返回总目录

CEILING（）函数

返回大于或等于指定数值表达式的下一个最高整数。

语法

CEILING(nExpression)

返值类型

数值型

参数描述

nExpression

指定的数值，CEILING（）返回其后续的整数。

说明

CEILING 将一个带有小数位的数取整为下一个大于它的整数。

示例

```
STORE 10.1 TO num1
STORE -10.9 TO num2
? CEILING(num1)  && 显示 11
? CEILING(num2)  && 显示 -10
? CEILING(10.0)  && 显示 10
```

```
? CEILING(-10.0) && 显示 -10
```

请参阅

FLOOR( ), ROUND( )

Century 属性

CEILING 将一个带有小数位的数取整为下一个大于它的整数。

语法

Object.Century[= nValue]

参数描述

nValue

取下列设置：

设置	说明
0	不显示。不显示一个日期的世纪部分。
1	（默认值）显示。显示一个日期的世纪部分。
2	SET CENTURY 设置决定了是否显示一个日期的世纪部分。如果 SET CENTURY 是 ON，则显示一个日期的世纪部分。如果 SET CENTURY 是 OFF，则不显示一个日期的世纪部

分。

说明

如果 DateFormat 属性设置为 Short 或 Long，则忽略 Century 属性的设置。注意，在早期版本的 Visual FoxPro 中，Century 属性的默认设置为 2。

应用于

文本框

请参阅

DateFormat 属性，DateMark 属性，Hours 属性，Seconds 属性，SETCENTURY，StrictDateEntry 属性

CHANGE 命令

显示要编辑的字段。

语法

```
CHANGE
  [FIELDS FieldList]
  [Scope] [FOR lExpression1] [WHILE lExpression2]
  [FONT cFontName [, nFontSize]]
  [STYLE cFontStyle]
```

[FREEZE FieldName]
[KEY eExpression1 [, eExpression2]]
[LAST | NOINIT]
[LPARTITION]
[NAME ObjectName]
[NOAPPEND]
[NOCLEAR]
[NODELETE]
[NOEDIT | NOMODIFY]
[NOLINK]
[NOMENU]
[NOOPTIMIZE]
[NORMAL]
[NOWAIT]
[PARTITION nColumnNumber [LEDIT] [REDIT]]
[PREFERENCE PreferenceName]
[REST]
[SAVE]
[TIMEOUT nSeconds]
[TITLE cTitleText]
[VALID [:F] lExpression3 [ERROR cMessageText]]
[WHEN lExpression4]

[WIDTH nFieldWidth]
[WINDOW WindowName1]
[IN [WINDOW] WindowNam2 | IN SCREEN | IN MACDESKTOP]]
[COLOR SCHEME nSchemeNumber
| COLOR ColorPairList]

参数描述

CHANGE 的参数与 EDIT 的参数相同。有关参数说明，请参阅 EDIT。

说明

CHANGE 功能与 EDIT 类似。

请参阅

BROWSE, EDIT, WTITLE()

CheckBox 控件

 创建一个复选框。

语法

CheckBox

说明

复选框用来在两个状态，如“真”(.T.)与“假”(.F.)，或“是”与“否”之间切换。若条件是“真”，就在复选框中显示一个“X”。

使用 Caption 属性可指定显示在复选框旁的文本。使用 Picture 属性可以为复选框指定一幅图片。

有关复选框控件的其他信息，请参阅《Microsoft Visual FoxPro6.0 中文版程序员指南》中的第十章“使用控件”。

属性

Alignment
Application
AutoSize
BackColor
BackStyle
BaseClass
Caption
Class
ClassLibrary
ColorScheme
ColorSource
Comment
ControlSource
DisabledBackColor
DisabledForeColor
DisabledPicture
DownPicture
DragIcon
DragMode
Enabled
FontBold
FontCondense
FontExtend
FontItalic
FontName
FontOutline
FontShadow
FontSize
FontStrikeThru
FontUnderline
ForeColor
Height
HelpContextID
Left
MouseIcon
MousePointer
Name
OLEDragMode
OLEDragPicture
OLEDropEffects
OLEDropHasData
OLEDropMode
Parent
ParentClass
Picture

续表

ReadOnly	RightToLeft	SpecialEffect
StatusBarText	Style	TabIndex
TabStop	Tag	TerminateRead
ToolTipText	Top	Value
Visible	WhatsThisHelpID	Width

事件

Click	DblClick	Destroy
DragDrop	DragOver	Error
ErrorMessage	GotFocus	Init
InteractiveChange	KeyPress	LostFocus
Message	MiddleClick	MouseDown
MouseMove	MouseUp	MouseWheel
OLECompleteDrag	OLEDragDrop	OLEDragOver
OLEGiveFeedBack	OLESetData	OLEStartDrag
ProgrammaticChange	RightClick	UIEnable
Valid	When	

方法

AddProperty	CloneObject	Drag

Move OLEDrag ReadExpression
ReadMethod Refresh ResetToDefault
SaveAsClass SetFocus ShowWhatsThis
WriteExpression WriteMethod ZOrder

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

CheckIn 方法

签入源代码管理器中项目的一个文件的更改。

语法

Object.CheckIn( )

说明

如果成功签入该文件，则返回“真” (.T.)。**如果文件不能签入源代码管理器，或者项目不在源代码管理器中，则返回“假”** (.F.)。

应用于

文件对象

请参阅

AddToSCC 方法, CheckOut 方法, GetLatestVersion 方法, RemoveFromSCC 方法, UndoCheckOut 方法

CheckOut 方法

签出源代码管理器中项目的一个文件，允许您更改该文件。

语法

Object.CheckOut( )

说明

如果成功签出该文件，则返回“真” (.T.)。**如果文件不能从源代码管理器签出，或者项目不在源代码管理器中，则返回“假”** (.F.)。

应用于

文件对象

请参阅

AddToSCC 方法, CheckIn 方法, GetLatestVersion 方法, RemoveFromSCC 方法, UndoCheckOut 方法

ChildAlias 属性

指定子表的别名。设计和运行时只读。

语法

DataEnvironment.Relation.ChildAlias[= cText]

参数描述

cText

指定关系中子表的别名。

说明

ChildAlias **属性设置必须与代表子表的临时表对象的** Alias **属性设置相同。**

应用于

关系

请参阅

Alias **属性**, ParentAlias **属性**

ChildOrder 属性

为表格控件或关系对象的记录源指定索引标识。设计时可用，运行时只读。

语法

Object.ChildOrder[= cTagName]

参数描述

cTagName

指定已存在的索引标识名称。

说明

若已设置了 ChildOrder 属性，则忽略子表临时表的 Order 属性。

使用这个属性可连接两个有一对多关系的表。例如，把一个包含每位顾客记录的顾客表与一个包含每位顾客多个订单的订单表相联接。对于这个一对多关系，应把 ChildOrder 属性设置为顾客 ID 索引标识字段。

ChildOrder 属性和 SET ORDER 语句的功能相似。

应用于

表格，关系

请参阅

ControlSource 属性, LinkMaster 属性, SET ORDER

CHR（）函数

根据指定的 ANSI 数值代码返回相应的字符。

语法

CHR(nANSICode)

返值类型

字符型

参数描述

nANSICode

指定一个介于 0 和 255 之间的数值，CHR（）返回与之对应的 ANSI 字符。使用 ASC（）可以返回一个指定字符的 ANSI 值。

说明

CHR（）根据当前代码页字符表中字符的数值位置返回其对应的字符。可用 CHR（）向打印机发送打印控制代码。

示例

下面示例显示从 65 到 75 的数值，并使用 CHR（）显示对应的字符值 A 到 K。

```
CLEAR
FOR nCOUNT = 65 TO 75
   ? nCount && 显示数值
   ?? ' ' + CHR(nCount) && 显示字符 r
```

ENDFOR

请参阅

ASC( ), INKEY( )

CHRSAW（）函数

确定键盘缓冲区中是否有字符。

语法

CHRSAW([nSeconds])

返值类型

逻辑型

参数描述

nSeconds

指定 CHRSAW（）**在检查键盘缓冲区之前等待的秒数。若省略** *nSeconds*，**则立刻检查键盘缓冲区。**

包含 *nSeconds* **可以使** CHRSAW（）**用于多种限时操作中。如若在指定秒数内未按下一键则关闭应用程序。**

说明

若一个字符在键盘缓冲区中出现，则 CHRSAW（）返回“真”(.T.)；否则返回“假”(.F.)。CHRSAW（）不影响键盘缓冲区内容。

示例

在以下示例中，系统显示一个包含输入字段（由 @...GET 命令创建）的窗口，并有 5 秒等候键盘输入。如果没有在这段时间中按下一个键，CHRSAW（）函数返回“假”(.F.) 并中断程序。

```
SET TALK OFF
DEFINE WINDOW wEnter FROM 7,10 to 13,70 PANEL
ACTIVATE WINDOW wEnter
@ 1,3 SAY 'Customer: '  GET gcCustomer  DEFAULT SPACE(40)
@ 3,3 SAY 'Address: '  GET gcAddress  DEFAULT SPACE(40)
WAIT WINDOW 'Waiting for input' NOWAIT
IF NOT CHRSAW(5)
    DEACTIVATE WINDOW wEnter
    CLEAR GETS
ELSE
    READ
    DEACTIVATE WINDOW wEnter
ENDIF
RELEASE WINDOW wEnter
WAIT
CLEAR
```

请参阅

INKEY ( ), KEYBOARD, READKEY( )

CHRTRAN（）函数

在一个字符表达式中，将与第二个表达式字符相匹配的字符替换为第三个表达式中的相应字符。

语法

CHRTRAN(cSearchedExpression, cSearchExpression, cReplacementExpression)

返值类型

字符型

参数描述

cSearchedExpression

指定字符表达式，CHRTRAN（）函数替换其中的字符。

cSearchExpression

指定字符表达式，CHRTRAN（）函数在 *cSearchedExpression* 中寻找此表达式的字符。

cReplacementExpression

指定包含替换字符的表达式，。

若在 *cSearchedExpression* 中发现一个 *cSearchExpression* 中的字符，则用 *cReplacementExpression* 中的一个字符替换。替换字符在 *cReplacementExpression* 中的位

置与被替换字符在 *cSearchExpression* 中的位置相同。

若 *cReplacenentExpression* 中的字符比 *cSearchExpression* 中的字符少，则删除 *cSearchExpression* 中多余的字符。若 *cReplacementExpression* 中的字符比 *cSearchExpression* 中的字符多，则忽略 *cReplacementExpression* 中的多余字符。

说明

CHRTRAN() 使用翻译表达式 *cSearchExpression* 和 *cReplacementExpression* 翻译 *cSearchedExpression*，并返回结果字符串。

示例

```
? CHRTRAN('ABCDEF', 'ACE', 'XYZ') && 显示 XBYDZF
? CHRTRAN('ABCD', 'ABC', 'YZ') && 显示 YZD
? CHRTRAN('ABCDEF', 'ACE', 'XYZQRST') && 显示 XBYDZF
```

请参阅

CHRTRANC(), SYS(15)

CHRTRANC() 函数

将第一个字符表达式中与第二个表达式的字符相匹配的字符替换为第三个表达式中相应的字符。

语法

CHRTRANC(cSearched, cSearchFor, cReplacement)

返值类型

字符型

参数描述

cSearched

指定的表达式，CHRTRANC（）函数对其中的字符进行替换。

cSearchFor

指定的表达式，CHRTRANC（）函数在 *cSearched* 中查找此表达式。

cReplacement

指定包含替换字符的表达式。

若在 *cSearched* 中发现一个 *cSearchFor* 中的字符，则用 *cReplacement* 中的一个字符替换。替换字符在 *cReplacement* 中的位置与被替换字符在 *cSearchFor* 中的位置相同。

若 *cReplacement* 中的字符比 *cSearchFor* 中的字符少，则删除 *cSearchFor* 中多余的字符。若 *cReplacement* 中的字符比 *cSearchFor* 中的字符多，则忽略 *cReplacement* 中多余的字符。

说明

用 CHRTRANC（）可以进行单字节字符与双字节字符之间的替换。如果表达式中只有单字节字符，则 CHRTRANC（）等同于 CHRTRAN()。

该函数适合于处理双字节字符集。

请参阅

CHRTRAN()，STRCONV()

Circle 方法

在表单上画一个圆或椭圆。

语法

Object.Circle (nRadius [, nXCoord, nYCoord [, nAspect]])

参数描述

nRadius

指定圆或椭圆的半径。度量单位由表单的 ScaleMode 属性确定。

nXCoord, nYCoord

指定圆或椭圆的中心坐标。度量单位由表单的 ScaleMode 属性确定。

nAspect

指定圆的纵横比。默认值是 1.0，生成一个正圆（非椭圆）。大于 1.0 的值将生成一个垂直方向椭圆；小于 1.0 的值将生成一个水平方向椭圆。

说明

要控制画圆或椭圆的线宽，可设置 DrawWidth 属性。

要控制在背景中画圆的方式，可设置 DrawMode 和 DrawStyle 属性。

要填充圆，可设置表单的 FillColor 和 FillStyle 属性。

当激活 Circle 方法时，CurrentX 和 CurrentY 属性被设置为中心点参量：*nXCoord*，*nYCoord*。

应用于

表单，_SCREEN

请参阅

BackColor 属性，CurrentX，CurrentY 属性，DrawMode 属性，DrawStyle 属性，DrawWidth 属性，FillColor 属性，FillStyle 属性，ForeColor 属性，ScaleMode 属性

Class 属性

返回一个对象的基类名称。设计和运行时只读。

语法

Object.Class

说明

在类设计器中，Class 属性返回类的实际名称。

应用于

ActiveDoc 对象，复选框，列，组合框，命令按钮，命令组，容器对象，控件对象，定制，编辑框，表单，表单集，表格，标头，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，ProjectHook 对象，形状，_SCREEN，微调，文本框，计时器，工具栏。

请参阅

ClassLibrary 属性

指定用户自定义类库名，此类库中包含自定义的对象类。

语法

Object.ClassLibrary

说明

设计和运行时只读。Visual FoxPro 基类对象已连编进产品中，没有 ClassLibrary 设置。

应用于

ActiveDoc 对象，复选框，列，组合框，命令按钮，命令组，容器对象，控件对象，定制，编辑框，表单，表单集，表格，标头，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，ProjectHook 对象，_SCREEN，形状，微调，文本框，计时器，工具栏

请参阅

BaseClass 属性

CleanUp 方法

通过删除标记为删除的记录，并且删除备注字段，来清理项目表。

语法

Object.CleanUp(lRemoveObjectCode)

参数描述

lRemoveObjectCode

指定是否删除保存在项目表对象字段中的对象代码。如果 lRemoveObjectCode 为“真” (.T.)，则从项目表中删除对象代码。如果 lRemoveObjectCode 为“假” (.F.) 或省略了，则不删除对象代码。

对象字段包含项目中程序 (.prg)、菜单 (.mnx) 和查询 (.qpr) 文件的对象代码。如果删除保存在项目表对象字段中的对象代码，则在下次重新连编项目时，重新编译源代码，并且再次将对象代码保存在对象字段中。

说明

每个项目都有相应的表，其中包含项目信息。使用 CleanUp 方法可以通过删除标记为删除的记录，并且删除备注字段来减小这个表的大小。清理项目表可以减少重新连编一个大项目的时间，或者减少根据一个大项目创建应用程序文件 (.app)、动态链接库 (.dll) 或可执行文件 (.exe) 的时间。

应用于

项目对象

请参阅

BUILD APP, BUILD DLL, BUILD EXE, Build **方法**, BUILD PROJECT, CREATE PROJECT, MODIFY PROJECT

CLEAR **命令**

从内存中释放指定项。

语法

```
CLEAR
  [ALL | CLASS ClassName | CLASSLIB ClassLibraryName | DEBUG
  | DLLS | EVENTS | FIELDS | GETS | MACROS | MEMORY
  | MENUS | POPUPS  | PROGRAM | PROMPT | READ [ALL]
  | RESOURCES [FileName] | TYPEAHEAD | WINDOWS]
```

参数描述

ALL

从内存中释放所有的变量和数组以及所有用户自定义菜单栏、菜单和窗口的定义。CLOSE ALL **也能关闭所有表，包括所有相关的索引、格式和备注文**

件，并且选择工作区 1。CLEAR ALL 还从内存中删除所有用 DECLARE-ALL 注册的外部 Windows 32 位动态链接库 (.DLLS)。

CLEAR ALL 不释放系统变量，也不清除已编译程序的缓冲区。应使用 CLEAR PROGRAM 来清除已编译程序的缓冲区。

在活动控制或对象的事件或方法内执行 CLEAR ALL 时，Visual FoxPro 会产生错误信息。当对象类型变量的相关控制或对象活动时，不能将其从内存中释放。

CLASS ClassName

从内存中清除类的定义。如果创建了类的一个实例，释放该实例以后，Visual FoxPro 仍在内存中保存类定义。释放实例以后，应使用 CLEAR CLASS 从内存中清除类定义。

CLASSLIB ClassLibraryName

从内存中清除所有包含在可视类库中的类定义。若类库中的类的实例仍然存在，则不从内存中清除类定义，并产生错误信息。但是，内存中所有没有实例的类定义会被清除。

DEBUG

清除调试器中的所有断点，并且将调试窗口（调用堆、跟踪、查看等）恢复到默认位置。。

如果当调试器是关闭时发出 Clear Debug 命令，则打开调试器，并且调试窗口位于默认位置。

在 fox 或调试器框架模式中的工作。

DLLS

从内存中清除所有用 DECLARE-DLL 注册的外部 Windows 32 位动态链接库(.DLLS)。有关注册外部 DLL 的详细内容，请参阅 DECLARE-DLL

EVENTS

停止以 READ EVENTS 开始的事件处理。当执行 CLEAR EVENTS 后，程序从 READ EVENTS 的下一条程序行继续执行。

FIELDS

释放用 SET FIELDS 命令创建的列表，并且执行 SET FIELDS OFF。CLEAR FIELDS 与 SET FIELDS TO 的不同之处在于，它释放所有工作区中的所有字段列表，不仅仅是当前工作区中的字段列表。另外，SET FIELD TO 不会隐含执行 SET FIELDS OFF。

GETS

释放所有等待的 @...GET 控制。执行 CLEAR 也可以释放所有等待的 @...GET 控制。

MACROS

从内存中释放所有键盘宏，包括任何用 SET FUNCTION 指定的功能键定义。可用 SAVE MACROS 将宏存入一个宏文件或备注字段中，并在以后用 RESTORE MACROS 命令还原。也可用 RESTORE MACROS 还原默认宏。

MEMORY

从内存中释放所有公共或私有变量和数组，但不释放系统变量。

MENUS

释放内存中所有的菜单栏定义。

POPUPS

释放内存中所有用 DEFINE POPUP 命令创建的菜单定义。

PROGRAM

清除已编译程序的缓冲区。Visual FoxPro 保存最近执行程序的缓冲区。Visual FoxPro 可能不认可对磁盘上程序文件所作的更改，但这种情况很少出现。CLEAR PROGRAM 迫使 Visual FoxPro 从磁盘上而不是程序缓冲区读程序。Visual FoxPro 不认可对程序文件所作更改的最可能的原因是使用了外部或常驻内存 (TSR) 编辑器修改程序文件。除此之外，不必使用 CLEAR PROGRAM。

PROMPT

释放用 @...PROMPT 创建的菜单项。

READ [ALL]

包含此子句是为了提供向后兼容性。可用 CLEAR EVENTS 代替。

RESOURCES [FileName]

指定将被清除出内存的位图 (bitmap)，图片，字体，光标，或图标等资源的文件。如果不指定具体文件名，所有这些文件就都将从内存中清除。

当 Visual FoxPro 显示位图，图片，字体，光标，或图标时，这些资源被调入到内存中；下次再使用它们时，直接由内存中读取，这样提高了程序的速度。但是，如果两个资源文件具有相同的名称（比如，第二次调用的位图与已在内存中的位图具有相同的名称，而实际上不是同一个资源），Visual FoxPro 并不重新调入新资源，从而可能引发错误。

所以，将图片等资源从内存中清除就十分有用，它迫使 Visual FoxPro 在下次使用资源

时重新从磁盘中读取。例如，一个报表需要显示数据库中的一系列图片，而所有图片的名称都是 TEMP；这时，如果您不每次使用 CLEAR RESOURCES 命令将显示过的资源清除出内存，Visual FoxPro 不会自动调入新的图片。

TYPEAHEAD

清除键盘缓冲区。若要在显示字段或提示之前禁止向字段输入或对提示应答，CLEAR TYPEAHEAD 很有用。

WINDOWS

释放内存中所有用户自定义窗口的定义，并从 Visual FoxPro 主窗口或活动的用户自定义窗口中清除窗口。使用 SAVE WINDOW 可将窗口定义存入文件或备注字段以备后用。

执行 CLEAR WINDOWS 将释放任何对表单的变量引用。例如，下列命令创建一个对表单的变量引用，然后显示有关变量的信息：

```
goMyForm = CREATEOBJECT('FORM')
DISPLAY MEMORY LIKE goMyForm && 显示 GOMYFORM O FORM
```

执行 CLEAR WINDOWS 释放变量引用，则当前变量中包含 null 值：

```
CLEAR WINDOWS
DISPLAY MEMORY LIKE goMyForm && 显示 GOMYFORM O .NULL.
```

说明

CLEAR 删除 Visual FoxPro 主窗口或当前用户自定义窗口，并且释放内存中所有等待的 @...GET 控制。可在格式文件中包含 CLEAR。

请参阅

@...CLEAR, CLOSE, DECLARE-DLL, READ, READ EVENTS, RELEASE, RELEASE CLASSLIB, RELEASE WINDOWS

Clear 方法

清除组合框或列表框控件的内容。

语法

Object.Clear

说明

为使 Clear **方法有效，必须将** RowSourceType **属性设置为** 0（**无**）。

应用于

组合框，列表框

请参阅

AddItem **方法**, RemoveItem **方法**

ClearData 方法

从 OLE 拖放 DataObject 对象中清除所有的数据和数据格式。只在设计时可用。

语法

oDataObject.ClearData

说明

ClearData 方法只能在 OLEStartDrag 事件中执行；从 OLEDragDrop 或 OLEDragOver 事件中调用 ClearData 方法，会产生错误。

应用于

DataObject 对象

请参阅

GetData 方法，GetFormat 方法，OLE drag-and-drop 概述，SetData 方法，SetFormat 方法

Click 事件

当在程序中包含触发此事件的代码，或者将鼠标指针放在一个控件上按下并释放鼠标左键，或者更改特定控件的值，或在表单空白区单击时，此事件发生，

语法

PROCEDURE Object.Click
[LPARAMETERS nIndex]

参数描述

nIndex

唯一标识控件数组中的一个控件。

说明

Click 事件发生在：

- 用鼠标左键单击复选框、命令按钮、列表框或选项按钮控件时。
- 在命令按钮、选项按钮或复选框有焦点时按 SPACEBAR。
- 表单中有 Default 属性设置为“真”(.T.)的命令按钮并且按 ENTER 时。
- 按一个控件的快捷键。例如，若一个命令按钮的标题为“\<Go”，则按 ALT+G 可触发 Click 事件。.
- 单击表单空白区。当指针位于标题栏或窗口边界上时，不发生表单的 Click

事件。

- 单击微调控件的文本输入区。
- 单击禁止的控件时，废止控件所在表单的 Click 事件发生。

Click 事件也可能由于包含了执行 MOUSE 命令的代码而发生。

示例

以下示例创建了一个“选项按钮组”控件并将控件放置在表单上。“选项按钮组”有三个按钮，并根据您选择的选项按钮，出现圆形、椭圆形或方形。使用 ButtonCount 属性指定“选项按钮组”的按钮数目。Buttons 和 Caption 属性用于指定显示紧接着每个选项按钮的文本。

使用“形状”控件创建圆形、椭圆形和方形。“选项按钮组”控件的 Click 事件使用 DO CASE...ENDCASE 结构和 Value 属性显示与您选择的选项按钮相对应的形状。

```
frmMyForm = CREATEOBJECT('Form') && 创建表单
frmMyForm.Closable = .F. && 禁止窗口弹出菜单

frmMyForm.AddObject('cmdCommand1','cmdMyCmndBtn') && 添加命令按钮
frmMyForm.AddObject('opgOptionGroup1','opgMyOptGrp') && 添加选项按钮组
frmMyForm.AddObject('shpCircle1','shpMyCircle') && 添加圆形
frmMyForm.AddObject('shpEllipse1','shpMyEllipse') && 添加椭圆形
frmMyForm.AddObject('shpSquare','shpMySquare') && 添加方框

frmMyForm.cmdCommand1.Visible =.T. && “退出”命令按钮可视

frmMyForm.opgOptionGroup1.Buttons(1).Caption = "圆形(\<C)"
frmMyForm.opgOptionGroup1.Buttons(2).Caption = "椭圆形(\<E)"
frmMyForm.opgOptionGroup1.Buttons(3).Caption = "方形(\<S)"
frmMyForm.opgOptionGroup1.SetAll("Width", 100) && 设置选项按钮组宽度
```

```
frmMyForm.opgOptionGroup1.Visible = .T. && 选项按钮组可视
frmMyForm.opgOptionGroup1.Click  && 显示圆形

frmMyForm.SHOW  && 显示表单
READ EVENTS  && 启动事件处理

DEFINE CLASS opgMyOptGrp AS OptionGroup  && 创建选项按钮组
    ButtonCount = 3  && 三个选项按钮
    Top = 10
    Left = 10
    Height = 75
    Width = 100

    PROCEDURE Click
        ThisForm.shpCircle1.Visible = .F.  && 隐藏圆形
        ThisForm.shpEllipse1.Visible = .F.  && 隐藏椭圆形
        ThisForm.shpSquare.Visible = .F.  && 隐藏方形

        DO CASE
            CASE ThisForm.opgOptionGroup1.Value = 1
                ThisForm.shpCircle1.Visible = .T. && 显示圆形
            CASE ThisForm.opgOptionGroup1.Value = 2
                ThisForm.shpEllipse1.Visible = .T.  && 显示椭圆形
            CASE ThisForm.opgOptionGroup1.Value = 3
                ThisForm.shpSquare.Visible = .T.  && 显示方形
        ENDCASE
ENDDEFINE
```

```
DEFINE CLASS cmdMyCmndBtn AS CommandButton && 创建命令按钮
    Caption = '退出(\<Q)' && 命令按钮上的标题
    Cancel = .T. && 默认取消命令按钮(Esc)
    Left = 125 && 命令按钮列
    Top = 210 && 命令按钮行
    Height = 25 && 命令按钮高

    PROCEDURE Click
        CLEAR EVENTS && 停止事件处理，关闭表单
ENDDEFINE

DEFINE CLASS shpMyCircle AS SHAPE && 创建圆形
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 99
    BackColor = RGB(255,0,0) && 红色
ENDDEFINE

DEFINE CLASS shpMyEllipse AS SHAPE && 创建椭圆形
    Top = 35
    Left = 200
    Width = 100
    Height = 50
    Curvature = 99
```

```
    BackColor = RGB(0,128,0) && 绿色
ENDDEFINE

DEFINE CLASS shpMySquare AS SHAPE && 创建方形
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 0
    BackColor = RGB(0,0,255) && 兰色
ENDDEFINE
```

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，标头，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

DblClick 事件，MiddleClick 事件，MOUSE，MouseDown 事件，MouseUp 事件，MouseWheel 事件，Value 属性

ClipControls 属性

确定 Paint 事件中的图形方法是重画整个对象还是只重画新露出的区域，也确定 Windows 操作环境是否创建一个剪裁区域来除去对象中的非图形控件。

语法

Object.ClipControls[= lExpr]

参数描述

lExpr

ClipControls 属性的设置为：

设置	说明
“真”(.T.)	（默认值） Paint 事件中的图形方法重画整个对象。Paint 事件发生之前在表单的非图形控件周围创建一个剪裁区。
“假”(.F.)	Paint 事件中的图形方法只重画新露出的区域。不在非图形控件周围创建剪裁区。

应用于

表格，_SCREEN

请参阅

_CLIPTEXT 系统变量

包含剪贴板的内容。

语法

_CLIPTEXT = cExpression

参数描述

cExpression

指定要存储于剪贴板中的字符表达式。

说明

可用 STORE 或“ = ”赋值操作符将表达式 *cExpression* 放置到剪贴板上。

请参阅

PRIVATE, PUBLIC, STORE

CloneControl 方法

复制对象，包括对象所有的属性、事件和方法。仅设计时可用。

语法

Object.CloneObject(NewName)

参数描述

NewName

指定新建或复制的对象名。

应用于

复选框，组合框，命令按钮，命令组，定制，编辑框，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，形状，微调，文本框，计时器，工具栏

请参阅

SaveAs 方法, SaveAsClass 方法

Closable 属性

指定能否由双击窗口弹出菜单框，或从窗口弹出菜单中选择“关闭”项，或通过单击“关闭”按钮来关闭表单。设计和运行时可用。

语法

Object.Closable[= lExpr]

参数描述

lExpr

下表列出了 Closable 属性的设置：

设置	说明
“真”(.T.)	（默认值）可关闭表单，并将“关闭”菜单命令添加到窗口弹出菜单中。
“假”(.F.)	不能使用窗口弹出菜单关闭表单，并从窗口弹出菜单中删除“关闭”菜单命令。

示例

下面示例演示了如何使用 Closable 属性禁止用窗口弹出菜单关闭表单。表单的 Closable 属性设置为“假”(.F.)，禁止用窗口弹出菜单关闭表单。

若表单的 closable 属性已经设置为“真”（.T.），并且已用窗口弹出菜单关闭了此表

```
    单，则必须执行 CLEAR EVENTS 终止事件处理，并退出程序。
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. && 使控件菜单框无效
                         && 并关闭按钮

frmMyForm.AddObject('shpLine','Line') && 向表单添加一个 Line 控件
frmMyForm.AddObject('cmdCmndBtn1','cmdMyCmndBtn1') && Up 命令按钮
frmMyForm.AddObject('cmdCmndBtn2','cmdMyCmndBtn2') && Down 命令按钮
frmMyForm.AddObject('cmdCmndBtn3','cmdMyCmndBtn3') && Quit 命令按钮

frmMyForm.shpLine.Visible = .T. && 使 Line 控件可见
frmMyForm.shpLine.Top = 20 && 指定 Line 控件所在行
frmMyForm.shpLine.Left = 125 && 指定 Line 控件所在列

frmMyForm.cmdCmndBtn1.Visible =.T. && Up 命令按钮可见
frmMyForm.cmdCmndBtn2.Visible =.T. && Down 命令按钮可见
frmMyForm.cmdCmndBtn3.Visible =.T. && Quit 命令按钮可见

frmMyForm.SHOW && 显示表单
READ EVENTS && 开始事件处理

DEFINE CLASS cmdMyCmndBtn1 AS COMMANDBUTTON && 创建命令按钮
    Caption = 'Slant \<Up' && 命令按钮标题
    Left = 50 && 命令按钮所在列
    Top = 100 && 命令按钮所在行
    Height = 25 && 命令按钮高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. && 隐藏 Line 控件 l
        ThisForm.shpLine.LineSlant ='/' && 向上倾斜
        ThisForm.shpLine.Visible = .T. && 显示 Line 控件
```

```
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn2 AS CommandButton && 创建命令按钮
    Caption = 'Slant \<Down' && 命令按钮标题
    Left = 200 && 命令按钮所在列
    Top = 100 && 命令按钮所在行
    Height = 25 && 命令按钮高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. && 隐藏 Line 控件
        ThisForm.shpLine.LineSlant ='\' && 向下倾斜
        ThisForm.shpLine.Visible = .T. && 显示 Line 控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn3 AS CommandButton &&创建命令按钮
    Caption = '\<Quit' && 命令按钮标题
    Cancel = .T. && 默认取消命令(Esc)
    Left = 125 && 命令按钮所在列
    Top = 150 && 命令按钮所在行
    Height = 25 && 命令按钮高度

    PROCEDURE Click
        CLEAR EVENTS && 停止事件处理， 关闭表单
ENDDEFINE
```

应用于

表格，_SCREEN

请参阅

ControlBox **属性**，TitleBar **属性**

CLOSE 命令

关闭各种类型的文件。

语法

CLOSE
[ALL | ALTERNATE | DATABASES [ALL] | DEBUGGER
| FORMAT | INDEXES | PROCEDURE | TABLES [ALL]]

参数描述

ALL

关闭所有工作区中打开的数据库、表和索引，并选择工作区 1。CLOSE ALL 也关闭任何用 FCREATE（）和 FOPEN（）低级文件功能打开的文件。

CLOSE ALL 关闭：

- 表单设计器
- 项目管理器
- 标签设计器
- 报表设计器
- 查询设计器

但 CLOSE ALL 不关闭：

- 命令窗口

- 调试窗口
- 帮助系统
- 跟踪窗口

CLOSE ALTERNATE

关闭用 SET ALTERNATE 打开的替代文件。

CLOSE DATABASES [ALL]

关闭当前数据库和表。若没有当前数据库，则关闭所有工作区内所有打开的自由表、索引和格式文件，并选择工作区 1。

ALL

指定关闭：

- 所有打开的数据库和其中的表。
- 所有打开的自由表。
- 所有工作区内所有索引和格式文件。

选择工作区 1。

CLOSE DEBUGGER

关闭 Visual FoxPro 调试器。

CLOSE FORMAT

关闭当前工作区内用 SET FORMAT 打开的格式文件。

CLOSE INDEXES

关闭当前工作区内所有打开的索引文件（只有单项索引 .IDX 和独立复合 .CDX 文件）。不关闭结构复合索引（与表自动同时打开的 .CDX 文件）。

CLOSE PROCEDURE

关闭用 SET PROCEDURE 打开的过程文件。

CLOSE TABLES [ALL]

关闭所有当前选中数据库中的所有表。若没有已打开的数据库，则 CLOSE TABLES 关闭所有工作区内的自由表。

包含 ALL 可以关闭所有数据库中的所有表以及自由表，但所有数据库保持打开。

若正在执行某一事务，则不应执行 CLOSE TABLES；否则 Visual FoxPro 产生错误信息。

请参阅

ADD TABLE, CLOSE MEMO, CREATE DATABASE

Close 方法

关闭项目，并且释放项目的项目挂接对象和项目对象。

语法

Object.Close( )

说明

执行 Close 方法会引起项目的 ProjectHook.Destroy 事件的发生。在 ProjectHook.Destroy

事件发生之后，会释放这个项目挂接对象，然后释放项目对象。

应用于

项目对象

请参阅

CLOSE 命令, Destroy 事件

CLOSE MEMO 命令

关闭一个或多个备注编辑窗口。

语法

CLOSE MEMO MemoFieldName1 [, MemoFieldName2 ...] | ALL

参数描述

MemoFieldName1 [, MemoFieldName2 ...]

指定备注字段名，以关闭其备注编辑窗口。若要关闭一组备注编辑窗口，可以包括一列用逗号分隔的备注字段名。通过包括表别名可以关闭另外一个工作区的打开表中的备注字段的备注编辑窗口。

ALL

关闭任何打开表中所有备注字段的备注编辑窗口。

说明

CLOSE MEMO 关闭用 MODIFY MEMO 打开或从浏览窗口或编辑窗口中打开的备注编辑窗口。CLOSE MEMO 保存对备注字段的任何更改。

关闭包含备注字段的表会保存所做的任何更改，并关闭备注窗口。

请参阅

BROWSE, MODIFY MEMO

CloseTables 方法

关闭与数据环境相关的表和视图。

语法

DataEnvironment.CloseTables

说明

通过调用 OpenTables 方法可以重新打开表和视图。

应用于

数据环境

请参阅

AfterCloseTables 事件, AutoCloseTables 属性, BeforeOpenTables 事件, OpenTables 方法

Cls 方法

清除表单中的图形和文本。

语法

Object.Cls

说明

Cls 清除运行期间图形和打印语句生成的文本和图形。Cls 方法不影响设计期间放置在表单上的背景位图。

Cls 方法将 CurrentX 和 CurrentY 属性重新设置为 0。

应用于

表单，_SCREEN

请参阅

Circle 方法，CurrentX，CurrentY 属性，Line 方法，Picture 属性

CLSID 属性

包含项目中一个服务程序的已注册 CLSID (Class Identifier)。设计和运行时只读。

语法

Object.CLSID

说明

CLSID 是在根据项目连编一个可执行文件 (.exe) 或动态链接库 (.dll) 时，在 Windows 注册表中为一个服务程序创建的。

应用于

Server 对象

请参阅

CREATEOBJECTEX(), ProgID 属性, TypeLibCLSID 属性, TypeLibDesc 属性, TypeLibName 属性

CMONTH（）函数

返回一个给定日期或时间表达式的月份。

语法

CMONTH(dExpression | tExpression)

返值类型

数值型

参数描述

dExpression

指定从 CMONTH（）返回月份名的日期表达式。

tExpression

指定从 CMONTH（）返回月份名的日期时间表达式。

说明

CMONTH（）返回的月份名称是以一定名词格式的字符串。

示例

```
? CMONTH(DATE( ))
STORE {^1998-02-16} TO gdDueDate
? 'Your payment was due in ', CMONTH(gdDueDate)
STORE gdDueDate+60 TO gdFinalDate
? 'You must pay by ', CMONTH(gdFinalDate)
```

请参阅

DMY( ), MDY( ) , MONTH( )

CNTBAR（）函数

返回系统菜单或用户定义菜单的栏目数。

语法

CNTBAR(cMenuName)

返值类型

数值型

参数描述

cMenuName

为 CNTBAR（）**函数指定菜单的名称，该函数返回菜单项的数目。有关** Visual FoxPro **系统菜单名，请参阅稍后部分的语言参考“系统菜单名称”。**

说明

如果在 DEFINE POPUP **命令中使用** PROMPT **选项创建用户自定义的菜单，当执行** ACTIVEATE POPUP **命令时** Visual FoxPro **计算菜单项数目。对于这样的菜单，** CNTBAR（）**只有在您激活菜单时才返回一个有意义的值。但是，如果使用** DEFINE

BAR 创建菜单中的菜单项，CNTBAR（）可以在执行 ACTIVEATE POPUP 命令之前确定菜单项的数目。

示例

在以下程序示例中，程序名为 CNTBAR.PRG，向系统菜单中添加一个菜单标题。创建的 popEnv 菜单具有四个菜单项。当选定一个菜单项时此程序为每个菜单项放置一个标记字符。在循环语句中使用 CNTBAR（）函数初始显示与每个相应菜单项紧接着的标记。

```
*** You must name this program CNTBAR.PRG ***
CLEAR
SET TALK OFF
DEFINE PAD padEnv OF _MSYSMENU PROMPT 'E\<nvironment';
    KEY ALT+R, 'ALT+R'
ON PAD padEnv OF _MSYSMENU ACTIVATE POPUP popEnv
DEFINE POPUP popEnv MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF popEnv PROMPT '\<Status Bar'
DEFINE BAR 2 OF popEnv PROMPT '\<Clock'
DEFINE BAR 3 OF popEnv PROMPT '\<Extended Video'
DEFINE BAR 4 OF popEnv PROMPT 'St\<icky'
ON SELECTION POPUP popEnv DO enviropop IN cntbar.prg
FOR i = 1 TO CNTBAR('popEnv')
    DO CASE
        CASE PRMBAR('popEnv', i) = 'Status Bar'
            IF _WINDOWS or _MAC
                SET MARK OF BAR i OF popEnv TO SET('STATUS BAR') = 'ON'
            ELSE
                SET MARK OF BAR i OF popEnv TO SET('STATUS') = 'ON'
            ENDIF
        CASE PRMBAR('popEnv', i) = 'Clock'
```

```
            SET MARK OF BAR i OF popEnv TO  SET('CLOCK') = 'ON'
        CASE PRMBAR('popEnv', i) = 'Extended Video'
            SET MARK OF BAR i OF popEnv TO  SROW ( ) > 25
        CASE PRMBAR('popEnv', i) = 'Sticky'
            SET MARK OF BAR i OF popEnv TO  SET('STICKY') = 'ON'
    ENDCASE
ENDFOR
PROCEDURE enviropop
DO CASE
    CASE PROMPT() = 'Status'
        IF mrkbar('popEnv', BAR( ))
            DO CASE
                CASE _WINDOWS OR _MAC
                    SET STATUS BAR OFF
                CASE _DOS
                    SET STATUS OFF
                OTHERWISE
            ENDCASE
            SET MARK OF BAR BAR ( ) OF popEnv TO .F.
        ELSE
            DO CASE
                CASE _WINDOWS OR _MAC
                    SET STATUS BAR ON
                CASE _DOS
                    SET STATUS ON
                OTHERWISE
            ENDCASE
            SET MARK OF BAR BAR ( ) OF popEnv TO .T.
        ENDIF
```

```
CASE PROMPT ( ) = 'Clock'
    IF mrkbar('popEnv', BAR( ))
        SET CLOCK OFF
        SET MARK OF BAR BAR ( ) OF popEnv TO .F.
    ELSE
        DO CASE
            CASE _WINDOWS OR _MAC
                SET STATUS BAR ON
                SET CLOCK STATUS
            CASE _DOS
                SET CLOCK ON
            OTHERWISE
        ENDCASE
        SET MARK OF BAR BAR ( ) OF popEnv TO .T.
    ENDIF
CASE PROMPT ( ) = 'Extended Video'
    IF MRKBAR('popEnv', BAR( ))
        SET DISPLAY TO VGA25
        SET MARK OF BAR BAR ( ) OF popEnv TO .F.
    ELSE
        SET DISPLAY TO VGA50
        SET MARK OF BAR BAR ( ) OF popEnv TO .T.
    ENDIF
CASE PROMPT ( ) = 'Sticky'
    IF MRKBAR('popEnv', BAR( ))
        DO CASE
            CASE _WINDOWS OR _MAC
                WAIT WINDOW 'STICKY is always on in this Visual FoxPro version'
            CASE _DOS
```

```
            SET STICKY OFF
         OTHERWISE
      ENDCASE
      SET MARK OF BAR BAR ( ) OF popEnv TO .F.
   ELSE
      DO CASE
         CASE _WINDOWS OR _MAC
            WAIT WINDOW 'STICKY is always ON in Visual FoxPro'
         CASE _DOS
            SET STICKY ON
         OTHERWISE
      ENDCASE
      SET MARK OF BAR BAR ( ) OF popEnv TO .T.
   ENDIF
ENDCASE
```

请参阅

ACTIVATE POPUP, CNTPAD( ), CREATE MENU, DEFINE BAR, DEFINE POPUP, GETBAR( ) , MRKBAR( ) , ON BAR, ON SELECTION BAR, PRMBAR( )

CNTPAD（）函数

返回用户自定义菜单栏或 Visual FoxPro 系统菜单栏上的菜单标题的数目。

语法

CNTPAD(cMenuBarName)

返值类型

数值型

参数描述

cMenuBarName

为 CNTPAD（）函数指定菜单栏数目，该函数返回菜单标题的数目。

示例

以下命令使用 CNTPAD（）函数显示在 Visual FoxPro 系统菜单栏中的菜单标题数。

```
? CNTPAD('_MSYSMENU')
```

请参阅

ACTIVATE MENU, CNTBAR( ), CREATE MENU, DEFINE PAD, DEFINE MENU, GETPAD( ) , MRKPAD( ) , ON PAD, ON SELECTION PAD, PRMPAD( )

CodePage 属性

包含项目中一个文件的代码页。设计和运行时只读。

语法

Object.CodePage

说明

代码页是一个数值，表明了文件所使用的字符集。代码页经常对应于不同的平台和语言，并且用在国际化应用程序中。

应用于

文件对象

请参阅

CPCONVERT( ), CPCURRENT( ), CPDBF( )

COL（） 函数

包含此函数为了向后兼容。可用 Current X 属性代替。

颜色概述

Visual FoxPro 对颜色的完全控制提供了一个复杂的命令系列。

默认情况下,Visual FoxPro 从操作系统的“控制面板”的设置中得到自身的颜色。在启动时，“控制面板”颜色映射为 Visual FoxPro 默认的颜色方案。可以直接使用 SET COLOR 命令设置颜色或在“控制面板”中交互地设置。有关使用“控制面板”设置颜色的详细信息，请参考您的 Windows 文档。

不是所有的 Visual FoxPro 界面元素都可以通过颜色方案来控制 — 例如：“数据工作期”和“命令”窗口、系统菜单栏等系统元素。

颜色术语

会在 FoxPro 文档中用到下列颜色术语。

颜色对

一个颜色对包含两个颜色代码，它们指定了前景和背景颜色。颜色对由两个用斜杠分隔的字母组成；第一个字母指定了前景颜色，第二个字母指定了背景颜色。

例如，下列颜色对指定在白色背景中的红色前景：

R/W

下表列出了可用的颜色和它们的代码。

颜色	代码
Black	N
Blank	X
Blue	B
Brown	GR
Cyan	BG
Green	G
Inverse	I
Magenta	RB
Red	R
White	W
Yellow	GR+
Underlined	U

在一个颜色代码的后面紧跟着一个星号 (*)，可以用来表明闪烁或亮度（这取决于您的

显示硬件和 SET BLINK 的设置）。在 Visual FoxPro 中，包含一个星号会使背景更亮，而不是闪烁。在一个颜色代码的后面紧跟着一个加号 (+)，可以用来表明背景颜色的高亮度。

对于单彩显示器，使用四种颜色可用： white (W), black (N), underlined (U), and inverse video (I)。 blank (X) 颜色可用于输入密码。

RGB 颜色对

也可以使用一组六个 RGB (red, green, and blue) 颜色值（用逗号分隔）指定一个颜色对。这些值的取值范围从 0（最低亮度或没颜色）到 255（高亮度或亮颜色）。每个前景和背景颜色都需要三个值，一个用于红色，一个用于绿色，一个用于蓝色。所以，一个颜色对需要六个值，三个用于指定前景，三个用于指定背景。

下面是深灰色（高亮度的黑色）上面的红色的 RGB 颜色代码：

```
RGB(255,0,0,64,64,64)
```

在上面的 RGB 表达式中，前三个值将前景颜色设置为红色，后三个值将背景颜色设置为深灰色。

下表列出了 Visual FoxPro 中可用的颜色，包括颜色代码和相应的 RGB 值。

颜色	颜色代码	RGB 值
White	W+	255,255,255
Black	N	0,0,0
Dark Gray	N+	64,64,64 (25% gray)

续表

Gray	W	192,192,192
Red	R+	255,0,0
Dark Red	R	128,0,0
Yellow	GR+	255,255,0
Dark Yellow	GR	128,128,0
Green	G+	0,255,0
Dark Green	G	0,128,0
Cyan	BG+	0,255,255
Dark Cyan	BG	0,128,128
Blue	B+	0,0,255
Dark Blue	B	0,0,128
Magenta	RB+	255,0,255
Dark Magenta	RB	128,0,128
Blank	X	N/A

当使用 RGB 值时，下列规则决定了颜色：

- 当所有的三个颜色值 (R, G or B) 小于 32 时，颜色是黑色的。
- 当所有的三个颜色值 (R, G or B) 在 32 和 64 之间时，字符颜色是深灰色的。
- 当所有的三个颜色值 (R, G or B) 在 65 和 191 之间时，字符颜色是灰色的。
- 当任何一个颜色值 (R, G or B) 大于 191 时，颜色是高亮度的 (+ 或 *)。

颜色对列表

一个颜色对列表包含 1 至 10 个用逗号分隔的颜色对。例如：
W+/B, W+/BG, GR+/B, GR+/B, R+/B, W+/GR, GR+/RB, N+/N, GR+/B, R+/B

一个单彩显示器的颜色对列表可以是：
W/N, N+/W, W+/N, W+/N, W/N, U+/N, W+/N, -, W+/N, W/N

单彩显示器的颜色对列表在第八个颜色对位置可能有一个短线 (-)，表明没有阴影。

颜色对也可以指定为一组 RGB (red, green, and blue) 值。一组 RGB 颜色值可以是：
RGB(0,255,0,255,0,0), RGB(127,255,0,0,0,0), ...

