Microsoft Visual FoxPro 6.0 Language Reference
中文版语言参考手册

北京希望电脑公司

Microsoft Press

美国微软出版社授权中文版系列书

Visual Foxpro 6.0 中文版

语言参考手册

Microsoft Corporation 著

希望图书创作室 译

龙启铭等 审校

1.本书配套电子书
2.《跨越 Photoshop 4.0 增强版》多媒体教学软件

北京希望电脑公司

1998

内容简介

美国微软公司为满足中国用户更好、更快地学习和掌握新的微机编程和开发工具，一种特别的做法是：新版软件配套的中文版使用手册和开发人员指南单独销售，这就为从事微软相关软件的开发群体和个人提供了最大的方便：通过配套手册的学习，可大大缩短掌握和应用相关软件的时间，从而提高开发效率。本次推出的与软件配套的手册包括：Visual Foxpro 6.0 中文版程序员指南、Visual Foxpro 6.0 中文版语言参考手册、Visual Basic 6.0 中文版程序员指南、Visual Basic 6.0 中文版语言参考手册、Visual C++6.0 程序员指南、Visual C++6.0 运行库参考手册、Visual C++6.0 模板库参考手册、Visual C++6.0 基本类库参考手册、Visual C++6.0 语言参考手册、Visual InterDev 6.0 程序员指南、Visual InterDev 6.0 Web 参考手册、Visual J++ 6.0 程序员指南、Visual J++ 6.0 基本类库参考手册。请广大的读者朋友密切留意出版信息。

《Visual Foxpro 6.0 中文版语言参考手册》是美国微软出版社授权的 Microsoft Visual Studio 中文版系列书之一。本书按字母顺序介绍了 Visual FoxPro 的命令、函数、控件和对象、属性、事件和方法等各种语言元素，每个语言主题均字母顺序出现并且提供一般说明、语法、参数描述、说明和示例等信息。

本书的内容十分丰富，它不但是从事 Visual Foxpro 6.0 应用和开发的所有用户的使用指南和工具书，同时也是大专院校相关专业的师生、科研院所的科技人员自学和教学的重要参考书。

本书还提供配套的电子书，以方便读者携带、学习和长久保存。

欲购本书或需技术支持的用户请直接与北京海淀 8721 信箱联系，邮政编码：100080，联系电话：010-62562329，62541992，传真：010-62579874，62633308。

Visual Foxpro 6.0 中文版语言参考手册

Microsoft Corporation 著
希望图书创作室译
责任编辑　　陆卫民
北京希望电脑公司　　出品
北京海淀路 82 号（100080）
北京双青印刷厂　印刷

新华书店、新华书店音像发行所发行、各地书店、软件专卖店经销
1998 年 12 月第 1 版　1998 年 12 月第 1 次印刷
开本：787×1092 1/16　　印张：90.5
字数：2094 千字　　印数：1-5000
新出音管[1998]164 号
ISBN 7-980021-21-5/TP·14
定价：138 元（1CD，含配套书）

致　谢

近 10 年来，我国从事计算机数据库应用和开发的人员数以千万计。其中 Microsoft 公司近年来推出的关系数据库 FoxPro、Visual FoxPro 由于其优秀的功能更是受到国内广大计算机新老用户的青睐。

最近由 Microsoft 公司又正式推出了 Visual FoxPro 6.0 中文版。由于它是一个 32 位的、面向对象的应用程序开发环境，并具有 Internet 开发功能，在产业界引起了巨大的反响。Visual FoxPro 6.0 的汉化工作是由北京汉扬天地科技发展有限公司承担。为满足国内广大用户的迫切需求，希望图书创作室根据用户和市场的需求与微软出版社合作，编译出版了本书。编译工作由参与软件汉化的北京汉扬天地科技发展有限公司的许震宇、陈曦、龚洁、饶丽、雷驹、范志宏等完成。

北京汉扬天地科技发展有限公司的韩冬晖先生在成书过程中给予了大量帮助，在此致以深深的谢意！

希望图书创作室

简介

欢迎使用 Microsoft Visual FoxPro ，使用这个关系型数据库系统可以简化数据的管理，优化应用程序的开发。

关于本书

《Visual FoxPro6.0 中文版语言参考》按字母顺序介绍了 Visual FoxPro 的语言。在本书中涉及到了以下语言元素：

- 命令
- 函数
- 控件和对象
- 属性
- 事件
- 方法

每个语言主题都是按字母顺序出现的，并且提供了以下信息：

- 一般说明
- 语法

- 参数和设置说明，还有附加说明
- 代码示例

有关如何使用 Visual Foxpro 编程语言及其元素的概述，请参阅“帮助”中的“语言概述”。有关编程的概念、使用类以及使用属性、事件和方法管理对象的详细内容，请参阅《*Microsoft Visual FoxPro 6.0* 中文版程序员指南》中的前四章。

获得帮助

Visual FoxPro 帮助系统可以使您快速了解有关使用 Visual FoxPro 设计工具和语言元素的信息。

需要使用帮助时，请按 F1

如果您正在使用一个命令或关键字，并且需要了解更多的相关内容，可以在“命令”窗口中突出显示该命令或关键字，并按 F1 键，以显示与该项内容有关的上下文相关的帮助主题。注意，为了获得帮助，在安装时一定要选择 MSDN Library 选项。

示例文件

在 Visual FoxPro 和 MSDN Library 中包括各种示例应用程序、数据库和文件，以演示编程技术。有关详细内容，请在启动 Visual Foxpro 时参见“欢迎使用 Visual FoxPro ”屏幕。

使用代码示例

帮助中的很多语言主题都包含代码示例，您可以在程序窗口中运行这些示例。在运行这些程序示例之前，必须先进行安装。

若要运行一个代码示例

1. 在帮助主题的顶部，单击“示例”。
2. 选中想要运行的代码，并复制到剪贴板上 (CTRL+C)。
3. 在 Visual FoxPro 中，单击“命令”窗口，键入 MODIFY COMMAND，以打开程序窗口，或者选择“文件”菜单中的“新建”命令，在“新建”对话框中单击“程序”选项，然后单击“新文件”按钮。
4. 将代码粘贴到程序窗口中 (CTRL+V)。
5. 单击 ! 按钮，以运行该程序。
6. 在提示时命名并保存该程序。在保存后，就会运行该程序。

文档约定

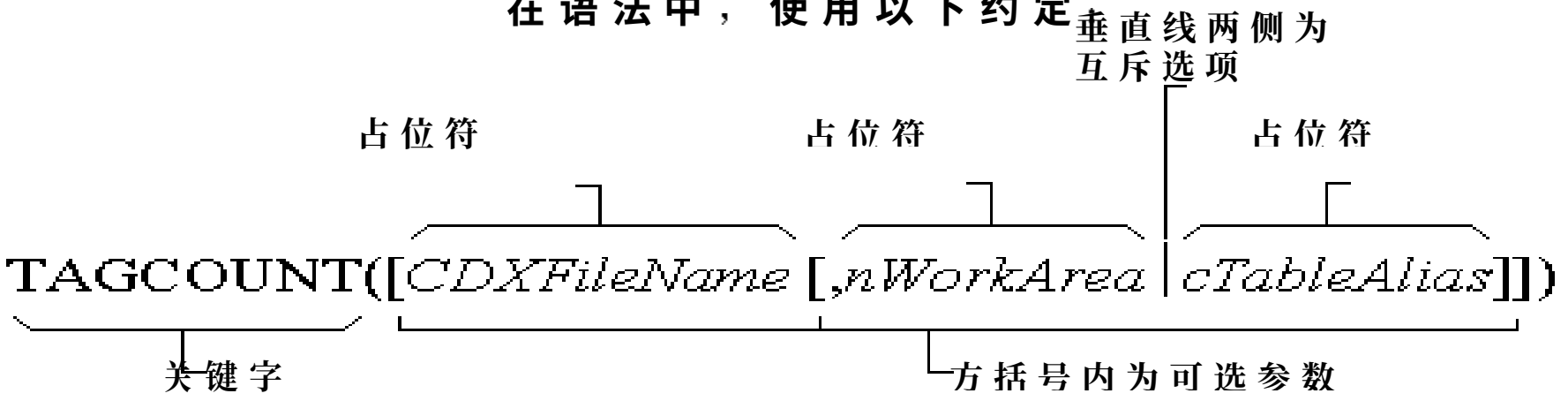
下面列出了在 Visual FoxPro 文档中使用的印刷约定：

示例	约定
setup	粗体表明必须键入的单词，而不是语言命令。
In the Query Designer toolbar, choose Add Table.	粗体也用在过程中，以突出显示界面元素，例如窗口、菜单、对话框、工具栏、按钮或选项的名称。
SET HELP TO	命令名、命令中的关键字、首字母缩略词、常量和设备名称等用大写字母表示。
Press the TAB key. Press SHIFT+F1.	大写字母表示键名。加号 (+) 表示组合键。
Buttons.vcx	开头字母大写表示文件名。
C:\My Computer\Working Files\Document	开头字母大写表示文件夹和目录。在路径中，文件夹、目录和文件名用反斜杠分隔。
http://www.microsoft.com/	小写字母用于 URL。服务程序、共享和文件名用斜杠分隔。
FontSize	开头字母大写表示对象、属性、事件和方法的名称。如果该名称多于一个单词，则后续单词的开头字母也大写。

续表

event-driven	斜体表明文本中第一次出现的定义术语。术语是产品词汇的一部分。单击斜体单词可以看到定义。
IF StatusText() = "Test" = MESSAGEBOX("OK") ENDIF	等宽字体表明需要键入的命令行、代码示例以及文本中对代码示例的引用。
USE customer	数据库、表和字段的名称用小写字母表示。
nTotal, cName	变量名和数组元素名的小写前缀表明它们的数据类型：*c* 代表字符型，*n* 代表数值型，*l* 代表逻辑型，*d* 表示日期型，*t* 表示时间型，*y* 表示货币型，*a* 表示数组，*e* 表示表达式，o 表示对象，u 表示未知。

在语法中，使用以下约定



示例	约定
DELETE VIEW ViewName	在语法项中，斜体单词为占位符。
[STYLE cStyleName]	在语法项中，方括号所括的是可选项。
SET BELL ON|OFF	在语法项中，由垂直线所隔开的两个选项中只能包含一个。
[, WindowName2 ...]	在语法项中，省略号表示列表中可包含任意数目的参数。各参数间用逗号隔开。

目　录

@...BOX 命令

@...CLASS 命令

@...CLEAR 命令

@...EDIT 编辑框命令

@...FILL 命令

@...GET-复选框命令

@...GET-组合框命令

@...GET- 命令按钮命令

@...GET-列表框命令

@...GET-选项按钮命令

@...GET-微调控件命令

@...GET-文本框命令

@...GET-透明按钮命令

@...MENU 命令

@...PROMPT 命令

@...SAY 命令

@...SAY-图片和 OLE 对象命令

@...SCROLL 命令

@...TO 命令

ABS( ) 函数

ACCEPT 命令

ACLASS( ) 函数

ACOPY( ) 函数

ACOS( ) 函数

Activate 事件

ACTIVATE MENU 命令

ACTIVATE POPUP 命令

ACTIVATE SCREEN 命令

ACTIVATE WINDOW 命令

ActivateCell 方法

ActiveColumn 属性

ActiveControl 属性

ActiveDoc 对象

ActiveForm 属性

ActivePage 属性

ActiveProject 属性

ActiveRow 属性

ADATABASES( ) 函数

ADBOBJECTS( ) 函数

Add 方法

ADD CLASS 命令

ADD TABLE 命令

ADDBS( ) 函数

AddColumn 方法

AddItem 方法

AddListItem 方法

AddObject 方法

AddProperty 方法

AddToSCC 方法

ADEL( ) 函数

ADIR( ) 函数

AELEMENT( ) 函数

AERROR( ) 函数

AFIELDS( ) 函数

AFONT( ) 函数

AfterBuild 事件

AfterCloseTables 事件

AfterDock 事件

AfterRowColChange 事件

AGETCLASS( ) 函数

AGETFILEVERSION( ) 函数

AINS( ) 函数

AINSTANCE( ) 函 数

ALEN( ) 函 数

Alias 属 性

ALIAS( ) 函 数

Align 属 性

Alignment 属 性

_ALIGNMENT 系 统 变 量

ALINES( ) 函 数

AllowAddNew 属 性

AllowHeaderSizing 属 性

AllowRowSizing 属 性

AllowTabs 属 性

ALLTRIM( ) 函 数

ALTER TABLE- SQL 命 令

AlwaysOnTop 属 性

AMEMBERS( ) 函 数

ANETRESOURCES( ) 函 数

AMOUSEOBJ( ) 函 数

ANSITOOEM( ) 函 数

APPEND 命 令

APPEND FROM 命 令

APPEND FROM ARRAY 命 令

APPEND GENERAL 命 令

APPEND MEMO 命 令

APPEND PROCEDURES 命 令

Application 对 象

Application 属 性

APRINTERS( ) 函 数

ASC( ) 函 数

ASCAN( ) 函 数

_ASCIICOLS 系 统 变 量

_ASCIIROWS 系 统 变 量

ASELOBJ( ) 函 数

ASIN( ) 函 数

ASORT( ) 函 数

ASSERT 命 令

ASSIST 命 令

_ASSIST 系 统 变 量

ASUBSCRIPT( ) 函 数

AT( ) 函 数

AT_C( ) 函 数

ATAN( ) 函 数

ATC( ) 函 数

ATCC( ) 函 数

ATCLINE( ) 函 数

ATLINE( ) 函 数

ATN2( ) 函 数

AUSED( ) 函 数

CAPSLOCK( ) 函 数

Caption 属 性

CD 或 CHDIR 命 令

CDOW( ) 函 数

CDX( ) 函 数

CEILING( ) 函 数

Century 属 性

CHANGE 命 令

CheckBox 控 件

CheckIn 方 法

CheckOut 方 法

ChildAlias 属 性

ChildOrder 属 性

CHR( ) 函 数

CHRSAW( ) 函 数

CHRTRAN( ) 函 数

CloseTables 方法

Cls 方法

CLSID 属性

CMONTH( ) 函数

CNTBAR( ) 函数

CNTPAD( ) 函数

CodePage 属性

COL( ) 函数

颜色概述

颜色术语

ColorScheme 属性

ColorSource 属性

Column 对象

ColumnCount 属性

ColumnLines 属性

ColumnOrder 属性

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

CREATE SQL VIEW 命令

CREATE TABLE- SQL 命令

CREATE TRIGGER 命令

CREATE VIEW 命令

CREATEBINARY( ) 函 数

CREATE OBJECT( ) 函 数

CREATEOBJECTEX( ) 函 数

CREATEOFFLINE( ) 函 数

CTOD( ) 函 数

CTOBIN( ) 函 数

CTOT( ) 函 数

CURDIR( ) 函 数

_CUROBJ 系 统 变 量

CurrentControl 属 性

CurrentX, CurrentY 属 性

Cursor 对 象

CURSORGETPROP( ) 函 数

CURSORSETPROP( ) 函 数

CursorSource 属 性

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

DBF( ) 函数

DoCmd 方法

DODEFAULT( ) 函数

DOEVENTS 命令

DO FORM 命令

DO WHILE...ENDDO 命令

Dock 方法

Docked 属性

DockPosition 属性

DocumentFile 属性

_DOS 系统变量

DoScroll 方法

DoVerb 方法

DOW( ) 函数

DownClick 事件

DownPicture 属性

Drag 方法

EXTERNAL 命令

FCHSIZE( ) 函数

FCLOSE( ) 函数

FCOUNT( ) 函数

FCREATE( ) 函数

FDATE( ) 函数

FEOF( ) 函数

FERROR( ) 函数

FFLUSH( ) 函数

FGETS( ) 函数

FIELD( ) 函数

FILE( ) 函数

FileClass 属性

FileClassLibrary 属性

File 对象

文件集合（Files Collection）

字 体 函 数

FontBold, FontItalic, FontStrikethru, FontUnderline 属 性

FontCondense, FontExtend 属 性

FONTMETRIC( ) 函 数

FontName 属 性

FontOutline 属 性

FontShadow 属 性

FontSize 属 性

FOPEN( ) 函 数

FOR...ENDFOR 命 令

FOR EACH ... ENDFOR 命 令

FOR( ) 函 数

FORCEEXT( ) 函 数

FORCEPATH( ) 函 数

Form 对 象

Format 属 性

GETFILE( ) 函 数

GETFLDSTATE( ) 函 数

GETFONT( ) 函 数

GetFormat 方 法

GETHOST( ) 函 数

GetLatestVersion 方 法

GETNEXTMODIFIED( ) 函 数

GETOBJECT( ) 函 数

GETPAD( ) 函 数

GETPEM( ) 函 数

GETPICT( ) 函 数

GETPRINTER( ) 函 数

GO 或 GOTO 命 令

GoBack 方 法

GoForward 方 法

GOMONTH( ) 函 数

GotFocus 事件

Grid 控件

GridHitTest 方法

GridLineColor 属性

GridLines 属性

GridLineWidth 属性

HalfHeightCaption 属性

Header 对象

HEADER( ) 函数

HeaderHeight 属性

Height 属性

HELP 命令

Help 方法

HelpContextID 属性

HIDE MENU 命令

Hide 方法

ISFLOCKED( ) 函 数

ISHOSTED( ) 函 数

ISLEADBYTE( ) 函 数

ISLOWER( ) 函 数

ISMOUSE( ) 函 数

ISNULL( ) 函 数

ISREADONLY( ) 函 数

ISRLOCKED( ) 函 数

ISUPPER( ) 函 数

Item 方 法

ItemBackColor, ItemForeColor 属 性

ItemData 属 性

ItemIDData 属 性

ItemIDToIndex 方 法

ItemTips 属 性

JOIN 命 令

JUSTDRIVE( ) 函 数

JUSTEXT( ) 函 数

JUSTFNAME( ) 函 数

JUSTPATH( ) 函 数

JUSTSTEM( ) 函 数

KEY( ) 函 数

KEYBOARD 命 令

KeyboardHighValue, KeyboardLowValue 属 性

KEYMATCH( ) 函 数

KeyPress 事 件

KeyPreview 属 性

LABEL 命 令

Label 控 件

LASTKEY( ) 函 数

LastModified 属 性

Left 属 性

LEFT( ) 函 数

LEFTC( ) 函 数

LeftColumn 属 性

LEN( ) 函 数

LENC( ) 函 数

LIKE( ) 函 数

LIKEC( ) 函 数

Line 控 件

Line 方 法

LINENO( ) 函 数

LineSlant 属 性

LinkMaster 属 性

LIST 命 令

LIST CONNECTIONS 命 令

LIST DATABASE 命 令

LIST DLLS 命 令

LIST OBJECTS 命令

LIST PROCEDURES 命令

List 属性

LIST TABLES 命令

LIST VIEWS 命令

ListBox 控件

ListCount 属性

ListIndex 属性

ListItem 属性

ListItemID 属性

_LMARGIN 系统变量

LOAD 命令

Load 事件

LOADPICTURE( ) 函数

LOCAL 命令

LOCATE 命令

LOCFILE( ) 函 数

LOCK( ) 函 数

LockScreen 属 性

LOG( ) 函 数

LOG10( ) 函 数

LOOKUP( ) 函 数

LostFocus 事 件

LOWER( ) 函 数

LPARAMETERS 命 令

LTRIM( ) 函 数

LUPDATE( ) 函 数

_MAC 系 统 变 量

MacDesktop 属 性

MainClass 属 性

MainFile 属 性

Margin 属 性

MAX( ) 函 数

MaxButton 属 性

MaxHeight 属 性

MaxLeft 属 性

MaxLength 属 性

MaxTop 属 性

MaxWidth 属 性

MCOL( ) 函 数

MD 或 MKDIR 命 令

MDIForm 属 性

MDOWN( ) 函 数

MDX( ) 函 数

MDY( ) 函 数

MEMLINES( ) 函 数

MEMORY( ) 函 数

MemoWindow 属 性

MENU 命令

MENU( ) 函数

MENU TO 命令

Message 事件

MESSAGE( ) 函数

MESSAGEBOX( ) 函数

MiddleClick 事件

MIN( ) 函数

MinButton 属性

MinHeight 属性

MINUTE( ) 函数

MinWidth 属性

MLINE( ) 函数

_MLINE 系统变量

MOD( ) 函数

Modify 方法

MODIFY CLASS 命令

MODIFY COMMAND 命令

MODIFY CONNECTION 命令

MODIFY DATABASE 命令

MODIFY FILE 命令

MODIFY FORM 命令

MODIFY GENERAL 命令

MODIFY LABEL 命令

MODIFY MEMO 命令

MODIFY MENU 命令

MODIFY PROCEDURE 命令

MODIFY PROJECT 命令

MODIFY QUERY 命令

MODIFY REPORT 命令

MODIFY SCREEN 命令

MODIFY STRUCTURE 命令

MODIFY VIEW 命令

MODIFY WINDOW 命令

MONTH( ) 函数

MOUSE 命令

MouseDown 事件

MouseIcon 属性

MouseMove 事件

MousePointer 属性

MouseUp 事件

MouseWheel 事件

Movable 属性

Move 方法

MOVE POPUP 命令

MOVE WINDOW 命令

Moved 事件

MoverBars 属性

NTOM( ) 函 数

NullDisplay 属 性

NumberOfElements 属 性

NUMLOCK( ) 函 数

NVL( ) 函 数

Object 属 性

对 象 集 合 （Object Collection）

OBJNUM( ) 函 数

OBJTOCLIENT( ) 函 数

OBJVAR( ) 函 数

OCCURS( ) 函 数

OEMTOANSI( ) 函 数

OLDVAL( ) 函 数

OLE Bound 控 件

OLE Container 控 件

OLEClass 属 性

OLECompleteDrag 事件

OLEDrag 方法

OLEDragDrop 事件

OLEDragMode 属性

OLEDragOver 事件

OLEDragPicture 属性

OLEDropEffects 属性

OLEDropHasData 属性

OLEDropMode 属性

OLEDropTextInsertion 属性

OLEGiveFeedback 事件

OLELCID 属性

OLERequestPendingTimeout 属性

OLEServerBusyRaiseError 属性

OLEServerBusyTimeout 属性

OLESetData 事件

OLEStartDrag 事件

OLETypeAllowed 属性

ON BAR 命令

ON ERROR 命令

ON ESCAPE 命令

ON EXIT BAR 命令

ON EXIT MENU 命令

ON EXIT PAD 命令

ON EXIT POPUP 命令

ON KEY 命令

ON KEY = 命令

ON KEY LABEL 命令

ON PAD 命令

ON PAGE 命令

ON READERROR 命令

ON SELECTION BAR 命令

ON SELECTION MENU 命令

ON SELECTION PAD 命令

ON SELECTION POPUP 命令

ON SHUTDOWN 命令

ON( ) 函数

OneToMany 属性

OPEN DATABASE 命令

OpenTables 方法

OpenViews 属性

OpenWindow 属性

OptionButton 控件

OptionGroup 控件

Order 属性

ORDER( ) 函数

OS( ) 函数

PACK 命令

PACK DATABASE 命令

PAD( ) 函数

PADL( )|PADR( )|PADC( ) 函数

_PADVANCE 系统变量

Page 对象

PageCount 属性

PageFrame 控件

PageHeight 属性

_PAGENO 系统变量

PageOrder 属性

Pages 属性

PageWidth 属性

Paint 事件

Panel 属性

PanelLink 属性

PARAMETERS 命令

PEMSTATUS( ) 函数

_PEPAGE 系统变量

PI( ) 函数

Picture 属性

PLAY MACRO 命令

_PLENGTH 系统变量

_PLINENO 系统变量

_PLOFFSET 系统变量

Point 方法

POP KEY 命令

POP MENU 命令

POP POPUP 命令

POPUP( ) 函数

_PPITCH 系统变量

_PQUALITY 系统变量

_PRETEXT 系统变量

PRIMARY( ) 函 数

Print 方 法

PRINTJOB...ENDPRINTJOB 命 令

PRINTSTATUS( ) 函 数

PRIVATE 命 令

PRMBAR( ) 函 数

PRMPAD( ) 函 数

PROCEDURE 命 令

ProgID 属 性

PROGRAM( ) 函 数

ProgrammaticChange 事 件

Projects 集 合

Project 对 象

ProjectHook 对 象

ProjectHook 属 性

ProjectHookClass 属 性

ProjectHookLibrary 属性

PROMPT( ) 函数

PROPER( ) 函数

PROW( ) 函数

PRTINFO( ) 函数

_PSCODE 系统变量

PSet 方法

_PSPACING 系统变量

PUBLIC 命令

PUSH KEY 命令

PUSH MENU 命令

PUSH POPUP 命令

PUTFILE( ) 函数

PV( ) 函数

_PWAIT 系统变量

QueryAddFile 事件

QueryModifyFile 事件

QueryRemoveFile 事件

QueryRunFile 事件

QueryUnload 事件

QUIT 命令

Quit 方法

RAND( ) 函数

RangeHigh 事件

RangeLow 事件

RAT( ) 函数

RATC( ) 函数

RATLINE( ) 函数

RD|RMDIR 命令

RDLEVEL( ) 函数

READ EVENTS 命令

READ 命令

READ MENU 命令

ReadActivate 事件

ReadBackColor, ReadForeColor 属性

ReadCycle 属性

ReadDeactivate 事件

ReadExpression 方法

READKEY( ) 函数

ReadLock 属性

ReadMethod 方法

ReadMouse 属性

ReadObject 属性

ReadOnly 属性

ReadSave 属性

ReadShow 事件

ReadTimeout 属性

ReadValid 事件

RelativeRow 属性

RELEASE 命令

RELEASE BAR 命令

RELEASE CLASSLIB 命令

RELEASE LIBRARY 命令

RELEASE MENUS 命令

Release 方法

RELEASE PAD 命令

RELEASE POPUPS 命令

RELEASE PROCEDURE 命令

RELEASE WINDOWS 命令

ReleaseType 属性

Remove 方法

RemoveFromSCC 方法

REMOVE CLASS 命令

REMOVE TABLE 命令

RemoveItem 方法

RemoveListItem 方法

RemoveObject 方法

RENAME 命令

RENAME CLASS 命令

RENAME CONNECTION 命令

RENAME TABLE 命令

RENAME VIEW 命令

REPLACE 命令

REPLACE FROM ARRAY 命令

REPLICATE( ) 函数

REPORT 命令

Requery 方法

REQUERY( ) 函数

RequestData 方法

Reset 方法

RLOCK( ) 函 数

_RMARGIN 系 统 变 量

ROLLBACK 命 令

ROUND( ) 函 数

ROW( ) 函 数

RowHeight 属 性

RowSource 属 性

RowSourceType 属 性

RTOD( ) 函 数

RTRIM( ) 函 数

RUN|! 命 令

Run 事 件

Run 方 法

_RUNACTIVEDOC 系 统 变 量

_SAMPLES 系 统 变 量

SAVE MACROS 命 令

SAVE SCREEN 命令

SAVE TO 命令

SAVE WINDOWS 命令

SaveAs 方法

SaveAsClass 方法

SAVEPICTURE( ) 函数

ScaleMode 属性

SCAN...ENDSCAN 命令

SCATTER 命令

SCCProvider 属性

SCCStatus 属性

_SCCTEXT 系统变量

SCHEME( ) 函数

SCOLS( ) 函数

_SCREEN 系统变量

SCROLL 命令

SelStart 属性

SelText 属性

Separator 对象

Server 对象

ServerClass 属性

ServerClassLibrary 属性

ServerHelpFile 属性

ServerName 属性

ServerProject 属性

Servers Collection

SET 命令

SET ALTERNATE 命令

SET ANSI 命令

SET ASSERTS 命令

SET AUTOSAVE 命令

SET BLOCKSIZE 命令

SET BORDER 命令

SET BROWSEIME 命令

SET BRSTATUS 命令

SET CARRY 命令

SET CENTURY 命令

SET CLASSLIB 命令

SET CLEAR 命令

SET CLOCK 命令

SET COLLATE 命令

SET COLOR OF 命令

SET COLOR OF SCHEME 命令

SET COLOR SET 命令

SET COLOR TO 命令

SET COMPATIBLE 命令

SET CONFIRM 命令

SET CONSOLE 命令

SET COVERAGE 命令

SET CPCOMPILE 命令

SET CPDIALOG 命令

SET CURRENCY 命令

SET CURSOR 命令

SET DATABASE 命令

SET DATASESSION 命令

SET DATE 命令

SET DEBUG 命令

SET DEBUGOUT 命令

SET DECIMALS 命令

SET DEFAULT 命令

SET DELETED 命令

SET DELIMITERS 命令

SET DEVELOPMENT 命令

SET DEVICE 命令

SET DISPLAY 命令

SET DOHISTORY 命令

SetData 方法

SET ECHO 命令

SET ESCAPE 命令

SET EVENTLIST 命令

SET EVENTTRACKING 命令

SET EXACT 命令

SET EXCLUSIVE 命令

SET FDOW 命令

SET FIELDS 命令

SET FILTER 命令

SET FIXED 命令

SET FORMAT 命令

SET FULLPATH 命令

SET FUNCTION 命令

SET FWEEK 命令

SetFormat 方法

SET HEADINGS 命令

SET HELP 命令

SET HELPFILTER 命令

SET HOURS 命令

SET INDEX 命令

SET INTENSITY 命令

SET KEY 命令

SET KEYCOMP 命令

SET LIBRARY 命令

SET LOCK 命令

SET LOGERRORS 命令

SET MACKEY 命令

SET MARGIN 命令

SET MARK OF 命令

SET MARK TO 命令

SET MEMOWIDTH 命令

SET MESSAGE 命令

SET MULTILOCKS 命令

SET NEAR 命令

SET NOCPTRANS 命令

SET NOTIFY 命令

SET NULL 命令

SET NULLDISPLAY 命令

SET ODOMETER 命令

SET OLEOBJECT 命令

SET OPTIMIZE 命令

SET ORDER 命令

SET PALETTE 命令

SET PATH 命令

SET PDSETUP 命令

SET POINT 命令

SET PRINTER 命令

SET PROCEDURE 命令

SET READBORDER 命令

SET REFRESH 命令

SET RELATION 命令

SET RELATION OFF 命令

SET REPROCESS 命令

SET RESOURCE 命令

SET SAFETY 命令

SET SECONDS 命令

SET SEPARATOR 命令

SET SKIP 命令

SET SKIP OF 命令

SET SPACE 命令

SET STATUS 命令

SET STATUS BAR 命令

SET STEP 命令

SET STRICTDATE 命令

SET SYSFORMATS 命令

SET SYSMENU 命令

SET TALK 命令

SET TEXTMERGE 命令

SET TEXTMERGE DELIMITERS 命令

SET TOPIC 命令

SET TOPIC ID 命令

SET TRBETWEEN 命令

SET TYPEAHEAD 命令

SET UDFPARMS 命令

SET UNIQUE 命令

SET VIEW 命令

SET WINDOW OF MEMO 命令

SET( ) 函数

SetAll 方法

SETFLDSTATE( ) 函数

SetFocus 方法

SetMain 方法

SetVar 方法 1

SetViewPort 方法

Shape 控件

_SHELL 系统变量

SHOW MENU 命令

Show 方法

SHOW OBJECT 命令

SHOW POPUP 命令

SHOW WINDOW 命令

ShowDoc 事件

ShowTips 属性

SQLSETPROP( ) 函 数

SQLSTRINGCONNECT( ) 函 数

SQLTABLES( ) 函 数

SQRT( ) 函 数

SROWS( ) 函 数

StartMode 属 性

_STARTUP 系 统 变 量

StatusBar 属 性

StatusBarText 属 性

STORE 命 令

STR( ) 函 数

STRCONV( ) 函 数

Stretch 属 性

StrictDateEntry 属 性

STRTOFILE( ) 函 数

STRTRAN( ) 函 数

STUFF( ) 函 数

STUFFC( ) 函 数

Style 属 性

SUBSTR( ) 函 数

SUBSTRC( ) 函 数

SUM 命 令

SUSPEND 命 令

SYS( ) 函 数 概 述

SYS(0)-网 络 机 器 信 息

SYS(1)-儒 略 (Julian)系 统 日 期

SYS(2)-自 午 夜 开 始 以 秒 计 的 时 间

SYS(3)-合 法 的 文 件 名

SYS(5)-默 认 驱 动 器

SYS(6)-当 前 打 印 设 备

SYS(7)-当 前 格 式 文 件

SYS(9)-Visual FoxPro 系 列 号

SYS(10)-来自儒略(Julian)日期的字符串

SYS(11)-儒略(Julian)日期

SYS(12)-可用内存的字节数

SYS(13)-打印机状态

SYS(14)-索引表达式

SYS(15)-字符转换

SYS(16)-执行程序文件名

SYS(17)-正在使用的处理器

SYS(18)- 当前控件

SYS(20) - 转换德文文本

SYS(21)- 控制索引编号

SYS(22)-控制标志或索引名

SYS(23)-Visual FoxPro 的 EMS 内存用法

SYS(24)- EMS 内存限制

SYS(100)-控制台设置

SYS(101)- 设备设置

SYS(102)- 打印机设置

SYS(103)-对话设置

SYS(1001)- Visual FoxPro 内存

SYS(1016)-用户对象的内存使用

SYS(1023)- 启用诊断帮助模式

SYS(1024)- 终止诊断帮助模式

SYS(1037)-“页面设置”对话框

SYS(1269)- 属性信息

SYS(1270)- 对象位置

SYS(1271)- 对象的 SCX 文件

SYS(1272)- 对象层次

SYS(1500)-激活系统菜单项

SYS(2000)-文件名通配符匹配

SYS(2001)- SET 命令状态

SYS(2002)-打开或关闭插入点

SYS(2003)-当前目录

SYS(2004)- Visual FoxPro 启动目录或文件夹

SYS(2005)- 当前资源文件

SYS(2006)- 当前图形适配卡

SYS(2007)- 检查求和值

SYS(2010)- CONFIG.SYS 文件设置

SYS(2011)- 当前锁定状态

SYS(2012)-备注字段块大小

SYS(2013)-系统菜单名称字符串

SYS(2014)- 最小化路径

SYS(2015)-唯一过程名

SYS(2016)-SHOW GETS WINDOW 名称

SYS(2017)- 显示启动屏幕

SYS(2018)- 错误信息参数

SYS(2019)-配置文件名称和位置

SYS(2020)- 默认磁盘空间

SYS(2021)-筛选表达式

SYS(2022)-磁盘簇（块）大小

SYS(2023)-临时文件驱动器

SYS(2029)-表类型

SYS(2333)-开启或关闭 ActiveX 的双界面支持

SYS(2334)- 自动服务启动

SYS(2335)-无人照管服务模式

SYS(3004)-返回环境 ID 值

SYS(3005)-设置环境 ID 值

SYS(3006)-设置语言和环境 ID 值

SYS(3050)-设置缓冲内存大小

SYS(3051)- 设定锁定重试间隔

SYS(3052)-忽略 SET REPROCESS 锁定

SYS(3053)- ODBC 环境句柄

SYS(3054)-显示 Rushmore 优化的级别

SYS(3055)- FOR and WHERE 子句复杂级别

SYS(3056)- 读取注册设置

SYS(4204)- Active Document 调试

SYSMETRIC( ) 函数

系统变量概述

System Menu Names

TabIndex 属性

TABLEREVERT( ) 函数

TABLEUPDATE( ) 函数

Tabs 属性

_TABS 系统变量

TabStop 属性

TabStretch 属性

TabStyle 属性

Tag 属性

TAG( ) 函数

TAGCOUNT( ) 函数

TAGNO( ) 函数

TYPE 命令

Type 属性

TYPE( ) 函数

TypeLibCLSID 属性

TypeLibDesc 属性

TypeLibName 属性

UIEnable 事件

UndoCheckOut 方法

UnDock 事件

UNIQUE( ) 函数

_UNIX 系统变量

Unload 事件

UNLOCK 命令

UpClick 事件

UPDATE- SQL 命令

UPDATE 命令

UPDATED( ) 函 数

UPPER( ) 函 数

USE 命 令

USED( ) 函 数

VAL( ) 函 数

VARTYPE( ) 函 数

Valid 事 件

VALIDATE DATABASE 命 令

Value 属 性

VARREAD( ) 函 数

VERSION( ) 函 数

Version 属 性

VersionComments 属 性

VersionCompany 属 性

VersionCopyright 属 性

VersionDescription 属 性

VersionLanguage 属性

VersionNumber 属性

VersionProduct 属性

VersionTrademarks 属性

_VFP 系统变量

View 属性

ViewPortHeight 属性

ViewPortLeft 属性

ViewPortTop 属性

ViewPortWidth 属性

Visible 属性

VScrollSmallChange 属性

WAIT 命令

WBORDER( ) 函数

WCHILD( ) 函数

WCOLS( ) 函数

WLCOL( ) 函 数

WLROW( ) 函 数

WMAXIMUM( ) 函 数

WMINIMUM( ) 函 数

WONTOP( ) 函 数

WordWrap 属 性

WOUTPUT( ) 函 数

WPARENT( ) 函 数

_WRAP 系 统 变 量

WREAD( ) 函 数

WriteExpression 方 法

WriteMethod 方 法

WROWS( ) 函 数

WTITLE( ) 函 数

WVISIBLE( ) 函 数

YEAR( ) 函 数

ZAP 命令

ZoomBox 属性

ZOOM WINDOW 命令

ZOrder 方法

返回总目录

#DEFINE ……#UNDEF 预处理器命令
#IF……#ENDIF 预处理器命令
#IFDEF|#IFNDEF……#ENDIF 预处理器命令
#INCLUDE 预处理器命令
:: 作用域操作符
$ 操作符
% 操作符
& 命令
&& 命令
* 命令
= 命令
\ 或 \\ 命令
? 或 ?? 命令
??? 命令
@ ……BOX 命令
@ ……CLASS 命令
@ ……CLEAR 命令
@ ……EDIT 编辑框命令
@ ……FILL 命令

ACTIVATE POPUP 命令
ACTIVATE SCREEN 命令
ACTIVATE WINDOW 命令
ActivateCell 方法
ActiveColumn 属性
ActiveControl 属性
ActiveDoc 对象
ActiveForm 属性
ActivePage 属性
ActiveProject 属性
ActiveRow 属性
ADATABASES（） 函数
ADBOBJECTS（） 函数
Add 方法
ADD CLASS 命令
ADD TABLE 命令
ADDBS（） 函数
AddColumn 方法
AddItem 方法

#DEFINE .....#UNDEF 预处理器命令

创建和释放编译期间所用的常量。

语法

#DEFINE ConstantName eExpression

.....

