请参阅

**Bind** 方法。

## Remove 方法（ActiveX 控件）

该方法从集合中删除一项。

应用于

LightSources 集合, Bands 集合, ComboItems 集合, ListImage 对象, ListImages 集合, ColumnHeader 对象, ColumnHeaders 集合, ListItem 对象, ListItems 集合, RepeaterBindings 集合, OLEObjects 集合, Panels 集合, Tabs 集合, Buttons 集合, Node 对象, Nodes 集合。

语法

*object.Remove (index)*

**Item** 方法的语法包含下面部分：

| 部分            | 描述   |
|---------------|--|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象   |
| <i>index</i>  | 一个整数或字符串，唯一标识集合中的对象。使用整数指明 <b>Index</b> 属性的值，使用字符串指明 <b>Key</b> 属性的值 |

## 说明

要删除集合中所有的成员，使用 **clear** 方法。

## 请参阅

**Clear** 方法 (ActiveX 控件)，**Key** 属性 (ActiveX 控件)，**Index** 属性 (ActiveX 控件)。

## 示例

在本例将 6 个 **Panel** 对象添加到一个 **StatusBar** 控件，共创建了 7 个 **Panel** 对象。在单击窗体时，代码将检查共有多少个 **Panel** 对象。如果只有一个 **Panel** 对象，程序将添加 6 个 **Panel** 对象。否则，它将删除第一个 **Panel**。如果要试运行本例，请将一个 **StatusBar** 控件放到一个窗体上，然后将下面的代码粘贴到“声明”部分。在运行示例时，每次开始单击窗体都将删除一个 **Panel** 对象，

然后添加 Panel 对象。

```
Private Sub Form_Load()
```

```
    Dim pnlX As Panel ' 为 Panel 对象声明对象变量。
```

```
    Dim i As Integer
```

```
    ' 在缺省的 Panel 对象中添加 6 个 Panel 对象，
```

```
    ' 得到 7 个 Panel 对象。
```

```
    For i = 1 To 6
```

```
        Set pnlX = StatusBar1.Panels.Add(, , i)
```

```
        pnlX.AutoSize = sbrSpring
```

```
    Next i
```

```
End Sub
```

```
Private Sub Form_Click()
```

```
    ' 如果集合的 Count 为 1，添加 6 个 Panel 对象。
```

```
    ' 否则，从集合中删除第一个 Panel。
```

```
    If StatusBar1.Panels.Count = 1 Then
```

```
        Dim sbrX As Panel
```

```
        Dim i As Integer
```

```
        For i = 1 To 6 ' 通过 i 设置每个 Panel 的样式。
```

```
            Set sbrX = StatusBar1.Panels.Add(, , i)
```

```
            sbrX.AutoSize = sbrSpring
```

```
Next i
Else ' 删除第一个 Panel。
    StatusBar1.Panels.Remove 1
End If
End Sub
```

## RightToLeft 属性（ActiveX 控件）

返回布尔值，指出文本显示方向并在双向系统上控制可见外观。

应用于

DataCombo 控件，DataList 控件，DBCombo 控件，DBList 控件。

语法

*object*.RightToLeft

*object* 置换元代表“应用于”中的一个对象的对象表达式。

## 设置

从 `RightToLeft` 属性可能返回的布尔值为：

| 设置值   | 描述   |
|-------|--|
| True  | 控件在双向平台上运行，比如在阿拉伯语的 Windows95 或希伯来语的 Windows 95 上运行，而且，文本从右到左运行。控件应修改其性能，比如在文本框或列表框的左边安放垂直滚动条，在文本框的右边放置标签，等等 |
| False | 控件应该就象在非双向平台上那样运行，比如就象在英语的 Windows95 上那样，而且，文本从左到右运行。如果容器不实现这一环境属性，则其为缺省值                                    |

## `SelLength`、`SelStart`、`SelText` 属性（ActiveX 控件）

- `SelLength`—返回或设置所选择的字符数。
- `SelStart`—返回或设置所选择的文本的起始点；如果没有文本被选中，则指出插入点的位置。
- `SelText`—返回或设置包含当前所选择文本的字符串；如果没有字符被选中，则为零长度字符串 ("").

