SUN AND IBM: ARE THERE ISSUES?

JAVA™ DEVELOPER'S JOURNAL

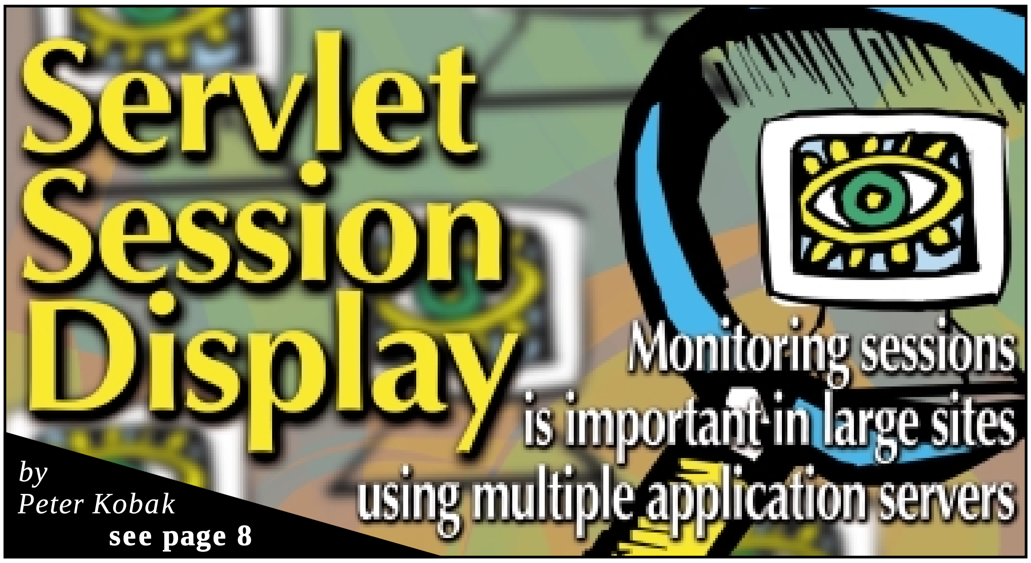
The World's Leading Java Resource

Volume: 5 Issue: 2, February 2000

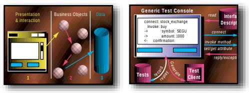
JAVADEVELOPERSJOURNAL.COM

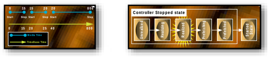
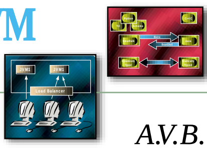
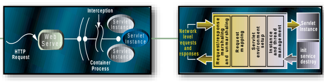
Announcing...
JavaCON 2000
See page 77 for details

SYS-CON PUBLICATIONS



Servlet Session Display
Monitoring sessions is important in large sites using multiple application servers
by Peter Kobak
see page 8



BEA

www.beasys.com

Protoview

www.protoview.com

KL Group

www.klgroup.com

JAVA DEVELOPERS JOURNAL

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
PRODUCTION EDITOR:	CHERYL VAN SISE
ASSOCIATE EDITOR:	NANCY VALENTINE
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

LINDEN DECARMO, RICK HIGHTOWER, THOMAS KERN, PETER KOBAK, ALEXANDRE LEMIEUX, GABOR LIPTAK, DAVID LYONS, JIM MILBERY, GEORGE PAOLINI, PAT PETERNOSTRO, TONY PATTON, SEAN RHODY, AJIT SAGAR, TODD SCALLAN, ROD SMITH, A.V.B. SUBRAHMANYAM, JASON WESTRA, ALAN WILLIAMSON

SUBSCRIPTIONS

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

SUBSCRIPTION HOTLINE: 800 513-7111

COVER PRICE: $4.99/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

PUBLISHER, PRESIDENT AND CEO:	FUAT A. KIRCAALI
VICE PRESIDENT, PRODUCTION:	JIM MORGAN
VICE PRESIDENT, MARKETING:	CARMEN GONZALEZ
CHIEF FINANCIAL OFFICER:	IGNACIO ARELLANO
ACCOUNTING MANAGER:	ELI HOROWITZ
CIRCULATION MANAGER:	MARY ANN MCBRIDE
ADVERTISING ACCOUNT MANAGERS:	ROBYN FORMA
	MEGAN RING
JDJSTORE.COM:	JACLYN REDMOND
ADVERTISING ASSISTANT:	CHRISTINE RUSSELL
GRAPHIC DESIGN INTERN:	AARATHI VENKATARAMAN
WEBMASTER:	ROBERT DIAMOND
WEB EDITOR:	BARD DEMA
WEB SERVICES CONSULTANT:	BRUNO Y. DECAUDIN
WEB SERVICES INTERN:	DIGANT B. DAVE
CUSTOMER SERVICE:	SIAN O'GORMAN
	ANN MARIE MILILLO
ONLINE CUSTOMER SERVICE:	AMANDA MOSKOWITZ

EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.
39 E. CENTRAL AVE., PEARL RIVER, NY 10965
TELEPHONE: 914 735-7300 FAX: 914 735-6547
SUBSCRIBE@SYS-CON.COM

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

© COPYRIGHT

Copyright © 1999 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
739 RIVER ROAD, NEW MILFORD NJ 07646-3048 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.

BPA INTERNATIONAL

SYS-CON PUBLICATIONS

SEAN RHODY, EDITOR-IN-CHIEF

To BEA or Not to BEA

Yech. I hate that title as much as you do, but it stuck in my brain and I can't get it out. Things are going on in the industry, and I think this is an appropriate time to cover them. We were at the Java Business Conference in December, covering what appeared to be more of a non-event than a true exposition. Probably the biggest disappointment was Sun's backing out of the standards process for Java. Alan Williamson's column goes into detail on that, expressing the feelings of many in the industry that were upset by this move. While the little guys were understandably upset, it's the big players that were really hurt. IBM in particular has made a large investment in Java, and has been looking to participate in the development of the language on more of a peer basis than this turn of events allows.

There have been other happenings in the field as well. Not one but two start-up EJB-based product companies have been acquired by larger corporations in the past few months. On the West Coast Ariba purchased Trading Dynamics for about $400M. And here in the East BEA purchased the Theory Center for about $100M. Trading Dynamics makes sophisticated auction software, which is EJB compliant and runs on WebLogic. The Theory Center makes a component development framework for WebLogic, and coincidentally has a set of vertical components that support a business-to-consumer type of auction.

Clearly, one of the hottest markets at the moment is exchange enablement. Exchanges, be they business to consumer, like eBay or Priceline.com, or business to business, such as Chematch.com or eSteel, are the focus of both the venture capital industry and the large-scale consulting practices. Egged on by the success of eBay, a large number of companies are competing to change the way consumers and businesses interact.

Several other significant events happened recently too. BEA and Warburg-Pinkus, which was an original investor in BEA and still owns a good part of it, combined to create a new, as yet unnamed company to develop software tools for the industry. The company's first act? To purchase VisualCafé from Symantec. I spoke with Joe Menard, president of BEA's E-Commerce Server division shortly after this was announced. They see this as a way to further integrate tools into J2EE – tools that will make it easier for everyone to develop distributed applications.

Interestingly enough, Persistence Software picked this moment to engage in a lawsuit with the Object People over Persistence's patents regarding object-relational mapping. I spoke with two Persistence VPs at the show, and the company is keen on protecting what it regards as a clear differentiator in the EJB marketplace.

Where's all of this headed? People have been telling the server companies two things lately – the products are what we want, but they're too hard to use. At the lower end you see products like ColdFusion and SilverStream taking away business from the EJB servers. These products are easy to use and fairly powerful in their own right, but they don't pack the wallop of a WebLogic, a WebSphere or a PowerTier.

Additionally, people – particularly start-ups – are looking for a much higher degree of vertical integration than previously offered. They don't want products, they want solutions, like Trading Dynamics.

It's going to come down to who offers the best set of tools and integration. IBM has a head start with hardware, software, database and development tools. BEA is quickly making up for lost ground and is buying everyone in sight. Persistence just announced an exchange package called Sold.

The first age of EJB is now over. In the beginning, there was plenty of room for everyone, and anyone could build a server. We're now in a consolidation phase. Expect to see plenty of work on integrating the presentation layer – JSP, JHTML and the like – into the business logic layer, and making it easy to work with via enhanced IDEs. I also expect there to be fewer server vendors this time next year. To BEA or not to BEA, that is the question.

sean@sys-con.com

AUTHOR BIO

Sean Rhody is the editor-in-chief of Java Developer's Journal. *He is also a principal consultant with Computer Sciences Corporation where he specialzes in application architecture – particularly distributed systems.*

Together Soft

www.togethersoft.com

WRITTEN BY GEORGE PAOLINI

Refining the Process

For many years I was the world's greatest parent. Then I had kids.

Before my kids were born, I knew all the answers to successful child-rearing. And as the self-appointed expert I was quick to impart my wisdom to friends and family. Things sure were simpler then, and best of all I never made mistakes.

It was while observing one of my precious progeny (the little tyke pouring gravy on the dog's head during the Thanksgiving dinner at Aunt Mimi's) that I had to face the fact that I had a lot to learn. Successful parenting, I came to realize, is a lot of trial and error and a whole lot harder in practice than in theory.

I think of this lesson in humility every time I pick up a trade magazine these days. In there, somewhere, I can easily find the latest unsolicited advice to Sun. Every industry pundit knows exactly what Sun should do to standardize Java; every executive with a vice president's title knows just what to do to make Java more "open." And if Sun would heed their advice, Java would be saved and we might just cure cancer in the process.

Maybe it's worth digging out the Baby Book and looking back four years, when Java was in its infancy. Let's frame the era. The Web is just starting to take off and there are a number of projects, all of them competing in the same space as Java: Smalltalk, Inferno and ActiveX. Some people actually believe VRML is the hottest thing going. Others are betting that Netscape can deliver a robust set of APIs on which to build network applications.

Zoom ahead four Internet years. Which one of those little toddlers has grown up to be class president? Well, let's look in the yearbook and see which can claim credit for:

- A platform with 2 million active programmers
- A book industry unto itself
- The default computing environment taught in college computer science classes
- A software environment used by virtually every Fortune 1000 company
- A software environment employed by technology vendors from smart cards to the data center

So how did we get from there to here? We did it by making lots of mistakes. Shepherding a technology, it turns out, has some things in common with parenting. It requires some give and take. It requires a lot of trial and error. It requires patience, tenacity and the unflappable belief that your "kid" has what it takes.

Along the way those mistakes led to a pretty unique process for building, evolving and disseminating a technology that began as a language, a VM and a few core class libraries, and it has matured to a full, robust platform for distributed, network computing.

To be sure, we didn't do it alone. It takes a village to raise a child, and it takes an entire industry to groom a technology as important as Java.

What began as an informal process for collaboration on the Java technology has been formalized in what we call the Java Community Process. Along the way, in four short years, the Java community has developed over 80 new interfaces.

All well and good, you say. But Java is grown up now and it's time to let go. Make it open.

The term *open* is bandied about these days and associated with lots of technologies. But what does it mean for a technology to be open? To my mind it must possess at least three qualities:

1. The specifications must be published.
2. The source code must be accessible.
3. There must be a process for allowing the community to innovate on the source base.

And to ensure that this technology has a practical purpose within the business world, we added a fourth requirement:

4. Any modifications to the source base must pass compatibility testing.

With these four steps we can ensure innovation while maintaining compatibility. Java meets these criteria, hands down, and that's why, to my mind, the technology has achieved such overwhelming success. It provides a stable base for the millions of developers such as you to innovate upon.

—continued on page 86

george.paolini@sun.com

AUTHOR BIO

George Paolini, vice president of marketing of Software Products and Platforms at Sun Microsystems, is responsible for managing all public relations, Internet, strategic and technical marketing branding, and trade shows and events within that business unit. George is also a member of the Java Developer's Journal *Editorial Board.*

Soft-
Wired

www.javamessaging.com/ibus

J D J   F E A T U R E

Servlet Session Display

Monitoring sessions is important in large sites using multiple application servers

WRITTEN BY PETER KOBAK

With the release of the Java 2 Platform, Enterprise Edition, Java-based Web application servers are gaining in popularity. Although application servers have been around for a few years, they forced programmers to be tied to a proprietary API. Support of J2EE by application server vendors standardizes the API we write to, easing training, staffing and support costs. Perhaps most important in our dynamic Web vendor environment, writing code to an industry standard reduces the huge risk a customer takes in choosing a particular vendor should that vendor disappear.

In this article I'll explain the basics of Java servlet sessions and why careful monitoring of your sessions will be important in a large site using multiple application servers. Then I'll present a simple utility to check your servlet sessions.

Application Servers

What is a Web "application server"? Although many full-fledged Web applications have been written using a Web server with cgi-bins, most are written using C or Perl. The cgi-bins typically contained display and business logic and talked to back-end processors and databases via proprietary interfaces (see Figure 1). This works fine for simple applications with low traffic, but when traffic increases or pages need to be redesigned, the situation gets ugly. Unless it was designed with saintlike attention to isolation and modularity, a page change, database change or business logic change ripples through all parts of the application. More important, as traffic goes up, more and more code must be dedicated to resource management: database connection management to keep the database from getting swamped, thread or process control to keep the Web server from thrashing, and load distribution to handle volume. Sound scary? Ask anyone who's had to maintain an older site for a few years using only a C compiler.

Application servers help reduce the housekeeping and management burdens, allowing developers to concentrate on the application itself. On the presentation layer a page layout mechanism enables the use of templates to insert data in dynamic pages. A transaction monitor allows you to manage the sequence of transactions, roll back across heterogeneous databases, use a connection pool and access a common API to address different databases. Business logic modularity is supported through formal mechanisms. Automatic caching is available for database resultsets and HTML pages. Across it all are management functions to perform load balancing and distribution among threads, to make full use of available processors and to allow for graceful degradation under stress. The most serious application servers also support load balancing or failover between servers.

Java-based application servers perform these functions using a standard set of APIs (see Figure 2). Instead of learning the quirks of your vendor's API, you can rely on information from multiple sources and have access to many resources – books, sites and newsgroups – not to mention the fine magazine you're reading now. You also avoid the sometimes deadly vendor lock. Using a proprietary system is like locking yourself in a jail cell; a standards-based system is similar, but you get to smuggle in a hacksaw.

Although four layers are shown in the application server in Figure 2, there's nothing stopping you from skipping layers or using a more complex pathway. For instance, you may find that for all their sophistication EJBs are overkill. You may choose to go directly from a servlet to a JDBC call. However, to get business-layer isolation you'll probably want to use a bean separate from servlet code. Also, it's not uncommon to have a chain of servlets to provide layered user data validation and transaction control.

Presentation Layer

At the presentation layer, user data validation and HTML page construction are accomplished with Java servlets and JavaServer Pages. Servlets and JSPs, or a chain of them, can be invoked. A servlet is a subclass of the HttpServlet interface. The application server invokes a method of your servlet through a mapping of URL nodes (see Figure 3 for an example). An HttpRequest object is passed to the invoke() method of your HttpServlet; HttpRequest contains everything that gets passed into a cgi-bin and more, but with easier access. The request is examined and processed, and your servlet returns a stream of HTML through another parameter object.