配色方案

配色方案是一组 10 个颜色对。使用 SCHEME（）或 RGBSCHEME（）函数可以返回一个配色方案的颜色对。

配色方案控制了界面元素的颜色，例如系统窗口、用户自定义窗口、菜单等等。在 Visual FoxPro 中，有些界面元素的颜色不由配色方案控制，例如，“数据工作期”窗口和“命令”窗口、系统菜单栏等等的颜色，就不由配色方案控制。

在配置文件中，可以指定自己的启动配色方案。对于每个想要改变的配色方案，可以包含下面的代码：
COLOR OF SCHEME nScheme = Colorpairlist

注意 在 Visual FoxPro 中，配色方案 13 到 15 保留为内部使用。在 FoxPro for Windows 中，配色方案 13 和 14 保留为内部使用。在 FoxPro for Macintosh 中，配色方案 13 到 16 保留为内部使用。请不要使用这些配色方案。

颜色集合

一个颜色集合包含 24 个配色方案。在颜色集合中，可以保存完整的颜色环境。在 FoxPro for Macintosh 中，不支持颜色集合。

与键盘宏和变量类似，颜色集合也可以保存起来，以后再用。可以使用 CREATE COLOR SET 保存一个颜色集合。颜色集合保存在 Foxuser.dbf 资源文件中。

可以使用 SET COLOR SET 加载一个颜色集合。

当第一次启动 Visual FoxPro 时，“控制面板”中的颜色设置就加载到默认的颜色集合中。若要在您的配置文件中指定一个启动颜色集合，可以包含以下代码：

```
COLOR SET = ColorSetName
```

请参阅

CREATE COLOR SET, GETCOLOR( ), RGB( ) , RGBSCHEME( ) , SET COLOR OF SCHEME, SET COLOR SET, SET COLOR TO

ColorScheme 属性

指定控件使用的配色方案。包含此属性是为了提供向后兼容性。可使用 BackColor、ForeColor 属性代替。

ColorSource 属性

确定如何设置控件的颜色。设计和运行时可用。

语法

Object.ColorSource[= nSource]

参数描述

nSource

下表列出了 *nSource* 的设置：

设置	说明
0	对象的 Color 属性。控件使用对象的 Color 属性设置（前景色、背景色等）。本设置对以下控件和对象可用： 复选框，组合框，命令按钮，命令按钮组，容器，控件对象，编辑框，表单，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，图形，微调，文本框和工具栏
1	表单配色方案。控件使用所在表单的配色方案。本设置对以下控件和对象可用： 复选框，组合框，命令按钮，编辑框，标签，列表框，选项按钮，图形，微调和文本框
2	ColorScheme 属性方案。控件使用 ColorScheme 属性指定的配色方案。本设置对以下控件和对象可用： 复选框，组合框，命令按钮，编辑框，标签，列表框，选项按钮，图形，微调和文本框
3	默认方案。控件使用默认配色方案。对大多数控件，默认配色方案是表单配色方案；对于编辑框和列表框，默认配色方案是方案 2，“用户菜单”。本设置对以下控件和对象可用： 复选框，组合框，命令按钮，编辑框，标签，列表框，选项按钮，图形，微调和文本框

续表

4	Windows控制面板（3D颜色）（默认值）。控件使用Windows控制面板中指定的3D颜色。本设置对以下控件和对象可用： 复选框，组合框，命令按钮，命令按钮组，容器，控件对象，编辑框，表单，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，图形，微调，文本框和工具栏
5	Windows控制面板（Windows颜色）。控件使用Windows控制面板中指定的颜色设置。本设置只对表单对象可用。

说明

设置一个对象的颜色属性（ForeColor、BackColor、SelectedForeColor等等），会覆盖ColorSource属性的设置。如果没有设置一个对象的颜色属性，则该对象的颜色由ColorSource属性的设置确定。

应用于

复选框，组合框，命令按钮，编辑框，标签，列表框，选项框按钮，形状，微调，文本框

请参阅

ColorScheme属性

Column 对象

在表格中创建列。

语法

Column

说明

表格中列的数目可用表格的 ColumnCount 属性指定。

表格中的列可包括表中的字段数据或表达式的值。可使用 DataSource 属性指定在列中显示的数据。

列也可包含控件。可以用表格所在表单的 Init 事件中的 AddObject 方法将控件添加到表格的列中。使用列的 CurrentControl 属性可以设置表格的列中的活动控件。使用该控件的 ControlSource 属性可以指定该控件的数据源。

注意：除非发生了表格的 Init 事件，否则不能使用列的标头和控件。

有关在表格中创建列的其他信息，请参阅稍后的“表格控件”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》中的第十章“使用控件”。

属性

Alignment	Application	BackColor
BaseClass	Bound	Class
ClassLibrary	ColumnOrder	Comment

续 表

ControlCount	Controls	ControlSource
CurrentControl	DynamicAlignment	DynamicBackColor
DynamicCurrentControl	DynamicFontBold	DynamicFontItalic
DynamicFontName	DynamicFontOutline	DynamicFontShadow
DynamicFontSize	DynamicFontStrikeThru	DynamicFontUnderline
DynamicForeColor	DynamicInputMask	Enabled
FontBold	FontCondense	FontExtend
FontItalic	FontName	FontOutline
FontShadow	FontSize	FontStrikeThru
FontUnderline	ForeColor	Format
InputMask	Movable	Name
Parent	ParentClass	ReadOnly
Resizable	SelectOnEntry	Sparse
Tag	Visible	Width

事 件

MouseMove	MouseWheel	Moved
Resize		

方法

AddObject	AddProperty	Move
ReadExpression	ReadMethod	Refresh
RemoveObject	ResetToDefault	SaveAsClass
SetAll	SetFocus	WriteExpression
WriteMethod	ZOrder	

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS, Grid **控件**, Header **对象**

ColumnCount **属性**

指定表格、组合框或列表框控件中列对象的数目。对表格而言，设计时可用，运行时只读或只写；对于组合框或列表框，设计和运行时可用。

语法

Object.ColumnCount[= nCol]

参数描述

nCol

对于表格控件，指定要显示的列数。默认的属性设置为-1，表明自动创建足够的列，以容纳数据源中的所有字段。最大列数是 255。**如果将** nCol**设置为**

一个正数，以指定特定的列数，可使用 ControlSource 属性设置在某列显示的数据。（如果没有为一列指定 ControlSource，则表格控件显示记录源中下一个可用的未显示字段。）

对于组合框或列表框控件，nCol 指定控件中所包含列的数目。若设置 ColumnCount 为 0，则根据 RowSource 属性或使用方法添加的项来显示第一列。

说明

对于表格，可使用 AddColumn 方法增加列的数目。

应用于

组合框，表格，列表框

请参阅

AddColumn 方法，Order 属性

ColumnLines 属性

显示或隐藏列之间的线条。设计和运行时可用。

语法

Control.ColumnLines[= lExpr]

参数描述

lExpr

指定是否显示分隔列的线条。下表列出了 *ColumnLines* **属性的设置：**

设置	**说明**
“真” (.T.)	**（默认值）显示列间线条。**
“假” (.F.)	**隐藏列间线条。**

应用于

组合框，列表框

请参阅

ColumnWidths **属性**

ColumnOrder **属性**

指定表格控件中列对象的相对顺序。设计和运行时可用。

语法

Column.ColumnOrder[= nExpr]

参数描述

nExpr

指定一列相对于表格中其他列的顺序。

说明

若表格包含了 5 列，并且要使第 3 列最后显示，则设置第 3 列的 *ColumnOrder* 属性为 5。第 4 列的 *ColumnOrder* 设置就变成了 3，第 5 列的 *ColumnOrder* 设置变成 4，等等。

注意 *ColumnOrder* 设置不必是连续的，可以有间隔。例如，若一个表格包含 3 列，可增加第 4 列并且设置第 4 列的 *ColumnOrder* 属性为 10。

应用于

列

请参阅

Activate 事件, Deactivate 事件, ZOrder 方法

Columns 属性

通过列号访问表格控件中单个列对象的数组。运行时只读或只写。

语法

Grid.Columns(nCol).Property [= Setting]

参数描述

nCol
指定表格中的列。最左边的列号为 1。
Property
指定要访问的列的属性。
Setting
为所有用 *nCol* 指定的单元指定新的属性设置。

说明

使用 Columns 属性可访问表格中指定列的属性。例如，SpecialGrid.Columns(1) .BackColor=RGB(255，0，0) 更改第一列中所有单元的 BackColor 属性为浅红色。

应用于

表格

请参阅

Column 对象，ColumnCount 属性，SetAll 方法

ColumnWidths 属性

指定组合框或列表框控件的列宽。设计或运行时可用。

语法

Control.ColumnWidths[= “cCol1Width, cCol2Width,...cColnWidth”]

参数描述

“cCol1Width, cCol2Width,...cColnWidth”

指定组合框或列表框中一列或一组列的宽度。例如，*cColumnWidths* = “5，7，9”指定第 1 列为 5 个单位宽，第 2 列为 7 个单位宽，第 3 列为 9 个单位宽，度量单位由表单的 ScaleMode 属性指定。使用逗号分隔每个宽度值。可用 ColumnCount 属性指定列数。

应用于

组合框，列表框

请参阅

ColumnCount 属性, ScaleMode 属性, Width 属性

ComboBox 控件

 创建一个组合框。

语法

ComboBox

说明

当被选中时，组合框打开，并显示项的列表，从中可选择一项。组合框控件结合了文本框控件和列表框控件的特性。可在文本框部分输入信息或从列表框部分选择一项。通过在组合框控件中的一项前面放一个反斜杠 (\) 可以让该项失效。如果该项以反斜杠开头，而且不应该失效，就再加一个反斜杠。

Style 属性确定组合框的类型。若 Style 属性设为 0，则创建一个下拉组合框；若 Style 属性设为 2，则创建一个下拉列表框。

有关复选框控件的其他信息，请参阅帮助中的“表单设计工具”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》中的第十章“使用控件”。

属性

Alignment	Application	BackColor
BaseClass	BorderColor	BorderStyle
BoundColumn	BoundTo	Class
ClassLibrary	ColorScheme	ColorSource
ColumnCount	ColumnLines	ColumnWidths
Comment	ControlSource	DisabledBackColor
DisabledForeColor	DisabledItemBackColor	DisabledItemForeColor
DisplayCount	DisplayValue	DragIcon
DragMode	Enabled	FirstElement
FontBold	FontCondense	FontExtend
FontItalic	FontName	FontOutline
FontShadow	FontSize	FontStrikeThru

续 表

FontUnderline	ForeColor	Format
Height	HelpContextID	HideSelection
IMEMode	IncrementalSearch	InputMask
ItemBackColor	ItemData	ItemForeColor
ItemIDData	ItemTips	Left
List	ListCount	ListIndex
ListItem	ListItemID	Margin
MouseIcon	MousePointer	Name
NewIndex	NewItemID	NullDisplay
NumberOfElements	OLEDragMode	OLEDragPicture
OLEDropEffects	OLEDropHasData	OLEDropMode
OLEDropTextInsert ion	Parent	ParentClass
Picture	ReadOnly	RightToLeft
RowSource	RowSourceType	Selected
SelectedBackColor	SelectedForeColor	SelectedID
SelectedItemBackC olor	SelectedItemForeC olor	SelectOnEntry
SelLength	SelStart	SelText
Sorted	SpecialEffect	StatusBarText
Style	TabIndex	TabStop
Tag	Text Property	TerminateRead
ToolTipText	Top	TopIndex

续 表

TopItemID	Value	Visible
WhatsThisHelpID	Width	

事 件

Click	DblClick	Destroy
DownClick	DragDrop	DragOver
DropDown	Error	ErrorMessage
GotFocus	Init	InteractiveChange
KeyPress	LostFocus	Message
MiddleClick	MouseDown	MouseMove
MouseUp	MouseWheel	OLECompleteDrag
OLEDragDrop	OLEDragOver	OLEGiveFeedBack
OLESetData	OLEStartDrag	ProgrammaticChange
RangeHigh	RangeLow	RightClick
UIEnable	UpClick	Valid
When		

方 法

AddItem	AddListItem	AddProperty
Clear	CloneObject	Drag
IndexToItemID	ItemIDToIndex	Move
OLEDrag	ReadExpression	ReadMethod
Refresh	RemoveItem	RemoveListItem
Requery	ResetToDefault	SaveAsClass
SetFocus	SetViewPort	ShowWhatsThis
WriteExpression	WriteMethod	ZOrder

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

COMARRAY（）函数

指定如何向 COM **对象传递数组。**

语法

COMARRAY(oObject [, nNewValue])

返值类型

数值型

参数描述

oObject

对 COM **对象的一个对象引用。**

nNewValue

指定如何向 oObjec **指定的** COM **对象传递数组。下表列出了** nNewValue **的设置，以及数组是如何传递给** COM **对象的。**

nNewValue	说明
0	数组是基于零的数组，并且按值传递。
1 (Default)	数组是基于一的数组，并且按值传递。 与早期版本的 Visual FoxPro 兼容。
10	数组是基于零的数组，并且按引用传递。
11	数组是基于一的数组，并且按引用传递。

使用 COMARRAY() 时不带 nNewValue 参数，可以返回当前设置。

说明

早期版本的 Visual FoxPro 只能向 COM 对象按值传递数组。另外，向 COM 对象传递的数组也是假设为基于一的，也就是说，数组中的第一个元素、行和列是用 1 引用的。

但是，有些 COM 对象需要将数组按引用传递给它们，或者数组是基于零的（数组中的第一个元素、行和列用 0 引用。），或者两者都需要。COMARRAY() 允许您指定如何向 COM 对象传递数组，并且假设您知道应该如何向 COM 对象传递数组。

注意，只有当使用下列语法向 COM 对象传递数组时才能使用 COMARRAY()：

```
oComObject.Method(@MyArray)
```

如果省略 @ 符号，则只向 COM 对象传递数组的第一个元素，COMARRAY() 函数不起作用。

请参阅

COMCLASSINFO( ), CREATEOBJECT( ), GETOBJECT( )

COMCLASSINFO（）函数

返回有关一个 COM 对象（例如一个 Visual FoxPro 自动服务程序）的注册信息。

语法

COMCLASSINFO(oObject [, nInfoType])

返值类型

字符型

参数描述

oObject

对一个 COM 或 OLE 对象的对象引用。

nInfoType

指定要返回的信息类型。下表列出了 nInfoType 的值，以及所返回的信息。

nInfoType	所返回的信息
1（默认值）	对象的程序标识符 (ProgID)。
A	一个 ProgID 是与 CLSID 相关的注册项。

续表

2	对象的 VersionIndependentProdID。VersionIndependentProgID 使得一个 ProgID 与 CLSID 相关。它用于确定一个对象应用程序的最新版本，引用该应用程序的类，而且不同版本的 VersionIndependentProgID 相同。
3	对象的友好名称
4	对象的类标识符 (CLSID)。CLSID 是全局唯一的标识符，它标识了一个 COM 类对象。

说明

如果指定对象的注册信息不可用，COMCLASSINFO( ) 返回空字符串。Visual FoxPro 自动服务程序是 COM 对象，是在项目管理器中创建的 .exe 执行文件或 .dll 动态链接库。

请参阅

COMARRAY( ), CREATE PROJECT, CREATEOBJECT( ), GETOBJECT( )

CommandButton 控件

创建单个命令按钮。

语法

CommandButton

说明

命令按钮通常用来启动一个事件，如关闭一个表单、移到不同记录、打印报表等动作。可使用 CommandGroup 控件创建一组命令按钮，并对其中的单个按钮或成组的按钮进行操作。

可使用 Caption 属性指定在命令按钮上显示的文本，使用 Picture 属性指定命令按钮的图片。

通过单击命令按钮就可以选择该命令按钮。若 Default 属性设置为“真”(.T.)，则可按 ENTER 键选择此命令按钮；若命令按钮的 Cancel 属性设置为“真”(.T.)，则可通过按 ESC 键选择此命令按钮。

有关命令按钮的其他信息，请参阅帮助中的“表单设计工具”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》中的第十章“使用控件”。

属性

Application
AutoSize
BaseClass
Cancel
Caption
ClassLibrary
Class
ColorScheme
ColorSource
Comment
Default
DisabledForeColor
DisabledPicture
DownPicture
DragIcon
DragMode
Enabled
FontBold
FontCondense
FontExtend
FontItalic
FontName
FontOutline
FontShadow

续 表

FontSize	FontStrikeThru	FontUnderline
ForeColor	Height	HelpContextID
Left	MouseIcon	MousePointer
Name	OLEDragMode	OLEDragPicture
OLEDropEffects	OLEDropHasData	OLEDropMode
Parent	ParentClass	Picture
RightToLeft	SpecialEffect	StatusBarText
Style	TabIndex	TabStop
Tag	TerminateRead	ToolTipText
Top	Visible	WhatsThisHelpID
Width	WordWrap	

事 件

Click	Destroy	DragDrop
DragOver	Error	ErrorMessage
GotFocus	Init	KeyPress
LostFocus	Message	MiddleClick
MouseDown	MouseMove	MouseUp
MouseWheel	OLECompleteDrag	OLEDragDrop
OLEDragOver	OLEGiveFeedBack	OLESetData
OLEStartDrag	RightClick	UIEnable
Valid	When	

方法

AddProperty	CloneObject	Drag
Move	OLEDrag	ReadExpression
ReadMethod	Refresh	ResetToDefault
SaveAsClass	SetFocus	ShowWhatsThis
WriteExpression	WriteMethod	ZOrder

示例

以下示例演示了命令按钮是如何添加到表单中的。使用 Caption 属性指定命令按钮上的文本和指明每个按钮的快捷访问键序列的文本。使用 Cancel 属性指定当按下 ESC 键时选取的按钮。

AddObject 方法用于向表单中添加三个命令按钮，允许您在“线条”控件中更改倾斜方向或关闭表单。

```
frmMyForm = CREATEOBJECT('Form') && 创建表单
frmMyForm.Closable = .F. && 禁止控件菜单框

frmMyForm.AddObject('shpLine','Line') && 向表单中添加线条控件
frmMyForm.AddObject('cmdCmndBtn1','cmdMyCmndBtn1') && 向上命令按钮
frmMyForm.AddObject('cmdCmndBtn2','cmdMyCmndBtn2') && 向下令按钮
frmMyForm.AddObject('cmdCmndBtn3','cmdMyCmndBtn3') && 退出命令按钮

frmMyForm.shpLine.Visible = .T. &&将线条控件设为可视的
frmMyForm.shpLine.Top = 20 && 指定线条控件行
frmMyForm.shpLine.Left = 125 && 指定线条控件列

frmMyForm.cmdCmndBtn1.Visible =.T. && “向上”命令按钮可视
```

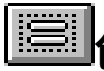
```
frmMyForm.cmdCmndBtn2.Visible =.T. && “向下”命令按钮可视
frmMyForm.cmdCmndBtn3.Visible =.T. && “退出”命令按钮可视

frmMyForm.SHOW &&显示表单
READ EVENTS &&开始事件处理

DEFINE CLASS cmdMyCmndBtn1 AS CommandButton &&创建命令按钮
    Caption = '向上倾斜(\<U)' &&命令按钮上的标题
    Left = 50 &&命令按钮列
    Top = 100 &&命令按钮行
    Height = 25 &&命令按钮高

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. &&隐藏线条控件
        ThisForm.shpLine.LineSlant ='/' &&向上倾斜
        ThisForm.shpLine.Visible = .T. &&显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn2 AS CommandButton &&创建命令按钮
    Caption = '向下倾斜(\<D)' &&命令按钮上的标题
    Left = 200 &&命令按钮列
    Top = 100 &&命令按钮行
    Height = 25 &&命令按钮高

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. &&隐藏线条控件
        ThisForm.shpLine.LineSlant ='\' &&向下倾斜
        ThisForm.shpLine.Visible = .T. &&显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn3 AS CommandButton &&创建命令按钮
    Caption = '退出(\<Q)' &&命令按钮上的标题
```

```
    Cancel = .T. && 默认“取消”命令按钮(Esc)
    Left = 125 && 命令按钮列
    Top = 150 && 命令按钮行
    Height = 25 && 命令按钮高
    PROCEDURE Click
       CLEAR EVENTS && 停止事件处理，关闭表单
ENDDEFINE
```

请参阅

CommandGroup 控件, CREATE CLASS 命令, CREATE FORM 命令, DEFINE CLASS 命令

CommandGroup 控件

 创建一组命令按钮。

语法

CommandGroup

说明

可使用 CommandGroup 控件创建命令按钮，并对其中的单个按钮或成组的按钮进行操作。使用 CommandButton 控件可创建单个命令按钮。

可使用 ButtonCount 属性指定组中的命令按钮数目，使用 Caption 属性指定命令按钮组的标签。

有关命令按钮组的其他信息，请参阅帮助中的“表单设计工具”和《Microsoft Visual FoxPro 6.0 中文版程序员指南》中的第十章“使用控件”。

属性

Application
AutoSize
BackColor
BackStyle
BaseClass
BorderColor
BorderStyle
ButtonCount
Buttons
Class
ClassLibrary
ColorSource
Comment
ControlSource
DragIcon
DragMode
Enabled
Height
HelpContextID
Left
MouseIcon
MousePointer
Name
OLEDragMode
OLEDragPicture
OLEDropEffects
OLEDropHasData
OLEDropMode
Parent
ParentClass
SpecialEffect
TabIndex
Tag
TerminateRead
Top
Value
Visible
WhatsThisHelpID
Width

事 件

Click
DblClick
DragDrop
DragOver
Error
ErrorMessage
Init
InteractiveChange
Message
MiddleClick
MouseDown
MouseMove
MouseUp
MouseWheel
OLECompleteDrag
OLEDragDrop
OLEDragOver
OLEGiveFeedBack
OLESetData
OLEStartDrag
ProgrammaticChange
RightClick
UIEnable
Valid
When

方 法

AddObject
AddProperty
CloneObject
Drag
Move
NewObject
OLEDrag
ReadExpression
ReadMethod
Refresh
RemoveObject
ResetToDefault
SaveAsClass
SetAll
ShowWhatsThis
WriteExpression
WriteMethod
ZOrder

请参阅

CommandButton **控件**, CREATE CLASS, CREATE FORM, DEFINE CLASS

CommandTargetExec **事件**

当 Active Document **的宿主应用程序通知** Active Document **要执行的命令时发生。**

语法

PROCEDURE Object.CommandTargetExec
[LPARAMETERS nCommandID, nExecOption, eArgIn, eArgOut]

参数描述

nCommandID

是 Active Document **的宿主应用程序向** CommandTargetExec **事件传递的一个参数，该参数指出了要执行的命令。下表列出了向** Visual FoxPro **可处理事件传递的值，以及要执行的相应命令。**

nCommandID	FOXPRO.H 常数	命令
1	CMDID_OPEN	“文件”菜单的“新建”命令。

续表

2	CMDID_NEW	“文件”菜单的“新建”命令。
3	CMDID_SAVE	“文件”菜单的“保存”命令。
4	CMDID_SAVEAS	“文件”菜单的“另存为”命令。
5	CMDID_SAVECOPYAS	“文件”菜单的“将复本保存为”命令。
6	CMDID_PRINT	“文件”菜单的“打印”命令。
7	CMDID_PRINTPREVIEW	“文件”菜单的“打印预览”命令。
8	CMDID_PAGESETUP	“文件”菜单的“页面设置”命令。
9	CMDID_SPELL	“工具”菜单的“拼写检查”命令。
10	CMDID_PROPERTIES	“文件”菜单的“属性”命令。
11	CMDID_CUT	“编辑”菜单的“剪切”命令。

续表

12	CMDID_COPY	“编辑”菜单的“复制”命令。
13	CMDID_PASTE	“编辑”菜单的“粘贴”命令。
14	CMDID_PASTESPECIAL	“编辑”菜单的“选择性粘贴”命令。
15	CMDID_UNDO	“编辑”菜单的“取消”命令。
16	CMDID_REDO	“编辑”菜单的“恢复”命令。
17	CMDID_SELECTALL	“编辑”菜单的“全选”命令。
18	CMDID_CLEARSELECTION	“编辑”菜单的“清除”命令。
19	CMDID_ZOOM	“视图”菜单的“显示比例”命令。
20	CMDID_GETZOOMRANGE	获取“视图”菜单的“显示比例”命令适用的缩放范围。

续表

21	CMDID_UPDATECOMMANDS	通知 Active Document 状态的改变。
22	CMDID_REFRESH	请求 Active Document 刷新它的显示。由 Active Document 完成。
23	CMDID_STOP	请求 Active Document 停止它的处理。
24	CMDID_HIDETOOLBARS	请求 Active Document 隐藏工具栏。由 Active Document 完成。
25	CMDID_SETPROGRESSMAX	设置进程指示器的最大值。
26	CMDID_SETPROGRESSPOS	设置进程指示器的当前值。
27	CMDID_PROGRESSTEXT	设置进程指示器中包含的文本。
28	CMDID_SETTITLE	设置标题栏文本。

续表

29	CMDID_DOWNLOADSTATE	当宿主程序下载状态更改时发送。
30	CMDID_STOPDOWNLOAD	当执行时停止下载。
31	CMDID_ONTOOLBARACTIVATED	容器的一个工具栏获得了焦点。
36	CMDID_ENABLE_INTERACTION	由 Active Document 宿主应用程序发送，告诉 Active Document 或者暂停或者重新开始 Active Document 中的任何多媒体（音频或动画）。 如果在 Active Document 一个控件中运行的多媒体文件应该重新开始，则 eArgIn 返回“真”(.T.)，如果该多媒体文件应该暂

续表

		停，则 eArgIn 返回“假”(.T.)。
37	CMDID_ONUNLOAD	Internet Explorer 4.0 使用这个命令，当一个 Active Document 最小化或完全被另一个窗口覆盖时通知该 Active Document，这样该 Active Document 可以暂停播放多媒体信息。当定位到另一个站点之前，或 Active Document 宿主关闭之前由该宿主程序发送。将 *eArgOut* 设置为“假”(.T.)可以防止一个 Active Document 被它的宿主程序关闭。将 *eArgOut* 设置为

续表

“真” (.T.)，可以允许一个 Active Document 由它的宿主程序关闭。

nExecOption

传递给 CommandTargetExec 事件的应该参数，表明指定命令所需的默认动作。下表列出了 *nExecOption* 的值，以及要执行的动作。

nExecOption	FOXPRO.H Constant	动作
0	CMDEXECOPT_DODEFAULT	使用默认的行为，提示或不提示用户输入。
1	CMDEXECOPT_PROMPTUSER	当获得用户输入之后执行命令。
2	CMDEXECOPT_DONTPROMPTUSER	执行命令而不提示用户。例如，单击“打印”工具栏按钮，会立即打印一篇文档，而不需用户的输入。

续表

3	CMDEXECOPT_SHOWHELP	显示相应命令的帮助信息，而实际不执行该命令。

eArgIn

从 Active Document 宿主程序传递到 CommandTargetExec 事件的参数。通常，这个参数是 null 值，当 nCommandID 是 36 时例外。当 nCommandID 是 36 时，eArgIn 包含一个值。有关传递给 eArgIn 的逻辑值的详细内容，请参阅 nCommandID 参数说明中的表格。

eArgOut

从 CommandTargetExec 事件返回给 Active Document 宿主程序的一个输出参数。通常，这个参数是 null 值，当 nCommandID 是 37 时例外。当 nCommandID 是 37 时（表明 Active Document 宿主程序将要关闭 Active Document 或定位到另一个站点），可以将 eArgOut 设置为“假”(.T.)，以防止 Active Document 被宿主程序关闭。将 eArgOut 设置为“真”(.T.)，可以允许 Active Document 被宿主程序关闭。

说明

CommandTargetExec 事件允许 Active Document 根据宿主程序发送的命令执行自定义动作。这些命令可能是由于响应用户选择一个菜单项或工具栏项而发送的，或者是由于触发了 Active Document 宿主程序中的一个事件而发送的。从 CommandTargetExec 事件的返回值会通知 Active Document 宿主程序您是否处理过该命令。如果您没有处理过该命令，Active Document 宿主程序会执行该命令默认的过程。

下表列出了从 CommandTargetExec 事件返回的值。

返回值	FOXPRO.H 常数	命令动作
0	CMD_OK	Active Document 处理了该命令。 当 Active Document 曾经处理了由 nCommandID 指定的命令时返回该值。例如，如果选择了宿主程序中“文件”菜单中的“打开”命令，则您的 Active Document 可以在 CommandTargetExec 事件中确定 nCommandID 为 1，然后执行自己的打开文件例程，可能是使用 Visual FoxPro GETFILE（）函数。
2	CMD_NOTSUPPORTED	Active Document 不支持该命令。 Active Document 不支持该命令。当 Active Document 不能识别 nCommandID 指定的命令时，返回这个值。
3	CMD_DISABLED	Active Document 废止了该命令。 当 nCommandID 指定的命令正被废止时，或 Active Document 不能执行该命令时，返回这个值。

续表

4	CMD_NOHELP	Active Document 没有该命令的帮助信息。 当 Active Document 没有 nCommandID 指定的命令的帮助信息时，返回这个值。
5	CMD_CANCELED	用户取消了该命令的执行。 当 Active Document 试图处理 nCommandID 指定的命令而用户取消了该操作时，返回这个值。

Active Document 不支持该命令。当 Active Document 不能识别 nCommandID 指定的命令时，返回这个值。

对于 CommandTargetExec 和 CommandTargetQuery 事件，Visual FoxPro 利用 IoleCommandTarget 接口。有关这些事件是如何实施的详细信息，可以在 MSDN library 中搜索 IoleCommandTarget。

应用于

ActiveDoc 对象

请参阅

CommandTargetQuery 事件

CommandTargetQuery 事件

当 Active Document 宿主程序更新自己的用户界面时发生该事件。

语法

PROCEDURE Object.CommandTargetQuery
[LPARAMETERS *aCommands*, *nCommandTextFlag*, *cCommandTextOut*]

参数描述

aCommands

一个二维数组，其中包含一组 Active Document 宿主程序支持的命令。数组的第一列包含一些对应于 Active Document 宿主程序支持的命令的数值。有关命令列表和数值的详细内容，请参阅 CommandTargetExec 事件中的 nCommandID 参数。

数组的第二列包含一些对应于每个命令的支持状态的数值。下表列出了第二列包含的数值，以及相应的每个命令的支持状态。

数值	FOXPRO.H 常数	说明
0	CMDF_NOTSUPPORTED	该对象不支持该命令。
1	CMDF_SUPPORTED	该对象支持该命令。
2	CMDF_ENABLED	该命令可用而且已经启用。

续表

4	CMDF_LATCHED	该命令可以开关切换，而且目前是打开的。
8	CMDF_NINCHED	该命令可以开关切换，但是由于在相关选择中，该命令的属性处于既开又关状态，所以该命令的状态无法确定。例如，这种状态对应于三种状态复选框的“不明状态”。

数组的第二列对于每个命令，初始都包含零。事件代码应该在第二列包含一个数值，以表明 Active Document 对每个命令的支持状态。可以任意组合这些值，以指定附加的支持级别。例如，一个命令支持 (1) 您的 Active Document 过程，而且可用并启用了 (2)，则可以在相应命令的第二列保存 3 (1 + 2)。

nCommandTextFlag

从 Active Document 宿主程序传递到 CommandTargetExec 事件的一个参数。*nCommandTextFlag* 表明了通过 *cCommandTextOut* 参数传递给 Active Document 宿主程序的命令信息的类型。下表列出了 *nCommandTextFlag* 的值，以及相应的传递给 Active Document 宿主程序的命令信息。

nCommandTextFlag	命令信息
0	不需要额外信息。
1	Active Document 应该提供命令的本地名称。

续表

2	Active Document 应该提供命令的本地状态栏字符串。

cCommandTextOut

从 CommandTargetExec 事件传递给 Active Document 宿主程序的一个参数。*cCommandTextOut* 是为一个命令显示的文本，通常显示在 Active Document 宿主程序的状态栏。*cCommandTextOut* 适用于数组中第一列的命令。

说明

CommandTargetQuery 事件允许通知一个 Active Document 的宿主程序，该 Active Document 支持哪些命令，而且每个命令可用的支持类型。也可以指定当选中一个命令（通常是一个菜单项）时显示的文本。

对于 CommandTargetExec 和 CommandTargetQuery 事件，Visual FoxPro 利用 IOleCommandTarget 接口。有关这些事件是如何实施的详细信息，可以在 MSDN library 中搜索 IOleCommandTarget。

注意，Internet Explorer 3.0 和 4.0 通常不需要 *cCommandTextOut* 的命令文本。因此，*nCommandTextFlag* 通常为零，而且 *cCommandTextOut* 通常包含 null 值。包含这些参数，以便将来与 IOleCommandTarget 接口兼容，而且在您的应用程序中可以安全地忽略它们。

应用于

ActiveDoc 对象

请参阅

CommandTargetExec 事件

Comment 属性

存储对象的信息。设计和运行时可用。

语法

Object.Comment[= cTextString]

参数描述

cTextString

指定一个文本字符串。

说明

与其他属性不同，Visual FoxPro 并不使用 Comment 属性的值。因此可使用这个属性标识或说明对象。

可使用这个属性指定对象的一个标识字符串而不影响对象的其他属性设置。将表单或控件作为变量传递给某一过程，在想要了解该表单或控件的情况时，Comment 属性很有用。

Comment 属性的默认值设置为零长度字符串（“”）。

当创建表单的一个新的实例时，应给 Comment 属性指定一个新的值。

应用于

ActiveDoc，复选框，组合框，命令按钮，命令组，容器对象，控件对象，定制，编辑，表单，表单集，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器

控件，选项按钮，选项组，页面，页框，ProjectHook 对象，_SCREEN，形状，微调，文本框，计时器，工具栏

请参阅

Caption 属性，Name 属性

COMPILE 命令

编译一个或多个文件，并为每一个源文件创建一个目标文件。

语法

```
COMPILE [CLASSLIB | LABEL | REPORT] FileName | FileSkeleton | ?
  [ENCRYPT] [NODEBUG]
  [AS nCodePage]
```

参数描述

CLASSLIB

指定要编译的文件是一个可视化类库 (.vcx)。可视化类库的源代码是存储在可视化类库表中的备注字段中。COMPILE CLASSLIB 将这些备注字段编译到存储在其他备注字段的对象代码中。

LABEL

指定要编译的文件是一个标签定义文件 (.lbx)。在一个附加的备注字段中编

译并存储与标签定义文件一起保存的数据环境源代码。

REPORT

指定要编译的文件是一个报表定义文件 (.frx)。在一个附加的备注字段中编译并存储与报表定义文件一起保存的数据环境源代码。

FileName | FileSkeleton

指定一个要编译的文件或编译一组与文件梗概相匹配的文件。该文件梗概可包含通配符如 * 和 ?。例如，要编译当前目录里所有带 .PRG 扩展名的程序文件，可执行 COMPILE *.PRG。

?