#UNDEF ConstantName

参数描述

ConstantName

指定编译期间所用的常量名。常量名必须是合法的 Visual FoxPro 常量名，第一个字符为字母或下划线，其他字符可以是字母、数字或下划线，最长可达 254 个字符。为提高程序的可读性和简化调试过程，应对常量名使用大写字母并遵循标准命名约定。

重要提示 请不要使用 Visual FoxPro 关键字作常量名。

要取消用#DEFINE 定义的编译期间常量，可使用#UNDEF *ConstantName*。

eExpression

指定编译期间所用常量的值。*eExpression* 可以是一个名称或一表达式。表达式的值可以是字符、数值、货币、日期、日期时间或逻辑值等。

重要提示 请不要在 *eExpression* 中使用系统变量。系统变量只有在运行时才具有值。

说明

#DEFINE 和#UNDEF 预处理器命令在程序中用来定义和取消编译期间所用的常量。用#DEFINE 定义编译期间所用常量，同使用变量相比，能够减少内存消耗，简化程序并增强程序运行性能。

要使用#DEFINE 定义常量，应该用 *ConstantName* 给出常量名，并用 *eExpression* 指定它的值表达式。编译程序时将执行文本替换，把程序中任何出现常量名的地方都用常量表达式替换过来。在程序中添加#UNDEF 命令可以停止这种替换。

上述替换只在定义常量的#DEFINE 语句行与取消该常量的#UNDEF 语句行之间发生。常量只在定义该常量的程序中有效。

如果#DEFINE 语句放在表单的一个事件或方法过程中，则定义的编译期间常量只在该事件或方法的过程中有效。要使得#DEFINE 语句定义的编译常量对表单中所有事件和方法过程都有效，请从“表单”菜单中选择“包含文件”菜单项，并指定一个头文件来包含#DEFINE 编译期间常量。

请注意位于引号中的编译期间常量不予承认。

示例

下面的程序定义了一个名为 MAXITEMS 的编译期间常量。该常量在 FOR ..... NEXT 循

环中用来显示数值 1 到 10。

```
#DEFINE MAXITEMS 10
CLEAR
FOR gnCount = 1 TO MAXITEMS
   ? gnCount
NEXT
```

请参阅

COMPILE,#IF .....#ENDIF,#IFDEF |#IFNDEF .....#ENDIF

#IF.....#ENDIF 预处理器命令

在编译时，根据条件决定是否编译某段源代码。

语法

```
#IF nExpression1 | lExpression1
  Commands
[#ELIF nExpression2 |#ELIF lExpression2
  Commands
  .....
#ELIF nExpressionN |#ELIF lExpressionN
```

Commands]
[#ELSE
Commands]
#ENDIF

参数描述

#IF nExpression1 | lExpression1
Commands

nExpression1 给出要计算的数值表达式。

- 如果该表达式的值不为 0，则编译紧接在#IF 命令后面的程序。然后退出#IF .....#ENDIF 结构，接着编译#ENDIF 命令后面的程序。
- 如果该表达式的值为 0，则不编译紧接在#IF 命令后面的程序，而计算#ELIF 命令后的表达式。

lExpression1 给出要计算的逻辑表达式。

- 如果逻辑表达式的值为“真”(.T.)，则编译紧接在#IF 命令后面的程序。然后退出#IF .....#ENDIF 结构，接着编译#ENDIF 命令后面的程序。
- 如果逻辑表达式的值为“假”(.F.)，则不编译紧接在#IF 命令后面的程序，而计算#ELIF 命令后的表达式。

注意 请不要为 *nExpression1* 和 *lExpression1* 指定系统变量。因为系统变量只有在运行时才具有值。

#ELIF nExpression2 |#ELIF lExpression2
Commands
.....
#ELIF nExpressionN |#ELIF lExpressionN
Commands

如果表达式 *nExpression1* 的值为 0，或者表达式 *lExpression1* 的值为“假”(.F.)，则计算#ELIF 命令。如果存在 *nExpression2* 或 *lExpression2*，则计算#ELIF 命令中表达式的值。如果表达式 *nExpression2* 的值不为 0，或者表达式 *lExpression2* 的值为“真”，则编译在#ELIF 命令后的程序代码，然后退出#IF .....#ENDIF 结构，接着编译#ENDIF 命令后面的程序。

如果表达式 *nExpression2* 的值为 0，或者表达式 *lExpression2* 的值为“假”(.F.)，则不编译#ELIF 命令后的程序，而继续处理下一个#ELIF 命令。

#ELSE Commands

如果没有#ELIF 命令，或者全部#ELIF 命令的表达式计算结果都为 0 或都为“假”(.F.)，#ELSE 的存在与否将决定#IF .....#ENDIF 结构中是否有其他需要编译的程序。

- 如果没有#ELIF 命令，或者全部#ELIF 命令的表达式计算结果都为 0 或都为“假”(.F.)，#ELSE 的存在与否将决定#IF .....#ENDIF 结构中是否有其他需要编译的程序。
- 如果不存在#ELSE 语句，在#IF 和#ENDIF 之间则没有要编译的代码。退出#IF .....#ENDIF 结构后，编译从#ENDIF 后面的第一行程序继续进行。

#ENDIF

表示#IF 语句的结束。

说明

#IF .....#ENDIF 能够提高源代码的可读性，缩小编译后程序的大小，以及在某些情况下提高程序的运行性能。

编译#IF .....#ENDIF 结构时，首先计算该结构中的一系列逻辑表达式或数值表达式的值，这些表达式的计算结果将决定编译哪些 Visual FoxPro 程序（如果存在）。

示例

下面的示例中，#IF.....#ENDIF 结构决定了用何种版本的 Visual FoxPro 来编译程序并显示适当的信息。

```
#IF 'WINDOWS' $ UPPER(VERSION( ) )
    ? 'This was compiled under Visual FoxPro for Windows'

#ELIF 'MAC' $ UPPER(VERSION( ) )
    ? 'This was compiled under Visual FoxPro for Macintosh'

#ELIF 'UNIX' $ UPPER(VERSION( ) )
    ? 'This was compiled under FoxPro for UNIX'

#ELSE
    ? 'This was compiled under FoxPro for MS-DOS'

#ENDIF
```

请参阅

COMPILE,#DEFINE ……#UNDEF,#IFDEF |#IFNDEF ……#ENDIF

#IFDEF|#IFNDEF ……#ENDIF 预处理器命令

在编译期间，根据是否定义了某一个编译常量，决定一段代码是否要编译。

语法

```
#IFDEF |#IFNDEF ConstantName
  Commands
[#ELSE
  Commands]
#ENDIF
```

参数描述

#IFDEF

如果已定义由 *ConstantName* **指定的编译常量，则编译一段程序代码。**

下面描述如何根据#IFDEF **预处理器命令确定要编译的程序代码：**

- **如果定义了** *ConstantName*，**则要编译从**#IFDEF **到**#ELSE **或者**#ENDIF **（取决于哪个在前面）之间的程序代码。**

- 如果没有定义 *ConstantName*，但有#ELSE 命令，则编译#ELSE 到#ENDIF 之间的程序代码。
- 如果没有定义 *ConstantName*，也没有#ELSE 命令，则#IFDEF .....#ENDIF 结构中的所有程序都不编译。

#IFNDEF

当没有定义 *ConstantName* 指定的编译常量时，指定要编译的一组程序代码。

下面描述如何根据#IFNDEF 预处理器命令决定要编译的程序代码：

- 如果没有定义 *ConstantName*，则编译#IFNDEF 到#ELSE 或者#ENDIF（取决于哪个在前面）之间的程序代码。
- 如果已经定义了 *ConstantName*，并且有#ELSE 命令，则编译#ELSE 到#ENDIF 之间的程序代码。
- 如果定义了 *ConstantName*，但没有#ELSE 命令，则#IFNDEF .....#ENDIF 结构中的所有程序都不编译。

ConstantName

指定编译期间所用的常量。编译常量由#DEFINE 命令定义。

Commands

指定要编译的程序代码。

说明

一个#IFDEF |#IFNDEF .....#ENDIF 结构可以嵌套另一个#IFDEF |#IFNDEF .....#ENDIF 结构。

注释可以放在#IFDEF、#IFNDEF、#ELSE 和#ENDIF 所在行的后面。这些注释在编译和程序运行期间将被忽略。

示例

下面的示例创建了一个名称为 MYDEFINE 的编译时期常量。如果编译时期常量已经定义，#IFDEF……#ENDIF 会显示信息。

```
#DEFINE MYDEFINE 1

#IFDEF MYDEFINE
    WAIT WINDOW "MYDEFINE exists"

#ELSE
    WAIT WINDOW "MYDEFINE does not exist"

#ENDIF
```

请参阅

COMPILE,#DEFINE ……#UNDEF,#IF……#ENDIF ,#INCLUDE

#INCLUDE 预处理器命令

告诉 Visual FoxPro 预处理器去处理指定头文件的内容，就好像头文件中的内容出现在

Visual FoxPro 程序中一样。

语法

#INCLUDE FileName

参数描述

FileName

指定头文件的文件名。头文件在编译时将合并到程序中去。

头文件名中可以包含路径。当头文件名中包含路径时，Visual FoxPro 只在指定路径里寻找头文件。

如果头文件名中不包含路径，则 Visual FoxPro 首先在默认的 Visual FoxPro 目录中寻找头文件，然后沿着 Visual FoxPro 路径寻找。Visual FoxPro 路径用 SET PATH 指定。

说明

您可以创建包含预处理器命令的头文件，然后在编译程序时用#INCLUDE 命令把它的内容合并到程序中去。编译时，头文件的内容放在程序中#INCLUDE 命令出现的位置。

头文件中，只有#DEFINE .....#UNDEF 和#IF .....#ENDIF 预处理器命令有效。头文件中的注释和 Visual FoxPro 命令无效。

一个程序可以包含任意多个#INCLUDE 命令，并且这些命令可以出现在程序中的任何位置。#INCLUDE 命令也可以出现在头文件中。Visual FoxPro 允许嵌套#INCLUDE 命令。

头文件一般具有 .H 扩展名，但是也可以带其他扩展名。Visual FoxPro 提供了一个头文

件 FOXPRO.H，它包含了本文档中所提到的许多常量。

示例

下面的示例中使用了两个文件：头文件 Const.h 与程序文件 Myprog.prg 。头文件包含了几个创建编译期间常量的 #DEFINE 命令。程序文件用 #INCLUDE 在编译时合并 Const.h 头文件，并且程序可使用头文件中的编译期间常量。

```
*** Header file CONST.H ***
#DEFINE ERROR_NODISK    1
#DEFINE ERROR_DISKFULL  2
#DEFINE ERROR_UNKNOWN  3

*** Program file MYPROG.PRG ***

#INCLUDE CONST.H

FUNCTION chkerror
PARAMETER errcode
    DO CASE
    CASE errcode = ERROR_NODISK
    ?"Error - No Disk"
    CASE errcode = ERROR_DISKFULL
    ?"Error - Disk Full"
    CASE errcode = ERROR_UNKNOWN
    ?"Unknown Error"
    ENDCASE
RETURN
```

请参阅

#DEFINE ……#UNDEF,#IF ……#ENDIF ,#IFDEF |#IFNDEF ……#ENDIF ,
_INCLUDE

:: 作用域操作符

在子类方法中运行父类的方法。

语法

cClassName::cMethod

说明

操作符 :: 从子类方法中执行父类的方法。在创建子类时，子类方法可以通过自动继承父类方法来得到。操作符 :: 使您在子类方法中执行完父类方法后，还能执行子类方法的其他一些操作。

有关作用域操作符 :: 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”。

示例

下面的示例创建了一个表单并添加了两个命令按钮。单击任意按钮，可以退出表单——第二个按钮 cmdAnotherButton 从 cmdQuit 调用 Click 过程。因为是子类，此操作是可行的。作用域操作符为子类对象调用父类代码。

```
frmMyForm = CREATEOBJECT("Form")
frmMyForm.Width  = 450
frmMyForm.Height  = 100
frmMyForm.Caption  = "Scope Resolution Example"
frmMyForm.AutoCenter =.T.
frmMyForm.AddObject("cmdQuit","cmdQuitButton")
frmMyForm.AddObject("cmdAnother","cmdAnotherButton")
frmMyForm.SHOW     && 显示表单
READ EVENTS      && 启动事件程序
```

下面的示例定义了两个命令按钮。第二个按钮作为第一个按钮的子类。为 cmdQuit 定义的 FontBold 与 ForeColor 属性可以使用于此子类，而在 cmdAnotherButton 则没有清楚的设置。我们将 cmdAnotherButton 定义为 cmdQuitButton 的子类。这样，该按钮会获得为 cmdQuitButton 定义的所有属性。

```
DEFINE CLASS cmdQuitButton AS CommandButton
    Caption  = "\<Quit"  && 给命令按钮增加说明
    Left  = 175   && 按钮左边距
    Top  = 60   && 按钮顶端位置
    Height  = 25   && 按钮高度
    Visible  = .T.   && 显示表单中的按钮
    FontItalic = .T.   && 设置斜体字
    ForeColor  = RGB(0,0,255)   && 改变按钮字体颜色

    PROCEDURE Click
    WAIT WINDOW "Executing the CLICK procedure for cmdQuit." TIMEOUT 1
    CLEAR EVENTS   && 终止事件程序，关闭表单
ENDDEFINE
DEFINE CLASS cmdAnotherButton AS cmdQuitButton
```

```
    Caption = "Click to quit"
    Left  = 175
    Top   = 30
    Height  = 25

    PROCEDURE Click
    WAIT WINDOW "Click event for button: cmdAnotherButton" TIMEOUT 1
    cmdQuitButton::Click
ENDDEFINE
```

请参阅

ADD CLASS, CREATE CLASS, CREATE CLASSLIB, CREATEOBJECT（）, DEFINE CLASS, DODEFAULT（）, GETOBJECT（）, MODIFY CLASS, RELEASE CLASSLIB, SET CLASSLIB, WITH ..... ENDWITH

$ 操作符

如果一个字符表达式包含在另一个字符表达式中，则返回“真”(.T.)；**否则，返回“假”**(.F.)。

语法

cSearchFor $ cSearchIn

返值类型

逻辑型

参数描述

cSearchFor

指定要在 *cSearchIn* 中查找的字符表达式。

cSearchIn

指定要在其中查找包含 *cSearchFor* 的字符表达式。

如果在 *cSearchIn* 中找到了 *cSearchFor*，则 $ 返回“真”(.T.)；否则，返回“假”(.F.)。*cSearchIn* 和 *cSearchFor* 可以是字符型变量或数组元素、字符型字段、原义字符串或任意长度的备注型字段。

备注型字段可以像字符表达式、表中字段、变量以及数组元素一样操作。例如，如果 MEMO_FLD 是一个备注型字段，则下面的语句有效：

```
LIST FOR 'FOX' $ UPPER(memo_fld)
```

说明

如果没有找到字符表达式，则返回“假”(.F.)。操作符 $ 区分大小写，但不能进行 Rushmore™ 优化。

示例

下面的示例创建了包含一个备注字段、名称为 memotest 的表。表中添加了三个记

录。LIST 用于显示这三个记录。可用美元符号($)来列出包含字符串“FOX.”的记录。然后为此示例而创建的文件将被删除。

```
CLOSE DATABASES
CLEAR
CREATE TABLE memotest (Text C(3), Memo M)
INSERT INTO  memotest (Text, Memo) VALUES ('Fox', 'Fox')
INSERT INTO  memotest (Text, Memo) VALUES ('Cat', 'Cat')
INSERT INTO  memotest (Text, Memo) VALUES ('FOX', 'FOX')
LIST FIELDS  Memo, Text FOR 'FOX' $ UPPER(Memo)
USE
DELETE FILE memotest.dbf
DELETE FILE memotest.fpt
```

请参阅

AT（）

% 操作符

返回两个数值表达式相除的余数。

语法

nDividend % nDivisor

参数描述

nDividend

指定被除数（被除数 *nDividend* 的数值表达式）。*nDividend* 中的小数位数决定结果数值的小数位数。

nDivisor

指定除数（被除数 *nDivisor* 的数值表达式）。当 *nDivisor* 为正数时，返回值为正数；当 *nDivisor* 为负数时，返回值为负数。除数 *nDivisor* 不能为零。

说明

取余操作符 (%) 和 MOD（） 返回相同的结果。

取余操作符 (%) 是一个算术操作符。算术操作符还有：+（加法）、-（减法）、*（乘法）、/（除法）和 ^（乘幂）。当数值表达式中包含这些操作符时，% 和 *、/ 的优先级相同。

为了进一步了解操作符和他们的优先级，请参阅“帮助”中的“操作符”。

示例

```
? 36 % 10       && 显示 6
? (4*9) % (90/9)     && 显示 6
? 25.250 % 5.0     && 显示 0.250
? IIF(YEAR(DATE（）) % 4 = 0, '今年是闰年';
    '今年不是闰年')
```

请参阅

& 命令

执行宏替换。

语法

& VarName[.cExpression]

参数描述

& VarName

指定宏替换中引用的变量名或数组元素名。请不要加上用于区分变量与字段的前缀“M.”，否则将产生语法错误。宏的长度不要超过 Visual FoxPro **中允许的最大语句长度。**

在宏替换中，变量不能递归引用自身。例如，下列语句将产生错误信息：

```
STORE '&gcX' TO gcX
? &gcX
```

出现在 DO WHILE、FOR **和** SCAN **中的宏替换语句只在循环开始时计算值，在后续的循环中则不再计算值。因此在循环内改变变量和数组元素的值对宏替换都无效。**

.cExpression

句点分隔符 (.) 和 *cExpression* 选项可用来在宏后面追加额外的字符。使用 *.cExpression* 附加在宏后面的 *cExpression* 也可以是一个宏。

说明

宏替换把变量和数组元素中的内容当作原义字符串。当连字符 (&) 位于字符型变量或数组元素前面时，变量和数组元素的内容将替代宏引用。宏替换可用在任何接受原义字符串的命令和函数中。

提示 请尽可能使用名称表达式来取代宏替换。名称表达式与宏替换作用相似，但是，名称表达式限于传递作为名称的字符串。当命令或函数接受名称（文件名、窗口名、菜单名等）时，使用名称表达式的处理速度要明显快得多。

如想了解更多有关名称表达方式的信息，请参阅“帮助”中的“语言概述”。

下列代码可以正确执行：

```
STORE 'customer' TO gcTableName
STORE 'company' TO gcTagName
USE &gcTableName ORDER &gcTagName
```

可以使用名称表达式代替：

```
USE (gcTableName) ORDER (gcTagName)
```

宏替换对于替换命令中的关键字是很有用的。在下面的示例中，把 TALK 设置保存在

变量中，以便在后面的程序能够恢复它。TALK 的原始设置使用宏替换恢复。

示例

```
STORE SET('TALK') TO gcSaveTalk
SET TALK OFF
*
*    其他程序代码
*
SET TALK &gcSaveTalk  && 恢复原始的 TALK 设置
```

请参阅

STORE

&& 命令

在程序文件中标明非执行内部注释行的开始。

语法

&& [Comments]

参数描述

Comments

标明其后是内部注释。例如：

STORE (20*12) TO gnPayments &&20 年的月工资之和

通过插入内部注释来标明 IF ..... ENDIF、DO 和 FOR ..... ENDFOR 等结构化命令的结束，能够大大地提高程序的可读性。

说明

在注释行的后面加入分号 (;)，表明下一行也是注释行。不能在命令行续行的分号后面加入 && 和注释。

示例

```
NOTE  Initialize the page number;
    variable.
STORE 1 to gnPageNum
* 建立循环
DO WHILE gnPageNum <= 25 && 循环 25 次
    gnPageNum = gnPageNum + 1
ENDDO && DO WHILE gnPageNum <= 25
```

请参阅

*, MODIFY COMMAND, MODIFY FILE, NOTE

* 命令

在程序文件中，标明非执行注释行的开始。

语法

* [Comments]

参数描述

Comments

给出注释行中的注释，例如：

* 这是一行注释

说明

在注释行的后面加入分号 (;)，表明下一行也是注释行。

示例

```
* Initialize the page number;
    variable.
STORE 1 to gnPageNum

* 建立循环
DO WHILE gnPageNum <= 25  && 循环 25 次
    gnPageNum = gnPageNum + 1
```

```
ENDDO && DO WHILE gnPageNum <= 25
```

请参阅

&&, MODIFY COMMAND, MODIFY FILE, NOTE

= 命令

计算表达式的值。

语法

= Expression1 [, Expression2 .....]

参数描述

Expression1 [, Expression2 .....]

指定 = 命令计算的表达式。

说明

= 命令计算一个或多个表达式的值，并且放弃返回值。这个命令对于那些只须获得所需效果，而不需要把返回值赋给变量、数组元素或字段的 Visual FoxPro 函数和用户自定义函数是非常有用的。

例如，要使用插入方式，可以执行命令：

= INSMODE(.T.)

通常情况下，INSMODE 返回“真”(.T.)或“假”(.F.)。但在上述示例中，函数执行后舍弃返回值。

如果只包含了一个表达式(*Expression1*)，则等号就是可选的。

注意 等号(=)还有其他两个与以上所述无关的用法。它可用在逻辑表达式中作为比较操作符，还可用来给变量和数组元素赋值。在这两种情况下，等号(=)是一个操作符而不是一个命令。有关在逻辑表达式中运用等号(=)的详细内容，请参阅帮助中的“关系操作符”。有关用等号(=)为变量和数组元素赋值的详细内容，请参阅 STORE。

请参阅

EVALUATE

\ 或 \\ 命令

输出文本行。

语法

\TextLine

- 或者 -

\\TextLine

参数描述

\TextLine

如果使用 \，文本行输出时先输出一个回车符和一个换行符。

\\TextLine

如果使用 \\，文本行输出前不回车换行。

\ 和 \\ 前面的空格不包含在输出行中，但是 \ 和 \\ 之后的空格包含在输出行中。

可以在文本行中嵌套一个表达式。如果表达式放在文本合并分隔符（默认情况下为 << 和 >>）之内，并且 SET TEXTMERGE 设置为 ON，则计算表达式的值，并将结果以文本形式输出。

说明

\ 和 \\ 简化了 Visual FoxPro 中的文本合并过程。文本合并允许把文本输出到一个文件，并以此创建套用信函或程序。

使用 \ 和 \\ 可以把文本行输出到当前的文本合并输出文件或屏幕。SET TEXTMERGE 用来指定文本合并输出文件。如果没有为文本合并指定文件，那么文本行将只输出到 Visual FoxPro 的主窗口中，或者输出到活动的用户自定义输出窗口中。SET TEXTMERGE NOSHOW 用来禁止输出到 Visual FoxPro 主窗口或者活动的用户自定义窗口。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer     && Open customer table
SET TEXTMERGE ON
SET TEXTMERGE TO letter.txt

\<<CDOW(DATE（） )>>, <<CMONTH(DATE（） )>>

\\  <<DAY(DATE（） )>>, <<YEAR(DATE（） )>>

\

\

\Dear <<contact>>

\Additional text

\

\Thank you,

\

\XYZ Corporation
CLOSE ALL
MODIFY FILE letter.txt NOEDIT
```

请 参 阅

_PRETEXT, SET TEXTMERGE, SET TEXTMERGE DELIMITERS, _TEXT, TEXT ..... ENDTEXT

? 或 ?? 命令

计算表达式的值，并输出计算结果。

语法

? | ?? Expression1
[PICTURE cFormatCodes] | [FUNCTION cFormatCodes] | [VnWidth]
[AT nColumn]
[FONT cFontName [, nFontSize] [STYLE cFontStyle | Expression2]]
[, Expression3] .....

参数描述

? Expression1

计算表达式 *Expression1* **的值，然后输出一个回车和换行符，再输出计算结果。计算结果显示在** Visual FoxPro **主窗口或者活动的用户自定义窗口的下一行，并且如果函数代码** *cFormatCodes* **或系统变量** _ALIGNMENT **没有指定其他值，则该结果打印在页面的左侧。**

如果省略了表达式，则显示或打印一个空行。当包含多个表达式时，表达式结果之间将插入一个空格。

?? Expression1

计算表达式 *Expression1* 的值，并把结果显示在 Visual FoxPro 主窗口、活动的用户自定义窗口，或者打印在打印机当前行的当前位置上。输出计算结果前不回车换行。

PICTURE cFormatCodes

指定显示表达式 *Expression1* 计算结果的图片格式。*cFormatCodes* 可以包括函数代码、图片代码或者两者的组合。表达式中可以使用与 @ ..... SAY 中相同的图片和函数代码。

函数代码影响结果输出的总体格式，图片代码则只对结果中的单个字符有影响。如果 *cFormatCodes* 中使用了函数代码，那么函数代码必须放在图片代码之前，并且代码前必须加 @ 。没有内嵌空格的多重函数代码可以紧接着放在 @ 之后。最后一个函数代码之后必须有一个或多个空格。空格标志着函数代码的结束和图片代码的开始。

FUNCTION cFormatCodes

指定在 ? 和 ?? 输出中的函数代码。如果包括函数子句，则不要把 @ 放在函数代码之前。当 PICTURE 中包括函数代码时，函数代码前必须使用 @ 。

VnWidth

指定一种特殊函数代码，能使字符表达式的结果在有限列中垂直伸展。*nWidth* 指定输出的列数。

?'函数代码 V 按如下示例方法使用';

FUNCTION 'V10'

AT nColumn

指定显示结果的列编号。这个选项使您能够在指定的若干列中对齐输出结果，以便创建一个表。数值表达式 *nColumn* 可以是返回数值的用户自定义函数。

FONT cFontName [, nFontSize]

指定用于 ? 或 ?? 输出的字体。*cFontName* 指定字体名称，*nFontSize* 指定字体的大小。例如，下面的命令将以 16 磅的 Courier 字体显示系统日期：

```
? DATE() FONT 'Courier', 16
```

如果给出 FONT 子句但是没有指定字体大小 *nFontSize*，此时字体大小为 16 磅。

如果省略了 FONT 子句，并且 ? 或 ?? 的输出结果放在 Visual FoxPro 主窗口中，则输出的字体为 Visual FoxPro 主窗口字体。如果省略了 FONT 子句，并且 ? 或 ?? 的输出结果放在用户自定义窗口中，则输出的字体为用户自定义窗口字体。

注意 在 Visual FoxPro 中，如果找不到指定的字体，则用具有相似字体特性的字体代替。

STYLE cFontStyle

指定用于 ? 或 ?? 输出的字形。如果省略 STYLE 子句，则使用“常规”字形。如果找不到指定的字形样式，则用具有相似字体特性的字形代替。

注意 当您使用 STYLE 子句指定字形时，必须包含 FONT 子句。

可以用 *cFontStyle* 指定的字形有：

字符	字形
B	粗体
I	斜体
N	常规
O	轮廓
Q	不透明
S	阴影
-	删除线
T	透明
U	下划线

可以使用多个字符的组合来指定字形。例如，下面的命令用 Courier 粗斜体来显示系统日期：

```
? DATE() FONT 'COURIER' STYLE 'BI'
```

说明

? 和 ?? 计算表达式的值，并把计算结果传送到 Visual FoxPro 主窗口、活动的用户自定义窗口或者打印机上。

如果 SET PRINTER 是 ON，则表达式的计算结果送到打印机和 Visual FoxPro 主窗口或活动的用户自定义窗口上。如果 SET PRINTER 为 ON，但 SET CONSOLE 为

OFF，则表达式计算结果只送到打印机。

示例

```
? 15 * (10+10)

? 'Welcome to ' PICTURE '@!'

