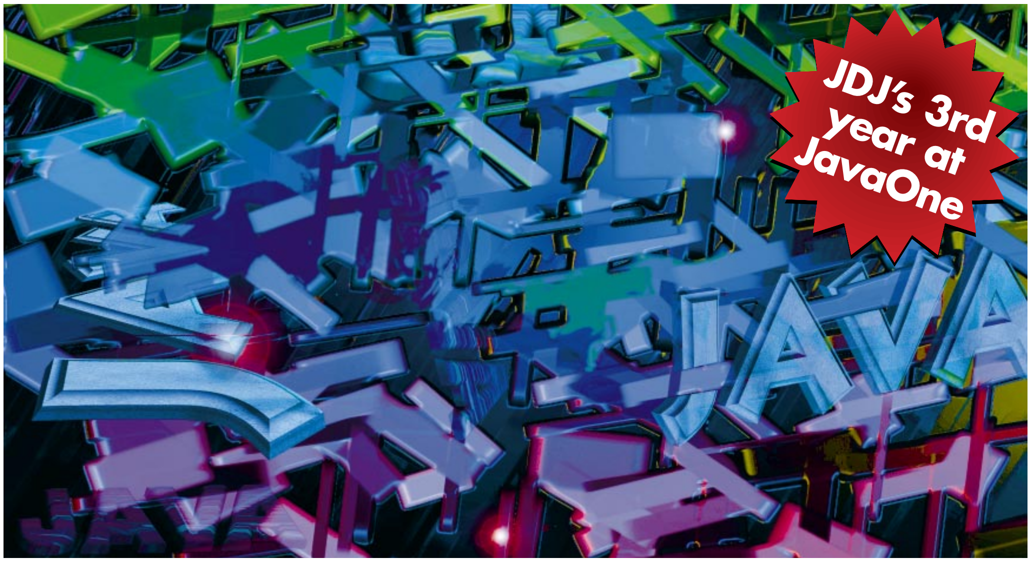
FREE SUBSCRIPTION AT JAVAONE

U.S. $4.99 (Canada $6.99)

JAVA™ DEVELOPER'S JOURNAL

JavaDevelopersJournal.com

Volume: 3 Issue: 3



JDJ's 3rd year at JavaOne

From the Editorial Board

Life Beyond the Browser by Arthur van Hoff pg. 70

Java and Corba by Richard Soley pg. 72

From the Industry

Java's First 1000 Days by George Paolini pg. 78

Microsoft Announces Visual J++ 6.0 by Bill Dunlap pg. 62

From the Java Lobby

Developers and the Java Alliance by Rick Ross pg. 24

JavaOne News

IBM Launches jCentral at JavaOne

.....

IBM Delivers San Francisco at JavaOne

pg. 80

Product Reviews

Visual Quantify Zebrowski pg. 74

he Sun

oding

59

The End o

by Joe S. Valle

JDJ Feature: **PersonalJava and Inferno** Steven Frank

New opportunities for application developers in an emerging market 8

JDJ Interview: **The World in Your Wallet** Scott Davison

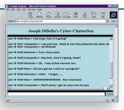
Tom Lebsack on Schlumberger's Cyberflex Java Smart Card 17

From the Ground Up: **Layout Managers** John Tabbone

An interface that provides rules for Component placement 27

JDJ Feature: **Building a Chat Applet** Joseph DiBella

Interfacing with CGI/Perl scripts on a Web server 30



JDJ Feature: **Interfacing Transaction Services** Maros Cunderlik

Making a critical task in enterprise computing easier 52

Visual Café: **Object Serialization** Alan Williamson

Saving and loading complete classes using sockets and JDK 1.1 68

CORBACorner: **CORBAScript** *Using one standard language as the glue for your components* Jeff Nelson 64

ІAL JDJ BUYER'S GUIDE INSERT

SYS-CON PUBLICATIONS

Record Circulation 65,000 COPIES

DEVELOPER'S JAVA JOURNAL

Art Director: Jim Morgan
Executive Editor: Scott Davison
Managing Editor: Gail S. Schultz
Editorial Assistant: Christy Wrightington
Copy Editor: Alix Lowenthal
Technical Editor: Bahadir Karuv
Visual J++ Editor: Ed Zebrowski
Visual Café Pro Editor: Alan Williamson
Product Review Editor: Jim Mathis
Games & Graphics Editor: Eric Ries
Tips & Techniques Editor: Brian Maso
Java Security Editor: Jay Heiser

WRITERS IN THIS ISSUE

Maros Cunderlik, Scott Davison, Joseph DiBella,
Steven Frank, Jeff Nelson, Jason Purdy, Achut Reddy,
John Tabbone, Joe S. Valley, Alan Williamson

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department

Subscription Hotline: 800 513-7111

Cover Price: $4.95/issue.
Domestic: $49/yr. (12 issues) *Canada/Mexico:* $69/yr.
Overseas: Basic subscription price plus air-mail postage
(U.S. Banks or Money Orders).*Back Issues:* $12 each

Publisher, President and CEO: Fuat A. Kircaali
Vice President, Production: Jim Morgan
Vice President, Marketing: Carmen Gonzalez
Advertising Manager: Claudia Jung
Advertising Sales: Diane Baird
Advertising Assistant: Erin O'Gorman
Accounting: Jennifer Patterson
Senior Designer: Robin Groves
Web Master: Robert Diamond
Web Designer: Corey Low
Customer Service: Rae Miranda
Sian O'Gorman
Mitchell Low

EDITORIAL OFFICES

SYS-CON Publications, Inc.
39 E. Central Ave., Pearl River, NY 10965
Telephone: 914 735-1900 Fax: 914 735-3922
Subscribe@SYS-CON.com

JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is published monthly (12 times a year) for $49.00 by SYS-CON Publications, Inc., 39 E. Central Ave., Pearl River, NY 10965-2306. Application to mail at Periodicals Postage rates is pending at Pearl River, NY 10965 *and additional mailing offices.*
POSTMASTER: Send address changes to:
JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc.,
39 E. Central Ave., Pearl River, NY 10965-2306.

ISSN # 1087-6944

DISTRIBUTED in USA by
International Periodical Distributors
674 Via De La Valle, Suite 204, Solana Beach, CA 92075 619 481-5928

BPA Membership Applied For August, 1996

FROM THE INDUSTRY

Tom Flynn

More Optimism for Distributed Computing

I was introduced to Java in the Fall of 1995 when most of the industry viewed Java as a language to be used for developing applets. At the time, several colleagues of mine were looking at Java with great excitement but not as an applet development platform. These were developers working to bring distributed computing solutions to the average corporate IS organization. Unfortunately, things were not going well. It was painfully obvious that distributed computing was just too difficult, and much of that difficulty revolved around development of business logic in C or C++. Java offered the promise of a development language that was easier to learn than C or C++ yet robust enough to be used for complex business applications. Today, that same group of developers that introduced me to Java is still developing distributed applications; however, they are a lot more optimistic about the future of distributed computing. Java is proving to be an outstanding language for business rules development in distributed systems.

The typical IS organization is not in a position to develop multi-tier applications that require significant expertise with C or C++. Given the current shortage of developers, most organizations are looking to train existing employees in the latest technologies. From this perspective, Java is a much better choice for large organizations as it is easier to learn than C. That does not make it simple. As most of us know, you still need a background in object-oriented principles to be successful with Java. There has been much talk of how easily C developers pick up Java. As someone who has been there, I can say that client/server developers skilled in PowerBuilder or Visual Basic can also overcome the Java learning curve in record time.

Performance is a common concern when considering the use of Java for middle-tier business logic. Those of us who have been promoting the use of Java over the past 24 months are relieved to see legitimate progress in the area of JVM performance. In particular, there have been some interesting benchmarks showing Just-In-Time compilers performing comparable to C++. I would recommend taking a look at some of the more recent benchmarks, which provide concrete evidence that the performance argument against Java is weakening. Some of the greatest challenges in distributed development are integration issues. This involves making disparate software such as databases, middleware and legacy applications all work together and communicate with one another. While C is still a sound choice in this regard, Java is rapidly gaining ground as all software vendors work on some type of Java integration. Java APIs provide the same type of integration that C developers enjoy when integrating application logic with third party products. Where Java APIs are not available, the Java Native Interface provides an alternative for leveraging C APIs from within Java code.

Finally, there is Java's promise of platform independence. I say promise because we are still seeing numerous inconsistencies across JVMs. Java's "write once, run anywhere" message is much more relevant on the server than it is on the clients. Anyone who has been involved in the development of a distributed application knows that you can encounter significant problems when attempting to migrate code from one server operating system to another. The ability to run Java classes on any platform's JVM will greatly reduce the aggravation that exists when integrating products within a distributed application.

Distributed computing must become accessible to the average developer if it is to achieve widespread success. The emergence of Web development and e-commerce adds a sense of urgency to the distributed computing problem. Java stands the best chance of being the answer to this problem largely due to the industry-wide support it enjoys. With recent estimates indicating the number of Java developers at 750,000 with 1,000 new developers being added daily, it seems unlikely that such momentum can be stopped.

About the Author

Tom Flynn is the Product Marketing Manager for Greenbrier & Russel. He spends much of his time working on the technical marketing aspects of Greenbrier & Russel's Java application server, Jamaica. Tom can be reached at tflynn@gr.com.

SYS-CON PUBLICATIONS

PHONE, ADDRESS & WEB DIRECTORY

CALL FOR SUBSCRIPTIONS

1 800 513-7111

International Subscriptions & Customer Service Inquiries

914 735-1900

or by fax: 914 735-3922

E-Mail: Subscribe@SYS-CON.com
http://www.SYS-CON.com

MAIL All Subscription Orders or Customer Service Inquiries to

JAVA DEVELOPER'S JOURNAL

Java Developer's Journal
http://www.JavaDevelopersJournal.com

VRML DEVELOPER'S JOURNAL

VRML Developer's Journal
VRMLJournal.com

National JAVA LEARNING CENTER

National Java Learning Center, Inc.

JAVA DEVELOPER'S JOURNAL 1998 JAVA Buyer's Guide

JDJ Buyer's Guide
JavaBuyersGuide.com

WEB-PRO DEVELOPER'S SUPPLEMENT

Web-Pro Developer's Supplement

SYS-CON Publications, Inc.
39 E. Central Ave.
Pearl River, NY 10965 – USA

EDITORIAL OFFICES
Phone: 914 735-1900
Fax: 914 735-3922

ADVERTISING & SALES OFFICE
Phone: 914 735-0300
Fax: 914 735-7302

CUSTOMER SERVICE
Phone: 914 735-1900
Fax: 914 735-3922

DESIGN & PRODUCTION
Phone: 914 735-7300
Fax: 914 735-6547

DISTRIBUTED in the USA by
International Periodical Distributors
674 Via De La Valle, Suite: 204
Solana Beach, CA 92075
Phone: 619 481-5928

Worldwide Distribution by
Curtis Circulation Company
739 River Road,
New Milford NJ 07646-3048
Phone: 201 634-7400

GUEST EDITORIAL

Jason Purdy

Where Do We Go from Here?

It's a funny thing, public opinion and the media and how they determine the direction of so many things. Presidential elections, foreign policy, the stock market, and yes, even Java. Java has survived 1997, but its coat doesn't have the same shine and sheen that it enjoyed in 1995, where it was the latest buzzword and managers were looking for Java developers with at least five years of experience. It's 1998 and in order for Java to make it into the 21st century, it needs a new infusion of both marketing and material.

Third party component vendors, such as Stingray Software and Rogue Wave, will continue to play a major role in Java's evolution. By filling in the gaps of the JDK, component vendors allow developers to go farther, pushing Java into a more serious contender as the language of choice. While the component vendor role is high in importance (even higher as Java is still a relatively new language), JavaSoft needs to recognize and encourage the vendors' efforts even more than has been done to date. (One can look to the well-established Visual Basic component marketplace to see how Microsoft started a cottage industry through outstanding developer relations.) Without establishing a mature developer relations group, many third-party vendors come and go and, in general, create a shaky market. JavaSoft needs to recognize their role and offer more support for those who are trying to bring Java to the next phase.

Another issue that is hurting the adoption of Java is the deployment of APIs and JDKs from Sun. It seems not a month goes by without an announcement of a new API set, whether it's something outstanding like the JavaBeans and JFC API or something obscure like the Java Input Method and Java Advanced Imaging API. It's as if Sun regards these APIs as spaghetti noodles, throwing them against the wall and seeing what sticks. The fallen noodles are determined by the Java community response and, as a fallen noodle, the time from announcement to deployment is extreme, either taking over a year or totally dropped. This causes a lot of uncertainty within the Java community and also hinders Java's growth.

After 2+ years of developing client-side components, the most daunting issue facing Java developers today is the QA issue. One of the main reasons that Java took off (if not the main reason) was that you could write a program in Java and then it would run on any computer that had the JVM installed. The impact of this statement is so overwhelming, it almost deserves being repeated. However, with the advent of platform-specific and vendor-specific JVMs (meaning a vendor, once licensed, could build a JVM following certain specifications), the "Write Once, Run Anywhere" slogan has changed to "Write Once, Test Everywhere." A typical QA engineer working on a Java client applet/application to be tested in multiple environments now has the non-enviable task of testing under multiple operating systems, each of which might have multiple JVMs under them (with different versions of those JVMs too!). While we've consistently been told that Java is a young language and that things will get better, two years has proven otherwise. While technologies such as Project Java Activator might help in the future, not every Java developer will be able to assume that their target audience will conveniently have (or be allowed to have) this technology enabled.

It is as Charles Dickens wrote, "It was the best of times, it was the worst of times." These are the times that will try Java developers' resolve. 1998 will be a pivotal year for Java. It is up to Sun and Java developers everywhere to deliver the killer apps and tools that will maintain Java's name and carry it into the 21st century. Otherwise, Java will become a fond memory and a display at the Smithsonian detailing how Java started with a bang, but ended in a murmur.

About the Author

Jason Purdy is the Java Evangelist for Stingray Software, the leading provider of C++ and Java development tools. He can be reached at jason@stingray.com.

PersonalJava and Inferno for Today's Consumer Devices

Opening the possibilities for low-cost consumer devices

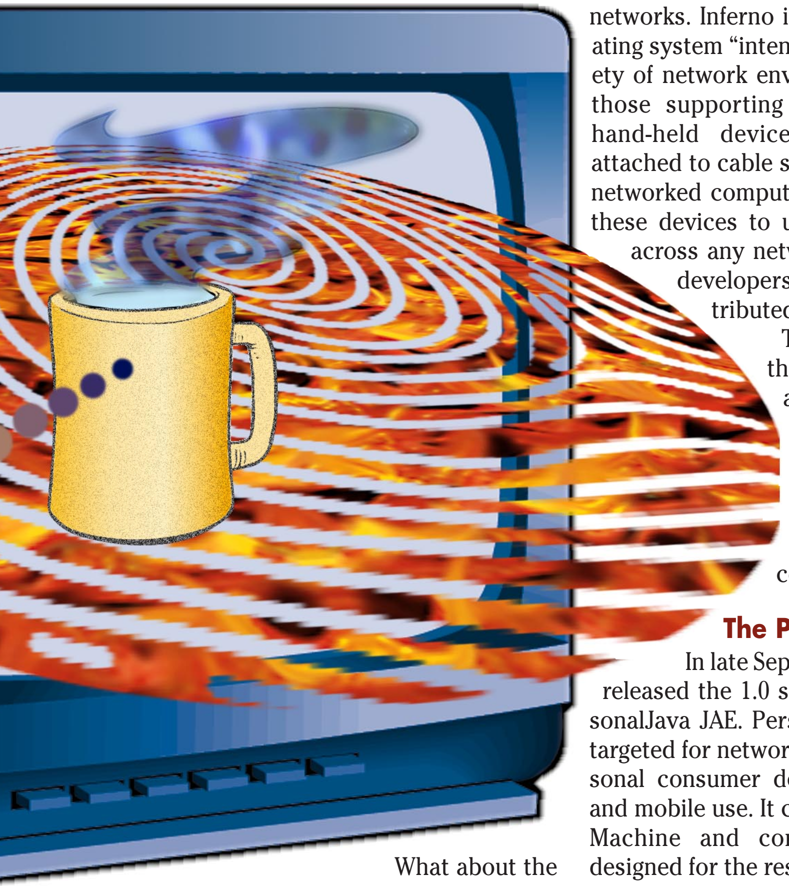
by **Steven Frank**

To some they are "gadgets." To others they are "gizmos." To many, gadgets and gizmos are nothing more than big-boys' expensive toys. Nevertheless, "toys" not unlike Dick Tracy's Wrist Communicator, or James Bond's combination cellular phone/fingerprint analyzer/car remote control/security scanner, are no longer only in Hollywood prop rooms or sci-fi writers' imaginations.

Yes, we are beginning to see an emergence of a wide range of gadgets and gizmos, *personal consumer devices* or *information appliances* as they are often called. Today's personal consumer devices, with the declining cost of electronic components and stiff competition in the marketplace, promise to be in the realm of affordability to the masses. Compared to the cost of desktop computers, some consumer devices are very inexpensive. We're talking about hand-held personal digital assistants, set-top boxes, game consoles and smart phones. Some manufacturers have already introduced products or prototypes of these devices with street prices in the range of a few hundred dollars.

Consumer devices introduce a new computing model with the potential for being distinctly different from the traditional desktop computing model that we've grown accustomed to. This new computer model presents many new opportunities and challenges for today's application developers. The target devices for our applications have many special requirements.

For example, their screens, or displays, can be nothing more than a small LCD panel capable of displaying only a few lines of text. Or they can be as big as your TV set. Or somewhere in between.



What about the user interface? Does the device have a mouse? A touch-screen? An IR remote? A keyboard?

What about the application environment? The development tools? The operating system?

These are just simple examples of what an application developer must take into consideration when designing and developing an application for consumer devices. A combination of two of the most important technologies introduced in the past few years helps you, the application developer, address these challenges as real opportunities.

Introduction to PersonalJava and Inferno

The minds at Sun Microsystems – the ones responsible for the Java revolution – have introduced a Java Application Environment (JAE) designed for personal consumer device applications.

"PersonalJava" is a Java API and Java Application Environment for networked applications running on personal consumer devices.[1] PersonalJava enables these devices to run Java programs, giving developers a direct link to providing applications for these devices.

The innovators at Lucent Technologies' Bell Labs – the same Labs that created Unix, C and C++ – are also providing technology for enabling distributed, network applications to run across multiple platforms and networks. Inferno is a small network operating system "intended to be used in a variety of network environments; for example those supporting advanced telephones, hand-held devices, TV set-top boxes attached to cable systems and inexpensive networked computers."[2] Inferno enables these devices to use resources scattered across any network, giving application developers the ability to create distributed applications.

The combination of these technologies gives application developers the necessary tools for designing, developing and deploying sophisticated, intelligent, usable real-world consumer devices.

The PersonalJava JAE

In late September of 1997, JavaSoft released the 1.0 specification for the PersonalJava JAE. PersonalJava is specifically targeted for networked applications on personal consumer devices for home, office and mobile use. It consists of a Java Virtual Machine and core and optional APIs designed for the resource-constrained environments and special requirements of consumer device applications. It can be considered a subset of Java in the sense that PersonalJava applications are upward-compatible to standard Java, defined by the JDK. On the other hand, it is also a superset of Java since it includes features, such as additional APIs, that are common to consumer devices.

PersonalJava is designed to benefit application developers in the following ways:

- PersonalJava-based applets also run in a JDK 1.1 applet environment and are compatible with JDK 1.1 packages supported by PersonalJava. This gives developers the ability to write applets for PersonalJava that can use the JDK 1.1 applet features when running in a JDK 1.1 applet environment.
- The PersonalJava environment can also run a large proportion of the applets written for the JDK 1.0.2 and the JDK 1.1.
- PersonalJava can adapt to the many input and output mechanisms used by consumer devices, including remote controls, television output and touch screens.
- PersonalJava requires less memory than the JDK 1.1 in terms of the Java system itself and of runtime requirements. The PersonalJava Virtual Machine and class libraries are designed to fit within 2 megabytes of ROM and execute in 1-2 megabytes of RAM. This makes PersonalJava highly scalable and configurable while using minimal memory.
- PersonalJava reduces time-to-market for consumer devices with the portability, flexibility and code reuse attributes inherent to Java.
- In the relatively short time that the PersonalJava specification has been publicly available, there are already a number of manufacturers and software developers using it for products such as hand-held computers, set-top boxes, game consoles, mobile devices and smart phones.

The Inferno Network Operating System

Even before Inferno's first commercial release in March of 1997, it was compared to and seen as a rival to Java. The two technologies do address some of the same issues, both in similar and distinctly different ways, but they are really not competitors[3]. There are, in fact, many aspects of Java that find complements within Inferno. Initially, Java was designed for the very market where Inferno is finding itself, that of small consumer appliances and devices.

Inferno is a complete, full-service operating system that includes a real-time kernel, a concurrent programming language (Limbo), a virtual machine (Dis) and a communication protocol (Styx). It provides many important features expected in an operating system designed for applications running on a wide range of consumer devices. Inferno includes built-in security, network independence, multithreading and multi-processor support, heap and stack management, interpreted and JIT (just-in-time) compilation, graphics libraries and support for IR input devices.

Inferno's definitive strength lies in its portability and versatility across several dimensions:

- Portability across processors: It currently runs on Intel, SPARC, MIPS, ARM, StrongARM, HP-PA and AMD 29K architectures and is being ported to others. This gives an application programmer the confidence that an application written for

Inferno will run on the popular architectures being used in consumer devices.

- Portability across environments: It runs as a native operating system on small devices and also as a user application (in Emulation mode) under Windows 95/NT and Solaris. In all of these environments, Inferno applications see an identical interface.
- Distributed design: An application can establish the identical environment at the client and at the server and each may use the resources of the other. Through the communications facilities of Inferno's runtime system, applications can be designed to be dynamically distributed between client and server, multiple servers or even multiple clients.
- Minimal hardware requirements: It runs complete applications on machines with as little as 1 MB of memory and does not require memory-mapping hardware.
- Dynamic adaptability: Applications may, depending on the hardware or other resources available, load different program modules to perform a specific function. For example, a video player application might use any of several different decoder modules.

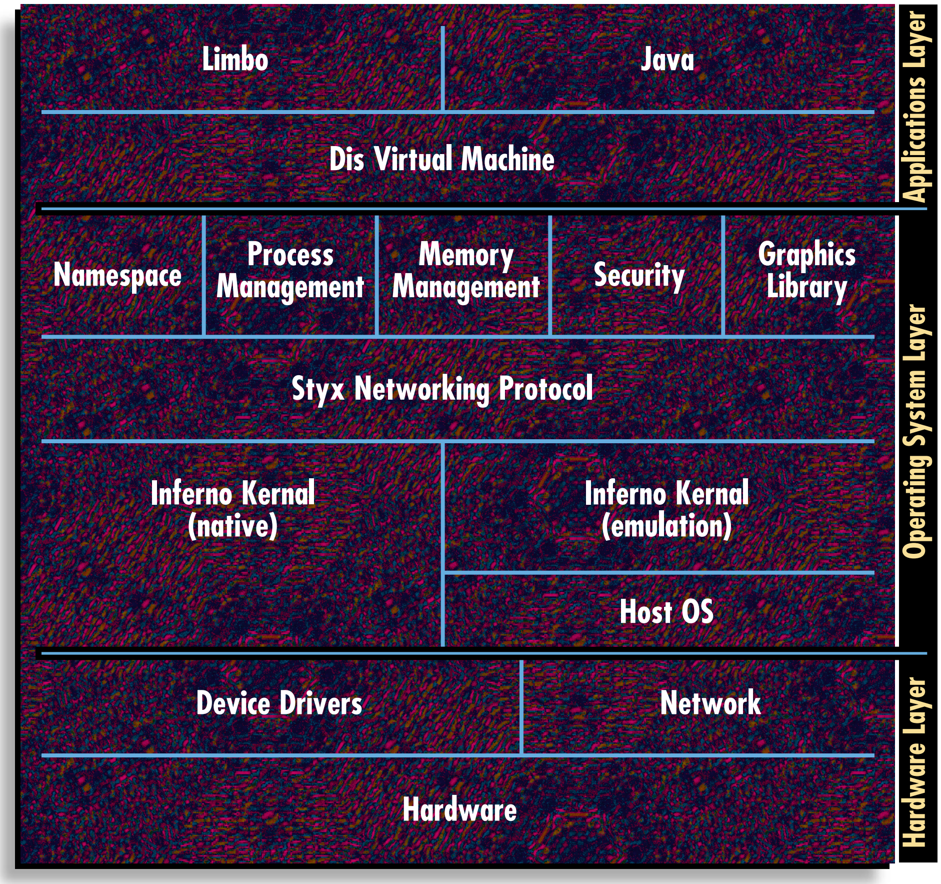


Figure 1. General Inferno architecture

The purpose of most Inferno applications is to present information or media to the user; thus, applications must locate the information sources in the network and construct a local representation of them. Inferno is designed to isolate the mess of network hardware and protocols from the application programmer. Inferno's design has three principles:

1. Most resources are named and accessed like files in hierarchical file systems.
2. The disjoint resource hierarchies provided by different services are joined together into a single, private, hierarchical namespace.
3. The Styx communication protocol is used by the system to access these resources, whether local or remote. This gives an application uniform acccess to resources, whether local or remote.

Many service providers are looking aggressively to expand their reaches by providing a vast array of services *and* media. This includes the giant industries of telecom, Internet and cable providers. In this arena, however, there are diverse network hardware and protocols, including POTS, ISDN, ASDL, ATM, broadcast TV, cable TV, digital TV, satellite, the Internet, etc. The destination of the media and services that must traverse this mélange of acronyms is the variety of consumer devices we have been talking about.

In less than a year since its first commercial release, Inferno has garnered partnerships with a large number of OEMs and ISVs. These partners are using Inferno in a variety of consumer devices, including smart phones, set-top boxes, game consoles and mobile devices, as well as a number of other non-consumer devices such as network elements and servers.

The Vices of Consumer Devices

As we've already discussed, consumer devices present many new opportunities and challenges for today's application developers. Some of the most significant of these challenges are memory and display constraints and the varying types of input and output sources.

Where's the Memory?

Consumer devices typically do not have the large amounts of RAM or ROM required by most of the software that we run on our desktop systems. It is not unreasonable for a device to have at most 1MB of RAM and 1MB of ROM available for the operating system, the applications *and* user data.

You Call <u>That</u> a Screen?

Displays vary significantly from one type of device to another and they rarely have the capabilities of a desktop's SVGA monitor. It doesn't seem all that long ago that a laptop's display looked more like a washed-out watercolor painting than a computer's screen. We've seen some tremendous advances in the quality of smaller displays in just the last couple of years. With that, though, comes our insatiable thirst to make them even smaller. How many clowns can you fit into a Volkswagen Beetle? Probably more than the number of pixels that fit on the display of some small consumer devices.

No Mouse? No Keyboard? What Do I Do?

The method by which the user will interact and interface with the device and the applications may not be via a mouse and keyboard; the device may have a touch screen, IR remote control, some kind of small keypad or voice recognition.

The PersonalJava and Inferno Solution

The combination of PersonalJava and Inferno solves many of the problems developers face when developing applications for consumer devices.

Lean and Mean

Amid the many praises of Java, one of the loud complaints has been its rather large memory requirements. Most typical JDK 1.1 implementations require 8, 16 or more MB of memory. PersonalJava, however, was designed with the more modest memory requirements befitting a consumer device. The intent, as stated in the specification, is that the Java Virtual Machine and class libraries fit in 2 MB of ROM and exe-

cute in 1-2MB of RAM.

This does not take into account the operating system or other system requirements. The Inferno operating system is very small and itself requires less than 1MB. So, in practice, it seems that 4MB of total memory is the lowest possible and 6 to 8MB would be recommended for most applications.

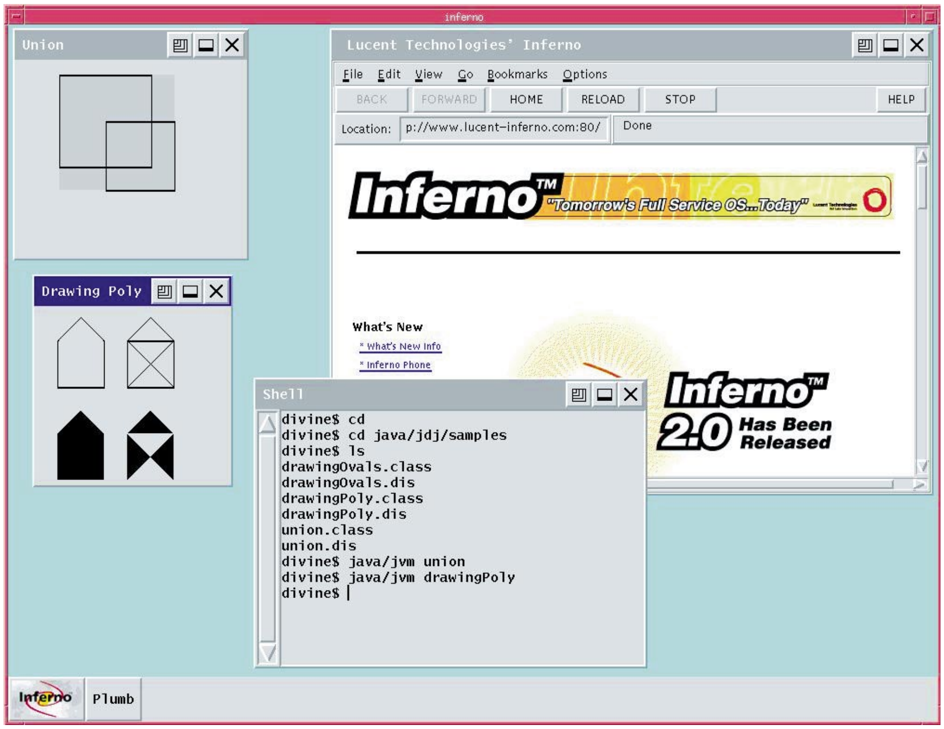
Although recently we have seen a dramatic drop in the price of the memory typically used in consumer devices, the fact remains that the less memory that is required, the cheaper the device can be manufactured. This is especially evident when you consider that some of these devices will be manufactured in relatively large quantities. For example, if a manufacturer can shave $20 off the cost to make a single set-top box by limiting the amount of memory, that translates into $20,000,000 savings in the production of 1,000,000 units. They must walk a fine line when determining how much memory a device needs to be useful verses the cost of production.