显示“编译”对话框，允许选择要编译的文件。

ENCRYPT

加密要编译的 Visual FoxPro 程序。不能使用 CLASSLIB、LABEL 和 REPORT 关键字。这可以防止任何对原始源程序的访问。为保护源代码，在编译要发布的程序时经常需要包含这个选项。

NODEBUG

使编译文件每行缩小 2 字节。编译文件中的这两个字节是对源文件中对应行的引用。删除这些字节不影响程序的性能，但是可缩小编译文件大小并节省磁盘空间。

当包含 NODEBUG 选项时，不能在跟踪窗口中显示程序的执行情况，也不能使用 MESSAGE(1) 返回产生错误的程序行的源代码。

AS nCodePage

指定编译程序的代码页。不能使用 CLASSLIB、LABEL 和 REPORT 关键字。该子句指定的编译代码页将改写由 SET CPCOMPILE 指定的全局编译代

码页。

说明

Visual FoxPro只执行目标文件。因此若源文件尚未编译，则在运行程序时会自动编译源文件，而源文件保持不变。创建的编译文件与源文件的基本名相同，但扩展名不同。下表列出了每种文件类型的源文件和编译文件扩展名。

文件类型	源文件扩展名	编译文件扩展名
程序文件	.prg	.fxp
表单代码	.spr	.spx
菜单代码	.mpr	.mpx
查询	.qpr	.qpx
格式	.fmt	.prx

编译器检查源文件中的任何语法错误。若SET LOGERRORS是ON，则编译文件时，编译错误信息被保存到一个文本文件中。错误日志文件与编译文件的基本名相同，但扩展名为.ERR；若SET LOGERRORS是OFF，则不创建错误日志文件。

请参阅

BUILD APP, BUILD PROJECT, #DEFINE...#UNDEF, #IF...#ENDIF, #IFDEF | #IFNDEF...#ENDIF, #INCLUDE, MODIFY COMMAND, MODIFY PROJECT, SET CPCOMPILE, SET LOGERRORS

COMPILE DATABASE 命令

编译数据库中的内部存储过程。

语法

COMPILE DATABASE DatabaseName

参数描述

DatabaseName

指定数据库名，该数据库中包含要编译的内部存储过程。可编译未打开数据库中的内部存储过程。

说明

使用 COMPILE DATABASE 在数据库设计器之外编译内部存储过程。内部存储过程可以由 MODIFY PROCEDURES 命令交互地创建和修改，或用 APPEND PROCEDURES 命令以编程方式创建和修改。

COMPILE DATABASE 为数据库删除 .dct 备注文件中的备注字段，以便从备注文件中删除未使用的空间。在数据库表中标记为删除的记录不从表中删除。

示例

下面的示例编译 testdata 数据库中所有的内部存储过程。

```
CLOSE DATABASES
COMPILE DATABASE (HOME(2) + 'data\testdata')
```

请参阅

APPEND PROCEDURES, COPY PROCEDURES, CREATE TRIGGER, DISPLAY PROCEDURES , MODIFY PROCEDURE, OPEN DATABASE

返回总目录

COPY STRUCTURE 命令
COPY STRUCTURE EXTENDED 命令
COPY TAG 命令
COPY TO 命令
COPY TO ARRAY 命令
COS（） 函数
COUNT 命令
Count 属性
_COVERAGE 系统变量
CPCONVERT（） 函数
CPCURRENT（） 函数
CPDBF（） 函数
CREATE 命令
CREATE CLASS 命令
CREATE CLASSLIB 命令
CREATE COLOR SET 命令
CREATE CONNECTION 命令
CREATE CURSOR – SQL 命令
CREATE DATABASE 命令
CREATE FORM 命令
CREAT FROM 命令
CREATE LABEL 命令

CREATE MENU 命令
CREATE PROJECT 命令
CREATE QUERY 命令
CREATE REPORT 命令
CREATE REPORT – Quick Report 命令
CREATE SCREEN –快速屏幕命令
CREATE SCREEN 命令

COMPILE FORM 命令

编译一个或多个表单对象。

语法

COMPILE FORM FormName | cFileSkeleton [ALL]

参数描述

FormName

指定要编译的表单的名称。

cFileSkeleton

指定一组要编译的表单文件。*cFileSkeleton* 是支持通配符的文件名梗概。例如，要编译所有以 A 开头的表单对象，可使用下列命令：

COMPILE FORM A*

ALL

编译供所有 Visual FoxPro 平台使用的表单中的所有记录。如果省略 ALL，则只编译供 Visual FoxPro 当前平台使用的记录。

说明

在表单设计器中保存表单时，自动编译表单。使用 COMPILE FORM 命令可以在表单设计器之外编译表单。

表单源代码存储于表单表的备注字段中。COMPILE FORM 将这些备注字段编译为目标代码，存储在附加的备注字段里。当发出 DO FORM 命令时，执行这个附加字段中的目标代码。

需要注意的是如果表单包含了一个包含文件 (.h) 而这个包含文件从它的原始位置移走了，就会发生一个编译错误并将错误列在 .err 错误日志文件中。表单可以运行，但是如果修改了表单，那么，直到您更正了包含文件的路径才能保存它。如果要更正包含文件的路径，使用 MODIFY FORM 命令打开表单，从“表单”菜单中选择“包含文件”，在“包含文件”对话框中指定包含文件新的路径。

请参阅

CREATE FORM, DO FORM, MODIFY FORM

COMPOBJ( ) 函数

比较两个对象的属性。若两者的属性和属性值相同，则返回“真” (.T.)。

语法

COMPOBJ(oExpression1, oExpression2)

参数描述

oExpression1, oExpression2

指定要比较的对象。*oExpression1* 和 *oExpression2* 可以是任何求值结果为对象

的表达式，如对象引用、对象变量或对象数组元素。

返值类型

逻辑值

说明

若一个对象具有另一个对象所没有的属性，或两个对象具有相同的属性但有一个或多个属性值不同，则 COMPOBJ() 返回“假”(.F.)。

示例

在下面的示例中，创建了两个名为 lstMyList1 和 lstMyList2 的列表框和一个名为 cmbMyCombo 的组合框，并显示每个列表框的 Name 属性。

COMPOBJ() 用来比较第一个列表框和组合框的属性。因为许多属性不同，所以返回 .F.。然后用 COMPOBJ() 比较第一个列表框和第二个列表框的属性。因为 Names 属性不同，所以返回 .F.。用第一个列表框 lstMyList1 代替第二个列表框 lstMyList2，并用 COMPOBJ() 来比较属性。因为这两个列表框的属性相同，所以返回 .T.。

```
lstMyList1 = CREATEOBJ('ListBox') &&创建一个“列表框”
lstMyList2 = CREATEOBJ('ListBox') &&创建第二个“列表框”
cmbMyCombo = CREATEOBJ('ComboBox') &&创建一个“组合框”

CLEAR
? lstMyList1.Name  &&显示 List1 Name 属性
? lstMyList2.Name  &&显示 List2 Name 属性

? COMPOBJ(lstMyList1, cmbMyCombo)    &&显示 .F.
? COMPOBJ(lstMyList1, lstMyList2)    &&显示 .F., 不同的“名称”
lstMyList2.Name = lstMyList1.Name
? COMPOBJ(lstMyList1, lstMyList2)    &&显示 .T., 相同的属性
```

请参阅

CREATEOBJECT（）, DEFINE CLASS, GETOBJECT（）, SET CLASSLIB

COMRETURNERROR（） 函数

使用自动服务客户程序能用来确定自动服务错误原因的信息填充 COM 异常结构。

语法

COMRETURNERROR(cExceptionSource, cExceptionText)

参数描述

cExceptionSource
指定异常原因的文本。

cExceptionText
指定异常说明的文本。

说明

COMRETURNERROR（） **允许** Visual FoxPro Automation **服务程序填充 COM 异常结构，这样 Automation 客户程序就可以确定在 Automation 服务程序中发生一个错误的原因。**

使用 COMRETURNERROR（） **函数可以将指定文本放在 COM 异常结构中，并且将控**

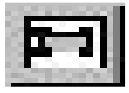
制返回给客户程序。在使用了 COMRETURNERROR（） 的位置中断 Automation 服务程序的执行。Automation 服务程序保留在内存中，在客户程序调用下一个方法时继续执行 Automation 服务程序。

Visual FoxPro 客户程序可以使用 AERROR（） 查看 COM 异常结构中的文本。

请参阅

AERROR（），ON ERROR

Container 对象



创建可包含其他对象的对象。

语法

Container

说明

容器对象可包含其他对象，并且允许访问被包含对象。例如，若创建了一个由两个列表框和两个命令按钮组成的容器对象，然后将容器对象添加到一个表单中，则可在设计和

运行时操作列表框和命令按钮。

有关容器对象和容器对象与其他对象和控件的区别的详细内容，请参阅《Visual FoxPro 6.0 中文版程序员指南》的第三章“面向对象的程序设计”。

属性

ActiveControl
Application
BackColor
BackStyle
BaseClass
BorderColor
BorderWidth
Class
ClassLibrary
ColorSource
Comment
ControlCount
Controls
DragIcon
DragMode
Enabled
ForeColor
Height
HelpContextID
Left
MouseIcon
MousePointer
Name
Objects
OLEDragMode
OLEDragPicture
OLEDropEffects
OLEDropHasData
OLEDropMode
Parent
ParentClass
Picture
SpecialEffect
TabIndex
TabStop
Tag
Top
Visible
WhatsThisHelpID
Width

事件

Click	DblClick	Destroy
DragDrop	DragOver	Error
GotFocus	Init	LostFocus
MiddleClick	MouseDown	MouseMove
MouseUp	MouseWheel	Moved
OLECompleteDrag	OLEDragDrop	OLEDragOver
OLEGiveFeedBack	OLESetData	OLEStartDrag
Resize	RightClick	UIEnable

方法

AddObject	AddProperty	Drag
Draw	Move	NewObject
OLEDrag	ReadExpression	ReadMethod
Refresh	RemoveObject	ResetToDefault
SaveAsClass	SetAll	SetFocus
ShowWhatsThis	WriteExpression	Zorder

请参阅

CREATE CLASS 命令, CREATE FORM 命令, DEFINE CLASS 命令

ContainerRelease 事件

当一个 Active Document 被自己的宿主程序释放时发生。

语法

PROCEDURE Object.ContainerRelease

说明

当关闭容器，Active Document 从容器的缓冲中抛出，或者当从 Active Document 中漫游时，该 Active Document 可以被容器释放。例如，当您从一个 Active Document 中漫游时，Microsoft Internet Explorer 4.0 for Windows 会释放该 Active Document。在 Microsoft Internet Explorer 3.0 for Windows 中，当一个 Active Document 从四页的缓冲中抛出时，会释放该 Active Document。

ContainerReleaseType 属性可以在此事件中设置，在该事件发生后 Visual FoxPro 检查此值以确定是否在 Visual FoxPro 运行时刻打开 Active Document。

应用于

ActiveDoc 对象

请参阅

ContainerReleaseType 属性

ContainerReleaseType 属性

指定一个 Active Document 被宿主程序释放时，该 Active Document 是否保持打开和运行。设计和运行时可用。

语法

Object.ContainerReleaseType[= nExpression]

参数描述

nExpression

nExpression 的设置有：

设置	说明
0	（默认值）当一个 Active Document 被宿主程序释放时，该 Active Document 在运行时刻保持打开。该 Active Document 在运行时刻 Visual FoxPro 的主窗口中继续运行。
1	关闭该 Active Document，并且停止运行。

说明

当一个 Active Document 被宿主程序释放时，发生 ContainerRelease 事件。在该事件中也可以查询 ContainerReleaseType 属性的值；如果该值为 1，您可以进行关闭过程，例如关闭打开的文件，清理 Visual FoxPro 环境。注意，当 ContainerReleaseType 设置为 1，并且关闭了 Active Document，在该 Active Document 的宿主程序也关闭之前，运行

时刻 Visual FoxPro 的一个实例 Vfp6run.exe 仍保持运行。
当关闭容器，当 Active Document 从容器的缓冲中抛出，或者当从 Active Document 中漫游时，该 Active Document 可以被容器释放。例如，当您从一个 Active Document 中漫游时，Microsoft Internet Explorer 4.0 for Windows 会释放该 Active Document。在 Microsoft Internet Explorer 3.0 for Windows 中，当一个 Active Document 从四页的缓冲中抛出时，会释放该 Active Document。
在 ContainerRelease 事件中可以设置该属性，以指定当从容器中释放 Active Document 时，该 Active Document 是否在运行时刻 Visual FoxPro 中打开。
应用于

ActiveDoc 对象
请参阅

ContainerRelease 事件

CONTINUE 命令

继续执行先前的 LOCATE 命令。
语法

CONTINUE

说明

CONTINUE 在用 LOCATE 查找到记录后使用，继续执行 LOCATE 操作。CONTINUE 移动记录指针到下一条逻辑表达式为“真”(.T.) 的记录，该逻辑表达式是在上一个 LOCATE 中指定的。

在到达文件尾或用 LOCATE 指定的作用范围结束之前，可重复使用 CONTINUE。若 CONTINUE 查找到一个记录，则 RECNO（） 返回该记录的记录号，FOUND（） 返回“真”(.T.)，EOF（） 返回“假”(.F.)。

如果 CONTINUE 找到记录，RECNO（） 返回“真”(.T.)，而 EOF（）返回“假”(.F.)。

如果 CONTINUE 没有找到记录，则 RECNO（） 返回表的记录个数加一，FOUND（）则返回“假”(.F.)，而 EOF（） 则返回“真”(.T.)。

示例

在以下示例中，计算所有从法国 (France) 来的顾客并显示总数。使用 LOCATE 命令和其后循环语句中的 CONTINUE 命令找到所有的记录。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表
SET TALK OFF
STORE 0 TO gnCount

LOCATE FOR ALLTRIM(UPPER(country)) = 'FRANCE'
DO WHILE FOUND（）
    gnCount = gnCount + 1
    CONTINUE
ENDDO
```

```
? 'Total customers from France: '+ LTRIM(STR(gnCount))
```

请参阅

EOF（）,FOUND（）,LOCATE, SEEK

ContinuousScroll 属性

指定在一个表单中的滚动是连续的，还是只当释放一个滚动框时才滚动。设计和运行时可用。

语法

Object.ContinuousScroll[= lExpression]

参数描述

lExpression

lExpression **的设置为：**

设置	**说明**
True (.T.)	**（默认值）在表单中的滚动是连续的。当移动滚动框时，在表单中的滚动是连续的。在表单滚动过程中，连续发生** Scrolled **事件。**

续表

False (.F.)	只有当释放一个滚动框时才滚动。在释放滚动框之前，表单保持不动，在释放之后，表单在新位置重画。当表单重画之后发生 Scrolled 事件。

说明

Scrollbars 属性决定一个表单是否有滚动条。

应用于

表单

请参阅

ScrollBars 属性, Scrolled 事件

Control 对象

创建能包含其他被保护对象的控件对象。

语法

Control

说明

Control 控件对象能包含其他对象，但是不能像容器对象那样允许访问被包含的对象。例如，若创建一个由两个列表框和两个命令按钮组成的控件对象，然后将控件对象添加到表单中，则在设计和运行时不能操作列表框和命令按钮。

有关控件对象及其与其他对象和控件的区别的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第三章“面向对象程序设计”。

属性

ActiveControl
Application
BackColor
BackStyle
BaseClass
BorderColor
BorderWidth
Class
ClassLibrary
ColorSource
Comment
ControlCount
Controls
DragIcon
DragMode
Enabled
ForeColor
Height
HelpContextID
Left
MouseIcon
MousePointer
Name
Objects
OLEDragMode
OLEDragPicture
OLEDropEffects
OLEDropHasData
OLEDropMode
Parent
ParentClass
Picture
SpecialEffect
TabIndex
TabStop
Tag
Top
Visible
WhatsThisHelpID
Width

事件

Click
DblClick
Destroy
DragDrop
DragOver
Error
GotFocus
Init
LostFocus
MiddleClick
MouseDown
MouseMove
MouseUp
MouseWheel
Moved
OLECompleteDrag
OLEDragDrop
OLEDragOver
OLEGiveFeedBack
OLESetData
OLEStartDrag
Resize
RightClick
UIEnable

方法

AddProperty
Drag
Draw
Move
OLEDrag
ReadExpression
ReadMethod
Refresh
ResetToDefault
SaveAsClass
SetFocus
ShowWhatsThis
WriteExpression
Zorder

请参阅

CREATE CLASS 命令, CREATE FORM 命令, DEFINE CLASS 命令

ControlBox 属性

指定运行时在表单或工具栏的左上角是否显示控件菜单框。设计和运行时可用。

语法

Object.ControlBox[= lExpr]

参数描述

lExpr

下表列出了 ControlBox 属性设置：

设置	说明
“真” (.T.)	（默认值）显示弹出菜单图标。
“假” (.F.)	不显示弹出菜单图标。

说明

设计和运行时可用。模式、无模式表单和工具栏都能包含弹出菜单图标。在运行时菜单命令是否可用取决于相关属性的设置。例如，设置 MaxButton 和 MinButton 为“假” (.F.)，可使弹出菜单上的“最大化”和“最小化”菜单项无效，但“移动”和“关闭”命令仍然可用。

注意 为 ControlBox、BorderStyle、MaxButton 和 MinButton 指定的设置只有在运行时才在表单上反映出来。

应用于

表单，_SCREEN，工具栏

请参阅

BorderStyle 属性，Closable 属性，MaxButton 属性，MinButton 属性

ControlCount 属性

指定容器对象中控件的数目。设计时不可用，运行时只读。

语法

Object.ControlCount

说明

可用此属性循环遍历容器内的所有控件，并在每个控件上执行一个操作。

应用于

列，容器对象，控件对象，表单，页面，_SCREEN，工具栏

请参阅

Controls 属性

控件和对象

本主题说明了对在以前版本的 FoxPro 中创建的控件和对象的支持，以及添加到 Visual FoxPro 中的新控件。

创建控件和对象的最简单的方法是使用“表单设计器”。有关使用“表单设计器”创建控件的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版开发指南》中的第九章“创建表单”。

也可以通过编程创建控件和对象。有关根据 Visual FoxPro 基类通过编程创建控件的详细内容，请参阅本语言参考后面的 DEFINE CLASS 命令，以及《Microsoft Visual FoxPro 6.0 中文版开发指南》中的第三章“面向对象程序设计”。

使用 Visual FoxPro for Windows 或 Visual FoxPro for Macintosh 时，有些控件的行为可能有些不同；在默认情况下，在 Visual FoxPro for Macintosh 中的控件遵循 Macintosh 应用程序的一般界面规范。例如，在默认情况下，在 Visual FoxPro for Macintosh 中的复选框、组合框和选项按钮不能接受焦点。有关控件行为的不同，请参阅本语言参考后面的 SET KEYCOMP 命令。

在 Visual FoxPro 中，对于在以前版本的 FoxPro 中创建的控件，都添加了一个 NAME 子句。对于使用 @ ... GET 和 @ ... EDIT 命令创建的控件，NAME 子句可以创建对这些控件的对象引用，这允许您使用 Visual FoxPro 的属性、事件和方法来操作该控件。NAME 子句提供了一步到位的方法，使您可以利用 Visual FoxPro 的面向对象编程技术升级您的应用程序。

与 FoxPro 2.x 控件的兼容性

下表在早期版本的 FoxPro 中可用的控件，以及可以在 Visual FoxPro 中用来通过编程创建控件的基类。

FoxPro 2.x 控件	相应的 Visual FoxPro 控件	Visual FoxPro 基类名
@ ... GET – Check Boxes	CheckBox 控件	CheckBox
@ ... GET – Lists	ListBox 控件	ListBox
@ ... GET – Popups	ComboBox 控件	ComboBox
@ ... GET – Push Buttons	CommandButton 控件	CommandButton
@ ... GET – Radio Buttons	OptionButton 控件	OptionButton
@ ... GET – Spinners	Spinner 控件	Spinner
@ ... GET – Text Boxes	TextBox 控件	TextBox
@ ... EDIT – Text Edit Regions	EditBox 控件	EditBox

Visual FoxPro 基类

另外，下列控件和对象是 Visual FoxPro 新加的，并且只能通过编程从 Visual FoxPro 基类创建。

控件和对象	基类名	说明
ActiveDoc 对象	ActiveDoc	创建一个可以包含在 Active Document 容器（例如 Microsoft Office Binder 和 Internet Explorer）中的 Active Document。
Column 对象	Column	创建表格中一个列。
CommandGroup 控件	CommandGroup	创建一组命令按钮。
Container 对象	Container	创建一个可以包含其他对象的对象。
Control 对象	Control	创建一个控件对象，它可以包含其他被保护的对象。

续表

Cursor 对象	Cursor	当一个表或视图添加到表单、表单集或报表的数据环境中时创建。
Custom 对象	Custom	创建一个自定义对象。
DataEnvironment 对象	DataEnvironment	当打开表单、表单集或报表时创建。
Form 对象	Form	创建一个表单。
FormSet 对象	FormSet	创建一个表单集.
Grid 控件	Grid	创建一个 Grid 控件。
Header 对象	Header	为表格中的一列创建一个标头。
Hyperlink 对象	HyperLink	创建一个超级链接对象，允许跳转到一个 URL (Uniform Resource Locator)。
Image 控件	Image	创建一个 Image 控件，它可以显示一个 .bmp 或 PICT 图。

续表

Label 控件	Label	创建一个显示文本的 Label 控件。
Line 控件	Line	创建一个显示水平线、垂直现或对角线的 Line 控件 OLE。
OLE Container 控件	OLEControl	创建一个 OLE 容器控件。
OLE Bound 控件	OLEBound Control	创建一个 OLE 绑定型控件。
OptionGroup 控件	OptionGroup	创建一组选项按钮。
Page 对象	Page	创建页框中的一个页。
PageFrame 控件	PageFrame	创建一个包含页的页框。
ProjectHook 对象	ProjectHook	每当打开一个项目时初始化，提供对项目事件的编程访问。
Relation 对象	Relation	当在表单、表单集或报表的“数据环境设计器”中建立关系时创建。

续表

Separator 对象	Separator	创建一个 Separator 对象，它可以在工具栏的控件之间设置空间。
Shape 控件	Shape	创建一个显示方框、圆或椭圆的 Shape 控件。
Timer 控件	Timer	创建一个可以按规则间隔执行代码的 Timer 控件。
ToolBar 对象	Toolbar	创建一个可以放置控件的工具栏。

请参阅

CREATE CLASS 命令, CREATE FORM 命令, DEFINE CLASS 命令

Controls 属性

访问容器对象中控件的数组。运行时可用。

语法

ContainerObject.Controls(Index).Property[= Expr]

参数描述

Expr
为包含在 *ContainerObject* 中的控件指定属性值。

说明

可使用 Controls 属性访问所包含对象的属性。

应用于

列，容器对象，控件对象，表单，页面，_SCREEN，工具栏

请参阅

ControlCount 属性

ControlSource 属性

指定与对象绑定的数据源。设计和运行时可用。

语法

Object.ControlSource[= cName]

参数描述

cName
对于文本框控件，*cName* 是一个变量或字段。

说明

若 ControlSource 属性设置为字段或变量，则 Value 属性将与 ControlSource 属性所设置的变量或字段具有相同的数值和数据类型。

对于文本框控件，*cName* 一般是字段。

在表格（Grid）控件中，如果没有为列指定 ControlSource 的设置，列将显示下一个可用的表格的记录来源中的未显示字段。

若列的 Bound 属性设置为“真”(.T.)，则列的 ControlSource 属性设置适用于列和它所包含的任何控件。若再试图设置控件的 ControlSource 属性，则产生一个错误。若列的 Bound 属性设置为“假”(.F.)，则可以直接设置控件的 ControlSource 属性。若又设置了列的 ControlSource 属性，则此设置覆盖控件的 ControlSource 设置。

应用于

复选框，列，组合框，命令组，编辑框，列表框，OLE 绑定型控件，选项按钮，选项组，微调，文本框

请参阅

Bound 属性，Order 属性，RecordSourceType 属性，Value 属性

_CONVERTER 系统变量

包含 Visual FoxPro 转换器的应用程序名称。

语法

_CONVERTER = cProgramName

参数描述

cProgramName

指定一个转换器应用程序。若转换器应用程序不在 Visual FoxPro 当前默认目录下，应在应用程序名中包含路径。

也可通过在 Visual FoxPro 配置文件中包含下行来指定一个转换器应用程序。该行使用如下语法：

_CONVERTER = cProgramName

说明

当试图打开在 FoxPro 早期版本中创建的屏幕、报表或项目时，_CONVERTER 系统变量包含了 Visual FoxPro 使用的应用程序名。_CONVERTER 默认地包含 CONVERT.APP。CONVERT.APP 安装在 Visual FoxPro 目录下。可以为转换器应用程序指定一个不同的名字。

请参阅

系统变量概述

COPY FILE 命令

复制任何类型的文件。

语法

COPY FILE FileName1 TO FileName2

说明

COPY FILE **创建文件** *FileName1* **的一个备份。可使用** COPY FILE **复制任何类型的文件。要复制的文件不能处于打开状态。源文件名** *FileName1* **和目标文件名** *FileName2* **都要包含扩展名。**

FileName1 **和** FileName2 **能包含如** * **和** ? **之类的通配符。例如，要创建当前目录下扩展名为**.prg **的所有文件的备份，执行命令** COPY FILE *.prg *.bak**。**

若使用 COPY FILE **复制含有备注字段、结构索引或两者兼有的表，则必须同时复制** .FPT **和** .CDX **文件。**

请参阅

DELETE FILE **命令**, RENAME **命令**, RENAME TABLE **命令**

COPY INDEXES 命令

从单项索引 .IDX 文件创建复合索引标识。

语法

```
COPY INDEXES IndexFileList | ALL
  [TO CDXFileName]
```

参数描述

IndexFileList

指定 .IDX 单项索引文件，该文件的索引表达式用于创建标识。用逗号分隔索引文件名，每个标识的名称即为相应单项索引文件的基本名。如果由索引文件创建的标识与已有标识同名，则显示一个对话框（若 SAFETY 是 ON），询问是否要覆盖该标识。

ALL

指定从所有打开的单项索引文件创建索引标识。

TO CDXFileName

创建非结构复合索引文件中的标识。使用 *CDXFileName* 指定非结构复合索引文件名。若指定的非结构复合索引文件不存在，则 Visual FoxPro 自动创建该文件。

说明

复合索引文件是包含独立索引项（称为标识）的索引文件。每个标识由各自的唯一标识名来区别。复合索引文件的默认扩展名是 .CDX。

在使用 COPY INDEXES 之前，必须先打开表和单项索引文件。单项索引文件的索引表达式用于创建新标识。

若省略 TO 子句，则新标识被添加到与表一起自动打开的结构复合索引文件中。若表没有结构复合索引文件，COPY INDEXES 将创建一个。

可以使用 COPY TAG 由复合索引文件标识创建单项索引文件。

请参阅

CDX（）, COPY TAG, DELETE TAG, INDEX, TAG（）, USE

COPY MEMO 命令

将当前记录中指定备注字段的内容复制到文本文件。

语法

```
COPY MEMO MemoFieldName TO FileName
  [ADDITIVE]
  [AS nCodePage]
```

参数描述

MemoFieldName
指定要复制到文本文件的备注字段名。

TO FileName
定一个新的或已有的文本文件名，备注字段将要复制到该文件中。若在 *FileName* 中没有提供扩展名，则指定扩展名为 .TXT。也可在文件名中包含路径。

ADDITIVE
将备注字段内容追加到指定文本文件尾。若省略 ADDITIVE，则用备注字段内容替换文本文件内容。

AS nCodePage
指定 COPY MEMO 创建的文本文件的代码页。Visual FoxPro 复制指定备注字段的内容，并且在复制数据的同时，根据文本文件的代码页自动转换数据。

若为 *nCodePage* 指定了不被支持的值，Visual FoxPro 产生错误信息。可使用 GETCP（）来显示代码页对话框，并允许为 Visual FoxPro 创建的文件指定代码页。

若省略 AS *nCodePage* 或 AS *nCodePage* 为 0，则不发生代码页转换。

示例

在以下示例中，将名为 notes 的备注字段的内容复制到名为 Test.txt 文件中。然后再次复制该备注字段并将其附加到文本文件中。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE employee  && 打开 Employee 表
COPY MEMO notes TO test.txt
WAIT WINDOW 'Memo contents now in test.txt' NOWAIT
MODIFY FILE test.txt
COPY MEMO notes TO test.txt ADDITIVE
WAIT WINDOW 'Memo contents added again to test.txt' NOWAIT
MODIFY FILE test.txt
DELETE FILE test.txt
```

请参阅

APPEND MEMO 命令, COPY FILE 命令, COPY TO 命令, MODIFY MEMO 命令

COPY PROCEDURES 命令

将当前数据库中的内部存储过程复制到文本文件。

语法

COPY PROCEDURES TO FileName
[AS nCodePage] [ADDITIVE]

参数描述

FileName

指定文本文件名。内部存储过程将被复制到此文本文件中。若该文件不存在，Visual FoxPro 自动创建它。

AS nCodePage

指定文本文件的代码页，内部存储过程将被复制到该文本文件中。Visual FoxPro 复制内部存储过程，并且在指定了代码页的情况下，自动将内部存储过程转换到指定的代码页。

若为 *nCodePage* 指定了一个不被支持的值，Visual FoxPro 产生错误信息。可使用 GETCP() 显示代码页对话框，并允许为文本文件指定代码页。内部存储过程将被复制到该文本文件。

若省略 AS *nCodePage* 或 AS *nCodePage* 为 0，则不进行代码页转换。

ADDITIVE

将内部存储过程追加到指定文本文件尾。若省略 ADDITIVE，则内部存储过程替代文本文件的内容。

说明

利用 COPY PROCEDURES 和 APPEND PROCEDURES 可以编程方式修改数据库中的内部存储过程。当执行 COPY PROCEDURES 时，数据库必须是打开的当前数据库，否则 Visual FoxPro 产生错误信息。

示例

下面的示例打开 testdata 数据库，并使用 COPY PROCEDURES 将内部存储过程复制到一个名为 MYPROC.TXT 的临时文件中，然后使用 MODIFY FILE 打开这个临时文

本文件。若数据库中没有内部存储过程，则该临时文本文件为空文件。

如果在数据库中没有存储过程，您可以运行 APPEND PROCEDURES 的示例向数据库添加过程。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
COPY PROCEDURES TO myproc.txt && 将存储过程复制到文件中
MODIFY FILE myproc.txt  && 打开文件
DELETE FILE myproc.txt  && 清除文件
```

请参阅

APPEND PROCEDURES 命令, CREATE TRIGGER 命令, DISPLAY PROCEDURES 命令, MODIFY PROCEDURE 命令, SET DATABASE 命令

COPY STRUCTURE 命令

用当前选择的表结构创建一个新的自由表。

语法

```
COPY STRUCTURE TO TableName
  [FIELDS FieldList] [[WITH] CDX | [WITH] PRODUCTION]
  [DATABASE cDatabaseName [NAME cTableName]]
```

参数描述

TableName
指定要创建的自由表名称。
在 Visual FoxPro 中，新自由表中每一个字段的默认值以及是否支持 null 值与当前选定表的这些设置是相同的。

FIELDS FieldList
只将 *FieldList* 指定的字段复制到新表。若省略 FIELDS *FieldList*，则把所有字段复制到新表。

[WITH] CDX | [WITH] PRODUCTION
创建与已有表的结构索引文件相同的新表的结构索引文件。原始结构索引文件的标识和索引表达式都复制到新的结构索引文件。CDX 等同于 PRODUCTION 子句。在 Visual FoxPro 中，当前选定表的主索引转换成新的自由表的候选索引。
CDX 和 PRODUCTION 子句是相同的。
在 Visual FoxPro 中，将当前选定表的主索引转换为新空表的候选索引。

DATABASE cDatabaseName
指定要添加新表的现有数据库的名称。注意，表和字段的属性没有复制到该数据库。

NAME cTableName
指定在数据库中出现的表的名称。

示例

在以下示例中，打开“ customer ”表，将它的结构复制到一个名为“ backup ”的表中

并打开“backup”。然后使用 APPEND FROM 命令将“customer”表中的记录附加到“backup”中，打开“浏览”窗口浏览“backup”表。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

COPY STRUCTURE TO backup
USE backup
APPEND FROM customer FOR country = 'UK'
BROWSE FIELDS contact, country
USE
DELETE FILE backup.dbf
```

请参阅

COPY STRUCTURE EXTENDED, CREATE, DISPLAY STRUCTURE

COPY STRUCTURE EXTENDED 命令

创建一个新表，其中的字段包含当前选定表的结构信息。

语法

COPY STRUCTURE EXTENDED TO FileName
[DATABASE DatabaseName [NAME LongTableName]]

[FIELDS FieldList]

参数描述

FileName

指定要创建的新表。

DATABASE DatabaseName

指定要添加新表的数据库。

NAME LongTableName

指定新表的长名称。长名称最多可以包含 128 **个字符并且可以在数据库中使用短文件名的地方使用它。**

FIELDS FieldList

指定在新表的记录中只包含由 *FieldList* **指定的字段。若省略** FIELDS *FieldList*，**则所有字段在新表中都有一个记录。**

说明

当前选定表内每个字段的信息被复制到新表的一条记录中。新表的结构在格式上固定，由 16 **个字段组成。**

下表列出了这 16 **个字段的名称和内容。**

字段	**字段类型**	**内容**
FIELD_NAME	**字符型**	**当前选定表的字段名**

续表

FIELD_TYPE	字符型	Field types: C =字符型 Y = 货币型 N = 数值型 F = 浮点型 I = 整型 B =双精度型 D = 日期型 T =日期时间型 L = 逻辑值 M = 备注型 G =通用型
FIELD_LEN	数值型	字段宽度
FIELD_DEC	数值型	数值字段中的小数位数
FIELD_NULL	逻辑值	字段支持 null 值
FIELD_NOCP	逻辑值	不允许代码页转换(只用于字符型字段和备注型字段)
FIELD_DEFA	备注型	字段默认值
FIELD_RULE	备注型	字段有效性规则
FIELD_ERR	备注型	字段有效性文本
TABLE_RULE	备注型	表有效性规则

续表

TABLE_ERR	备注型	表有效性文本
TABLE_NAME	字符型	长表名(只用于第一个记录)
INS_TRIG	备注型	插入触发器表达式(只有第一个记录)
UPD_TRIG	备注型	更新触发器表达式(只有第一个记录)
DEL_TRIG	备注型	删除触发器表达式(只有第一个记录)
TABLE_CMT	备注型	表注释(只有第一个记录)

可对此新表进行修改，然后使用 CREATE FROM 创建一个不同结构的新表。COPY STRUCTURE 和 CREATE FROM 可以用来以编程方式更改表的结构。

在以前版本的 Visual FoxPro 、FoxPro for Windows 和 FoxPro for MS-DOS 中，FIELD_NAME 字段的宽度为 10。为了将 CREATE FROM 用于一个在 Visual FoxPro 5.0 或以前版本使用 COPY STRUCTURE EXTENDED 创建的表，必须将 FIELD_NAME 字段的宽度改为 10 个字符。注意，在 Visual FoxPro 3.0 以及以前版本中不支持有些字段。

示例

以下示例显示了 orders 表的结构，将扩展的结构复制到 temp 表中，浏览 temp 表，从 temp 表中创建 backup 表并显示 backup 的结构。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE orders  &&打开 Orders 表
CLEAR
DISPLAY STRUCTURE

WAIT WINDOW 'Structure of the orders table' NOWAIT
```

```
COPY STRUCTURE EXTENDED TO temp
USE temp
WAIT WINDOW 'The temp table - 1 row per field in orders' NOWAIT
BROWSE
CREATE backup FROM temp
USE backup
DISPLAY STRUCTURE
WAIT WINDOW 'Backup.dbf has the same structure as orders' NOWAIT
USE
DELETE FILE temp.dbf
DELETE FILE backup.dbf
```

请参阅

AFIELDS（），CREATE FROM

COPY TAG 命令

根据复合索引文件的标识创建单项索引 (.idx) 文件。

语法

```
COPY TAG TagName [OF CDXFileName]
  TO IndexFileName
```

参数描述

TagName

指定用来创建单项索引 .idx 文件的标识。

OF CDXFileName

指定包含标识的复合索引文件。若在打开的多个复合索引文件中有同名标识，则需要包含这个子句。若省略 OF *CDXFileName*，Visual FoxPro 先在结构索引文件中查找标识；若找不到，则 Visual FoxPro 查找所有打开的非结构复合索引文件。

TO IndexFileName

指定要创建的单项索引 .idx 文件名。

说明

使用 COPY TAG 可由 .cdx 复合索引文件的标识创建一个新的单项索引 .idx 文件。

用来创建单项索引 .idx 文件的复合索引文件必须是打开的。当打开表时，自动打开结构复合索引文件。而非结构复合索引则必须用 USE ... INDEX 或 SET INDEX 明确打开。

有关复合索引文件的详细内容，请参阅 INDEX。

可使用 COPY INDEX 从单项索引 .idx 文件创建复合索引文件的标识。

请参阅

CDX（），COPY INDEXES，DELETE TAG，INDEX，TAG（），USE

COPY TO 命令

用当前选定表的内容创建新文件。

语法

```
COPY TO FileName
  [DATABASE DatabaseName [NAME LongTableName]]
  [FIELDS FieldList
  | FIELDS LIKE Skeleton
  | FIELDS EXCEPT Skeleton]
  [Scope] [FOR lExpression1] [WHILE lExpression2]
  [[WITH] CDX] | [[WITH] PRODUCTION]
  [NOOPTIMIZE]
  [[TYPE] [FOXPLUS | FOX2X | DIF | MOD
  | SDF | SYLK | WK1 | WKS | WR1 | WRK | CVS | | XLS | XL5
  | DELIMITED [WITH Delimiter | WITH BLANK | WITH TAB
  | WITH CHARACTER Delimiter]]]
  [AS nCodePage]
```

参数描述

FileName

指定 COPY TO 要创建的新文件名。若文件名中不包含扩展名，则指定扩展名为文件类型的默认扩展名。若不指定文件类型，则 COPY TO 创建一个新的 Visual FoxPro 表，并且用默认扩展名 .DBF 指定表文件名。

DATABASE DatabaseName

指定要添加新表的数据库。

NAME LongTableName

指定新表的长名称。长名称最多可以包含到 128 个字符并且可以在数据库中使用短文件名的地方使用它。

FIELDS FieldList

指定要复制到新文件的字段。若省略 FIELDS *FieldLsit*，则将所有字段复制到新文件。若要创建的文件不是表，则即使备注字段名包含在字段列表中，也不把备注字段复制到新文件。

FIELDS LIKE Skeleton

指定与所给字段梗概相匹配的原始表中的字段。使用 COPY TO 创建的新文件中包含 Skeleton。

FIELDS EXCEPT Skeleton

指定在 COPY TO 创建的新文件中包含除了与字段梗概 *Skeleton* 匹配的所有字段。

字段梗概 *Skeleton* 支持通配符。例如，要想在新文件中包含以字母 A 和 P 开头的所有字段，可以使用下列代码：

```
COPY TO mytable FIELDS LIKE A*,P*
```

LIKE 子句可以和 EXCEPT 子句组合使用：

```
COPY TO mytable FIELDS LIKE A*,P* EXCEPT PARTNO*
```

Scope

指定要复制到新文件的记录范围。只有在范围内的记录才被复制。Scope 子句为：ALL、NEXT *nRecord*、RECORD *nRecordNumber* 和 REST。有关信息，请参阅帮助中的 Scope Clauses。

FOR lExpression1

指定只复制逻辑条件 *lExpression1* 为“真”(.T.) 的记录到文件中。包含 FOR *lExpression1* 可按条件复制记录，筛选出不想要的记录。

若 *lExpression1* 是可优化表达式，则 Rushmore 优化有 FOR *lExpression1* 子句的 COPY TO 命令。为获得最佳效果，可在 FOR *lExpression1* 子句中使用可优化表达式。

有关优化表达式的内容，请参阅稍后部分的 SET OPTIMIZE 命令和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”的“掌握 Rushmore 技术”。

WHILE lExpression2

指定一个条件，只有当逻辑表达式 lExpression2 为“真”(.T.) 时才复制记录。

[WITH] CDX | [WITH] PRODUCTION

创建一个与已有表的结构索引文件相同的新表结构索引文件。原始结构索引文件的标识和索引表达式被复制到新结构索引文件。CDX 等同于 PRODUCTION

子句。
若不是复制到 Visual FoxPro 新表，则不要包含 CDX 或 PRODUCTION。

NOOPTIMIZE

使 COPY TO 的 Rushmore 优化无效。
有关详细内容，请参阅稍后部分的 SET OPTIMIZE 命令。

TYPE

若要创建的文件不是 Visual FoxPro 表，则指定该文件类型。指定文件类型时不必包含 TYPE 关键字。

FOXPLUS

Visual FoxPro 备注文件与 FoxBASE+ 备注文件有不同的结构。若 Visual FoxPro 源表包含备注字段，则包含 FOXPLUS 子句可创建一个能在 FoxBASE+ 中使用的表。Visual FoxPro 备注字段不能包含二进制数据，因为 FoxBASE+ 不支持备注字段中出现二进制数据。

FOX2X

创建能在 FoxPro for Windows、FoxPro for Macintosh 和 FoxPro for MS-DOS 的早期版本（2.0，2.5 和 2.6 版）中打开的新表。
对于数值型、浮点型、整型、双精度型和货币型字段，源表中的 null 值被转换为新表中的 0。对于其他字段类型，源表中的 null 值被转换为新表中的空值（blank value)。有关空值 (blank value) 的详细内容，请参阅 ISBLANK（）。
下表列出了当包含 FOX2X 参数时，被转换为新表中不同字段类型的 Visual FoxPro 字段类型。

Visual 字段类型	FoxPro 2.x 字段类型
货币型	浮点型
日期时间型	日期型
双精度型	浮点型
整型	数值型

DIF

创建一个 VisiCalc .DIF（数据交换格式）文件。Visual FoxPro 表的字段变为矢量（列），记录变为元组（行）。若在 *FileName* 中不包含扩展名，则指定新文件的扩展名为 .DIF。

创建 Microsoft Multiplan 4.01 版文件。若不包含扩展名，则指定新的 Microsoft Multiplan 文件的扩展名为 .MOD。

创建 SDF（系统数据格式）文件。SDF 文件是 ASCII 文本文件，其中记录都有固定长度，并以回车和换行符结尾。字段不分隔。若不包含扩展名，则指定 SDF 文件的扩展名为 .TXT。注意，当使用 COPY TO 命令创建 SDF 文件时，会忽略 SET CENTURY 的设置。

创建 SYLK（符号连接）交换文件。SYLK 文件用于 Microsoft Multiplan。每个当前选定表中的字段变为电子表格中的一列，每个记录变为一行。SYLK 文件没有扩展名。

创建 Lotus 1-2-3 2.x 版的电子表格文件。每个当前选定表中的字段变为电子表格中的一列，每条记录变为一行。新建电子表格的文件扩展名指定为 .WK1。

WKS

创建 Lotus 1-2-3 1a 版的电子表格文件。当前选定表中的每个字段变为电子表

格中的一列，每条记录变为一行。新建电子表格的文件扩展名指定为 .WKS。

创建 Lotus Symphony 1.1 或 1.2 版的电子表格文件。当前选定表中的每个字段变为电子表格中的一列，每条记录变为一行。新建电子表格的文件扩展名指定为 .WR1。

WRK

创建 Lotus Symphony 1.0 版的电子表格文件。当前选定表中的每个字段变为电子表格中的一列，每条记录变为一行。新建电子表格的文件扩展名指定为 .WRK。

CVS

创建一个用逗号分隔各值的文件。一个 CSV 文件的第一行是字段名，文件中余下的字段值用逗号分隔。

创建 Microsoft Excel 2.0 版的电子表格文件。当前选定表中的每个字段变为电子表格中的一列，每条记录变为一行。若不包含文件扩展名，则新建电子表格的文件扩展名指定为 .XLS。

XL5

创建 Microsoft Excel 5.0 版的电子表格文件。当前选定表中的每个字段变为电子表格中的一列，每条记录变为一行。若不包含文件扩展名，则新建电子表格的扩展名指定为 .XLS。

创建分隔文件。分隔文件是 ASCII 文本文件，其中每条记录以一个回车和换行符结尾。默认的字段分隔符是逗号。因为字符型数据可能包含逗号，所以另外用双引号标识字符型字段。

除非另外指定，否则所有新建 DELIMITED 文件的扩展名都指定为 .TXT。

DELIMITED WITH Delimiter

用字符而不是等号分隔文件中的字符型字段。用字符 ***Delimiter*** 代替引号标识字符型字段。

DELIMITED WITH BLANK

用空格代替逗号分隔文件中的各个字段。

DELIMITED WITH TAB

用制表符代替逗号分隔文件中的各个字段。

DELIMITED WITH CHARACTER Delimiter

创建的文本文件中，各个字段由指定的 ***Delimite*** 分隔。若指定 ***Delimite*** 为分号，注意分号应用引号括起来，因为分号在 Visual Foxpro 中有特殊的意义：同一命令被分写在不同行中时，用分号作为行的结束。***Delimite*** 还可以是 BLANK 或 TAB。

注意 WITH ***Delimiter*** 可与 WITH CHARACTER ***Delimiter*** 配合使用。比如，在下面的例子中，创建了一个文本文件，其中各个字段由分号 ;分隔，字符字段用下划线 _ 标识。

```
COPY TO mytxt.txt DELIMITED WITH _ WITH CHARACTER ';'
```

AS nCodePage

指定 COPY TO 创建的表或文件的代码页。Visual FoxPro 复制当前选定表的内容，并且在复制数据的同时，自动将数据转换到为新表或文件指定的代码页。如可能，Visual FoxPro 用指定的代码页标记新建的表或文件。若为 nCodePage 指定了不被支持的值，Visual FoxPro 产生一条错误信息。可使用 GETCP（） 显示代码页对话框，并允许为

Visual FoxPro 创建的表或文件指定代码页。

若省略 AS nCodePage，则将新建的表或文件转换到 Visual FoxPro 当前代码页。

若 nCodePage 是 0，则不发生代码页转换，并且不用代码页标记新建的表或文件。

说明

若已设置了索引排序方式，则按主索引顺序复制记录。

示例

在下面的示例中，打开 Customer 表并且复制三个连续的记录到名为 TEMP.TXT 的新建分隔格式的数据文件中。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

COPY NEXT 3 TO temp TYPE DELIMITED
WAIT WINDOW 'This is the delimited text file' NOWAIT
MODIFY FILE temp.txt
DELETE FILE temp.txt
```

请参阅

APPEND FROM, COPY FILE, GETCP（）, EXPORT, IMPORT, RENAME TABLE

COPY TO ARRAY 命令

将当前选定表中的数据复制到数组。

语法

COPY TO ARRAY ArrayName
[FIELDS FieldList]
[Scope] [FOR lExpression1] [WHILE lExpression2]
[NOOPTIMIZE]

参数描述

ArrayName

指定数组名，将当前选定表中的数据复制到该数组中。

FIELDS FieldList

指定只将由 *FieldList* 指定的字段复制到数组。若省略 FIELDS *FieldList*，只要数组有足够的列，则复制所有字段到数组。

Scope

指定要复制到数组中的记录的范围。只有在范围内的记录才被复制。Scope 子句有：ALL、NEXT *nRecords*、RECORD *nRecordNumber* 和 REST。

有关 scope 子句的详细内容，请参阅帮助中的“Scope 子句”。

COPY TO ARRAY 的默认范围是 ALL 记录。

FOR lExpression1

指定只复制符合逻辑表达式 *lExpression1* 的记录到数组。包含 FOR 子句可按条件复制记录到数组，筛选出不想要的记录。

若 *lExpression1* 是可优化表达式，则 Rushmore 优化包含 FOR *lExpression1* 的 COPY TO ARRAY 查询。为获得最佳效果，可在 FOR 子句中使用可优化表达式。

有关优化表达式的内容，请参阅稍后部分的 SET OPTIMIZE 命令和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”的“掌握 Rushmore 技术”。

WHILE lExpression2

指定条件，只有当逻辑表达式 *lExpression2* 为“真”(.T.)

