Column: War Stories

Figures:
File for Subscriber Downloads:

What If We Slice The Bread Before We Sell It?

Whil Hentzen

I'm slightly schizophrenic when it comes to new technology. On the one hand, I'm the first one to install a new beta (and the first one to complain if it doesn't work.) I love playing with the new stuff as much as the next guy (or maybe even more, considering how many of my friends steadfastly insist, "I don't do OS betas!")

For example, the new Windows ME has this "revert to a prior point" feature where you can roll back your system to a previous known configuration – it's truly awesome. We use it when we need to temporarily use an old piece of software. For example, HTML Help Workshop doesn't work well with Office 2000 files. What we do, then, is take a machine that has O2K, temporarily install Office 97, do our work, and then revert the machine back to before we had Office 97 installed – leaving plain old O2K on it. Very nifty.

On the other hand, as you've probably noticed, I'm often advocating reticence when it comes to jumping on the bandwagon when there's no visible benefit to doing so. For instance, over the last couple of years, we've seen more and more hype about new software architectures from all of the major players – and many wannabes – but that's really all it is – hype disguised as technical briefings.

The marketing talk sounds impressive, the demos look snazzy, but when you take one of the techies aside, you find out that there are plenty of missing pieces, that performance isn't at acceptable levels, that the documentation isn't complete, and that it can't be depended upon to produce reliable applications – those that have been deployed require armies of baby sitters attending to them around the clock.

And no one at any of these vendors has considered helping us out with the fundamental premise of engineering – being able to create reliable applications time and time again with repeatable success. As soon as something appears to work, the product development folks are off on another generation, leaving us in the real world to fend for ourselves.

In each of these cases, the benefit of jumping into a brand new technology/platform/architecture isn't sufficient to justify the gut-wrenching that takes place during implementation. Let's let it simmer for a while. Let's let it mature.

But technologies do mature, of course, and they become very useable. Remember DDE? Where the functionality was so rudimentary and feature-incomplete that it was unusable except for the most desperate of uses. DDE has become "Automation" (with a few acronym name changes along the way), it's a great tool now, and it's been worth the wait.

Some of you have been using Automation for years, and you're vacillating between suppressing a yawn and turning to the next page. Others haven't gotten there, and may be a little uncomfortable exploring this new technology. Until you get on the horse a few times, it's new and strange – and just doesn't feel "quite right." It's now time to approach Ol' Paint and try to jump on.

Recently, I've integrated automated email into applications for my customers (at least for customers who have a stable email system) and they love it. Of course, it means upgrading to Outlook 2000, but that was an easy decision when they see they can email 90% of their purchase orders and shipping confirmations as soon as the Save button is clicked. And you don't need to spend days and days reading books, poring through the docs, and asking questions on the various forums. You can get up and running in, oh, about 500 words – here ya go.

How to create a simple email merge program

Suppose you wanted to send an email confirmation to a customer during the order entry process. This confirmation would contain information about what they bought, when it shipped, how it was paid for (or was to be paid for in the future), and some other information.

Somewhere in your order entry screen, you could place a button called “Send Email” that would launch a “Send Email to Customer” form like that shown in Figure 1.

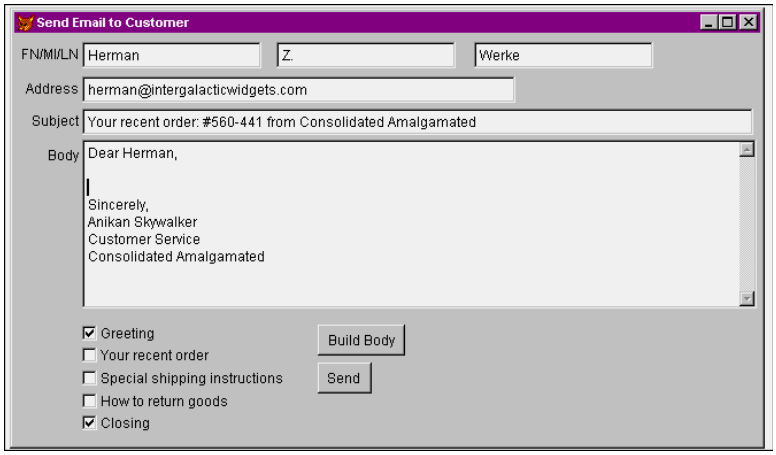

Figure 1. A sample email capture form.