?? 'Visual FoxPro'
```

请参阅

???, @ ..... SAY, SET MEMOWIDTH, SET PRINTER, SET SPACE

??? 命令

把结果直接输出到打印机。

语法

??? cExpression

参数描述

cExpression

指定要输出到打印机的字符。

说明

三个问号不通过打印机驱动程序而直接把 *cExpression* 的内容传送到打印机上。*cExpression* 必须包含有效的打印机代码。

打印机控件代码能够重置打印机，改变打印样式和大小，还能启用或取消黑体打印等。这些代码可由两种字符的任意组合来组成，这两种字符是由正在使用的用户打印机的可打印字符和非打印字符组成。将控件代码发送到打印机的方法有很多：

- 使用 + 把 CHR（） 和引号括起的字符串合并到一起，然后直接把 ASCII 字符串传送给打印机。
- 使用引号把包含打印代码的字符串或 ASCII 字符串送入打印机。
- 在开始打印前和打印结束后，可使用 _PSCODE 和 _PECODE 系统变量把代码传送到打印机上。有关详细内容，请参阅 _PSCODE 和 _PECODE。

打印机控件代码随打印机的不同而不同。有关打印机控件代码信息的内容，最好参阅打印机的随机手册。

请参阅

? | ??, @ ..... SAY, CHR（）, **打印对话框**

@.....BOX 命令

用指定的坐标画一个方框。包含此命令是为了提供向后兼容性。对于 Visual FoxPro 应

用程序，可用形状控件来代替。

@ .....CLASS 命令

创建可以用 READ 激活的控件或对象。

语法

@ nRow, nColumn CLASS ClassName NAME ObjectName

参数描述

@ nRow, nColumn

指定控件或对象的位置。控件或对象的宽度和高度由类的默认宽度和高度值确定。

行从上向下编号。Visual FoxPro 主窗口或用户自定义窗口中的第一行编号为 0。Visual FoxPro 的第 0 行就是紧接着 Visual FoxPro 系统菜单栏的那一行。

列从左向右编号。Visual FoxPro 主窗口或用户自定义窗口中的第一列编号为 0。当把一个对象或控件添加到用户自定义窗口中时，行和列坐标与用户自定义窗口，而不是 Visual FoxPro 主窗口相关。

在 Visual FoxPro 中，Visual FoxPro 主窗口或用户自定义窗口中的某一位置，由 Visual FoxPro 主窗口或用户自定义窗口的字体决定。大多数字体可以用不同大小显示。有一

些字体能够按比例留间隔。行的高度与当前字体的高度一致，列的宽度则与当前字体字符的平均宽度一致。

在 Visual FoxPro 中，可以使用小数指定控件或对象的行、列坐标。

CLASS ClassName

指定控件或对象的类。*ClassName* 可以是 Visual FoxPro 的一个基类，也可以是用户自定义类。下表列出了 *ClassName* 可指定的 Visual FoxPro 基类。

基类名称		
CheckBox	Column	ComboBox
CommandButton	CommandGroup	Container
Control	Cursor	Custom
DataEnvironment	EditBox	Grid
Header	Image	Label
Line	ListBox	OLEBoundControl
OLEControl	OptionButton	OptionGroup
Page	PageFrame	Relation
Separator	Shape	Spinner
TextBox	Timer	

NAME ObjectName

指定对象引用变量的名称，NAME 子句可以创建此变量。控件或对象的面向对象属性、事件和方法都可以通过引用这个变量来操纵。

说明

@ ..... CLASS 提供了一个捷径，能把 FoxPro 早期版本中创建的应用程序移植到更受欢迎的 Visual FoxPro 面向对象编程方法中来。有关与 FoxPro 2.x 控件的向后兼容性的附加信息，请参阅本书稍后的“控件与对象”。

有关在 Visual FoxPro 中的面向对象编程，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”。

示例

下面的例子说明了如何把 @ ..... CLASS 同早期 FoxPro 版本中的编程技术结合起来使用（在这个例子中，使用 READ 激活控件）。@ ..... CLASS 创建了一个文本框。这个文本框的属性可以用 Visual FoxPro 面向对象的编程技术来改变。

使用 ON KEY LABEL，可以在按 ALT + W 时显示“窗口颜色”对话框。用 @ ..... CLASS 命令把文本框添加到 Visual FoxPro 主窗口中，然后用 READ 激活文本框。

```
CLEAR
ON KEY LABEL CTRL+I _SCREEN.PageFrame1.Page1.goFirstName.BackColor;
   = GETCOLOR()
@ 2,2 SAY '按下 Ctrl+I 键改变背景颜色'

@ 4,2 CLASS TextBox NAME goFirstName
READ
CLEAR
```

请参阅

CREATEOBJECT()，DEFINE CLASS，READ，_SCREEN

@ ……CLEAR 命令

清除 Visual FoxPro 主窗口或用户自定义窗口的部分区域。

语法

@ nRow1, nColumn1 [CLEAR | CLEAR TO nRow2, nColumn2]

参数描述

@ nRow1, nColumn1 CLEAR

清除一长方形区域。该长方形区域的左上角是第 *nRow1* 行第 *nColumn1* 列，右下角是 Visual FoxPro 主窗口或用户自定义窗口的右下角。

CLEAR TO nRow2, nColumn2

清除一长方形区域。该长方形区域左上角是第 *nRow1* 行第 *nColumn1* 列，右下角是第 *nRow2* 行第 *nColumn2* 列。

说明

如果省略 CLEAR 或 CLEAR TO，Visual FoxPro 将清除第 *nRow1* 行中从 *nColumn1* 列开始到行尾的内容。

示例

以下示例清除屏幕上 Visual FoxPro 主窗口或用户自定义窗口从第二行开始到窗口右下角的区域。

```
@ 2,0 CLEAR
```

以下示例清除从第 10 行第 0 列开始一直到第 20 行 20 列的长方形区域。

```
@ 10,0 CLEAR TO 20,20
```

请参阅

CLEAR

@ ......EDIT 编辑框命令

创建一个编辑框。包含此命令是为了提供向后兼容性。对于 Visual FoxPro 应用程序，请使用编辑框控件代替此命令。

@ ..... FILL 命令

更改屏幕某区域中文本的颜色。

语法

@ nRow1, nColumn1 FILL TO nRow2, nColumn2
[COLOR SCHEME nSchemeNumber | COLOR ColorPairList]

参数描述

@ nRow1, nColumn1
指定要更改颜色区域的左上角。

FILL TO nRow2, nColumn2
指定要更改颜色区域的右下角。

COLOR SCHEME nSchemeNumber
指定区域的颜色。只有指定配色方案中的第一个颜色对决定了区域的颜色。

COLOR ColorPairList
指定区域的颜色。只有指定颜色对列表中第一个颜色对决定了区域的颜色。

如果省略 COLOR SCHEME 或 COLOR 子句，则清除指定的长方形区域。清除一长方形区域也可使用 @ ..... CLEAR 命令。

有关配色方案与颜色对的信息，请参阅稍后的“语言参考”中的“颜色概述”。

说明

这个命令用来更改 Visual FoxPro 主窗口或活动的用户自定义窗口中长方形区域的文本颜色。使用该命令，只能为正在显示的文本设置前景和背景颜色属性。所有使用 @ ……FILL 命令之后输出到该指定区域中的文本，仍以屏幕或窗口默认的颜色显示。

示例

以下示例先清除 Visual FoxPro 主窗口，然后用彩色填充一个区域。

```
ACTIVATE SCREEN
CLEAR

@ 4,1 FILL  TO 10, 8 COLOR GR+/B
```

请参阅

@ …… SAY，颜色概览

@ ……GET - 复选框命令

此命令用于创建复选框。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，请使用复选框控件代替此命令。

@ ......GET - 组合框命令

创建组合框。包含此命令是为了提供向后兼容性。对于 Visual FoxPro 应用程序，请使用组合框控件代替此命令。

@ ......GET - 命令按钮命令

创建一组命令按钮。包括此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，请使用命令按钮控件代替此命令。

@ ......GET-列表框命令

创建列表框。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，请使

用列表框控件代替此命令。

@ ..... GET-选项按钮命令

创建一组选项按钮。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，请使用选项组控件代替此命令。

@ .....GET-微调控件命令

创建微调控件。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，请使用微调控件代替此命令。

@ ……GET -文本框命令

创建文本框。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，请使用文本框控件代替此命令。

@ ……GET -透明按钮命令

创建一个透明的命令按钮。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序，请使用命令按钮控件代替此命令。

@ ……MENU 命令

创建一个菜单。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，可

用菜单设计器和 CREATE MENU 代替。

@ .....PROMPT 命令

创建菜单栏。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，可用菜单设计器和 CREATE MENU 代替。

@ .....SAY 命令

在指定的行列位置显示或打印输出结果。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，可用标签控件命令来显示文本，用文本框控件显示字段与变量内容。

@ ……SAY-图片和 OLE 对象命令

显示图片和 OLE 对象，或调用 OLE 服务程序来执行 OLE 对象操作。包含此命令是为了提供向后兼容性。在 Visual FoxPro 应用程序中，可用图像控件、OLE 绑定型控件和 OLE 容器控件代替。

@ ……SCROLL 命令

将 Visual FoxPro 主窗口或用户自定义窗口的某个区域向上、下、左、右滚动。

语法

```
@ nRow1, nColumn1 TO nRow2, nColumn2 SCROLL
  [UP | DOWN | LEFT | RIGHT]
  [BY nMoveAmount]
```

参数描述

@ nRow1, nColumn1 TO nRow2, nColumn2 SCROLL

移动矩形区域，该区域左上角坐标为 *nRow1, nColumn1*，右下角坐标为 *nRow2, nColumn2*。

UP | DOWN | LEFT | RIGHT

指定矩形区域的移动方向。若省略该子句，区域将向上移动。

BY nMoveAmount

指定区域移过多少行或列。若省略 BY *nMoveAmount* 参数，则区域移动一行或一列。

请参阅

SCROLL

@ ……TO 命令

绘制方框、圆或椭圆。包含此命令是为了提供向后兼容性。在 Visual FoxPro 的应用程序中，可以用形状控件命令代替。

ABS（） 函数

返回指定数值表达式的绝对值。

语法

ABS(nExpression)

返值类型

数值型

参数描述

nExpression

指定需由 ABS（） 返回绝对值的数值表达式。

示例

```
? ABS(-45)      && 显示 45
? ABS(10-30)    && 显示 20
? ABS(30-10)    && 显示 20
STORE 40 TO gnNumber1
STORE 2 TO gnNumber2
? ABS(gnNumber2-gnNumber1)    && 显示 38
```

请参阅

INT（），ROUND（），SIGN（）

ACCEPT 命令

从屏幕接收字符串数据，包含此命令是为了提供向后兼容性，在 VisualFoxpro 中，可使用文本框控件命令代替。

ACLASS（） 函数

将一个对象的类名和祖先类名存放到一个变量中。

语法

ACLASS(ArrayName, oExpression)

返值类型

数值型

参数描述

ArrayName

指定存放类名的数组名。如果指定的数组名不存在，Visual FoxPro将自动创建此数组；如果指定的数组已存在，但它的大小不足以容纳所有父类名，则Visual FoxPro自动增加数组的大小；如果已存在的数组超过所需大小，将截掉多余部分；如果指定数组是已存在的二维数组，则将它转化为一维数组。

oExpression

指定一个对象，此对象的类名及其父类名将存放在指定的数组中。*oExpression*可以为任意的对象表达式，例如，对象引用、对象变量或对象数组元素。

说明

ACLASS()函数创建一个一维数组，数组包含了指定对象的类名和祖先类名。第一个数组元素包含此对象的类名，第二个元素包含对象的父类名；第三个元素包含对象的祖父类名，依此类推。

ACLASS()函数返回数组中类名的数目。如果不能创建此数组，则ACLASS()函数的返回值为0。

示例

下面的示例从Visual FoxPro 的Form基类创建了两个自定义类，FormChild和FormGrandChild。ACLASS()用来创建一个包含类名称、名为gaNewarray的数组，并且会显示出来。

```
CLEAR
frmMyForm = CREATEOBJECT("FormGrandChild")
```

```
FOR nCount = 1 TO ACLASS(gaNewarray, frmMyForm)    && 生成数组
   ? gaNewarray(nCount)  && 显示 类名
ENDFOR
RELEASE frmMyForm

DEFINE CLASS FormChild AS FORM
ENDDEFINE

DEFINE CLASS FormGrandChild AS FormChild
ENDDEFINE
```

请参阅

ADD CLASS, AMEMBERS（）, CREATE CLASS, CREATE CLASSLIB, CREATEOBJECT（）, DEFINE CLASS

ACOPY（） 函数

把数组的元素复制到另一个数组中。

语法

ACOPY(SourceArrayName, DestinationArrayName
[, nFirstSourceElement [, nNumberElements [, nFirstDestElement]]])

返值类型

数值型

参数描述

SourceArrayName, DestinationArrayName

将源数组 *SourceArrayName* 中的元素一对一地复制到目标数组 *DestinationArrayName* 中，源数组中的元素将替换目标数组的元素。

数组可以是一维或二维数组。如目标数组不存在，Visual FoxPro 将自动创建目标数组。这时，目标数组的大小与源数组大小相同。

注意 可以用两种方法引用二维变量数组的元素。第一种是用两个下标值指定元素在数组中的行和列，另一种则指定单个元素的编号。此函数和其他操作二维数组的函数需要单个元素编号（此处为 nFirstSourceElement 和 nFirstDestElement）。使用 AELEMENT( ) 函数，可以根据元素的行和列下标，返回二维数组中某元素的正确元素编号。.

nFirstSourceElement

指定源数组中第一个被复制元素的编号，复制从此元素开始（复制时，包含编号为 *nFirstSourceElement* 的元素）。如果复制时不使用元素编号 *nFirstSourceElement*，则从源数组的第一个元素开始复制。

nNumberElements

指定复制的源数组的元素数目。如果 *nNumberElement* 为 -1，则从元素 *nFirstSourceElement* 开始，复制源数组中所有的元素。

nFirstDestElement

指定目标数组中第一个被替换的元素。

说明

ACOPY() 函数返回复制到目标数组中的元素个数。

示例

下面的示例从 customer 表中选定的记录创建一个数组，并且用 ACOPY() 创建一个新数组。

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

SELECT DISTINCT company ;
    FROM customer ;
    ORDER BY company ;
    WHERE country = 'Germany';
    INTO ARRAY gaCompanies
= ACOPY(gaCompanies, gaCompaniesTemp) && 复制数组
CLEAR
DISPLAY MEMORY LIKE gaCompaniesTemp
```

请参阅

ADEL()，AELEMENT()，AINS()，ASCAN()，ASORT()，DIMENSION

ACOS（） 函数

返回指定数值表达式的反余弦值。

语法

ACOS(nExpression)

返值类型

数值型

参数描述

nExpression

指定的数值表达式，ACOS（） 函数返回此表达式的反余弦值。*nExpression* 参数的取值范围从 -1 到 +1，ACOS（） 函数的值域从 0 到 π (3.141592)。

ACOS（） 函数返回数值的小数位数由 SET DECIMALS 指定。

RTOD（） 函数可把弧度值转化为角度值。

说明

函数返回的反余弦值为弧度值。

示例

```
CLEAR
? RTOD(ACOS(0)) && 显示 90.00
```

```
STORE -1 to gnArcAngle
? RTOD(ACOS(gnArcAngle)) && 显示 180.00
? RTOD(ACOS(SQRT(2)/2)) && 显示 45.00
```

请参阅

ASIN（）,ATAN（）,ATN2（）,COS（）,DTOR（）,RTOD（）,SET DECIMALS,SIN（）,TAN（）

Activate 事件

当激活表单、表单集或页对象，或者显示工具栏对象时，将发生 Activate 事件。

语法

PROCEDURE Object.Activate

说明

此事件的触发取决于对象的类型：

- **当表单集中的一个表单获得焦点，或调用表单集的 Show 方法时，激活表单集对象。**
- **当用户单击一个表单或单击一个控件，或者调用表单对象的 Show 方法时，**

激活表单对象。

- 当用户单击页面的选项卡，单击页面上的控件，或者将包含页对象的页框的 ActivePage 属性设置为此页对象对应页码时，激活页对象。
- 当调用工具栏的 Show 方法时，激活工具栏。

使用表单集的 Show 方法时，将显示所有 Visible 属性为“真”(.T.) 的表单。Activate 事件触发后，首先激活表单集，然后是表单，最后是页面。

应用于

表单，表单集，页面，工具栏

请参阅

ActivePage 属性, Deactivate 事件, GotFocus 事件, LostFocus 事件, SetFocus 方法, Show 方法, Visible 属性

ACTIVATE MENU 命令

显示并激活一个菜单栏。

语法

ACTIVATE MENU MenuBarName

[NOWAIT]
[PAD MenuTitleName]

参数描述

MenuBarName
指定要激活的菜单栏的名称。

NOWAIT
指定在程序执行时，不等待用户在已激活菜单栏中选择菜单或按 ESC 键，而是继续执行程序。发出 DEACTIVATE MENU 命令后，如果菜单是用 NOWAIT 选项激活的，则程序的执行不返回到 ACTIVATE MENU 命令的下一行命令。

PAD MenuTitleName
指定一个菜单标题名，当激活菜单栏时，选中此菜单标题名。如果不指定菜单标题名，菜单栏激活后，默认激活第一个菜单标题名。

说明

该命令显示并且激活 *MenuBarName* 指定的菜单栏，需要结合 DEFINE MENU 和 DEFINE PAD 命令使用。

提示 当应用程序中包含 Visual FoxPro 系统菜单栏 (_MSYSMENU) 时，不需激活此菜单，而是执行 SET SYSMENU AUTOMATIC 命令。

示例

下面的示例用 ACTIVATE MENU 来显示和激活一个用户自定义的菜单系统。首先用

SET SYSMENU SAVE命令将当前系统菜单栏存入内存，然后用 SET SYSMENU TO 命令删除所有系统菜单标题。

用 DEFINE PAD命令创建两个菜单标题，并且用 DEFINE POPUP命令为每一个菜单标题创建一个下拉菜单，再用 DEFINE BAR命令为每个菜单创建菜单项。当选择一个菜单标题后，ON PAD 命令通过 ACTIVATE POPUP子句激活相应的菜单。

ACTIVATE MENU显示和激活菜单栏。

从菜单中选择某个菜单项后，将执行 CHOICE过程。CHOICE过程将显示选定的菜单项名和对应菜单名。

```
*** Name this program ACTIMENU.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 actimenu;
    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 ' ;
    KEY ALT+C, ''
ON SELECTION POPUP cardinfo;
    DO choice IN actimenu WITH PROMPT ( ) , POPUP ( )
```

ACTIVATE MENU example
DEACTIVATE MENU example
RELEASE MENU example EXTENDED
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

CLEAR MENUS, CREATE MENU, DEACTIVATE MENU, DEFINE MENU, DEFINE PAD , HIDE MENU, SET SYSMENU, SHOW MENU

ACTIVATE POPUP 命令

显示并且激活一个菜单。

语法

ACTIVATE POPUP MenuName
[AT nRow, nColumn]
[BAR nMenuItemNumber]
[NOWAIT]
[REST]

参数描述

MenuName
指定要激活的菜单的名称。

AT nRow, nColumn

指定菜单在屏幕或用户自定义窗口中显示的位置，行和列的坐标值为菜单左上角的坐标值。用此参数确定的位置优先于 DEFINE POPUP 命令的 FROM 参数指定的位置。

BAR nMenuItemNumber

指定菜单激活后自动选定的菜单项。例如，如果 *nMenuItemNumber* 为 2，则菜单激活后将自动选择第二个菜单项。如果省略参数 BAR *nMenuItemNumber*，或者 *nMenuItemNumber* 的数值大于菜单中的项数，则菜单激活后将选择第一个菜单项。

NOWAIT

指定程序运行时不等待用户选择菜单项，而是继续执行程序。

REST

如果 DEFINE POPUP 命令中带有 PROMPT FIELD 子句，则它所创建菜单的各项就是每个记录指定字段的内容。此菜单激活时，即使包含此字段的表的记录指针没有指向第一个记录，初始时还是选定第一个菜单项。

包含 REST 选项，则指定此菜单激活时选定与表中当前记录指针相对应的项。

说明

ACTIVATE POPUP 与 DEFINE POPUP 命令结合使用，用于创建菜单，而 DEFINE BAR 命令则用于创建菜单中的菜单项。

示例

这个示例在选择一个菜单标题后，用 ACTIVATE POPUP 命令激活菜单。首先用 SET

SYSMENU SAVE 命令将当前系统菜单栏存入内存，然后用 SET SYSMENU TO 命令删除所有系统菜单标题。

用 DEFINE PAD 命令创建两个新的系统菜单标题，并且用 DEFINE POPUP 命令为每一个菜单标题创建一个菜单，再用 DEFINE BAR 命令为每个菜单创建菜单项。当选择一个菜单标题后，ON PAD 命令通过 ACTIVATE POPUP 子句激活相应的菜单。

从菜单中选择某个菜单项后，将执行 CHOICE 过程。CHOICE 过程将显示选定的菜单项名和对应菜单名。如果从 Card Info 菜单中选择 Exit 菜单项，将恢复为原来的 Visual FoxPro 系统菜单。

```
*** 命名此程序为 ACTIPOP.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 actipop 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' ;
    KEY ALT+C, ''
DEFINE BAR 6 OF cardinfo PROMPT '\-'
DEFINE BAR 7 OF cardinfo PROMPT 'E\<xit';
    KEY ALT+X, ''
ON SELECTION POPUP cardinfo;
DO choice IN actipop 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

CLEAR POPUPS, CREATE MENU, DEACTIVATE POPUP, DEFINE BAR, DEFINE POPUP, HIDE POPUP, MOVE POPUP, ON SELECTION POPUP, POP POPUP, POPUP（）, PROMPT（）, PUSH POPUP, SHOW POPUP

ACTIVATE SCREEN 命令

把结果输出到 Visual FoxPro 主窗口，而不是用户自定义活动的窗口。

语法

ACTIVATE SCREEN

说明

用 ACTIVATE WINDOW 命令可使结果输出到用户自定义的窗口中。

请参阅

ACTIVATE WINDOW, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, SHOW WINDOW

ACTIVATE WINDOW 命令

显示并且激活一个或多个用户自定义窗口或 Visual FoxPro 系统窗口。

语法

```
ACTIVATE WINDOW WindowName1 [, WindowName2 ......]
  | ALL
  [IN [WINDOW] WindowName3 | IN SCREEN
  [BOTTOM | TOP | SAME]
  [NOSHOW]
```

参数描述

WindowName1 [, WindowName2 ......]

指定要激活的窗口的名称，窗口名用逗号分开。在 Visual FoxPro 中，可以指定一个待激活工具栏的名称。若要列出 Visual FoxPro 所有的工具栏名，请参阅 SHOW WINDOW。

ALL

指定激活所有窗口，最后一个被激活的窗口为活动的输出窗口。

IN [WINDOW] WindowName3

指定父窗口名，要激活的窗口放入这个父窗口中并激活，激活的窗口变成一

个子窗口。一个父窗口可以有多个子窗口，在父窗口内激活的子窗口不能移出父窗口。如移动父窗口，子窗口也随之一起移动。

注意 父窗口对于每一个可视的子窗口来说必须是可视的。

IN SCREEN

在 Visual FoxPro 主窗口中放置并激活窗口。创建一个窗口时，可用 DEFINE WINDOW 的 IN WINDOW 子句将它放入一个父窗口中。用包含 IN SCREEN 子句的 ACTIVATE WINDOW 命令激活窗口时，DEFINE WINDOW 命令中的 IN WINDOW 子句将不起作用。

BOTTOM | TOP | SAME

指定被激活窗口对应其他已激活窗口的位置。默认情况下，窗口激活后为最顶层的窗口。如果使用 BOTTOM 子句，窗口激活后位于所有其他窗口之后；如果使用 TOP 子句，则激活窗口位于所有其他窗口之前。如果使用 SAME 子句，窗口激活后并不影响窗口的前后位置。

NOSHOW

激活一个窗口，并使输出结果输出至此窗口，但不显示这个窗口。

说明

可以使用 DEFINE WINDOW 命令创建用户自定义窗口。

激活一个窗口，使它成为最顶层的窗口，并且所有的输出结果都输出到这个窗口。输出结果一次只能输出到一个窗口中。只有在活动的输出窗口变为不活动的或被释放时，或者将另一个窗口或 Visual FoxPro 主窗口激活时，才将其他窗口设置为活动的输出窗口。

用户自定义窗口的名称显示在窗口菜单底部。活动的用户自定义窗口用复选标记来标识。

在 Visual FoxPro 主窗口中，能同时放置多个窗口，但输出窗口只能是最后一个被激活的窗口。当打开多个窗口时，使输出窗口由活动变为非活动将从 Visual FoxPro 主窗口中移去此窗口，并把以后的输出结果送至另一个窗口。如果没有活动的输出窗口，输出结果将输出到 Visual FoxPro 主窗口。

注意 当活动输出窗口变为非活动窗口时，为了保证输出结果定向到一个确定的窗口，您必须用 ACTIVATE WINDOW 命令明确地激活这个窗口。

所有活动窗口只要不用 DEACTIVATE WINDOW 或 HIDE WINDOW 命令把它从屏幕上移去，就一直显示。但是执行这两个命令只是从屏幕上移去窗口，并不从内存中删除窗口。事实上，执行 ACTIVATE WINDOW 或 SHOW WINDOW 命令，又可使窗口重新显示。

要从屏幕和内存中同时删除窗口，可使用 CLEAR WINDOWS、RELEASE WINDOWS 或 CLEAR ALL 命令。从内存中删除的窗口必须重新定义，才能重新放置到 Visual FoxPro 主窗口中。

您可以用 ACTIVATE WINDOW 命令把 Visual FoxPro 系统窗口放在 Visual FoxPro 主窗口或一个父窗口中。

下列系统窗口可用 ACTIVATE WINDOW 命令打开：

- 命令窗口
- 调用堆栈窗口

- 数据工作期窗口
- 调试窗口
- 调试输出窗口
- 局部窗口
- 跟踪窗口
- 查看窗口

若要激活一个（Visual FoxPro 中的）系统窗口或工具栏，要用引号把整个系统窗口名或工具栏名括起来。例如，要在 Visual FoxPro 中激活“报表控件”工具栏，可发出下面命令：

```
ACTIVATE WINDOW " Report Controls "
```

用 HIDE WINDOW 或 RELEASE WINDOW 命令可从 Visual FoxPro 主窗口或父窗口中移去系统窗口。

示例

下面的示例定义了一个名为 output 的窗口并激活它，将它置于 Visual FoxPro 主窗口中。WAIT 命令能暂停执行，窗口被隐藏，然后再重新显示。

```
CLEAR
DEFINE WINDOW output FROM 2,1 TO 13,75 TITLE 'Output' ;
    CLOSE FLOAT GROW ZOOM
ACTIVATE WINDOW output
WAIT WINDOW 'Press any key to hide window output'
HIDE WINDOW output
```

```
WAIT WINDOW 'Press any key to show window output'
SHOW WINDOW output
WAIT WINDOW 'Press any key to release window output'
RELEASE WINDOW output
```

请参阅

CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, RELEASE WINDOWS, SHOW WINDOW

ActivateCell 方法

激活表格控件中的一个单元。

语法

Grid.ActivateCell(nRow, nCol)

参数描述

nRow, nCol

指定活动单元所在的行和列。

应用于

表格

请参阅

ActiveColumn **属性**, ActiveRow **属性**, **表格** HitTest **方法**

ActiveColumn **属性**

返回一个整数，表明表格控件中包含活动单元的列的编号。设计时不可用，运行时只读。

语法

Grid.ActiveColumn

说明

如果表格没有焦点，则 ActiveColumn **返回零。**

应用于

表格

请参阅

ActivateCell **方法**, ActiveRow **属性**, grid itTest **方法**

ActiveControl 属性

引用对象上的活动控件。设计时不可用，运行时只读。

语法

Object.ActiveControl.Property[= Value]

参数描述

Property

要返回或设置的属性。

Value

当前属性值或新的属性值。

说明

若指定对象的所有控件都不可见或被废止，则此属性也不可用。

如果对象是活动的，则 ActiveControl 属性引用有焦点的控件；如果对象处于非活动状态，则产生错误。

应用于

容器，表单，页面，_SCREEN，工具栏

请参阅

ActiveDoc 对象

创建可以包含于 Active Document 容器（例如 Microsoft Internet Explorer）中的一个 Active Document。

语法

ActiveDoc

说明

ActiveDoc 对象允许通过编程访问当包含一个 Active Document 时发生的事件，并且可以访问该 Active Document 的属性。它也可作为能够创建独立的.APP 或 .EXE 的应用程序的基类。

有关创建 Active Documents 的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第三十一章“互操作性和 Internet”中的“Active Document”。

属 性

Application	BaseClass	Caption
ContainerReleaseType	Class	ClassLibrary
Comment	Name	Parent
ParentClass	Tag	

事 件

CommandTargetExec	CommandTargetQuery	ContainerRelease
Destroy	Error	HideDoc
Init	Run	ShowDoc

方 法

AddProperty	ReadExpression	ReadMethod
ResetToDefault	SaveAsClass	WriteExpression

请 参 阅

COMCLASSINFO（），GETHOST（），ISHOSTED（）

ActiveForm 属性

引用表单集中或 _SCREEN 对象中活动的表单对象。设计时不可用，运行时只读。

语法

Object.ActiveForm.Property [= Setting]

- 或者 -

Object.ActiveForm.Method

参数描述

Property

指定表单集中活动表单的任一属性，例如 Caption 属性。

Setting

Property 属性已有的或新的设置。

Method

指定表单集中活动表单的任一方法，例如 Move 方法。

说明

如果表单集对象是活动的，则 ActiveForm 引用有焦点的表单对象；如果表单集处于非活动状态，则产生错误。

使用 ActiveForm 属性可以访问活动表单对象的属性和方法。

应用于

表单，表单集，_SCREEN

请参阅

Activate 事件，Deactivate 事件

ActivePage 属性

返回页框对象中活动页面的页码。设计时可用，运行时可读写。

语法

PageFrame.ActivePage

应用于

页框

请参阅

Activate 事件，GotFocus 事件

ActiveProject 属性

对于当前活动项目管理器窗口，包含对项目对象的一个对象引用。设计和运行时只读。

语法

Object.ActiveProject

说明

当打开多个项目管理器窗口时，ActiveProject 属性包含一个对象引用，引用最前面项目管理器窗口中项目对象。如果当没有打开项目管理器窗口时，试图访问 ActiveProject 属性会产生一个错误。有关项目的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第三十二章“应用程序开发和开发者的生产率”中的“项目管理器挂接程序”。

应用于

应用程序对象，_VFP

请参阅

File 对象，文件集合，Project 对象，ProjectHook 对象，项目集合，Server 对象，Servers Collection

ActiveRow 属性

指定表格控件中包含活动单元的行。设计时不可用，运行时只读。

语法

Grid.ActiveRow

说明

ActiveRow 属性的返回值与索引表中 RECNO（） 函数的返回值不同。如果表格没有焦点，则 ActiveRow 返回零。

应用于

表格

请参阅

ActivateCell 方法, ActiveColumn 属性, GridHitTest 方法, RECNO（）

ADATABASES（） 函数

将所有打开的数据库的名称和路径放到变量数组中。

语法

ADATABASES(ArrayName)

返值类型

数值型

参数描述

ArrayName

指定的数组名。如果指定数组不存在，则 Visual ForPro 自动创建一个数组。如果数组存在，但大小不足以包含所有数据库信息，则 Visual ForPro 自动增加数组大小，使得数组能容纳所有信息；若数组超过了所需值，则 Visual FoxPro 将截掉多余部分；若数组存在，并且由于没有打开的数据库而导致 ADATABASES（） 的返回值为 0，则不更改已有的数组；如果数组不存在，并且 ADATABASES（） 函数的返回值为 0，则不创建指定数组。

说明

在当前数据工作期，所有打开数据库的名称都将放置在变量数组里。

ADATABASES（） 函数创建一个二维数组。数组的第一列存放打开数据库的名称，

第二列存放数据库的路径。

ADATABASES() 函数返回数组中数据库名的数目（行数）。如果没有打开的数据库，则 ADATABASES() 函数的返回值为 0，并且不创建数组。

示例

下面的示例先打开数据库 testdata，然后用 ADATABASES() 函数创建一个包含所有打开数据库名的数组 gaDatabase。

```
SET PATH TO (HOME(2) + 'data\')   && 设置到数据库的路径
OPEN DATABASE testdata && 打开数据库
CLEAR
?ADATABASES(gaDatabase)   && 创建一个包含打开数据库的数组
DISPLAY MEMORY LIKE gadatabase && 显示数组的内容
CLOSE DATABASES
```

请参阅

CREATE DATABASE, DISPLAY DATABASE, LIST DATABASE, MODIFY DATABASE, OPEN DATABASE

ADBOBJECTS() 函数

把当前数据库中的命名连接名、关系名、表名或 SQL 视图名放到一个变量数组中。

语法

ADBOBJECTS(ArrayName, cSetting)

返值类型

数值型

参数描述

ArrayName

指定存放数据库名称的数组名。若指定数组不存在，Visual FoxPro将自动创建一个数组。如果数组存在，但数组大小不足以包含所有名称，则Visual FoxPro自动增大数组，使得数组能容纳所有名称。如果数组的大小超过所需值，Visual FoxPro将截掉多余部分；如果数组存在，但由于没找到任何名称导致ADBOBJECTS（）函数返回值为零，则数组内容将保持不变；如果数组不存在，并且ADBOBJECTS（）函数返回值为零，则不创建指定数组。

如果一维数组在创建时指定为CONNECTION、TABLE或VIEW，则一维数组中每行包含数据库中的连接名、表名或视图名。

如果创建二维数组时指定为RELATION，则二维数组的每行对应数据库中的一个关系。数组第一列存放子表名，第二列存放父表名，第三列存放子表的索引标识名，第四列存放父表的索引标识名。

数组的第五列存放参照完整性信息。如果关系没有参照完整性规则，则这一列为空。如果关系具有参照完整性规则，则这一列存放一些字符，这些字符决定了修改、删除和插入的参照完整性规则类型。

第一个字符代表更新规则类型，第二个字符代表删除规则类型，第三个字符代表插入规则类型。

更新和删除的可选字符值为“C”、“R”和“I”，其中“C”表示级联，“R”表示约束，“I”表示忽略。插入的可选字符值为“R”和“I”，其中“R”表示约束，“I”代表忽略。例如，如果某个关系的参照完整性规则是级联更新、约束删除和忽略插入，则第五列存放字符串“CRI”。

cSetting

指定哪些名称放在变量数组中。下表列出了参数 *cSetting* 的可选值和放在数组中的相应名称。

CSetting 值	名称
CONNECTION	连接名
RELATION	表关系
TABLE	表名
VIEW	视图名

CONNECTION、RELATION、TABLE 和 VIEW 设置不能缩写。

说明

运行 ADBOBJECTS() 函数时，必须有一个数据库是打开的，并且为当前数据库，否则 Visual FoxPro 将产生错误信息。

示例

下面的示例首先打开数据库 testdata，然后用 ADBOBJECTS() 函数创建名为

gaTables 的数组，数组将包含数据库中的表名，最后显示这些表名。

* 关闭所有数据库
```
CLOSE DATABASES
```

* 清除桌面，准备显示数组
```
CLEAR
```

* 打开示例用 testdata 数据库
```
OPEN DATABASE (HOME(2) + 'Data\testdata')
```

* 调用函数
```
=ADBOBJECTS(gaTables, "TABLE")
```

* 显示由 ADBOBJECTS() 函数生成的数组
```
DISPLAY MEMORY LIKE gaTables
```

请参阅

ADATABASES(), CREATE, CREATE CONNECTION, CREATE DATABASE, CREATE SQL VIEW, CREATE TABLE - SQL, DISPLAY DATABASE, INDBC(), LIST DATABASE, MODIFY DATABASE, SET DATABASE

Add 方法

将一个文件添加到项目中。

语法

Object.Add(cFileName)

返值类型

对象

参数描述

cFileName

指定要添加到项目中的文件的名称。如果所指定的文件不存在，会产生一个错误。如果打开了项目管理器窗口，则在添加了该文件之后刷新该窗口。

说明

Add 方法是文件集合的方法。当使用 Add 方法将一个文件添加到项目中时，会为该文件创建一个 File 对象，并且该 File 对象添加到文件集合中。

如果新文件成功地添加到项目，则会返回该文件的一个对象引用。如果该文件不能添加到项目中，会返回 null 值。

当一个文件添加到项目之前，会发生 QueryAddFile 事件。如果在 QueryAddFile 事件中指定了 NODEFAULT，则该文件就不会添加到项目中。在 QueryAddFile 事件中包含 NODEFAULT，会防止将一个文件添加到项目中。

应用于

文件集合

请参阅

File 对象，QueryAddFile 事件

ADD CLASS 命令

向 .VCX 可视类库中添加一个类定义。

语法

ADD CLASS ClassName [OF ClassLibraryName1] TO ClassLibraryName2 [OVERWRITE]

参数描述

ClassName

指定添加到 .VCX 可视类库 *ClassLibraryName2* 中的类定义名。

如果忽略可选的 OF *ClassLibraryName1* 子句，Visual FoxPro 将在所有以 SET CLASSLIB 命令打开的可视类库 .VCX 中寻找这个类定义。

如果找不到这个类定义，或者具有指定名称的类定义在 *ClassLibraryName2* 中已经存在，Visual FoxPro 将产生错误。

OF ClassLibraryName1

指定可从中复制类定义的 .VCX 可视类库名。

TO ClassLibraryName2

指定 .VCX 可视类库名，类定义将加入到这个可视类库中。如果指定的 .VCX 可视类库不存在，则 Visual FoxPro 将自动创建指定的可视类库，并把类定义加入到新创建的库中。

OVERWRITE

改写与指定类定义 *ClassName* 同名的类定义。如果省略参数 OVERWRITE，并且在 .VCX 可视类库中存在与 *ClassName* 同名的类定义，Visual FoxPro 将产生错误信息。

说明

使用 ADD CLASS 命令可以往类库中添加一个类定义，或者将某个类定义从一个可视类库（.VCX）复制到另一个类库中去。不能从一个 Visual FoxPro 程序或应用程序（.prg 或 .app），或一个过程文件添加一个类定义。

请参阅

ACLASS（）, AMEMBERS（）, CREATE CLASS, CREATE CLASSLIB, DEFINE CLASS, MODIFY CLASS, RELEASE CLASSLIB, REMOVE CLASS,

ADD TABLE 命令

向当前数据库中添加表。

语法

ADD TABLE TableName | ?
[NAME LongTableName]

参数描述

TableName
指定添加到数据库中的表的名称。

?
显示“打开”对话框，从中可以选择添加到数据库中的表。

NAME LongTableName
指定表的长名。长名可以包含 128 个字符，可用来取代扩展名为 .DBF 的短文件名。

说明

将表添加到数据库后，可以像其他数据库表一样对该表进行操作。

表添加到数据库后，就不再是自由表。但是使用 REMOVE TABLE 命令又可以使数据库中的任何一个表成为自由表。

要添加的表必须具备下列条件：

- 该表是有效的 .DBF 文件。
- 除非为表指定一个唯一的长名称，否则表不能与打开数据库中已有的表同名。
- 表不能同时放在另一个数据库中。使用 REMOVE TABLE 命令可把表从另一个数据库中移去。

事务处理中不能包含添加表的数据库。

示例

下面的示例创建名为 mydbc1 和 mydbc2 的两个数据库和一个名为 table1 的表。在创建表时，把表添加到数据库 mydbc1 中，然后关闭这个表，并将它从数据库 mydbc1 中移去。接着使用 ADD TABLE 命令把表添加到 mydbc2 中，最后使用 RENAME 命令将表名由 table1 改为 table2。

