## Cool Tool
## Version icon: 3.0
## Platform icon: Windows

## Figures: NONE
## File for Companion Disk: 05COOL.ZIP

Better Access to Your Runtime Objects

Whil Hentzen

Every time you're done with the Debugger or Trace Window in Visual FoxPro, don't you feel like you should start a campfire in front of the cave using a couple of dry sticks, and then take your stone axe or bow and arrow and go hunt down dinner? Here's an alternative to the somewhat long-in-the-tooth tools until we see Visual FoxPro integrated in the Developer Studio IDE.

Don't get me wrong - I love the Debug Window in FoxPro 2.x. If you've ever seen me develop, you know that I keep the Debug window open all the time - you can't beat it for monitoring the status of your application, (and it's kind of a handy calculation besides.) But I keep it open in 3.0 primarily from force of habit - despite the extension from 16 to 32 expressions, Debug is plain inadequate for examining running forms and classes.

Let's look at a couple of reasons why. In earlier versions of FoxPro, you could enter either a memory variable, a field name, or an expression in the left pane of Debug, and see the evaluated value display in the right side. And that's all there was to it - you could easily determine the value of anything object in your application.

In Visual FoxPro, there are a great many more objects, and the description of those objects varies a great deal more. This, in fact, poses the first quandry - what DID you name the seventh text box on the second page of the smaller form? Sure, you can use AMEMBERS and ACLASS functions to access information about the objects in your application, and you can use the results of those functions to puzzle out the names, but jeesh! Who's got the time? Or the patience?

Second, should you decide you've got the time and the patience, do you have the typing skills to enter an expression like

```
_screen.activeform.pgfBase2.page4.obgbase2.option5.value
```

more than once or twice in your lifetime?

Finally, even if you have the patience and the skills, most objects have several dozen properties. The 32 allowable items in Debug don't take you very far.

A number of tools have been developed that take into account these shortcomings, and we'll discuss them over the next few months. For example, Markus Egger's Object Explorer provides a unique view of the object oriented paradigm, and you'll see it discussed in an upcoming column. This month, we're going to look at Dave Frankenbach's Object Inspector, a utility that displays all PEMs for a selected object during run-time. Dave is the author of the Object Oriented Programming volume in Pinnacle Publishing's The Pros Talk Visual FoxPro series and the Object Inspector, or Inspect for short, is the topic of one of the chapters in his book.

## 05HENTZ1.BMP
INSPECT allows you to view and change properties of a running form.

Inspect is more than a Debug window on steroids. Inspect's functionality includes the following abilities:

- Scroll through all properties, events, methods and contained objects of the selected object,
- Change the properties of the selected object from the Object Inspector's dialog,
- Call the events and methods of the selected object from the Object Inspector's dialog,
- Drill down through multiple levels of containers either in the same or another Inspect dialog, and
- Call up Visual FoxPro Help for detailed information on any entry.

Furthermore, the Object Inspector's own interface is customizable. You can resize the form, implement a right-click context menu to change font and colors, and set the Inspect options. Inspect is also persistent, that is, it remembers it's last size and position.

Let's take an example. On the Companion Disk, I've included a simple form that contains four command buttons and a label. The top two command buttons change the caption of the label, the middle button runs Inspect, and the Close button, obviously, closes the form. This will give us enough to play with.

## 05HENTZ2.BMP
The form TESTFORM.SCX has several objects that can be inspected with INSPECT.

After running the form, press the top two buttons to see how they affect the caption of the label below them. Then run INSPECT by pressing the Inspect command button. (You can run INSPECT in an application or from any form by attaching the code

```
ON KEY LABEL x = inspect( _screen.activeform.activecontrol )
```

to a hotkey like F12.) Now that INSPECT is running, you can view and modify the properties of the form as well as the events and methods of the form. You can also drill down into the objects of the form by selecting the Object tab, selecting the desired object, and right-clicking on the name. This will bring up a second copy of INSPECT that allows you to manipulate that object.

You can manipulate an object by selecting the name of the property and then pressing the = key or the Equal button on the INSPECT form. This will bring up a dialog that shows you the current value and allows you to enter a new value. A similar dialog allows you to execute a method, optionally with parameters. Simply select the method and press the ! button on INSPECT.

## 05HENTZ3.BMP
The Change dialog allows you to change the value of the currently selected property.

As you can see, INSPECT provides all of the functionality you wanted in the Debug window, plus a lot more. Inspecting and manipulating the environment is a new adventure for FoxPro developers, and you'll need all the help you can get to gain mastery of this skill. INSPECT will be a valuable tool in your arsenal to do so.

Where to Get the Object Inspector
Inspect is part of the source code that comes with the Dave's Object Oriented Programming with Visual FoxPro book. The .APP file is included on this month's Companion Disk. If your appetite is whetted to the point that you want the source code and more indepth coverage (the chapter in the book runs 45 pages), then you'll need to give Pinnacle a call for ordering information. (Contact information is on the inside front cover of FoxTalk.) Go ahead and play with INSPECT for a while, and see if it's not the most valuable programming aid for Visual FoxPro yet. You can also contact Dave Frankenbach on CompuServe at 72147,2635.

Whil Hentzen is editor of FoxTalk and president of Hentzenwerke Corp, a Milwaukee Wisconsin software development firm that specializes in strategic FoxPro-based business applications for Fortune 500 companies. 414.224.7654, CompuServe 70651,2270.

Book of the Month
This month's book has been suggested reading around the "FoxPro circuit" for years - but if you haven't grabbed it yet, you really ought to do so. With Visual FoxPro and other Microsoft visual applications becoming the standard development tools, it's time to realize that GUI programming is not the same as the mainframe-like character-based applications we've done in DOS.

Bruce Tognazzini has been developing human-machine interfaces for better than 30 years, and he's spent the last two decades as the Human Interface Evangelist at Apple Computer. His book, TOG on Interface, (Addison-Wesley Publishing, ISBN 0-201-60842-1) is a compilation of columns he wrote for Apple Direct, a magazine for Mac developers and some other miscellanous writings.

Don't let the "Macintosh" tie-in sway you from picking this baby up. Much of Tog's material has more to do with the human-machine interface independent of a specific platform. His writing is witty, wide-ranging and engaging. For example, he discusses the requirement of interfaces having multiple visual clues as to their functionality by introducing entropy and the resulting solution of redundancy: “Shannon referred to the process of losing information and gaining noise as entry. (He was going to call it ‘uncertainty’ but the mathematician Von Neumann counselled him, explaining that since no one understood what entropy means, no one would be able to challenge him.) Information theory states that information tends toward entropy... Shannon not only identified the entropy problem, he offered a solution: redundancy.”

Chapters cover topics such as Command Keys vs. The Mouse, User Testing on the Cheap, How to Make an Extra Couple Million in the 1990’s, the Cure for Trash Can Madness, Fitt’s Law: Why Pull-Down Menus Work Best, and the Use of Timed-Closure Modal Dialogs. Some ideas you’ll be able to use immediately (“I knew there was an answer to <name of issue currently distressing you>“) and others you’ll want to dwell on for a period of time.

Since it’s a series of columns, the material is emminently readable - much in bite-sized chunks. I didn’t even read it cover to cover; rather, I just opened the book at random and read through a chapter every time I had 15 minutes to kill. The book won’t change your life, but it will open your eyes in terms of the way you communicate to your users with your interfaces.