Use the Toolbox!

This tool does everything the Form Controls toolbar does, but better and more easily. Plus it gives you much more functionality.

Tamar E. Granor, Ph.D.

Until now, I've focused this column on the FoxPro language, covering commands and functions. In this issue, I want to take a look at one of my favorite recent tools, the Toolbox, introduced in VFP 8.

When FoxPro 2.x's Screen Builder morphed into the Form Designer in VFP 3, it included three ways to put controls on forms and classes.

The most basic and visible way to drop controls is the Form Controls toolbar. By default, it opens automatically when you open the Form Designer or Class Designer; to drop a control, you click on the control in the toolbar, and then click where you want to put it. This toolbar was so "in your face" that many people never looked any further.

The Project Manager offers a second approach. Switch to the Classes tab or expand the Class Libraries section in the All tab, then expand the relevant library and you can drag a control and drop it where you want it. Many VFP developers settled on this as their standard way of creating forms and classes.

There was a third choice, as well. The Class Browser is a separate tool actually written (by Ken Levy) in VFP code. It lets you open one or more class libraries (or even an entire project) and explore their contents. You can drop a control onto a form or class by clicking the class in the Class Browser, then dragging the icon in the upper left corner to the position where you want it. (In fact, that approach even works on live forms, so you can do the parlor trick of creating a blank form, dropping live controls onto it, and then using those controls.)

While all three techniques work, each of them has quirks and drawbacks, things that make it clumsy or inefficient or cases it doesn't handle. I won't catalog all of the issues with the old tools here, but I will list the main ones that bother me.

With the Form Controls toolbar, the biggest issues relate to how you get your own class libraries shown. The toolbar can show either the native VFP controls, all registered ActiveX controls, or the controls in a single class library. Although the toolbar does remember how you left it, when you close VFP and reopen it, any class libraries other than the one currently displayed are removed from the list.

You can register class libraries on the Controls page of the Tools | Options dialog to make them available in every VFP session, but there's no easy mechanism for setting up different lists of libraries for different projects.

Another problem is that, by default, all controls based on the same base class look the same, and you have to rely on their tooltips to figure out which is the one you want. In my experience, the order changes when you edit classes in the library, so you can't even rely on their position. (You can change the icon for a class using the Class Info dialog that's available from the Class Designer, but in my experience, very few people ever do so.)

All three tools share another big weakness; there's no easy way to create new forms based on a custom form class. For me, this is a fairly common activity when developing and the workarounds for it (registering a form class in the Tools | Options dialog, or using the CREATE FORM command with the AS clause) are clumsy; registering a form class in the Tools| Option dialog doesn't address the idea that you may need to work with several form classes.

Enter the Component Gallery

VFP 5 brought the Component Gallery, an alternate face for the Class Browser that offers a way to organize classes and much more into catalogs based on your own organizational structure rather than where they live or what project they belong to. It's an incredibly flexible tool, with tremendous extensibility built in.

However, it's also hard to understand and use and never gained much traction in the community.

The Toolbox—Easy to use and flexible

So, when the Toolbox was introduced in VFP 8, I was thrilled to find that it covers pretty much every use case I have. The Toolbox organizes "tools" (just about anything you want to include) into *categories*, which you can define. Each category contains *items*, which can be dropped onto forms and classes, or

into code windows, or into other kinds of windows (even windows belonging to other applications). While each type of item has default behavior for what happens when you drop it, you can specify different behavior. The Toolbox has five types of items built in, but you can define other item types fairly easily.

The Toolbox addresses the weaknesses of the other tools. Creating new forms from a form class is as simple as right-clicking on the class and choosing Create Form. Classes are listed in the Toolbox using a name you specify (by default, the name of the class). The Toolbox includes a filter mechanism, so you can organize its contents by project or any other way you want; a category can belong to multiple filters.

In other words, the Toolbox finally provides an easy, flexible way to put controls onto forms and classes. In fact, it offers much, much more.

Toolbox basics

To run the Toolbox, start it from the Tools menu or the standard toolbar, or run it with the command DO (_Toolbox). When it initially opens, it looks more or less like Figure 1. "More or less" because, first, I've changed from the default font and size, and second, because this is actually a filtered view of my complete Toolbox, showing only the items that are there by default.