这些属性在设计时是不可用的。

应用于

Masked Edit 控件，DataCombo 控件，Slider 控件。

语法

*object.SelLength* [= *number*]

*object.SelStart* [= *index*]

*object.SelText* [= *value*]

SelLength、SelStart、和 SelText 属性的语法包含下面部分：

| 部分            | 描述  |
|---------------|---|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的一个对象  |
| <i>number</i> | 一个用来指定被选择字符数的数值表达式。对于 SelLength 和 SelStart，设置值的有效范围是 0 到文本长度—在 ComboBox 或 TextBox 控件编辑区中字符的总数 |
| <i>index</i>  | 一个用来指定所选择文本的起始点的数值表达式，“设置值”中有详细描述   |
| <i>value</i>  | 包含所选择文本的字符串表达式  |

## 说明

对下面这些任务应使用这些属性，如设置插入点、建立插入范围、在控件中选择子串、或清除文本等。与 **Clipboard** 对象联合使用，这些属性对于复制、剪切、和粘贴操作是很有用的。

当使用这些属性时：

- **SelLength** 的设置比 0 小会导致一个运行时错误。
- **SelStart** 的设置比文本长度大，会使该属性设置为现有文本长度；**SelStart** 的改变将使选择改变到插入点并将 **SelLength** 设置为 0。
- **SelText** 的设置为新值，会将 **SelLength** 设置为 0 并用新字符串代替所选择的文本。

请参阅

**SelText** 属性（**Masked** 编辑控件）。



# SetData 方法（ActiveX 控件）

用指定的数据格式将数据插入 DataObject 对象。

应用于

DataObject 对象（ActiveX 控件）。

语法

*object*.SetData [*data*], [*format*]

SetData 方法的语法具有以下几部分：

| 部分            | 描述                            |
|---------------|-------------------------------|
| <i>object</i> | 必需的。对象表达式 其值是“应用于”列表中的对象      |
| <i>data</i>   | 可选的。变量，包含传递给 DataObject 对象的数据 |
| <i>format</i> | 可选的。常量或者值，它指定所传递的数据的格式        |

设置

format 的设置值为：

| 常量            | 值     | 描述                |
|---------------|-------|-------------------|
| vbCFText      | 1     | 文本 (.txt 文件)      |
| vbCFBitmap    | 2     | 位图 (.bmp 文件)      |
| vbCFMetafile  | 3     | 元文件 (.wmf 文件)     |
| vbCFEMetafile | 14    | 增强的元文件 (.emf 文件)  |
| vbCFDIB       | 8     | 与设备无关的位图 (DIB)    |
| vbCFPalette   | 9     | 调色板               |
| vbCFFiles     | 15    | 文件的列表             |
| vbCFRTF       | -     | 丰富的文本格式 (.rtf 文件) |
|               | 16639 |                   |

## 说明

在 Visual Basic (VB) 的对象浏览器的对象库中列举这些常量。

`data` 参数是可选的。因此可以设置几种不同的格式，不必对每种格式都分别加载数据，源部件就可支持这些格式。经过几次调用 `SetData` 来设置多重格式，每次都使用一种不同格式。如果想要刷新，则应使用 `Clear` 方法清除 `DataObject` 的所有数据和格式信息。

`format` 参数 也是可选的，但无论是 `data`，还是 `format` 参数，都必须是被指定的。如果指定了 `data`，但未指定 `format`，则 Visual Basic 将试图确定数据的格式。如果未能成功，则生成一个错误。当目标要求指定的数据和格式，

但未提供数据，则发生源 `OLESetData` 事件，而且资源可以提供所需的数据类型。

对 `GetData` 和 `SetData` 方法可以使用不同于设置值中列举的数据格式，包括用户定义的经 Windows 通过 `RegisterClipboardFormat()` API 函数注册的格式。但有几点需要告诫：

- 当数据未识别指定的数据格式时，`SetData` 方法要求数据具有字节数组的形式。
- 当数据具有不能识别的数据格式时，尽管 `Visual Basic` 可透明地将返回的字节数组转换成其它数据类型，比如字符串类型，`GetData` 方法也总是返回字节数组中的数据。
- 对某个在其数组结尾具有任意字节的操作系统，当在这个操作系统上运行时，`GetData` 返回的字节数组将大于实际数据。原因在于，`Visual Basic` 不知道数据的格式，它只知道操作系统分配给数据的内存大小。分配的内存通常要比数据实际所需的大。所以，在所分配的内存段的结尾可能有多余字节。结果，必须用适当的函数以有意义的方式解释返回的数据（例如，如果数据具有文本格式，则按具体长度截断字符串）。

## ShowTips 属性（ActiveX 控件）

返回一值，决定是否对对象显示工具提示。

应用于

StatusBar 控件， TabStrip 控件， Toolbar 控件。

语法

*object*.ShowTips

**object** 置换元表示一个对象表达式，其值为“应用于”列表中的一个对象。

返回值

一个布尔表达式，指出是否显示工具提示，如“设置”中所描述的。

设置

**value** 的设置值为：

| 设置    | 描述  |
|-------|---|
| True  | （缺省）控件中的每个对象都能显示一个相关联的字符串，这是 <b>ToolTipText</b> 属性的设置，该属性在对象之下的一个小矩形中。在运行时，当用户的光标在对象上逗留达一秒钟时，工具提示就会出现 |
| False | 在运行时，对象将不显示工具提示   |

