Java™ for Competitive Advantage

Chip Kelly

Abstract

Many companies are finding the web and its associated technologies a key ingredient for moving forward in business today. The distribution of application logic throughout the enterprise, the use of network computers to execute those applications, and the ability to extend the use of those same applications to distributors and customers are compelling reasons to use the web. One of the most fascinating advances in computer linguistics, Java, is a major contributor to the success of the web in many organizations. This paper will present some of the benefits that the Java language, and the Java platform, bring to users of the SAS® System of software.

To get the most out of this presentation, you should

- be interested in the web as a business tool,
- be familiar with the capabilities of the SAS System,
- have some familiarity with Java as a language and platform.

Introduction

Java has been around for a couple of years now, and still a day does not go by that you don't see an article or an opinion about it. Why is that? What is the big deal about Java? And why has it remained on the forefront of the technology tide known as the web? We hope to answer these and other questions you may have about Java, and provide you with a framework for understanding the potential value to your organization that Java represents.

We'll start with a brief history of Java, just to make certain that we are all speaking the same language. Then we will provide a landscape view of the current Java architecture, noting the significance of certain features for applications developers, followed by a brief glimpse into the nearterm plans for Java from the folks at Javasoft, a subsidiary of Sun Microsystems Inc.

SAS Institute is committed to using Java in many different aspects of SAS software. The next few sections will briefly describe some of the Research and Development that is ongoing with respect to Java, in all areas of the Institute.

Finally, based on the information we've given you about Java, and how SAS Institute is planning to use it, we'll enumerate some of the aspects of Java that may help you gain competitive advantage in the marketplace.

The Java Phenomenon

Before this information fades completely into urban legend, let's look back at the history of the language we now know as Java.

A Brief History of Java

Java was invented in a lab at Sun Microsystems by a research team lead by James Gosling. They was trying to find an elegant way of programmatically controlling small, reliable devices that only possessed small amounts of memory, used extremely simple processors, were typically distributed in heterogeneous, mobile environments, and required robust, real-time embedded systems. The year was 1990. None of the languages of the day were working to his expectations, so Gosling started prototyping a new language that he named "Oak".

Oak was a compact language, like C. It was an object oriented language, like C++ and Smalltalk. It was not designed for numerically intensive applications like FORTRAN, and it was not as verbose as COBOL, although it does well with character-based algorithms. What Oak did well can be summarized in the following bullets excerpted from *The Java Language: A White Paper*[1]:

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure
- Architecture Neutral
- Portable
- High Performance
- Multithreaded

[1] The Java Language: A White Paper, 1994, Sun Microsystems, Inc.

- Dynamic

The Oak language was suitable for small portable devices for the same reasons that it was ideal for creating portable executables for the Internet. Realizing that Oak could be used to produce programs that could run on any computer that was connected to the Internet, Sun Microsystems began re-tooling the Oak project, creating the beginnings of the Java phenomenon.

Most of the initial desirable aspects of Java were appealing to programmers. The first programs written in Java for the Internet were applets that could be embedded in browsers. The language was simple. From the perspective of C and C++ programmers, it was small, familiar, and did not use pointers for referencing objects, one of the biggest source of bugs in a C/C++ program.

By the time most of us had heard about Java, it had been around for nearly a year, and tinkered with by thousands of programmers. Sun Microsystems placed the initial Alpha release of Java on their web site for free downloads, and word of the "fun" spread like wildfire. Once Netscape announced that its Navigator browser would support Java applets, Java's popularity skyrocketed, as did Sun's commitment of resources to the project. Sun's JavaSoft subsidiary was created to manage the increasing importance of Java as not only a programming language, but as a potential platform competitor to the reigning client platform – Windows.

Current Java Architecture

To understand the strategic importance of Java for your organization, we need to take a somewhat comprehensive look at the architecture that Sun Microsystems has implemented for the Version 1.1 release of the Java Developers Kit, or JDK™1.1. Earlier versions of the JDK are still in use since many web browsers only fully support the JDK1.02 specification. But the real value of Java to your company is starting to be realized with JDK1.1, so let's take a look at what's inside.