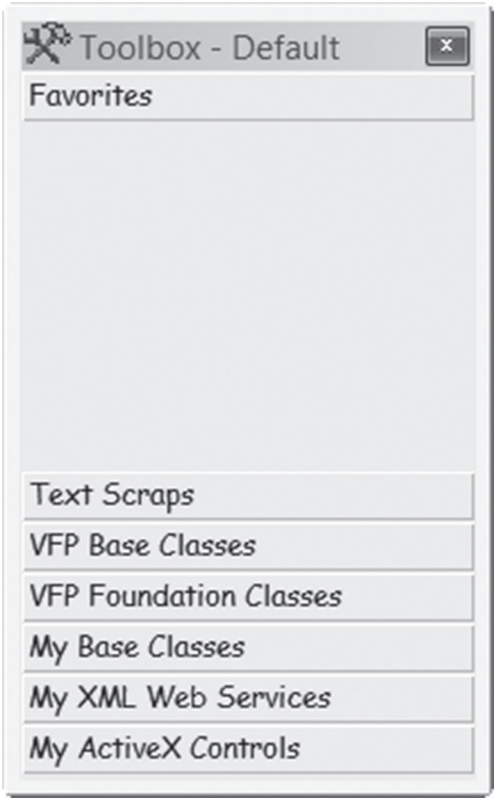

Figure 1. The basic Toolbox. Here, the font has been changed and a filter applied to show only the built-in items.

The bars you see are the built-in categories. Click on a category to open it and see the items within. In Figure 1, the Favorites category is open, but it's empty. In Figure 2, the category My Base Classes has been clicked. (Figure 1 was captured on a Vista machine, while Figure 2 comes from Windows XP.)

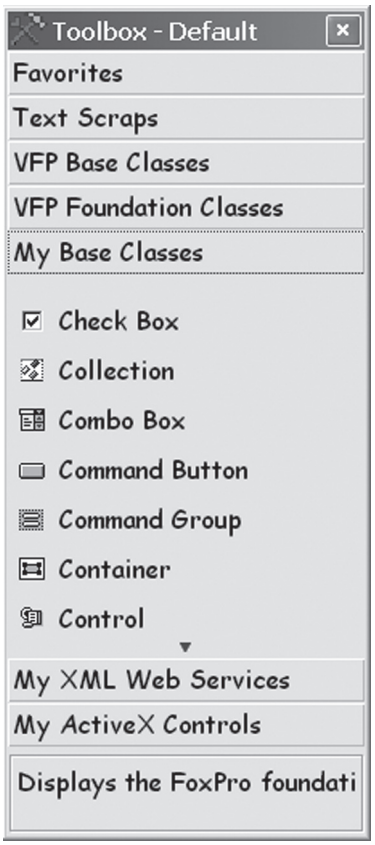

Figure 2. When you click a category, it opens to show the items within.

There are two ways to put a control, whether a native VFP control or a custom class, onto a form or another class. With the form or class open in the appropriate designer, you can either drag and drop from the Toolbox or double-click the control class. If you drag and drop, the control, of course, lands where you drop it. If you double-click, it goes in the upper-left corner.

One of the cool features of the Toolbox is that you can also drag and drop classes into code windows. When you do so, you generate a CreateObject() or NewObject() call. For example, dragging the Check Box class from the My Base Classes category into a code window results in this line of code (as a single line without the continuation):

```
_checkbox = NEWOBJECT("_checkbox", ;
                      "_BASE.VCX")
```

While you'd rarely want to instantiate a checkbox that way, this approach is very nice for classes that perform control tasks. For example, the VFP Foundation Classes category includes the Registry

class from the FoxPro Foundation classes. Drag that to a code window and you see (again, on a single line):

```
registry = NEWOBJECT("registry", ;
                     "REGISTRY.VCX")
```

Putting controls into containers

One of the tasks that's aggravating (if not actually difficult) with the older tools is putting controls inside a container, such as a page or grid. You have to click on the container, then right-click and choose Edit, and then drop the control. With the Toolbox, it's a piece of cake.

To add a control to a container, simply drag it and drop it onto the container. The container doesn't have to be selected. Figure 3 and Figure 4 show the process of dropping a listbox onto a page. Note that the page is not initially selected. Once you drop the control, the new control is selected.

