

Please visit  
the Web sites  
of our  
advertising  
partners  
who make it  
possible for us  
to bring you this  
Digital Edition  
(PDF) of *JDJ*

## ADVERTISER INDEX

ADVERTISER	URL	PH	PG
4TH PASS	WWW.4THPASS.COM	877-484-7277	63
AJILE SYSTEMS	WWW.AJILE.COM	408-557-0829	100
ALLAIRE CORPORATION	WWW.ALLAIRE.COM/DOWNLOAD	888-939-2545	21
AMAZON.COM	WWW.AMAZON.COM/JAVA		37
APPEAL VIRTUAL MACHINES	WWW.JROCKIT.COM	468-402-2873	61
BEA	WWW.BEA.COM	800-817-4BEA	9
BULL	WWW.BULL.COM		17
CAPE CLEAR	WWW.CAPECLEAR.COM	353-1-241-9900	25
CAREER OPPORTUNITY ADVERTISERS		800-846-7591	102,118-133
CEREBELLUM SOFTWARE	WWW.CEREBELLUMSOFTWARE.COM	888-862-9898	35
CLOUDSCAPE	WWW.CLOUDSCAPE.COM	510-239-1900	12-13
COMPUTERWORK.COM	WWW.COMPUTERWORK.COM	800-691-8413	99
COMPUWARE	WWW.COMPUWARE.COM/NUMEGA	800-4-NUMEGA	19
ELIXIR TECHNOLOGY	WWW.ELIXIRTECH.COM/ELIXIRIDE	65 532-4300	57
FIORANO	WWW.FIORANO.COM	800-663-3621	65
FLASHLINE.COM, INC.	WWW.FLASHLINE.COM	800-259-1961	49
GEMSTONE	WWW.GEMSTONE.COM/WELCOME	503-533-3000	11
HOTDISPATCH.COM	WWW.HOTDISPATCH.COM	650-234-9752	27
IAM CONSULTING	WWW.IAMX.COM	212-580-2700	73
INETSOFT TECHNOLOGY CORP	WWW.INETSOFTCORP.COM	732-235-0137	55,98
INTUITIVE SYSTEMS, INC	WWW.OPTIMIZEIT.COM	408-245-8540	47
IONA TECHNOLOGIES	WWW.IONA.COM	800-672-4948	79
JAVACON2000	WWW.JAVACON2000.COM		84-85
JAVACON2000	WWW.JAVACON2000.COM		103-117
JDJ READERS' CHOICE AWARDS	WWW.SYS-CON.COM		86
JDJ STORE.COM	WWW.JDJSTORE.COM	888-303-JAVA	01
JV SEARCH	WWW.JVSEARCH.COM		20
KL GROUP INC	WWW.KLGROUP.COM/POWER	888-361-3264	15
KL GROUP INC	WWW.KLGROUP.COM/COURSE	888-361-3264	81
KL GROUP INC	WWW.KLGROUP.COM/GREAT	888-361-3264	136
MACROMEDIA	WWW.MACROMEDIA.COM/ULTRADEV	415-252-2000	32-33
METAMATA, INC.	WWW.METAMATA.COM	510-796-0915	77
NETDIVE.COM	WWW.NETDIVE.COM	415-981-4546	95
NEW ATLANTA	WWW.SERVLETEXEC.COM	800-GO-UNIFY	53
NO MAGIC	WWW.MAGICDRAW.COM	303-914-8074	7
NORTHWOODS SOFTWARE CORPORATION	WWW.NWOODS.COM	800-226-4662	70
OBJECTWAVE	WWW.OBJECTWAVE.COM	321-269-0111	70
OOP.COM	WWW.OOP.COM	877-667-6070	68-69
PARASOFT	WWW.PARASOFT.COM	888-305-0041	83
PERSISTENCE	WWW.PERSISTENCE.COM	650-372-3600	29
PRAMATI	WWW.PRAMATI.COM	877-667-PRAMATI	71
PROGRESS SOFTWARE	WWW.SONICMQ.COM/AD11.HTM	800-989-3773	2
PROTOVIEW	WWW.PROTOVIEW.COM	800-231-8588	3,135
QUEST SOFTWARE	WWW.QUEST.COM	949-754-8000	39
QUICKSTREAM SOFTWARE	WWW.QUICKSTREAM.COM	888-769-9898	96
SEGUE SOFTWARE	WWW.SEGUE.COM	800-287-1329	4
SIC CORPORATION	WWW.SIC21.COM	822.227.398801	75
SLANGSOFT	WWW.SLANGSOFT.COM	972-2-648-2424	43
SOFTWARE AG	WWW.SOFTWAREAG.COM/BOLERO	925-472-4900	67
STARBASE	WWW.STARBASE.COM	888-STAR700	93
SYBASE	WWW.SYBASE.COM/PRODUCTS/SERVER	800-8-SYBASE	45
SYNTION AG	WWW.SYNTION.COM		97
THINWEB TECHNOLOGIES	WWW.THINWEB.COM	877-THINWEB	51
TIDESTONE TECHNOLOGIES	WWW.TIDESTONE.COM	800-884-8665	23
TOGETHERSOFT CORPORATION	WWW.TOGETHERSOFT.COM	919-833-5550	6
UNIFY CORPORATION	WWW.EWAVECOMMERCE.COM	800-GO UNIFY	59
VISICOMP	WWW.VISICOMP.COM/JDJ7	831-335-1820	31
VISUALIZE	WWW.VISUALIZEINC.COM	602-861-0999	20
WEBVISION	WWW.WEBVISION.COM	800-531-5057	91
WIRELESS DEVELOPER	WWW.WIRELESSDEVELOPER.COM	800-594-5102	100
YOUCENTRIC	WWW.YOUCENTRIC.COM/NOBRAINER	888-462-6703	89
ZUCOTTO	WWW.ZUCOTTO.COM	613-789-0090	41

# JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

September 2000 Volume: 5 Issue: 9

JAVA DEVELOPERS JOURNAL.COM

**XML DevCon FALL 2000**

November 12-15, 2000

Announcing...

December 3-5, 2000

**Wireless DevCon**

From the Editor

**Source Notes**

by Sean Rhody pg. 5

Guest Editorial

**Enterprise Development**

by Joe Menard pg. 7

Industry Watch

**What's It All About?**

by Alan Williamson pg. 24

Product Reviews

**Ensemble Streams 3.2**

by Ensemble Systems pg. 84

**JSmartGrid 1.0**

by Eliad Technologies pg. 134

Interview

**Paul Chambers**

of GemStone Systems pg. 88

VisualAge Repository

**Developing Web Applications**

by Anita Huang & Tim deBoer pg. 92

**SYS.CON MEDIA**

## JAVA UNPLUGGED

### TRANSLATING JAVA TO THE WIRELESS PARADIGM

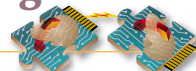
by Kristian Cibulskis  
page 62

**Feature: CORBA Components**  
*Taking EJBs to the next level*



Jon Siegel  
8

**EJB Home: Transactions and Exception Handling**  
*Create more fault-tolerant code in your new EJB applications*



Jason Westra  
16

**Feature: Introducing JavaSpaces**  
*A new and futuristic paradigm far from traditional invocation of methods on remote objects*



Sanjay Mahapatra

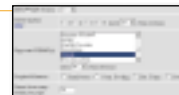
THIS IS UNENCODED DATA  
ENCODER  
28

**Java Techniques: Encoded Streams**  
*Read and write encoded data with Java I/O streams*



Mike Jasnowski  
36

**Feature: Implementing Fowler's Analysis Validator Pattern in Java**  
*Reusable self-validation GUI components increase developer productivity*



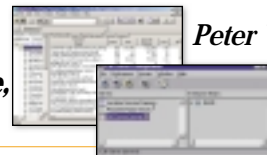
Dr. Sara Stoeklin & Dr. Clement Allen  
44

**JDBC Basics: Programming with Databases**  
*Using Java JDBC - a must-have tool for your programming toolbox*

Robert J. Brunner

56

**App Building: Challenges of Developing Distributed Java Applications**  
*Build reliable, high-performance applications and components*



Peter Varhol

72

**Feature: Understanding Workflow**  
*An introduction to workflow and workflow management systems*



Bobby Woolf

96

SEAN RHODY, EDITOR-IN-CHIEF



## EDITORIAL ADVISORY BOARD

TED COOMBS, BILL DUNLAP, DAVID GEE, MICHEL GERIN,  
ARTHUR VAN HOFF, JOHN OLSON, GEORGE PAOLINI, KIM POLESE,  
SEAN RHODY, RICK ROSS, AJIT SAGAR, RICHARD SOLEY, ALAN WILLIAMSON

EDITOR-IN-CHIEF: SEAN RHODY  
EXECUTIVE EDITOR: M'LOU PINKHAM  
ART DIRECTOR: ALEX BOTERO  
MANAGING EDITOR: CHERYL VAN SISE  
ASSOCIATE EDITOR: NANCY VALENTINE  
EDITORIAL ASSISTANT: JAMIE MATUSOW  
EDITORIAL CONSULTANT: SCOTT DAVISON  
TECHNICAL EDITOR: BAHADIR KARUV  
PRODUCT REVIEW EDITOR: ED ZEBROWSKI  
INDUSTRY NEWS EDITOR: ALAN WILLIAMSON  
E-COMMERCE EDITOR: AJIT SAGAR

## WRITERS IN THIS ISSUE

CLEMENT ALLEN, GREG BOLLELLA, ROBERT BRUNNER,  
KRISTIAN CIBULSKIS, TIM DEBOER, BRUNO Y. DECAUDIN, PETER HAGGAR,  
ANITA HUANG, MIKE JASNOWSKI, SANJAY MAHAPATRA, JOE MENARD,  
JIM MILBERY, SEAN RHODY, EKKEHARD ROHWEDDER,  
SHERRY SHAWOR, JON SIEGEL, SARA STOECKLIN, PETER VARHOL,  
JASON WESTRA, ALAN WILLIAMSON, BOBBY WOOLF

## SUBSCRIPTIONS

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,  
PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT

SUBSCRIPTION HOTLINE: 800 513-7111

COVER PRICE: \$4.99/ISSUE

DOMESTIC: \$49/YR. (12 ISSUES) CANADA/MEXICO: \$69/YR.  
OVERSEAS: BASIC SUBSCRIPTION PRICE PLUS AIRMAIL POSTAGE  
(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: \$12 EACH

PRESIDENT AND CEO: FUJAT A. KIRCAALI  
VICE PRESIDENT, PRODUCTION: JIM MORGAN  
VICE PRESIDENT, MARKETING: CARMEN GONZALEZ  
GROUP PUBLISHER: LISE ST. AMANT  
COMPTROLLER: BRUCE MILLER  
ADVERTISING ACCOUNT MANAGERS: ROBYN FORMA  
MEGAN RING  
AMANDA MOSKOWITZ  
IDISTORE.COM: AMANDA MOSKOWITZ  
ADVERTISING ASSISTANT: CHRISTINE RUSSELL  
ADVERTISING INTERN: ALISON NOVICK  
GRAPHIC DESIGNERS: ABRAHAM ADDO  
CATHRYN BURAK  
AARATHI VENKATARAMAN  
LOUIS F. CUFFARI  
WEBMASTER: ROBERT DIAMOND  
WEB SERVICES CONSULTANT: BRUNO Y. DECAUDIN  
WEB DESIGNERS: STEPHEN KILMURRAY  
GINA ALAYAN  
SYS-CON EVENTS MANAGER: ANTHONY D. SPITZER  
CUSTOMER SERVICE: ELLEN MOSKOWITZ

## EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.  
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645  
TELEPHONE: 201 802-3000 FAX: 201 782-9600  
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., 135 Chestnut Ridge Road, Montvale, NJ 07645.  
Periodicals Postage rates are paid at  
Montvale, NJ 07645 and additional mailing offices.  
POSTMASTER: Send address changes to:  
JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc.,  
135 Chestnut Ridge Road, Montvale, NJ 07645.

## © COPYRIGHT

Copyright © 2000 by SYS-CON Publications, Inc. All rights reserved.  
No part of this publication may be reproduced or transmitted in any form or by any  
means, electronic or mechanical, including photocopy or any information storage and  
retrieval system, without written permission. For promotional reprints, contact reprint  
coordinator, SYS-CON Publications, Inc., reserves the right to revise, republish and  
authorize its readers to use the articles submitted for publication.

WORLDWIDE DISTRIBUTION BY  
CURTIS CIRCULATION COMPANY

730 RIVER ROAD, NEW MILFORD NJ 07446-3048 PHONE: 201 634-7400

Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc.,  
in the United States and other countries. SYS-CON Publications, Inc., is independent of Sun  
Microsystems, Inc. All brand and product names used on these pages are trade names,  
service marks or trademarks of their respective companies.



## Source Notes

Years ago, when I was in college, I decided to pursue a minor in music to offset the insanity of getting a degree in physics. I spent a bit of time learning the key signatures, and how to transpose music written in one key to another, usually simpler, key (since I'm not much of a musician). I'm not sure how many sharps are in the key of C sharp at this point, but I think it had to be one of the more difficult keys to read, if not play.

I've received a good deal of mail and other input regarding Microsoft's new C Sharp specification, including some very interesting comparisons from one reader who outlined the similarities between C Sharp and Java in detail, including keyword counts (with synonyms even). He made his point pretty easily: it's not hard to see that C Sharp looks a great deal like Java.

Now, Microsoft is proclaiming this as the next step in the evolution of the C language. C++ added object-oriented programming to C, although it's still possible to write completely ordinary procedural code with C++. C++ also brought with it the ugly concept of multiple inheritance. Granted there are times when the ability to treat an object as more than one class, depending on need, is important. It's just that the C++ approach is one of the ugliest ways to do it I've ever seen. Add to that the memory management jungle that exists within C/C++ and it's easy to understand why Java appeals to programmers who want to be productive.

C Sharp promises to provide a new path for C. As if we needed another one. It's simple to see that C Sharp is aimed squarely at the Java community. This new language has the same syntax as Java, which is understandable since Java evolved largely out of C/C++. It also removes the concept of multiple inheritance, same as Java. And it does away with memory management and provides garbage collection as well, all similar to Java.

And we all can see where this is going. After years of trying unsuccessfully to co-opt the Java standard due to its overwhelming need to control and dominate, Microsoft has finally acknowledged that it can't break this market. So the egomaniacs in Redmond have decided to create a new language that will look like Java, but not have any ties to the standard.

Like most people, I use Microsoft products every day. I'm writing this in Microsoft Word on a computer that runs Windows NT. I use Internet Explorer. The desktop is still important. I won't rehash the tired arguments about Microsoft domination; I know we all know them. Despite the importance of the desktop, there are clearly other arenas that have evolved around or even in spite of Windows. There's Linux, Internet Appliances, Palm Pilots, Wireless PDAs – the list grows daily. Personal computing has grown to be more than just a PC. And the language that's driving that revolution is Java.

So the entrance of this new language doesn't bother me the way it might have several years ago. Java has a clear edge in the marketplace as the language of the Internet. It runs Web sites everywhere, powering B2C, B2B and other e-commerce applications throughout the world. JSP has truly helped bring Java to the browser, and EJB has standardized the way we write distributed applications.

C Sharp won't change that. About all that it will do is further muddy the story on Java from Microsoft. Fortunately, there are numerous options for Java VMs from other sources. While it would be interesting to see a strong VM from Microsoft, in point of fact it's largely irrelevant. C Sharp signals Microsoft's exit from the Java world, on a sour note. But it means business as usual for those who use Java on Microsoft products. ☪

sean@sys-con.com

AUTHOR BIO

Sean Rhody is editor-in-chief of Java Developer's Journal.  
He is also a respected industry expert and a consultant with a leading Internet service company.



# Enterprise Development — Beyond the Java IDE

Are you part of a small team that's building e-business systems in one- to three-month cycles, creating and reusing enterprise business components while integrating disparate platforms?

Are you wondering if the long-heralded era of component-based application assembly is going to arrive within your lifetime? Are you waiting for the day when you can select from a wide array of prefabricated, generalized, "certified compatible" e-biz and e-com components?

Does your vision of the perfect enterprise development environment extend beyond your current Java IDE? Are you tired of Java IDEs that don't support important Java standards, UML or object/relational mapping, or don't integrate well with other Web tools?

Does your IT organization support a mixed environment of application servers from iPlanet, IBM, BEA and others while eschewing standardization on any one of them in the near future?

If you answered "yes" to any of the above, you reflect today's professional Java developer, based on our latest round of research with over 100 enterprise Java developers.

Some of the main points from our research:

- Tools integration remains the main sticking point for Java developers. They're either poorly integrated or the tool suites lack best-of-breed point solutions.
- Developers want IDEs that stay in tight step with Sun's J2EE technical standard, including JSP support.
- Enterprise shops want complementary integration between XML and Java.
- Everyone's building reusable object- or component-based frameworks. Many say that's the "strategic direction" their application architectures need to take to go forward.
- There's a growing trend to capture business requirements in UML models early in the development life cycle. Developers want richer, easier and better integrated UML modeling facilities.
- Most shops are targeting multiple application servers, databases and operating systems – from the S/390 on down to Windows NT/2000.
- Most shops are targeting multiple SQL databases. Developers want richer and easier object-relational component-based mapping tools.

Given this feedback, it's clear that the challenges faced by enterprise developers today overtax the capabilities of even the best Java IDEs. What's needed is a new approach to enterprise development, something beyond the traditional Java IDE that integrates best-of-breed life-cycle tools, Web technologies and Java application servers – a comprehensive, integrated system that spans UI development to database persistence.

Over the past six months WebGain has acquired, licensed and integrated key development technologies as part of our flagship product, WebGain Studio. Developers should spend their time building applications, not evaluating, integrating and testing tools. A solution is needed that integrates best-of-breed technologies and works seamlessly with existing Web standards, databases, OS platforms and Java application servers, and that includes integrated modeling.

To develop EJBs efficiently, fast and easy UML modeling is evolving as a requirement. The integration between the Java and UML environments is crucial to ensure that the code and models are always synchronized.

The solution should also include an integrated object-relational mapping technology that eliminates tedious manual coding associated with the mapping of objects to relational databases, and an integrated HTML and JSP environment to streamline the development of attractive, customized interfaces to end users.

According to our research, these solutions are what Java developers want. What enterprises need goes beyond that to include simple application assembly so the power of Java can be used by non-Java programmers and business analysts. This addresses both time-to-market considerations and a shortage of skilled Java resources. New application assembly technology recently acquired by WebGain will soon make that possibility a reality.

Given all this, what choices does a Java developer have to create that application in a month? They go well beyond the IDE...to integrated suites that allow easier development from the browser to the database, that allow application modeling and that ultimately allow simple and easy application assembly on multiple application servers. At WebGain we're close to being there. ☺

[joseph.menard@webgain.com](mailto:joseph.menard@webgain.com)

## AUTHOR BIO

Joe Menard, CEO of WebGain Inc., has 20 years' experience in general management, marketing, sales and product development. He holds a BS in electrical engineering from Worcester Polytechnic Institute and an MBA from Babson College.

# CORBA COMPONENTS:

WRITTEN BY JON SIEGEL


**I**ncreasingly, business applications are evolving into a client side that interacts with the user, and a server side that stores and retrieves data and manipulates it in various ways. The client side may run on a number of different hardware types including telephones, pagers and handhelds, in addition to the usual assortment of desktop and laptop computers. The intricacies of dealing with this assortment of client types from a single server would make a good article but will have to wait for another issue, because our subject today is the server side.

Responsible for satisfying requests from all of these client types, the server side is the key to success in e-business. But, unlike even the relative simplicity of years past, today's server side must integrate many functions and has grown increasingly complex as a result. Customers expect even a simple e-commerce sales application to check stock levels and tell them – truthfully! – whether or not their item is available today. Then they expect the application to interact with bookkeeping to charge their credit card and with shipping to set up their order. After they receive an e-mail confirmation with their order number, they'll expect to punch it into an Order Status Web page and find out if their stuff has shipped yet, and then to retrieve a tracking number that lets them follow up with the delivery service as their package wends its way from a warehouse to their house.

This article is based, in part, on Chapter 5 of the book *CORBA 3 Fundamentals and Programming*, by Jon Siegel (John Wiley & Sons). Published by permission of OMG and the author. Check out Chapter 5 for more on the CCM, and a miniprogramming example contributed by the authors of the CCM specification.



# Taking EJBs to the Next Level



Clearly, this assortment of functionality is too complex to run on a single server. These days it takes a number of servers, each typically load-balanced on several to many computers, to provide the service level required by even a medium-sized enterprise. This makes the server problem inherently distributed. In addition, a server must have these four essential characteristics for businesses to rely on it:

- **Scalability:** A server must perform robustly and deliver the same good performance even when (or, perhaps, *especially* when) heavily loaded – financial trading applications need to handle the days when the markets swing way up or down and everyone wants to get in or out of positions; travel reservations systems have to handle the load when bad weather over half the country cancels flights and everyone needs new reservations.
- **Transactionality:** Business transactions, once completed, must be reliably committed to persistent storage and immediately become visible to every client. A trader must be allowed to sell his stock in XYZ company only once, and the last seat on the last flight to San Francisco must be assigned to only one traveler.
- **Reliability:** When the client side crashes, only one person notices. When your server crashes, every user notices. If your site is big enough, the outage becomes a headline in *The Wall Street Journal* and even more people notice. The ideal server never stops running – not for crashes, nor for upgrades of hardware, operating systems or even applications.
- **Security:** It doesn't matter whether you're buying books, bonds or seats – security is essential. We don't have to say why; you already know.

The first product tailored to this type of application was the transaction monitor, or TM. Based on transaction processing systems, TMs added scalability and robustness to business computing. But they were more of a genre than standards-based software, and each vendor's product used its own proprietary interfaces to provide essentially the same service as the other vendors'.

As client load increased the need for servers with these characteristics, advances in software architecture provided better ways to build them. Server-side infrastructure products combined resource control, load balancing and redundancy techniques in various ways, many using object-oriented techniques, until finally a number of standardized architectures emerged. We'll start with one you may already be familiar with – Enterprise JavaBeans (EJBs) – and see how the OMG has extended it into the CORBA Component Model (CCM). EJB is a good place to start because CCM is a superset of EJB, extended in two directions: programming language coverage and features, as we'll see.

## Architectural Basis

Before components, it took a lot of skill and technical know-how to code a server application with the characteristics we listed – scalability, transactionality, reliability and security. For example, even if you had a TM-based infrastructure, you had to begin and end each transaction in hard code. This forced programmers to acquire a set of server-programmer skills above and beyond the knowledge they needed to code the business logic that was, after all, the reason the server was being built.

In the new server component architectures, in contrast, scalability, transactionality, reliability and security have become runtime characteristics of the system rather than the coded-in properties of any individual part. They're built into the component infrastructure, which provides them to every component instance as a service at runtime, through standardized interfaces. You buy a component system with development and runtime parts. Your system administrator installs the runtime, which includes a transaction-processing system, security, load balancing and possibly provisions for fault-tolerant operation. Components running in this environment automatically become scalable, transactional, reliable and secure.

After they've been built using the development environment, assembled into an application and configured properly (more on this coming up), components are installed in, and run in, a container whose standardized interfaces communicate between the environment and the component implementation. The container manages component instances' life cycle (creation and destruction) and resource control (activation and passivation), and provides infrastructure services including state persistence and transactionality via standardized interfaces.

In contrast to client-component interactions, which may be (and, virtually always will be) remote, component-container interactions must be local – that is, the container and component instance must reside on the same physical machine. The EJB 1.1 specification doesn't say much about load balancing, but we know that multimachine EJB servers are available from several vendors; the CCM specifically allows containers to span machines for load-balancing purposes.

Let's look at how EJB (and, looking ahead, the CCM) provides the four capabilities we listed:

- **Scalability:** This is provided mainly by server resource control. EJB instances don't run constantly; they may be passivated by their container and their memory resources reclaimed when idle, and activated automatically when invoked. (CCM uses the POA for this.) Additional scalability is provided by load-balancing techniques, although these aren't standardized by EJB. (Load balancing in CCM is supported by IOP features that we won't detail here.)
- **Transactionality:** Both EJB and CCM runtimes include a transaction-processing system that provides two-phase commit and rollback behavior and all the other features of TP. You don't have to code transactionality into your EJBs or CCMs – you can get the behavior just by specifying it in your deployment descriptor files (container-managed transaction demarcation). Alternatively, you can code the beginning and ending of transactions in your components if you wish (bean- or component-managed transaction demarcation).
- **Reliability:** Although the EJB specification doesn't contain any explicit provisions for reliability, EJBs' encapsulation lets vendors build redundancy and other fault-tolerant features into their runtime. (CORBA has a separate fault-tolerance specification that can be applied to a CCM environment.)
- **Security:** The EJB 1.1 specification defines security functionality and a number of security interfaces. If you don't want to code security into your application, you can omit the calls and let the application assembler and deployer set security policies in configuration files. All security functionality resides in the container, of course. (CCM uses the same structure with CORBA security interfaces.)

### Building an EJB Server and Application

Both the EJB and CCM specifications divide the creation and deployment of a server application into roles. We'll present the six roles described in the EJB specification; differences between these and the corresponding CCM roles are subtle enough that we don't have to discuss them separately. The first two roles are extremely technical, since these players create the development and runtime environments. These designers and programmers program transactionality, scalability, reliability and security into a product that can be used by any business to create applications that gain these desirable attributes from the system that runs them. They are:

1. **Server provider:** This player provides a development and runtime environment that is transactional, reliable, scalable and secure. It need not conform to any of the EJB interfaces, however – in fact, many TM suppliers could fill this role.
2. **Container provider:** This player builds the container. On the outside, the container interacts with the server using its proprietary interfaces. On the inside, the container provides the standardized EJB functionality to the enterprise beans installed within it through the interfaces defined in the specification. This player must also provide configuration, installation and runtime support.

Freed by the container architecture from the need to program technical details, domain business experts can now step in and play roles 3

and 4. The application they produce will then combine their best business logic with the enterprise qualities that the server and container providers built into the base system. These business roles are:

3. **Enterprise bean (or component) provider:** This person is typically an application domain expert. Because the container provides scalability, transactionality, reliability and security, and the architecture allows for transparent distribution, he or she doesn't have to be an expert in these (or any other) system-level programming techniques. Still, this person should understand reusability enough to produce EJBs that can be used in a range of applications.
4. **Application assembler:** This person assembles the coded enterprise beans into an application. Sometimes the provider and the assembler will be the same person, especially for applications written from scratch with no (or little) reuse of previously developed beans. In projects that reuse beans, the assembler's role is to put together an application that combines EJBs from multiple sources.

The architecture carries the division-of-labor principle one step beyond what we've described so far. By dividing application creation into two steps – bean provider and application assembler – it puts into place the foundation of a component marketplace where multiple supplier companies develop independent components that can later be assembled into multiple applications to fit most precisely the needs of the end user.

5. **Application deployer:** This person takes one or more `ejb-jar` files and deploys and configures them so that all references are resolved.
6. **System administrator:** This is the person who makes sure that everything runs, and continues to run, as the users take advantage of everything players one through five have created.

## Some Technical Details

Four categories of EJBs allow developers to fine-tune resource management; these carry through without much change into the CCM, as we'll soon see. There are two types of session beans, which don't have identity and can be used by only one client: stateless and stateful. The former can only be called once. They do their thing on that one call, using only data that you've supplied as input arguments, and may be considered destroyed after the call completes. Stateful session beans, on the other hand, are conversational; they maintain their state from one call to the next, but typically don't store any of it persistently in the database.

There are also two types of entity beans, which have identity and can be registered in JNDI, or whatever naming service you have, and used by multiple clients. They typically represent entities in your database, adding whatever functionality you program into them. EJB divides this category into beans with container-managed and self-managed persistence. CCM divides this category into Process and Entity component types, differing in the way instances are identified; both of these types let you choose between container-managed and self-managed persistence.

EJB Homes add classlike functionality to the environment. Each type has a home, which provides at least its create operation; for entity beans the home also provides a find operation using the identity and a table of instances that the runtime maintains. This concept carries over into CCM.

Distribution is provided by RMI and, increasingly (since Sun has just made this a requirement in the new EJB 2.0 draft specification), by RMI-IIOP. Since the bean-container relationship is a local one, only client-bean (and client-home) interaction goes over the network.

## And Now, to CORBA

OMG members, working closely with Sun's Java people, wrote the CCM specification to work closely with EJBs. How closely? The specification defines two levels of components. *Basic-level* CCM components have the same characteristics as revision 1.1 EJBs, except that their inter-

faces are defined in OMG IDL and the components themselves may therefore be written in any mapped programming language, not just Java. That's why we discussed EJBs in such detail. *Extended-level* components have additional features. We like these features a lot, and think that the extended CCM environment will be especially productive.

Before we go off into the features of extended CCMs, however, let's take a closer look at the architectural advantages we get from the tying of basic CCMs to EJBs. Remember that assembly step – the one where we built an application from a bunch of different components? Typically, a final server-side application will consist of multiple component types that call each other to get the entire job done. Each provides a subset of the application's functionality. In spite of their differences, the commonality of CCMs and EJBs lets them also call each other. (This will require a bridge, but only an extremely thin and efficient one due to the commonality of the models.) The bottom line is, when we assemble an application, we can use all of the components in our library – EJBs and basic CCMs together. Configuration files tell our system how to direct calls from one environment to another.

This architectural convergence adds CORBA's multilanguage capability to EJBs, allowing us to build components in whatever languages we need (C++ comes to mind right away) in addition to Java, and assemble them into an application along with our EJBs. EJB programmers can contribute to the libraries of a multilanguage environment without having to learn anything new, while CORBA programmers build components that can be used with the substantial libraries of EJBs, written by programmers who never learned to work in CORBA.