Granted, most of us don't care much about the manufacturing costs of millions of units. What we do care about is the hardware specs of the device that we want to run our application on. It may mean that we have to approach writing our applications very differently if the device has 1MB versus 4MB of memory.

Inferno, in particular its virtual machine (named Dis), was designed with the flexibility of providing a multi-language programming environment. Applications written in Inferno's native programming language, Limbo, can run simultaneously with PersonalJava applications.

One of Limbo's many features is portability[4]. Limbo programs, like Java programs, are compiled to a hardware-independent binary image (bytecode) for execution in Inferno's virtual machine. Like Java, Limbo can be interpreted or compiled on the fly for portable execution, use garbage collection for memory management and support the Unicode character set. For Limbo, though, concurrency and inter-task communication are intrinsic components to the language and virtual machine.

Limbo is a procedural language that uses the concept of "modules" with separate interfaces and implementations. A typical Limbo application is composed of a set of dynamically interacting modules. The flexibility of the module model contributes to the compactness, efficiency and adaptability of Limbo applications.


inferno
Union
Drawing Poly
Lucent Technologies' Inferno
File Edit View Go Bookmarks Options
BACK FORWARD HOME RELOAD STOP HELP
Location: p://www.lucent-inferno.com:80/ Done
Inferno™ "Tomorrow's Full Service OS...Today"
What's New
* What's New Info
* Inferno Phone
Inferno™ 2.0 Has Been Released
Shell
Inferno Plumb

```
divine$ cd
divine$ cd java/jdj/samples
divine$ ls
drawingOvals.class
drawingOvals.dis
drawingPoly.class
drawingPoly.dis
union.class
union.dis
divine$ java/jvm union
divine$ java/jvm drawingPoly
divine$ |
```

Figure 2. Inferno on a Solaris workstation running two Java apps (union and drawingPoly), as well as a Limbo Web browser.

Figure 3. Inferno running in a set-top box environment.

Size Doesn't Matter

Small screens do not mean that we are returning to the world of pre-GUI days. Users still expect and demand the aesthetically pleasing, graphical point-and-click interfaces. For some devices, however, this just won't be practical. Another JAE, EmbeddedJava™, is better suited for devices with even greater constraints.

PersonalJava addresses this issue by modifying some of the API packages, in particular java.awt. For example, the `Component.setCursor()` method may ignore the specified cursor since some platforms may not support cursors and others may limit the types of cursors displayed for usability reasons.

It may not make sense on some devices to have a general-purpose window manager that allows a user to manipulate overlapping windows, such as resize and move. PersonalJava implementations might not support the creation of a `Frame` or `Window` beyond the "root" window.

Additionally, PersonalJava provides a special API for performing double buffering

Personal Java

PersonalJava™ is a new Java Application Environment for network-connectable applications on personal consumer devices for home, office and mobile use. It is designed specifically for resource limited environments.

PersonalJava is designed to be highly scalable and configurable while requiring minimal system resources – a key factor in embedded software applications. It is targeted at developers writing applications for consumers who may not be computer-savvy. PersonalJava supports robust user interfaces and the downloading and execution of applets. These applications, like the JavaCard, are upward-compatible and will run on "enterprise" Java platforms.

Inferno

Inferno™ is an operating system for creating distributed services and is intended for use in a variety of network environments, including hand-held devices, set top boxes and inexpensive networked computers as well as traditional computing systems. It creates a standard environment for applications and identical application programs can run under any instance of this environment, even in distributed fashion, and see the same resources. The system architecture is constrained only by the desired markets and available interconnection and server technologies, not by the software.

(for platforms that support it) to make painting and updating graphics appear smoother. There are ways to accomplish this by explicitly creating an off-screen image, drawing into it and then using the `drawImage` method to display it. However, since this can take large amounts of memory, it may not work on devices with limited memory. (Since this can have benefits beyond just consumer devices, it is likely that it will be included in a future version of the JDK.)

PersonalJava provides a new method, `isDoubleBuffered`, for class `Component`:

```
public boolean isDoubleBuffered();
```

A return value of true indicates all drawing performed by the paint and update methods are double buffered.

Through Inferno's namespace, where all application resources are represented as a hierarchical file system, even devices such as the display device (console) can be imported and exported across the network. They can be dynamically loaded and unloaded, depending on the application's needs.

"Look, Ma! No Mouse!"

For the last 30 years or so, a computer has been synonymous with a keyboard of some type. However, many of today's consumer devices don't have a keyboard or if they do, it's some convoluted mixture of shift/control keys.

In conjunction with traditional GUIs, some kind of pointing device is generally required, whether it's a mouse, trackball or pen. Yet it isn't too likely that we'll see a mouse port for a cellular phone. In mouseless environments, the users typically can navigate from one on-screen component to another using keys or buttons on the device. For example, after navigating to an on-screen button component, the user might press the Go key on a remote control to indicate that the on-screen button is to be "pressed."

PersonalJava provides additional APIs for specifying ways of interfacing with components in a mouseless environment The API includes four interfaces in the java.awt package that allow developers to make it easier to adapt to mouseless environments:

```
public interface NoInputPreferred {}
public interface KeyboardInputPreferred {}
public interface ActionInputPreferred {}
public interface PositionalInputPreferred {}
```

Inferno enables an application to dynamically load the support modules for the input device in use. For example, when an application is running on a set-top box that uses a remote control, it can load the IR interface module; when it is running on a desktop or a system that uses a keyboard for input, it can load a keyboard IR simulation module.

Summary

So, whether they are just "gadgets" or "gizmos" to you, the emerging market of consumer devices presents many exciting opportunities for application developers. With these opportunities, we must take into consideration their distinct, if not peculiar, application requirements.

The PersonalJava JAE is sure to put the Java trademark on a wide range of consumer devices. Its design gives application developers the ability to add another verse to the "Write Once, Run Anywhere" mantra.

PersonalJava applications running in Inferno can take advantage of Inferno's extensive features, such as security and network independence. Through Inferno's native programming language, Limbo, application developers can easily tap into the advanced low-level features of Inferno.

The strengths of Inferno and PersonalJava can open up the possibilities for application developers to tap into the growing and lucrative market of lower-cost consumer devices. These applications can give users easy access to a world of remote information and resources.

Resources

1. PersonalJava 1.0 Specification. Sun Microsystems, Inc. http://www.javasoft.com/products/personaljava/spec-1-0-0/personalJavaSpec.html
2. Inferno: la Commedia Interattiva. Lucent Technologies, Inc. http://inferno.lucent.com/inferno/infernosum.html
3. ***Java Developer's Journal***. "The Developers of Inferno." Volume 1, Issue 4.
4. *Inferno Programmer's Guide*. Lucent Technologies, Inc. Murray Hill, NJ. 1997.

About the Author

Steven P. Frank is a member of the technical staff at Lucent Technologies/Bell Labs in Murray Hill, NJ, working in the Inferno venture group for the past year. Most recently, he has been working with the Java Integration team responsible for PersonalJava support in Inferno. Steven can be reached via e-mail at spf@bell-labs.com

spf@bell-labs.com

Clockwise from top left:
Bertrand du Castel heads the team that invented the first Smart Card; Marc Valderrama has been on the Smart Card team for 2 years; Hervé Garcia develops hardware interfaces and cryptography for the Smart Card; Pam Saegert helps explain the card to the world; Mike Montgomery works on next-generation Java Card products; Neville Pattinson is Product Manager for the Motorola "Solo" License, and Ksheerabdhi Krishna focuses on software development and Java Card technology.

JDJ FEATURE

The World in Your Wallet

Who wouldn't be attracted to a Smart Card that supports Java?
Schlumberger
Cyberflex,
the card for Java

Schlumberger's Cyberflex Java Smart Card

An interview with Tom Lebsack, head of Marketing for Java card

by **Scott Davison**

JDJ: For those readers who aren't familiar with Schlumberger Electronic Transactions, could you please give us a short history of your corporate background and structure plus your own responsibilities?

TL: Schlumberger has been in business since the late 1920s. The company was founded based on a new technology developed by Conrad and Marcel Schlumberger, two French brothers who invented a method to determine, through electrical measurements, the presence of oil and gas in subsurface rock formations. The technique, which became known as well logging, utilizes sophisticated instrumentation lowered into wells during and just after drilling; it's used in virtually all wells drilled today.

From this start, the company grew to be the largest oilfield services company in the world, providing well logging, testing, pumping, drilling and seismic services to the oil and gas industry. A major part of the business involves data collection, transmission and interpretation on a worldwide basis. New technologies have always been a major force in the success of the company.

Other divisions of the company include Schlumberger ATE, which is a leading supplier of manufacturing and test equipment to the semiconductor industry.

The parent company, Schlumberger Ltd, is headquartered in New York and Paris, and has about 60,000 people operating in over 100 countries around the world. Last year's revenue was over $10 billion.

Schlumberger's smart card activities are part of our Electronic Transactions business unit. We got into the smart card business in the late 1970s, investing in the development of this new chip card technology in France. The first commercial chip cards were memory cards used in the French telephone industry. Those of your readers who have been to Europe, particularly France, know that you can use a chip card to make calls from public pay phones. From that beginning, the technology was adopted by the French banks, and spread to other banks and telecom operators in Europe and elsewhere that found chip cards ideally suited for secure financial transactions. Schlumberger has played a role in the development of the technology to adapt to the new market segments, including the financial applications like credit and ATM cards, health care cards and digital wireless communications – the GSM (Global Standard for Mobile Communications) cell phones that are used around the world. These telephone applications have been the fastest growing area for smart cards recently. If your readers outside North America have a cell phone, there's probably a 90% chance that it has a smart card inside.

Today Schlumberger is a leading manufacturer of chip cards, and the largest manufacturer of secure cards, chip and mag stripe. We have ten plants in France, Spain, the United Kingdom, United States, Mexico and China producing over a million chip cards every day.

As for myself, I am the Director of Schlumberger's Information Security and Multimedia business segment. I started in oilfield services as a field engineer doing oil well logging, and later moved into marketing and management positions in the company. I moved over to Electronic Transactions four years ago and am now directing our Java card marketing activities worldwide.

> A smart card is a plastic card with a computer chip embedded in it. It enables technology for many new applications due to its unique features: computational power; portable data storage; and high security.
>
> The Schlumberger Electronic Transactions group of Schlumberger, Ltd. supplies cards, terminals and management systems across the entire range of magnetic and chip card applications. Cyberflex, their smart card, runs programs written in Java™ that enable the creation of new smart card programs for consumer businesses, financial institutions and government services agaences that adopt this technology.

Who wouldn't be attracted to a Smart Card that supports Java?
Schlumberger
Cyberflex, the card for Java

JDJ: What type of market do Cyberflex cards have now?

TL: Cyberflex is a Java card, a smart card with a Java Virtual Machine. Its invention has energized the card industry and is the major focus for new developments by chip manufacturers and card suppliers around the world. All of the major players in the financial and telecommunications industries, the two largest smart card markets, have strategies that involve Java cards. One very significant example is Visa's Open Platform project, now under development. It is a Java-based multiple application card tailored to the needs of the Visa issuing banks, and they expect to be in trials this year. That said, the market for Cyberflex cards is in the development stage, the technology having been invented less than 1-1/2 years ago. We have been selling Cyberflex Development Kits by the hundreds since the middle of last year, and many of your readers have been able to develop their own smart card applications and try them out on real smart cards, something they could not do before. We expect to see some announcements of major Java card programs this year.

Our studies indicate that Java cards will command a significant share of the growing market for microprocessor smart cards, particularly where multiple application cards are required and where companies need the capability to quickly introduce card programs and update or modify the smart card applications after the cards are issued. The prime markets are financial, retail (such as loyalty and frequency marketing applications), telecommunications, information security, health care and travel and leisure. Card issuers, merchants and operators in these segments are looking for ways to introduce quickly differentiated products and services, often with partners. These markets are international. Cyberflex meets these needs perfectly since it features inherent security between applications and fast development and deployment times.

JDJ: Could you describe the impact that the Cyberflex cards are having as a result of their having the Java Virtual Machine and operating system?

TL: Cyberflex is revolutionizing the smart card industry. Schlumberger's objectives in developing the technology were to expand the market for smart cards. We felt that we could do this by introducing technology that would improve the business cases for smart card programs, first by reducing the time-to-market for smart card-based products and services, then by enabling secure multiple application cards to bring opportunities for co-branding with partners. We also felt that it was time, after almost twenty years, that the smart card industry entered the world of mainstream computing by allowing developers to program cards with open, industry-standard languages and tools. Since our initial Cyberflex announcement in October 1996, just after Sun's Java Card announcement, industry response has been overwhelming.

One way to understand the impact is by comparing conventional cards to Java cards in the areas of development and distribution. Conventional microprocessor smart cards have a CPU, RAM, ROM and EEPROM memories for native functions, application programs and data. The operating system and application programs are hard-masked into ROM at the time of manufacture, and the EEPROM is rewritable memory used for data. This architecture has allowed the industry to maximize the functionality of a card even when faced with very limited memory and CPU resources, a few KILO bytes of ROM and often less than one K byte of EEPROM in the early days. But the limitations have not been so bad when you consider that smart cards have been used as peripheral devices, with no GUI or overly complex I/O involved.

The conventional architecture has had an unfortunate result, though. Applications for conventional cards are chip-specific and not at all portable. If you developed your application on a particular smart card and later wanted to move it to a similar one from another manufacturer, you had to change the application to match that company's operating system. This has been one of the factors in limiting the growth of smart cards, lack of interoperability between cards at the API level.

Time-to-market is another concern. A new smart card program can take easily 9 months to a year to introduce, allowing for the development, masking and validation time required. You can't make too many mistakes and hope to have your application introduced on a card in any reasonable length of time. Fortunately, there are a few skilled professionals, mainly in the smart card companies, that are up to the task. But even if the code is perfect, there is still a chance that the features of the application might need to be changed after the cards have been introduced. This means another pass through the process and several months lost with new cards to manufacture and distribute.

A Java card is also a microprocessor card, with hardware that is similar or even identical to today's higher-end conventional cards. It has a CPU, RAM, ROM and EEPROM memories for native functions, application programs and data. The card operating system, virtual machine and interpreter are installed in ROM when the card is manufactured. Applications and data are stored in the rewritable non-volatile memory (EEPROM), and the RAM is used for instructions and data during operation.

The impact of this architecture on smart cards is enormous because they now start to look a lot like regular computers. With Cyberflex, the applications are developed in Java using standard tools and bytecodes are loaded onto the card. You don't have to be an expert in 6805 or 8051 assembly languages. You don't even have to know about them. What's more, if you change your mind while validating the application in a test or pilot program, or even after introducing hundreds of thousands of cards, the application can be modified and downloaded onto the cards.

JDJ: How do you distribute the changes to the cards?

TL: It depends on the type of application and the infrastructure that the card issuer has in place. If you are talking about a financial card, for example, the card issuing bank will have a system to download new applications

when the card is used at an ATM, for example, or at home through an online banking Web site. For a card issued by a GSM telecom operator, the application upgrade would be done over the air, right into your cell phone handset.

Security is extremely important. For a bank card, for example, neither you nor the card issuer would want unauthorized applications to get onto the card, so there are safeguards being built into the systems to keep rogue cardlets off cards. Java's security model is a major benefit too. In fact, it's the main reason that we chose Java initially over other high level languages. Java applications cannot interfere with other applications or data, and it has been tested a lot.

> "The most important thing is that, if you want to write smart card applications, you have to know what to do not only from a technical viewpoint, but from the business viewpoint."

JDJ: You've come out with a new upgrade, Cyberflex 2.0 Multi8K. How has this affected new development?

TL: Our previous Cyberflex version had a 4K EEPROM space and only about 2.8K of that was available. We were very limited in what we could do in single applications and there was little chance of loading multiple applications. 8K gives us a lot more room. In addition, there are improvements in the programming itself that reduce the amount of space that applications take up. We allow a couple of new methods to access the default processing on the card so that you don't have to write all of the code in the application. You just invoke a method to access those functions. It is a real improvement in terms of the space available from a physical point of view as well as from a logical point of view.

JDJ: How do the smart card terminals impact the Cyberflex development cycle?

TL: When we talk about a terminal, we are talking about a device that has an application running in it that requires a smart card for execution. A terminal can be anything from a pretty simple device, such as card readers that attach to PCs, all the way to a PC or an ATM. In between are the point-of-sale terminals used with cash registers and vending machine terminals.

Existing terminals can use Java cards since a Java card application can be developed that emulates exactly the existing smart card used with the terminals. This means that Java cards and conventional cards can co-exist in a system. To take advantage of the power of a Java card, though, one might want to use the terminal to interact with the card in a more innovative way; for example, dynamically loading applications.

Integrated development environments are coming that will allow the developer to work on the applications on both the terminal and card sides interactively, so that the applications can be tested as developed.

JDJ: Microsoft and Schlumberger have made some announcements in the past few months about smart cards and Windows 95 and Windows NT. How do these relate to Java Card?

TL: Microsoft and Schlumberger, along with several other companies, have worked together in the PC/SC Workgroup to establish specifications for smart card readers and smart cards themselves to interact with personal computers. This effort started in the middle of 1996 and the first version of the specifications was issued at the end of that year. Then, at the end of 1997, final specifications were released. In the meantime, all of the companies involved in PC/SC were developing products that would use the specifications. Microsoft announced their Smart Card SDK for Windows in September of last year, and Schlumberger announced their smart card and smart card reader drivers for that architecture.

PC/SC establishes the specifications for drivers that reside on the PC. While the specs are designed to be platform-neutral, Microsoft is the only company to implement them in their OS. The smart card drivers, called Smart Card Service Providers, allow the developer of a PC application that uses smart cards to avoid writing drivers for each variety of smart card that he or she wants. PC/SC standardizes the interface to the application above the Service Provider and therefore makes the development of smart card-enabled applications much easier.

Java Card and PC/SC are complementary. Schlumberger has actively supported PC/SC for some of the same reasons that we became involved with Java card. The smart card industry has a strong need to open itself up to the world of mainstream computing and standards for smart cards to meet their potential as a computing platform.

JDJ: Security is a major issue and cards with a cash capability are the biggest target. Have you found any need for additional security beyond the native Java security?

TL: The native Java security is wonderful because it isolates applications from each other. This is very important if money or sensitive information is involved, such as with financial applications, network access, or health care, among others. There are other areas that are addressed besides this aspect of security, too. For example, earlier we talked about loading applications securely onto the cards. Java cards will incorporate mechanisms to allow only signed applications to be loaded.

But in addition to application-level security, there are other security aspects that must be considered, an important one being user authentication. This is typically done with PIN codes – something you know – but, depending on the level of security required, higher levels of security may be warranted, such as biometrics – something you are. Smart cards are capable of supporting biometrics.

JDJ: What about shipping the Developer's Kit outside the US? Do you have any problems with the encryption algorithms being exported?

TL: The Cyberflex Development Kit is exportable outside the US, plus we fulfill orders from plants both in the US and France.

JDJ: What are the risks of losing a smart card if it has cash value?

TL: You're speaking of a smart card that has a stored value, or e-purse, application on it. The risks depend on the way the issuer has developed and implemented the program, and there are trade-offs involved. In principle, any stored value card can be PIN protected, but requiring the user to enter a PIN at the point of sale can slow down the transaction considerably. You would want to avoid this for low-dollar purchase, such as at soda vending machines and fast-food restaurants. With most the of the programs, such as Visa Cash, if you lose the card, you've effectively lost the cash. On the other hand, during stored value trials around the world, consumers report feeling an increased level of security since they do not expose cash at the point of sale, and initial fears about losing cards have not been a significant issue. Besides, the amount of cash that you carry on the card is up to the user. If you are not comfortable carrying $100, you can load $20 or $50 instead.

JDJ: What technical innovations do you see affecting smart cards in the future?

TL: The most important innovations will be in chip memory capacity and CPU performance. Java card, public key cryptography and GSM applications are putting demands on the existing devices and the chip suppliers are responding with faster chips with co-processors and bigger memories, with even smaller die sizes. The evolution will be similar to what we saw in personal computer technology in the 80s and early 90s. If you don't have the memory you need today, don't worry, it'll be there before you know it.

JDJ: Could you tell our readers about Schlumberger's Smart Village? What kind of market do you see for the kinds of smart card applications that you have described for the Smart Village?

TL: The Smart Village is Schlumberger's vision of the role of smart cards in the future, where these cards are used to help make people's lives more productive, convenient and safe. The vision builds on the three fundamental attributes that make smart cards ideal for many applications. They are secure, portable and personal. What does this vision mean practically today? It means that credit and ATM applications on a smart card are more secure than conventional cards. It means that an issuer can implement a stored value program because the security and convenience of the card make it practical. It means that health care information can be stored securely, so that only the cardholders and their health care providers can get access.

"...if you change your mind while validating the application in a test or pilot program... the application can be modified and downloaded onto the cards."

And in the near future, it means that people will be using multiple application cards in a variety of new and innovative ways, such as to pay for goods in person or over the Internet, to store and track frequent flier miles and electronic coupons at the supermarket, to authenticate themselves for single system sign-on to their company's Intranet, and so on.

As an example, a new smart card technology, contactless cards, is now being introduced that will bring a new level of convenience to people using urban mass transit systems. These cards operate much like a stored value card and replace the paper or coin tokens common in most transit systems today – convenient and secure.

The applications that are needed to implement the vision come from the minds of developers who understand their customer's businesses and who want to be part of the smart card computing wave.

The Smart Village also relates to the product offer from Schlumberger. The different divisions of Schlumberger Electronic Transactions supply smart cards, smart card readers for personal computers, terminals for the banking and retail industry, automatic vending terminals, pay telephones and transportation-related applications, and all of these devices accept smart cards.

JDJ: You advertise for developers who "could assume the responsibility for end to end development of Java cards on families of microprocessors." You listed a number of skills, but what skills do you see as necessary for developers who want to be involved in smart card technology?

TL: The ad you are talking about was one that we were running to attract people to work on our system software, the operating system. The software business and those skills are different from the ones that developers are going to need to write applications.

The most important thing is that if you want to write smart card applications, you have to know what to do not only from a technical viewpoint, but from the business viewpoint. You have to know how real people are going to use the application. People who can translate that business model into the design of the program and from that into code are in real demand.

JDJ: What training do you have available for smart card developers here and in Europe?

TL: Schlumberger provides training courses as part of a strategy to increase the usage of smart cards, providing people with technology, tools and training. Our Cyberflex training program is part of an overall training program that we sponsor for Schlumberger Business Associates and smart card developers. We have put on several Cyberflex classes, starting with three in Europe and one in the US since last fall. Another one is scheduled for the US this spring. Our Web site has the information (www.cyberflex.slb.com).

JDJ: What is the most interesting smart card application you have come across?

TL: It's hard to choose, but I think from a business point of view the GSM digital telephone application would get my vote. GSM is tailor-made for Java smart cards for the developer, the operator and the user. In the GSM world, they already have two-way communications between the smart cards and the servers. The present day SIM (Subscriber Identity Module) smart cards that are used in GSM phones are loaded with the user's account information and preferences. With a Java-based SIM card, telecom-based multiple applications become a reality, such as loyalty programs, network access or banking. And the applications can be downloaded onto the cards over the air. This has real power for differentiating services and offering new levels of convenience to consumers.

If I could choose another, I think network security is one of the most obvious applications for smart cards that there is. When companies want to implement public key security for their internal and external networks, there is really only one choice for storing keys and certificates: the smart card. It is the most secure place to store a private key, and it never needs to be exposed during digital signature or challenge/response sequences. The usual alternative to smart cards is to store them on hard drives or floppies and this is just unbelievable. After implementing a strong public key infrastructure, to leave the keys on a hard drive or floppy is like leaving them under the doormat. In the past year or so, smart cards with strong RSA cryptographic capabilities have come onto the market. Schlumberger brought out the Cryptoflex card at the end of 1996; it features 1024-bit digital signature and 4K EEPROM for key and certificate storage.

JDJ: Many of our readers are interested in the people behind the products. Could you tell us a little bit about your group? Some of the people we interview are risk takers. They not only start new businesses, but also like sports such as mountain biking, snow boarding and sky diving.

TL: We try to keep these guys from taking any physical risks at all. Just kidding. The smart card group in the Austin Product Center comprises a dozen people out of about three hundred professionals altogether. We are devoted to innovation in smart card technology, especially at the operating system and language levels. Our team provides support for Schlumberger's application development teams in Montrouge, France, San Jose and Hong Kong, as well as for the developers in other companies.

The smart card team has the best people we can find. In fact, we have a very high portion of the top software scientists and engineers in Schlumberger assigned to the team. Nationalities represented include American, French, Indian, British, Chinese and Vietnamese. It's really a world class team.

JDJ: Do you see a super card in the future that will replace your wallet and let you do everything from a single card?

TL: That's down the road a bit. The technology will be there before the business cases, probably. While having a super card that does everything has a lot of intellectual appeal, the ownership of the card and the control of the applications that go onto the card could get a little complicated. In the short term, I think that it will be the individual applications that catch hold and are put together on multi-app Java cards where it makes sense to do so. Which applications? Well, the best applications that really ignited the market for any computer platform you can name didn't come from the technology providers. For PCs, Visicalc and 1-2-3 came from designers and developers with a vision of how the new technology could best be used in ways not imagined before. I think the same will be true with smart cards, maybe even coming from readers of ***Java Developer's Journal***.

Developers and the Java Alliance

It's time for the major players to set an example for the small developers

by Rick Ross

A question I have been asking myself lately is whether high-profile alliances of large corporations are actually the best way to advance the technology initiatives that shape our development environment and the products we build and use. It's fine and well that Sun, IBM, Netscape, Oracle, Sybase, Novell and others are working with one another to be the technology leaders of this new Java platform that we all care about. But how much does this alliance actually benefit you and me, and what are its potential drawbacks?

There have been several alliances in the recent past which have either failed to reach their goals or completely fallen apart due to the competition that has historically set the tone for the relationships between these major corporations. Java is not shielded from this reality and one doesn't need to look hard to see some of the pressure points! In most of these major alliances each of the big companies expects the others to shoulder the burden of making the alliance successful, though all seem happy enough to share the credit. As we witness Apple and Netscape falling by the wayside in the Java alliance, we should be aware that others may be just as vulnerable. Oracle and Sybase both have taken significant hits in their market valuation during recent months, and it will be much harder for them to be strong partners in the Java alliance if their fundamental markets are endangered.

The real problem with the big Java alliance is that the activities of the alliance partners are granted far too much significance, and the meaningful advances of anyone not in the alliance are too easily undervalued. The press and public look at events like Netscape's statement that JavaGator is "on hold" and conclude that Java itself must also be "on hold." It is as if they presume that since Netscape cannot succeed at what it tries to do, then the rest of us must not be able to either! Likewise, Corel's failure to deliver on its over-hyped effort was similarly damaging.

> *"Some of the greatest innovations come from the most unexpected places..."*

The world places far too much significance on these moments. Let's just imagine that Netscape had 500 developers focused on JavaGator and that Corel had a similar number working on their Java office suite. We'll guess that a total of 1000 Java developers were involved in these two projects. Let's also conjecture that the estimates of 600,000 to one million Java developers are somewhat overstated. Instead, we'll postulate that there are only 100,000 Java developers around the world.

OK, fine! In this scenario the 1000 developers on the Netscape and Corel projects account for only ***one percent*** of the world's Java developers! Should their amazingly publicized failures really have been able to cast such a long shadow of doubt over the activities of the overwhelming majority of the rest of us? ***No way!***

The long term success or failure of Java will be measured by the efforts of tens of thousands of us, not just by the comparatively small groups represented by the alliance partners. I wish the world would stop looking at every move these big companies make as though it should give us all cause to reconsider our Java plans and directions. Their activities are significant, but there is a lot going on in the Java world beyond what the alliance is doing.

There's another level at which the highly publicized partnerships of major players should give us concern. Recently we heard about a partnership between Sun and cable-television giant, TCI. It appears that TCI will be adopting PersonalJava for as many as five million of its cable-TV set-top boxes. On the surface this is fantastic news for Java (and probably for TV viewers, too!). I searched for details, however, and as far as I can tell, this deal is great for those two big companies but it has yet to create opportunity for any of the rest of us. While five million boxes is virtually an "instant platform", I have not been able to find any useful information about how you or I can get our own Java innovations onto that new platform.

Then there is the "Motorola deal." Motorola has entered into a licensing agreement with Sun Microsystems, Inc. for the use and distribution of its full family of Java platform technologies. Additionally, both companies will cooperate to bring Motorola's embedded and communications expertise and technologies to the Java environment. Scott McNealy proclaimed this one as "the largest technology license agreement in the history of the Java platform." The power-punch of Java combined with Motorola's fantastic array of innovative technologies could truly reshape how we communicate.

It is essential, however, that these power players make sure to invite the rest of us to the party. Some of the greatest innovations come from the most unexpected places, and the true opportunity for Sun, Motorola and TCI is to create foundation technologies on which hundreds or thousands of innovative third-party solutions can be built. Java will shine its brightest in the near-future technology where wired and wireless smart devices are able to communicate with each other in a way that makes our lives richer, easier and more fun.

It will be a disaster if Sun, Motorola and TCI do not regard the significant example set by another technology partnership Motorola participated in: General Magic. Several of the mightiest firms in the world were involved in the General Magic alliance and that still was not enough to make it succeed. The problem was that they simply forgot to make sure to set a place at the table for developers who were not part of their alliance! They acted in a way that was typical of big companies doing big business deals in big partnerships with each other.

Java has remarkable strengths and a remarkably diverse foundation of support in the developer world. My hope is that the major industry players will set a good example for others to follow. I hope that Sun, IBM, Motorola, TCI and the rest will all make sure to have rich and open programs for third-party developers to help cultivate these emerging new platforms. Then the Java alliance will be truly complete. Developers are the key, and I hope we can all enjoy success working with the industry leaders.