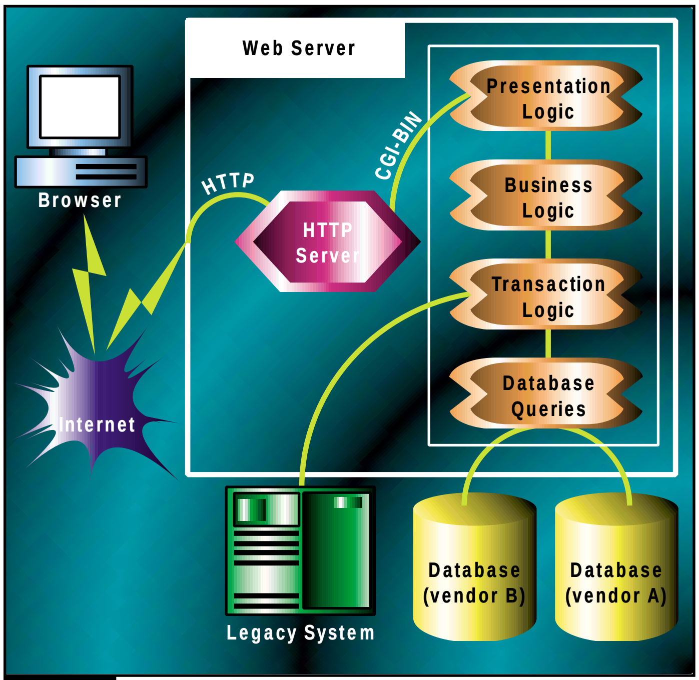


FIGURE 1 Web server application model

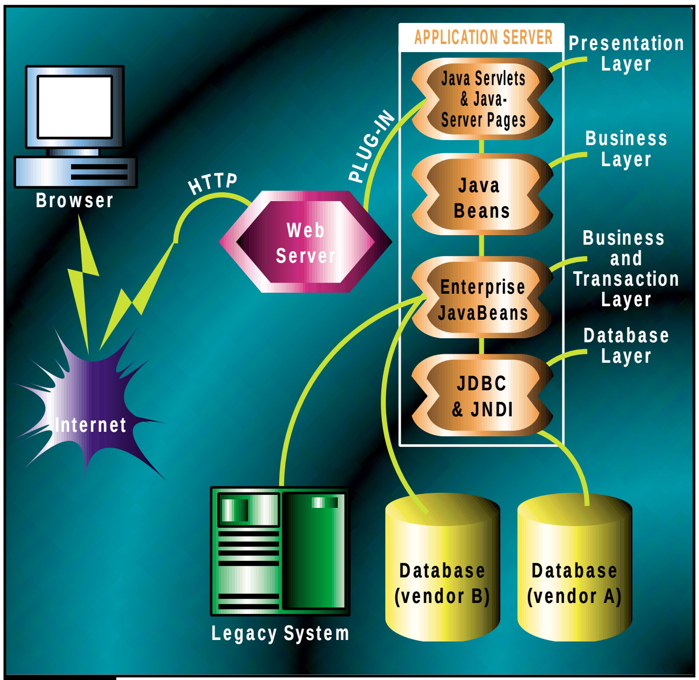


FIGURE 2 Java application server model

LIFESPAN	MEANING	USAGE
Page	Lives only during invocation of this JSP	Not too useful, but available if you love beans
Request	Lives during current HttpRequest	The bean can be used by other JSPs, but is lost once control returns from the initial invoke()
Session	Lives in the HttpSession; continues to live between requests	Used to store information about the user or the user's transaction; a good place to keep a shopping cart
Application	Lives forever (that is, until the application or server is rebooted)	Application-global information

TABLE 1 Bean lifetimes

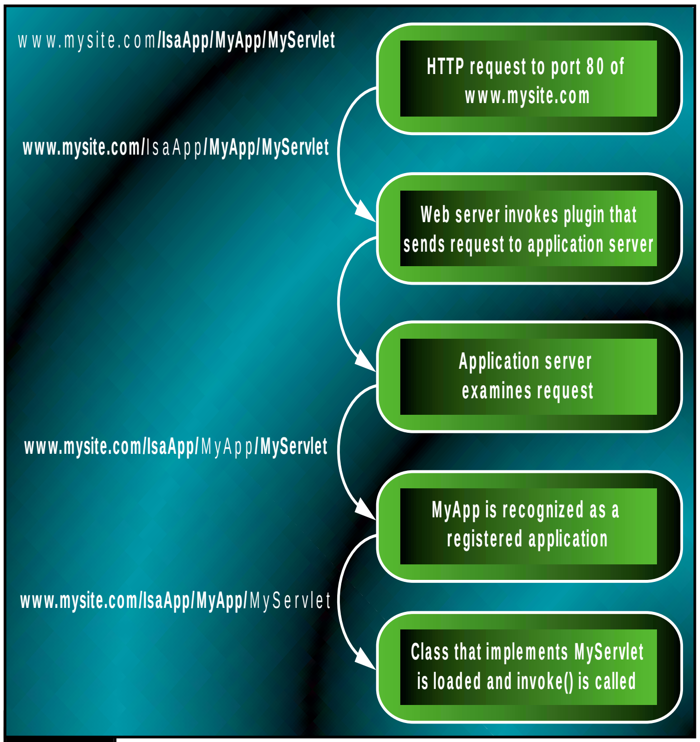

www.mysite.com/IsaApp/MyApp/MyServlet
HTTP request to port 80 of www.mysite.com
www.mysite.com/IsaApp/MyApp/MyServlet
Web server invokes plugin that sends request to application server
Application server examines request
www.mysite.com/IsaApp/MyApp/MyServlet
MyApp is recognized as a registered application
www.mysite.com/IsaApp/MyApp/MyServlet
Class that implements MyServlet is loaded and invoke() is called

FIGURE 3 URL decomposition

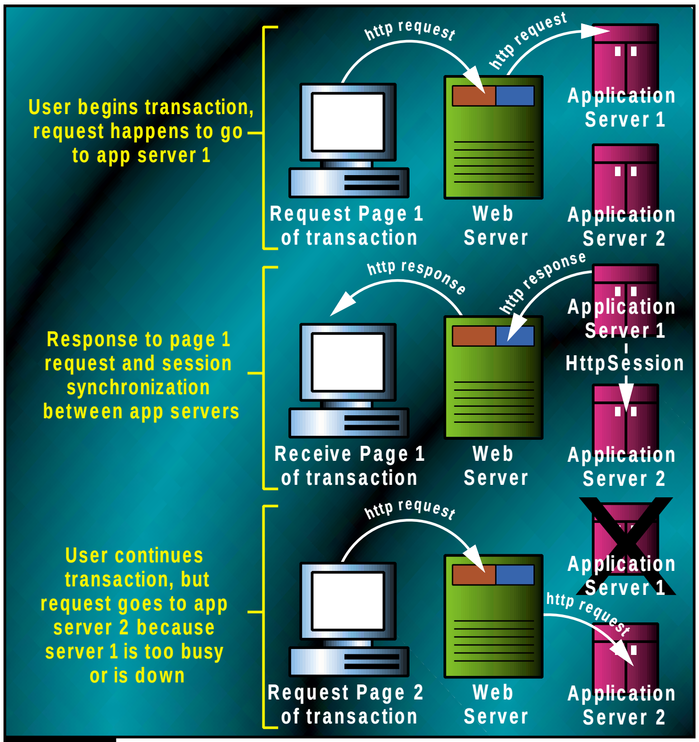

User begins transaction, request happens to go to app server 1
http request
http request
Request Page 1 of transaction
Web Server
Application Server 1
Application Server 2
Response to page 1 request and session synchronization between app servers
http response
http response
Receive Page 1 of transaction
Web Server
Application Server 1
HttpSession
Application Server 2
User continues transaction, but request goes to app server 2 because server 1 is too busy or is down
http request
http request
Request Page 2 of transaction
Web Server
Application Server 1
Application Server 2

FIGURE 4 Failover scenario

JSPs allow you to mix servlet code with HTML, which gives you a nice layout of static HTML with the dynamic bits placed as Java code. JSP source is compiled into an HTML file and a Java file, but the application server does it for you. Some servers even recompile on the fly, recognizing a change to JSP source and doing all the compiling and class reloading automatically (pretty slick). JSPs can do most things a servlet can, and in addition support JavaBeans.

You probably know about beans, but for the purpose of JSPs, the only important rule to follow is the coding pattern of using getProperty() and setProperty() methods – where "Property" is a bean property name that's usually just a field of the bean class. Since arbitrary Java code can be executed in a JSP, a bean can be used just by invoking it as a class in the usual way. Beyond this, though, there's support for beans. You can create and use a bean for a period that extends beyond the life of the current request (see Table 1). There's also a JSP syntax to set and get bean properties, although standard Java syntax works just as well.

Typically, the beans invoked from a JSP contain your business logic, and directly or indirectly invoke back-end systems and databases. Even if your logic is simple, you're better off minimizing the amount of Java in a JSP. Think of a JSP's Java code as performing just enough logic to invoke business beans. The beans do their work, then the JSP extracts bean properties to put data on the HTML page.

Sessions

An HttpSession object is part of the HttpRequest object passed to a servlet or JSP; HttpSession is the easy way to have data associated with a user (actually the user's browser). Behind the scenes, the application server uses cookies or URL encoding to indirectly store the data. This is probably nothing new to you; cookies or URL codes are the only way to store data on the browser so it gets returned to the server on subsequent HTTP requests. The friendliness of HttpSession is twofold. First, you don't have to think about cookies or URLs since the server does that dirty work for you. In fact, some servers will try to use cookies and if they can't (if they're not accepted by the browser, for instance), the server will attempt to use URL rewriting instead. Second, the application data is stored on the server, not the browser. The cookie data is merely a key to finding the session data. This gives you a good measure of security for your data as well as freeing you (mostly) from the space constraints of storing data directly in a cookie.

You can store an object in a session by invoking a method of HttpSession and giving the object a name. On the same or subsequent requests, invoking a get method with the object's name can retrieve the object. A servlet accesses the HttpSession directly. A JSP can also use the session directly but, as you probably guessed, a bean stored with a "session" lifetime is stored into and retrieved from the HttpSession for you.

Distributing Servlet Sessions

If you buy a big, serious Java application server, one of the features your money buys is load balancing and failover mechanisms. Within a server box, your app server package is expected to partition work among available resources as efficiently as possible. Among other things, that means clever distribution of work between available CPUs and pooling connections to back-end systems.

Distributing fine-grained work between servers is even more clever. However, if your application is supposed to work with a consistent HttpSession, the session must be maintained on multiple servers (see Figure 4). Application servers can have more than one policy for sharing sessions, but it's clear that if you need to have robust transactions (that is, user transactions that can survive if a server goes down) your HttpSessions must somehow be distributed to the peer server(s). Although it's technically possible for a server to implement distributed sessions without using serialization, the serialization mechanism native to Java is the obvious and simplest way for a server to distribute the data from the objects living in the session. Even though the app server does the hard work of enforcing session distribution policies, you have to make sure all the objects you store into HttpSession are serializable.

Serializing Session Objects

If you're not familiar with Java serialization, here's a brief overview. For more information, see the Javadoc for the serializable interface in the standard JDK.

Serialization is at the core of several Java standards involving the movement or storage of the object state outside of system memory. It's a powerful mechanism but simple to use. An object that implements the

Microsoft

www.microsoft.com

java.io.Serializable interface can have its state, and the state of all constituent objects, written to a java.io.ObjectOutputStream. The object can then be reconstituted by reading the same stream from an ObjectInputStream. To allow an object to be serialized, a programmer need only specify "implements Serializable" and be sure all constituent objects are serializable (or primitive) as well. Notably, a programmer doesn't need to code any methods at all to support serialization. Just about any JDK class you'd sensibly want to serialize already implements serialization.

Although it's easy to mark an object as serializable, the subtle danger is that it's easy to forget to mark an object deep in a hierarchy. You'd learn about the oversight only at runtime when ObjectOutputStream throws a NotSerializableException, and then only if you're vigilant. The size of a serial stream can be difficult to predict for the same reason: it's often not surprising to find that a field of your object was set to an object pointing to a large tree of objects. During serialization, you may find that the tree-following serialization process dutifully saves a lot of data you don't actually need.

Session Monitor

I've written a simple HttpSession display utility that could be useful to you in different ways, depending on where you are on the path toward J2EE.

1. *In a development environment, to monitor the size and structure of objects stored in sessions, possibly by different development teams.* This was the original purpose of the utility.
2. *To help study the composition of sessions while you're experimenting and learning.* To that end, if you don't want to buy a commercial app server yet (for the price of a car), you can learn with the reference version of J2EE generously made available from Sun at http://java.sun.com/j2ee.
3. *To give you a simple example of a JSP working with a JavaBean.* Also, it demonstrates the power of manipulating classes as objects.

The JSP "SessionTester.jsp" is in Listing 1 and the worker bean "SessionTesterBean.java" is in Listing 2. In the JSP you can see how easily Java and HTML are mixed. Note that "<%" … "%>" surrounds Java statements, while "<%=" … "%>" surrounds a Java expression that's converted to a string and inserted in the surrounding HTML. (I have to comment how wonderful it was to start using JSP after building cgi-bins for years in C. It was like emerging from a programmer's gulag.)

It's a stretch to call the SessionTesterBean class a bean. But the term *bean* is an accurate way to differentiate an independent Java class in an app server from a servlet or JSP. Plus, it sounds better than "Java worker class." The SessionTesterBean consists of a few static methods, one of which is called from the JSP. Each value in the session is found, then dumped by writing HTML to the "out" stream. Note the direct way I find the serialized size of each session value object – I just write the object to an ObjectOutputStream and find the number of bytes. I also check for exceptions and report them, which helps find any nonserializable object in your object tree.

Installing the utility can be different for every server vendor. Compile the .java file with the compiler in your app server's JDK. If necessary, "compile" the .jsp file, although many app servers automatically recognize, compile and install new JSPs. You have to put the .jsp file in a location that allows the app server to invoke it with a URL, and you have to put the .class file in a package directory, SessionTesterBeans, along the CLASSPATH that's active when the JSP is active. This may involve configuration efforts in your app server, but if you've already done some playing with JSPs and beans, you probably already know where to install the files in an existing configuration.

To use the utility, establish some HttpSession data in your own servlet or JSP, then simply use the URL needed to invoke SessionTester.jsp. The page returned shows a summary of the objects in the current session along with their sizes. Pressing a button displays session parameters and the composition of each object within it.

AUTHOR BIO

Peter Kobak is a technical lead for a major financial institution's Web site. He's currently part of a team working to migrate the site from a proprietary application infrastructure to a Java-based application server.

kobak@ieee.org

Listing 1

```
<HTML>
<TITLE>Session Unit Tester</TITLE>
<H1>Session Unit Tester</H1>

<FONT COLOR="Teal">Although caching is
 turned off on this page, it is safer
 to do a manual refresh to be sure the
 most current
 session is examined.</FONT>

<%
   //Turn off all browser caching
   response.setHeader("pragma","no- cache");
   response.setHeader("cache control","no");
   response.setHeader("expires","0");

   HttpSession session =
      request.getSession(false);
   if (session != null)
   {
      boolean fDeep =
         request.getParameter("De-
         tails") != null;
      String buttonName = fDeep ?
           "VALUE=\"Hide Details\"" :
           "NAME=\"Details\" "+
              "VALUE=\"Show Details\"";
%>
<FORM METHOD="GET">
   <INPUT TYPE="SUBMIT" <%= button-
Name%> >
</FORM>
<%
      SessionTesterBeans.SessionTesterBean.
         dumpSession( session, out, fDeep );
   }
   else
   {
%>

<H3>Couldn't get a session :-(</H3>

Suggestions for getting a session:<OL>
<LI>Hit the refresh button.
<LI>Make sure you have a session in
 your application; it may have expired.
 Try to start a new session in your
 application.
<LI>Check the application's session
 configuration.
</OL>

<%
   } // end else
%>

</HTML>
```

Listing 2

```
package SessionTesterBeans;

import java.beans.*;
import java.util.*;
import java.text.*;
import java.io.*;
import java.lang.reflect.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.beans.*;
import java.util.Date;

/**
* A set of static methods to test and
* dump servlet sessions. Normally to be
* used from SessionTester.jsp, but can
* also be used from the command line
* (see main()) or by calling these
* methods directly.
* <P>
* All dumping methods accept an object
* and an output stream.
* The display of the object is created
* as HTML and written to the stream.
*/
public class SessionTesterBean {

// don't display property values longer
// than this
static final int MAX_DISPLAYABLE_STRING = 100;

// time format used by epochTimeToString()
static final SimpleDateFormat myDateFormat =
   new SimpleDateFormat( "HH:mm:ss.SSS
                        'on' "+
                        "MM/dd/yyyy" );

/**
* Diplays the content of a session,
* including indentification of
* each object stored in the session.
* If fDeep is true, each such object
* is also displayed using dumpSerializ-
* able() and dumpBean() in this class.
* @param session Session to be dis-
* played.
* @param out The stream to which HTML
```

Microsoft

www.microsoft.com

```
* is written.
* @param fDeep True to perform a dump
* of each session object.
*/
public static void
dumpSession( HttpSession session,
             PrintWriter out,
             boolean fDeep )
{
   if (session == null) {
      out.println( "<BR>Null session
      passed to "+
         "SessionUnitTester.dumpSes-
          sion()" );
      return;
   }

   if (fDeep) {
      out.println( "<BR>Current  time: " +
         myDateFormat.format(new Date()));
      out.println( "<BR>Access   time: " +
         myDateFormat.format(new Date(
            session.getCreationTime())) );
      out.println( "<BR>Creation time: " +
         myDateFormat.format(new Date(
            session.getLastAccessed-
            Time())) );
      out.println( "<BR>session id: " +
         session.getId() );
      out.println( "<BR>timeout: " +
         session.getMaxInactiveInterval() );
      out.println( "<BR>new: " +
         session.isNew() );
   }

   String[] names = session.getValue-
   Names();
   out.println( "<P>Found "+
                names.length +
                " session
                objects:<OL>" );

   Object sessObj;
   int    totalObjSize = 0;

   for (int i = 0; i < names.length; ++i) {
      out.println( "<P><LI>" + names[i] );
      sessObj = session.getValue( names[i] );
      if (sessObj == null) {
         out.println(
            "<BR><B>Can't get this ses-
             sion "+
            "object.</B>" +
            " Possibly non-Serializ-
              able." );
      }
      else {
         totalObjSize +=
            dumpSerializable( sessObj,
                              out,
                              fDeep );

         /* Bean dumping code that's
         /* too long to show in this
         /* article
            if (fDeep) {
               dumpBean( sessObj, out );
            }
         */
      }
   }
   out.println(
      "</OL>Total size of session
       objects = " +
      totalObjSize + "<BR>" );
}

/**
* Displays a (supposedly) Serializable
* object by attempting to write the
* object to determine if the object is
* serializable, and if so, how big
* the object is. If fDeep is true, dis-
* plays the name of each field in the
* class.
* <P>The purpose of this display is to
* assist the developer in determining
* if an object can be serialized in a
* servlet session, and to report the
* size and composition of the object
* so it can be controlled at an early
* stage in the development process.
* @param sessObj An object to be dis-
* played.
*        Although Serializable objects
*        are expected, any object can
*        be used.
*        Non-serializable objects
*        (including non-serializable
*        objects embedded in
*        sessObj) will be reported as
*        such.
* @param out The stream to which HTML
* is written.
* @param fDeep True to display the
* fields of the object.
* @return Serialized size of sessObj
* (zero if couldn't be serialized).
*/
public static int
dumpSerializable( Object sessObj,
                  PrintWriter out,
                  boolean fDeep )
{
   if (sessObj == null) {
      out.println( "<BR>Null object
      passed to "+
         "SessionUnitTester.dumpSerial-
          izable()" );
      return 0;
   }

   Class sessClass = sessObj.get-
   Class();
   out.println( "<BR>Class name = "+
                sessClass.getName()
                +"<BR>" );

   int sessObjSize = 0;

   try {
      ByteArrayOutputStream baSessStrm
=
         new ByteArrayOutputStream();
      ObjectOutputStream objSessStrm =
         new ObjectOutputStream(
         baSessStrm );
      objSessStrm.writeObject( sessObj );
      objSessStrm.close();
      sessObjSize = baSessStrm.size();
      out.println( "Serialized size = "+
                   sessObjSize );
   }
   catch (Exception xcpt) {
      if (xcpt instanceof
          NotSerializableException) {
         out.println( "<B>This object
                      is not "+
                      "serializ-
                      able.</B><BR>" );
      }
      out.println( "While attempting to "+
                   "serialize, this
                   exception "+
                   "occurred:<BR>" );
      xcpt.printStackTrace( out );
   }

   if (fDeep && !(sessClass.isInterface() ||
                  sessClass.isPrimitive() ||
                  sessClass.isArray()
) ) {
      Field[] fields =
         sessClass.getDeclaredFields();
      if (fields.length > 0) {
         out.println( "<P>found "+
                      fields.length +
                      " object
                      fields:<OL>" );
         for (int ifield = 0;
              ifield < fields.length;
              ++ifield) {
            out.println( "<LI>" +
               fields[ifield].to-
               String() );
         }
         out.println( "</OL>" );
      }
      else {
         out.println(
            "<P>no object fields found" );
      }
   }

   return sessObjSize;
}

/**
* Stand-alone test for an object.
* The program argument is the name of a
* class.
* If the class can be loaded and an
* object created, that object is dis-
* played (with
* HTML tags) using <B>dumpSerializ-
* able</B>.
*/
public static void main( String[] args
)
{
   PrintWriter out =
      new PrintWriter( System.out );

   out.println( "Argument is name of
                class to "+
                "test for session
                properties." );
   if (args.length == 1) {
      final String testClassName =
      args[0];
      try {
         Class testClass =
            Class.forName( testClass-
            Name );
         Object testObject =
            testClass.newInstance();
         dumpSerializable( testObject, out,
                           true );
      }
      catch (Exception xcpt) {
         out.println(
            "Couldn't create an
            instance of "+
            testClassName );
         xcpt.printStackTrace( out );
      }
   }
   out.flush();
}

} // end class SessionTesterBean
```

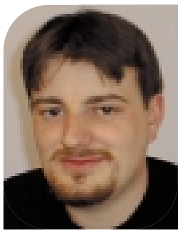
Download the Code!
The code listing for this article can also be located at
www.JavaDevelopersJournal.com

Microsoft

www.microsoft.com

Battle of the Titans

Welcome one and all to this month's dose of nonsense and trivia from the world of Java. December was a rather fun-filled month, with many things happening that will affect us all in the near future. I'm sure you've all heard about the controversy with Sun and IBM. But more on that later. Can't be getting into heavy stuff this early on. I have to ease myself into it gently!