```
CREATE DATABASE mydbc1
CREATE DATABASE mydbc2
SET DATABASE TO mydbc1
CREATE TABLE table1 (cField1 C(10), n N(10)) && 往 mydbc1 中添加表
CLOSE TABLES    && 要从数据库中移去表，应先将该表关闭
REMOVE TABLE table1
SET DATABASE TO mydbc2
ADD TABLE table1
RENAME TABLE table1 TO table2
```

请参阅

CLOSE DATABASES, CREATE DATABASE, DISPLAY TABLES, FREE TABLE, OPEN DATABASE, REMOVE TABLE

ADDBS( ) 函数

向一个路径表达式添加一个反斜杠（如果必要）。

语法

ADDBS(*cPath*)

返值类型

字符型

参数描述

cPath **指定要添加反斜杠的路径名。**

请参阅

DEFAULTEXT( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTDRIVE( ), JUSTEXT( ), JUSTFNAME( ), JUSTPATH( ),

AddColumn 方法

向表格控件中添加列对象。

语法

Grid AddColumn(nIndex)

参数描述

nIndex

指定一个表示位置的数，新列将添加到表格中的此位置上。原有的列向右移动，但是 ColumnCount 属性的值不增加。

说明

调用 AddColum（） 时,现有的列向右移动，并且 ColumnOrder 属性值相应增加。表格的 ColumnCount 属性值也会增加。

表格的 Controls 属性数组通过一个元素扩大，并且对新列的引用放置在新元素中。将赋予新列一个唯一名称。

例如，为了添加新列，再赋予此列一个新名称，可以这样做。

```
THISFORM.Grid1.AddColumn(1)  && Insert column at left.
```

```
THISFORM.Grid 1.Columns(THISFORM.Grid 1.ColumnCount).Name = "NewColumn"
THISFORM.Grid 1.NewColumn.ControlSource = "Customer.CustID"
```

应用于

表格

请参阅

AddObject **方法**, ColumnCount **属性**, DeleteColumn **方法**

AddItem **方法**

在组合框或列表框中添加一个新数据项，并且可以指定数据项索引。

语法

Control.AddItem(cItem [, nIndex] [, nColumn])

参数描述

cItem

指定添加到控件中的字符串表达式。

nIndex

指定控件中放置数据项的位置。如果指定了有效的 *nIndex* **值，** *cItem* **将放置在控件的正确位置。如果指定的** *nIndex* **已经存在，数据项将插入到这个位**

置，在这个数据项后面的其他所有数据项在组合框或列表框控件的列表区中向下移一个位置。

如果忽略参数 *nIndex*，并且 Sorted 属性设置为“真”(.T.)，则 *cItem* 数据按字母排序方式添加到队列；如果忽略参数 *nIndex*，并且 Sorted 属性设置为“假”(.F.)，则 *cItem* 将添加到组合框或列表框控件的列表区未尾。

nColumn

指定控件的列，新数据项加入到此列中。默认值为 1。

说明

当 RowSourceType 属性设置为 0（无）时，可使用 AddItem 方法或 AddListItem 方法。

每个添加到组合框或列表框中的数据项都有两个标识号：

- nItemID，这是一个与控件中数据项的唯一标识 ID 相关的整数。除非指定了其他的 *nItemID* 值，第一个数据项的 *nItemID* 值为 1。
- nIndex，这是一个与控件中数据项显示顺序有关的整数，在控件中第一个数据项的 *nIndex* 值为 1。

应用于

组合框，列表框

请参阅

AddListItem 方法，Clear 方法，ListIndex 属性，ListItemID 属性，RowSourceType 属性，RemoveItem 方法，Sorted 属性

返回主目录

AINSTANCE（）函数
ALEN（）函数
Alias 属性
ALIAS（）函数
Align 属性
Alignment 属性
_ALIGNMENT 系统变量
ALINES（）函数
AllowAddNew 属性
AllowHeaderSizing 属性
AllowRowSizing 属性
AllowTabs 属性
ALLTRIM（）函数
ALTER TABLE-SQL 命令
AlwaysOnTop 属性
AMEMBERS（）函数
ANETRESOURCES（）函数
AMOUSEOBJ（）函数
ANSITOOEM（）函数
APPEND 命令
APPEND FROM 命令

APPEND FROM ARRAY 命令
APPEND GENERAL 命令
APPEND MEMO 命令

AddListItem 方法

在组合框或列表框控件中添加新的数据项，并且可以指定数据项的 ID 值。

语法

Control.AddListItem(cItem [, nItemID] [, nColumn])

参数描述

cItem

指定添加到控件中的数据项。

nItemID

指定一个整数，此值表示数据项在控件中唯一的 ID。nItemID 可以指定的最大值为 32,767。

如果忽略参数 *nItemID*，并且 Sorted 属性设置为“真”(.T.)，则 *cItem* 按字母顺序添加到组合框或列表框的列表中；如果忽略参数 *nItemID*，并且 Sorted 属性设置为“假”(.F.)，则 *cItem* 添加到组合框或列表框的列表未尾。

nColumn

指定新数据项加入到控件的哪一列。默认值为 1。

说明

当 RowSource 属性设置为 0（无）时，可使用 AddItem 或 AddListItem 方法。

每个添加到组合框或列表框中的数据项都有两个标识号：

- nItemID，这是一个与控件中数据项的唯一标识 ID 相关的整数。除非指定了其他的 *nItemID* 值，否则第一个数据项的 *nItemID* 值为 1。
- nIndex，这是一个与控件中数据项显示顺序有关的整数，在控件中第一个数据项的 *nIndex* 值为 1。

应用于

组合框，列表框

请参阅

AddItem 方法，Clear 方法，ListIndex 属性，ListItemID 属性，RowSourceType 属性，RemoveItem 方法，Sorted 属性

AddObject 方法

运行时，在容器对象中添加对象。

语法

object.AddObject(cName, cClass [, cOLEClass] [, aInit1, aInit2 ...])

参数描述

cName
指定引用新对象的名称。

cClass
指定添加对象所在的类。

cOLEClass
指定添加对象的 OLE 类。

aInit1, aInit2
指定传给新对象的 Init 事件的参数。

说明

调用 AddObject 方法时，将触发新添加对象的 Init 事件。在表单集中加入表单时，Load 事件在 Init 事件之前发生。

注意 当用 AddObject 方法往容器中加入对象时，对象的 Visible 属性设置为“假”(.F.)。因此您可以设置对象的属性，而不看更改对象外观时的一些中间效果。

示例

下面的示例介绍怎样使用 AddObject 方法将对象或控件添加到表单中。此例用 AddObject 方法往表单中加入一个 Line 控件和三个命令按钮。

Line 控件和命令按钮的 Visible 属性设置为“真”(.T.)。在默认情况下，将对象或控件添加到表单后，它们是不可见的。

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

```
frmMyForm.Addobject('shpLine','Line')  && 在表单中添加 Line 控件
frmMyForm.Addobject('cmdCmndBtn1','cmdMyCmndBtn1')  && 向上命令按钮
frmMyForm.Addobject('cmdCmndBtn2','cmdMyCmndBtn2')  && 向下命令按钮
frmMyForm.Addobject('cmdCmndBtn3','cmdMyCmndBtn3')  && 退出命令按钮

frmMyForm.shpLine.Visible = .T.  && 设置 Line 控件可见
frmMyForm.shpLine.Top = 20  && 指定 Line 控件所在的行
frmMyForm.shpLine.Left = 125  && 指定 Line 控件所在的列
frmMyForm.cmdCmndBtn1.Visible =.T.  && 使向上命令按钮可视
frmMyForm.cmdCmndBtn2.Visible =.T.  && 使向下命令按钮可视
frmMyForm.cmdCmndBtn3.Visible =.T.  && 使退出命令按钮可视

frmMyForm.SHOW  &&显示表单
READ EVENTS  && 开始事务处理

DEFINE CLASS cmdMyCmndBtn1 AS COMMANDBUTTON  && 创建命令按钮
    Caption = 'Slant \<Up'  && 命令按钮的标题
    Left = 50  && 命令按钮所在的列
    Top = 100  && 命令按钮所在的行
    Height = 25  && 命令按钮的高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F.  && 隐藏 Line 控件
        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'  && 命令按钮的标题
```

```
    Left = 200   && 命令按钮所在的列
    Top = 100   &&命令按钮所在的行
    Height = 25   && 命令按钮的高度

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

DEFINE CLASS cmdMyCmndBtn3 AS CommandButton   && 创建命令按钮
    Caption = '\<Quit'   &&命令按钮的标题
    Cancel = .T.   && 默认的取消(ESC)命令按钮
    Left = 125   &&命令按钮所在的列
    Top = 150   && 命令按钮所在的行
    Height = 25   && 命令按钮的高度

    PROCEDURE Click
        CLEAR EVENTS   && 结束事件处理，关闭表单
ENDDEFINE
```

应用于

列，命令组，容器对象，定制，数据环境，表单，表单集，表格，选项组，页面，页框，_SCREEN，工具栏

请参阅

Init 事件，Load 事件，RemoveObject 方法，Visible 属性

AddProperty 方法

向一个对象添加一个新属性。

语法

Object.AddProperty(cPropertyName [, eNewValue])

返值类型

逻辑值

参数描述

cPropertyName

指定要添加到对象中的新属性的名称。

eNewValue

指定新属性的设置值。如果省略 eNewValue，如果该属性已经存在，或者新属性设置为“假” (.F.)，则新属性的值不变。

说明

AddProperty（）方法允许您在设计时将一个属性添加到一个对象。新属性添加为一个 PUBLIC 属性。

如果该属性成功添加到对象中，AddProperty（）返回一个逻辑值“真” (.T.)；否则返

回一个逻辑值“假”(.F.)。如果尝试添加的属性已经存在，会产生一个错误。下例将演示如何给一个对象添加属性。

```
oMyForm = CREATEObject('Form')
oMyForm.AddProperty('MyArray(2)', 1)   && 增加一个数组属性
oMyForm.MyArray(2) = 'Two'
CLEAR
? oMyForm.MyArray(1)   && 显示 1
? oMyForm.MyArray(2)   && 显示 2
```

如果所指定的属性名称不存在，将创建这个属性，并将返回一个逻辑真值(.T.)。

如果所指定的属性名称已经存在，将返回一个如下值：

- 真(.T.)。如果新属性为数组属性并且现有的属性也是数组属性。数组的大小将重新设置为新数组的大小。如果用 eNewValue 指定了值，数组中的所有元素都设置为该值。如果省略了 eNewValue，所有的数组元素都设置为假(F)。
- 真(.T.)。如果新属性不是数组属性并且现有的属性是数组属性，属性将保留为数组属性。如果用 eNewVValue 指定了值，数组中的所有元素都设置为该值。如果省略了 eNewValue，数组元素保持不变。
- 真(.T.)。如果新属性不是数组属性并且现有的属性也不是数组属性或只读的 VisualFoxPro 本地属性。如果用 eNewValue 指定了值，现有的属性都设置为该值。如果省略了 eNewVValue，现有的属性值保持不变。
- 假(F)。如果新属性是数组属性并且现有的属性不是数组属性。现有的属性

保持不变。

- 如果现有的属性是只读的 VisualFoxPro 本地属性，例如 BaseClass 属性时，将出现“属性<*属性 Name*>是只读的”错误信息。
- 如果属性名称是无效的（属性名称包括了空格或其他非法字符），将出现“不正确的属性名称”的错误信息。

应用于

ActiveDoc，复选框，列，组合框，命令按钮，命令按钮组，容器对象，控件对象，临时表对象，自定义对象，数据环境对象，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，选项按钮组，页面，页框，项目挂接对象，关系对象，屏幕，分隔符对象，形状，微调，文本框，计时器，工具栏

请参阅

DEFINE CLASS, NewObject 方法, SaveAs 方法, SaveAsClass 方法

AddToSCC 方法

将项目中的一个文件添加到源代码管理器。

语法

Objcet.AddToSCC( )

说明

如果该文件成功地添加到源代码管理器，则返回“真” (.T.)。**如果该文件不能添加到源代码管理器，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

CheckIn **方法**, CheckOut **方法**, GetLatestVersion **方法**, RemoveFromSCC **方法**, UndoCheckOut **方法**

ADEL（） **函数**

从一个数组中删除一个、一行或一列元素。

语法

ADEL(ArrayName, nElementNumber [, 2])

返值类型

数值型

参数描述

ArrayName

指定一个数组，ADEL()函数删除这个数组的一个元素或一行一列。

nElementNumber

指定从数组中删除第几个元素，或者第几行或第几列。如果要从数组中删除一列，必须包含可选参数 2。

有关如何引用数组中元素的详细内容，请参阅 DIMENSION。

2

从数组中删除一列。

说明

从数组中删除一个、一行或一列元素，但并不改变数组的大小；后续元素、行或列元素向数组的起始方向移动，并把最后一个元素、行或列设置为“假”(.F.)。

如果成功地删除了一个元素、行或列，则返回 1。

示例

下例将建立并填充一个数组，搜索一个公司的名称并删除它。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer    && 打开 customer 表
SELECT company FROM customer ;
    WHERE country = 'UK' ;
    INTO ARRAY gaCompanies
gnCount = _TALLY
gcName = 'Seven Seas Imports'
```

```
CLEAR
DISPLAY MEMORY LIKE gaCompanies
gnPos = ASCAN(gaCompanies, gcName)  && 搜索公司名
IF gnPos != 0
    * Company found, remove it from the array
    = ADEL(gaCompanies, gnPos)
    gnCount = gnCount - 1
ENDIF
DISPLAY MEMORY LIKE gaCompanies
```

请参阅

ACOPY( ), ADIR( ), AELEMENT( ), AFIELDS( ), AINS( ), ALEN( ), ASCAN( ), ASORT( ), ASUBSCRIPT( ), DIMENSION

ADIR（）函数

将文件信息存放到数组中。

语法

ADIR(ArrayName [, cFileSkeleton [, cAttribute]])

返值类型

数值型

参数描述

ArrayName

指定数组名。如果数组不存在，Visual FoxPro将自动创建此数组。如果数组存在，但其大小不足以包含所有信息，则Visual FoxPro自动增加数组大小，使得数组能容纳所有信息。如果数组超过了所需大小，Visual FoxPro将截掉多余部分。如果数组存在，并且ADIR（）函数由于没找到匹配文件而返回0，则数组保持不变。如果数组不存在，并且ADIR（）函数返回零，则不创建数组。

下表说明了数组中每列的内容及其数据类型：

列	数组内容	数值类型
1	文件名	字符型
2	文件大小	数值型
3	文件日期	日期型
4	文件时间	字符型
5	文件属性	字符型

数组的最后一列包含匹配文件的文件属性。每个文件属性值由一个字母表示，一个文件可有多个属性。下表说明每个字母表示的文件属性含义：

字母	属性
A	档案文件 — 可读写
H	隐藏文件

续表

R	只读文件
S	系统文件
D	目录

cFileSkeleton

指定文件梗概，以便存储满足搜索条件的文件名或扩展名的文件信息。例如，条件可以是所有表、所有文本文件、所有文件名第一个字母为“A”的文件，等等。这些通配查询在 *cFileSkeleton* 中可以包含通配符 * 和 ?。其中问号代表单个字符，星号代表任意字符串。在文件梗概中，可在任意位置使用任意个数的通配符。

可以指定驱动器和目录名，程序将在此驱动器和目录下搜索匹配文件。如果不指定驱动器和目录名，将把当前目录下的文件信息存入数组中。

CAttribute

在返回内容中，指定包含子目录或嵌套文件夹、隐藏或系统文件、或者卷名。

CAttribute 可以是 D、H 和 S 的任意组合。如果包含 D，除了返回匹配 *cFileSkeleton* 的文件名外，还将返回当前目录的子目录或当前文件夹的嵌套文件夹。如果包含 H，将返回与 *cFileSkeleton* 指定相匹配的隐藏文件信息。如果包含 S，将返回与 *cFileSkeleton* 指定通配文件名相匹配的系统文件信息。

如果 *cFileSkeleton* 为空字符串，仅返回子目录或嵌套文件夹名、隐藏或系统文件。

CAttribute 参数中包含字符 V，将返回当前驱动器的卷名。如果 V 和 D、H 或 S 一起包

含在数组中，则只返回卷名。卷名存入数组的第一个元素中，并截去数组的其余部分。

说明

对于每一个文件，ADIR（）将文件名、大小、日期、时间和属性信息存入数组。

示例

ADIR( ) 创建了一个包含数据库信息的数组。另外还显示了数据库的名称。

```
CLOSE DATABASES
SET PATH TO (HOME(2) + 'Data')

gnDbcnumber = ADIR(gaDatabase, '*.DBC') && 创建数组

CLEAR
FOR nCount = 1 TO gnDbcnumber && 循环数据库的编号
   ? gaDatabase(nCount,1) && 显示数据库名称
ENDFOR
SET PATH TO HOME（） && 设置 Visual FoxPro 目录的路径
```

请参阅

A , AELEMENT( ) , AFIELDS( ) , AINS( ) , ALEN( ) , ANETRESOURCES( ) , ASCAN( ) , ASORT( ) , ASUBSCRIPT( ) , DIMENSION, DIR 或 DIRECTORY

AELEMENT() 函数

由元素下标返回数组元素的编号。

语法

AELEMENT(ArrayName, nRowSubscript[, nColumnSubscript])

返值类型

数值型

参数描述

ArrayName

指定想要返回元素编号的数组名。

NRowSubscript

指定行下标。如果数组为一维数组，AELEMENT() **函数的返回值为** *nRowSubscript*。

如果仅有参数 *nRowSubscript*，**并且此值大于数组的行总数，将产生错误信息。**

NColumnSubscript

Visual FoxPro **指定列下标。如果数组为二维数组，则需要** *nRowSubscript* **和** *ColumnSubscript* **两个参数。**

说明

有两种方法引用二维数组中的元素。第一种方法用两个下标值指定元素在数组中的行和列，另一种方法则给出单个元素的编号。在第一种方法中提供元素的行和列下标后，AELEMENT()函数能返回元素的编号。

Visual FoxPro函数ADEL()、ADIR()、AFIELDS()、AINS()、ALEN()、ASCAN()、ASORT()和ASUBSCRIPT()可以操作二维数组，并且需要通过元素编号引用元素。AELEMENT()为这些函数提供了由下标值转化为元素编号的方法。使用ASUBSCRIPT()函数可由编号值返回相应的行和列下标值。

下面的示例说明了如何创建一个具有二行三列的数组。DISPLAY MEMORY命令按元素编号顺序显示每个元素的内容。

```
DIMENSION gaMyArray(2,3)
DISPLAY MEMORY LIKE gaMyArray
gaMyArray  Pub  A
   ( 1, 1)  L  .F. (element number 1)
   ( 1, 2)  L  .F. (element number 2)
   ( 1, 3)  L  .F. (element number 3)
   ( 2, 1)  L  .F. (element number 4)
   ( 2, 2)  L  .F. (element number 5)
   ( 2, 3)  L  .F. (element number 6)
```

一个元素可通过下标或编号来引用。命令STORE 'INVOICE' TO gaMyArray(2,1)和STORE 'INVOICE' TO gaMyArray(4)都将字符串INVOICE存入同一数组元素中。在一维数组中，元素编号与行下标相同。

因此对于一维数组，没有必要使用AELEMENT()函数。

请参阅

ADEL( ), ADIR( ) , AFIELDS( ) , AINS( ) , ALEN( ) , ASCAN( ) , ASORT( ) , ASUBSCRIPT( ) , DIMENSION, DISPLAY MEMORY

AERROR（）函数

创建一个变量数组，数组中包含最近的 Visual FoxPro、OLE **或** ODBC **的错误信息。**

语法

AERROR(ArrayName)

返值类型

数值型

参数描述

ArrayName

指定 AERROR（）**函数创建的数组名。**

说明

AERROR（）**函数创建的数组有六列，并且返回数组的行数。行数由产生的错误类型决定。**

下表描述了 Visual FoxPro 产生错误时，数组中每个元素的内容。当发生 Visual FoxPro 错误时，数组只有一行。

元素编号	说明
1	数值型，这是一个错误编号，与 ERROR（）函数返回的值相同。
2	字符型，错误文本信息，与 MESSAGE（）函数返回的值相同。
3	null 值，但是如果错误具有附加错误参数，则包含错误参数的文本信息，与 SYS(2018) 的返回值相同。
4	null 值，但是在适当的时候，包含发生错误的工作区编号。
5	null 值，但是当触发失败时（错误 1539），包含下列数值之一： 1 –插入触发失败。 2 更新触发失败。 3 –删除触发失败。
6	null 值。
7	null 值。

下表描述了发生 OLE 错误 1427 和 1429 时各元素的内容。当这些 OLE 错误发生时，数组只有一行。

元素编号	说明
1	数值型，为 1427 或 1429。
2	字符型，Visual FoxPro 的错误信息文本。
3	字符型，OLE 错误信息文本。
4	字符型，应用程序名（例如，Microsoft Excel）。.
5	null 值或字符，如果能从应用程序的帮助文件中得到更详细的有关错误的信息，则此处包含应用程序中保存这些信息的帮助文件名，否则为 null 值。
6	如果能从应用程序中得到有关信息，此处存放相应帮助主题的帮助文本中的主题标识，否则为 null 值。.
7	数值型，OLE 2.0 的异常编号。

下表描述了发生 ODBC 错误 1526 时各元素的内容。当发生 ODBC 错误时，数组可能包含两行或更多行，每一行为一个 ODBC 错误。

元素编号	说明
1	数值型.为 1526.
2	字符型，错误信息文本。

续表

3	字符型，ODBC 错误信息文本。
4	字符型，当前的 ODBC SQL 状态。
5	数值型，ODBC 数据源的错误编号。
6	数值型，ODBC 连接句柄。
7	null 值。

错误信息的数组，并且显示此信息。

示例

下面的示例运用 ON ERROR 指定了名称为 errhand 的错误处理例程。错误是由发出拼写错误的命令(BRWS) 引起的。

```
ON ERROR DO errhand   && errhand 是一个错误处理程序

BRWS &&产生语法错误
ON ERROR &&恢复系统错误处理程序

PROCEDURE errhand
   =AERROR(aErrorArray) &&反映最新错误的数据
   CLEAR
   ? 'The error provided the following information' &&显示信息
   FOR n = 1 TO 7 &&显示数组的所有元素
      ? aErrorArray(n)
   ENDFOR
```

请参阅

COMRETURNERROR(), CREATE TRIGGER, ERROR(), MESSAGE(), ON ERROR, SYS(2018)

AFIELDS（）函数

把当前表的结构信息存放在指定数组中，并且返回表的字段数。

语法

AFIELDS(ArrayName [, nWorkArea | cTableAlias])

返值类型

数值型

参数描述

ArrayName

指定的数组名，将表结构信息存放在这个数组中。如果 AFIELDS（）**函数指定的数组不存在，** Visual FoxPro **将自动创建此数组。如果数组存在，但大小不足以包含** AFIELDS（）**函数返回的所有信息，** Visual FoxPro **将自动增加数组大小，使数组能容纳所有信息。**

nWorkArea

指定表所在的工作区，在这个工作区中打开的表的结构信息要存放到数组中。

cTableAlias

指定表的别名。此表的结构信息要存放到数组中。

如果省略参数 *nWorkArea* 和 *cTableAlias*，将把当前选定工作区中的表结构信息存入数组中。

下表描述了数组中每列的内容和每列信息的数据类型，每行对应表的一个字段。

列号	字段信息	数据类型
1	字段名	字符型
2	字段类型: C = 字符型 D = 日期型 L = 逻辑型 M = 备注型 N = 数值型 F = 浮点型 I = 整型 B = 双精度型 Y = 货币型 T = 日期时间型 G = 通用型	字符型
3	字段宽	数值型
4	小数点位数	数值型
5	是否允许 null 值	逻辑型
6	是否不允许代码转化	逻辑型
7	字段有效性规则	字符型

续表

8	字段有效性说明	字符型
9	字段默认值	字符型
10	表有效性规则	字符型
11	表有效性说明	字符型
12	长表格名称	字符型
13	插入触发器表达式	字符型
14	更新触发器表达式	字符型
15	删除触发器表达式	字符型
16	表格注释	字符型

说明

AFIELDS() 函数返回表的字段数。数组包含 11 列，行数与表中字段数相同。

使用 COPY STRUCTURE EXTENDED 命令可以将类似的信息复制到一个表而不是一个数组中。

示例

下面的示例创建了名称为 gaMyArray 的数组，它包含了关于 customer 表格的字段信息，并且显示了字段名称。

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

gnFieldcount = AFIELDS(gaMyArray)  && 创建数组
```

```
CLEAR
FOR nCount = 1 TO gnFieldcount
    ? gaMyArray(nCount,1)  && 显示字段名称
ENDFOR
```

请参阅

ADEL( ), ADIR( ) , AELEMENT( ) , AINS( ) , ALEN( ) , ALTER TABLE-SQL, ASCAN( ) , ASORT( ) , ASUBSCRIPT( ) , COPY STRUCTURE EXTENDED, CREATE, CREATE TABLE, DIMENSION

AFONT( ) 函数

将可用字体的信息放到一个数组中。

语法

AFONT(ArrayName [, cFontName [, nFontSize]])

返值类型

逻辑型

参数描述

ArrayName
指定存放可用字体名的变量数组。如果数组大小不足以包含所有字体，Visual FoxPro 将自动增加数组大小。如果指定的是已有的二维数组，Visual FoxPro 将把数组转化为一维数组。

如果创建数组成功，则 AFONT（）函数的返回值为“真”(.T.)，否则返回值为“假”(.F.)。

cFontName
指定需存放信息到数组中的字体名。

如果指定字体仅支持离散的字体尺寸大小（8 磅、10 磅...），这些字体尺寸大小将存入数组，并且 AFONT（）函数的返回值为“真”(.T.)。如果参数 *cFontName* 指定的字体为可缩放的（支持连续字体大小值），数组中将包含一个值为 -1 的元素，并且 AFONT（）函数的返回值为“真”(.T.)。

如果指定了无效的字体名，将不创建数组并且 AFONT（）函数返回“假”(.F.)。

nFontSize
指定字体 *cFontName* 的大小。

如果指定的 *cFontSize* 对字体 *FontName* 可用，则数组将包含一个值为“真”(.T.) 的元素，并且 AFONT（）函数返回“真”(.T.)。如果指定的字体大小对指定字体无效，将

不创建数组并且 AFONT（）函数返回“假”(.F.)。

说明

AFONT（）函数将可用字体的名称存放在数组中，此函数还可用于确定有效的字体大小或字体是否可缩放。用 GETFONT（）函数可以显示包含有效字体、字体大小和样式信息的对话框。

示例

下面的示例运用 AFONT( )创建了一个包含所有的可用字体的名称的数组，并将显示每种字体的名称和示范。如果安装了 10 种以上的字体，将只显示前 10 种。

```
CLEAR
=AFONT(gaFontArray) && 包含字体名称的数组
gnNumFonts = ALEN(gaFontArray) && 字体编号
IF gnNumFonts > 10
   gnNumFonts = 10 && 显示前 10 种字体
ENDIF

FOR nCount = 1 TO gnNumFonts
   ? ALLTRIM(gaFontArray(nCount)) && 显示字体名称
   ?? '  This is an example of ' ;
      + ALLTRIM(gaFontArray(nCount)) FONT gaFontArray(nCount), 8
ENDFOR
```

请参阅

FONTMETRIC( ), GETFONT( ), TXTWIDTH( ), SYSMETRIC( ), WFONT( )

AfterBuild 事件

当重新连编一个项目之后，或者从一个项目创建一个应用程序文件 (.app)、动态链接库 (.dll) 或可执行文件 (.exe) 之后发生。

语法

PROCEDURE Object.AfterBuild
[LPARAMETERS nError]

参数描述

nError

本参数是当重新连编一个项目之后或创建一个 .app、.dll 或 .exe 文件之后，Visual FoxPro 返回的错误代码。如果 nError 为 0，当重新连编一个项目时或创建一个 .app、.dll 或 .exe 文件时,就不会发生错误。

应用于

项目挂接对象

请参阅

BeforeBuild 事件, Build 方法

AfterCloseTables 事件

在表单、表单集或报表的数据环境中，释放指定表或视图后，将发生此事件。

语法

PROCEDURE DataEnvironment.AfterCloseTables

说明

对于表单或表单集，AfterCloseTable 事件发生在表单集或表单的 Unload 事件发生之后，另外还发生在由数据环境打开的表或视图关闭之后。

对于报表，AfterCloseTable 事件发生在由数据环境打开的任意表和视图关闭之后。

在任何时候调用 CloseTables 方法，都会发生 AftercloseTables 事件。AfterCloseTable 事件发生后，将发生数据环境和其相关对象的 Destroy 事件。

应用于

数据环境

请参阅

CloseTables 方法, Destroy 事件, Unload 事件

AfterDock 事件

停放工具栏后，将发生此事件。

语法

PROCEDURE ToolBar.AfterDock
[LPARAMETERS nIndex]

参数描述

nIndex

当一个控件存放在一个控件数组中时，指定它在控件数组中的唯一索引标识。

说明

当用户停放一个工具栏或调用 Dock 方法时，发生 AfterDock 事件。

应用于

工具栏

请参阅

BeforeDock 事件, Dock 方法, Undock 事件

AfterRowColChange 事件

当用户移到表格的另一行或列时，新单元获得焦点以及新行或列中对象的 When 事件发生后，发生此事件。如果新行或列中对象的 When 事件不返回“真”(.T.)，则不触发 AfterRowColChange 事件。

语法

PROCEDURE Grid.AfterRowColChange
LPARAMETERS nColIndex

参数描述

nColIndex

返回最新选择的行或列的索引。

说明

触发 AfterRowColChange 事件的方法可以通过交互式使用鼠标或键盘，或用编程的方法，或调用 ActivateCell 方法。

应用于

表格

请参阅

ActivateCell 方法, BeforeRowColChange 事件, When 事件

AGETCLASS( ) 函数

在“打开”对话框中显示类库，并且创建一个包含该类库和所选类名称的数组。

语法

AGETCLASS(*ArrayName* [, cLibraryName [, cClassName [, cTitleText [, cFileNameCaption [, cButtonCaption]]]]])

返值类型

逻辑型

参数描述

ArrayName

指定数组的名称，在该数组中保存类库和类的名称。如果所指定的数组不存在，Visual FoxPro 会自动创建该数组。如果该数组存在，但是大小不足以容纳类库和类的名称，则 Visual FoxPro 自动增加该数组的大小。如果该数组比所需的要大，则 Visual FoxPro 会截短该数组。如果该数组存在，并且由于关闭了“类库”对话框（通过按 ESC 键、选择了“取消”命令或单击了“关闭”按钮）而使 AGETCLASS( ) 返回了“假” (.F.)，则该数组保持不变。

如果该数组不存在，并且 AGETCLASS（）返回了“假”(.F.)，则不创建该数组。

下表列出了当选择一个类时所创建数组的每个元素的内容。

元素	内容
1	所选类库的名称。
2	所选类库的名称。

cLibraryName

指定当显示“打开”对话框时初始选中的类库的名称。所指定的类库的名称显示在“文件名”文本框中。如果所指定的类库不存在，或者 *cLibraryName* 是空字符串或 null 值，则会产生一个错误。

cClassName

指定当显示“打开”对话框时在“类名”列表中初始选中的类的名称。如果所指定的类不存在，则选中“类名”列表中的第一个类。如果省略 cLibraryName，或 cClassName 是 null 值，则会产生一个错误。

cTitleText

指定在“打开”对话框的标题栏显示的文本。在默认情况下，显示“打开”。

cFileNameCaption

指定在“文件名”文本框旁边显示的文本。在默认情况下，显示“文件名”。

cButtonCaption

指定 OK 按钮的标题。

说明

如果您选择了一个类，则 AGETCLASS（）返回“真“ (.T.)，并且创建一个包含两个元素的一维数组。第一个元素包含所选类库的名称；第二个元素包含所选类的名称。如果退出了“类库”对话框（通过按 ESC 键、选择了“取消”命令或单击了“关闭”按钮），则返回“假” (.T.)。

示例

下面的示例创建了一个名为 aClassLib 的数组。目录更改到包含示范类的子目录 Samples。AGETCLASS( )显示带有 Buttons 类库和选定的 VCR 类的对话框。如果选择了“修改”按扭，类库的名称和选定的类将存储在数组中。然后在“类设计器”中打开类。

```
LOCAL aClassLib(2) && 创建一个数组，初始化为 .F.
cCurrentDir = CURDIR（） && 保存当前目录
CD HOME(2) + 'CLASSES' && 转换目录

AGETCLASS(aClassLib, 'BUTTONS.VCX', 'VCR', 'Modify Class', ;
    'Class File:', 'Modify') && 显示对话框
CD (cCurrentDir) && 转换到先前目录

IF TYPE('aClassLib(2)') = 'C' && 选择了类？
    MODIFY CLASS (aClassLib(2)) OF (aClassLib(1)) && 打开并修改
ENDIF
```

请参阅

ACLASS( ), AMEMBERS( ), AVCXCLASSES( ) , CREATE CLASS

AGETFILEVERSION（）函数

创建一个数组，其中包含有关文件的 Windows 版本资源的信息，例如 .exe、.dll 和 .fll 文件，或在 Visual FoxPro 中创建的自动服务文件。

语法

AGETFILEVERSION(*ArrayName*, cFileName)

返值类型

数值型

参数描述

ArrayName

指定数组的名称，在该数组中放置文件信息。所指定的数组不存在，Visual FoxPro 会自动创建该数组。如果该数组存在，但是大小不足以容纳文件信息，则 Visual FoxPro 自动增加该数组的大小。如果该数组比所需的要大，则 Visual FoxPro 会截短该数组。

下表列出了该数组每个元素的内容。

元素	内容
1	注释
2	公司名
3	文件说明
4	文件版本
5	内部名称
6	合法版权
7	合法商标
8	原有文件名
9	私有连编
10	产品名
11	产品版本
12	特殊连编
13	OLE 自注册（如果文件支持自注册，则包含 "OLESelfRegister"；否则包含空字符串）
14	语言（从导出）

续表

15	翻译代码。例如，可以用以下代码来判断 VisualFoxPro 可执行文件的 LocaleID： DIMENSION aFiles[1] AGETFILEVERSION(aFiles,"VFP6.EXE") ? EVAL("0x"+LEFT(aFiles[15],4)) ** Returns 1033 for US version

cFileName

指定文件名，该文件的信息放在数组中。所指定的类库名显示在“文件名”文本框中。如果所指定的类库不存在，会产生一个错误。

说明

通常，AGETFILEVERSION（）用于获得文件的 Windows 版本资源的信息，例如 .exe、.dll 和 .fll 文件，或在 Visual FoxPro 中创建的自动服务文件。为了获得 Windows 版本资源，必须在“EXE 版本”对话框中为一个 Visual FoxPro 自动服务程序至少指定一项。

AGETFILEVERSION（）返回数组的元素数。如果所指定的文件没有 Windows 版本资源，会返回零，并且数组（如果已经创建）保持不变。

AGETFILEVERSION（）最小可以截短为 5 个字符。

请参阅

ADIR(), DIR 或 DIRECTORY

AINS（）函数

往一维数组中插入一个元素，或者向二维数组中插入一行或一列元素。

语法

AINS(ArrayName, nElementNumber[, 2])

返值类型

数值型

参数描述

ArrayName

指定需插入元素的数组名。

NElementNumber

指定在数组的第几个元素，或第几行（或列）处插入新元素。

若要往一维数组中插入新元素时，请包含参数 *ArrayName* 和插入位置 *nElementNumber*，新元素插到元素 *nElementNumber* 前。若要往二维数组中插入行时，请包含参数 *ArrayName* 和行号 *nElementNumber*，新行正好插到参数 *nElementNumber* 指定的行前。

有关用下标引用数组元素列的详细内容，请参阅 DIMENSION。

2

往二维数组中插入一列。新列正好插到参数 *nElementNumber* 指定的列前。

说明

往数组中插入一个元素、一行或一列，并不改变数组大小。后续的元素、行或列将依次往数组末尾移一步，并丢弃数组中最后一个元素、一行或一列的数据。新插入的元素、行或列初始化为“假”(.F.)。

如果元素、行或列成功地插入到数组中，则 AINS() 函数返回 1。

示例

下面的示例创建并用公司名称填充一个数组，还在数组中查找指定的公司名称。如果没有找到，则在数组中插入遗漏的公司名称。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer    && 打开 customer 表
SELECT company FROM customer ;
    WHERE country = 'Germany' ;
    INTO ARRAY gaCompanies

gnCount = _TALLY
gcName = 'Seven Seas Imports'
CLEAR
DISPLAY MEMORY LIKE gaCompanies

IF ASCAN(gaCompanies, gcName) = 0  && 查找公司
```

```
*** Company not found-add it ***
    DIMENSION gaCompanies[gnCount+1,1]
    = AINS(gaCompanies, gnCount-1)
    gaCompanies[gnCount-1] = gcName
ENDIF
DISPLAY MEMORY LIKE gaCompanies
```