- Language –The virtues and shortcomings of the Java language when compared to C and/or C++ is documented elsewhere[2], and will not be discussed at length here. Suffice to say that it is a compiled language, the compiled output is portable and interpreted by a virtual machine that creates in realtime the actual executable binary. From the first production release, JDK1.02 to the now current JDK1.1, significant changes in what is called the "event model" allow applets/applications written in JDK1.1 to exercise much better control over aspects of the applet/application that interact with the user and other components.
- Classes – Since Java is an object-oriented language, the components of the language itself are also structured as objects and classes. Related aspects of the language are grouped into packages, all with similar, and sometimes inherited behavior. The packages in JDK1.1 include:
  - Applet – controls that attributes of applets, small applications that execute inside a web browser.
  - Java Beans™ - the component architecture framework for Java.
  - Advanced Window Toolkit – contains all the classes for creating user interfaces, and for painting graphics and images. The new event model for JDK1.1 is embodied in this package.
  - IO – contains classes for building input and output data streams.
  - Lang – the core classes of the Java language are in this package. This is where the characteristics of an object are defined, and other indivisible entities like strings and threads.
  - Net – contains classes for implementing networking applications.
  - Util – contains miscellaneous classes that assist with the hashtables, dates, random numbers, etc.

Other related pieces of the Java Developers Kit

- Security - a framework for developers that includes cryptography with digital signatures, encryption, and authentication.
- Internationalization –a framework for using the execution locale of an application to determine the language chosen for display of user interfaces and error messages.
- JAR – the Java Archive format, a platform-neutral method of compressing and storing Java applications.
- RMI – Remote Method Invocation, a framework that allows distributed Java-to-Java communications, in which the

[2] Java for C/C++ Programmers, 1996, Michael C. Daconta, John Wiley & Sons, Inc.

methods of remote Java objects can be invoked from other Java virtual machines.

- Object Serialization –supports the encoding of objects, and the objects reachable from them, into a stream of bytes; and it supports the complementary reconstruction of the object graph from the stream. Used for lightweight persistence
- JDBC™ - a standard SQL database access interface, providing uniform access to relational databases
- Inner Classes - allows classes to be defined as members of other classes, locally within a block of statements, or (anonymously) within an expression.
- JNI - Java Native Interface is a standard programming interface for writing Java native methods and embedding the Java virtual machine into native applications.

Those bullets really describe only the extensive capabilities of the Java language. To get a more complete picture of how Sun Microsystems is attempting to mature Java into a platform, additional APIs are currently available that extend the reach of Java into electronic commerce and multimedia[3]:

- Java Foundation Classes
  extend the original Abstract Windowing Toolkit (AWT) by adding a set of portable graphical user interface class libraries.
- Java Mail
  A set of abstract classes that models a mail system.
- Java Management
  Aids the development of system, network, and system management solutions for heterogeneous networks.
- Java Electronic Commerce Framework
  A virtual point-of-sale terminal accessible by any Java-enabled browser that provides a framework for payment methods. (Java Wallet)
- Java Media and Communication APIs, including 2D, 3D, and Java Telephony
  Allows developers and users to take advantage of a wide range of interactive media on the Web. Includes: Media Framework, Sound, Animation, 2D, 3D, Java Telephony, Speech and Collaboration.

In addition to those extensions of the JDK, the following Java products further enlarge the development environment for companies that choose a Java solution:

- JavaSpaces™
  Provides a distributed persistence and data exchange mechanisms for code written in the Java programming language.
- Java Activator
  Provides enterprise customers with the ability to specify the use of Sun's implementation of the Java Runtime Environment (JRE) in Internet Explorer 3.02 or later, and Netscape Navigator 3.0 or later instead of the browsers default Java virtual machine (JVM).
- JavaPC™
  A software solution that converts older DOS PCs into Java-enabled network computers.
- JavaHelp™
  Java-based help system
- Java Blend
  Simplifies database application development. A tool and a runtime environment.

Early users of Java limited their endeavors to applets that execute inside web browsers. Small device vendors, e.g. pagers, telephones, credit card producers, quickly picked up on the unique qualities of Java that allow it to deliver customized executables to a personal device. Sun Microsystems solicited cooperation from selected vendors and jointly developed the following products and APIs for that high growth market:

- Java Card
  Provides detailed information about the Java Card v2.0 language standards, APIs and developing Java Card applets to run in smartcard devices.
- JavaOS
  A new platform optimized to run Java on a variety of computing and consumer platforms. JavaOS provides a runtime specifically tuned to run Java applications directly on hardware

[3] The APIs and their brief descriptions are excerpted from http://java.sun.com/products/index.html , 1995-97 Sun Microsystems, Inc.

platforms without requiring a host operating system.