About the Author

Rick Ross is the founder of the Java Lobby (www.Javalobby.org), which currently has more than 13,000 members. He can be reached at rick@javalobby.org.

Full pg

If you're interested in VRML development and virtual reality on the Web, where do you get information you can use?

INTRODUCING

VRML Developer's Journal

VRML Developer's Journal (VRDJ) is the first, and only, print publication devoted exclusively to VRML and the creation of virtual worlds on the World Wide Web. Working closely with the VRML Consortium, the organization charged with maintaining and extending the VRML specification, VRDJ brings unbiased, vendor-neutral information to help developers fully realize VRML's potential in creating Web-based presences and integrating other technologies such as Java, JavaScript and ActiveX to add functionality and richness to these projects.

VIRTUAL WORLDS ON THE WEB

U.S. $7.99 (Canada $6.99)

VRML DEVELOPER'S JOURNAL

VRMLDevelopersJournal.com

Volume: 1 Issue: 1

From the Consortium
Welcome to the Future
Neil Trevett
page 5

From the Publisher
A Look Ahead
Fuat Kircaali
page 7

Ask the Expert
"Horses for Courses"
Choosing the right channel for your content
by David Frerichs
page 34

Tutorial
VRML-Bär
How to animate your favorite diet
by Markus Roskothen
page 48

A Look Inside
Using V-Realm Builder
Using VRML in its native scene graph format
by Bernie Roehl
page 26

SYS-CON PUBLICATIONS

VRDJ Feature: VRML Humanoids
Setting the standard for humanoid characters 8 *Bernie Roehl*

VRDJ Interview: Continuing Commitment To a Growing Technology *A talk with one of the original writers of the VRML spec: Tony Parisi* 38 *Scott Davison*

Virtual Environments: *Strait Dope*
Exploiting the possibilities of a new display system 16 *Margaret Dolinsky*

Integrating JavaScript into VRML
How to use VRML elements to build 3D objects 20 *Guy Huggins*

VRDJ Feature: Adding *VR* to *VRML*
Solving tough issues to build a better foundation 30 *Paul Beardow*

Extending VRML 2 with Java
The external authoring interface explored 42 *Bruce Campbell*

Each issue of VRML Developer's Journal includes:

● Practical examples of successful deployments of VRML including enhancements as a result of the integration of additional technologies. ● Profiles of VRML's movers and shakers ● Reviews of tools and resources of interest to VRML developers ● Coverage of industry news and events

$ Satisfaction Guaranteed $

If you are not satisfied with VRDJ at any time, we will refund the unused portion of your subscription. ***Guaranteed!***

Editorial Board

The VRML Developer's Journal Editorial Advisory Board is a veritable who's who of VRML masters. Their knowledge and experience help guide and shape VRDJ into the valuable information resource you can trust to give you the accurate and insightful information that will help with your development efforts.

Neil Trevett
Chairman
President of the VRML Consortium

Tony Parisi
President and CEO of Intervista Software, Inc.

Miko Matsumura
Java Evangelist for JavaSoft.

Mark Pesce
Co-creator of VRML

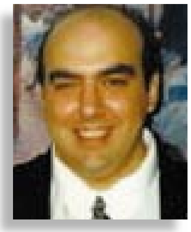
Sam Chen
Cyber Architect at Silicon Graphics

Subscribe Today and take advantage of our Charter Subscription Rates. Save **$10.00 off** the regular subscription rate! As a Charter Subscriber, you pay only **$39.00** for a full year – 6 issues in all!

To Subscribe call: **800-513-7111**
Outside the U.S. 914-735-1900
Fax: 914-735-3922
Order online: **http://www.sys-con.com**

One year is 6 issues. Savings is based on regular annual subscription rate of $49.99 for six issues. Subscription begins upon receipt of payment. Please allow 4-6 weeks for delivery of first issue. Payment must be payable in US dollars, with checks drawn on a US bank. Canada/Mexico: $59.99/year. All other countries: $69.99.

Layout Managers



What, why and what to watch out for

by **John Tabbone**

My last column covered Components, Containers and Events. The material discussed Java's different kinds of Components, the class structure, how Components generate events and how those events can be handled. We discussed the differences between Components and Containers. To recap, Component is the base class for all user interface widgets. A Container is a descendant of Component whose sole purpose is to hold other Components. For example, if you were to create a user interface and you wanted to add several buttons and text areas, you would create instances of the appropriate classes (which all descend from Component) and add them to some kind of Container (like a Panel).

If you have experimented with the AWT classes (or if you have been doing your assignments) you have probably noticed Java's distinct lack of a precise method to place Components. Basically, the programmer is offered a method similar to add(Component). There are no arguments indicating coordinates or sizes; nothing in the method call indicates where the Component may wind up. All you know is that it will be added, somewhere. Most new Java programmers have spent a lot of time wondering, "Where will the Component be added after this method is executed?" Even after experimenting with the add(Component) method, most programmers get unpredictable results. Either the Component is added in some awkward place, *or* it resizes funny, *or* it never shows up on the screen at all. If this has happened to you, don't despair. These are common side effects of using one of Java's layout managers.

A LayoutManager is an interface whose implementation provides rules for Component placement. Containers use LayoutManagers to indicate where Components will appear when they are added (using the add(Component) method). LayoutManagers also provide a mechanism to describe how Components will resize if their parent Container grows or shrinks. For example, a LayoutManager might indicate, and a certain one does, that the parent Container is to be broken up into a grid of a predetermined number of rows and columns. Components are placed into equally sized grid cells, starting at the upper left and proceeding to the lower right. When the Container resizes, new cell dimensions are calculated and the Components placed in those cells assume the corresponding size.

LayoutManagers can often be awkward to use even if the rules are very simple, as in the brief GridLayout example above. User interface requirements are often very complicated, and a simple set of rules usually does not accommodate the many exceptions needed to put a screen together aesthetically. It is possible to use many Containers, each of which uses a different LayoutManager, to create a nice looking screen, but to the developer this usually seems like a clumsy approach. There always seems to be one button or text field that has to grow differently, or needs to be placed in an odd place.

So why does Java promote the use of LayoutManagers? Why shouldn't Components to be placed by coordinate values? If the developer wants a button at an exact location, why do they have to go through the headache of creating a LayoutManager, adding the component and often settling for whatever happens? Well, it turns out that the programmer's investment in LayoutManagers pays off pretty well. Java is a cross-platform language. Machines that run Java programs can have a wide variety of screen resolutions. If Java programs hard-coded Component placement with pixel coordinates, they might be unusable on certain systems.

Imagine yourself in a few years using your favorite Java-based personal organization software package. While on the road, you would probably use the software to keep appointments and such, and you will probably run the program on your tiny little Java-enabled watch-computer. When you come home, you might like to run the same program on your-wall sized flat panel Java-enabled television. If the writers of the software package only had your watch in mind when hard coding the coordinates of Components, the whole program might take up about a square inch of the upper left corner of your wall-sized television. But since they used LayoutManagers to design the screen, the application could grow to fit the whole size of your television.

In today's terms, you cannot be sure of the monitor resolution of end user machines. A program developed with Components hard coded to fit on an 800X600 screen will be clipped when running on machines in 640X480. Java uses LayoutManagers to ensure that your programs look similar on all platforms.

A LayoutManager is actually an object that implements one of two interfaces: LayoutManager or LayoutManager2. The interfaces are defined in the java.awt package. LayoutManager is an artifact from Java 1.0X and is used by all of the LayoutManager implementations included in java.awt. The LayoutManager2 interface is new to Java 1.1. The methods defined in this interface allow for extra information (constraint information) to be stored with each Component added to a Container. The constraints may hold some useful data indicating how a particular Component sizes when more space becomes available, or rules describing how much distance one Component must be from another.

The methods defined in these interfaces rarely get called by the programmer. Containers call the methods when they have to resize the screen. Programmers only have to work with specific methods when creating their own LayoutManager implementations.

The steps involved in getting a Container to work with a LayoutManager are fairly straightforward. First, you need to create an instance of a LayoutManager. Next, you will want to tell a Container to use this new LayoutManager instance. Finally, you add your Components to the Container. The Container will rely on the logic defined in the LayoutManager to determine where the Components will be placed. See the example code in Listing 1.

As I mentioned earlier, Java comes with some LayoutManagers. Though some of them are crude, and others are a little tricky to use, they are all that a programmer really needs to create a decent user interface.

Java has five LayoutManagers. This article will discuss four. The fifth is a complicated beast called GridBagLayout which I will discuss in my next column. The following are descriptions of the others.

FlowLayout

FlowLayout is the default LayoutManager for all Panels. This means that if you don't explicitly assign a LayoutManager to a Panel, you get FlowLayout. FlowLayout can be very frustrating for new programmers to use because it does not resize Components. Components controlled by a FlowLayout are added to the Container one row at a time, much the same way that letters are typed into a document. When there is no more space available in the current row, FlowLayout will 'carriage-return' to the next row and start adding Components there until the end of the row is reached. FlowLayout defines several constants to indicate whether Components will be added with a 'left justification', 'right justification' or 'center justification.' The names of the constants are LEFT, RIGHT and CENTER.

As mentioned earlier, Components added to a FlowLayout do not resize. The LayoutManager gives each Component its preferredSize. A very common mishap occurs when the underlining Container does not have enough space to fit all of the Components that are added to it. Under these circumstances, a Component will appear clipped or will not be drawn at all. It is very common for Java programmers to spend hours wondering why the Component they just added does not show up on the screen, even though everything compiles fine and the code has been examined several times. If you are using FlowLayout, chances are that the Container is not big enough!

> *"A LayoutManager is an interface whose implementation provides rules for Component placement."*

BorderLayout

BorderLayout is the default LayoutManager for Windows. BorderLayout divides the screen into five regions: North, South, East, West and Center. Components are added to a region, using the add(String name, Component c) method, and expand to fill the entire area. The Name argument indicates the region in which to add the Component. Components added to the North or South fill the entire width of the Container and assume their preferred height. If the Container is not big enough to accommodate the sum of the two preferred heights, the Component added first will be drawn on top of the Component added second. Components added to the East and West regions have whatever height is left after the North and South Components take their share. The East and West Components will contend for width the same way the North and South Components contend for height. The Center Component gets whatever is left over.

Programmers often get into trouble when using BorderLayout. One common mistake is to use the wrong add method. If you call add(Component) and do not specify a region to add the Component to, your code will compile fine but the Component will not show up on the screen! Also, North, South, East, West and Center are caps-sensitive. So if your code makes a call with the wrong capitalization (i.e., add("south", b1);), you will not see b1 appear in the center region of the Container and you will not receive any compiler errors! Finally, if you try to add two Components to the same region, only the second one will appear on the screen.

GridLayout

GridLayout is the simplest, most predictable LayoutManager. GridLayout divides the Container into equally sized rows and columns. Components are added to each cell starting from the upper left and going from row to row until finishing up in the lower right cell of the grid. Components added resize themselves to fill up an entire cell.

GridLayout's Constructor describes the number of rows and columns the Layout will contain. For example, GridLayout gl = new GridLayout(10,9); divides the Container into a grid of 90 cells. The grid has 10 rows and 9 columns.

A common 'feature' encountered by programmers when using GridLayout occurs when the developer has not taken the time to count exactly how many Components will be added to their Container. In many cases, GridLayout's grid will often reshape itself to better hold its Components. For example, if you specify a 3X3 grid in GridLayout's Constructor, but only add four Components instead of nine, the grid may actually appear to be 2X2 in size.

CardLayout

CardLayout is a peculiar LayoutManager used to organize Containers. For example, if you had five Panels, each of which had some Buttons, TextAreas, etc. and you wanted to display these five Panels within a particular Container, one at a time, you would use CardLayout. Using this, you could change which Panel is displayed and which are hidden. Probably the most popular example of CardLayout is the tabbed dialog. A tabbed dialog has a single display area (Container) and some tabs running along the top. When a tab is pressed, a different screen (Panel) appears in the display area.

About the Author

John V. Tabbone is a lecturer at New York University's Information Technologies Institute, where he teaches two Java programming courses and advises on curriculum development. He has been a professional Java programmer since early 1996, and continues to consult on and develop systems for a variety of New York based businesses.

john.tabbone@nyu.edu

Listing 1

```
Panel   p   = new Panel();
Button   b1 = new Button( "Button1" );
Button   b2 = new Button( "Button2" );
Button   b3 = new Button( "Button3" );
GridLayout myLayout = new GridLayout( 3,1 );    // First, create an
instance of the
// LayoutManager . In this example we use
// GridLayout
p.setLayout( myLayout );
// Next, tell the Container to use the
// LayoutManager
p.add( b1 );
p.add( b2 );
p.add( b3 );  // Finally, add the Components. When the Components
get added to the
// Container, the LayoutManager determines where they will be
positioned on
// the screen. In this case, the Panel is divided into three rows
and one column,
// each of which will contain one Button.
```

BUILDING A Chat Applet

Interfacing Java applets with Perl scripts using CGI

by **Joseph DiBella**

Java enables us to embed miniature applications called Applets within Web pages which can process data, perform graphical animations and access databases, among other things, in a dynamic fashion. In addition, these Applets work fine on several different types of computers accessing these Web pages over the Internet. The dilemma with using Java in real world Internet Web solutions for small businesses is not necessarily a limitation of the Java language, but limitations placed upon it by Internet service providers.

Most small businesses do not maintain their own Web servers; instead, they employ the services of a Web hosting company. In fact, about 80 percent of my clients use Web hosting companies to store their Web pages. When using these other companies, you typically have to take what they give you. Many of these hosting services use some sort of Unix-based Web server, while others use Windows NT server with Internet information server.

It wasn't long before I realized that I would want to use Java for more than just the animation and basic ticker-tape applets that made it famous in the first place. In previous articles (***JDJ***, Vol. 2, Issue 5), I have demonstrated how easy it is to write a Web server in Java. Using such a Web server, I could receive data from a client-side Java Applet and then process that data in any way I chose. I may want to log this data to a file to be stored on the Web server and/or return data to the Applet. However, this is where I hit a brick wall.

Just because I can write a wonderful server application in Java doesn't necessarily mean that the Web hosting company will allow me to run it on their Web server. Since many of these companies are responsible for hosting several hundred Web sites, it is not hard to see why they may become hesitant to run my server application. Rest assured, if lightning comes through the window and strikes their Web server, my Java application would be blamed.

The one thing I have found that most Web hosting companies seem to have in common is that they have Perl installed on their Web servers. Hence, it may be more practical to use Java Applets on the client side and Perl scripts on the server side. For some reason, Web hosting services tend to trust Perl programs and as a result, I decided to learn how to code in Perl out of necessity.

In going to the bookstore, the first book I picked up was *"Perl Programming for Dummies."* Although it seems like a fine book, I saw a chapter entitled "Ten reasons why Perl is better than Java." I quickly put that book back on the shelf. After all, everyone knows that Java is the world's finest programming language. I have written programs in Basic, Fortran, Cobol, PL/1, RPG, Assembler, C, C++, Clarion and Java. Programming in Perl feels like learning to ride a bike again. You will fall down and get bloody knees over and over again, but you have to keep trying. As you will see here, Perl is an incredibly cryptic and hard to learn language at first, second and third glances. However, it is a powerful and useful tool to add to your programming tool chest. My advice is to buy a few Perl books and start small and simple. In this article, you will see that you only need to know the basics of Perl to write effective back ends to your Java applets.

Why do we need a back end in the first place? Java Applets can easily pull data from a Web server; however, they can not easily push data to a server for storage. This is due to the very nature of the mechanism provided by http Web servers. Typically, a Web browser requests a Web page and the server sends it. CGI extends Web servers so that data can be sent to a server and processed by programs which the server specifies. One of these programs is Perl.

The Perl Interpreter, along with a Perl script, can process the data sent to the Web server by the browsing client and send back a dynamically created Web page based on the processed data. For example, I can write a Perl script called addTwoNumbers.pl in an ordinary text editor. The Perl Interpreter will interpret the script so it does not need to be compiled by a compiler. I can place this script in an executable directory on a Web server which has been set up to use Perl. I can write an HTML form which allows a browsing client to enter two numbers in a form. When the submit button on the form is pressed, the browser sends the two numbers which the person has typed to the server. This, in turn, invokes the Perl Interpreter to process this data along with the Perl Script. Perl can then create an HTML document on the fly and send it back to the browser with the answer. I could also use this method to do other server side things, for instance store data to a log file.

> "Java Applets can easily pull data from a Web server; however, they can not easily push data to a server for storage."

Using Perl in this way is sort of a clunky way to send data back and forth to the server; however, it is perhaps the most popular. Each time data is sent, a new Web page must be reloaded. Any one who has done shopping online this past holiday season can attest to the amateur way in which transactions are handled.

In this article, I will write a Chat Applet in Java. This applet will exemplify the benefits of a Perl back end and it will interface with a Perl Script on the server. A broad overview of this project is as follows:

The applet will read a log file stored on the server repeatedly at a given interval of time. This log file will store the last twenty chat submissions. I will use a thread to read the log file every 8-10 seconds. Any new data in the log file will be appended to the text in a TextArea. I will need a way of parsing the log file and extracting new data only. I will send Chat data to the server's Perl script in a separate thread. This will emulate the process examined in the above HTML/Perl example. Each person will be required to enter a name to chat with. When a person enters or leaves the Web pages with the chat applet on it, a message will be submitted to the server that informs others that he/she has either entered or exited the chat room.

Now it becomes apparent why the Perl script is needed. While it would be a trivial task to read from the log file in a Java Applet, sending data (to be appended) to the log file, using http, becomes a challenging task. Each line of the log file will represent one line submitted to chat. I will configure the Perl script to log the 20 most recent chat submissions in a file, which will be read later by my applet. In doing this, I can be assured that the log file will contain 20 or less lines of text, thus keeping it very small in size. Each chat line received will be appended to the beginning of the log file so that the most recent submissions will appear at the top of the file. This is like making pancakes – the hottest (most recent) one is on top of the stack.

Each time the Java Applet reads the log file, it will read the most recent chat line first, since the Perl script is creating the file with each new submission appended to the top. The Applet will compare each line to the first line read in the previous reading. If these lines are different, then the chat log has been updated since the applet last read it. The Applet will continue to read the file until it either finds a match or reaches the end of the log file. I will store each line read in a temporary String, appending each line read to the beginning of the String. This will re-order the chat data with oldest chat first to most recent chat last. This is like taking the pancakes off the plate one at a time and placing them on another plate – now the hottest one is on the bottom. Then I will append the temporary String to the end of the TextArea, where the users are reading the chat data.

To submit a chat, I need to emulate the interaction that happens when a Web browser uses CGI to send data to a Web server. A Web browser can use GET or POST to send data to a Web server. If you made a form in HTML and used the GET method, when a person pressed the Submit button the browser would request the following URL (or something similar):

```
http://www.myserver.com/cgi-
bin/joeTest.cgi?name=joe&age=32&eyecolor=blue
```

The URL up to the question mark would name the Perl Script, in this case joeTest.cgi. After the question mark would be the data, which has been URL-encoded to replace spaces and other illegal characters, to be sent to the Perl script and to be used like command line variables. This data is called a query string. The Perl program decodes the data and makes chunks that look like: Variable=Value. Then, as you've probably guessed, it parses these chunks into variables and assigns the appropriate values to them. Now these variables can be used for something useful in the Perl program.

The POST method works almost the same way as the GET. The main difference is that the data is read from standard-in instead of the query string. This data is still URL-encoded, so I will still need to decode it in the Perl script. It is almost always a good idea to use the POST method over the GET, since the GET method has limitations.

With this preliminary planning complete, let's jump right in and start this project! First, we'll examine the Perl program:

The first line describes the path to the Perl Interpreter. This line is not needed if your Web server is Windows NT with IIS. In Perl, a comment is denoted by the # in the same way that // are in Java. Next, you will find the line that says:

```
require 'ctime.pl';
```

This is similar to a Java import. It states that this Perl script requires the ctime.pl file, which is included with Perl. I will use this file for date and time utilities, since I would like to record the time that each chat line has taken place. The "define variables" portion of this script defines a variable called $maxChat and assigns it the value of 20. This is where I can state the number of lines I will maintain in the chat log file. Notice how variables in Perl start with a $. These are called Scalar variables. Using scalar variables, I do not need to define a type for the variable, as I do with Java. Perl figures it out based on the context of how the variable is used.

The next three lines create a variable called $date and assign a String value to it, which is returned by the function &ctime(time). This function will create a string with the date and time. Chop removes the terminating or new-line character from the end of $date. I will then extract a substring from $date and store it back in $date, since I am not interested in all of the date-time data.

```
$date = &ctime(time);
 chop($date);
 $date = substr($date,4,12);
```

The entire "Get Input" area does quite a lot. First, it will read the data, which will be sent by my Java Applet, via standard-in and place its contents into a variable named $buffer. The line,

```
@pairs = split(/&/, $buffer);
```

splits the buffer into chunks, delimited by the & symbol. Each chunk will be added to an array called @pairs. Notice that I did not need to specify an array length in Perl and that arrays are denoted by the @ symbol. The foreach loop iterates through the array and states that each value will be referenced by $pair. In the loop, I am taking each $pair, splitting it delimited by the = sign and placing the results in temporary variables called $name and $value. The next two lines decode the URL-encoded lines. Next, I stuff the $value into an associative array called $contents and index it by $name. This works much like Java's Hashtable object.

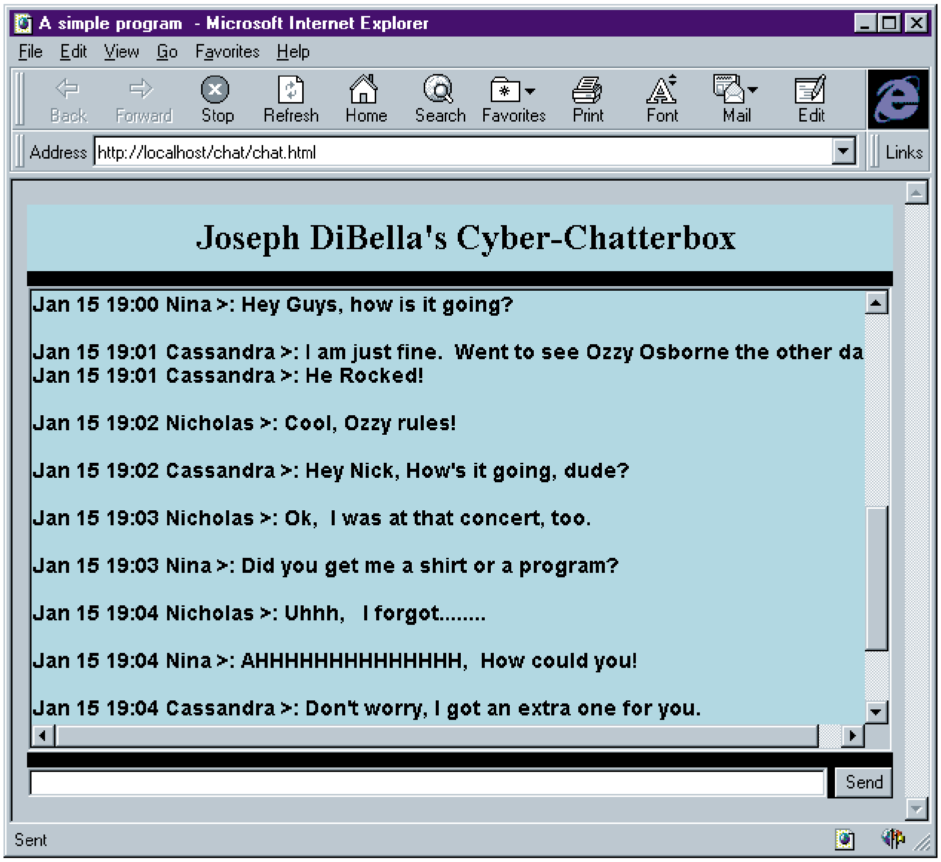
In the next section, I specify the full path and file name to the log file. On my computer, this is the specific path to the log file, which must be placed where the applet can read it. I get the log file name from $contents{'chatfile'} and append a .log extension. I am doing this so that a hacker cannot make a chatfile.bat file with commands in it to format my hard drive. Such a chat file name would result in chatfile.bat.log. Call me paranoid, but I don't like to take chances. Since I use Windows NT, I am using an MS-DOS style path. In UNIX, I would use a UNIX style path. Regardless, this path must place the log file in a directory that will be accessible by the Applet. Thus, I provide a path to the exact location where the HTML file which runs the Applet resides. I will do something similar and specify a file with a .lock extension as well. The "lock file" will be used later so that only one person can write to the chat file at a time.

The next line, &get_file_lock; will call a function which I have defined at the end of the script. Let's jump down there now to look at it.

```
sub get_file_lock {
    $endtime = time + 60;
    while(-e $lock_file && time < $endtime){}
    open(LOCK_FILE, ">$lock_file");
}
```

The lock file is simply a flag which indicates that someone has opened the log file. When the user is finished with the log file, he/she will delete the lock file. The idea behind this lock file is:

- I check if a lock file already exists. If it exists, I know that someone else is in the process of writing to the log file. I will continue to check for its presence for 60 seconds.
- If the lock file disappears within the 60 second period, I create a new lock file so that others will know that I am now using the log file. When I am done with the log file, I will later delete the lock file using the &release_file_lock function so that others can use the log file.
- If the lock file does not disappear, I will assume that something went wrong. If the Web server was rebooted while the lock file exists, for instance, it will reboot, in error, with a lock file existing. Sixty seconds is more than enough time to wait. After waiting this time period, I will take control of the lock file for myself.



Joseph DiBella's Cyber-Chatterbox

Jan 15 19:00 Nina >: Hey Guys, how is it going?

Jan 15 19:01 Cassandra >: I am just fine. Went to see Ozzy Osborne the other da
Jan 15 19:01 Cassandra >: He Rocked!

Jan 15 19:02 Nicholas >: Cool, Ozzy rules!

Jan 15 19:02 Cassandra >: Hey Nick, How's it going, dude?

Jan 15 19:03 Nicholas >: Ok, I was at that concert, too.

Jan 15 19:03 Nina >: Did you get me a shirt or a program?

Jan 15 19:04 Nicholas >: Uhhh,   I forgot........

Jan 15 19:04 Nina >: AHHHHHHHHHHHHHH, How could you!

Jan 15 19:04 Cassandra >: Don't worry, I got an extra one for you.

Figure 1

In detail, the function (subroutine) creates a variable called $endtime and sets its value to whatever the current time is, plus 60 seconds. The while loop will check if the $lock_file exists with the -e. If it does exist, it will continue to check for its disappearance until $endtime, while performing a do-nothing loop. Then it will open the lock file for writing by using the > sign.

While we are down here, let's look at the &release_file_lock subroutine, which I will use later to delete the lock file from the system. After closing the file, the unlink command will delete it from the hard-drive (see Listing 1).

The "read the log file" section uses the open command to open $filename for reading and assigns LOG as the file handle. It reads the entire file into an array called @logcache, making each line an element of the array. Next, I will chop @logcache to remove each new-line character from each element of the array. I will then close the log file, since I am actually storing its contents in my array. The unshift command will take an array and add an element to it as its first element. This pushes all the other array elements down and makes the array grow by one, like a stack. Perl uses the period (.) to append strings, like Java uses the + sign.

The element I am adding to the array is a string, which consists of the $date, a space and the chat line. The next line will determine how many elements are contained by the array. If I have exceeded the desired array length, I will pop the array. This will take the last element off of the array and lessen its length by one.

Now, all I have to do is write this array to the log file and I'm home free, at least on the Perl side.

```
open(LOG,">$filename");
foreach $sendchat (@logcache){
print LOG "$sendchat \n";
}
close(LOG);
```

I will open the log file for output and this

will overwrite the old logfile. Then, for each element of @logcache, I will write its contents to the log file and append a new-line character to each line. When I am finished, I will close the log file. Now all I have to do is release the lock file and send a reply back to the applet, which will finish the Perl program.

```
print "Content-type: text/html\n\n";
print “Chat Received and Processed by Server”;
```

The last part of this month's article will be to examine the Java Applet code, which will submit the chat data to the CGI/PERL script. I have created a class called SubmitToChatServer for this purpose. Since this class implements Runnable, you can assume that I will use a separate Thread to submit the data. This object has the standard Start() method to create and spawn the Thread. Objects of this class will be constructed with the following parameters:

- String text – This represents the actual chat text to be sent to the chat server.
- Applet app – This is a handle to the applet which is needed to construct the URL in order to send the chatline, as well as to show status in the applet.
- String chatFileName – This specifies the name of the file on the Web Server which will store the chat log.

Note that the Perl Script will append a .log extension to this name. Therefore, when I construct this object I will leave the extension off of the file name. I will extract this file name from a parameter listed in the HTML file, which loads the Applet. The reason I might want to do this is so that I can use a single Chat applet to open several chat rooms. The HTML page, which has the name of the chat log file embedded in it as a parameter, can be created on the fly by another Perl script. This other Perl script may specify the chat log file name based off of user input.

The constructor will create a String, named completeMessage, by encoding the data that will be sent to the Server by using a utility method in the URLEncoder class, called encode(). I must separate each variable with an & symbol and use the = sign to assign the value. The URLEncoder.encode() method will replace illegal characters, such as spaces, with characters that can be transmitted to the server without problems.

```
completeMessage =
"chatfile="+URLEncoder.encode(chatFileName)+"&chatline="+
               URLEncoder.encode(message);
```

As with most Runnable objects, the heart of this object is in its run() method (see Listing 2).

Here, within a try block, I will create a new URL object which will point to my Perl script on the Web server, named chat.cgi. Next, I will need to get a URLConnection object for that URL. I will do this by calling the openConnection() method for chatServer, which is the URL I just created. One method I will need to call is setDoOutput(true). This will give me the ability to send my chat data to the Web server. I will not want to cache data, so I will call the setUseCaches(false) method on my URLConnection object.

There are just a few more things I need to specify before I can send the data. I will need to set some Request Properties for my URLConnection object. The Post method requires that I specify what type of data I am sending and its length, in bytes. Since I am Posting the data, I will need to specify the Content-type and Content-length using the setRequestProperty() method. The parameter sent as: “+completeMessage.length()” uses a kind of hack to convert the returned int from completeMessage.length() into a String.

