Java-based Application Development Technology

by: Lynn Greiner

Docid: 00018686

Publication Date: 0807

Publication Type: TUTORIAL

Preview

A year after Java was released to open source, the language's footprint continues to expand, helped in large part by the agreements between Sun and Microsoft that eliminated the competitive distractions of their litigious past. Java is now firmly entrenched in the corporate world alongside Microsoft's .NET platform. This report examines its tumultuous past, its evolving present and its promising future.

Report Contents:

Executive Summary

[return to top of this report]

Created by Sun Microsystems, Java is an object-oriented programming language, meaning it allows modular software pieces to be reused and shared from one application to the next. The language found early success as an embedded device platform and as a way to enhance Web sites. Java, however, can also be used for a number of other development purposes, many of them highly sophisticated.

Java's history has been shaped significantly by Sun's relationships with other IT industry companies, relationships that have alternated frequently between cooperation and bitter dispute. The most notable relationship Sun has been involved with regards Microsoft. The parties have now settled their disputes

and stated that they plan to cooperate, but in some respects they remain competitors. Microsoft's .NET platform is an increasingly more compelling alternative to Sun's Java Enterprise Edition (Java EE), which is designed for the development of enterprise-caliber applications. The decision whether to use Java or an alternative for a particular application should be influenced heavily by the preferences and experiences of an organization's IT staff. Using a development platform that the IT staff is skilled with will help to reduce costs, speed development cycles, and yield better finished software.

Sun announced at its 2007 JavaOne conference that it had completed the transition of Java to open source. With the exception of a few "encumbered" technologies, all of Java is now open source, licensed under GPL v2. Some critics think that Sun made the decision too late, as rival technologies such as Microsoft's .NET may have already made Java less attractive an alternative than before, but that is for the market to decide.

Description

[return to top of this report]

Originally developed by Sun Microsystems, Java is a programming language designed for use on the Web, in intranet applications, and embedded devices, among other uses. A number of qualities make Java well suited to today's networked computing environment, and as developers are looking to link legacy back-end systems to dynamic interfaces online, many are turning to Java.

Like C++, Java is an object-oriented programming language. The goal of object-oriented programming is to allow independent software components to be reused and interchanged between programs. Software objects are designed to work together at run time, interoperating regardless of programming language or platform. When it comes to creating software objects, the key distinction between C++ and Java is the learning curve. Java was originally designed as a purely object oriented language, in contrast to C++, which added object-oriented capabilities to traditional C language programming. As a result, C++ is significantly more complex than Java.

Java is an interpreted language, which means Java source code is compiled to an intermediate syntax at run time. Compiled Java source yields byte codes, which are, in turn, executed by the local version of the Java Virtual Machine (JVM). The JVM translates byte codes to the host platform’s native instruction set. The JVM component of the interpreter is also called the Java Runtime Environment (JRE). The byte code format is architecture-neutral, and this is what makes Java programs “platform independent.” Java’s claims of portability rely on the fact that it is fairly easy to port the JVM to new platforms. High-level Java source code sees only the byte code compiler, and in theory, this interface is invariant. This differs from the more complex compile-link-run cycle of languages like C and C++, where the executable program is initially built for a specific target platform and its unique instruction set.

Java's portability is ideal for applications distributed over the Internet or other networks, and for creating platform-neutral applications for PCs. Though non-portable programs can exist in Java, they are relatively easy for programmers to avoid. But what Java offers in portability, it lacks in speed. Because Java is an interpreted language, it is not as fast as a entirely compiled language like C. Java advocates claim that the types of applications for which Java is best suited--GUI and networked-based--do not need as much speed as others. Java's speed--or lack thereof--has become an issue, however.

Part of the Java language developers' goal was to simplify C++ by removing some of the lesser used and more bug-prone elements of the C language, such as pointers, and adding other more efficient capabilities, such as the garbage collection feature that automates memory management. Java is multithreaded. It can handle different tasks simultaneously--improving interactivity and performance, and working with threads in Java is much easier than doing so in C++. In the range of programming languages available, Java falls in the middle--somewhere between the high-level interpreted scripting languages like Tcl and UNIX shells and low-level compiled languages like C and C++. Java attempts to combine the portability of the high level languages with the speed of the low-level variety.