### CCM Foundations

CCM's interfaces are defined in OMG IDL, extended by the CCM specification to include new keywords including (no surprise here) *component*. IDL separates interface from implementation for CCMs just as it does for CORBA objects, transparently extending the CCM to the multilanguage CORBA environment. One caveat we hear from people who like the technical aspects of EJBs is, "But it's a single-language environment!" Whether you want to use other languages available today or are worried about the language that will come along sometime in the future and replace Java, the compatibility of EJBs and CCMs should set your mind at ease – on this score, anyway.

The CCM container is a specialized CORBA POA (portable object adapter). Designed to support scalable servers, the POA architecture allows an application to set policies that control (among other things) activation/passivation patterns for the executing code and data that constitute a CCM instance, which, by the way, is called a *servant*. In terms of resource control, a POA is more flexible than an EJB container, able to support several hundred patterns of instance activation/passivation defined by combinations of seven policy types.

This is our first sighting of a pattern that carries through all of the CCM. In much of CORBA there is a tremendous amount of flexibility, which requires a fair amount of technical knowledge to program. In the CCM, however, the pattern is to wrap these CORBA features in a layer that exposes a simpler, higher-level interface with fewer choices. This makes it easy for business-level developers to pick the right one, and lets the service provide the behavior using generated code. In many parts of CCM the coding required for a service is reduced to zero for developers who are willing to accept default behavior; for example, persistence, transactionality and security all work this way. If you love CORBA but feel intimidated by the programming, CCM is the environment for you!

So, instead of the hundreds of instance activation patterns available to a POA programmer, a CCM programmer gets a choice of four and they're patterned on the EJBs we just examined. There are two variants with nonpersistent component references: the service component, which corresponds to a stateless session EJB, and the session component, which corresponds to a stateful session EJB.

The two CCM types with persistent (and therefore sharable) references don't correspond directly to the two EJB entity types even though one uses that name. Persistence, provided by OMG's new Persistent State Service (PSS), may be controlled by either the component or the con-

tainer for either of these types. The Process component lacks a key findable via its Home component while the Entity component has this key. Lacking a key, the Process component typically represents a process with a beginning and an end, one that may be interrupted and restarted later: applying for a mortgage, opening a bank account or shopping (where the shopping cart would be a Process component that lives only for the duration of a single shopping trip). Having a key and therefore retrievable by any client, the Entity component well represents permanent constructs, including the mortgage or bank account that we created with our Process components, or a Customer, or whatever.

With this, we leave the basic components behind and enter the extended world.

### Multiple Interfaces, Navigation, Segmented Persistence

Figure 1 shows an extended CORBA component with the four interface types that it uses to communicate in standard ways with the world. (There's a separate container-centric view that we won't cover because of its similarity to EJBs.)

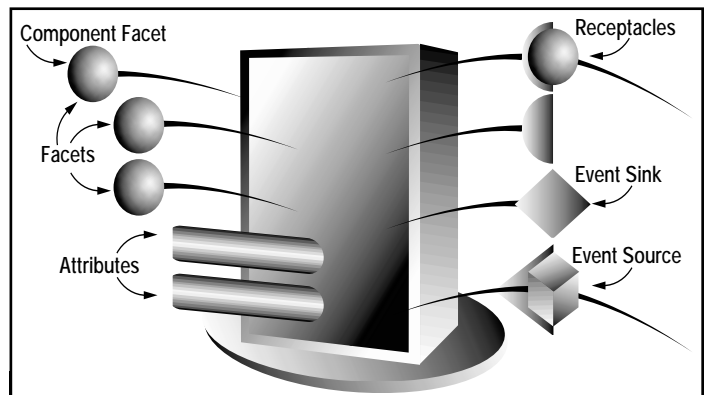


FIGURE 1 Extended CORBA component

*Component* is a new CORBA type specifically defined to support multiple interfaces and navigation among them. The multiple interfaces of a CCM are termed *facets*. The CCM infrastructure takes care of all the navigation, generating both the additional IDL operations and the code that supports them. Clients can navigate from any of a component's facets to its main facet, where they can get a list of all supported facets – in many ways, a CORBA analog for IUnknown. Or, if a client already knows where it wants to go, it can navigate from any facet directly to any other one. Extending this concept is segmented persistence, in which the unit of activation/passivation is the facet and its associated code and data, instead of the entire component.

### Attributes and Configuration

CCMs may be configured at installation. In a component marketplace this gives a single CCM product the flexibility to fit into multiple application assemblies with varying requirements (as long as they don't vary too much!). Configuration is accomplished by setting values of attributes (see Figure 1) at installation time. Once this is complete, a call to `configuration_complete` freezes the configuration and enables calls to the CCM's facets.

### Receptacles

Receptacles (see Figure 1) are the client-side interfaces that a CCM uses to invoke operations on other CCMs, analogous to the `ejb-link` feature of EJBs. In CCM, when you can specify client-object couplings in your application configuration file, the system automatically feeds the component reference of its target to a receptacle at invocation time.

### CCMs' Persistent State

Persistence of extended CCMs' state is handled by OMG's Persistent State Service. It's a lot more standardized than EJB's persistence, in which you configure container-managed persistence (either automati-



cally or manually) via vendor-specific tools at deployment time, or hard-code database calls into the bean itself for bean-managed persistence. (For basic CCMs the persistence environment is the same as for EJBs.) The PSS lets you define your persistent state in either of two modes: transparent persistence, in which you declare persistent variables in your programming language code, or by using Persistent State Definition Language (PSDL), a superset of OMG IDL that combines the flexibility of bean-managed persistence with standards-based portability.

### Event Handling

Another difference between EJBs and extended CCMs is event handling. Based on CORBA's Notification Service, event handling in CCM is totally distributed as all channels, endpoints, sources and sinks (see Figure 1) are either CORBA objects or clients. This lets you couple CCMs in one container with those in another, even across CCM application boundaries. Architecturally, it unifies messaging among components within a container with those in different containers, simplifying structure as applications grow larger. The Notification Service is very capable and flexible, with an elegant structured payload in addition to QoS controls and event typing and filtering. Since Java doesn't define a distributed event service, EJBs lack this capability.

### Multithreading

While EJBs allow only one thread to enter a container at a time, extended CCMs allow components that are written thread-safe to execute in a multithreaded container. The setting – either single- or multithreaded – is made in the configuration files.

## Conclusions

Many programmers (and their managers!) find the single-language Java environment confining despite its many delightful qualities. Others don't mind being confined to Java now, but know that this limitation will have undesirable consequences in the future. (Even Java applications will wear the "legacy" label someday!) For these people the CCM provides a way out – they can program in EJBs, and either integrate or switch to other languages now or anytime in the future.

We expect many others to look at the features of the extended CCM and adopt it soon after products become available. As we've shown here, the extended features provide the foundation for application architectures that would be much less elegant written any other way.

All of CORBA 3, including the CCM, is now OMG adopted technology. The CCM is currently undergoing its first maintenance revision; under its new procedure OMG waits for this first revision stage to complete before issuing a formal release with the incremented release number. This is scheduled for late 2000; products implementing the CCM are expected to hit the marketplace around the same time.

## References

For more on EJBs check any of Jason Westra's articles in back issues of **JDJ** including June and November 1999 (Vol. 4, issues 6, 11), and Rodrigues and Raj's article in August 1999 (Vol. 4, issue 8). For more on CORBA servants and the POA, see my **CORBA Corner** column in the December 1999 **JDJ** (Vol. 4, issue 12).

For access to all of OMG's specifications, surf to <http://www.omg.org/gettingstarted/specsandprods.htm>. For specifications that are part of CORBA 3, see <http://www.omg.org/technology/corba/corba3releaseinfo.htm>. ☪

### AUTHOR BIO

*Jon Siegel, the Object Management Group's director of technology transfer, was an early practitioner of distributed computing and OO software development. Jon writes articles and presents tutorials and seminars about CORBA.*

[siegel@omg.org](mailto:siegel@omg.org)

# Transactions and Exception Handling in EJB 1.1

Create more fault-tolerant code in your new EJB applications



WRITTEN BY  
JASON WESTRA

Many of you have been developing EJB applications since the 1.0 version of the specification. In the EJB 1.1 specification the approach toward EJB exception handling has changed slightly regarding the exceptions and transaction management responsibilities between bean providers and container vendors.

Those of you who made the conversion from 1.0 to 1.1 may not have enjoyed this “tightening” of the spec. Exceptions are thrown at many different levels in large applications; each exception has a specific meaning and is handled differently by your application. To change how your application handles exceptions can be a daunting task. For instance, you have to make sense of the container’s responsibilities versus yours as a bean provider. Also, you must understand how to use new exceptions correctly and which exceptions have been deprecated from certain usage.

This month I’ll talk about the main types of exceptions you’ll encounter when developing EJBs to the 1.1 specification. I’ll discuss the differences between exception and transaction management in the 1.0 and 1.1 EJB specifications and describe each type of exception, categorizing commonly thrown exceptions for you. My intention is to shed some light on exception handling in 1.1-compliant EJBs to either smooth your conversion or ease your new EJB development effort.

## EJB Exceptions in 1.0

The 1.0 specification divides exceptions and their effects on the current transaction based on who initiated the transaction and how exceptions are handled in the scope of the transaction. It can initiate a transaction three ways: client-managed, bean-managed and container-managed. Depending on your choice for transaction management, your recourse on a failed action is extremely different. Let’s examine how the 1.0 specification handled transactions and exceptions to better understand the differences between exception handling in a 1.0- and 1.1-compliant container.

## Client-Managed Transactions in 1.0

Client-managed transactions are transactions initiated on the client that generally span multiple calls to one or more EJBs. Because the client manages the scope of the transaction, it can assess exceptions thrown to it and determine whether to retry the business function or simply roll back the transaction, essentially calling it quits! The ability to retry calls to your EJBs without the transaction rolling back is a great feature, since remote method invocations are inherently less reliable than local calls. In a clustered environment we may be able to retry the same EJB on another server without disrupting the execution of the transaction.

## Bean-Managed Transactions in 1.0

Exceptions thrown in entity or stateless session beans that manage their own transactions automatically signal the container to roll back the transaction. The container doesn’t distinguish exception types; it simply marks the transaction for immediate rollback. Stateful session beans that manage their own transactions have more latitude than their counterparts. They may throw any exception, except an unchecked one, and still maintain their transactional state. This can be problematic and bean providers are advised to call `EJBContext.setRollbackOnly` to enforce transactional integrity.

## Container-Managed Transactions in 1.0

The flexibility of transaction management is near zero with container-managed transactions in 1.0. All exceptions, including application and system exceptions thrown from an EJB (not handled gracefully within the EJB), caused an automatic rollback of the transaction.

Containers can’t tell if the transaction’s integrity can be upheld in 1.0 unless you call `EJBContext.setRollbackOnly` to prevent a commit. However, calling `setRollbackOnly` seems like an extra step to take for the bean provider to cause a rollback, especially when throwing any exception has the same effect.

Due to a lack of foresight that the EJB specification would adapt, many early EJB applications allocated too much time to migrate to EJB 1.1 exception handling. The migration effort generally involves determining which application exceptions are fatal and adding `setRollbackOnly` calls into the EJB’s code.

## EJB Exceptions in 1.1

The EJB 1.1 specification groups exceptions into two categories to enable bean developers, client developers and container providers to handle exceptions and recoverability of transactions more effectively. Three types of exceptions may be thrown from an EJB: application, system and checked subsystem. The three types are detailed below.

### Application Exceptions

Application exceptions don’t subclass `java.lang.RuntimeException` or `java.rmi.RemoteException`. They represent errors in business logic specific to a use case or business service provided by your EJBs – for example, `AccountOverdrawnException` in a bank application or `ProcessOrderException` in an order management system. `ProcessOrderException` would be thrown from an `OrderMgrEJB` because the user didn’t include sufficient credit card information to correctly process an order.

The EJB specification even provides some basic application exceptions for use by bean providers. Standard application exceptions include `CreateException`, `Dupli-`

cateKeyException, FinderException, ObjectNotFoundException and RemoveException. Table 1 describes these exceptions and some user-defined ones.

Enhancing Standard EJB Exceptions:

Standard exceptions can be extended, if necessary, to show the client that a specific error occurred. However, I haven't had luck with subclassing mainly because Java doesn't support multiple inheritance. For instance, you subclass `javax.ejb.CreateException` to indicate more specifically that the parameters passed into `ejbCreate` are incorrect. The new exception name is `InvalidDataException`. Later, when you perform data validation in an update method, the `InvalidDataException` isn't reusable because it applies (or should apply) only to create methods, since it inherits from `CreateException`. Therefore a generic exception that indicates data is invalid warrants an ancestor separate from the predefined `javax.ejb` exceptions. This allows it to be reused in update and insert scenarios.



Subclassing standard EJB exceptions can be effective for other reasons, such as providing a better description of root errors. A subclass exception that allows nesting of exceptions is common. The application exceptions provided for you don't have nesting capabilities to allow handlers to query more information about the error. Currently they have empty constructors or their constructors take only a String message about the error. If your application wants this more informative exception tracking, you'll need to enhance it through inheritance.

### System Exceptions

System exceptions include `RuntimeException`, `RemoteException` and all exceptions that inherit from them. The EJB specification provides not only standard application exceptions, but version 1.1 also introduced a standard system exception, `EJBException`, that subclasses `RuntimeException`.

The EJB 1.1 specification describes the container's responsibility in the exception-handling process in more detail than in 1.0. It slightly modifies the way containers handle system excep-

tions versus application exceptions. System exceptions cause the container to automatically roll back its transaction if executing within one, while any application exception is allowed to pass to the client for handling.

Table 2 details how an EJB container handles system exceptions. First, a container always marks an existing transaction for rollback whether or not the transaction was initiated by the container or a client.

Next, the exception is logged to allow system administrators to monitor errors in the application that might be system related. Logging of system exceptions is mandated in the EJB 1.1 specification; however, the implementation is left up to the vendor. Some vendors have simple logging facilities, while others allow you to notify third-party monitoring tools and send pages to system administrators.

The container must remove the bean instance, discarding it to prevent its use in another transaction. At this point the instance may have corrupt information and might not be cleared correctly before being reused. Discarding the whole bean instance eliminates any doubt about interacting with unsafe information. The process of discarding a bean instance includes removing it from any instance pooling cycles, dereferencing the object and allowing it to be garbage collected.

When a bean instance is discarded, the client isn't affected as long as the EJB is either stateless or an entity bean. Either of these can be re-created without loss of state since entity bean state is re-created from persistent storage and stateless beans well...don't have state! However, when a stateful bean rethrows a system exception, the container discards it and the client loses its reference back to the EJB and any session data it held.

Last, the container handling the system exception will throw a `RemoteException` if the transaction was initiated from the container, or `TransactionRollbackException` if a client is managing the transaction. `TransactionRollbackException`, a subclass of `RemoteException`, indicates that the client's transaction has failed and a new one should be started.

Table 3 details common system exceptions encountered in EJB applications.

### Checked Subsystem Exceptions

Checked subsystem exceptions are exceptions thrown from various components for a J2EE server including JNDI, JDBC and JMS. Don't rethrow system exceptions or checked subsystem exceptions if caught by your application.

Exception	Defined	Entity or Session	Purpose
<code>CreateException</code>	<code>javax.ejb</code> package, EJB 1.0	Both	Thrown from <code>ejbCreate</code> to indicate an error in creation of the EJB.
<code>DuplicateKeyException</code>	<code>javax.ejb</code> package, EJB 1.0	Entity	Thrown in <code>ejbCreate</code> to indicate a duplicate entity already exists for the primarykey.
<code>RemoveException</code>	<code>javax.ejb</code> package, EJB 1.0	Both	Thrown from <code>ejbRemove</code> to indicate an error in removal of EJB.
<code>FinderException</code>	<code>javax.ejb</code> package, EJB 1.0	Entity	Thrown from EJBHome finder methods to indicate an error occurred processing the finder. Not thrown if finder completes and finds no rows!
<code>ObjectNotFoundException</code>	<code>javax.ejb</code> package, EJB 1.0	Both	For session beans, indicates the session has been terminated and the object reference is no longer valid. For entity beans, indicates the entity has been deleted from persistent storage.
<code>AccountOverdrawnException</code>	User-defined	Both	An example of a user-defined application exception that indicates an account balance is overdrawn. It could be thrown from a session bean or the account entity bean itself.
<code>ProcessOrderException</code>	User-defined	Session	An example of a user-defined application exception that's thrown when an order isn't fulfilled because of missing or invalid information.

TABLE 1 Application exception examples

Transaction Type	Container Responsibilities
Container-managed (Required, RequiresNew only)	Roll back current transaction Log error to system error log Remove invalid bean instance from container Rethrow java.rmi.RemoteException to caller
Client-managed	Mark client's transaction for rollback only Log error to system error log Remove invalid bean instance from container Throw javax.transaction.TransactionRollbackException to caller

TABLE 2 Container's responsibilities for system exceptions

Exception	Defined	Entity or Session	Purpose
RemoteException	java.rmi package	Neither, container thrown only	Thrown by container to indicate a communications error between distributed objects or to a remote service.
RuntimeException or	java.lang package	Both or Container	Thrown if a runtime error is encountered such as a null pointer or an array index is out-of-bounds.
EJBException	javax.ejb package, EJB 1.1	Both	Thrown from within an EJB's code to indicate an irrecoverable action and have the container roll back transaction.
NoSuchEntityException	javax.ejb package, EJB 1.1	Entity	A subclass of EJBException, it indicates that the underlying entity bean was removed from persistent storage.
TransactionRollbackException	javax.transaction package, JTA	Neither, container thrown	Thrown to indicate that the transaction is set for rollback.

TABLE 3 System exception examples

Exception	Defined	Subsystem	Purpose
NamingException(s)	javax.naming package	JNDI	Indicates that an error in a Context or DirContext of the naming service. Subclasses help pinpoint exactly what happened.
SQLException	java.sql package	JDBC	Thrown if an exception occurs processing SQL through JDBC.

TABLE 4 Checked subsystem exception examples

Typically, look for checked exceptions from J2EE subsystems such as JNDI, JDBC, JMS and RMI, and nest them in an EJBException, throwing the EJBException to indicate to the container that it should mark the current transaction for rollback. Table 4 details two common checked subsystem exceptions, NamingException and SQLException.

Listing 1 shows an example of wrapping a checked subsystem exception. In this example, if a JDBC call fails to `stmt.execute("select * from orders")`

correctly, the SQLException should be rethrown as an EJBException instead of declaring SQLException in the throws clause. Adding numerous checked exceptions to the throws statements of a method signature opens up the client to the numerous subsystems of the J2EE platform and should be avoided to prevent complexity. *Note:* There's neither logging nor a print of its call stack – this duty is left up to the container, which must report all system exceptions as indicated by the 1.1 specification.

Although EJBException has multiple constructors, I generally use the one that nests the original exception to allow the caller to get the root cause out of it when it's caught on the client side. Because EJBException is a subclass of RuntimeException, you don't need to code a throws clause. However, this does put a burden on the client developer, who must look explicitly for EJBException and open it up, calling `getCauseByException` to get to the root cause of the error. Client exception handlers should be aware that if EJBException's nested constructor isn't used, the `getCauseByException` will be null. You should code a check for "null" and handle this appropriately to prevent throwing a NullPointerException at runtime.

## Summary

The EJB 1.1 specification clarifies exception handling and transaction management for both bean providers and container vendors. Specifically, the spec made the distinction between system and application exception handling for each role. This distinction allows bean providers and client developers to create more fault-tolerant code that retries transactions when application exceptions occur. Also, the requirement that all system exceptions trigger containers to automatically roll back transactions helps define the scope of accountability for transaction management in containers versus client and beans. I hope this month's **EJB Home** was informative and can be applied to your current or next EJB project. ☺

[jwestra@vergecorp.com](mailto:jwestra@vergecorp.com)

### Listing 1: Handling Subsystem Exceptions

```
try {
    // call helper method to get
    a connection
    java.sql.Connection conn =
    this.getConnection();
    java.sql.Statement stmt =
    conn.createStatement();
    stmt.execute("select * from
    orders");

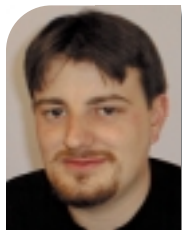
    // handle ResultSet
    ...
}
catch(SQLException ex) {
    // wrap subsystem exception
    and throw EJBException
    throw new EJBException(ex);
}
```

#### AUTHOR BIO

Jason Westra is the CTO of Verge Technologies Group, Inc. ([www.vergecorp.com](http://www.vergecorp.com)). Verge is a Boulder, Colorado-based firm specializing in e-business solutions with Enterprise JavaBeans.

# What's It All About?

Maybe not *a//*about Java, but of interest anyway . . .



WRITTEN BY  
ALAN WILLIAMSON

know what you're thinking: What happened to Straight Talking? It's a good question, and most certainly deserves an answer. As seasoned *JDJ* readers know, Straight Talking was a regular column for over two years. But as much as I loved "talking" to you every month, the time has come to move on and introduce a whole new column/format.

As you know, I have the honor of sitting on the editorial board of *JDJ* with the title of "Industry News Editor." So what does this mean? Well, in a nutshell, I get to hear about all the juicy things going on in the industry a wee bit before the masses are informed. My inbox regularly fills up with press releases, announcements and must-reads from companies all over the globe. Most of it I merely filtered and passed on down the chain to other *JDJ* departments.

Many of the news items aren't that specific to Java but to the industry as a whole, and although this is a Java magazine, I felt that some of this information might be of interest to you. So with that in mind, I made the tough decision to step back from *JDJ* for a month or two, refocus my writing efforts and present the first in a new column devoted to filtering through the news stories and presenting you with an overview of what's been happening in the world.

Our more educated readers will have twigged that *JDJ* is a monthly publication and having a monthly news report just isn't going to be that cutting edge. To that end, we've built a complete Web site to accompany this column. On it you'll find the latest press releases complete with a weekly summary of the goings-on. The site will serve as a conduit of all things Java related, collating views on all aspects of our industry. But we need your help.

[www.n-ary.com/industrywatch/](http://www.n-ary.com/industrywatch/)

The Web site needs your input. It's not a static bunch of HTML pages but a dynamic, content-rich experience, and we want your reviews and your articles. We aim to raise the standard on community-driven Web sites. So work with us to build something rather exciting.

Keith and I have been working hard, practicing and perfecting our radio show. We've finally prepared the necessary infrastructure to broadcast to you on a daily basis (we also want to hear from budding radio DJs interested in hosting their own shows). So **Industry Watch** is coming at you not only in electronic and print form, but in audio format.

**Industry Watch** will be a monthly review/summary of what's been going on in the world at large. I suspect the overall formula will need a little tweaking and changing as we gather more feedback, so work with us in the early days and we'll soon have a well-oiled machine.

## dot-com Fever Breaks

It's fair to say that everyone reading this magazine has an interest in Java in some shape or form. Furthermore, it's probably not going out on a limb to say that the majority of you are involved in the Internet either in B2B or B2C activities. Now, in the middle of the third quarter, many companies have released their second quarter earnings, and for the dot-coms of the world it makes pretty sorry reading.

Here in Europe we've seen the demise of two high-profile e-tailers: boo.com and clickmango.com. Both ran out of money and couldn't raise the necessary funds to keep them going until they hit more profitable times. Of the two, I found the story behind boo.com to be the most interesting. Apparently, one of the reasons for its failure to pick up the necessary user base was that the technology employed at its Web site was too cutting edge: only those with a high bandwidth connection were able to utilize the site properly.



It's interesting to see that while we all strive to use the latest and greatest tools, sometimes it doesn't pay to be on the bleeding edge. Solutions have to be delivered that will allow the widest user base to utilize them. The notion of faster bandwidth has been promised for many years now, but the majority of users are still surfing the Net with their dial-up analog modems. They're trying not to get too impatient waiting for that huge JPEG image to download, which will do nothing to enhance the experience except to make them wary of clicking on the next link to drive them deeper into the site.

I've read many reports and press releases claiming B2C is dead and long live B2B. I can safely say that the majority of them that have come from the application server vendors have played on their B2B features as opposed to their wider appeal. So what's happening to the poor old consumers? Are they going to be left behind?

I hope not, although it would appear that making profits in this area is increasingly difficult. Intel and SAP recently announced they were giving up the ghost and shutting down their operations as they couldn't see a path to profit. Amazon, no stranger to losses, announced they had lost \$115.7M in the last quarter - which wasn't as bad as the first quarter, which recorded a loss of \$121M. Even Barnes and Noble, a traditional "bricks and mortar" company, posted losses of \$45.4M for the last quarter for its online operation.

So just who is making money in the B2C world? Well, it appears that companies that aren't warehousing any stock aren't doing too badly. eBay posted profits of \$11.6M, but Expedia recorded a \$42.2M loss. It just doesn't make sense.

I recently spoke to a VC company about all of this. They said that a year ago many of the original investors were quite happy to wait the two to four years for market share and realization of profits. But with all the bad press, some are beginning to get a little itchy and impatient, and fail to come up with the once promised second and third rounds of funding.

That said, the B2B world is doing just fine and dandy. Sun posted record earnings: \$5 billion in revenue for just one quarter, totaling more than \$15 billion at year's end. So if anyone ever asks if Sun is making money out of Java, I think you can take the answer as Yes. The application server market is very buoyant, with Sun's iPlant securing over \$1 billion in revenues alone. Not bad for a company that's primarily a box shifter.

Looking at all the trade journals and financial papers, it appears that those making money out of the Internet are those that are tooling up the rest of the industry. The dot-coms that have invested heavily in these tools appear to be taking the beating, fighting vigorously in the B2C marketplace, trying to stay ahead before the grim reaper (the CFO) comes a'knocking.

For us Java developers it would seem on the face of it that that book on EJBS

might not seem such an extravagant purchase after all. As an entrepreneur, I don't know what it signals. Reading through the posts of the UK's premier forum for start-ups, First Tuesday ([www.firsttuesday.com](http://www.firsttuesday.com)), it seems that investors have gone cold on the B2C ideas, and are looking to take the safer route of B2B instead.

I wonder how the industry will look in 12 months' time. What technologies should we all be looking to invest our skill and energy in, to stay ahead of the competition? Investing in Java alone is no longer sufficient - we have to narrow our focus even further.

Speaking of Java, I noted that Kim Polese, the "mother of Java," has stepped down from the CEO post at Marimba to take on a less prominent role as chair and CSO. Now there's a company we don't hear much from. Marimba didn't fare well when push technology suddenly went out of vogue. But with a repositioning, they adapted their technology to manage corporate software on open networks. An astute move now that they recently posted their first-ever earnings, with a \$1.3M profit. I tip my hat to Ms. Polese for not allowing her ego to get in the way of her company's growth. A rare thing for a founder to voluntarily step down and allow someone else to run her

baby. I can't image Larry Ellison or Scott McNealy doing the same.

As this column was being prepared, Microsoft featured heavily in many areas of news. If they aren't being investigated by yet another government for alleged market dominance, they seem to be still moving forward, announcing new alliances and products. The recent U.S. court case doesn't seem to have slowed down their business in the slightest. One even gets the feeling they're cashing in on all the free publicity.

Next month I'll review what's been happening on the C# front, Microsoft's recently announced new language that officially has nothing to do with Java. Yeah, sure!

Remember to check out [www.n-ary.com/industrywatch/](http://www.n-ary.com/industrywatch/) for regular updates on the moves of our great industry. ☺

#### AUTHOR BIO

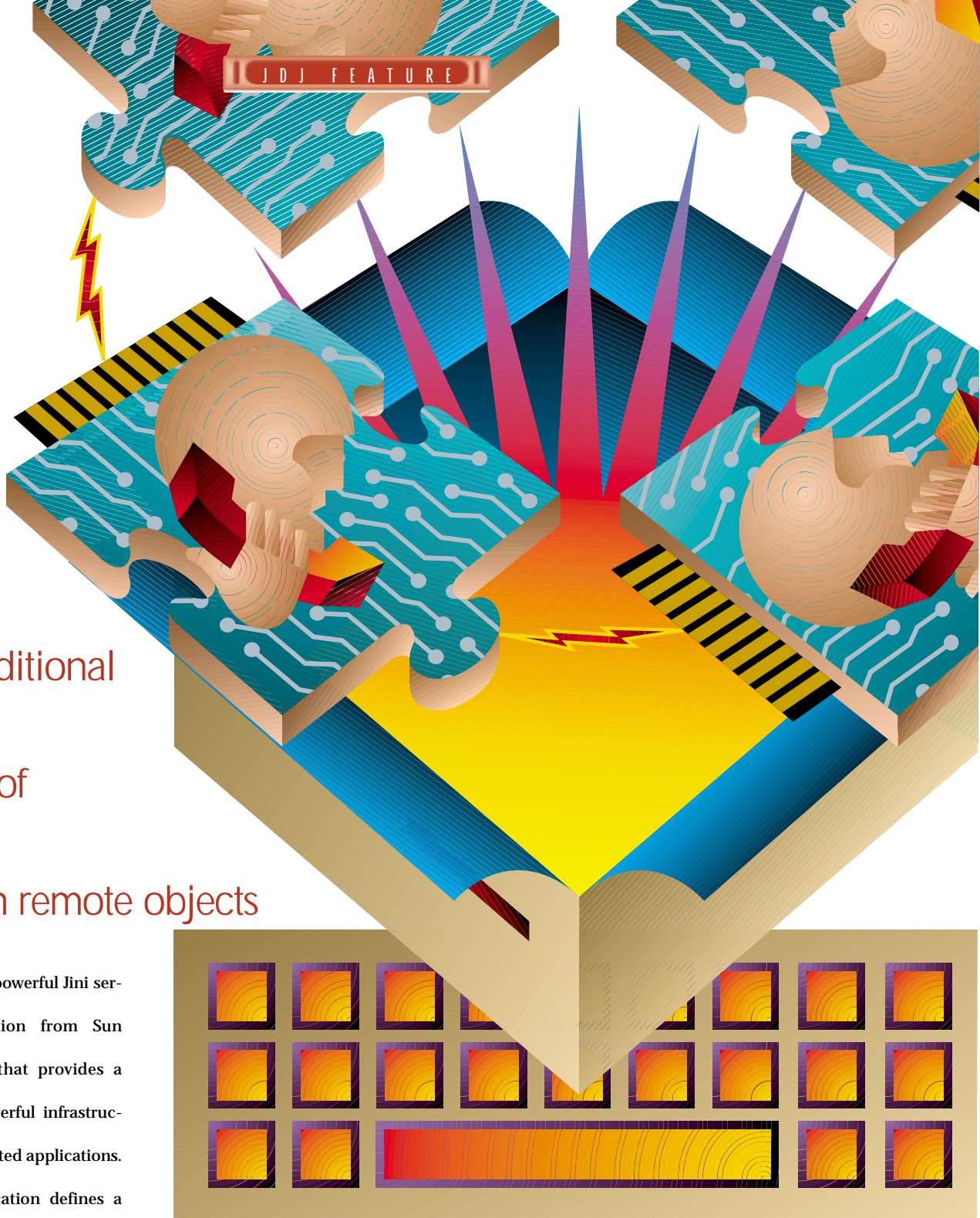
*Alan Williamson is CEO of the first pure Java company in the UK, n-ary (consulting) Ltd ([www.n-ary.com](http://www.n-ary.com)), a Java solutions company specializing in delivering real-world applications with real-world Java. Alan has authored two Java Servlet books and contributed to the Servlet API.*

[alan@n-ary.com](mailto:alan@n-ary.com)

WRITTEN BY SANJAY MAHAPATRA

A new and  
futuristic  
paradigm  
far from traditional  
invocation of  
methods on remote objects

**J**avaSpaces is a powerful Jini service specification from Sun Microsystems that provides a simple yet powerful infrastructure for building distributed applications. The JavaSpaces specification defines a reliable distributed repository for objects, along with support for distributed transactions, events and leasing. In the JavaSpaces programming model, applications are viewed as a group of processes, cooperating via the flow of objects into and out of "spaces."