Once the URLConnection is set up like this, I can create a DataOutputStream object with an output stream from my URLConnection object. Using this DataOutputStream, I can then send the data to the Web Server with the writeBytes() method. When I am finished, I simply need to close the stream.

Although, I do not really need to get any data back from the server for my program, I chose to anyway. I have spent many hours trying to figure out why certain Applets like this worked with Internet Explorer and did not work with Netscape. I discovered that Netscape needs to get data back from the Server, while Microsoft Internet Explorer does not. Hence, since getting data back will not hurt anything on MSIE, I did it anyway. I will get back an acknowledgement from the Perl script that I will print out to the console using System.out.println(). To do this, I will open a DataInputStream to the URLConnection object and read from it in a while loop, in the standard way.

Using the information covered in this month's article, you can set up any applet to interface with CGI/Perl scripts on a Web server. This particular project will use this technology to create a Chat Room-style Applet. If you choose, this method can also be used to access data from databases using Perl and ODBC, without the need for JDBC. Next month, however, I will complete the Chat Applet and discuss how to optimize it on your Web Server.

About the Author

Joseph M. DiBella is the Senior Java Instructor and Curriculum Developer for Computer Educational Services in New York City. He also is the President of HMJ Electronics, a computer consulting company which develops software and Java-enhanced Web sites. Joe can be reached at lite-n-sweet@java-joe.com

lite-n-sweet@java-joe.com

Listing 1: Unlink command.

```
# read the log file data
 open(LOG,"$filename");
 @logcache = <LOG>;
 chop @logcache;
 close(IN);
 unshift(@logcache,$date.' '.$contents{'chatline'});
# Calculate the number of lines our chat log contains
 $numberOfChatLines = ($#logcache + 1 );
# check to see if we have exceeded our maximum log size
 if ($numberOfChatLines > $maxChat){
     # Remove the oldest chat line
     pop(@logcache);
 }
```

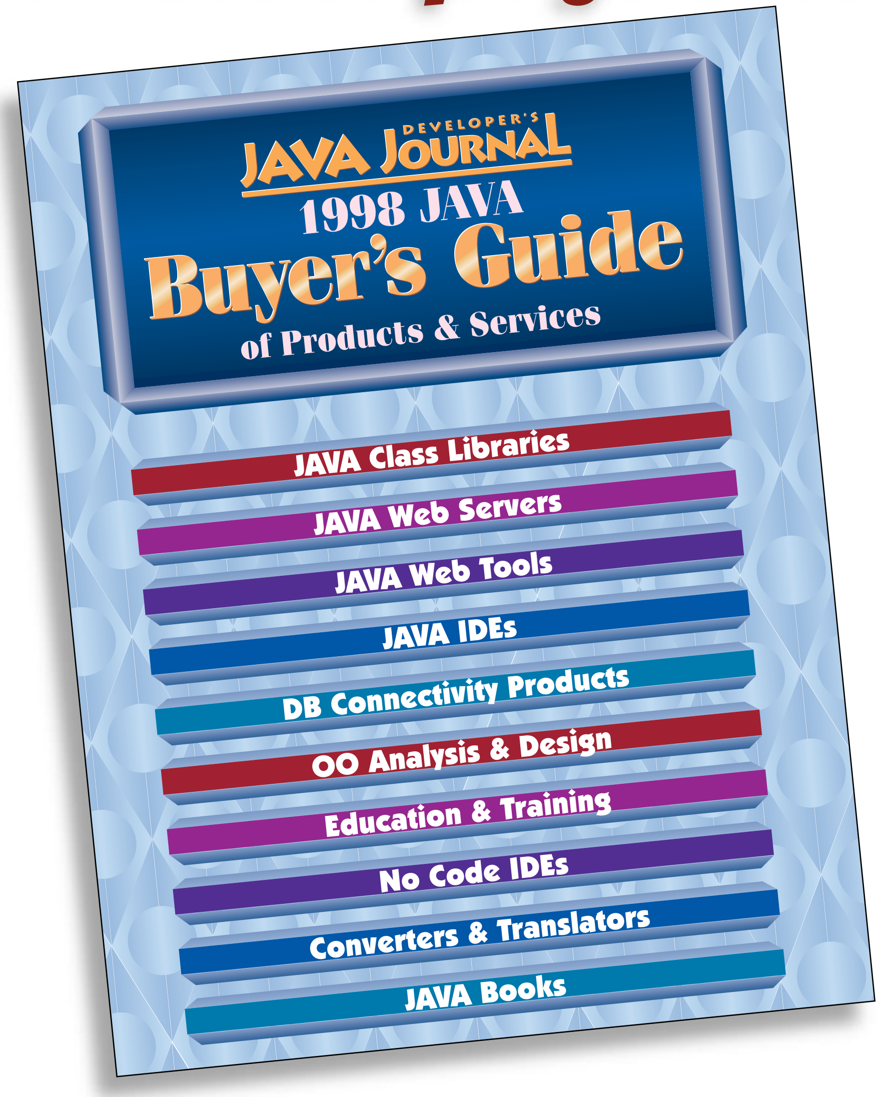
Listing 2: Run ( ) method.

```
   chatServer = new URL(app.getDocumentBase(),cgiPath+"chat.cgi");
   chatServerConnection = chatServer.openConnection();
   chatServerConnection.setDoOutput(true);
   chatServerConnection.setUseCaches(false);
   chatServerConnection.setRequestProperty("Content-type","application/octet-stream");
   chatServerConnection.setRequestProperty("Content-length",""+completeMessage.length());

   DataOutputStream send = new DataOutputStream(chatServerConnection.getOutputStream());
   send.writeBytes(completeMessage);
   send.close();
```

Listing 3: Chat.CGI may be found on our Web site at www.sys-con.com/java

An Excerpt from…



DEVELOPER'S JAVA JOURNAL

1998 JAVA

Buyer's Guide

of Products & Services

JAVA Class Libraries

JAVA Web Servers

JAVA Web Tools

JAVA IDEs

DB Connectivity Products

OO Analysis & Design

Education & Training

No Code IDEs

Converters & Translators

JAVA Books

This Special Insert includes the following listings from the Java Developer's Journal Buyer's Guide

- **Java Class Libraries**
- **Java Web Tools**
- **Java IDEs**

Java Web Tools

ActionLine™

ActionLine™ is a Web page authoring tool for creating dynamic, interactive multimedia-rich Web pages using Java™ technology. No programming or scripting is required. *ActionLine* is ideal for graphic artists, designers and Webmasters who are building intranet and Internet sites with animation, interactive buttons, transitions, scrolling and other multimedia features.

Interactive Media Corporation
Peter N. Rosenthal
650 948-0745
www.imcinfo.com

Applet FX

Applet FX Commercial Edition is a package of 60 high-quality Java™ effects and tools for your home page and includes Visual Applet Configurator (VAC), a powerful configuration and installation tool.

Demicron
Anibal Wainstein
+46 8 53174266
www.demicron.se

Attaché

Attaché is a set of application programming interfaces (APIs) for Itinerary Web Presenter, Contigo Software's flagship product which brings completely new and unique communications capabilities to Internet and intranet users. *Attaché* enables Java™ developers to build new applications to work in conjunction with Itinerary engine. In addition, Contigo has launched the Itinerary Developer's Program in support of Java development on the Itinerary software platform.

Contigo Software LLC
Doug Cooper
619 278-5900
www.contigo.com

BackOnLine

BackOnLine is a 100% Java™-based backup client/server system for the Internet with features such as 56-bit DES encryption, compression, easy server-side administration and almost no client administration. The client can be run as an applet, GUI application or console application. The multi-threaded, multi-user server can also be used by Java developers as an object repository for custom applets/applications via the *BackOnLine* client side API.

Divya, Inc.
Anil Hemrajani
703 861-JAVA
www.divya.com

Billboard™ Applet

Billboard™ is a Java™ applet that draws attention to your Web site. Breaking news, events or other messages are displayed via a single-line, HTML controlled, auto-scroll or stationary text box. No knowledge of Java or JavaScript is required. All parameters and text are controlled from your HTML file.

Cribsheet Software
J.P.Evans
310 532-6212
members.aol.com/cribsheet/home.htm

Cartlet

Cartlet is a Java™ shopping cart designed to work without server side programming. It includes configurable applet classes for cart display, order form and buy buttons, plus tiplet pop-up windows to highlight special featured products.

Motivational Marketing Associates, LLC
William Newell
716 689-1593
www.mmaweb.com/replet

Castanet

Castanet is for businesses that wish to lower the cost and simplify the complexity of destroying, managing and updating applications and information over corporate networks and the Internet. *Castanet* channels are applications or services distributed by *Castanet* Transmitters. Once a channel is created, it is published to a *Castanet* Transmitter running on a network server. The Transmitter then communicates with *Castanet* Tuners to automatically distribute the channel's content.

Marimba, Inc.
Beth Johnson
650 930-5224
www.marimba.com

ChartBlaster

ChartBlaster is a cross-platform, graphically based, chart editing application. It allows users to easily create, modify and publish a wide variety of scientific and business-oriented charts. Since *ChartBlaster* was developed using the Java™ software language, it can be used on many different platforms, including Win95, WinNT, MacOS and most UNIX platforms. Additionally, since *ChartBlaster* is 100% Java, it can also be used on all Java-compliant Network Computers.

NetFactory
Mitch I. Selbiger
301 625-5600
www.netcharts.com

Crosstie

Crosstie is the terminal emulation solution for Web browsers. It installs on any HTTP server and works with any Java™-capable browser, allowing users to access existing UNIX applications from a Web page. No plug-ins or client downloads are required. *Crosstie* offers an unparalleled level of emulation accuracy and true cross-platform support.

Maximum Computer Technologies, Inc.
Charlotte Canup
770 428-5000
www.maxtech.com

CUChat

CUChat is a Java™ Visual Chat client/server platform. Clients, upon arriving at a Web site enabled with *CUChat,* will be able to communicate/chat visually (represented as Avatars) with one another. There is no need to install a plug-in since *CUChat* is in 100% Java.

NetDIVE Corporation
Sue Berna
415 474-3756
www.netdive.com

Cyber Billboard

Cyber Billboard is an image slide show with variable transitions, links and audio capabilities. This high powered Java™ display is ideal for Web advertising.

Sea-Tech Australia
Mark Griffiths
9584 9557
www.ozemail.com.au/~seatecha

CyberCAT

CyberCAT Java™ *Catalog* is a Java™ applet for deploying Internet catalog and product information easily and inexpensively without programming. Simply add your product information, images, advertisements and user interface graphics. *CyberCAT Java*™ *Catalog* takes care of the rest.

Betacorp
John J. Bekto
905 564-2424
www.betacorp.com

Dynamo Developer's Kit

Dynamo Developer's Kit is a powerful Java™ application framework for Web developers who want to create a new class of consumer and community-based applications. *Dynamo Developer's Kit* provides a complete server-side Java platform with the full power of Java's multi-threaded, object-oriented, type and memory-safe programming techniques. It also allows Java server-side applications to seamlessly integrate with client-side components.

Art Technology Group
Karen McPhillips
617 859-1212
www.atg-dynamo.com

Dynamo Retail Studio

Dynamo Retail Station is the robust storefront management environment for businesses that want to rapidly build, customize and manage feature-rich, online storefronts. *Dynamo Retail Station* dynamically delivers catalog content and promotional programs to create compelling, personalized shopping experiences. It integrates easily with existing databases and order-processing systems to allow maximum flexibility.

Art Technology Group
Karen McPhillips
617 859-1212
www.atg-dynamo.com

ebGATE pro 2.0

ebGate pro 2.0 provides client-side password protection for your Web pages. This powerful Java™ applet is the perfect client-side solution for non-programmers and Web developers who do not have access to the CGI bin or server-side configuration. Try our online demo.

Electric Butterfly
Dave Wooldridge
www.ebutterfly.com

FindItFast

DoubleOLogic Software sells Java™ applets that provide database query and reporting capabilities on any Web page. Web page database applets read data from HTML parameters or a sequential data file and offer searching, sorting and innovative graphical display of data in an applet window. These applets are designed for the Webmaster to plug in to any HTML document.

DoubleOLogic Software
Diana Luckevich
425 649-1296
www.doubleologic.com

1st Java™ Bundle

Add Java™ tabs and/or tree views to your Web pages – no Java or HTML programming experience necessary! The latest ultimate "all in one" Web site navigation bundle consists of all three of our top selling products in one hit: The 1st JAVATab provides the most simple Web site navigation for your visitors with the ease of tabs.

Auscomp World Wide
Guenther
www.auscomp.com

GuruSuite

GuruSuite is an online buyer's guide database system specifically designed to increase Internet advertising revenue for publishing companies. Written in 100% Pure Java™.

Intelligent Databases International Ltd.
Sean Krakiwsky
403 272-4413
www.inteldb.com

Hybrid Shopping Cart

Hybrid Shopping Cart is a Java™ applet that provides a complete user interface package for Internet shopping Web sites.

A "hybrid" is defined as an offspring of two varieties – an item produced by the blending of two diverse traditions. That's exactly what the *Hybrid Shopping Cart* offers; a blending of the best features from our CGI and Java shopping products. The most powerful aspects of Java technology are combined with the most desirable features of our CGI shopping cart.

Eastland Data Systems
Ed Zarrella
301 831-9681
www.eastland.com/hybrid.html

I.R.A

The award-winning *Internet Retirement Assessor* gives investors access to current statement/investment-related data in the context of a set of analysis tools to better access their retirement needs. Investment providers can then keep their investor base more informed via the timely data as well as the optional newsletter concerning the provider.

Clearview
Stephen Randolph
617 961-8800

infoBook

infoBook is a Java™-based information presentation manager for Web pages. With flexible screen configuration options, CGI scripting, extensibility through Javascript, connectivity to any data source and over 70 configurable parameters, it is a Web page developer's dream tool. *infoBook* has won such awards as JARS' Top 1% and Gamelan's Featured Applet.

Divya, Inc.
Anil Hemrajani
703 861-JAVA
www.divya.com

InstallAnywhere

InstallAnywhere, the installer utility designed specifically for Java™, allows developers to quickly and easily create a single, universal installer that will run on any Java platform, even installing a virtual machine on the client if one isn't available. *InstallAnywhere* is a complete software distribution and setup solution, producing commercial-quality installers for Java software and traditional platform-specific software as well.

Zero G Software, Inc.
Patrick Collins
415 512-7771 (ext. 29)
www.zerog.com

Instant Messaging

imChat's *Instant Messaging* application is designed to be used in active forums either Internet or intranet. It has rotating banner daemon integrated for advertising purposes, and fast load time and two ways to access. Users can e-mail or chat with each other. This is not a full blown chat applet and was designed to let users see who's at the site.

imChat
Mark Porter
888 387-5350
www.imchat.com

Internet Shopping with Java™

Welcome to the latest in Internet shopping user interface technology – the new way to shop on the Internet. This Java™ Applet provides a complete user interface package for Internet Shopping Web Sites.

Eastland Data Systems
Ed Zarrella
301 831-9681
www.eastland.com/shopping.html

Interrogate

The *Interrogate* search agent integrates with Web browsers to provide a simple user interface for searching files – either within Web sites or on disk or CD-ROM. It operates entirely on the local computer, removing any necessity for server-side scripting when searching pages on the World Wide Web.

Bigfoot Partners, L.P.
Chris Lacey
www.bigfoot.com

IPnetWATCHER Java™ Edition

IPnetWATCHER 1.0 Java™ Edition combines the latest in Java™, Web and Push technologies to deliver real-time performance/fault-monitoring and alerting to managers and users of IP networks.

IPnetWATCHER finds and concurrently tests both on-site and remote IP and SNMP devices, services and even applications without the need for additional software or agents.

Caravelle, Inc.
Diane Dekuysscher
613 225-1172
www.caravelle,com

Itinerary Web Presenter

Itinerary is a Java™-based Web site presentation and collaboration software which facilitates efficient, synchronized interaction for remote viewing of a common Web site. Via any Java-enabled browser, *Itinerary* allows someone to assume the role of "pilot" and remotely navigate one or more "passengers" through a live presentation of one or multiple Web sites.

Contigo Software, LLC
Doug Cooper
619 278-5900
www.contigo.com

Jshrink

Jshrink removes names and unused data from compiled Java™ class files, resulting in smaller files that load faster and that yield less information when decompiled.

Jshrink does not change public or protected member names and can, therefore, be used with redistributable components, such as class libraries.

Eastridge Technology
Nick Eastridge
609 252-0825
www.e-t.com

MerzScope™

MerzScope™ is the revolutionary new way to map and surf on-line documents, whether on the Web or on your corporation's intranet. Compatible with any Java™-enabled browser, *MerzScope™* is a mapping and navigating tool that allows you to produce dynamic graphical maps of any collection of Web pages and links.

MerzCom, Inc.
Sharon Kroo
514 736-1647
www.merzcom.com

NetResults™ v. 1.2

NetResults™ by Innotech is a 100% Pure Java™ search engine for Web site or intranet use. With the "auto-config" feature, installation, configuration and index set-up functions are all achieved with minimal input from the user. In three easy steps, information on your site will become searchable by information seekers.

Innotech Multimedia Corporation
Stephen Paterson
416 492-3838
www.netresults-search.com

NewsWire™ Applet

NewsWire™ is a Java™ applet that draws attention to your Web site. Breaking news, events or other messages are displayed via a multi-line, HTML controlled, auto-scroll or stationary text box. No knowledge of Java or JavaScript is required. All parameters and NewsWire text are controlled from your HTML file.

Cribsheet Software
J.P. Evans
310 532-6212
members.aol.com/cribsheet/home.htm

Object/LM

Object/LM effectively manages the deployment of distributed applications by providing the industry's only out-of-the-box solution for controlling and tracking object utilization and associated costs. *Object/LM* provides a secure, easy-to-use, universal access control and metering system that conforms to the CORBA standard, supplying IDL interfaces with mappings to C++ and Java™.

Black & White Software, Inc.
Susan Karas
408 369-7400
www.blackwhite.com

Object/Observer

Object/Observer offers diagnostic and trace mechanisms for effectively monitoring distributed object communication using CORBA. Method invocations can be visually monitored and can be recorded for subsequent analysis. Monitoring covers invocation requests with parameter values, replies with return values and thrown exceptions. Each method can be tracked for a finite number of invocations or indefinitely.

Black & White Software, Inc.
Susan Karas
408 369-7400
www.blackwhite.com

OC://WebConnect Pro

OC://WebConnect Pro provides secure access to mainframe and mid-range SNA applications from any Java™ enabled Web browser.

OC://WebConnect Pro provides enterprise-wide application delivery for 3270, 5250 or VT220 applications. *OC://WebConnect Pro* supports any standard TN3270, SNA Gateway or mainframe TCP/IP stack for communications.

OpenConnect Systems, Inc.
Kristi Williams
972 888-0693
www.openconnect.com

OptimizeIt

OptimizeIt allows developers to test and improve the performance of their Java™ applets, applications or JavaBeans™. *OptimizeIt* goes behind the scenes of the Java Virtual Machine and shows how an application uses computer resources. Using *OptimizeIt*, Java programmers can easily spot the code responsible for excessive memory allocations or inefficient CPU usage.

Intuitive Systems, Inc.
Berangere Noyau
408 245-8540
www.optimizeit.com

Orb/Enable

Orb/Enable is a productivity toolset which greatly simplifies CORBA application development. *Orb/Enable* includes an IDL Editor with visual compiler front end, an Interface Repository Browser with automatic code generation and a Server Manager for administration and deployment.

Black & White Software, Inc.
Susan Karas
408 369-7400
www.blackwhite.com

PARTS HPGL

View and zoom your CAD files in your Web browser. Use a Java™ applet to view and zoom optimized HPGL files which are converted by our *PARTS HPGL* image converter in batch or single file mode.

Vanguard Software GmbH
Robert Siegel
+43 1 2714866
www.vanguard.at/products

Passport IntRprise

Passport Corporation is the premier provider of easy to use tools for the development and deployment of Internet-enabled, enterprise-wide, fault tolerant client/server applications.

Passport Corporation
Christopher Zosa
201 634-1100
www.passport4gl.com

PECOS.net

PECOS.net extends the PECOS family of secure, multimedia, electronic commerce systems to the World Wide Web. Companies gain the competitive advantages of both our robust full-circle catalog and ordering systems – enabling interactive, enterprise-wide commerce and the convenience, openness and accessibility to the world-wide market of the Web.

Elcom Systems, Inc.
Pat Breslin
617 407-5003
www.elcom.com

PeopleNet HelpDesk Manager

Help Desk Manager, an integrated multi-user Windows package, provides complete front-line support and management. Our WWW interface gives any browser complete access to *Help Desk Manager's* impressive features.

PeopleNet, Inc.
Bart Huitema
818 783-0606
www.helpdesk.peoplenet.net

ProxyReporter

ProxyReporter Internet access management software generates concise reports on end-user Web usage. It is designed to help both departmental managers and network administrators track and analyze employee Internet activity. This activity can be monitored across the entire enterprise, site or department. An authorized user from a single interface can connect to and query multiple servers.

Wavecrestcomputing
Dennis W. McCabe
407 953-5351
www.wavecrestcomputing.com

rBannerD

rBannerD is a rotating banner daemon that e-mails advertising statistics. It has several uses including Logo display on other Web sites with full control of multiple images, each with a different URL and message. Add a service? Add a banner to the rotation and all the sites sporting your logo will automatically update. If your company puts its 88x31 gif on other Web sites, you need this.

imChat
Mark Porter
888 387-5350
www.imchat.com

Rendezvous

Rendezvous is a Java™-based Internet whiteboard. It allows users to conduct whiteboard conferences over the Internet. *Rendezvous* is based on a client/server architecture and is ideally suited for Network computers and intranets.

Visualtek Solutions, Inc.
Prashant Parekh
510 353-0952
www.visualtek.com

RSVP

RSVP turns Java™ into the ideal language for developing fast, demanding, interactive and dynamic Web server applications. *RSVP* applications run inside the Web server, interacting with the user through dynamically generated HTML pages, forms and applets. *RSVP* applications use JDBC to connect to any database server.

Withinreach
Assaf Arkin
www.withinreach.co.il/products

Sapphire/Web 4.0

Sapphire/Web 4.0 is an application server solution which is architected to provide enterprises with complete datasource integration, openness, development and deployment versatility, easy manageability and unlimited scalability. It is a comprehensive environment for building Internet, intranet and extranet applications.

Bluestone Software, Inc.
Anastasia Bullinger
609 727-4600
www.bluestone.com

SiteScope

SiteScope is Freshwater Software's Java™-based server monitoring and administration software for NT and Unix. *SiteScope* checks for URL availability, CPU usage, disk space and critical server processes like FTP, DNS, Telnet, e-mail and more.

Freshwater Software
John Meier
303 443-2266
www.freshtech.com

SpaceSQL 3.0

Designed for today's networked world, *SpaceSQL 3.0* is a server-based solution for deploying enterprise-wide decision support across the Web. *SpaceSQL 3.0* enables any organization to leverage its existing investments in data warehousing and to extend the benefits of business intelligence to hundreds, even thousands of users through a familiar, cost-effective interface: the Web browser.

Infospace, Inc.
650 685-3000
www.infospace-inc.com

Start Page

The imChat *Start Page* is free to all. We can also customize it for your business. It has several features that make it stand out from the advertising giants.

imChat
Mark Porter
888 387-5350
www.imchat.com

StoryServer 3

Vignette StoryServer 3 is a Web content application system designed to meet the demanding requirements of content-driven service applications. *StoryServer 3* delivers a collaborative content management environment tightly integrated with a dynamic content application server.

Vignette Corporation
Jessica McMahon
512 502-0223
www.vignette.com

3Space Publisher

Add fabulous, attention-grabbing 3D animation to your Web site and CD-ROM titles. With *3Space Publisher* it's easy to create photo-realistic, animated 3D banners, bullets, icons, text and images.

Template Graphics Software, Inc.
619 457-5359
www.tgs.com

Visara Network Computer

The *Visara Network Computer* combines multiple host connectivity, the ability to run PC applications and Internet/intranet access in a scalable, centrally managed workstation. *Visara* provides more performance and increases flexibility over the traditional host terminal. It is less expensive to own and easier to support than a PC.

Affinity Systems
John Curran
215 412-0555
www.affinitysys.com

Visual Page

Symantec's *Visual Page 1.1* is the latest version of the award-winning Web authoring program for novice to professional Web developers. *Visual Page* is a powerful Web authoring tool with an intuitive interface, pure HTML code generation (no proprietary code or extensions) and true and real-time WYSIWYG capability. With its easy interface, *Visual Page* is truly designed for novice and expert Web designers who want a quick Web solution, not just another arduous learning experience. A free fully functional evaluation version of *Visual Page 1.1* is available on the Symantec Web site.

Symantec Corporation
Susan Miller
408 447-8317
www.cafe.symantec.com

Web Automation Suite

Web Automation integrates data from the Web with business applications: automatically, reliably, securely – with or without a browser. *Web Automation* enables business information systems to exchange data with any Web site in the world to access the vast store of data now on corporate intranets, extranets and the public Internet.

Web Automation Platform is an API for automating access to all Web addresses (URLs); it combines rich HTML parsing, text pattern matching and a Web Interface Definition Language (WIDL) that binds data objects to program variables with HTTP protocol handling and an object repository which supports complex queries across multiple documents.

webMethods, Inc.
Caren DeWitt
703 352-8501
www.webmethods.com

Web Button Wiz

Add some sparkle to your Web pages with interactive Java™ buttons. You don't need to know Java or HTML. *Web Button Wiz* is so simple that anyone can create dazzling effects in minutes.

Formula Software Pty Ltd
David Mitchie
+61 2 9810 1688
www.formulagraphics.com

Web Integrity

For intranet managers, corporate communications teams and knowledge workers, *Web Integrity* enables team development of Web sites and automates processes for approving content.

Mortice Kern Systems, Inc.
800 265-2797
www.mks.com

Web/Enable

Web/Enable is a member of Black & White's OrbixBuilder family of CORBA development products. It builds upon OrbixBuilder for UIM/X to provide the only Java™ and C++ development solution that incorporates visual drag and drop techniques, a built-in test mode and automatic generation of code that is CORBA IIOP enabled.

Black & White Software, Inc.
Susan Karas
408 369-7400
www.blackwhite.com

Woodstock Hypertext

Woodstock Hypertext is a structured information processor written entirely in Java™ which allows you to treat documents as objects. Use it to generate HTML and XML documents – with and without templates – and to parse and understand the content of documents.

Vtopia, Inc.
Jeff Mackay
847 438-1381
www.vtopia.com

Java Class Libraries

Ambrosia™ Event Management System

Using the *Ambrosia™ Event Management System*, corporate developers and vendors of packaged application software can quickly build event-based business applications that provide secure and reliable inter-application communication over the Internet and/or within corporate networks.

Open Horizon, Inc.
Audrey Kalman
415 869-2263
www.openhorizon.com

Bitmap Graphics Pack

Display bitmap (BMP) images in your Java™ applets and applications. *Bitmap Graphics Pack* renders bitmaps in either original or specified size, and draws in the background which frees your app to do other tasks. *Bitmap Graphics Pack* is Java 1.0.2 and Java 1.1 AWT-compliant.

Solid Logic Computer Solutions, Inc.
Bob Barker
612 949-0140
www.slogic.com

BSF

The *Business Sight Framework (BSF)* is an object-relational Java™ class library that allows Java objects to be easily saved and retrieved from relational databases. Although Java is truly an object-oriented language, applications that access relational databases must deal with tables, rows and columns.

Objectmatter, Inc.
Enrique Travieso
305 718-9101
www.objectmatter.com

CinnaMoney

Overcome Java™ floating point problems and handle large calculations accurately. *CinnaMoney* is an easy to use discretionary and arbitrary precision math class. It handles complete calculations (up to 2 billion digits) without floating point problems. Built in routines easily convert Java datatypes (such as string, float, int and long) into *CinnaMoney* objects. Java programmers determine the precision of calculations as well as the precision of the display and printing of results.

Thought, Inc.
Dan Wilson
415 836-9199
www.thoughtinc.com

Cornix, The Web Power Reader

A Java™ language version of our patent-pending Machine Assisted Reading Software (MARS) technology, *Cornix* beta 10 is designed for the Web. New features include user control of color, type face and type size.

Tenay SE
Cliff High
360 866-1686
www.halcyon.com/chigh/cornix.html

Developer

This is an independent Java™ and on-line developer.

Frederic Najman
331 53 32 84 94

iIntegrale

The Interstream *iIntegrale Class Library* is a set of controls which may be used in a Java™ applet or application to present an "image-smart" user interface. These advanced Java User Interface controls are useful to both Webmasters and Java developers and have been fully tested under IBM's OS/2 Java and the OS/2 Netscape browsers.

InterStream, Inc.
Emily Roberson
512 263-5469
www.i-stream.com

iIntegrale SCE

The Interstream *iIntegrale Class Library SCE* is a set of controls which may be used in a Java™ applet or application to present an "image-smart" user interface. These advanced Java user-interface controls are ideal for both Webmasters and Java developers, and have been fully tested under IBM's OS/2 Java and the OS/2 Netscape browsers.

InterStream, Inc.
Emily Roberson
512 263-5469
www.i-stream.com

ILOG JViews™

ILOG JViews™ is a 100% Java™/E class library for developing high-performance, intuitive 2D structured graphics displays. It complements existing GUI components to create interfaces such as network topologies, maps or customized editors, while achieving outstanding performance and reliability. For Java applications requiring compelling 2D graphics, *ILOG JViews* provides the competitive edge.

ILOG
Ed Kiraly
650 944-7134
www.ilog.com

Intel.Agent Library

Version 3 of this large class library for building Java™ Intelligent Agents includes intelligent modules (rule-based, lisp and neural nets), simulation and modeling tools and graphical tools. It also has a large set of interfaces to conventional systems and databases. It is ideal for systems integration tasks.

Bitpix
Thomas John
512 335-7315
www.bitpix.com/business/main/bitpix.htm

Intraprise Framework™

Intraprise Framework for Java™ is a set of component-based Java applets that serve as a blueprint for development teams. Application developers, who may not be familiar with the intricacies of developing Java applications, can easily obtain and assemble component-based applets to quickly generate business solutions.