- Embedded Java
  An API and application environment for high-volume embedded devices.
- Personal Java
  An API and application environment for network-connectable applications on personal consumer devices.

To allow Java to be in used in complex applications that are typically found supporting enterprise level endeavors, Sun Microsystems developed the Java Enterprise APIs[4]:

- Enterprise JavaBeans
  The Enterprise JavaBeans specification defines an API that will make it easy for developers to create, deploy and manage cross-platform, component-based enterprise applications that work within the framework of the systems currently in use.
- Java Naming and Directory Interface (JNDI)
  Provides uniform, industry-standard, seamless connectivity from the Java platform to business information assets, thus allowing developers to deliver Java applications with unified access to multiple naming and directory services across the enterprise.
- Java IDL
  Provides interoperability with CORBA, the industry standard for heterogeneous computing. Java IDL includes an IDL-to-Java compiler and a lightweight ORB that supports IIOP.
- JDBC™ Database Access API
  Provides programmers with a uniform interface to a wide range of relational databases, and provides a common base on which higher-level tools and interfaces can be built.

Most of the activity in Java thus far has centered on applets/applications that execute on the client. There is some detectable movement in the industry that leads Sun Microsystems to believe that Java is a good platform for server-centric applications as well, and they have come up with the JavaServer™ Product Family that support that aspect of Java usage[5]:

- Java Web Server
- Server Toolkit
  Simplifies the task of developing network-centric servers in the Java programming language. (Available for Licensees Only)
- Servlet Development Kit
  Includes a servlet engine for running and testing servlets, the javax.servlet.* sources, the API documentation for javax.servlet.*, sun.servlet.*, and support for Netscape, Microsoft, and Apache web servers.

The Impact of Java

As you can see from the preceding section, Sun Microsystems has been very busy in the last two years, and has produced an amazing spectrum of capabilities for Java. When you couple Java with the the web for use in an enterprise level application, the infrastructure that becomes available to you on many heterogeneous hardware platforms is a compelling reason for considering Java as part of your solution. This process has been unfolding in the labs of the major

- hardware vendors(IBM®, HP, SUN, Apple)
- major operating system vendors(IBM, Sun, Microsoft, Apple)
- systems software vendors(Symantec, Borland, Roguewave, Computer Associates)
- database vendors(IBM, Oracle, Sybase, Informix)
- commercial applications development vendors(SAS Institute Inc., IBM, IBI to name only a few),
- large in-house development groups(pick your favorites from the Fortune 2000).

The list leaves out very, very few important aspects of the information industry. Not all reports of progress are rosy. Some vendors are experiencing more difficulty with implementations than others. But the bottom line is the overwhelming acceptance of Java as a player in areas that are of concern to SAS

[4] excerpted from http://java.sun.com/products/index.html , 1995-97 Sun Microsystems, Inc.

[5] excerpted from http://java.sun.com/products/index.html , 1995-97 Sun Microsystems, Inc.

Institute and its customers. For that reason, SAS Institute chose to join the frontline of hardware, platform, and software vendors that licensed the Java environment from Sun Microsystems.

SAS Institute Inc. and Java

Becoming a Sun Microsystems licensee enabled SAS Institute's Research and Development staff to begin developing close working relationships with their Sun Microsystems counterparts, and to participate in the design phases of new components and APIs that significantly impact SAS software.

As an example, one of the earliest implementations of the Java Database Connectivity (JDBC) classes was done by Barbara Walters in producing the SAS/SHARE*NET™ Driver for JDBC that is now being distributed with SAS/IntrNet™ software. This highly collaborative effort actually started in advance of SAS Institute becoming a licensee, and paved the way for cooperation in other areas. The expertise of SAS Institute R&D in areas germane to Sun Microsystems's design goals, e.g., portability, architecture neutrality, handling missing values, is a key ingredient for Sun Microsystems engineers in the ongoing process of maturing the Java language and environment.

The relationship between the two software development groups has shown advantages to both. At SAS Institute, the use of Java as a potential solution component has spread to all of the Research and Development divisions. Java "experiments" are cropping up all over, some of which are advancing to product integration phases.