请参阅

ACOPY( ), ADEL( ) , ADIR( ) , AELEMENT( ) , AFIELDS( ) , ALEN( ) , ASCAN( ) , ASORT( ) , ASUBSCRIPT( ) , DIMENSION

AINSTANCE( ) 函数

将一个类的实例存放到变量数组中，并且返回数组中存放的实例个数。

语法

AINSTANCE(ArrayName, cClassName)

返值类型

数值型

参数描述

ArrayName

指定存放实例的数组名。如果指定数组不存在，Visual FoxPro 将自动创建指定的数组。如果数组存在，但其大小不足以包含所有实例，Visual FoxPro 则自动增加数组大小，使数组能容纳所有实例。如果数组大小大于所需值，Visual FoxPro 将截掉多余部分。如果数组存在，但由于没找到实例使 AINSTANCE（）函数的返回值为 0，则不改变数组。如果数组不存在，并且 AINSTANCE（）函数的返回值为 0，将不创建指定数组。

只有用 CREATEOBJECT（）赋给变量和数组元素的类实例才存放到数组中。

CClassName

指定 Visual FoxPro 基类名或用户自定义类名。下表列出了可以给 *cClassName* 指定的 Visual FoxPro 基类。

基类名称		
ActiveDoc	CheckBox	Column
ComboBox	CommandButton	CommandGroup
Container	Control	Cursor
Custom	DataEnvironment	EditBox
Form	FormSet	Grid
Header	Hyperlink	Image
Label	Line	ListBox
OLEBoundControl	OLEControl	OptionButton

续表

OptionGroup	Page	PageFrame
ProjectHook	Relation	Separator
Shape	Spinner	TextBox
Timer	ToolBar	

示例

在下面的示例中，将用 CREATEOBJECT（）函数创建 VisualFoxPro 表单基类的两个实例。然后用 AINSTANCE（）函数创建数组 gaMyArray，数组中包括每个表单实例的变量引用（goINSTANCE1 和 goINSTANCE2）。最后显示数组的内容。

```
CLEAR ALL
goINSTANCE1 = CREATEOBJECT('Form')
goINSTANCE2 = CREATEOBJECT('Form')

CLEAR
? AINSTANCE(gaMyArray, 'Form') && 返回 2, 两个表单间距
DISPLAY MEMORY LIKE gaMyArray && 显示内容
```

请参阅

ADD CLASS, AMEMBERS(), CREATE CLASS, CREATE CLASSLIB, CREATEOBJECT(), DEFINE CLASS

ALEN（）函数

返回数组中元素、行或列的数目。

语法

ALEN(ArrayName [, nArrayAttribute])

返值类型

数值型

参数描述

ArrayName

指定数组名。如果参数仅包含数组名，ALEN（）函数则返回元素的数目。

NArrayAttribute

确定 ALEN（）函数返回的是数组元素的数目、数组的行数、还是数组的列数。

NArrayAttribute 可以取值为 0、1 或 2。

0	指定返回数组元素数目。省略 *nArrayAttribute* 与指定 *nArrayAttribute* 为 0 作用相同。
1	指定返回数组的行数。

续表

2	指定返回数组的列数。如果数组是一维数组，则 ALEN（）函数返回 0（没有列）。

示例

下面的示例用 AFONT( )创建了一个包含所有可用字体的名称的数组。ALEN（）可用于判断数组的行数。将显示每个字体名称和示范。如果安装了 10 种以上的字体，则只显示前 10 种。

```
CLEAR
=AFONT(gaFontArray) && 包含可用字体名称的数组
gnNumFonts= ALEN(gaFontArray) && 字体名称
IF gnNumFonts > 10
   gnNumFonts = 10 && 显示前 10 种字体
ENDIF

FOR nCount = 1 TO gnNumFonts
   ? ALLTRIM(gaFontArray(nCount)) && 显示字体名称
   ?? '  This is an example of ' ;
      + ALLTRIM(gaFontArray(nCount)) FONT gaFontArray(nCount), 8
ENDFOR
```

请参阅

ADEL( ), ADIR( ), AELEMENT( ), AFIELDS( ), AINS( ), ASCAN( ), ASORT( ), ASUBSCRIPT( ), DIMENSION, STORE

Alias 属性

指定与临时表对象相关的每个表或视图的别名。

语法

DataEnvironment.Cursor.Alias[= cText]

参数描述

cText

指定与临时表对象相关的表或视图的别名。

说明

加载数据环境后，与临时表对象相关的表或视图都分配了一个别名。在默认状态下，别名与表或视图名相同。使用 Alias 属性可以改写默认的别名。

Alias 属性与 USE 命令的 Alias 子句功能相似。

应用于

临时表

请参阅

USE

ALIAS（）函数

返回当前表或指定工作区表的别名。

语法

ALIAS([nWorkArea | cTableAlias])

返值类型

字符型

参数描述

nWorkArea

指定的工作区编号，ALIAS（）函数返回此工作区表的别名。

cTableAlias

指定的表别名，ALIAS（）函数返回此表的别名。

如果省略参数 *nWorkArea* 或 *cTableAlias*，ALIAS（）函数将返回在当前工作区中打开的表的别名。如果当前或指定工作区没有打开的表，则函数返回空字符串。

示例

下面的示例打开表 customer，并且显示别名。表 customer 在另一个工作区又被打开，赋予 MyCustomer 的别名，此别名也被显示。

```
CLOSE DATABASES
```

```
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer    && 打开 customer 表
CLEAR
?ALIAS()  && 显示别名
SELECT 0
USE customer AGAIN ALIAS MyCustomer  && 使用不同的别名
?ALIAS()  && 显示别名
```

请参阅

DBF( ), SELECT( )

Align 属性

指定表单上 ActiveX™ 控件 (.OCX) 的对齐方式。

语法

OLEContainerControl.Align[= nAlign]

参数描述

nAlign

Align **属性的设置是：**

设置	说明
0	标准对齐方式。在表单上 ActiveX 控件的位置与 OLE 容器控件的位置相同。
1	顶部。ActiveX 控件放置在表单的顶部。
2	底部。ActiveX 控件放置在表单的底部。
3	左边。ActiveX 控件放置在表单的左边。
4	右边。ActiveX 控件放置在表单的右边。

说明

ActiveX 控件（.OCX 文件）放置在 OLE 容器控件中。

Align 属性仅用于那些对更改对齐方式提供支持的 ActiveX 控件。可插入的 OLE 对象（如 Microsoft Excel 工作表）不支持 Align 属性。

Align 属性仅用于表单上的 ActiveX 控件。

如果有二个或多个 ActiveX 控件具有相同的 Align 属性设置，则这些控件将相互重叠。

应用于

OLE 容器控件

请参阅

OLE 容器控件

Alignment 属性

指定与控件相关的文本的对齐方式。设计和运行时可用。

语法

Control.Alignment[= nAlign]

参数描述

nAlign

对于复选框或选项按钮控件，Alignment 属性的设置可以为：

设置	说明
0	（默认设置）左对齐。控件左对齐，文本安排在控件的右边。
1	（中东版 Windows 的默认设置）右对齐。控件右对齐，文本安排在控件的左边。

对于组合框、编辑框、标题、标签或微调控件，Alignment 属性的设置可以为：

设置	说明
0	（默认设置）左对齐。组合框、标题、和标签的默认设置。
1	（中东版 Windows 的默认设置）右对齐。微调的默认设置。
2	居中对齐。文本放置正中间。左右所留空间相等。

续表

3	自动。根据控件源的数据类型对齐文本。

对于文本框控件，Alignment 属性的设置可以为：

设置	说明
0	左对齐。
1	右对齐。
2	居中对齐。文本放置在正中间，左右所留空间相等。
3	（默认设置）自动。对于不包含在列中的文本框控件，文本对齐方式取决于控件源的数据类型。数值类型（数值型、双精度型、浮点型、货币型和整型）采用右对齐；其他数据类型采用左对齐。对于包含在列中的文本框控件，列的对齐方式决定了文本框中文本的对齐方式。

对于列，Alignment 属性的设置可以为：

设置	说明
0	中左对齐。将文本左对齐，并且垂直居中。
1	中右对齐。将文本右对齐，并且垂直居中。
2	居中对齐。文本放置在正中间，左右所留空间相等，并且垂直居中。

续表

3	（默认设置）自动。文本对齐方式取决于控件源的数据类型。数值类型（数值型、双精度型、浮点型、货币型和整型）采用右对齐；其他数据类型采用左对齐。
4	上左对齐。将文本在列的上面左对齐。
5	上右对齐。将文本在列的上面右对齐。
6	上中对齐。将文本在列的上面居中对齐。
7	下左对齐。将文本在列的下面左对齐。
8	下右对齐。将文本在列的下面右对齐。
9	下中对齐。将文本在列的下面居中对齐。

说明

对于组合框，仅当 Style 属性为 0 时，才可用。

应用于

复选框，列，组合框，编辑框，标头，标签，选项按扭，微调，文本框

请参阅

AutoSize 属性，Caption 属性

_ALIGNMENT 系统变量

在页边距之间对齐文本。可用报表设计器代替。

ALINES（）函数

将一个字符表达式的或备注字段中的每一行复制到一个数组相应行。

语法

ALINES(ArrayName, cExpression [, lTrim])

返值类型

数值型

参数描述

ArrayName

指定数组的名称，将字符表达式或备注字段中的每一行复制到该一个数组中。如果所指定的数组不存在，Visual FoxPro会自动创建该数组。如果该数

组存在，但是大小不足以容纳备注字段中的每一行，则 Visual FoxPro 自动增加该数组的大小。如果该数组比所需的要大，则 Visual FoxPro 会截短该数组。

cExpression

指定字符表达式或备注字段，其中包含要复制到数组中的行。如果 cExpression 是空字符串或 null 值，会创建一个具有单行的数组，并且该行包含空字符串。

lTrim

指定是否从复制到数组的行中删除前导和后缀空格。如果 lTrim 为“真” (.T.)，则从行中删除前导和后缀空格。如果 lTrim 为“假” (.F.) 或省略，则不删除前导和后缀空格。

说明

ALINES() 返回数组的行数（或者，同样地返回字符表达式或备注字段的行数）。字符表达式或备注字段的第一行复制到数组的第一行，字符表达式或备注字段的第二行复制到数组的第二行，依次类推。

一个换行符 (CHR(10)) 或回车符 (CHR(13)) 表明了一行的结束。也可以使用这些字符的组合 (CHR(10) + CHR(13)或 CHR(13) + CHR(10)) 表明一行的结束。

ALINES() 提供了分析字符表达式或备注字段中各行的简单方法。虽然 MLINES() 也用来分析字符表达式或备注字段，但是 ALINES() 更快，并且需要更少的编程。另外，ALINES() 不受 SET MEMOWIDTH 的值的影响。

为了将字符表达式或备注字段中的行复制到数组，必须有足够的内存。如果内存不

足，Visual FoxPro 会生成一条错误信息。

示例

下面的程序在数据库 Testdata 中打开表 Employee。ALINES( )用于复制 Notes 备注型字段中的直线到名为 aMyArray 的数组中，然后显示数组的内容。

本示例中，ALINES( )返回 1，因为在备注字段中输入雇员信息时，句子后没有按 Enter 键。

```
CLOSE DATABASES
CLEAR
SET TALK OFF
OPEN DATABASE (HOME(2) + 'data\testdata')
USE employee  && 打开 Employee 表
? ALINES(aMyArray, employee.notes)  && 显示 1
? aMyArray(1)
```

请参阅

MEMLINES( ), MLINE( ) , _MLINE, SCATTER

AllowAddNew 属性

指定是否可以从一个表格中将新记录添加到表中。设计和运行时可用。

语法

Grid.AllowAddNew[= lExpr]

参数描述

lExpr

取下列一个值：

设置	说明
“真” (.T.)	可以从一个表格中将新记录添加到表中。
“假” (.F.)	（默认值）不可以从一个表格中将新记录添加到表中。

说明

如果 AllowAddNew 设置为“真” (.T.)，只要表格是可读写的，通过将记录指针放在表格的最后一个记录，并且按向下箭头，就可以将一个新记录添加到表中。如果表格是只读的（如果 RecordSourceType 是一个查询，表是只读的，等等），则不可以从一个表格中将新记录添加到表中。

应用于

表格

请参阅

APPEND

AllowHeaderSizing 属性

指定在运行时刻，一个表格的标头高度是否可以改变。设计和运行时可用。

语法

Grid.AllowHeaderSizing[= lExpr]

参数描述

lExpr

取下列一个值：

设置	说明
“真” (.T.)	（默认值）在运行时刻表格的标头高度可以通过拖动标头改变。
“假” (.F.)	在运行时刻表格的标头高度不可以通过拖动标头改变。

应用于

表格

请参阅

AllowRowSizing 属性

AllowRowSizing 属性

指定在运行时刻，一个表格的记录高度是否可以改变。设计和运行时可用。

语法

Grid.AllowRowSizing[= lExpr]

参数描述

lExpr

取下列一个值：

设置	说明
“真” (.T.)	（默认值）在运行时刻表格的标头高度可以通过拖动标头改变。
“假” (.F.)	在运行时刻表格的标头高度不可以通过拖动标头改变。

应用于

表格

请参阅

AllowHeaderSizing 属性

AllowTabs 属性

指定编辑控件中是否允许使用选项卡。

语法

EditBox.AllowTabs[= lExpr]

参数描述

lExpr

AllowTabs **属性的设置如下：**

设置	说明
“真” (.T.)	允许使用选项卡，并且按 CTRL+TAB 键退出。
“假” (.F.)	（默认）不允许在编辑框中使用选项卡，按 TAB 键使焦点移到下一控件。

应用于

编辑框

请参阅

TabIndex **属性**, TabStop **属性**

ALLTRIM（）函数

删除指定字符表达式的前后空格符，并将删除空格符后的字符串返回。

语法

ALLTRIM(cExpression)

返值类型

字符型

参数描述

cExpression

指定删除首尾空格的字符表达式。

说明

使用 ALLTRIM（）函数能确保删除用户输入数据首尾的空格字符。

示例

下面的示例用 AFONT()创建了一个包含所有可用字体的名称的数组。ALLTRIM()可用于删除字体名称中的前导与后续空格。将显示削减过的字体名称和示范。如果安装了 10 种以上的字体，则只显示前 10 种。

```
CLEAR
=AFONT(gaFontArray) && 显示包含可用字体名称的数组
```

```
gnNumFonts= ALEN(gaFontArray) && 字体的种类
IF gnNumFonts > 10
   gnNumFonts = 10 && 显示前 10种字体
ENDIF

FOR nCount = 1 TO gnNumFonts
   ?ALLTRIM(gaFontArray(nCount)) && 显示字体名称
   ?? '  This is an example of ' ;
      +ALLTRIM(gaFontArray(nCount)) FONT gaFontArray(nCount), 8
ENDFOR
```

请参阅

LTRIM( ), RTRIM( ) , TRIM( )

ALTER TABLE-SQL 命令

以编程方式修改表的结构。

语法

```
ALTER TABLE TableName1
  ADD | ALTER [COLUMN] FieldName1
     FieldType [(nFieldWidth [, nPrecision])]
     [NULL | NOT NULL]
```

```
    [CHECK lExpression1 [ERROR cMessageText1]]
    [DEFAULT eExpression1]
    [PRIMARY KEY | UNIQUE]
    [REFERENCES TableName2 [TAG TagName1]]
    [NOCPTRANS]
```

– 或 者 –

```
ALTER TABLE TableName1
  ALTER [COLUMN] FieldName2
    [NULL | NOT NULL]
    [SET DEFAULT eExpression2]
    [SET CHECK lExpression2 [ERROR cMessageText2]]
    [DROP DEFAULT]
    [DROP CHECK]
```

– 或 者 –

```
ALTER TABLE TableName1
  [DROP [COLUMN] FieldName3]
  [SET CHECK lExpression3 [ERROR cMessageText3]]
  [DROP CHECK]
  [ADD PRIMARY KEY eExpression3 TAG TagName2 [FOR
lExpression4]]
```

[DROP PRIMARY KEY]
[ADD UNIQUE eExpression4 [TAG TagName3 [FOR lExpression5]]]
[DROP UNIQUE TAG TagName4]
[ADD FOREIGN KEY [eExpression5] TAG TagName4 [FOR lExpression6]
REFERENCES TableName2 [TAG TagName5]]
[DROP FOREIGN KEY TAG TagName6 [SAVE]]
[RENAME COLUMN FieldName4 TO FieldName5]
[NOVALIDATE]

参数描述

TableName1
指定要修改其结构的表名。

ADD [COLUMN] FieldName1
指定要添加的字段名。一个表最多可含 255 个字段。如果一或多个字段允许空值，则最多可含 254 个字段。

ALTER [COLUMN] FieldName1
指定要修改的字段名（字段已存在）。

FieldType [(nFieldWidth [, nPrecision])]
指定新字段或待修改字段的字段类型、字段宽度和字段精度（小数点后的位数）。

参数 *FieldType* 是表示字段数据类型的单个字符。有些字段类型还需要参数 *nFiledWidth* 或 *nPrecision*，或者两者皆要。

下表列出了参数 *FiledType* 的值及其对应参数 *nFiledWidth* 和 *nPression* 的取舍情况。

字段类型	字段宽度	精度	说明
C	n	–	宽度为 *n* 的字符字段
D	–	–	日期
T	–	–	日期时间
N	n	d	宽度为 *n* 的数值型字段，小数点后保留 *d* 位
F	n	d	宽度为 *n* 的浮点型字段，小数点后保留 *d* 位
I	–	–	整型
B	–	d	双精度型
Y	–	–	货币型
L	–	–	逻辑型
M	–	–	备注型
G	–	–	通用型
P	–	–	图片型

对于 D、T、I、Y、L、M、G 和 P 型数据，省略参数 *nFieldWidth* 和 *nPrecision*。如果

对 N、F 或 B 型数据没有给出参数 *nPrecision* 的值，其默认值为 0。

NULL | NOT NULL

允许（或不允许）字段为 null 值。如果一个或更多的字段可以包含.null.值，那么表拥有的最多的字段数将减少一个，从 255 降到 254。

如果省略 NULL 和 NOT NULL，当前的 SET NULL 设置将决定字段是否允许为 null；并且如果命令中带有 PRIMARY KEY 或 UNIQUE 子句，当前的 SET NULL 设置不起作用，字段默认为非 null。

CHECK lExpression1

指定字段的有效性规则。*lExpression1* 为逻辑表达式值；可以是用户定义的函数或内部存储过程。注意：每添加一个空记录时，都将进行有效性检查。如果有效性规则不允许添加的记录中有空字段值，Visual FoxPro 将产生错误信息。

ERROR cMessageText1

指定字段有效性检查出现错误时显示的错误信息。只有在浏览或编辑窗口中修改数据时，此信息才可能显示。

DEFAULT eExpression1

指定字段默认值。*eExpression1* 的数据类型必须与字段的数据类型相同。

PRIMARY KEY

创建主索引标识。索引标识与字段同名。

UNIQUE

创建与字段同名的候选索引标识。

有关候选索引标识的更多信息，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第七章“表的操作”。

注意 候选索引（由 UNIQUE 选项创建，具有在 ALTER TABLE 或 CREATE TABLE 中的 ANSI 兼容性。）与带 UNIQUE 选项的 INDEX 命令创建的索引不同。用带 UNIQUE 选项的 INDEX 命令创建的索引允许有重复索引关键字，而候选索引不允许有重复索引关键字。

在主索引或候选索引字段中，不允许有 null 值和重复记录。

如果用 ADD COLUMN 子句创建一个新字段，若给允许 null 值的字段创建主索引或候选索引，Microsoft Visual FoxPro 不会产生错误。但是，如果您试图往用作主索引或候选索引的字段中输入 null 值或重复值，Visual FoxPro 将产生错误信息。

如果修改已有的字段和由字段组成的主索引表达式或候选表达式时，将检查字段是否为 null 值或重复记录。如果为 null 值或重复记录，Visual FoxPro 将产生错误，并且不对表进行修改。

REFERENCES TableName2 TAG TagName1

指定与之建立永久关系的父表。参数 TAG *TagName1* 指定父表索引标识，关系建立在此父表索引标识基础上。索引标识最长为 10 个字。

NOCPTRANS

防止对字符串或备注字段进行代码页转换。如果需要将表转换到另一代码页，那么指定了 NOCPTRANS 的字段不进行转换。只能对字符字段和备注字段指定 NOCPTRANS。

下面的示例创建一个包含两个字符型字段和两个备注型字段，名称为 MYTABLE 的

表。第 2 个字符型字段 char2 与第 2 个备注型字段 memo2 包括了 NOCPTRANS，以避免翻译。

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

ALTER [COLUMN] FieldName2

指定要修改的已有的字段名。注意在单独的 ALTER TABLE 命令中更改字段一个以上的属性时，需要多个 ALTER COLUMN 子句。请参阅 ALTER TABLE 示例，了解 ALTER COLUMN 子句是如何构造的。

SET DEFAULT eExpression2

指定已有字段的新默认值。*eExpression2* 的数据类型必须与字段数据类型相同。

SET CHECK lExpression2

为已有字段指定新的有效性规则。*lExpression2* 值必须为逻辑表达式，也可以为用户自定义函数或已有的过程。

ERROR cMessageText2

指定有效性检查出现错误时显示的错误信息。只有在“浏览”窗口或“编辑”窗口改变数据时，才可能显示此信息。

DROP DEFAULT

删除已有字段的默认值。

DROP CHECK

删除已有字段的有效性规则。

DROP [COLUMN] FieldName3

从表中删除一个字段。删除一个字段的同时也删除了字段的默认值和字段有效性规则。

字段被删除后，索引关键字或引用此字段的触发器表达式将变为无效。在这种情况下，删除字段并不产生错误，但是在运行时刻，无效的索引关键字或触发器表达式将导致错误。

SET CHECK lExpression3

指定表的有效性规则。*lExpression3* 必须是逻辑表达式，也可以是用户自定义函数或已有的过程。

ERROR cMessageText3

指定表的有效性检查出现错误时显示的错误信息。只有在浏览窗口或编辑窗口中改变数据值时，才可能显示此信息。

DROP CHECK

删除表的有效性规则。

ADD PRIMARY KEY eExpression3 TAG TagName2 [FOR lExpression4]

往表中添加主索引，*eExpression* 指定主索引关键字表达式，*TagName2* 指定主索引标识名，索引标识名最长为 10 个字符。如果省略 TAG *TagName2* 而 *eExpression3* 是一个字段，主关键索引标识与指定的 *eExpression3* 同名。

包含 FOR lExpression4 子句，可以指定只有满足筛选表达式 lExpression4 的记录才可以显示和访问；主索引关键字是在所有文件中只为符合这个筛选表达式的记录创建的。注意，应该避免使用 FOR 子句创建一个主索引；主索引关键字的唯一性只限制

满足筛选表达式 lExpression4 的记录。相反，应该使用带 FOR 子句的 INDEX 命令创建一个筛选索引。

如果 lExpression4 是一个可优化的表达式，则 Rushmore 技术优化这个 ALTER TABLE ... FOR lExpression4 命令。为了得到最好的性能，可在 FOR 子句中使用一个可优化的表达式。

详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”。

DROP PRIMARY KEY

删除主索引及其标识。因为表只能有一个主关键字，所以不必指定关键字的名称。删除主索引也将删除所有基于此关键字的永久关系。

ADD UNIQUE eExpression4 [TAG TagName3 [FOR lExpression5]]

往表中添加候选索引。*eExpression4* 指定候选索引关键字表达式，*TagName3* 指定候选索引标识名。候选标识名最长可为 10 个字符。如果省略参数 TAG *TagName3* 并且 *eExpression4* 为单个字段，候选索引标识与 *eExpression4* 中的指定的字段同名。

包含 FOR lExpression5 子句，可以指定只有满足筛选表达式 lExpression5 的记录才可以显示和访问；候选索引关键字是在所有文件中只为符合这个筛选表达式的记录创建的。

如果 lExpression5 是一个可优化的表达式，则 Rushmore 技术优化这个 ALTER TABLE ... FOR lExpression5 命令。为了得到最好的性能，可在 FOR 子句中使用一个可优化的表达式。

详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应

用程序”。

DROP UNIQUE TAG TagName4

删除候选索引及其标识。因为表可能有多个候选关键字，所以必须指定候选索引标识名。

ADD FOREIGN KEY [eExpression5] TAG TagName4 [FOR lExpression6]

往表中添加外部关键字（非主关键字）索引。*eExpression5* 指定外部索引关键字表达式，*TagName4* 指定外部索引标识名。索引标识名最长为 10 个字符。

包含 FOR lExpression6 子句，可以指定只有满足筛选表达式 lExpression6 的记录才可以显示和访问；外部索引关键字是在所有文件中只为符合这个筛选表达式的记录创建的。

如果 lExpression6 是一个可优化的表达式，则 Rushmore 技术优化这个 ALTER TABLE…FOR lExpression6 命令。为了得到最好的性能，可在 FOR 子句中使用一个可优化的表达式。

详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”。

REFERENCES TableName2 [TAG TagName5]

指定在其上创建了永久关系的父表。使用 TAG *TagName5* 指定一个已有的索引标识，基于此索引标识建立表与父表的一个关系。索引标识名最长可以为 10 个字符。如果省略参数 TAG *TagName5*，则使用父表的主索引标识建立关系。

DROP FOREIGN KEY TAG TagName6 [SAVE]

删除索引标识为 *TagName6* 的外部关键字。如果省略 SAVE 参数，将从结构索引中删除索引标识。如果加入 SAVE 参数，则不从结构索引中删除索引标识。

RENAME COLUMN FieldName4 TO FieldName5

允许改变表中字段的字段名。*FieldName4* 指定待更改的字段名，*FieldName5* 指定新的字段名。

警告　改变表的字段名时一定要小心：索引表达式、字段和表的有效性规则、命令、函数等等可能仍会引用原始字段名。

NOVALIDATE

选用这一选项后，Visual FoxPro 修改表的结构不受表中数据完整性的约束。默认时，Visual FoxPro 改变表结构将受到表中数据的完整性约束。使用 NOVALIDATE 参数将使默认情况无效。

说明

ALTER TABLE 命令可以用于修改还没有添加到数据库中的表的结构。然而在修改自由表时，如果加入 DEFAULT、FOREIGN KEY、PRIMARY REFERENCES 或 SET 子句，Visual FoxPro 将出现错误。

ALTER TABLE 命令可以通过建立新表头和往表头中添加记录来重建表。例如，改变字段类型或字段宽度。

表经过重建后，将对所有改变了类型或宽度的字段执行字段有效性规则。如果修改了表中所有字段的类型或宽度，将执行表的有效性规则。

如果对已含有记录的表修改其字段有效规则或表有效性规则，Visual FoxPro 将检查新的字段或表有效性规则是否与存在数据相符合，并在发现有不符之处时发出警告。如果要修改的表在数据库中，ALTER TABLE-SQL 命令需要独占使用数据库。若要独占打开数据库，可使用包含 EXCLUSIVE 子句的 OPEN DATABASE 命令。

示例

示例 1 往表 customer 中添加字段 fax，并且允许字段有 null 值。
示例 2 使 cust_id 字段为 customer 表的主关键字。
示例 3 给 orders 表中的 quantity 字段添加有效性规则，使字段 quantity 的值非负。
示例 4 基于表 customer 的关键字 cust_id 和表 orders 中的候选关键字 cust_id，建立 customer 和 orders 间的一对多永久关系。
示例 5 删除表 orders 的 quantity 字段的有效性规则。
示例 6 删除表 customer 和表 orders 间的永久关系，但保持 orders 表中的 cust_id 索引标识。
示例 7 添加名为 fax2 的字段到 customer 中以防止字段包含 null 值。将显示表的新结构。两个 ALTER COLUMN 子句用于允许字段有 null 值，并且将字段的默认值设置为 null 值。注意在单独的 ALTER TABLE 命令中更改字段一个以上的属性时，需要多个 ALTER COLUMN 子句。新字段将被从表中删除，表恢复到原来的状态。

```
* 示例 1
SET PATH TO (HOME(2) + 'Data\')    && 为表设置路径
ALTER TABLE customer ADD COLUMN fax c(20) NULL

* 示例 2
ALTER TABLE customer ADD PRIMARY KEY cust_id TAG cust_id
```

```
ALTER TABLE customer ALTER COLUMN cust_id c(5) PRIMARY KEY
```

*** 示 例** 3

```
ALTER TABLE orders;
    ALTER COLUMN quantity SET CHECK quantity >= 0;
    ERROR "Quantities must be non-negative"
```

*** 示 例** 4

```
ALTER TABLE orders;
    ADD FOREIGN KEY cust_id TAG cust_id REFERENCES customer
```

*** 示 例** 5

```
ALTER TABLE orders ALTER COLUMN quantity DROP CHECK
```

*** 示 例** 6

```
ALTER TABLE orders DROP FOREIGN KEY TAG cust_id SAVE
```

*** 示 例** 7

```
CLEAR
ALTER TABLE customer ADD COLUMN fax2 c(20) NOT NULL
DISPLAY STRUCTURE

ALTER TABLE customer;
    ALTER COLUMN fax2 NULL;
    ALTER COLUMN fax2 SET DEFAULT .NULL.

ALTER TABLE customer DROP COLUMN fax2
```

请 参 阅

CREATE TABLE-SQL, INDEX, MODIFY STRUCTURE, OPEN DATABASE

AlwaysOnTop 属性

避免其他窗口覆盖表单窗口。设计时可用，运行时可读写。

语法

Object.AlwaysOnTop[= lExpr]

参数描述

lExpr

AlwaysOnTop 属性设置为：

设置	说明
“真” (.T.)	表单总是显示在最顶层（只有 AlwaysOnTop 属性设为 .T. 的窗口可显示在表单之上）。
“假” (.F.)	默认值。表单可被其他窗口覆盖。

说明

注意 AlwaysOnTop 属性只能应用于 MDI (多文档界面)表单，此表单是 ShowWindow 属性设置为 1—在顶层表单中时创建的。

应用于

表格, _SCREEN

请参阅

ShowWindow **属性**, ZOrder **方法**

AMEMBERS（）函数

将一个对象的属性名、过程名和成员对象存入变量数组。

语法

AMEMBERS(ArrayName, ObjectName | cClassName [, 1 | 2])

返值类型

数值型

参数描述

ArrayName

指定存放对象的成员属性名的数组。如果指定的数组不存在， Visual FoxPro **将自动创建该数组。如果数组大小不足以包含所有名称，** Visual FoxPro **将自动增加数组大小。**

ObjectName

指定对象名，此对象的成员属性将存入指定的变量数组 *ArrayName* **中。**

ObjectName 可以是任意的对象表达式，如对象引用、对象变量或对象数组中的一个元素。

cClassName

指定类名。它的成员属性放在 ArrayName 指定的变量数组中。

1

指定数组中既包含对象的方法和成员信息，还包含其对象的属性信息。生成的数组是二维数组，第一列放置成员信息，第二列放置成员的类型信息。第二列的可选值有 Property、Event、Method 和 Object。

2

指定数组中包含 ObjectName 对象的成员对象名。结果数组是一维的。

AMEMBERS() 函数返回对象的成员对象、属性和过程的总数。若不能创建数组则返回零。如果省略参数 1 和 2，将创建包含对象 *ObjectName* 属性的一维数组。

示例

下面的示例用 CREATE OBJECT() 创建了一个名称为 goForm1 的表单对象。用 AMEMBERS()创建一个包含此表单可用的属性、名称为 gaPropArray 的数组；然后显示属性。

```
CLEAR
goForm1 = CREATEOBJECT("Form") && 创建表单
= AMEMBERS(gaPropArray, goForm1, 1) && 包含表单属性的数组
DISPLAY MEMORY LIKE gaPropArray && 显示表单属性
```

请参阅

ADD CLASS, AINSTANCE(), CREATE CLASS, CREATE CLASSLIB, CREATEOBJECT( ) , DEFINE CLASS

ANETRESOURCES（）函数

将网络共享或打印机的名称放到一个数组中，然后返回资源的数目。

语法

ANETRESOURCES(*ArrayName, cNetworkName, nResourceType*)

返值类型

数值型

参数描述

ArrayName

指定数组的名称，该数组中包含网络共享或打印机的信息。如果所指定的数组不存在，Visual FoxPro 会自动创建该数组。如果该数组存在，但是大小不足以容纳所有的信息，则 Visual FoxPro 自动增加该数组的大小。如果该数组比所需的要大，则 Visual FoxPro 会截短该数组。

如果数组存在，并且由于没有找到或打印机使得 ANETRESOURCES（）返回 0，则该数组保持不变。如果该数组不存在，并且 ANETRESOURCES（）返回 0，则不创建该数组。

cNetworkName

指定网络的名称，在该网络上返回了共享或打印机信息。网络名的格式应该是“\\NetworkName”。不必与所指定的网络连接，而且指定一个网络也不会使您与该网络连接。

nResourceType

指定网络资源的类型，将返回该资源的信息。如果 *nResourceType* 的值为 1，则返回网络上共享的名称。如果 *nResourceType* 的值为 2，则返回网络上打印机的名称。

说明

ANETRESOURCES（）返回所找到的网络共享或打印机的数量（等于数组的行数）。如果所指定网络上没有共享或打印机，或者所指定的网络不存在，则 ANETRESOURCES（）返回零。

请参阅

ADIR( )

AMOUSEOBJ( ) 函数

创建一个数组，其中包含有关鼠标指针位置以及鼠标指针下对象的信息。

语法

AMOUSEOBJ(ArrayName [, 1])

返值类型

数值型

参数描述

ArrayName

指定数组的名称，该数组中包含有关鼠标指针的信息。如果所指定的数组不存在，Visual FoxPro会自动创建该数组。如果该数组存在，但是大小不足以容纳所有的信息，则Visual FoxPro自动增加该数组的大小。如果该数组比所需的要大，则Visual FoxPro会截短该数组。

所创建的数组包含四行。下表说明了数组每行的内容：

数组行	说明
1	包含一个对象引用，当执行 AMOUSEOBJ（）函数时，鼠标指针位于该对象上。
2	包含一个对象容器的对象引用，当执行 AMOUSEOBJ（）函数时，鼠标指针位于该对象上。
3	包含鼠标指针相对于对象容器的水平坐标 (X)，单位为像素，当执行 AMOUSEOBJ（）函数时，鼠标指针位于该对象上。
4	包含鼠标指针相对于对象容器的垂直坐标 (Y)，单位为像素，当执行 AMOUSEOBJ（）函数时，鼠标指针位于该对象上。

注意，如果鼠标位于一个普通容器上，例如页框上，数组的第一行和第二行可以包含相同的值。

[, 1]

这个可选的参数指定数组中包含的鼠标指针信息是相对于当前表单的 (THISFORM)。如果包含了这个选项，数组的第二行总是包含对当前表单的一个对象引用，而第三第四行包含鼠标指针相对于当前表单的坐标。

说明

MOUSEOBJ（）也可以用来在设计时刻确定鼠标指针的位置。下表列出了数组在设计时刻的元素，以及数组每行包含的值：

设计时刻的元素	数组内容
表单和类设计器	第一行——对控件的对象引用。
	第二行——对表单的对象引用。
	第三行——相对于表单的鼠标指针水平坐标 (X)。
	第四行——相对于表单的鼠标指针垂直坐标 (Y)。
项目管理器	第一行——对项目的对象引用。
	第二行——对象引用 to 项目.
	第三行——零。
	第四行——零。
Visual FoxPro 桌面	第一行——对桌面的对象引用。
	第二行——对桌面的对象引用。
	第三行——相对于桌面的鼠标指针水平坐标 (X)。
	第四行——相对于桌面的鼠标指针垂直坐标 (X)。

如果鼠标指针位于上述区域，则 AMOUSEOBJ（）返回 4（数组的行数）。如果鼠标指针位于其他区域，AMOUSEOBJ（）返回零，并且如果所指定的数组存在，则该数组不变。如果所指定的数组不存在，则不创建它。

请参阅

M , MROW( ) , SYS(1270)——对象位置

ANSITOOEM（）函数

包含此项是为了提供向后兼容性。请使用 GETCP()。

APPEND 命令

在表的末尾添加一个或多个新记录。

语法

APPEND [BLANK]
[IN nWorkArea | cTableAlias]
[NOMENU]

参数描述

BLANK

在当前表的末尾添加一个空记录。

Visual FoxPro 在发出 APPEND BLANK 命令时并不打开编辑窗口。可以使用

BROWSE、CHANGE 或 EDIT 命令编辑新记录。

IN nWorkArea

指定要添加新记录的表所在的工作区。

IN cTableAlias

指定要添加新记录的表的别名。

如果省略 *nWorkArea* 和 *cTableAlias*，新记录将添加到当前选定工作区的表中。如果发出 APPEND 命令，空记录将添加到由 *nWorkArea* 或 *cTableAlias* 指定的工作区的表中，并且自动选定该表；如果发出 APPEND BLANK 命令，空记录将添加到指定的 *nWorkarea* 或 *cTableAlias* 工作区的表中，但不选定表。

NOMENU

此参数指定将“表”菜单标题从系统菜单栏中删除，以避免改变编辑窗口的格式。

说明

当发出 APPEND 或 APPEND BLANK 命令，并且没有在当前选定工作区中打开表时，将显示一个打开对话框，您可以在对话框中选择需要添加记录的表。

APPEND 命令打开一个编辑窗口，您可以在其中输入一个或多个新记录。增加新记录后，Visual FoxPro 将自动修改打开的所有索引。

示例

下面的示例用 APPNED BLANK 创建了一个包含了随机值、有 10 个记录的表，然后显示了表中的最大值与最小值。

```
CLOSE DATABASES
CREATE TABLE Random (cValue N(3))
FOR nItem = 1 TO 10  && 增加 10 个记录
    APPEND BLANK
    REPLACE cValue WITH 1 + 100 * RAND（）  && 插入随机值
ENDFOR

CLEAR
LIST  && 显示这些值
gnMaximum = 1  && 初始化最小值
gnMinimum = 100  && 初始化最大值
SCAN
    gnMinimum = MIN(gnMinimum, cValue)
    gnMaximum = MAX(gnMaximum, cValue)
ENDSCAN
? ‘The minimum value is: ‘, gnMinimum  && 显示最小值
? ‘The maximum value is: ‘, gnMaximum  && 显示最大值
```

请参阅

APPEND FROM ARRAY, BROWSE, CHANGE, EDIT, INSERT-SQL, REPLACE

APPEND FROM 命令

从一个文件中读入记录，追加到当前表的尾部。

语法

```
APPEND FROM FileName | ?
  [FIELDS FieldList]
  [FOR lExpression]
  [[TYPE] [DELIMITED [WITH Delimiter | WITH BLANK | WITH TAB
  | WITH character Delimiter]
  | DIF | FW2 | MOD | PDOX | RPD | SDF | SYLK
      | WK1 | WK3 | WKS | WR1 | WRK | CVS
      | XLS | XL5 [SHEET cSheetName] | XL8 [SHEET cSheetName]]]
  [AS nCodePage]
```

参数描述

FileName

指定从哪个文件中读入记录。如果给出的文件名不包含扩展名，则将文件默认为 Visual FoxPro 表，扩展名为 .DBF。如果文件是 Visual FoxPro 表，无论 SET DELETED 为何种设置，表中标记为删除的记录也将添加到当前表中。

?

