October, 2002

Editor's View

Why a Framework?

Frameworks have a significant learning curve. Are they worth the bother?

Recently, I've been studying one of the commercial frameworks for Visual FoxPro. Because there's a third-party book about this framework, as well as the Help file and some tutorials, I've been able to get some quick results. Nonetheless, when working with this product, I often feel like I'm programming with gloves on.

If all I want is "standard" stuff, the framework has tools to create it for me. But whenever I want to do anything a little bit differently, I can't just knock out the code as I usually would. Instead, I have to dig into the book or the Help file or the examples or even the class libraries that comprise the framework to figure out how this particular task is accomplished through the framework, rather than working around it.

This exercise has led me to think about the learning process in our business. I've spent many years learning VFP (and FoxPro and FoxBase+ before it). More than a dozen years of experience with the product means that I have large chunks of both syntax and semantics at my fingertips. When I want to do something straightforward, I can generally sit down at the keyboard and know exactly which commands to use. Obviously, more complex tasks require more forethought, but it's rare for me to encounter a problem in VFP for which I need to look up more than syntax (and, of course, with IntelliSense in VFP 7, I rarely even have to do that).

A few years, I became intrigued by Automation, and since then, have spent a fair amount of time writing code to automate the Microsoft Office products. While I've barely scratched the surface of the various object models, I have reached the point, at least with Word, that I can write a great deal of fundamental code without looking up more than syntax(where, again, IntelliSense saves a lot of time). My last few client projects have involved writing VBA code within the Office products, and my experience automating them from VFP has made this transition fairly easy (though I don't think I'll ever remember to put THEN at the end of an IF statement).

My experience with both VFP and the Office products means that I've been able to work with other object models comfortably. But somehow, using this framework is much more difficult. I know that, if I spend enough time with it, I'll get to the point of comfort, but the learning curve is still steep.

Why is learning a framework fundamentally different from learning the object model for an automation server? I think it's that a framework requires you to rethink the whole structure of your application, and even the process by which you create it.

When you automate a server, it's a side process in some sense. You're still writing VFP (or VBA) code; you're just using it to talk to another object and you can view the server's commands much like UDFs you've written yourself.

Working inside a framework, though, means getting your head around the way someone else has structured an application. It's much like coming in to maintain an existing application. You have to get a feel for the way the application is organized. At first, you only make tiny little changes, to see what repercussions they have. The longer you work with the application, the more confident you become in your knowledge of its structure and operation.

If it's this hard to do, why bother learning a framework? That one's easy. The framework contains code to handle all kinds of things I need in pretty much every application I write, things like conflict resolution, validation of data, and so forth. Never writing that code again seems worth some initial struggles.

If you've already implemented your own framework for your applications, and it does what you need, stick with it. If you haven't, you'd be well-advised to look at the various frameworks available. You'll find ads for most of them in FoxPro Advisor. Also, have a look at the framework comparison chart and related topics at http://fox.wikis.com/wc.dll?Wiki~FrameworkFeatureChart~VFP.

Still haven't upgraded?

By now, most of you know that VFP 8 (code-named Toledo) is in development. (See my August column for a list of some of the features coming in that version.) In fact, those attending DevCon will go home with a copy of the beta. However, it's still going to be some time before it hits the street.

In the meantime, if you haven't yet upgraded to VFP 7, what are you waiting for? As I said in the July, 2001 issue, IntelliSense alone is worth the price of the upgrade. In fact, I've heard of quite a few people who still have to deliver applications in VFP 6, but are doing their development in VFP 7, and then compiling in VFP 6. Besides IntelliSense, VFP 7 includes increased support for XML, the ability to create and consume Web Services, better tools for COM development, improvements in interface creation, and much more. (For a detailed list, check out the topic "What's New" in the online documentation at http://msdn.microsoft.com/vfoxpro/technical/documentation.asp.")

In addition to the benefits to you as a developer, upgrading tells Microsoft that you're still interested in VFP. Microsoft is, after all, a business, and the larger the audience for any product, the more resources it's likely to get.

So, don't wait for VFP 8. Upgrade to VFP 7 now, download the service pack (follow the link from http://msdn.microsoft.com/vfoxpro/), and find out what you're missing.