NOOPTIMIZE

使 COPY TO ARRAY 的 Rushmore 优化无效。有关详细内容，请参阅请参阅稍后部分的 SET OPTIMIZE 命令和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”的“掌握 Rushmore 技术”。

说明

COPY TO ARRAY 和 SCATTER 相似。但 COPY TO ARRAY 将多个记录复制到数组，而 SCATTER 只复制一条记录到数组或一组变量。若指定的数组不存在，则 COPY TO ARRAY 和 SCATTER 都创建一个新的数组。

要将单个记录复制到数组，可指定一维数组。指定的一维数组的元素数目必须与表中字段的数目相同，但不包括备注字段。在 COPY TO ARRAY 中不考虑备注字段。

若指定了一维数组，则记录的第一个字段存储到数组的第一个元素，第二个字段存储到

数组的第二个元素，依此类推。若一维数组的元素数目比表中字段的数目多，则多余元素保持不变。若数组元素比表中字段少，则忽略多余字段。

要将多个记录或整个表复制到数组，则指定一个二维数组。数组的行数就是数组能容纳的记录数，数组的列数就是数组能容纳的字段数。

每个记录存入数组的一行，记录的每个字段存入数组的一列。对每个记录，第一个字段存储在数组的第一列，第二个字段存储在数组的第二列，依此类推。若数组的列比表中的字段多，则不更改多余的列。若数组的列比表中的字段少，则多余的字段不存储到数组中。

数组中每一行都填充了表中的一条记录的内容。若数组的行比表中的记录多，则不更改任何多余的行。若数组的行比表中的记录少，则多余的记录不存储到数组中。

可以用 APPEND FROM ARRAY 将数组中的数据复制到新表中的记录。也可以用 GATHER 将数组或一组变量中的数据复制到表中的记录。

示例

在下面的示例中，先打开 customer 表。然后创建一个二维数组，并且复制 customer 的前三条记录到数组。DISPLAY MEMORY 显示存储在数组中的数据。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

DIMENSION gaTemp(3,10)
COPY NEXT 3 TO ARRAY gaTemp
DISPLAY MEMORY LIKE gaTemp
```

请参阅

APPEND FROM ARRAY 命令, DECLARE 命令, DIMENSION GATHER 命令, PUBLIC 命令, SCATTER 命令, STORE 命令

COS( ) 函数

返回数值表达式的余弦值。

语法

COS(nExpression)

返值类型

数值型

参数描述

nExpression

指定一个需要返回余弦值的数值表达式。*nExpression* 可以是任意值。

说明

COS( ) 返回以弧度表示的 *nExpression* 的余弦值。可用 DTOR( ) 把角度转换成弧度。COS( ) 返回值的小数位数由 SET DECIMALS 指定。COS( ) 返回值范围在 –1 到 1 之间。

示例

```
CLEAR
```

```
? COS(0) && 显示数值 1.00
? COS(PI() ) && 显示数值 -1.00
? COS(DTOR(180)) && 显示数值 -1.00
STORE PI() * 3 TO gnAngle
? COS(gnAngle) && 显示数值 -1.00
```

请参阅

ACOS()，DTOR()，RTOD()，SET DECIMALS，SIN()

COUNT 命令

统计表中记录数目。

语法

```
COUNT
  [Scope] [FOR lExpression1] [WHILE lExpression2]
  [TO VarName]
  [NOOPTIMIZE]
```

参数描述

Scope

指定需要统计的记录范围。scope **子句是** All、NEXT *nRecords*、RECORD *nRecordNumber* **和** REST。**包含有** *Scope* **参数的命令只能对活动工作区中的表进行操作。**

对 COUNT 命令，默认的范围是全部记录，即 ALL。

FOR lExpression1

指定只有满足逻辑条件 *lExpression1* 的记录才进行计数。包含 FOR 子句可以有条件地对记录进行计数，筛选出不合要求的记录。

如果 *lExpression1* 是可优化表达式，那么，Rushmore 将优化 COUNT FOR 查询。为获得最佳性能，请在 FOR 子句中使用可优化表达式。

有关优化表达式的内容，请参阅稍后部分的 SET OPTIMIZE 命令和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”的“掌握 Rushmore 技术”。

WHILE lExpression2

指定对记录进行计数的条件：只要逻辑表达式 *lExpression2* 的值为“真”(.T.)，则进行计数，直至遇到使该表达式的值为“假”(.F.)的记录为止。

TO VarName

指定用于存储记录数目的变量或数组。如果所指定的变量不存在，Visual FoxPro 会创建它。

NOOPTIMIZE

禁止 COUNT 进行 Rushmore 优化。有关详细内容，请参阅稍后部分的 SET OPTIMIZE 命令和《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”的“掌握 Rushmore 技术”。

说明

COUNT 在 FOR 或 WHILE 条件为“真”时，对一定范围内的记录进行计数。如果 SET TALK 是 ON，则显示记录的数目。

如果 SET DELETE 是 OFF，则带有删除标记的记录也包括在计数中。

有关 Null 值如何影响 COUNT 的讨论，请参阅帮助中的“语言概述”。

示例

下面的示例统计并显示在巴黎的顾客数目。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表

CLEAR
COUNT FOR UPPER(city) = 'PARIS'
DISPLAY FIELDS company, contact FOR UPPER(city) = 'PARIS'
```

请参阅

CALCULATE, SET DELETED, SET TALK

Count 属性

包含一个项目集合、文件集合或服务程序集合中的项目对象、文件对象或服务程序对象的数目。设计和运行时只读。

语法

Object.Count

说明

对于项目集合，如果没有项目管理器的实例，则 Count 属性包含零。对于文件或服务程序集合，如果在文件或服务程序集合中没有文件或服务程序，则 Count 属性包含零。

应用于

文件集合，项目集合，服务程序集合

请参阅

Item 方法

_COVERAGE 系统变量

包含一个 Visual FoxPro 应用程序的名称，该应用程序可以创建调试器代码输出范围。

语法

_COVERAGE = cProgramName

参数描述

cProgramName

指定一个代码范围输出应用程序。如果您指定的代码范围输出应用程序不在 Visual FoxPro 的当前默认目录中，需要在应用程序名称中包含路径。

您也可以在您的 Visual FoxPro 配置文件中指定一个代码范围输出应用程序，需要使用以

下语法：

```
_COVERAGE = cProgramName
```

说明

在默认情况下，_COVERAGE 包含 Coverage.app，它安装在您的 Visual FoxPro 目录中。您可以为这个代码范围输出应用程序指定其他名称。

请参阅

SET COVERAGE, SET EVENTTRACKING

CPCONVERT（） 函数

把字符、备注字段或字符表达式转换到其他代码页。

语法

```
CPCONVERT(nCurrentCodePage, nNewCodePage, cExpression)
```

参数描述

nCurrentCodePage

指定要转换 *cExpression* 的源代码页。

nNewCodePage

指定将 *cExpression* 转换到某一目标代码页。

ä 的字符：：

```
CPCONVERT(10000, 1252, gcCharExpr)
```

请参阅有关代码页的内容和 Visual FoxPro 的国际化支持，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十八章“开发国际化应用程序”中的“Visual FoxPro 支持的代码页”。

请参阅

CPCURRENT（），CPDBF（），MODIFY COMMAND，MODIFY FILE，SET NOCPTRANS

CPCURRENT（） 函数

返回 Visual FoxPro 配置文件中的代码页设置（若存在），或返回当前操作系统代码页。

语法

CPCURRENT([1 | 2])

说明

CPCURRENT（） 返回下面中的一个值。

- I在 Visual FoxPro 中，如果配置文件中不包含 CODEPAGE 配置项，返回当前操作系统代码页。在 FoxPro 的早期版本中，如果配置文件中不包含 CODEPAGE 配置项，则返回 0。
- 返回 CODEPAGE 配置项中指定的代码页编号。例如，如果配置文件中包含有下面的代码，CPCURRENT（） 返回 852：

CODEPAGE = 852

- 如果在配置文件中包含有下面的代码，则返回当前操作系统代码页：

CODEPAGE = AUTO

在 Visual FoxPro 中，不管配置项 CODEPAGE 如何设置，CPCURRENT(1) 都返回 Windows 代码页。

不管配置项 CODEPAGE 如何设置，CPCURRENT(2) 总是返回基础操作系统代码页。例如，如果正在运行 Windows，CPCURRENT(2) 返回 MS-DOS 代码页。

有关代码页的内容和 Visual FoxPro 的国际化支持，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十八章“开发国际化应用程序”中的“Visual FoxPro 支持的代码页”。

请参阅

CPCONVERT（）,CPDBF（）,MODIFY COMMAND,MODIFY FILE,
SETNOCPTRANS

CPDBF（） 函数

返回一个打开表所使用的代码页。

语法

CPDBF([nWorkArea | cTableAlias])

参数描述

nWorkArea
指定的工作区号。

cTableAlias
指定的表别名。

若要指定一个在非当前工作区中打开的表，请包含参数 *nWorkArea* 或 *cTableAlias*。如果在指定的工作区中没有打开的表，CPDBF（） 返回 0。如果不存在具有指定别名的表，Visual FoxPro 产生错误信息。

说明

DISPLAY STRUCTURE 也显示一个打开表所使用的代码页。

有关代码页的内容和 Visual FoxPro 的国际化支持，请参阅《Microsoft Visual FoxPro 6.0

中文版程序员指南》第十八章“开发国际化应用程序”中的“Visual FoxPro 支持的代码页”。

请参阅

CPCONVERT（）,CPCURRENT（）,MODIFY COMMAND,MODIFY FILE,SETNOCPTRANS

CREATE 命令

生成一个新表。

语法

CREATE [FileName | ?]

参数描述

FileName

指定要创建的表名。

?

显示“创建”对话框，提示您为正在创建的表命名。

说明

在 Visual FoxPro 中，如果在创建表时打开了一个数据库，则表自动添加到该数据库中。

在 Visual FoxPro 中，不能创建和 MS-DOS 设备同名的表，如 CON、NUL、PRN 和 COM1。应该避免在表名中使用连字符，因为用连字符连起来的表名不能出现在“查看”窗口中，而且会和别名指示符 (->) 弄混。
在 Visual FoxPro 中，不能创建和 MS-DOS 设备同名的表，如 CON、NUL、PRN 和 COM1。应该避免在表名中使用连字符，因为用连字符连起来的表名不能出现在“查看”窗口中，而且会和别名指示符 (->) 弄混。

请参阅

ADD TABLE, ALTER TABLE – SQL, CREATE DATABASE, CREATE TABLE – SQL, MODIFY STRUCTURE

CREATE CLASS 命令

打开类设计器，创建一个新的类定义。

语法

```
CREATE CLASS ClassName | ? [OF ClassLibraryName1 | ?]
  [AS cBaseClassName [FROM ClassLibraryName2]] [NOWAIT]
```

参数描述

ClassName

指定要创建的类定义的名称。

?

显示“新类”对话框，在框内指定要创建的类定义的名称。

OF ClassLibraryName1

指定要创建的 .VCX 可视类库的名称。如果已经存在 .VCX 可视类库，则可在其中添加类定义。

可视类库的文件扩展名一般为 .VCX。如果指定的可视类库的文件扩展名不是 .VCX，则应该将这个扩展名明确地包含在文件名中。

?

显示“新类”对话框，在框内指定新的或已存在的 .VCX 可视类库的名称，以便向其中添加类定义。

AS cBaseClassName

指定一个类，新类派生于这个类。*cBaseClassName* 可以是除了 Column、Cursor、DataEnvironment、Header、Page 和 Relation 之外的任何 Visual FoxPro 基类。

如果省略 AS *cBaseClassName*，则类定义派生于 Visual FoxPro 的 FormSet 基类。

FROM ClassLibraryName2

指定 .VCX 可视类库的名称，该可视类库应包含有 *cBaseClassName* 指定的用户自定义类。

NOWAIT

在类设计器打开之后继续程序的执行。程序不必等待类设计器关闭，而是继续执行 CREATE CLASS NOWAIT 之后的程序行。如果省略 NOWAIT，当在程序中发出 CREATE CLASS 时，类设计器打开，并且在类设计器关闭之前暂停

执行程序。

NOWAIT 仅在程序中才有效。当在命令窗口中发出 MODIFY CLASS NOWAIT 时，该项无效。

说明

用 CREATE CLASS 可以创建类定义并把它保存到 .VCX 可视类库中。这样的 .VCX 可视类库可以用 SET CLASSLIB 打开，并可以访问其中的类定义。

请参阅

ADD CLASS, _BROWSER, CREATE CLASSLIB, DEFINE CLASS, _INCLUDE, MODIFY CLASS, RELEASE CLASSLIB, SET CLASSLIB

CREATE CLASSLIB 命令

创建一个新的、空的可视类库 (.VCX) 文件。

语法

CREATE CLASSLIB ClassLibraryName

参数描述

ClassLibraryName

指定要创建的可视类库的名称。如果指定的可视类库名称已存在，并且 SET SAFETY 是 ON，Visual FoxPro 会询问是否要改写已存在的可视类库。如果

SET SAFETY 是 OFF，则已存在的文件被自动改写。

如果没有为文件指定扩展名，Visual FoxPro 自动指定扩展名为 .VCX。

说明

可用 ADD CLASS 和 CREATE CLASS 将类定义添加到可视类库中。

示例

下面的示例用 CREATE CLASSLIB 创建可视类库 1_clslib。同时，创建派生于 Visual FoxPro 基类 FORM 的 myform 类，并把它存储在 1_clslib 可视类库中。SET CLASSLIB 打开 1_clslib 可视类库，以便能够使用其中的类。

```
CREATE CLASSLIB myclslib    && 创建一个新的 .VCX 可视类库
CREATE CLASS myform OF myclslib AS "Form"  && 创建新类
SET CLASSLIB TO myclslib ADDITIVE    && 打开 1_clslib.VCX
```

请参阅

ADD CLASS, _BROWSER, CREATE CLASS, DEFINE CLASS, MODIFY CLASS, RELEASE CLASSLIB, SET CLASS

CREATE COLOR SET 命令

从当前颜色设置中创建一个新的颜色设置。

语法

CREATE COLOR SET ColorSetName

参数描述

ColorSetName
指定要创建的颜色设置的名称。

说明

配色方案的每个颜色对都存储在用户创建的颜色设置中。在 Visual FoxPro 中，一个颜色设置的名称最长可至 24 个字符（在 FoxPro 早期版本中是 10 个字符），可以包含数字和下划线，但不能以数字开头。

如果创建了颜色集合，便可用 SET COLOR SET 来加载它。

颜色设置存储在 Visual FoxPro 资源文件中。如果已存在同名的颜色设置，则改写之。

有关颜色方案和颜色对的详细内容，请参阅稍前部分的“颜色概述”。

请参阅

SET COLOR OF SCHEME, SET COLOR SET, SET COLOR TO

CREATE CONNECTION 命令

创建一个命名连接，并把它存储在当前数据库中。

语法

```
CREATE CONNECTION [ConnectionName | ?]
  [DATASOURCE cDataSourceName]
  [USERID cUserID] [PASSWORD cPassWord]
  [DATABASE cDatabaseName]
  | CONNSTRING cConnectionString]
```

参数描述

ConnectionName
指定要创建的连接的名称。

?
显示连接设计器，可在其中创建和存储连接。

DATASOURCE cDataSourceName
为连接指定 ODBC 数据源的名称。

USERID cUserID
为 ODBC 数据源指定用户标识。

PASSWORD cPassWord
为 ODBC 数据源指定密码。

DATABASE cDatabaseName
指定连接服务器上的数据库。

CONNSTRING cConnectionString
为 ODBC 数据源指定一个连接串。可使用连接串来代替 ODBC 数据源、用户

标识和密码。

说明

如果省略了可选的参数，连接设计器对话框就会出现，允许交互地创建一个连接。

示例

下面的示例假设 ODBC 数据源 MyFoxSQLNT 是可用的。数据源的用户 ID 是“Sa”。首先打开 testdata 数据库，并创建 Myconn 连接。然后用 DISPLAY CONNECTIONS 显示数据库中的命名连接，最后用 DELETE CONNECTION 从数据库中删除该连接。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
CREATE CONNECTION Myconn DATASOURCE "MyFoxSQLNT" USERID "sa"

CLEAR
DISPLAY CONNECTIONS &&在数据库中显示命名连接
DELETE CONNECTION Myconn &&删除刚刚创建的连接
```

请参阅

DELETE CONNECTION, DBGETPROP（）, DISPLAY CONNECTIONS, LISTCONNECTIONS , MODIFY CONNECTION, OPEN DATABASE, RENAMECONNECTION

CREATE CURSOR – SQL 命令

创建一个临时表。

语法

```
CREATE CURSOR alias_name
  (fname1 type [(precision [, scale])
      [NULL | NOT NULL]
      [CHECK lExpression [ERROR cMessageText]]
      [DEFAULT eExpression]
      [UNIQUE]
      [NOCPTRANS]]
  [, fname2 ...])
  | FROM ARRAY ArrayName
```

参数描述

alias_name

指定要创建的临时表名， alias_name **可以是一个名称表达式。**

fname

指定临时表中的字段名。每个 *fname* **可以是一个名称表达式。**

type

指定一个字母来表明字段的数据类型。

precision

指定由 *fname* 指定的字段的宽度。有些数据类型要求指定 *precision*。

scale

为指定的数据类型指定小数位数。有些数据类型要求指定 *scale*。

下表列出了可用的 *type*、*precision* 和 *scale*：

Field Type	nField Width	nPrecision	说明
C	n	–	宽度为 n 的字符字段
D	–	–	日期型
T	–	–	日期时间型
N	n	d	有 d 个小数位、宽度为 n 的数值型字段。
F	n	d	有 d 个小数位、宽度为 n 的浮点数值型字段。
I	–	–	整型
B	–	d	双精度型
Y	–	–	货币型
L	–	–	逻辑值
M	–	–	备注型
G	–	–	通用型
P	–	–	图片

nFieldwidth 和 *nPrecision* 对 D、T、Y、L、M、G 和 P 类型不适用。对于 N、F 或 B 类

型，若没有包含 *nPrecision*，则默认为零（没有小数位）。

NULL

在字段中允许 null 值。

NOT NULL

在字段中不允许 null 值。

如果省略了 NULL 和 NOT NULL，则 SET NULL 的当前设置决定字段中是否允许 null 值。但是，如果省略了 NULL 和 NOT NULL，并且包含 PRIMARY KEY 或 UNIQUE 子句，SET NULL 的当前设置无效，字段默认为 NOT NULL。

CHECK lExpression

指定字段的有效性规则，*lExpression* 可以是用户自定义函数。

ERROR cMessageText

指定当字段有效性规则产生错误时，Visual FoxPro 显示的错误信息。只有在浏览窗口或编辑窗口中的数据改变时，才显示该信息。

DEFAULT eExpression

指定字段的默认值，*eExpression* 的数据类型必须和字段的数据类型相同。

UNIQUE

创建字段的候选索引。候选索引标识和字段同名。

有关候选索引的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第七章“处理表”的“表的索引”。

注意　候选索引（通过包含 UNIQUE 选项所创建）和用 INDEX 命令中的 UNIQUE 选项创建的索引不同。用 INDEX 命令中的 UNIQUE 选项创建的索引允许重复的索引关键字，候选索引不允许重复的索引关键字。

用作候选索引的字段中不允许有 null 值和重复记录。但是，如果为支持 null 值的字段创建候选索引，Visual FoxPro 并不产生错误。如果试图在使用候选索引的字段中输入 null 值或重复值，Visual FoxPro 将产生错误。

NOCPTRANS

防止字符字段和备注字段转换到不同的代码页。如果临时表转换到其他代码页，那么，指定了 NOCPTRANS 的字段并不转换。只能对字符字段和备注字段指定 NOCPTRANS。

以下示例创建了一个名为 MYCURSOR 的临时表，该表中包含了两个字符字段和两个备注字段。第二个字符字段 CHAR2 和第二个备注字段 MEMO2 包含 NOCPTRANS 关键字以防止翻译。

```
CREATE CURSOR mycursor (char1 C(10), char2 C(10) NOCPTRANS,;
    memo1 M, memo2 M NOCPTRANS)
```

FROM ARRAY ArrayName

指定一个已存在数组的名称，其中包含有临时表的每个字段的名称、类型、精度和比例。有关数组内容的正确格式，请参阅 AFIELDS( )。

说明

CREATE CURSOR 创建了一个在被关闭之前一直存在的临时表。用 CREATE CURSOR 创建的临时表可像其他表一样操作——可进行浏览和索引，也能追加和修改记录。

临时表在最低的可用工作区中打开，并可以通过它的别名来访问它。临时表中的每个字段均由名称、类型、精度和比例定义。这些定义可从命令本身或从数组中获得。不管 SET EXCLUSIVE 的设置如何，临时表都是独占地打开。

示例

下面的示例创建了临时表 MYCURSOR，该表中包含有两个字符字段和两个备注字段。第二个字符字段 CHAR2 和第二个备注字段 MEMO2 包含了 NOCPTRANS 来防止转换。

```
CLOSE DATABASES
CLEAR
CREATE CURSOR employee ;
    (EmpID N(5), Name C(20), Address C(30), City C(30), ;
     PostalCode C(10), OfficeNo C(8) NULL, Specialty M)
DISPLAY STRUCTURE
WAIT WINDOW "Press a key to add a record."

INSERT INTO employee (EmpId, Name, Address, City, PostalCode, ;
        OfficeNo, Specialty);
    VALUES (1002, "Dr. Bonnie Doren", "University of Oregon", "Eugene", ;
            "98403", "", "Secondary Special Education")
BROWSE

* At this point you could copy this record to a permanent table
CLOSE ALL  && 一旦临时表关闭，所有数据从内存中清除
CLEAR
```

请参阅

AFIELDS（），CREATE，CREATE QUERY，CREATE TABLE – SQL，INSERT – SQL，MODIFY QUERY，SELECT – SQL

CREATE DATABASE 命令

创建一个数据库，并打开它。

语法

CREATE DATABASE [DatabaseName | ?]

参数描述

DatabaseName

指定要创建的数据库的名称。

如果 SAFETY 设为 ON，并且所指定的数据库和某一已存在的数据库有相同的名称和路径，Visual FoxPro 将显示一个警告对话框，提示您为数据库指定一个新的路径或名称。

?

显示“创建”对话框，在这个框中指定要创建的数据库名称。

说明

据库文件的扩展名为 .DBC，关联的数据库备注文件扩展名为 .DCT，关联的索引文件扩展名为 .DCX。

不管 SET EXCLUSIVE 的设置如何，数据库都独占地打开。由于 CREATE DATABASE 创建并打开数据库，因而不必随后再发出 OPEN DATABASE 命令来打开数据库。

如果发出 CREATE DATABASE 时不带任何可选参数，将显示“创建”对话框，提示您指定数据库的名称。

示例

本示例创建了数据库 people，同时创建表 friends 并自动把它添加到数据库中。DISPLAY TABLES 用来显示数据库中的表，而 DISPLAY DATABASES 用来显示数据库中表的信息。

```
CREATE DATABASE people
CREATE TABLE friends (FirstName C(20), LastName C(20))
CLEAR
DISPLAY TABLES  && 显示数据库中的表
DISPLAY DATABASES  && 显示表的信息
```

请参阅

ADD TABLE, CLOSE DATABASES, DBC（）, DBGETPROP（）, DBSETPROP（）, DELETE DATABASE, DISPLAY TABLES, FREE TABLE, MODIFY DATABASE, OPEN DATABASE, PACK DATABASE, REMOVE TABLE, SET DATABASE, VALIDATE DATABASE

CREATE FORM 命令

打开表单设计器。

语法

CREATE FORM [FormName | ?]

[AS cClassName FROM cClassLibraryName | ?]
[NOWAIT] [SAVE] [DEFAULT]
[[WINDOW WindowName1]
[IN [WINDOW] WindowName2 | IN SCREEN]]

参数描述

FormName

指定表单的文件名。如果没有为文件名指定扩展名，Visual FoxPro 自动指定 .SCX 为扩展名。如果指定的表单文件名已经存在，那么将提示您是否要改写已存在的文件（如果 SET SAFETY 设为 ON）。

?

显示“创建”对话框，可从中选择表单或输入要创建的新表单名。

AS cClassName FROM cClassLibraryName | ?

根据一个 .vcx 可视类库中的表单类创建一个新表单。cClassName 指定了用户自定义表单类的名称，根据该类创建新表单。如果 cClassName 不是基于一个表单类，则会产生错误。cClassLibraryName 指定了可视类库的名称，其中包含 cClassName 指定了表单类。包含 ? 可以打开一个“打开”对话框，允许指定一个可视类库。

NOWAIT

在表单设计器打开之后继续程序的执行。程序不必等待表单设计器关闭，而是继续执行 CREATE FORM NOWAIT 之后的程序行。如果省略 NOWAIT，当在程序中发出 CREATE FORM 时，表单设计器打开，并且在表单设计器关闭之前暂停程序的执行。

当在命令窗口中发出 CREATE FORM 命令时，NOWAIT 不起作用。

SAVE

在程序中发出包含 SAVE 子句的命令，在激活另一个窗口之后，表单设计器仍然打开。从命令窗口中发出命令时包含 SAVE，则无此作用。

DEFAULT

指定用默认的 Visual FoxPro 表单模板打开表单设计器，覆盖选项对话框中表单选项卡指定的默认表单模板。

有关表单模板的其他内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第九章“创建表单”。

WINDOW WindowName1

指定一个窗口，表单设计器采用该窗口的特性。例如，如果用 DEFINE WINDOW 的 FLOAT 选项创建窗口，表单设计器可以移动。窗口不必是活动的或可见的，但必须是已定义的

表单设计器的默认尺寸可能比它的特性窗口大。在这种情况下，表单设计器仍采用它所在的窗口的特性。表单设计器的左上角和窗口的左上角坐标相同，但其尺寸超出了窗口的边界。

IN [WINDOW] WindowName2

指定打开表单设计器的父窗口。表单设计器不采用父窗口的特性，也不能移出父窗口之外。如果父窗口移动，表单设计器也随之移动。

要访问表单设计器，父窗口必须先用 DEFINE WINDOW 定义，而且必须可见。

IN SCREEN

表单设计器放在父窗口中之后，指定在 Visual FoxPro 主窗口中打开该表单设

计器。通过在命令中包含 IN WINDOW 子句，可以把表单设计器放在父窗口中。

说明

发出不带任何附加参数的 CREATE FORM 命令，可以在表单设计器中打开一个新的表单。退出表单设计器时，会提示您用不同的名称存储表单。

请参阅

_BROWSER, COMPILE FORM, DO FORM, GETPEM（）, _INCLUDE, MODIFY FORM, PEMSTATUS（）, SYS(1269), SYS(1270), SYS(1271), SYS(1272)

CREAT FROM 命令

从 COPY STRUCTURE EXTENDED 文件创建一个表。

语法

```
CREATE
  [FileName1 [DATABASE DatabaseName [NAME LongTableName]]]
  FROM [FileName2]
```

参数描述

FileName1

要创建的新表的名称。

DATABASE DatabaseName

新表将被加入到本处指定的数据库中。

NAME LongTableName

指定新表在数据库中的长表名，最多可包含 128 个字符。若不指定，则在数据库中使用 *FileName1* 作为表名。

FileName2

FileName2 是一张表，其中记录 *FileName1* 中各个字段的信息。*FileName2* 可由 COPY STRUCTURE 命令创建，或由手工生成。

说明

CREATE 假设由 *FileName2* 指定的表已经用 COPY STRUCTURE EXTENDED 或用人工方式创建了。利用 *FileName2* 中说明的结构创建新表 *FileName1*。新创建的表成为活动表。

如果命令中没有包括 *FileName1* 或 *FileName2*，或者两个都不包括，则出现一个对话框，可在其中指定要创建的文件或 FROM 文件，或两者都指定。

请注意 *FileName2* 中的所有记录，包括那些作了删除标记的记录，都用来创建 *FileName1*。

示例

以下示例显示了 orders 表的结构，复制扩展的结构复制到 temp 表，浏览 temp，从 temp 中创建名为 backup 的表并显示 backup 的表结构

```
CLOSE DATABASES
```

```
CLEAR
SET PATH TO (HOME(2) + 'Data\')    && 设置数据库的路径
USE orders
DISPLAY STRUCTURE
WAIT WINDOW 'Structure of the orders table' NOWAIT
COPY STRUCTURE EXTENDED TO temp
USE temp
WAIT WINDOW 'Temp table has 1 row per field in ORDERS' NOWAIT
BROWSE
CREATE backup FROM temp
USE backup
DISPLAY STRUCTURE
WAIT WINDOW 'Backup.dbf has the same structure as ORDERS' NOWAIT
USE
DELETE FILE temp.dbf
DELETE FILE backup.dbf
```

请参阅

COPY STRUCTURE EXTENDED

CREATE LABEL 命令

打开标签设计器，以创建标签。

语法

CREATE LABEL [FileName | ?]
[NOWAIT] [SAVE]
[WINDOW WindowName1]
[IN [WINDOW] WindowName2 | IN SCREEN

参数描述

FileName

指定标签的文件名。如没有给该文件指定一个扩展名，Visual FoxPro 自动指定 .LBX 为扩展名。

?

显示“创建”对话框，提示您为要创建的标签命名。

NOWAIT

在标签设计器打开之后继续执行程序。程序不必等待标签设计器关闭，而是继续执行 CREATE LABLE NOWAIT 后面的程序行。如果在程序中发出 CREATE LABLE 时省略 NOWAIT，则在标签设计器打开之后、暂停程序的执行，直至关闭标签设计器。

当在命令窗口中发出 CREATE LABLE 命令时，NOWAIT 不起作用。

SAVE

在激活其他窗口后，标签设计器保持打开。如果省略 SAVE，在其他窗口被激活时关闭标签设计器。

在命令窗口中发出命令时包含 SAVE，则无此作用。

WINDOW WindowName1

指定一个窗口，标签设计器采用该窗口的特性。例如，如果该窗口是用 DEFINE WINDOW 命令中 FLOAT 选项来创建的，则标签设计器可以移动。该窗口不必是活动的或可见的，但必须是已定义的。

标签设计器的默认尺寸可能比它的特性窗口大。在这种情况下，标签设计器仍采用该窗口的特性。标签设计器的左上角坐标和窗口的左上角坐标相同，但其尺寸超出了窗口的边界。

IN [WINDOW] WindowName2

指定打开标签设计器的父窗口。标签设计器不采用父窗口的特性，也不能移出父窗口之外。如父窗口移动，标签设计器也随之移动。

要访问标签设计器，父窗口必须先用 DEFINE WINDOW 定义，而且必须是可见的。

IN SCREEN

将标签设计器放入父窗口之后，指定在 Visual FoxPro 主窗口中打开标签设计器。可通过在命令中包含 IN WINDOW 子句，把标签设计器放入父窗口。

说明

用 CREATE LABEL 可按标准格式创建标签或设计自定义标签。也可以使用标签向导来创建标签。

有关创建标签的详细内容，请参阅帮助中的“用户指南”中的第七章“设计报表和标签”。

请参阅

LABEL, MODIFY LABEL

CREATE MENU 命令

打开菜单设计器。

语法

```
CREATE MENU [FileName | ?]
  [NOWAIT] [SAVE]
  [WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN
```

参数描述

FileName

指定菜单表的文件名。如果没有为文件指定扩展名，Visual FoxPro 自动指定 .MNX 为扩展名。

?

显示“创建”对话框，提示您为正创建的菜单命名。

NOWAIT

在菜单设计器打开之后继续执行程序。程序不必等待菜单设计器关闭，而是继续执行 CREATE MENU NOWAIT 后面的程序行。如果省略 NOWAIT，当在程序中发出 CREATE MENU 时，菜单设计器打开，并且在它关闭之前暂停程序的执行。

如果从命令窗口发出带有 NOWAIT 的 CREATE MENU 命令，则不显示“新建菜单”对话框。在“新建菜单”对话框可以指定要创建的菜单的类型（标准或快捷）。

SAVE

在激活其他窗口之后，保持菜单设计器打开。如省略 SAVE，当激活其他窗口时，菜单设计器关闭。如果从命令窗口发出命令时包含 SAVE，则无此作用。

WINDOW WindowName1

指定一个窗口，菜单设计器采用该窗口的特性。例如，如果窗口是用 DEFINE WINDOW 命令中 FLOAT 选项创建的，则菜单设计器可以移动。该窗口不必是活动的或可见的，但必须是已定义的。

菜单设计器的默认尺寸可能比它的特性窗口大。在这种情况下，菜单设计器仍然采用该窗口的特性。菜单设计器的左上角坐标和窗口的左上角坐标相同，但其尺寸超出窗口的边界。

IN [WINDOW] WindowName2

指定打开菜单设计器的父窗口。菜单设计器不采用父窗口的特性，也不能移到父窗口之外。如父窗口移动，菜单设计器也随之移动。

要访问菜单设计器，父窗口必须首先用 DEFINE WINDOW 定义，并且必须是可见的。

IN SCREEN

在菜单设计器放到父窗口中后，指定在 Visual FoxPro 主窗口中打开菜单设计器。可通过在命令中包含 IN WINDOW，把菜单设计器放到父窗口中。

说明

发出不带任何附加参数的 CREATE MENU 命令，将打开菜单设计器，您可在其中定义一个菜单系统，暂时给菜单定义指定名称“菜单 1”。退出菜单设计器时，可用不同的

名称存储菜单定义。

有关创建菜单的详细内容，请参阅《Microsoft Visual FoxPro 6.0中文版程序员指南》第十一章“设计菜单和工具栏”。

请参阅

DEFINE MENU, MODIFY MENU

CREATE PROJECT 命令

打开项目管理器并创建一个项目。

语法

```
CREATE PROJECT [FileName | ?]
  [NOWAIT] [SAVE]
  [WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN
  [NOSHOW] [NOPROJECTHOOK]
```

参数描述

FileName

指定项目表的文件名。如果没有为文件指定一个扩展名，Visual FoxPro自动

指定 .PJX 为扩展名。

?

显示“创建”对话框，提示您为正创建的项目命名。

NOWAIT

在项目管理器打开之后继续执行程序。程序不必等待项目管理器关闭，而是继续执行 CREATE PROJECT NOWAIT 之后的程序行。如果省略 NOWAIT，当在程序中发出 CREATE PROJECT 时，项目管理器打开，并且在它关闭之前暂停程序的执行。

当在命令窗口中发出 CREATE PROJECT 时，NOWAIT 不起作用。

SAVE

在激活其他窗口之后，保持项目管理器打开。如省略 SAVE，在激活其他窗口时，关闭项目管理器。当从命令窗口发出命令时包含 SAVE，则无此作用。

WINDOW WindowName1

指定一个窗口，项目管理器具有该窗口的特性。例如，如果用 DEFINE WINDOW 命令中的 FLOAT 选项创建该窗口，则项目管理器可以移动。该窗口不必是活动的或可见的，但必须是已定义的。

项目管理器的默认尺寸可能比它的特性窗口大。在这种情况下，项目管理器仍然使用该窗口的特性。项目管理器的左上角坐标和窗口的左上角坐标相同，但其尺寸超出了窗口的边界。

IN [WINDOW] WindowName2

指定打开项目管理器的父窗口。项目管理器不采用父窗口的特性，也不能移到父窗口之外。如父窗口移动，项目管理器也随之移动。

要访问项目管理器，必须先用 DEFINE WINDOW 定义父窗口，且父窗口必须是可见的。

IN SCREEN

在项目管理器放入父窗口之后，指定在 Visual FoxPro 主窗口中打开项目管理器。可通过包含 IN WINDOW 子句把项目管理器放到父窗口中。

NOSHOW

指定当打开项目管理器（Visible 属性设置为“假” (.F.)）时，隐藏它。为了显示项目管理器，可将项目管理器的 Visible 属性设置为“真” (.T.)。NOSHOW 允许您在项目管理器中显示一个项目之前管理它。注意，为了避免与 NOSHADOW 关键字混淆，不可以将 NOSHOW 缩减为少于 5 个字符。

NOPROJECTHOOK

指定当打开项目管理器时，不创建一个项目管理器对象。对于由项目管理器挂接程序通过编程管理的项目，可以包含 NOPROJECTHOOK 关键字。应该注意的是无论何时打开项目文件 (.pjx),都会同时创建项目对象。

说明

项目是一个表，它记录了创建一个应用程序所需的所有文件以及文件间的所有关系、引用和连接。每个项目表都有一个 .PJX 扩展名以及一个以 .PJT 为扩展名的备注文件。在项目中，可以为应用程序指定它所要求的所有设置，Visual FoxPro 确保根据最新的源文件编译文件。

可用 USE 打开项目表，并可像操作其他 Visual FoxPro 表那样操作它。

发出不带任何附加参数的 CREATE PROJECT 命令会显示“创建”对话框，允许您为项目指定名称。

有关编译项目的详细内容，请参阅《Microsoft Visual FoxPro 6.0中文版程序员指南》第十三章“编译应用程序”。

请参阅

BUILD APP, BUILD PROJECT, MODIFY PROJECT

CREATE QUERY 命令

打开查询设计器。

语法

```
CREATE QUERY [FileName | ?]
  [NOWAIT]
```

参数描述

FileName

指定查询的文件名。如果没有为文件指定一个扩展名，Visual FoxPro自动指定 .QPR为扩展名。

?

显示“创建”对话框，提示您为要创建的查询命名。

NOWAIT

打开查询设计器之后继续执行程序。程序不必等待查询设计器关闭，而是继续

执行 CREATE QUERY NOWAIT 后面的程序行。如果省略 NOWAIT，当在程序中发出 CREATE QUERY 命令时，查询设计器打开，并且在它关闭之前暂停程序的执行。

只有在程序中发出命令时包含 NOWAIT 才有效。当在命令窗口中发出命令时包含 NOWAIT，则无此作用。

说明

CREATE QUERY 打开查询设计器，以便能交互地创建查询。

SQL SELECT 命令用来从表中检索数据。SELECT 非常强大，并能代替一系列 Visual FoxPro 命令。由于一个 SQL SELECT 实现了一系列 Visual FoxPro 命令的功能，所以 SELECT 能优化程序的性能。

可把 SELECT 看作是向 Visual FoxPro 提出查询以从表中获得信息的一种方法。SELECT 允许您指定所需要的信息，而不用告诉 Visual FoxPro 如何去检索这些信息。由 Visual FoxPro 决定检索信息的最好途径。

创建了查询之后，该查询作为 Visual FoxPro 程序文件以 .QPR 扩展名存储起来，查询程序可以用 DO 来执行。在用 DO 执行查询时，必须包含查询文件扩展名，如下面的示例所示。

```
DO my_query.qpr
```

发出一个不带任何参数的 CREATE QUERY 命令时，打开一个新的查询窗口，并为该查询指定文件名“查询 1”。退出查询窗口时，可用不同的名称存储该查询。

有关创建查询的详细内容，请参阅帮助中的“用户指南”第四章“检索数据”中的“创建查询”。

请参阅

CREATE REPORT 命令

在报表设计器中打开报表。

语法

```
CREATE REPORT [FileName | ?]
  [NOWAIT] [SAVE]
  [WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN
```

参数描述

FileName

指定报表的文件名。如果没有为文件指定一个扩展名，Visual FoxPro 自动指定 .FRX 为扩展名。如果所指定的报表名称已经存在，将提示您是否要改写已存在的文件（如果 SET SAFETY 是 ON）。

?

显示“创建”对话框，提示您为正创建的报表命名。

NOWAIT

在报表设计器打开之后继续执行程序。程序不必等待报表设计器关闭，而是继

续执行 CREATE REPORT NOWAIT 后面的程序行。如果省略 NOWAIT，当在程序中发出 CREATE REPORT 时，报表设计器打开，并且在它关闭之前暂停程序的执行。

在命令窗口中发出 CREATE REPORT 命令时，NOWAIT 不起作用。

SAVE

在激活其他窗口之后，保持报表设计器打开。如省略 SAVE，在激活其他窗口时关闭报表设计器。

当从命令窗口发出命令时包含 SAVE，则无此作用。

WINDOW WindowName1

指定一个窗口，报表设计器具有该窗口的特性。例如，如果窗口用 DEFINE WINDOW 命令中的 FLOAT 选项来创建，则报表设计器可以移动。该窗口不必是活动的或可见的，但必须已定义。

报表设计器的默认尺寸可能比该窗口大。在这种情况下，报表设计器仍采用该窗口的特性。报表设计器的左上角和窗口的左上角坐标相同，但其尺寸超出窗口的边界。

IN [WINDOW] WindowName2

指定打开报表设计器的父窗口。报表设计器不采用父窗口的特性，也不能移到父窗口之外。如果父窗口移动，报表设计器也随之移动。

要访问报表设计器，必须先用 DEFINE WINDOW 定义父窗口，且父窗口必须是可见的。

IN SCREEN

在报表设计器放入父窗口之后，指定在 Visual FoxPro 主窗口中打开报表设计器。可通过包含 IN WINDOW 子句，把报表设计器放到父窗口中。

说明

发出不带任何附加参数的 CREATE REPORT 命令将打开报表设计器，同时为新报表指定名称“报表 1”。退出报表设计器时，可用不同的名称存储报表。

CREATE REPORT 也可用来生成快速报表而不用打开报表设计器。有关详细内容，请参阅 CREATE REPORT - 快速报表命令。

有关报表和标签的其他内容，请参阅帮助中“用户指南”第七章“设计报表和标签”。

请参阅

ASCIICOLS, _ASCIIROWS, CREATE REPORT – Quick Report, MODIFY REPORT, REPORT

CREATE REPORT – Quick Report 命令

以编程方式创建报表。

语法