显示打开对话框，从中可以选择从哪个表中读入记录。

FIELDS FieldList

指定添加哪些字段数据。

FOR lExpression

为当前选定表中每一条 *lExpression* 为“真”(.T.) 的记录追加新记录，直至达到当前选定表的末尾。如果省略 FOR，则整个源文件记录都追加到当前表中。

TYPE

指定源文件类型。如果指定的源文件类型不是 Visual FoxPro 表，则必须指定文件类型，但不必包括 TYPE 关键字。您可以从各种类型文件（包括分隔 ASCII 文本文件）中读入信息添加到表中，在这些文件中可以指定字段分隔符。

如果要追加的源文件扩展名不是默认的扩展名，源文件名必须包括文件扩展名。例如，Microsoft Excel 工作表通常具有 .XLS 扩展名。如果要追加的 Microsoft Excel 工作表扩展名不是 .XLS，一定要指定扩展名。

注意 如果要追加的记录来自工作表，工作表中的数据必须以主行序而非主列序存储，这样才能使追加的工作表数据符合表结构。

DELIMITED

指定源文件为分隔数据文件。分隔数据文件是 ASCII 文本文件，文件中每条记录以回车和换行符结尾。各字段内容默认地由逗号分开，字符字段值还需要用引号括上。例如：

“Smith”,9999999,”TELEPHONE”

所有分隔数据文件的扩展名默认为 .TXT 数据的格式。
如果日期格式正确，可以从分隔文件中导入日期数据，日期的默认格式为 mm/dd/yy。您还可以选择加入世纪信息。Visual FoxPro 导入的数据（如 12/25/95）不包含世纪信息，世纪信息的默认值为 20 世纪。日期分隔符可以为任意非数值字符，但不能使用分隔文件中字段的分隔符。
如果其他一些日期格式与 SET DATE 中可以使用的格式相匹配，Visual FoxPro 也可以导入这些格式的日期数据。若要导入非默认格式的日期，应在使用 APPEND FROM 前先发出 SET DATE 修改数据格式设置。要想检查日期格式是否能成功地导入，可使用 CTOD（）函数。如果 CTOD（）函数接收此日期值，则日期数据就能正确地导入。

DELIMITED WITH Delimiter
字符字段由 ***Delimiter*** 标识，而非引号。

DELIMITED WITH BLANK
由空格符 (BLANK) 分隔字段，而不是用逗号分隔字段。

DELIMITED WITH TAB
各字段由制表符 (TAB) 来分隔，而非逗号。

DELIMITED WITH Character Delimiter
字段之间由给定的 *Delimiter* 分隔。如果 *Delimite* 是分号，应用引号括起来，因为引号在 Visual FoxPro 中有特殊的意义：一个命令分在多行中书写时，用分号作为行的结束。*Delimiter* 可以是 BLANK 或 TAB。

WITH *Delimiter* 子句可与 WITH CHARACTER 子句同时使用。例如，在下面的例子中，添加记录的来源是一个文本文件。该文本文件中，字符字段用下划线 _ 标识，而字段之间用星号 * 分割。

```
APPEND FROM mytxt.txt DELIMITED WITH _ ;
    WITH CHARACTER *
```

DIF

选用 DIF 可从 VisiCalc .dif（数据交换格式）文件中导入数据。矢量（列）对应当前选定表的字段，元组（行）对应表的记录。DIF 文件的默认扩展名为 .DIF。

FW2

选用 FW2 可从由 Framework II 创建的文件中导入数据。FW2 文件的默认扩展名为 .FW2。

MOD

选用 MOD 可从 Microsoft Multiplan 4.01 版本的文件中导入数据。MOD 文件由 Microsoft Multiplan 4.01 版本创建，默认扩展名为 .MOD。

PDOX

选用 PDOX 可从 Paradox 3.5 版或 4.0 版数据库文件中导入数据。Paradox 文件名的默认扩展名为 .DB。

RPD

选用 RPD 可从由 RapidFile 1.2 版本创建的文件中导入数据。RapidFile 文件名的默认扩展名为 .RPD。

SDF

选用 SDF 可从系统数据格式文件中导入数据。SDF 文件是一种 ASCII 文本文件，记录有固定长度，并且以回车和换行符结尾，各字段不分隔开。文件的默认扩展名为 .TXT。

SYLK

选用 SYLK 可从 SYLK（符号链接）交换格式文件中导入数据。SYLK 文件用于 Microsoft MultiPlan 中。SYLK 文件中的列对应 Visual FoxPro 表的字段，行对应表的记录。SYLK 文件没有扩展名。

WK1

选用 WK1 可从 Lotus 1-2-3 2.x 版本的电子表格中导入数据。电子表格的每列为表的一个字段，每行为表的一条记录。Lotus 1-2-3 2.x 版本创建的电子表格扩展名为 WK1。

WK3

选用 WK3 可从 Lotus1-2-3 的电子表格中导入数据，电子表格的每列为表的一个字段，每行为表的一条记录。Lotus1-2-3 版本 3.X 创建的电子表格扩展名为 .WK3。

WKS

选用 WKS 可从 Lotus1-2-3 1-A 版的电子表格中导入数据。电子表格的每列为表的一个字段，每行为表的一条记录。Lotus1-2-3 1-A 版本创建的文件扩展名为 .WKS。

WR1

选用 WR1 可从 Lotus Symphony 1.1 或 1.2 版的电子表格中导入数据。电子表格的每列为表的一个字段，每行为表的一条记录。Symphony1.1 或 1.2 版创建的电子表格扩展名为 .WR1。

WRK

选用 WRK 可从 Lotus Symphony 1.0 版的电子表格中导入数据。电子表格中的每列为表的一个字段，每行为表的一条记录。Symphony 1.0 版创建的电子表格扩展名为 .WRK。

CVS

选用 CVS 可从一个各值用逗号分隔的文件中导入数据。一个 CSV 文件的第一行是字段名；当导入该文件时，会忽略这个字段名。

XLS

选用 XLS 可从 Microsoft Excel 工作表中导入数据。工作表的每列为表的一个字段，每行为表的一条记录。由 Microsoft Excel 创建的工作表扩展名为 .XLS。

XL5

选用 XL5 可从 Microsoft Excel 5.0 版中导入数据。工作表的每列为表的一个字段，每行为表的一条记录。工作表文件的扩展名为 .XLS。

如果省略 SHEET 子句，会导入 Sheet1 中的数据。为了导入特定工作表中的数据，需要包含 SHEET 关键字，并且使用 cSheetName 指定工作表的名称。

XL8

包含 XL8 快导入 Microsoft Excel 97 的数据。工作表的列变成表中的字段；工作表的行变成表中的记录。在 Microsoft Excel 中创建的工作表文件的扩展名是 .xls。

如果省略 SHEET 子句，会导入 Sheet1 中的数据。为了导入特定工作表中的数据，需要包含 SHEET 关键字，并且使用 cSheetName 指定工作表的名称。

AS nCodePage

指定源表或源文件的代码页。Visual FoxPro 将复制源表或源文件的内容，并在复制时自动把数据转换到当前表的代码页中。

如果指定的 *nCodePage* 值无法使用，Visual FoxPro 将产生一条错误信息。您可以用 GETCP（）函数显示代码页对话框，在对话框中可以为追加的表或文件指定代码页。

如果省略 AS *nCodePage* 子句，并且 Visual FoxPro 不能判定源表或文件的代码页，Visual FoxPro 将复制源表或文件内容，并在复制数据的过程中，自动将数据转换到当前的 Visual FoxPro 代码页中。如果 SET CPDIALOG 为 ON，当前选定工作区中的表以代码页标记。如果要从没有代码页标记的表中读入数据并添加到表中时，将显示代码页对话框，您可以在其中选择表的代码页。当前的 Visual FoxPro 代码页可以由 CPCURRENT（）函数设定。

如果省略 AS *nCodePage* 子句，并且 Visual FoxPro 可以确定追加记录的表或文件的代码页，Visual FoxPro 将复制表或文件的内容，并在复制数据的过程中，自动将数据转换到当前选定表的代码页中。

如果 *nCodePage* 为零，Visual FoxPro 认为需追加记录的表和文件的代码页与当前选定表的代码页相同，并且不进行代码页的转换。

说明

如果从其中追加数据的文件是 Visual FoxPro 表或在 FoxPro 早期版本中创建的表，其扩展名为 .DBF。如果其扩展名不是 .DBF，您必须指定扩展名。如果文件不是 Visual FoxPro 表或 FoxPro 早期版本创建的表，还必须指定文件的类型。

在从 DBASE IV 或 DBASE V 创建的包含备注字段的表中追加记录前，您必须先用 USE 命令在 Visual FoxPro 中打开此表，当提示信息询问您是否要转换文件时，请选择“是”。

如果从 FoxPro 早期版本的表或 Visual FoxPro 表中读入记录，此表可以在另一工作区打开。对于源表中有删除标记的记录，一旦添加到表中，将会去掉删除标记。

使用 DBF() 函数可以从一个只读的临时表追加数据，该临时表是使用 SELECT-SQL 命令创建的。可象下例这样在 DBF() 函数中包含临时表的名称：

```
APPEND FROM DBF('<Cursor Name>')
```

示例

在下面的示例中，表 customer 被打开，复制它的结构到表 backup 中，并且表 backup 也被打开。然后 VisualFoxPro 从表 customer 的 Finland 追加所有记录。这些记录被复制到一个新的分隔文件 temp.txt 中。

```
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 = 'Finland'
COPY TO temp TYPE DELIMITED
MODIFY FILE temp.txt
USE
DELETE FILE backup.dbf
DELETE FILE temp.txt
```

请参阅

COPY FILE, COPY TO, EXPORT, GETCP(), IMPORT

APPEND FROM ARRAY 命令

对应数组中的每一行，追加一条记录到当前选定表中，并从相应的数组行中取出数据添加到记录中。

语法

```
APPEND FROM ARRAY ArrayName
  [FOR lExpression]
  [FIELDS FieldList
  | FIELDS LIKE Skeleton
  | FIELDS EXCEPT Skeleton]
```

参数描述

ArrayName

指定数组名，该数组包含要复制到新记录中的数据。命令将把数组中所有的行都追加到表中。

FOR lExpression

为数组中用于追加的记录指定条件。在 *lExpression* 条件表达式中必须包含目标字段名。

在数组中的行追加到表中的记录之前，先检查与 *lExpression* 中指定的目标字段对应的数组元素是否满足 *lExpression* 中的条件。如果数组元素满足条件，则将记录追加表中。

如果数组元素不满足条件，这一数组行不追加到表中，并继续检查下一行。

FIELDS FieldList

指定只有 *FieldList* 中的字段才从数组进行更新。*FieldList* 中的第一个字段用数组第一个元素的内容更新，第二个字段用第二个元素更新，依此类推。

FIELDS LIKE Skeleton

指定从数组中更新与字段梗概 Skeleton 匹配的字段。

FIELDS EXCEPT Skeleton

指定从数组中更新所有与字段梗概 Skeleton 不匹配的字段。

字段梗概 Skeleton 支持通配符。例如，为了指定从数组更新所有以字母 A 与 P 开头的字段，可以用：

```
APPEND FROM ARRAY aMyArray FIELDS LIKE A*,P*
```

LIKE 子句可以同 EXCEPT 子句组合使用。

```
APPEND FROM ARRAY aMyArray FIELDS LIKE A*,P* EXCEPT PARTNO*
```

说明

在 APPEND FROM ARRAY 命令中，备注字段和通用字段将被忽略。如果表处于打开状态并被共享使用，在追加记录时，APPEND FROM ARRAY 命令将锁定表头。

如果数组是一维的，APPEND FROM ARRAY 命令只在表中添加一个记录。第一个数组元素的内容将填充到新添加记录的第一个字段，第二个元素的内容将填充到记录的第二字段，… 依此类推。

如果一维数组元素的个数多于表字段数，将忽略多余的元素。如果表字段数多于数组元素的个数，多出的字段将初始化为默认的空值。下面给出了各种字段类型对应的默认空值。

字段类型	默认值
字符型	空格
数值型	0
货币型	0
浮点型	0
整型	0

续表

双精度型	0
日期型	空日期（例如 CTOD(''））
日期时间型	空日期时间（例如 CTOT(''））
逻辑型	假 (.F.)
备注型	空（无内容）

当指定数组为二维数组，则为数组中的每一行在表中添加一个新记录。例如，如果数组有 4 行，则在表中追加 4 个新记录。

数组中第一列的内容赋值给新添加记录的第一个字段，第二列内容赋值给新记录的第二个字段，依此类推。例如，数组有四行三列，数组中的第一列元素分别赋值给四个新记录的第一个字段。

如果二维数组的列数多于表中的字段数，多余的列将被忽略。如果表字段数多于数组列数，多出的字段将初始化为空值 (empty value)。

例如数组元素数据与相应的字段数据类型兼容，那么即使相应的数组元素的数据类型与字段数据类型不匹配，APPEND FROM ARRAY 也能填充字段。如果数据不兼容，字段将被初始化为空值。

示例

如下示例生成一个新表，并用 APPEND FROM ARRAY **命令给新表添加一条记录。**

```
LOCAL ARRAY aNewRec(3)

* Create the table
CREATE TABLE Test FREE  (OBJECT C(10), Color C(16), SqFt n(6,2))
SCATTER TO aNewRec BLANK  && 从表中创建一个新数组
aNewRec[1]="Box"       && 填充数组
aNewRec[2]="Red"
aNewRec[3]=12.5

APPEND  FROM  ARRAY aNewRec  && 增加包含数组内容的记录
             && 到表中
```

请参阅

APPEND, COPY TO ARRAY, DIMENSION, GATHER, SCATTER

APPEND GENERAL 命令

从文件中导入 OLE **对象，并将其放入通用字段中。**

语法

APPEND GENERAL GeneralFieldName
[FROM FileName]
[DATA cExpression]
[LINK]
[CLASS OLEClassName]

参数描述

GeneralFieldName

指定放置 OLE **对象的通用字段名。可以用带有表别名的字段名来指定在非当前工作区中打开的表的通用字段。**

FROM FileName

指定包含 OLE **对象的文件。必须给出文件全名，包括扩展名。如果文件不在当前目录或当前文件夹中，还需要给出文件的路径。**

DATA cExpression

指定字符表达式，此表达式作为一个字符串存入 OLE **对象的通用字段中。**OLE **对象必须能接收和处理字符串。例如，不能往** Paintbrush™ **的图片对象中存入字符串。**

LINK

建立 OLE **对象和包含对象的文件间的链接。**OLE **对象出现在通用字段，但对象定义仍在文件中。如果省略** LINK，OLE **对象将嵌入到通用字段中。**

CLASS OLEClassName
为 OLE 对象指定具体的 OLE 类，而不用其默认类。

提示 您可以通过运行 REGEDIT 并双击某一 OLE 对象来确定该对象的类名，类名列在“标识符”后。

当包含 OLE 对象的文件的扩展名不同于默认扩展名，并且要强制类行为时，您可以指定类名。如果默认扩展名可用于多个 OLE 服务程序，可用该类指定具体的服务程序。

说明

如果在通用字段中已有一个 OLE 对象，它将被源文件中的 OLE 对象取代。若要从通用字段中删除一个 OLE 对象，可不带任何附加参数地使用 APPEND GENERAL GeneralFieldName 命令（GeneralFieldName 是要清理的通用字段的名称）。

其它相关信息，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十六章“添加 OLE”。

示例

以下示例从 Excel 目录或文件夹中导入 Microsoft Excel 图表，存入通用字段 mygenfield 中。

```
CREATE TABLE MyGenTbl (mygenfield G)
APPEND BLANK  && 添加一个空记录
APPEND GENERAL mygenfield FROM C:\EXCEL\BOOK1.XLS CLASS EXCELCHART
```

请参阅

APPEND MEMO 命令

将文本文件的内容复制到备注字段中。

语法

APPEND MEMO MemoFieldName FROM FileName
[OVERWRITE] [AS nCodePage]

参数描述

MemoFieldName
指定备注字段名，文件内容将追加到此备注字段中。

FROM *FileName*
指定文本文件，其内容将复制到备注字段中。此参数必须包含完整的文本文件名，包括扩展名。

OVERWRITE
用文件的内容替换备注字段当前的内容。

AS nCodePage
指定复制到备注字段中的文本文件的代码页。Visual FoxPro 复制文本文件的内容，并在将数据复制到备注字段的过程中，自动将数据从指定代码页转换成备注字段所在表

的代码页。如果包含备注字段的表没有使用代码页标记，Visual FoxPro 自动将数据从指定代码页转换到当前 Visual FoxPro 代码页。

如果指定的 *nCodePage* 值无效，Visual FoxPro 将产生一条错误信息。您可以使用 GETCP() 函数显示“代码页”对话框，并从中指定要追加的表或文件的代码页。

如果忽略 AS *nCodePage* 子句，或者指定 *nCodePage* 值为零，将不转换文本文件的代码页。

说明

如果忽略参数 OVERWRITE，文本文件的全部内容将追加到当前记录的指定备注字段中。

示例

下面的示例将复制备注字段 notes 的内容到名为 Test.txt 的文件中。Test.txt 则被追加到备注字段的内容中。最后，Test.txt 的内容替换了备注字段的现有内容。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE employee  && 打开 Employee 表
WAIT WINDOW 'Employee notes memo field - press ESC' NOWAIT
MODIFY MEMO notes NOEDIT  && 打开 notes 备注字段
COPY MEMO notes TO test.txt  && 从备注字段创建 test 文件
WAIT WINDOW 'TEST.TXT text file - press ESC' NOWAIT
MODIFY FILE test.txt NOEDIT && 打开 text 文件
WAIT WINDOW 'Employee notes now appended - press ESC' NOWAIT
APPEND MEMO notes FROM test.txt  && 将 text 文件的内容加入
MODIFY MEMO notes NOEDIT  && 再次显示备注字段
WAIT WINDOW 'Overwrite Employee notes- press ESC' NOWAIT
```

```
APPEND MEMO notes FROM test.txt OVERWRITE && Replace notes
MODIFY MEMO notes NOEDIT NOWAIT
DELETE FILE test.txt
```

请参阅

COPY MEMO, GETCP()

返回主目录

BITAND（）函数

返回两个数值在按位进行 AND 运算后的结果。

语法

BITAND(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1, nExpression2

指定按位进行 AND 运算的两个数值。如果 *nExpression1* 和 *nExpression2* 不是整数，那么它们在按位进行 AND 运算之前将被转换为整数。

说明

BITAND（）将 *nExpression1* 的每一位同 *nExpression2* 的相应位进行比较。如果 *nExpression1* 和 *nExpression2* 的位都是 1，相应的结果位就是 1；否则相应的结果位是 0。

下表列出对 *nExpression1* 和 *nExpression2* 按位进行 AND 运算的结果：

nExpression1 位	nExpression2 位	结果位
0	0	0
0	1	0
1	1	1
1	0	0

示例

x = 3 && **二进制为** 0011
y = 6 && **二进制为** 0110

? BITAND(x,y) && return 4, **二进制为** 0010

请参阅

BITCLEAR( ), BITLSHIFT( ), BITNOT( ), BITOR( ), BITRSHIFT( ), BITSET( ), BITTEST( ), BITXOR( )

BITCLEAR（）函数

清除一个数值中的指定位（将此位设置成 0），并返回结果值。

语法

BITCLEAR(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1

指定要清除位的数值。如果 *nExpression1* 不是整数，那么在清除位之前，将被转换成整数。

nExpression2

指定 *nExpression1* 要清除位的位置。*nExpression2* 的取值范围为 0 到 31，0 表示最右端位。

示例

```
x = 7   && 二进制为 0111
y = 1   && 第二位（0 是第一位）
? BITCLEAR(x,y) && 返值类型 5, 二进制为 0101
```

请参阅

BITAND( ), BITLSHIFT( ), BITNOT( ), BITOR( ), BITRSHIFT( ), BITSET( ), BITTEST( ), BITXOR( )

BITLSHIFT（）函数

返回一个数值向左移动给定位后的结果。

语法

BITLSHIFT(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1

指定要左移的数值。如果 *nExpression1* 不是整数，那么在左移之前，将被转换成整数。

nExpression2

指定要左移的位数。如果 *nExpression2* 不是整数，那么将被转换成整数。

示例

```
x = 5   && 二进制为 0101
y = 1   && 左移一位

? BITLSHIFT(x,y)  && 返回值 10, 二进制为 1010
```

请参阅

BITAND( ), BITCLEAR( ), BITNOT( ), BITOR( ), BITRSHIFT( ), BITSET( ), BITTEST( ), BITXOR( )

BITNOT（）函数

返回一个数值按位进行 NOT 运算的结果。

语法

BITNOT(nExpression)

返值类型

数值型

参数描述

nExpression

指定按位进行 NOT 运算的数值。如果 *nExpression* 不是整数，那么在位操作之前将被转换成整数。

说明

BITNOT（）**返回 *nExpression* 按位取反操作的结果。返回值是将 *nExpression* 的每一**

位按 0 转换成 1、1 转换成 0 后得到的结果。

下表列出了对 *nExpression* 按位进行 NOT 运算的结果：

nExpression 位	结果位
0	1
1	0

示例

```
x = 5   && 二进制为 0101
? BITNOT(x)  && 返回值为 -6
```

请参阅

BITAND( ), BITCLEAR( ), BITLSHIFT( ), BITOR( ), BITRSHIFT( ), BITSET( ), BITTEST( ), BITXOR( )

BITOR（）函数

返回两个数值按位进行 OR 运算的结果。

语法

BITOR(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1, nExpression2

指定按位进行 OR 运算的数值表达式。如果 *nExpression1* 和 *nExpression2* 不是整数，它们在进行 OR 运算之前将被转换为整数。

说明

BITOR（）将 *nExpression1* 的每一位同 *nExpression2* 的相应位进行比较。如果 *nExpression1* 和 *nExpression2* 的相应位有一个是 1，相应的结果位就是 1；否则相应的结果位是 0。.

下表列出对 *nExpression1* 和 *nExpression2* 按位进行 OR 运算的结果：

nExpression1 位	nExpression2 位	结果位
0	0	0
0	1	1
1	0	1
1	1	1

示例

x = 5 && 二进制 0101
y = 6 && 二进制 0110

? BITOR(x,y) && 返回值为 7, 二进制 0111

请参阅

BITAND( ), BITCLEAR( ), BITLSHIFT( ), BITNOT( ), BITRSHIFT( ), BITSET( ), BITTEST( ), BITXOR( )

BITRSHIFT（）函数

将一个数值向右移动指定数目的位数，并返回转换后的结果。

语法

BITRSHIFT(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1

指定要右移的数值。如果 *nExpression1* **不是整数，那么在右移之前，将被转换成整数。**

nExpression2

指定 nExpression1 **要置位为** 1 **的位置。** nExpression2 ***取值范围为 0 到 31，***

0 表示最右端位。

示例

```
x = 5   && 二进制为 0101
y = 1   && 右移一位

? BITRSHIFT(x,y)   && 返回值是 2, 二进制为 0010
```

请参阅

BITAND( ), BITCLEAR( ), BITLSHIFT( ), BITNOT( ), BITOR( ), BITSET( ), BITTEST( ), BITXOR( )

BITSET（）函数

将一个数值的某位设置为 1，并返回结果。

语法

BITSET(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1

指定要置位的数值。如果 *nExpression1* 不是整数，那么在位设置之前，将被转换成整数。

nExpression2

指定 *nExpression1* 要置位为 1 的位置。*nExpression2* 的取值范围为 0 到 31，0 表示最右端位。

示例

```
x = 5 && 二进制为 0101
y = 1 && 第二位（0 = 第一位）
? BITSET(x,y) && 返回值 7, 二进制为 0111
```

请参阅

BITAND( ), BITCLEAR( ), BITLSHIFT( ), BITNOT( ), BITOR( ), BITRSHIFT( ), BITTEST( ), BITXOR( )

BITTEST（）函数

确定一个数值的指定位是否为 1。如果为 1，返回“真”(.T.)，否则返回“假”(.F.)。

语法

BITTEST(nExpression1, nExpression2)

返值类型

逻辑值

参数描述

nExpression1

指定要检查位的数值。如果 *nExpression1* **不是整数，那么该数在检查之前，将被转换成整数。**

nExpression2

指定 *nExpression1* **需要检查的位置。** *nExpression2* **的取值范围为** 0 **到** 31，0 **表示最右端位。**

示例

以下示例使用 BITTEST（）**函数确定整数系列是否为奇数。如果整数为奇数，函数** IsEven **返回“真”** (.T.)，**否则，返回“假”** (.F.)。

```
CLEAR
? '2 even? '
?? IsEven(2) &&如为奇数，返回“真” (.T.)
? '3 even? '
?? IsEven(3) &&如不是奇数，返回“假” (.F.)
? '0 even? '
?? IsEven(0) &&如为奇数，返回“真” (.T.)
? '-13 even? '
?? IsEven(-13) &&如不是奇数，返回“假” (.F.)

Function IsEven
```

```
PARAMETER nInteger
RETURN NOT BITTEST(nInteger, 0)
```

请参阅

BITAND( ), BITCLEAR( ), BITLSHIFT( ), BITNOT( ), BITOR( ), BITRSHIFT( ), BITSET( ), BITXOR( )

BITXOR( ) 函数

返回两个数值按位进行 XOR 运算的结果。

语法

BITXOR(nExpression1, nExpression2)

返值类型

数值型

参数描述

nExpression1, nExpression2

指定要按位进行 XOR 运算的两个数值。如果 *nExpression1* 和 *nExpression2* 不是整数，那么它们在位操作之前，将被转换成整数。

说明

BITXOR() 函数比较 *nExpression1* 和 *nExpression2* 的每个对应位。如果一个数的某一位为 0，并且另一个数相应位为 1，那么操作结果的对应位等于 1；否则等于 0。下表列出对 *nExpression1* 和 *nExpression2* 按位进行 XOR 运算的结果：

nExpression1 位	nExpression2 位	结果位
0	0	0
0	1	1
1	0	1
1	1	0

示例

```
x = 5   && 二进制为 0101
y = 6   && 二进制为 0110

? BITXOR(x,y) && 返回值为 3，二进制为 0011
```

请参阅

BITAND(), BITCLEAR(), BITLSHIFT(), BITNOT(), BITOR(), BITRSHIFT(), BITSET(), BITTEST()

BLANK 命令

如果发出该命令时不带任何参数，则清除当前记录中所有字段的数据。

语法

BLANK
[FIELDS FieldList]
[Scope]
[FOR lExpression1]
[WHILE lExpression2]
[NOOPTIMIZE]

参数描述

FIELDS FieldList

仅清除由 *Fieldlist* 指定的字段。默认情况下，如果省略了 FIELDS 子句，则清除该记录的所有字段的数据。在非选定工作区中指定的任何字段名都必须以工作区别名开始。

要点 如果记录指针已指向当前工作区的文件末尾，那么 BLANK 命令不清除另一个相关工作区中记录的字段数据。要使 BLANK 命令能够作用在其他相关记录的字段上，记录指针必须指向当前工作区中一个已排序的记录。

Scope

指定要清除的记录范围。只有在范围之内的记录才被清除。Scope 子句包括：ALL、NEXT *nRecords*、RECORD *nRecordNumber* 以及 REST。

包含 *Scope* 子句的命令仅对活动工作区中的表进行处理。

BLANK 默认的作用域范围为当前记录 (NEXT 1)。

FOR lExpression1

清除条件 *lExpression1* 为“真”(.T.) 的记录中的字段数据。若 *lExpression1* 为可优化表达式，则 Rushmore 优化 BLANK FOR。其它相关信息，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十五章“最优化应用程序”。

WHILE lExpression2

指定要清除的记录字段数据应满足的条件，即令逻辑表达式 *lExpression2* 为“真”(.T.)。

NOOPTIMIZE

禁止 BLANK 进行 Rushmore 优化。请参阅 SETOPTIMIZE 命令，和《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十五章“最优化应用程序”。

说明

使用 APPEND BLANK 在表尾添加一个新的空记录，使用 ISBLANK() 来确定一个记录的某一字段是否为空。

示例

以下示例打开了数据库“testdata”中的“customer”表。显示第一个记录的内容使用SCATTER命令将记录的内容保存到数组中。使用BLANK命令清除记录，并重新显示记录的内容。使用GATHER命令恢复原始记录的内容，并重新显示已恢复的记录内容。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开客户表

CLEAR
DISPLAY   && 显示当前记录
SCATTER TO gaCustomer   && 根据记录内容建立数组
BLANK   && 清除记录
DISPLAY   &&显示空记录
GATHER FROM gaCustomer   && 恢复原始记录内容
DISPLAY   && 显示经过恢复的记录
```

请参阅

APPEND, EMPTY( ), ISBLANK( ) , REPLACE, SET OPTIMIZE

BOF（）函数

确定当前记录指针是否在表头。

语法

BOF([nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea

指定在非当前工作区中打开的表所在的工作区号。

cTableAlias

指定在非当前工作区中打开的表别名。

如果受测试的表在非当前选定工作区中打开，那么使用这些可选的参数为表指定别名或所在的工作区号。若此表未在指定的工作区中打开，则 BOF（）返回“假”(.F.)。

说明

BOF（）函数可用来测试一个表的记录指针是否已到文件头。如果用户试图将记录指针移动到表的第一条记录之前，则 BOF（）返回“真” (.T.)。

示例

以下示例打开了“customer”表并一次在一页上列出了公司 (Company) 名称，以表中的最后一个记录开始。继续列表直到到达文件的开头或您选择了“取消”。

```
CLOSE DATABASES
CLEAR
OPEN DATABASE (HOME() + “samples\data\testdata”)
USE customer
GO BOTTOM
local recCtr, btnValue
```

```
recCtr = 0
btnValue = 1
DO WHILE btnValue = 1 AND NOT BOF()
    ? “Company : “ + company
    recCtr = recCtr + 1
    if (recCtr % 20) = 0 then
        btnValue =MESSAGEBOX (“单击‘确定’继续，‘取消’退出。”，33)
        clear
    endif
    Skip -1   && 上移一个记录
ENDDO
=MESSAGEBOX(“Listing complete.”,48)
```

请参阅

EOF( )

BorderColor 属性

指定对象的边框颜色。设计和运行时可用。

语法

Object.BorderColor[= nColor]

参数描述

nColor

指定边框的颜色值，
下表列出了常用的颜色值：

颜色	RGB值	nColor值
白色	255, 255, 255	16777215
黑色	0, 0, 0	0
灰色	192, 192, 192	12632256
深灰色	128, 128, 128	8421504
红色	255, 0, 0	255
深红色	128, 0, 0	128
黄色	255, 255, 0	65535
深黄色	128, 128, 0	32896
绿色	0, 255, 0	65280
深绿色	0, 128, 0	32768
青色	0, 255, 255	16776960
深青色	0, 128, 128	8421376
蓝色	0, 0, 255	16711680
深蓝色	0, 0, 128	8388608
洋红色	255, 0 ,255	16711935
深洋红色	128, 0, 128	8388736

