Visual FoxPro
The Fox Hunt

By Whil Hentzen

The usual response of a FoxPro 2.x programmer when cracking open the box of Visual FoxPro and starting to create a form is unbridled joy. Unlike the 13 controls available up to now (and four of them were marginally useful graphics controls), VFP starts out with 19 controls, and further provides the ability to add more through third party OCXs.

However, in order to take full advantage of Visual FoxPro's object oriented features, you'll not want to use these controls to build your forms. Instead, you'll want to subclass each control, and use those subclassed versions as your starting point. This month, we're going to discuss how to do so. (If you're a little shaky on the concepts and benefits of subclassing, refer to the inaugural issue of IVF.)

There are 24 objects that can be subclassed - each of the 19 controls, plus a few specialized ones. We'll subclass the controls and discuss the others in a later article. In order to start, create a fresh directory that will contain your class libraries, similar to the directory that held all of your common code in earlier versions of FoxPro. I call mine COMMON30.

Next, issue the CREATE CLASS command, either from the Command Window or from the File, New, Class menu option. The New Class dialog appears. There are three objects in this dialog that we'll fill in. The Class Name is the name of the class that you are creating. It's a good idea to take care in naming your classes, because you'll use them frequently. Furthermore, you'll generally use them as starting points to create further subclasses, so a good naming convention will help you distinguish between your base classes and your subclasses.

The base classes that ship with VFP are named checkbox, textbox, listbox, and so on. The Microsoft-suggested naming convention suggests that classes based on textboxes begin with 'txt', listboxes begin with 'lst', labels begin with 'lbl', and so on. The complete list of conventions are listed in Help, under Contents, Technical Reference, Programming, Naming Conventions, Objects.

I name my classes like so: txtBase, lblBase, cmdBase, and so on. Then, when I subclass those classes, I append additional descriptors to the end of the class. For example, VFP's command button base class is called commandbutton. My command button base class (based on the VFP base class) is called cmdBase. Then, when I create a class, say, for Quit buttons, that is based on cmdBase, I'd name it cmdBaseDone. A class for a Search button would be called cmdBaseSearch. This way, it's easy to tell that the Search command button class is subclassed from the cmdBase class.

The next control to use in the New Class dialog allows you to select which Visual FoxPro base class to use as the base class for your new class.

The third control in the New Class dialog prompts you for a file to store your new class in. Class libraries are tables, just like everything else in VFP, but these tables have extensions of VCX (and VCT for the associated memo field.). How you organize your classes into various libraries is up to you. You'll generally want to keep similar classes in a single library, as opposed to creating a separate library for each class, but you don't want to overload a single library with hundreds of classes. I've found it helpful to have a half-dozen or so standard libraries, one for base controls, one for base forms, one for non-visual classes, and so on. Then I create another set of classes, based on my base classes, for a specific application as demands require.

How to name your class libraries? Again, a consistent naming standard will help you down the road. I've used the names BASECTRL, BASEFORM, BASENONV for my control, form, and non-visual base classes.

Once you have finished with the New Class dialog, the Class Designer appears, with the object that you are subclassing.

Change the object as desired. For example, several controls have white background colors by default. If you're taken with the grey forms that everyone seems to be using, you've found that placing a Visual FoxPro base class on one of these forms looks ugly - the background outline of the control is white. Thus, you're going to want to change the backcolor property of your base classes to be grey. You might also want to change the fontname and fontsize.

When you're finished, save the class. Don't get too worried about getting your base class definition just exactly perfect - after all, if you decide you don't like the result after you've created dozens of controls that reference your base class, you can simply change the class!

Continue these three steps for each class you want to create. Note that you don't have to remember the name of the class library each time you create a new class; you can use the three dot command button to bring forward a list of class libraries.

Once you’ve created your classes, how do you use them on a form? As with about everything in Visual FoxPro, there are a dozen ways to get started. The easiest way is to bring forward the Form Controls toolbar, click on the icon to the right of the arrow (the books icon), and select the Add menu option. You’ll be given a list of available class libraries and you can choose the one you want. The standard Form Controls toolbar will be replaced with a custom toolbar that contains icons for each class in the library. If you’ve created several classes based on the same base class, the icon will be reused. In order to tell them apart, you can position your mouse over an icon and see that the class name shows up as the tooltip. You can also specify a toolbar icon for a class by opening the class in the Class Designer, and then selecting the Class, Class Info menu option, and selecting a bitmap or icon file.

Experiment with creating classes, and then placing controls on forms that are instantiated from your new base classes. Then go back and change the class definition, and see how your forms change as well. Next month, we’ll discuss how to extend your classes, and control the use and access of your class’s properties and methods.