# INTRODUCING JAVASPACE

## Inspired by “Tuple-Spaces”

JavaSpaces is based on the concept of “tuple-spaces” first described in 1982 in the Linda programming language and system originally pro-pounded by Dr. David Gelernter at Yale University. The public-domain Linda system is a coordination language for expressing parallel process-ing algorithms without reference to any specific computer or network architecture and provides interprocess coordination via virtual shared memories or tuple-spaces that can be accessed associatively.

The tuple-space model is especially useful for concurrent algorithms. Although JavaSpaces technology is strongly influenced by the Linda sys-tem, it differs from Linda in several ways – such as Java’s richer typing, object orientation, subtype matching and transactional support span-ning multiple spaces, leasing and events.

## JavaSpaces in Today’s Networked World

Although the programming model and concepts underlying Java-Spaces may at first seem abstract and theoretical, JavaSpaces does in fact provide an elegant and practical approach to solving common real-world scenarios in today’s increasingly networked world. Distributed, collaborative and parallel applications can be solved using a JavaSpaces-based solution since JavaSpaces lends itself to applications such as trad-ing services, reservation systems, online ordering systems, online auc-tion systems, large computation-intensive jobs, workflow systems, mobile offices and agent technology.

## A Distributed Algorithm as a Flow of Objects Between Spaces

Any application developed using JavaSpaces technology would need to be modeled as a flow of objects through one or more spaces. Java-Spaces represents a fundamentally different approach from that of the more traditional invocation of methods on remote objects or direct exchange of information between processes. With the method-invo-cation approach, specific remote interfaces are needed for each operation. With the JavaSpaces’ flow-of-objects approach, on the other hand, only one interface is required: namely, the JavaSpaces interface (from the package `net.jini.space`).

## What’s a Space?

The term *space* refers to an implementation of the JavaSpaces service specification. Like the rest of the Java technology, there’s a separation between the specification and its implementation of the specification. A number of vendors may provide implementations. A JavaSpaces client is expected to receive identical service and functionality from all such space implementations, though their internal designs may vary. A space, then, is a particular implementation of the specification and represents a persistent object exchange “area” via which remote processes can coordinate their actions and exchange data. The space thus provides a ubiquitous, cross-platform framework for distributed computing.

## A SHORT NOTE ON JINI

Jini (not an acronym for anything and pronounced like the word “genie”) is technology from Sun Microsystems that represents a network-centric and object-oriented paradigm for distributed com-puting. Jini enables a wide variety of devices from diverse manufacturers to interact via interfaces to objects rather than via common network protocols, and raises the level of abstraction from the communication protocol level to an object-oriented, software-interface level.



## Advantages of JavaSpaces

The JavaSpaces model has numerous advantages:

1. Multiple processes can access data concurrently.
2. There is a loose coupling between senders and receivers, which enhances software reuse and flexibility of design.
3. The model has extremely high scalability.
4. At the same time it is simple, flexible and reliable.
5. The space provides support for distributed events and leasing.
6. Atomicity, transactional security, synchronization and coordinated concurrent access are inherently built into JavaSpaces.

## Characteristics of the “Space”

A space is a shared, persistent, associative, transactionally secure and network-accessible repository for objects that allows processes to read, remove and insert objects into itself and thus communicate by such an exchange of objects through one or more spaces. Let’s review the chief characteristics:

- **Shared:** A space is a network-accessible, shared region of memory with which multiple remote processes can concurrently interact. The space manages the details of concurrent access, leaving you to focus on your application’s protocols.
- **Persistent:** A space provides a reliable storage mechanism. An object that is stored in a space will reliably remain in the space until it’s removed from the space or until its lease time – as specified at the time of writing into the space – is up.
- **Associative:** A space supports an associative lookup mechanism. An “associative lookup” provides a means of finding objects of interest according to their content and type, rather than by name or memory location.
- **Transactionally secure:** A space supports a transaction model, which ensures that all operations on a space are atomic and transactionally secure. A single space transaction may include multiple operations on one or more spaces.

## “Space” Rules

While within the space, objects can’t be modified or methods invoked on them. Processes must exclusively remove objects from the space, update or invoke methods on the objects only while outside the space, and then reinsert objects back into the space, as required from the appli-cation standpoint. The insertion and removal of an entry into and from a space are guaranteed to be atomic.

## JavaSpaces Interfaces

Each implementation of a JavaSpaces service will export objects to the local client that implement the JavaSpaces interface (from package `net.jini.space`), via which the local client can interact with the remote JavaSpaces service. The JavaSpaces interface isn’t a remote interface.

## What Can Be Placed into the Space

A space stores “entries.” An entry is a class in the Java platform that implements the Entry interface (package `net.jini.core.entry`) as defined in the Jini Entry Specification. Once you’ve created an entry, you’re ready to perform operations that interact with the “space.” Any class whose instances are intended to be written into a space or used as the basis of an associative lookup must implement the Entry interface.

## Associative Lookup Using Templates

The associative lookup functionality provided by the space is based on the template’s fields. A template is an entry object of the same class or a superclass of the entry of interest, with its fields set to values that will be matched against entries within the space in a read or take oper-



ation. Null fields in the template act as wild cards. In case there are multiple entries that match a template, only one will be returned. The space makes an arbitrary choice – there's no ordering of entries within the space.

## Operations

The four primary operations that can be applied to a space are write, read, take and notify.

### Write

The write() operation places a copy of an entry object into the space such that the entry can subsequently be accessed via a read or take operation.

### Read

The read() operation returns an entry that matches the template, without removing the entry from the space. Multiple processes can read the same entry at the same time. If a matching entry isn't available immediately, the read() operation will wait for an entry to be introduced into the space, or until its timeout period is up. The constant `JavaSpaces.NO_WAIT` may be used as the timeout value, in which case the read request will return immediately, even if no matching entry is found. The `readIfExists()` works like the `read()` except that it returns immediately without waiting for the timeout period unless a matching entry happens to exist within a currently occurring transaction – in which case `readIfExists()` will wait up to its timeout value for that transaction to complete. Thus `readIfExists()` returns immediately except in cases when a matching value exists within a transaction.

### Take

The take() operation returns an entry that matches the template and simultaneously removes the entry from the space. The take() operation is atomic, thus ensuring synchronization and concurrency. The `takeIfExists()` operation works in a manner similar to the `readIfExists()` operation.

### Notify

The notify() method is used to register interest in the arrival of an entry into the space that matches a specified template. The space notifies you that an entry that matches your template has arrived into the space. This notification and distributed event mechanism relies on the Jini distributed event model as described in the Jini Distributed Event Specification.

The concepts of entry, template and operations are central to JavaSpaces. Though fairly straightforward, they're immensely powerful.

## Leasing and the Lease Interface

The Lease interface from the package `net.jini.lease` includes signatures for methods including `getExpiration()` and `renew()`. A request to write an entry into a space is accompanied by the requested lease time, the time for which the caller requests the entry to be stored within the space. The write operation actually returns a Lease object representing the lease time granted by the space, which may be less than the requested lease time. When the granted lease time is up, the entry is removed from the space. Thus any entry written into a space may be removed either explicitly with the take() operation or when its granted lease time is up. Once a lease has been granted for a finite period of time, it may

subsequently be renewed. The lease time associated with a write operation may be requested and granted for the constant `Lease.FOREVER`, which means that the entry will remain in the space persistently until such time that it is explicitly removed from the space via a take operation. Leasing, as applicable to JavaSpaces, is defined in the Jini Distributed Leasing Specification. Leases and lease times may also be applied to transactions. Just as an entry is removed from a space when its lease time is up, a transaction will be aborted if it has not completed when its lease time is up, in which case none of the transaction's constituent operations will be committed.

## Distributed Events

JavaSpaces technology provides a distributed event model that uses the Jini event model as described in the Jini Distributed Event Specification. Just as the event model in the single JVM environment comprises the event source, event object and event listener, the Jini distributed model comprises the event source, the remote event object and the remote event listener. The space acts as the event source that fires events when entries are written into it and notifies processes that have registered interest in entries that match specified templates.

## Transactions

Transactional support is an essential aspect of any distributed service framework, and JavaSpaces technology provides a distributed transaction service based on the Jini transaction model as described in the Jini Distributed Transaction Specification. The JavaSpaces transaction model simplifies the use of the more general Jini transaction model, and provides a means of grouping multiple space operations on one or more spaces. A process that requires transactional functionality will need to look up the transaction manager (a remote service), obtain the transaction from the manager, then pass the transaction to each space-based application that's intended to be part of the transaction. The transaction manager oversees all transactions. A transaction is actually a leased resource; if a

transaction is neither explicitly committed nor explicitly aborted, the transaction manager will abort it upon expiration of its lease. Transactions affect space operations in several ways: for instance, an entry that is returned by a read() operation that's part of a transaction can't be taken by a take() operation that's part of another transaction until the first transaction is completed.

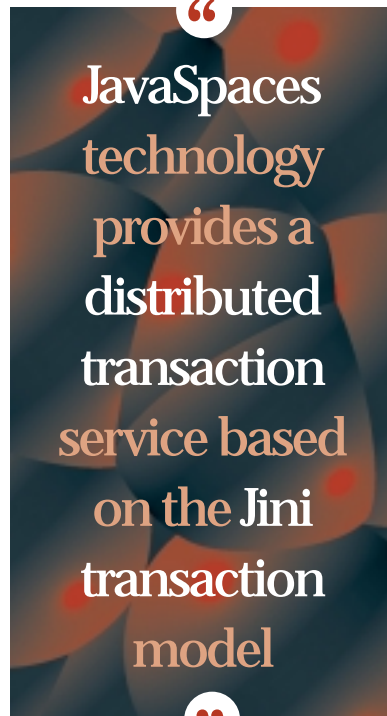
## Understanding JavaSpaces Technology in Relation to Other Technologies

### JavaSpaces and Three-tier Architecture

A JavaSpaces application typically represents the middle tier of a three-tier model, since clients and servers throughout the network can exchange objects via the space. The main advantage is high scalability and concurrency. JavaSpaces technology supports the thin-client model.

### JavaSpaces and J2EE

The J2EE technology groups several technologies – EJB, Java Servlets, JavaServer Pages, JavaMail, Java Message Service, JDBC, CORBA technology and so on. The emphasis is on interaction with existing enterprise resources and standardization so that the consumer's past and current investment in IT efforts is protected by the use of standard interfaces,



ease of legacy integration and application server vendor/implementation independence. JavaSpaces and Jini represent a new and futuristic paradigm that endeavors to meet the complex needs of tomorrow's increasingly networked world by emphasizing dynamic communication, spontaneous "lookup 'n' discovery" and network enablement.

### *JavaSpaces and Messaging Systems*

JavaSpaces goes far beyond a mere messaging system. It's a powerful and flexible service that provides a distributed object repository with persistence storage along with support for events, leasing and transactions. JavaSpaces isn't limited to any particular architecture such as point-to-point or publish/subscribe. For instance, publishers and subscribers in the publish/subscribe messaging architecture use subject-based messaging, while JavaSpaces provides a more flexible associative lookup by value and type.

### *JavaSpaces and JMS*

Java Message Service provides a high-level generic interface to messaging products that support the JMS API. While both JavaSpaces and JMS represent a model with uncoupled senders and receivers, JavaSpaces provides support for leasing and represents a more generic model that is not limited to the point-to-point and publish/subscribe models supported by JMS.

### *JavaSpaces and Databases*

Although JavaSpaces provides a distributed object persistence service, it's neither a relational nor an object database. A space is an unordered collection of objects that may contain data as well as the behavior. JavaSpaces focuses on supporting ease of writing distributed applications rather than merely providing a data repository functionality. There's also a difference in the querying mechanism supported by JavaSpaces in that querying is only for exact-match-or-don't-care for a particular field. A space can find, match and reference objects by both type and value, which means any object-based program or device can join a JavaSpaces system.

## Summary

The JavaSpaces-related API is small-sized and simple, yet the concepts of space operations, distributed events, leases and transactions when applied together form a powerful infrastructure for building robust, large-scale, distributed applications.

## References

1. JavaSpaces Specification 1.0, Sun Microsystems 1999.
2. Freeman, E., Hupfer, S., and Arnold, K. (1999). *JavaSpaces: Principles, Patterns and Practice*. Addison-Wesley.
3. Waldo, J. "The End of Protocols." Java developer connection: <http://developer.java.sun.com/developer/technicalArticles/jini/protocols.html>
4. Freeman, E., and Hupfer, S. *Make Room for JavaSpaces, Parts 1, 2 and 3*. <http://www.javaworld.com/javaworld/jw-11-1999/jw-11-jiniology-2.html>
5. Jini Entry Specification 1.0.1, Sun Microsystems 1999.
6. Jini Entry Utilities Specification 1.0.1, Sun Microsystems 1999.
7. Jini Distributed Leasing Specification 1.0.1, Sun Microsystems 1999.
8. Jini Distributed Event Specification 1.0.1, Sun Microsystems 1999.
9. Jini Distributed Transaction Specification 1.0.1, Sun Microsystems 1999. ☺

### AUTHOR BIO

Sanjay Mahapatra is a Sun certified Java 1.1 programmer and architect (Java Platform 2) and works for Cook Systems International, Inc., a consulting and solutions company.

[sanzem@hushmail.com](mailto:sanzem@hushmail.com)

# Encoded Streams

## Read and write encoded data with Java I/O streams



WRITTEN BY  
MIKE JASNOWSKI

Two basic types of data – text and binary – are used in applications to create files such as documents, images, video, text and executables. Certain applications, however, may need to alter a file to make it available to other applications; for example, e-mail requires text and binary data to be encoded before it's sent.

This article discusses a technique used to read and write encoded data using Java I/O streams. We'll define encoding and cover some of its history, examine two I/O stream classes and an interface, then finish by applying this technique to both a text and a binary file using the Base64 encoding scheme. With this technique you can provide encoding in your applications as well as encoded user information for authenticating against HTTP servers. This technique is provided using a standard, familiar group of Java classes: the I/O streams.

### What Is Encoding?

Encoding manipulates and reorganizes bytes so they can be understood by other applications (see Figure 1). This is done primarily for Internet e-mail systems, but is also used in places like basic authentication. Basic authentication requires the user ID and password to be encoded using Base64. Although encoding has been around awhile, you probably never knew it. For example, your e-mail and attachments could be encoded before being sent and decoded when received. E-mails specify encoded content by using the Content-Transfer-Encoding header. This header field can have the following values:

- 7Bit
- Quoted-Printable
- Base64
- 8Bit
- Binary

One side effect of encoding is a possible increase in the size of your data. It all depends on the encoding scheme you're using.

Now that we have some basics, let's look at the `EncodedInputStream` and `EncodedOutputStream` classes, which are used to read and write encoded data.



FIGURE 1 Basic flow of encoding and decoding

### EncodedInputStream

The `EncodedInputStream` takes encoded data and give it back as a byte array. Convert this data to any form you wish, such as text (see Listing 1). Its constructor takes two arguments: `InputStream` and `EncodingScheme`. The `InputStream` course could be a `FileInputStream` or even a socket.

```
Base64EncodingScheme scheme = new
Base64EncodingScheme();
EncodedInputStream eIn = new Encoded-
InputStream(new
FileInputStream("encoded.txt"),scheme
);
Byte data[] = eIn.readEncoded();
```

This class overrides the `read` method and adds a method called `readEncoded`, which reads encoded data and returns it as a byte array. The `read` method has been overridden to always return a -1. Initially this was done because the `read` method returns single bytes; when decoding data, you may be working with more than a single byte at a time.

### EncodedOutputStream

The `EncodedOutputStream` writes out data using whatever encoding scheme you specify (see Listing 2). Its constructor takes two arguments: `OutputStream` and `EncodingScheme`. The `OutputStream` can be almost any kind of

stream, such as a `FileOutputStream` or a socket.

```
Base64EncodingScheme scheme = new
Base64EncodingScheme();
EncodedOutputStream eOut = new
EncodedOutputStream(new FileOutput-
Stream("encoded.txt"),scheme);
eOut.write("This is unencoded
data".getBytes());
```

This class will buffer output as it's written to the class, encode the data, then write it out to the actual `OutputStream` specified in the constructor. Use it as you would any other I/O stream – just write either an integer or a byte array and the data will be encoded using the scheme you passed into the constructor.

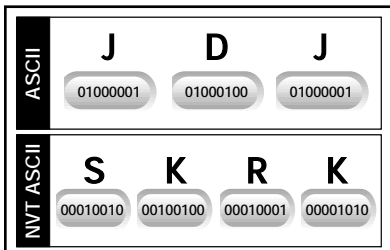
### EncodingScheme

Let's look at the `EncodingScheme` interface. It's a class that provides different encoding implementations such as the Base64 used in this article (see Listing 3). Its two methods are `encode` and `decode`. The `EncodedInputStream` and `EncodedOutputStream` delegate to this class when writing and reading the data. Rather than impose different encoding scheme implementations on a user of the stream, developers can plug in different encoding schemes (Quoted-Printable, 7Bit and Base64) and use familiar methods to read and write data without requiring significant changes to their code.

### Base64 Encoding Scheme

Before moving to our sample application, we need to implement an encoding scheme; I'll show the Base64 encoding scheme. This scheme basically reorganizes three 8-bit chunks into four 6-bit chunks (see Figure 2). These four 6-

bit chunks are represented using a special NVT ASCII character set. The “=” sign is used to pad chunks that aren’t a multiple of 3 bytes. You must also organize encoded data into chunks no greater than 76 bytes each. A more formal explanation is available in RFC 2045. As noted previously, encoding increases the size of your data. Base64 increases the size by approximately one-third.



**FIGURE 2** Three 8-bit chunks reorganized as four 6-bit chunks

The basic flow of the encode method is to work with 3 byte chunks at all times. When you reach the end of your data, pad with the “=” character. After each iteration of the loop, 4 bytes will be writ-

ten out to the buffer. When the loop has completely passed through all the data, padding is added and the encoded byte array is returned. The decode method operates almost the same except it works with 4 byte chunks instead of 3 and ignores the padding character (see Listing 4).

### Sample Application

Let’s put our encoding scheme to use. Our first example encodes a Java source file, then decodes it (see Listing 5). Compile EncodingSample and then run it, specifying HelloWorld.java as the argument (see Listing 6). Once it’s finished running, look at the contents of the encoded.txt file to see what the file looks like in its encoded state.

Now take the HelloWorld Java class file, encode it and then decode it. If you haven’t already done so, compile the HelloWorld.java file and then run EncodingSample, specifying HelloWorld.class as the argument. Then look at encoded.txt file to see what the file looked like encoded. To prove the file was successfully decoded, type “java

HelloWorld” – you should see “HelloWorld” printed out.

### Enhancements

While EncodedInputStream and EncodedOutputStream allow you to easily read and write encoded data, some enhancements can be made. Buffering large datasets makes it easy to decode all at once but may cause intermittent OutOfMemoryErrors. Alternatively, data can be encoded and decoded in chunks rather than all at once. Due to time constraints I was unable to implement this feature.

### Summary

It’s easy to provide an extensible means to read and write encoded data using ordinary Java I/O streams. You can also provide your own EncodingScheme implementations and plug them into your code without changes. For all you sun.misc.BASE64Encoder users, you now have a documented way to use Base64 encoding. Good Luck! ☺

[boopan@msn.com](mailto:boopan@msn.com)

#### AUTHOR BIO

Mike Jasnowski, a Sun-certified Java programmer, has over 17 years of programming experience and over three years with Java. He works for a software company in Kansas City, Missouri.

#### Listing 1

```

/*
 *
 * EncodedInputStream
 *
 * This class is used to decode a stream of data that has
 * been encoded
 *
 *
 * @author Mike Jasnowski
 * @version 1.0 , 06/01/2000
 */

import java.io.InputStream;
import java.io.IOException;
import java.io.ByteArrayOutputStream;

public class EncodedInputStream extends InputStream{

    private EncodingScheme encoding_scheme;
    private InputStream in_stream;
    private ByteArrayOutputStream in = new ByteArrayOutputStream();

    public EncodedInputStream(InputStream in,EncodingScheme
    scheme){
        in_stream = in;
        encoding_scheme = scheme;
    }

    public int read() throws IOException{
        int nill = -1;
        return nill;
    }

    public byte[] readEncoded() throws IOException{

int read = 0;
        byte decoded[] = null;

while ((read = in_stream.read())!=-1)
        in.write(read);

```

```

        decoded = encoding_scheme.decode(in.toByteArray());

return decoded;
    }

    public void close() throws IOException{
super.close();
in_stream.close();
    }
}

```

#### Listing 2

```

/*
 *
 * EncodedOutputStream
 *
 * This class is used to encode a stream of data
 *
 * @author Mike Jasnowski
 * @version 1.0 , 06/01/2000
 */

import java.io.OutputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;

public class EncodedOutputStream extends OutputStream{

    private OutputStream out_stream;
    private ByteArrayOutputStream out = new ByteArrayOutputStream();
    private EncodingScheme encoding_scheme;

    public EncodedOutputStream(OutputStream out,EncodingScheme
    scheme){
        out_stream = out;
        encoding_scheme = scheme;
    }

    public void write(int b) throws IOException{
        /* Encoding needs to be done here before it's written
        to Outputstream */
        out.write(b);

```

```

    }

    public void write(byte[] b) throws IOException{
        write(b,0,b.length);
    }

    public void write(byte[] b,int offset,int length) throws
    IOException{
        for (int i = 0;i < length;i++)
            write(b[offset + i]);
    }

    public void close() throws IOException{
        super.close();
        out_stream.write(encoding_scheme.encode(out.toByteArray()));
        out_stream.close();
    }
}

```

### Listing 3

```

/*
 *
 *
 * EncodingScheme - The interface class for all Encod-
ingSchemes
 *
 * @author Mike Jasnowski
 * @version 1.0 , 06/01/2000
 */

public interface EncodingScheme{

    /* This method is called by EncodedOutputStream */
    public byte[] encode(byte[] to_encode);

    /* This method is called by EncodedInputStream */
    public byte[] decode(byte[] to_decode);
}

```

### Listing 4

```

/*
 *
 * This class carries the encode/decode logic for the scheme
 *
 * @author Mike Jasnowski
 * @version 1.0 , 06/01/2000
 */

import java.io.ByteArrayOutputStream;

public class Base64EncodingScheme implements EncodingScheme{

    char NVT_ASCII[] =
    {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P',
    'Q','R','S','T','U',
    'V','W','X','Y','Z','a','b','c','d','e','f','g','h','i','j','k',
    'l','m','n','o','p','q','r','s','t','u',
    'v','w','x','y','z','0','1','2','3','4','5','6','7','8','9','+',
    '/'};

    private boolean isPadding = false;

    public Base64EncodingScheme(){

    public byte[] encode(byte[] data){

        char temp[] = new char[4];
        int byte1=0,byte2=0,byte3=0,byte4=0;
        int num_padding = 3;
        byte[] hold_buffer = null;
        String encoded = "";
        int chunk = 0;
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int counter = 0;

        for (int i = 0;i < data.length;i+=3){
            hold_buffer = new byte[3];
            temp = new char[4];

```

```

            num_padding=3;
            if (i < data.length){ hold_buffer[0] = data[i];num_padding-
            ;}
            if (i+1 < data.length){ hold_buffer[1] =
            data[i+1];num_padding--;}
            if (i+2 < data.length){ hold_buffer[2] =
            data[i+2];num_padding--;}

            /* This puts padding in place */
            if ((i+2 > data.length) && (hold_buffer[1] == 0))
            {hold_buffer[1] = 61;isPadding=true;}
            if ((i+3 > data.length) && (hold_buffer[2] == 0))
            {hold_buffer[2] = 61;isPadding=true;}

            /* Encoding Starts Here */
            byte1 = unsigned(hold_buffer[0]) >>> 2;
            byte1 = byte1 & 0xFF;

            temp[0] = NVT_ASCII[byte1]; /* first byte encoded */

            byte1 = unsigned(hold_buffer[0]) << 6;
            byte1 = byte1 & 0xFF;
            byte1 = byte1 >>> 2;
            if (hold_buffer[1] != 0x3D | !isPadding)
                byte2 = unsigned(hold_buffer[1]) >>> 4;
            else
                byte2 = 0;
            byte3 = byte1 | byte2;

            temp[1] = NVT_ASCII[byte3]; /* second byte
            encoded */

            if (hold_buffer[1] != 0x3D | !isPadding){

            byte1 = unsigned(hold_buffer[1]) << 4;
            byte1 = byte1 & 0xFF;
            byte1 = byte1 >>> 2;
            byte2 = unsigned(hold_buffer[2]) >>> 6;
            byte3 = byte1 | byte2;

            temp[2] = NVT_ASCII[byte3]; /* third byte encod-
            ed */

            }

            if (hold_buffer[2] != 0x3D | !isPadding){
            byte1 = unsigned(hold_buffer[2]) << 2;
            byte1 = byte1 & 0xFF;
            byte1 = byte1 >> 2;
            temp[3]= NVT_ASCII[byte1]; /* fourth byte encod-
            ed */

            }

            counter = 0;

            for (int j = 0;j<temp.length;j++){

            if (temp[j] != 0){
                counter++;
                out.write((byte)temp[j]);
            }

            }

            chunk+=4;

            if (chunk == 76){

                out.write(13);
                out.write(10);
                chunk=0;
            }

            }

            /*Write out padding */
            for (int j = 0;j<num_padding;j++)
                out.write(61);

            /* Write final CRLF */
            out.write(13);
            out.write(10);

            return out.toByteArray();

```

```

}

public byte[] decode(byte[] data){

byte decoded[] = new byte[3];
byte hold_buffer[] = new byte[4];
int byte1=0,byte2=0,byte3=0;
int running_length = 0;
ByteArrayOutputStream out = new ByteArrayOutputStream();
ByteArrayOutputStream temp = new ByteArrayOutputStream();
int remove_padding = 0;

/* Strip out CRLF - Chunk markers */
for (int c = 0;c<data.length;c++){
    if (data[c] != 0x0D && data[c] != 0x0A)
        temp.write(data[c]);
}

byte newdata[] = temp.toByteArray();

running_length = newdata.length;

for (int i = 0;i < running_length;i+=4){

    hold_buffer = new byte[4];
    decoded = new byte[3];
    remove_padding = 0;

    hold_buffer[0] = newdata[i];
    hold_buffer[1] = newdata[i+1];
    hold_buffer[2] = newdata[i+2];
    hold_buffer[3] = newdata[i+3];

    if (hold_buffer[2] == 61) remove_padding++;
    if (hold_buffer[3] == 61) remove_padding++;

    byte1 = nvt_lookup((char)hold_buffer[0]) << 2;

    byte2 = nvt_lookup((char)hold_buffer[1]) >> 4;
    byte3 = byte1 | byte2;

    decoded[0] = (byte)byte3; /* First Byte Decoded */

    byte1 = nvt_lookup((char)hold_buffer[1]) << 4;
    byte2 = nvt_lookup((char)hold_buffer[2]) >> 2;
    byte3 = byte1 | byte2;

    decoded[1] = (byte)byte3; /* Second Byte
Decoded */

    byte1 = nvt_lookup((char)hold_buffer[2]) << 6;
    byte2 = nvt_lookup((char)hold_buffer[3]) ;
    byte3 = byte1 | byte2;

    decoded[2] = (byte)byte3; /* Third Byte Decoded */

    out.write(decoded,0,decoded.length-remove_padding);
}

return out.toByteArray();
}

private int nvt_lookup(char c){
for (int i = 0;i<NVT_ASCII.length;i++){
    if (c == NVT_ASCII[i])
        return i;
}
return 0;
}

private int unsigned(int value){
    int newvalue = value << 24;
    return newvalue >>> 24;
}
}
}

```

#### Listing 5

```

/*
*

```

```

* This sample performs the following:
*
* 1) Encodes some sample text using the "EncodedOutput-
Stream"
* 2) Displays the encoded text from a file
* 3) Decodes the sample text by reading in the file using
the "EncodedInputStream"
*
* @author Mike Jasnowski
* @version 1.0 , 06/01/2000
*/

import java.io.*;

public class EncodingSample{

    public static void main(String args[]){
        new EncodingSample(args[0]);
    }

    public EncodingSample(String filename){

        Base64EncodingScheme scheme = new Base64EncodingScheme();

        try {

            /* Write it out and encode */

            EncodedOutputStream encOut = new EncodedOutput-
Stream(new FileOutputStream("encoded.txt"),scheme);
            ByteArrayOutputStream buffer = new ByteArrayOutputStream();
            FileInputStream fin = new FileInputStream(filename);

            int ch = 0;
            while ((ch = fin.read())!=-1)
                buffer.write(ch);

            byte edata[] = buffer.toByteArray();

            encOut.write(edata);

            encOut.close();

            System.out.println(filename + " has been encoded");

            /* Read it back in and decode */

            EncodedInputStream encIn = new EncodedInputStream(new
FileInputStream("encoded.txt"),scheme);
            int r = 0;
            byte ddata[] = encIn.readEncoded();
            encIn.close();

            FileOutputStream fout = new FileOutputStream(filename);

            fout.write(ddata,0,ddata.length);
            fout.close();

            System.out.println(filename + " has been decoded");

        }catch(IOException e){
            System.out.println(e);
        }
    }
}

```

#### Listing 6

```

public class HelloWorld{

    public static void main(String args[]){
        new HelloWorld();
    }

    public HelloWorld(){
        System.out.println("HelloWorld");
    }

}

```



# Implementing Fowler's Analysis Validator Pattern in Java

*Reusable*

*self-validating*

*GUI components*

*increase*

*developer*

*productivity*

WRITTEN BY DR. SARA STOEKLIN AND DR. CLEMENT ALLEN

**B**uilding large systems requires the difficult and time-consuming activities of elicitation and representation of software requirements. During these analysis activities, particular analysis abstractions emerge.