WRITTEN BY
ALAN WILLIAMSON

December saw me in New York, hosting **SYS-CON Radio** with my esteemed colleague, Keith Douglas, at the Java Business Conference. We conducted a marathon of radio interviews, talking to a vast array of different companies from all over the planet. Well, I say planet, but let's be honest: the majority of them came from the inner sanctum of the Valley. I guess that's still the place to go if you intend running a tech-based company. That said, many observations were made at the show, so let me share some of them with you.

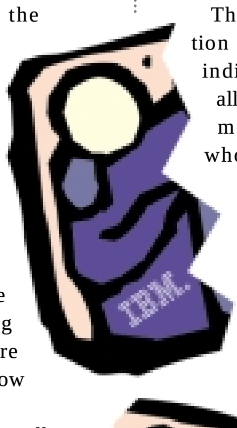
One of the more comical phrases I heard was that this was the East Coast version of JavaOne. Ha! Wishful thinking, methinks. For a start, there simply wasn't the number of people. Hardly anyone turned up. After the show ended, we got to talking to one of the ZD-Net conference organizers, who shall remain nameless, but she said that it was a big disappointment all around…nowhere near the expected number of people turned out. So why the low footfall?

A variety of possible reasons were discussed. One was oversaturation. An XML show was hosted in the same week, while a week later an e-commerce show was going to be held in the same venue. People simply can't attend them all. I think that's definitely a contributing factor, but personally I think it's all gotten rather….The pizazz of the Java revolution is beginning to wane. I think people are getting bored – and you know what? I don't blame them.

We spent a whole week talking, listening, interviewing and generally networking with companies from all walks of the industry. But none of them had anything exciting. I'm sorry if that sounds harsh, but compared to the JavaOnes of yore, no company had anything exciting or remotely cool to show off. I'm sure I'm going to get a whole raft of e-mails from companies complaining that this is a bit dramatic, but I'm sticking to what I said. Any company that makes an application server or virtual machine need not apply. I guess that wipes out the majority of complaints!

Seriously, talk about jumping on a bandwagon. Everyone and their dog seem to be producing an application server. Okay, chaps, I think the market is saturated enough now, so if you're on the verge of announcing a new application server, save it. The world doesn't need it. Have you seen the number of solutions available? It's madness. The only reason I think we're seeing a serious surge in this new market is money.

Java-based companies are finding it increasingly difficult to make money. Application servers are the only way they feel they can place high-ticket price tags on pieces of software. In an industry that's hell-bent on giving away software, the justification to charge for it is becoming increasingly hard. The reality is that we're in this game to make money, and "free" simply doesn't pay the rent.



This leads me to my next observation regarding a company and an individual that I think are letting us all down. Oh, but that reminds me. Let me thank those of you who came up to me and expressed kindness regarding this column. It was much appreciated. That said, would you believe I was accused by one gentleman of becoming soft? Soft? I ask you. He commented that of late I hadn't attacked any company. Well, for the record, this is something I don't do just to fill column space. If a company is in the wrong, I merely bring it to the attention of the world at large. If a company has done well, I do the same thing. It's just that people always seem to remember the bad news as opposed to the praise. Funny how human nature works.

"Talk about jumping on a bandwagon. Everyone and their dog seem to be producing an application server"

But let me give the companies of this industry advance warning: I'm a'comin' to get you. We're about to conduct a major test of the support this industry provides. We'll contact all the big names and some of the smaller ones, asking for help on their products. Don't worry. We won't be using our real names – wouldn't want you giving me any special treatment just because I'm a ***JDJ*** columnist. That wouldn't be fair to the normal developer. It'll be very revealing, and the good news is that you'll be able to hear some of the more interesting help through our **Straight Talking** radio show. Each month I'll update you on our findings. Our aim is to raise the standard of support in the Java world.

Okay, now that I've given you advance warning, back to the thread of conversation we were on before I drifted.

As you know, pre-Christmas '99 we had a major falling out of two of the largest companies in the Java industry, Sun and IBM. A severe case of toys being thrown out of the pram and storming off in a huff, taking all the toys with them. Sun may be playing a dangerous game, and may already be alienating developers.

Hit Software

www.hit.com

A Bit of Background

Essentially, there was a move to have Java's future controlled by a consortium of companies as opposed to just one, Sun. IBM was a major player in this consortium, and was keen to move forward with this. Sun, however, became increasingly nervous, and one week into December said they would be pulling out of this initiative. IBM retaliated by announcing that there may be a version of Java that may not be compatible with the present one. A threat if ever I heard one. Sun then threw out the fact that 80% of the existing Java is tied up in legal copyright. But that leaves 20% that isn't. It takes only 1% for it to become non-compatible. A scary state of affairs, if you ask me.

At the time of this writing the position of the companies hadn't changed. I spoke to an IBM'er close to the action who stated that IBM didn't want to see more standards/names for Java. The latest Java Enterprise Edition hasn't impressed IBM, and they feel it's a step too far. I tend to agree with them. The whole version-numbering of Java has simply gotten out of hand, and something needs to be done. We have Java 2.0, but the actual release is 1.3. Talk about introducing confusion into an already version-happy world. Sun…sort it out.

So Sun doesn't want to relinquish control of Java. They want to control it all themselves, and want us to believe that it's in everyone's best interest to allow them to continue. Gee, this sounds familiar. Ring any bells?

The man who may have lost the respect of the development community is Scott McNealy. For a while he was the developer's friend, speaking out against the empire we all love to hate and giving us the resources to allow us to fight the onslaught of Microsoft. But he's gone and buggered up his power base by showing his true colors, which ironically seem to be the same shade as Mr. Gates's.

McNealy is famous for slandering Microsoft in his keynotes, and his constant attack on all things Gates was beginning to get boring. But from a man who once retorted he probably wouldn't do anything different if he were in Bill Gates's shoes, it looks as though he's setting himself up for this sort of power base. McNealy may end up as a man no longer to be trusted, and Java may no longer be safe in his hands. Let me illustrate why.

For years people have asked where Sun was making its money, and for years the standard answer was "from the sale of its servers." People were led to believe that Java would run faster on a Sun server than any other, so they'd flock to Sun, leaving other server manufacturers out in the cold. If you've read the recent Sun book, you know this was the plan back in the beginning. Didn't quite work out like that. For a start, Sun's servers didn't outperform the competition in quite the way Sun had hoped. Also, the virtual machines offered by other vendors knocked the spots off Sun's own offering. So the original plan to make money from the sale of servers wasn't working out. Sun needed an income from Java.

> "The whole version-numbering of Java has simply gotten out of hand, and something needs to be done"

Well, here's an idea. Since you control the fate of Java, why don't you charge people for using the name in their products? Nah, that would be like charging for the use of the word *Internet* in every product. Silly. Sound ridiculous? Well, guess what? That's what McNealy is doing. Yup, if you have a product that's compliant with the J2EE API, then you owe Mr. McNealy 3% of your profits. Tip of the iceberg. What else is Sun going to charge for? One cent for every download of your applet? As bizarre as that sounds, it's heading that way.

McNealy has gone and done the one thing he said he'd never do, and that is to charge for Java. If IBM plays this right, they could become the developer's new champion and take on a new version of Java that's both open and free.

I never thought I'd be throwing my support behind Big Blue, but Sun has lost my respect and they'll have to do an awful lot to get it back again. For a start, I no longer trust what McNealy tells me. They don't want to share the development of Java, and their belief that they're the only ones that can really do it justice will be Java's downfall. What I'd love to see is companies releasing products that are only 97% compatible – where the 3% is lost since they don't want to pay royalties on the use of a name.

Mailing List

With that I move on to the mailing list that's based on this column. We've been discussing this very subject in depth, and no one has posted any mail that supports Sun's stance. All have stated their feellings about what Scott McNealy is up to. It's good to hear your views on this and other topics of interest. Keep them coming and let us know what you think. I'd like to point out that we have a new list server in place, which means that the directions for signing up that I've given out in previous columns are no longer valid. We have a new Web-based system that makes it easier for everyone. Point your browser toward the URL http://listserv.n-ary.com/mailman/listinfo/straight_talking for all the details. It's easy and very quick. I look forward to hearing from you.

Alternatively, if you want to hear from me, then head along to http://radio.sys-con.com/ to hear the daily **Straight Talking** radio show. It's a mix of Java chat, music and a bit of insight into all sorts of useless and interesting trivia.

Salute of the Month

This month the salute goes to Bruno Decaudin and the **SYS-CON Radio** production team, who turned the interview sessions into one slick, well-oiled operation. Keith and I thank you for your efforts – we thoroughly enjoyed having you as part of the team and look forward to working with you at JavaOne.

• • •

On that note I'd better wind up this month's installment. The good news is that I think my Dolly Parton phase is weakening. I can't be too sure though, as I recently took delivery of a portable MP3 player, which means that Miss Parton goes with me to the gym as I continue my quest for the body beautiful. Oh dear, maybe I still have a long way to go!

AUTHOR BIO

Alan Williamson is CEO of n-ary (consulting) Ltd, the first pure Java company in the United Kingdom. The firm, which specializes solely in Java at the server side, has offices in Scotland, England and Australia. Alan is the author of two Java servlet books, and contributed to the Servlet API. He has a Web site at www.n-ary.com.

alan@sys-con.com

Segue

www.segue.com

How to implement a JSP component-based framework by applying application server techniques

Creating a JSP JavaBeans Framework

WRITTEN BY DAVID LYONS

JavaServer Pages (JSP), an API layer that extends the servlet architect, provide developers with a standard for creating template-based HTML applications. The JSP specification marries scripting tags and Java code in an HTML template with nonvisual JavaBeans and servlets running in the JSP/servlet engine. It provides a flexible solution that caters to Web developers who use a scripting approach with display logic embedded in a Web page and application developers who prefer to separate the HTML from the controlling logic by using a component-driven approach.

This article shows how to build JSP applications using a component-based framework. By leveraging nonvisual JavaBean components, the Java event model and a bean-aware IDE, I'll demonstrate how you can realize many of the rapid development advantages of component development without having to use an application server-specific IDE and API.

HTML, Application Servers and JSP

Before JSP, application server vendors created their own server-side API to display HTML Web pages. Using products such as Sun NetDynamics, SilverStream and Bluestone Web/Sapphire, developers can create a containment hierarchy of controls, set properties via property sheets and bind the controls to a database. Tags corresponding to their controls are then inserted into the HTML and used during the rendering process. Vendors developed some productive approaches to this process (they also provided input to the JSP specification); however, each vendor developed a different approach. This is now a problem since the productivity resulted from using the vendor's own API, which, of course, ran only on that vendor's server. The approach discussed in this article isn't very different from the vendors'. However, we're using commonly available tools and leveraging existing standards to improve productivity while creating a portable framework across JSP implementations.

Application server vendors are now beginning to support JSP – a good development. The products already provide most of the enterprise services needed by the JSP presentation layer. Yet base-level support of JSP differs from productive JSP support. In many of these products productive support in the form of IDE enhancements and API extensions is six months to a year away. In the meantime, you can start leveraging JSP productivity today while the vendors work toward more comprehensive solutions that differentiate their products.

Before We Begin

I'll now show what you can currently build using the tools you already own (or that are readily available). The code is limited in scope, functionality and error checking. In other words, don't try this in production!

First we'll build a JSP framework, which will consist of several JavaBeans and base classes for our application. Next we'll create a small Web application (one page) that uses the framework and beans to build the display of a JSP page. To further limit the scope of the code in demoland, we won't connect to a database or an EJB (Enterprise JavaBean) even though we would in the real world. For now we'll focus on the display aspects of a page. Most (though not all) of the code is given at the end of this article. The complete source is located on **SYS-CON**'s Web site at www.JavaDevelopersJournal.com.

Using Components with JSP

Our JavaBeans components will encapsulate display aspects of HTML or XML tags. These tag components will reside on a JavaBeans palette in your IDE just like visual Swing JavaBeans components. Our beans will comprise HTML tags such as an input button, input text or just static HTML text. The framework is extendible, so all types of data-driven, text-based objects can be built. Does using a component-based approach sound vaguely familiar? It should. Although developing component-based JSP applications is similar to developing client-side JavaBean applications, there are several major differences. One example is event flow. In applets and applications, user actions can occur in varying sequences. In JSP/HTML applications the set of options is more defined; once the user chooses an action, the display process is a controlled event flow or batch process as the display of the page follows a predictable order of events. On demand, our tag components will "render" themselves using their properties and data. Our tags will also fire Java events during the rendering process to provide "user exits" for developers to add code that dynamically changes the runtime display. This provides a flexible architecture that the developer can exploit for Web pages with complex display logic and business rules.

JSP Scopes and the Framework

An important part of JSP and servlet development is knowing the thread safety rules for each JSP scope. JavaBeans can be instantiated in different scopes ranging from page scope (local to a page rendering) to application scope (global to all JSPs and servlets in an application). The JSPs in this article create page scope beans. This means bean instance variables are okay because the bean itself is local to a single page rendering. While page scope eliminates thread-safety concerns, heavy use of page-scope objects also creates more overhead and drags down performance.

Fortunately, the JSP framework developer can choose one of two approaches when using a framework such as this in a wider scope without making life harder for the JSP application developer. The first approach is object pooling of the beans. Application servers such as Sun NetDynamics use this technique extensively to limit the creation of new objects while serving a large number of concurrent users. The concept is the same as database connection pooling in which an instance is checked out, utilized, then checked back into the pool. The second approach separates the framework into two sets of classes: lighter-weight property objects created in a page scope that contain variable data specific to each page, and corresponding application scope objects that contain design-time property settings and methods that operate on the data. Again, our objects will be page scope objects so these measures aren't needed but I did want to address this important issue.

First Up, Events and Listeners

First we'll define the events the DataTag bean can fire. For example, the bean could fire a preDataFetch event to allow a developer to set some defaults before data is received. In another case a preHtmlOutput event that enables a developer to change the completed tag before display could be fired, or he or she could decide not to display if the current user has inadequate security. Since we're working with JavaBean components, many IDEs will recognize the events and provide visual support for easily adding them to our source code. This speeds development while providing structure to our process. For this article we'll define one event, formatOutputProperties, that fires when we're ready to add formatting to our display value. For example, we may want to format a date or add a currency format to our value. See Listing 1 for the event and listener code.

Creating a DataAdapter Class

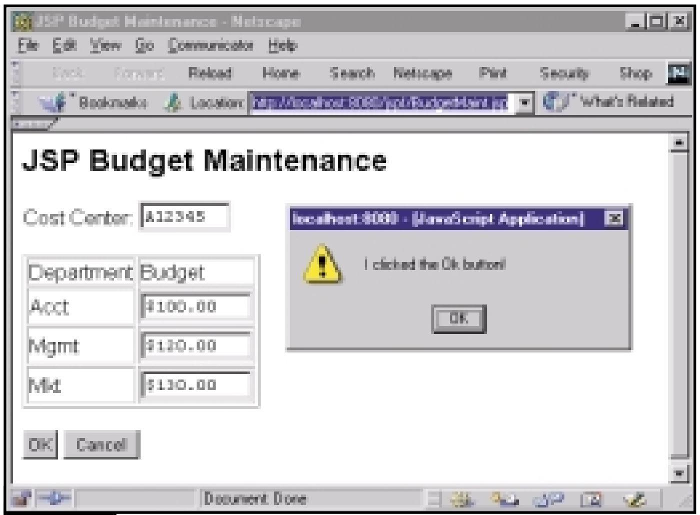
Next, we'll define a DataAdapter class. This wrapper class retrieves values from an instance (or member) field and provides the value to the DataTag. The DataAdapter concept can become quite flexible, and subclasses could support access to JDBC resultsets and EJB components. For now we'll keep our class simple and support only single-value and array objects that use the toString() method to convert their values to string format. Wrapping an object is completed by simply passing it via the appropriate constructor. The adapter then returns values as requested, using getValue() methods. The method signatures are shown below. See Listing 2 for the complete source.

```
Public DataAdapter(
  Object value)
Public DataAdapter(
  Object[] value)
Public Object getValue()
Public Object getValue(
  int Index)
```

Adding the DataTag Class

Now we're ready to define the DataTag. This is the core class in the framework. It controls the process of rendering text by retrieving data from a source and firing events the developer uses to customize the display process. We'll create the class and then extend it to render HTML Input Button and Text tags along with static HTML text. Our framework users will see the extensions to this object as components that are residing on the palette as JavaBeans.