说明

当 SpecialEffect 属性设置为 1（即 Plain）时，BorderColor 只应用于文本框和编辑框。利用 RGB（）函数可以将三部分颜色转换成一个复合的 *nColor*。

应用于

组合框，命令组，容器对象，控件对象，图像，线条，列表框，选项组，页框，形状，文本框

请参阅

BackColor 属性，颜色概览，GetColor( )　，RGB( )　，SpecialEffect 属性

BorderStyle 属性

指定对象的边框样式。设计和运行时可用。

语法

Object.BorderStyle[= nStyle]

参数描述

nStyle

对于命令组、编辑框、图像、标签、选项组或文本框，BorderStyle 属性可取

下列值：

设置	说明
0	无。图像和标签的默认值。
1	固定单线。命令组、编辑框、选项组或文本框的默认值。

对于线条或形状，BorderStyle 属性可取下列值：

设置	说明
0	透明
1	（默认值）实线。边框的外边缘就是形状的外边缘。
2	虚线
3	点线
4	点划线
5	双点划线
6	内实线

对于一个表单对象，BorderStyle 属性可取下列值：

设置	说明
0	无边框
1	单线边框
2	双线边框
3	（默认值）大小可调边框

说明

如果 BorderWidth 属性的设置大于 1，那么不管 BorderStyle 的设置如何，都将绘成实线边框；若 BorderWidth 设置为 1，BorderStyle 的效果将如上表所述。

应用于

命令组，编辑框，表单，图像，标签，线条，选项组，_SCREEN，形状，文本框

请参阅

BorderColor 属性，BorderWidth 属性，DrawStyle 属性，DrawWidth 属性

BorderWidth 属性

指定一个控件的边框宽度。设计和运行时可用。

语法

Control.BorderWidth[= nWidth]

参数描述

nWidth

指定边框宽度。范围是 0 ~ 8192。

注意 如果 BorderWidth 属性的设置大于 1，则忽略 BorderStyle 属性的设置。

说明

当 BorderWidth 属性设置为 0 时，控件没有边框。对 PageFrame 控件不能改变 BorderWidth 属性设置。

应用于

容器对象，控件对象，线条，页框，形状

请参阅

BorderStyle 属性

Bound 属性

确定一个列对象里的控件是否与列的控件源绑定。设计和运行时可用。

语法

Column.Bound[= lExpr]

参数描述

lExpr

Bound 属性的设置如下：

设置	说明
“真”(.T.)	（默认值）控件与列的控件源绑定。
“假”(.F.)	控件不与列的控件源绑定。

说明

如果列的 Bound 属性设置为“真”(.T.)，则列的 ControlSource 属性设置适用于该列和它的所有控件，这时若想再修改控件的 ControlSource 属性，则会出错。如果列的 Bound 属性设成“假”(.F.)，那么用户可以直接设定列的控件的 ControlSource 属性。如果用户再设置列的 ControlSource 属性，那么该值将覆盖控件的 ControlSource 原值。

通常，控件对应的数据类型如下表所示：

列控件基类	常用的列数据类型
复选框	逻辑值，数值型
组合框	字符型，数值型
命令按钮	字符型，数值型
编辑框	字符型
列表框	字符型，数值型
选择按钮	字符型，数值型
微调	字符型，数值型
文本框	除通用型或备注型外的任何数据类型

注意 不能将一个控件与一个通用型或备注型的列绑定在一起。

应用于

列

请参阅

AddObject 方法, ControlSource 属性, CurrentControl 属性, Sparse 属性

BoundColumn 属性

对一个多列的列表框或组合框，确定哪个列与该控件的 Value 属性绑定。设计和运行时可用。

语法

Control.BoundColumn[= nCol]

参数描述

nCol

指定与 Value 属性绑定的列的编号，*nCol* 的默认值为 1。

说明

在一个多列的列表框或组合框中，当用户需要将某一列的数据（不是第一列）放置在

控件的 Value 属性中时，应使用 BoundColumn 属性。

应用于

组合框，列表框

请参阅

Value 属性

BoundTo 属性

指定组合框或列表框的 Value 属性是否由 List 或 ListIndex 属性确定。设计和运行时可用。

语法

[Form.]Control.BoundTo[= lExpression]

参数描述

lExpression

BoundTo 属性的设置有：

设置	说明
.T.	Value 属性由 List 属性确定。
.F.	（默认值）。Value 属性由 ControlSource 属性指定的字段或变量的数据类型确定。

如果 ControlSource 属性设置的字段或变量是字符型，则 Value 属性由 List 属性确定。
如果 ControlSource 属性设置的字段或变量是数值型，则 Value 属性使用 ListIndex 属性指定的索引号。
这个设置提供了与 Microsoft Visual FoxPro 3.0 和 2.x 版本的 FoxPro 的兼容性。

说明

在 Visual FoxPro 3.0 中，如果一个组合框或列表框的 ControlSource 属性是数值型的，则指定项的 ListIndex 属性值保存在与该控件绑定的变量或字段中。将 BoundTo 属性设置为“真” (.T.)，可以将 List 属性值保存在与该控件绑定的变量或字段中。

应用于

组合框、列表框

请参阅

ControlSource 属性, ListIndex 属性, Value 属性

Box 方法

在表单对象上画矩形。

语法

Object.Box(nXCoord1, nYCoord1, nXCoord2, nYCoord2)

–或者–

Object.Box(nXCoord2, nYCoord2)

参数描述

nXCoord1, nYCoord1

指定矩形起始点的坐标。度量单位由表单的 ScaleMode 方法确定。如果省略了这些参数，则使用 CurrentX 和 CurrentY 值。

nXCoord2, nYCoord2

指定矩形终点的坐标。

说明

矩形的线宽由 DrawWidth 属性确定。如何在背景上绘出一个矩形取决于 DrawMode 和 DrawStyle 属性设置。当调用 Box 方法时，以当前的 CurrentX 和 CurrentY 属性设置终点的坐标。

应用于

表单，_SCREEN

请参阅

CurrentX, CurrentY 属性, DrawMode 属性, DrawStyle 属性, DrawWidth 属性, ScaleMode 属性

_BOX 系统变量

包含此变量是为了提供向后兼容性。可用报表设计器代替。

BROWSE 命令

打开浏览窗口，显示当前或选定表的记录。

语法

BROWSE

[FIELDS FieldList]
[FONT cFontName [, nFontSize]]
[STYLE cFontStyle]
[FOR lExpression1 [REST]]
[FORMAT]
[FREEZE FieldName]
[KEY eExpression1 [, eExpression2]]
[LAST | NOINIT]
[LOCK nNumberOfFields]
[LPARTITION]
[NAME ObjectName]
[NOAPPEND]
[NODELETE]
[NOEDIT | NOMODIFY]
[NOLGRID] [NORGRID]
[NOLINK]
[NOMENU]
[NOOPTIMIZE]
[NOREFRESH]
[NORMAL]
[NOWAIT]

[PARTITION nColumnNumber [LEDIT] [REDIT]]
[PREFERENCE PreferenceName]
[SAVE]
[TIMEOUT nSeconds]
[TITLE cTitleText]
[VALID [:F] lExpression2 [ERROR cMessageText]]
[WHEN lExpression3]
[WIDTH nFieldWidth]
[WINDOW WindowName1]
[IN [WINDOW] WindowName2 | IN SCREEN]
[COLOR SCHEME nSchemeNumber]

参数描述

FIELDS FieldList

指定显示在浏览窗口中的字段。这些字段以 *Fieldlist* **指定的顺序显示。在该字段列表中可包含其他相关表中的字段。在包含一个相关表的字段时，应在字段名前面放一个句号及相关表的别名。**

如果忽略 FIELDS **子句，表中的所有字段按其在表结构中出现的顺序显示。**

FONT cFontName [, nFontSize]

指定浏览窗口的字体及字体大小。字符型表达式 *cFontName* **指定字体，数值型表达式** *nFontSize* **指定字体大小。例如，下列子句指定显示于浏览窗口的字段为** 16 **磅** Courier **字体。**

```
FONT 'Courier',16
```

如果在包含 FONT 子句的同时省略了字体大小 *nFontSize*，则浏览窗口采用 10 磅字体。
如果省略 FONT 子句，则采用 8 磅 MS Sans Serif 字体。
如果用户指定的字体不可用，则用具有相似字体特征的字体代替。

```
 STYLE cFontStyle
```

指定浏览窗口的字形。如果省略 STYLE 子句，则采用常规字形。
如果指定的字形不可用，则使用具有相似特征的字形代替，或使用 Normal 字体。

字符	字形
B	粗体
I	斜体
N	常规
O	轮廓线
Q	不透明
S	阴影
–	删除线
T	透明
U	下划线

可用多个字符的组合来指定字形。下例打开一个浏览窗口，并采用下划线字形。

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

```
USE customer  && 打开 customer 表
IF _WINDOWS
     BROWSE FIELDS contact FONT 'System', 15  STYLE 'NU'
ENDIF
IF _MAC
BROWSE FIELDS contact FONT 'Geneva', 14  STYLE 'NU'
ENDIF
```

FOR lExpression1

指定一个条件，只有 *lExpression1* 为“真”的记录才显示于浏览窗口。

若 *lExpression1* 是一个可优化表达式，则 Rushmore 优化由 BROWSE FOR 指定的查询。为达到最佳性能，需在 FOR 子句中采用可优化表达式。其它相关信息，请参阅本书稍后 SET OPTIMIZE 命令和《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十五章“最优化应用程序”。

包含 FOR 子句使记录指针移向符合该条件的第一条记录。包含 REST 子句使记录指针处于当前位置。

REST

防止在 FOR 子句打开浏览窗口时，记录指针从当前位置移向表顶部。若不包含 REST，默认情况下，BROWSE 将记录指针放置在表顶部。

FORMAT

使用格式文件来控制浏览窗口的显示和输入数据的格式。首先应用 SET FORMAT 打开格式文件。以下是从格式文件中提取的、应用于浏览窗口的信息：

- 要浏览的字段列表

- 所有 VALID 子句
- 所有 WHEN 子句
- 所有 RANGE 子句
- 字段大小（如 PICTURE 子句所指定）
- 所有 SAY 表达式（包含 BROWSE 计算结果字段）段

下例采用格式文件检查输入到浏览窗口中的数据，忽略由 @ ... GET 指定的位置。第一行创建一个浏览字段 (cust_id)，该字段为 5 个字符宽，并且只允许输入字母和数字。第二行创建一个浏览字段 (company)，该字段不能包含空 (blank) 值，且最多只能包含 20 个字母字符。第三行创建一个浏览字段 (contact)，只在该字段为空 (blank) 时才输入数据。

以下是 CUSTENTR.FMT 格式文件的内容，用于检查输入到 customer 表中的数据：

```
@ 3,0 GET cust_id PICTURE 'NNNNN'
@ 3,0 GET company VALID company != SPACE(40) ;
     PICTURE 'AAAAAAAAAAAAAAAAAAAA'
@ 3,0 GET contact WHEN contact = SPACE(40)
```

```
* This is the program that uses the format file
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && Open customer table
SET FORMAT TO custentr.fmt
```

BROWSE FORMAT

FREEZE FieldName

允许在浏览窗口中只修改一个字段。使用 *FieldName* **指定该字段，其他字段可显示但不能编辑。**

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && Open customer table
BROWSE FIELDS phone :H = 'Phone Number:' , ;
     company :H = 'Company:' ;
     FREEZE phone
```

KEY eExpression1 [, eExpression2]

限制浏览窗口显示的记录范围。用 KEY **指定浏览窗口显示的记录的一个索引关键字值 (***eExpression1***) 或关键字值的范围 (***eExpression1***,** *eExpression2***)。所浏览的表必须已建立索引，并且索引关键字值或** KEY **子句的值必须与主索引文件或标识的索引表达式具有相同的数据类型。**

例如，customer **表包含一个保存邮政编码的字符字段。如果该表已建立邮政编码字段索引，则可在** KEY **子句指定邮政编码的范围。**

下例中，只有在 10,000 **到** 30,000 **范围内的邮政编码记录才显示在浏览窗口。**

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && Open customer table
```

```
SET ORDER TO postalcode
BROWSE KEY '10000', '30000'
```

LAST | NOINIT

保存浏览窗口外观的任何变更。这些变更可保存于 FOXUSER 文件中，包括字段列表、浏览窗口的位置及尺寸等内容的变更。

如果用户发出带有 LAST 或 NOINIT 子句的 BROWSE 命令，在 SET RESOURCE 为 ON 的情况下，以最后保存于 FOXUSER 文件中的配置打开浏览窗口，恢复最后一个 BROWSE 命令创建的浏览窗口。如果在命令窗口中最后发出的 BROWSE 命令带有一长串子句，那么使用包含 LAST 或 NOINIT 选项的 BROWSE 命令，可以避免重新键入命令。有关 FOXUSER 文件的详细内容，请参阅 SET RESOURCE。

如果最后一个浏览窗口是由一个包含 PREFERENCE 子句的 BROWSE 命令打开的，BROWSE LAST 将不再恢复原配置。如果用户按 CTRL+Q 退出浏览，则当前工作区中的浏览窗口配置的任何变更，都不会被保存。LAST 与 NOINIT 作用相同。NOINIT 是为与 dBASE 兼容而提供的。

LOCK nNumberOfFields

指定不需滚动就能在浏览窗口左分区中可见的字段数。分区的大小可自动调整以显示 *nNumberOfFields* 指定的字段数。

LPARTITION

指定光标位置在浏览窗口左分区的第一个字段。默认情况下，打开浏览窗口时，光标位于右分区的第一字段。

NAME ObjectName

为浏览窗口创建一个对象引用，允许在操作浏览窗口时使用表格控件的面向对象属性。有关在 Visual FoxPro 中的面向对象编程的其他信息，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”。有关您能够为使用 NAME 子句创建的“浏览”窗口指定的表格控件属性的内容，请参阅稍后的“表格控件”。

NOAPPEND

禁止用户通过如下方式添加表记录：按 CTRL+Y 键，或选择“表”菜单的“追加记录”命令。

要点 包含 NOAPPEND 并不禁止用户在程序中向浏览窗口中添加记录（由 VALID、WHEN 或 ON KEY LABEL 等命令创建）。

NODELETE

禁止在浏览窗口中为记录作删除标记。默认情况下，按 CTRL+T 键、选择 Visual FoxPro 中的表菜单或 FoxPro 早期版本浏览菜单的切换删除标记项，或单击要删除的记录的最左边一列，可为记录作删除标记。包含 NODELETE 不禁止用户在程序中为浏览窗口中的记录作删除标记。

NOEDIT | NOMODIFY

禁止用户修改表。NOEDIT 等同于 NOMODIFY。包含其中任何一个子句，用户可以浏览或搜索表，可以添加或删除记录，但不能编辑。

NOLGRID

删除浏览窗口左分区中字段的网格线。

NORGRID

删除浏览窗口右分区中字段的网格线。

NOLINK

将浏览窗口的左右两部分断开联系。默认情况下，浏览窗口的左分区和右分区链接在一起，即滚动其中一区时，另一区也随之滚动

NOMENU

从系统菜单栏中删除“表”菜单标题，防止访问“浏览”菜单。

NOOPTIMIZE

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

NOREFRESH

禁止对浏览窗口进行刷新。浏览窗口以 SET REPRESH 确定的速度进行刷新。NOREPRESH 对只读文件非常有用，并且可以改进性能。

NORMAL

使用正常的默认设置打开浏览窗口，如颜色、尺寸、标题、位置和控制选项（如 GROW、FLOAT、ZOOM 等等）。如果省略 NORMAL，而当前输出窗口为用户自定义窗口，浏览窗口使用用户自定义设置。

NOWAIT

打开浏览窗口之后继续运行程序。程序不必等待浏览窗口关闭，而是继续执行 BROWSE NOWAIT 之后的程序行。若在程序中发出 BROWSE 命令时省略 NOWAIT，则浏览窗口打开后，暂停执行程序，直至浏览窗口关闭。NOWAIT 只用于程序中。在

命令窗口中发出 BROWSE 命令时包含 NOWAIT 无此作用。

PARTITION nColumnNumber

将浏览窗口拆分为左右分区，*nColumnNumber* 指定拆分条所在的列。例如，*nColumnNumber* 为 20，则拆分条放置于浏览窗口的第 20 列。

LEDIT

指定浏览窗口的左分区以编辑方式显示。

REDIT

指定浏览窗口的右分区以编辑方式显示。下例打开浏览窗口，使拆分条位于第 20 列，并以编辑方式打开右分区。

若同时包含 LEDIT、REDIT 两个关键字，以编辑方式打开两个分区。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表
BROWSE PARTITION 20 REDIT
```

PREFERENCE PreferenceName

保存浏览窗口的属性和选项供以后使用。PREFERENCE 不像 LAST 以先前工作区中出现的方式打开浏览窗口，而是将浏览窗口的属性随时存储于 FOXUSER 资源文件中。PREFERENCE 可在任何时候恢复。

在第一次发出带有指定设置的 BROWSE 命令时，在 FOXUSER 文件中创建一个保存浏览窗口配置的数据项。以后用同样的状态名发出 BROWSE 命令时，恢复浏览窗口该状

态。在浏览窗口关闭时，更新该状态。

状态名可为 10 个字符长，必须以字母或下划线开头，可包括字母、数字或下划线的任意组合。

如果您喜爱某种状态，可禁止其被更改。关闭浏览窗口，发出 SET RESOURCE OFF 命令，将 FOXUSER 作为一个表打开，并将逻辑字段 READONLY 的值更改为“真”(.T.)，可将包含状态的记录改为只读记录。

有关 FOXUSER 资源文件的详细内容，请参阅 SET RESOURCE。

若按 CTRL+Q 键退出浏览窗口，则资源文件不保存任何对浏览窗口的更改。

SAVE

保持浏览窗口及其所有备注字段的文本编辑窗口为活动和可视（打开）的。用键盘或鼠标访问其他打开的窗口后，可以返回浏览窗口。SAVE 仅用于程序中。在命令窗口中发出 BROWSE 命令时包含 SAVE，则无此作用。在默认情况下，BROWSE SAVE 以交互方式工作。

TIMEOUT nSeconds

指定浏览窗口等待输入的时间。数值型表达式 *nSeconds* 指定在没有任何输入的情况下，浏览窗口保持多少秒后才自动关闭。

TIMEOUT 仅用于程序中。在命令窗口中发出 BROWSE 命令时包含 TIMEOUT，则无此作用。下例中，如果在 10 秒内没有输入，则关闭浏览窗口：

```
DEFINE WINDOW wBrowse FROM 1,1 TO 24,40 ;
    CLOSE ;
    GROW ;
```

```
    COLOR SCHEME 10
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表
BROWSE WINDOW wBrowse ;
    FIELDS phone :H = 'Phone Number:' , ;
    company :H = 'Company:' ;
    TIMEOUT 10
RELEASE WINDOW wBrowse
```

TITLE cTitleText

以 *cTitleText* 指定的标题改写显示于浏览窗口标题栏中的默认表名或别名。否则，要浏览的表的名称或别名显示于标题栏中。

如果发出 BROWSE WINDOW 命令将浏览窗口置于用户自定义窗口中，则浏览窗口的标题替换用户自定义窗口的标题。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && 打开 customer 表
BROWSE;
    TITLE 'My Browse Window' ;
    FIELDS phone :H = 'Phone Number' , ;
    company :H = 'Company:'
```

VALID lExpression2

在浏览窗口中执行记录级有效性检查。VALID 子句仅在记录已被更改并且试图将光标移至另一记录时执行。如果只是对备注字段的更改，不执行 VALID 子句。

如果 VALID 返回值为“真”(.T.)，用户可将光标移向另一记录；如果 VALID 返回值为“假”(.F.)，光标停留在当前记录上，Visual FoxPro 产生一条错误信息。如果 VALID 返回值为 0，光标停留在当前记录上，但不显示错误信息。

不要把 VALID 子句和启用字段级有效性的检验选项 (:V) 相混淆。

:F

在用户将光标移至下一个记录之前，强制执行 VALID 子句。这种情况下，即使记录没更改，也执行 VALID 。

ERROR cMessageText

指定一条错误信息来替换系统默认的错误信息。在 VALID 返回值为“假”(.F.) 时，Visual FoxPro 显示 *cMessageText*。

WHEN lExpression3

在用户把指针移到另一条记录时做出评估，若评估为真，则用户可以修改将要移到的记录，若评估为假，则该记录为只读或用户不能修改它。

指定一条错误信息来改写系统默认的错误信息。在 VALID 返回值为“假”(.F.) 时，Visual FoxPro 显示 *cMessageText*。

在激活另一窗口时，不执行 WHEN 子句。

WIDTH nFieldWidth

将浏览窗口中所有字段的字符数限制为 *nFieldWidth*，使用左右箭头或水平滚动栏，可水平滚动字段内容。使用 WIDTH 子句并不会修改表中字段的大小，而只修改字段在浏览窗口中的显示方式。如果已由 FIELDS 子句指定一个字段的宽度，它将替换 WIDTH 子句所指定的该字段的宽度。

WINDOW WindowName1

指定一个用户自定义窗口，浏览窗口将采用此窗口的特性。例如，如果该用户自定义窗口是由 FLOAT 子句创建的，则浏览窗口可移动。所指定的窗口不需要是活动的和可见的，但必须是已定义的。

IN [WINDOW] WindowName2

指定打开浏览窗口的父窗口，该浏览窗口并不采用父窗口的特性。在父窗口内激活的浏览窗口不能移出该父窗口。如果父窗口移动，则浏览窗口随之移动。

要访问浏览窗口，必须首先用 DEFINE WINDOW 定义父窗口，而且父窗口必须是活动的和可见的。

IN SCREEN

在用户自定义窗口为活动窗口时，将浏览窗口放置在 Viusal FoxPro 主窗口中。

COLOR SCHEME nSchemeNumber

指定用于浏览窗口的配色方案的编号。

浏览窗口假定配色方案已由 Windows 的颜色控制面板建立。

说明

浏览窗口允许查看、编辑表中的记录，并追加额外的记录。Visual FoxPro 允许同时打开多个浏览窗口

若按 Esc 键退出浏览窗口，则放弃对最后一个字段所做的更改。然而，若在修改一个字段后移向另一记录，则可保存对此字段所做的更改。

有关在“浏览”窗口中定位的信息，请参阅帮助中的《用户指南》第二章“创建表和索引”。

字段列表可以指定为字段或计算结果字段的任意组合。

字段列表的语法是：

```
FieldName1
    [:R]
    [:nColumnWidth]
    [:V = lExpression1 [:F] [:E = cMessageText]]
    [:P = cFormatCodes]
    [:B = eLowerBound, eUpperBound [:F]]
    [:H = cHeadingText]
    [:W = lExpression2]
    [, FieldName2 [:R]... ]
```

计算结果字段... 字段列表可包含创建计算结果字段的语句。计算结果字段包含由表达式创建的只读数据。该表达式可以是任何格式，但必须是有效的 Visual FoxPro 表达

式。

用于创建计算结果字段的语句格式为：

```
CalculatedFieldName = eExpression
```

下面的示例创建一个名为 location 的计算结果字段：

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + ‘data\testdata’)
USE customer  && 打开 customer 表
BROWSE FIELDS location = ALLTRIM(city) + ‘, ‘ + country
```

city 和 country 都是当前选定表的字段名。

FIELDS 子句的字段列表包含对显示于浏览窗口中的字段进行特殊处理的选项：

```
:R
```

指定字段为只读字段，可显示但不可编辑字段包含的数据。

下例中打开的浏览窗口包含 cust_id 和 company 字段。Cust_id 字段为只读字段，不可更改。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + ‘data\testdata’)
USE customer  && 打开 customer 表
BROWSE FIELDS cust_id:R, company
```

:nColumnWidth

指定列中字段显示的大小，*:nColumnWidth* 值不影响表中字段的大小，只影响字段在浏览窗口中显示的方式。

:V = lExpression1 [:F] [:E = cMessageText]

检验选项 (:V) 在浏览窗口中执行字段级数据有效性检查。若从一个字段移去光标时，*lExpression1* 值为“真” (.T.)，则认为输入到该字段的数据正确，光标移向下一字段。

若 *lExpression1* 值为“假” (.F.)，则认为所输入数据不正确，光标停留在当前字段并显示一条错误信息。若 *lExpression1* 值为 0，则认为所输入数据不正确，光标仍停留在当前字段但不显示错误信息。

对于备注字段不执行此检验选项。

默认情况下，*lExpression1* 只在修改字段时求值。欲强制执行检验，需包括 :F 选项。

用户可包含下面所述的 :E 选项来显示自己定义的错误信息。

:F

决定当从一个字段移去光标或激活另一个窗口时，是否对检验选项中的表达式求值。若不包括 :F，则 *lExpression1* 仅在修改字段时求值。若包括 :F，则即使没有修改字段也对 *lExpression1* 求值。

:E = cMessageText

如果有效性检测表达式 :V = *lExpression1* 值为“真”，则允许光标移动。如果该表达式值为“假”，光标停留在当前字段，Visual FoxPro 产生一个错误信息。

如果包含错误选项 (:E)，*cMessageText* 代替系统错误信息出现。*CMessageText* 仅在 SET NOTIFY 为 ON 时出现。若 SET BELL 为 ON，则响铃。

如果 :V = *lExpression1* 值为 0，则不出现错误信息，光标停留于当前有效字段。这使得用户可以在有效性检测例程中显示自己定义的错误信息。

下例打开 products 表并显示 product_id 和 prod_name 字段。Product_id 字段是一个可接收 5 个数字的数值型字段。在该例中，假定大于 100 的 product_id 是无效的。

:V 指定有效性准则。不论数据是否更改，:F 都强制执行有效性检查。:E 使用用户自定义错误信息来代替 Visual FoxPro 系统错误信息。在 Visual FoxPro、ForPro for Windows 和 FoxPro for Macintosh 中，错误信息显示于 Visual FoxPro 主窗口底部的状态栏中。

按下 ESC 键可关闭浏览窗口。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && Open products table
IF _WINDOWS OR _MAC
     SET STATUS BAR ON
ENDIF
USE products
BROWSE FIELDS in_stock :V = in_stock < 100 ;
      :F ;
      :E = 'The stock amount must be less than 100'
```

:P = cFormatCodes

如果包含 FIELDS 子句，用户也可为列表中的每一字段指定一个图片选项 (:P)。图片选项可创建一个代码列表，该列表控制浏览窗口中每一字段的显示和数据输入。*CFormatCodes* 即为代码列表。

下例采用图片选项，只允许以指定格式的数值型数据输入到 unit_price 字段中。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && Open products table
BROWSE FIELDS unit_price :P = '99,999.99'
```

有关图片选项代码的详细内容，请参阅稍后的“Format 和 InputMask 属性”。

:B = eLowerBound, eUpperBound [:F]

指定一对边界，字段数据必须处于边界范围内。边界表达式 *eLowerBound* 和 *eUpperBound* 必须与字段的数据类型匹配，并且不能是用户自定义函数。如果数据不处在 *eLowerBound* 和 *eUpperBound* 之间，则出现一条系统错误信息，指出数据允许的范围。

默认情况下，仅在修改字段内容时对输入的数据作边界值检查。要想强制执行边界值检查，需包含强制有效性选项 (:F)。

下例确保 in_stock 字段值处于 1 和 100 之间。按 ESC 键可关闭浏览窗口。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && Open products table
BROWSE FIELDS in_stock :B = 1, 100   :F
```

:H = cHeadingText

用 *cHeadingText* 指定的自定义标头替换默认字段名。默认情况下，在浏览窗口中用字段名作列标头。

下例为所显示的字段提供自定义列标头。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && Open products table
BROWSE FIELDS prod_name :H = 'Product Name:', ;
     unit_price :H = 'Price per Unit:'
```

:W = lExpression2

根据逻辑表达式 *lExpression2* 的值，用户可以禁止光标移向某一字段。如果 *lExpression2* 值为“假” (.F.)，则禁止光标移向该字段。如果 *lExpression2* 值为“真” (.T.)，光标可移向该字段。*LExpression2* 支持用户自定义函数。

如果禁止光标移向所有字段，则当前记录标记为只读记录。只有在每一字段均包含一个值为“假”的 WHEN 子句时，这种情况才出现。

SET SKIP 支持... 允许用户在两表之间建立一对多关系。对于父表中的每一条记录，均可在子表中有多个相关记录。如果创建一对多关系，那么使用 BROWSE 命令可以查看父表和子表中的记录。

一旦显示父记录，子表的第一条匹配记录接着显示。随后的任何匹配记录均在父记录和第一条匹配记录之后显示。

如果记录指针位于一个父记录上，可以在浏览窗口中，按 CTRL+DOWN ARROW 可以将记录指针移动到后一个父记录，按 CTRL+UP ARROW 可以将记录指针移动到前一个父记录。有关一对多关系，请参阅稍后的 SET SKIP 命令。

FIELDS 子句的字段列表包含父表和子表的记录。字段名前应加上表别名（orders 或 customer）和句点分隔符。

```
CLEAR
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer ORDER cust_id IN 0  && 父表
USE orders ORDER cust_id IN 0  && 子表
SELECT customer    && 返回父表工作区
SET RELATION TO cust_id INTO orders    && 创建关系
SET SKIP TO orders  && 一对多关系
WAIT WINDOW 'Scroll to see shipping dates for each customer' NOWAIT
BROWSE FIELDS customer.cust_id :H='Customer Number', ;
    customer.city :H='Customer City', orders.shipped_on
```

有用的函数... 几个 Visual FoxPro 函数返回有关浏览窗口的有用信息。

函数	说明
VARREAD( )	RECNO( )
返回活动浏览窗口中光标所在的字段名。	返回活动浏览窗口中选定的记录号。

请参阅

CHANGE, EDIT, Grid Control, SET SKIP, WTITLE( )

_BROWSER 系统变量

包含类浏览器的应用程序名称。

语法

_BROWSER = cProgramName

参数描述

cProgramName

指定类浏览器应用程序。如果已安装类浏览器应用程序，_BROWSER 包含 BROWSER.APP。Visual FoxPro 的专业版中有 BROWSER.APP。

如果类浏览器位于非当前默认文件夹中，应该包含具有类浏览器应用程序名称的路径。

使用以下语法，可以在 Visual FoxPro 配置文件中指定类浏览器应用程序。

_BROWSER = cProgramName

可以在命令窗口或程序中通过向 _BROWSER 存储新的应用程序名而指定另一个类浏览器。

说明

Visual FoxPro 的专业版提供一个名为 BROWSER.APP 的类浏览器。BROWSER.APP 显示可视

类库 (.VCX) 中类之间的关系。默认情况下，BROWSER.APP 安装在 Visual FoxPro 目录中。

用户可发出下列命令运行类浏览器：

```
DO (_BROWSER)
```

发出该命令在工具菜单中添加一个类浏览器菜单项，并启动类浏览器。选择该类浏览器菜单项，可再次运行类浏览器。

若要在工具菜单中添加一个类浏览器菜单项但不启动类浏览器。可发出以下命令：

```
DO (_BROWSER) WITH 0
```

用户也可运行一个特殊可视类库或一个对象引用的类浏览器，也可任意包含一个初始选定的类名。例如，下面的命令启动类浏览器，打开按钮可视类库并选择 VCR 类：

```
DO (_BROWSER) WITH HOME（）+ '\SAMPLES\CONTROLS\BUTTONS.VCX', 'VCR'
```

其它相关信息，请参阅“使用‘类浏览器’管理类”。

请参阅

CREATE CLASS, CREATE CLASSLIB, CREATE FORM, CREATEOBJECT( ), DEFINE CLASS, **系统变量概述**

BufferMode 属性

指定保守式更新还是开放式更新记录。设计和运行时可用。

语法

Object.BufferMode[= nValue]

参数描述

nValue

BufferMode 属性的设置为：

设置	说明
0	（默认值）无。在开始编辑时，记录锁定。在记录指针移动时写字段。模拟 FoxPro 2.x 行为。
1	保守式。在开始编辑时，记录锁定，在记录指针移动时写字段。可用 TABLEREVERT（）来撤消对当前记录的更改。
2	开放式。编辑时记录不锁定。当使用 TABLEUPDATE（）将记录写向磁盘时，Visual FoxPro 尝试锁定记录。

说明

若 BufferMode 设置为 1 或 2，表格控件所用的任何临时表采用表缓冲。数据的其他任何控件均采用行缓冲。

应用于

表单，表单集，_SCREEN

请参阅

BufferModeOverride 属性，TABLEREVERT()，TABLEUPDATE()

BufferModeOverride 属性

指定是否改写表单级或表单集级的 BufferMode 属性设置。设计和运行时可用。

语法

DataEnvironment.Cursor.BufferModeOverride[= nValue]

参数描述

nValue

下表列出了 BufferModeOverride 属性的设置：

设置	说明
0	无。不使用缓冲。
1	（默认值）使用表单设置。在表单级或表单集级使用 BufferMode 属性设置。
2	保守式行缓冲。锁定记录并缓冲更改，直至记录指针移动。可使用 TABLEREVERT（）函数撤消更改。
3	开放式行缓冲。允许编辑单个记录，仅在将记录写入磁盘时锁定记录。可使用 TABLEREVERT（）函数撤消更改。

续表

4	保守式表缓冲。锁定每个编辑的记录，但在调用 TABLEUPDATE（）函数前记录不写入磁盘。可使用 TABLEREVERT（）函数撤消更改。
5	开放式表缓冲。允许编辑所有记录，在以 TABLEUPDATE（）函数将记录写入磁盘前不锁定该记录。可使用 TABLEREVERT（）函数撤消更改。

说明

如果临时表基于本地视图或远程视图，BufferModeOverride 只能设置为 3 和 5。如果表单集或表单的 BufferMode 属性设置为 1（保守式），则基于视图的临时表的 BufferModeOverride 默认设置为 3（开放式行缓冲）。

应用于

临时表

请参阅

BufferMode 属性，TABLEREVERT()，TABLEUPDATE()

BUILD APP 命令

使用项目文件生成以 .APP 为扩展名的应用程序。

语法

BUILD APP APPFileName FROM ProjectName [RECOMPILE]

参数描述

APPFileName

指定要生成的应用程序文件名称。默认应用程序文件扩展名为 .APP。

FROM ProjectName

指定一个要生成 .APP 应用程序的项目名。

RECOMPILE

在生成应用程序前重新编译整个项目。项目中的所有组件：程序，表单，标签，报表，可视类库，内部存储过程等都要重新编译。

说明

用 CREATE PROJECT 或 MODIFY PROJECT 命令创建一个项目文件。项目文件是一个表，该表的文件扩展名为 .PJX（表文件）和 .PJT（备注文件）。

在使用 BUILD APP 之前，确保项目文件包含应用程序所需的所有文件。如果在生成

期间遗漏所需文件，Visual FoxPro 产生一个错误。这个错误和编译时产生的其他错误都存储于一个扩展名为 .ERR 的错误文件中。错误文件名的前 8 个字符和项目名的前 8 个字符相同。

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

请参阅

BUILD EXE 命令, Build 方法, BUILD PROJECT, CREATE PROJECT, MODIFY PROJECT 命令

BUILD DLL 命令

使用项目文件生成以.dll 为扩展名的动态链接库（DLL）。

语法

BUILD DLL DLLFileName FROM ProjectName [RECOMPILE]

参数描述

DLLFileName

指定动态链接库的文件名。默认的后缀名是 .DLL。

FROM ProjectName

指定项目的名称。BUILD DLL 命令根据这个项目创建动态链接库。

在这个项目中必须包含一个指定为 OLEPUBLIC 的类，否则系统将产生一个错误信息。若要将一个类指定为 OLEPUBLIC，可以在定义该类的 DEFINE CLASS 命令中包含 OLEPUBLIC 关键字，或在使用“类设计器”定义类时，从“类”菜单中选择“类信息”，并在“类信息”对话框中选择“OLE 公有”。

RECOMPILE

指定在生成动态链接库前，重新编译项目。项目中的所有组件：程序，格式文件，表单，报表，标签，可视类库，内部存储过程 (stored procedures)，等等都将被重新编译。

说明

BUILD DLL 在生成动态链接库后，自动注册它，并在“项目信息”对话框中的“服务器”选项卡中的“服务器类”列表框中显示它。

有关创建自定义 Automation 服务程序的详细信息，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十六章“添加 OLE”中的“创建 Automation 服务程序”。

请参阅

BUILD EXE 命令, Build 方法, BUILD PROJECT, CREATE PROJECT, MODIFY PROJECT 命令

BUILD EXE 命令

从项目生成一个可执行文件。

语法

BUILD EXE EXEFileName FROM ProjectName [RECOMPILE]

参数描述

EXEFileName

指定要生成的可执行文件名。如果一个 .APP 应用程序文件和现有可执行文件有相同的根文件名，则删除该应用程序文件。请注意，如果已有一个可执行文件，而用户生成一个有相同文件名的 .APP 文件，则删除该可执行文件。.