These abstractions, called analysis patterns, represent reusable patterns for subsequent analysis efforts in various domains. As an example, software developers use an analysis abstraction called Person to represent a person from different application domains, such as a student person, employer person or customer person. Martin Fowler, in his book Analysis Patterns, has defined a higher abstraction to represent either a person or an organization labeled the Party pattern.

The most popular patterns – design patterns – deal with those patterns useful in both object-oriented design and programming. Design patterns represent reusable software structures needed for implementation. By contrast, Fowler’s Analysis Patterns represents reusable abstractions needed in the elicitation and representation of the software requirements, hence analysis patterns. Analysis patterns represent conceptual domain structures denoting the model of a business system domain, rather than the design of computer programs. While analysis patterns don’t deal directly with the details of implementation, they do influence how code is designed.

## Why Use Analysis Patterns?

We’ve used patterns in structured programming and they proved to be beneficial. When writing a new program, most programmers copied examples of similar programs (program patterns) and modified them to meet the needs of the new program, rather than writing from scratch and risking simple compilation and logic errors. Structured pattern programs proved beneficial because they saved development and maintenance time needed for new programs and for reviewing existing programs.

When object-oriented analysis, design and programming were introduced, they proclaimed the notable advantage of reuse. These proclamations gave hope to the idea of fast, easy software development using reusable objects or components. Many object-oriented programmers aren’t realizing reductions in their work efforts due to reuse. However, Fowler’s analysis patterns will allow the proclamations to be realized in object-oriented language solutions.

The great advantage of using patterns in object-oriented development is the increase in productivity and therefore a reduction in cost. Analysis patterns leverage the capabilities of the reusability of object-oriented components. These components ease the complexity of problem solutions of new development and lower the maintenance for an application using these components. Application builders who have built systems using analysis patterns have experienced reduced development time, ease of development and low maintenance costs.

## Fowler’s First Pattern – The Party Pattern

The first analysis pattern described by Fowler is the Party pattern. A Party, according to Fowler, is an abstraction to define persons or organizations. He models a Party class with subclasses of Person and Organization. The Party could have different roles. For example, a person could be an employer, employee, doctor or mother. An organization could be a business entity, a shelter or a hospital. Figure 1 depicts the class representation of the Party pattern.

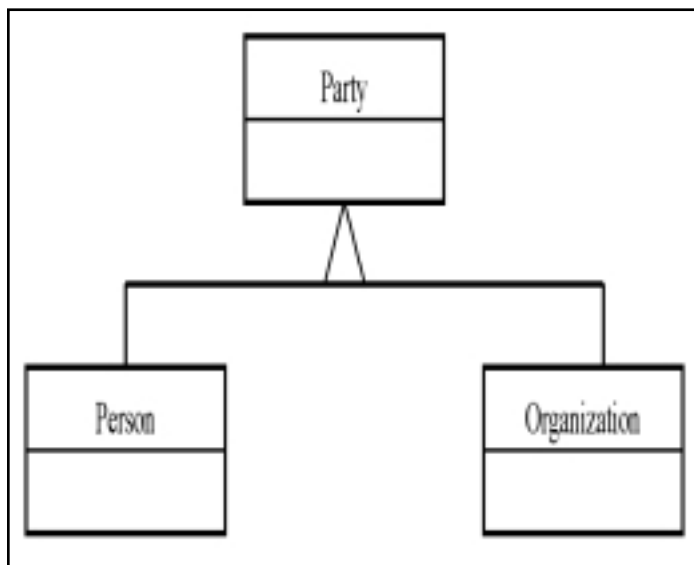


FIGURE 1 Party pattern

## Observation Pattern

Another analysis pattern described by Fowler is the Observation pattern used to model information about the real world, generally representing attributes observed about a party. Observations play an important role in information systems, since these observations are usually stored in databases and rehashed to form statistical analysis of data. Examples of observations are a person’s eye color, hair color, weight or height. Other examples include a test score of Johnny Doe on December 4, 1999, or Jane Doe’s telephone number. The Observation pattern is an abstraction that describes the quantification about a given attribute type, called a *phenomenon*, related to a Party. As an example, a test score type of attribute – or, as labeled by Fowler, a test score phenomenon – has a quantification of 64, which is related to a person, Johnny Doe. For example, Johnny Doe made a score of 64 on a particular test.

Using the Observation pattern, new observations are defined by extending the behavior of the observation and writing new code to define their type. From a position paper by Joseph Yoder, “Patterns for Developing Successful Object-Oriented Frameworks,” we realize that there’s not only an observation in the pattern, but an observation type is needed to describe the subject of the associated observation. Figure 2 depicts the relationship between a party as well as a portion of the Observation pattern, including the observation and observation type. The observation type allows modeling of the observation phenomenon as a type. This pattern allows creation of a phenomenon, such as test scores, without creating a different class for each phenomenon needed within a system. One instance of the party class, such as Johnny Doe, may have multiple instances of observations, and each of those observations is defined as a specific observation type. For example, John Doe has an observation of blue observation type called eye color, as well as an observation of 64 observation-type test scores.

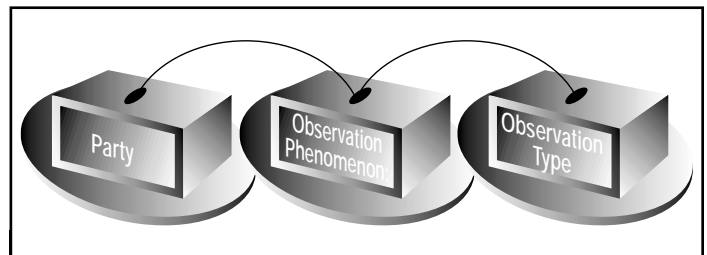


FIGURE 2 Party associated with an observation

One of the key characteristics of observations with information systems is the entry of observation information by users and the potential database storage of that information. This input scenario requires that information gathered from the user must be validated prior to placing the values into database storage. Integers, entered as strings in text fields, require validations by ensuring that first, they’re composed of the characters 0–9 and second, that the value is indeed storable as an integer and the value of the entered data falls within the range defined by a business rule. A salary within an organization, represented as a float, may have a business rule that requires the salary to be more than \$1 and less than \$100,000 per year. In most structured systems, the validation routines necessary would be written into each program that allows entry of the field. In object-oriented systems, one might call a class that contains the method to validate salary. Using Fowler’s analysis patterns, validation for this salary observation is done using the Validator pattern.

## Validator Pattern

The Validator pattern is used in collaboration with the Observation pattern to validate observations. This architecture allows different types of observations to be associated with their relevant applications and businesses. Therefore, the observation types are extended with a validator type that associates the instance of the observation type with a validator strategy.



The Validator pattern is an abstraction that models the procedures for validation of different types of observations using three different validators. The Validator class (see Figure 3) depicts the three validators: discrete, range and null.

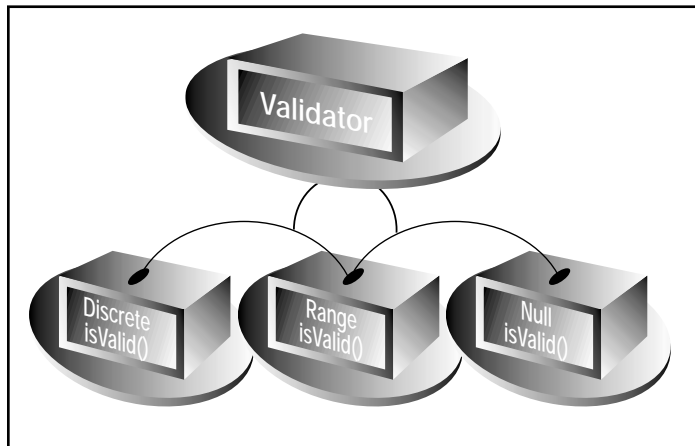


FIGURE 3 Validator pattern

When using the pattern, all observations are validated. Therefore, a need exists for a null validator in those fields that require no validation. The null validator is implemented using the Null Object pattern.

The Discrete validator authenticates values such as the items found in a typical code value table. A common example of discrete validation is eye color, in which valid values include a set with blue, green, hazel, brown and black members. Another common example is a two-member set representing sex as the discrete values of male or female. A user entering data in a field with a discrete validator would view a table of valid values and select one. This is possible if the number of values in the code table isn't too large. Large code tables often require the user to key in a few characters before a table of valid values is displayed. Observations in which the valid values are extremely large may not allow the code values to be displayed, but instead require the user to key in the entire value before validation. This is the scenario addressed by the discrete validation described in this paper.

The Range validator is used to validate an observation expected to be within a valid range of values according to a particular business rule. The salary, which must fall within the range of \$1 to \$100,000 per year, is an example of such an observation. A range validation routine requires a minimum and maximum value to define the valid range. The Validator Pattern classes are shown in Figure 3, each with a method `isValid()`.

Not only does the validator abstraction allow users the freedom to reuse the validator component in many different domains, it also allows users to modify their validation requirements dynamically at runtime.

When code tables were introduced in the 1980s, the user gained the ability to add, modify and delete values from the code table without causing a program maintenance activity. Using the Range validator allows the user the same freedom – to change the minimum (min) and maximum (max) values for range-validated fields without asking a programmer to perform a maintenance activity. The Validator pattern uses the data dictionary table to obtain the min and max values for a range-validated field. The users have the freedom to change these data dictionary min and max values. Keeping this close collaboration between the data dictionary and the observation entry allows user control of validation rules and gives dynamic validation of observations at runtime. It demands a data dictionary that acts more as a usable software engineering tool rather than a documentation table.

Each observation, defined in the data dictionary, is linked to an observation type defining the validator needed for that observation. The observation “shoe size” is defined in the data dictionary with a minimum of four and maximum of 20. The shoe size observation type is called Shoosize and the validator defined by that type is RangeValidator.

## Building Reusable GUI Components Using the Validator Pattern

Ease of development and maintenance is the goal in using analysis patterns. The Validator pattern allows easy building and maintenance of observations in an application domain. It also allows development of reusable graphical user interface (GUI) components. To show the technique of using the Validator pattern to build these reusable GUI components, this article concentrates on the Range Validator.

The data dictionary used to store the min and max necessary for range validation could also contain information regarding the needed GUI component for the observation. Items that prove useful in dynamically building GUI components for observations include the field length to dictate the size of the needed textfield, observation type (such as eye color) to attach a validator to the observation type, the default label needed on the GUI component, a standard default message for invalid data and the name of the validator (in this case range validator).

To build a reusable GUI panel component responsible for validating the users observation inputs, we define a panel named *Observation Panel*. The panel contains two components: the textfield of the length defined in the data dictionary and the label of the field with the name taken from the data dictionary, if desired. The panel contains a field that defines the data dictionary item contained in a vector named a data dictionary (DD) vector. The structure needed for this reusable component is shown in Figure 4.

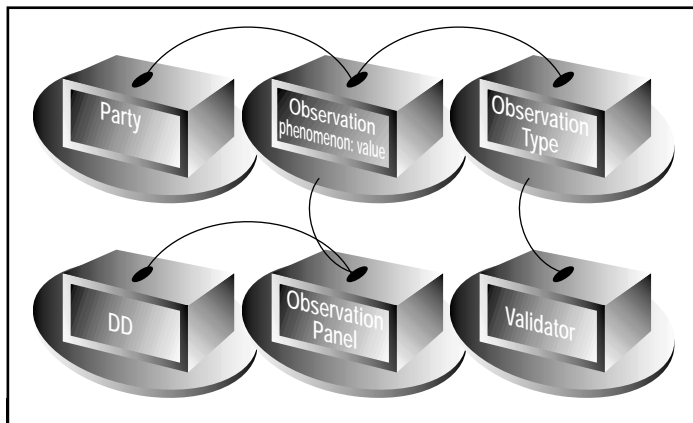


FIGURE 4 Reusable GUI component and the Validator pattern

To understand fully how these GUI components would be used with the Observation and Validator patterns, we define a typical application programmer building a GUI under the present technology without analysis patterns, then with them.

A typical application program building a GUI screen follows a development scenario similar to the one below:

1. Create the container for the screen
2. Repeat for each component needed on the screen:
  - a. Drag a reusable component (such as a label) on the screen
  - b. Change the properties for the component (size, name, position on the screen, etc.)
  - c. Write the validation routine for the component entry
  - d. Write the error-routine display routine
  - e. Test properties, location, validation routine, error routine iteratively
3. Scenario complete

This scenario is time consuming and yields components that may have inconsistent labels, error messages and sizes across multiple screens. Even if programmers build components for each field entered, the validation and error routine must be written and tested for each component.

Using the analysis patterns described and a reusable bean – Observation Panel – component modeled in Figure 4, the scenario is as follows:

1. Create the container for the screen
2. Build the reusable Observation Panel Bean (once)
3. Test the Observation Panel Bean
4. Repeat for each needed component
  - a. Drag the Observation Panel Bean on the screen
  - b. Change the Data Dictionary Property Name for this field
  - c. Test property DD name is correct
5. Scenario complete

This scenario allows an expert bean builder to build and test the Observation Panel Bean (OPB) before the application programmer begins building the GUI screens using the OPB. The application developer reuses the Panel Bean component for each GUI component on each screen, which reduces the design, programming and testing effort significantly. Figure 5 shows the application developer changing the name of the data dictionary data element with IBM's VisualAge for Java.

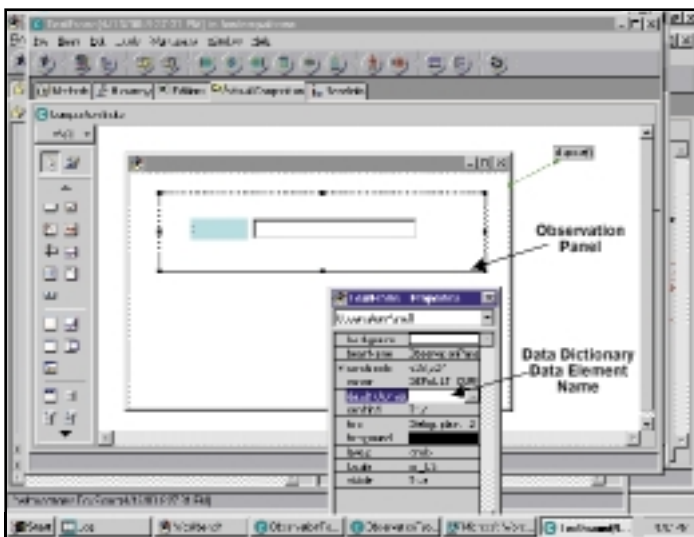


FIGURE 5 Application builder changing the Observation Panel Bean property

## Runtime Scenarios

At runtime, the OPB displays itself and its components making itself capable of accepting and validating the entered data. When the OPB is displayed the following scenario happens.

1. The new GUI screen frame creates an instance of the OPB.
2. The OPB displays the label and textfield using information obtained from the data dictionary regarding size and label.
3. The observation and observation-type instance are created. The specific validator is created using the Reflection pattern, and the needed data such as the min and max values are loaded.

This scenario is the result of executing the code shown in Listing 1, which includes the classes for the Observation, ObservationType, ObservationPanel and DataDictionaryRecord. Not all of the code is shown – only the parts of interest are displayed. The remainder of the code for the Validator and RangeValidator classes is shown in Listing 2.

After the OPB is displayed with the label and textfield, the data is entered by the user and validated with the Validator pattern as in the following scenario.

1. An action is performed indicating that the field is entered and ready for validation. This may be a submit button on the GUI screen frame, a carriage return on the textfield or other actions.
2. The OPB requests the text from the textfield using `getText()`.
3. The panel asks the observation to validate itself. This observation field will later hold the valid data in the needed type. The observation, expecting to receive a notification of valid data, asks the observation

type to validate the string entered in the textfield using the validator defined according to the data dictionary information regarding the data type and valid ranges.

4. The observation type sends the data dictionary member to the validator for the actual validation of the string data entered.
5. If the data isn't valid, an error routine is executed. In our case an invalid message obtained from the data dictionary is displayed on the panel as a label. The message is erased if the user enters more data.
6. If the data is valid, the panel requests the observation panel to convert the valid data to the value needed.

Remember, the only action performed by the application builder was to drag the OPB on the GUI screen frame and modify the bean property to be the specific data dictionary element. The remainder of the actions, from the two scenarios described above, happened as a result of methods and procedures contained in the reusable OPB, the data dictionary and the Validator analysis pattern. This same OPB and Validator can be reused for any data dictionary field on any GUI screen.

## Conclusion

Analysis and design patterns are useful in solving problems when defining requirements for a system, irrespective of the domain. They're helpful in guiding the development of reusable components. Fowler's Observation patterns are examples of analysis patterns that have many variations and extensions. The OPB reusable component and the Validator pattern (see Figure 4) use the Observation pattern with an object-oriented language and its concepts to promote the reusability of the Validator and the OPB. The discrete validator would be implemented similarly with the name of the valid code table values contained in the data dictionary. Use of the Validator pattern certainly changes the face of GUI development.

While Fowler's patterns are powerful abstractions in information systems, implementing them in different languages often leads to many nuances and problems that require further understanding of the patterns and the problems solved by them.

In the future, validation functions could be linked to observations, composite validation rules added to composite observations, and other GUI components and debugging tools developed. These types of abstractions and the resulting codes do take time, but the payoff in productivity is certainly large enough to justify the cost. 🍌

## Resources

1. Gamma, E., Helms, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.
2. Coplien, J.O., and Schmidt, D.C. (1995). *Pattern Languages of Program Design*. Reading, MA: Addison-Wesley.
3. Fowler, M. (1997). *Analysis Patterns: Reusable Object Models*. Reading, MA: Addison-Wesley.
4. Grand, M. (1998). *Patterns in Java*. Vol. 1. New York, NY: John Wiley & Sons, Inc.
5. Grand, M. (1999). *Patterns in Java*. Vol. 2. New York, NY: John Wiley & Sons, Inc.
6. Yoder, J. "Patterns for Developing Successful Object-Oriented Frameworks." Workshop Position Paper, OOPSLA, 1997.

### AUTHOR BIOS

Dr. Sara Stoecklin serves as graduate director in the Department of Computer and Information Sciences at Florida A&M University where she teaches software engineering classes in industry and academia. Dr. Stoecklin holds an MS from East Tennessee State University and a PhD from Florida State University.

Dr. Clement Allen teaches advanced Java at Florida A&M University. He holds an MS from Howard University and a PhD from the University of Alabama at Birmingham.

stoeckli@cis.famu.edu allen@cis.famu.edu

## Listing 1

```

//Observation Class with a few pertinent methods
public class Observation {
    private java.util.Date recordedDate;
    private java.util.Date observedDate;
    private int duration;
    private ObservationType type;
    private String observationTypeName;
    public boolean isValid (String obsValue) {
        return getType().isValid( obsValue );
    } end isValid
} // end Observation Class

// Observation Type with pertinent methods
public class ObservationType {
    private String phenomenon;
    private Validator dataElementValidator = null;
    private String phenomenonType;
    public ObservationType(String ddElementName) {
        DataDictionaryRecord drecord = DDManager.getMember(dElementName );
        setPhenomenon( ddElementName );
        setPhenomenonType( drecord.getDataElementType());
        try {
            // This uses the validator name stored in the data dictionary to //
            build an instance of the needed validator (either a Range // or Discrete Validator) using the Reflection pattern.
            Class validatorClass =
                Class.forName("fowlerspatterns."+drecord.getValidatorName());
            setDataElementValidator ( (Validator)
                validatorClass.newInstance());
            getDataElementValidator().setDDRecord( drecord);
        } catch(Exception e) { e.printStackTrace(); }
    } // end ObservationType constructor
    public boolean isValid(String obsValue) {
        return getDataElementValidator().isValid(obsValue );
    } // end isValid
} // end Observation Type

// Observation Panel

public class ObservationPanel extends java.awt.Panel
implements java.awt.event.ActionListener {
    protected transient
        java.beans.PropertyChangeSupport propertyChange;
    private String fieldDataDictionaryElementName = new
String();
    private java.awt.Label ivjObservationLabel = null;
    private java.awt.TextField
ivjObservationTextField = null;
    private Observation dataElementObservation = null;
    private java.awt.Label ivjErrorLabel = null;
    private boolean observationValid = true;
    private String panelObservation;
    private String panelObservationText;
    private String errorText;

    public ObservationPanel() {
        super(); initialize();
    } // end ObservationPanel constructor

    public static void main(java.lang.String[] args) {
        try { java.awt.Frame frame;
            try {
                Class aFrameClass =
                    Class.forName("com.ibm.uvm.abt.edit.TestFrame");
                frame = (java.awt.Frame)aFrameClass.newInstance();
            } catch (java.lang.Throwable ivjExc)
                {frame = new java.awt.Frame();}
            ObservationPanel aObservationPanel;
            aObservationPanel = new ObservationPanel();
            frame.add("Center", aObservationPanel);
            frame.setSize(aObservationPanel.getSize());
            frame.setVisible(true);
        } catch (Throwable exception) {
            System.err.println("Exception occurred in main
Panel");
            exception.printStackTrace(System.out);
        }
    }
}

} // end main

public void actionPerformed(java.awt.event.ActionEvent e)
{
    if ((e.getSource() == getObservationTextField()) ) {
        validateThisObservation(e);
    }
} // end actionPerformed

private void validateThisObservation
    (java.awt.event.ActionEvent arg1) {
    try {this.validateObservation();}
    catch (java.lang.Throwable ivjExc)
        {handleException(ivjExc);}
    } // end validateThisObservation

private java.awt.Label getErrorLabel() {
    if (ivjErrorLabel == null) {
        try {
            ivjErrorLabel = new java.awt.Label();
            ivjErrorLabel.setName("ErrorLabel");
            ivjErrorLabel.setText("");
            ivjErrorLabel.setBounds(45, 61, 269, 23);
        } catch (java.lang.Throwable ivjExc)
            {handleException(ivjExc);}
    }; // end if
    return ivjErrorLabel;
} // end getErrorLabel

private java.awt.Label getObservationLabel() {
    if (ivjObservationLabel == null) {
        try {
            ivjObservationLabel = new java.awt.Label();
            ivjObservationLabel.setName("ObservationLabel");
            ivjObservationLabel.setText("
");
            ivjObservationLabel.setBackground(java.awt.Color.cyan);
            ivjObservationLabel.setBounds(36, 29, 66, 23);
        } catch (java.lang.Throwable ivjExc)
            {handleException(ivjExc);}
    }; // end if
    return ivjObservationLabel;
} // end getObservationLabel

private java.awt.TextField getObservationTextField() {
    if (ivjObservationTextField == null) {
        try {
            ivjObservationTextField = new
java.awt.TextField();
            ivjObservationTextField.setName
                ("ObservationTextField");
            ivjObservationTextField.setBounds(108, 29, 188,
23);
        } catch (java.lang.Throwable ivjExc)
            {handleException(ivjExc);}
    }; // end if
    return ivjObservationTextField;
} // end getObservationTextField

protected java.beans.PropertyChangeSupport getProperty-
Change() {
    if (propertyChange == null) {
        propertyChange = new
            java.beans.PropertyChangeSupport(this);
    }; // end if
    return propertyChange;
} // end PropertyChangeSupport

private void initialize() {
    setName("ObservationPanel"); setLayout(null); setSize(375,
88);
    add(getObservationTextField(),
getObservationTextField().getName());
    add(getObservationLabel(), getObservationLabel().get-
Name());
    add(getErrorLabel(), getErrorLabel().getName());
    initConnections();
    DataDictionaryRecord drecord =
        DDManager.getMember(getDataDictionaryElementName());
    getObservationLabel().setText( drecord.getLabelName() );
    setErrorText( drecord.getInvalidObservationLabel() );
}

```

```

// The label size, textfield size, and panel size should
// be adjusted using the length of the label from the
// data dictionary.
    setDataElementObservation( new Observation(
getDataDictionaryElementName() ) );
} // end initialize

public void setDataDictionaryElementName
    (String dataDictionaryElementName) {
String oldValue = fieldDataDictionaryElementName;
fieldDataDictionaryElementName = dataDictionaryElement-
Name;
firePropertyChange
    ("dataDictionaryElementName", oldValue,
    dataDictionaryElementName);
} // end setDataDictionaryElementName

public void validateObservation() {
/* Perform the validateObservation method. */
getErrorLabel().setText("");
setObservationValid(true);
setPanelObservationText( getObservationTextField().get-
Text() );
setObservationValid
    (getDataElementObservation().isValid
    (getPanelObservationText()));
if( !getObservationValid() ) {
    getErrorLabel().setText(getErrorText());
} // end if
} // end validateObservation
}end Observation Panel class

// DataDictionary Record class an item in DD Vector
public class DataDictionaryRecord {
    private String dataElementName;
    private Validator dataElementValidator;
    private String min;
    private String max;
    private String tableName;
    private java.util.Vector tableVector;
    private String validatorName;
    private int length;
    private String labelName = new String();
    private String dataElementType;
    private String invalidObservationLabel;
} // end DDRRecord class

```

#### Listing 2

```

// validator class

public class Validator {
    protected DataDictionaryRecord dDRecord;
    public boolean isValid(String obs ) { return false; } //
end isValid
} // end validator class

// RangeValidator Class
public class RangeValidator extends Validator {
    public boolean isValid(String obs ) {
    int obsInt;
    if(getDDRRecord().getDataElementType().equals("integer")) {
        // validate that obs is an integer
        // ,Ã¶.
        try {obsInt = Integer.parseInt( obs );}
        catch(NumberFormatException e) {return false;}
        // validate that obs meets the rules of min and max
        int maxInt = Integer.parseInt(getDDRRecord().getMax());
        int minInt = Integer.parseInt(getDDRRecord().getMin());
        return ((minInt <= obsInt )&&(maxInt>=obsInt));
    } // end if
    return false;
    } // end isValid
} // end rangevalidator

```



# Programming with Databases Using Java

JDBC — a must-have tool for your programming toolbox



WRITTEN BY  
ROBERT J. BRUNNER

A primary benefit of using the Java programming language is the wide range of packages available for simplifying a variety of programming tasks. One of these tasks is to provide a persistent storage for Java programs. Actually, this can be accomplished using several different techniques, including Serialization, SQLJ, JDBC and eventually JDO.

JDBC (Java Database Connectivity) is routinely covered by many different authors in varying detail; however, the fundamental basics of using it to connect a Java application to a database is often casually discussed or, worse, ignored completely. With the spread of CASE tools for simplifying Java-database interactions (using, perhaps, EJB), many users have become insulated from the actual details of what is going on “under the hood.”

This is a critical point in really understanding what your application is doing, especially when you consider that you might want to connect to a database from a variety of Java applications, including applets, servlets, JavaServer Pages, Swing applications and Enterprise JavaBeans. This article provides a gentle introduction to the fundamentals so you can eventually tackle more challenging projects with greater confidence.

Before delving any further into the details of JDBC, let’s make sure we all understand what JDBC does and doesn’t do. Primarily, JDBC encapsulates the specifics of connecting to a database, sending SQL statements over the established connection, and processing the results from executed SQL statements. In addition, JDBC provides access to specific metadata (data that describes data) for both the database you’re working with and the metadata for the result of the SQL query.

The phrase “JDBC encapsulates...” implies two things. One, that the JDBC API is composed primarily of interfaces. This is important, because it indicates that someone else must supply the implementation, better known as the JDBC driver (which we’ll explore later in this arti-

cle). The second point is that the JDBC API hides the differences between different databases, which allows you to migrate your application quickly between different database vendors. If you’re not careful, of course, there’s enough flexibility within the JDBC API to quickly tie yourself to a database vendor through the use of specific functionality. However, by following good object-oriented program-

ming practices (also known as best practices), any potential liabilities can be minimized.

