Issue Date: FoxTalk December 1996

Visual FoxPro 5.0: Interactive Looping Constructs, the New FOR EACH Construct, and Support for the 21st Century

John V. Petersen

Run looping constructs interactively

Have you ever needed to test code that was contained in a looping construct? Specifically, testing a FOREACH loop and testing to see if a SCANENDSCAN will be processed are two common tasks. In version 3.0 and earlier, you need to write a program to test the code. Usually, this temporary program is called TEST, and more than likely there's already a program called TEST. Do you save the old program because it may be required later, or do you overwrite it? The end result is usually a set of programs called TEST1, TEST2, and so forth.

In version 5.0, things have become much more efficient. The following steps illustrate the process:

1. Type the following code in the Command Window, making sure to press Ctrl-Enter after each line:

```
FOR i = 1 TO 10
   ?i
NEXT i
```

2. Next, highlight the three lines of code.
3. With the three lines of code selected, position the mouse pointer over the selection and click the right mouse button.
4. Choose Execute Selection from the menu.

The numbers 1 to 10 will be echoed to the VFP Screen. If you find that the code can be used in a program, form, or class, simply highlight the code again, drag it to the appropriate destination, and drop the code.

New FOR EACH construct

Now that we've discussed how to test looping interactively in the Command Window, we can use the technique to illustrate a new looping construct, FOR EACH. For those of you who've worked with Visual Basic, this construct should look familiar because it works in the same manner. The FOR EACH command repeats a group of commands for each element in an array or collection. The following is the syntax for this new construct from the VFP online Help file:

```
FOR EACH Var IN Group
 Commands
 [EXIT]
 [LOOP]
ENDFOR | NEXT [Var]
```

- **Var:** a variable or array element used to iterate through the elements of Group.
- **Group:** a Visual FoxPro array, an OLE array, a Visual FoxPro collection, or an OLE collection.
- **Commands:** specifies the Visual FoxPro commands to be executed for each element in Group. Commands can include any number of commands.
- **EXIT:** transfers control from within the FOR EACH ... ENDFOR loop to the command immediately following ENDFOR. You can place EXIT anywhere between FOR EACH and ENDFOR.
- **LOOP:** returns control directly back to the FOR EACH clause without executing the statements between LOOP and ENDFOR. LOOP can be placed anywhere between FOR EACH and ENDFOR.

The FOR EACH construct provides a cleaner method of interacting with collections. Consider an example of looping through all the elements of an array. One such application would be in populating the new Grid ActiveX control that ships with Visual FoxPro. With the FOR...ENDFOR construct, you would need the following code to examine each element in an array:

```
FOR x = 1 TO ALEN(laArray,1)
   FOR y = 1 TO ALEN(laArray,2)
      ?laArray[x,y]
   NEXT y
NEXT x
```

With the new FOR EACH construct, the code is simplified as follows:

```
FOR EACH x IN laArray
   ?x
NEXT x
```

The FOR EACH construct will also work with collections such as the Forms collection of _SCREEN and the Controls Collection of a container.

Windows 95 Help
By now, you've probably noticed the new features present in Windows 95 style help. While Visual FoxPro 3.0 ran under Windows 95, version 5.0 is the first version to fully embrace all of the new features associated with Windows 95-style help. To see these new features, let's direct our attention to the Form Designer.

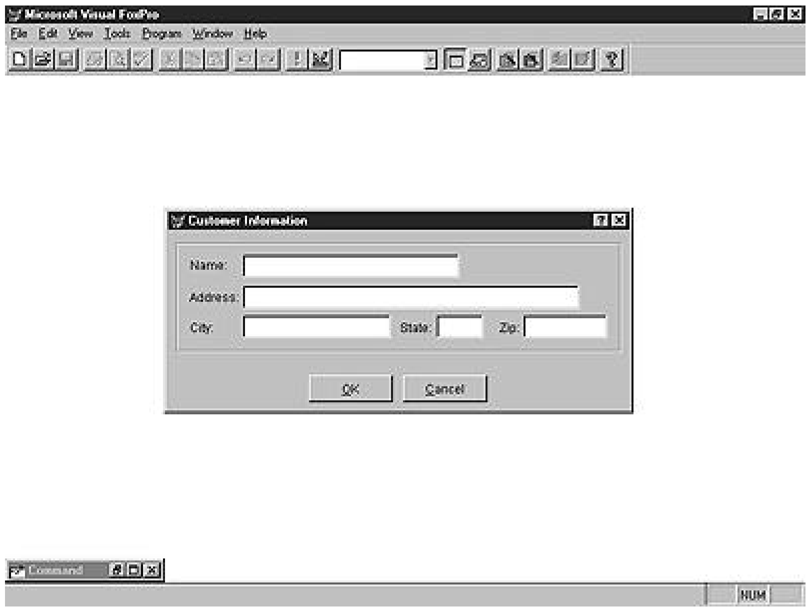
The following is a listing of new properties and methods available in Visual FoxPro 5.0 that enable users to create Windows 95 help-enabled applications:

