Looking for the Killer App

Mark Hunter

While working on Martin Knopp’s review of Ice Tea Active Pages, I got to thinking about what “killer apps” have existed for SQLWindows developers, and which ones might surface in the future. Normally, when you use the term “killer app,” you’re talking about something that end users simply must have. In this case, I’m thinking about features or add-ons that are exciting to developers.

I guess SQLWindows 1.0 was the original killer app. Right from the start it had the outline interface, still unbeatable. And it had tight integration with databases, eliminating a lot of low-level function calls. (Table windows were still tough; the SalTbIPopulate function would come later.)

Later on, I think the Component Developer Kit was the killer app, but no one described it as such. The CDK enabled many software companies to create great SQLWindows add-ons, and even ordinary developers could take advantage of its power. Like the outline interface, it continues to be an indispensable part of SQLWindows even today.

ForeSite had the potential to be a killer app. Its integration module with Centura Web Developer means that you could embed calls to Centura Team Developer apps within Web pages, and mix those calls with calls to other data sources and other development tools. Congruity Corp. took over that integration module after the Centura/ForeSite divorce, and they still make success stories with it.

Pro Publishing teamed up with Pivato Consulting to manufacture and sell XSalCOM, because we think it’s a killer app, or close to it. By allowing CTD developers to expose their applications to the outside world, it eliminates the proprietary, closed nature of those applications, which is fatal to development tools in today’s interconnected world. With COM you can easily set up n-tier architectures without needing to run proprietary middleware (especially now that DCOM has been conquered, as described in our cover article this month). And turning your apps into COM automation servers also means you can access them from some Web servers, especially those using Microsoft Active Server Pages.

Eyes Wide Shut Mark Hunter, Eyes Wide Open Dian Schaffhauser, Eyes on the Numbers Shelley Doyle, Eye Candy Paul Gould, Eyes on the Milkbones Mocha
Distributed Centura . . .
Continued from page 1

SQL handles to get this information, his machine would be tied up for several minutes. If the part is out of stock in Orlando, he may also want to check somewhere else. You can see the problem. With DCOM, his client machine makes the call If fnGetPart( ORLANDO, CLSID, nPartNum, nReceivePrice, dtReceiveShipDate ) to the machine ORLANDO. “ORLANDO” sits next to the database server in the Orlando warehouse, runs the SQL statement in 24 seconds and returns TRUE, the price and ship date if it finds the part, FALSE if it doesn’t find the part. A single client can make calls to multiple application servers hitting different databases in different states or even countries, minimizing the need for large result sets to be passed over a WAN. The application server does all the heavy work and returns a result to the client.

If you really want to make yourself insane but indispensable to your employer to maintain the systems you write, three-tier architecture can even be expanded to n-tiers with application servers making calls to application servers making calls to database servers to the nth degree.

DCOM, almost . . .
Here comes the “almost.” To be able to truly qualify as DCOM, a client application should be able to make function calls remotely to multiple servers and switch between them programatically. Due to an unfortunate shortfall in how Centura creates an instance of its COM servers, this isn’t currently possible. In its Object.Create() call, Centura calls CoCreateInstance(), which only allows the passing of a CLSID and not the specifying of a remote server machine, whereas CoCreateInstanceEx() allows the remote server’s host machine to be named as in our previous example. This means your Centura client has no way of creating a server executable that exists on a remote machine. Centura is addressing this in the 2.0 version of CTD. In the meantime, there’s a workaround, which I’ll discuss a bit later.

The following procedures were developed by running our application servers on Windows NT Server and running our clients on NT Workstation and Windows 95.

There are five major pieces to making your Centura apps into DCOM components:

- Writing class functions and converting them into a COM server.
- Creating your client application.
- Registering your client and server applications.
- Enabling DCOM on your client and server machines.
- Setting user rights.

Creating the COM server
First, create a functional class that contains your functions and instance variables. I found it helpful to

Looking for the Killer App . . .
Continued from page 2

But many large-scale Web sites don’t use ASP because they don’t think it scales well. Now Ice Tea Active Pages provides a simple way to make calls to CTD apps from inside HTML pages, and it has excellent scalability. The CTD apps don’t need to be turned into COM servers, either, so changes to existing apps are small. Is ITAP a killer app? Well, it’s very good at what it does, as evidenced by the sites that are already using it. I guess it depends on how important it is to get your CTD business logic out to the Web.

If you simply want to provide data to a Web server, there are many products to choose from. Java can serve up database rows to Web pages. Cold Fusion and Net Dynamics are pioneer data publishing products. net.db is a simple, effective solution, too. But most of these products generate the HTML pages themselves. There’s always a struggle by developers to get just the look they want, and each new version of such tools features “better” HTML generation that still doesn’t quite satisfy. I think it’s kind of hopeless to expect data publishing tools to progress as quickly as Web page design does.

If you need business logic instead of just data publishing, you can make your CTD apps talk to a Web server using ForeSite, using XSalCOM and ASP, using Centura Web Developer without ForeSite, and now using Ice Tea Active Pages. Which one is the right choice for which requirement? It’s not easy to tell. Centura Pro will simply keep supplying as much information as possible so that your choice is an informed one. All of these tools must deal with the current CTD runtime, which isn’t multithreading. When CTD 2.0 arrives next year, the leaner runtime will make it practical to run many CTD application servers on one machine, and all these tools will take advantage of that.

What do you think? Did I overlook your favorite “killer app” in the SQLWindows world? What would a future killer app look like to you? Do you need to have Web servers making calls to functions in your CTD apps? If you do, what approach are you going to take? Send me e-mail at huntersoftware@netscape.net and let me know.

CP