## JDBC Architecture

The basic JDBC architecture (which roughly translates to the JDBC 1.2 API) is remarkably simple, given all that it’s designed to accomplish. Essentially,

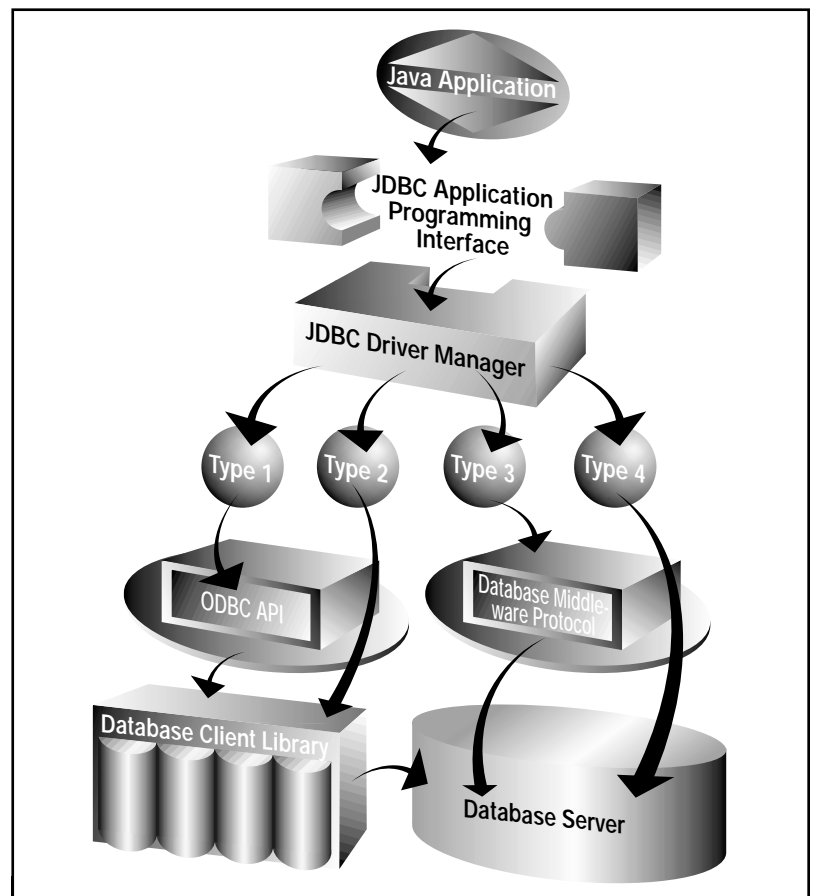


FIGURE 1 Outline of the JDBC architecture

vendors who implement the JDBC API must shoulder the burden of providing the detailed implementation, but that's a good thing for you as a consumer, of course, as it spurs competition. JDBC was designed to provide a "call-level" SQL interface for Java applications, which means that a user must have a working knowledge of SQL to actually use JDBC.

While this might seem counterproductive, it actually works to your advantage as it simplifies the development of tools, either custom developed or commercially produced, that can automate the complexities of database programming in Java. As an example, tools for creating Java classes from database schema (and vice versa), known as schema mapping, can be purchased from several major vendors.

The architecture that results from using JDBC to handle database interactions with a Java application is outlined in Figure 1. It may appear a bit confusing at first glance, but that's primarily a result of the flexibility inherent in the JDBC model as well as in the four types of JDBC drivers (discussed later). Any Java application (e.g., an applet, servlet or JSP) that uses the JDBC API to communicate with a database system will utilize an appropriate JDBC driver to handle all interactions with the database.

Interestingly enough, the developer doesn't directly control which driver is used; this is handled by the Driver Manager (note that JDBC 2.0 introduces an alternative, the Data Source Object, which uses a slightly different architecture). This design allows a single application to interact with different databases using different JDBC drivers. In fact, the Driver Manager will select the appropriate driver object to use from the pool of drivers that it knows about based on the database contact information (the JDBC URL) supplied by the developer.

The rest of the information flow through the JDBC architecture depends on the specific type of JDBC driver used. The drivers come in four different flavors, cleverly named by Sun Microsystems as Type 1, Type 2, Type 3 and Type 4.

As previously indicated, the Type 1 driver is the Sun-provided JDBC-ODBC bridge, which translates the JDBC API into the Microsoft-developed ODBC API. Using this bridge, a Java application can interact with any ODBC-aware application, including the entire Microsoft Office suite. However, this simple flexibility introduces an extra layer of indirection, which, among other things, can

severely limit performance: not only does the JDBC have to be transformed into ODBC, but the ODBC must be transformed into the application's own API, typically involving the equivalent of a database client library API.

On the other hand, the Type 2 JDBC drivers (also known as partial Java drivers) translate the JDBC API directly into the database client library API, reducing the level of complexity. This requires that the client (the machine running the Java application) must be running a subset of the (possible binary) code

“  
The first step in designing a JDBC application, of course, is to select the database to use, never an easy task, especially in a 'team' environment  
”

from the database-specific client API. This probably requires an extra license per client, as well as potential code distribution nightmares (for example, does your database vendor support all of your potential client platforms? How will you keep all of your clients synchronized with the latest version of the database libraries?).

Type 3 drivers are written in pure Java, and as a result overcome many of the limitations of the previous two types of drivers. This class of JDBC drivers communicates in a middleware protocol that provides an extra layer of flexibility. The middleware component can interact with many different database systems as this type of driver provides a server (i.e., the middleware component) that handles the specific database communications, allowing different applications to use the same JDBC driver to communicate with different database systems.

The Type 4 driver, also a pure Java driver, communicates directly with the database server in a database-specific

protocol. This architecture, while conceptually the cleanest, isn't optimized for any specific hardware and operating system platforms (which can be done for Type 3 drivers). Instead, it's generally optimized for a specific database server, and as a result can provide significant performance benefits if you don't mind being tied to a specific database system (e.g., Oracle).

## Choosing the Storage

The first step in designing a JDBC application, of course, is to select the database to use, never an easy task, especially in a "team" environment. In fact, if you're working on a corporate LAN, your ability to pick and choose a database system may be significantly limited due to security restrictions. (This results from the fact that most database systems will need to run as a daemon that will likely provide a network accessible port – a major security concern if not done properly.) On the other hand, a corporate LAN may already have both a database system and a JDBC driver available, which can easily be used for evaluation on new projects.

Assuming free reign on picking a database, three different classes of database systems can be used to develop JDBC applications. The first class is simple databases, of which Microsoft Access is the best example. These are generally so prolific in their distribution that you can usually find one readily available. In fact, using the JDBC-ODBC bridge driver provided by Sun with the JDBC API, you can treat any data source that has an ODBC interface as potential persistent storage. Although these systems can be useful either for learning how to program with JDBC or for very simple Java database applications, they're not recommended for Internet-based applications (e.g., e-commerce) as they're not designed for large traffic volume or distributed transactions.

The second class is lightweight database systems, which generally, at a minimum, provide a basic persistent storage implementation but without a significant amount of the bells and whistles found in the next class of systems. Many of the databases that fall into this class are either open source or very reasonably priced. For example, mSQL is a moderately powerful relational database system that is free for noncommercial applications. Another product that falls in this category is MySQL, which has rather liberal licensing policies, including a GPL for specific versions.

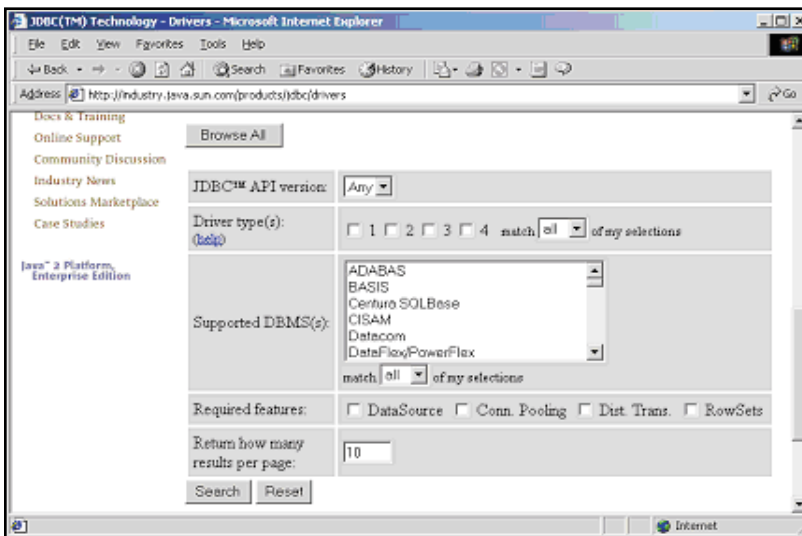


FIGURE 2 JavaSoft driver query Web page

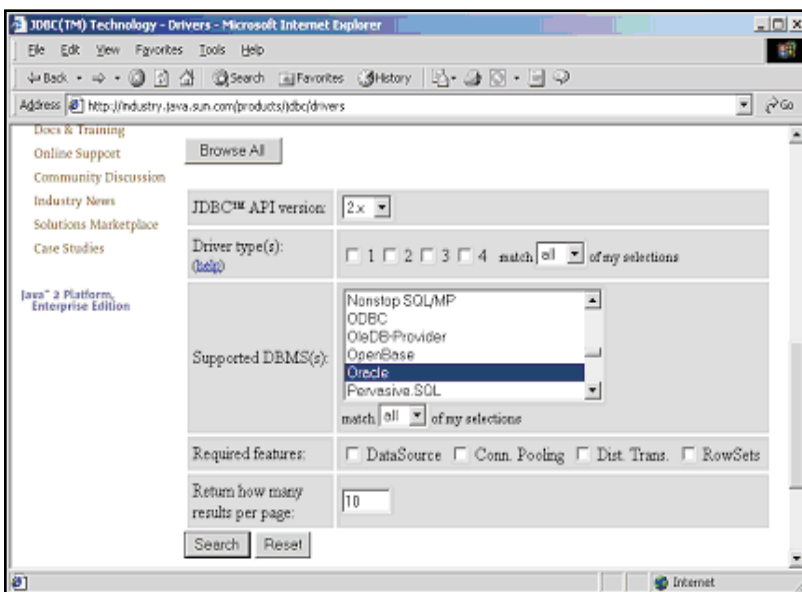


FIGURE 3 JavaSoft driver query Web page as completed to find a JDBC 2.x Oracle-compliant driver

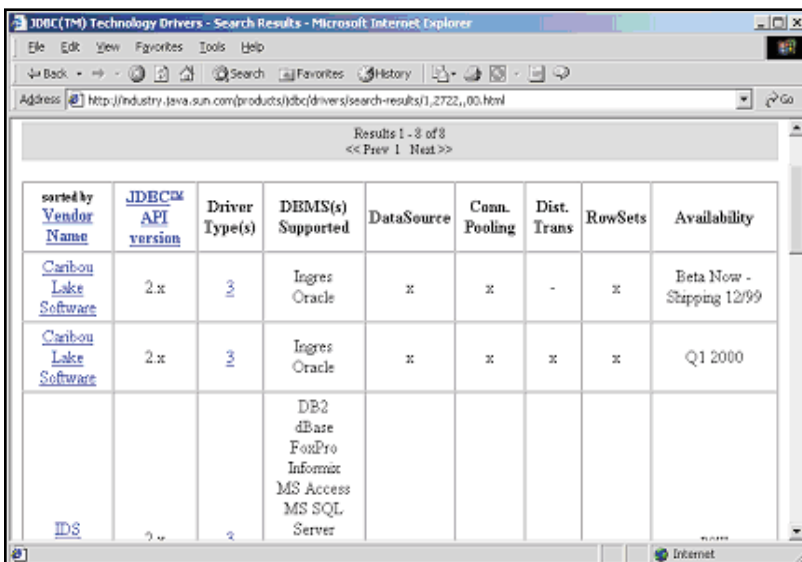


FIGURE 4 Result of the driver query shown in Figure 3

The final class of database systems includes the household names (well, maybe only those households that are occupied by programming enthusiasts or Internet investors) of the database industry, including Oracle, Sybase, SQL Server and Informix. These systems are powerful, full-featured, network-ready software systems that are “ready for prime time” straight out of the box. Of course, the old adage that you get what you pay for certainly applies here, as these systems can take a major bite out of your wallet. However, if you’re developing an e-commerce site, these systems are generally preferred due to their high performance standards, as well as their ability to scale along with your application. Finally, these systems often provide value-added enhancements such as the ability to serve as EJB containers and SQLJ interfaces, and include a well-stocked toolbox. In general, full-featured versions of these database systems are available for evaluation.

### Finding the Right Driver

Once a database has been selected, the last step before jumping into writing code is to choose an appropriate JDBC driver. Fortunately, this last step has been simplified by the Sun JDBC team. A Web page at the JavaSoft Web site (see Figure 2; <http://industry.java.sun.com/products/jdbc/drivers>) provides a simple form that you can use to find all drivers that have been registered with Sun. On this form you enter specific information regarding your project’s requirements – for instance, a specific database system or version of JDBC. As a demonstration, Figure 3 shows the form filled out to find all drivers that support the JDBC 2.x API and connect to an Oracle database. As of late spring 2000, the driver query returned eight separate drivers. As can be seen in Figure 4, the query returns not only the name of the driver, but also its type, the JDBC version it supports, the database systems it supports as well as any supported JDBC standard extensions, and the date of availability.

### Conclusion

This article will hopefully prepare newcomers for the more specific JDBC example articles and discussions commonly available. Programming with databases using Java is a rewarding and challenging task, and for those who enjoy working on cutting-edge technology, a must-have tool for their programming toolbox. ☺

[rjbrunner@yahoo.com](mailto:rjbrunner@yahoo.com)

**AUTHOR BIO**  
Robert Brunner is a member of the research staff at the California Institute of Technology, where he focuses on very large (multiterabyte) databases, particularly on KDD (Knowledge Discovery in Databases) and advanced indexing techniques. He has used Java and databases for more than three years, and has been the Java database instructor for the Java Programming Certificate at California State University Pomona for the past two years.

# JAVA UNPLUGGED

WRITTEN BY KRISTIAN CIBULSKIS

**T**he Web is moving to wireless and Java is making it happen! How is a wireless environment different from the Web? What languages are used for wireless devices and what features do they have? Most important, what role does the Java 2 Enterprise Edition (J2EE) play in a wireless architecture?

## TRANSLATING JAVA TO THE WIRELESS PARADIGM

### Wireless vs Web

There are several key differences between clients in a Web world and those in a wireless world. For example, in the wireless world network connections have a lower bandwidth and a higher latency than a typical Internet connection. Also, wireless connections are unstable and unpredictable since they're built on a wireless network - a fact you're likely familiar with from your cell phone service! The Wireless Access Protocol (WAP) was created to address network computing in this restricted environment. It defines two key aspects of wireless communication: an end-to-end application protocol and an application environment.

As shown in Figure 1, the WAP's end-to-end application protocol is built on three major components: an application server, a WAP gateway and a wireless device. The WAP gateway is responsible for routing and translating WAP data on the wireless network to and from HTTP data on a TCP/IP-based network such as the Internet. Some WAP gateways even convert HTML to languages that WAP-enabled devices can understand in a process called *transcoding*. The end-to-end aspect of the WAP protocol consists of four layers carried by various bearers or providers. Figure 2 compares the WAP protocol stack with a typical Internet one.



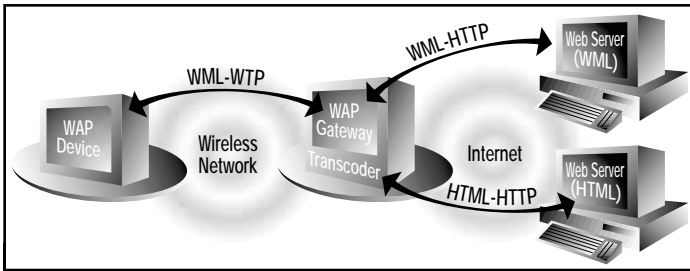


FIGURE 1 Wireless architecture

INTERNET	WAP
HTML JavaScript	Wireless Application Environment (WAE)
HTTP	Wireless Session Protocol (WSP) Wireless Transaction Protocol (WTP)
TLS-SSL	Wireless Transport Layer Security (WTLS)
TCP/IP UDP/IP	Wireless Datagram Protocol (WDP) Bearers (SMS, USSD, CDMA, CDPD, etc.)

FIGURE 2 WAP and Internet protocol stacks

In addition to the end-to-end application protocol, the WAP specification also defines the application environment. This environment consists of a WML browser for rendering content and a script interpreter for executing applications on the user device. The Wireless Markup Language (WML) defines the content to be rendered by the browser using WMLScript as the scripting language. This is analogous to the Internet environment in which HTML is rendered by the browser and JavaScript is the scripting language.

The application environment was included in the WAP specification to ensure that the myriad of WAP-enabled devices can execute the same applications. No doubt you're familiar with browser compatibility problems on the Internet. Imagine if every cell phone manufacturer created their own proprietary browser – there would be hundreds maybe thousands of different browsers. What a mess!

## WML and WMLScript

WML is similar to HTML in its structure and elements. However, two major distinctions are worth mentioning. The first is that WML is a subset of XML and all the rules regarding standard XML apply. This means that the entire WML document must be well formed and adhere to a DTD that describes all WML documents. The second is that WML provides extremely limited control over presentation. If you thought HTML was restrictive, wait until you start playing with WML. These severe restrictions allow for flexibility on the rendering device. Only the lowest common denominator can be assumed across all client devices. It's not just missing attributes in WML tags. Even the rendering of individual tags can appear radically different on different phones. The model for WML rendering, however, is quite similar to Java's approach. Each platform has the freedom to determine how to render a particular widget, as long as it achieves the intended functionality in a manner consistent for that platform. Although the language definition and functionality is rigid to ensure compatibility, the implementation is loosely defined to enable support for a wide variety of devices.

WMLScript is analogous to JavaScript as it's executed on the client browser. WMLScript provides a basic set of libraries to perform string manipulation, math functions, URL facilities and interaction with the WML browser. Additional libraries are being developed that allow the WMLScript programmer to access telephony aspects of the device, such as dialing a phone number or accessing the phonebook. WMLScript's purpose is to provide a richer user environment over static WML.

The easiest way to begin learning both WML and WMLScript is to work with an example. Listing 1 is an extremely simple business model: purchasing your favorite java-based product, such as a cup of coffee or a latte. A user can select the items for purchase, calculate the total and check out. The basic application flow is shown in Figure 3. The code was developed and simulated under Nokia's WAP SDK (see "References" for links).

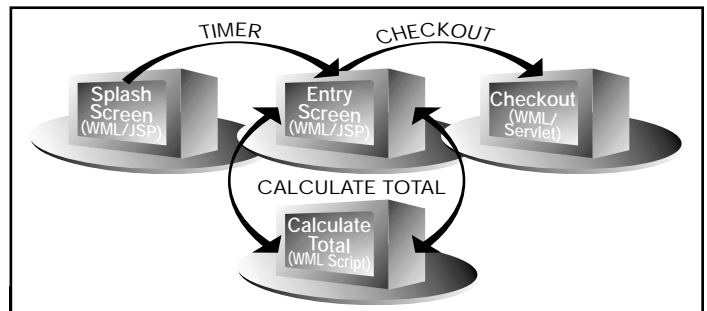


FIGURE 3 Application flow

Let's look at the code for the main WML page in Listing 1.

- **Lines 1-2:** Standard XML syntax that indicates this document is XML and conforms to the DTD specified at [www.wapforum.org/DTD/wml\\_1.1.xml](http://www.wapforum.org/DTD/wml_1.1.xml).
- **Line 3:** The root of a WML document is the <wml> element, similar to the <html> element of an HTML document. Since it's an XML document, it must be well formed. Without the <wml> element and the corresponding </wml> element at line 39, the document won't be processed.
- **Line 4:** The basic working unit of a WML document is a card. A WML document, or deck, can be composed of one or more cards. Each card is completely self-contained and represents an individual interaction with the user. Due to the high latency inherent in wireless connections, a collection of related cards is transmitted in one deck to improve overall performance for the end user. Each card must have a unique identifier specified by the ID attribute. A title for the card can be specified as well. The title is usually displayed across the top of the screen, but that can change due to the creative liberty the browser takes during rendering. The newcontext attribute indicates that each time this card is entered, any variable or state should be reset. Since this first card is a splash screen, it's a good place to reset any variables that may be lingering in the context of the browser. The final attribute is set in line 4 in the ontimer attribute. The value for the ontimer attribute is a URL, which can be either a complete location, such as <http://java.sun.com>, or simply a bookmark that references another card in this deck, such as #main. In this case, when a timer expires within this card, the user will be directed to the card identified as "main."
- **Line 5:** This single line sets the timer referred to in line 4. The value for this tag indicates a length of time in tenths of a second until the timer expires. Once the timer expires, the ontimer event is triggered.
- **Lines 6-13:** This looks like standard HTML. The display tags of WML are extremely similar to HTML, although they're severely restricted when it comes to adjusting the display of data. For example, there's no tag to set the vertical alignment of the columns within a table. However, if you're familiar with HTML, you can probably jump right into writing WML display logic.

Although HTML and WML provide functionality for displaying images, the WML specification only allows for WBMP image types. Many freeware converters and plug-ins can convert existing image files into WBMP format. However, since most phones are monochromatic, images that are converted from a high-resolution color source may need some tweaking in order to display nicely.

• **Line 15:** This is the beginning tag for the main card in our application.

• **Lines 17-18:** Here are our first input elements. Unlike HTML, there's no need for a `<form>` tag to enclose your input fields. Later on we'll see how these elements make it into the next request. For now, these input elements set the value of local variables within the browser context. The name of the variable is indicated by the name attribute in the input tag. These variables can be displayed elsewhere on the page with the syntax `$(variablename)`. To display a single dollar sign, a double dollar sign (`$$`) is used.

It's worth mentioning the format attribute for an input tag. This attribute indicates to the rendering device what sort of character patterns are acceptable as input. For example, the format mask in line 17 indicates that an arbitrary number of alphanumeric characters may be entered, whereas the format mask in line 18 indicates that only four characters may be entered. Format masks may also contain characters to be inserted into the input field. These characters are indicated by prefixing the character with a `\` in the format mask. For example, the format mask `"NNNNN\~NNNN"` could be used to format a zip code entry field by supplying a dash between the first five numbers and the last four numbers during entry.

The final attribute used for input elements is the title element. It's often used when entry for an input field actually takes place on a special edit screen. The WML browser can display this title on that screen for identification purposes.

• **Lines 19-25:** The `<select>` element, also similar to HTML, provides a selection list that enables multiple selections as determined by the multiple attribute. In this example we want our clients to be able to purchase more than one item at a time, so the multiple attribute is set to true. Enclosed within the select element are the various choices or options to be displayed. The text within the element is displayed to the user, while the value attribute indicates the value assigned to the variable that's defined by the name attribute in the select tag. For a multiple selection element the values will all be placed into this variable with semicolon delimiters.

• **Line 26:** This line displays the contents of the totalPurchase variable. Since we haven't defined it yet, the rendering agent will display a blank value. We'll update this variable using WMLScript in just a bit.

• **Lines 27-29:** The `<do>` tag is one of the few action tags available in WML. The type attribute indicates what sort of action is required. This value determines which physical button should be linked to this action. You don't have much control over which button is chosen. On most phones the accept type used in this example is linked to the OK button on a mobile phone. Other types include prev, help and delete. Enclosed within the `<do>` tag is the action to be performed – in this case, to navigate to another URL. This is actually a call to a WMLScript function, so we'll hold off.

• **Lines 30-36:** We have another `<do>` tag here with the same action type, which is perfectly legal; the user agent determines how to render it. On many cell phones it's rendered by providing a menu to select which action to perform when the button linked to this type is pressed. Here we're navigating to a different URL. The method attribute indicates whether we're performing a GET or a POST operation to this URL. And a new tag is enclosed within the `<go>` tag. The `<postfield>` tag is used to add data to a request string. The name/value pairs are provided as attributes to the `<postfield>` tag, and it's up to the WML browser to properly add these to the request as a query string in the case of a GET or as data fields for a POST.

At this point we have a nearly functional application. The only mysterious functionality is the URL defined in line 28, which doesn't look like a standard HTML or WML page location. This location is actually the syntax for calling a WMLScript function. The first part of the location, `JavaCafe.wmls`, identifies the location of the WMLScript library in which the function is contained. The second part of the URL, `#calculateTotal`, identifies the function we want to call. The final part of the URL, `('$purchase')`,

passes along the value of the variable purchase as a parameter to this function. The actual syntax of WMLScript is fairly similar to JavaScript, so let's look at Listing 2 to see what the `JavaCafe.wmls` code is doing.

• **Line 1:** It declares the method signature for this function. The keyword `extern` indicates that this function may be called from outside this package, similar to the `public` keyword in Java. Our function takes a single parameter named `inputPurchase`. It's not necessary to declare the type of the parameter since WMLScript is a weakly typed language. Data is converted automatically from one data type to another on the fly by the script execution engine.

• **Lines 2-4:** Declare and initialize three variables. Again, variables are weakly typed so no data types are required. The function call in the initialization of the purchase variable translates the escaped character string required by HTTP back into a standard string. For example, `%20` is translated back into a space character.

• **Lines 5-17:** This simple "for" loop iterates over the values in the purchase variable that are separated by semicolons. The elements method counts the number of elements in the string separated by the provided delimiter. The `elementAt` method returns the *n*th element of the string delimited by the provided delimiter. The "if" block is then used to calculate the total price of the order.

• **Line 18:** This is the key of the entire script. The call to `WMLBrowser.setVar` sets the value of a variable in the scope of the WML browser to the supplied value. Here we're changing the value of the totalPurchase variable, which is displayed in line 25 of `JavaCafe.wml`. The `String.format` method, which is similar to the C function `printf`, is used to perform basic message formatting.

• **Line 19:** The call to `WMLBrowser.refresh()` causes the WML browser to redisplay the current card with the latest variable information. However, this refresh is performed on the browser side. No trip back to the server is required.

A look at this example shows the similarities between WML/WMLScript and HTML/JavaScript to be fairly obvious. Although WML/WMLScript doesn't provide as much functionality as HTML/JavaScript, the basic purpose of each technology remains the same. WML and HTML are used for displaying a page of information, while WMLScript and JavaScript allow for a more interactive user experience within a page. However, one question remains: How do we generate dynamic content for the WML decks?

## JSPs, Servlets and Wireless Devices

Although there are many variations within J2EE architectures, in general, JSPs develop presentation-oriented dynamic content and servlets process user input and perform conditional navigation. Ideally, all your business logic would be located outside the JSPs and servlets in either EJBs or pure Java classes. Once the business logic tier is written, different presentation layers, such as WAP, can provide the interface to the business logic.

This paradigm is extremely applicable in current Internet technology. Many businesses operate transactional HTML sites. If their business logic is kept separate from the presentation logic, providing a new presentation layer such as WAP while still using a common layer for business functionality is a fairly straightforward process.

A few simple changes need to be made to enable our JSP/Servlet engine to serve up JSPs to a wireless device. The first set of changes must be made to the Web server that serves up the output of the JSP/Servlet engine in order to recognize the new MIME types for the content being displayed. Table 1 shows the typical MIME types.

To set the proper MIME type the next change must be made to the pages or decks. We can rename `JavaCafe.wml` from Listing 1 to `JavaCafe.jsp` and simply add the following code after line 2:

```
<%@ page contentType="text/vnd.wap.wml" %>
```

After these changes, your JSP/Servlet engine shouldn't have a problem serving up JSPs to a simulated WAP device. Of course, to serve up these responses to an actual WAP device you'd need a WAP gateway that linked your network to the cellular network. However, the JSP/Servlet engine setup would remain the same.

Using servlets in this model is just as simple. For example, the Check Out action (see Listing 3) directs the user to a servlet, then returns a simple WML deck with a Thank You message that includes the user's name. The user's name was retrieved as a field posted in the request to this URL. It's a simple modification to change this servlet to interact with any existing Java purchase logic. However, keeping in line with the clean separation of business and presentation logic, this servlet could also make calls to the business tier to perform the required business logic for checkout, then redirect to a JSP page to produce the presentation to be returned to the mobile user.

FILE EXTENSION	MIME TYPE
wml	text/vnd.wap.wml
wmls	text/vnd.wap.wmlscript
wmlc	application/vnd.wap.wmlc
wmlsc	application/vnd.wap.wmlscriptc
wbmp	image/vnd.wap.wbmp

TABLE 1 WAP MIME types

## Industry Support

Nearly every application server on the market is moving to support wireless in some fashion. At the most basic level, any Web server or JSP/Servlet engine can become WAP-enabled simply by defining the MIME types discussed above. However, many application servers are beginning to offer transcoding-type services. Transcoding services provide a semi- to fully automatic way to convert existing HTML or XML pages to WML. The idea is that by using a transcoding application server, your existing site will be available to any wireless device with a minimal amount of modification to existing code. Many application server vendors are also partnering with wireless vendors to provide a complete end-to-end wireless technology stack in a single solution.

## Conclusion

The Internet revolution has spawned billions if not trillions of dollars in new revenue streams. This is due in large part to the huge number of previously isolated people who are now empowered by the PC and the Internet to communicate in an easy and effective manner. Wireless technology extends the Internet's entry point beyond the PC. The number of wireless phones is predicted to surpass the number of land-based phones within the next three years. Most of them will be connected to some sort of wireless data service, which in turn will be connected to the Internet. The impact of this vast number of Internet clients is unimaginable. However, the implementation success that Java has had in the B2B and B2C space easily translates over to the wireless paradigm. In terms of scalability, reliability and maintainability, all the J2EE features can still be used in nearly the same way. Although it's certain that wireless architectures will evolve over time, Java will be there to help blaze the way. ☘

## References

1. Nokia's WAP information center, free SDK available: [www.nokia.com/corporate/wap](http://www.nokia.com/corporate/wap)
2. Home of the WAP specification: [www.wapforum.org](http://www.wapforum.org)
3. Great repository of information on wireless technology: [www.AnywhereYouGo.com](http://www.AnywhereYouGo.com)
4. Another excellent wireless developer portal site: [www.phone.com](http://www.phone.com)

### AUTHOR BIO

Kristian Cibulskis is an architect at Sapient Corporation in Boston, where he works with the Enterprise Java competency group. He holds a BS in computer science from Cornell University.