Current View

[return to top of this report]

Java has evolved from a programming language to a system of development platforms. Deployed applications range from embedded systems for handheld devices to enterprise-wide, mission-critical applications. Java is clearly a force in distributed computing, but it is impossible to talk about Java without talking about Sun Microsystems. Sun owns the Java copyright and has tenaciously guided its development. Depending on whom you ask, Sun has either ushered in Java's adoption or hindered it irrevocably.

In a move that some consider too late, Sun announced plans in May 2006 to release an open source version of Java. The company fulfilled that promise, announcing a year later at JavaOne 2007 that it had completed the release to open source, with the exception of a few components still under licensing encumbrances. Sun is still working to resolve those issues.

Some of the other issues currently surrounding Java are outlined below.

Java Tools Vendors Target Enterprise Developers

Seeing an opportunity, Java tools vendors have taken the initiative, wrestling a degree of control of Java’s direction from Sun. Respected development technology players such as Borland integrated Java language tools with a broad spectrum of development technologies. Borland’s acquisition of program modeling software vendor TogetherSoft brought Java into a mix that facilitates program design, coding, and maintenance. The modeling component allowed developers to better manage both initial coding and ongoing maintenance by providing a functional roadmap for the behavior of software. In addition, Borland acquired tools vendors BoldSoft and Starbase. It later created a new division, CodeGear, for its development tool portfolio, and as of July 1, 2008, sold this division to Embarcadero Technologies, effectively removing itself from the development tools market. CodeGear (now Embarcadero) has addressed a long standing disconnect between application development and modeling. Its new suite of tools can dynamically synchronize code when the program model is updated, radically decreasing the cost of maintenance programming and improving reliability of updated code. In a similar move, IBM acquired modeling tools provider Rational Software, whose modeling tools buttressed its already comprehensive suite of development tools.

Microsoft and Sun Agree to Cooperate

After several years of acrimonious debate in the marketplace and courts, Microsoft and Sun entered into a broad technology collaboration arrangement to enable their products to work better together and to settle all pending litigation between the two companies. The companies have also entered into agreements on patents and other issues.

The agreements involve payments of $700 million USD to Sun by Microsoft to resolve pending antitrust issues and $900 million to resolve patent issues. In addition, Sun and Microsoft have agreed to pay royalties for use of each other's technology, with Microsoft making an up-front payment of $350 million, and Sun making payments when this technology is incorporated into its server products. The agreements also include the following elements:

- **Technical Collaboration--**The Technical Collaboration Agreement will provide both companies with access to aspects of each other's server-based technology and will enable them to use this information to develop new server software products that will work better together. The cooperation will initially center on Windows 2003 Server and Windows Client, but will eventually include other important areas, including email and database software. As a result of this agreement, Sun and Microsoft engineers will cooperate to allow identity information to be easily shared between Microsoft Active Directory and the Sun Java System Identity Server, resulting in less complex and more secure computing environments.
- **Microsoft Communications Protocol Program--**Sun agreed to sign a license for the Windows desktop OS communications protocols under Microsoft's Communications Protocol Program, established pursuant to Microsoft's consent decree and final judgment with the US Department of Justice and 18 state attorneys general.
- **Microsoft Support for Java--**The companies agreed that Microsoft could continue to provide product support for the Microsoft Java Virtual Machine that customers have deployed in Microsoft's products but this support terminated at the end of 2007.
- **Windows Certification for Sun Server--**Sun and Microsoft announced Windows certification for Sun's Xeon servers, and in September 2007 Sun signed on as a Windows Server OEM.
- **Future Collaboration for Java and .NET--**Sun and Microsoft agreed to work together to improve technical collaboration between their Java and .NET technologies. Thus far, little progress has been made on this front.
- **Patents and Intellectual Property--**The parties have agreed to a broad covenant not to sue with respect to all past patent infringement claims they may have against each other. The agreement also provides for potential future extensions of this type of covenant.

In September 2007 the agreement was expanded, adding the following elements:

- **Windows Server OEM agreement.** Sun will offer Windows Server on their x64 hardware and will provide additional utilities and value-added software offerings to server systems carrying Windows Server. Windows Server 2003 will be available on Sun x64 systems within 90 days of the agreement.

- **Sun x64 systems and storage.** Microsoft recognizes Sun’s compelling x64 server and storage products in the market today. Microsoft and Sun will continue to work together to test and validate the Windows platform on these systems for scale-up enterprise computing.