Rexton InterActive Corporation
Lindsey Williams
800 700-1224 ext. 606
www.GoRex.com

J/CRYPTO V.2.0

J/CRYPTO is a general purpose cryptography toolkit which can be used to provide security to any Java™ application or applet. Support for full strength keys (128-bit RSA keys) is available world-wide.

Baltimore Technologies
Sharon Grufferty
+353 1 605 4399
www.baltimore.ie

Java™ Generic Library

The *Java™ Generic Library (JGL)* is the most comprehensive set of reusable containers and algorithms available for Java today. Designed for general purpose programming, the *Java Generic Library* contains a full set of features including over 10 highly optimized data structures such as sets, lists and maps and 70 generic algorithms such as sort, union and intersect. It comes with full source code, online HTML documentation, comprehensive examples and a suite of performance benchmarks.

The Java Generic Library is efficient and multi-thread safe. Its containers protect public methods with the synchronized keyword to work correctly in the presence of multiple threads.

ObjectSpace, Inc.
Stephen Andrews
1 800-Object1
www.objectspace.com

JChart

JChart JavaBeans™ enables you to incorporate customizable, dynamic charts in your Java™ applets and applications. Choose from built-in basic chart types, or use the charting primitives included in *JChart* to create your own chart type of any complexity or detail, using a Beans builder or using the pure Java classes programmatically. *JChart's* data model provides methods for dynamic updates of charted data, and built-in callback mechanisms allow you to provide drill-down capability for various portions of a chart. *JChart* includes these basic 2D, 2-1/2D and 3D chart types: multi-row, multi-column and stacked bar charts; multi-row and stacked area charts; and line, pie and scatter-plot charts.

Rogue Wave Software, Inc.
Billie Chapman
541 754-3010
www.roguewave.com

JConfig

JConfig is a set of utility classes designed to help you create shrink-wrap style Java™ applications on Windows and Mac.

JConfig supplements Java's File and Process classes and the Runtime.exec() method.

Samizdat Productions
Chris Kelly
310 226-8065
www.tolstoy.com

JCT 2.0

JCT 2.0 is the next release of Shafir's award winning Java™ Controls Toolkit. All Controls are ported to the JavaBean™ spec and support JDK.1.1.

Shafir Inc.
Steve Hardwick
512 258-0930
www.shafir.com

JCT Chart 1.0

Add 2D and 3D charts to Java™ applications with Shafir's *JCTChart.* One of Shafir's family of toolkits, *JCTChart* includes pie and bar charts, live updates, use bitmaps for chart surfaces and rotation controls. Chart data can be sourced from static text, local/remote files and databases.

Shafir Inc.
Steve Hardwick
512 258-0930
www.shafir.com

JDBTools

JDBTools delivers sophisticated database access and manipulation capabilities from an intuitive, object-oriented API. *JDBTools* is pure Java™ and builds on important Java language standards such as JDBC. In addition, *JDBTools* provides powerful core technology that transparently and seamlessly maps Java classes to specific SQL dialects, and a sophisticated class hierarchy that enhances JDBC.

Rogue Wave Software, Inc.
Billie Chapman
541 754-3010
www.roguewave.com

Jeevan

Jeevan is a platform-independent, extensible and easy-to-use object-oriented database for Java™. Using *Jeevan*, you can now create a database to store and retrieve Java objects with minimal programming. Its powerful yet easy-to-use features provide immediate productivity for building robust, full featured Java applications.

W3apps, Inc.
Tushar Kale
954 389-8529
www.w3apps.com

Jelp

Jelp is a context-sensitive help system for Java™ applications/applets and is written entirely in 100% Java. *Jelp* converts RTF files, generated from your favorite help authoring tool, to *Jelp* booklets that may be distributed with our viewer in your application/applet royalty free.

CreativeSoft
Joe Spinelli
214 373-8720
www.jelp.com

JGL™

ObjectSpace JGL™ is a powerful add-on for the JDK that provides a series of advanced collections and more than 50 generic algorithms. *JGL* is designed to complement, not replace, the basic features found in JDK and is ideal for enterprise Java™ developers. *JGL 3.0* enhances the *JGL* offering with distributed collection support, allowing the remote construction, access and persistence of all JGL containers using ObjectSpace Voyager, the agent ORB for Java. *JGL* is free for most commercial use at the Web address shown here.

ObjectSpace, Inc.
972 726-4100
www.objectspace.com

Jprinter

Jprinter improves Java™ printing by giving you complete printer-feature access while isolating you from browser and platform dependencies on Windows, Network Computers and UNIX. By leveraging and extending the JDK AWT (Abstract Windows Toolkit) graphics class with *Jprinter*, Java developers can add powerful WYSIWYG cross-platform printing to their Java applications and applets. Using a thin-client and network server architecture, *Jprinter* can print locally from Windows or across a network in a mixed-platform computing environment. Even without Windows, users can select printer models, change print properties and send their Java application output to PostScript Level 1 or 2, Hewlett-Packard's PCL 5 or 6 and Hewlett-Packard's HPGL-2RTL (large format plotter) devices.

Bristol Technology Inc.
Leslie Evans
203 798-1007
www.bristol.com

JSpell

JSpell is a 100% Java™ spell checking solution. It can be integrated with your applets and applications and is redistributable royalty free in your products. It includes two versions, standalone and client/server, which you can use depending on your applets and applications working environment.

JSpell offers international language support and is lightweight and fast.

Wall Street Wise Software
212 348-5031
www.wallstreetwise.com

Microline Component Toolkit 3.0

The *MCT* provides Java™ developers with a powerful set of advanced GUI objects for intranet and Internet application development. The tool kit includes grid, tree, tab, parse and progress components and supports JavaBeans™. Please visit the Web address below for more information and free downloads.

Microline Software Sales
408 245-5116
www.mlsoft.com

NetResults-API

NetResults-API is Innotech's Java™ toolkit for Java developers looking to add indexing, search and retrieval capabilities and *NetResults'* core services into their software products.

Innotech Multimedia Corporation
Stephen Paterson
416 492-3838
www.netresults-search.com

Nutmeg

Managing lists of information has never been easier than with our Java™ container classes. Some of the classes and features to meet your Java programming needs are arrays, indexed collections, ordered and sorted collections, sets, user-createable sub classes, etc. Users can define Java-style or Smalltalk-style error handling. Programmers can integrate their own custom sorting. *Nutmeg* has a simple, powerful and high-performance interface for generic "list management" tasks. *Nutmeg* is available for trial and purchase from the Web site below.

Thought, Inc.
Dan Wilson
415 836-9199
www.thoughtinc.com

Objective Blend 1.0

Objective Blend 1.0 is a package of over ten AWT extensions that makes writing, cutting-edge graphical applications. Some of the components include a tree control, tabled windows, masked edit, combo box and more!

Stingray Software, Inc.
Aris
919 461-0672
www.stingsoft.com

Objective Blend 1.1

Objective Blend 1.1 is a significant update to Stingray Software's popular family of class component libraries, incorporating many new features requested by Java™ developers. New features include JDK 1.1 and 1.02 support; Smart Edits for date, time and currency; calculator and calendar controls; enhanced tab controls; multicolumn list and tree controls; enhanced slider control; IDE integration; JavaBeans™ support and prebuilt Beans. *Objective Blend 1.1* ships with full source code and 60 days of technical support. It is available for $295 and subscriptions are available for $145.

Stingray Software
Shannon Stamey
919 461-0672
www.stingsoft.com

Objective Grid for Java™

Objective Grid for Java™ is a 100% Java™ grid control that features multiple sheet support, unlimited undo/redo, JDBC data binding and many other features.

Stingray Software, Inc.
Aris
919 461-0672
www.stingsoft.com

Objective Grid/J 1.1

Objective Grid/J 1.1 is a significant update to Stingray's popular grid control for Java™ developers. New features include JDK 1.1 and 1.02 support; new control types; enhanced database binding; performance improvement; new printing support; JavaBeans support and prebuilt Beans. *Objective Grid/J 1.1* ships with full source code and 60 days of technical support.

Stingray Software
Shannon Stamey
919 461-0672
www.stingsoft.com

ObjectSpace Voyager™

ObjectSpace Voyager™ Core Technology (*Voyager*) is an advanced, 100% Java™ Object Request Broker (ORB) designed as a Java-centric distributed computing platform. Its object model is based on the Java language, simplifying development and reducing time to market.

ObjectSpace, Inc.
972 726-4100
www.objectspace.com

PCX Graphics Pack

Display PCX images in your Java™ applets and applications. *PCX Graphics Pack* renders PCX images in either original or specified size, and draws in the background which frees your app to do other tasks. Java 1.0.2 and Java 1.1 AWT compliant.

Solid Logic Computer Solutions, Inc.
Bob Barker
612 949-0140
www.slogic.com

PECOS.net

PECOS.net extends the PECOS family of secure, multimedia, electronic commerce systems to the World Wide Web. Companies gain the competitive advantages of both our robust full-circle catalog and ordering systems – enabling interactive, enterprise-wide commerce and the convenience, openness and accessibility to the world-wide market of the Web.

Elcom Systems, Inc.
Pat Breslin
617 407-5003
www.elcom.com

Phaos Base Crypto

The *Phaos Base Crypto Toolkit* provides the core algorithms for cryptography in Java™. *Phaos Base Crypto* includes fast and lightweight Java implementations of DES, triple-DES and Blowfish encryption, MD5 and SHA message digests, Diffie-Hellman Key Agreement, public key support and much more. With *Phaos Base Crypto*, you'll write once, and run on any version of Java, from 1.0.2 to 1.1.4.

Phaos Technology Corporation
Lynn Hahn
212 929-7515
www.phaos.com

ProtoView CalendarJ

CalendarJ, a JavaBeans™ calendar component, is the third in the ProtoView JavaBeans line of products to be released. *CalendarJ* is a standalone calendar control with an easy-to-use, intuitive user interface. It features:

- Display of the currently selected date
- Complete font selection with 3D styles
- International language support
- DatePlus component with drop-down calendar
- Multi-month selection between calendars
- One month, quarter or half-year or full-year view
- Various color or border styles
- Dropdown month and year

ProtoView Development Corporation
Russell Frith
609 655-5000
www.protoview.com

ProtoView DataTableJ

The latest in a series of powerful grid components, the *DataTableJ* is based on its C++ predecessor, which was chosen by both Oracle and Informix to front-end their development tools.

This industrial-strength grid component is built for maximum speed and high performance. Its quick scrolling and crisp painting give Web applications professional quality and appeal. Built for handling large amounts of data with high-speed performance, the *DataTableJ* rises above other Java™ grids by easily accommodating thousands of rows.

ProtoView Development Corporation
Russell Frith
609 655-5000
www.protoview.com

ProtoView JSuite

The *Protoview JSuite* is a low-priced bundle of 17 feature-rich JavaBeans™ components. The *JSuite* includes the following products: CalendarJ, DataTableJ, TabJ, TreeViewJ and the WinJ Component Library.

ProtoView Development Corporation
Russell Frith
609 655-5000
www.protoview.com

ProtoView TabJ

The *ProtoView TabJ* is a flexible JavaBeans™ tab component. Along with a variety of font, border and shadow styles, the *TabJ* supports rounded or square tab styles, scrolling by tab or by page and placement of tabs above or below pages. The *TabJ* is available with or without source code.

ProtoView Development Corporation
Russell Frith
609 655-5000
www.protoview.com

ProtoView TreeViewJ

ProtoView TreeViewJ is a robust JavaBeans™ component for displaying outlines and tree hierarchies. It features standard TreeView behavior with in-place branch editing, vertical and horizontal scrolling and branch expand and collapse functionality. With an easy-to-use, flexible programming interface, its small size and quick load time make it the perfect addition to any Web site.

ProtoView Development Corporation
Russell Frith
609 655-5000
www.protoview.com

ProtoView WinJ

The *WinJ* Component Library is a rich collection of ProtoView JavaBeans™. While geared towards the professional developer, these feature-rich controls are simple enough for even the most basic Java™ programmer.

All of the *WinJ* components support these display options:

- Multiple border styles and shadows
- Aureole effect – This feature is exclusive to Protoview JavaBeans and creates a sunburst style color outline around each individual character of any display font (the color for this effect can be determined by the developer).
- Formatted display
- Odometer display
- Exclusive LED font in either bold or thin display

ProtoView Development Corporation
Russell Frith
609 655-5000
www.protoview.com

ScreamingBeans Fixed-Income Toolkit

The ScreamingBeans Fixed-Income Toolkit contains JavaBeans™ to aid in the development of fixed-income applications for GUI clients or application servers. The toolkit contains both visual and non-visual Beans; e.g., a GUI widget to adjust a yield curve business object, yield-to-yield calculators and fixed-income instruments.

Screaming Solutions
Mark Kerbel
416 783-1700
www.screamingsolutions.com

ScsGrid Control

ScsGrid Control enables you to have many of the common features found in grid controls for displaying and editing data fields. *ScsGrid Control* was designed to provide an easy-to-use grid control with fast uploads on the client machine. You have the ability to do background images, cell colors, freeze-column, multi-line header area, width auto-resize and many other graphically-oriented characteristics. Two versions are available: JDK1.0 and JDK1.1/JavaBean. *ScsGrid* may be purchased for as low as $39.00 (class files) and $79.00 (source code).

SofTech Computer Systems, Inc.
814 696-3715
www.scscompany.com

SSLava Toolkit

The *SSLava Toolkit* allows you to communicate securely in Java™ with the SSL (Secure Sockets Layer) protocol. Available since mid-1996, *SSLava* has been used successfully by numerous organizations – from Fortune 500 companies to e-commerce startups – for securing their Internet applications and services. Release 1.11 not only includes the complete SSL protocol implementation, but also includes such features as Diffie-Hellman with DSS support, SSL tunneling and JSAFE support for RSA cryptography. The *SSLava Toolkit* is 100% Java and is compatible with all JDK versions from 1.0.2 to 1.1.4.

Phaos Technology Corporation
Lynn Hahn
212 929-7515
www.phaos.com

Ultimate Grid for Java™

Ultimate Grid for Java™ supports true mouse scrolling, cursor changes during resize operations and column swapping. It is 100% Pure Java™ and includes full source code. The open architecture binds to any datasource, including proprietary.

Dundas Software Ltd.
Donna Marchand
416 239-7472
www.dundas.com

Vanilla Search

Add Unicode wild card searching to any Java™ app with *Vanilla Search*. *Vanilla Search* offers powerful pattern-matching search capability using industry standard "regular expression" syntax like Perl and Grep. *Vanilla Search* is very easy to use and reliable. It understands Java data-types such as string and character, patterns and searchable data of any length. It offers seamless integration with other Thought Inc. products and is available for trial and purchase from the Thought Inc. Web site below.

Thought, Inc.
Dan Wilson
415 836-9199
www.thoughtinc.com

Java IDEs

CodeWarrior Gold 10

CodeWarrior Gold, the world's best-selling suite of Macintosh development tools, offers everything you need for industrial-strength programming.

Metrowerks
Roger Aradi
512 873-4757
www.metrowerks.com

Discover Programming with Java™

This software/literature combination offers everything you need to start programming with Java™. You'll find the award-winning CodeWarrior IDE and extensive Java toolset, an electronic version of "*Learn Java on the Macintosh"* by Barry Boone with Dave Mark.

Metrowerks
Roger Acadi
512 873-4757
www.metrowerks.com

Grinder for Java™

Grinder for Java™ maximizes your programming power, software quality and portability, and object discovery and reuse. No other package allows you to explore Java™ from Sun, Microsoft or third party providers like *Grinder*. And we now support any standard Java compiler including Sun and Microsoft's (JDK's) for lightning fast compiles.

The Paradigm Exchange
James Earle
770 395-1705
www.tpex.com

GRIT 4

GRIT 4 Application Developer is an object-oriented software development tool set that enables the implementation of simple and complex applications across a wide range of platforms. *GRIT 4* features support for C/C++ and Java™. It is an integrated user interface design tool and offers database connectivity.

GFT Software Gmbh
Graham Byrne
+351-1-4781311
www.gftsoftware.com

Java™ WorkShop

Java™ *WorkShop* is a true multi-platform Java™ development environment. Designed for professional software developers, it delivers on the promise of Java write-once-run-everywhere, visual, intuitive application development, including applets, applications and JavaBeans™ components.

Java WorkShop provides a fully integrated graphical toolset which allows programmers to design, edit, compile, debug and tune Java applets and full-scale, client/server Java applications. Wizards for project management allow users to share, manage and distribute information to the whole development team. In addition, it comes with a Java profiler to help developers identify bottlenecks in their applications and tune for performance.

Sun Microsystems
Jill Reed
415 974-7242
www.sun.com

KAWA

KAWA is a simple yet powerful IDE for your Java™ development. *KAWA* is a wrapper on Sun's JDK. It is a Win 32 platform and is version independent. *KAWA* features include integrated syntax coloring editor, class browser, project manager, integrated context sensitive help and much more. Download a free evaluation from the Tek-Tools, Inc. Web site.

Tek-Tools, Inc.
Cindy Schlette
972 980-2890
www.tek-tools.com/kawa/

OrbixBuilder

OrbixBuilder is a family of CORBA development products for Java™ and C++ that plug into popular graphical development environments, such as Symantec Visual Café on Windows NT/95 and UIM/X on UNIX. *OrbixBuilder* includes the Orbix or OrbixWeb ORB, integrated graphical CORBA utilities, automatic code generation and visual CORBA tutorials.

Black & White Software, Inc.
Susan Karas
408 369-7400
www.blackwhite.com

PARTS for Java™

PARTS for Java™ is a complete visual programming environment that provides maximum productivity and ease of use, plus the ability to create distributed applications. *PARTS* offers complete support for JavaBeans™. Developers can also make use of CORBA and Java RMI to create distributed applications. *PARTS* features a full-featured graphical debugger, powerful code browsing tools and complete project management.

Objectshare
Thomas Murphy
714 513-3000
www.objectshare.com

PARTS for Java™ Pro

PARTS for Java™ *Professional* builds upon the high-productivity tools provided in PARTS for Java, adding integrated version control and the ability to automatically build applications utilizing JDBC. You also get support for the use and creation of ActiveX/COM components and support for the new JFC/Swing widget set.

Objectshare
Thomas Murphy
714 513-3000
www.objectshare.com

Passport IntRprise

Passport Corporation is the premier provider of easy to use tools for the development and deployment of Internet-enabled, enterprise-wide, fault tolerant client/server applications. *Passport IntRprise* is designed for creating highly interactive, event-driven applications and functions independently of the database, operating system, windowing system and network. Supports Windows 95, NT, 3.x, Unix (16 flavors).

Passport Corporation
Christopher Zosa
201 634-1100
www.passportcorp.com

PowerJ Enterprise

PowerJ Enterprise is a comprehensive, highly productive Java™ system that revolutionizes client/server and OLTP application development for the Web by enabling the creation, testing and deployment of enterprise-class applications. It is designed to deliver rich, universally accessible Java applets, applications and servlets for low-cost deployment and maintenance in a multi-tier architecture to extend the reach of your enterprise.

Sybase
Tracey Ryan
519 883-6365
www.sybase.com

Prometheus System

The *Prometheus System* is a hardware system designed by Indelible Blue specifically to serve the needs of Java™ software developers. *Prometheus* features IBM's VisualAge for Java Professional development environment, pre-loaded across three different operating systems: Microsoft's Windows 95, Windows NT Workstation and IBM's OS/2 Warp 4.

Indelible Blue, Inc.
Kelly Edwards
800 776-8284
www.indelible-blue.com

SuperCede for Java™ and ActiveX

SuperCede for Java™ is the first in a line of SuperCede development environments. It has an on-the-fly environment which enables developers to modify running applications and to "work at the rate that they think". Its "Flash Compiler" technology brings C++ levels of performance to Java by generating true native executables.

Supercede, Inc.
Jim O'Farrell
206 637-2488
www.supercede.com

SuperCede Java™ Edition

SuperCede Java™ *Edition* is an on-the-fly Java™ development environment. *SuperCede*'s patent-pending flash compiler lets you add new methods and behaviors and modify existing code, on-the-fly, without stopping your running application. Comparisons with other Java tools show that *SuperCede* developer productivity is enhanced by a factor of 5 due to the flash compiler technology. *SuperCede* also supports the generation of Java Byte Code PLUS Windows 32-bit executables and DLLs, thus shattering the Java application performance barrier. *SuperCede*'s Visual Basic-like user interface makes the development of Java applications and applets as easy as "drag and drop". Additionally, *SuperCede*'s learning is fast because MindQ's "Learning Java with *SuperCede*" interactive CD tutorial ($49.95 value) is included FREE with *SuperCede Java Edition.*

SuperCede Corporation
Jim O' Farrell
206 637-2488
www.supercede.com

UIM/X Platinum

UIM/X Platinum Toolsuite Edition 3.0 (PTE) vastly extends the ability of UIM/X users to design platform independent interfaces. Moreover, *PTE* catapults your legacy code for use in Java™ and Internet development. Using our sophisticated Migration Assistant and Refinement Technology (MART), *PTE* will automatically convert your Motif, UIM/X and Cross Platform Toolset (CPT) interfaces into 100% Pure Java.

Visual Edge Software
Eric Rubin
408 973-7823
www.visualedge.com

Vibe DE

Vibe is an integrated development environment used to build and deploy Java™ based solutions. *Vibe* combines an intuitive user interface with a Java virtual machine, compiler, debugger, editing tools, interface construction tools and extensive runtime classes into a comprehensive environment. Using *Vibe*, developers can write an application once and deploy it on Windows, OS/2, Macintosh and UNIX without making changes to the source code.

Visix Software, Inc.
Katherine Johnson
703 758-2833
www.visix.com

Visaj

Visaj is a new breed of Java™ development tool, written entirely in Java. *Visaj* gives developers the ability to build truly cross-platform Java applications, a key accomplishment in the platform-neutral world of the Internet and World Wide Web. *Visaj* enhances developers' productivity by providing an intuitive visual environment for building 100% Pure Java applications which greatly reduces development time. *Visaj* provides layout editors for all AWT layout controls, allowing for the development of complex interface designs that are difficult to achieve with other development tools.

Imperial Software Technology
650 688-0200
www.ist.co.uk

WingEditor

WingEditor is a simple Java™ IDE implemented in Pure Java. It provides a Java-oriented editor environment, an error-browser capable environment and a GUI debugger.

WingSoft Company
Kathy Ho
510 744-1866
www.wingsoft.com

WinGEN® for Java™ v 2.0

WinGEN for Java™ is a 32-bit source code application generator for Java™.

WinGEN
David Johnson
519 742-9521
www.wingen.com

JavaBuyersGuide.com

1998 Java Buyer's Guide

The Most Complete Java Source of the year...

Available in three convenient formats:

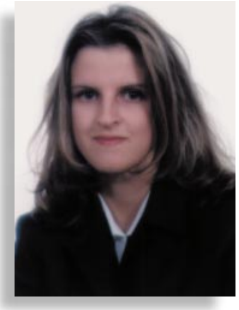
1. Print,
2. CD-Rom &
3. Online

(JavaBuyersGuide.com)

SUBMIT YOUR PRODUCT TO The Buyer's Guide

Click on this button at **JavaBuyersGuide.com** to get your product into the **1998 Buyer's Guide!**

Gail Schultz
Editor,
Java Developer's Journal
gschultz@sys-con.com



Christy Wrightington
Editor,
Java Buyer's Guide
christyw@sys-con.com

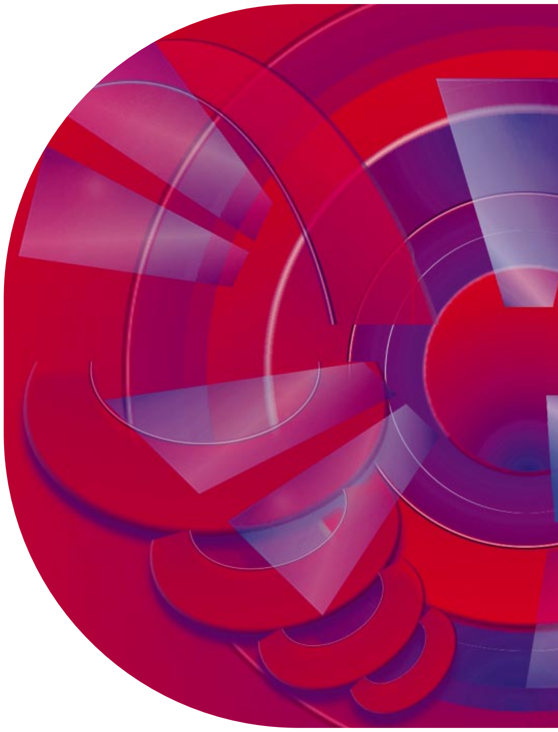
Java Developer's Journal DAILY

JavaBuyersGuide.com

INTERFACING TRANSACTION SERVICES

A critical task in enterprise computing

by Maros Cunderlik



Introduction

With wider acceptance of component development and distributed object technologies, applications can be 'assembled' as a set of collaborating components. For nonvisual business components, where delivery time, integration costs and maintainability are the determining factors of success, Java is increasingly the implementation language of choice. When writing components in Java, we must be able to access a set of services commonly provided to business objects in enterprise computing, most notably resource management, object pooling and transaction processing.

In this article, we will explore how Java components can access transaction services. We will describe the common framework for transaction processing: Distributed Transaction Processing (DTP) Model. We will also take a closer look at two common applications of DTP-like transaction models. In Part I, we will cover the concepts of the Microsoft Transaction Server (MTS), and OLE Transactions and build a sample MTS-based order validation application in Java. In Part II, we will explore the CORBA Object Transaction Service (OTS) and its commercial implementation. In the end, you will have a good understanding of transaction models, their underlying principles and how Java applications can participate in transaction processing. Before we start, I want to emphasize that the purpose of this article is to focus on understanding transaction processing concepts. We will not attempt to 'rate' various commercial implementations of TP monitors or component standards. You should also have a basic understanding of COM and CORBA.

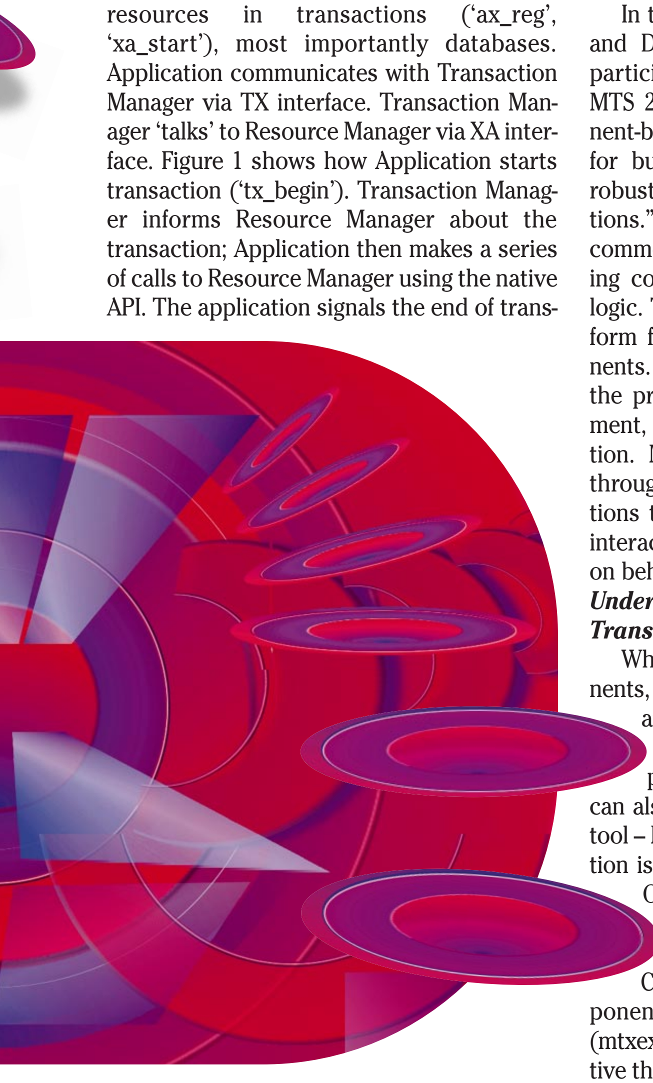
Distributed Transaction Processing – X/Open DTP Model

Conceptually, a large number of business problems are transaction-based. Inventory management, order processing and job scheduling are only a few of the common manufacturing tasks that can be viewed as transactional. The concept of transaction is the centerpiece of all banking systems. Banks most certainly do not want to credit an account without debiting another and vice versa. In general, we describe transactions as:

- **Atomic:** A transaction is a single logical unit of work – all changes are always either successfully committed or rolled back in case of failure. A transaction will not be partially committed or partially aborted.
- **Consistent:** A transaction is a unit of work which takes a system from one consistent state to another.
- **Isolated:** While a transaction is executing, its partial results are hidden from other entities accessing the system, and
- **Durable:** The results of a transaction are persistent.

These four properties are also known as ACID properties and form the foundation of transactional systems. In truth, it is not that difficult to achieve ACID properties in highly integrated systems. However, maintaining ACID properties becomes increasingly complex in systems spanning multiple machines, platforms, databases or database systems.

To address transaction processing in the distributed and heterogeneous environment, X/Open group defined the Distributed Transaction Processing (DTP) Model. The basic X/Open DTP model consists of three main entities and two interfaces. Transaction Manager manages the scope (context) of transaction, provides a unique transaction identifier and determines the outcome of transaction. Resource Manager enlists transaction resources in transactions ('ax_reg', 'xa_start'), most importantly databases. Application communicates with Transaction Manager via TX interface. Transaction Manager 'talks' to Resource Manager via XA interface. Figure 1 shows how Application starts transaction ('tx_begin'). Transaction Manager informs Resource Manager about the transaction; Application then makes a series of calls to Resource Manager using the native API. The application signals the end of transaction by calling Transaction Manager ('tx_commit'). At this point, Transaction Manager starts the two-phase commit process using XA interface. In a simplified view of two-phase commit, Transaction Manager instructs all participating Resource Managers to prepare the commit ('xa_prepare'). Resource Managers communicate with their resources to prepare commit and send an acknowledge massage ('commit ready', or 'commit aborted') to the Transaction Manager. The Transaction Manager awaits the responses. If all resource managers responded with 'commit ready', the final commit command ('xa_commit') is issued by the Transaction Manager and all changes are committed. The information about successful commit is communicated back to the client. If one or more resource managers could not prepare to commit, the transaction manager issues the rollback message and all resource managers ensure that the initial state is restored. Finally, the transaction manager informs the client that the changes failed and the initial state was restored.