- **WhatsThisHelpID Property** -- Specifies a help topic context ID to provide What's This Help for an object. Applies to: CheckBox, ComboBox, CommandButton, CommandGroup, Container Object, Control Object, EditBox, Form, Grid, Header, Image, Label, Line, ListBox, OLE Bound Control, OLE Container Control, OptionButton, OptionGroup, Shape, Spinner, TextBox, Timer, and ToolBar.
- **WhatsThisHelp Property** -- Specifies whether context-sensitive help uses What's This Help or the Windows Help file that's specified with Set Help. Applies to: Form.
- **WhatsThisButton Property** -- Specifies whether or not the What's This button appears in a form's title bar. Applies to: Form.
- **WhatsThisMode() Method** -- Displays the What's This Help question mark mouse pointer and enables Whats This Help mode. Applies to: Form.

Consider the sample form in Figure 1. Notice the question mark button in the upper right portion of the form. This is the What's This Button. For the button to be visible the following form properties must be set as follows:

- WhatsThisButton = .T.
- MinButton = .F.
- MaxButton = .F.
- BorderStyle > 0 (cannot be set to 0 - None)

***Figure 1:* Sample form.**

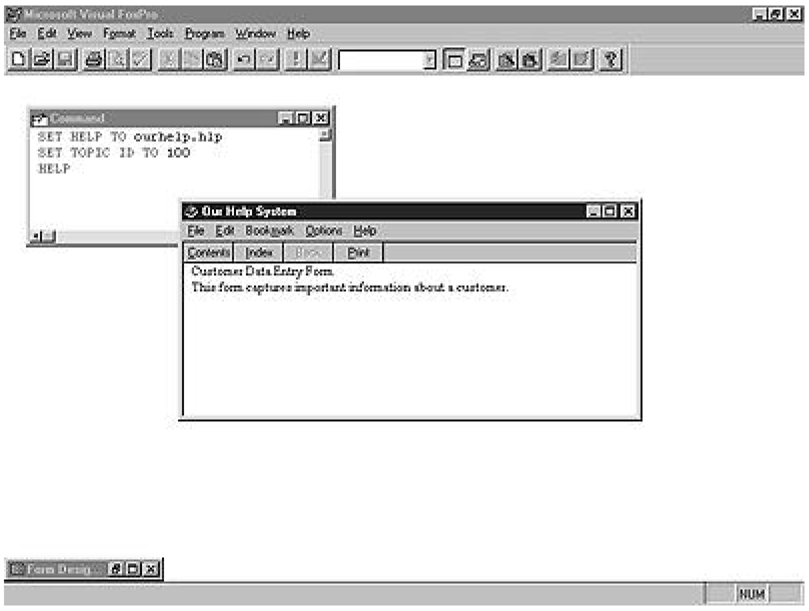


With these properties set accordingly, the form is ready to support the new Windows 95 Help features. The only remaining task at this point is to set some additional properties both for the form and its contained objects. Most baseclasses have a WhatsThisHelpID Property. The WhatsThisHelpID is very much like the HelpContextID Property that's present in both versions 3.0 and 5.0. Consider the following example.

Our form has an associated help file called OURHELP.HLP. The form has a HelpContextID Property of 100. Issuing the following code results in a view of the help system (Figure 2) we're all familiar with:

```
SET TOPIC ID TO 100
HELP
```