FROM ProjectName

指定项目名，在该项目中生成可执行文件。

RECOMPILE

在生成应用程序前重新编译整个项目。项目中的所有组件：程序，表单，标签，报表，可视类库，内部存储过程，等等都要重新编译。

说明

使用 BUILD EXE 创建的可执行文件需要两个支撑文件：VFP6r.DLL 和 VFP6renu.DLL

(文件名中的 EN 表示 English)。这两个文件必须与可执行文件包含在同一个目录中。
如果在可执行文件中包含 OLEPUBLIC 类定义，BUILD EXE 会自动把 OLEPUBLIC 类定义注册到系统注册表中。在“项目信息”对话框中，“服务器”标签所在页的“服务器类”列表中显示该 OLEPUBLIC 类定义。
如果在可执行文件中包含 OLEPUBLIC 类定义，BUILD EXE 会自动把 OLEPUBLIC 类定义注册到系统注册表中。在“项目信息”对话框中，“服务器”标签所在页的“服务器类”列表中显示该 OLEPUBLIC 类定义。
BUILD EXE 同时创建了与可执行文件同名.vbr（注册）和.tlb（类型库）文件。当可执行文件移动到其他计算机上时，.vbr 文件允许您在系统注册表中注册类定义。.tlb 文件用于对象浏览器。
有关在可执行文件中注册 OLEPUBLIC 类定义的详细信息，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十六章“添加 OLE”。
请参阅

BUILD APP 命令, Build 方法, BUILD PROJECT 命令

Build 方法

重新连编项目，或者根据项目创建一个应用程序文件 (.app)、动态链接库 (.dll) 或可执

行文件 (.exe)。

语法

Object.Build([cOutputName] [, nBuildAction] [, lRebuildAll] [, lShowErrors] [, lBuildNewGUIDs])

参数描述

cOutputName

指定要创建的应用程序、动态链接库或可执行文件的名称。

如果 cOutputName 包含一个扩展名，并且省略了 nBuildAction，则 cOutputName 指定的扩展名确定了要连编文件的类型。

nBuildAction

指定是重新连编该项目，还是创建一个应用程序、动态链接库或可执行文件。

下表列出了 nBuildAction 的值，以及每个值的说明。

nBuildAction	FoxPro.h 常数	说明
1	BUILDACTION_REBUILD	（缺省）重新连编项目。
2	BUILDACTION_BUILDAPP	创建一个 .app 文件。

续表

3	BUILDACTION_BUILDEXE	创建一个 .exe 文件。
4	BUILDACTION_BUILDDLL	创建一个 .dll 文件。

如果 cOutPutName 不包含扩展名，会添加相应的扩展名。

lRebuildAll

指定在创建一个 .app、.dll 或 .exe 文件之前，是否重新编译项目中的文件。如果 lRebuildAll 为“真” (.T.)，则重新编译以下文件：

- 程序文件。
- 格式文件。
- 表单、报表和可视类库中的源代码。
- 数据库中的存储过程。

如果 lRebuildAll 为“假” (.F.) 或省略，则在创建一个 .app、.dll 或 .exe 文件之前，不重新编译项目中的文件。

lShowErrors

指定在连编之后是否在一个编辑窗口中显示编译错误。如果 *lShowErrors* 为“真” (.T.)，则显示编译错误。如果 *lShowErrors* 为“假” (.F.) 或省略，则不显示编译错误。

lBuildNewGUIDs

指定当创建一个可执行文件或动态链接库时是否生成新的注册 GUID。如果

lBuildNewGUIDs 为“真”(.T.)，则生成新的 GUID。如果 lBuildNewGUIDs 为“假”(.F.)或省略，则不生成新的 GUID。如果 nBuildAction 小于 3，则不生成新的 GUID。。

说明

如果项目成功地重新连编，或者成功地创建了 .app、.dll 或 .exe 文件，则返回一个逻辑“真”(.T.)；否则返回“假”(.F.)。

应用于

项目对象

请参阅

BUILD APP 命令, BUILD DLL 命令, BUILD EXE 命令, BUILD PROJECT 命令, CREATE PROJECT 命令, MODIFY PROJECT 命令, Refresh 方法

BUILD PROJECT 命令

创建并生成一个项目文件。

语法

BUILD PROJECT ProjectFileName [RECOMPILE]

[FROM ProgramName1 | MenuName1 | ReportName1 | LabelName1
| FormName1 | LibraryName1
[, ProgramName2 | MenuName2 | ReportName2 | LabelName2
| FormName2 | LibraryName2 ...]]

参数描述

ProjectFileName

指定要生成的项目文件名。

RECOMPILE

编译项目中的所有文件。如果省略 RECOMPILE，**当生成项目时，只编译被更改过的文件。**

FROM ProgramName1 | MenuName1 | ReportName1 | LabelName1
| FormName1 | LibraryName1

指定项目所包含的文件。可指定一个或多个程序、菜单、报表、标签、表单或库文件。项目跟踪这些文件的变化情况，包括它们之间的相关性、引用和连接等。

默认情况下， FROM **子句中的第一个可执行程序或菜单文件是项目的主程序文件。**

说明

通过打开并处理一个或多个指定的程序、菜单、报表、列表、标签或库文件， BUILD PROJECT **自动生成一个文件扩展名为** .PJX **的项目文件。可以使用项目文件生成以下两种类型程序：以** .APP **为扩展名的程序文件或以** .EXE **为扩展名的可执行文件。项目文件保持对生成一个应用程序所需的所有文件的追踪，包括文件间的相关性、引用以**

及关系。一旦用户指定了项目的内容，Visual FoxPro 确保应用程序是基于最新的源文件。

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

在 BUILD PROJECT 生成一个项目文件时，如果 Visual FoxPro 遇到一个程序、菜单或表单文件，它将查找其编译文件并比较两个文件的时间和日期标记。如果源文件的时间和日期标记比编译文件晚，Visual FoxPro 将编译源文件。

每一个项目文件均包含一个时间日期戳，便于用户在更改项目文件或更改相关性时刷新项目文件。这便于确保项目文件生成的所有应用程序一直使用最新的源文件。发出不包含 FROM 子句的 BUILD PROJECT 命令，可刷新项目文件，并更新所指定的项目。

在发出 BUILD PROJECT 命令时，Visual FoxPro 将报告不能识别的引用和其他错误，但不禁止生成项目文件列表。这样，即使所有必需内容并未实际创建或不可用，仍然可以生成项目。通过以后刷新项目文件或以 MODIFY PROJECT 命令手工修改存储于项目文件中的信息，可以解决不能识别的引用和其他问题。

请参阅

BUILD APP 命令, BUILD EXE 命令, Build METHOD 命令, BuildDateTime 属性, CREATE PROJECT 命令, EXTERNAL 命令, MODIFY PROJECT 命令

BuildDateTime 属性

包含项目最后的连编日期和时间。只读。

语法

Object.BuildDateTime

说明

可以使用 BUILD PROJECT 或者在“连编选项”对话框选择“重新连编项目”按钮，可以连编项目。

如果项目没有被编译，则 BuildDateTime 包含空的日期时间值。

应用于

项目对象

请参阅

Build 选项, BUILD PROJECT

_BUILDER 系统变量

如果项目没有被编译，则 BuildDateTime 包含空的日期时间值。

语法

_BUILDER = cProgramName

参数描述

cProgramName

指定一个生成器应用程序。如果生成器应用程序不在当前 Visual FoxPro 默认目录中，应在应用程序名中包含路径。

说明

当用户在表单设计器中选择一个生成器时，_BUILDER 系统变量包含 Visual FoxPro 使用的应用程序名。默认情况下，_BUILDER 包含安装在 Visual FoxPro 目录下的 BUILDER.APP。用户也可指定一个不同的生成器应用程序。

请参阅

系统变量概览

ButtonCount 属性

指定命令组或选项组中的按钮数。设计和运行时可用。

语法

Control.ButtonCount[= nNumber]

参数描述

nNumber

为控件指定按钮数目。

说明

使用 ButtonCount 属性来动态设置包含于命令组或选项组中的按钮数。

如果在运行时更改按钮数，则自动给定按钮命名。命令组中按钮被命名为 Command*N*，选项组中按钮被命名为 Option*N*，其中 *N* 为所添加的按钮数。例如，如果命令组中有四个按钮而 ButtonCount 属性设置更改为 5，则新按钮命名为 Command5。

示例

下例创建一个选项组控件并将该控件放置在表单中。选项组控件有三个按钮，根据用户单击的选项按钮，显示一个圆、椭圆或正方形。ButtonCount 属性用于指定选项组中

的按钮数。Buttons 和 Caption 属性用于指定显示在每一选项按钮旁的文本。

形状控件用于创建圆、椭圆和正方形。选项组控件的 Click 事件采用 DO CASE…ENDCASE 结构，并在用户单击一个选项按钮时，Value 属性显示相应的形状。

```
frmMyForm = CREATEOBJECT('Form') && 创建窗体
frmMyForm.Closable = .F. && 禁止控件菜单栏

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" 命令按钮可视

frmMyForm.opgOptionGroup1.Buttons(1).Caption = "\<Circle"
frmMyForm.opgOptionGroup1.Buttons(2).Caption = "\<Ellipse"
frmMyForm.opgOptionGroup1.Buttons(3).Caption = "\<Square"
frmMyForm.opgOptionGroup1.SetAll("Width", 100) && 设定选项组宽度
frmMyForm.opgOptionGroup1.Visible = .T. && 选项组可视
frmMyForm.opgOptionGroup1.Click &&显示圆

frmMyForm.SHOW && 显示表单
READ EVENTS && 启动事件程序
```

```
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'  && 给命令按钮加标题
    Cancel = .T.  && 默认取消命令按钮(Esc键)
    Left = 125  && 命令按钮列
```

```
    Top = 210 && 命令按钮行
    Height = 25 && 命令按钮高

    PROCEDURE Click
        CLEAR EVENT && 终止事件程序，关闭表单
ENDDEFINE
DEFINE CLASS shpMyCircle AS SHAPE && 创建圆
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 99
    BackColor = RGB(255,0,0) && 背景为红色
ENDDEFINE
DEFINE CLASS shpMyEllipse AS SHAPE && 生成椭圆
    Top = 35
    Left = 200
    Width = 100
    Height = 50
    Curvature = 99
    BackColor = RGB(0,128,0) && 背景为绿色
ENDDEFINE

DEFINE CLASS shpMySquare AS SHAPE && 生成正方形
    Top = 10
    Left = 200
    Width = 100
```

```
    Height = 100
    Curvature = 0
    BackColor = RGB(0,0,255) && 背景为蓝色
ENDDEFINE
```

应用于

命令组，选项组

请参阅

Buttons 属性

Buttons 属性

访问一个组内每个按钮的数组。设计时不可用。

语法

Control.Buttons (nIndex).Property= Value

–**或者**–

Control.Buttons (nIndex).Method

参数描述

nIndex
是 1 和 *Control* 的 ButtonCount 属性所指定的按钮数之间的一个整数。

Property
命令按钮或选项按钮的属性。

Value
Property 的值。

Method
命令按钮或选项按钮的方法。

说明

可使用 Buttons 属性为组内的所有按钮设置属性和调用方法。Buttons 属性是在创建按钮组时创建的一个数组。

应用于

命令组，选项组

请参阅

ButtonCount 属性，SetAll 方法

示例

下例创建一个选项组控件并在一个表单上设置该控件。选项组有三个按钮，根据用户单击的选项按钮，显示一个圆、椭圆或正方形。ButtonCount 属性用于指定选项组的按钮数，Buttons 和 Caption 属性用于指定显示在每一选项按钮旁的文本。
形状控件用于创建圆、椭圆和正方形。选项组控件的 Click 事件采用 DO CASE ...

ENDCASE 结构，并在用户单击一个选项按钮时，Value 属性显示相应的形状。

```
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. &&废止“控件”菜单框
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') && 添加框
frmMyForm.cmdCommand1.Visible =.T. && "Quit" “退出”命令按钮可视
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. &&选项组可见
frmMyForm.opgOptionGroup1.Click &&显示圆
frmMyForm.SHOW && 显示表单
READ EVENTS && 启动事件处理
DEFINE CLASS opgMyOptGrp AS OptionGroup && 创建一个选项组
    ButtonCount = 3 && 三个选择按钮
    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' && Caption on the 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 && 命令按钮高度
    PROCEDURE Click
        CLEAR EVENTS && 终止事件处理，关闭表单
ENDDEFINE
DEFINE CLASS shpMyCircle AS SHAPE && 创建一个圆
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 99
    BackColor = RGB(255,0,0) && 红色
ENDDEFINE
DEFINE CLASS shpMyEllipse AS SHAPE && 创建一个椭圆
    Top = 35
    Left = 200
    Width = 100
```

```
    Height = 50
    Curvature = 99
    BackColor = RGB(0,128,0) && 绿色
ENDDEFINE
DEFINE CLASS shpMySquare AS SHAPE && 创建一个正方形
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 0
    BackColor = RGB(0,0,255) && 蓝色
ENDDEFINE
```

_CALCMEM 系统变量

包含 Visual FoxPro 存储在计算器内存中的数值。

语法

_CALCMEM = nCalculatorValue

参数描述

nCalculatorValue

指定由 _CALCMEM 存储到内存中的数值。

说明

可在计算器内存中保存计算结果，并将结果返回给一个程序，或者在使用计算器之前向计算器内存中设置一个指定值。

要初始化计算器内存，可用 STORE 或 = 赋值操作符在 _CALCMEM 中存储一个数值。当使用计算器时，内存中就包含了指定的值。

示例

下列程序示例将数值常量 1234 存储到 _CALCMEM 中。然后程序显示计算器并用字母“R”填充键盘缓冲区。键击 R 可以显示存储于计算器内存中的值。

```
STORE 1234 TO _CALCMEM
ACTIVATE WINDOW calculator
CLEAR TYPEAHEAD
KEYBOARD CHR(82)
```

请参阅

ACTIVATE WINDOW 命令, _CALCVALUE, STORE 命令

CALCULATE 命令

对表中的字段或包含字段的表达式进行金融和统计操作。

语法

CALCULATE eExpressionList
[Scope] [FOR lExpression1] [WHILE lExpression2]
[TO VarList | TO ARRAY ArrayName],
[NOOPTIMIZE]

参数描述

eExpressionList

指定表达式，该表达式可以包含下列函数的任意组合：

AVG(nExpression)
CNT()
MAX(eExpression)
MIN(eExpression)
NPV(nExpression1, nExpression2 [, nExpression3])
STD(nExpression)
SUM(nExpression)
VAR(nExpression)

用逗号分隔表达式列表 *eExpressionList* **中的函数。这些函数仅用于** CALCULATE **命令。在本段后面将详细说明这些函数。不要与有相似名称的独立函数相混淆。例如，** CALCULATE MIN（）**与** MIN（）**不同。**

Scope

指定计算中所使用记录的范围。只有在范围之内的记录才进行计算。 Scope **子句有：** ALL、NEXT *nRecords*、RECORD *nRecordNumber* **和** REST。**关于** *Scope* **子句，请参阅帮助中的“** Scope Classes**”。包含** *Scope* **子句的命令只能在活动工作区内的表上操作。**

CALCULATE **默认的范围是“全部”记录** (ALL)。

FOR lExpression1

指定只有满足逻辑条件 *lExpression1* 的记录才进行计算。在计算中包含 FOR 子句可以有条件地选择记录，筛选出不想要的记录。

若 *lExpression1* 是可优化的表达式，Rushmore 技术将优化 CALCULATE...FOR 查询。为了获得最佳性能，应在 FOR 子句中使用可优化表达式。有关 Rushmore 优化表达式的详细信息，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》的第十五章“优化应用程序”中的“掌握 Rushimore 技术”。

WHILE lExpression2

指定一个条件，只要逻辑表达式 *lExpression2* 计算为“真”(.T.)，记录就进行计算，直至遇到使该表达式的值为“假”(.F.) 的记录为止。

TO VarList

指定一个或多个用以存储计算结果的变量。若指定的变量名不存在，Visual FoxPro 自动用该名称创建变量。

TO ARRAY ArrayName

指定存储计算结果的数组。若指定的数组名不存在，Visual FoxPro 自动用该名称创建数组；若该数组存在，但容纳不下所有的计算结果，Visual FoxPro 自动增大数组以容纳信息。若数组的大小比需要的大，那么数组中多余单元的内容保持不变。计算结果按照 CALCULATE 命令指定的顺序保存到数组中。

NOOPTIMIZE

禁止 CALCULATE 进行 Rushmore 优化。有关详细信息，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》的第十五章“优化应用程序”中的“掌握 Rushimore 技术”。

AVG(nExpression)

计算 *nExpression* 的算术平均值。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

CNT()

返回表中记录的数目。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

MAX(eExpression)

返回 *eExpression* 的最大值或最新值。在 MAX() 子句中可指定任何字符型、日期型、日期时间型、数值型、浮点型、整型、双精度型或货币型字段，或包含这些类型字段的表达式。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

MIN(eExpression)

返回 *eExpression* 的最小值或最早值。*eExpression* 中可包括任何字符型、日期型、日期时间型、数值型、浮点型、整型、双精度型或货币型字段，或任何使用这些类型字段的有效表达式。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

NPV(nExpression1, nExpression2 [, nExpression3])

计算一个固定周期利率下，一系列现金流的净现值。

nExpression1 指定用十进制表示的利率。

nExpression2 指定代表一系列现金流的字段、字段表达式或数值表达式。每个现金流可正可负。当 *nExpression2* 是字段时，每个记录的字段值都认为是一个现金流。

nExpression3 指定可选的初始投资。如果不包括初始投资，则假定初始投资发生在第一阶段末。这个初始投资就是第一个记录，而且是负的，代表现金流出。

只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

STD(nExpression)

计算 *nExpression* 的标准偏差。标准偏差用以衡量字段或包含字段的表达式的值偏离平均值的程度。标准偏差越小，这些值偏离平均值就越少。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

SUM(nExpression)

对 *nExpression* 的值求和。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

VAR(nExpression)

从 *nExpression* 的平均值中计算方差。方差越小，值偏离平均值就越少。只有满足 *Scope* 和/或可选的 FOR 或 WHILE 条件的记录才包括到结果中。

说明

含有 null 值的记录不包含在 CALCULATE 的操作中。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE orders  && 打开 Orders 表
```

```
SET TALK ON
CLEAR

CALCULATE AVG(order_amt), MIN(order_amt), MAX(order_amt)
CALCULATE STD(order_amt), VAR(order_amt) TO gnStd, gnVar
```

请参阅

AVERAGE, COUNT, DIMENSION, FV( ), MAX( ) , MIN( ) , PAYMENT( ) , PV( ) , SUM

_CALCVALUE 系统变量

包含计算器显示的数值。

语法

_CALCVALUE = nCalculatorDisplayValue

参数描述

nCalculatorDisplayValue

指定计算器显示的数值。

说明

在使用计算器之前，可向程序返回计算器结果，或设置一个特定数值显示在计算器中。

要初始化计算器显示，可用 STORE 或 = 赋值操作符在 _CALCVALVE 中存储一个数值。当使用计算器时，该值就显示在计算器中。

示例

在下列示例中，先将 1234 保存到 _CALCVALUE 中，然后计算器显示值 1234。

```
STORE 1234 TO _CALCVALUE
ACTIVATE WINDOW calculator
```

请参阅

ACTIVATE WINDOW, _CALCMEM, STORE

CALL 命令

包含此命令是为了提供向后兼容性。可用 SET LIBRARY 代替。

CANCEL 命令

停止当前 Visual FoxPro 程序的执行。

语法

CANCEL

说明

当交互使用 Visual FoxPro 时，控制权返回命令窗口。

若执行一个独立的发布应用程序，CANCEL 终止该应用程序并将控制权返回 Windows。若设计时在 Visual FoxPro 中执行一个程序，CANCEL 终止该程序，并将控制权返回命令窗口。

执行 CANCEL 将释放所有私有变量。

示例

以下示例模仿了程序执行的循环语句。每次进入循环语句都向您提问是否继续。如果单击“取消”按钮，程序执行停止。

```
DO WHILE .T.
   IF MESSAGEBOX("Do you want to continue?",36) <> 6
      CANCEL
   ENDIF
ENDDO
```

请参阅

QUIT, RESUME, RETURN, SUSPEND

Cancel 属性

指定一个命令按钮或 OLE 容器控件是否为“取消”按钮；即当用户按 ESC 键时，“取消”按钮的 Click 事件发生。设计和运行时可用。

语法

Object.Cancel[= lExpr]

参数描述

lExpr

Cancel **属性的设置是：**

设置	说明
“真” (.T.)	命令按钮或 OLE 容器控件是“取消”按钮。
“假” (.F.)	（默认值）命令按钮或 OLE 容器控件不是“取消”按钮。

说明

Cancel 属性仅用于特定的 OLE 容器控件，此容器控件包含“动作类似于按钮”的 ActiveX 控件。

示例

下列示例创建了一个命令按钮和一个选项组控件，并把它们放在一个表单中。Cancel 属性用来将命令按钮指定为“取消”按钮。若按 ESC 键，命令按钮的 Click 事件发生，并且 Click 事件执行 CLEAR EVENTS 来关闭表单和终止事件处理。

选项组控件有三个按钮，根据用户单击的选项按钮，可显示一个圆、椭圆或正方形。Shape 控件用来创建圆、椭圆和正方形。在选项组控件的 Click 事件中使用 DO CASE ... ENDCASE 结构和 Value 属性，按下选项按钮时可以显示适当的形状。

```
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. && 禁止“控件”菜单框

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. && “退出”命令按钮可见

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.  && 选项组可见
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  && 创建一个选项组
    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  && Create 创建命令按钮
    Caption = '\<Quit'  && 命令按钮标题
    Cancel = .T.  && 默认取消按钮(Esc)
    Left = 125  && 命令按钮列
    Top = 210  && 命令按钮行
    Height = 25  && 命令按钮高度

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

DEFINE CLASS shpMyCircle AS SHAPE  && 创建一个圆
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 99
    BackColor = RGB(255,0,0)  && 红色
ENDDEFINE

DEFINE CLASS shpMyEllipse AS SHAPE  && 创建一个椭圆
```

```
    Top = 35
    Left = 200
    Width = 100
    Height = 50
    Curvature = 99
    BackColor = RGB(0,128,0) && 绿色
ENDDEFINE

DEFINE CLASS shpMySquare AS SHAPE && 创建一个正方形
    Top = 10
    Left = 200
    Width = 100
    Height = 100
    Curvature = 0
    BackColor = RGB(0,0,255) && 蓝色
ENDDEFINE
```

应用于

命令按钮， OLE **容器控件**

请参阅

Click **事件**, Default **属性**

CANDIDATE（）函数

如果一个索引标识是候选索引标识，则返回“真”(.T.)；否则，返回“假”(.F.)。

语法

CANDIDATE([nIndexNumber] [, nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nIndexNumber

指定的索引标识编号，CANDIDATE（）返回其候选状态。当 *nIndexNumber* 从 1 递增到结构复合索引标识和独立复合索引标识总数时，CANDIDATE（）按下列顺序返回候选状态：

1. 首先返回结构复合索引（若存在）中每个标识的候选状态。按照标识在结构索引中的创建顺序返回它们的候选状态。
2. 然后返回任何打开的独立复合索引中每个标识的候选状态。按照标识在独立复合索引中的创建顺序返回它们的候选状态。

若省略 *nIndexNumber*，CANDIDATE（）检查主控索引标识是否为候选索引标识。若

没有主控索引标识，则 CANDIDATE（）返回“假”(.F.)。

nWorkArea

指定由 *nIndexNumber* 指定的索引标识所在的工作区。

cTableAlias

指定由 *nIndexNumber* 指定的索引标识所在的表的别名。

若省略 *nWorkArea* 和 *cTableAlias*，则 CANDIDATE（）检查当前选定工作区中的索引标识是否为候选索引标识。

说明

由于候选索引标识不包含 null 或重复值，因此可以作为主索引标识。

示例

以下示例打开了“testdata”数据库中的“customer”表。使用 FOR...ENDFOR 建立循环语句，在其中显示检查到的“customer”结构索引的每个索引标识的候选状态。显示每个结构索引的名称和它们的候选状态。

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

FOR nCount = 1 TO 254
   IF !EMPTY(TAG(nCount)) && 检查索引中标识
   ? TAG(nCount) && 显示标识名
   ? CANDIDATE(nCount) && 显示候选状态
   ELSE
      EXIT && 当不再发现标识时退出循环
   ENDIF
```

ENDFOR

请参阅

ALTER TABLE - SQL, CREATE TABLE - SQL, INDEX, PRIMARY()

CAPSLOCK（）函数

返回 CAPS LOCK **键的当前状态，或把** CAPS LOCK **键状态设置为“开”或“关”。**

语法

CAPSLOCK([lExpression])

返值类型

逻辑值

参数描述

lExpression

可包含 *lExpression* **来打开、关闭** CAPS LOCK **键。** CAPSLOCK(.T.) **打开** CAPS LOCK **键；而** CAPSLOCK(.F.) **关闭** CAPS LOCK **键。在执行** CAPSLOCK(.T.) **或** CAPSLOCK(.F.) **之前，根据** CAPS LOCK **的设置将返回一个逻辑值。**

说明

不带参数运行 CAPSLOCK（）时，若 CAPS LOCK 打开则返回“真”，若 CAPS LOCK 关闭则返回“假”。

示例

以下代码将 CAPSLOCK（）的状态存储到系统变量。= 命令执行 CAPSLOCK（）函数将 CAPS LOCK 设置为开启状态。= 命令执行 CAPSLOCK（）函数将 CAPS LOCK 恢复到它原来的状态。

```
glOldLock = CAPSLOCK（）    && 保存原始设置
= CAPSLOCK(.T.)   && 将 CAPS LOCK 打开

*** 执行任意数目的语句***

= CAPSLOCK(glOldLock) && 返回到原始设置

*** 或，将 CapsLock 切换到相反的值再返回 ***

= CAPSLOCK(!CAPSLOCK( ))
WAIT WINDOW
= CAPSLOCK(!CAPSLOCK( ))
WAIT WINDOW
= CAPSLOCK(glOldLock) && 返回到原始设置
```

请参阅

INSMODE( ), NUMLOCK( )

Caption 属性

指定在对象标题中显示的文本。设计和运行时可用。

语法

Object.Caption[= cText]

参数描述

cText

指定在对象标题中显示的文本。

说明

对象不同，标题的显示也不同：

- 对于表单，Caption 属性指定显示在表单标题栏中的文本。若将表单最小化，文本就显示在表单图标的下面。
- 对于页框对象中的页面，Caption 属性指定显示在每页选项卡上的文本。
- 对于控件，标题属性指定显示在控件上或控件旁的文本。
- 对于 Style 属性设置为 1（图形）的控件以及被最小化的表单，标题显示在图标下面。
- 对于 Active Documents，Caption 属性确定了当 Active Document 的帮助菜单

与容器的帮助菜单合并时在 Active Document 的帮助菜单项显示的文本。Active Document 的帮助菜单项中的文本后面是“Help”。例如，如果 Caption 属性设置为“My Application”，则 Active Document 的帮助菜单项显示为“My Application Help”。默认的 Caption 属性值为“ActiveDoc1”。如果 Caption 属性设置为空字符串，则帮助菜单项显示显示为“Microsoft Visual FoxPro”

当创建一个新表单或控件时，默认标题与默认 Name 属性的设置值相同。这个默认标题包括对象类名和一个整数，如 Command1、Combo1 或 Form1。

Name 属性指定如何在代码中引用对象，Caption 属性指定屏幕上用以标识控件的内容。这两个属性初始值相同，但以后可以独立设置。

若没有特别为控件设置 Width 属性值，则控件自动改变大小以容纳标题。

对于命令组和选项组对象，只有 BorderStyle 属性设置为单实线 1 时，才显示标题。

对于标签控件，设置 AutoSize 属性为“真”时，控件可自动调整大小以容纳标题。

Caption 属性的最大字符数是 256。

若要为一个控件分配一个访问键，在标题中将反斜杠和小于号 (\<) 放在访问键字符的前面。用户可以按 ALT 键和指定的字符，将焦点移动到该控件上。如果该控件是命令按钮、复选框或选项按钮，按 ALT 键和指定字符的作用与单击该控件相同。

只有当 KEYCOMP 设置为 WINDOWS（默认值）时，访问键才可用。

应用于

ActiveDoc 对象，复选框，命令按钮，表单，标头，标签，选项按钮，页面，_SCREEN，工具栏

请参阅

AutoSize 属性, BorderStyle 属性, Name 属性, Style 属性, Width 属性

CD 或 CHDIR 命令

将 Visual FoxPro 的默认目录更改为指定的目录。

语法

CD cPath | CHDIR cPath

参数描述

cPath

指定下列中的一个：

- 驱动器指示符
- 带目录的驱动器指示符
- 子目录
- 任何使用 MS-DOS 速记符号（\或 ..）的上述内容。注意：当使用 .. 符号更改到父目录时，必须在 CD（或 CHDIR）与两句点之间包含一个空格。

说明

使用 CD 或 CHDIR 可指定 Visual FoxPro 目录，Visual FoxPro 在目录中搜索文件。若 Visual FoxPro 在目录中找不到文件，则搜索已指定的 Visual FoxPro 路径。可以使用

SET PATH 指定 Visual FoxPro 路径。

如果创建一个文件，但没有指定存放在何处，则该文件存放在默认 Visual FoxPro 子目录下。

示例

以下示例使用 MKDIR 创建新的目录，名为“mytstdir”，然后使用 CHDIR 命令将工作目录更改到新目录。GETDIR() 用于显示目录结构，使用 RMDIR 移除新建的目录，再用 GETDIR() 显示目录结构

```
SET DEFAULT TO HOME() && 恢复 Visual FoxPro 目录
MKDIR mytstdir && 创建新目录

CHDIR mytstdir && 更改到新的目录
= GETDIR() && 显示“选取目录”对话框
SET DEFAULT TO HOME() && 恢复 Visual FoxPro 目录
RMDIR mytstdir && 移除新目录
= GETDIR() && 显示“选取目录”对话框
```

请参阅

DIRECTORY, DIRECTORY(), GETDIR(), HOME(), MD 或 MKDIR, RD 或 RMDIR, SET DEFAULT, SET PATH, SYS(5), SYS(2003)——当前目录, SYS(2004)——VisualFoxPro 起始目录

CDOW（）函数

从给定的日期表达式中返回星期值。

语法

CDOW(dExpression | tExpression)

返值类型

字符型

参数描述

dExpression

指定的日期，CDOW（）返回其星期值。

tExpression

指定的日期时间，CDOW( )返回其星期值。

说明

CDOW（）是字符型星期值函数，可从日期表达式中返回星期值。

示例

```
STORE {^1998-02-16} TO gdDate
CLEAR
? CDOW(gdDate) && 显示 Monday
```

请参阅

DAY()，DOW()，SET FDOW，SET FWEEK，SYS（）**函数概述**

CDX（）函数

根据指定的索引位置编号，返回打开的复合索引(.CDX)文件名称。

语法

CDX(nIndexNumber [, nWorkArea | cTableAlias])

返值类型

字符型

参数描述

nIndexNumber

下列规则适用于具有一个结构复合索引以及一个或多个复合索引的表：

nIndexNumber	说明
1	CDX（）返回结构索引文件名（一般与表名相同）。
2	CDX（）返回 USE 命令 INDEX 子句或 SET INDEX 命令指定的第一个复合索引文件名。
3	CDX（）返回第二个复合索引文件名。依次类推。
大于打开 CDX 文件的数目	CDX（）返回空字符串。

下列规则适用于没有结构复合索引而有一个或多个复合索引的表：

nIndexNumber	说明
1	CDX（）返回 USE 命令 INDEX 子句或 SET INDEX 命令指定的第一个复合索引文件名。
2	CDX（）返回第二个复合索引文件名。依次类推。
大于打开 CDX 文件的数目	CDX（）返回空字符串。

nWorkArea

指定表所在的工作区编号，CDX（）函数返回该工作区中打开表的复合索引文件的文件名。

cTableAlias

指定表的别名，CDX（）函数返回该表的复合索引文件的文件名。

若省略 *nWorkArea* 和 *cTableAlias*，则返回当前选定工作区中表的复合索引文件名。

说明

CDX（）函数等同于 MDX（）函数。

一个 .CDX（复合）索引由一个物理文件组成，文件中包含许多索引标识，每个标识都是对相关表中索引的引用。

.CDX 文件有两类：标准复合索引 (.CDX) 和结构 .CDX。标准复合索引 (.CDX) 可以与相关表不同名，并可保存在与相关表不同的目录下。一个表可以有多个复合索引文件。用 USE 命令的 INDEX 子句或 SET INDEX 命令可以打开一个复合索引。

结构 .CDX 文件必须与相关表同名并保存于同一目录下。一个表只能有一个结构索引文件。当用 USE 打开相关表时，结构 .CDX 文件自动打开和更新。

CDX（）忽略 USE 或 SET INDEX 中指定的任何 .IDX（FoxBASE+ 和 FoxPro 1.0 兼容索引）文件。

使用 TAG（）可以返回包含于 .CDX 文件中的单个标识名。使用 NDX（）可以返回打开的 .IDX 文件名。

当 SET FULLPATH 为 ON 时，CDX（）返回 .CDX 文件的路径和名称；当 SET FULLPATH 为 OFF 时，CDX（）返回 .CDX 文件所在的驱动器及其名称。

示例

以下示例打开了“testdata”数据库中的“customer”表。使用 FOR...ENDFOR 创建循环语句，在循环语句中显示结构索引名。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer    && 打开 customer 表
CLEAR
FOR nCount = 1 TO 254
   IF !EMPTY(TAG(nCount))  && 检查索引中标识
   ? CDX(nCount)    && 显示结构索引名称
   ELSE
      EXIT  && 当不再发现标识时，退出循环
   ENDIF
ENDFOR
```

请参阅

INDEX, MDX(), NDX() , SET FULLPATH, SET INDEX, SYS(14), SYS(21) ,SYS(22) , SYS(2021), TAG() , USE