The form would be called with a few parameters, such as the contact’s name, email address, and perhaps some specific order information (order number, list of products, shipping dates.) Wait, maybe you wouldn’t want to pass all this information as parameters – you’d end up with 50 or 60 of them. You could pass a parameter object, or perhaps just have this form inherit the data environment with the appropriately populated cursors. At any rate, it isn’t important, in this example, how you get the data to this form. All that’s important is that it’s available once you get here.

Once in this form, you’ll have the user’s name and email already filled in the appropriate text boxes (I make those editable just in case you want to make a last minute change.) The next thing to do is create a subject line and the body of the email.

Here’s where you can have some fun. In the init of the form, examine that data you have available, and customize the subject based on that data. For example, if there’s just an order about to ship, you could include the order number in the subject: “Your recent order- #560-441 from Consolidated Amalgamated – is has been scheduled to ship.” If, on the other hand, they only have a back-order, you could include that: “Your recent order - #560-441 from Consolidated Amalgamated – has been received and is on back order.”

Next, you can start to craft the body. Figure 1 shows a “Build Body” command button that takes information from the existing data and creates the body of the email. The series of five checkboxes to the left of the Build Body button act as logical flags for the construction of the body – so the user can pick and choose how they want to customize the body. If you wanted to get more sophisticated, you could include a pick list of standardized comments and values to be inserted in the body, much like secretaries did when creating documents from a series of boilerplate paragraphs.

The real magic, of course, happens when you hit the Send button. Here’s the code:

```
m.lcError = ""
if empty(thisform.hwtxtcEmail.value)
 m.lcError = "No Email address provided."
endif
if !("@" $ thisform.hwtxtcEmail.value)
 m.lcError = "No '@' in Email address."
endif
if !empty(m.lcError)
 messagebox(m.lcError, 0, oCust.cMsgBoxCaption)
 return
endif

if messagebox("Do you want to send this email?", ;
   292, oCust.cMsgBoxCaption) = 6
 oOutLookObject = CreateObject("Outlook.Application")
 olNameSpace = oOutlookObject.GetNameSpace("MAPI")
```

```
 oMailOut = oOutLookObject.CreateItem(olMailItem)
 oMailOut.To =  thisform.hwtxtcEmail.value
 oMailOut.Subject = thisform.hwtxtSubject.value
 oMailOut.body = thisform.hwedtBody.value
 oMailOut.send
 oOutlookObject = .null.
 release oOutlookOjbect
else
 wait window nowait "No email sent"
endif
thisform.close()
```

The first part of the code just helps protect you from yourself – making sure there's a valid email address before mailing. You could spruce it up by checking for a subject and/or a body as well.

The second part creates an object reference to Outlook, creates a mail item, sets some properties like the recipient, the subject, and so on, and then sends it. Finally, the object reference is terminated, and you're done. It really is this simple!

Naturally, you can get more sophisticated if you like – but this will get you started and running. If you need to set more properties, you can use the Object Browser in VB to open up Outlook and investigate the object model in more detail. If you keep your back issues, FoxTalk also ran several articles by John Petersen on the Outlook '97 and '98 object model, which is similar in many ways to Outlook 2000.

In the future, we'll have additional articles that cover the nuances of Automation with Outlook, as well as integrating VFP with other parts of the Office suite.

By the way, Outlook has had a fairly checkered history – with the initial release of Outlook 97 quickly superceded by Outlook '98. This article was written for and is running on Outlook 2000; you might try it on previous versions but your mileage may vary.

Yes, once a technology has had time to mature, you can finally build reliable applications and repeat that success in a quantifiable fashion. With Automation, we can have our (sliced) bread and eat it too.