Microsoft Transaction Server (MTS)

In this section, we will explore how COM and DCOM-based components create and participate in transactions. According to MTS 2.0 documentation: "MTS is a component-based transaction processing system for building, deploying, and administering robust Internet and Intranet server applications." More importantly, MTS provides a common framework of services for developing components that encapsulate business logic. The MTS runtime is a middle-tier platform for running these COM-based components. Arguably, the key benefits of MTS are the programming model, resource management, object pooling and ease of administration. MTS provides transactional services through leveraging existing OLE Transactions technology. MTS runtime then simply interacts with OLE Transactions participants on behalf of the business components.

Understanding MTS and OLE Transactions

When developing MTS application components, we declare component's transaction attribute, which determines whether associated resources should participate in the transaction. This attribute can also be changed via MTS administrative tool – MTS Explorer. The transaction information is contained in Context Object. Context Object is created automatically for each MTS component. Among other properties, it also contains Transaction ID. Context Object is loaded with MTS component as a part of MTS Executive (mtxex.dll). The components and MTS executive then execute either in client's process or, more typically, in separate host process provided by MTS (mtx.exe).

MTS Executive has an 'intimate' knowledge of the application component and interacts with MTS runtime on its behalf. Figure 2 illustrates the core components of MTS runtime that participate in transaction processing. If a component is declared as transactional, MTS runtime contacts Microsoft Distributed Transaction Coordinator (MS DTC) and automatically starts a new transaction, or joins the existing one, when the component is instantiated.

As illustrated in Figure 2, MTS Executive requests resources from Dispenser Manager. Dispenser Manager maintains resource pools with the help of Inventory Statistics Manager and Holder. The Holder component lists the resource inventory for each Resource Dispenser. Resource Dispenser has an 'intimate' knowledge of the underlying resource. It knows how to allocate and reclaim the resource. On start-up, Resource Dispenser registers with Dispenser Manager and enlists its resource with Resource Manager and Holder. The dispenser usually provides public API or COM interfaces that are used by client applications to access the resource. Resource Manager is a component that is part of the OLE Transactions model. The resource manager communicates with MS DTC and participates in a two-phase commit process.

Dispenser Manager periodically (every 10 seconds in MTS 2.0) interrogates each Holder to allow them to readjust their inventory. Each Holder, in turn, calls Inventory Statistics Manager to suggest the appropriate inventory levels. As a result, Holder instructs Resource Dispenser to either create or destroy some resources. After the resource is allocated, Dispenser Manager asks MTS Executive whether the application component is running within a transaction. If so, Dispenser Manager asks Resource Dispenser to enlist the resource in the transaction. Resource Dispenser, in turn, contacts Resource Manager to enlist the resource with MS DTC.

OLE Transactions

As we mentioned earlier, MTS relies on OLE Transactions to provide full support for ACID properties. The OLE Transactions model is based loosely on the DTP model. It also defines three entities in the transaction processing: the client application, the resource manager and the transaction manager. OLE Transactions protocol differs from the DTP model mainly because it is object-based and provides support for COM interfaces. As a proprietary standard, OLE Transactions take advantage of Windows-specific features and can be extended to support transactions across resources that were traditionally thought of as non-transactional (e.g., voice, video, directory services). Microsoft provides an implementation of the transaction manager – MS DTC. OLE Transactions also define a set of COM interfaces that MS DTC and each resource manager support. The protocol details how the client, MS DTC, and the resource managers carry out the two-phase commit process. The commit process is fundamentally very similar to the one described in the DTP model.

MTS Programming in Java

One of the major advantages of the MTS programming model is its simplicity. Virtually all of the detail about how the transactions are carried out is hidden from the client. Figure 3 depicts the complete set of Java interfaces and classes exposed by MTS in com.ms.mtx package.

Because MTS is COM-based, the process of building an MTS application is basically

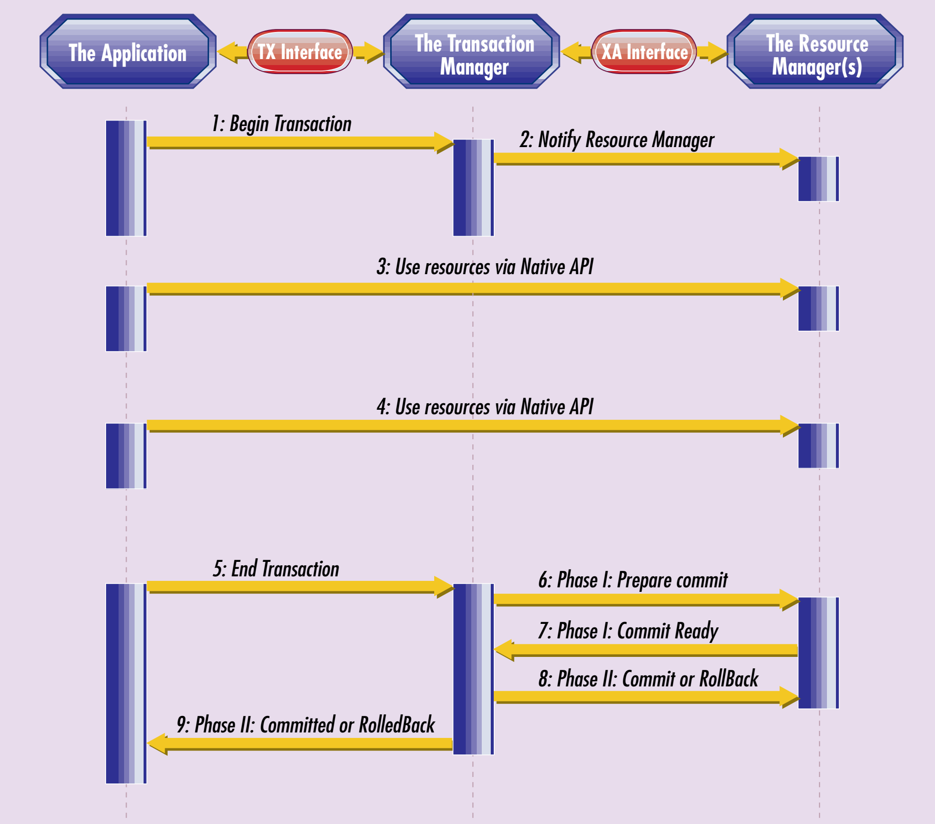

The Application
TX Interface
The Transaction Manager
XA Interface
The Resource Manager(s)
1: Begin Transaction
2: Notify Resource Manager
3: Use resources via Native API
4: Use resources via Native API
5: End Transaction
6: Phase I: Prepare commit
7: Phase I: Commit Ready
8: Phase II: Commit or RollBack
9: Phase II: Committed or RolledBack

Figure 1: X/Open DTP Model – Sequence Diagram

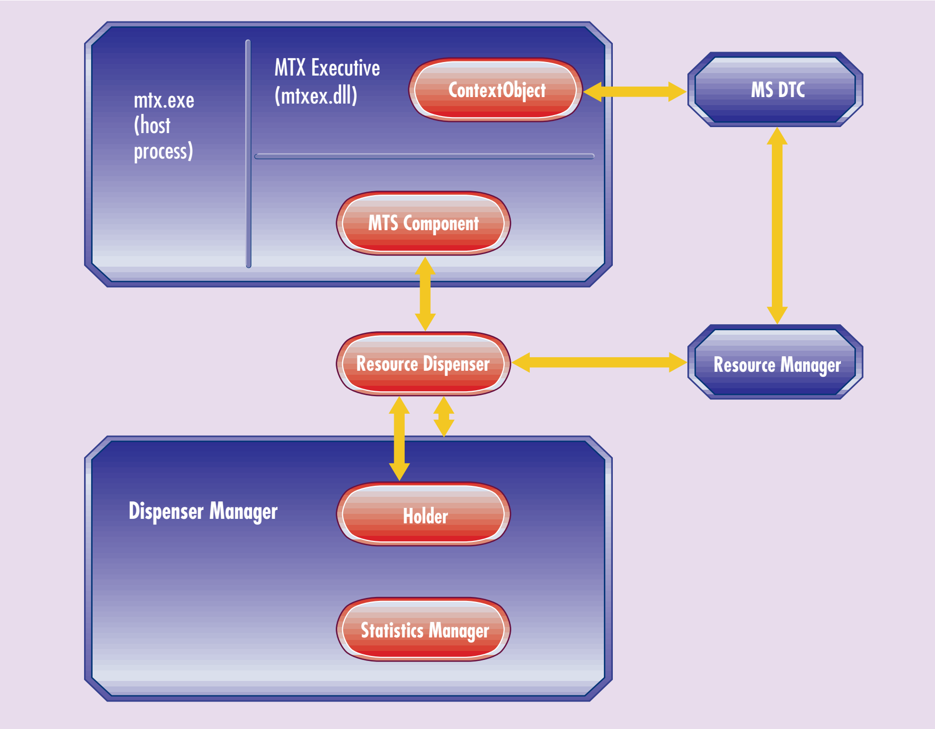


Figure 2: MTS Architecture

the same as building COM components in Java. However, when attempting to create scalable systems we must consider issues such as an objects' state, persistence, granularity, object pooling and resource management. There are key differences between building the scalable applications in MTS and plain COM. For example, resource management and object pooling make acquiring and destroying resources in MTS relatively 'cheap'. Therefore, the traditional model of acquiring resources once and then holding onto them as long as they are needed is no longer valid in MTS. The 'typical' MTS component would create and destroy resources as needed and commit/rollback changes as soon as possible. The component sends commit/rollback messages by calling SetComplete/SetAbort methods on IObjectContext interface. In addition to committing changes, the SetComplete/SetAbort calls cause the object instance to be 'recycled' using the object pool. MTS version 2.0 supports the most basic object pooling – instances are created and destroyed on demand, no object pool is maintained. As a result, clients cannot rely on MTS objects to maintain their state after MTS objects issue SetComplete/SetAbort commands. This behavior constitutes the single most important paradigm change in the COM programming model.

MTS also introduces a new security model on top of NT domain security. The centerpiece of MTS security model is the concept of roles. A role usually refers to a group or type of users for a set of components; e.g., Manager, Sales Representative or Customer.

To demonstrate these concepts we will build a sample order validation system. First a little background. Our company – M.C. Manufacturing (MCM) – is adopting a new business model. It allows its customers to order any sample and experimental products. After a few successful sample orders, the customers would presumably place a large order and the product could be eventually added to the standard product line. MCM's IT department must now build an order validation system that could preprocess the large volume of custom-made products. Pre-processing orders will help to minimize the lead times. We will attempt to build the system in the following steps:

1. Design conceptual-level interfaces that describe desired systemís behavior.
2. Design physical, MTS COM interfaces and classes using Interface Definition Language (IDL).
3. Map IDL interfaces and classes into Java interfaces. Create Java implementation classes.
4. Create and deploy the system.

Designing conceptual-level interfaces

MCM's new system must accomplish a single logical step – an activity, in MTS speak: process an order. As it turns out, to validate an order the system must: validate order attributes (size, quantity, etc.), make sure that the item satisfies general rules (production specifications, etc.), notify demand planning system and contact work scheduling system to pre-schedule plant capacity. For simplicity, we assume that there is only one item per order. We also omit security, which usually would be part of the system. Given these facts, Listing 1 shows a simplified set of Java interfaces describing the new system. IItem interface represents an item on an order. Before submitting an order, each item must be validated by calling IItem.ValidateAt-

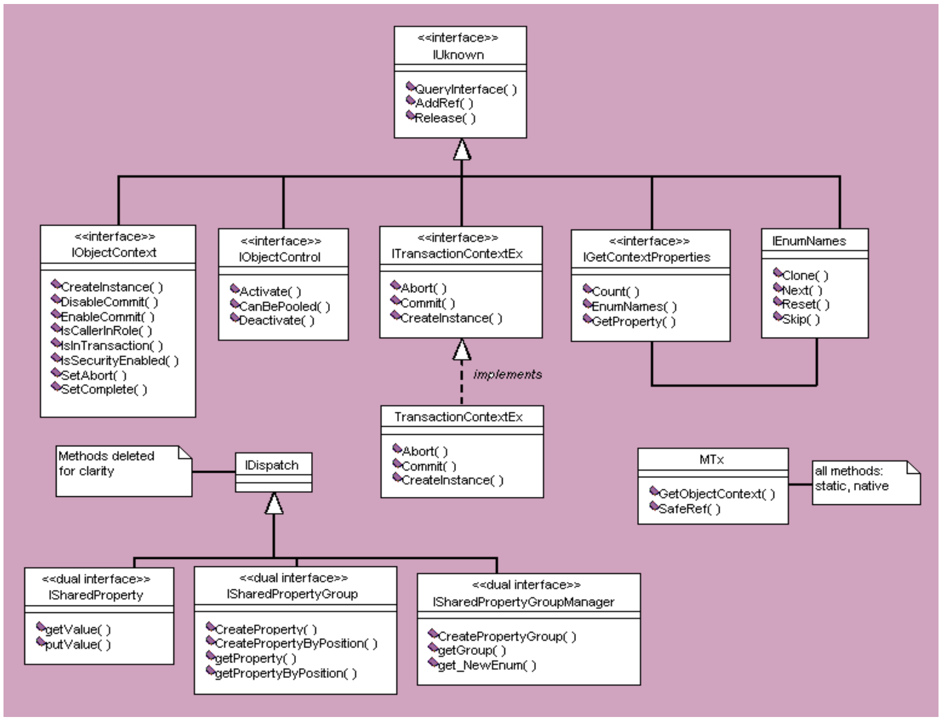
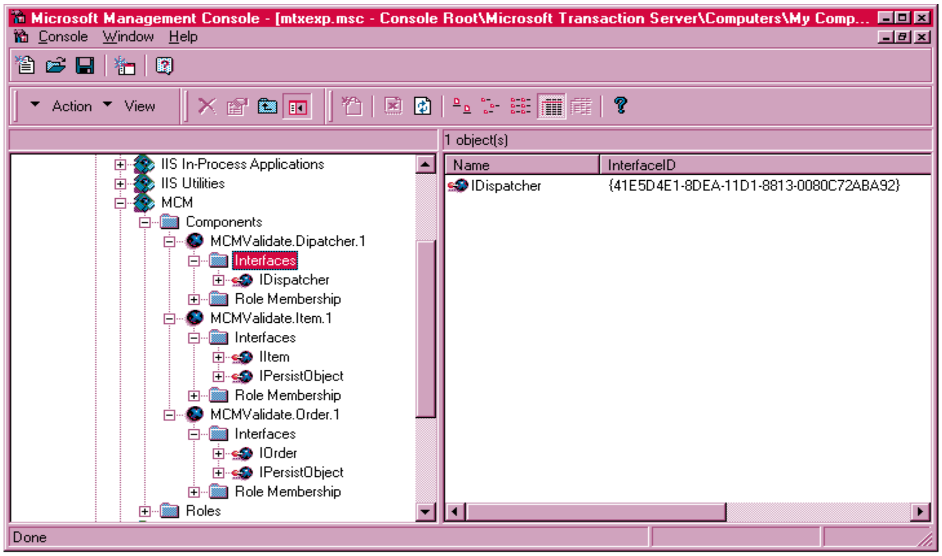


Figure 3: Microsoft Transaction Server: com.ms.mtx. Java package



1 object(s)

Name	InterfaceID
IDispatcher	{41E5D4E1-8DEA-11D1-8813-0080C72ABA92}

Figure 4: MTS Explorer – MCM Order Validation System Components

tributes() and IItem.ValidateRules(). Each order, represented by IOrder interface, contains one item. Using IOrder interface, we can add an item (IOrder.AddItem()), and submit the order (IOrder.Submit()). When submitting an order, the new system uses IDispacher interface to interact with scheduling system (IDispatcher. Schedule()), and demand planning system (IDispatcher. InformDemandPlanning()).

Designing physical, MTS COM interfaces using COM IDL

COM IDL provides a language-neutral way of defining interfaces in COM. These definitions are 'compiled' by an MIDL compiler that produces necessary marshaling code, header files and 'type library' – a binary representation of IDL code. The type library is then used to generate language mappings for interfaces and implementation classes described in IDL. Before we create an IDL file, however, we must re-design the conceptual-level interfaces with respect to MTS and COM.

Traditionally, inside of Order class we would also create an instance of Item and Dispatcher classes and use them when validating and submitting order. These classes would, therefore, remain in memory for the lifetime of order instance. Clearly, this is not the best use of system resources. Consider the following scenario. When entering an order, we create an instance of Order component. When an item is added, we will obtain and keep a reference to it for the lifetime of instance of Order class. We will use an object manager to create an instance of Item class as needed. The object manager will 'know' how to create an instance of a class. It also 'knows' how to restore object's state based on the reference. What we are describing here is a process of adding persistence. COM has a concept of monikers that are used for implementing persistence. However, monikers are not supported in Java and, therefore, we must implement persistence ourselves. Our implementation is simple. Every class will implement an IPersistObject interface that defines how to load state, save state, return a persistent object reference to itself and notify about changes in state. Clients then use a static BindToObject() method of PersistManager class to recreate state of persistent objects. Listing 2 shows the IDL definition of IPersistObject interface.

Listing 2 describes the IDL definitions of all interfaces and COM implementation classes. Note that the IDL file contains TRANSACTION REQUIRED and JAVACLASS attributes. First attribute marks the component as transactional and causes MTS to start a new transaction or include it in the existing one. The JAVACLASS attribute provides a class name of concrete Java implementation class. For example, Java MCMValidateImpl.Order class will implement COM class COrder.

Mapping IDL files and creating implementation classes

We use a Jactivex tool to generate Java code from the type library. The Jactivex tool is included in Microsoft's Java SDK. Jactivex by default generates .java files for all interfaces and coclasses (COM classes) in the type library. In Listing 3 we describe the batch file that automates all necessary commands to create MTS components. Listing 4 shows the implementation classes for Order, Item and Dispatcher components. All business implementation logic is omitted for clarity. Note that all business classes implement IPersistObject to provide persistence support. All business classes also implement the IObjectControl interface. By implementing this MTS interface, objects can declare their support for object pooling. If an object supports pooling, the Activate() method is called when a new instance is assigned from the object pool. Therefore, rather than using constructor, you should always put any initialization code into Activate() method.

Implementation of Order and Dispatcher components illustrates the importance of state awareness in MTS. Since we want to keep order's state until it is submitted, AddItem() method does not call SetComplete(). On the other hand, Submit() method uses SetComplete()/SetAbort() to commit changes and to recycle the object instance. Dispatcher component is an example of a so-called 'stateless' component. It does not have any private state, all information is passed in via parameters and each method calls SetComplete()/SetAbort() upon completion. This design pattern minimizes use of system resources and also allows clients to use Dispatcher component without having to restore the component's state between method calls. In implementing large systems, it is clearly critical for clients to fully understand the MTS components' state behavior and use them accordingly.

Creating, and deploying MTS components.

We can now compile all Java classes and create an in-process, MTS-compatible dynamically linked library using the exegen tool shipped with MTS (see Listing 4). After creating MCMValidate.dll, we are ready to deploy components in MTS. First we create a new package and then import the newly created DLL by using 'Install new component(s)' option in MTS Explorer. Finally, MCMValidate.Order.1, MCMValidate.Item.1, and MCMValidate.Dispatcher.1 components will appear in MTS Explorer (see Figure 4). You can verify the install by examining component properties in MTS Explorer.

Conclusion

Transaction processing is one of the critical tasks in enterprise computing. To write effective Java business components, we must be able to access transaction services. Transactions are usually defined by four properties: atomicity, consistency, isolation and durability. One of the most common frameworks for providing transaction support is the Distributed Transaction Processing Model by X/Open. The DTP model defines the set of standard interfaces, entities and behaviors for implementing transactions with ACID properties.

Microsoft Transaction Server (MTS) has emerged as one of the most important frameworks for component development. MTS provides COM components with common services such as resource management, object pooling and transactions. MTS components access transaction services by using OLE Transactions technology. MTS runtime participates in OLE Transactions on behalf of MTS components. In addition to transaction processing, MTS introduces the new COM programming model. To build effective MTS components, we must consider additional design issues such as state management, resource management and object pooling.

In Part II we will explore the CORBA OTS specification. We will review and explain OTS-defined interfaces, entities and behaviors. We will also examine a commercial implementation of OTS, and demonstrate how Java clients can interact with OTS entities.

References and Resources

Arnold, K., and Gosling, J., "*The Java Programming Language*", Addison-Wesley, Reading, MA, 1996

Box, D., "*Essential COM*," Addison-Wesley, Reading, MA, 1998

Chappell, D., "How Microsoft Transaction Server Changes the COM Programming Model," *Microsoft Systems Journal,* January, 1998, pp. 19-28.

Sessions, R., "*COM & DCOM: Microsoft's Vision for Distributed Objects"*, John Wiley & Sons, New York, NY, 1998

"*Distributed TP: XA Specification, X /Open Document C193.*", X/Open Company Ltd., Reading, UK, 1992

Microsoft Transaction Server 2.0 online Help.

Microsoft Transaction Server SDK (ftp.microsoft.com/bussys/viper/SDK).

Microsoft Developer Network, "Microsoft Transaction Server", January 1998 (www.microsoft.com/msdn)

Maros Cunderlik is a consultant for Connect Computer Co., a regional consulting and system integration firm based in Minneapolis, MN. He focuses on OO design and distributed object architecture. He can be reached at maros.cunderlik@connects.com

maros.cunderlik@connects.com

Access the source code for this and other articles appearing in this issue at JavaDevelopersJournal.com

ADVERTISER INDEX

Advertiser		Page
3D Graphics www.threedgraphics.com	310 553-3313	15
Bristol Technology www.bristol.com	203 438-6969	57
COSMO cosmo.sgi.com	888 912-6766	70
Greenbrier & Russel www.gr.com/java	800 453-0347	13
Halcyon www.halcyon.com	408 378-9898	21
IBM www.ibm.com	800 426-5900	50 & 51
iEC-Expo www.iec-expo.com	800 222-8734	77
ILOG www.ILOG.com	800 367-4564	75
Imperial www.Imperial.com	415 688-0200	58
InnoVal www.innoval.com	914 835-3838	19
KEO www.keo.com	978 463-5900	55
KEO www.keo.com	978 463-5900	69

Advertiser		Page
KL Group Inc. www.klg.com	800 663-4723	84
Live Software www.livesoftware.com	619 643-1919	33
MindQ www.mindq.com	800 847-0904	55
Object Matter www.objectmatter.com	305 718-9109	58
Object Management Group www.omg.org	508 820-4300	81
The Object People www.objectpeople.com	919 852-2200	23
Open Link www.openlinksw.com	800 495-ODBC	25
Phaos www.Phaos.com	212 229-1450	63
PreEmptive Solutions www.preemptive.com	216 732-5895	73
ProtoView www.protoview.com	609 655-5000	3
Rogue Wave www.roguewave.com/ad/graph	800 487-3217	29
Sales Vision www.salesvision.com	704 567-9111	79

Advertiser		Page
Schlumberger www.cyberflex.austin.et.slb.com	800 732-6868	43
SilverStream www.silverstream.com	888 823-9700	83
Sockem Software www.sockem.com	814 696-3715	81
Stingray Software Inc. www.stingsoft.com	800 924-4223	2
SunTest www.suntest.com	415 336-2005	4
Symantec cafe.symantec.com	800 453-1059 ext. 9NE5	11
SYS-CON Publications www.sys.con.com	914 735-1900	26
SYS-CON Publications www.sys.con.com	914 735-1900	46, 47
Thought, Inc. www.thought.com	415 836-9199	61
Visionary www.visionary.com	800 660-9633	67
webMethods www.webmethods.com	703 352-8501	39
Zero G. Software www.zerog.com	415 512-7771	54

A Common Coding Style for Java

by Achut Reddy

Coding style means nothing to a compiler; it will happily accept any legal code, no matter how badly formatted. Not so for us humans. Our ability to read and understand a program written by another is extremely sensitive to format. Imagine trying to read this article if every letter was in a different font and each line randomly aligned. Clearly, format affects readability, and it is just as true for reading programs as for text.

A consistent style also leads to lower maintenance costs by making it easier to debug and enhance the code. This is especially true for teams of programmers working on the same project, who must often read and edit each other's code. In this case, it is important that each team member follow not only a consistent style, but the same style. A common style also enables the development of automated tools to assist in program development, such as formatting tools and editor macros.

In spite of these benefits, some developers are reluctant to adopt a common style. One reason might be a belief that it requires too much effort. It is true that it takes a bit of time initially to learn the new rules (and unlearn old bad habits). However, once learned, they become automatic, and it takes no more effort to write code in a good style than in a poor style. In fact, it probably *saves* time by allowing programmers to concentrate on the semantics of the program, instead of consciously worry about the right format for each statement.

A larger reason for lack of a consistent style, however, is the lack of widely available, comprehensive and well thought out coding standards. This article is one attempt at improving this situation.

Space does not permit the presentation of an entire coding standard here. Rather, this article gives a selected set of the most important coding style rules, in somewhat condensed form. Listing 1 shows a code example demonstrating most of these style rules.

Indentation

One of the most important rules addresses the horizontal alignment of source lines. **Line indentation is always four spaces per indentation level.** [This is a difference from the predominant indentation style of eight spaces used in C programs; it is an acknowledgment that typical Java programs tend to have more levels of nesting than typical C programs.] The construction of the indentation may include tabs as well as spaces in order to reduce the file size; however, do *not* change the hard tab settings of your editor to accomplish this. Hard tabs *must* be set every 8 spaces. Soft tabs may be set every 4 spaces if your editor supports them.

Line Length

Lines should be limited to 80 columns. Longer lines should be broken into one or more continuation lines, as needed. All the continuation lines should be aligned and indented from the first line of the statement. The amount of the indentation depends on the type of statement.

If the statement must be broken in the middle of a parenthesized expression, such as for the parameter list in a method invocation or declaration, the next line should be aligned with the first character to the right of the first unmatched left parenthesis in the previous line. In all other cases, the continuation lines should be indented by a full standard indentation (4 spaces).

Braces Style

The preferred style for placement of braces in compound statements is the style commonly referred to as the "K&R" style. [Named after Brian Kernighan and Dennis Ritchie, authors of the first book on the C language.] Unlike C, in Java this style is used in *all* cases, including class and instance initializers and array initializers. This style is specified as follows:

1. The opening left brace "{" is at the end of the line beginning the statement.
2. The closing right brace "}" is alone on a line, indented to the same column as the first line of the statement.
3. The statements inside the enclosed braces are indented one more level than in the statement.

Naming Conventions

Another aspect of coding style that makes a big difference is consistency of naming.

Package names should use only lowercase letters and digits, and no underscore. If the package will be publicly distributed, prepend a unique prefix to the package by using the Internet domain name components in reverse order (e.g., `com.sun, graphics.util`).

All class and interface names should use the *InfixCaps* style. Start with an **upper case** letter and capitalize the first letter of any subsequent word in the name. All other characters in the names are lower case. Do not use underscores to separate words. Class names should be nouns or noun phrases. Interface names depend on the purpose of the interface. If it is primarily to endow an object with a particular *capability,* then the name should be an adjective that describes the capability (e.g., `Searchable, Sortable`). Otherwise, use nouns or noun phrases.

Variables (except static final types, see below) use the *infixCaps* style. Start with a **lower case** letter and capitalize the first letter of any subsequent word in the name, as well as any letters that are part of an acronym. all other characters in the name are lower case. Do not use underscores to separate words. Variable names should be nouns or noun phrases (e.g., `numberOfPoints`).

Static final primitive types being used as constants should be all upper case, with underscores separating words (e.g., MAX_BUFFER_SIZE).

Method names also use the infixCaps style. Method names should be verbs or verb phrases (e.g., `drawCircle()`).

White Space Usage

Blank lines can improve readability by grouping sections of the code that are logically related. A blank line should also be used:

- After the package declaration and import section
- Before a class or method declaration
- Before a block or single-line comment, unless it is the first line in a block.

A single blank space (not tab) should be used to separate all tokens in a statement, except for the following. Blanks should *not* be used:

- Between a constructor or method name and its opening parenthesis
- Before or after a . (dot) operator
- Between a unary operator and its operand
- Between a cast and the expression being casted
- After an opening parenthesis or square bracket, or before a closing parenthesis or square bracket

Comments

Comments can help the reader understand the code. Here are some general guidelines for comment usage:

- Comments should help a reader understand the purpose of the code. They should guide the reader through the flow of the program, focusing especially on areas which might be confusing or obscure.
- Avoid comments that are obvious from the code, as in the famously bad comment example:

```
i=i+1;    //Add one to i
```

- Remember that misleading comments are worse than no comments at all.
- Avoid putting any information into comments that is likely to become out-of-date.

About the Author

Achut Reddy is the Project Lead for Java Performance Tools at Sun Microsystems in Menlo Park, CA. Achut has been with Sun for 9 years and has worked on a number of products, including Java WorkShop, C/C++ Workshop and SPARCworks. He received his degree in Computer Engineering from UCLA and has worked in the software industry for 15 years.

