Craig S. Mullins

August 1998

From IDUG Solutions Journal...

Java: Hype or Hope?

By Craig S. Mullins

Everybody has heard about Java and how it is going transform the world of IT — unless you've been living under a rock. But just because you've heard about it doesn't mean you understand it. And even if you know a bit about it, there is always more to discover. Let's face it, there's a lot of hype out there regarding anything that concerns the Internet. Is it all hype, or is there some hope for a brighter multi-platform world? Let's examine what Java means to the world of DB2.

What is Java?
First and foremost, Java is an object-oriented programming language. Developed by Sun Microsystems, Java was modeled after, and most closely resembles, C++. But it requires a smaller footprint and eliminates some of the more complex features of C and C++ (e.g. pointer management).

Java enables animation for and interaction with the World Wide Web (WWW). Although web interaction is

Java's most touted feature, it is a fully functional programming language that can be used for developing general purpose programs (independent from the web).

Using HTML, developers can run Java programs, called applets, over the web. But Java is a completely different language than HTML, and it does not replace HTML. Java applets are automatically downloaded and executed by users as they surf the web. The Java applet is run by the web browser.

What makes Java special is that it was designed to be multi-platform. In theory, regardless of the machine and operating system you are running, the Java program should be able to run. Many possible benefits accrue because Java enables developers to write an application once and then distribute it to be run on any platform. Benefits may include reduced development and maintenance costs, lower systems management costs, and more flexible hardware and software configurations.

So, to summarize, the major qualities of Java are:

- its similarity to other popular languages
- its ability to enable web interaction
- its ability to enable executable web content
- its ability to run on multiple platforms

Java Database Connectivity (JDBC)
JDBC is an API that enables Java to access relational

databases. Similar to ODBC, JDBC consists of a set of classes and interfaces that can be used to access relational data. Anyone familiar with application programming and ODBC (or any call-level interface) can get up and running with JDBC quickly.

Java, IBM, and DB2

Java for OS/390 is IBM's mainframe Java development environment. Using Java, developers can build web-based and general purpose business applications on the mainframe. Likewise, IBM provides Java support for its DB2 Universal Database (UDB) platforms, AIX and OS/2, (as do other operating systems suppliers such as Microsoft with Windows NT/95, Hewlett-Packard with HP-UX, and of course Sun with Solaris). With DB2 UDB developers also can code user-defined functions and stored procedures that run on the server.

In combination with the Java development environment, IBM also provides JDBC application support for Version 5 of DB2 for OS/390 and DB2 UDB. Using Java and JDBC, users can create applications that might otherwise be written in COBOL, PL/I, C, or C++.

The intended benefit of JDBC is to provide vendor-independent connections to relational databases from Java programs. IBM delivers this (as do most other RDBMS vendors), but from an industry-wide perspective, JDBC is still immature. This, coupled with other problems such as lack of support from Microsoft and slow performance, will probably limit JDBC acceptance and growth through the middle of 1999. But JDBC will most likely succeed and achieve wide-spread acceptance by the year 2000.

Hype or Hope?

So, given all of the above background, should we be wary of Java hype or full of Java hope? I think the best answer is "yes, both!"

Hype

Java is very simply the most hyped phenomenon since the Internet and the WWW. This does not mean it is without merit; just that its merits are oversold. There is simply no chance that 100% compatibility of Java across platforms will ever become reality. Differences will exist from platform to platform and system to system. Witness the recent events where HP delivered a Java Virtual Machines (JavaVM) for embedded systems that closely conforms to, but is independent from, the Sun standard. A JavaVM is required for each platform on which Java is to run. Key characteristics of the Java language require the presence of a substantial runtime environment, the JavaVM.

Differences will continue to occur because each vendor's offerings are different and require differences to optimize their platforms. Sun has sued Microsoft to make it remove the Java logo from its web site because its Java implementation does not conform to Sun's standard. Furthermore, vendors want differences; users are the only ones who want conformity. Once again, witness Microsoft's efforts to promote ActiveX as an alternative to Java for interactive web applications. Without differences, hardware and software becomes a commodity and the only ones who win are the users. That would be nice, but it is not likely to occur.

Furthermore, there are other reasons why Java is not all it is cracked up to be. Java is an interpreted language. This makes it slower than compiled languages, so performance will be an issue. Many vendors (including IBM) are releasing just-in-time (JIT) compilers to react to this deficiency, but even a JIT compiler will be slower than a truly compiled and optimized language. Java is most useful for applications that require a high productivity development environment and high portability for the resultant programs. If instead the application requires maximum performance, platform specific processing, or the use of robust compiler technology, then C++ (or another time-tested 3GL) will usually be preferable to Java.

Finally, there is a lack of infrastructure and tools available for the Java environment. Java is only just three years old and organizations are not yet fully competent on how to implement, administer, secure, and maintain the Java environment. And vendors have not yet delivered robust tools needed to provide a robust application development lifecycle (testing, change management, debugging, implementation, etc.).

Hope
But, none of these issues will sound the death knell for Java. It is truly here to stay because of its many benefits. Although 100% platform independence is unlikely, Java will get us closer than we ever have gotten before. At least there is a standard at the onset of the Java revolution that can be used as a gauge for successful or unsuccessful Java compliance.

And the list of current deficiencies will slowly evaporate over the course of the next few years. Every vendor is working on Java tools that will solve the infrastructure and development problems. This is true because more and more corporations are using Java to develop new and exciting applications. According to a 1997 survey by Forrester Research, 52% of the Fortune 1000 firms surveyed were actively building applications with Java and 81% of those plan to build Java mission-critical systems by 1999. The same survey also shows that using Java to drive database updates is increasing in popularity with 54%, versus 4% the year before. Java is being used currently, and its use will continue to grow.

JIT Java compilers are getting faster and vendors are working on Java compilers that will probably resolve the speed issue once and for all.

A big benefit of Java applications that is hard to ignore is a lower cost of ownership compared to client/server applications. With client/server applications there are typically numerous DLLs scattered across machines that require modification, removal, or additional DLLs each time the application is updated. With Java you have a better model because the code is downloaded when it needs to be run. Of course, this can impact performance when the code is being downloaded, but at least the application will be more likely to run correctly.

For DB2 users, Java is supported by Net.Data. With Net.Data and Java, developers can create Java applets to process the results of Net.Data applications and create graphs, charts, and other interactive elements. As Java matures, it will become the way that you write stored procedures, user-defined functions, and triggers. Oracle, Sybase, and Informix stored procedures are based on proprietary SQL; DB2's rely on 3GL programs. Java stored procedures have the promise of running securely in any database — providing a possible huge benefit.

Eventually, developers will be able to embed SQL directly into Java programs, call Java programs from SQL statements, and even store Java object instances into DB2 columns. All of these things will enable users to create active DB2 databases. This is goodness because it places the responsibility for taking the proper action in a single, central, secure place — the database.

Finally, Java's ability to make the web interactive will ensure the success of Java. In as much as we as DB2 developers want to enable our users to access DB2 data over the web, we will rely on Java to help us do that.

Synopsis
There is enough hope for Java that we should all be able to put up with the hype that is out there. With time, patience, and a little luck, we will have web-enabled, multi-platform, distributed, Java applications accessing our active DB2 databases.

From *IDUG Solutions Journal*, August 1998.