[kcibulskis@sapient.com](mailto:kcibulskis@sapient.com)

### Listing 1

```

1: <?xml version="1.0"?>
2: <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">
3: <wml>
4:   <card id="splash" title="JavaCafe" newcontext="true"
   ontimer="#main">
5:     <timer value="30"/>
6:     <p>
7:       <table columns="2">
8:         <tr>
9:           <td></td>
10:          <td><small>Welcome to the JavaCafe</small></td>
11:        </tr>
12:      </table>
13:    </p>
14:  </card>
15:  <card id="main" title="JavaCafe">
16:    <p>
17:      UserId: <input name="userid" title="UserId:" format="*M"/>
18:      Pin: <input name="pin" title="PIN:" format="NNNN" />
19:      Purchase:
20:      <select name="purchase" title="Purchase" multiple="true">
21:        <option value="COFFEE">Coffee</option>
22:        <option value="MOCHA">Mocha</option>
23:        <option value="LATTE">Latte</option>
24:        <option value="TEA">Tea</option>
25:      </select>
26:      Total: $(totalPurchase)
27:      <do type="accept" label="Calculate Total">
28:        <go href="JavaCafe.wmls#calculateTotal('${(purchase)}')"/>
29:      </do>
30:      <do type="accept" label="Check Out">
31:        <go href="http://localhost:7001/Checkout"
   method="post">
32:          <postfield name="userid" value="$(userid)"/>
33:          <postfield name="pin" value="$(pin)"/>
34:          <postfield name="purchase" value="$(purchase)"/>
35:        </go>
36:      </do>
37:    </p>
38:  </card>
39: </wml>

```

### Listing 2

```

1: extern function calculateTotal(inputPurchase) {
2:   var total = 0;
3:   var purchase = URL.unescapeString(inputPurchase);
4:   var currentItem;
5:   for (var i=0; i<String.elements(purchase, ";"); i++) {
6:     currentItem = String.elementAt(purchase, i, ";");
7:
8:     if(currentItem == "COFFEE") {
9:       total += 1.25;
10:    } else if(currentItem == "MOCHA") {
11:      total += 2.25;
12:    } else if(currentItem == "LATTE") {
13:      total += 2.00;
14:    } else if(currentItem == "TEA") {
15:      total += 0.85;
16:    }
17:  }
18:  WMLBrowser.setVar("totalPurchase",
   String.format("%$5.2f",total) );
19:  WMLBrowser.refresh();
20: }

```

### Listing 3

```

1: import javax.servlet.*;
2: import javax.servlet.http.*;
3: import java.io.*;
4: public class Checkout extends HttpServlet {
5:   public void doPost(HttpServletRequest req, HttpServletResponse res)
6:     throws IOException {
7:     // Must set the content type first
8:     res.setContentType("text/vnd.wap.wml");
9:
10:    // Now we can obtain a PrintWriter
11:    PrintWriter out = res.getWriter();
12:    out.println("<?xml version=\"1.0\"?>");
13:    out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"+
   \"http://www.wapforum.org/DTD/wml_1.1.xml\">");
14:    out.println("<wml>");
15:    out.println("<card id=\"checkout\" title=\"JavaCafe\" >");
16:    out.println("<p align=\"center\">Thank You "+
   req.getParameter("userid") +"!</p>");
17:    out.println("</card>");
18:  }
19: }

```

# The Challenges of Developing Distributed Java Applications

Build reliable, high-performance applications and components with Java technology



WRITTEN BY  
PETER VARHOL

In just a few years the Java language and platform has become the technical approach of choice for building complex, distributed and Web-enabled applications across the enterprise. Thanks to its cross-platform runtime environment, object-oriented development model, and facilities for working with object request brokers and other code components, Java is well equipped for building such applications.

Java enables software developers to provide seamless communication and application access to the rapidly growing world of Internet computers and communications devices, from UNIX servers and PCs to cell phones and beyond. IDC Research estimates that the Java products market will grow by 85% annually through 2004.

Java applications – especially those designed to work with other applications and components (often written in other languages) in a distributed environment – have different development requirements than traditional applications. For example, Java's execution model virtually eliminates traditional memory errors, but can introduce performance problems stemming from poor resource utilization. For those used to addressing traditional programming errors and other issues, Java's unique execution model and language characteristics may make building error-free and efficient applications more difficult.

The distributed nature of many Java applications can also make it difficult to pinpoint performance issues or diagnose programming errors. As a result, Java developers need software development tools and techniques for viewing and analyzing code execution on multiple systems on the network and from multiple code bases.

## Issues Surrounding the Development of Distributed Java

Java is unique – it's a mainstream programming language that works like no other. Its rules aren't well understood yet by many application developers. Part of the reason for this is that its capabilities and limitations haven't been fully explored.

As a result, while many of the problems may be similar, recognizing them and knowing what to do when you find them remains challenging, even to experienced Java developers. What follows are just a few of the development issues and what they mean for Java.

### Performance

Performance is a concern of applications written in any language. Most programmers are familiar with common performance issues using a conventional language such as C with a stand-alone or even a client/server application. Such issues often involve improper allocation, deallocation of memory and poor use of system APIs.

These aren't even characteristics of the Java language. For example, relating Java code to how the JVM manages memory is difficult and error-prone. However, it's vital to improve Java performance because its execution model has additional overhead that tends to degrade performance more than native applications do.

In addition to typical performance requirements and issues surrounding traditional stand-alone applications, distributed Java applications must contend with problems surrounding the interactions between components running on different systems. Performance problems may manifest themselves in unexpected ways or appear to be caused by different parts of the code other than the actual problem area. Identifying and locating performance bottlenecks rapidly is a significant challenge in distributed application development.

### Reliability

Enterprise Java applications, especially distributed ones, are often mis-

sion-critical in nature: all aspects of the application must work perfectly at all times. Developers and development managers must be able to gauge the reliability of their applications accurately. While the characteristics of the language tend to make Java applications less error-prone, there are still plenty of ways to introduce runtime errors.

With distributed Java applications the reliability equation is even more difficult. It means assessing not only the individual applications but also the components as they interact. Java also makes it possible to write highly threaded applications that make sense in a distributed environment. But using threads means that problems with resource contention and deadlock are much greater.

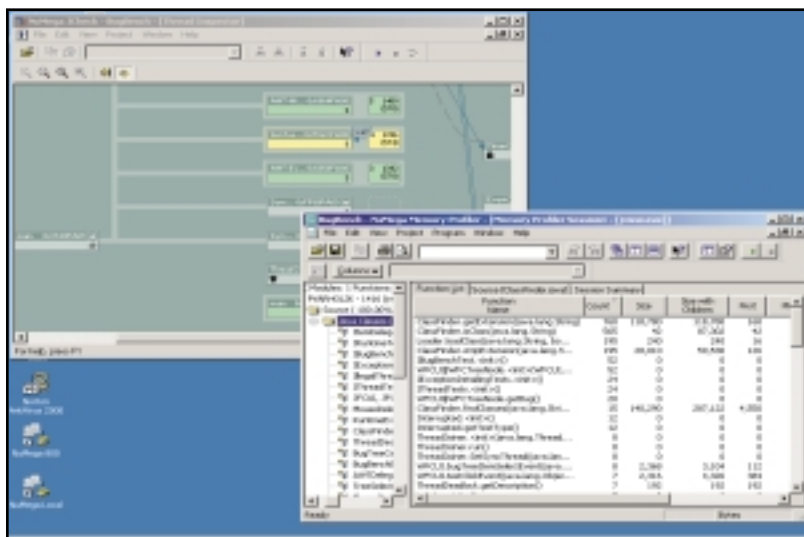
### Testing

Java applications face the same testing problems as traditional ones. They must be tested as thoroughly as possible before fielded, and developers should have a good idea of the extent of the test coverage before certifying an application. Distributed software systems written in Java, however, are extraordinarily difficult to test and debug. Because components reside on different computers and must work together perfectly for the application to work properly, all components must be tested simultaneously.

### Memory Management

Since JVMs perform all the memory management tasks for applications, understanding the memory usage of the underlying platform and influencing memory allocation and use to affect performance is difficult to do and not intuitive. That's especially true because developers have no easy way of deter-

mining the relationship between code and the underlying memory use. Analyzing underlying memory use is a key component of building efficient applications (see Figure 1).



**FIGURE 1** Inspecting threads and analyzing underlying memory use are essential in building distributed Java applications.

## Software Tools Can Make Java Transparent

Several of the integrated development environments for Java are quite good, combining visual development, context-sensitive editing, JIT compilation and runtime debugging. What they lack, however, is the ability to determine the efficiency and reliability of the application, especially if it's distributed across several servers.

Most developers do without such tools, due in part to the relatively poor selection. There are many different Java-oriented development environments, but few tools to move code beyond the development stage. A few tools, such as the Compuware DevPartner for Java suite, combine components that evaluate performance issues, examine memory usage, analyze threads and track testing progress (see Figure 2).

Many Java developers fail to recognize that it's not enough for Java applications to be debugged within the development environment because of several myths regarding the use of Java as a development language and environment. One myth is that the VM eliminates programming errors and bugs. While direct memory errors aren't usually possible in Java development, it's still possible for Java applications to contain serious errors that affect the proper operation of the program.

Another myth is that the application developer has no control over the performance of the software since the VM

manages the low-level details that determine how fast the code runs. How developers use specific language instructions can have a significant impact on application performance. Often a few simple

changes can greatly improve performance if developers know their underlying effect in advance.

Java development tools assist and accelerate the development of reliable, high-performance applications, especially distributed ones. They go beyond the traditional development environments to include components that help make applications more reliable and efficient. When choosing a set of tools to supplement those found in development environments and improve the reliability and performance of Java applications, developers and develop-

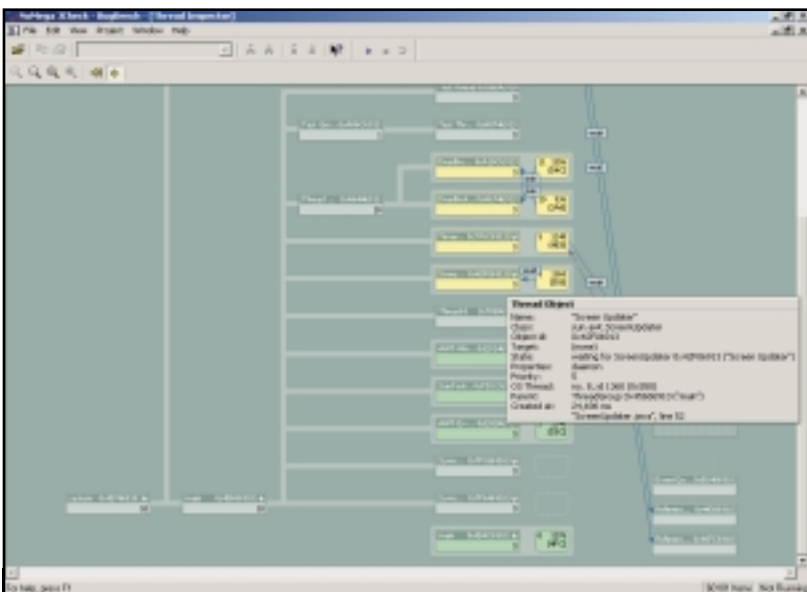
ment teams should take the following characteristics into account.

- **Support for multiple, unmodified VMs:** Different operating systems often use different JVMs with different performance and behavior characteristics. In addition, some Java tools require the use of specially modified VMs that may not represent the characteristics of production systems.

To obtain accurate information to improve the performance and reliability of distributed Java and mixed-language applications, developers should select tools that run in the actual deployment environment. This way they can be certain that the behavior observed and analyzed during development and testing will be the same once the application is deployed.

- **Minimal impact on Java runtime resources:** During application testing and analysis, the Java development tools that are used can often be intrusive – their presence may influence the test results. If developers attempt to deploy the tools for further testing and tuning in the production environment, they may also encounter unacceptable performance degradation due to high resource utilization.

If developers need accurate information on resource utilization, performance and system load, they should consider software tools that don't use large amounts of Java runtime resources. Tools with minimal Java and computing resource impact are more likely to provide the accurate information needed to improve the performance and reliability of distributed applications.



**FIGURE 2** The DevPartner Java Edition thread inspector enables developers to visually analyze Java threads.

- **Support for Web technologies such as JavaScript and Active Server Pages (ASP):** Large distributed applications that use Java often go beyond it to include many different software technologies, including JavaScript and ASP. Web software technologies can be a source of reliability and performance problems, and software tools that are unable to test for and identify these problems are of limited use.

Application developers need tools that work with whatever software technologies they're using. Multilanguage tools, especially those that support industry-standard Web languages, provide the range of capabilities needed to find performance and reliability problems anywhere in the application (see Figure 3).

starvation and code thrashing. These error conditions don't prevent the application from running; however, they cause severe performance bottlenecks and may even cause the application to hang while running.

The problems are especially prevalent in Java applications in which multiple running threads contend for limited virtual machine resources. To ensure the reliability of deployed applications, Java developers require software tools that enable them to identify and locate the complex combination of conditions that can cause resource starvation and thrashing.

- **Ability to measure code base stability:** Rapidly changing code during debugging and testing usually means that

will be deployed across several different types of systems. This is especially true of Java-based applications, which are designed to run unmodified on multiple operating systems.

Application development tools have to support multiple development and runtime environments to eliminate the expense of purchasing different tool sets for different platforms, and to reduce the need for developer training on multiple tool sets and platforms.

## Delivering Higher Quality Java Applications

Today's Web-enabled, distributed applications combine many different technologies and are prone to performance and reliability problems. Software developers using Java technology can spend a substantial amount of time trying to resolve these problems, leading to schedule delays and applications with ongoing problems.

Most of the Java development environments available are excellent for writing small to medium-sized standalone applications. It's growing increasingly difficult to write applications that are large or distributed, and to work with legacy components or databases. Performance and reliability issues overshadow the advantages of rapid, object-oriented development.

Visualizing these types of problems is an important aspect of debugging, tuning and testing applications because developers can quickly identify and localize the code responsible. If the developer can see where the deadlocked thread is, it's easier to pinpoint the resource that's deadlocked and its cause.

The kinds of problems that don't arise or are trivial in smaller applications take on critical importance in distributed processing. By using performance analyzers, memory profilers and thread inspectors such as those found in DevPartner Java Edition, distributed application developers can be sure their applications will run as expected.

Large-scale Java and distributed applications are easy to write but difficult to write well. To bridge the gap between mediocre or poorly performing applications and highly efficient ones, developers need to leverage software tools that help build reliable high-performance applications and components with Java technology. Such tools would allow Java developers to quickly and easily identify problems in key areas like runtime performance, memory utilization and multithreading. 🍌

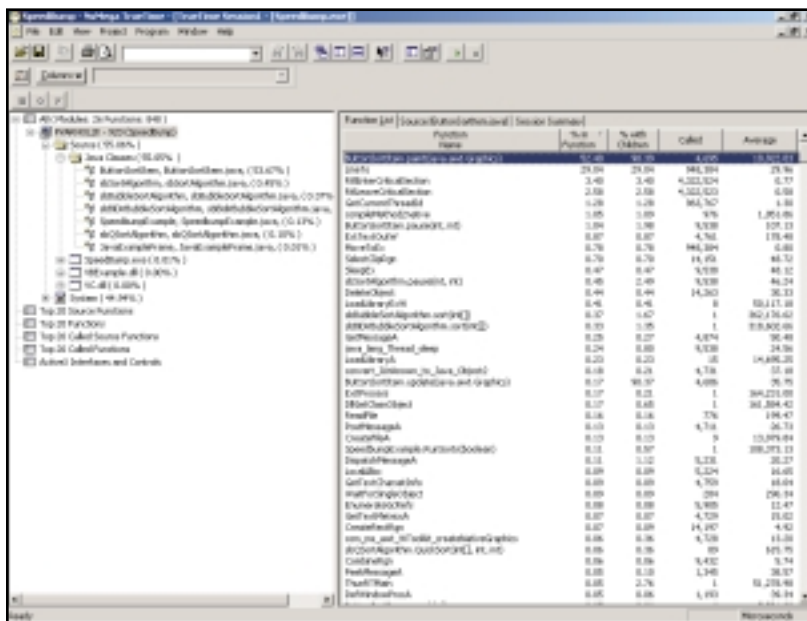


FIGURE 3 Analyzing the performance of distributed applications

- **Ability to track Java memory utilization to the function and line level:** Knowing you have a performance or reliability problem isn't any good unless you're able to pinpoint the source rapidly and easily. The more precise the diagnosis and analysis, the faster application developers can address the problem.

Java application development tools should focus as specifically as possible on the exact location of a performance bottleneck or software error, especially for large, distributed applications. Developers working on deadlines need the most exacting information possible from their tools.

- **Ability to find thrashing and starvation conditions graphically:** Some of the toughest software problems to find are those that involve resource

the application may be unreliable or needs additional testing before deployment. Conversely, an application with few changes to the code base during debugging and testing will more likely be fully tested and stable.

It's important, therefore, for application developers to understand how much and how rapidly their application code is changing during the latter stages of the development process. An application development tool should measure the stability of the code base to enable both the development team and the managers to determine when the application can be used reliably in production.

- **Support for multiple operating systems:** Mixed operating environments are the rule rather than the exception. Even if developers code on one platform, it's increasingly likely that the application

### AUTHOR BIO

Peter Varhol is a technical evangelist for the Computware NuMega product line. He holds graduate degrees in computer science, applied mathematics and psychology.

[peter.varhol@numega.com](mailto:peter.varhol@numega.com)

# PART III FINALE: A SQLJ MAGIC SHOW

Concluding our series on SQLJ,  
the standard for embedding database  
SQL statements in Java programs

WRITTEN BY EKKEHARD ROHWEDDER

**S**everal fun and important secrets of SQLJ will be unlocked today. Our show will include the following numbers:

- Some magic tricks for taming the SQLJ translator to do your bidding.
- Some incantations to turn you into a SQLJ debug-mon.
- An initiation to the mysteries of execution contexts - for getting full control over executing SQL statements - and of connection contexts.
- A happy story of brotherly love between JDBC and SQLJ.
- A map of the hidden location containing all of the remaining SQLJ details that the wonderful folks from *Java Developer's Journal* neither would or could let me fit into this column.

Little space and much to say - let's get started right away!

## Your Own Private Translator . . . Is Talking Back

Don't fret if you're stuck and need help - the SQLJ translator may just be able to give you what you need. Enjoy the following translator command-line options:

- **sqlj -help** shows a short help message with the important SQLJ command line options. And when you just say *sqlj* by itself this is also recognized as a cry for help.

## The Translator-Chameleon Is Serving Up Bugs

How I wish that SQLJ translation were built into the javac Java compiler. But tough luck – it ain't! The translator strives hard to make it appear however as if the only difference is saying *sqlj* instead of *javac*.

*Nice:* You can freely mix .sqlj and .java source files on the command line. The translator also has some implicit make capability similar to javac. Oh yes, it automatically invokes your Java compiler – but you already knew that!

*Nice:* The translator reports error messages from your Java compiler on the original SQLJ file and not on the generated Java file.

*Not Nice:* When your program throws exceptions at runtime, line numbers – such as those issued by `printStackTrace()` – are shown in terms of the generated Java files.

### INCANTATION #1

Add the flag `-linemap` to your command line during translation. Then the translator will fix up file names and the line numbers in those class files to show the original SQLJ files. If you want this on by default (which it probably should have been in the first place), then create an environment variable `SQLJ_OPTIONS` with the value `"-linemap"`. Or you can add the line `sqlj.linemap` to a `sqlj.properties` file.

*Still Not Nice:* If you use Sun's Java debugger `jdb` to debug your SQLJ program, you'll see that...this tip does not work: `jdb` refuses to show the SQLJ source. No wonder — they only taught it about .java source files!

### MAGIC SPELL #2

Shout `-jdblinemap` instead of `-linemap` whenever you must trick that silly little (de)bugger.

*God Mode:* Well, not quite...You can trace the goings-on in the SQLJ runtime by installing a profile auditor in your SQLJ profile files (remember those pesky little .ser files that hold the static SQL information of your program). After the usual translation and compilation you add tracing with the following command:

```
sqlj -P-debug *.ser
```

Have fun drinking from the fire hose!

## More Dizzying Debugging Spells

Remember the JDBC Genie? On many platforms (such as Oracle's), SQLJ runtime calls turn into calls to JDBC at the end of the day. You can trace JDBC calls with `DriverManager.setLogStream()` (or `setLogWriter()`).

Don't count out them IDEs: Oracle's JDeveloper has been supporting SQLJ programming and debugging for a while. Other IDEs may offer SQLJ support Real Soon Now.

*Smokey Bear Says:* Only you can prevent runtime bugs. You should always provide the translator with the

```
-user=<name>/<password>
```

option, so that it can check your SQL code for real.

## Become a Control Freak — with ExecutionContexts

At times you need to obtain additional information about a SQL statement that just ran or you need to control exactly how you want SQL statements to be executed. Take the following examples:

- The statement might have resulted in a warning (not an exception) that you want to inspect.
- An UPDATE or DELETE statement reports the number of rows that was changed or – respectively – removed.
- You want to set a timeout or designate a prefetch size for queries.
- You want to batch several SQL statements and have them all executed together, rather than pay a round trip to the database for each one. In particular, you want to combine the repeated execution of a DML statement, such as an INSERT, an UPDATE or a DELETE that takes on

different values for its bind-variables.

All of this – and more – is available on the `sqlj.runtime.ExecutionContext`. Every connection context (remember, this is the SQLJ equivalent to JDBC connections) has an associated `ExecutionContext` which can be accessed with `getExecutionContext()`. Your static `DefaultContext` also has one. Consider this example:

```
import sqlj.runtime.ref.DefaultContext;
import sqlj.runtime.ExecutionContext;
...
#sql { UPDATE emp SET sal = sal * 2 };
ExecutionContext ec = DefaultContext.getDefaultContext().getExecutionContext();
System.out.println( ec.getUpdateCount() + " employees are rejoicing.");
```

Or, you can terminate a whole bunch of bad guys in a single batch as follows.

```
String[] badGuys = { "HORRIBLE", "TERRIBLE", "NOGOOD", "VERYBAD" };
ec.setBatching(true);
for (int i=0; i<badGuys.length; i++) {
    #sql { DELETE FROM emp WHERE ename = :{badGuys[i]} };
ec.executeBatch();
```

If we hadn't issued `executeBatch()` explicitly, then the `badGuys`' execution would happen implicitly when a different `#sql` statement is encountered. Alternatively, we could have used `ec.setBatchLimit(4)`; to tell SQLJ to always flush a batch implicitly once 4 rows have accumulated. This convenience feature is not (yet?) offered by JDBC.

## Being Well Connected — Explicitly

Until now you've been brainwashed. You were led to believe that there's only a single, static connection in your SQLJ program, set once with `DefaultContext.setDefaultContext(...)` and then forgotten about. Even though this makes great marketing copy, it's not the (only) way the world works.

If you're connecting to more than one schema, if you're running an applet in a browser or if you're connecting to the database in a multithreaded program, then you should – nay, you must – use explicit SQLJ connections.

Don't worry: it's really easy. You just put the connection context instance (or an expression evaluating to one) in those square brackets:

```
[context].
```

An example:

```
import sqlj.runtime.ref.DefaultContext;
...
DefaultContext ctx =
    new DefaultContext("jdbc:oracle:oci8:@", "scott", "tiger",
false);
#sql [ctx] { UPDATE emp SET sal = sal * 2 };
```

*Tip:* Always use explicit connection contexts, unless you know that your program owns the world and requires only a single, static database connection.

## Rolling Your Own Connection

Sometimes you need to distinguish between different schemas or databases. Consider a program that establishes connections to two different database schemas. The `PILOTS` schema has personnel data, flight hours and so on for pilots, and the `JETS` schema contains maintenance and flight data of aircraft. Your program is working on two different sets of tables, views and stored procedures. Now you need to verify your SQLJ statements against both of these schemas. Typed connection contexts are what let you do this. First, declare two different context types:



```
#sql context Pilots;
#sql context Jets;
```

At runtime, you connect to each of the schemas using the appropriate type:

```
Pilots pconn = new
Pilots("jdbc:oracle:oci8:@", "pilots", "ace", false);
Jets jconn = new
Jets("jdbc:oracle:oci8:@", "jets", "stratos", false);
```

Then the connection context type of the #sql statement clearly shows whether you want a Pilots or a Jets connection:

```
#sql [pconn] { INSERT INTO pilot VALUES ( .... ) };
// Pilots context
#sql [jconn] { UPDATE maintenance SET status = Checkup( .... ) };
// Jets context
```

Of course at translate time you also need to explain how to connect to the database for both connection context types:

```
sqlj -user@Pilots=pilots/ace -user@Jets=jets/stratos MyFile.sqlj
```

## SQLJ and JDBC: Living in Perfect Harmony

SQLJ works just fine and dandy with static SQL – where you know the shape of SQL statements and queries beforehand. But what if your application has to make up the WHERE clause in a SELECT statement on the fly – guess you'd better forget all about SQLJ, right?

Not so quick – SQLJ and JDBC are actually close-knit buddies. JDBC connections and SQLJ connection contexts are mutually convertible and so are `java.sql.ResultSets` and SQLJ iterators. Let's look at the specifics.

### Connecting from JDBC to SQLJ

All connection context constructors and initializers can take an existing JDBC connection. Example:

```
java.sql.Connection conn = DriverManager.getConnection(...);
DefaultContext ctx = new DefaultContext(conn);
```

Now SQLJ and JDBC share the same session. Closing the SQLJ context will also clean up the JDBC connection.

### Connecting from SQLJ to JDBC

All SQLJ connection contexts have the `getConnection()` method that allows you to retrieve an underlying JDBC connection. For example, if your program uses the static default context, the following will do:

```
java.sql.Connection conn =
DefaultContext.getDefaultContext().getConnection();
```

### Passing Result Sets from JDBC to SQLJ

If you want to pass off a JDBC result set as a SQLJ iterator, you can do so with a SQLJ CAST statement:

```
SomeIterator iter;
java.sql.ResultSet rs = stmt.executeQuery();
#sql iter = { CAST :rs };
```

Why that CAST statement and not simply a Java constructor? So that any vendor's SQLJ runtime implementation can scrutinize that result set very closely.

### Passing Iterators from SQLJ to JDBC

This one's a breeze. You just call the iterator's `getResultSet()` method and – voilà – your JDBC ResultSet.

Exercise 1: Why might it be useful to convert a JDBC result set into a SQLJ iterator or vice versa?

## Jump into the Fray

Hope you enjoyed the journey, even though we've not yet seen all the vistas. We stopped short of covering the new fun JDBC 2.0 features that made it into SQLJ, such as support for various SQL LOB types, named SQL types, DataSources and scrollable iterators. Nor did we look at how to put iterator subclassing to use or how connection caching meshes with SQLJ or even speculate what directions SQLJ might take in the future. Is there a place to learn more? In my totally biased opinion (Oracle pays my bills, after all) the vastest amount of information on SQLJ is on the Oracle Technology Network site at <http://technet.oracle.com>. You'll find the world's most humongous SQLJ manual that answers more questions than you could ever dream of, as well as demos, samples, papers, an FAQ and so on. Beware, though, the Oracle-specific is happily stirred in with the generic.

Better yet, *do it*. If you want to get involved on the standards side, subscribe to the SQLJ partners mailing list at [sqlj-ext@eng.sun.com](mailto:sqlj-ext@eng.sun.com). Or, if you want to hack the SQLJ reference implementation – yep, all source, all public domain – go to [www.sqlj.org](http://www.sqlj.org). Or download one of the SQLJ implementations from the IBM, Informix, or Oracle Web site and get started. ☪

## SUMMARY OF SQLJ SYNTAX

- Connection context declaration and use:

```
#sql context CtxType;
...
CtxType ctx = new CtxType(url, user, pwd, auto-commit);
#sql [ctx] { ...sql statement... };
```

- Setting the translate-time connection for CtxType:

```
sqlj -user@CtxType=user/pwd ...
```

- ExecutionContext declaration and use:

```
import sqlj.runtime.ExecutionContext;
...
ExecutionContext ec = new ExecutionContext();
... set properties on ec ...
#sql [ec] { ...sql statement... };
... retrieve warnings, update count, side-channel results, etc. from
ec ...
```

- Using an explicit ExecutionContext and an explicit connection context:

```
#sql [ctx, ec] { ...sql statement... };
```

- Casting SQLJ iterators to JDBC result sets:

```
SqljIterator iter;
#sql iter = { CAST :jdbc_result_set };
```

## AUTHOR BIO

Ekkehard Rohwedder leads the SQLJ development at Oracle. SQLJ has come about as a collaboration of many people from several companies. Special kudos go to the members – past and present – of the Oracle SQLJ group and of the intercompany SQLJ working group. Thank you for being coauthors!

[erohwedd@us.oracle.com](mailto:erohwedd@us.oracle.com)

# Ensemble Streams 3.2

by Ensemble Systems Inc.

REVIEWED BY JIM MILBERY



**AUTHOR BIO**

Jim Milbery is a software consultant with Kuromaku Partners LLC ([www.kuromaku.com](http://www.kuromaku.com)), based in Easton, Pennsylvania. He has over 16 years of experience in application development and relational databases.

[jmilbery@kuromaku.com](mailto:jmilbery@kuromaku.com)



Ensemble Streams 3.2  
 Ensemble Systems Inc.  
 280-5200 Hollybridge Way  
 Richmond, BC Canada V7C 4N3

Phone: 604 231-9510  
 Web: [www.ensemble-systems.com](http://www.ensemble-systems.com)

Pricing: Streams Professional 3.2: \$995/Developer

Test Environment  
 Gateway GPi, 196MB RAM, 30 gigabyte disk drive,  
 Windows 2000

Moore's Law essentially states that the processor speed for chips doubles every 18 months, and it's proved to be a fundamental tenet of the high-tech industry. Milbery's Law, on the other hand, has been less rigorously proved. It states that developers have to double their output with fewer resources every time they're asked to do so. Machines get faster and faster and developers get further and further behind. One solution to this endless cycle is to aggressively model and analyze your business requirements before you start slinging code. Ideally, the closer you track your software requirements with the needs of the business, the more likely it is that your code will match the business requirements on time and on budget. Toward that end I had the chance to look at Ensemble Systems' Ensemble Streams Professional 3.2, a business-process tool for communicating workflow models to end users, analysts and teams of software developers.