Internal Database Division

The earliest purveyors of Java technologies at the Institute are part of the Internal Data Base Division, and are responsible for the delivery of the SAS/SHARE*NET Driver for JDBC, the JConnect class libraries, JTunnel, and the JViewer applet, all available now as part of the SAS/IntrNet product. This group is still actively engaged with Java technology, and is currently working on the following key projects:

- **Java Warehouse Viewer** – an applet for data warehouse clients that presents the contents of the Warehouse by navigating the metadata generated by SAS/Warehouse Adminstrator™ software.

  The metadata provides logical groupings of objects, along with detail information concerning the object type, location (on which server machine), and business notes provided by the administrator. Using a point and click interface, the user can select objects registered in the Data Warehouse and display the object's contents. Because the Java Warehouse Viewer is an applet written entirely in Java, the user needs only a Java capable browser installed on their machine. The applet is dynamically downloaded. No SAS or Data Warehouse specific software is installed on the user's machine.

  Features included in Warehouse Viewer:
  - Viewers for LOG, OUTPUT, SOURCE, GRSEG, IMAGE, FORMAT, FORMATC, tables, summary tables and MDDB's
  - sorting by subject, by type, by owner)
  - metadata searching (object metadata and column metadata)
  - user profiling (viewer preferences and scoping) (this is a future enhancement
- Java components for server access included with the SWAN IDE described below
- Additional **samples** written in Java will be delivered in upcoming releases of SAS/IntrNet software, addressing a wide array of user needs.

For more information about the ongoing Java projects in the Internal Database Division, see *Overview of Java™ Components and Applets in SAS/IntrNet™ Software*, by Barbara Walters and Don Chapman, delivered at this SUGI.

Display Products Division

Another Java hotspot at SAS Institute is the Display Products Division, creators of the SAS/AF® product, FRAMEs, the Screen Control Language®, and many other full-screen, interactive capabilities of SAS software. The popularity of SAS/AF as an applications development platform for SAS-based applications both on client and server platforms has sparked significant interest in the use of Java as a potential method for delivering those applications.

Current projects on display at this SUGI include:

- **Interactive Development Environment(IDE)**

A visual application builder that generates 100% Pure Java applets and applications. It contains full support for JavaBeans and includes a rich set of JavaBeans-compliant components and includes built-in support for access to SAS data and the full set of SAS server capabilities.
Features:

- Visual drag-and-drop application builder
- Simple drag-and-drop component wiring including model/view connections and property linkages
- 100% Pure Java source code generation
- JDK 1.1 compliant development environment
- Remote Java procedure access using RMI.
- Remote SAS server access (currently using either TCP/IP or HTTP tunneling, with planned support for DCOM and CORBA) as follows:
  - Access to remote SCL classes as local Java objects
  - Access to remote data using JDBC
  - Access to server-side SAS language and procedures
- Comprehensive support for JavaBeans
  - customizers and property editors
  - Introspection using reflection and BeanInfo
  - JavaBeans event model
  - Serialization
  - Creation of JavaBeans components
  - Import JavaBeans from JAR files onto the component toolbar
  - A rich set of more than 80 components that includes standard UI widgets, data access components, and utility components.
- A powerful integrated development environment with the following key features:
  - A Project Navigator
  - Tabbed component toolbars.
  - Output region with tabbed views of output, the Java log, user tool output, and find in files output. Support double-click navigation to the associated source error or search hit.
  - Language-aware source editor
  - Application debugger
  - Property sheet that supports JavaBeans property editors and customizers.
  - Component state is saved using serialization.
  - Class browser
  - Class editor that can generate standard Java classes, JavaBeans compliant components, or specialized components for SAS data access.
  - Project wizard for application setup
  - Event wizard for connecting components to one another.
- **Remote Object Class Factory(ROCF) –** an API and framework for enabling the remote execution of SCL classes from any Java client. This work is the springboard of SAS Institute's efforts to ensure openness of the SAS System. ROCF enables the use of industry standard communications protocols like CORBA and DCOM to bridge new client interfaces with SAS-based objects.