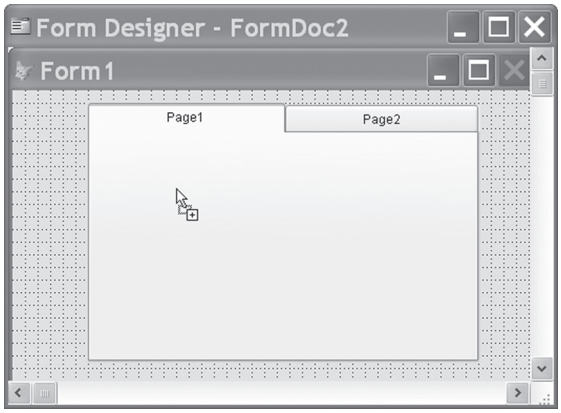

Figure 3. To add a control to a container like a page, just drag and drop onto the container. You don't have to select the container first.

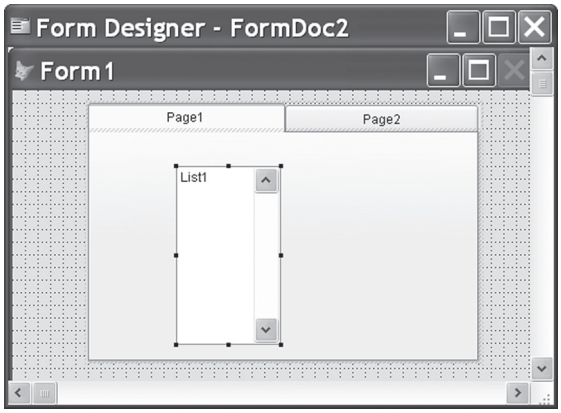

Figure 4. Once you drop a control onto a container, it becomes the selected object.

You can also add controls to containers using double-click. In that case, the appropriate container must be selected, but you don't have to right-click and choose Edit. The new control is placed in the upper-left corner of the container (though you can actually change the default). In this case, the newly added control is not selected. Instead, the container remains selection, so you can add a series of controls quickly.

While this technique is handy for pages and containers and so forth, where it really shines is with grids. Getting the right controls into grid columns is tedious using any of the older tools. The Toolbox makes it easy.

Once you add a grid to a form or class, select the grid, and you can start adding controls by double-clicking on them or dragging and dropping them. There are three distinct behaviors, depending on the situation.

If you double-click on a control in the Toolbox with a grid selected, you're prompted to add a column to the grid to hold that control, as in Figure 5. If you drag and drop a control to an empty space in the grid, you see the same behavior.

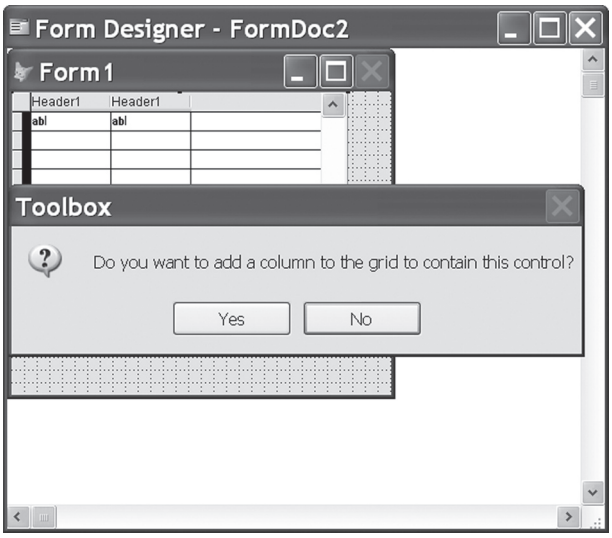

Figure 5. You can add a new column to a grid by double-clicking on the appropriate control in the Toolbox.

But if you drag and drop a control into an existing column, the behavior you see depends on what's already there. If the column contains a textbox named Text1, you can replace that control with the one you're dropping, as in Figure 6. (You control whether that message appears, through a Toolbox option.)