Our base bean tag has two primary methods and several important properties: (1) display(int index) drives the tag-rendering process, firing events as needed – we implement this in the DataTag; (2) render(int index)


JSP Budget Maintenance - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop
Bookmarks Location: http://localhost:8080/jspl/BudgetMaint.jsp
What's Related
Document: Done

JSP Budget Maintenance

Cost Center: A12345

Department	Budget
Acct	$100.00
Mgmt	$120.00
Mkt	$130.00

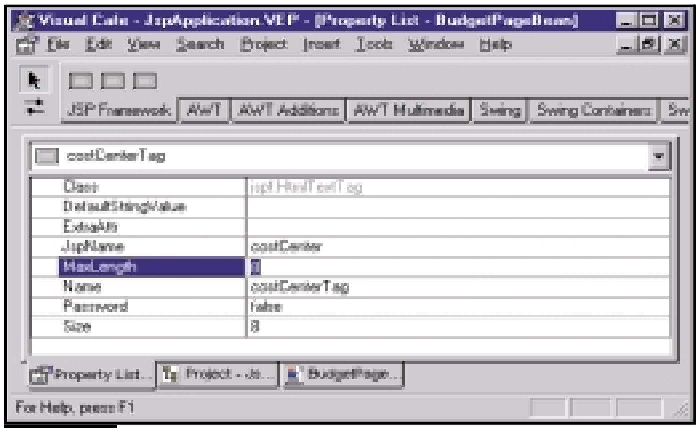
OK Cancel

localhost:8080 - [JavaScript Application]

I clicked the Ok button!

OK

FIGURE 1 JSP page in browser with JavaScript message displaying


Visual Cafe - JspApplication.VEP - [Property List - BudgetPageBean]
File Edit View Search Project Insert Tools Window Help
JSP Framework AWT AWT Additions AWT Multimedia Swing Swing Containers
Property List... Project - Js... BudgetPage...
For Help, press F1

costCenterTag

Class	jspl.HtmlTextTag
DefaultStringValue	
ExtraAttr	
JspName	costCenter
MaxLength	8
Name	costCenterTag
Password	false
Size	8

FIGURE 2 CostCenterTag's HtmlTextTag properties as displayed in VisualCafé

is an abstract polymorphic method implemented in our subclasses – this method is called during the display process and creates and returns the HTML. Our class also contains several instance variables (see Figure 1):

- ***String jspName:*** Name applied to a form tag for "name" and servlet request parameter purposes
- ***DataAdapter valueSource:*** Source object from which the tag's data value will be retrieved
- ***String defaultStringValue:*** Value used when a tag isn't bound to a source
- ***String stringValue:*** Temporary value changed on each rendering by retrieving either a data value from the source or a copy of the default value; developer formats this value in the formatOutputProperties event
- ***String extraAttr:*** Free-form field used by the developer to add JavaScript or other scripting information – appended as last property
- ***Vector displayListeners:*** Event source support

Other properties work as well. For example, an extraAttributes tag, extraAttr, is included for adding JavaScript to our tags. All of our variables that have accessor methods conforming to the JavaBean specification (or defined in BeanInfo) can be set in the bean's property sheet. However, the stringValue property is meant solely for use via the API. No problem. We can choose to hide it from the property sheet by creating a BeanInfo class. The source code for the key instance variables and methods in DataTag is located in Listing 3.

Now let's examine the display method in closer detail. This is where the real action takes place. It executes property get and set statements and fires DataTag defined events. As mentioned earlier, it makes sense to define events that fire at different points in the display process to allow runtime customization. For example, the snippet of the display logic below checks for the existence of a DataAdapter as a source of our value. If none is found, it uses a default string value provided via a property sheet. This is another example where we could consider defining an event prior to executing this logic. Why? On a tag-by-tag basis, we could then listen for the event and execute logic to change the default value or change the source DataAdapter based on user-specific information. Let's say we're including news headlines on our JSP page. One user may prefer *Wall Street Journal* headlines while another prefers *Newsweek*. We could swap the source at runtime before the value is retrieved without changing the core display process.

```
if(getValueSource() == null) {
 setStringValue(
  getDefaultStringValue());
} else {
 setStringValue(
  getValueSource().
  getValue(index).toString());
}
```

Once we set our string value for display, we execute our own event, formatOutputProperties. To utilize the event, all our framework users need to do is set up a listener and add the business logic. Since most IDEs provide event and listener code-generation support, it's a snap to implement.

With our DataTag base class coded, we can now extend the class to create other tags. Only a single method, render(int index), must be implemented, though many tags will add additional properties. For example, an HTML text tag requires size and maxlength properties. The HtmlTextTag demonstrates this in Listing 4 (property getters and setters have been eliminated to save space). The complete source and the source for other tags used in the application can be found at www.JavaDevelopersJournal.com.

Finishing Our Framework

The last class to create in our framework is the container class for the DataTags. We'll call this class PageBean because it'll be created based on a reference in a JSP page (see Listing 5 for the complete source). The bean will be the connector between the JSP page and our DataTags. We could also use it to pass JSP variables, such as the pageContext, to our DataTags. The PageBean maintains a hashtable of DataTags to the JSP page. When a developer codes one of the following tags in the JSP template,

```
<%=pageBean.display(
  "costCenter")%>
<%=pageBean.display(i,
  "costCenter")%>
```

the method on our PageBean directs the call to the correct DataTag using the following code:

```
return ((DataTag) tagTable
 .get(name)).display(index);
```

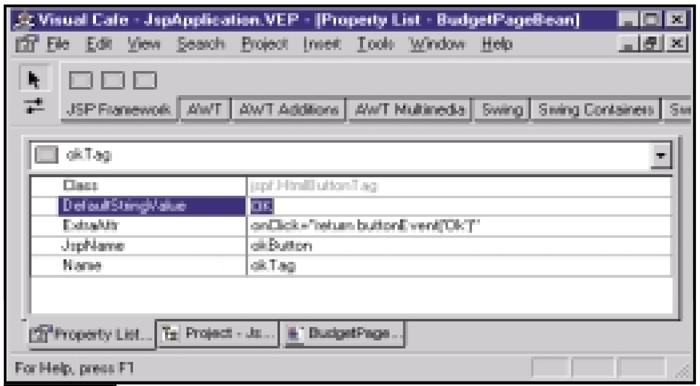
With our lightweight framework complete, we can deploy it and build the JSP application. After we JAR the classes and install them as components in our JavaBean palette, the role of the framework developer is complete. We can now leverage our IDE's capabilities to utilize our framework to speed development of the JSP.

Creating the JSP Application

For this example we'll create a budget maintenance page to list departments and their annual budgets (see Figure 2). First we create the BudgetPageBean class to extend the PageBean class (see Listing 6). Next, we use our IDE (I'm using VisualCafé) to drag and drop DataTags representing HTML tags onto the BudgetPageBean. I did this for each HTML tag on the page and then renamed the tags and set the properties. For example, two of our tags are a cost center text field and an OK button (see Figure 3). We may be able to drag and drop the objects to our bean using our editor. If not, we can do it by adding them to the source as instance variables. We'll also code several set methods to initialize properties. Again, an IDE may create this code for us via property sheet changes or we'll code the changes ourselves in a constructor or init event.

Modis

www.modis.com



Visual Cafe - JspApplication.VEP - [Property List - BudgetPageBean]
File Edit View Search Project Insert Tools Window Help
JSP Framework AWT AWT Additions AWT Multimedia Swing Swing Containers
okTag
Property List... Project - Js... BudgetPage...
For Help, press F1

Class	jspf.HtmlButtonTag
DefaultStringValue	OK
ExtraAttr	onClick="return buttonEvent('OK')"
JspName	okButton
Name	okTag

FIGURE 3 okTag's HtmlButtonTag properties as displayed in VisualCafé

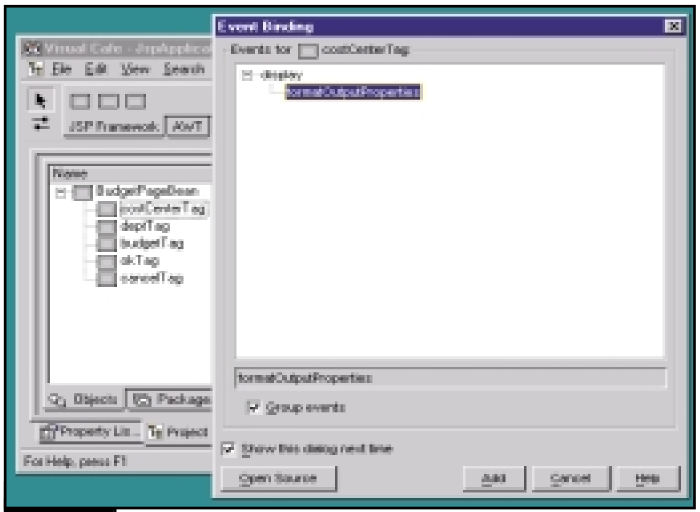


FIGURE 4 VisualCafé's graphical way of binding events to listeners

```
costCenterTag.setJSPName(
  "costCenter");
costCenterTag.setMaxLength(8);
costCenterTag.setSize(8);
okTag.setJSPName("okButton");
okTag.setDefaultStringValue(
  "OK");
okTag.setExtraAttr(
  "onClick=\"return
  buttonEvent(\'Ok\')\"");
```

Next, we'll add our instance variables that are our source data items. These values will be set by our business logic prior to displaying the page. *Note:* We include support for both single-value items and array items. This structure corresponds to the servlet API's support for both types via getParameter() and getParameterValues().

With all our variables defined, we'll complete the initial setup by linking the tag to its container and data source. enableTag() adds the tag to the pageBean's hashtable and setValueSource() sets the object from which we'll retrieve data values.

```
enableTag(costCenterTag);
costCenterTag.setValueSource(
  costCenter);
```

Let's make this more interesting by adding currency formatting to the budget text field for display purposes. We'll just register an event listener with the budgetTag so we can respond to the formatOutputProperties event (see Figure 4). Your IDE may even add all the plumbing for the listener. In your listener, add the business logic to make the change. The code that changes the formatting is shown below.

```
DataTag tag = (DataTag)
 event.getSource();
tag.setStringValue("$" +
 tag.getStringValue() + ".00");
```

Creating the JSP Page

The last step in our process is to create the JSP page. Actually, this can be your first step, depending on your preferred approach. We'll create a basic JSP page with some single-value tags along with an HTML table. Note that very little business logic or display code is needed in the JSP page. This keeps our HTML clean and provides for easier graphical HTML editing via DreamWeaver or FrontPage. The controlling display logic is maintained in our PageBean-derived class instead of being interspersed with HTML in the JSP template. The JSP source is in Listing 7. Take note that we have easily incorporated an HTML table into our source. Since our framework supports arrays, we just pass a row number to our source.

```
<%for(int i=0; i<3; i++) {%>
  <tr><td><%=pageBean.display(
    i,"dept")%></td>
  <td><%=pageBean.display(
    i,"budget")%></td></tr>
<%}%>
```

There you have it! We've implemented a JSP component-based framework and built our first Web page by applying it. Using this approach isolates the HTML template from the dynamic display control logic, allows us to use a component-based approach and leverages the rapid application capabilities of today's IDEs.

AUTHOR BIO

David Lyons is a technology director with Virtualogic in Bethesda, Maryland. A Sun NetDynamics certified instructor, David has developed Web applications using several Java application servers.

dlyons@virtualogic.com

Listing 1: Event and Listener source

```
package jspf;
public class DisplayEvent extends
    java.util.EventObject {
  private int rowIndex;

  // pass rowIndex so tag can retrive
  // multivalued items
  public DisplayEvent(Object source,
      int rowIndex) {
    super(source);
    this.rowIndex = rowIndex;
  }
  public int getRowIndex() {
    return rowIndex;
  }
}

package jspf;
public interface DisplayListener
    extends java.util.EventListener {
  public void formatOutputProperties(
      DisplayEvent e);
}
```

Listing 2: Data Adapter source

```
package jspf;
public class DataAdapter extends Object
    implements java.io.Serializable {
  private Object     value;
  private Object[]   arrayValue;
  private boolean    multivalued = false;

  // wrap a single-value item
  public DataAdapter(Object value) {
    this.value = value;
    multivalued = false;
  }
  // wrap an array item
  public DataAdapter(Object[] value) {
    this.arrayValue = value;
    multivalued = true;
  }
  public Object getValue() {
    if(multivalued) {
      return arrayValue[0];
    } else {
      return value;
    }
  }
```

YouCentric

www.youcentric.com

```
   // return the requested value. Note:
   // works with both single- and multi-
   // value items
   public Object getValue(int index) {
    if(multivalued) {
       return arrayValue[index];
     } else {
       return value;
     }
   }
}
```

Listing 3: DataTag key instance variables and methods

```
   private String jspName = "";
   private DataAdapter valueSource;
   private String defaultStringValue = "";
   private String stringValue = "";
   private String extraAttr = "";
   private Vector displayListeners = new
   Vector();

   abstract public String render(int index);

   protected String display(int index) {
     // reset our temp value before display
     setStringValue("");
     // get default value or get dynamic value
     // from our data source
     if(getValueSource() == null) {
       setStringValue(getDefaultString-
       Value());
     } else {
       setStringValue(getValueSource().
           getValue(index).toString());
     }
     // fire format event so developers can
     // customize stringValue to display
     Vector v1 = (Vector)displayListen-
     ers.clone();
     for(int i=0; i < v1.size(); i++)
       ((DisplayListener)
v1.elementAt(i)).
           formatOutputProperties(
           new DisplayEvent(this, index));
     // create html by calling tag's
     // render process
     return render(index);
   }
```

Listing 4: HtmlTextTag

```
package jspf;
public class HtmlTextTag extends DataTag {
   private int maxLength = 10;
   private int size = 10;
   private boolean password = false;

   public String render(int index) {
     StringBuffer  html = new String-
     Buffer();

     html.append("<INPUT TYPE=");
     // check for password type
     if(isPassword())
       html.append("PASSWORD ");
     else
       html.append("TEXT ");
     html.append("NAME=").append("\"").
         append(getJSPName()).append("\"  ");
     html.append("VALUE=").append("\"").

append(getStringValue()).append("\" ");
     html.append("MAXLENGTH=\"").

append(getMaxLength()).append("\" ");
     html.append("SIZE=\"").append(get-
Size()).
         append("\"");
     // if extra attributes included,
     // append them.
    if(! getExtraAttr().equals(""))
      html.append(" ").append(getExtraAttr());
    html.append(">");
     return html.toString();
   }
   // property get / set methods excluded
   // from listing – see complete source
```

Listing 5: PageBean source

```
package jspf;
import java.util.Hashtable;

public abstract class PageBean
    implements java.io.Serializable {
  private Hashtable tagTable = new
Hashtable();

  // single value display process
  public String display(String name) {
    return processDisplay(-1, name);
  }
  // overloaded display process multi-
  // value items
  public String display(int index,
  String name) {
    return processDisplay(index, name);
  }
  // forward display request to named tag
  protected String processDisplay(int index,
      String name) {
    return ((DataTag)
tagTable.get(name)).
          display(index);
  }
  // add tag to hashtable of tags we
  // can display
  protected synchronized void enableTag(
      DataTag tag) {
    tagTable.put(tag.getJSPName(), tag);
  }
}
```

Listing 6: BudgetPageBean source

```
package jspf;

public class BudgetPageBean extends
jspf.PageBean{
  String costCenter = "Value to   replace";
  String[] dept = {"Acct", "Mgmt", "Mkt"};
  Integer[] budget = new Integer[3];

  public BudgetPageBean() {
    // visual cafe init - varies per IDE
    vcInit();
    // load budget array
    budget[0] = new Integer(100);
    budget[1] = new Integer(120);
    budget[2] = new Integer(130);
    // load tags we'll display from JSP
    enableTag(costCenterTag);
    enableTag(deptTag);
    enableTag(budgetTag);
    enableTag(okTag);
    enableTag(cancelTag);
    // set sourceValues
    costCenterTag.setValueSource(cost-
    Center);
    deptTag.setValueSource(dept);
    budgetTag.setValueSource(budget);

    SymDisplay lSymDisplay = new
    SymDisplay();
     budgetTag.addDisplayListen-
     er(lSymDisplay);

   costCenterTag.addDisplayListener(lSymDis play);
  }

  public void vcInit() {
    costCenterTag.setJSPName("costCenter");
    costCenterTag.setMaxLength(8);
    costCenterTag.setSize(8);
    deptTag.setJSPName("dept");
    budgetTag.setJSPName("budget");
    okTag.setExtraAttr(
      "onClick=\"return
       buttonEvent(\'Ok\')\"");
    okTag.setDefaultStringValue("OK");
    okTag.setJSPName("okButton");
    cancelTag.setExtraAttr(
    "onClick=\"return buttonEvent(\'Can-
     cel\')\"");
    cancelTag.setDefaultStringValue("Cancel");
    cancelTag.setJSPName("cancelButton");
  }
```

```
  HtmlTextTag costCenterTag = new Html-
  TextTag();
  HtmlStaticTextTag deptTag = new
    HtmlStaticTextTag();
  HtmlTextTag budgetTag = new HtmlText Tag();
  HtmlButtonTag okTag = new HtmlButtonTag();
  HtmlButtonTag cancelTag = new Html-
  ButtonTag();

  // diplaylistener class added by IDE
  class SymDisplay implements Dis-
  playListener {
    public void formatOutputProperties(
        DisplayEvent event) {
      Object object = event.get-
      Source();
      if (object == budgetTag)
        budgetTag_formatOutputProper-
        ties(event);
  else if (object == costCenterTag)
costCenterTag_formatOutputProperties(event);
    }
  }
  // add budget currency formatting
  void budgetTag_formatOutputProperties(
      DisplayEvent event) {
    DataTag tag = (DataTag) event.get-
    Source();
    tag.setStringValue("$" + tag.get-
StringValue()
        + ".00");
  }
  // set cost center value on the fly
  // via code
  void costCenterTag_formatOutputProperties(
      jspf.DisplayEvent event) {
    DataTag tag = (DataTag) event.get-
    Source();
    tag.setStringValue("A12345");
  }
}
```

Listing 7: Budget.jsp source

```
<%@page language="java" session="true"%>
<jsp:useBean id="pageBean" scope="page"
 class="jspf.BudgetPageBean"/>
<html><head><title>JSP Budget Mainte-
nance</title>
<SCRIPT LANGUAGE="JavaScript">
  // sample Javascript function we call
  // via setting extraAttr property
  function buttonEvent(button) {
    alert("I clicked the " + button + "
    button!");
    return false;
  }
</SCRIPT>
</head><body bgcolor="#FFFFFF">
<h2>JSP Budget Maintenance</h2>
<form method="post" action="tp.jsp"
name="frm1">
<p>Cost Center:

<%=pageBean.display("costCenter")%></p>
  <table border="1">

<tr><td>Department</td><td>Budget</td></tr>
    <%for(int i = 0; i < 3; i++) {%>
    <tr>

<td><%=pageBean.display(i,"dept")%></td>
      <td><%=pageBean.display(i,"bud-
get")%></td>
    </tr>
    <%}%>
  </table>
<p><%=pageBean.display("okButton")%>
<%=pageBean.display("cancelButton")%></p
>
</form></body></html>
```

Download the Code!
The code listing for this article can also be located at
www.JavaDevelopersJournal.com

Flashline

www.flashline.com

What's Going On with Your Downloads?

