June, 1997

Advisor Answers

Visual FoxPro

Q: How can I print everything about a form in VFP? In FoxPro 2.x, it was easy to print the source program and get some idea of what was going on. But in VFP, we have a lot of forms and code is put in many places. It takes a long time to examine each method.

–Narit Phumdontree (via the Internet)

A: One of the tools that comes with Visual FoxPro makes it easy to look at all the code for a form, although the object-oriented nature of VFP means that it's not quite as simple as in FoxPro 2.x.

Start the Class Browser (included with the Professional edition of VFP). In the File Open dialog that appears, change the Files of Type dropdown to Form and choose the form you're interested in. When the Class Browser opens, choose the View Class Code button and a window opens up that contains code equivalent to your form. You can examine or print the code from that window.

A few warnings are in order. First, the generated code probably can't be executed. The method that generates the code takes a shortcut to handle container objects in the form. For example, if the form contains a grid, you might have code similar to this:

```
ADD OBJECT form1.grid1.column1.header1 AS header WITH ;
             Caption = "order_id", ;
             Name = "Header1"
```

However, VFP doesn't actually let you add nested objects with ADD OBJECT. So, this line would fail if you tried to run the code.

In addition, the output only includes code that's actually contained in the form. It doesn't look through the class hierarchy and include inherited code. To print out that code, you'd need to load the classes involved into the Browser and choose View Class Code.

Despite these limitations, the output can be very helpful in documenting and understanding the forms you create in VFP.

If you want to get code for a number of forms or classes, you can use the Class Browser programmatically. The code below generates and prints the code for a form called MyForm.SCX.

```
DO (_Browser) WITH "MyForm.scx",,,,1
cAllCode = _oBrowser.ExportClass()
cClassCode = SUBSTR(cAllCode,AT("*",cAllCode))
nHandle = FCREATE("output.txt")
FWRITE(nHandle,cClassCode)
FCLOSE(nHandle)
_oBrowser.Release()
TYPE output.txt TO PRINT
```

The system variable _Browser contains the path and file name of the Class Browser. When you start it, you can pass a number of parameters. The first is the name of the file to open. The fifth parameter (passed as 1 above) indicates that the Browser should be minimized.

The second line of code calls the Browser's ExportClass method, which is the one that actually generates and returns the code for the current class or form. When form code is generated, the Browser treats it like a class and automatically adds some lines that instantiate the class. The third line above strips that code out. Then, the next three lines store the code to a file (using low-level file functions). Finally, the Browser is released and the file is printed.

The same approach can be used to print a collection of forms and/or classes. Since a project is just a table, it's not hard to write code that goes through the project and sends code for every form or every class to a file. Here's the basic structure for a program that processes all the forms in a project:

```
* Open the output file
nHandle = FCREATE("output.txt")

USE MyProj.PJX ALIAS Proj
SCAN FOR Type="K"
   * Strip off trailing CHR(0)
   cName = LEFT(Name,LEN(TRIM(Name))-1)
   IF TYPE("_oBrowser")="U" OR ISNULL(_oBrowser)
      * Open the browser the first time.
      DO (_Browser) WITH cName,,,,1
   ELSE
      * Open the file in the Browser
      _oBrowser.OpenFile(cName)
   ENDIF

   * Now generate the code
   cAllCode = _oBrowser.ExportClass()
   cClassCode = SUBSTR(cAllCode,AT("*",cAllCode))
   FWRITE(nHandle,cClassCode)
ENDSCAN

* Clean up
_oBrowser.Release()
FCLOSE(nHandle)
USE IN Proj
```

An alternative approach to finding out what's in your forms is to use the Documenting Wizard, also a part of the Professional Edition of VFP. This tool is a successor to the FoxDoc documentation program which was included with earlier versions of FoxPro. It allows you to reformat your code as well as produce a number of reports. To get output from a form, the desired report is Source Code Listing.

Unfortunately, the wizard's output for a form is not very usable. There's a single line for each non-default property, but there's nothing that indicates which member of the form a particular property belongs to, in many cases. All method code is included, but again, there's no indication which object a particular method is associated with. In addition, like the Class Browser's output, the file created by the Documenting Wizard cannot be run.

On the whole, the Class Browser provides more useful source listings than the Documenting Wizard.

–Tamar