```
CREATE REPORT FileName1 | ? FROM FileName2
  [FORM | COLUMN] [FIELDS FieldList] [ALIAS]
  [NOOVERWRITE] [WIDTH nColumns]
```

参数描述

FileName1

指定报表的文件名。如果没有为文件指定扩展名，Visual FoxPro 自动指定 .FRX 为扩展名。

?

显示“创建”对话框，提示您为正创建的报表命名。

FROM FileName2

指定用来创建报表的表的名称。不用打开该表。

FORM

指定在细节带区中按照由上向下排列字段及字段名的方式创建报表。

COLUMN

指定在细节带区中按照由左向右横跨页面排列字段的方式创建报表。字段名放在页标头带区中。如果省略 FORM 和 COLUMN，报表默认为 COLUMN 格式。

FIELDS FieldList

指定出现在报表中的表字段。在 *FieldList* 中用逗号隔开字段。

ALIAS

指定在报表字段名前添加表别名。

NOOVERWRITE

指定不能改写已存在的报表。如果由 *FileName1* 指定的报表名已存在，则不能创建该报表。

WIDTH nColumns

以列的形式指定报表页的宽度。

说明

这种形式的 CREATE REPORT，可以不打开报表设计器就能创建一个快速报表。这种创建报表的方式类似于从“报表”菜单中选择“快速报表”。

我们已经在前面的主题中讨论过 CREATE REPORT 命令的其他形式，打开“报表设计器”可以让您交互地创建报表。

有关报表和标签的其他内容，请参阅帮助中的“用户指南”第七章“设计报表和标签”。

请参阅

CREATE REPORT, MODIFY REPORT, REPORT

CREATE SCREEN –快速屏幕命令

包含此命令是为了提供向后兼容性。请使用表单设计器。

CREATE SCREEN 命令

打开表单设计器。包含此命令是为了提供向后兼容性。请使用 CREATE FORM **命令。**

返回总目录

CURVAL（）函数
Curvature 属性
Custom 对象
Database 属性
DataEnvironment 对象
DataEnvironment 属性
DataObject 对象
DataSession 属性
DataSessionID 属性
DataToClip 方法
DATE（）函数
DateFormat 属性
DateMark 属性
DATETIME（）函数
DAY（）函数
DBC（）函数

CREATE SQL VIEW 命令

显示视图设计器来创建 SQL 视图。

语法

```
CREATE SQL VIEW [ViewName ] [REMOTE]
  [CONNECTION ConnectionName [SHARE]
  | CONNECTION DataSourceName]
  [AS SQLSELECTStatement]
```

参数描述

ViewName

指定要创建的视图的名称。

REMOTE

指定创建使用远程表的远程视图。如省略 REMOTE，则用本地表创建视图。

CONNECTION ConnectionName [SHARE]

打开视图时，指定建立一个命名连接。如果包含 SHARE 子句，并且共享连接可用，Visual FoxPro 将使用共享连接。如果共享连接不可用，则在打开视图时创建一个唯一连接，其他视图不能使用该连接。

CONNECTION DataSourceName

指定一个已存在的、建立了连接的数据源。

AS SQLSELECTStatement

指定视图定义。*SQLSELECTStatement* 必须是一个合法的 SQL SELECT 语句，而且不必用引号括起来。对于本地视图，通过用数据库名和惊叹号(!)作为视图或表名的前缀，可以指定数据库（不是当前数据库）中的表和视图。例如，下面的命令创建了 SQL 视图 mysqlview。该视图选择了 customer 数据库的 orders 表中的所有字段：

```
CREATE SQL VIEW mysqlview AS SELECT * FROM customer!orders
```

有关 SQL SELECT 语句的其他内容，请参阅 SELECT - SQL。

可通过创建一个参数化视图，而不用为每个记录子集都创建一个单独的视图，来限制视图的范围。参数化视图利用 WHERE 子句，通过提供一个值作参数，把记录限制在那些符合条件的记录上。例如，可以创建一个 SQL 视图，根据您提供的国家名称下载有关该国的记录。

所提供的参数被当作 Visual FoxPro 表达式来计算。如果计算失败，Visual FoxPro 提示参数值。例如，假定 Testdata 数据库的 customer 表放在一个远程服务器上，下面的示例创建了一个参数化远程视图，它把视图限制在某个国家的顾客上，该国国名由参数 ?cCountry 的值提供：

```
OPEN DATABASE testdata
CREATE SQL VIEW customer_remote_view CONNECTION remote_01 ;
    AS SELECT * FROM customer WHERE customer.country = ?cCountry
```

提示 如果参数是一个表达式，在括号中将参数表达式包围起来。这样就可以将整个表达式作为参数的一部分进行赋值。

有关带参数视图的其他内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第八章“创建视图”。

说明

在创建 SQL 视图之前，必须独占地打开数据库，可用包含 EXCLUSIVE 关键字的 OPEN DATABASE 命令独占地打开一个已存在的数据库，或用 CREATE DATABASE 来创建一个新的数据库。

示例

下面的示例打开 testdata 数据库。用 CREATE SQL VIEW 创建一个本地 SQL 视图 myview，这个视图利用 SELECT-SQL 语句，从 customer 表中选择了所有的记录。然后显示视图设计器，允许修改 SQL 视图。最后关闭视图设计器，删除 SQL 视图。

```
CLOSE DATABASES
* Open testdata database
OPEN DATABASE (HOME(2) + 'data\testdata')

* Create view with initial select criteria from customer table
CREATE SQL VIEW myview AS SELECT * FROM testdata!customer;
    WHERE country="Mexico"

* Activate View Designer to modify or run query
MODIFY VIEW myview  &&Activates View Designer

* Delete view after View Designer closes
DELETE VIEW myview
```

请参阅

CREATE DATABASE, CREATE VIEW, DELETE VIEW, DISPLAY VIEWS,

LIST VIEWS, MODIFY VIEW, OPEN DATABASE, RENAME VIEW, SELECT-SQL, USE

CREATE TABLE-SQL 命令

创建一个含有指定字段的表。

语法

```
CREATE TABLE | DBF TableName1 [NAME LongTableName] [FREE]
  (FieldName1 FieldType [(nFieldWidth [, nPrecision])]
      [NULL | NOT NULL]
      [CHECK lExpression1 [ERROR cMessageText1]]
      [DEFAULT eExpression1]
      [PRIMARY KEY | UNIQUE]
      [REFERENCES TableName2 [TAG TagName1]]
      [NOCPTRANS]
  [, FieldName2 ...]
      [, PRIMARY KEY eExpression2 TAG TagName2
      |, UNIQUE eExpression3 TAG TagName3]
      [, FOREIGN KEY eExpression4 TAG TagName4 [NODUP]
```

REFERENCES TableName3 [TAG TagName5]]
[, CHECK lExpression2 [ERROR cMessageText2]])
| FROM ARRAY ArrayName

参数描述

TableName1

指定要创建的表的名称。TABLE **和** DBF **选项作用相同。**

NAME LongTableName

指定表的长名。因为长表名存储在数据库中，只有在打开数据库时才能指定长表名。

长名最多可包括 128 **个字符，在数据库中可用来代替短名。**

FREE

指定所创建的表不添加到数据库中。如果没有打开数据库则不需要 FREE**。**

(FieldName1 FieldType [(nFieldWidth [, nPrecision])]

分别指定字段名、字段类型、字段宽度和字段精度（小数位数）。

一个表最多可以包含 255 **个字段。如果有一个以上的字段允许** Null **值，此限制将减至** 254 **个字段。**

FieldType **是指定字段数据类型的单个字母。有些字段数据类型要求指定** *nFieldWidth* **或** *nPrecision* **或两者都要指定。**

下表列出了 *FieldType* **的值及是否需要指定** *nFieldWidth* **和** *nPrecision*。

Field Type	NField Width	nPrecision	说明
C	n	–	宽度为 *n* 的字符字段
D	–	–	日期型
T	–	–	日期时间型
N	n	d	宽度为 *n*、有 *d* 位小数的数值型字段
F	n	d	宽度为 *n*、有 *d* 位小数的浮点数值型字段
I	–	–	整型
B	–	d	双精度型
Y	–	–	货币型
L	–	–	逻辑值
M	–	–	备注型
G	–	–	通用型

nFieldWidth 和 *nPrecision* 不适用于 D、T、I、Y、L、M、G 和 P 类型。对于 N、F 或 B 类型，若不包含 *nPrecision*，则 *nPrecision* 默认为零（没有小数位）。

NULL

在字段中允许 null 值。如果有一个以上的字段允许 Null 值，表可以包含的最多字段数将减少一个，从 255 减至 254。

NOT NULL

在字段中不允许 null 值。

如果省略了 NULL 或 NOT NULL，SET NULL 的当前设置将决定在字段中是否允许 null 值。但是，如果省略 NULL 和 NOT NULL 而包含 PRIMARY KEY 或 UNIQUE 子句，SET NULL 的当前设置无效，字段默认为 NOT NULL。

CHECK lExpression1

指定字段的有效性规则。*lExpression1* 可以是用户自定义函数。请注意，当追加空记录时，就要检查有效性规则。如果有效性规则不允许在追加记录中有空字段值，则产生错误。

ERROR cMessageText1

指定当字段规则产生错误时，Visual FoxPro 显示的错误信息。只有当数据在浏览窗口或编辑窗口中作了修改时，才显示信息。

DEFAULT eExpression1

指定字段的默认值。*eExpression1* 的数据类型必须和字段的数据类型相同。

PRIMARY KEY

将此字段作为主索引。主索引标识名和字段名相同。

UNIQUE

将此字段作为一个候选索引。候选索引标识名和字段名相同。有关候选索引的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第七章“处理表”中的“表的索引”。

注意 候选索引（通过在 CREATE TABLE 或 ALTER TABLE-SQL 中包含 UNIQUE 选项来创建）和用 INDEX 命令的 UNIQUE 选项创建的索引不同。用 INDEX 命令的 UNIQUE 选项创建的索引允许重复索引关键

字，候选索引不允许重复索引关键字。有关 UNIQUE 选项的其他内容，请参阅 INDEX。

在主索引或候选索引字段中不允许 null 值和重复记录。但是，如果为支持 null 值的字段创建主索引或候选索引，Visual FoxPro 将不会产生错误。但是，如试图把 null 值或重复值输入到用作主索引或候选索引的字段中时，Visual FoxPro 将产生错误。

REFERENCES TableName2 [TAG TagName1]

指定与之建立永久关系的父表。如果省略 TAG *TagName1*，则使用父表的主索引关键字建立关系。如果父表没有主索引，则 Visual FoxPro 产生错误。

可包含 TAG *TagName1* 来与父表建立一个基于现有索引标识的关系，索引标识名最多可包含 10 个字符。

父表不能是自由表。

NOCPTRANS

防止字符字段和备注字段转换到另一个代码页。如果要将表转换到其他代码页，则指定了 NOCPTRANS 的字段不转换。只能为字符字段和备注字段指定 NOCPTRANS。

以下示例创建了一个名为 MYTABLE 的表，该表中包含了两个字符字段和两个备注字段。第二个字符字段 CHAR2 和第二个备注字段 MEMO2 包含 NOCPTRANS 关键字以防止翻译。

```
CREATE TABLE mytable (char1 C(10), char2 C(10) NOCPTRANS,;
    memo1 M, memo2 M NOCPTRANS)
```

PRIMARY KEY eExpression2 TAG TagName2

指定要创建的主索引。*eExpression2* 指定表中的任一个字段或字段组合。TAG *TagName2* 指定要创建的主索引标识的名称。索引标识名最多可包含 10 个字符。

因为表只能有一个主索引，如已经创建了一个主索引字段，则命令中不能包含本子句。如果在 CREATE TABLE 中包含多个 PRIMARY KEY 子句，Visual FoxPro 将产生错误。

UNIQUE eExpression3 TAG TagName3

创建侯选索引。*eExpression3* 指定表中的任一字段或字段组合。但是，如果已经用一个 PRIMARY KEY 选项创建了一个主索引，则不能包含指定为主索引的字段。TAG *TagName3* 为要创建的候选索引标识指定了标识名。索引标识名最多可包含 10 个字符。

一个表可以有多个侯选索引。

FOREIGN KEY eExpression4 TAG TagName4 [NODUP]

创建一个外部索引（非主索引），并建立和父表的关系。*eExpression4* 指定外部索引关键字表达式，而 *TagName4* 为要创建的外部索引关键字标识指定名称。索引标识名最多可包含 10 个字符。包含 NODUP 来创建一个候选外部索引。

可以为表创建多个外部索引，但外部索引表达式必须指定表中的不同字段。

REFERENCES TableName3 [TAG TagName5]

指定与之建立永久关系的父表。可包含 TAG *TagName5* 来与父表建立一个基于索引标识的关系。索引标识名最多可包含 10 个字符。如果省略 TAG

TagName5，则默认用父表的主索引关键字建立关系。

CHECK eExpression2 [ERROR cMessageText2]

指定表的有效性规则。ERROR *cMessageText2* 指定当有效性规则执行时，Visual FoxPro 显示的错误信息。只有当数据在浏览窗口或编辑窗口中作了修改时，才显示该信息。

FROM ARRAY ArrayName

指定一个已存在的数组名称，数组中包含表的每个字段的名称、类型、精度以及宽度。数组的内容可用 AFIELDS() 函数来定义。

说明

新表在最低的可用工作区中打开，并可通过它的别名来访问。不管 SET EXCLUSIVE 的当前设置如何，新表都以独占方式打开。

如果打开了数据库，并且没有包含 FREE 子句，则新表添加到数据库中。不能创建和数据库中的表同名的新表。

如果在创建新表时没有打开数据库，且包含了 NAME、CHECK、DEFAULT、FOREIGN KEY、PRIMARY KEY 或 REFERENCES 子句，则将产生错误。

需要注意的是 CREATE TABLE 命令的语法使用逗号分隔一定的 CREATE TABLE 选项。同时 NULL、NOT NULL、CHECK、DEFAULT、PRIMARY KEY 和 UNIQUE 子句必须将其放在包含列定义的括号中。

示例

下面的示例创建了一个新的数据库 Mydata1。用 CREATE TABLE 创建了两个表 Salesman 和 Customer，它们自动添加到数据库中。第二个 CREATE TABLE 命令中的 FOREIGN KEY 和 REFERENCE 子句创建了 Salesman 和 Customer 表间的一个永久的

一对多关系，然后用 MODIFY DATABASE 来显示两表间的关系。第三个 DEFAULT clauses in the third CREATE TABLE 命令中的 DEFAULT 子句用于建立默认值，CHECK 和 ERROR 子句为输入特定字段内容建立商务规则。使用 MODIFY DATABASE 命令显示三个表的关系。

```
CLOSE DATABASES
CLEAR

*在当前目录或文件夹中创建“我的数据”的数据库
CREATE DATABASE mydata1

*用主关键词创建一个销售人员表
CREATE TABLE salesman ;
    (SalesID c(6) PRIMARY KEY, ;
    SaleName C(20))

*创建一个客户表并与销售人员表建立相关关系
CREATE TABLE customer ;
    (SalesID c(6), ;
    CustId i PRIMARY KEY, ;
    CustName c(20) UNIQUE,  ;
    SalesBranch c(3), ;
    FOREIGN KEY SalesId TAG SalesId REFERENCES salesman)

*根据客户表的主关键词建立一个与之相关的定单表
*使用关键词或一些商业术语如拖欠或支票付款等
CREATE TABLE orders ;
    (OrderId i PRIMARY KEY, ;
        CustId i REFERENCES customer TAG CustId, ;
        OrderAmt y(4), ;
        OrderQty i ;
```

```
    DEFAULT 10 ;
    CHECK (OrderQty > 9) ;
    ERROR "Order Quantity must be at least 10", ;
        DiscPercent n(6,2) NULL ;
    DEFAULT .NULL., ;
    CHECK (OrderAmt > 0) ERROR "Order Amount Must be > 0" )
```

*显示新数据库、新表和它们之间的关系
MODIFY DATABASE

*删除这些范例文件
SET SAFETY OFF && To suppress verification message
CLOSE DATABASES &&删除前关闭数据库
DELETE DATABASE mydata1 DELETETABLES

请参阅

AFIELDS（）, ALTER TABLE-SQL, CREATE, CREATE QUERY, INSERT-SQL, MODIFY QUERY, MODIFY STRUCTURE, OPEN DATABASE, SELECT-SQL, SETNOCPTRANS , SQL 命令概览

CREATE TRIGGER 命令

创建表的触发器。

语法

CREATE TRIGGER ON TableName
FOR DELETE | INSERT | UPDATE AS lExpression

参数描述

TableName

在当前数据库中指定要创建触发器的表。

FOR DELETE | INSERT | UPDATE

指定 Visual FoxPro **创建的触发器类型。**

如果所指定类型的触发器已经存在并且 SET SAFETY **是** ON，Visual FoxPro **会提示您是否要改写已存在的触发器。如果** SET SAFETY **是** OFF，**则已存在的触发器被自动改写。**

AS lExpression

指定触发器激发时，要计算的逻辑表达式。 *lExpression* **可以是返回逻辑值的用户自定义函数或内部存储过程。可用** MODIFY PROCEDURE **创建表的内部存储过程。**

用户自定义函数或内部存储过程可通过 AERROR() **决定激发触发器的表的名称以及触发器的类型。**

如果 *lExpression* **计算结果为“真”**(.T.)，**则执行激发触发器的命令或事件。**

如果 *lExpression* **的计算结果为“假”**(.F.)，**不执行激发触发器的命令或事件。如果** ON ERROR **过程有效，则执行** ON ERROR **过程来代替激发触发器的命令或事件。如果** ON ERROR **过程无效，则不执行激发触发器的命令或事件，同时** Visual FoxPro **产生错误信息**

说明

利用 CREATE TRIGGER 来俘获引起表中记录被删除、添加或更改的事件。只有数据库表能创建删除、插入或更新触发器。可用 CREATE DATABASE 来创建数据库，用 ADD TABLE 把表添加到数据库中。CREATE TRIGGER 要求独占使用数据库。若要以独占方式打开数据库，可在 OPEN DATABASE 中包含 EXCLUSIVE。
下面的列表描述了激发删除、插入或更新触发器的事件。

删除触发器

- 发出 DELETE 命令。
- 发出 ZAP 命令不会删除触发器。
- 注意执行 ZAP 命令不会引起“删除”触发器发生。

插入触发器

- 发出 APPEND FROM 命令
- 发出 APPEND FROM ARRAY 命令
- 发出 APPEND BLANK 命令
- 从浏览窗口或编辑窗口的“表”菜单追加记录
- 发出 IMPORT 命令
- 发出 INSERT-SQL 命令
- 发出 RECALL 命令
- 从浏览窗口或编辑窗口的“表”菜单中清除记录的删除标记

更新触发器

- 发出 GATHER 命令
- 发出 REPLACE 命令
- 发出 REPLACE FROM ARRAY 命令
- 发出 UPDATE-SQL 命令
- 引起修改记录的其他事件，如表单更改字段中的内容时。

下面的规则适用于用 CREATE TRIGGER 创建的触发器：

- 不能对有触发器的表使用 INSERT，但可使用 INSERT-SQL
- 发出 PACK 不会激发任何触发器
- 发出 ZAP 不会激发删除触发器
- 如更新具有删除标记的记录，不会激发触发器
- 触发器可能不会立即被激发，这取决于当前缓冲模式：

如果表缓冲起作用，当发出 TABLEUPDATE（）时，激发更新触发器，表中的每个缓冲记录都被更新。

示例

下面的示例创建了更新触发器。防止大于 50 的值输入到 customer 表的 maxordamt 字段中。第一个 REPLACE 命令执行时，产生错误信息，因为 maxordamt 字段的值大于 50。第二个 REPLACE 命令没有产生错误，因为 maxordamt 的值小于或等于 50。

```
CLOSE DATABASES
```

```
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表

* Set trigger on maxordamt field to fail with values <= 50
CREATE TRIGGER ON customer FOR UPDATE AS maxordamt <= 50

ON ERROR  && 还原系统错误处理程序

WAIT WINDOW "Press a key to test trigger with value of 60"+CHR(13);
 +"When you get the error message, press Ignore."
REPLACE maxordamt WITH 60    && 显示错误信息
? maxordamt

WAIT WINDOW "Press a key to test with value of 50."
REPLACE maxordamt WITH 50    && 值是可接受的
? maxordamt
DELETE TRIGGER ON customer FOR UPDATE  && Remove the trigger
```

请参阅

ADD TABLE, AERROR（）, CREATE DATABASE, DELETE TRIGGER, DISPLAY DATABASE, LIST DATABASE, OPEN DATABASE

CREATE VIEW 命令

从 Visual FoxPro 环境创建视图文件。

语法

CREATE VIEW FileName

参数描述

FileName

指定要创建的视图文件的名称。

说明

CREATE VIEW 生成一个新的包含有 Visual FoxPro 环境信息的视图。SET VIEW 可恢复保存在视图文件中的环境。用 CREATE VIEW 创建的视图文件被指定一个 .VUE 扩展名。

存储在视图文件中的信息包含：

- 当前在所有工作区中打开的表、索引、替补文件以及格式文件
- 包括在 SET FIELDS 列表中的所有字段
- 在打开的表间建立的所有关系
- 对打开表有效的所有过滤器
- DEFAULT 和 PATH 设置

- 过程文件设置
- 当前的帮助文件
- 当前的资源文件
- SET SKIP 的状态
- 状态栏的状态（ON 或 OFF）

视图文件在程序中或是调试时都是有用的。恢复环境时，只需执行一条 SET VIEW TO *FileName* 命令。在调试时，可把环境设置存储在视图文件中，进行调试之后，再恢复环境来继续执行程序。

请参阅

CREATE SQL VIEW, SET VIEW

CREATEBINARY（）函数

将 Visual FoxPro 字符型数据转换为二进制字符型数据。您在向 ActiveX 控件或 OLE 对象传递数据时需要使用二进制字符型数据。

语法

CREATEBINARY(cExpression)

返值类型

字符型

参数描述

cExpression

希望转换的字符表达式。

说明

在 Visual FoxPro 中，二进制数据可以包含在字符串中。但 ActiveX 控件和 OLE 对象中的字符串 (OLE VT_BSTR 数据类型) 不能包含二进制数据。ActiveX 控件和 OLE 对象通过 VT_UI1 类型数组与 Visual FoxPro 传递二进制形式的数据。

ActiveX 控件和 OLE 对象通过 VT_UI1 类型的数组将二进制数据传递给 Visual FoxPro 后。Visual FoxPro 自动将它转换为 Visual FoxPro 字符串的形式，但 Visual FoxPro 内部会把由 ActiveX 控件和 OLE 对象传递来的字符串标记为二进制字符串。当需要把数据传回 ActiveX 控件和 OLE 对象时，Visual FoxPro 再自动将它们转换为 VT_UI1 类型的数组。

如果您需要把一个 Visaul FoxPro 字符串传递给 ActiveX 控件和 OLE 对象，您必须首先使用 CRETEBINARY（）函数将它转换为二进制字符串形式。CRETEBINARY（）的缩写不能少于 7 个字符。

有关 ActiveX 控件和 automation 对象，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十六章“添加 OLE”。

请参阅

OLE Bound 控件，OLE 容器控件

CREATE OBJECT（）函数

从类定义或可用 OLE 应用程序中创建对象。

语法

CREATEOBJECT(ClassName [, eParameter1 , eParameter2, ...])

返值类型

Object

参数描述

ClassName

指定用以创建新对象的类或 OLE 对象。Visual FoxPro 按下面的顺序搜索类或 OLE 对象：

1. Visual FoxPro 基类。
2. 在内存中按加载顺序排列的用户自定义类定义。
3. 当前程序中的类。
4. 用 SET CLASSLIB 打开的 .VCX 类库中的类。
5. 用 SET PROCEDURE 打开的过程文件中的类。
6. 在 Visual FoxPro 程序执行链中的类。

7. OLE 注册表（如果 SET OLEOBJECT 为 ON）。

可对 *ClassName* 使用下面语法创建 OLE 对象：

ApplicationName.Class

例如，若要创建 Microsoft Excel 工作表（支持 OLE 自动化），可用下面的语法：

```
x = CREATEOBJECT('Excel.Sheet')
```

当运行这个代码时，启动 Microsoft Excel（如果还没有运行），并创建一个新的工作表。

一个类库有一个别名。若要用别名来指定类库中的对象，可包含类库别名，后接一个句点号和对象名。

注意，ClassName 不能是 Visual FoxPro OLE 容器控件的基类。

ePameter1, eParameter2, ...

这些可选参数用来把值传给类的 Init 事件过程。当发出 CREATEOBJECT（）时，Init 被执行，并进行对象初始化工作。

说明

可用 CREATEOBJECT（）从类定义或支持 OLE 自动化的应用程序中创建对象，并将对象引用赋给变量或数组元素。

由用户自定义类创建对象之前，用户自定义类必须先用 DEFINE CLASS 来创建，或者必须是用 SET CLASSLIB 打开的 .VCX 可视类库中的类。

可用 = 或 STORE 来将对象引用赋给变量或数组元素。如果释放指定给变量或数组元

素的对象，则变量或数组元素包含 null 值。可用 RELEASE 从内存中移去变量或数组元素。

示例

下面的示例利用 DEFINE CLASS 和 CREATEOBJECT（），由 Visual FoxPro 表单基类创建了两个自定义类 FormChild 和 FormGrandChild。ACLASS（）用来创建一个存放类名的数组 gaNewarray，然后显示类名。

```
CLEAR

* Verify current class library 设置
cCurClassLib=SET("CLASSLIB")
IF LEN(ALLTRIM(cCurClassLib))=0
    cCurClassLib="None"
ENDIF
WAIT WINDOW "Current class library is: " + cCurClassLib + CHR(13);
    + "Press any key to continue..."

frmMyForm = CREATEOBJECT("FormGrandChild")

* 建立一个数组
FOR nCount = 1 TO ACLASS(gaNewarray, frmMyForm)
    ? gaNewarray(nCount)  && 显示类名
ENDFOR

RELEASE frmMyForm

*在基类表格中创建孩子表格
DEFINE CLASS FormChild AS FORM
ENDDEFINE

*在用户定义的基类表格中创建孙子表格
```

```
DEFINE CLASS FormGrandChild AS FormChild
ENDDEFINE
```

请参阅

_BROWSER, COMCLASSINFO（）, CREATEOBJECTEX（）, DEFINE CLASS, GETOBJECT（）, RELEASE, SET CLASSLIB, SET OLEOBJECT

CREATEOBJECTEX（）函数

在一个远程计算机上创建一个已注册 COM 对象（例如一个 Visual FoxPro Automation 服务程序）的一个实例。

语法

CREATEOBJECTEX(cCLSID | cPROGID, cComputerName)

返值类型

Object

参数描述

cCLSID | cPROGID

为了实例化 COM 对象，为它指定 CLSID (Class Identifier) 或 PROGID (Programmatic Identifier)。如果包含 CLSID，则必须在使用 cComputerName 指定的远程服务器上注册该 COM 对象。如果包含 PROGID，则必须在本地

计算机上和使用 cComputerName 指定的远程服务器上注册该 COM 对象。试图使用一个 PROGID，而不首先在本地计算机上注册它，会造成 OLE 错误 0x800401f3，既“无效的类串”。

对于在本地计算机上创建的 Visual FoxPro 自动服务程序，可以使用该服务程序对象的 CLSID 和 PROGID 属性确定本地的 CLSID 和 PROGID 值。

cComputerName 指定远程计算机的，将在该计算机上实例化 COM 对象。

如果 cComputerName 是空字符串，则在本地计算机上或重定向的计算机（在注册表中指定）上实例化该 COM 对象。使用 Clireg32.exe 或 Racmgr32.exe 可以通过注册表将本地计算机重定向为另一台计算机。

CComputerName 支持 UNC (Universal Naming Convention) 名称（例如 \ 和“myserver,”）和 DNS (Domain Name System) 名称（例如 m 、“www.microsoft.com”和“207.68.137.56”）。

说明

如果成功实例化了，CREATEOBJECTEX（）返回对该 COM 对象的一个对象引用。

CREATEOBJECTEX（）不能用来实例化 Visual FoxPro 类，例如表单，可使用 CREATEOBJECT（）实例化 Visual FoxPro 类。注意，只能将 CREATEOBJECTEX（）最小缩短为 13 个字符，以区别于 CREATEOBJECT（）函数。

CREATEOBJECTEX（）需要 Windows NT 4.0 或以后版本，或者安装了的 DCOM95 的 Windows 95。

有关使用 Visual FoxPro 创建 Automation 服务程序的其他内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十六章“添加 OLE”。

请参阅

CLSID 属性, CREATEOBJECT（）函数, ProgID 属性, Server 对象

CREATEOFFLINE（）函数

由已存在的视图创建一个游离视图。

语法

CREATEOFFLINE(ViewName [, cPath])

返值类型

逻辑值

参数描述

ViewName

指定一个已存在的视图。若要从一个已存在的视图创建游离视图，这个视图所在的数据库必须处于打开状态。

cPath

若要将游离视图的结果保存在一个 .DBF 文件中，*cPath* 为游离视图指定保存的路径和文件名。

说明

如果游离视图创建成功，CREATEOFFLINE（）返回真(.T.)；否则返回假(.F.)。
若要打开游离视图，请使用USE命令。游离视图打开后，您可以添加或修改其中的记录。但不能使用CREAT TRIGGER，INSERT，PACK，或ZAP命令。您如果希望将您的修改更新到数据库中，可以使用包含ONLINE子句的USE命令打开游离视图，再使用TABLEUPDATE（）函数。
只有使用USE及ONLINE子句打开游离视图时，才能使用来自服务器的数据刷新游离视图的内容。
使用DROPOFFLNE（）可以将游离视图重新连接到数据库中并放弃所有的修改。

请参阅

DROPOFFLINE（），USE

CTOD（）函数

把字符表达式转换成日期表达式。

语法

CTOD(cExpression)

返值类型

日期型

参数描述

cExpression
指定的字符表达式，CTOD（）把它转换成日期型的值。

说明

CTOD（）是将字符转换成日期的函数，从字符表达式返回一个日期型的值。有关模糊日期的详细内容，请参阅《Microsoft Visual FoxPro 6.0中文版程序员指南》第三十三章“对编程的改进”中的“对2000年日期的支持”。

请参阅

DATE（），SET STRICTDATE

CTOBIN（）函数

将二进制字符型表示转换为整数。

语法

CTOBIN(cExpression)

返值类型

数值型

参数描述

cExpression character
要转换的二进制字符型数据。

说明

CTOBIN（）将由 BINTOC（）函数得到的二进制字符型数据还原为整数。

请参阅

BINTOC（）函数

CTOT（）函数

从字符表达式返回一个日期时间值。

语法

CTOT(cCharacterExpression)

返值类型

日期时间型

参数描述

cCharacterExpression

指定要返回日期时间值的字符表达式。

说明

注意 CTOT（）不能创建模糊日期时间值，并且当 SET STRICTDATE 设置为 2 时会产生编译错误。使用 DATETIME（）代替创建非模糊日期值，有关模糊日期的详细内

容，请参阅《Microsoft Visual FoxPro 6.0中文版程序员指南》第三十三章“对编程的改进”中的“对 2000 年日期的支持”。

请参阅

DATETIME（），SET STRICTDATE

CURDIR（）函数

返回当前目录。

语法

CURDIR([cExpression])

返值类型

字符型

参数描述

cExpression

CURDIR（）返回当前目录所在的驱动器或卷。如省略 *cExpression*，则指定当前默认驱动器或卷；如果 *cExpression* 中指定的驱动器或卷不存在，则返回空字符串。

说明

CURDIR（）以字符串形式返回指定驱动器的当前 MS-DOS 目录。

示例

以下示例存储了当前可用目录，设置的默认值为启动 Visual FoxPro 的目录。显示新目录，将默认值返回到原始目录，并显示原始目录。

```
CLEAR
? 'Current directory: ', CURDIR（）
gcOldDir = SET('DEFAULT') + SYS(2003)
SET DEFAULT TO (HOME（）)
? 'Visual FoxPro directory: ', CURDIR（）
SET DEFAULT TO (gcOldDir)
? 'Current directory: ', CURDIR（）
```

请参阅

FULLPATH（），HOME（）

_CUROBJ 系统变量

存储当前选定的控制编号。包含此变量是为了提供向后兼容性。对新表单可用 ActiveControl 属性代替。

CurrentControl 属性

指定列对象中的哪一个控件用来显示活动单元格的值。设计时可用，运行时可读写。

语法

Column.CurrentControl[= cName]

参数描述

cName

指定 Column 对象中显示和接受活动单元数据的控件名。

说明

默认控件是一个 Name 属性值为 Text1 的文本框。可用 AddObject 方法添加其他控件。

如果列的 Sparse 属性设为“真”（.T.），则只有列中活动单元使用 CurrentControl 属性中指定的对象，其他单元用文本框显示数据；如果 Sparse 属性设为“假”（.F.），则列中所有单元都用 CurrentControl 属性显示数据。

注意 可用 AddObject 方法把任意数目的控件添加到 Column 对象中，也可以设置 CurrentControl 属性，使任何时候都只有一个控件。

应用于

列

请参阅

AddObject 方法, Bound 属性, RemoveObject 方法, Sparse 属性

CurrentX, CurrentY 属性

指定供下一个绘图方法使用的横坐标 (X) 和纵坐标 (Y)。设计时不可用，运行时只读只写。

语法

Object.CurrentX[= nXCoord]
Object.CurrentY[= nYCoord]

参数描述

nXCoord
以表单 ScaleMode 属性指定的度量单位来确定表单的横坐标。

nYCoord
以表单 ScaleMode 属性指定的度量单位来确定表单的纵坐标。

说明

坐标从对象的左上角开始计算。对象左上角 CurrentX 是 0，CurrentY 是 0，用 foxels 或 ScaleMode 属性定义的当前度量单位来表示。

当使用下面的图形方法时，CurrentX 和 CurrentY 设置改变如下：

方法	CurrentX，CurrentY 设为
Box	由后两个参数指定的框的终点。
Circle	对象的中心。
Cls	0, 0.
Line	线的终点。
Print	下一个打印位置。
Pset	所画的点。

应用于

Form, _SCREEN

请参阅

Box 方法, Circle 方法, Cls 方法, DrawMode 属性, DrawStyle 属性, Left 属性, Line 方法, PSet 方法, ScaleMode 属性, SYS(1270), Top 属性

Cursor 对象

把表或视图添加到表单、表单集或报表的数据环境时创建。

语法

Cursor

说明

在运行表单、表单集或报表时，可借助临时表对象指定或确定表或视图的属性。

请注意：在运行时，设置临时表对象属性将产生错误（Filter 或 Order 属性除外，它们可以在运行时设置）。要使新的属性设置生效，必须调用数据环境的 CloseTables 和 OpenTables 方法。此外，如果运行时改变拥有一对多关系的临时表的 Order 属性，将破坏这个一对多关系。

有关表单和表单集数据环境的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第九章“创建表单”中的“设置数据环境”。

有关报表数据环境的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十二章“添加查询和报表”。

属性

Alias	Application	BufferModeOverride
CursorSource	Comment	Database
Exclusive	Filter	Name
NoDataOnLoad	Order	ReadOnly
Tag		

事件

Destroy	Error	Init

方法

AddProperty　ReadExpression　ReadMethod
ResetToDefault　WriteExpression

请参阅

DataEnvironment **对象**, Relation **对象**

CURSORGETPROP（）**函数**

返回 Visual FoxPro **表或临时表的当前属性设置。**

语法

CURSORGETPROP(cProperty [, nWorkArea | cTableAlias])

返值类型

字符型、数值型或逻辑型

参数描述

cProperty

指定要返回的属性设置。

下表列出了可以返回的属性设置、返回值的数据类型以及返回值说明。

cProperty	类型	描述
BatchUpdateCount*	N	送到缓冲表后端的更新语句的数目。1 是默认值。调整该值可大大提高更新的性能。 可读写。
Buffering	N	1-（默认值）行缓冲和表缓冲关闭。 2-打开保守式行缓冲。 3-打开开放式行缓冲。 4-打开保守式表缓冲。 5-打开开放式表缓冲。 可读写。
CompareMemo	L	若用于检测更新冲突的 WHERE 子句中包含备注字段或通用字段，返回 .T.（默认）；否则，返回 .F.。本属性只用于视图。 可读写。
ConnectHandle	N	连接句柄。只有在数据源中包含远程表时此属性才有效。 只读。
ConnectName	C	在创建临时表时使用的命名连接。 只读。

Database	C	包含表或视图的数据库名称。如表是自由表，包含空字符串。 只读。
FetchAsNeeded	L	若只有在需要时才将数据调入，返回 .T.；否则，返回 .F.（默认）。 可读写。
FetchMemo*	L	如果视图结果中含有备注字段，则为“真”(.T.)；否则为“假”(.F.)（默认值）。 可读写。
FetchSize*	N	从远程表结果集合中逐步提取的行数。默认是100行。设置 FetchSize 为 -1 将检索整个结果集（但受 MaxRecords 设置限制）。可读写。
KeyFieldList	C	逗号分隔的临时表的主字段列表。 可读写。
MaxRecords*	N	当返回结果集时所提取的最大行数。默认值是-1（返回所有行）。 0 值指定运行视图，但不提取结果。 可读写。

Prepared	L	如果 SQL 语句已准备好，可以执行 REQUERY（），返回 .T.；否则，返回 .F.（默认）。 REQUERY（）用于从一个 SQL 视图中重新检索数据。有关 SQL 语句的准备，请参阅 SQLPREPARE（）。本属性只用于视图。 可读写。
SendUpdates	L	如果 SQL 更新查询被执行以更新远程表，则为“真”(.T.)；否则为“假”(.F.)（默认值）。 可读写。
SourceName	C	包含数据库中的 SQL 视图或表的长名，或者自由表的文件路径和表名。 可读写。
SourceType	N	1-指定数据源是本地 SQL 视图。 2-指定数据源是远程 SQL 视图。 3-指定数据源是表。 只读。
SQL	C	Th 创建临时表时执行的 SQL 语句。 只读。
Tables	C	逗号分隔的远程表名的列表。 可读写。

UpdatableFieldList	C	逗号分隔的指定给临时表的远程字段名和本地字段名的列表。 可用该选项将临时表的无效 Visual FoxPro 字段名指定为有效的 Visual FoxPro 字段名。 可读写。

续表

UpdateNameList	C	逗号分隔的视图中字段的列表，该列表包含有来自本地和远程表的字段。 可读写。
UpdateType	N	1-指定用新数据更新旧数据（默认值）。 2-指定通过删除旧数据，插入新数据来进行更新。 可读写。
UseMemoSize*	N	使得结果返回到备注字段的列的最小宽度（按字节）。例如，如果一个列的结果的宽度比 UseMemoSize 的值大，列结果存储在备注字段中。UseMemoSize 可以从 1 到 255，默认值是 255。 可读写。

续表

WhereType	N	更新远程表的 WHERE 子句。WhereType 可以指定为下列值： 1 或 DB_KEY（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句仅包含由 KeyFieldList 属性指定的主关键字段。 2 或 DB_KEYANDUPDATABLE（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句包含由 keyFieldList 属性指定的主关键字段及任何可更新的字段。 3 或 DB_KEYANDMODIFIED（源于 FOXPRO.H）（默认值）。对远程表进行更新的 WHERE 子句包含由 keyFieldList 属性指定的主关键字段及任何已修改字段。 4 或 DB_KEYANDTIMESTAMP（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句包含由 keyFieldList 属性指定的主关键字段和时间戳。 可读写。

*此属性的返回值只对远程视图才是重要的，如果您对本地视图取得此属性，CURSORGETPROP 函数返回默认值。

下表中包含可以返回设置的属性、返回值的数据类型及返回值的说明的列表。

cProperty	Type	说明
BatchUpdate Count*	N	送到缓冲表后端的更新语句的数目。1 是默认值。调整该值可大大提高更新的性能。 可读写。
Buffering	N	1-（默认值）行缓冲和表缓冲关闭。 2-打开保守式行缓冲。 3-打开开放式行缓冲。 4-打开保守式表缓冲。 5-打开开放式表缓冲。 可读写。
CompareMemo	L	若用于检测更新冲突的 WHERE 子句中包含备注字段或通用字段，返回 .T.（默认）；否则，返回 .F.。本属性只用于视图。 可读写。
ConnectHandle	N	连接句柄。只有在数据源中包含远程表时此属性才有效。 只读。
ConnectName	C	在创建临时表时使用的命名连接。只有在数据源中包含远程表时此属性才有效。 只读。

续表

Database	C	包含表或视图的数据库名称。如表是自由表，包含空字符串。 只读。
FetchAsNeeded	L	若只有在需要时才将数据调入，返回 .T.；否则，返回 .F.（默认）。 可读写。
FetchMemo*	L	如果视图结果中含有备注字段，则为“真”(.T.)；否则为“假”(.F.)（默认值）。 可读写。
FetchSize*	N	从远程表结果集合中逐步提取的行数。默认是100行。设置 FetchSize 为 -1 将检索整个结果集（但受 MaxRecords 设置限制）。 可读写。
KeyFieldList	C	逗号分隔的临时表的主字段列表。 可读写。
MaxRecords*	N	当返回结果集时所提取的最大行数。默认值是-1（返回所有行）。 0值指定运行视图，但不提取结果。 可读写。

续表

Prepared	L	如果 SQL 语句已准备好，可以执行 REQUERY（），返回 .T.；否则，返回 .F.（默认）。 REQUERY（）用于从一个 SQL 视图中重新检索数据。有关 SQL 语句的准备，请参阅 SQLPREPARE（）。本属性只用于视图。 可读写。
SendUpdates	L	如果 SQL 更新查询被执行以更新远程表，则为“真”(.T.)；否则为“假”(.F.)（默认值）。 可读写
SourceName	C	包含数据库中的 SQL 视图或表的长名，或者自由表的文件路径和表名。 只读。
SourceType	N	1-指定数据源是本地 SQL 视图。 2-指定数据源是远程 SQL 视图。 3-指定数据源是表。 只读。
SQL	C	创建临时表时执行的 SQL 语句。 只读。
列表	C	逗号分隔的远程表名的列表。 可读写。

续表

Updatable FieldList	C	逗号分隔的指定给临时表的远程字段名和本地字段名的列表。 可用该选项将临时表的无效 Visual FoxPro 字段名指定为有效的 Visual FoxPro 字段名。 可读写。
UpdateNameList	C	逗号分隔的视图中字段的列表，该列表包含有来自本地和远程表的字段。 可读写。
UpdateType	N	1-指定用新数据更新旧数据（默认值）。 2-指定通过删除旧数据，插入新数据来进行更新。 可读写。
UseMemoSize*	N	使得结果返回到备注字段的列的最小宽度（按字节）。例如，如果一个列的结果的宽度比 UseMemoSize 的值大，列结果存储在备注字段中。UseMemoSize 可以从 1 到 255，默认值是 255。 可读写。

续表

WhereType	N	更新远程表的 WHERE 子句。WhereType 可以指定为下列值： 1 或 DB_KEY（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句仅包含由 KeyFieldList 属性指定的主关键字段。 2 或 DB_KEYANDUPDATABLE（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句包含由 keyFieldList 属性指定的主关键字段及任何可更新的字段。 3 或 DB_KEYANDMODIFIED（源于 FOXPRO.H）（默认值）。对远程表进行更新的 WHERE 子句包含由 keyFieldList 属性指定的主关键字段及任何已修改字段。 4 或 DB_KEYANDTIMESTAMP（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句包含由 keyFieldList 属性指定的主关键字段和时间戳。 可读写。

* 此属性的返回值只对远程视图才是重要的，如果您对本地视图取得此属性，CURSORGETPROP 函数返回默认值。

nWorkArea

从返回的属性设置可以指定表的工作区域。如果指定工作区域为 0，CURSORGETPROP（）函数返回环境设置。

cTableAlias

从返回的属性设置可以指定工作表 的别名。

说明

可用 CURSORSETPROP（）设置 Visual FoxPro 表或临时表的具体属性。有关每个属性及它们的设置，请参阅 CURSORSETPROP（）。

如果发出不带可选 *cTableAlias* 或 *nWorkArea* 参数的 CURSORGETPROP（）命令，则在当前选定工作区中打开的表或临时表的当前属性设置被返回。

示例

下面示例打开了 testdata 数据库中的 customer 表。然后用 CURSORGETPROP（）来显示表的缓冲方式以及包含该表的数据库名称。