- **Solaris and Windows virtualization.** Sun and Microsoft will work together to ensure that Solaris runs well as a guest on Microsoft virtualization technologies and that Windows Server

runs well as a guest in Sun's virtualization technologies. Sun and Microsoft will work together on a support process for customers using the virtualization solutions. This joint commitment to customers ensures that Windows and Solaris will provide a solid virtualization experience.

- **Expanded IPTV partnership.** Sun and Microsoft will continue to collaborate to advance the worldwide deployment of the Microsoft Mediaroom IPTV and multimedia platform on Sun server and storage systems. AT&T U-verse digital TV offering has chosen Sun for one of world's largest deployments of the Microsoft Mediaroom platform, which includes server and client software. The combination of Sun's server solutions and Microsoft's TV software technology and expertise is expected to speed time to market for IPTV services, while providing superior cost and performance characteristics for communications service providers worldwide.

- **Expanded investment in interoperability.** Microsoft and Sun will build an Interoperability Center on Microsoft's Redmond campus. The center will include a demonstration area for Sun x64 systems, act as a working lab for Windows on Sun benchmarks and sales tools, and support customers running proofs of concept for projects focused on Windows on Sun x64 systems, including joint Sun/Microsoft solutions in areas such as databases, e-mail and messaging, virtualization, and Remote Desktop Protocol (RDP) support in Sun Ray thin clients. The Interoperability Center will expand Sun's presence on the Microsoft main campus, adding to existing Sun systems showcased and customer-tested in the Microsoft Enterprise Engineering Center.

While Microsoft has discontinued building its Microsoft Java Virtual Machine into its products, and stopped supporting the products at the end of 2007, the two companies have created a basis for tighter interoperability between Platform Enterprise Edition (Java EE), the Microsoft .NET Framework 3.0 and Windows Communication Foundation in Sun's Web services interoperability technologies (called Project Tango). Sun is also a founding member of the Microsoft Interop Vendor Alliance and a Microsoft Gold Certified Partner. This is the best of both worlds for Sun, and for the development community it is a major step toward encouraging use of Java when working with Microsoft applications.

A Confrontation Over Standards

With promises of "Write once, run anywhere for the server," Java EE (formerly J2EE) is designed to create applications for the networked, e-commerce-enabled environment. Detractors, however, have noted that Java EE is not significantly different from Java and, more than anything, has created confusion and dissension in the Java community. With Java EE, Sun has essentially put all Java server technology under one umbrella. Java EE runs on top of Standard Java 2 and includes compatibility tests for the entire platform, an Application Programming Model, and a Connector Architecture to allow EIS systems to plug into compatible application servers. IBM has called for Sun to unbundle Java EE, allowing developers to use products in the package separately.

The Glassfish project is a community project that is developing an open source reference implementation of Java EE. To address concerns about the compatibility of all of the open source Java components, the Technology Compatibility Kit (TCK), a mandatory compatibility test, has been released; it is not open source, although Sun is finalizing licensing terms for its use on OpenJava.

To further enhance compatibility among implementations, the Java Community Process, Sun's governing body for Java specification development, voted to adopt the Web Services Interoperability

Organization’s interpretation of Key Web Services Standards. Adoption of the Web Services Interoperability group’s Basic Profile endows Java with a set of non-proprietary Web Services specifications, including a set of clarifications that promote interoperability.

At JavaOne 2007, Sun announced the OpenJDK Interim Governance Board, whose charter was to draft and gain ratification of a new constitution for the OpenJDK Community within the next year, with active participation from the membership. The goal of OpenJDK governance is to be representative of a broad and inclusive consensus of OpenJDK members. The community will then hold an election to replace the Interim Governance Board with a duly elected board in accordance with the OpenJDK constitution. However, during the initial year it became apparent that the community had other priorities, so the constitution was put on the back burner and the interim governance board's tenure extended for another year while developers worked on IcedTea, an open source Java development project designed to allow the use of the OpenJDK without needing any proprietary software.

JavaBeans, Java’s Reusable Component Architecture

Sun’s vision of Java EE, now being developed by the open source community as JSR-88, reflects the importance of multi-tiered architectures in the enterprise market. Developing for the enterprise involves building n-tier applications for large scale client/server models. Java EE promises to upgrade the client/server model by improving upon functions like software reuse, deployment and support, scalability, and platform independence. A crucial part of the enterprise platform is the JavaBeans component architecture. JavaBeans provide an interface that allows independent program modules to communicate. As reusable objects, JavaBeans also reduce development time and reliability.