Listing 1.

```
Example code demonstrating coding style rules)
/*
 * @#CodingStyleExample.java         1.0 98/01/23 Achut Reddy
 *
 * Copyright (c) 1994-1998 Sun Microsystems, Inc. All Rights Reserved.
 */
package com.sun.examples;

import java.applet.Applet;
import java.awt.Point;
/**
 * A class to demonstrate good coding style.
 */
public class CodingStyleExample extends Applet implements Runnable {
    static final int              BUFFER_SIZE = 4096;    // default buffer size
    StringBuffer               name;                                    // my name
    Point                            starshipCoordinates[];        // ship locations
    /**
     * Compute the total distance between a set of Points.
     * @param starshipCoordinates the locations of all known starships
     * @param numberOfPoints       the number of points in the array
     * @return the total distance
     */
    public int computeDistances(Point starshipCoordinates[],
                                   int numberOfPoints) throws Exception {
        int     distance = 0;    // accumulates distances
        // Compute distance to each starship and add it to the total
        for (int i = 0; i < numberOfPoints; i++) {
            distance += Math.sqrt((double)((starshipCoordinates[i].x *
                                   starshipCoordinates[i].x) +)
                                   (starshipCoordinates[i].y *
                                   starshipCoordinates[i].y)));
        }
        if (distance > 100000) {
            throw new Exception
        }
        return distance;
    }
    /**
     * Called whenever Thread.start() is called for this class
     */
    public void run() {
         try {
              name.append  ("X");
              System.out.println(name);
         } catch  (Exception  e) {
              name = new StringBuffer(BUFFER_SIZE);
         }
     }
}
```

Microsoft Announces Visual J++ 6.0 Technology Preview

by **Bill Dunlap**

The introduction of Java brought several advances to the discipline of software development. Based on lessons learned from years of building software in C++, the Java language was designed to simplify the development of object-oriented, network-aware applications. It brimmed with potential and was accompanied by a wide variety of promises, including enhanced programmer productivity, seamless cross-platform deployment and an architecture for building large-scale applications.

Over the past three years, the promise of Java has been put to the test repeatedly. What has emerged is a better picture of what customers want from their software as well as what Java can and cannot do. To its credit, the design of the Java language has demonstrated tremendous programmer productivity benefits over C++. Language features such as automatic garbage collection, exception handling and inherent multi-threading support allow developers to more rapidly write high quality code. Unfortunately, Java has not delivered on some of the other promises. Despite major investment from top platform and Internet technology vendors, cross-platform execution of large applications has proven easier said than done. The sensitivity of the Java runtime to differences in underlying hardware architectures, operating systems and the virtual machines themselves has given rise to the developer cry of "write once, debug everywhere." Further, the sacrifices in performance, functionality and integration necessary to pursue portability have made it difficult to deliver compelling applications.

> *"... a better picture of what customers want from their software as well as what Java can and cannot do."*

The high-tech industry is a very competitive environment, one based on meeting customer demand. Given a choice between two applications, if one provides more features and better integration with their present hardware and software investments, users will chose the richer, more functional one every time. In an article in the *San Jose Mercury News* (Feb. 9, 1998), Eric Schmidt, Sun's former Chief Technology Officer, admitted as much when he said, "The problem with client side (Java) is that many customers are satisfied with the Windows-only client." Ultimately, application adoption boils down to functionality, performance and the ability to reuse existing systems.

Where is Java Headed?

As a result of better understanding of these success factors, the market is altering its expectations as to how Java can best be used. More and more, developers are turning to the native OS to access key services in a competitive world. A recent Microsoft Developer Tracking Study showed that over 50% of Java developers use native methods. Even Java ISVs rely on native OS access for commercial development projects. Many of the top platform vendors have recognized Java's limitations and are encouraging access to native services or natively compiled executables in order to meet customer needs. As operating system technology evolves, developers who take advantage of the latest features will have an advantage.

Another recent trend is the focus on Java as a server-side technology due, in part, to the problems encountered deploying to diverse clients. With greater control over deployment targets, developers have been successful in building simple server-side applications. But building complex middle-tier logic requires access to the full range and functionality of the server's application services. It is only by taking advantage of these services that developers can create applications providing the performance, scalability, and robustness required of enterprise systems. It is interesting to note that the Enterprise JavaBeans (EJB) specification is limited in these key areas because the contributing – and fiercely competitive – vendors could not find common ground.

Turning Development Trends into Real Solutions

While the Java market has moved towards greater native access, development tools have not matched this trend. Currently, most Java tools focus on client-side applications, emphasizing cross-platform deployment over other considerations, such as performance and functionality. With these tradeoffs, applications created with today's Java environments are just not competitive against those written for the native platform using languages like Visual Basic and C++. Server-side support is also limited, as the architecture for database connectivity is immature, and there is no support for accessing the full range of platform services required by modern enterprise business systems.

Microsoft Visual J++ 6.0, Technology Preview 1, meets today's changing market needs. By combining the power of Windows and the productivity of Java, it offers the fastest way to build and deploy high-performance, data-driven client and server solutions for Windows and the Web. With Visual J++ 6.0, developers can construct powerful business applications using the Windows Foundation Classes (WFC) for Java, an object-oriented framework that encapsulates, simplifies and unifies the Win32 and Dynamic HTML programming models.

By directly accessing the Win32 API, Visual J++ 6.0 makes it possible for Java developers to produce feature-rich applications, leverage existing software investments and build powerful middle-tier business objects that integrate with a host of Windows NT application services. Moreover, through integration with the language-neutral Microsoft Component Object Model (COM), developers can build applications using "cooperating components" written in a wide variety of languages.

Visual J++ 6.0 is the first Java development tool to provide easier access to the underlying platform, and offers end-to-end solutions for developers looking to build powerful, full-featured Java applications. Other tools will soon follow, as several top Java tool vendors plan to incorporate WFC into their offerings, providing developers with even more choices and flexibility. And on other platforms, vendors will continue to expose more native functionality to Java. As the market for native Java tools grows, so too will the opportunities to build commercial quality applications, creating a wealth of new application choices for businesses and consumers.

About the Author

Bill Dunlap is the Visual J++ Technical Product Manager at Microsoft Corporation.

CORBASCRIPT

Using JavaScript with CORBA to glue your large components together

by Jeff Nelson

Scripting languages provide a powerful tool for easily gluing together components of a system. The CORBA community has recently begun work on a CORBA Component framework which incorporates many of the architectural insights of JavaBeans. This article explores how JavaScript could serve as a scripting language for such CORBA components to quickly create script applications which draw on mission critical network services.

Introduction

Scripting languages have established a role in the software industry as a powerful tool for quickly whipping together applications. The applications developed with scripting languages, called scripts, can be used as the glue language for large components. Some of the benefits of scripting languages include speed of development, ease of use, lack of strong typing and runtime customization.

CORBA is an incredibly popular, open standard for developing complex distributed systems. More than 800 companies have joined the Object Management Group, the standards body which maintains the CORBA standard.

CORBA stresses different goals from those of scripting languages. While scripts are short programs that solve simple problems, CORBA was designed to provide scalable solutions to complex problems. While scripts are not compiled, most languages used to build CORBA objects, such as Java and C++, are. Finally, while scripts rarely use strong typing, CORBA mandates that every object define its operations using a strongly typed interface.

That both tools have different goals is an indication that a synergy might exist in combining them into a scripting language that leverages the power of CORBA called, say, “CorbaScript”. Though CORBA applications are written in strongly typed languages and developed to scale to the enterprise, scripts might provide a path to quickly composing these objects to solve complex problems. Here, the objects used in the scripting language is not a simple “button” or “number”, as might be found in your typical scripting language; instead, the target of these scripts is CORBA objects solving enterprise scale problems. CorbaScript addresses the easy integration of existing CORBA systems in such an application.

Another benefit of a CORBA scripting language is the ability of entry level developers to learn and use scripts. That scripts often don’t have data types or require compilation steps makes them easy to use. Developers may start with a small set of example script commands and combine them in different ways to achieve their desired effect, slowly learning the complete set of script commands over time. This approach to learning a scripting language is common in the Visual Basic and JavaScript communities and has proven remarkably successful. CorbaScript would leverage these benefits of scripting languages, allowing entry level developers to become immediately productive in the development of CORBA software while gradually learning to perform more complex tasks over time.

Some scripting languages also have come under the spotlight of the Internet. JavaScript, in particular, has been recognized as an international standard supported in both Netscape and Internet Explorer. Since these Web browsers are already installed on virtually every computer in the world, a CORBA scripting engine based on JavaScript could come pre-installed on every machine.

With these observations in mind, Netscape has designed a CORBA scripting facility for JavaScript provided within their Component Development Kit. This facility integrates with Borland’s VisiBroker to access CORBA objects from within the JavaScript environment. VisiBroker is wrapped as part of Netscape’s Enterprise Server so that developers do not necessarily need to purchase separate licenses from Borland to begin development immediately. Furthermore, Netscape’s flagship JavaScript development suite, Visual JavaScript, also integrates well with the CORBA component model. Developers can add CORBA components to the Visual JavaScript component palette and then add these components into applications through simple drag and drop mouse operations. Entry level developers don’t necessarily even need to be aware that they are working with CORBA under the covers, just drag and drop components!

Under the covers, the standard VisiBroker tools like Web Naming are still used to coordinate the access to the CORBA servers. JavaScript has always had a fairly nice integration model with Java. This new support for CORBA objects is merely a fairly straightforward extension of this Java integration. The IDL file is compiled into Java stubs and skeletons under the covers; then the Java stubs and skeletons are integrated into JavaScript. The end result is the illusion that the IDL file magically became a JavaScript component because the developer doesn’t necessarily need to be aware of the underly-

In this issue, Jeff Nelson returns once again to continue the discussion of the upcoming CORBA Component Model. This time, Jeff’s focus is on scripting: what is the planned CORBA Scripting Language, what does it have to do with extant scripting languages (like Javascript, Perl, REXX and TCL) and how does it relate to component composition? Since several scripting languages have been experimentally linked with CORBA systems to support linking of components, this is a timely topic.

Richard Soley
Editor, CORBACORNER
President and Technical Director of the Object Management Group, Inc.

ing implementation details.

Hello, World!

Let’s face it. Most software developers, myself included, don’t understand a new technology until they see a “Hello, World!” example. For those of us who think in source code, here is such a demo.

In order to follow along with developing this yourself, the first task is to collect the software that you will need to run this example. You should obtain the following from Netscape if you don’t already have them:

- Communicator 4.03 or better
- Enterprise Server 3.0 or better
- Visual JavaScript

You also need a Java development environment. Netscape recommends JDK 1.1.2 or better for component development. As long as your Java development environment is compatible with Java 1.1.2.\, your environment should work.

Next, be sure that your CLASSPATH includes nisb.zip and wai.zip from your installation of Netscape Enterprise Server 3.0. These contain the required Java classes related to VisiBroker bundled into Enterprise Server.

We are now ready to write an IDL file. For this example, let’s just use the following simple interface that has only a single operation returning a string. Save this in a file called Hello.idl.

```
// IDL
interface Hello
{
  string getMessage();
};
```

The next step is to compile this CORBA interface. Provided your CLASSPATH is set up appropriately and you have all the right software, the command for compiling this interface is the following:

```
> java com.visigenic.vbroker.tools.idl2java
Hello.idl
```

In the above command, the “>” represents the prompt on the command line or shell that you are using. This varies quite a bit between operating systems and shells. In Microsoft operating systems, it looks something like “C:\>”.

This compilation step produces a number of Java source files, including Hello.java, HelloHelper.java, HelloHolder.java, _st_Hello.java, _sk_Hello.java and others. These are the normal files provided by a CORBA compiler to perform its task of providing the illusion of network transparency to the distributed objects. These files are not related directly to this example of integrating CORBA with JavaScript other than their normal role in a CORBA-based distributed system.

In order to proceed with building an example of access to a CORBA server from within a JavaScript application, we also need to create a CORBA server. Since the design and construction of such a CORBA server in Java isn’t the point of this article, the code is presented in Listing 1 without additional explanation.

Once you have the server ready, you can compile the example within your favorite Java development environment. Something as simple as the following could be used with the JDK.

```
%javac *.java
```

The next step is to create a directory for CORBA objects on your Enterprise Server. The Web Naming feature of VisiBroker makes use of this directory to store the object references for your CORBA servers. If your Enterprise Server is not already running on port 80, start it on that port.

> *“That scripts often don't have data types or require compilation steps makes them easy to use.”*

- Open up the System Administration URL on your Enterprise Server and confirm that “Web Publishing” is set to “On”.
- Choose the HTTPD that you would like to broadcast with CORBA objects.
- Select “Content Management” and then “Additional Document Directories”. A new URL prefix is required to host your CORBA objects. Netscape requires this to be called “iiop_objects”.
- Next, provide a directory for the contents of iiop_objects. This must be directly under your primary document directory in Enterprise Server.
- Finally, go under the “Server Preferences”, then "Restrict Access" frames, to configure the access permissions of the new “iiop_objects” directory.
- Select Edit Access Control. Confirm that access control is turned on, but configure it to “Allow anyone from anyplace all rights”. In practice, you wouldn’t want to do this for a Web server that contains mission critical documents because it could represent a security risk. For the purposes of this simple example, though, not much risk is present.
- Save and apply the changes you’ve made in the System Administrator.

Now that Enterprise Server is completely configured to run the example, let’s try to start the CORBA server and see if it runs properly. We don’t have a client yet, so it won’t do anything interesting. However, if it starts up and at least stays running, that probably is a good indication that Netscape Enterprise Server is configured correctly.

You can run the example from within your Java development environment or on the command line with the JDK. From the command line, something like the following command is required:

```
> java -DDISABLE_ORB_LOCATOR HelloImpl
"Hello, World!"
```

The -DDISABLE_ORB_LOCATOR command line argument instructs the VisiBroker ORB not to worry about using the normal ORB locator feature built into VisiBroker. This example uses Web Naming and doesn’t need the ORB locator functionality. If you are running this example from within a Java development environment, be sure to specify this command line option by configuring the execution options for HelloImpl. How this configuration is performed varies considerably between development environments, though, so you are on your own!

If HelloImpl doesn’t crash within about two minutes of starting up, everything is probably configured correctly up to this point. Leave the server running for now; we will use it later once the JavaScript is ready.

The next step is to use Visual JavaScript to make use of the CORBA components. Visual JavaScript uses a component palette paradigm for software development. The application developer just clicks and drags appropriate components into an application to make use of the component. Little or no source coding is required to create a complete application in this manner.

First, create a new project and add a new blank page to the project. This page will serve as a host for our JavaScript once it is developed. Next, create a palette called “Corba” to hold CORBA components. If you need help in doing this, see the ShowText example provided with Visual JavaScript. To import the Hello component into this new palette, select the “Corba” tab on your Components Palette. Click the right mouse button on the "Corba" tab to view a context menu. Choose "Install" and then "Corba Component". Use the File Dialog box to navigate to the location of the Hello.idl file and select this file. The component should now appear in your "Corba" palette.

Now we are ready to develop a JavaScript application with the Hello component. Click and drag the component onto the blank page of your project to add the component to the project. The CORBA component is now part of the project, although we haven't configured it to communicate with the server yet. In order to do that, double click on the Hello component in the project (not the one in the palette). You should see a prompt for "Object URL Address". The Hello component in this project needs to be connected with the HelloImpl server by specifying the URL for the CORBA server. This URL was configured in the source code of HelloImpl and should be something like "http://localhost/iiop_objects/Hello". Type this URL in as the value of the "Object URL Address" for the Hello component. Now the component in your JavaScript application can find the CORBA server.

Finally, let's create a simple form to display the results of this CORBA invocation. Select the tab labeled form elements. Click on a button and drag it onto the blank page. You should see a new form automatically created for you in the project with your button in it. Double click on the button and change the "Button Label" to "Get Message". Next, click on a "Text Field" and drag it into the form. Select the connection point of the button and drag it on top of the Hello component. A Connection Builder dialog box will appear asking you to configure how the Button and Hello component are connected to each other. Choose "View Parameters" inside the Connection Builder. Enter the action as getMessage and then press the "View JavaScript" button. In the JavaScript event handler box, change whatever is shown there to read:

```
document.Form1.Text1.value = Hello1.getMessage()
```

The JavaScript is ready to publish as a Web page. Choose "Apply", "Close", then select the Deploy button on the toolbar to load this project into your Enterprise Server.

Start up Communicator and navigate to the URL where you deployed the Web page. The form you created a second ago using Visual JavaScript should appear. Just click the "Get Message" button to invoke the CORBA object! You should see the message "Hello, World!".

Not so fancy you say? Well, remember that this is a distributed system. The JavaScript in this Web page is invoking the CORBA component, which could be located anywhere in the world. It doesn't have to be on your machine; it might be on another Web server. The CORBA component might implement the accounting system for a sales organization or provide the electronic commerce API for customers. The CORBA component in the JavaScript locates the component, even though it might be running on another machine and communicates with it transparently. This empowers you, the developer, to build complex JavaScript applications that make use of all kinds of network services.

> *"...the goal is to standardize a single language as the common glue for a CORBA Component model."*

The Details

When JavaScript accesses a CORBA object, each of the constructs in the CORBA object must map onto some equivalent constructs in the JavaScript language. Constructs such as data types, structures, object references and modules in CORBA must have corresponding features in JavaScript in order to provide the API sufficient for the JavaScript developer to manipulate CORBA objects.

CORBA has a fairly rich set of data types, including float, double, short, long, char and others. JavaScript, like many scripting languages, tries to minimize the software development overhead of data types by providing only one or a small set of data types. The main four data types in JavaScript are object, number, string and boolean. JavaScript doesn't differentiate types of numeric data types like long, short, float and double. Rather, all the numeric data types in CORBA are mapped to the JavaScript number.

Perhaps surprisingly, CORBA char and wchar data types, representing characters, also map into the JavaScript number. No character data type is present in JavaScript, the closest data types being string and number. The designers of the mapping made the choice to map the char and wchar onto the corresponding numeric values of the character. The JavaScript application may then determine which character was returned by testing the numeric value.

The CORBA boolean and string data types both map naturally into the JavaScript string. CORBA strings are normally strings of 8-bit chars, but the conversion between the CORBA 8-bit char string and the JavaScript 16-bit char string is automatic. An exception is raised if one of the characters can not be converted back and forth between the different formats of string, as might happen if a particular 16-bit character could not map into the limited set of available ASCII 8-bit characters. CORBA also recently added support for a new wstring data type, which is a string of 16-bit characters. The wstring also maps well into the JavaScript string.

CORBA also has a set of less intuitive data types present in other languages, such as enum, struct, union, array, and exception to name a few. All of these map into a JavaScript object. A particular custom JavaScript object is created for each of these data types to provide the capabilities of these data types within a JavaScript application. When a JavaScript application requires an enum to invoke a particular CORBA server that happens to use an enum as a parameter, the script must simply instantiate an object of the appropriate enum type and pass it into the invocation. In the reverse case, JavaScript objects of the enum type might come back as a return or out parameter in a CORBA invocation.

Do all of these many-to-one mappings of different data types in CORBA to a single data type in JavaScript cause problems? Not really. The CORBA stubs and skeletons contain all the data type information about the interface, including the types of all the parameters for a particular operation. The different data type can be translated back and forth easily under the covers by CORBA and JavaScript. When a developer passes a number data type as part of an invocation of a CORBA Calculator object, the stub for the Calculator defines the appropriate data type to pass into the invocation, perhaps double. JavaScript converts the number data type into the Java double data type in the process of invoking the invocation on the Calculator stub. Developers don't have to spend a lot of time worrying about doing it themselves. However, a basic awareness of the process can help eliminate tricky bugs down the road that might be the result of data type conversions going on under the covers. For example, don't try to pass the number 100,000 into a CORBA invocation that expects a short or you'll get undefined behavior.

One of the final and most important questions is how to actually access the CORBA objects from within JavaScript. That is, when a CORBA object is running someplace across the Internet – say, on the Netscape Web server – and you are writing a JavaScript that uses that CORBA object, how do you make your script communicate with the remote CORBA object? JavaScript introduced the concept of an ObjectURL, a normal Internet URL specifying the location of the instance of the CORBA object running out there on the Internet. By setting the ObjectURL of the CORBA component to the appropriate URL of the CORBA object, the component knows how to invoke the appropriate CORBA server.

Other Approaches

Is scripting of CORBA distributed systems a radical new technology? Not necessarily. Several scripting languages have had CORBA support for some time, such as TCL, Perl and Python. Interpreted object-oriented languages such as Smalltalk and Lisp also have a long history of CORBA support. Should we standardize a single "CorbaScript" language when all these other languages are already available?

Listing 1.

```
// Java

public class HelloImpl
  extends _sk_HelloCorba
{
  public String message = "The default message.";

  public HelloImpl(String n)
  {
    super(name);
  }

  public String getMessage()
  {
    return message;
  }

  public static void main(String args[])
  {
    // Use first command line argument as the message, if provid-
ed.
    if (args.length > 0)
      message = args[0];

    try
      {
        org.omg.CORBA.ORB orb = orb.omg.CORBA.ORB.init();
        org.omg.CORBA.BOA boa = orb.BOA_init();
        HelloImpl h = new HelloImpl("Hello");
        boa.obj_is_ready(h);
        Naming.register("http://localhost/iiop_objects/Hello",h);
        boa.impl_is_ready();
      }
    catch(Exception e)
      {
        System.err.println("HelloImpl: " + e);
        e.printStackTrace();
      }
  }
};
```

In my opinion, the key seems to be that the CORBA scripting language is not aimed at the simple integration of CORBA with a scripting or interpreted language. Rather, the goal is to standardize a single language as the common glue for a CORBA Component model. This approach is analogous to the use of Visual Basic as the standard scripting language for COM-based components. Such COM components can be accessed to a limited extent in other languages, such as C and C++, but Visual Basic has become the de facto standard that the majority of the COM community has rallied behind.

CorbaScript and CorbaBeans are inseparable in this sense. You can't have a CorbaScript language before you talk about the CORBA component model formed by CorbaBeans. Netscape is far ahead of the game by integrating CORBA components with JavaScript.

Where to Go From Here

The example described in this article can be downloaded from http://www.DistributedObjects.com.

Information on CORBA standards can be found at http://www.omg.org.

Netscape product information can be found at http://www.netscape.com.

About the Author

Jeff Nelson is a distributed systems architect with DiaLogos Incorporated, experts in CORBA and Java Technologies (http://dialogosweb.com) and active participants in the Object Management Group. He has 8 years of experience in distributed computing and object technology. Jeff can be found on the Web at http://www.distributedobjects.com/ or by e-mail at jnelson@distributedobjects.com.

✉ jnelson@distributedobjects.com

The Data Series: Object Serialization

Reading and writing object data made easy

by **Alan Williamson**

Data is one of those housekeeping duties that has to be performed by everyone who uses a computer. Whatever level they are on, somewhere along the line data is stored, files are created and directories are made.

Last month I introduced the whole notion of data and suggested ways in which we might handle various quantities of the stuff. One of the greatest problems faced by the developer is that of storage. Every situation seems to call for a different solution. For example, last month we looked at storing small amounts of data through the use of INI, or program information files. A complete class for this type of manipulation was developed and by the time implementation was complete we had a class that allowed the storing and retrieval of data arranged in key/value pairs.

In this column we will look at the saving and loading of complete classes through the use of the Object Serialization mechanism introduced in the 1.1 version of the JDK.

"Our object in the construction of the state is the greatest happiness of the whole, and not that of any one class."

These words were uttered by Plato around 400 B.C., and it makes you wonder just how old this supposedly new object-orientation really is. While we ponder this in the back of our minds, let's have a look at Object Serialization.

Object Serialization

In a nutshell, Object Serialization is a mechanism that allows an object to be represented as a byte stream. This byte stream can then be passed on to any of the Java streams, such as sockets and files, and then reassembled back into the original object with no data loss. Using this technique, not only can objects be safely saved and restored, but they can also be passed onto other machines in a distributed environment. In fact, Remote Method Invocation, or RMI, uses serialization in exactly this manner.

At this point you may be thinking that it sounds a little complicated. More hassle than it's really worth. Be prepared to be pleasantly surprised. Object Serialization has been made so easy to use, there's really no excuse for not taking advantage of it. Let's look at an example.

Listing 1 shows the saving of both a String and a Date object. You will notice the first thing that is created is an instance to an OutputStream. In this instance we are using a FileOutputStream, but this could have been any output stream. Next we create an ObjectOutputStream instance, which is what we will use to write the data to the output stream. Writing the actual objects out to disk is then a simple matter of calling the writeObject( ) method for each object that is to be saved.

The ObjectOutput interface is part of the java.io package. Listing 2 shows the main methods for this interface, of which ObjectOutputStream is an implementation.

As seen in Listing 2, the most important method is writeObject(...), which takes as a parameter a reference to an Object which can be transposed to a stream of bytes.

Recreating the objects is as simple as writing them in the first place. For each output-orientated method/class, the equivalent input methods exist. For example, to read back in the objects we wrote out earlier, we can use the code shown in Listing 3.

Instead of using writeObject(...), we use readObject(...), casting the objects necessary. One important point must be made, and that is the order in which objects are read and written. Generally, they have to be read in the same order they were written out in. This ensures the correct data is associated with each object.

The majority of the classes in JDK1.1 can be read and written using this technique. For example, let's say you had a Hashtable and it was populated with hundreds of String objects. Writing this out to disk can be performed using one call to writeObject(...) with the hash table as the object reference passed in. This will write out all the data associated with the hash table, including all the String objects. This is one of the greatest features of object serialization, the ability to present the developer with an interface that makes the saving and retrieval of objects a really trivial matter.

If the majority of the classes in the JDK are already for serialization, how do you go about making your own classes serializable? This is a very easy thing to do. Simply have your class implement the java.io.Serializable interface. Listing 4 illustrates this.

That's it! It's as simple as that. Once your class implements this interface, it can be read and saved as easily as the examples earlier in this article.

Summary

This article took a look at the Object Serialization method that was introduced in the 1.1 release of the JDK. As was demonstrated, reading and writing of object data is made extremely easy. But this isn't limited to just saving data in files. The same technique can be used to send complete objects to processes running on other machines via Sockets.

In next month's article we will look at JDBC and how we can start reading and writing volumes of data stored in an external database.

About the Author

Alan Williamson is on the Board of Directors at N-ARY Limited, a UK-based Java software company, specializing solely in JDBC and Java Servlets. He has recently completed his second book, focusing purely on Java Servlets, with his first book looking at using Java/JDBC/Servlets to provide an efficient database solution. He can be reached at alan@n-ary.com (http://www.n-ary.com) and welcomes all suggestions and comments.

alan@n-ary.com

Listing 1: Saving string and date object.

```
String tText = ìPlease save me!î;
Date   tWhen = new Date();

FileOutputStream     FS = new FileOutputStream("saveme.txt");
ObjectOutputStream   OS = new ObjectOutputStream(FS);
OS.writeObject( tText );
OS.writeObject( tWhen );
OS.flush();
```

Listing 2: Main methods of ObjectOutput interface.

```
public interface ObjectOutput extends DataOutput
{
  public void writeObject(Object obj) throws IOException;
  public void write(int b) throws IOException;
  public void write(byte b[]) throws IOException;
  public void write(byte b[], int off, int len) throws IOException;
  public void flush() throws IOException;
  public void close() throws IOException;
}
```

Listing 3.

```
String tText;
Date   tWhen;

FileIntputStream     FS = new FileInputStream("saveme.txt");
ObjectOutputStream   IS = new ObjectInputStream(FS);
tText                    = (String)IS.readObject();
tWhen                    = (Date)IS.readObject();
```

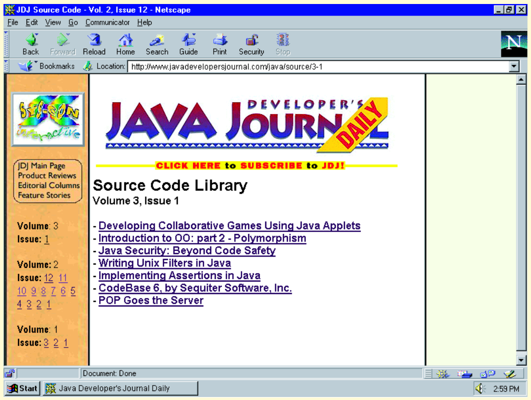
Listing 4: Implementing java.io.Serializable interface.

```
public class myClass implements java.io.Serializable
{
  String Name = ìî;
  int age = 39;

  public class myClass(){
    Name = ìCeriî;
    age  = 19;
  }
}
```

Don't Type it... Download it!

Access the source code for this and other articles appearing in this issue at JavaDevelopersJournal.com



JDJ Source Code - Vol. 2, Issue 12 - Netscape

Location: http://www.javadevelopersjournal.com/java/source/3-1

CLICK HERE to SUBSCRIBE to JDJ!

Source Code Library

Volume 3, Issue 1

- Developing Collaborative Games Using Java Applets
- Introduction to OO: part 2 - Polymorphism
- Java Security: Beyond Code Safety
- Writing Unix Filters in Java
- Implementing Assertions in Java
- CodeBase 6, by Sequiter Software, Inc.
- POP Goes the Server

JDJ Main Page
Product Reviews
Editorial Columns
Feature Stories

Volume: 3
Issue: 1

Volume: 2
Issue: 12 11 10 9 8 7 6 5 4 3 2 1

Volume: 1
Issue: 3 2 1

Document: Done

File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks
JAVA DEVELOPER'S JOURNAL DAILY
SYS-CON interactive
Start
Java Developer's Journal Daily
2:59 PM

FROM THE JDJ EDITORIAL BOARD

Java: Life Beyond the Browser

A need to leverage the strengths that Java offers

by **Arthur van Hoff**

Although Java is a topic for great discussion and debate, most would agree that it has much technical merit. The expectations and realities of Java have been visited frequently and there have been numerous speculations about where Java is headed. What has become clear over the past couple of years, is that Java is well on its way to becoming a highly successful language and an even greater marketing success. One question that this situation has raised is, "Is Java well suited for deploying real enterprise applications?"

When Sun first introduced Java in May of 1995 with great fanfare, it was promoted as a programming language that would provide a solution for more robust, interactive Web pages – something that at that time was very unfamiliar to the Internet-savvy. Programmers at that time used Java to do just that: build Web pages with more interactive media. But as Sun took an even bolder marketing approach, many felt that Java had become a heavily hyped language that was touted in Silicon Valley as the language that could potentially change the direction and weight of platform compatibility. The expectations for Java were high and it aimed to be an innovative solution for many of the problems programmers encountered with native code. But instead, over time Java, in many ways, fell short of making Web pages more meaningful and was not arguably a substantial improvement over native code.