```
CLOSE DATABASES
CLEAR

OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer    && 打开客户表

? CURSORGETPROP("Buffering") && 返回缓冲模式
? CURSORGETPROP("Database")  && 返回数据库的名字
```

请参阅

CURSORSETPROP（），SQLGETPROP（），SQLSETPROP（）

CURSORSETPROP（）函数

设置 Visual FoxPro 表或临时表的属性。

语法

CURSORSETPROP(cProperty [, eExpression] [, cTableAlias | nWorkArea])

返值类型

逻辑值

参数描述

cProperty

指定要设置的表或临时表属性。请注意，对 Visual FoxPro 表只能指定 Buffering 属性。

eExpression

为 *cProperty* 属性指定值。如省略 *eExpression*，属性设为默认值。

下表列出可为 *cProperty* 指定的属性以及能指定的 *eExpression* 值。

属性	eExpression 值
BatchUpdateCount*	发送到缓冲表后端的更新语句的数目。1 是默认值，调整该值可以大大提高更新性能。

续表

Buffering	1-设置行缓冲和表缓冲关闭。在 FoxPro 早期版本中，锁定记录和写数据操作是等价的（默认值）。 2-打开保守式行缓冲。 3-打开开放式行缓冲。 4-打开保守式表缓冲。 5-打开开放式表缓冲。 除了方式 1 之外，SET MULTILOCKS 都必须是 ON。
Compare Memo	.T-检测更新冲突的 WHERE 子句中包含备注字段或通用字段 .F-检测更新冲突的 WHERE 子句中不包含备注字段或通用字段
FetchAs Needed	.T. –按照要求选取记录。 .F.-对于远程视图，选取记录的数目决定于 MaxRecord 的属性；对于当地视图，选取所有记录。
FetchMemo*	. .T.-视图结果中包含备注字段。 .F.-视图结果中不包含备注字段。
FetchSize*	从远程结果集合中逐步提取的行数，默认值是 100 行。 FetchSize 设为 -1 将检索全部结果集（但受 MaxRecords 设置限制）。
KeyFieldList	逗号分隔的临时表主字段的列表。 无默认值。必须包含字段名的列表。
MaxRecords*	当结果集返回时所提取的最大行数。默认值是-1（返回所有行）。0 值指定执行视图，但不提取结果。

Prepared	.T.-准备 SQL 语句，以执行 REQUERY（）。REQUERY（）用于从一个 SQL 视图中重新检索数据。有关 SQL 语句的准备，请参阅 SQLPREPARE（）。本属性只用于视图。 默认为 .F.
SendUpdates	.T.-执行 SQL 更新查询来更新远程表。 .F.-不执行 SQL 更新查询。
Tables	逗号分隔的远程表名称的列表。 无默认值。必须包含表名称的列表。
Updatable FieldList	逗号分隔的指定给临时表的远程字段名和本地字段名的列表。 可用该选项来为包含无效 Visual FoxPro 字段名的临时表指定有效的 Visual FoxPro 字段名。
Update NameList	逗号分隔视图中字段的列表。此列表可以包含来自本地和远程表的字段。您必须包含将要进行更新的字段的列表。
UpdateType	1-用新数据更新旧数据（默认值）。 2-通过删除旧数据，插入新数据来进行更新。
UseMemoSize *	使得结果返回到备注字段的列的最小宽度（按字节计）。例如，如果列结果的宽度比 UseMemoSize 值大，列结果存储在备注字段中。UseMemoSize 可从 1 到 255，默认值是 255。

WhereType	更新远程表的 WHERE 子句。WhereType 可以设为下列值： 1 或 DB_KDY（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句仅包含由 keyFieldList 属性指定的主关键字段。 2 或 DB_KEYANDUPDATABLE（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句包含由 KeyFieldList 属性指定的主关键字段及任何可更新字段。 3 或 DB_KEYANDMODIFIED（源于 FOXPRO.H）（默认值）。对远程表进行更新的 WHERE 子句包含由 KeyFieldList 属性指定的主关键字段及其他已修改的字段。 4 或 DB_KEYANDTIMESTAMP（源于 FOXPRO.H）。对远程表进行更新的 WHERE 子句包含由 KeyFieldList 属性指定的主关键字段和时间戳。

*此属性主要用于远程视图，设置它对本地视图没有影响。但是，您可以为将要升迁的本地视图预先设置此属性。

cTableAlias

指定要设置属性的表或临时表的别名。

nWorkArea

指定要设置属性的表或临时表的工作区。如指定 *nWorkArea* 为 0，CURSORSETPROP() 为后面所有的表或临时表设定环境设置。

说明

如果 Visual FoxPro 成功地设置指定的属性，则 CURSORSETPROP() 返回“真”（.T.）；如果不能设置指定的属性，Visual FoxPro 将产生错误。

CURSORSETPROP() 缓冲属性的设置决定 Visual FoxPro 如何执行记录锁定和更新缓

冲。有关记录锁定和更新缓冲的详细内容，请参阅“帮助”中的缓冲访问数据主题。

CURSORSETPROP() 的 WhereType 属性设置决定了在远程表上如何执行更新。

可用 CURSORSETPROP() 改写临时表的 FetchSize SQLSETPROP() 属性，默认情况下该属性从临时表的连接句柄继承而来。

对于 Visual FoxPro 表和为表创建的临时表，可用 CURSORGETPROP() 返回其当前属性设置。

如执行不带可选 *cTableAlias* 或 *nWorkArea* 参数的 CURSORSETPROP() 命令，可以为当前选定工作区中打开的表或临时表指定属性设置。

示例

下面示例显示了如何用 CURSORSETPROP() 激活开放式表缓冲，表缓冲要求 MULTILOCKS 设为 ON。首先打开 testdata 数据库的 customer 表，然后用 CURSORSETPROP() 设置缓冲方式为开放式表缓冲（5）。如果成功地激活开放式表缓冲，则显示“真”（.T.），否则显示“假”（.F.）。

```
CLOSE DATABASES
CLEAR

SET MULTILOCKS ON
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer    && 打开客户表

* Set buffering mode and store logical result
lSuccess=CURSORSETPROP("Buffering", 5, "customer")
IF lSuccess = .T.
    =MESSAGEBOX("Operation successful!",0,"Operation Status")
ELSE
    =MESSAGEBOX("Operation NOT successful!",0,"Operation Status")
```

```
ENDIF
```

请参阅

CURSORGETPROP（）,SET MULTILOCKS,SQLGETPROP（）,SQLSETPROP（）

CursorSource 属性

指定与临时表对象相关的表或视图名。设计时只读，运行时可读写。

语法

DataEnvironment.Cursor.CursorSource[= cText]

参数描述

cText

对于视图，指定数据库中视图的名称；对于数据库中的表，指定长表名；对于自由表，指定到自由表的绝对路径。

应用于

临时表对象

请参阅

Database 属性

CURVAL（）函数

从磁盘上的表或远程数据源中直接返回字段值。

语法

CURVAL(cExpression [, cTableAlias | nWorkArea])

返值类型

字符型、货币型、日期型、日期时间型、双精度型、浮点型、逻辑型、数值型或备注型

参数描述

cExpression

指定一个表达式，CURVAL（）从表或远程数据源返回该表达式的值。*cExpression* 通常是表或远程数据源的一个字段或包含一组字段的表达式。

cTableAlias

指定表的别名，可从磁盘上的表或远程数据源返回该表的字段值。

nWorkArea

指定表的工作区，可从磁盘上的表或远程数据源返回该表的字段值。

说明

CURVAL（）和 OLDVAL（）返回的字段值可进行比较，以决定编辑字段时，网络中的其他用户是否改变了字段值。当激活开放式行缓冲或表缓冲时，CURVAL（）和

OLDVAL() 必须返回不同的值。可用 CURSORSETPROP() 激活开放式行缓冲或表缓冲。.

注意 如果您在多用户环境中处理视图，CURVAL() 的返回值可能不会取得最新的数据除非您事先调用了 REFRESH() 函数。视图返回的数据放在缓冲区内，CURVAL() 函数从缓冲区中读取值。但是，如果其他用户已经更改了视图所基于的表中的数据，除非调用了 REFRESH() 函数，否则不会更新缓冲区中的数据。

CURVAL() 返回当前记录的字段值，返回值的数据类型由 *cExpression* 决定。

如果执行不带可选 *cTableAlias* 或 *nWorkArea* 参数的 CURVAL() 命令，则返回当前选定工作区中打开的表或临时表的对应值。

示例

此示例创建了一个名为 mytable 的自由表并向 cDigit 字段插入了一个值“One”。SET MULTILOCKS ON 和 CURSORSETPROP() 启用开放式表缓冲。然后再将“Two”插入 cDigit 字段，使用 CURVAL() 及 OLDVAL() 函数显示原始的 cDigit 值。使用 TABLEUPDATE() 函数将更改提交到表中，再次使用 CURVAL() 和 OLDVAL() 显示新的 cDigit 值。注意这是一个单用户示例，CURVAL() 和 OLDVAL() 返回相同的值。

```
CLOSE DATABASES
CLEAR

CREATE TABLE mytable FREE (cDigit C(10))
* 存储原始数值
INSERT INTO mytable (cDigit) VALUES ("One")
SET MULTILOCKS ON      && 认可表缓冲模式
= CURSORSETPROP("Buffering",5)  && 打开表缓冲模式
```

```
REPLACE cDigit WITH "Two"   && 新值

? "Current value: " + CURVAL("cDigit", "mytable")
? "Old value: " + OLDVAL("cDigit", "mytable")
= TABLEUPDATE(.T.)      && 认可对表的修改或更新
? "Table changes committed"
? "New current value: " + CURVAL("cDigit", "mytable")
? "New old value: " + OLDVAL("cDigit", "mytable")
```

请参阅

GETFLDSTATE（）,OLDVAL（）,TABLEREVERT（）,TABLEUPDATE（）,CURSORSETPROP（）

Curvature 属性

指定形状控件的弯角曲率。设计和运行时可用。

语法

Shape.Curvature[= nCurve]

参数描述

nCurve

下表列出了 Curvature 属性的设置：

设置	说明
0	指定无曲率。创建方角。
1 到 98	指定圆角。数字越大，曲率越大。
99	指定最大曲率。创建一个圆或椭圆。

示例

下面示例显示了如何使用形状控件和 Curvature 属性，在表单上显示圆、椭圆或矩形。创建表单，在表单上放置一组选项按钮和命令按钮。当单击一个选项按钮时，表单上显示相应的形状。每个形状的 Height、Width 和 Curvature 属性决定了要创建的形状类型（圆、椭圆或正方形）。Curvature 属性为 99 时，则创建圆和椭圆。

```
frmMyForm = CREATEOBJECT('Form') && 创建表单
frmMyForm.Closable = .F. && 使“控件”菜单框失效

frmMyForm.AddObject('cmdCommand1','cmdMyCmndBtn') && 添加命令按钮
frmMyForm.AddObject('opgOptionGroup1','opgMyOptGrp') && 添加选项组
frmMyForm.AddObject('shpCircle1','shpMyCircle') && 添加圆
frmMyForm.AddObject('shpEllipse1','shpMyEllipse') && 添加椭圆
frmMyForm.AddObject('shpSquare','shpMySquare') && 添加正方形

frmMyForm.cmdCommand1.Visible =.T. && "Quit" Command button visible

frmMyForm.opgOptionGroup1.Buttons(1).Caption = "\<Circle"
frmMyForm.opgOptionGroup1.Buttons(2).Caption = "\<Ellipse"
frmMyForm.opgOptionGroup1.Buttons(3).Caption = "\<Square"

frmMyForm.opgOptionGroup1.SetAll("Width", 100) && 设置选项组宽度
```

```
frmMyForm.opgOptionGroup1.Visible = .T. && 选项组可视
frmMyForm.opgOptionGroup1.Click  && 显示圆

frmMyForm.SHOW  && 显示表单
READ EVENTS  && 开始事件处理

DEFINE CLASS opgMyOptGrp AS OptionGroup  && 创建选项组
    ButtonCount = 3  && 三个选项按钮
    Top = 10
    Left = 10
    Height = 75
    Width = 100

    PROCEDURE Click
        ThisForm.shpCircle1.Visible = .F.  && 隐藏圆
        ThisForm.shpEllipse1.Visible = .F.  && 隐藏椭圆
        ThisForm.shpSquare.Visible = .F.  && 隐藏正方形

        DO CASE
            CASE ThisForm.opgOptionGroup1.Value = 1
                ThisForm.shpCircle1.Visible = .T. && 显示圆
            CASE ThisForm.opgOptionGroup1.Value = 2
                ThisForm.shpEllipse1.Visible = .T.  && 显示椭圆
            CASE ThisForm.opgOptionGroup1.Value = 3
                ThisForm.shpSquare.Visible = .T.  && 显示正方形
        ENDCASE
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn AS CommandButton  && 创建命令按钮
    Caption = '\<Quit'  && 命令按钮的标题
    Cancel = .T.  && 默认的取消命令按钮（Esc）
    Left = 125  && 命令按钮列
```

```
    Top = 210  && 命令按钮行
    Height = 25  && 命令按钮的高度

    PROCEDURE Click
        CLEAR EVENTS  && Stop event processing, close Form
ENDDEFINE

DEFINE CLASS shpMyCircle AS SHAPE  && 创建圆
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 99
    BackColor = RGB(255,0,0)  && 红色
ENDDEFINE


DEFINE CLASS shpMyEllipse AS SHAPE  && 创建椭圆
    Top = 35
    Left = 200
    Width = 100
    Height = 50
    Curvature = 99
    BackColor = RGB(0,128,0)  && 绿色
ENDDEFINE

DEFINE CLASS shpMySquare AS SHAPE  && 创建正方形
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 0
```

```
    BackColor = RGB(0,0,255) && 蓝色
ENDDEFINE
```

应用于

形状

请参阅

Shape 控件

Custom 对象

创建用户自定义对象。

语法

Custom

说明

可使用 DEFINE CLASS **或类设计器来创建用户自定义类，执行** CREATE CLASS **将显示类设计器。**

用户自定义类具有属性、事件和方法，但没有可视的外观。所有用于定义其他类型的类的通用规则也适用于定义自定义类。

有关创建非可视化类的自定义对象的其他内容，请参阅《Microsoft Visual FoxPro 6.0 **中文版程序员指南》第三章“面向对象的程序设计”。**

属 性

Application	BaseClass	Class
ClassLibrary	Comment	ControlCount
Controls	Height	HelpContextID
Left	Name	Object
Parent	ParentClass	Picture
Tag	Top	WhatsThisHelpID
Width		

事 件

Destroy	Error	Init

方 法

AddObject	AddProperty	NewObject
ReadExpression	ReadMethod	RemoveObject
ResetToDefault	SaveAsClass	ShowWhatsThis
WriteExpression	WriteMethod	

请 参 阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

Database 属性

指定数据库的路径，该数据库包含与临时表对象相关联的表或视图。设计时只读，运行时可读写。

语法

DataEnvironment.Cursor.Database[= cPath]

参数描述

cPath

Specifies the full path to the database (.dbc) file.

说明

当用 CURSORSETPROP( ) 访问对象时，Database 属性在运行时只读。如果临时表对象建立在自由表基础上，则该属性返回一个空字符串 (" ")。

应用于

临时表

请参阅

CursorSource 属性

DataEnvironment 对象

当创建表单、表单集或报表时，创建该对象。

语法

DataEnvironment

说明

数据环境对象是临时表对象或关系对象的容器对象，而这些临时表对象或关系对象与表单、表单集或报表相关联。

注意：在运行时设置数据环境对象属性将产生错误。要使新的属性设置生效，必须为数据环境对象调用 CloseTables 和 OpenTables 方法。

有关表单和表单集数据环境的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第九章“创建表单”。

有关报表数据环境的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十二章“添加查询和报表”。

属性

Application AutoCloseTables AutoOpenTables
Comment InitialSelectedAlias Name
OpenViews Tag

事件

AfterCloseTables BeforeOpenTables Destroy
Error Init

方法

AddObject AddProperty CloseTables
OpenTables ReadExpression ReadMethod
RemoveObject ResetToDefault SaveAsClass
WriteExpression

请参阅

Cursor 对象, Relation 对象

DataEnvironment 属性

引用对象的数据环境。设计和运行时只读。

语法

Object.DataEnvironment

应用于

表单，表单集

请参阅

DataEnvironment **对象**

DataObject 对象

是从一个 OLE 拖动源传输到一个 OLE 放落目标的数据的容器。只在设计时可用。

语法

oDataObject

说明

DataObject 对象从一个 OLE 拖动源传输到一个 OLE 放落目标的数据的容器，并且只在一个 OLE 拖放事件过程中存在。不能通过编程创建 DataObject 对象，并且在 OLE 拖放操作之后引用它也是无效的。DataObject 是作为 oDataObject 参数在 OLEDragDrop、OLEDragOver、OLESetData 和 OLEStartDrag 事件中传送的。

DataObject 可以按不同的格式保存多组值。可使用 GetFormat 和 GetData 方法确定 DataObject 中的数据格式和数据。使用 SetFormat、SetData 或 ClearData 方法将数据格式和数据添加到 DataObject 中，或者从 DataObject 中清除所有的数据格式和数据。

方法

ClearData　GetData　GetFormat
SetData　SetFormat

请参阅

OLE Drag-and-Drop 概览, OLEDragDrop 事件, OLEDragOver 事件, OLESetData 事件, OLEStartDrag 事件

DataSession 属性

指定一个表单、表单集或工具栏能否在自身的数据工作期中运行以及能否具有独立的数据环境。设计时可用，运行时只读。

语法

Object.DataSession[= nSession]

参数描述

nSession

下表列出了 DataSession 属性的设置：

设置	说明
1	（默认值）默认数据工作期。
2	私有数据工作期。为表单、表单集或工具栏的每个实例创建一个新的数据工作期。

应用于

表单，表单集，_SCREEN，工具栏

请参阅

DataSessionID 属性

DataSessionID 属性

返回数据工作期 ID 标识号，用来标识表单集、表单或工具栏的私有数据工作期。设计时只读，运行时可读写。

如果表单、表单集或工具栏的 DataSession 属性设置为 1（默认数据工作期），则返回默认数据工作期的 ID 标识号。

语法

Object.DataSessionID

说明

只有当表单集、表单或工具栏的 DataSession 属性设置为 2（私有数据工作期）时，才可使用该属性。

可以使用 SET DATASESSION 和 DataSessionID 属性改变数据工作期。

当设置 DataSessionID 属性时，将影响容器对象和它包含的所有对象的数据工作期。DataSessionID 属性设置不影响未包含的对象，也不影响用 CREATOBJECT() 创建的对象。

改变 DataSessionID 属性设置后，将增加改变后数据工作期的引用次数，而减少改变前数据工作期的引用次数。然而，如果数据工作期是通过将 DataSession 属性设置为 2（私有数据工作期）创建的，则改变属性设置不会释放最初的数据工作期，在这种情况下，必须通过释放对象来释放初始数据工作期。

有关多个数据工作期的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十七章“共享访问程序设计”。

警告 对于包含数据绑定型控件的对象，当改变其中的 DataSessionID 属性设置时，将丢失原始数据源。通常对不包含数据绑定型控件的对象使用 DataSessionID。

应用于

表单，表单集，_SCREEN，工具栏

请参阅

CREATEOBJECT()，DataSession 属性，SET DATASESSION

DataToClip 方法

将一组记录作为文本复制到剪贴板上。

语法

ApplicationObject.DataToClip([nWorkArea | cTableAlias] [, nRecords] [, nClipFormat])

参数描述

nWorkArea

指定表的工作区编号，从该表中将记录复制到剪贴板上。如果省略 cTableAlias 和 nWorkArea，则从当前工作区的打开表中复制记录。

cTableAlias

指定表的别名，从该表中将记录复制到剪贴板上。

nRecords

指定要复制到剪贴板上的记录数。如果 nRecords 比表中的记录多，则将所有记录复制到剪贴板上。如果省略 nRecords 和 nClipFormat，则将当前记录和所有其他记录复制到剪贴板上。

nClipFormat

指定如何分隔字段。NClipFormat 的设置有：

nClipFormat	说明
1	（默认值）使用空格分隔字段。
3	使用制表符分隔字段。

如果省略 nClipFormat，也使用空格分隔字段。

说明

字段名位于文本的第一行，后面是一行一个记录。

应用于

Application 对象，_VFP 系统变量

请参阅

Eval 方法, SetVar 方法

DATE（）函数

返回由操作系统控制的当前系统日期，或创建一个与 2000 年兼容的日期值。

语法

DATE([nYear, nMonth, nDay])

返值类型

日期型

参数描述

nYear

指定返回的年份在与2000年兼容的日期值之间。nYear值的范围从100到9999。

nMonth

指定返回的月份在与2000年兼容的日期值之间。nMonth值的范围从1到12。

nDay

指定返回的日期在与2000年兼容的日期值之间。nDay值的范围从1到31。

说明

如果执行不带可选参数的DATE（）函数将返回当前的系统日期。包含可选参数将返回与2000年兼容的日期值。有关创建与2000年兼容的日期值的详细内容，请参阅《Microsoft Visual FoxPro 6.0中文版程序员指南》第三十三章“对编程的改进”中的“对2000年日期的支持”。

Visual FoxFro命令或函数都不能直接改变系统日期。

示例

下面例子显示带有世纪和不带世纪的当前系统日期。

```
CLEAR
SET CENTURY OFF
?DATE（） &&显示不带世纪部分的日期值
SET CENTURY ON
?DATE（） &&显示带世纪部分的日期值
```

```
? DATE(1998, 02, 16) && 显示与 2000 年兼容的日期值
```

请参阅

CTOD（），DATETIME（），DTOC（），SET CENTURY，SET DATE，SET MARK TO，SYS（）**函数概览**

DateFormat 属性

指定在文本框中显示的 Date **和** DateTime **值的格式。设计和运行时可用。**

语法

Object.DateFormat[= nValue]

参数描述

nValue

取下列设置之一：

设置	说明	格式
0	（默认值）。所显示的日期格式由 SET DATE 的设置确定。	由 SET DATE 的设置确定
1	American	mm/dd/yy

续表

2	ANSI	yy.mm.dd
3	British	dd/mm/yy
4	Italian	dd-mm-yy
5	French	dd/mm/yy
6	German	dd.mm.yy
7	Japan	yy/mm/dd
8	Taiwan	yy/mm/dd
9	USA	mm-dd-yy
10	MDY	mm/dd/yy
11	DMY	dd/mm/yy
12	YMD	yy/mm/dd
13	Short	由 Windows“控制面板”确定短日期设置
14	Long	由 Windows“控制面板”确定长日期设置

说明

当一个文本框没有焦点时，其中显示的 Date 或 DateTime 值的格式可能不同。您在 Windows“控制面板”中指定的长短日期格式不对应于有效的 Visual Foxpro 格式，并且按默认格式显示。

当 Format 属性设置为 YS、YL、E 或 D 时，Format 属性覆盖 DateFormat 属性的设置。

应用于

文本框

请参阅

Century **属性**, DateMark **属性**, Hours **属性**, Seconds **属性**, SET DATE, StrictDateEntry **属性**

DateMark **属性**

指定在文本框中显示的 Date **和** DateTime **值的分隔符。设计和运行时可用。**

语法

Object.DateMark[= cDateMarkCharacter]

参数描述

cDateMarkCharacter

指定 Date **和** DateTime **值的分隔符。如果** cDateMarkCharacter **是空字符串，则分隔符由** SET MARK **的设置确定。如果** cDateMarkCharacter **是一个空格，则使用文本框的** DateFormat **属性的当前设置。**

在“属性”窗口中，使用 = **命令可以将** DateMark **属性指定为空字符串** (=“”) **或空格** (=“ ”)。

说明

如果 DateFormat 属性设置为 Short 或 Long，则忽略 DateMark 属性的设置。

应用于

文本框

请参阅

Century 属性, DateFormat 属性, Hours 属性, Seconds 属性, SET MARK TO, StrictDateEntry 属性

DATETIME() 函数

以日期时间值返回当前的日期和时间。

语法

DATETIME([nYear, nMonth, nDay [, nHours [, nMinutes [, nSeconds]]]])

返值类型

日期时间型

参数描述

nYear

指定返回的年份在与 2000 年兼容的日期时间值之间。nYear 值的范围从 100 到 9999。

nMonth
指定返回的月份在与 2000 年兼容的日期时间值之间。nMonth 值的范围从 1 到 12。

nDay
指定返回的日期在与 2000 年兼容的日期时间值之间。nDay 值的范围从 1 到 31。

nHours
指定返回的小时在与 2000 年兼容的日期时间值之间。nHours 值的范围从 0（午夜）到 23（11 P.M)。如果忽略，默认为 0。

nMinutes
指定返回的分钟在与 2000 年兼容的日期值时间之间。nMinutes 值的范围从 0 到 59。如果忽略，默认为 0。

nSeconds
指定返回的秒在与 2000 年兼容的日期值时间之间。nSeconds 值的范围从 0 到 59。如果忽略，默认为 0。

说明

如果执行不带可选参数 DATETIME() 命令则返回当前的系统日期时间值。包含可选参数将返回与 2000 年兼容的日期时间值。有关创建与 2000 年兼容的日期时间值的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第三十三章“对编程的改进”中的“对 2000 年日期的支持”。

示例

第一个示例将新的年份存储在一个名为 tNewyear 的变量中，并且将当前的日期时

间存储在名为 tToday 的变量中。然后显示当前的日期时间与下一年此时之间的秒数。

第二个示例使用 DATETIME() 函数创建与 2000 年兼容的日期时间值。

```
tNewyear = DATETIME(YEAR(DATE() ) + 1, 1, 1)  && 下一个新年
tToday = DATETIME()
nSecondstonewyear = tNewyear - tToday
CLEAR
? "There are " + ALLTRIM (STR(nSecondstonewyear)) ;
  + " seconds to the next New Year."

CLEAR
SET CENTURY ON
SET DATE TO AMERICAN
? DATETIME(1998, 02, 16, 12, 34, 56)  && 显示数值 02/16/1998 12:34:56 PM
```

请参阅

CTOT(), DATE(), DTOT() HOUR(), SEC(), SECONDS(), SET SECONDS, SETSYSFORMATS, TIME(), TTOC(), TTOD()

DAY() 函数

以数值型返回给定日期表达式是某月中的第几天。

语法

DAY(dExpression | tExpression)

返值类型

数值型

参数描述

dExpression

指定的一个日期，DAY（）返回该日期是某月中的第几天。*dExpression* 可以是一个日期字符串，或是一个日期型变量、数组元素或字段。

tExpression

指定的一个日期时间，DAY（）返回该日期时间是某月中的第几天。*tExpression* 可以是一个日期时间字符串，或是一个日期时间型变量、数组元素或字段。

说明

DAY（）返回 1 到 31 之间的数。

示例

```
STORE {^1998-03-05} TO gdBDate

CLEAR
? CDOW(gdBDate) && 显示 星期四
? DAY(gdBDate) && 显示 5
? 'That date is ', CMONTH(gdBDate), STR(DAY(gdBDate),2)
```

请参阅

CDOW(), DOW(), SET FDOW, SET FWEEK, SYS（） 函数概览

DBC（）函数

返回当前数据库的名称和路径。

语法

DBC（）

返值类型

字符型

说明

如果没有当前数据库，则 DBC（）返回空字符串。

可使用 SET DATABASE 指定当前数据库。

示例

下面例子打开 testdata 数据库，并用 DBC（）显示该数据库的信息。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata') && 打开 DBC.

CLEAR
? DBC（）    && 显示数据库的路径和名称
```

请参阅

ADD TABLE, CLOSE DATABASES, CREATE DATABASE, DISPLAY

TABLES, OPEN DATABASE, REMOVE TABLE, SET DATABASE

返回总目录

DBF（）函数

返回指定工作区中打开的表名，或根据表别名返回表名。

语法

DBF([cTableAlias | nWorkArea])

返值类型

字符型

参数描述

cTableAlias

指定表的别名。

nWorkArea

指定工作区编号。

如果省略 *cTableAlias* 和 *nWorkArea*，DBF（）返回当前工作区中打开的表名；如果指定的工作区中没有打开的表，DBF（）返回一个空字符串；如果表没有 *cTableAlias* 别名，则 Visual FoxPro 产生错误。

有关创建表别名的详细内容，请参阅 USE。

说明

当 SET FULLPATH 设置为 ON 时，DBF（）返回表的路径和表名；当 SET

FULLPATH 设置为 OFF 时，DBF（）返回表所在的驱动器和表名。

示例

以下示例从工作区中返回表名急其别名，并且在所有表关闭之后返回空字串。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer IN 2 ALIAS mycust
CLEAR
? DBF(2) && 显示 customer.dbf文件及其路径
? DBF('mycust') && 显示 customer.dbf文件及其路径
CLOSE DATABASES
? DBF（）    && 显示空列
```

请参阅

CDX(), FIELD() , NDX() , SET FULLPATH, USE

DBGETPROP（）函数

返回当前数据库的属性，或者返回当前数据库中字段、命名连接、表或视图的属性。

语法

DBGETPROP(cName, cType, cProperty)

返值类型

字符型，数值型，或者逻辑值

参数描述

cName

指定数据库、字段、命名连接、表或视图的名称，DBGETPROP（）函数返回有关信息。

要返回表或视图中字段的信息，可将包含该字段的表或视图的名称放在该字段名前面。例如，要返回 customer 表中 custid 字段的信息，可指定 *cName* 为：

```
customer.custid
```

cType

指定 *cName* 是否为当前数据库，或者当前数据库中的一个字段、命名连接、表或视图。下表列出了 *cType* 的允许值：

cType	说明
CONNECTION	*cName* 是当前数据库中的一个命名连接。
DATABASE	*cName* 是当前数据库。
FIELD	*cName* 是当前数据库中的一个字段。
TABLE	*cName* 是当前数据库中的一个表。
VIEW	*cName* 是当前数据库中的一个视图。

cProperty

指定属性名称，DBGETPROP（）函数返回该属性的信息。下表列出了 *cProperty* 的允许值、返回值类型以及属性说明，每个说明都包括属性的读写权限。如果属性是只读的，则它的值就不能用 DBSETPROP（）更改。有关

更改属性值的详细内容，请参阅 DBSETPROP()。

Connection 属性

cProperty	类型	说明
Asynchronous	L	连接方式。 默认值为“假”(.F.)，同步连接方式 “真”(.T.)为异步连接方式。 可读写。
BatchMode	L	批处理方式。 默认值为“真”(.T.)，以批处理方式操作连接。 可读写。
Comment	C	连接的注释文本。 可读写。
ConnectString	C	注册连接字符串。 可读写。
Connect Timeout	N	以秒为单位的连接超时间隔。默认值为 0（无限期等待）。 可读写。
Database	C	使用带有 DATABASE 子句的 CREATE CONNECTION 命令或“连接设计器”指定服务器数据库名。 可读写。

续表

DataSource	C	ODBC.INI 文件中定义的数据源名。 可读写。
DispLogin	N	决定 ODBC 的注册对话框何时显示。DispLogin 可以是以下值： 1 或 DB_PROMPTCOMPLETE（在 FOXPRO.H 中定义的常量）。 1 是默认值。 2 或 DB_PROMPTALWAYS（在 FOXPRO.H 中定义的常量）。 3 或 DB_PROMPTNEVER（在 FOXPRO.H 中定义的常量）。 如果指定为 1 或 DB_PROMPTCOMPLETE，只有当缺少所需信息时，Visual FoxPro 才显示 ODBC 的注册对话框。 如果指定 2 或 DB_PROMPTALWAYS，那么每次都显示 ODBC 的注册对话框，并允许在连接前改变设置。 如果指定 3 或 DB_PROMPTNEVER，不显示 ODBC 注册对话框，并且当不能获得所需要的注册信息时，Visual FoxPro 产生错误信息。 可读写。

续表

DispWarnings	L	包含一个逻辑值，决定是否显示远程表、ODBC 或 Visual FoxPro 中不可俘获的警告信息。 默认值为“真”(.T.)，指定显示不可俘获的警告信息。 可读写。
IdleTimeout	N	以秒为单位的空闲超时间隔，在指定的时间间隔后，活动连接变成不活动。默认值为 0（无限期地等待）。 可读写。
PacketSize	N	连接用的网络包大小，调整这个值可以提高性能，默认值为 4096 个字节（4K）。 可读写。
PassWord	C	连接密码。 可读写。
QueryTimeout	N	以秒为单位的查询超时间隔。默认值为 0（无限期地等待）。 可读写。

续表

Transactions	N	决定连接如何管理远程表上的事务，Transactions 可为以下值： 1 or DB_TRANSAUTO (from Foxpro.h). 1 或 DB_TRANSAUTO（在 FDOXPRO.H 中定义的常量） 2 或 DB_TRANSMANUAL（在 FOXPRO.H 中定义的常量）。用 SQLCOMMIT（）和 SQLROLLBACK（）人工处理事务。 通过 SQLCOMMIT（）和 SQLROLLBACK（）函数人工处理事务处理进程。 可读写。
UserId	C	用户标识。 可读写。
WaitTime	N	在 Visual FoxPro 检查 SQL 语句是否完成以前，以毫秒为单位计算所消耗的时间总和。默认值为 100 毫秒。 可读写。

Database 属性

cProperty	类型	说明
Comment	C	数据库的注释文本。 可读写。
Version	N	数据库版本号。 只读。

表字段属性

cProperty	类型	说明
Caption	C	字段标题。 可读写。
Comment	C	字段的注释文本。 可读写。
DefaultValue	C	字段默认值。 只读。
DisplayClass	C	字段所映象的类的名称。 可读写。
DisplayClass Library	C	字段所映象的类所在的类库（包括详细的路径）。 可读写。
Format	C	字段显示格式。有关格式设置的列表，请参阅稍后部分的语言参考“Format 属性”。 可读写。
InputMask	C	字段输入格式。有关输入掩码设置的列表，请参阅稍后部分的语言参考“InputMask 属性”。 可读写。
RuleExpression	C	字段规则表达式。 只读。

续表

RuleText	C	字段规则错误文本。 只读。

视图字段属性

cProperty	类型	说明
Caption	C	字段标题。 可读写。
Comment	C	字段的注释文本。 可读写。
DataType	C	视图中字段的数据类型。最初设置为数据源字段的数据类型 若要用 DBSETPROP() 函数为字段指定不同的数据类型，可使用 CREATE TABLE - SQL 语法创建字段。 例如，要将 Mytable 表中 iCost 字段由整型改为长度为 4 带 2 个小数位的数值型，可使用如下命令： DBSETPROP('mytable.icost', 'field', ; 'DataType', 'N(4,2)')

续表

		也可包含 NOCPTRANS 子句禁止将字符字段和备注字段转换到不同的代码页。
		对于远程视图可读写。
		本地视图忽略这个属性。
DefaultValue	C	字段默认值。
		可读写。
KeyField	L	如果在索引关键表达式中指定了这个字段，则为“真”(.T.)；否则为“假”(.F.)。
		可读写。
RuleExpression	C	字段规则表达式。
		可读写。
RuleText	C	字段规则错误文本。
		可读写。
Updatable	L	如果可以更改字段，则为“真”(.T.)；否则为假(.F.)。
		可读写。

续表

UpdateName	C	如果可以更改字段，则为“真”(.T.)；否则为假(.F.)。 可读写。

Table 属性

cProperty	Type	说明
Comment	C	表的注释文本。 可读写。
DeleteTrigger	C	删除触发器表达式。 只读。
InsertTrigger	C	插入触发器表达式。 只读。
Path	C	表的路径。 只读。
PrimaryKey	C	主关键字的标识名。 只读。
RuleExpression	C	行规则表达式。 只读。

续表

RuleText	C	行规则错误文本。 只读。
UpdateTrigger	C	更新触发器表达式。 只读。

视图属性

cProperty	类型	说明
BatchUpdate Count	N	发送到视图后端的更新语句数目。1为默认值。调整这个值能极大提高更新的性能。 可读写。
Comment	C	视图的注释文本。 可读写。
CompareMemo	L	若用于检测更新冲突的WHERE子句中包含备注字段或通用字段，返回.T.（默认）；否则，返回.F.。 可读写。
ConnectName	C	打开视图时所用的命名连接。 只读。

续表

FetchAsNeeded	L	若只有在需要时才将数据调入，返回 .T.；否则，返回 .F.（默认）。 可读写。
FetchMemo	L	如果视图结果含有备注字段和通用字段，则为“真”(.T.)（默认值）；否则为假(.F.)。 可读写。
FetchSize	N	当可逐步读取记录时，每次从远程表中所取得的记录数。默认值为 100 条记录。如果将 FetchSize 设置为 -1，一次读取整个结果集合（只受 MaxRecords 设置的限制）。 可读写。
MaxRecords	N	返回结果集合时获取的最大行数。默认值为 -1（返回所有行）。0 表示执行视图，但不获取任何结果。 可读写。
Offline	L	若为游离视图，返回 .T.。只读。

续表

ParameterList	C	WHERE 子句参数。参数的格式为 ''ParameterName1, 'Type1'; ParameterName2, 'Type2'; ...'' ，其中 Type 是下列字符之一，用来指定参数类型： C— 字符型 D — 日期型 T — 日期时间型 N — 数值型 F — 浮点型 B — 双精度型 I — 整型 Y — 货币型 L — 逻辑型 例如，''MyParam1, 'C' '' 指定一个名为 MyParaml 的单个字符型参数。 关于创建带参数视图的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第八章“创建视图”。 可读写。

续表

Prepared	L	如果SQL语句已准备好，可以执行REQUERY()，返回.T.；否则，返回.F.（默认）。REQUERY（）用于从一个SQL视图中重新检索数据。有关SQL语句的准备，请参阅SQLPREPARE()。本属性只用于视图。 默认值为.F.假。 可读写。
RuleExpression	C	行规则表达式。 可读写。
RuleText	C	在浏览窗口或编辑窗口中编辑数据时，若出现错误，所显示的规则文本表达式。 可读写。
SendUpdates	L	如果由SQL更新查询去更新远程表，则为“真”(.T.)；否则为“假”(.F.)（默认值）。 可读写。
ShareConnection	L	如果视图能与其他连接共享其连接句柄，则为“真”(.T.)；否则为“假”(.F.)。 可读写。

续表

SourceType	N	视图源。SourceType 可以是以下值： 1. 视图使用本地表。 2. 视图使用远程表。 只读。
SQL	C	打开视图时执行的 SQL 语句。 只读。
Tables	C	一个用逗号分隔的表名列表。 可读写。
UpdateType	N	更新类型。有效值为： 1 或 DB_UPDATE（在 FOXPRO.H 中定义的常量），表示用新数据更新旧数据（默认值）。 2 或 DB_DELETEINSERT（在 FOXPRO.H 中定义的常量）。删除旧数据并插入新数据。 可读写。
UseMemoSize	N	将结果返回到备注字段的列的最小宽度（以字节为单位）。例如，如果一个列结果的宽度大于 UseMemoSize 的值，那么列结果就存放在一个备注字段中。UseMemoSize 的取值从 1 到 255，默认值为 255。 可读写。

续表

WhereType	N	WHERE 子句更新远程表，WhereType 可以为以下值： 1 或 DB_KEY（在 FOXPRO.H 中定义的常量）。用来更新远程表的 WHERE 子句仅由 KeyFieldList 属性指定的主关键字段组成。 2 或 DB_KEYANDUPDATBLE（在 FOXPRO.H 中定义的常量）。用来更新远程表的 WHERE 子句由 KeyFieldList 属性指定的主关键字段和所有可更新字段组成。 3 或 DB_KEYANDMODIFIED（在 FOXPRO.H 中定义的常量）（默认值）。更新远程表的 WHERE 子句包含 KeyFieldList 属性指定的主关键字段和其他已修改字段。 4 或 DB_KEYANDTIMESTAMP（在 FOXPRO.H 中定义的常量）。更新远程表的 WHERE 子句由 KeyFieldList 属性指定的主关键字段和比较时间戳组成。 关于“WhereType 属性”的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第八章“创建视图”。 可读写。

说明

在获得一个数据库的属性或数据库的连接、表、视图或字段的属性之前，该数据库必须打开。在获得一个表或视图的属性之前，不必执行 USE 命令。

有关可以访问的连接、数据库、字段、表及视图属性的细节，请参考下表所列主题。

有关细节	参考区域	《Microsoft Visual FoxPro 6.0 中文版程序员指南》章名
数据库属性	“显示和设置数据库属性”	第六章“创建数据库”
连接	“访问远程数据”	第八章“创建视图”
视图		第八章“创建视图”
触发器	“使用触发器”	第七章“处理表“
标题	“创建字段标题”	第七章“处理表”
默认值	“创建字段默认值”	第七章“处理表”
	“创建视图字段默认值”	第八章“创建视图”
注释	“添加字段注释”	第七章“处理表”
	“显示和设置数据库属性”	第六章“创建数据库”
主关键字	“控制重复值”	第七章“处理表”

续表

规则	“强制商业规则”	第七章“处理表”
	“设置或更改字段级或表规则”	第七章“处理表”
	“在视图字段和记录中创建规则”	第八章“创建视图”

示例

以下示例显示 customer 表主关键字的名称。然后显示 customer 表中 cust_id 字段的字段注释。如果此字段没有注释，显示一条表示没有字段注释的消息。如果要添加说明，请参阅稍后部分的语言参考“DBSETPROP()函数”的示例。

```
CLOSE DATABASES
CLEAR

OPEN DATABASE (HOME(2) + 'Data\testdata')

