THE FUTURE OF JAVA APPLICATIONS

James W. Cooper

When Java was first introduced, we treated it as a language for building much nicer web pages. We assumed it would be well supported in web browsers and that the best way to make really elegant pages would be using Java. However, poor planning and powerful competition led to an arena where no major browser supports Java very well and the idea of the nice features of Java 2's Swing user interface just haven't caught on in the browser world.

Java on Your Server

Meanwhile, we had completely overlooked the client-server implications of Java. Somewhere in the interim, Sun introduced Java RMI, and again it looked like Java on the web browser client could communicate easily with servers using a simple coding protocol. They can communicate quite easily, and if your applet can use RMI, it is simply terrific. However, since firewalls (and the religious views of corporate security types) can extinguish this flame, we find that RMI also is limited in acceptance. Then there was this great idea called CORBA that was supposed to be the panacea to object communication. Not only could the objects be in different languages and platforms, they could be on different planets, and the magic of IDLs would make this all work out. It does, in a homogeneous environment, but in the messy real world, it's just too hard to understand, let alone implement.

Finally the JavaServer Pages flag was run up the pole. Lots of us have saluted, and while we're still rigidly at attention, it also hasn't been all we hoped. One major problem is that the great unwashed of Web hosting services don't understand or support JSPs, and only a few even support Microsoft's simpler ASPs. A simple scan through the web hosting services you can turn up by searching, shows almost no server side Java support at all. "We have Java 1.02 installed, but we do not offer Java servlets at this time." Wonder why?

Now there are plenty of services that do support Java, but they are really aimed at the absolute high end of services. These are not the $29.95 a month variety of web hosting services, but the thousands of dollars a month and we'll bill your for our design-time services. Great for big companies, but not much use to the small consumer of web services.

I use JSPs all the time at the office and have a server running under my desk that serves JSPs on IIS. Anyone can do this. Why hasn't it caught on? Well, people feel you have to install something pretty big and expensive like Websphere or BEA or IPlanet to support all this stuff as well as EJBs, which no one in this space understands at all. EJBs seem to exist in a land of recursive acronyms and impenetrable jargon that keep a lot of services away from them. ISPs that cater to small business and non-profits have no incentive to go down this path. Thus, in an odd way, EJBs may be inhibiting the growth of Java at the low end.

Actually, JSPs are no longer much trouble to get going. Sun has created a JSP 1.1 server implementation and donated it to the Apache project as Tomcat 3.1. There is even a DLL that makes JSPs run automatically under IIS, the previous lone holdout. Of course, this DLL has to be installed by an administrator, not a remote user.

If you run a small home business and have a hosted web site, it is thus pretty unlikely that you are using Java on your web site. RMI is too limited, EJBs are too daunting, JSPs are unsupported and CORBA is just too opaque. So what is left?

One thing to keep your eye on is SOAP (Simple Object Application Protocol). It hasn't risen above many people's event horizon yet, but it is an HTTP-only protocol (meaning no firewall problems) that communicates to the (Java) server using objects encoded in XML. If this can be made simple enough to get buried in low maintenance servers like IIS or Apache, this could be the killer server app of the next season. Even though SOAP was more or less hatched by Microsoft (the Evil Empire!) this is a really terrific (cross platform!) technology. It can even be language independent and server independent. And it's from Microsoft? G'waan! Not only that, but other big players like IBM and Sun are working hard to adopt it as well. I'll write more after I get some examples working, but look at the IBM Alphaworks site (alphaworks.ibm.com) for a complete Java SOAP implementation, and the Microsoft site for an unusually lucid explanation of it.

Java on Your Client

The real question I have is why you don't see much in the way of Java client applications. Here it is, the great OO language of the millennium. It's really hard to write poorly constructed code in Java, and it has most of the facilities of other popular client development languages, like Visual Basic. And it's miles easier to use than Visual C++ or Borland C++. Why no apps?

Possibly the reason is inertia and the innate FUD of the untried. Also, Java is still slooower, if only on startup. When that's fixed by embedding more of Java in the OS, we'll see more and better Java applications on the desktop.

The single most important announcement I heard at JavaOne was Apple CEO Steve Job's announcement, right there on the stage with Sun's Scott McNealy, that Apple's forthcoming Mac OS-X would have the "best Java on the planet." If you've been following this, you know that Java on the Mac has always been a poor undeveloped relation. It was back level, buggy and poorly supported in a sort of game of chicken or hot potato between Apple and Sun. Now Steve Jobs announces that their offices are right across the street from Sun's and that they will be working together to get this planet-wide best Java onto the Mac. If they do it, this could be the start of the avalanche.

Imagine this. You can write a great-looking Java program that runs identically on the PC, the Mac and your friendly Linux box. What a relief! This could be the start of a great sea change in software development. Never mind the Windows hegemony. With tools from Sun and Apple, we can write a great application that we can deploy anywhere! What a unique experience!

What will we need to get there that we don't have yet? Well forget per-copy runtime licensing of embedded program tools. I'll pay $200-500 or more for a tool if I can use it without runtime licensing. I certainly won't pay $700 and then $70 per copy when I want to sell the app for under $100. Another thing still needed are really good development environments that are as simple and transparent as Visual Basic is. Even though Enprise/Borland and Webgain/Visual Café have been trying hard, these tools are still too slow and hard to use to compete head to head with VB.

It could be that this problem of building the quick visual app can never be solved in Java because of the layout manager problem. You don't want a lot of generated code that says, "for goodness

sake don't change any of this!" stuffed inside the program you thought you were writing. A better approach could be to store the entire look and feel in XML and have a jar file that interprets it and constructs your visual layout. This is pretty much what VB does anyway with its Form file format.

Another thing we're going to need is a more or less free database we can access via JDBC and ship with our desktop applications. After all, many desktop programs use databases to store whatever they've got, so a good database is simply required. I have great hopes for Cloudscape, Pointbase and MySQL in this space, if they can only develop licensing models suitable for consumer apps, low flat fee and unlimited distribution of a version that allows less than 5 connections or users. When we have that we'll be in great shape.

You may think that this is really more kvetching than positive predictions, but I believe strongly that Java can and will be an important desktop player. Don't worry too much about speed, since the recent Consumer Reports suggests that your minimum home desktop system should be an 800MHz Pentium III. I'm embarking on a project this fall to convert a major consumer application to Java, and will be reporting back to you on what I find: the peaks and the valleys. We are really poised to take off on a terrific new Java adventure here, and I hope you'll come along for the ride.