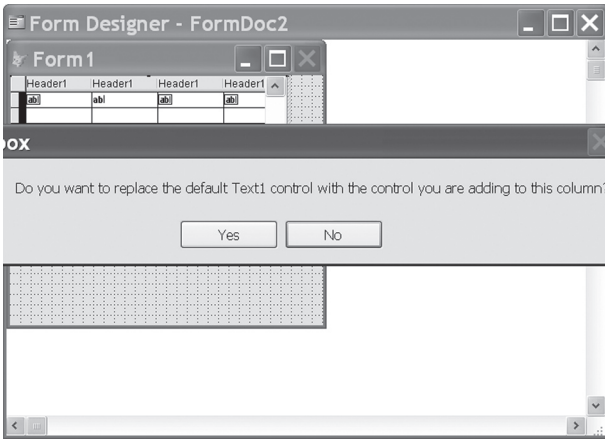

Figure 6. When you drop a control into a column containing only a textbox named Text1, you can remove the textbox at the same time.

If the column is empty or contains any other controls, the control you drop is added to the column. You can determine through one of the Toolbox's options whether the newly added control becomes the CurrentControl for the column.

Adding your own classes

All this functionality wouldn't be very useful if you could only apply it to the items that come in the Toolbox. But of course, you can add your own categories and your own classes. While you don't have to add categories to add classes, you probably will want to, so I'll show you how to do that first.

A category is just a named collection of items. By default, the Toolbox supports five types of categories, though any item can actually go into any type of category. For now, we'll look only at the General type of category, which is intended primarily for VFP objects.

To add a new category, right-click on the Toolbox and choose Add Category. The Add Category dialog (Figure 7) appears.

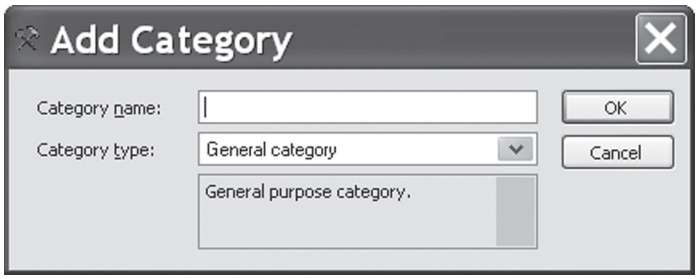

Figure 7. Use this dialog to add a new category. For VFP classes, choose the General category type.

Give the category a name. I tend to use an abbreviation for the relevant project or client, followed by a description of the type of items I'll put in this category. So, for example, for a project abbreviated HAP, I have categories HAP Base, HAP Forms, HAP Controls, and so forth. When you click OK, the new category is added at the bottom of the Toolbox. (You can control the display order of categories in the Customize Toolbox dialog.) Of course, the new category is empty.

To add classes, click on the Category name to open it. Then right-click and choose Add Class Library, as in Figure 8. In the Add Class Library dialog that appears (Figure 9), point to the class library whose classes you want to add.

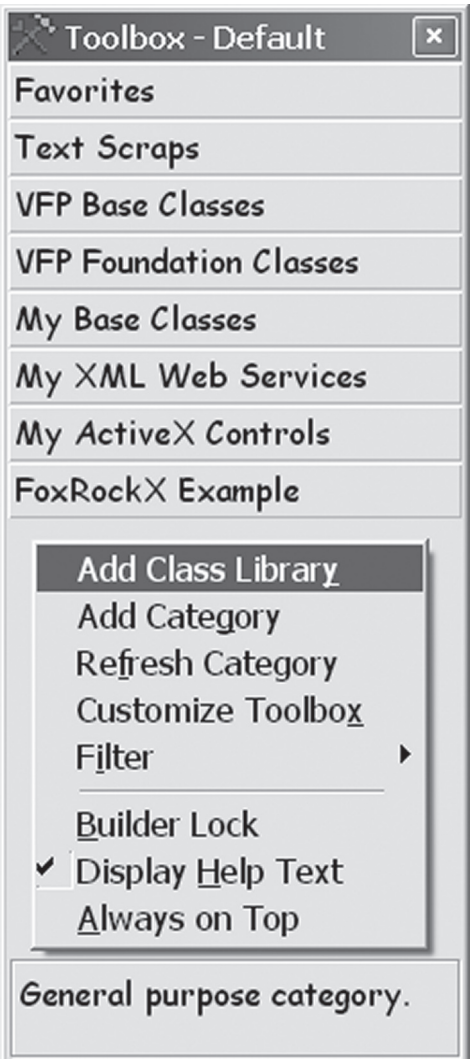

Figure 8. To add classes to the Toolbox, right-click in the appropriate category and choose Add Class Library.