* display the primary key field
cResults = DBGETPROP("customer", "Table", "PrimaryKey")
=MESSAGEBOX(cResults)  &&显示 cust_id 字段

* display comments for the field 'cust_id'
cResults = DBGETPROP("customer.cust_id", "Field", "Comment")
IF LEN(ALLTRIM(cResults)) = 0
    =MESSAGEBOX("No Comment for this field." + CHR(13) + ;
     CHR (13) + "Use DBSETPROP() to add comments.")
ELSE
    =MESSAGEBOX("Cust_id field comments: " + cRESULTS)
ENDIF
```

请参阅

ADD TABLE, CREATE CONNECTION, CREATE DATABASE, CREATE SQL VIEW, CURSORGETPROP( ), CURSORSETPROP( ), DBSETPROP( ), DISPLAY DATABASE, LIST DATABASE, RENAME CONNECTION, SQLCOMMIT( ), SQLROLLBACK( )

DblClick 事件

当连续两次快速按下鼠标左按钮（主按钮）并释放时，此事件发生。

语法

PROCEDURE Object.DblClick
[LPARAMETERS nIndex]

参数描述

nIndex

如果控件在数组中，用以唯一指定控件的索引号。

说明

当从列表框或组合框中选择一个选项并按 ENTER **键时，也发生** DblClick **事件。**

如果在系统指定的双击时间间隔内不发生 DblClick **事件，那么对象认为这种操作是一个** Click **事件。因此，当向这些相关事件中添加过程时，必须确认这些事件不冲突。**

另外，不响应 DblClick 事件的控件可能会将一个双击事件确认为两个单击事件。

注意　要区分鼠标的左、右、中按钮，可使用 MouseDown 和 MouseUp 事件。

应用于

复选框，组合框，命令组，容器对象，控件对象，编辑框，表单，表格，标头，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

Click 事件, MouseDown 事件, MouseUp 事件

_DBLCLICK 系统变量

指定双击鼠标和三击鼠标的最大时间间隔。

语法

_DBLCLICK = nTicks

参数描述

nTicks

以秒为单位指定时间间隔。内部计时器以时钟的“滴答”为单位，每“滴答”一次大约 1/18 秒。如果在 _DBLCLICK 中存放一个整数或小数，Visual FoxPro 会存放一个略有不同的值，同时按“滴答声”对之四舍五入。

_DBLCLICK 的默认值为 0.5 秒，可变范围为 0.05 秒到 5.5 秒（1 到 100 次“滴答”）。

说明

DBLCLICK 的设置改写 Windows 控制面板中“鼠标”对话框的指定设置。

_ DBLCLICK 包含一个数值，它指定了一个时间间隔，Visual FoxPro 用这个时间间隔来确定连续两次鼠标击键或三次鼠标击键是否为双击或三击鼠标。_DBLCLICK 是鼠标击键的时间间隔。例如，如果 _DBLCLICK 设定为 0.5 秒，则在 1/2 秒时间内两次单击表示一个双击，在 1 秒内三次单击表示一个三击。

_DBLCLICK 的值越大，Visual FoxPro 在把两次单击解释成一次双击时，第一次与第二次单击间的等待时间就可以越长；如果 _DBLCLICK 设成非常小的值，即使快速地双击（或三击）也可能被解释成两个（或三个）单击。

请参阅

INKEY( )

DBSETPROP（） 函数

给当前数据库或当前数据库中的字段、命名连接、表或视图设置一个属性。

语法

DBSETPROP(cName, cType, cProperty, ePropertyValue)

返值类型

逻辑值

参数描述

cName

指定要设置属性的数据库、字段、命名连接、表或视图的名称。

若要给表或视图中的字段设置一个属性，应将包含该字段的表或视图的名称加在字段名前面。例如，要为 customer 表中的 custid 字段设置属性，应该将 *cName* 指定为：customer.custid

cType

指定 *cName* 是当前数据库还是当前数据库中的一个字段、命名连接、表或视图。

下表列出了可以为 *cType* 指定的值。

cType	说明
CONNECTION	*cName* 是当前数据库中的命名连接。
DATABASE	*cName* 是当前数据库。
FIELD	*cName* 是当前数据库中的字段。
TABLE	*cName* 是当前数据库的一个表。
VIEW	*cName* 是当前数据库中的一个视图。

cProperty

指定要设置的属性名。如果某个属性是只读，它的值就不能用 DBSETPROP（）函数修改。如果要设置只读属性，Visual FoxPro 会产生错误信息。

有关使用 cProperty 可以指定的属性及其数据类型的详细内容，请参阅稍前部分的语言参考“DBGETPROP（）函数”

ePropertyValue

指定 *cProperty* 的设定值，*ePropertyValue* 的数据类型必须和属性的数据类型相同。

警告：可用 DBSETPROP（）函数为某一属性设置无效值，Visual FoxPro 并不验证为属性指定的值是否有效。例如，可以用 DBSETPROP（）函数为一个字段的规则表达式设置一个无效表达式，而 Visual FoxPro 并不产生错误。

说明

使用 DBSETPROP（）函数可为当前数据库或当前数据库中的字段、命名连接、表或视图设置属性，使用 DBGETPROP（）函数则可确定当前属性值。

DBSETPROP（）函数要求以独占方式使用当前数据库。要以独占方式打开一个数据库，可在 OPEN DATABASE 中包含 EXCLUSIVE 子句。

欲了解详情，请参阅下表。

有关细节	参考区域	《Microsoft Visual FoxPro 6.0中文版程序员指南》章名
数据库属性	“显示和设置数据库属性”	第六章“创建数据库”
连接	“访问远程数据”	第八章“创建视图”
视图		第八章“创建视图”
触发器	“使用触发器”	第七章“处理表”
标题	“创建字段标题”	第七章“处理表”
默认值	“创建字段默认值”	第七章“处理表”
	“创建视图字段默认值”	第八章“创建视图”
注释	“添加字段注释”	第七章“处理表“
	“显示和设置数据库属性”	第六章“创建数据库”
主关键字	“控制重复值”	第七章“处理表“
规则	“强制商业规则”	第七章“处理表“
	“设置或更改字段级或表规则”	第七章“处理表“
	“在视图字段和记录中创建规则”	第八章“创建视图”

示例

以下示例使用 DBSETPROP( ) 函数指定 customer 表中的 cust_id 字段的字段注释。DBGETPROP( ) 用于显示注释。

```
CLOSE DATABASES
CLEAR

OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer    && 打开 customer 表

= DBSETPROP("customer.cust_id", "Field", "Comment", ;
  "Property has been set by DBSETPROP.")  && 新建字段注释
cRESULTS = DBGETPROP("customer.cust_id", "Field", "Comment")
WAIT WINDOW "Cust_id field comments: "+ cRESULTS  && 显示注释
```

请参阅

ADD TABLE, CREATE DATABASE, CURSORGETPROP( ), CURSORSETPROP( ), DBGETPROP( ), DISPLAY DATABASE, LIST DATABASE, OPEN DATABASE, SQLCOMMIT( ), SQLROLLBACK( )

DBUSED( ) 函数

当指定的数据库已打开时，返回“真”(.T.)。

语法

DBUSED(cDatabaseName)

返值类型

逻辑值

参数描述

cDatabaseName

指定数据库的名称。DBUSED（）函数返回一个逻辑值，表明该数据库是否打开。

说明

如果指定的数据库已经打开，则 DBUSED（）函数返回“真”(.T.)；否则返回“假”(.F.)。

示例

以下示例打开了 TESTDATA 数据库，然后使用 DBUSED（）函数确定数据库 TESTDATA 和名为 TEST 的数据库是否打开了。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')

CLEAR
? 'Testdata database open? '
?? DBUSED('testdata')    && 显示 .T.
? 'Test database open? '
?? DBUSED('test')    && 显示 .F.
```

请参阅

DDE 函数

在 Visual FoxPro 和其他 Microsoft Windows 应用程序之间交换数据。

说明

Visual FoxPro 可以作为服务程序向 Microsoft Windows 应用程序发送数据，也可作为客户从其他的 Microsoft Windows 应用程序接收数据。

在支持 DDE（动态数据交换）的应用程序中使用以下命名约定。

Name	说明
Service Name	当一客户要访问服务程序时，服务程序响应的服务名称。一个服务程序能支持很多服务名。
Topic Name	指定数据的主题名。对于基于文件的应用程序，主题名通常为文件名；而在其他程序中，主题名会根据程序的不同而不同。要访问服务程序，客户除了指定服务程序服务名外，还要指定主题名。
Item Name	指定服务程序可以传送给客户的数据单元名称。

要从其他应用程序请求数据，可创建一个 Visual FoxPro 程序，将 Visual FoxPro 作为一客户。下面简述一个过程，按照这个过程可以创建一个向其他应用程序请求数据的

Visual FoxPro 程序：

- 用 DDEInitiate（）函数建立一个服务程序的链接。
- 如果成功地建立了链接，则可以使用 DDERequest（）函数向服务程序请求数据，DDERequest（）函数能重复发送以请求更多的数据。
- 接收到数据后，使用 DDETerminate（）函数终止与服务程序的链接，释放系统资源。

以上函数建立了一个冷链接，当客户对应用程序间的所有通讯初始化时，存在冷链接。有关其他类型链接的讨论，请参阅 DDEAdvise()。

下面简述一个过程，此过程建立一个作为服务程序的 Visual FoxPro 应用程序：

- 利用 DDEService（）函数创建一个服务并指定服务类型。
- 利用 DDESetTopic（）函数创建服务主题，并为该主题指定一个过程。当客户请求中指定了该主题时，执行这个过程。
- 创建 DDESetTopic（）函数中指定的过程，以接收传给该过程的参数。
- 在过程中处理请求，如果合适，向客户返回这些数据。

注意　这些 DDE 函数在以下几个方面与以前的 Visual FoxPro 函数有不同约定：

- 这些函数名的前 4 个字符不能唯一确定该函数。
- 函数名必须超过 10 个字符且不能缩写。

DDE 函数	说明
DDEAbortTrans( )	结束一个异步 DDE 事务。
DDEAdvise( )	创建 DDE 中的报告链接或自动链接。

续表

DDEEnabled( )	启用或禁止 DDE 处理，或者返回 DDE 处理状态。
DDEExecute( )	使用 DDE，向另一个应用程序发送一条命令。
DDEInitiate( )	建立 Visual FoxPro 和另一个 Microsoft Windows 应用程序之间的 DDE 通道。
DDELastError( )	返回最后执行的 DDE 函数错误编号。
DDEPoke( )	DDE 会话过程中，在客户和服务程序之间发送数据。
DDERequest( )	在动态 DDE 会话中，向一个服务程序请求数据。
DDESetOption( )	更改或返回 DDE 设置。
DDESetService( )	创建、释放或修改 Visual FoxPro 中的服务名和设置。
DDESetTopic( )	DDE 会话过程中，在服务名中创建或释放一个主题名。
DDETerminate( )	关闭一个用 DDEInitiate（）函数建立的通道。

请参阅

DDEAbortTrans( ), DDEAdvise( ), DDEEnabled( ), DDEExecute( ), DDEInitiate( ), DDELastError( ), DDEPoke( ), DDERequest( ), DDESetOption( ), DDESetService( ), DDESetTopic( ), DDETerminate( )

DDEAbortTrans（）函数

结束一次异步动态数据交换（DDE）处理。

语法

DDEAbortTrans(nTransactionNumber)

返值类型

逻辑值

参数描述

nTransactionNumber

当事务发送到服务程序时，该参数指定DDEExecute（）函数、DDEPoke（）函数或DDERequest（）函数返回的事务编号。

说明

异步事务允许Visual FoxPro程序继续执行，而不必等待服务程序响应数据请求。

在服务程序响应时，除非指定执行用户自定义函数，否则DDExecute（）函数、DDEPoke（）函数和DDERequst（）函数要等待一定的服务程序响应时间，这个时间由DDESetOption（）函数指定。指定一个用户自定义函数并在这些函数中执行该函数，会创建一个异步事务。

如果服务程序请求处理完成前调用了DDEAbortTrans（）函数，则在事务中不会调用用户自定义函数。

如果成功地结束异步事务，则 DDEAbortTrans（）函数返回“真”(.T.)；如果不能结束异步事务，则返回“假”(.T.)。使用 DDELastError（）函数可确定不能结束该事务的原因。

请参阅

DDEExecute( ), DDELastError( ), DDEPoke( ), DDERequest( )

DDEAdvise（）函数

创建一个报告链接或自动链接，用来进行动态数据交换（DDE）。

语法

DDEAdvise(nChannelNumber, cItemName, cUDFName, nLinkType)

返值类型

逻辑值

参数描述

nChannelNumber

指定通道号。

cItemName

指定项名。例如，Microsoft Excel 使用行标志和列标志指出工作表的单元，项名 R1C1 指出工作表第一行、第一列的单元。

cUDFName

当建立一个报告链接或自动链接并且修改 *cItemName* 时，指定要执行的用户自定义函数。执行用户自定义函数时，按下面给定顺序给用户自定义函数传递六个参数：

参数	内容
Channel Number	服务程序的通道号
Action	ADVISE 或 TERMINATE
Item	项名。如：R1C1 表示一个 Microsoft Excel 的工作表单元
Data	新数据（自动链接）或空字符串（报告链接）
Format	数据格式。如：CF_TEXT
Advise Status	链接类型（0 = 人工，1 = 报告，2 = 自动）

用户自定义函数在它的 LPARAMETER 或 PARAMTER 语句中应该有六个参数接受从服务程序传递的值。如果建立的是报告链接，则执行用户自定义函数时，Data 参数中将传入一个空字符串，在此之后可以用 DDERequest（）函数获取这些数据；如果建立自动链接，则执行用户自定义函数，并通过 Data 参数传送数据。

当链接由服务程序更新时，Action 参数包含 ADVISE。当客户或服务程序关闭链接时，调用用户自定义函数，并且 Action 参数包含 TERMINATE。

任何用户自定义函数返回的值都被忽略。

nLinkType
可以指定以下链接类型：

nLinkType	链接类型
0	人工
1	报告
2	自动

可通过将 *nLinkType* 指定为 0 来关闭服务程序发来的通知。如果项名改变了，就不执行用户自定义函数。

说明

DDEAdvise（）函数用来为服务程序中的一个项名创建报告链接或自动链接。当用 DDEAdvise（）函数创建报告链接时，服务程序会告诉 Visual FoxPro 项名已被更新；如果创建自动链接，服务程序就通知 Visual FoxPro，告诉它项名已被更新，同时将新数据传送给 Visual FoxPro。

在创建一个链接前，必须用 DDEInitiate（）函数建立一个通向服务程序的通道。

可以用 DDEAdvise（）函数关闭服务程序的通知。

当 DDEAdvise（）函数执行成功，返回“真”(.T.)；否则返回“假”(.F.)。

示例

以下示例演示了如何建立一个名为 Sheet1 的 Microsoft Excel 工作表的 DDE 通道。使用 DDEAdvise（）建立两个工作表单元格中（R1C1 和 R1C2）数据的链接。当这两个单元格中的任意一个数据更改时，执行用户自定义函数 NEWDATA 。此用户自定义的函数检测 item 和 advise 参数以确定是哪一个参数更改了和已建立了什么类型的链接。

```
PUBLIC mchannum
mchannum = DDEInitiate('Excel', 'Sheet1')
IF mchannum != -1
    = DDEAdvise(mchannum, 'R1C1', 'newdata', 1)    && 通知链接
    = DDEAdvise(mchannum, 'R1C2', 'newdata', 2)    && 自动链接
    WAIT WINDOW 'Enter data in first two cells in Excel.'
ENDIF
PROCEDURE newdata
PARAMETERS channel, action, item, data, format, advise
IF action = 'ADVISE'
    DO CASE
        CASE item = 'R1C1'   && Notify link
            newvalue = DDERequest(channel, item)
            ? 'R1C1 notify link: ' + newvalue
        CASE item = 'R1C2'   && Automatic link
            newvalue = data
            ? 'R1C2 automatic link: ' + newvalue
    ENDCASE
ELSE
    IF action != "TERMINATE"
        = DDETerminate(mchannum)
    ENDIF
ENDIF
```

请参阅

DDEInitiate( ), DDELastError( ), DDESetOption( ), DDETerminate( )

DDEEnabled（）函数

启用或禁止动态数据交换(DDE)处理，或返回DDE处理状态。

语法

DDEEnabled([lExpression1 | nChannelNumber [, lExpression2]])

返值类型

逻辑值

参数描述

lExpression1

当为“真”(.T.)或“假”(.F.)时，全局启用或禁止DDE处理。如果成功地启用或禁止DDE，则DDEEnabled（）函数返回“真”(.T.)；否则返回“假”(.F.)。

nChannelNumber

指定通道号，DDEEnabled（）函数返回该通道的DDE处理状态。如果已经启用指定通道的DDE处理，则DDEEnabled（）函数返回“真”(.T.)；如果禁止DDE处理，则返回“假”(.F.)。

lExpression2

要为指定的通道启用 DDE 处理，应包含通道号 (*nChannelNumber*)，并将 *lExpression2* 指定为“真” (.T.)；若要为指定的通道禁止 DDE 处理，则包含通道编号 (*nChannelNumber*) 并将 *lExpression2* 指定为“假” (.F.)。

说明

使用 DDEEnabled（）可以全局启用或禁止 DDE 处理，也可以为特定的通道启用或禁止 DDE 处理。

DDEEnabled（）函数能在短时间内保护关键代码或禁止链接。当 DDE 处理禁止时，客户请求加入队列等待，直到 DDE 处理激活。

如果 DDEEnabled（）不带任何选项参数运行，则返回全局 DDE 处理状态。如果 DDE 处理已经全局启用，则 DDEEnabled（）函数返回“真” (.T.)；如果 DDE 处理已经全局禁止，则返回“假” (.F.)。

请参阅

DDEAbortTrans( ), DDEAdvise( ), DDEExecute( ), DDEInitiate( ), DDELastError( ), DDEPoke( ), DDERequest( ), DDESetOption( ), DDESetService( ), DDESetTopic( ), DDETerminate( )

DDEExecute（）函数

使用动态数据交换 (DDE) 向另一个应用程序发送命令。

语法

DDEExecute(nChannelNumber, cCommand [, cUDFName])

返值类型

逻辑值

参数描述

nChannelNumber

指定通道号。

cCommand

指定要发往另一个应用程序的命令。命令格式由接收命令的程序决定，查阅应用程序的文档可以了解正确语法。

cUDFName

允许异步命令执行请求。如果省略 *cUDFName*，客户应用程序等待的时间由 DDESetOption（）函数指定。如果用 *cUDFName* 指定一个用户自定义函数，则客户程序在发出命令执行请求后立即执行下一个命令。

当服务程序执行命令后，就执行 *cUDFName* 指定的用户自定义函数，用户自定义函数接受六个参数，它们按下列顺序给出：

参数	内容
Channel Number	服务程序的通道号
Action	XACTCOMPLETE（执行成功时） XACTFAIL（命令执行失败时）
Item	项名，例如 R1C1 表示 Microsoft Excel 工作表单元
Data	新数据 (REQUEST)、传递的数据（POKE 或 EXECUTED）
Format	数据格式，例如 CF_TEXT
Transaction Number	DDEExecute（）函数返回的事务编号

使用 DDEAbortTrans（）函数可以取消一个未完成的事务。如果事务失败，可用 DDElastError（）函数确定失败的原因。

当包含 *cUDFName* 时，DDEExecute（）函数并不返回一个逻辑值，而返回一个事务号。如果发生错误则返回 -1。

说明

DDEExecute（）函数发出的命令必须能被应用程序解释。在执行命令前，用 DDEInitiate（）函数建立一个通向服务程序的通道。

例如，Mircrosoft Excel 有一个扩展的宏命令集合，包括允许 Microsoft Excel 向 Visual FoxPro 请求数据的 DDE 命令。如果建立了通向 Microsoft Excel 的通道，就可以用 DDEExecute（）函数从 Visual FoxPro 向 Microsoft Excel 发送宏命令。

如果接收命令的应用程序成功地执行命令，则 DDEExecute（）返回“真”(.T.)；如果接收命令的应用程序没有成功地执行命令，或者包含的通道号无效，则 DDEExecute

（）返回“假”(.F.)。如果包含了可选的异步用户自定义函数 *cUDFName*，则返回一个事务号；若发生错误，则 DDEExecute（）返回 -1。

示例

以下示例使用 DDEInitiate（）函数建立 Visual FoxPro 与 Microsoft Excel 工作表 Sheet1 之间的 DDE 通道。使用 DDEExecute（）函数执行 Microsoft Excel 命令，最大化 Microsoft Excel 窗口。

```
gnChanNum = DDEInitiate('Excel', 'Sheet1')
IF gnChanNum != -1
    glExecute = DDEExecute(gnChanNum, '[App.Maximize]')
    IF glExecute != .F.
        WAIT WINDOW 'EXCEL window has been zoomed out.'
    ENDIF
    = DDETerminate(gnChanNum)  && 关闭通道
ENDIF
```

请参阅

DDEAbortTrans( ), DDEInitiate( ), DDELastError( ), DDESetOption( ), DDETerminate( )

DDEInitiate（）函数

在 Visual FoxPro 和另一个 Microsoft Windows 应用程序之间建立一个动态数据交换

(DDE)通道。

语法

DDEInitiate(cServiceName, cTopicName)

返值类型

数值型

参数描述

cServiceName

指定服务程序的名称。通常情况下，是一个不带扩展名的可执行文件名，Visual FoxPro默认的服务名是Visual FoxPro。如果建立通向Microsoft Excel的通道，*cServiceName*为Excel。

cTopicName

指定主题名。主题根据应用程序的不同而不同，因此指定的主题必须能让应用程序理解。例如，绝大多数DDE服务程序提供一个System主题。有关应用程序支持的服务名和主题名内容，请参阅应用程序文档。

说明

DDEInitiate（）在Visual FoxPro和DDE服务程序间建立一个DDE通道。通道建立后，Visual FoxPro就可以在后续函数中通过指定通道从服务程序中请求数据。Visual FoxPro也可以作为一个客户通过通道从服务程序中请求数据。

如果成功建立通道，则DDElnitiate（）返回通道号。通道号是非负值，所能建立的通道数量仅受系统资源的限制。

如果不能建立通道，则DDEInitiate（）返回–1；如果没有打开服务程序，则Visual

FoxPro 询问是否要打开。选择“是”，Visual FoxPro 会试图打开这个应用程序（可以用 DDELastError() 确定不能建立通道的原因）。

要避免询问是否打开应用程序，可设置 DDESetOption 的“SAFETY”选项，也可以使用带 /N 选项的 RUN 命令启动应用程序。

可以使用 DDETerminate() 关闭一个通道。

示例

下面的示例使用 DDEInitiate() 在 Visual FoxPro 和 Microsoft Excel 的工作表 Sheetl 之间建立 DDE 通道。‘Excel’是服务名，‘Sheetl’是主题名。通道号存放在变量 mchannum 中，可以供后续 DDE 函数使用。

```
mchannum = DDEInitiate('Excel', 'Sheet1')
IF mchannum != -1
   * Process 进行客户操作
   = DDETerminate(mchannum) &&关闭通道
ENDIF
```

请参阅

DDEAdvise(), DDEEnabled(), DDEExecute(), DDELastError(), DDEPoke(), DDERequest(), DDESetOption(), DDETerminate(), RUN | !

DDELastError（）函数

返回最近一个动态数据交换 (DDE) 函数的错误号。

语法

DDELastError( )

返值类型

数值型

说明

当 DDE 函数执行失败时，可以使用 DDELastError（）确定出错的原因。

如果最近的 DDE 函数执行成功，则 DDELastError（）返回 0；如果不成功，则返回一个非零值。下表列出了错误号和相应的说明。

错误号	说明
1	服务繁忙
2	主题繁忙
3	通道繁忙
4	没有这样的服务
5	没有这样的主题
6	错误通道
7	内存不足

续 表

8	接收超时 (Acknowledge timeout)
9	请求超时
10	没有 DDEInitiate( )
11	客户机尝试服务程序事务
12	执行超时
13	参数错误
14	低内存
15	内存出错
16	连接失败
17	请求失败
18	发送超时
19	不能显示信息
20	多个同步事务
21	服务程序已关闭
22	内部 DDE 错误
23	建议超时 (Advise timeout)
24	无效的事务标识符
25	不能识别

请参阅

DDEAbortTrans( ), DDEAdvise( ), DDEEnabled( ), DDEExecute( ),

DDEInitiate( ), DDEPoke( ), DDERequest( ), DDESetOption( ),
DDESetService( ), DDESetTopic( ), DDETerminate( )

DDEPoke（）函数

在动态数据交换 (DDE) 会话中，在客户和服务程序之间传送数据。

语法

DDEPoke(nChannelNumber, cItemName, cDataSent
[, cDataFormat [, cUDFName]])

返值类型

逻辑值

参数描述

nChannelNumber

指定发送应用程序数据的通道号。如果是服务程序通道，则 DDEPoke（）响应一个请求，或响应以前建立的报告链接或自动链接发送数据。

cItemName

指定接收数据的项名。项名根据应用程序的不同而不同，并且必须能被应用程序理解。例如，Microsoft Excel 支持 R1C1 为有效项名，它指明工作表中的第一个单元。

cDataSent

指定发送给项名的数据，项名由 *cItemName* 指定。

cDataFormat

指定发送数据的格式，默认格式是 CF_TEXT。在这种格式中，字段用 Tab 键分隔，记录用回车和换行符分隔。

cUDFName

允许异步数据传送。如果省略 *cUDFName*，客户会等待一段时间，时间长短由 DDESetOption（）指定。如果用 *cUDFName* 指定用户自定义函数，客户程序在请求之后继续执行下行代码。

当从服务程序得到数据时，执行 *cUDFName* 指定的用户自定义函数。用户自定义程序接受六个参数，参数按下列顺序传递：

参数	内容
Channel Number	服务程序的通道号
Action	XACTCOMPLETE（成功的事务） XACTFAIL（失败的事务）
Item	项名。例如，R1C1 表示 Microsoft Excel 工作表的单元
Data	新数据 (REQUEST) 或已传送的数据（POKE 或 EXECUTED）
Format	数据格式。例如，CF_TEXT
Transaction Number	DDEPoke（）返回的事务编号

可使用 DDEAbortTrans（）取消未完成的事务。如果事务失败了，可以用

DDELastError（）确定失败的原因。

若包含 *cUDFName*，事务成功时，DDEPoke（）返回一个事务号；发生错误时返回 -1。

说明

DDEPoke（）将数据以字符串形式发送给通道号指定的应用程序中的项名。

如果数据发送成功，则 DDEPoke（）返回“真”(.T.)；如果不能发送数据，则返回“假”(.F.)。如果包含一个异步用户自定义函数 *cUDFName*，则 DDEPoke（）返回一个事务号；如果发生错误，DDEPoke（）返回 -1。

请参阅

DDEAbortTrans( ), DDEInitiate( ), DDELastError( ), DDESetOption( ), DDETerminate( )

DDERequest（）函数

在动态数据交换 (DDE) 会话中，向一个服务程序请求数据。

语法

DDERequest(nChannelNumber, cItemName [, cDataFormat [, cUDFName]])

返值类型

字符型

参数描述

nChannelNumber

指定服务程序的通道号。

cItemName

指定接收数据的项名，项名根据应用程序不同而不同，因此，指定的项名必须能被应用程序理解。例如，Microsoft Excel 支持 R1C1 为有效项名，它指明工作表的第一个单元。

cDataFormat

指定发送数据的格式，默认格式是 CF_TEXT。在这种格式中，字段用 Tab 键分隔，记录用回车和换行符分隔。

cUDFName

允许异步数据传送。如果省略 *cUDFName*，Visual FoxPro 为从服务程序获取数据而等待一段时间，时间长短由 DDESetOption（）指定。如果用 *cUDFName* 指定用户自定义函数，Visual FoxPro 程序在请求之后马上执行下一行代码。

当从服务程序得到数据时，就执行 *cUDFName* 指定的用户自定义函数。用户自定义程序接受六个参数，参数按下列顺序传递：

参数	内容
Channel Number	服务程序的通道号
Action	XACTCOMPLETE（成功的事务） XACTFAIL（失败的事务）

续表

Item	项名。例如，R1C1 表示 Microsoft Excel 工作单的单元
Data	新数据 (REQUEST) 或已传送的数据（POKE 或 EXECUTED）
Format	数据格式。例如，CF_TEXT
Transaction Number	DDEPoke（）返回的事务编号

使用 DDEAbortTrans（）取消未完成的事务。如果事务失败，可用 DDELastError（）确定失败的原因。

当包含 *cUDFName* 时，事务成功则 DDEPoke（）返回一个大于等于 0 的事务号，出错则返回 -1。

说明

在用 DDERequest（）请求数据之前，必须用 DDEInitiate（）建立通向服务程序的通道。

如果数据请求成功，DDERequest（）以字符串的形式返回数据；如果请求失败，DDERequest（）返回一个空字符串，且 DDELastError（）返回一个非零值。如果包含异步用户自定义函数 *cUDFName*，则在事务成功时，DDERequest（）返回一个事务号；发生错误则返回 –1 。

示例

以下示例使用了 DDEInitiate（）函数建立 Visual FoxPro 与名为 Sheet1 的 Microsoft Excel 工作表之间的 DDE 通道。“Excel”是服务名称，“Sheet1”是主题名称。通道号存储在变量 mchannum 中，以便在随后的 DDE 函数中使用。

DDERequest() 要求数据项名称为 R1C1，电子表格 Sheet1 第一行和第一列中的数据。

```
mchannum = DDEInitiate('Excel', 'Sheet1')
IF mchannum != -1
    mrequest = DDERequest(mchannum, 'R1C1')
    IF !EMPTY(mrequest) AND DDELastError() = 0    && 成功
        WAIT WINDOW 'R1C1 contents: ' + mrequest
    ENDIF
    = DDETerminate(mchannum)          && 关闭通道
ENDIF
```

请参阅

DDEAbortTrans(), DDEInitiate(), DDELastError(), DDESetOption(), DDETerminate()

DDESetOption() 函数

更改或返回动态数据交换 (DDE) 的设置。

语法

DDESetOption(cOption [, nTimeoutValue | lExpression])

返值类型

逻辑值或数值型

参数描述

cOption

指定设置选项。

cOption	设置	默认值	说明
TIMEOUT	NTimeout Value（毫秒数）	2000	DDE 函数等待服务程序响应的毫秒数。如果省略 *nTimeoutValue*，则返回当前的 TIMEOUT 值。
SAFETY	*lExpression*（“真”(.T.) 或“假”(.F.)）	.T.	使用 DDEInitiate（）建立一个通向服务程序的通道，且应用程序没有响应时，指定是否显示对话框。如果省略 *lExpression*，则返回当前的 SAFETY 设置

nTimeoutValue

指定超时值。

lExpression

启用或禁止对话框的显示。

说明

使用 DDESetOption（）可以改变或返回 DDE 设置。有两个选项可用：TIMEOUT 和 SAFETY。

请参阅

DDEAdvise( ), DDEEnabled( ), DDEExecute( ), DDEInitiate( ), DDEPoke( ),

DDERequest( ), DDESetService( ), DDETerminate( )

DDESetService（）函数

创建、释放或更新 DDE 服务名和设置。

语法

DDESetService(cServiceName, cOption [, cDataFormat | lExpression])

返值类型

逻辑值

参数描述

cServiceName

指定要创建、释放、更改或返回信息的服务名。

cOption

指定是创建、释放或更改一个服务名，还是返回服务名的信息。下表列出了可用 *cOption* 指定的选项、选项的默认值以及对每个选项的说明。

选项	默认值	说明
DEFINE	–	创建新的服务名
RELEASE	–	释放已有的服务名
ADVISE	.F.	启用或禁止有关改变项名的报告

续表

EXECUTE	.F.	启用或禁止命令的执行
POKE	.F.	启用或禁止客户向服务的发送
REQUEST	.T.	启用或禁止对服务的请求
FORMATS	CF_TEXT	规定数据格式

DEFINE

创建一个新的服务名。例如，下面的命令创建服务名 myservice：

```
glNewService = DDESetService('myservice', 'DEFINE')
```

RELEASE

释放一个已有的服务名，从而释放系统资源。当服务名释放后，这个服务的所有主题名也跟着被释放。

下面的命令释放前一个例子的服务名

```
glRelease = DDESetService('myservice', 'RELEASE')
```

要释放默认的 Visual FoxPro 服务，可执行下面这条命令：

```
glRelFox = DDESetService('FoxPro', 'RELEASE')
```

ADVISE

当一个项中的数据改变时，指定是否通知客户，或者返回服务名的当前建议

状态。有关向客户提供建议的其他信息，请参阅 DDEAdvise()。

要启用客户通知，可指定 *lExpression* 为“真”(.T.)；将 *lExpression* 指定为“假”(.F.)可禁止客户通知。

要返回服务名的当前客户通知状态，省略 *lExpression* 即可。如果已经启用了服务名的客户通知，则 DDESetService() 返回“真”；如果已禁止客户机通知，则返回“假”。

EXECUTE

允许启用或禁止向一个服务名的命令请求，或确定服务名的当前执行状态。

要启用客户执行命令的请求，可以指定 *lExpression* 为“真”(.T.)；要禁止客户执行命令的请求，则可以指定 *lExpression* 为“假”(.F.)。“假”(.F.)为默认值。

要返回服务名的当前命令执行状态，省略 *lExpression* 即可。如果客户对服务的命令执行请求已启用，则 DDESetService() 返回“真”；否则返回“假”。

下面的命令启用服务名 myservice 的命令执行，并禁止客户应用程序对这个服务名的数据请求，然后显示 myservice 的当前命令执行状态：

```
glExecute = DDESetService('myservice', 'EXECUTE', .T.)
glRequest = DDESetService('myservice', 'REQUEST', .F.)
? DDESetService('myservice', 'EXECUTE')
```

POKE

允许启用或禁止对服务名的发送请求，或确定服务名的当前发送状态。有关向服务程序或客户发送数据的详细信息，请参阅 DDEPoke()。

要启用客户发送请求，可以将 *lExpression* 指定为“真”(.T.)；将 *lExpression* 指定为

“假”(.F.)，则可以禁止客户发送请求。“假”(.F.)是默认值。

要返回服务名的当前发送状态，省略 *lExpression* 即可。如果启用了服务名的发送请求，则 DDESetService() 返回“真”，否则返回“假”。

REQUEST

可以用 REQUEST 启用或者禁止对服务名的用户请求，或返回对服务名的当前请求状态。

要启用对服务名的客户请求，需将 *lExpression* 指定为“真”(.T.)；将 *lExpression* 指定为“假”(.F.)，则可以禁止对服务名的客户请求。“真”(.T.)是默认值。

要想返回服务名的当前请求状态，省略 *lExpression* 即可。如果启用了服务名的发送请求，则 DDESetService() 返回“真”，否则返回“假”。

下面的命令禁止从客户应用程序对服务名 myservice 的请求，并显示 myservice 的当前请求状态：

```
glRequest = DDESetService('myservice', 'REQUEST', .F.)
? DDESetService('myservice', 'REQUEST')
```

FORMATS [cDataFormat]

指定服务名支持的数据格式，没有用 *cDataFormat* 指定格式的服务程序请求会被拒绝。当指定数据格式时，请用逗号分隔列表中的格式。例如：

```
=DDESetService('myservice', 'FORMATS', 'CF_TEXT, CF_SYLK')
```

如果省略 *cDataFormat*，则只支持 CF_TEXT 格式。

lExpression

指定 REQUEST、EXECUTE、POKE 或 ADVISE 选项。将 *lExpression* 指定为“真”(.T.)可以启用一个选项，指定为“假”(.F.)则可以禁止一个选项。

说明

Visual FoxPro 可以作为一个动态数据交换 (DDE) 服务程序向 Microsoft Windows 的客户应用程序发送数据，DDESetService（）用来创建、释放或更改 Visual FoxPro 中的服务名和设置。每个服务名可以有多个由 DDESetSrvice（）创建的主题名，客户应用程序向 DDE 主题名请求数据。

如果成功地创建、释放或修改了服务名，则 DDESetService（）返回“真”(.T.)；如果服务名不能被创建、释放或修改，则 DDESetService（）返回“假”(.F.)。

DDESetservice（）也能返回有关服务名的信息。Visual FoxPro 有默认的服务名 Visual FoxPro，Visual FoxPro 服务名有一个称为 System 的主题名。下表列出了 System 主题支持的所有项名。

项名	项
Topics	可用的主题名列表
Formats	服务名支持格式列表
Status	繁忙或等待
SysItems	项名列表

可以使用 DDESetTopic（）修改 Visual FoxPro 服务名或释放服务名。有关 Visual FxPro 服务名操作的详细内容，请参阅 DDESetTopic（）选项。

请参阅

DDEEnabled( ), DDELastError( ), DDEPoke( ), DDESetTopic( )

DDESetTopic（）函数

在动态数据交换 (DDE) 会话中，创建或释放一个服务名的主题名。

语法

DDESetTopic(cServiceName, cTopicName [, cUDFName])

返值类型

逻辑值

参数描述

cServiceName

指定服务名。服务名可用 DDESetService（）创建。

cTopicName

指定要创建或释放的主题名。如果包括 *cUDFName*，则 DDESetTopic（）创建主题名 *cTopicName*；如果省略 *cUDFName*，则释放主题名 *cTopicName*。若 *cTopicName* 是一个空字符串，对任何没有明确声明的主题名，执行 *cUDFName* 指定的用户自定义函数。

cUDFName

在客户应用程序向主题名发出请求时，指定要执行的用户自定义函数。如果省略了

cUDFName，则从服务名中释放主题名 *cTopicName*。

参数	内容
Channel Number	客户通道号
Action	ADVISE，EXECUTE，INITIATE，POKE，REQUEST 或 TERMINATE
Item	项名。例如，R1C1 表示一个 Microsoft Excel 工作表单元
Data	来源于客户的数据
Format	数据格式。例如，CF_TEXT
Advise Status	链接类型（0 = 人工，2 = 报告或自动）

Item、Data 和 Advise Status 参数值取决于 Action 参数，下表列出了 Action 参数值以及包含在 Item、Data 和 Advise Status 参数中的值。短横线 (-) 表明参数值是个空字符串。

Action 值	Item 值	Data 值	Advise status 值
INITIATE	–	主题名	–
TERMINATE	–	–	–
POKE	项名	新数据	–
REQUEST	项名	–	–
EXECUTE	–	新数据	–
ADVISE	项名	–	链接类型

如果用户自定义函数成功地处理了客户请求，则用户自定义函数返回“真”(.T.)；如果

用户自定义函数不能处理请求或发生错误，则用户自定义函数应返回“假”(.F.)。如果Action参数值为INITIATE时返回“假”，则拒绝客户主题名的请求；如果在Action参数值是POKE、REQUEST或EXECUTE时返回“假”，则忽略请求；如果这个参数值为ADVISE时返回“假”，则拒绝客户报告或自动链接的请求。

说明

创建主题名后，任何对主题名的客户请求都会使Visual FoxPro执行*cUDFName*指定的用户自定义函数。向用户自定义函数传递的一组参数值是由客户请求决定的。用户自定义函数的返回值由DDEPoke（）函数传递给客户，其返回值是一个逻辑值，该值表明主题能否提供客户请求的服务。

如果成功地创建或释放主题名，则DDESetTopic（）返回“真”(.T.)；如果不能创建或释放主题名，则返回“假”(.F.)。使用DDELastError（）可确定不能创建或释放主题名的原因。

示例

以下示例创建了一个名为myserver的基本示例服务程序，此程序支持来自客户应用程序的Visual FoxPro 命令的执行。客户应用程序通过DO主题向myserver发出请求。

通过使用DO主题和宏替换执行客户的命令。

```
*** Set Visual FoxPro up as a DDE server ***
= DDESetService('myserver', 'DEFINE')
= DDESetService('myserver', 'EXECUTE', .T.)
= DDESetTopic('myserver', 'DO', 'DOTOPIC')
WAIT WINDOW 'Server portion service setup ... ' NOWAIT
*** Use Visual FoxPro as a DDE client ***
gnChannel = DDEInitiate('myserver','DO')
=DDEExecute(gnChannel, 'WAIT WINDOW "Command Executed ... "')
```

```
=DDETerminate(gnChannel)
PROCEDURE dotopic
PARAMETERS gnChannel, gcAction, gcItem, gData, gcFormat, gnAdvise
glResult = .F.
*** It's necessary to return .T. from an   ***
*** INITIATE action or no connection is made ***
IF gcAction = 'INITIATE'
    glResult = .T.
ENDIF
IF gcAction = 'EXECUTE'
    &gData
    glResult = .T.
ENDIF
IF gcAction = 'TERMINATE'
    WAIT WINDOW 'Goodbye ... ' NOWAIT
    glResult = .T.
ENDIF
RETURN glResult
```

在运行此示例程序之后，您已经建立了 Visual FoxPro 服务，其他应用程序可以访问此服务。如果有 Microsoft Excel，您可以运行以下 Excel 宏：

```
gnMyChan = INITIATE("myserver","DO")
=EXECUTE(MyChan,"WAIT WINDOW 'Hi, this is EXCEL speaking'")
=RETURN( )
```

请参阅

DDEEnabled( ), DDELastError( ), DDEPoke( ), DDESetService( )

DDETerminate（）函数

关闭用 DDEInitiate（）建立的动态数据交换 (DDE) 通道。

语法

DDETerminate(nChannelNumber | cServiceName)

返值类型

逻辑值

参数描述

nChannelNumber

指定要关闭的通道号。

cServiceName

指定要关闭的服务名。

说明

如果通道关闭成功，则 DDETerminate（）返回“真”(.T.)；如果通道不能关闭，DDETerminate（）返回“假”(.F.)。

不再需要通道时，应将它们关闭，节约系统资源。

如果通过选择“文件”菜单中的“退出”来退出 Visual FoxPro，或者通过命令窗口或从程序内执行 QUIT 退出 Visual FoxPro，那么通道将自动关闭。

请参阅

DDEAbortTrans( ), DDEAdvise( ), DDEExecute( ), DDEInitiate( ), DDELastError( ), DDEPoke( ), DDERequest( ), DDESetOption( )

Deactivate 事件

当一个容器对象（例如一个表单）因为所包含的对象没有焦点而不再处于活动状态时，该事件发生。对于一个工具栏来说，当使用 Hide 方法隐藏工具栏时，该事件发生。

语法

PROCEDURE Object.Deactivate

说明

在应用程序中只有移动焦点时，才发生 Activate 与 Deactivate 事件。将焦点移入或移出另一个应用程序的表单时，都不触发这两个事件。当卸载表单时，不发生 Deactivate 事件。

无论何时以编程方式或交互方式激活新对象时，都会触发原先活动对象的 Deactivate 事件，同时触发新建对象的 Activate 事件。

应用于

表单，表单集，页面，工具栏

请参阅

Activate 事件, Hide 方法, LostFocus 事件, Show 方法, Visible 属性

DEACTIVATE MENU 命令

使一个用户自定义菜单栏失效，并将它从屏幕上移开，但并不从内存中删除菜单栏的定义。

语法

DEACTIVATE MENU MenuName1 [, MenuName2 ...] | ALL

参数描述

MenuName1 [, MenuName2 ...]