  Build Time Features
  - Generate proxies for remote SAS data or SCL objects using, JConnect.(RMI, CORBA, and DCOM are planned).
  - Drag-and-drop client server connections and support for JavaBeans customizers to alter the remote connection setup.
  - Single-click remote proxy generation

  Run Time Features:
  - Proxies support remote access at runtime
  - Supports HTTP tunneling and web browser security models
  - Ability to switch servers at runtime
  - Ability to execute as a servlet

- **SWAN Business Viewer**
  Java application designed to view OLAP data held in MDDBs or viewable via HOLAP. It generates JavaBeans, which can be imported into other Java applications if required.

Advanced Visualization Division
Computer Graphics Research Division

The Advanced Visualization Division(AVD) has delivered outstanding 2-D and 3-D graphics capabilities to SAS software in

SAS/SPECTRAVIEW® software product. They recently became involved in the delivery of 3D graphics images to the web with the creation of the VRML viewer, a Java class that allows controlled viewing of files that conform to a subset of the 2.0 release of the Virtual Reality Modeling Language(VRML) specification. The VRML viewer is distributed with the SAS/IntrNet product, and used by the JViewer applet to display VRML files generated by the SAS procedures.

The VRML Viewer was the tip of the iceberg for AVD's commitment to Java. They are now working in conjunction with the Computer Graphics Research Division to deliver high resolution graphics to SAS software clients with highly efficient, compact Java classes that are easily transported across a network and executed on clients architectures of all types. Application programmers using these classes don't have to be concerned about the capabilities of the recipient clients, only that they support Java executions.

For more information about what the Advanced Visualization Division is doing with Java technologies, see *Data Visualization Using Java and VRML* by Lingxiao Li and Art Barnes, delivered at this SUGI.

Business Solutions Division

The Business Solutions Division(BSD) supports the CFO Vision™, Enterprise Miner™, SAS/EIS®, Enterprise Reporter™, SAS/GEO™, HR Vision™, IT Service Vision™, SAS/MDDB™ Server software, SAS/Ph-Clinical and Risk Management products. The role of Java in delivering these products is currently under scrutiny, but some candidates are coming to the surface.

Based on the efforts of the Display Products Division to add Java applications development capabilities to their SWAN IDE, SAS/EIS is a likely beneficiary for a Java makeover. Another area of interest is the availability of Java-based graphics capabilities that could find their way into a number of BSD products. Enterprise Miner may lead the way with the inclusion of VRML file generators that use AVD's VRML Viewer for display on the client.

Applications Division

The Applications Division develops many of the technical products in the SAS System including SAS/STAT®, SAS/ETS®, SAS/OR®, SAS/QC®, and SAS/INSIGHT®. The division is also developing the Risk Management product and significant portions of the Enterprise Miner in partnership with the Business Solutions Division. There is particular interest within the division in creating better user interfaces for the technical product areas as demonstrated in part by the Analyst Frame application for statistical analysis.

Developers in the Applications Division have collaborated with developers in the Internal Database and the Display Products Divisions on several experimental Java efforts. These experiments include applets that focus on a specific task, applications that present a framework for doing multiple tasks, and client/server applications using the Remote Object Class Factory.

It is too early to describe a specific role for Java in product development in the Applications Division. Developers in the division have a strong interest in Java and will use the Java IDE and ROCF to capitalize on the competitive advantages that Java offers.

Java for Competitive Advantage

This paper gave you two concepts:

- Java is important to the information industry, SAS Institute Inc., and you because of its scope, acceptance, and aspirations.
- SAS Institute is serious about the use of Java in delivering solutions to you, and is already moving quickly to give you the tools, infrastructure and products that will be of best benefit in the coming years.

As a voting member of the information industry, you choose, with your purchasing decisions, what is important and what is not. Our hope is that you now have enough information about Java, the language and environment, and about how we are using it to move you into the next generation of decision support software, to vote for what benefits you and your organization.

Summary

Java presents us with a unique opportunity to begin solving many longstanding issues in the deployment of distributed applications. The consolidation of languages on both client and server platforms, a scaleable architecture that encompasses the full range of devices and operating environments, compliance and exploitation of industry standards, and the real possibility of becoming an industry standard are all compelling reasons to consider Java in your arsenal of tools for combat in the marketplace. Whether you compete in the hardware or software industries, or use information technology to manage your company or government agency, Java is worth a look.

Author: Chip Kelly
sascwk@sas.com
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 677-8000 x7033