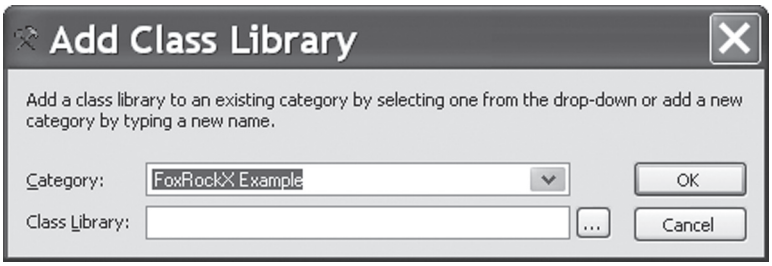

Figure 9. Use this dialog to specify the class library you want to add to the Toolbox.

Once you've added a class library, all of its classes are available in the specified category. You can actually control whether particular classes show in the Toolbox. You can hide an individual class by right-clicking on it, selecting Properties, and then checking the Inactive checkbox in the dia-

log that appears, as in Figure 10. It's a good idea to hide abstract classes that shouldn't be used. (The same dialog lets you change the name that appears for the class in the Toolbox. Modify the Item name to control what you see in the Toolbox.)

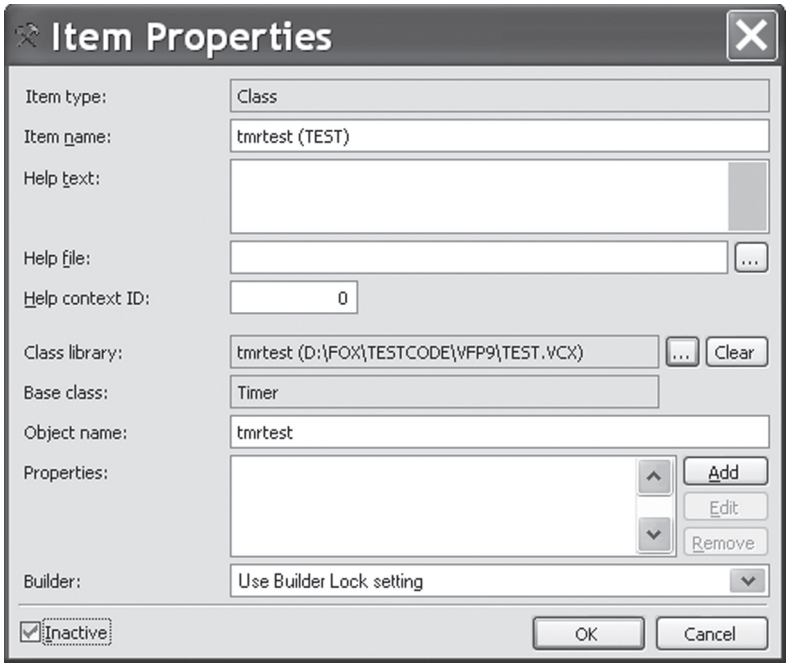

Figure 10. To hide a particular class, check the Inactive checkbox in the class's Item Properties dialog. To change the name displayed in the Toolbox, modify Item name.

Hiding classes one by one could get pretty tedious. Fortunately, there's another way. Right-click on the Toolbox and choose Customize Toolbox. In the Customize Toolbox dialog (Figure 11), choose your category, and you can then uncheck whichever classes you want to hide.

This dialog also offers the ability to change the name that appears for each class. Note that doing so doesn't change the class's name, just what you see for it in the Toolbox.

You can also re-order the classes within a category, either alphabetically or to an order you prefer, using the button bar that appears above the list of classes.

Making sure the tedious happens

There are various properties that need to be set just about every time you drop a control onto a form or class. You always should change the name of the control from the default. For some control types, you need to specify a caption, as well. While you're not likely to ship a form without providing the right caption, it's easy to leave the default name.

The Toolbox offers an easy way to make sure you specify the name for each control you drop. First, you can actually specify the "base" name used when you drag and drop. The Object name field in the Item Properties dialog (Figure 10) is used for that purpose. When you drop a control onto a form or class, a number is added to the object name to provide the control's name.