JavaBeans encapsulate user interfaces on the client side and business process logic in their enterprise format, Enterprise JavaBeans. Enterprise JavaBeans offer vendor independence and portability across different platforms and application servers, providing consistent implementation of business process support. In this role, they offer great advantages as a middle-tier of software objects, interoperable with back-end systems, and linking distributed application integration technologies like CORBA.

Java’s Runtime Personalities: Servlets, Applets, and Applications

Simply put, Java executables come in three basic formats: servlets, applets, and applications. Servlets are Web server extensions that are alternatives to traditional CGI scripts. They sit on the server and dynamically create HTML page descriptions. Unlike applets, they are never transferred to the client, but instead run entirely on the Web server. Applets, on the other hand, are bundles of Java code that are downloaded to a browser with a Web page. Performing the "HTML glue" functions that CGI scripts traditionally have done, both are used to create graphical elements like rotating ad banners and to perform e-commerce-related functions such as transferring user input from a Web page back to a data base or server. Because servlets are on a Web server, they perform better than CGI scripts. JavaScript, which is a scripting language from Netscape, performs similar glue functions, often tying Java applets together.

The front-end of the Java Enterprise System includes such technologies as Java Server Pages (JSP) and servlets. These technologies are essentially the front door to Java EE development. They allow for dynamic and changing content on the client side without additional application development. Servlets require explicit server side support, both for development and for use. Some popular Java software development programs include NetBeans, Embarcadero (formerly CodeGear) J Builder, JetBrains

IntelliJ IDEA, and IBM's WebSphere environment.

Recognizing that Java lacks the tools to produce rich multimedia experiences, Sun has developed JavaFX, a Flash-like addition to the technologies that will consist of a runtime environment, widgets and development tools. At JavaOne, it released the first pieces: JavaFX Script and JavaFX Mobile. Although there are currently no developer tools for JavaFX, the JavaFX SDK is expected to go into private preview in July 2008 and several other components will be available in late summer and in the spring of 2009.

AJAX

A relatively new addition to the Java world is Asynchronous JavaScript and XML, better known as AJAX. AJAX is a Web development technique that enables developers to develop interactive Web applications, and forms the basis for Web 2.0. AJAX helps developers create applications that are more responsive to users by exchanging small amounts of data with the server behind the view of users, essentially allowing a Web page to bypass having to reload each time a user makes a change. AJAX looks to increase a Web page's interactivity, speed, and usability. While AJAX has both its positives and negatives, the use of AJAX, as well as solutions supporting the technology, continues to grow.

Outlook

[return to top of this report]

Sun's open source efforts started with the open-sourcing of its Solaris operating system. Following Sun's announcement that Jonathan Schwartz would replace Scott McNealy as the company's CEO, Sun announced that the company would release an open-source version of Java. In June 2006, Simon Phipps, the chief open-source officer for Sun, said the company the release of open source Java was "months" away (a year later, Sun announced that the release had been completed), though the company was struggling with two major issues over the move: how to keep Java compatible, and how to ensure no one company used market forces as muscle for its own implementation. The latter move would threaten Java's "write once, run anywhere" mantra. The key to maintaining compatibility, according to Sun, is determined by the company's decisions in the areas of licensing and governance: the founding of the OpenJDK Governance Board and Java's release under GPL v2.

As developers look to speed up Java's performance, they are driving a move toward server-based architecture, where the majority of the computing responsibilities fall on the server, not the client. One result of this is the migration of the functions of applets to servlet technology. Java's performance issues have brought about developments in other areas as well. New compilers attempt to mitigate the slow performance resulting from Java's byte code architecture. The Java Virtual Machine (JVM), the traditional method of interpreting Java, has benefited from the incorporation of advanced compiler technology. In the newer approach, the compiler converts Java source code into native machine code all at once instead of on an as-needed basis, as JVMs do. This makes a program platform dependent, but it executes at native machine speeds. In addition to Sun's product, IBM, HP, Novell, and Intel also developed compilers. Some compilers give developers a choice of either platform-specific or portable output, which may be the direction Java development takes as attempts are made balance issues of performance and portability.