## 说明

在设计时可在控件的“属性页”对话框中的通用选项卡上设置 **ShowTips** 属性。在运行时为只读。

## Tag 属性（ActiveX 控件）

返回或设置一个表达式用来存储程序中需要的额外数据。与其它属性不同，**Tag** 属性值不被 **Visual Basic** 使用；可以用该属性来标识对象。

## 应用于

**MAPI**Session 控件，**MAPI**Messages 控件，**Masked** 编辑控件，**Multimedia MCI**

控件,PictureClip 控件,Animation 控件,MSChart 控件,Band 对象,DateTimePicker 控件,MSHFlexGrid 控件,MSFlexGrid 控件,FlatScrollBar 控件,ImageList 控件,ListImage 对象,ListImages 集合,Microsoft Internet Transfer 控件,ListView 控件,ColumnHeader 对象,ColumnHeaders 集合,ListItem 对象,ListItems 集合,Winsock 控件,ProgressBar 控件,RepeaterBindings 对象,RichTextBox 控件,Slider 控件,StatusBar 控件,Panel 对象,SysInfo 控件,TabStrip 控件,Tab 对象,ToolBar 控件,Button 对象,Node 对象,Nodes 集合,UpDown 控件。

语法

```
object.Tag [= expression]
```

Tag 属性语法包含下面部分：

| 部分                | 描述                            |
|-------------------|-------------------------------|
| <i>object</i>     | 对象表达式，其值是“应用于”列表中的一个对象        |
| <i>expression</i> | 字符串表达式用来标识对象，缺省值为零长度字符串 ("" ) |

说明

利用该属性可以给对象赋予一个标识字符串，而不会影响其任何其它属性设置值或引起副作用。当需要检查控件或作为变量传递给过程的 MDIForm 对象的标识时，Tag 属性是有用的。

**提示：**创建一个新的窗口实例时，给 Tag 属性赋予唯一值。

**注意：**Tag 属性对于 Toolbar Button 对象、TreeView Node 对象、ListView ListItem 和 ColumnHeader 对象、ImageList ListImage 对象、TabStrip Tab 对象、以及 StatusBar Panel 对象的 ActiveX 控件来说是 Variant 类型。可以使用 Tag 属性来传递值，但是不允许传递对象。

## Text 属性（ActiveX 控件）

返回或设置包含在对象中的文本。

应用于

Masked 编辑控件，AxisTitle 对象，DataPointLabel 对象，Footnote 对象，Title 对象，DataGrid 控件，DataCombo 控件，DataList 控件，ImageCombo 控件，ColumnHeader 对象，ColumnHeaders 集合，ListItem 对象，ListItems 集合，RichTextBox 控件，Slider 控件，Panel 对象，Node 对象，Nodes 集合，DBCombo 控件，DBList 控件，Column 对象。

## 语法

*object*.Text [= *string*]

Text 属性的语法具有以下几部分：

| 部分            | 描述                   |
|---------------|----------------------|
| <i>object</i> | 对象表达式，其值是“应用于”列表中的对象 |
| <i>string</i> | 字符串表达式，指定出现在对象中的文本   |

## 请参阅

SelLength, SelStart, SelText 属性（ActiveX 控件），Value 属性（ActiveX 控件）。

## 示例

该示例将具有文件标题的 TreeView 控件置入一个 ListBox 控件。在单击 TreeView 控件中的项目时，窗体的 Label 中将显示 Text 属性。为了试一下这个示例，在窗体上放置 TreeView、Label 和 ListBox 控件，并将代码粘贴到窗体的声明部分。运行示例并单击任一项目，看看它的 Text 属性。

```
Private Sub Form_Load()
```



Dim nodX As Node '对节点声明对象变量。

Dim i As Integer '声明变量，将其作为计数器来使用。

'将一个 Node 添加到 TreeView 控件，并调用它到第一个节点

Set nodX = TreeView1.Nodes.Add()

nodX.Text = "First Node"

'置入 ListBox

List1.AddItem "Node1" ' 将每个项目添加到列表。

List1.AddItem "Node2"

List1.AddItem "Node3"

List1.AddItem "Node4"

List1.AddItem "Node5"

List1.AddItem "Node6"

List1.AddItem "Node7"

'将子节点添加到第一个 Node 对象。用

' ListBox 置入控件。

For i = 0 To List1.ListCount - 1

Set nodX = TreeView1.Nodes.Add(1, tvwChild)

nodX.Text = List1.List(i)

Next i

Treeview1.Nodes(1).Selected = True

```
nodX.EnsureVisible '确保节点是可见的。
```

```
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
```

```
'显示已单击的 Node 对象的 Text 属性。
```

```
Label1.Caption = Node.Text
```

```
End Sub
```