However, there's a better choice. You can actually get prompted for the name as part of adding the control. The Toolbox has the ability to set instance properties, that is, properties of the control you're adding, much the way a Builder does.

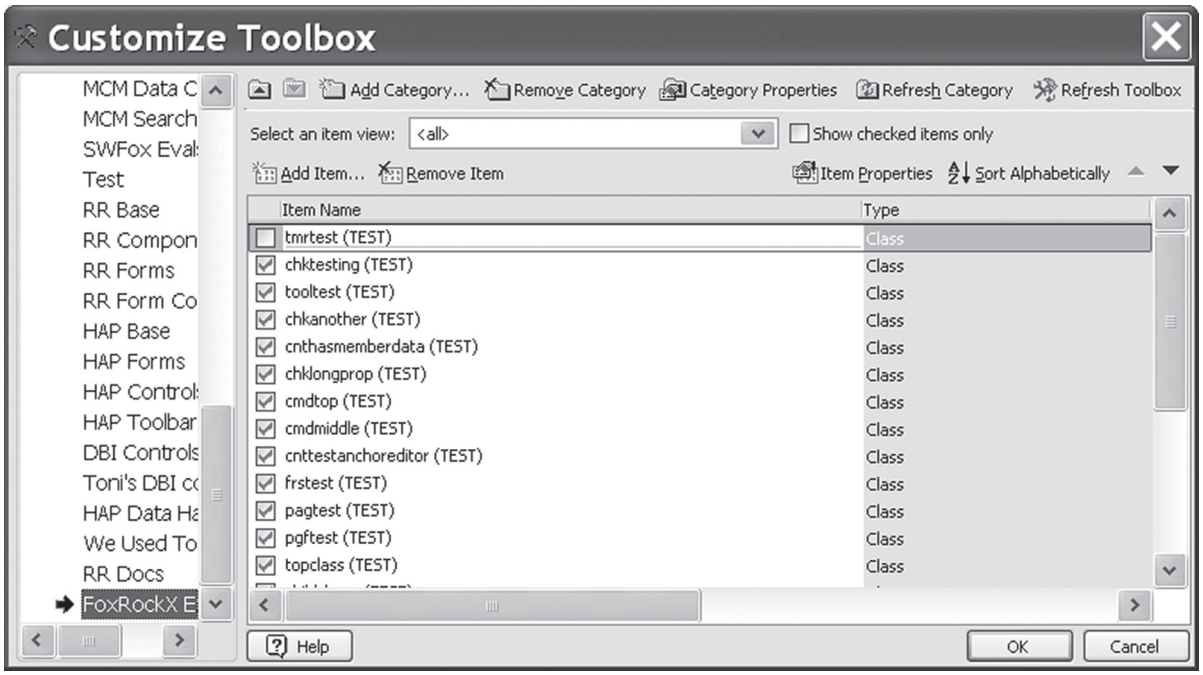

Figure 11. The Customize Toolbox dialog offers many ways to modify what you see in the Toolbox. Among them are hiding individual classes and changing the name that appears for a class.

To set it up, click the Add button in the Item Properties dialog. The Set Object Property dialog (Figure 12) appears. In the dropdown, choose the property you want to set, in this case, Name. In the textbox, specify the value to set that property to. What makes this ability so useful is that the "value" doesn't have to be a constant. In this case, it's an expression that calls the InputBox() function, so you can enter the name you want. Note that the parentheses around the expression are required in order to have the expression evaluated before assigning it:

```
(inputbox("Checkbox name", ;
          "Adding checkbox", "chk"))
```

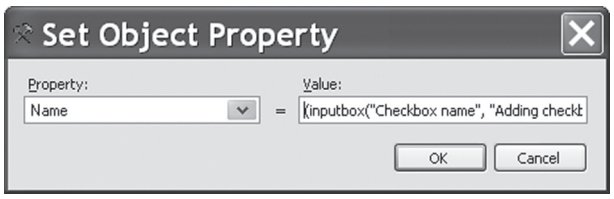

Figure 12. The Set Object Property dialog lets you specify properties to be modified as you add the control to a form or class.

When you drop this checkbox class onto a form or class, either by drag or drop or by double-clicking, you see the dialog in Figure 13. Type the name you want (or use right-arrow to unhighlight "chk" and then add the rest of the name) and the control is added with that name. No need to go to the Property Sheet to specify the control's name.