How to provide the details your clients are asking for

WRITTEN BY
ALEXANDRE LEMIEUX

With the Internet widely available, many software clients are asking for built-in networking capabilities. Data exchange is an advantage in any software product, and software that offers networking functions has a better chance of being selected.

Although the Internet has spread throughout the world, the rate of data transmission is still a problem, even on your business's local area network. Getting files from the Internet easily takes a few minutes even with a high-speed connection. Users want to see what's going on with their software when it's looking for data on the Internet. Telling them when it's over isn't sufficient. Programmers need to show them, by any means possible, that data is flowing between the server and their software. Elements such as the completion percentage, transfer rate, progress bar, estimated remaining download time and animation will make users wait more patiently in front of their screen. No more "Please wait..." messages.

I'll propose two classes that'll enable you to provide the details your clients are asking for. Your data transfer won't be faster, but users will be more likely to wait until the end rather than click the "Cancel" button in desperation.

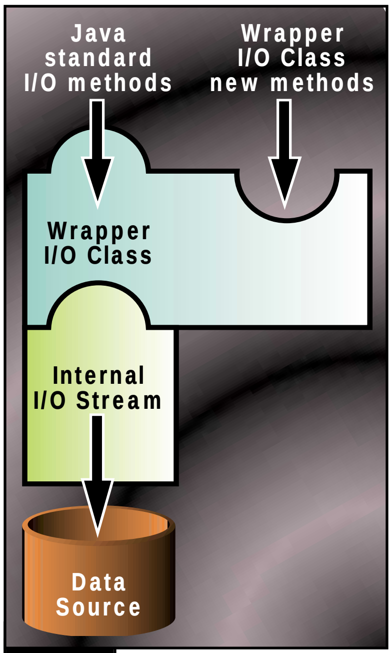


FIGURE 1 Class hierarchy diagram

I/O Handling

Before going into implementation details, I'll do a brief overview of I/O handling in Java. As you probably know, there are two I/O families: the input and output streams and the reader/writers.

Streams are low-level classes that allow you to directly use the disk, networking and peripherals to read and write raw bytes. Readers and writers are wrapper classes that translate unformatted data provided by the streams in data types, thus allowing them to be handled by programmers (boolean, int, float, string, even serialized objects). Since I'm willing to provide a source-dependent solution, I'd implement streams as base classes rather than readers and writers because they're too specific to be used in this situation.

There are more than 20 classes for handling data I/O in the stream mode, but they all have something in common: they inherit from the java.io.InputStream or java.io.OutputStream classes. Studying these classes, I've found that only two of their methods are abstract: the read() method (from the InputStream) and the write() method (from the OutputStream). All I had to do was implement these methods and add new ones.

The Architecture Used

I'll create wrapper classes that will plug into other classes to add functions to the old ones, the same way reader and writer classes do. See Figure 1 for the class hierarchy diagram.

The inheritance technique isn't usable because in order to be source-dependent, the solution can't be as specific as FileInputStream, SocketInputStream and all the other classes. It's simply impossible to extend all these classes. Rather, I'd extend the abstract InputStream (and OutputStream) to create wrapper classes.

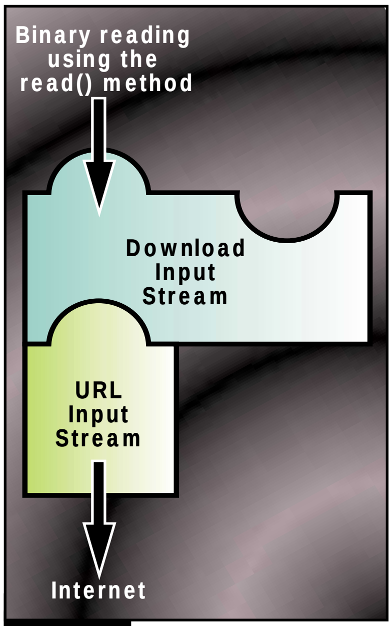


FIGURE 2A

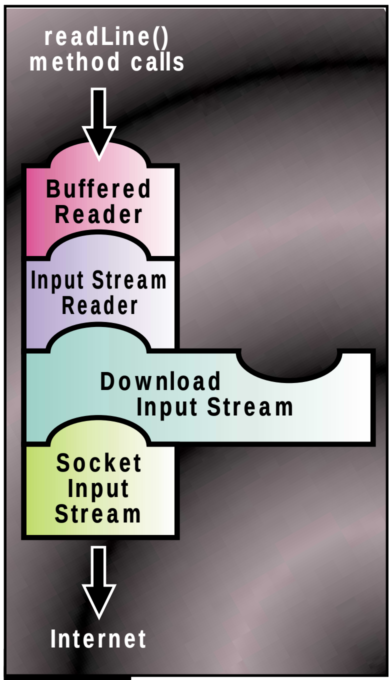


FIGURE 2B Possible uses of classes

Persistence

www.persistence.com

Since my classes will extend InputStream (or OutputStream), they can be used in the same way as any base class, even in conjunction with reader/writer classes. The main goal is to provide programmers with a class that acts the same way as the InputStream and OutputStream classes.

Figures 2A and B give examples of possible uses of my classes: binary reading from a URL and formatted reading from a socket connection.

Defining the Problem

Which variables should you show to your users to make their wait a little easier? Here's a short list of data users want to see:

- *Number of bytes read*
- *Total size*
- *Elapsed downloading or uploading time*
- *Mean transfer rate*
- *Estimated remaining download or upload time*

To provide your users with this data, you need only a few variables – start time, number of bytes read and total size. With these variables you'll be able to compute any of the values listed above. It must be said, however, that some of these variables aren't always available. While downloading data from a servlet, for example, you won't know the full length of what you're downloading until it's over (and that's too late). This is also the case for most custom protocols over a socket connection.

Implementing the Classes

Since these classes must be able to be used from any source, they must inherit from java.io.InputStream and java.io.OutputStream. As I've said before, it's impossible to know the exact length of an InputStream or an OutputStream. This leaves us with two possibilities: force the developer to give it through a constructor, which allows us to provide needed values, or simply ignore it, in which case some values won't be available. The only exception to this problem is the java.net.URL object. With the java.net.URLConnection class, the developer will usually know the size of a download before reading it. To simplify the process, a special constructor has been made to handle URLs.

AUTHOR BIO

Alexandre Lemieux is studying computer engineering at the Université du Québec à Chicoutimi in Canada. He also teaches UNIX, PERL and Java at the Humanis, Centre de Formation Continue, and manages a software development project for Humanis using Java servlets and distributed objects with RMI.

Here are the constructors for both wrapper classes:

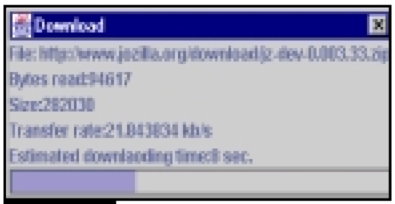
- *DownloadInputStream(InputStream is, int size)*
- *DownloadInputStream(InputStream is)*
- *DownloadInputStream(URL url)*
- *UploadOutputStream(OutputStream os, int size)*
- *UploadOutputStream(OutputStream os)*

Almost every method call is simply a call to its equivalent on the private internal instance. The available() method of DownloadInputStream calls the available() method on the private instance of the InputStream given through the constructor.

```
public int available() throws IOEx-
ception {
   return internalInputStream.avail-
able();
}
```

You need to know the starting time and the number of bytes being read up to now. To provide the user with this data, you need to reimplement the read(...) and write(...) methods so that the information is computed as the stream is being used. The following code shows the reimplementation of the read(...) method of the DownloadInputStream.

```
public int read(byte[] b, int off,
int len)
      throws IOException {
   if (startTime == -1)
      startTime = System.current-
TimeMillis();

   int read =
internalInputStream.read(b,off,len);
   bytesRead += read;
   return read;
}
```



File: http://www.jozilla.org/download/j2-dev-0.003.33.zip
Bytes read:94617
Size:282030
Transfer rate:21.843834 kb/s
Estimated downloading time:8 sec.

FIGURE 3 A running example of the download example

All you have to do is implement new methods that'll compute all the information you need to know about these streams with the new variables.

The transfer rate is quite simple – divide the number of bytes being read by the elapsed download time. The result is in bytes per millisecond. See Listing 1 for the exact implementation of this method.

The estimated remaining download or upload time is computed with an interpolation that relies on the transfer rate you've just computed. The hypothesis is that the rest of the data to be transferred will be downloaded/uploaded at approximately the same rate as the first part. The -1 value is returned if the full length of the input or output stream is unknown. The exact implementation of this method is shown in Listing 2.

EXAMPLE

Finally, since a bit of code is worth a thousand words, I've provided an example to prove how simple it can be to use these classes. It's a short program that downloads a given file from a URL on the local disk. While the file is being downloaded, data such as transfer rate, estimated remaining download time and a progress bar will show the user the state of his or her download (see Figure 3). See Listing 3 for the full example code.

Limitations

Keep in mind that the data provided by these classes must be carefully interpreted. If you build a program that calls the read(...) method every five minutes, you won't get the exact transfer rate and estimated download time whether you're using a 100Mb LAN or a dial-up modem. Since the interpolation is really simple, it can't handle extreme cases such as this.

With this in mind, you'll now be able to use these classes to provide your users with all the data they want to know. Since these classes can be used with the same methods as the common Java Input and Output Streams, you won't have to go through a learning curve.

fortrel@fortrel.net

Listing 1: DownloadInputStream.java

```
package net.fortrel.io;

import java.io.*;
import java.net.*;

/**
 * This is the Wrapper class for the Input
 * Stream.
 */
public class DownloadInputStream
      extends InputStream {

   // The Input Stream from which to read.
   private InputStream internalInputStream;

   // The length of the input stream, the
   // starting time of download and the number
   // of bytes being read.
   private int length = -1;
   private long startTime = -1;
```

Tidestone

www.tidestone.com

```
    private long bytesRead = 0;

    /**
     * Constructor if we know the length of the
     * input stream.
     */
    public DownloadInputStream(Input
    Stream is,
                          int length) {
       internalInputStream = is;
       this.length = length;
    }

    /**
     * Constructor if the length is unknown.
     */
    public DownloadInputStream(Input
    Stream is) {
       internalInputStream = is;
    }

    /**
     * Special constructor for a URL.
     */
    public DownloadInputStream(URL url)
          throws IOException {
       URLConnection connection =
          url.openConnection();
       this.length = connection.getContentLength();

       internalInputStream =
          connection.getInputStream();
    }

    /**
     * Get the current download rate in bytes
     * per millisecond.
     */
    public float getDownloadRate() {
       if (startTime != -1) {
          long currentTime =
             System.currentTimeMillis();

          return bytesRead / (float)(currentTime -
             startTime);
       }
       else
          return -1.0f;
    }

    /**
     * Get the time since the beginning of the
     * download in milliseconds.
     */
    public long getDownloadTime() {
       if (startTime != -1)
          return System.currentTimeMillis() -
             startTime;
       else
          return 0;
    }

    /**
     * Get the completion percent.
     */
    public float getDownloadedPercent()
{
       if (startTime == -1 || length == -1)
          return -1.0f;
       else {
          return (float)bytesRead /
(float)length;
       }
    }

    /**
     * Get the number of bytes read until now.
     */
    public long getBytesRead() {
       return bytesRead;
    }

    /**
     * Get the length of the input stream.
     * Returns -1 if the length is unknown.
     */
    public long getBytesTotal() {
       return length;
    }

    /**
     * Get the estimated remaining download
     * time in milliseconds.
     */
    public long getEstimatedDownloadTime() {
       if (startTime == -1 || length == -1)
          return -1;

       long currentTime =
          System.currentTimeMillis();

       return (long)((length - bytesRead) /
          getDownloadRate());
    }

/****************************************
 * All other methods are the implemen-
 * tation of the InputStream methods.
 ****************************************/

    public int available() throws IOException {
       return internalInputStream.available();
    }

    public void close() throws IOException {
       internalInputStream.close();
    }

    public void mark(int readlimit) {
      internalInputStream.mark(readlimit);
    }

    public boolean markSupported() {
       return internalInputStream.markSupported();
    }

    public int read() throws IOException
{
       if (startTime == -1)
          startTime = System.currentTimeMillis();

       int b = internalInputStream.read();

       if (b != -1)
          bytesRead++;

       return b;
    }

    public int read(byte[] b, int off, int len)
          throws IOException {
       if (startTime == -1)
          startTime = System.current-
          TimeMillis();

       int read = internalInput-
       Stream.read(b,off,
                    len);

       bytesRead += read;

       return read;
    }

    public void reset() throws IOException {
       startTime = -1;
       bytesRead = 0;

       internalInputStream.reset();
    }

    public long skip(long n) throws
    IOException {
       long skipped = internalInput-
       Stream.skip(n);
       bytesRead += skipped;

       return skipped;
    }
}
```

Listing 2: UploadOutputStream.java

```
package net.fortrel.io;

import java.io.*;

/**
 * This is the Wrapper class for the
 * Output Stream.
 */
public class UploadOutputStream
      extends OutputStream {

   // Private internal stream.
   private OutputStream internalOutputStream;

   // The length of the output stream, the
   // upload start time and the number of
   // bytes being writen.
   private int length=-1;
   private long startTime=-1;
   private int bytesWriten=0;

   /**
    * Constructor to use if the length is
    * unknown.
    */
   public UploadOutputStream(Output-
   Stream os) {
      internalOutputStream = os;
   }

   /**
    * Constructor to use when the
    * length is known.
    */
   public UploadOutputStream(Output-
   Stream os,
                       int length) {
      internalOutputStream = os;
      this.length = length;
   }

   /**
    * Get the upload transfer rate in
    * bytes per millisecond.
    */
   public float getUploadRate() {
      if (startTime != -1) {
         long currentTime =
            System.currentTimeMillis();

         return bytesWriten /
            (float)(currentTime-startTime);
      }
      else
         return -1.0f;
   }

   /**
    * Get the time since the beginning
    * of the upload in milliseconds.
    */
   public long getUploadTime() {
      if (startTime != -1)
         return System.currentTimeMillis() -
            startTime;
      else
         return -0;
   }

   /**
    * Get the completion percent.
    */
   public float getUploadPercent() {
      if (startTime == -1 || length == -1)
```

Object Switch

www.objectswitch.com

```
         return -1.0f;
      else {
         return (float)bytesWriten /
            (float)length;
      }
   }

   /**
    * Get the number of bytes being
    * writen by now.
    */
   public long getBytesWriten() {
      return bytesWriten;
   }

   /**
    * Get the length of the output
    * stream content to be writen.
    */
   public long getBytesTotal() {
      return length;
   }

   /**
    * Get the estimated uploading time
    * in milliseconds.
    */
   public long getEstimatedUploadTime()
{
      if (startTime == -1 || length == -1)
         return -1;

      long currentTime =
         System.currentTimeMillis();

      return (long)((length -
      bytesWriten) /
         getUploadRate());
   }

/**************************************
    * All other methods are the imple-
    * mentation of the OuputStream methods.
**************************************/
   public void close() throws IOException {
      internalOutputStream.close();
   }

   public void flush() throws IOException {
      internalOutputStream.flush();
   }

   public void write(byte[] b, int off, int len)
         throws IOException {
      if (startTime == -1)
         startTime = System.current-
         TimeMillis();

      bytesWriten += len;

internalOutputStream.write(b,off,len);
   }

   public void write(int b) throws
   IOException {
      if (startTime == -1)
         startTime = System.current-
         TimeMillis();
      bytesWriten++;

      internalOutputStream.write(b);
   }
}
```