> *"Was Java's vision really misunderstood and is its true destiny outside the browser definition?"*

But Sun's bolder approach to marketing helped Java infiltrate much of Silicon Valley and it was implemented in a myriad of ways by many industry and technology-leading companies, including Corel, CNET and even Microsoft. But perhaps the most influential use came when the Silicon Valley giant, Netscape, announced that it would Java-enable its browser, demonstrating that Java can work easily across multiple platforms. Although at first this decision was influential in building confidence for the language, Netscape, too, experienced difficulties achieving compatibility across all platforms using Java. Many large corporations considering Java as the ultimate development platform began to lose confidence in it, recognizing Java's abilities may have been spread too thin. Netscape's recent decision to remove Java from their browser will have an important effect on Java's future. It poses the questions: Was Java's vision really misunderstood and is its true destiny outside the browser definition?

Raising these questions has helped redefine where Java's true strengths lie in the marketplace and has allowed it to migrate to a space that it is well suited for: deploying both client and server-side enterprise applications. Java is successful in the server environment partly because no user interface is required. Initial response to this change of direction has been quite positive, and many companies are re-assessing the value of Java and realizing its worth within the extended enterprise.

Allowing Java to evolve into this new space brings new challenges. Bandwidth-efficient deployment of network applications, software update challenges and overall reduction of the cost and complexity of managing an efficient network-computing environment are the most common challenges. With a strong architecture in place, these problems can be solved efficiently. Technology, such as Marimba's Castanet, has been developed to provide structure in this type of environment and overcome the hurdles corporations encounter when deploying rich, network applications in the intranet, extranet and Internet. Castanet works with and complements Java by providing a simple, elegant solution for reducing the cost and complexity of managing applications. Companies such as Ericsson, Ingram Micro and Toshiba have embraced the combination of Java and Castanet, recognizing it as a viable solution for enabling self-installing applications and supporting them through the entire lifecycle of distributing, managing and updating applications.

After much debate and scrutiny, little about the actual Java language has changed. This is a testimonial to the fact that while relatively new, the language is strong and mature. In order to stay competitive in a rapidly evolving market, businesses now recognize that they need to leverage the strengths that Java offers for developing and deploying industrial strength enterprise applications.

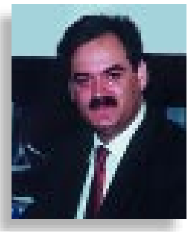
About the Author

Arthur van Hoff is one of the original members of the ***JDJ Editorial Board****. He is Chief Technology Officer of Marimba, Inc. (www.marimba.com). Marimba is dedicated to improving the experiences of users and developers on the Internet.*

Java and CORBA

Keeping the complicated simple

by **Richard Soley**



Even two years after its public debut, the Java juggernaut shows no sign of slowing. In fact, more than two years after its public debut, its popularity is still increasing. Businesses are in a headlong rush to move to Java to take advantage of the cost savings that applications running on the Java Virtual Machine have to offer. In fact, there are many who believe that soon not only will everything be programmed in Java but also that all the old non-Java programs will be scrapped and replaced with Java.

While it would be nice to believe that Java will soon rule the world and that knowing languages like C, C++ or Smalltalk will soon be like knowing Latin or ancient Greek, this isn't really what will happen. Java won't become king but will more likely be first among equals. And there will be many equals. And Java will need to be able to communicate with these equals. That communication mechanism is CORBA. Right now, CORBA ties all these languages together. And for the foreseeable future CORBA offers the best path for integrating the hot new Java programs being written with the legacy applications that sit on mainframes, minis and Unix boxes scattered all over God's green earth.

> *"...for the foreseeable future CORBA offers the best path for integrating the hot new Java programs being written with the legacy applications that sit on mainframes..."*

What Java Offers

Java offers portability – something that is in great demand in this age of increasingly networked systems. Java allows users real flexibility in their choice of hardware, software and so on. No longer are they limited by their proprietary solution. But Java still has limits of its own that need to be overcome. Because its native ORB RMI is meant for Java-to-Java communication, communication with languages other than Java, while possible, is difficult to do. That's where CORBA comes in. CORBA is a technology created to allow just this kind of inter-language communication. And now that CORBA has been mapped to Java both in IDL-to-Java mappings and Java-to-IDL mappings, this communication has been rendered much easier.

What CORBA Offers

CORBA offers interoperability – something that ties together the information and the technologies of the past and present with the rapid advances of the future. CORBA not only creates the interoperability that users demand but it gives them a rich set of services that are all part of a well-planned architecture. CORBA continues to evolve as user needs and demands change and grow. OMG's membership continues to update and expand the architecture to ensure its continued usability with the latest technologies, while maintaining its ability to keep ties with older technologies.

Where They Meet And Why It's Great

What must be remembered at the end of the day is not that one language is better than another or that any specific technology works best. What is important now is that they all work together. The computing world is immensely complicated and will only become more so as time goes on. What users want is simple – they want to get to their information as quickly and painlessly as possible – without having to agonize over how it gets done. With Java, users now get the portability they are seeking for their applications. With CORBA they get the interoperability they need to tie all their applications together.

About the Author

Dr. Richard Mark Soley is President and Technical Director of the Object Management Group, Inc. (OMG). He leads the OMG Technology Committees, which are responsible for producing standards documents, adopting OMG-standard technology and proposal of new technologies. He is a member of the ***Editorial Board of Java Developer's Journal*** *and is the editor of* ***JDJ's*** *CORBACorner column.*

Visual Quantify

by Rational Software

Identify, track and even fix performance bottlenecks in your applications

by **Ed Zebrowski**

ou've just finished up that Java program you've been working on for a while. You sit back n your chair and feel roud of yourself. It's time to show your finished work to the boss. Half an hour later he calls you into the office. "What's this?" he asks as he holds up one of the floppies that you proudly gave him just a little while ago. You try to hide the confusion and fear in your voice as you answer. "It's the application I've been working on. It's finished, and I just wanted you to have a look at it." He gestures you to his side of the desk. You watch him open the application on his machine. It opens just fine, but as soon as he begins to use it, it slows down, way down. Almost to a complete stop. "I thought it was hanging up my system," he says to you as he munches his sandwich, "but after I came back from getting my food, it started to run – a little bit, anyway." He gives you an encouraging word as he puts his hand on your shoulder to escort you to his door. "Fix it," he says as he closes the door behind you.

You just don't understand. The application runs just fine on your 300 with 128 megs of RAM. Why is he still running that ancient 166 with only 16 megs? Why should you adjust your applications for people still running machines made when the Romans invaded Carthage? You calm down a little bit and begin to think. It makes sense that a successful program will run on many different machines. Not everyone has a 300. You need to adjust the program so it can run smoothly on "lesser" processors. You need to find a way to pinpoint exactly where the application slows down.

In the old days, this was quite a challenge. You would have to decompile the code, go through it step by step and see if you could identify the problems that way. This was a tedious process that led to lots of banging on the desk and cursing. You don't have time for this. You need a quicker, easier way to do this. You need Visual Quantify!

Visual Quantify is the new application that lets you identify, track and even fix performance "bottlenecks" in your applications. It is amazingly simple to install and use.

Installation

We found Visual Quantify to be very simple to install. Installation was no more difficult than any of the current popular game software. If you're using Microsoft Developer Studio 97, VQ automatically integrates itself into that application. The next time Studio is opened, the VQ menu and toolbar are visible. This makes it possible to utilize VQ's features without leaving your development environment.

Using Visual Quantify

When VQ is first opened, it offers three options:

1. Run will pull an application through and begin instant analysis.
2. Open will pull up any files saved in VQ format (*.qfy).
3. Proceed will open VQ in idle mode, displaying a ready-to-use window.

VQ opens in a "run summary" window, which features real-time status monitoring for the threads in the program. This features a toolbar which allows the user to clear all the phases of execution that are not of interest, so only certain aspects of the program can be concentrated on. Data recording can also be controlled automatically from within the application by using an API function that embeds into the code of the program. Data recording tools allow the user to pause and resume the data recording, clear data or take a snapshot.

Visual Quantify

Rational Software Corporation
18880 Homestead Rd.
Cupertino, CA 95014
Phone: 408 863-9900
Fax: 408 863-4120
Web: http://www.rational.com
E-mail: info@rational.com

Requirements: Visual Quantify is designed to run on any system that meets the following criteria: Intel architecture; WINDOWS NT 3.51 or above, Visual C++ 2.2 or above, Java Virtual machine, Internet Explorer 4.0/Netscape Communicator 4.0 or above.

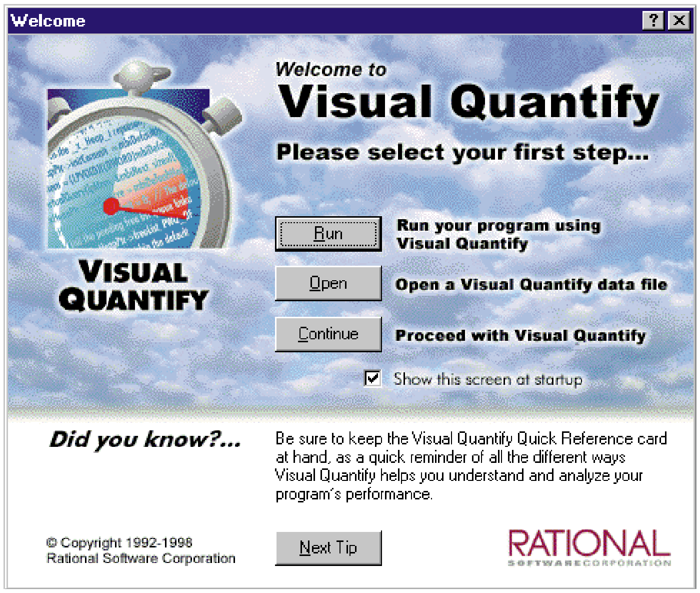
Price: $748 per 1-year subscription (includes upgrades and support for one year)
$1198.00 subscription with Purify.

Most applications feature so many paths, it boggles the mind. How is it possible to focus only on what's important and ignore what's not?

VQ features a "call graph" which maps out the application as it runs. A linear graph then displays the 20 most demanding functions of the program. Thinner lines between functions indicate quicker paths, whereas thicker lines indicate slower or more expensive paths. This allows the user to immediately pinpoint where the problems may lie.

Highlighting can isolate a particularly suspicious function. After a function has been highlighted, there are two ways of focusing on it:

- Filtering commands: This allows the user to hide functions or delete them. By hiding them, the call time to their callers is rolled up. Delete discards them completely.
- Subtree commands: Let's suppose a function is showing as particularly expensive. By right clicking on this function, it is isolated and a menu opens. Clicking on "focus on subtree" will then adjust the dataset so it will display only the troubled function and all its descendants. By clicking on "expand top 20 descendants," only the subtrees of the slow, expensive functions remain visible. Once again,



Welcome to
Visual Quantify

Please select your first step...

Run your program using Visual Quantify

Open a Visual Quantify data file

Proceed with Visual Quantify

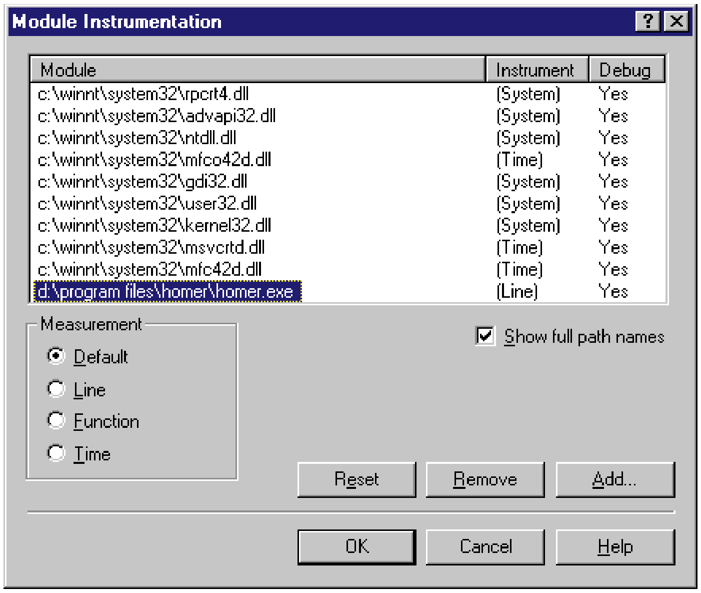
☑ Show this screen at startup

Did you know?...

Be sure to keep the Visual Quantify Quick Reference card at hand, as a quick reminder of all the different ways Visual Quantify helps you understand and analyze your program's performance.

© Copyright 1992-1998
Rational Software Corporation

Figure 1: Run, open and proceed are the three options given when the application is opened.



Module	Instrument	Debug
c:\winnt\system32\rpcrt4.dll	(System)	Yes
c:\winnt\system32\advapi32.dll	(System)	Yes
c:\winnt\system32\ntdll.dll	(System)	Yes
c:\winnt\system32\mfco42d.dll	(Time)	Yes
c:\winnt\system32\gdi32.dll	(System)	Yes
c:\winnt\system32\user32.dll	(System)	Yes
c:\winnt\system32\kernel32.dll	(System)	Yes
c:\winnt\system32\msvcrtd.dll	(Time)	Yes
c:\winnt\system32\mfc42d.dll	(Time)	Yes
d:\program files\homer\homer.exe	(Line)	Yes

Measurement
● Default
○ Line
○ Function
○ Time

☑ Show full path names

Figure 2: VQ can compare the performance of the original beside the re-engineered application.

thicker lines, allowing immediate identification of the problem area, highlight the slower paths. This feature includes a series of expand and collapse commands that work within the subtrees. Clicking the expand and collapse functions will only effect the call graph display, not change the contents of the dataset. VQ also has an undo feature, so you can go back to any previous data configuration.

Clicking on the "function list" tool changes the output from a graph to a table, which displays numerical data.

These are just a few of the problems VQ can locate:

- Unnecessary computation: VQ can point out when lists are sorted, even though the user has made no request to have the list displayed. An option would then be given to delete the sort function.
- Redundant computation: If a value has been computed twice, VQ can point out that it is more efficient to cache the results the first time.
- Redundant service requests: VQ displays the advantage of a few large operating system requests over many small ones.
- Unnecessary waiting for services to complete: It may be possible for a function to continue on, rather than wait for a particular service to complete.

It is also possible to fine tune VQ's depth and speed of data collection. When an application is analyzed, data is instrumented. A copy is made and then VQ inserts data collection instructions. When the program runs, these instructions collect data at a preset level. VQ has three levels of instrumentation:

1. Line: (Default) This setting provides the most in-depth and detailed performance data. VQ counts how often each line executes during a run, then computes data based on the number of cycles needed for one execution. This is the most demanding and time consuming setting.
2. Function: This setting provides the same accuracy as line level, but with less detail.
3. Time: VQ starts and stops a timer when each function begins and ends. This setting is accurate for the current run, but is influenced by microprocessor state and memory effects. This is the least taxing of the settings.

Not only can VQ find performance problems, it can simulate the performance of an application once the troubled function is changed. VQ will discard the unnecessary subtree from the dataset and recompute the remaining data, showing performance without the subtree. This new version can be saved as a VQ data file.

Clicking on the delta button (opens a "compare runs" function. This opens the Diff Call Graph, which allows the original application to be run side by side with the re-engineered one, comparing the efficiency of the two. New paths that show performance improvement are highlighted in green. If necessary, execution time for runs can be merged into one by using the epsilon button.

VQ doesn't stop short at analyzing only your data. It provides data for all the components in your program, including Windows DLLs and Microsoft Foundation Classes (MFC). ActiveX controls under Internet Explorer and Microsoft Office plug-ins such as Excel and Word are also covered.

When it comes to identifying and correcting performance bottlenecks on finished applications, Visual Quantify is one of the most useful and clever tools we've ever seen. That's where both its strength and its weakness lie. What if it is necessary to detect problems during the building process? VQ does not have the ability to identify and correct bottlenecks during the programming phase, only after the initial programming is complete. What about other problems beside bottlenecks? Again, VQ falls short, as it is useful only on problems of bottlenecking.

On Jan. 8, during a press conference in Las Vegas, Rational Software announced the extension of the profiling capabilities of Visual Quantify 4.0 to Visual Basic and Windows CE-based applications. This means that the same attributes of VQ are now available to those who work in these environments as well. This undoubtedly will dramatically enhance the marketability of this product as these are two of the more prominent programming tools on the market today.

Visual Quantify isn't going to be a magic cure-all to all of your programming woes. After all, it isn't going to write all of the code for you; it will just help you through some of the bottlenecking rough spots. It is, however a powerful and useful tool in the ongoing struggle to develop fast, clean, and efficient applications in today's highly competitive environment. With the use of VQ, and other tools like it, we will see a dramatic rise in the speed and efficiency of available applications. If you're using Java, C++, Visual Basic, or Windows CE, we think you might want to give it a try.

About the Author

Edward Zebrowski is a technical writer based in the Orlando, FL area. Ed runs his own Web development company, ZebraWeb, and can be reached on the net at zebra@rock-n-roll.com

zebra@rock-n-roll.com

Java's First 1000 Days

Continued commitment to an open process

by **George Paolini**

It's been about 1000 days since the Java platform was introduced. By all measures, we've come a long way since then. The magazine you're holding, ***Java Developer's Journal***, is proof of that.

Since the introduction of the Java platform, Sun's goals have been to provide platform completeness, platform ubiquity, business profitability and successful applications. These four criteria have been part of our game plan for making the Java platform successful and we're pursuing them for the long term.

From the start, we have singlemindedly focused on providing a comprehensive platform for developers, a broad environment to enable them to create rich applications. We have achieved this. And we didn't do it alone.

In the last 1,000 days, engineers at Sun, along with our partners and the Java community on the World Wide Web have been at work building the Java platform. JavaBeans™ was built by Sun with help from Borland, IBM, Lotus, Symantec and many others. The Java Foundation Classes, the graphical user interface tool for the Java platform, was built with help from Netscape, IBM, Apple and others. There are many other examples of successful APIs: Remote Method Invocation, Java Database Connectivity and Enterprise JavaBeans are just a few.

There is tremendous intellectual property invested in the Java platform not only by Sun, but also by countless other companies and independent developers. The Java platform was grown and evolved by a global community of developers on the World Wide Web. And this community insists on open standards and technical excellence. The Java platform is a product of this unique, open, industry-participative development process.

> *"From the start, we have... focused on providing a comprehensive platform for developers..."*

We remain committed to this open process.

As we move forward, we're finding that the developer community, which first asked us for functionality, is now finding the Java platform is solidifying and becoming complete. And the Java platform is providing opportunities for developers to create applications for a wide spectrum of opportunities.

In the enterprise, developers can cash in on the Java platform's value proposition by writing new middle-tier business logic using the Java platform and Enterprise JavaBeans. The Java platform lets developers use enterprise Java application programming interfaces for accessing existing middleware services and applications. These applications can then be deployed on any server, running any operating system, with any installed base of middleware. So the APIs and runtime libraries that support these interfaces allow the new applications to access existing databases, TP monitors and directory servers. The Java platform gives developers a whole new level of flexibility and interactivity.

The Java platform provides a wide range of opportunities for developers in the consumer space as well. There is enormous market potential for Java developers to build applications for devices as small as smart cards. Projections by UBS Global Research show that this year there will be 500 million smart cards on the market with microchips embedded in them; by 2000, the number of smart cards which run Java applications will skyrocket to more than 1.2 billion. The story is the same in personal and embedded devices with an explosion of potential Java devices being forecast by all the major research houses.

This explosion provides a business opportunity for developers, fueled by the Java platform's ability to provide a standard development environment that allows devices to connect to the network.

Developers who are building network-ready enterprise applications need a consistent deployment environment upon which their applications will run. We have a technology code named Project Java Activator to help ensure a consistent deployment environment. This technology lets developers take full advantage of the latest Java platform capabilities on either Internet Explorer or Netscape Navigator by allowing the browser to utilize a fully compatible, up-to-date implementation of the Java platform rather than the one bundled with the browser. IS organizations, developers and end-users no longer need to worry about multiple Java platforms on each system. One can install a single Java Virtual Machine that is utilized by all the browsers. Project Java Activator even allows new, up-to-date implementation of the Java platform on old browsers.

So now developers have a rich platform and a consistent deployment environment. Our next focus is on providing developers with the performance they need to be successful.

An recent article published in February on the Web cites independent tests which show that performance of the Java platform is as fast as C++. The study tested the Java platform with a Symantec JIT against C++ with a JIT and found that the performance gap is closing. But JITs are not the whole performance story. We're working on a technology, code named HotSpot, which is a performance turbocharger. While these tests examined just loops, not complete applications, we expect that HotSpot will provide performance equal to natively compiled C++ code and close the performance gap entirely.

The Java platform presents enormous opportunities for developers. With our commitment to equipping developers for success "by providing a rich platform, a consistent deployment environment and a renewed commitment to performance, stability, completeness and ubiquity" we're laying the foundation for a new era of computing.

About the Author

George Paolini is Director of Corporate Marketing at JavaSoft, a Sun Microsystems Inc. business unit devoted to developing Java.

Sales

Vision

JAVAONE NEWS

IBM Launches jCentral at JavaOne

(March 24, San Francisco, CA) - IBM's jCentral Web site is being launched today at JavaOne. jCentral will be the largest Java resource center on the Internet for developers to locate Java Applets, JavaBeans™, Java source code and Java newsgroup articles that meet their needs. One of the main strengths of jCentral over existing technologies is its capability to automatically crawl and "analyze" Java resources from the Internet.

One of the Most Ambitious Java Initiatives Delivered at JavaOne

(March 24, San Francisco, CA) - Here at JavaOne, IBM is delivering its second installment of application frameworks – for building Order Management and Warehouse Management applications – in its San Francisco effort. Also included will be selected international language support.

San Francisco is the most ambitious Java initiative to date for bringing line-of-business applications to the server. Over 260 companies worldwide have already licensed San Francisco and another 3,000 have downloaded its evaluation code.

Additional information on IBM's San Francisco initiative is available at www.ibm.com/java/sanfrancisco.

Microsoft Announces Visual J++ 6.0 Technology Preview

(March 24, Redmond, WA) - Microsoft Corp. today released Visual J++ 6.0 Technology Preview. Combining the power of Windows and the productivity of Java, it offers a way to build and deploy high-performance, data-driven client and server solutions for Windows and the Web. By directly accessing the Win32 API, Visual J++ 6.0 allows developers to produce feature-rich applications and build powerful middle-tier business objects that integrate with a host of Windows NT application services. For more information, a copy of Visual J++ 6.0 is included with this issue of ***JDJ***.

Zero G Announces InstallAnywhere 2.0

(March 24, San Francisco, CA) - Today, Zero G Software released InstallAnywhere 2.0, a development tool that enables applications to be installed easily by the end-user, but is also easy for the developer to use. InstallAnywhere allows the creation of a single universal installer that will operate on any Java-enabled platform. You may find a first look at InstallAnywhere in the next issue of ***JDJ***. For more information, see their Web site at www.ZeroG.com.

Lotus Ships eSuite DevPack

(March 24, San Francisco, CA) - Today Lotus announced shipment of the eSuite DevPack. The new product provides a comprehensive set of Java-based business productivity applets and technologies that enable the rapid development of highly interactive Web applications. Developers can embed the eSuite DevPack applets – spreadsheet, chart, word processor, project scheduler, presentation graphics and data access – into their Web applications.

The eSuite DevPack Preview includes sample applications, eSuite applets and documentation. For more information, see their Web site at esuite.lotus.com.

Java Boutique and Symantec Announce Applet Contest

(Orlando, FL) - The Java Boutique and Symantec have announced the Java Boutique Applet contest. Categories include educational, visual, text, audio, utility, games and overall. Winners will receive Java development software from Symantec.

The contest runs through April 15, 1998. Winning applets will be featured on the Java Boutique and special award buttons will be given out for each category. Each category winner will receive a copy of Symantec's Visual Café Professional Developers Edition. The overall winner will receive a copy of Symantec's Visual Café Database Developer's Edition. No purchase is necessary and developers may enter as many applets as they wish.

For more information about the contest, visit the Java Boutique Applet Contest Web page at javaboutique.internet.com/contest.html.

Petronio Technology Group Announces Java Programming Workshops

(Lexington, MA) - Petronio Technology Group, Inc., provider of training and consulting services, has announced two new programming workshops: "Moving from Java 1.1 to Java 1.2" and "JavaBeans." These courses are available as on-site or public-enrollment workshops.

"Moving from Java 1.1 to Java 1.2" is a one-day course that provides the Java programmer with a concise introduction to the 1.2 release. It covers the API enhancements, with special emphasis on the new JFC.

"JavaBeans" is a four-day, hands-on programming workshop. Topics include inner classes and the 1.1 Event Delegation Model, how the builder tool generates a meaningful GUI to interact with the Bean, publishing a Bean's interface and creating indexed, bound and constrained properties.

For more information, call 781 778-2000, send e-mail to info@petronio.com or see their Web site at www.petronio.com.

SofTech Announces New Approach to Data Display and Entry

(Altoona, PA) - SofTech Computer Systems, Inc. has released Java ScsGrid Control V. 2.1/JavaBean. This control offers a new approach to presenting data in a grid format with emphasis on usability and presentation of data.

The new grid component offers you the ability to dynamically resize or hide columns and include images as part of the background and within cells. Data entry is handled with popular edit components such as spinner, drop-down, toggle, button and text box controls. Another popular option is the ability to freeze columns as the user scrolls horizontally.

ScsGrid Control V.2.1 is priced at $79 for class files and $189 for source codes and comes with 90 days of technical support. For more information and promotional offers, call 814 696-3715 or see the demos at www.scscompany.com/main/grid.htm.

THE GRIND

by Joe S. Valley

The End of the World
As We Know It

I have been waking up in the middle of the night, palms sweating, heart pounding, eyes wide open. It is THAT dream again. The end is near... Armageddon for the personal computer industry.

Apple is below 5 percent market share. SGI is on the ropes. Even the 'Last Warrior', Sun, is porting to Intel's Merced. Netscape is taking on water, talking about Java as dead weight instead of a lifeboat. Cheap PCs and decent profit margins for no one, except Intel and Microsoft.

I had to make the trek once more to see Nostrajava: sage, seer, mystic and former Webmaster to the White House interns. Armed with nothing but a flask of cheap gin and a couple of Doobie Brothers 8-tracks, my journey began on a cold and rainy day. As in any journey of knowledge and enlightenment, the road is long and perilous. In my case, the road winds through the worst freeway in California. Dodging Soccer Moms in three-ton Suburbans and crazed teenagers in lowered Hondas, I worked my way up to the top of Mt. Hamilton.

At the top of the mountain, I saw the familiar beat-up Ford Van of the Keeper of the Flame, Nostrajava. Like his namesake, Nostradamus, he was in a trance, staring into a brass pot mounted on a tripod.

"Oh Great One," I called out. "Any wisdom in your brass pot; your window into the future?"

"Yo, Joey baby. What it is!" Nostrajava responded in a very un-mystic voice. "I can sense you are worried, seeing the dominance of the Intel-Microsoft machine. You fret over the future and long for the days of high margins and good gin."

"Amazing!" I responded. "How could you know?"

"Your 'Bill Gates Sucks' T-shirt for starters, but Intel and Microsoft are what everyone comes to see me about lately. Unfortunately, I can't help you, I'm out of the seer gig."

I was shocked. "But, what will you do?"

"I'm opening a microbrewery in Cupertino. Got a hell of a deal on an old building that used to house Apple Computer's Creative Services group. All kinds of cool stuff still on the wall. The place will have a 'Back to the Margin' theme"

"Look, the trend for drinking is down, and the valley is glutted with microbreweries. I thought you were a marketing wiz!" I exclaimed. "All that everyone, except me, drinks is fruit smoothies and espresso!"

"Well, Joe, you may be a trendsetter for once. Businesses already have most of the PC horsepower they need and everyone is looking to the consumer to drive the market. Next step in marketing to the consumer is a price war. Just wait until the effects of the $800 PCs hit the streets. No margins, no profit sharing, no high salaries. Trust me, people will be drinking heavily!"

Yeah, I guess he's right. The market penetration of PCs in the home is stuck at about 37 percent. The other 63 percent of the households don't need a computer or don't want to deal with the hassle of using Windows. Considering the fact that I reboot my PC two or three times a day, I understand the problem. Simply updating my browser version had my system down for a couple of days, for God's sake!

To open the market beyond the current 37 percent of households in the US, two things will have to happen. Systems will have to actually work and people will need a reason to own one. My mother will not reboot her TV, so why would she want to deal with Windows? Why do people really need a PC? Web browsing? E-mail? WebTV has it; crude but effective.

So, who is buying all these PC products right now? Current home and small business PC users. It is cheaper to buy a new system than to upgrade your old 486DX. So you buy yourself a new Pentium, and give the old system to your kids. Sales of PC products will really hit the wall as there becomes no great benefit to upgrading. A 200 Mhz Pentium is about as fast as most of us need until we get T1-speed lines into our homes. OK, OK, so there are the fringe-element Ultima players, but that's a small market.

Microsoft and Intel will do well. Microsoft is moving in on Web Commerce and corporate intranet applications, both of which continue to grow. And Windows CE will kill all other competitors in the consumer area.

Intel will move up to the high-margin server business through Sun and HP, as those companies wind down their RISC chip investment. Corporations overseas are still buying desktop systems and they buy from the big guys such as HP, Compaq and IBM. Intel gets those processors.

"So, Nostrajava, what should a humble hacker like me do in the face of such misfortune?" I asked.

"Keep that resume updated. I would hire you on as a bartender in my microbrewery, but I'm afraid you would drink up my margin." Noting the dejection on my face, he added, "One last bit of prophecy: think WebTV for the masses."

Driving back down the mountain, I saw a beautiful sunset. Overall, the Valley will do OK, the biomed, networking and specialty chip businesses are good. Pet Rocks, AIDS drugs and IP Routers will always be needed. But the ground will quake as the personal computer industry hits the wall this year.

Biomedical industry... I need to look into that. I wonder if sequencing DNA is like programming in Java?

"You fret over the future, and long for the days of high margins and good gin."

Joe S. Valley is a scarred veteran of the Silicon Valley wars. It was either writing this column or heading back into therapy. His company can't afford mental health care coverage anymore, so writing is the only option. There are a million stories in the Valley and Joe knows lots of them. Got a good story? E-mail him at Joe@sys-con.com

Joe@sys-con.com