指定失效的菜单栏名。包含逗号分隔的菜单栏列表可以使一组菜单栏失效。

ALL

使所有的活动菜单失效。

说明

DEACTIVATE MENU 从 Visual FoxPro 主窗口或用户自定义窗口中移去一个活动菜单栏或一组菜单栏，而不从内存中删除菜单栏的定义。可以用 ACTIVATE MENU 命令和菜单栏名重新激活菜单栏。

提示 当在应用程序中包含系统菜单栏(_MSYSMENU)时，不必定义、激活或禁止该菜单栏，只需发送 SET SYSMENU AUTOMATIC 命令即可。

要从内存中释放特定的一个或一组菜单栏，可使用 RELEASE MENUS 命令。可用 CLEAR MENUS 或 CLEAR 命令从内存中释放所有的菜单栏。

除非使用 DEFINE MENU BAR 创建菜单栏或使用 ACTIVATE MENU NOWAIT 激活菜单栏，否则程序控制回到激活菜单栏的命令行之后的程序行。

示例

以下示例使用 DEACTIVATE MENU 将取消菜单的激活并将它从屏幕中移去。使用 SET SYSMENU SAVE 将当前的系统菜单栏保存到内存中，并使用 SET SYSMENU TO 移去所有的系统菜单标题。

使用 DEFINE PAD 创建两个菜单标题，并使用 DEFINE POPUP 命令创建每个菜单标题的菜单。DEFINE BAR 命令创建每个菜单上的菜单项。当选择菜单标题时，ON PAD 使用 ACTIVATE POPUP 激活相应的菜单。ACTIVATE MENU 显示并激活菜单栏。

当从菜单中选择菜单项时，执行 CHOICE 过程。CHOICE 命令显示已选择的菜单项的名称和包含菜单项的菜单名。在 ACTIVATE MENU 行之后继续程序控制。

最后，取消菜单激活并将它从屏幕中移除，然后使用 RELEASE MENUS EXTENDED 命令从内存中释放菜单。

```
*** Name this program DEACMENU.PRG ***
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
ON KEY LABEL ESC KEYBOARD CHR(13)
DEFINE MENU EXAMPLE BAR AT LINE 1
DEFINE PAD convpad OF EXAMPLE PROMPT '\<Conversions' COLOR SCHEME 3 ;
```

```
    KEY ALT+C, ''
DEFINE PAD cardpad OF EXAMPLE PROMPT 'Card \<Info' COLOR SCHEME 3 ;
    KEY ALT+I, ''
ON PAD convpad OF EXAMPLE ACTIVATE POPUP conversion
ON PAD cardpad OF EXAMPLE ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' ;
    KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
    KEY CTRL+L, '^L'
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
    KEY CTRL+S, '^S'
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
    KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
    KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
    KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
    KEY CTRL+M, '^M'
ON SELECTION POPUP conversion DO choice IN deacmenu WITH PROMPT( ), POPUP( )
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
    KEY ALT+V, ''
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
    KEY ALT+P, ''
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' ;
    KEY ALT+W, ''
DEFINE BAR 4 OF cardinfo PROMPT '\-'
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
ON SELECTION POPUP cardinfo;
    DO choice IN deacmenu WITH PROMPT( ), POPUP( )
```

```
ACTIVATE MENU EXAMPLE
DEACTIVATE MENU EXAMPLE
RELEASE MENU EXAMPLE EXTENDED
SET SYSMENU NOSAVE
SET SYSMENU TO DEFAULT
ON KEY LABEL ESC
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ;
    ' from popup ' + mpopup NOWAIT
```

请参阅

ACTIVATE MENU, CLEAR ALL, CLEAR MENUS, CREATE MENU, DEFINE MENU, HIDE MENU, RELEASE, SHOW MENU

DEACTIVATE POPUP 命令

使 DEFINE POPUP 创建的菜单失效。

语法

DEACTIVATE POPUP MenuName1 [, MenuName2 ...] | ALL

参数描述

MenuName1 [, MenuName2 ...]

指定一个或多个要失效的菜单名。包含逗号分隔的多个菜单名可以使一组菜单失效。

ALL

使所有的活动菜单失效

说明

DEACTIVATE POPUP 只从 Visual FoxPro 主窗口或用户自定义窗口中移去一个或一组活动菜单，而不从内存中删除菜单定义。可以用 ACTIVATE POPUP 和菜单名重新激活菜单。

可以使用 RELEASE POPUPS 和菜单名从内存中释放特定的一个或一组菜单，用 CLEAR POPUPS 或 CLEAR ALL 从内存中释放所有菜单。

除非用 ACTIVATE POPUP NOWAIT 激活菜单，否则，程序控制将回到激活菜单的命令行的下一程序行。

请参阅

ACTIVATE POPUP, CLEAR ALL CLEAR POPUPS, CREATE MENU, DEFINE POPUP , HIDE POPUP, RELEASE POPUPS, SHOW POPUP

DEACTIVATE WINDOW 命令

使用户自定义窗口或 Visual FoxPro 系统窗口失效，并将它们从屏幕上移去，但不从内存中删除。

语法

DEACTIVATE WINDOW WindowName1 [,WindowName2 ...] | ALL

参数描述

WindowName1 [,WindowName2 ...]

指定一个或多个要失效的窗口。可以指定 Visual FoxPro 系统窗口，例如命令窗口或浏览窗口。

ALL

使所有的活动窗口失效。

若要重新显示 FoxPro 主窗口，可以从“窗口”菜单中选择“Visual FoxPro 屏幕”，或者发出 ACTIVATE WINDOW SCREEN 或 SHOW WINDOW SCREEN 命令。

说明

同时可放置多个用户自定义窗口，但只能向最近被激活的用户自定义窗口输出。当存在多个用户自定义窗口时，使当前窗口无效会清除该窗口的内容，并从屏幕上移走该窗口，将输出发送到前一个被激活的用户自定义窗口中。如果没有输出窗口，则输出将直接送入 Visual FoxPro 主窗口。

可使用 CLEAR WINDOWS 或 RELECASE WINDOWS 从屏幕和内存中删除窗口。

要使 Visual FoxPro 的系统窗口或工具栏无效，应将整个系统窗口或工具栏名包括在引号中。例如，要使 Visual FoxPro 的“报表控制”工具栏失效，可使用下列命令：

```
DEACTIVATE WINDOW "Report Controls"
```

示例

以下示例中，定义一个名为 wOutput1 的窗口并激活它。在显示来自 customer 表的记录之后，程序等待用户按键，随后取消窗口激活。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && Opens Customer table

CLEAR
DEFINE WINDOW wOutput1 FROM 2,1 TO 13,75 TITLE 'Output' ;
   CLOSE FLOAT GROW ZOOM
ACTIVATE WINDOW wOutput1

DISPLAY
WAIT WINDOW 'Press a key to deactivate the window'
DEACTIVATE WINDOW wOutput1
RELEASE WINDOW wOutput1
```

请参阅

ACTIVATE WINDOW, CLEAR WINDOWS DEFINE WINDOW, HIDE WINDOW, RELEASE WINDOWS, SHOW WINDOW

DEBUG 命令

启动 Visual FoxPro **调试器。**

语法

DEBUG

说明

调试环境设置 (FoxPro Frame 或 Debugger Frame) 决定了调试器是显示在 Visual FoxPro **桌面上还是在单独的窗口中。您可以在“选项”对话框中的“调试”选项卡中切换这两个设置。**

如果在您执行 DEBUG **命令时，调试器已经处于开启状态，则该命令激活调试器窗口。**

请参阅

CLOSE DEBUGGER, Debugging Tab, Options Dialog Box, DEBUGOUT, SET DEBUG

Debug 属性

指定在项目的已编译源代码中，是否包含调试信息。只在设计时可用。

语法

Object.Debug[= lExpression]

设置

lExpression

Debug 属性的设置有：

设置	说明
“真” (.T.)	（默认值）在已编译源代码中包含调试信息。
“假” (.F.)	不包含调试信息，并且不能在“跟踪”窗口中查看一个程序的运行，或者使用 MESSAGE(1) 返回引起错误的源代码。将 lExpression 设置为“假” (.F.)，与在 COMPILE 命令中包含 NODEBUG 相同。

说明

项目中的源代码包括程序和格式文件、表单、标签、报表和可视类库的源代码和数据库中的存储过程。Debug 属性对应于“项目信息”对话框中的“调试信息”复选框。

应用于

项目对象

请参阅

COMPILE, Project Information 对话框

DEBUGOUT 命令

在“调试输出”窗口中显示一个表达式的结果。

语法

DEBUGOUT eExpression

参数描述

eExpression

指定要在“调试输出”窗口中显示其结果的表达式。

说明

您可以使用 DEBUGOUT 命令通知调试者：程序的执行已进入到子程序或函数中。例如，您可以在每个过程的开始编写一句 DEBUGOUT 命令，在“调试输出”窗口中显示一段信息，表明已进入子程序。

注意，DEBUGOUT 的缩写不得少于 6 个字符，以免与 DEBUG 混淆。

请参阅

DEBUG, Debug 输出窗口, SET DEBUG

DECLARE 命令

创建一维或二维数组。

语法

DECLARE ArrayName1 (nRows1 [, nColumns1])
[, ArrayName2 (nRows2 [, nColumns2])] ...

说明

DECLARE **在操作和语法上与** DIMENSION **相同。有关详细内容，请参阅** DIMENSION。

请参阅

APPEND FROM ARRAY, COPY TO ARRAY DIMENSION, GATHER, PRIVATE, PUBLIC, SCATTER, SET COMPATIBLE, STORE

DECLARE – DLL 命令

注册外部 Windows 32 **位动态链接库** (.DLL) **中的一个函数。**

语法

```
DECLARE [cFunctionType] FunctionName IN LibraryName [AS
AliasName]
  [cParamType1 [@] ParamName1,
     cParamType2 [@] ParamName2, ...]
```

参数描述

cFunctionType

表明 32 **位** Windows .DLL **函数是否有返回值。如果函数有返回值，则包含** *cFunctionType*，**否则省略** *cFunctionType*。

cFunctionType **可以是以下值：**

cFunction Type	说明
SHORT	16 **位整数**
INTEGER	32 **位整数**
SINGLE	32 **位浮点数**

续表

DOUBLE	64 位浮点数
LONG	32 位长整数
STRING	字符串

FunctionName

指定在 Visual FoxPro 中要注册的 32 位 Windows .DLL 函数名。*FunctionName* 区分大小写。

注意 32 位 Windows .DLL 函数名可以与 Win32® API 手册中规定的不同。例如，MessageBox 函数应该命名为 MessageBoxA（对于单字节字符）和而 MessageBoxW（对于 UNICODE）。如果 Visual FoxPro 使用 cannot locate the DLL function you specify with FunctionName 参数无法定位您指定的 DLL 函数，在函数名称后附加字母 A 并且 Visual FoxPro 使用新名称再次搜索函数。

如果指定的 32 位 Windows .DLL 函数与一个 Visual FoxPro 函数同名，或者函数名不是合法的 Visual FoxPro 函数名，则将 32 位 Windows .DLL 函数名作为 *FunctionName*，并用 AS 子句给该函数取一个合法的 Visual FoxPro 名，这点在后面说明。

IN LibraryName

指定外部 Windows .DLL 的名称，该动态链接库包含 *FunctionName* 指定的 Windows 32 位 .DLL 函数。

如果为 *LibraryName* 指定 WIN32API，则 Visual FoxPro 就在 KERNEL32.DLL、GDI32.DLL、USER32.DLL、MPR.DLL 和 ADVAPI32.DLL 中搜索 32 位 Windows .DLL 函数。

AS AliasName

为一个与 Visual FoxPro 函数同名或不合法的 Windows 32 位 .DLL 函数名指定别名。*AliasName* 不能是 Visual FoxPro 的保留字，也不能是由 Visual FoxPro 注册的 Windows 32 位 .DLL 函数名。

可使用 *AliasName* 去调用 Windows 32 位 .DLL 函数。*AliasName* 不区分大小写。

cParameterType1 [@] *ParamName1*, *cParameterType2* [@] *ParamName2*, ...

指定传递给 Windows 32 位 .DLL 函数的参数类型。

Windows 32 位 .DLL 函数由 *cParameterType* 指定需要的参数类型。*cParameterType* 可以为下列某一类型：

cParameterType	说明
INTEGER	32 位整数
SINGLE	32 位浮点数
DOUBLE	64 位浮点数
LONG	32 位长整数
STRING	字符串

如果参数不是 Windows 32 位 .DLL 函数所需要的类型，Visual FoxPro 产生一个错误。null 值可以作为字符串传递。

调用函数时，若按引用传递一个参数，必须在这条命令的参数 *cParameterType* 后面、在调用函数相应变量前面包含 @。如果在 DECLARE 中或在调用函数中没有包含 @，参数就按值传递。有关需要 @ 按引用传递参数的共享库函数的详细内容，请参阅您的操作系统或环境的中文版程序员指南（例如有关向 Windows DLL 传递参数，请参考

或共享库函数使用。您可以将包含这些参数将它们作为函数接收的名称的提示和参数类型。

说明

在从 Visual FoxPro 调用一个 32 位 Windows .DLL 函数前，必须发出 DECLARE 命令，该命令带有函数名、包含该函数的 Windows .DLL 名以及函数将要接收的参数类型。

为了提供向后兼容性，Visual FoxPro 允许使用 SET LIBRARY 命令调用外部 API 库（使用 SET LIBRARY 命令，您可以访问 Foxtools.fll 中的函数）。但是，最好使用 DECLARE 来注册共享的库函数。

有关调用共享库函数的进一步内容，请参阅您的操作系统或环境的中文版程序员指南（例如有调用 DLL 的内容，请参考《Microsoft Win32 程序员指南》）。

执行 DISPLAY STATUS 或 LIST STATUS 可以显示注册函数的名称，执行 CLEAR ALL 或 CLEAR DLLS 可从内存中删除注册函数。

示例

此 Windows 示例当您切换到其他 Windows 应用程序时，返回 Visual FoxPro 句柄或零。当显示“等待”窗口时，您有 5 秒钟按下 ALT+TAB 切换到其他 Windows 应用程序，或可以离开作为活动应用程序的 Visual FoxPro 。

```
CLEAR
DECLARE INTEGER GetActiveWindow IN win32api
WAIT WINDOW "You can switch to another application now" TIMEOUT 5
? GetActiveWindow( )
```

请参阅

CALL, CLEAR, DISPLAY DLLS, DISPLAY STATUS, LIST DLLS, LOAD, SETLIBRARY

Default 属性

若活动表单上有两个或更多命令按钮，在按下 ENTER 时，指定哪个命令按钮或 OLE 容器控件做出响应。设计和运行时可用。

语法

Object.Default[= lExpr]

参数描述

lExpr

下表列出了 Default 属性的设置：

设置	说明
“真” (.T.)	当按钮的 Default 属性设置为“真” (.T.) 且父表单处于活动状态时，用户可以通过按下 ENTER 执行按钮命令。（如果焦点位于一个编辑框中，用户可以按 CTRL+ENTER。） 如果 KEYCOMPlngSET_KEYCOMP 设置为 WINDOWS，并且焦点移

续表

	到另一个命令按钮上时，按下回车键，不会执行默认按钮的命令。这时，按下回车键只能影响带焦点的按钮。
“假”(.F.)	（默认值）本按钮不是默认按钮。

说明

Default 属性只应用于包含了类似按钮的 ActiveX 控件 (.ocx) 的 OLE 容器控件。

在表单上只能有一个命令按钮或 OLE 容器控件作为默认的命令按钮。当一个命令按钮或 OLE 容器控件的 Default 属性设置为“真”(.T.) 时，表单上所有其他命令按钮或 OLE 容器控件都自动设置为“假”(.F.)。在工具栏中无法创建默认的命令按钮或 OLE 容器控件。

应用于

命令按钮，OLE 容器控件

请参阅

Enabled 属性，KeyPress 事件

DEFAULTEXT( ) 函数

如果一个文件没有扩展名，则返回一个带新扩展名的文件名。

语法

DEFAULTEXT(*cFileName*, *cDefault*)

返值类型

字符型

参数描述

cFileName

cFileName 指定要返回的文件名（带有或不带有路径或扩展名）。

cDefault

cDefault 指定默认的扩展名，没有点号。

请参阅

ADDBS( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTDRIVE( ), JUSTEXT( ), JUSTFNAME( ), JUSTPATH( ), JUSTSTEM( )

DefaultFilePath 属性

指定默认的一个 Application 对象使用的驱动器和目录。运行时可用。

语法

ApplicationObject.DefaultFilePath[= cPath]

参数描述

cPath

指定取下列一个值：

- 一个驱动器指示符。
- 一个带有目录名的驱动器指示符。
- 一个子目录名。
- 使用 MS-DOS 速记符号 (\ 或 ..) 以上内容。

说明

DefaultFilePath 属性类似于 SET DEFAULT 命令。

Visual FoxPro 在 Visual FoxPro 的默认目录中搜索一个文件。默认目录是启动 Visual FoxPro 的目录。但是，也可以使用 DefaultFilePath 属性指定另一个默认目录。如果 Visual FoxPro 无法在默认目录中找到一个文件，它会搜索 Visual FoxPro 的路径（如果指定了）。使用 SET PATH 可以指定 Visual FoxPro 的路径。

应用于

Application 对象，_VFP 系统变量

请参阅

SET DEFAULT, SET PATH

DEFINE BAR 命令

在 DEFINE POPUP 创建的菜单中创建一个菜单项。

语法

```
DEFINE BAR nMenuItemNumber1 | SystemItemName
OF MenuName PROMPT cMenuItemText
  [BEFORE nMenuItemNumber2 | AFTER nMenuItemNumber3]
  [FONT cFontName [, nFontSize]]
  [STYLE cFontStyle]
  [KEY KeyLabel [, cKeyText]]
  [MARK cMarkCharacter]
  [MESSAGE cMessageText]
  [SKIP [FOR lExpression]]
  [COLOR SCHEME nSchemeNumber
  | COLOR ColorPairList]
```

参数描述

nMenuItemNumber1

指定菜单项编号，以便在其他命令或函数中引用此菜单项。

SystemItemName

指定 Visual FoxPro 系统菜单中的一个菜单项。例如，要访问“打印”菜单项，可发出如下命令：

```
DEFINE BAR _MFI_PRINT OF popMyPopup PROMPT "Print..."
```

不是所有的 Visual FoxPro 系统菜单项都可用，使用 SYS(2013) 可返回可用的 Visual FoxPro 系统菜单列表。

OF MenuName

指定放置菜单项的菜单名。

PROMPT cMenuItemText

指定真正显示的菜单项文本。

为 *cMenuItemText* 指定一个反斜杠和破折号 (\-)，可创建一个分隔线。分隔线用来分隔菜单中的项组。例如，在菜单定义中包含如下命令，将在第三和第四个菜单项之间创建一条分隔线：

```
DEFINE BAR 4 OF popMyPopup PROMPT '\-'
```

在 cMenuItemText ***的开始指定一个反斜杠和竖线*** (\|)，可以创建多列菜单。该菜单项开始一个新列，并且后续菜单项放在相同的列中，直到遇到另一个以 \| 开头的菜单项。例如，在菜单定义中包含以下命令，可以在该菜单中创建一个新列，

```
DEFINE BAR 4 OF popMyPopup PROMPT '\|Start a new column'
```

在用作快捷键的字符前放置一个反斜杠和一个小于号(\<)可以创建一个快捷键。例如：

```
DEFINE POPUP popReceive
DEFINE BAR 1 OF popReceive PROMPT '\<Invoices'
DEFINE BAR 2 OF popReceive PROMPT 'In\<quiry'
ACTIVATE POPUP popReceive
```

用户可按下“I”键从 Receive 菜单中选择 Invoices，按下“Q”键从同样的菜单中选择 Inquiry。

BEFORE nMenuItemNumber2

在 *nMenuItemNumber2* 指定的菜单项前面放置一个菜单项。

AFTER nMenuItemNumber3

在 *nMenuItemNumber3* 指定的菜单项后面放置一个菜单项。

注意 为了使 BEFORE 或 AFTER 有效，用 DEFINE POPUP 创建菜单时必须包含 RELATIVE 子句。

也可以在 BEFORE 和 AFTER 子句中包含 _MFIRST 和 _MLAST。如果 BEFORE 子句中包含 _MFIRST，则菜单项就是菜单中的第一项；如果 AFTER 子句中包含 _MFIRST，则菜单项就是菜单上的第二项；如果在 AFTER 子句中包含 _MLAST，则菜单项就是菜单上的最后一项；如果在 BEFORE 子句中包含 _MLAST，则菜单项就是菜单上的倒数第二项。

用 DEFINE POPUP RELATIVE 创建的菜单不会给没有定义的菜单项保留空间。例如，如果在一个菜单上定义了项 1，2，4 和 5，则不为项 3 保留空间，可在以后插入项 3，使菜单扩充来容纳该项。

为菜单项指定字体。*cFontName* 指定字体的名称，*nFontSize* 指定字体磅值。例如，下面的命令用 12 磅 Courier 字体创建一个菜单项：

```
*** RELATIVE EXAMPLE ***
DEFINE POPUP popRelatYes RELATIVE FROM 1,1
DEFINE BAR 4  OF popRelatYes PROMPT '4444'
DEFINE BAR 3  OF popRelatYes PROMPT '3333'
DEFINE BAR 2  OF popRelatYes PROMPT '2222'
DEFINE BAR 1  OF popRelatYes PROMPT '1111'
DEFINE BAR 6  OF popRelatYes PROMPT '6666' BEFORE 4
ACTIVATE POPUP popRelatYes

*** NON-RELATIVE EXAMPLE ***
DEFINE POPUP popRelatNo FROM 1,10
DEFINE BAR 4 OF popRelatNo PROMPT '4444'
DEFINE BAR 3 OF popRelatNo PROMPT '3333'
DEFINE BAR 2 OF popRelatNo PROMPT '2222'
DEFINE BAR 1 OF popRelatNo PROMPT '1111'
DEFINE BAR 6 OF popRelatNo PROMPT '6666'
ACTIVATE POPUP popRelatNo
```

FONT cFontName [, nFontSize]

为菜单项指定字体。*cFontName* 指定字体的名称，*nFontSize* 指定字体磅值。例如，下面的命令用 12 磅 Courier 字体创建一个菜单项：

```
DEFINE BAR 1 OF popReceive PROMPT '\<Invoices' FONT 'Courier', 12
```

如果指定的字体不可用，则用相似的字体代替。如果包含 FONT 子句但省略磅值 *nFontSize*，则使用 10 磅。

STYLE cFontStyle

为菜单项指定字形。如果省略 STYLE 子句，则使用常规的字形。在 Visual FoxPro 中，如果指定的字形不可用，则用相似的字形代替。

cFontStyle 可指定以下字形：

字符	字体样式
B	粗体
I	斜体
N	常规体
Q	不透明
–	删除线
T	透明
U	下划线

可组合多个字符指定一个字形。例如，下面的命令指定字形为“粗斜体”：

```
DEFINE BAR 1 OF popReceive PROMPT '\<Invoices' STYLE 'BI'
```

KEY KeyLabel [, cKeyText]

为菜单项指定访问键或组合键。与使用反斜杠和小于号 (\<) 指定访问键不同，此时若要选择菜单项时，不一定要激活菜单。有关可用键、可用键组合以及键标记名称列表的内容，请参阅 ON KEY LABEL。

注意 如果对同样的键定义了键标记和键盘宏，则键盘宏优先，不能用指定的键或组合键去选择菜单项。

包含 *cKeyText* 可用自己的文本替换键标记。例如，如果包含 KEY CTRL+B，则将菜单上的文本 CTRL+B 放在菜单项名的右端；如果指定的不是 KEY CTRL+B，而是 "^B"，则 ^B 就会出现在菜单上。也可以通过指定 *cKeyText* 为空字符串而不输出一个键标记的显示。

MARK cMarkCharacter

指定出现在菜单项左端的标记字符。通过包含 MARK 可以将默认的标记字符改为 *cMarkCharacter* 指定的字符。如果 *cMarkCharacter* 包含不止一个字符，则只有第一个字符用作标记字符。

默认的标记字符为一个对勾。

在 Visual FoxPro 中，如果包含菜单项的菜单合并到 Visual FoxPro 系统菜单中，则忽略 MARK 子句而使用默认的标记字符。如果包含菜单项的菜单放置在 Visual FoxPro 主窗口或用户自定义窗口中，而 FoxFont 不是该窗口的字体，则忽略 MARK 子句。

注意 标记字符并不标记一个菜单项，可使用 SET MARK OF 来标记一个菜单项。

在 DEFINE BAR 中指定的标记字符优先于 DEFINE POPUP 中用 MARK 指定的标记字

符。SET MARK OF 用来将标记字符切换为开或关，也可用来为单个菜单项或所有菜单项指定标记字符。

MESSAGE cMessageText

当选择一个菜单项时显示一条信息。在 Visual FoxPro 中，消息放在图形方式状态栏中。如果用 SET STATUS BAR OFF 关闭图形方式状态栏，则消息放到 Visual foxPro 主窗口最后一行的中央。

SKIP [FOR lExpression]

指定一个条件。如果 *lExpression* 计算为“真”(.T.)，则禁止菜单项，不让用户选择它；如果为“假”(.F.)，则激活此菜单项。无效的菜单项以暗灰色显示。

也可以通过在提示文本前加一个反斜杠 (\) 使一个菜单项无效。例如：

```
DEFINE BAR 1 OF popReceive PROMPT '\Invoices'
```

在 Visual FoxPro 中不能选择用 SKIP 或 \ 禁止的菜单项。当拼写检查或向导处于活动状态时，所创建的包含 SKIP FOR 表达式的菜单可能不会正确执行。

SKIP FOR 表达式通常依赖于跳过变量 (skip variable) 的值，但当拼写检查或向导处于活动状态时，跳过变量对菜单是不可见的。在拼写检查和向导应用程序（分别为 SPELLCHK.APP 和 GENGRAPH.APP）的初始化代码中都有意设置了 PRIVATE ALL，这将会隐藏用户自定义菜单中的跳过变量，并且在选择一个菜单时会产生错误信息。要纠正这种错误，可将下面代码放在拼写检查和向导应用程序的开头：

```
IF TYPE("_memvarmask") = "C" and !EMPTY(_memvarmask)
   PRIVATE ALL EXCEPT &_memvarmask
```

```
ELSE
   PRIVATE ALL
ENDIF
```

假定名为“skipvar”的变量计算为“真”时要跳过某一菜单项。可以在菜单启动代码中包含下面几行来利用 _MEMVARMASK 变量：

```
PUBLIC _memvarmask
_memvarmask = "skipvar"
STORE .T. TO skipvar    && 初始时跳过。
```

要创建一组跳过变量，可以在菜单启动代码中包含下面几行：

```
PUBLIC _memvarmask
_memvarmask = "skip*"
STORE .T. TO skipthis, skipthat  && 初始时跳过。
```

当运行拼写检查和向导时，它们不隐藏 SKIP FOR 表达式中的变量，这就避免了在拼写检查或向导处于活动状态时可能遇到的错误信息。

注意：_MEMVARMASK 不是一个系统变量。

COLOR SCHEME nSchemeNumber

为单个菜单项指定颜色，这会改写默认颜色或 DEFINE POPUP 指定的颜色。

COLOR ColorPairList

为单个菜单项指定颜色，这将改写默认颜色或 DEFINE POPUP 指定的颜

色。可为所有菜单项、标记字符以及信息指定颜色。
默认情况下，菜单项颜色由当前颜色集的配色方案 2 决定。
有关颜色方案和颜色对的详细内容，请参阅稍前部分的语言参考“颜色概述”。

说明

DEFINE BAR 和 DEFINE POPUP 共同来创建菜单，用 DEFINE POPUP 创建菜单并给它命名，用一组 DEFINE BAR 命令在菜单上放置菜单项。
如果使用菜单设计器创建菜单，则没有必要使用这些命令。菜单设计器自动为菜单创建一个命令。菜单设计器使用 Visual FoxPro 系统菜单，您可以增加菜单项来更改它。
有关创建菜单的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十一章“设计菜单和工具栏”。
也可创建一个菜单，使其包含位于磁盘上的表或一组文件中的记录或字段。有关详细内容，请参阅 DEFINE POPUP 中的 PROMPT FIELD 子句、PROMPT STRUCTURE 子句和 PROMPT FILES 子句。
使用 ON BAR 可为菜单项创建一个层叠式子菜单。

示例

以下示例使用 DEFINE BAR 命令创建菜单上的菜单项。首先使用 SET SYSMENU SAVE 命令将当前系统菜单栏保存在内存中，然后使用 SET SYSMENU TO 命令移除所有系统菜单标题。
使用 DEFINE PAD 命令创建两个系统菜单标题，并使用 DEFINE POPUP 命令为每个菜单标题创建一个下拉式菜单。使用 DEFINE BAR 命令创建菜单上的菜单项。当选择菜单标题时，ON PAD 使用 ACTIVATE POPUP 激活相应的菜单。
当从菜单中选取菜单项时，ON SELECTION POPUP 使用 PROMPT() 和 POPUP()

向 CHOICE 过程传递菜单项数目和菜单名称。CHOICE 显示已选取菜单项的提示和包含菜单项的菜单名。如果从 Card Info 菜单中选取 Exit 项，恢复原始的 Visual FoxPro 系统菜单。

```
*** Name this program DEFINBAR.PRG ***
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
DEFINE PAD convpad OF _MSYSMENU PROMPT '\<Conversions' COLOR SCHEME 3 ;
    KEY ALT+C, ''
DEFINE PAD cardpad OF _MSYSMENU PROMPT 'Card \<Info' COLOR SCHEME 3 ;
    KEY ALT+I, ''
ON PAD convpad OF _MSYSMENU ACTIVATE POPUP conversion
ON PAD cardpad OF _MSYSMENU ACTIVATE POPUP cardinfo
DEFINE POPUP conversion MARGIN RELATIVE COLOR SCHEME 4
DEFINE BAR 1 OF conversion PROMPT 'Ar\<ea' KEY CTRL+E, '^E'
DEFINE BAR 2 OF conversion PROMPT '\<Length' ;
    KEY CTRL+L, '^L'
DEFINE BAR 3 OF conversion PROMPT 'Ma\<ss' ;
    KEY CTRL+S, '^S'
DEFINE BAR 4 OF conversion PROMPT 'Spee\<d' ;
    KEY CTRL+D, '^D'
DEFINE BAR 5 OF conversion PROMPT '\<Temperature' ;
    KEY CTRL+T, '^T'
DEFINE BAR 6 OF conversion PROMPT 'T\<ime' ;
    KEY CTRL+I, '^I'
DEFINE BAR 7 OF conversion PROMPT 'Volu\<me' ;
    KEY CTRL+M, '^M'
ON SELECTION POPUP conversion;
    DO choice IN definbar WITH PROMPT( ), POPUP( )
DEFINE POPUP cardinfo MARGIN RELATIVE COLOR SCHEME 4
```

```
DEFINE BAR 1 OF cardinfo PROMPT '\<View Charges' ;
    KEY ALT+V, ''
DEFINE BAR 2 OF cardinfo PROMPT 'View \<Payments' ;
    KEY ALT+P, ''
DEFINE BAR 3 OF cardinfo PROMPT 'Vie\<w Users' KEY ALT+W, ''
DEFINE BAR 4 OF cardinfo PROMPT '\-'
DEFINE BAR 5 OF cardinfo PROMPT '\<Charges '
DEFINE BAR 6 OF cardinfo PROMPT '\-'
DEFINE BAR 7 OF cardinfo PROMPT 'E\<xit '
ON SELECTION POPUP cardinfo;
    DO choice IN definbar WITH PROMPT( ), POPUP( )
PROCEDURE choice
PARAMETERS mprompt, mpopup
WAIT WINDOW 'You chose ' + mprompt + ;
    ' from popup ' + mpopup NOWAIT
IF mprompt = 'Exit'
    SET SYSMENU TO DEFAULT
ENDIF
```

请参阅

ACTIVATE POPUP, DEACTIVATE POPUP, DEFINE POPUP, HIDE POPUP, RELEASE BAR, SET MESSAGE, SHOW POPUP

DEFINE BOX 命令

在打印文本周围画一个方框。可用报表设计器代替。

DEFINE CLASS 命令

创建一个用户自定义类或子类，并为创建的类或子类指定属性、事件和方法。

语法

```
DEFINE CLASS ClassName1 AS ParentClass [OLEPUBLIC]
  [[PROTECTED | HIDDEN PropertyName1, PropertyName2 ...]
     [Object.]PropertyName = eExpression ...]
  [ADD OBJECT [PROTECTED] ObjectName AS ClassName2 [NOINIT]
     [WITH cPropertylist]]...
  [[PROTECTED | HIDDEN] FUNCTION | PROCEDURE Name[_ACCESS |
_ASSIGN]
  | THIS_ACCESS [NODEFAULT]
     cStatements
```

[ENDFUNC | ENDPROC]]...
ENDDEFINE

参数描述

ClassName1

指定要创建的类名。

AS ParentClass

指定派生类或子类的父类，父类可以是一个 Visual FoxPro **基类（例如** Form **类）或者是另一个用户自定义的类或子类。**

下表列出了 Visual FoxPro **的基类：**

基类名

ActiveDoc	CheckBox	Column
ComboBox	CommandButton	CommandGroup
Container	Control	Cursor
Custom	DataEnvironment	EditBox
Form	FormSet	Grid
Header	Hyperlink	Image
ProjectHook	Label	Line
ListBox	OLEBoundControl	OLEControl
OptionButton	OptionGroup	Page
PageFrame	Relation	Separator
Shape	Spinner	TextBox
Timer	ToolBar	

通过指定 *ParentClass* 为 Custom，可以创建一个非可视的用户自定义类。
在以下示例中，从表单 (Form) 基类中创建了一个名为 MyForm 的子类。当单击 MyForm 时，创建的 Click 方法显示一个对话框。

```
DEFINE CLASS MyForm AS Form
   PROCEDURE Click
     = MESSAGEBOX('MyForm has been clicked!')
   ENDPROC
ENDDEFINE
```

OLEPUBLIC
指明当前定义的类将要作为一个用户自定义的 OLE 服务程序 (custom OLE server)。这个类可以被其他 OLE 客户程序 (OLE client) 访问。
如果一个项目中包含带有 OLEPUBLIC 的类定义，那么，当使用项目管理器（或使用 BUILD EXE 或 BUILD DLL 命令）生成 .EXE 或 .DLL 时，.EXE 或 .DLL 文件既自动注册到操作系统中。这时，其他 OLE 客户程序就可使用它了。
有关创建自定义 Automation 服务程序的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十六章“添加 OLE”中的“创建 Automation 服务程序”。
[PROTECTED | HIDDEN PropertyName1, PropertyName2 ...]
[Object.]PropertyName = eExpression ... 创建一个类属性或子类属性，并给属性赋默认值。这些属性是类的命名属性，它们为类定义了特性和行为。类和子类可以有多个属性。
可使用“=”为属性赋值。以下示例创建了一个用户自定义名为 MyClass 类别并创建了

名为 Name 和 Version 的两个属性。将 Name property 初始化为空字符串，Version 属性初始化为字符串 1.0。

```
DEFINE CLASS MyClass AS Custom
   Name = ''
   Version = '1.0'
ENDDEFINE
```

当用 CREATEOBJECT（）函数创建对象后，可以在类或子类的定义之外访问属性：

```
MyOjbect = CREATEOBJECT('MyClass')
```

访问属性的语法如下：

```
ObjectName.Property
```

在属性名前加 PROTECTED 可以阻止从类定义或子类定义的外部访问和改变这些属性，在类或子类内部定义的方法和事件可以访问被保护属性。

以下示例向表单中添加了 Outline ActiveX 控件。在创建 Outline 控件之前，使用 Object 关键字为其指定属性。

```
PUBLIC frmOLETest
frmOLETest = CREATEOBJECT('Form')
frmOLETest.Visible = .T.

frmOLETest.ADDOBJECT('OCXTest', 'BlueOLEControl', ;
   'MSOutl.Outline')
```

```
frmOLETest.OCXTest.AddItem('Item One')
frmOLETest.OCXTest.AddItem('Item Two')

DEFINE CLASS BlueOLEControl AS OLEControl

    *设置 ActiveX 控件的属性 l
    .Object.Backcolor = 16776960

    *设置 OLE Container 控件的属性
    Visible = .T.
    Height = 100
    Width = 200
ENDDEFINE
```

包含 PROTECTED 和属性名称的列表以防止从类或子类定义以外访问或更改属性。类或子类定义中的方法和事件可以访问被保护的属性。
在以下示例中，保护了 Version 属性，防止从类定义以外访问和更改它。但是没有保护 Name 属性，可以访问并更改该属性。

```
DEFINE CLASS MyClass AS Custom
    PROTECTED Version
    Name = ''
    Version = '1.0'
ENDDEFINE
```

包含 HIDDEN 和属性名称的列表以防止从类定义以外访问和更改属性。只有在类定义中的方法和事件才可以访问隐藏的属性。通过类定义的子类可以被保护的属性，但只有从类定义中才能访问隐藏属性。

ADD OBJECT

从 Visual FoxPro 的一个基类、用户自定义类（子类）或从 OLE 自定义控件向一个类定义或子类定义中添加一个对象。

PROTECTED

禁止从类定义或子类定义的外部访问和修改对象的属性，PROTECTED 关键字必须直接放在 *ObjectName* 之前，否则 FoxPro 将产生语法错误。

ObjectName

在类定义或子类定义创建一个对象后，*ObjectName* 指定对象名，并用来从类定义或子类定义内部引用对象。

AS ClassName2

指定类或子类名。该类或子类中包含添加到类定义中的对象。

```
DEFINE CLASS MyClass AS Custom
   ADD OBJECT CB1 AS CommandButton
   ADD OBJECT LIST1 AS ListBox
ENDDEFINE
```

NOINIT

当添加对象时，不执行对象的 Init 方法。

WITH cPropertyList

为添加到类定义或子类定义中的对象指定一列属性和属性值。例如，以下类定义创建了一个名为 MyClass 的类，向类定义中添加一个命令按钮，为命令按钮指定 Caption 和 BackColor 属性。

```
DEFINE CLASS MyClass AS CUSTOM
   ADD OBJECT CB1 AS CommandButton;
    WITH Caption = 'Cancel', BackColor = 2
ENDDEFINE
```

FUNCTION | PROCEDURE Name[_ACCESS | _ASSIGN] | THIS_ACCESS

为类（或者子类）创建事件和方法，而事件和方法是作为一组函数或过程来创建的。

可以在类定义或子类定义中创建事件函数或过程去响应一个事件。一个事件是一次活动（例如一次鼠标单击），它可以由类定义或子类定义创建的对象去识别。有关 Visual FoxPro 事件处理的其他内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第四章“深入了解事件模型”中的“为事件指定代码”。

事件可用下面的语法调用：

```
ObjectName.Event
```

也可以在一个类或子类的定义中创建一个方法。一个方法就是一个过程，它作用于由类或子类定义创建的对象。方法可用下列语法调用：

```
ObjectName.Method
```

ACCESS 和 _ASSIGN 前缀可以添加到一个过程或函数名前，以便为具有相同名称的属性创建一个 Access 或 Assign 方法。每当查询该属性时，都执行 Access 方法中的代码。每当更改该属性时，都执行 Assign 方法中的代码。

另外，您可以创建一个 THIS_ACCESS 函数过程，它在您试图更改对象程序或查询对象程序时执行。

有关使用 DEFINE CLASS 创建 Access 和 Assign 方法的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》中的三十三章“对编程的改进”。

NODEFAULT

NODEFAULT 会导致 Visual FoxPro 不对它本身的事件和方法进行默认处理。例如，如果发生了 KeyPress 事件，将 NODEFAULT 放在 KeyPress 过程或函数中，可以防止 Visual FoxPro 将按下的键放在 Visual FoxPro 键盘缓冲中。这就允许您创建一个 KeyPress 过程，在将按下的键放在键盘缓冲中之前检查按下了哪个键。

NODEFAULT 可以放在事件或方法过程的任何位置。注意，NODEFAULT 也可放在表单设计器的事件或方法过程中。

cStatements

[ENDFUNC | ENDPROC]]...

ENDDEFINE

cStatements 是 Visual FoxPro 命令，当执行一个事件或方法时就执行它。

通过将 PARAMETERS 或 LPARAMETERS 作为函数或过程的第一个可执行语句，可以让事件和方法函数和过程接受参数。

与大多数 Visual FoxPro 关键字不同，不能缩写 ENDFUNC 和 ENDPROC，

主要是为了避免与 ENDFOR 和 ENDPRINTJOB 关键字发生冲突。

下列示例演示了如何创建一个当单击命令按钮时显示一条消息的事件过程。这个事件过程会取代命令按钮默认的 Click 事件。

```
DEFINE CLASS MyClass AS Custom
   ADD OBJECT MyButton AS CommandButton
   ADD OBJECT MyList AS ListBox
   PROCEDURE MyButton.Click
    = MESSAGEBOX('This is my click event procedure')
   ENDPROC
ENDDEFINE
```

说明

用户自定义类是放在程序文件中的一组命令，类似一个过程。与过程一样，这些类（或子类）定义后的命令并不执行，它们只定义类（或子类）的属性、事件和方法。

注意 在一个程序文件中，在过程后面不能包含普通的可执行程序代码；在程序文件中，在第一个 DEFINE CLASS、PROCEDURE 或 FUNCTION 命令之后只能包含类定义、过程和用户自定义函数。

不能在结构化编程命令中使用 DEFINE CLASS 创建类定义或子类定义（例如 IF…ENDIF 或 DO CASE…ENDCASE），它们也不能放在循环语句中（例如 DO WHILE…ENDDO 或 FOR…ENDFOR）。

要根据一个类定义或子类定义创建对象，应发出包含该类名或子类名的 CREATEOBJECT() 命令。

示例

以下示例使用 DEFINE CLASS 和 CREATEOBJECT() 从 Visual FoxPro 表单 (Form) 基类中创建了两个自定义类，名为 FormChild 和 FormGrandChild。使用 ACLASS() 创建一个名为 gaNewarray 的数组来包含类名称，该类名称随后显示。

```
CLEAR
frmMyForm = CREATEOBJECT("FormGrandChild")
FOR nCount = 1 TO ACLASS(gaNewarray, frmMyForm)    && Creates an array
    ? gaNewarray(nCount)  && display the names of the classes
ENDFOR
RELEASE frmMyForm

DEFINE CLASS FormChild AS FORM
ENDDEFINE
DEFINE CLASS FormGrandChild AS FormChild
ENDDEFINE
```

请参阅

:: Scope Resolution Operator, ADD CLASS, _BROWSER, CREATE CLASS, CREATE CLASSLIB , CREATEOBJECT( ) , DODEFAULT( ), GETOBJECT( ) , MODIFY CLASS , RELEASE CLASSLIB, SET CLASSLIB, WITH ... ENDWITH