Listing 3: DownloadDialog.java

```
package net.fortrel.io;

import java.io.*;
import java.net.*;
import java.awt.*;
import javax.swing.*;

/**
 * An example on using the DownloadIn-
 * putStream.
 */
public class DownloadDialog extends JDialog
      implements Runnable {
   private JProgressBar bar;
   private JLabel lTransfertRate,
               lEstimatedDownloadTime,
               lBytesRead,
               lBytesTotal;

   private DownloadInputStream dis;
   private OutputStream os;

   private Thread myThread;

   public DownloadDialog(Frame owner,
                       URL url,
                       String outputFile)
         throws IOException {
      super(owner, "Download", false);

      // Build the User Interface.
      buildUI(url.toString());
      pack();

      // Open every streams.
      dis = new DownloadInputStream(url);
      os = new FileOutputStream(outputFile);

      // Start the downloading thread.
      myThread = new Thread(this);
      myThread.start();
   }

   /**
    * A private method to build the
    * User Interface.
    */
   private void buildUI(String url) {
      JLabel lFile = new
      JLabel("File: " + url);
      lBytesRead = new JLabel("Bytes
      read:");
      lBytesTotal = new
      JLabel("Size:");
      lTransfertRate = new
         JLabel("Transfer rate:");
      lEstimatedDownloadTime = new
         JLabel("Estimated downloading
         time:");
      bar = new JProgressBar(
                 JProgressBar.HORIZON
                 TAL, 0, 100);

      // Do the layout.
      getContentPane().setLayout(new
      GridLayout(0,1));

      getContentPane().add(lFile);
      getContentPane().add(lBytesRead);
      getContentPane().add(lBytesTotal);
      getContentPane().add(lTransfertRate);
      getContentPane().add(lEstimated-
      DownloadTime);
      getContentPane().add(bar);
   }

   /**
    * The downloading thread method.
    */
   public void run() {
      byte buffer[] = new byte[1024];

      while(true) {
         try {
            int b = dis.read(buffer);

            if (b <= 0) {
               // The end of the down-
               // load is reached.
               os.close();

               System.exit(0);
            }
            else {
               // Save the data
               os.write(buffer, 0, b);

               // Update the on-screen
               // fields.

lBytesRead.setText("Bytes read:" +
                  dis.getBytesRead());

lBytesTotal.setText("Size:" +
                  dis.getBytesTotal());
               lTransfertRate.setText(
                  "Transfer rate:" +
                  dis.getDownloadRate() *
                  (1000f/1024f) +
                  " kb/s");

lEstimatedDownloadTime.setText(
                  "Estimated downlaod-
                   ing time:" +
                  (int)(dis.getEsti-
                  matedDownloadTime()
                  / 1000) + " sec.");
               bar.setValue(
                  (int)(dis.getDown-
                  loadedPercent() *
                  100));
            }
         }
         catch(IOException e) {
            System.err.println("Error:
" + e);
            break;
         }
      }
   }

   /**
    * The main method.
    * Use two parameters: the URL and
    * the filename to create.
    */
   public static void main(String   args[]) {
      if (args.length != 2) {
         System.err.println("Please
         give the URL"+
            " to connect to and the
              file store"+
            " the downloaded data.");
         System.exit(-1);
      }

      try {
         URL url = new URL(args[0]);
         String filename = args[1];

         DownloadDialog dd =
            new DownloadDialog(new
            JFrame(),
                           url,
                           filename);

         dd.show();
      }
      catch(Exception e) {
         System.err.println("Error:" + e);

         e.printStackTrace();
      }
   }
}
```

Download the Code!
The code listing for this article can also be located at
www.JavaDevelopersJournal.com

Compuware
NuMega

www.compuware.com

Interview... with PAUL LIPTON

DIRECTOR OF OBJECT TECHNOLOGY, COMPUTER ASSOCIATES

JDJ: Tell us about your Jasmine product.
Lipton: Most people are familiar with the current release of Jasmine, 1.21 – it's a powerful object database with some characteristics that people tend to associate with application servers. The typical object database is a passive repository for the state of an object, but in the case of Jasmine objects have both state and behavior. In a Jasmine 1.21 environment objects aren't just sitting there being persisted but are actually doing all kinds of work. That's how the current technology works, but of course what we're most excited about is what we're working on now.

JDJ: How about the new product?
Lipton: The new product is called Jasmine *ii* – Jasmine Intelligence for the Internet. It's a complete environment for rapid development of highly manageable, intelligent e-business solutions. We enable e-business applications to be intelligent using our patented Neugent technology as a Jasmine *ii* option. As you know, Neugents are the advanced neural network technology that we've developed, and I'll speak more about them in a minute. But Jasmine *ii* is also an open infrastructure for distributing, deploying and integrating all data providers and applications throughout the enterprise to the Internet and beyond. It's a synthesis of intelligence, middleware, development tools, persistent storage, integration capabilities and modeling. Also, Jasmine *ii* is an object-model neutral platform. We work with CORBA, COM and EJB in terms of object models and integrate with a vast array of databases and other unique data providers – everything from CICS to LDAP.

JDJ: Is Jasmine ii still a database?
Lipton: Jasmine *ii* is more than just a database. However, there will be an object database option called Jasmine *ii* ODB for those interested in using the object database subset to do the kinds of things that the object database does well, like optimized object persistence without the overhead of mapping, object caching and managing complex content. But what Jasmine *ii* is really about is building robust e-business applications and transforming existing systems into Web assets. To do this we had to go way beyond databases and what they can do. We had to add messaging, ORBs, global resource pooling and optimization, EJB support, Neugents, and so much more. We added modeling capabilities. And to make all this easy for developers to use, we created integration for the popular Java IDEs like VisualCafé, JBuilder and VisualAge. We don't force developers to use a particular IDE. In fact, we offer the same freedom to non-Java developers using Visual Basic or C++. The need for this kind of infrastructure hits developers all the time and in many different ways when they try to live up to object-oriented principles like code reuse.

There are many reasons why programmers don't reuse code that much. Let's face it, most of the data in the enterprise isn't even an object. About 80% of the data isn't even in relational databases. It's in prerelational databases, VSAM files on mainframes, e-mails, on green-screen terminals or whatever. In a world where e-business is demanding applications that are integrated into an extremely wide range of applications and data providers, as well as with partners and customers, that's one important issue. The other issue is that we do have these conflicting component models like EJB, COM and CORBA. To a varying extent they're not really that interoperable. How do you keep track of all their objects and organize them? The lack of a central catalog that understands all object models is also a problem. That's something that an object database can do really well. Does that mean that Jasmine *ii* is just an object database? Or that it only talks to object databases? Absolutely not. Jasmine *ii* talks to relational databases, too – such as Oracle, DB2, Sybase and Ingres, all the relational data providers but also nonrelational data providers.

In the real world there are real systems running on things like IMS, VSAM, IDMS and Datacom, and these systems are quite important. I know we'd like to erase it all and just start with this Java purity, but the reality of the matter is that these systems are running the bulk of today's businesses. If we doubt that, just look at the trillions of dollars that were spent on Y2K. So there's a real need not to be religious about this and integrate at multiple levels.

JDJ: So, is Jasmine ii all about integrating in a really big way?
Lipton: Integration is just one way that we think it will be uniquely positioned in the marketplace, because Jasmine *ii* is an infrastructure for building intelligent applications using the Neugent technology I mentioned earlier. Neugents work very differently from the conventional rule-based expert systems of the '80s and '90s. Rather than have a developer program in the business rules or do some kind of statistical analysis – which may be expensive and difficult to adapt to rapidly changing business conditions – Neugents are self-adapting and learn the way people do, from their own experience. We've been using them successfully in our Unicenter TNG product for quite a while now. Since Unicenter is an enterprise management product, it uses them to make predictions about events that will happen in the future. You know, instead of waiting for your e-commerce server to crash so you can panic, it posts a warning that says something like "there is a 90% probability of the e-commerce server crashing in 45 minutes." This is a lifesaver for serious business on the Internet. It happens because Neugents recognize complex patterns and learn from experience, just as humans do. In the case of Jasmine *ii* we've integrated the Neugent technology as an option allowing developers to create applications that can adapt to changing business environments.

JDJ: What would be the correct term for Jasmine ii? Is it an application server?
Lipton: The new term is *infrastructure.* Jasmine *ii* is an end-to-end infrastructure for creating e-business applications, particularly intelligent applications. The reason Jasmine *ii* isn't an application server is that the vision of it is so much greater. We've taken Java and added to it all our technologies in the areas of networks, enterprise management, security, databases, and much more that have taken us over two decades to develop. And no matter how talented you are, and how many clever people in your garage or office are working on the newest and greatest Java app server, you can't leverage decades of experience and technology if you don't have them to use.

JDJ: Are you going to be running sessions, etc., to train the new breed of Jasmine developers?
Lipton: Well, of course, we'll be offering extensive training, but any Java developer can be a Jasmine developer quite easily. Jasmine also has interfaces for C++ and COM for Visual Basic programmers because, again, we have to face the reality that these systems aren't going to disappear overnight. While we support Java, we have to be responsive to the genuine needs of all our clients.

Jasmine's architecture is quite extensible in terms of the kinds of services we can add to it. For example, it was easy to add XML support to Jasmine *ii*. But e-business today is putting tremendous competitive pressure on companies to differentiate their Web sites and their business systems. That's where the intelligence of Neugents comes in. It directly translates to an "above the line" competitive advantage.

JDJ: Basically, just as Java brought the threads and multiprocessing to the user very easily, you're bringing the world the neural nets and all the knock-on effects that it has to the end user.
Lipton: Personally, I see it continuing to evolve into an ever-more-useful, intelligent, and self-adaptive infrastructure. For example, at the core of Jasmine *ii* there's a lot of attention given to visualization technologies, particularly in the area of 3D. The object database is multimedia aware, so we're leveraging that to the next generation and applying Neugent intelligence and visualization technology in a sort of synergy. In the next few years what's going to really distinguish one e-business site from another is exactly the kind of application that Jasmine *ii* makes possible: an intelligent, self-learning, multimedia-aware application.

JDJ: When is it available?
Lipton: Well, it's in open beta, with some clients already using the new functionality for developing production systems. We're very enthusiastic about our delivery schedule, and are looking at a time frame measured in months to go to full general availability.

VisiComp

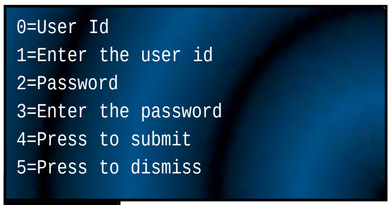
www.visicomp.com

An AWT Tip Window Control

A simple help mechanism for your Java program's GUI controls

WRITTEN BY
PAT PATERNOSTRO

A tip window (also known as a tool tip window) is a small popup window that displays a single line of descriptive text. Tip windows are usually displayed over toolbar buttons to provide textual help about a toolbar button's functionality. The tip window control is available for Swing components via the JComponent class's setToolTipText() method; unfortunately, the AWT doesn't have an equivalent control at its disposal. This article details the classes that constitute my implementation of an AWT tip window control.



```
0=User Id
1=Enter the user id
2=Password
3=Enter the password
4=Press to submit
5=Press to dismiss
```

FIGURE 1 The helptext.properties file contents

Implementation

The tip window control is made up of two classes, TipWindow and TipWindowMouseAdapter, located in TipWindow.java.

The TipWindow class (see Listing 1), which extends the java.awt.Window class, contains a single constructor and overrides the java.awt.Container paint() method. The constructor requires four parameters: a java.awt.Frame reference, a java.lang.String reference and two integer values. The constructor first calls the superclass's constructor, passing it the java.awt.Frame reference, which is required when constructing a Window object. Next, the java.lang.String reference is saved to a string instance variable that represents the tip window's help text. The tip window's background color is set to the ubiquitous yellow found on all tip window controls. A font is chosen and set for the tip window, and a java.awt.FontMetrics object is retrieved on the font currently set. The FontMetrics object is used (1) to calculate a *y* coordinate for the tip window's help text via the FontMetrics getAscent() method, and (2) to size the tip window using the values returned from the FontMetrics stringWidth() and getHeight() methods. The constant integer variable slack is added to the tip window's width and height to provide a buffer between the help text and the edges of the tip window. The tip window's location is set from the passed integer parameters, which represent an *x* and *y* coordinate, and the tip window is finally made visible via the java.awt.Component setVisible() method. Since the Window class doesn't contain any window decorations (e.g., border, title), the overridden paint() method draws a black border around the tip window via the java.awt.Graphics drawRect() method and draws the help text via the java.awt.Graphics drawString() method.

The TipWindowMouseAdapter class (see Listing 2) extends the java.awt.event.MouseAdapter class and contains a constructor and two overridden methods: mouseEntered() and mouseExited(). An adapter class is a convenience class that implements a specific event listener interface (e.g., MouseListener, WindowListener). The adapter class makes it easier to listen for specific events without having to implement all the event listener interface methods since they're already implemented in the adapter class. The TipWindowMouseAdapter constructor requires two parameters: a java.awt.Frame reference and a java.lang.String reference. The Frame reference is the same one that's passed to the TipWindow constructor and represents the tip window's container. The string reference is the name of a properties file (minus the .properties extension), which is a simple text file that contains key-value pairs. The key identifies the name of a GUI control and the value consists of the control's help text that's displayed in the tip window (more on this later). The properties file is opened and its contents placed in a hashtable for later retrieval via the java.util.ResourceBundle getBundle() method. Resource bundles, backed by properties files, are used primarily to internationalize a Java program. To learn more about resource bundles and internationalizing your Java program, look at the Internationalization section of Sun's Java Tutorial located at http://java.sun.com/docs/books/tutorial/i18n/index.html.

The mouseEntered() method first retrieves the component on which the mouse event occurred and the compo-

TIP WINDOWS IN AN APPLET

You can also create tip windows for an applet's GUI controls. Use the following code to retrieve the applet's java.awt.Frame reference:

```
public void init()
{
 Object parent;

 parent = getParent();
 while (!(parent instanceof Frame))
  parent = ((Component) parent).getParent();
}
```

You'll need to cast the java.lang.Object reference variable, parent, to a java.awt.Frame reference when calling the TipWindowMouseAdapter constructor. Also be aware that a warning banner will display on the applet's tip windows unless the applet is digitally signed.

I had mixed results creating tip windows for my applet's GUI controls. I created a JAR file containing all the necessary files (class and properties) since my ISP doesn't allow any files with a .properties extension. I launched the applet using Netscape Communicator 4.6 and kept receiving the following error: Applet exception: error: java.lang.ClassFormatError: Bad magic number. This error normally signifies that a class file is corrupt, most often because the file was incorrectly FTP'ed (i.e., binary mode wasn't specified on the transfer). Having checked that my FTP process was correct, I tried launching the applet again, this time using Netscape Communicator 4.7, and was successful – to a point. The previous error never manifested itself; however, the tip windows didn't display with a warning banner (as I expected) and their screen location was offset incorrectly, often nowhere near the control. I performed my next test with Microsoft's Internet Explorer version 4.0 (4.72.3110.8, to be exact). I didn't receive any errors, and the tip windows displayed – sort of. The problem with IE 4.0 is that the warning banner completely obscures the tip windows, rendering the control completely useless. Your mileage may vary with different versions of either browser.

IBM

www.ibm.com

nent's screen location. The tip window is created by instantiating a TipWindow constructor only if the ResourceBundle variable isn't null. The TipWindow constructor is passed the java.awt.Frame reference, the component's help text (retrieved via the java.util.ResourceBundle getString() method using the component's name as the key) and the component's *x* and *y* screen coordinates that are added to the mouse's *x* and *y* coordinates when the event occurred. The *y* coordinate has an extra value added to it to offset the tip window just below the cursor when it's displayed. The mouseExited() method disposes of the tip window and sets the tip window variable to null.

I've included a sample program (see Listing 3) that demonstrates using the tip window control. There are three requirements for using it:

1. A TipWindowMouseAdapter object must be instantiated.
2. A mouse listener must be added to every control that requires a tip window.
3. Every control that requires a tip window must be named via the java.awt.Component setName() method.

The last requirement is the most important, as the given component names are used as keys in the properties file that contains the tip window help text. Figure 1 shows the contents of the properties file used in the sample program. I chose a simple naming scheme that assigns a zero-offset numerical value to each component as it was added to the layout. Figure 2 shows the program with a tip window displayed.

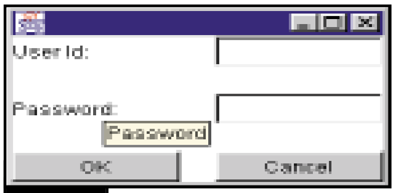


FIGURE 2 The TipWindowTest program

Summary

The code presented here is fully encapsulated and dynamic. No code changes to the tip window classes are necessary when GUI controls are added to or deleted from a Java AWT program. The tip window can be used as a simple help mechanism for your Java program's GUI controls.

ppaternostro@tricomgroup.com

AUTHOR BIO
Pat Paternostro is an associate partner with the Tri-Com Consulting Group located in Rocky Hill, Connecticut. Tri-Com provides programming services for a wide variety of development tasks.