Java still has some kinks to iron out before it becomes as widely used as other languages such as C++. Its garbage collection function, for example, can create glitches when it runs, which is one thing on the desktop and quite another for embedded systems. Sun has provided tuning advice to help alleviate the problems. The challenge in writing Java for embedded systems has been dealing with the slow speed and memory demands of compiling the byte code. Attempts are being made on the hardware side to make Java run more smoothly in embedded systems.

Java is a significant player in the embedded devices market. The Java Micro Edition (Java ME, formerly J2ME) has been incorporated into Palm handhelds, the RIM BlackBerry, and Motorola pagers, as well as other products such as cell phones and car navigation systems. PersonalJava was a specification for Java in networked embedded devices such as set-top boxes; its functionality has been rolled into Java ME.

Another application of Java in the embedded technologies market is Jini, Sun’s Java-based distributed computing environment in which devices can be dynamically and automatically incorporated in a network. Several US government organizations, including the Department of Defense and the FAA, have implemented employee and contractor identification in Java-based smart card systems. These cards store biometric information about the bearer, as well as the usual data, rendering the cards useless if stolen.

There are a number of Java APIs designed for multimedia in client/server applications. Sun and IBM's Java Media Framework is designed to enable streaming audio and video to be captured and transmitted across major operating systems. This adds a media rich element to Java's enterprise application capabilities. Its small footprint makes it a good fit for applications in portable multimedia systems such as video players, kiosks, and Web-toys. The newly announced JavaFX provides a development platform for the rich content.

The need for a platform-independent language to handle complex e-commerce functions is paramount, and Java has seen great success in this niche. Large-scale Java applications are in common use today, new developers are cropping up all the time, and its presence in consumer products is promising. Java is the platform of choice for use in portable handheld devices like cell phones and PDAs (with, of course, the exception of the Pocket PC).

Java's future is more secure now that Sun has, as promised, released it to open source. As long as the governance group prevents incompatibilities from creeping in, the platform has a chance of becoming a truly universal tool.

Recommendations

[return to top of this report]

Java can be used in many ways in many scenarios, from small uses that have little impact on an organization's overall IT development to far-reaching uses that affect an organization's entire application development and IT strategies.

Although Java may have absolute advantages in some areas of development, such as development for

embedded devices, viable alternatives exist for any project a developer may wish to complete. Many organizations choose between Java and a competing platform based on the preferences and experiences of their internal IT staff members. In most cases, this is a reasonable way to choose. Selecting a platform that an organization's existing programming team is experienced with will help to shorten development lifecycles, reduce staff and training costs, and minimize flaws in the finished product.

The area in which the decision whether to deploy Java becomes most significant is in choosing between Java EE and Microsoft .NET. While Java remains more prevalent than .NET, usage of the platforms is becoming more even. Java had a head start on .NET in respect to popularity and capability, but the narrowing of the capability gap is prompting a narrowing of the popularity gap. Selecting .NET can limit users to choices of development tools. There are a much broader range of software tools available to Java EE developers. Developers working with .NET will rely on Microsoft's Visual Studio. Provided that users are comfortable with Visual Studio, the lack of alternatives may be of little relevance. Visual Studio is a sophisticated, well-designed application. Developers who have particular preferences for other products and features not available in Visual Studio, however, may prefer to continue using Java EE. The important consideration is that the choice between the platforms needs to be in part a choice between development tools. Organizations that have standardized on Microsoft technology are best off using .NET.

Microsoft and Sun have settled their disputes (at least officially), and they have publicly committed to making Java EE and .NET more interoperable. Microsoft has released considerable guidance on the use of .NET with various Java platforms, finally admitting that heterogeneous systems are the norm. This bodes well for the future of both Java and .NET.

About the Author

Lynn Greiner is Vice President, Technical Services for a division of a multi-national corporation, and also an award-winning computer industry journalist. Ms. Greiner is a regular contributor to Faulkner Information Services and a member of the Advisor Panel.

Web Links

[return to top of this report]

Embarcadero Technologies: http://www.codegear.com/
DevX Java Zone: http://www.devx.com/Java/Door/6972
ECMA International: http://www.ecma-international.org/
Hewlett-Packard: http://www.hp.com/java
IBM: http://www.ibm.com/developerworks/java
JetBrains: http://www.jetbrains.com/
Microsoft: http://www.microsoft.com/
Novell: http://www.novell.com/
Sun Microsystems: http://www.java.sun.com/

[return to ***top*** of this report]