## ToolTipText 属性（ActiveX 控件）

返回或设置一个工具提示。

应用于

CheckBox 控件, ComboBox 控件, CommandButton 控件, Data 控件, DirListBox 控件, DriveListBox 控件, FileListBox 控件, Frame 控件, Image 控件, Label 控件, ListBox 控件, OptionButton 控件, PictureBox 控件, TextBox 控件

语法

```
object.ToolTipText [= ToolTipText]
```

ToolTipText 属性的语法有下面几部分：

| 部分                 | 描述                        |
|--------------------|---------------------------|
| <i>object</i>      | 一个 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>toolTipText</i> | 当鼠标指针停留在对象上时显示的字符串        |

## 说明

如果仅用图像作为对象的标签，那么能够使用此属性以较少的话解释每个对象。

在设计时仅可以在控件的属性对话框中设置 ToolTipText 属性字符串。

对于 Toolbar 和 TabStrip 控件，为了显示 ToolTips 必须设置 ShowTips 属性为 True

## Value 属性（ActiveX 控件）

返回或设置对象的值。

# 应用于

DateTimePicker 控件, FlateScrollBar 控件, ProgressBar 控件, Slider 控件, Button 控件, UpDown 控件

## 语法

*object.Value* [= *integer*]

Value 属性的语法具有以下几部分:

| 部分             | 描述  |
|----------------|---|
| <i>object</i>  | 对象表达式, 其值是“应用于”列表中的对象   |
| <i>integer</i> | 对 Slider 控件的一个长整数, 指定滑块的当前位置。对 ProgressBar 控件的一个整数, 指定 ProgressBar 控件的值。对其它控件, 请参阅以下设置值 |

## 设置

对 Button 对象, integer 的设置值为:

| 常量           | 值 | 描述                   |
|--------------|---|----------------------|
| TbrUnPressed | 0 | (缺省)。当前未按下按钮, 也未复选按钮 |
| TbrPressed   | 1 | 当前按下或复选按钮            |

## 说明

- **Slider** 控件—返回或设置滑块的当前位置。 **Value** 总是在 **Slider** 控件的 **Max** 属性值和 **Min** 属性值之间。
- **ProgressBar** 控件—返回或设置一值，指出操作大致接近完成。增加 **Value** 属性并不会按照 **Value** 属性的精确值改变 **ProgressBar** 控件的外观。**Value** 总是在 **Max** 属性值和 **Min** 属性值之间。在设计时不可用。

## 请参阅

**Click** 事件（**ActiveX** 控件），**Max,Min** 属性（**ActiveX** 控件），**Toolbar** 控件常量

## 示例

该示例使用 **Value** 属性来决定在 **Toolbar** 控件上显示相关联的 **ImageList** 控件的哪个图标。试以下示例，将 **Toolbar** 控件放在窗体上，并将代码粘贴在窗体的声明部分。然后运行示例。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
```

```
    '使用键值来决定已单击哪个按钮。
```

```
    Select Case Button.Key
```

Case "Done" '复选按钮。

If Button.Value = vbUnchecked Then

'未复选按钮。

Button.Value = vbChecked '复选按钮。

' 假定有一个具有"down"键的

' ListImage 对象。

Button.Image = "down"

Else ' Uncheck the 按钮

Button.Value = vbUnchecked

' 假定有一个具有"down"键的

' ListImage 对象。

Button.Image = "up"

End If

' 更多的可能。

End Select

End Sub

# Visible 属性（ActiveX 控件）

返回或设置一指示对象为可见或隐藏的值。

应用于

Masked 编辑控件，Multimedia MCI 控件，Animation 控件，AxisTitle 对象，Location 对象，Marker 对象，MSChart 控件，Band 对象，DateTimePicker 控件，MSHFlexGrid 控件，MSFlexGrid 控件，FlatScrollBar 控件，ImageCombo 控件，ListView 控件，ProgressBar 控件，RichTextBox 控件，Slider 控件，StatusBar 控件，Panel 对象，TabStrip 控件，ToolBar 控件，Button 对象，Node 对象，Nodes 集合，UpDown 控件

语法：

*object.Visible* [= *boolean*]

Visible 属性语法包含下面部分：

| 部分             | 描述                     |
|----------------|------------------------|
| <i>object</i>  | 对象表达式，其值是“应用于”列表中的一个对象 |
| <i>boolean</i> | 布尔表达式指定对象是可见还是隐藏       |

## 设置

*boolean* 的设置值为：

| 设置值   | 描述          |
|-------|-------------|
| True  | （缺省值）对象是可见的 |
| False | 对象是隐藏的      |

## 说明

要在启动时隐藏一个对象，在设计时将 **Visible** 属性设置为 **False**。在代码中设置该属性能够在运行时隐藏然后又重新显示控件以响应某特别事件。

**注意：**对窗体用 **Show** 或 **Hide** 方法，和在代码中将 **Visible** 属性分别设置为 **True** 或 **False** 的效果是一样的。