Listing 1

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class TipWindow extends Window
{
 private String message;
 private int tipYCoord;
 private final int slack = 4;

 public TipWindow(Frame parent, String
 message, int xCoord, int yCoord)
 {
  super(parent);

  /* Set the message to display */
  this.message = message;

  /* Set the tip window's background color */
  setBackground(new Color(255,255,220));

  /* Set the tip window's font */
  Font f = new
  Font("Arial",Font.PLAIN,12);
  setFont(f);

  /* Retrieve the font's metrics */
  FontMetrics fm = getFontMetrics(f);
  tipYCoord = fm.getAscent() + slack/2;

  /* Set the tip window's size based on
     the font metrics */
  setSize(fm.stringWidth(message) +
  slack,fm.getHeight() + slack);

  /* Set the tip window's location */
  setLocation(xCoord,yCoord);

  /* Display the tip window */
  setVisible(true);
 }

 public void paint(Graphics g)
 {
  /* Draw the tip window's border */
  g.drawRect(0,0,getSize().width -
  1,getSize().height - 1);

  /* Display the message */
  g.drawString(message,slack/2,tipYCoord);
 }
}
```

Listing 2

```
class TipWindowMouseAdapter extends
MouseAdapter
{
 private TipWindow tp;
 private Frame parent;
 private ResourceBundle rb;

 public TipWindowMouseAdapter(Frame par-
 ent, String propFile)
 {
  this.parent = parent;
  try {rb =  ResourceBundle.get-
  Bundle(propFile);}
  catch(MissingResourceException mre){}
 }

 public void mouseEntered(MouseEvent evt)
 {
  Component comp = evt.getComponent();
  Point p = comp.getLocationOnScreen();

  try
  {
   if (rb != null)
    tp = new TipWindow(parent,rb.get-
    String(comp.getName()),p.x +
    evt.getX(),p.y + evt.getY() + 20);
  }
  catch(MissingResourceException mre){}
 }

 public void mouseExited(MouseEvent evt)
 {
  if (tp != null)
  {
   tp.dispose();
   tp = null;
  }
 }
}
```

Listing 3

```
import java.awt.event.*;
import java.awt.*;

public class TipWindowTest {

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

class TipWindowTestFrame extends Frame {
 Label lblUserId = new Label("User  Id:");
 Label lblPassword = new Label("Password:");
 TextField tfUserid = new TextField(10);
 TextField tfPassword = new TextField(10);
 Button ok = new Button("OK");
 Button cancel = new Button("Cancel");

 TipWindowTestFrame() {
  super();

  /* Set the layout */
  setLayout(new GridLayout(3,2,20,20));

  /* Add components */
  add(lblUserId);
  add(tfUserid);
  add(lblPassword);
  add(tfPassword);
  add(ok);
  add(cancel);

  MouseListener ml = new TipWindow-
  MouseAdapter(this,"helptext");

  /* Get the frame's components */
  Component[] comps = getComponents();

  /* Cycle through adding mouse listen-
   er and naming components */
  for (int i = 0; i < comps.length; i++)
  {
   comps[i].addMouseListener(ml);
   comps[i].setName("" + i);
  }

  /* Add the window listener */
  addWindowListener(new WindowAdapter()
{
   public void windowClosing(WindowEvent
                             evt) {
    dispose(); System.exit(0);}});

  /* Size the frame */
  pack();

  /* Center the frame */
  Dimension screenDim = Toolkit.getDe-
  faultToolkit().getScreenSize();
  Rectangle frameDim = getBounds();
  setLocation((screenDim.width - frameD-
  im.width) / 2,(screenDim.height -
  frameDim.height) / 2);

  /* Show the frame */
  setVisible(true);
 }
}
```

Download the Code!
The code listing for this article can also be located at
www.JavaDevelopersJournal.com

Digital
Parahna

www.digitalparahna.com

Progress SonicMQ

A solution for distributed application development

REVIEWED BY JIM MILBERY

AUTHOR BIO

Jim Milbery is a software consultant with Kuromaku Partners LLC, based in Easton, Pennsylvania . He has over 15 years of experience in application development and relational databases. Jim can be reached via the company Web site at www.kuromaku.com.

jmilbery@kuromaku.com

JAVA DEVELOPER'S JOURNAL
JDJ
WORLD CLASS AWARD

Test Environment
Client/Server:
Dell 410 Precision, 128MB RAM, 14 Gigabyte disk drive, Windows NT 4.0 (Service Pack4)

Progress Software Corporation
14 Oak Park
Bedford, MA 01730
Phone: 800 477-6473
www.progress.com

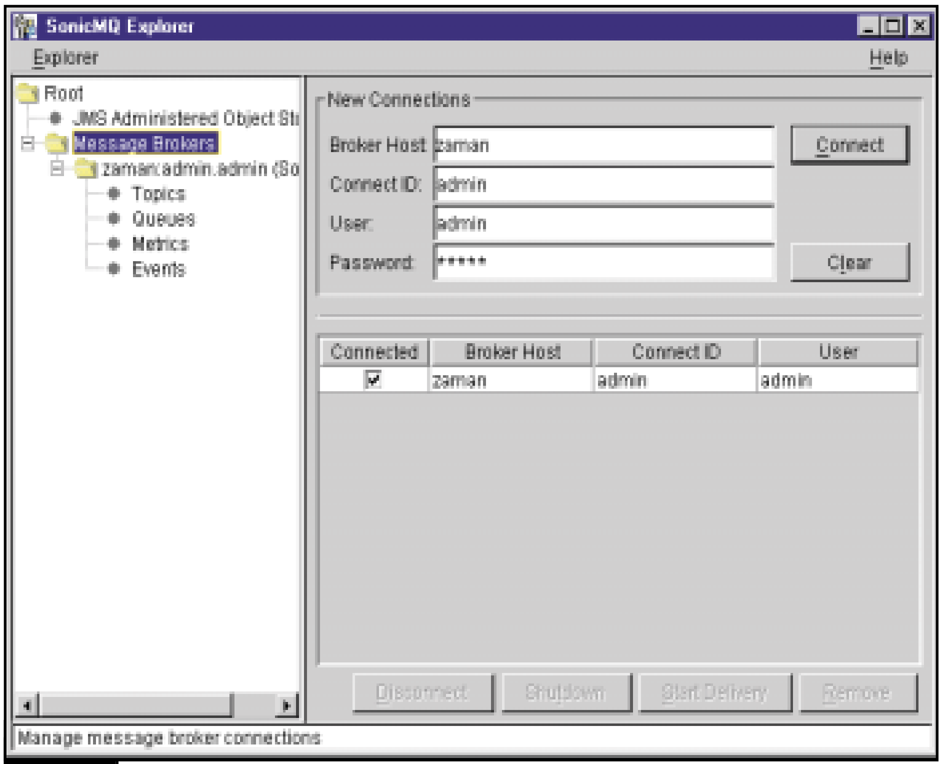
Distributed applications require the reliable transfer of information between the various layers of software. Guaranteeing the reliable movement of data between these layers is what keeps application architects up at night. This is particularly true when the application in question is deployed on the Internet and interfaces to disparate systems and environments. One solution to this problem is to make use of a message queue layer that can serve as the delivery mechanism for data. Message queue technology isn't a new idea, but thus far most of the solutions have been highly proprietary. With the release of their SonicMQ product Progress Software has weighed in with a new offering based on the Java Messaging API.

Progress first announced this product back in June when they demonstrated it as part of their application server. While many of the app server vendors are following a similar path, Progress elected to decouple the message queueing component from their server and offer it as a separate product, independent of any single application server. SonicMQ is built around JavaSoft's Java Message Service, a standardized framework for managing interapplication communications. One of the keys to understanding the need for message queue services is that the connection between disparate applications isn't necessarily a synchronous operation. Many corporate systems that offer a powerful, graphical user interface perform the majority of their transaction processing on older, more proprietary systems and applications. In such an environment it doesn't make sense to force the user (or front-end application) to wait for these back-office applications to complete their processing. As soon as you allow the front-end application to continue working while the back-end system is dealing with the transaction, you've crossed the line from synchronous processing to asynchronous processing. The trick is to ensure that the data is successfully processed by the back-end application. Placing a message queue in the middle of these applications is the answer to this problem.

Progress Software offers SonicMQ in three different packaging options. The Enterprise Edition is the most comprehensive, with support for an unlimited number of connections and a multibroker architecture. Small businesses (or interdepartment applications) can make use of the Small Business Edition, which has a limit of 50 concurrent clients. Programmers and third-party developers can work with SonicMQ through the Developer Edition, which is fully functional but has a limit of five connected clients and limited technical support.

Working with SonicMQ

I had the opportunity to see a preview of the Enterprise Edition, but I also took the time to register on the Progress Software SonicMQ site and download the Developer Edition of the software. The installation of SonicMQ under Windows NT was a breeze; I was able to get the software installed and configured in a matter of minutes. JMS itself comes complete with a set of high-level interfaces, but it's up to the mes-



Broker Host: zaman
Connect ID: admin
User: admin
Password: *****

Connected	Broker Host	Connect ID	User
✓	zaman	admin	admin

Manage message broker connections

FIGURE 1 SonicMQ Explorer

Evergreen

www.evergreen.com

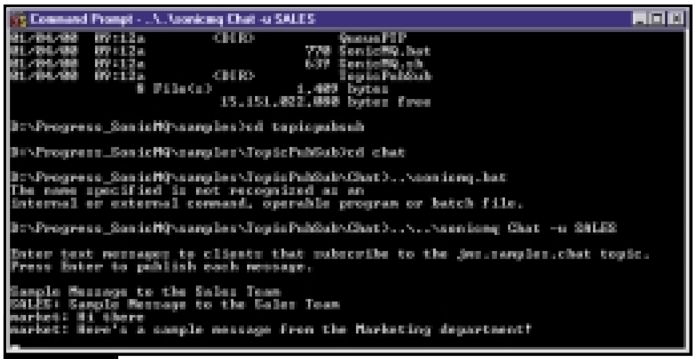
sage queue vendor (Progress in this case) to implement the various services such as load balancing, error handling, security and administration. Once you have SonicMQ installed, it's a simple process to start the service, and then you can use the Explorer to take a look at your message brokers (see Figure 1).

I found the user interface with Explorer to be a little sluggish, and some of the user interface objects performed erratically, but the interface itself is reasonably well designed. Through the Explorer you can track events, monitor the various queues you create and display charts that monitor key statistics, such as memory usage and response time. As the data store for the messages, SonicMQ supports a number of relational databases, including Oracle8*i*, and comes equipped with an embedded Java database that is automatically configured for you when you install the NT version, making it easy to get up and running with SonicMQ. You can control some aspects of the database configuration with SonicMQ, but you have to use external configuration files and batch scripts to do so. Even the command-line admin tool doesn't allow you to configure the database parameters, which is a little disappointing. It would be nice to have a single interface that managed all aspects of the server, but this isn't a fatal flaw by any means.

Progress supports the publish-and-subscribe model as well as the point-to-point model of messaging. They provide thorough sets of examples for working with both kinds of messaging with SonicMQ. I was very impressed with the wide array of samples that came with the development kit for SonicMQ. The folks at Progress took the time and effort to build a series of well-constructed examples and then documented them carefully in the online documentation (which was very thorough). Messaging can be a complex topic for developers, and it helps to have simple and concise examples at hand to learn the basics. I was particularly impressed with the Java source files that came with the examples. The developers left them clean and well organized, so those of you who aren't familiar with the ins and outs of JMS will find them easy to read.

As shown in Figure 2, I used the simplest example in the bunch to get started. Through SonicMQ you can broadcast a message to users who've asked to subscribe to the service. I found this to be a useful example of how to get started with JMS. Once you understand the basics, you can move on to the more sophisticated point-to-point sample applications. Most developers are familiar with the publish-and-subscribe model from applications such as chat. With this model a single message is delivered to groups of users that have registered themselves as being interested in the "topic" under which the message has been posted. You might leverage this capability in your own applications for broadcasting system or application messages to all currently connected users. Using JMS as the programming model provides you with a standardized framework, and using a commercial product such as SonicMQ allows you to focus on the logic and not the plumbing. However, from an e-commerce perspective the publish-and-subscribe model doesn't quite meet the needs of application developers. The alternative methodology is what JMS calls *point-to-point* messaging. Under this model a message can be delivered to a specific queue for a single client (the client being an application). Such a design ensures that a message is read once and only once by a specified application. In a scenario in which you're submitting an order to a back-office fulfillment system, this model becomes important to the success of the overall application. Once again, the sample applications were especially helpful in working with this messaging model. SonicMQ provides a "Talk" application that sends messages from one client application through a specified queue to a second application using the point-to-point model. The default installation of SonicMQ comes equipped with the necessary message queues already configured to make this example work; it was a simple process to start two separate client applications and send messages between the two. Part of the larger value proposition of a commercial message queue application is the ability to make them independent of the server. The Enterprise Edition of SonicMQ offers the ability to cluster brokers, but you can still test remote access with the Developer Edition. To prove this point, I stopped one of the client applications on my local server and moved the code to a remote workstation. The sample applications all accept parameters for the host address, port number and queue name, and it was a simple process to restart the client and have it point to the existing message queue on the other server. Just like that I was passing messages between client applications on different servers. (I even sent a message from the first client application before I restarted the second client on the new desktop, and SonicMQ kept the message queued up until the client application reconnected with SonicMQ.) Most of Progress's effort has gone into the design of the message server itself, so most of the sample programs contain only Java Message Server code. SonicMQ does come with two Java package APIs, one for interfacing with ActiveX and one for the SonicMQ Java client. The Java client package provides classes for dealing with some extensions to the JMS standard, such as handling XML messages. However, as the sample programs demonstrate, you can stick to the JMS standard API and build applications that are independent of the SonicMQ platform if you choose to.

"I'd recommend taking a look at SonicMQ as a solution for distributed application development"



```
Command Prompt - ..\..\sonicmq Chat -u SALES
01/04/00  09:12a      <DIR>          QueuePTP
01/04/00  09:12a                 770 SonicMQ.bat
01/04/00  09:12a                 639 SonicMQ.sh
01/04/00  09:12a      <DIR>          TopicPubSub
               8 File(s)          1,409 bytes
                         15,151,022,080 bytes free

D:\Progress_SonicMQ\samples>cd topicpubsub

D:\Progress_SonicMQ\samples\TopicPubSub>cd chat

D:\Progress_SonicMQ\samples\TopicPubSub\Chat>..\sonicmq.bat
The name specified is not recognized as an
internal or external command, operable program or batch file.

D:\Progress_SonicMQ\samples\TopicPubSub\Chat>..\..\sonicmq Chat -u SALES

Enter text messages to clients that subscribe to the jms.samples.chat topic.
Press Enter to publish each message.