## Ensemble Streams Overview

Ensemble Systems was founded in 1995 as a consulting services company with a specialization in object-oriented technology. They offer a number of software products in addition to providing consulting services to clients. As software consultants they have first-hand experience working closely with business analysts and client sponsors within corporations. I expect the idea for Streams grew out of their own experiences helping these clients design and implement complex applications.

Ensemble positions Streams as a tool for bridging the gap between business

and software professionals. While you'll frequently use it to model software applications, it can be used to model any type of business process. The interface is designed to allow business analysts to model out the workflow of business process scenarios. Professional developers can use this same "business analyst" model as the basis for building a complete technical software solution. Streams can bridge the gap between business analysts and programmers by providing tools that allow analysts to include business terminology and workflows directly within a UML (Unified Modeling Language) model. This model can include actors, classes and use cases, which can be transferred into Rational Software's Rational Rose tool where it can be enhanced by professional developers and used to generate complete applications.

Application designers and systems analysts have rallied around the Object Management Group's UML. Ensemble supports standard UML 1.3 modeling, but also lets you extend the model with custom symbols and notations. Streams allows you to add documents and screenshots to your models, making them much more than just UML diagrams. Despite these additions, the product is a UML tool at its core. Whether you're a business analyst or a professional programmer, you'll need to have some familiarity with UML modeling to work successfully with Streams.

## First Impressions

Ensemble provides a downloadable version of Streams on their Web site, but I installed the product using a CD-ROM provided with the press package from **JDI**. Streams supports Windows-based operating systems and uses Demo Shield for installation. After viewing a short pre-

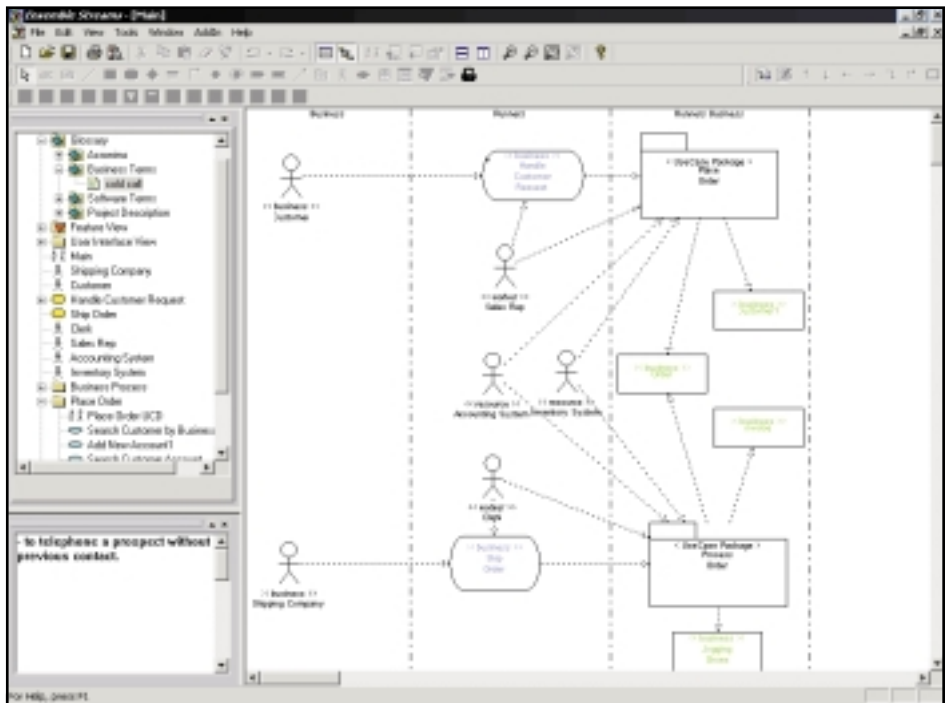


FIGURE 1 Streams 3.2 interface

sentation from the CD, I installed and configured Streams Professional in just a few minutes. Streams is licensed on either a fixed license basis or by a series of floating licenses (all provided by Globetrotter's FlexLM). However, the demo version comes equipped with a 30-day trial license, so you can get started without having to deal with licensing issues. The Streams interface is clean and well designed, as shown in Figure 1.

Seven basic elements are part of the Streams user interface. The various functions and commands are organized into three layers: menu bar, toolbar and toolbox. The toolbar serves as a shortcut for options available within the menu, while the toolbox contains all the notational elements that can be placed within a diagram. You're free to rearrange elements of the toolbar and toolbox to match your preferences.

The three main elements of the interface are the drawing area, the browser and the documentation panel. Notational elements can be selected from the toolbox and placed on the drawing area, as shown in Figure 1. As you add elements to your model, they'll appear in the browser window in an outline control, and detailed descriptions of each selected element are displayed in the documentation editor. By default, these three views appear as docked panels, but you can undock them into separate panels if you wish. Status information and messages are displayed in the status bar at the bottom of the panel.

Ensemble doesn't provide a tutorial for Streams, but they do provide product documentation in Adobe PDF format and a white paper that details the process of using Streams with a fictional specialty sneaker store. I used the "Runners" sample case and example model (see Figure 1) to get my bearings with Streams, but the document isn't a tutorial in the classic sense of the word. The installation CD also comes equipped with several additional white papers and PowerPoint presentations - I wasn't overly impressed with the written materials. It's my guess that most business analysts would have some trouble with Streams unless they were already familiar with the basics of UML modeling. Don't get me wrong. The interface is well designed, and it's certainly easier to start modeling with Streams as compared to jumping directly into the Rational Rose 2000 product (from a business-analyst perspective that is). However, Streams still makes use of UML and you'll need to know UML modeling before working with the product.

The folks at Ensemble have added some clever features that make Streams useful as a tool for defining workflows and business rules. For example, a business analyst typically designs a workflow or a set of business rules and then works with end users to validate the model. To facilitate this process Streams allows the analyst to include extra elements on the activity diagrams such as notes, hyperlinks, documents and even sample screen designs. Thus the diagram becomes much more than a standard UML model. When it's time to actually build software from the model, Streams exports the UML portions directly into Rational Rose (see Figure 2).

Ensemble provides additional products, one of which connects Rational Rose to a number of leading Java development environments such as Oracle's JDeveloper. Theoretically you can then take a project all the way "from concept to code" using Streams, Rational and the Java IDE. However, you can also use Streams to output your project into a series of HTML pages (see Figure 3) that can be shared with users across an intranet. Although the HTML add-in isn't unique to Streams, I found it to be a nice addition to the interface. Streams generates a hierarchical HTML index page with links for each activity and subactivity as defined in the model. I can easily envision a business analyst working with Streams to define workflows, then using the company intranet to share the work-in-process with business sponsors. If you require a more rigorous tool for walking users through the model, Streams provides a "storyboarding" facility. Through this interface you can add some basic animation and extra description for the model. As the product description states, Streams isn't limited to software development; it can be used with almost any business process. The CD even includes a presentation that describes the use of Streams to build an activity-based costing (ABC) model of a business flow. Thus those organizations looking to optimize some of their business processes using methodologies such as ABC and Six Sigma can also benefit from a product like Streams.

## Summary

I found Streams to be a clever product for designing workflows and software processes, and I liked the concept of working at a higher level of abstraction from the model. You'll need to purchase companion products to implement these models as software, but I didn't find this a burdensome requirement. If you're not familiar with UML modeling, however, I'd suggest you invest in some UML training before diving into Streams. 🍌

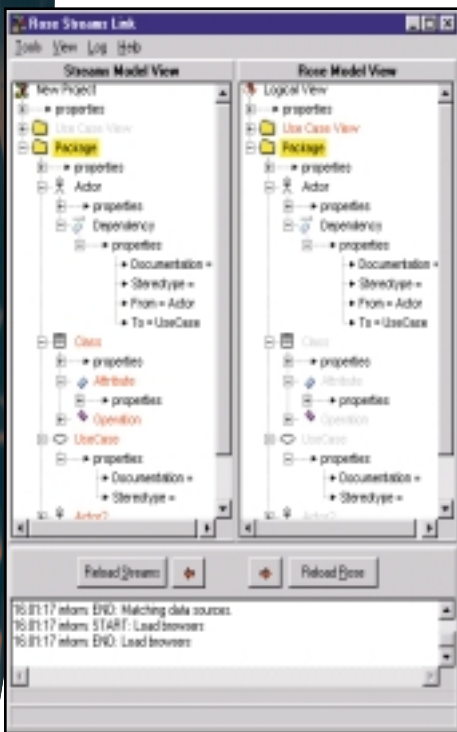


FIGURE 2 Streams RoseLink

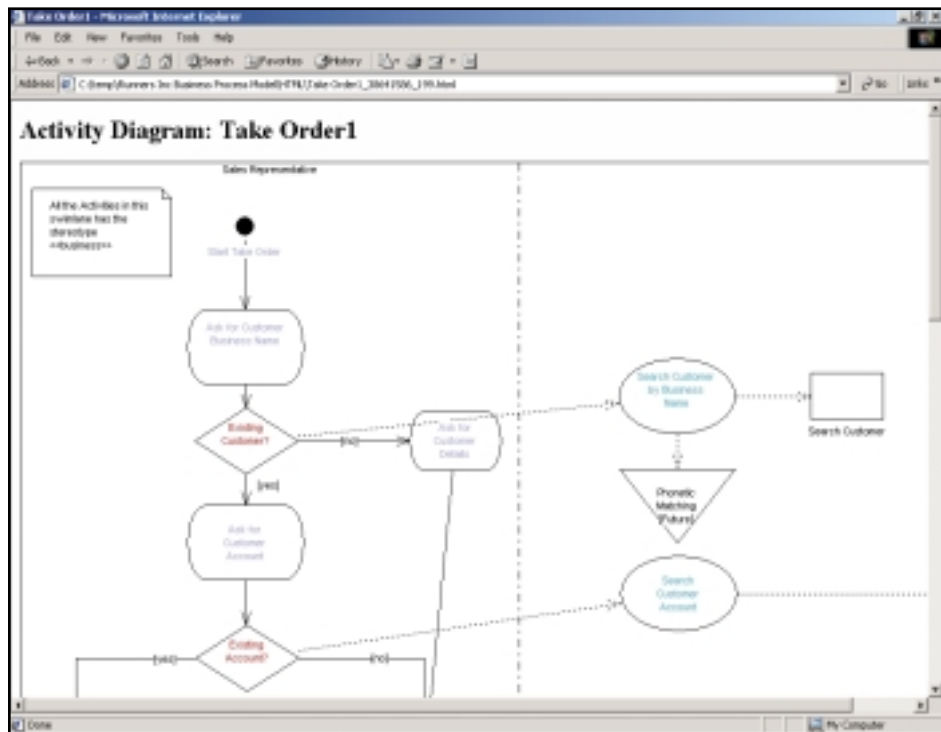


FIGURE 3 Streams HTML Output



# Interview...with Paul Chambers

## PART 1

## CTO OF GEMSTONE SYSTEMS (EUROPE)

AN INTERVIEW BY JASON WESTRA

**JDJ:** Paul, I'd like you to give us some technology trends and talk a little bit about GemStone's role in the wireless market. Tell us some of the key players in that market right now.

**Chambers:** The wireless market is a little bit different for people in software and it's been driven, really, by the telecommunications industry, which makes it a bit different. I think that if your name is on the play list, you can see where it's going to go within the next few years. The sorts of players that you've got in the market are telecommunications equipment manufacturers and the northern mobile operators, so you've got companies like Nokia and Ericsson driving the standards, because they have rolled out infrastructure which you've got to have to support the standards. They ship the kit, like mobile phones and small phones, which actually can support the standards as well. So they pretty much have a big say in how things go. I think they're becoming the ones to watch in terms of companies and the market.

**JDJ:** What exactly is driving this trend in the wireless market?

**Chambers:** Over the next few years it's going to be quite a big revolution in terms of telecommunications and technology. With the fundamental technology out there, you can see by the software applications and standards for applications where they're going to go in the next five years. You've got the North American market and the European market and the standards are a bit different and not necessarily compatible. In Europe at the moment what we have is a GSM network that's operating at about 14 kilobits per second. Now it's just variable for data, it supports data. You can run things like WAP, the Wireless Application Protocol, over there, but it's pretty slow, and if



reception is a bit bad, it cuts even further down to 5 or even 4 kilobits per second. So it's pretty primitive stuff we've been running. You can do the same thing in the United States – in the States is a standard called CDMA, and it's supposed to send this sort of bandwidth and people are learning WAP over it. So again, it's pretty primitive, it's pretty slow and the key thing that's going to happen in Europe this year is a new technology coming out called GPRS, which is an upgrade to the GSM system. Now that delivers about 115 kilobits per second. That's pretty useful and you can start doing stuff, especially with small sorts of applications. Another thing that is a big difference is it administers IP, mobile TCP/IP, so it's always on – it's packet switched – so it's moving away from circuit-switched systems to packet-switched systems. The same thing is going to happen in North America as well with CDMA 2000; it's going to go to packet switching. Now that's much more interesting for applications, much more usable.

**JDJ:** It sounds like there are a number of standards that are emerging. How does J2EE fit in, and how does GemStone offer it in wireless solutions in the future?

**Chambers:** Really, it's the ongoing world for J2EE. I think maybe the way to view J2EE is as an aggregator, an independent aggregator. It's activating information at the back end – services and communication, data, etc. – and at the front end it's becoming an aggregator for different channels out there. So the things that can be predominant out there aren't just the Internet with PCs. You're seeing digital TV, Web phones, mobile devices – all sorts of different devices out there. And the role for J2EE is to be the aggregator and deliver to all of them, with all the channels out there independently. With GemStone, what we've been doing is working with clients who actually deliver wireless applications with GemStone at the back end, so the standards, such as the servlets and JSPs, work with most of the wireless products, such as the WAP application servers. The WAP application servers, essentially, are very much like the Web servers: they serve up pages. They solved some of the problems of the narrow bandwidth so instead of binary messages to the mobile device, they handle different protocol stacks so they're supporting things like circuit-switched data over GSM, circuit-switched over CDMA here in the States. Obviously, that rolls over into the new

technologies we were just talking about. So they have a new set of challenges to hit. You've got mobile devices, less memory, you've got an over-the-air network with less bandwidth, so GemStone fits in behind that. What GemStone actually does is deliver more of what all applications should do. It's delivering a level of smartness, a level of caching. If you use a lot of the wireless applications every minute, oh man, they're bad! And, if you can put a smart system like GemStone behind the scenes, you can get much more user-friendly, much smarter systems. When you remove that, you get interrupted, you lose the connections, etc., etc. You need some smartness both from the device and on the back end so that you can resume correctly, so you can remember what you were doing. That's a key bit with GemStone – to be smart. And the final piece is robust, scalable, delivering to a high through-put. If you look at the volume, certainly in Europe the number of people who have wireless connections over PCs is about two and a half times fixed-line access – two and a half times in the mobile world, and the rest of it is going to be so upset by things like digital TV. Both of those platforms are going to supersede PC access. The numbers that are going to go through these systems are going to be huge – concurrent users and concurrent transactions. With J2EE and products like GemStone/! it's critical to have them at the back end to deliver what we've been delivering on the Web.

**JDJ:** Tell us a little about the role of the persistent cache architecture that GemStone has and how that's a key player in the wireless.

**Chambers:** There are two key bits. Dr. Lougie Anderson, GemStone's vice president of engineering, was talking about two things we have: a persistent cache architecture in GemStone and extreme

clustering. The extreme clustering really tries to solve all the scalability problems. Basically, what it's trying to do is implement a multivirtual machine architecture to get around all the scaling issues with single VM architectures. So you've got scalable garbage collections, scalable resource management. And it's robust and can run for a very long time without being brought down. That's what the extreme clustering gives and that's what we're bringing to wireless, that channel-neutral platform. We have clients that actually run dual channels into the same application – it's no problem with our servlet technology. Somebody in the request is from and just formatting it slightly differently, it's not a big challenge. In terms of the PCA – what the PCA (the persistent cache) and all in GemStone is – it's really a shared cache where all the VMs can collaborate. It's like a community of VMs in a cluster can actually collaborate and that's the way you can put the smartness in. This is where you can build in protection again so when people lose their signal on wire-

less applications, the system has to remember where they were and get back to a current state and you can do that in a scalable fashion. PCA is really enabling that sort of smartness and robustness.

**JDJ:** *Can the PCA, the persistent cache architecture, keep track of those transactions and pick up where they left off?*

**Chambers:** Absolutely. If you take the Web technology and the wireless WAP technology, it's got the concept, it can do some sessions, so that's in the cache itself. Tracking will be done via cookie support and session support, so when people lose connections and resume connections they'll see what they were previously doing. What we really need to do, though, is see the next generation of phones, because you need smartness at the phone end as well. When you lose a connection, the last thing you want to do is lose the whole service. You want to be able to continue... you don't want to lose what you previously did. That's a concern there, you do lose that. When we talk again we can talk about how the devices are evolving

over the next few years as well and see how that will change things.

**JDJ:** *There have got to be some customers that are driving all these changes in wireless and this whole trend. Could you give an indication of some of the customers that you have dealt with in Europe as well as the United States that are driving GemStone's movement to the wireless?*

**Chambers:** Yes. What we're seeing is the finance industry is moving pretty quickly on this. Some of the typical applications we are going with are Internet banking for all the mobiles. The other thing we're seeing is B2B as well, basically for field people who need status updates at all times. We have a client here in the States called Buildscape. Buildscape is actually a B2B exchange for the building industry, you know, the building merchants. A lot of people are out on sites, and they use their Palm Pilots and WAP phones, etc. They actually place orders, check on the status of things, check on deliveries. Another company we're working with on their wireless drive is eConnections. They were previous-

ly Marshall Industries. They're rating similar things where they've got field staffing to actually have access to the current status and they're also tracking just where things are at, they track where their orders are at and where deliveries are at in their logistics system. On the banking side, another company we're working with is First Rand in South Africa. They're running a dual strategy both with the Web and on wireless applications. Again, Internet banking, aggregating all the financial services, stuff that you like to do on the fly, stuff that you want to do. You don't want to put on a whole PC just to make a payment – it's a big thing. You just want to switch your phone on, switch your Palm Pilot on, and do it there and then. So people like going down those routes for those sorts of items.

**JDJ:** *We'll be back again next month to continue our conversation with Paul Chambers.* ☪

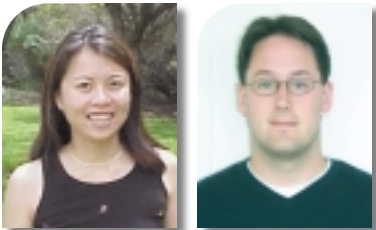
*Jason Westra is CTO of Verge Technologies Group, Inc., and a columnist for JDJ.*

[jwestra@vergecorp.com](mailto:jwestra@vergecorp.com)

# Developing Web Applications Using VisualAge for Java and WebSphere Studio

Part 1

WRITTEN BY  
ANITA HUANG &  
TIM DEBOER



To ease the server-side programming process, it makes sense to build the application using a role-based approach:

- **Business logic:** Determines the content generated
- **Presentation style and content:** Determines how the information is presented

Whether you're a Java programmer who wants to work on the server side of a Web application or a graphic designer who may be more interested in working on the client view of the application, VisualAge for Java and WebSphere Studio provide the tools you need.

In creating a Web application, it's common to create Enterprise JavaBeans to access the back-end data (the business logic side of the application), servlets to control the flow through the application, and JavaServer Pages to display the information to the user (the presentation style and content of the application). In this scenario the Java programmer would concentrate on developing the EJB and the graphic designer would concentrate on the JSP design. Depending on skill or complexity, both could work on the servlets.

IBM provides VisualAge for Java for your server-side EJB development and WebSphere Studio for your client-side JSP/Servlet development. In Part 1 of this series we provide you with an overview of the tools available for Web application development and how they can be used to build and debug a complete end-to-end Web application. In Part 2 we'll build a complete Web site from the EJB created in

our team has been assigned to build an end-to-end Web application. As a Java programmer, you need to focus on the code, to ensure that it can successfully call the required data. Your graphic and Web designers need to focus on the actual presentation to the user, to determine how best to display the information.

the article "Building Enterprise Beans with VisualAge for Java" (*JDI*, Vol. 5, issue 6).

## Summary of Tools

### WebSphere Studio

WebSphere Studio is a suite of tools that allows everyone on the Web development team to work together. It helps page designers, graphic artists, programmers and Web masters to work in such a way that each particular expert in the field can concentrate on doing his or her job well. The tools provided with WebSphere Studio help you to create, assemble, publish and maintain dynamic interactive Web applications powered by IBM's WebSphere Application Server. Use WebSphere Studio tools to create HTML, JSP and servlet files.

WebSphere Studio provides the following tools that will be useful for your client-side development.

- **JavaBean Wizard:** Allows you to import a JavaBean and then generate from it an HTML input page, a servlet that uses the bean, a servlet configuration file and a JSP file to dynamically display the results to the user.
- **Page Designer:** Advanced HTML editor that allows you to easily edit and create HTML and JSP files; provides support for servlets, JavaBeans and JavaScript so you can integrate these technologies with the generated client files. Also provides scripts for you to incorporate into your Web pages so you don't have to write the code yourself.
- **WebArt Designer and Animated GIF**

**Designer:** These tools enable you to create your own images.

### VisualAge for Java

VisualAge for Java, an enterprise-level Java development environment, provides a complete team-programming environment with all the tools you need as a professional Java developer. It's used to create JavaBeans, complex servlets and EJB, as well as to run, test and debug your Web application.

VisualAge for Java provides the following tools for Web-based development.

- **WebSphere Test Environment:** One of the biggest reasons to use VisualAge for Java as a development environment. This feature is a fully functional version of WebSphere Application Server that can run entirely within the VisualAge for Java IDE. This means you can run your entire application within the development environment, making use of the integrated tools and debugging support.
  - As you build the application, you can configure the WebSphere Test Environment to match your deployment platform, including the creation of multiple Web applications, virtual paths and aliases. This allows you to re-create your entire deployment environment for debugging and testing before you deploy the application.
- **EJB server:** The EJB development and test environment was covered in the June article referenced earlier. It also allows you to run the WebSphere

Advanced EJB server inside VisualAge for Java.

Figure 1 shows the EJB Server Configuration dialog. In the left pane you can start the Persistent Name Server and any number of EJB servers. The right pane shows the EJB groups and EJBs within the selected server. You can use

this dialog to configure deployment descriptors for your EJBs and to set up the properties of the various servers.

To test your EJBs, you can run both the EJB server and any number of client applications, all within VisualAge for Java. If the client application isn't finished (or for rapid testing of EJBs), a test

client can be generated automatically. This test client is a simple application that allows you to test access to both the home and remote interfaces of each EJB. When you want to access the EJBs from a client application, VisualAge for Java can generate access EJBs that allow a client application to connect to your EJBs without writing any EJB code.

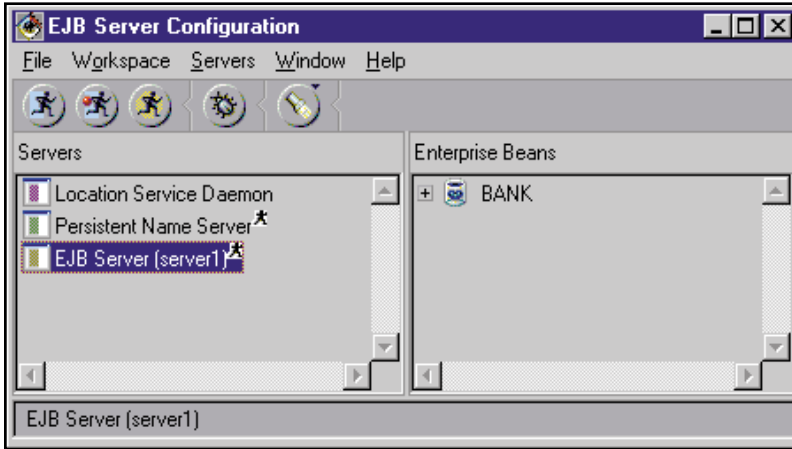


FIGURE 1 EJB Server Configuration window

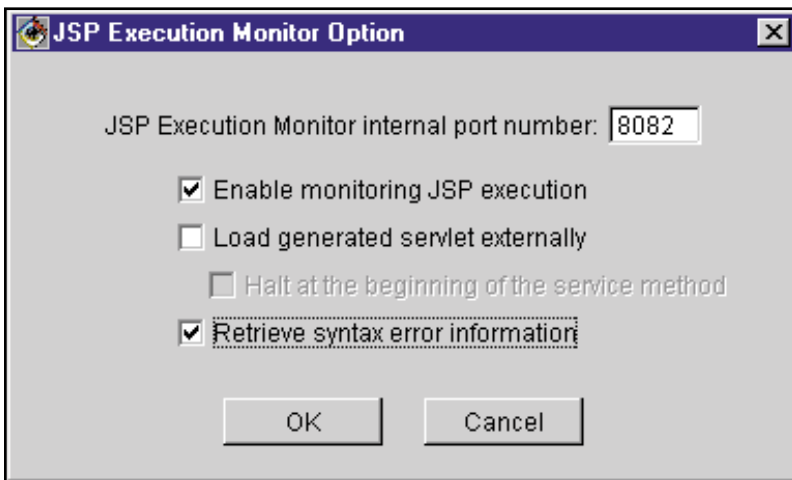


FIGURE 2 Options for JSP Execution Monitor

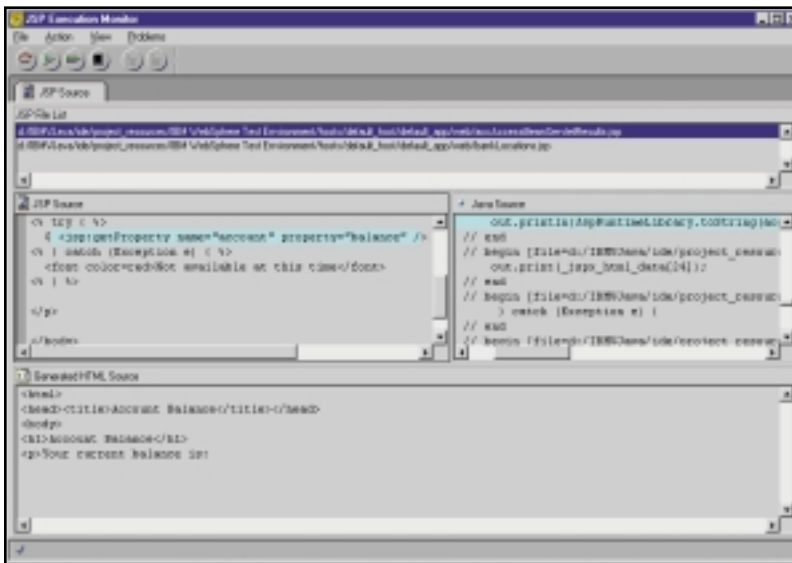


FIGURE 3 JSP Execution Monitor

- Integrated Debugger:** As with any program running inside VisualAge for Java, your Web applications can also make use of the VisualAge for Java integrated Debugger. This tool allows you to debug your servlets, EJBs and JavaBeans just as you would in any other application, including setting breakpoints and setting up watches on variables. Perhaps the biggest bonus comes with the incremental compiler built into the VisualAge for Java IDE. When you find a bug, you can fix it and have it compiled directly into the running code without stopping and restarting the Web or EJB servers! This feature alone can save plenty of time during development.

- JSP Execution Monitor:** Allows you to debug JSP pages within the IDE. Figure 2 shows the JSP Execution Monitor options, which allow you to turn JSP page debugging on or off and set which options to use. The highlighted option, "Retrieve syntax error information," allows you to debug Java syntax errors in the JSP page.

Figure 3 shows the JSP Execution Monitor window during execution. In the top pane are the JSP pages that have been run inside the Jmonitor. When these pages are initially compiled, the two middle frames show, respectively, the original JSP page source and the servlet generated from that source. As you step through the page using the toolbar, the current source is highlighted in both panes, and the HTML output from this source is displayed in the bottom pane. With familiar debugging features like breakpoints and step-through, the JSP Execution Monitor allows you to rapidly diagnose and fix errors in your JSP pages, whether they're Java coding errors or incorrect output from JavaBeans used by the JSP page.

• • •

Next month we'll provide a detailed tutorial that walks you through the steps involved in developing a Web application using VisualAge for Java and WebSphere Studio. ☛

**AUTHOR BIOS**

Anita Huang is currently working on IBM's WebSphere Developer Domain site, providing in-depth samples and tutorials that incorporate the WebSphere software platform for e-business. Previously, she worked on the VisualAge for Java Information Development team, focusing primarily on componentry to build enterprise applications.

Tim deBoer currently develops tools to build applications that run on WebSphere Application Server. He previously worked with the VisualAge for Java Technical Support group, providing support to enterprise developers working with VisualAge for Java.

[anitah@ca.ibm.com](mailto:anitah@ca.ibm.com)

[deboer@ca.ibm.com](mailto:deboer@ca.ibm.com)

# Understanding Workflow

WRITTEN BY BOBBY WOOLF



The Java 2 Platform, Enterprise Edition (J2EE), especially its Enterprise JavaBeans technology, provides an industry standard for the development of distributed enterprise applications. EJB helps solve a major problem: providing distributed access to persistent data. But it doesn't solve a related problem: modeling the business processes that applications use to access and manipulate that data.

Workflow helps solve the problem of modeling and implementing business processes within enterprise applications. It's just as important a part of enterprise computing as data persistence and distribution. EJB models behavior at the object level and limited interactions with any one client. Workflow models behavior across objects, applications and even systems, coordinating multiple clients while externalizing the processes from the code so they're easier to understand, change and manage.