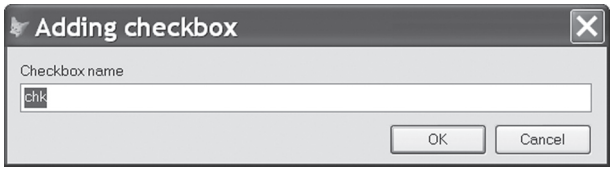

Figure 13. After you set things up as in Figure 12, this dialog appears each time you add the class to a form or class.

You can use the same trick to set the Caption as you add the control and, in fact, you can set many properties this way. However, don't use this ability as a substitute for subclassing. If you need a control that always has certain properties set to certain values, create a class with those properties set.

Creating forms

In my projects, I tend to have a few different form classes. Typically, there's a "base" form class, then subclasses of that one for dialogs, for data entry forms, and for reporting. (Sometimes, of course, additional subclasses are called for, as well.) The older tools don't provide any way of creating a new form based on a class. You have to work from the Command Window.

With the Toolbox, it's easy. Right-click on the form class and choose Create Form, as in Figure 14. The Form Designer opens with the new form based on the specified class.

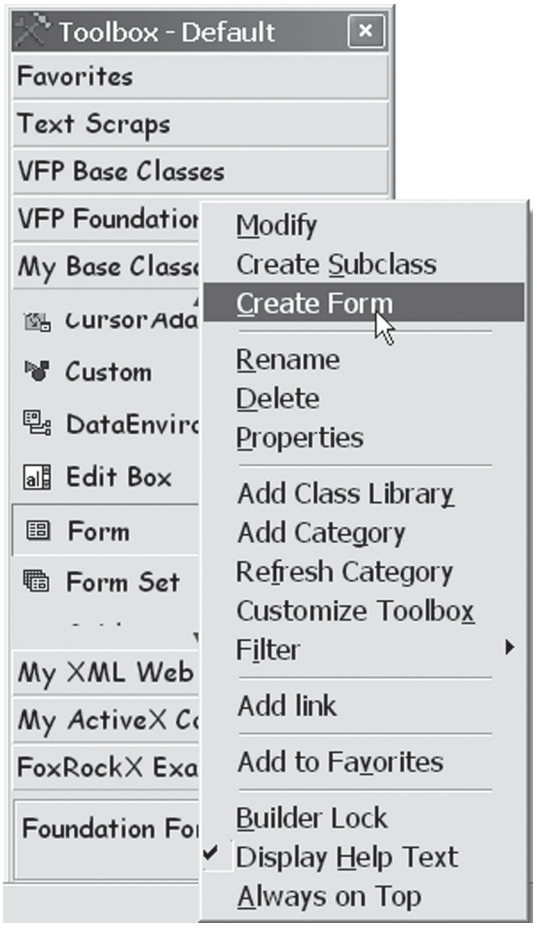

Figure 14. Creating a form based on a class is a simple operation in the Toolbox.

Filtering

One of my issues with the Form Controls toolbar is that all registered class libraries show up, making it hard to be sure you're using the right classes for a given project.

The Toolbox lets you filter what's shown, so that you can look at only the classes you should for a given project. A filter contains a subset of the categories defined.

To create a filter, right-click on the Toolbox and choose Customize Toolbox. Then click on Filters in the General section of the left-hand list. Then, click New Filter on the button bar at the top. In the Filter name textbox, specify the name for your filter and click Update to update it in the list below.

Then, check each category you want to include in this filter. Figure 15 shows the dialog after creating and naming a new filter and adding a category to it.

