March, 1998

Advisor Answers

Visual FoxPro 5.0 and 3.0

Q: Can you give me an example of the use of ResetToDefault? I need to reset quite a few color properties for text boxes and combo boxes so they pick up the property values from the class.

–Jim Huwaldt (via CompuServe)

A: Every property and method of an object has a default value or behavior inherited from the class on which the object is based. Similarly, the properties and methods of a class have default behavior which is based on its parent class. When you specify a value for a property or add code to a method, you override that default. (Many methods also have some basic, built-in behaviors, which are not overridden by custom code. For example, the OpenTables method of the data environment opens all the tables in the DE unless you include the NODEFAULT statement in your custom code.)

Occasionally, having changed a property or method, you need to get rid of your changes and again allow the object to inherit. In the Form and Class Designers, you can right-click on a property or method and choose "Reset To Default". (See Ask Advisor in the October '95 issue for details on that approach.)

However, the original version of Visual FoxPro had no programmatic way to restore the default values of properties and methods.

Fortunately, Microsoft quickly understood our need to make such changes in code, and added the ResetToDefault method in VFP 3.0b. Every base class has this method. To use it, you pass the name of the property or method to be reset. For example, to reset the BackColor property of the current object, use:

```
THIS.ResetToDefault("BackColor")
```

The method works at both design-time and run-time. This means that you can actually change the behavior of a method at run-time, though needing to do so probably indicates a design problem with your application.

I think the most likely use of this method is in builders, whether they're custom builders to perform complex tasks or simple builders you write to solve immediate problems. For example, in your situation, say you want to reset the BackColor and ForeColor properties of all your text boxes and combo boxes, as well as the ItemBackColor and ItemForeColor property of the combos. You can write a fairly simple program to do the job, along these lines:

```
MODIFY FORM MyForm NOWAIT && Open the form designer
MOUSE AT 2,2 WINDOW (WONTOP()) && Get the mouse onto the form
WAIT TIMEOUT .1  && Give the mouse time to get there
oForm = SYS(1270)     && Get a reference to the form

* Now process controls
```

```
FOR EACH oControl in oForm.Controls
* in VFP3, use FOR with ControlCount
   DO CASE
   CASE UPPER(oControl.BaseClass)="TEXTBOX"
      oControl.ResetToDefault("BackColor")
      oControl.ResetToDefault("ForeColor")
   CASE UPPER(oControl.BaseClass)="COMBOBOX"
      oControl.ResetToDefault("BackColor")
      oControl.ResetToDefault("ForeColor")
      oControl.ResetToDefault("ItemBackColor")
      oControl.ResetToDefault("ItemForeColor")
   ENDCASE
ENDFOR

* Save the changed form
KEYBOARD "{ENTER}"
KEYBOARD "{CTRL+S}"
=INKEY()
=INKEY()

* Clean up
RELEASE WINDOW (WONTOP())
```

The sample code only processes text boxes and combos that are right on the form. To do this right, you actually need to drill down into any container objects on the form, such as pageframes and grids.

This example also demonstrates some of the cooler features of VFP including the ability to manipulate objects in the Form and Class Designers programmatically, the MOUSE command that lets you position the mouse and even click it where you want, SYS(1270) which gives you a reference to the object under the mouse, and VFP5's FOR EACH loop, which lets you process all the item in an array or collection without knowing how many there are.

–Tamar