In this article I'll provide an introduction to workflow concepts and how they relate to developing J2EE-style systems. I'll adhere as much as possible to the concepts described by the Workflow Management Coalition (WfMC) and use terms defined in their glossary. However, because workflow standards are still being developed and its concepts can be as much opinion as fact, I'll also describe workflow in terms of the Java-based workflow automation software I use – the Verve process engine. In describing workflow I'll explain how it relates to other major parts of your system, the components of workflow and two major styles for using workflow. With this information you'll be prepared to evaluate how you should use workflow as part of your J2EE systems.

## An introduction to workflow and workflow management systems

### What Is Workflow?

The WfMC describes workflow as the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action according to a set of procedural rules. As a simple example, let's think about the basics of processing an insurance claim. Here's the gist of how the claims process works:

1. Fill out the claim form.
2. Approve or deny the claim.
3. If approved, send the insured a check.
4. If denied, send the insured a rejection letter.

This simple example could easily be enhanced to handle improperly filled-out forms,

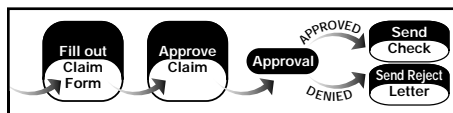


FIGURE 1 Insurance claim activity diagram

fraudulent claims and so forth. The activity diagram for this workflow is shown in Figure 1.

This workflow is a process – in this case one for processing an insurance claim. It contains four activities, each a task to be performed, each potentially performed by a different person. The workflow manages a set of information passed between the tasks: namely, the claim form. Besides the activities, the workflow also contains a decision point that splits the workflow into two parallel branches – approved and denied – and decides which branch to follow. Notice that it specifies what order the activities should be performed in, but not what work should be done during each activity. Workflow is more concerned with linking the activities together than with what work any particular activity does.

The first function we've performed here is to model the business process. That may seem like a trivial accomplishment for this simple example, but the requirements-gathering and mod-

eling can be the most difficult part of a workflow effort. The second function is a question of how to enact the workflow. Before computers, workflows were enacted by passing a file folder of papers from one person's desk to another. Today we use computer systems to store the documents electronically, but we still need a way to manage the process of passing the document from one person's computer to the next.

### Need for Tool Support

The WfMC defines a workflow management system (WfMS) as a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, and is then able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications. A WfMS manages a workflow in two ways:

1. **Process definition**
2. **Process enactment**



Process definition is the act of “coding” the workflow, defining it in such a way as to describe what it will do when it runs. Depending on the WfMS, defining can be implemented through a declarative coding model or through a visual programming model (which nonprogrammers find easier to use). Either way, the resulting workflow model is a data structure that can be stored in XML or any other data format of choice. Process enactment is the act of running a process definition, much the way bytecodes are run by the virtual machine. I’ll discuss the details of enactment shortly.

Workflow is best handled by embedding a separate WfMS tool within your application. The question of why your application needs a separate workflow tool is similar to asking why your application needs a separate database management system. Back when applications ran on mainframe computers and didn’t share data, each application contained its own code to manage its data. But with the need to share data between applications, distribute it across networks and manage overhead issues like concurrency and security, DBMSs evolved. They help alleviate the need for applications to manage their own data. Similarly, workflow management systems help alleviate the need for applications to manage their business processes. The application can then delegate its business processes to the workflow engine, allowing the application to focus on using the business processes rather than on implementing them.

### Role in Application

A workflow management system is neither a database management system nor an application server, although the three are frequently used together. Figure 2 shows how a WfMS fits into a typical system architecture.

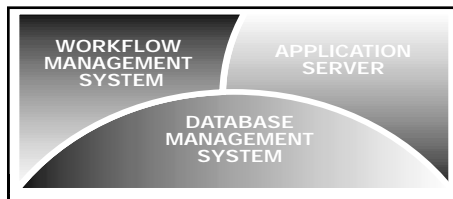


FIGURE 2 Typical system architecture

The application server manages running applications and provides clients access to those applications through understood APIs. The server doesn’t define the application, but it does store, execute and provide access to it. The problem with an application server is that it doesn’t know how to coordinate a workflow. It allows a single client to access an application and coordinates several clients accessing an application, each within its own session. Workflow cuts across these sessions, specifying a series of such sessions – requiring that when one session ends, others must be scheduled to begin, and performing work that occurs outside the context of any client sessions.

EJB wasn’t designed to provide workflow functionality. Entity beans are persistent and

transactional, but they don’t manage the session state or process, only domain object behavior. Session beans manage small bits of process, but only for a single client/server session (usually in a single thread), and provide poor support for transactions and persistence. If the application crashes or is shut down, entity beans preserve the state, but the session beans don’t remember what they were doing when they stopped. A session bean isn’t designed to coordinate a series of transactions coordinating multiple clients through a lengthy process during which the system could crash and restart.

A DBMS provides transactions and persistence (which supports the container’s entity bean implementation), but it doesn’t have a good way to tie together a common series of transactions as a workflow. Applications frequently need to perform a series of transactions, one after another, to implement a process. Because the DBMS doesn’t help manage this series of transactions, the responsibility falls on the application. But because the application session management has poor persistence and transactions, it’s not well suited for remembering which transactions have been completed so far and which still need to be run. Furthermore, application code that manages such transactions tends to be difficult to understand and maintain, so the processes they implement become buried and lost.

This is where a workflow management system comes in. It should be persistent and transactional (often by being implemented on top of a database management system, or perhaps as an EJB application), simplify modeling processes separately from the code that implements them, and make certain that when each transaction in a process completes, the next transaction begins. This frees the application from these concerns and allows it to concentrate on modeling the domain that the DBMS stores and that the WfMS manipulates.

### Workflow Enactment

Previously I mentioned that a process is defined, then enacted. Enactment is a little more complicated than simply running the workflow. Each separate enactment is represented by a work item. The WfMC defines a work item as a representation of the work to be processed (by a workflow participant) in the context of an activity within a process instance. When a process is enacted, the WfMS creates a work item to represent that particular enactment of that particular process. In this way, when a process is run multiple times (by multiple users or repeatedly by a single user), each separate run is represented by a work item.

When a WfMS enacts a process, it enacts each of the process’s activities in the order defined by the process. Just as enacting a process creates a work item to represent that enactment, enacting an activity creates a work item to represent the execution of that activity. If a particular activity is enacted several times, such as in a process loop, each enactment creates a separate work item.

An activity can be automated or manual. The work item for an automated activity is managed automatically by the workflow management system. For example, when a process work item is created, the WfMS automatically manages the work item by enacting the process’s activities. The work item for a manual activity must be managed by an entity external to the workflow management system. Such an entity is usually a person – a user of the application – which the WfMC calls a *participant* (something I’ll discuss later under Organizational Knowledge).

Manual work items are queued up on worklists. Each participant and organizational role (discussed below) has its own worklist. The items on a particular worklist represent the work that is available for the worklist owner to perform. If a worklist becomes too large, this indicates that the workflows are producing work requests faster than the worklist owner can perform them. A worklist is associated with an owner, not a workflow, so a single worklist often gets work items added to it by several different workflows.

## Workflow Components

Now that we know what a workflow is, how does it fit into a WfMS? And how does it interface with the rest of the application? Workflow consists of three parts that work together:

1. **Organizational knowledge**
2. **Domain knowledge**
3. **Process knowledge**

The relationship of these three parts is shown in Figure 3.

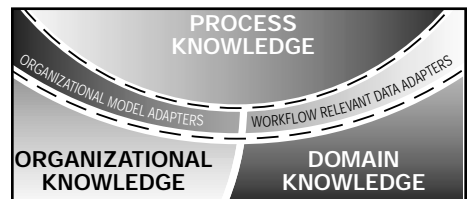


FIGURE 3 Workflow components

### Organizational Knowledge

*Organizational knowledge* is the set of users of the system and the groups they’re in. Why is this important to the workflow? Because the activities of a workflow are performed by people, so the workflow management system needs to know which people are allowed to perform what activities. Each manual activity is assigned to an organizational role – a description of the person within the organization who should perform this work. The set of people within a role is defined by users’ permissions and their interests and responsibilities within the organization, and the intent of the workflow developer. If an activity should be performed by a particular person, the role will describe just that person. The profiles we might define for the insurance claim example are shown in Table 1.

Each workflow user is represented as a participant – someone (or something) capable of per-

INSURANCE CLAIM PROCESS ACTIVITY	ORGANIZATIONAL ROLE
Fill out claim form	Customer
Approve claim	Adjuster
Send check	Secretary
Send reject letter	Secretary

TABLE 1 Profiles for workflow activities

forming work defined by an activity. Which activities a participant can perform depend on which roles the participant is a member of. If the participant is a member of a particular role, and an activity is assigned to that role, then – when that activity is enacted and a work item is created – the participant is allowed to perform that work item. If the participant weren't a member of that role, he or she wouldn't be allowed to perform the work item.

How does the WfMS know who the participants and roles are, and how they fit together? The WfMS accesses this organizational knowledge through an organizational model adapter. The model contains the organization entities (participants and roles) and their relationships. It gets its data from an external database, usually the databases that the enterprise already uses to model its employees and other users, such as LDAP.

### Domain Knowledge

*Domain knowledge* is the business domain that the application models. In our example the domain is insurance – specifically, claims processing. The WfMC distinguishes between application data – domain data that the workflow management system never uses – and workflow relevant data (WfRD) – domain data that the WfMS must be able to access. Most domain data is application data, but what we're interested in here is (as its name implies) workflow-relevant data.

A workflow is surprisingly unaware of and uninterested in most of what's going on in the domain. This is because a particular activity isn't much concerned with what work it represents, only that whatever work it represents is done when it needs to be done. The WfMS tells the application to "do this work now" and the application does it; it's up to the application to decide what it means exactly to do the work. So while the appli-

cation typically needs lots of domain data to perform its work, the workflows tend not to be interested. However, workflows are interested in some domain data, especially to help make decisions within the workflow.

Using our insurance example, after approving or denying the claim, the workflow then needs to decide whether to send a check or a rejection letter. How does it decide? The approval activity should have modified the claim object to set an approved flag. (It may also set an "evaluated by" field so we know which adjuster approved or denied the claim, but the workflow isn't interested in that.) The workflow will look at this flag on the claim to determine which activity to perform next. When the workflow accesses the claim as workflow-relevant data, it will typically ignore the multitude of fields and relationships having to do with who submitted the claim, the specifics of the claim and so forth. The WfRD will look at the claim as nothing more than a big approval flag container. The workflow accesses its workflow-relevant data through a workflow-relevant data adapter. The adapter gathers the data from within the domain and presents it to the workflow in a simple way that's just what the workflow needs. A single workflow may use several different adapters to access different sets of data, and multiple workflows that want the same data presented in the same way can share the same adapter code (although they probably won't be able to share the same adapter instances).

### Process Knowledge

*Process knowledge* is the set of process definitions. These are the workflows that the system knows how to run. What's really interesting here isn't so much what process knowledge is, but what it isn't. Our insurance workflow example revolves around the insurance claim. The claim is created by the first activity, and is then used by all the subsequent activities. It's difficult to imagine an activity for this workflow that wouldn't somehow use the claim to do its work. Yet the workflow doesn't contain the claim. The claim object is domain knowledge, not process knowledge. The workflow does contain a reference to the claim, a key or handle that uniquely identifies the claim within the application. This way, for example, when an adjuster gets a work item to approve a claim, the application knows which claim to present to him or her for approval.

Similarly, when the workflow assigns the approval activity to the adjuster role, the workflow has no idea what the adjuster role really means or who within the organization is allowed to perform adjuster tasks. When the application asks the WfMS what work is available for a particular user, the WfMS runs that user's participant through the organizational model adapter(s) to determine what roles he or she fulfills. It then finds the worklists for those roles and tells the application that the work items on those lists are available for that user.

This separation of responsibilities can be confusing at first, but it's ultimately clean and powerful, and one of the strongest advantages of using a workflow management system. Although the WfMS has its fingers into lots of organizational and domain knowledge, it really separates the workflows from that knowledge. Then the workflows can focus on what work needs to be done and what sorts of people will do it, but workflows don't focus on the specific people who will do it or how they'll do it.

This separation allows the workflow designer to work fairly independently of the application designer and the LDAP administrator. It focuses the workflow designer on the work to be done and away from *how* it will be done. It allows enterprises to make major changes to their business processes while minimizing the impact on the applications that enable those processes.

### Workflow Styles

In my work with designing workflows I've discovered two distinct approaches to using workflow:

1. **User-centric**
2. **Automation-centric**

A particular workflow can use either approach, or a combination of both. In practice, a workflow may be 100% user-centric, but it's rarely 100% automation-centric. Even the most automation-centric architecture still needs some small portion that's user-centric – as a last defense for error handling, if nothing else. In other words, it's difficult and not very desirable to automate everything.

# ANNOUNCING...

## Wireless Developer's Journal

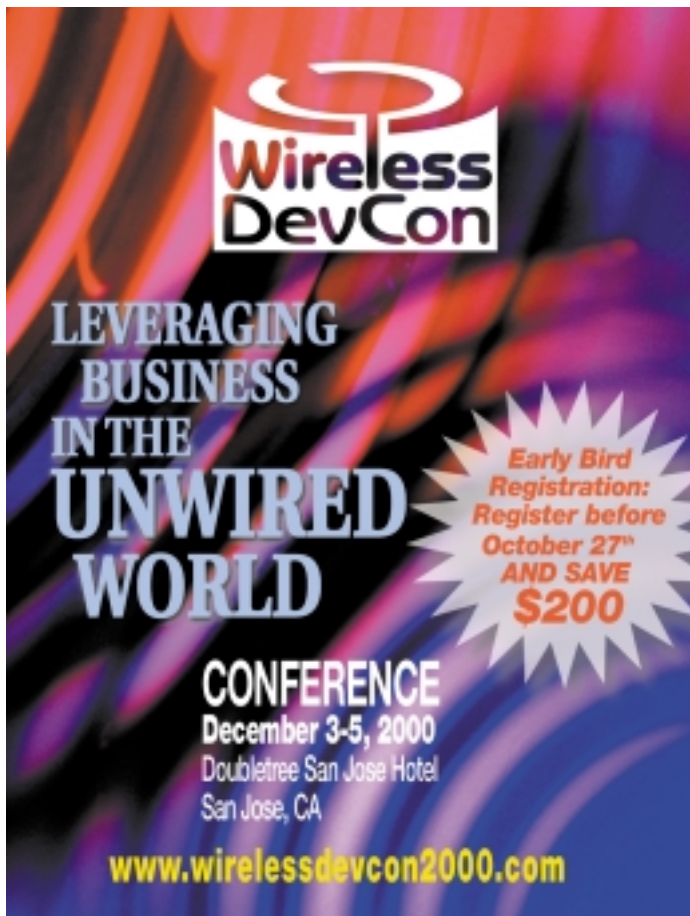
SYS-CON Media, the world's leading publisher of Internet technology magazines for developers, software architects and e-commerce professionals, becomes the first to serve the rapidly growing wireless application development community!



Look for it on your newsstand and worldwide in September!



[www.wireless-journal.com](http://www.wireless-journal.com)



**Wireless DevCon**

**LEVERAGING BUSINESS IN THE UNWIRED WORLD**

**CONFERENCE**  
December 3-5, 2000  
Doubletree San Jose Hotel  
San Jose, CA

**Early Bird Registration: Register before October 27<sup>th</sup> AND SAVE \$200**

[www.wirelessdevcon2000.com](http://www.wirelessdevcon2000.com)

### User-centric

This is the classic workflow approach. A person performs the work for each manual activity. The person sees the work item on a worklist and performs the work described by that work item. The person will typically interact with the application and the WfMS through a GUI (either in an AWT/Swing-style native window or through an HTML Web browser). These systems are relatively simple to build. The application developers have to implement the relevant GUIs for the users and add the code that lets the GUIs interact not only with the domain but also with the WfMS through its API. These GUIs then become a client part of the application.

### Automation-centric

This is the approach for automating workflow so that large amounts of work can be performed with a minimum of human intervention. The work for a manual activity is performed (whenever possible) by an automated system that's external to the WfMS and probably external to the application as well. The external systems interact with the WfMS through an API. For example, in our insurance claim example the "send check" and "send reject letter" activities could probably be automated by systems that print and mail the documents.

Systems that automate activities tend to be more complex than user-centric ones because of the difficulty in interfacing the WfMS to the external systems that do the work. The WfMS queues the work on worklists, just as it would for people. But whereas people have GUIs that give them access to those worklists, a typical out-of-the-box back-end system has no idea how to interface to worklists. They can use the same WfMS APIs that the GUIs use, but somebody has to write the code to tie together the WfMS APIs with the back-end system APIs.

This work can be simplified somewhat by using a messaging system, such as one that implements the Java Message Service API. This way, WfMS work requests can be queued as messages and the messaging system then has to worry about getting the back-end system to perform the messages. This also allows other systems besides the WfMS to make requests of the back-end systems in an asynchronous, persistent, transactional way.

The issue is then interfacing the WfMS API to a messaging API. As this interface code moves work requests between the worklist queues and the messaging queues, it has to avoid losing or duplicating any of the requests. Ideally, moving the requests should be performed transactionally, which requires a two-phase (distributed) transaction between the WfMS and the messaging system. Many such systems (both workflow and messaging) don't support external distributed transactions at this time. Likewise, a work request will often produce results data that needs to be stored in the database before the request is considered complete. This involves a three-way distributed transaction between the messaging, database and workflow systems.

### Conclusion

We've now seen the importance of workflow and how it relates to the rest of our application, and basic workflow concepts. Workflow models business processes, something that applications can't do well and that databases can't do at all. A workflow management system separates the business process from the applications that hook into the business processes and manages the execution of those processes. It separates the process from the organization that performs the work and the domain in which the work is performed. Finally, a workflow management system prepares work to be performed by human users or automated systems, or a combination of both.

Much like database management systems 10 or 20 years ago, the workflow management systems' time has come. They are rapidly becoming an indispensable part of an enterprise application architecture. 🍌

### AUTHOR BIO

Bobby Woolf is a senior architect at GemStone Systems ([www.gemstone.com](http://www.gemstone.com)) and a member of their Advanced Application Architecture Team ([www.gemstone.com/javasuccess](http://www.gemstone.com/javasuccess)), where he specializes in workflow and the application of the Verve process engine ([www.verveinc.com](http://www.verveinc.com)) within J2EE applications.

[bobbyw@gemstone.com](mailto:bobbyw@gemstone.com) and [woolf@acm.org](mailto:woolf@acm.org)




**GET YOUR OWN!**


Special TechWave '99 Issue!  
PowerBuilder Journal  
WEB DEVELOPMENT DOES THE TANGO  
DEVELOPER'S JOURNAL  
BUILDING ENTERPRISE  
COLDFUSION  
XML: IT'S THE 'X' THAT MATTERS  
XML JOURNAL

**Call and Subscribe today!**  
Get Your Own Subscription to the Finest Technical Journals in the Industry!  
1-800-513-7111 [www.sys-con.com](http://www.sys-con.com)


## ProtoView Launches PowerChart Java Component

(Cranbury, NJ) – ProtoView Development, a leading provider of component technology, has released version 1.0 of its new Java charting component. PowerChart is a rich and  robust set of integrated JavaBeans that provides 2D and 3D rendering of a wide variety of graph types to any Java application. Further information and a comprehensive new feature list can be found on the ProtoView Web site at [www.protoview.com](http://www.protoview.com).


## Quest Software in Alliance with Siebel Systems


(Irvine, CA) – Quest Software, Inc., a leading supplier of application and information availability software, has announced an alliance with Siebel Systems, Inc., a provider of e-business application software, to provide tightly integrated performance, monitoring and high-availability solutions for deployment with Siebel e-business applications for sales, marketing and customer relationship management.  [www.quest.com](http://www.quest.com)

## AbriaSoft Announces MySQL Database Software

(Fremont, CA) – AbriaSoft has announced the launch of Abria MySQL Lite, which reduces the cumbersome process of installing and configuring MySQL and Apache and offers an integrated, turnkey install of RedHat Package Modules for MySQL 3.22, Apache Web Server, PHP3 and Perl. Abria MySQL Lite is available for unrestricted free download from the company's Web site.  [www.abriasoft.com](http://www.abriasoft.com)

## Websprocket Releases JEMini

(Sunnyvale, CA) – Websprocket, a provider of software and software tools for networked devices, has released JEMini, a free, open-source Java programming language for 

embedded networked systems. JEMini includes an array of foundation libraries found in commercial Java distribution and libraries suitable for device drivers, embedded system design and embedded system design automation.  [www.websprocket.com](http://www.websprocket.com)


## Introducing RoboHELP Office 9.0

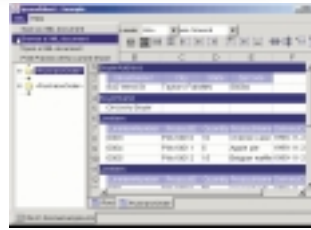
(San Diego, CA) – RoboHELP Office 9.0 from eHelp Corporation offers new and improved features to support the creation of integrated help and user assistance for Web sites and Web server-based applications.  [www.ehelp.com/robohelp](http://www.ehelp.com/robohelp)

## COOL:Joe 1.1, v. 1.1, on the Market

(Islandia, NY) – Computer Associates International, Inc.'s new version of COOL:Joe is J2EE compliant, ensuring complete support for the latest standards. New features include a task advisor, Web server support, a context-sensitive editor, enhanced smart expansion and integration with leading configuration management tools.  [www.ca.com](http://www.ca.com)

## JSmartGrid for the Web from Eliad

(Paris, France) – Eliad Technologies, Inc., has released JSmartGrid 1.0, the smallest Swing grid 



component (162 KB) specifically designed for the Java 2 Platform v1.3.0. JSmartGrid's extensive API with its scalable presentation abilities simplifies design for Web, database and B2B applications and gives developers more flexibility in displaying all types of data, including XML documents. A free evaluation copy can be downloaded from the Eliad Web site. [www.eliad.com](http://www.eliad.com)

## GemStone Teams with RSW Software

(Beaverton, OR) – GemStone Systems, Inc., and RSW Software, Inc., have formed a strategic alliance that provides tools for load testing and optimizing the performance of enterprise applications from development through deployment. As a result of the partnership, GemStone and its customers can use RSW's e-TEST suite and EJB-test to analyze the scalability and functionality of enterprise-class applications built on the GemStone/J 4.0 application server using Java 2 Platform, Enterprise Edition. [www.gemstone.com](http://www.gemstone.com)

## Allaire Licenses Sun's J2EE

(Newton, MA) – Allaire Corporation, in a strategic alliance with Sun Microsystems, Inc., will formally license Sun's Java 2 Platform, Enterprise Edition.

Allaire is currently incorporating the J2EE technology into its new Java Application Server, JRun 3.0, announced at JavaOne. JRun 3.0 is a full, clean-room implementation of the J2EE specifications and a key component of the Allaire Business Platform.  [www.allaire.com](http://www.allaire.com)

## Macromedia Unleashes Flash 5

(San Francisco, CA) – Flash 5, the latest version of the product for producing high-impact Web experiences, will soon be available from Macromedia. Flash 5 includes new creative tools, "seamless integration" with Macromedia FreeHand and Fireworks, ActionScript, a JavaScript-like language for creating interactivity, and support for XML data interchange.   A new Bezier pen tool complements the product's natural drawing tools and enables traditional illustrators accustomed to technical drawing tools to work easily within the authoring environment. [www.macromedia.com](http://www.macromedia.com)

## WebGain Studio Available, New Version of WebGain's TopLink for BEA WebLogic

(Santa Clara, CA) – WebGain, Inc., is shipping its flagship product, WebGain Studio, an integrated e-commerce development environment that enables developers to rapidly build Java component-based e-commerce applications. WebGain Studio builds on industry-leading tools and technologies acquired from Symantec, Tendril Software and The Object People, and licensed from Macromedia.  [www.webgain.com](http://www.webgain.com)

WebGain, Inc., is also introducing WebGain TopLink v 2.5 for BEA Systems application servers, BEA WebLogic Server 5.1 and BEA WebLogic Enterprise 5.1.

Compatible with Java 2 Platform, Enterprise Edition application servers, the product frees Java developers from the rigors of integrating object-oriented applications with non-object data sources such as relational databases. By

integrating TopLink with BEA WebLogic Server, developers can efficiently build components for application servers that run on Java, while significantly cutting application development time and expense. [www.webgain.com](http://www.webgain.com)

## BEA First Independent Company to Achieve J2EE Certification

(San Jose, CA) – BEA Systems, Inc.'s WebLogic family of application servers has successfully completed Sun Microsystems' J2EE certification, the industry's only J2EE-comprehensive test suite for assessing compliance with the J2EE specifications.



This milestone validates BEA's continuing and rigorous adherence to J2EE. ☛

[www.bea.com](http://www.bea.com)

## JDJ Editor-in-Chief Joins Verge Board

(Boulder, CO) – Verge Technologies Group, Inc., a J2EE consulting and hosting firm specializing in Enterprise JavaBeans, has named Sean Rhody, editor-in-chief of **Java Developer's Journal**, to its board of advisors.

A former CIO and successful consultant in the B2B NetMarket industry responsible for the architecture of several leading B2B exchange sites, Rhody is a respected industry expert and frequent speaker at Java events and conferences.



Jason Westra, CTO of Verge and editor of **JDJ's EJB Home** column, stated: "We are excited to have Sean's skills and technical vision behind us as we continue to lead the industry in J2EE ventures...."

"I think Verge brings the depth of knowledge and commitment to excellence around J2EE that will allow the company to differentiate itself from the competition. Verge's ejip.net offering, the first of its kind in the industry, shows the depth of thinking and the versatility of the organization," said Rhody. ☛

[www.vergecorp.com](http://www.vergecorp.com)

## Holt New Head of Secant Technologies

(Cleveland, OH) – Former Computer Associates senior vice president and general manager Jim Holt has been named Secant's new president. Holt takes the reins at



Secant with more than 20 years in leadership positions at innovative technology and system integration companies and an impressive record of accomplishment in building global organizations.

In his new role Holt will manage Secant's worldwide operations from the company's headquarters in Cleveland. He



will work closely with CEO John Schwalm to provide overall leadership, build operations, penetrate vertical markets and identify new opportunities for the company. ☛

[www.secant.com](http://www.secant.com)

## ParaSoft Partners with CodeMarket

(Monrovia, CA / New City, NY) – ParaSoft, which provides software error prevention and error detection solutions, has announced a partnership with CodeMarket, a global software development network where software developers and development managers can find and purchase both free-lance development work and ready-



to-run Java components. The partnership was developed to bring more certainty to the marketplace for outsourced software component development. ParaSoft's Jtest will be the standard tool by which all components outsourced or purchased through CodeMarket will be tested. ☛

[www.parasoft.com](http://www.parasoft.com) [www.codemarket.net](http://www.codemarket.net)

# JSmartGrid

# 1.0



by Eliad Technologies, Inc.

REVIEWED BY BRUNO Y. DECAUDIN



**AUTHOR BIO**

Bruno Y. Decaudin has accumulated 20 years' experience in the computer field in Europe and the U.S., where he settled down. A graduate of the EDHEC in France, Bruno worked for major corporations and consulting firms before creating his own company in 1991, focusing first on application development and then on network design and Internet development.

[bruno@sys-con.com](mailto:bruno@sys-con.com)

Eliad Technologies is a young company with a mission: to bring data visualization components to the Java community in a very small footprint (i.e., 162KB).

The fruit of their labor is JSmartGrid, the smallest Swing grid component designed specifically for the Java 2 Platform version 1.3.0. JSmartGrid can visualize data in tables, grids and spreadsheets. It was introduced in June at JavaOne 2000 and at the XML DevCon 2000 in New York City.

A product evaluation CD-ROM was also distributed at both shows, but Eliad Technologies provides an updated downloadable version of 1.0 on their Web site.

## Features

The JSmartGrid, a Swing component and a bean, installs easily on every IDE loading bean such as JBuilder 3.5 and Forté. The only other requirement is the JDK Java 2 Platform version 1.3.0 or JRE Java 2 Platform version 1.3.0; they can be downloaded from Sun Microsystems' Web site, [www.sun.com](http://www.sun.com).

JSmartGrid offers a wide range of features that make their product quite attractive. This 100% Java Swing component allows the display of data as text, numbers, graphics and images in GIF or JPEG format, or as Swing components such as buttons and combo-boxes. Such flexibility enables you to replace your old static HTML tables with new dynamic Java tables that allow users to edit, reformat and sort data. You can't do that with lame HTML code! With JSmartGrid you have the flexibility

to define the content and the behavior of every individual cell in your spreadsheet. A unique feature is the ability to drag cells, rows and columns and to merge cells into bigger rectangular arrays called *spans*.

On the technical side JSmartGrid simplifies application design through its flexible models (data, span and style) and offers improvements in performance on the client side thanks to its integration with the Java 2 Platform v1.3.0. If you're familiar with the JTable Swing component from Sun, you shouldn't experience much difficulty getting started with this component. Last but not least is its ability to connect to data through JDBC and to display XML documents.

## Impressions

JSmartGrid is a good, stable product proving once more that a small company can compete in the big leagues. Their product is innovative and useful since they provide features that competitive products lack. What impressed me most is the quality of the documentation, which is exhaustive. Every class is documented in the format used by Sun and O'Reilly & Associates with complete hyperlinks in a tree fashion. In addition, you can download examples, documentation and tutorials from their Web site, which should make learning the product much easier. ☺

JSmartGrid 1.0, Eliad Technologies, Inc.  
19925 Stevens Creek Blvd, Cupertino, CA 95014

Web: [www.eliad.com](http://www.eliad.com) • Phone: 408 973-7216 • E-mail: [info@eliad.com](mailto:info@eliad.com)

Test Environment: Pentium II 350MHz, 128MB RAM, 26GB disk drive,  
Windows NT Workstation 4.0 SP6 • Pricing: \$179.00