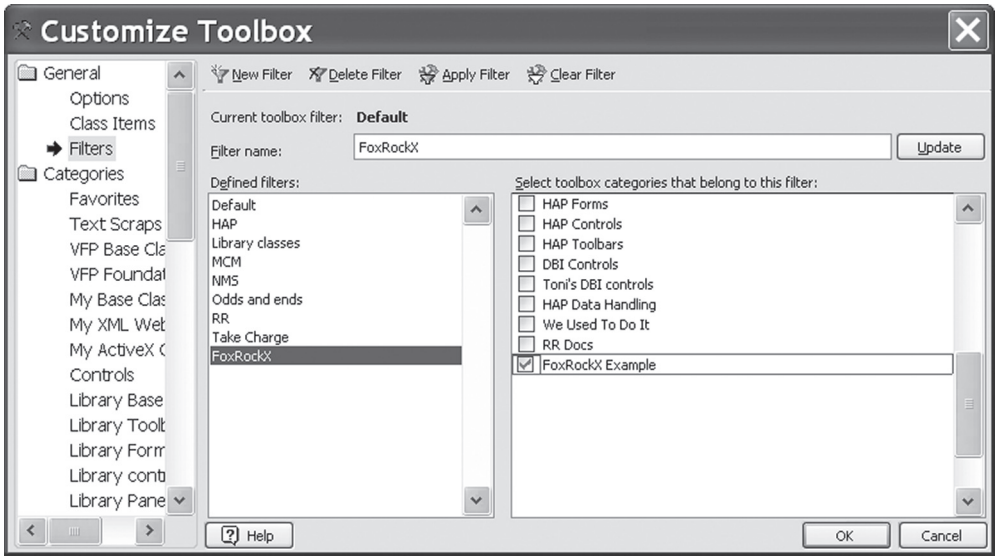

Figure 15. Filters let you show only a subset of the defined categories.

Any category can appear in as many filters as you want, and you can define many filters. I use one filter for the default Toolbox contents (important to me because I do demonstrations of the Toolbox), and one for each project I'm working on.

Once you've created filters, a Filter item appears in the Toolbox's context menu; it has a submenu listing all the defined filters. Choose one and the Toolbox shows only the categories included in that filter. Figure 16 shows the Toolbox after selecting the newly-defined FoxRockX filter.

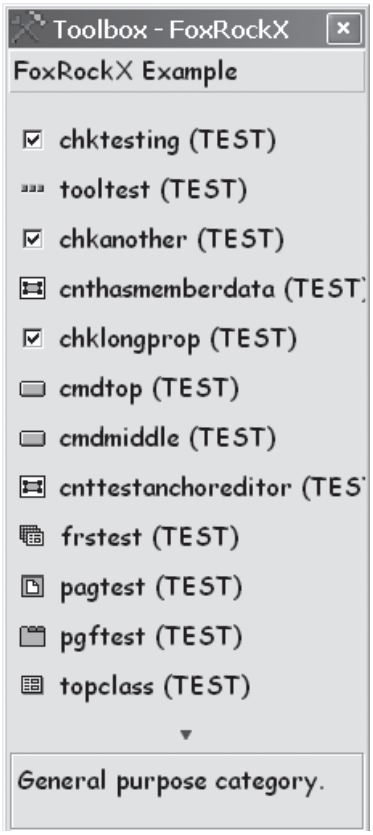

Figure 16. When you apply a filter, only the categories in that filter appear. This makes it easier to see what you have, and helps to use the right classes for each project.

But wait, there's more!

What I've covered in this article is just a little bit of what the Toolbox offers. Covering it all would take many, many more pages. If you've never worked with the Toolbox, I strongly urge you to open it up and spend some time just playing, to see what you can do and whether it's a comfortable tool for you.

There are a couple of additional resources for the Toolbox available. The chapter I wrote on the Toolbox for the book "What's New in VFP 8" is available at http://www.hentzenwerke.com/samplechapters/zsamplechapters.htm. A white paper by Beth Massi showing how to extend the Toolbox is at http://msdn.microsoft.com/en-us/library/ms965183.aspx.

Author Profile

Tamar E. Granor, Ph.D. is the owner of Tomorrow's Solutions, LLC. She has developed and enhanced numerous Visual FoxPro applications for businesses and other organizations. She currently focuses on working with other developers through consulting and subcontracting. Tamar is author or co-author of ten books including the award winning Hacker's Guide to Visual FoxPro, Microsoft Office Automation with Visual FoxPro and Taming Visual FoxPro's SQL. Her latest collaboration is Making Sense of Sedna and SP2, coming out this year. Her books are available from Hentzenwerke Publishing (www.hentzenwerke.com). Tamar is a Microsoft Support Most Valuable Professional. In 2007, Tamar received the Visual FoxPro Community Lifetime chievement Award. You can reach her at tamar@thegranors.com or through www.tomorrowssolutionsllc.com.