***Figure 2:* OURHELP.HLP a view of the help system .**

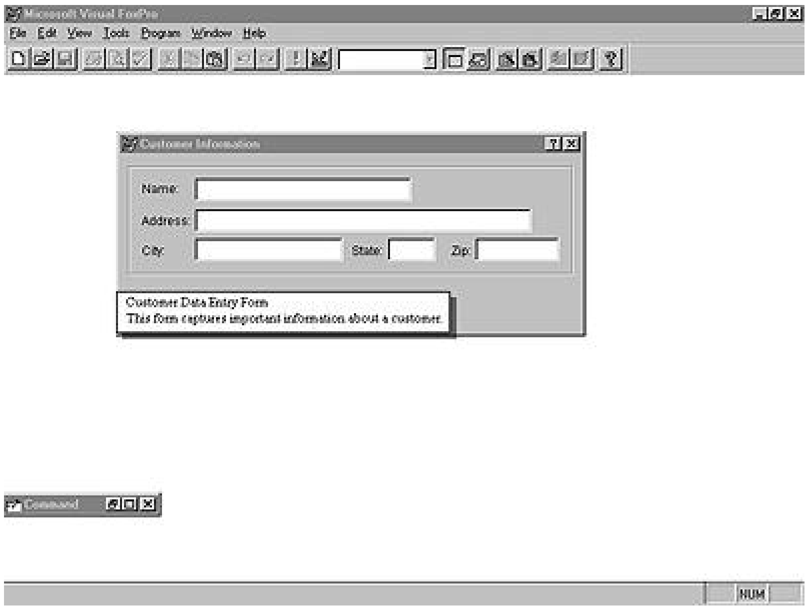

Microsoft Visual FoxPro
File Edit View Format Tools Program Window Help
Command
Our Help System
File Edit Bookmark Options Help
Contents Index Back Print
Form Desig...
NUM

```
SET HELP TO ourhelp.hlp
SET TOPIC ID TO 100
HELP
```

Customer Data Entry Form
This form captures important information about a customer.

Let's look at what the new help features give us. By setting the WhatsThisHelpID property of the form to 100, the What's This Help will use the same topic as presented in **Figure 2**. The only difference is the appearance. To invoke What's This Help for the Form, just click the What's This Help Button and then click the form. Your screen should look like **Figure 3**.

***Figure 3:* What's This Help Button form.**



Customer Data Entry Form
This form captures important information about a customer.

When you click the What's This Button, a question mark is attached to the mouse pointer. By clicking on any object with a WhatsThisHelpID Property assigned, a help window will pop up complete with details about the selected object. If a WhatsThisHelpID Property hasn't been assigned a value, the same pop-up window will appear stating that no help topic has been associated with the item.

These new Windows 95 Help features only work with graphical style help. If a .DBF help file is active while the What's This Help Button is enabled, it won't work. Finally, check out the new Help Workshop that ships with Visual FoxPro 5.0. It's a very nice utility for organizing help projects and makes it simple to take advantage of all the new features in Windows 95 Help Systems.

Turn-of-the-century support

Visual FoxPro finally has full support for the 21st century. In previous versions, consider the following:

```
SET CENTURY ON
ldDate = {01/01/20}
?ldDate && 01/01/1920
```

Assume when "20" was entered for the century, the user meant 2020, not 1920. In previous versions, you needed to make sure SET CENTURY was on so that the century portion of the year could be entered. This would tell FoxPro exactly which year is correct.

In version 5.0 of Visual FoxPro, SET CENTURY has two new optional clauses that remove ambiguity with a changing millenium.

Syntax

```
SET CENTURY ON | OFF | TO [nCentury [ROLLOVER nYear]]
```

Arguments

ON -- Specifies a four-digit year in a format that includes 10 characters (including date delimiters).
OFF -- (Default) Specifies a two-digit year in a format that includes eight characters and assumes the 20th century for date calculations.
TO nCentury -- A number from 1 to 99 that specifies the current century. When a date has a two-digit year, nCentury determines in which century the year occurs.
ROLLOVER nYear -- A number from 0 to 99 that specifies the year above which is the current century and below which is the next century.

Let's assume that any year with the last two digits in the range of 00 to 59 relates to the years 2000 through 2059. Years with the last two digits of 60 to 99 relate to the years 1960 through 1999. To accomplish this, let's take advantage of the new TO and ROLLOVER clause:

```
SET CENTURY TO 19 ROLLOVER 60
```

By making 19 the TO clause, we're telling Visual FoxPro that the current century is the 20th century. By making 60 the ROLLOVER Clause, we're telling Visual FoxPro to consider digits 6099 as part of the 20th century and digits 0059 as part of the 21st century. Consider the following code samples:

```
SET CENTURY TO 19 ROLLOVER 60
ldDate = {01/01/64}
?ldDate && 01/01/1964
ldDate = {01/01/20}
?ldDate && 01/01/2020
```