Sample Message to the Sales Team
SALES: Sample Message to the Sales Team
market: Hi there
market: Here's a sample message from the Marketing department!
```

FIGURE 2 Chat example

Although the administrative interface with SonicMQ was workable, I was not as impressed with the overall functionality of the Explorer UI. Given the large number of enterprise features in the message server itself, I'd expect this to improve as SonicMQ matures.

Summary

Many of the application server vendors have announced plans to integrate message queue software into their application server products (including Progress). SonicMQ offers a cost-effective, flexible solution that's independent of the application server layer. Progress has architected the product for high performance and reliability (although I didn't test these claims). I'd recommend taking a look at SonicMQ as a solution for distributed application development, especially in consideration of the wealth of example code and documentation that comes with the product.

Applied
Reasoning

www.appliedreasoning.com

Functional Testing of Middle-Tier Servers

Use standard middleware to implement a generic test tool

WRITTEN BY
TODD SCALLAN
& THOMAS KERN

Critically important to the reliability of an application is how software components work together and how resilient they are to change. This article discusses how to perform functional testing on servers in the middle tier of distributed applications. We'll also address key middleware standards from the testing perspective, namely CORBA and Enterprise JavaBeans.

Testing plays a critical role in ensuring the reliability of software applications. To better understand what this means, we must consider the following questions:

- *How is software tested?*
- *When is software tested during the different phases of a project?*

Let's address the first question. Two methods of testing are used: functional and structural. The goal of the former is to prove that a software program conforms to its specification. Usually the tester reads the specification documents, creates a test plan and test data, applies the tests and checks whether the software behaves correctly. This method is commonly referred to as "black box" testing. Structural testing, on the other hand, uses the software program's source code for the creation of test cases. For example, you should ensure that all logical paths in the program code are executed at least once. This is known as "white box" testing. By looking at the source code, dependencies with other components can be determined and input data can be chosen accordingly.

Present-day software development often follows the object-oriented methodology. Here the waterfall model of "requirements—>design—>implementation—>test" has given way to a highly iterative and incremental development process.

This raises the second question: When is software tested? Software testing occurs at critical steps during the development and maintenance phases of a project:

- *Component testing* locates errors within new software components during development.
- *Integration testing* identifies errors in interactions and interfaces of new, untested modules. This can occur during the development and maintenance phases.
- *Regression testing* occurs during the maintenance phase, ensuring that software continues to work after modifications are made.

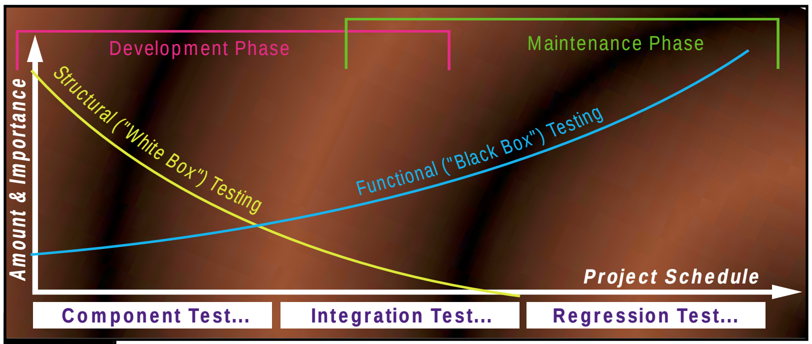


FIGURE 1 Functional testing increases in amount and importance during a project while structural testing decreases.

As a project advances, the amount and importance of functional testing increases, while the amount and importance of structural testing decreases (see Figure 1). Structural testing is very important during the development of a software component. However, during integration the focus shifts to interfaces and the interactions between software components and modules, not to the software's source code. Furthermore, if a project is based on third-party components, the source code is typically not even available, rendering structural testing infeasible. When a project enters regression testing, the source code isn't consulted at all.

Structural testing is limited to the definition and execution of tests by the software developer for a particular component's source code. (For Java classes it's also possible to perform structural testing to a limited extent without source code by analyzing the compiled Java and then generating test cases.) Far more important to the reliability of an application is how components work together and how resilient they are to change, hence the importance of functional testing.

Functional Testing of Nondistributed Applications

Before exploring how to test multitier applications, let's first look briefly at how nondistributed applications are tested. Tests can be performed at two levels: the user interface and the application programming interface.

At the UI level a tester simulates the end user, attempting to "break" the application. Many automated tools are available for recording and playing back user interactions with the UI. Although this method is popular with quality assurance departments, it's entirely black box, so the origin of errors is usually unknown.

At the API level a developer writes specialized test drivers. The objects or modules tested through the API must be linked with the drivers. If they don't have published interfaces, the developer usually performs the API-level testing.

Implications of Multitier Applications

Modern distributed applications are typically implemented in multiple tiers. Distribution has a significant impact on functional testing, as we'll see shortly. The three-tier architecture shown in Figure 2 has become quite pervasive, especially as applications continue to be enabled for Web access.

PointBase

www.pointbase.com

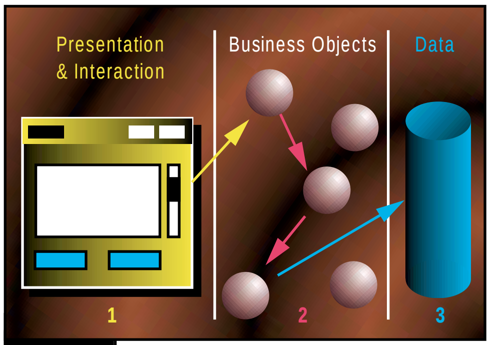


FIGURE 2 Three-tier architecture

Within the three-tier architecture the first tier represents the presentation and interaction layer, such as a Web browser in a typical e-business application. The middle tier consists of the application logic, which can be constructed as business objects. These business objects may be new applications or existing ones that are encapsulated so they can be integrated into the environment. The third tier includes data repositories, such as relational or object-oriented databases.

Business objects, the most prominent part of the three-tier architecture, are the building blocks of a distributed application. As such, they can be located anywhere across a network and can be accessed transparently by client programs regardless of physical location. The underlying communication infrastructure is responsible for ensuring that a business object can be found and accessed.

Business objects also provide transparency to application developers by hiding networking and communication details. This allows an object to be implemented in the programming language that's most appropriate for a particular task. Ideally, method invocations can be made in the native programming language of the caller and automatically mapped into the language of the target object.

Performing functional tests on distributed applications requires a significant amount of API-level testing, which is considerably more complex in comparison to testing nondistributed applications. A typical distributed application comprises many application modules that span different operating systems and network connections. There are multiple participants in the development and testing of a distributed application, which makes coordination a challenge. In the Internet age, changes to software can occur frequently with very short delivery cycles. Furthermore, changes to distributed applications, such as those supporting e-commerce on the Web, must be introduced incrementally without bringing down the entire system.

All of this complexity poses a dilemma for the tester. Which part of the application has failed? Who's responsible – the UI developer, business object developer or database administrator? What's going on within the business object? Using standard middleware can greatly alleviate this situation, as evidenced by the growing use of CORBA as the communication backplane for business objects, and Enterprise JavaBeans as a standard for server application components.

Standard Middleware for Objects and Components

Middleware provides a means for integrating business objects in the middle tier (see Figure 3). CORBA is a proven middleware standard for enabling object-level communication in multitier, multilanguage systems. From an application perspective, CORBA provides tremendous flexibility, but may present an abstraction level that's too low for some projects. In other words, the object paradigm supported by CORBA forces the programmer to define the application framework in addition to the business objects. (*Note:* The recently adopted CORBA Component Model should raise CORBA's abstraction level to be comparable with Enterprise JavaBeans. Please read on.)

The EJB specification is another middleware standard. It's much newer than CORBA, but is evolving – that is, maturing technically and gaining market acceptance – rapidly. EJB is a component paradigm that has gained recent prominence with the advent of application servers for three-tier development. In the EJB model a programmer writes code to implement specific business objects. A standard container supplied by someone else handles the rest. All requests from outside the programmer's component are directed to the container, which is responsible for executing the correct code within the object implementation. The container and the object communicate through a protocol that is well defined by the EJB specification.

EJB can be thought of as an abstraction layer that can exist on top of CORBA. So while standard middleware eliminates some of the uncertainty within a distributed environment, having multiple standards may also require testing at different levels.

Testing Middle-Tier Servers

Standard middleware provides a convenient way to develop business objects, but is it possible to perform functional testing of objects independent of an application user interface? Fortunately, the answer to this question is Yes.

The basic approach to functionally testing business objects is to substitute a generic test tool in the "Presentation & Interaction" tier (see Figure 3) that can work directly with the business objects. In other words, impersonate the expected caller of the server. Such a test tool must address the following three challenges:

1. *How to obtain information about object interfaces:* To solve this first challenge, published interface descriptions can be accessed through mechanisms provided by standard middleware. For example, CORBA provides the Interface Repository, which contains interface descriptions that can be accessed at runtime. EJB utilizes JNDI for locating objects and metadata descriptions for home and remote interfaces, bean type and business methods. A generic test tool would have a user interface through which information about a business object's interface could be

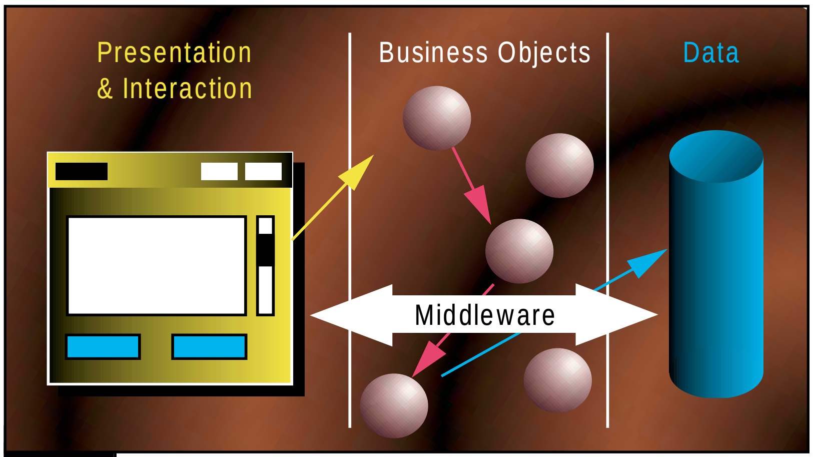


FIGURE 3 Three-tier architecture utilizing standard middleware

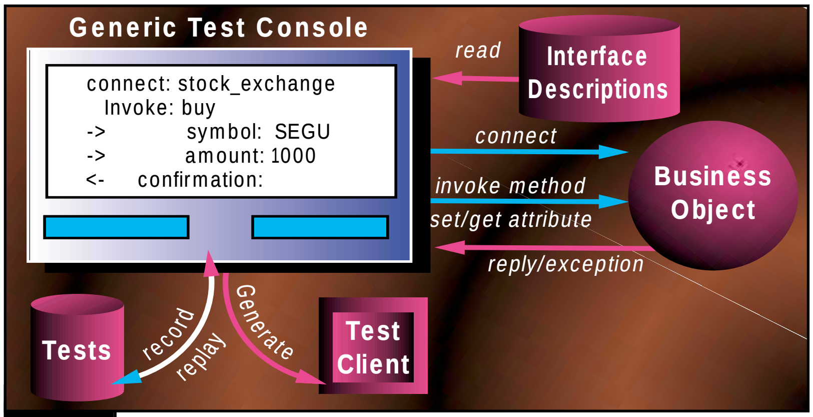


FIGURE 4 Testing a business object in the middle tier

presented. Details about an interface include available methods, method parameters, return values and possible exceptions.

2. *How to invoke methods:* In a generic test tool it would be impractical to require compilation of static object descriptions into the tool. Instead, dynamic invocation should be used to create method calls as needed by the tester. Fortunately, CORBA's dynamic invocation interface (DII) and Java's reflection API support dynamic discovery of interfaces and the creation of method calls on the fly. The test tool's usage model must allow selection of a method to test, visual creation and editing of arguments, and immediate viewing of results and exceptions.

3. *How to record and replay:* Requests and replies made through a tool should be saved using a suitable external data representation, such as XML. The user should be allowed to decide which actions to record and save for future use. The test tool would then interpret the external data to replay a session. The external data could also be used to generate standalone test clients. These capabilities are required to support regression testing.

Standard middleware, such as CORBA and EJB, makes it possible to implement a generic test tool (see Figure 4). This is achieved by meeting the challenges of obtaining interface details, constructing and executing method invocations, and recording test sessions.

Conclusion

When considering how to perform funtional tests on middle-tier servers, look for the following:

- *Time and money savings:* A good test tool should eliminate the need to build custom test programs and to manually create regression test cases.
- *Investment protection:* Test tools and procedures should be built on open industry standards so that technology investments are protected.
- *Productivity improvement:* Test procedures should be clear and intuitive. Test automation tools should offer ease of use.
- *Successful deployment:* All server components within a distributed system should be properly tested, thereby increasing your confidence that the application will work in production.

tscallan@segue.com / tkern@segue.com

AUTHOR BIOS

Todd Scallan is the director of product management for Segue Software's distributed computing products. He holds a BS in electrical engineering from Lehigh and an MS in computer engineering from Syracuse.

Thomas Kern has 12 years of experience in systems programming and application development for UNIX and NT. He is coauthor of a book about programming from X Window System and Motif.

QuickStream

www.quickstream.com

JProbe ServerSide Suite, V 2.5

A tool with powerful servlet and server-side application-tuning capabilities

REVIEWED BY GABOR LIPTAK

AUTHOR BIO

Gabor Liptak is an independent consultant with more than 10 years of industry experience. He is currently an architect of a Java e-commerce project.

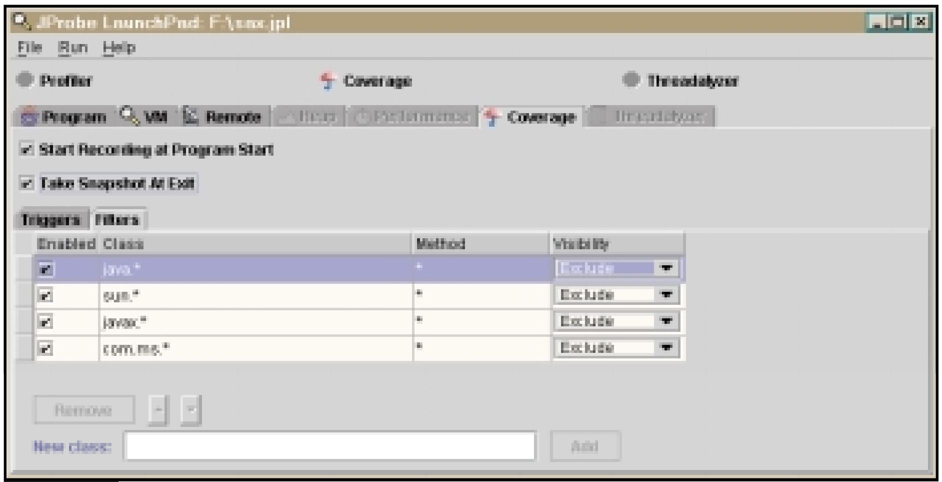
http://gliptak.homepage.com/

KL Group
260 King St. East
Toronto, Ontario
Canada M5A 4L5
Phone: 800 663-4723
www.klgroup.com
e-mail: javainfo@klgroup.com

Installation Requirements:
On Microsoft Windows 4.0 or Windows 95/98:
64MB RAM — 40MB disk space
Java 2 installed separately

On Solaris 2.6/2.7:
96MB RAM — 70MB disk space
Java 2 (reference version) installed separately

Pricing: ServerSide Suite: $1,899
(lower prices available for separate components and Professional Edition)



☑ Start Recording at Program Start
☑ Take Snapshot At Exit

Enabled	Class	Method	Visibility
✓	java.*	*	Exclude
✓	sun.*	*	Exclude
✓	javax.*	*	Exclude
✓	com.ms.*	*	Exclude

FIGURE 1 JProbe LaunchPad

The JProbe ServerSide Suite, version 2.5, consists of three related tools: a Profiler/Memory Debugger, a Threadalyzer and a Coverage product. All can use the LaunchPad to start profiling sessions against applications, applets and servlets, or use their built-in launch facilities

Each product takes a different approach to making your Java code more solid. The Coverage tool helps test all source code paths, the Profiler/Memory Debugger is used during development and later during production to tune the code, and the Threadalyzer helps to identify (potential) locking/synchronization issues.

With the direct support of Java 2 (using the JVM debugger and profiler interfaces), KL Group removed one of the main differentiators of the product. Earlier versions relied solely on instrumented JVMs to provide accurate measurements, but only approximated the real production environment (instrumented versions of JDK 1.1.7 and 1.1.8 are still supplied with the product). The Java 2 support is excellent, providing line-by-line detail not directly available using the JVMDI/JVMPI APIs. Note that the Solaris production version of Java 2 doesn't support the JVMDI or JVMPI required by JProbe, so you're still only approximating the real environment on Solaris. KL Group is working toward providing support for the Solaris Java 2 production version.

All three products use a similar metaphor of "snapshots" that can be loaded from and saved to a disk, given names and unloaded. These show up on the left-hand panels of the products. Snapshots can be redisplayed, analyzed and merged if appropriate. The right-hand panels display information specific to the snapshot, allowing drill-down actions where appropriate. Pattern-based filtering can be used to (de)select which packages/classes are of importance. Triggers are also provided that permit specific actions (e.g., starting/pausing/stopping/clearing event recording) when a class (or one of its specified methods) is entered or exited. Filtering and triggers allow you to set up sophisticated testing schemes, but you pay a maintenance price, especially on "moving target" complex products with frequent API changes. The tools let you control a JVM that's executing remotely, but the detailed debug information generated needs to be delivered in a separate way (perhaps mounted drives or something similar). Although there's no way to run all three tools in parallel against the same JVM, the same start-up files can be used for the different tools from the command line. For many tables in the product there are popup menus that allow you to show more columns in the tables. (In most tables package names weren't displayed by default.) I noticed GUI refresh problems and misreported line numbers; also, the keyboard scrolling doesn't seem to work in the built-in source browser. Command-line options are available for all three tools and their API is also documented in the help files.

Let's look at the tools one by one. When using the LaunchPad, you need to select one of the three modes of operation; the selected viewing tool can then be attached to the JVM that's running (see Figure 1). LaunchPad functionality can be invoked from the command line, allowing extensive configuration file and parameter-based scripting. In addition to controlling the data collection of LaunchPad by the other tools, this allows standalone data collection operation, with the resulting files later analyzed by the other tools. I found some inconsistent behavior when using the LaunchPad. It didn't restore the JVM version selected or the triggers for other modes of operation when switched between them. Sometimes the interaction between the LaunchPad and the other tools wasn't well synchronized. For example, in some cases the Profiler and the LaunchPad weren't pointing to the same snapshot directory or were running different JVMs, causing strange errors. These different settings may be desirable when running several instances of LaunchPad on different TCP/IP ports, but one would think that the two appli-

Sybase

www.sybase.com

cations would be synchronized during connection time. Programs can be run from the tools without using the LaunchPad, but in this case there seems to be no way to load/save the arguments set.

The Profiler/Memory Debugger has the busiest interface of the three products, because of its extensive visualization and drill-down capabilities (see Figure 2). The tool presents a twofold approach in which the Profiler generates both the timing and the number of object allocation snapshots, and the Memory Debugger generates object reference graphs in memory. While the timing approach is the more usual one, the number of objects generating information is of great importance because of the cost of creating/destroying objects. Also, analyzing object reference graphs helps identify potential memory leaks, when objects are kept from garbage collection, by referencing them unintentionally. Instead of generating differences between snapshots taken at the different times, the tool allows you to do counts relative to a cleared baseline. Even when helped with the well-done UI of the tool, the information presented doesn't always map easily into a good understanding of where the bottlenecks are in complex situations. Finding real problem areas requires iterations, and the ability to set up/save filtering and triggers definitely helps.

Threadalyzer (see Figure 3) has the fattest documentation of the three tools, indicating that it's conceptually different from the usual profiling tools. As described below, different levels of analysis are available, and running some of them slows the application to a crawl as it cross-inspects variables and where they're accessed. According to the documentation, thread is expected to behave differently under different JVMs, so it's wise to run your production JVM with the Threadalyzer tool, which is a concern with Solaris. The tool looks for the following thread-related problems:

- *Deadlocks:* When resources are locked in a different order on more than one thread
- *Potential deadlocks:* Holding a resource while in a wait state and locking order differences between threads
- *Thread stalls:* When a thread is in a wait state longer than the threshold set
- *Data races:* When different threads try to read/write the same variable at the same time; detected using "happens before" and "lock covers" algorithms

As analyzing requires no instrumentation of the code, it can also be used to discover problems with system or third-party libraries, and it may prove very useful in testing for multithreading issues in Swing code. A visualizer option is also included, allowing a quick overview of all threads and their state. The user will likely see many false alarms when using the tool, as mentioned in the documentation supplied, but the nice tool layout will help you narrow in on the important ones. The tool is useful to stress-test code with complex threading, but because thread-related states are difficult to (re)create, you may still want to continue reviewing any code with complex threading.

The easy-to-use Coverage tool fits nicely with the other debug tools, as it uses the line number in file information to create a line-level detail coverage map (see Figure 4). It lets you merge generated coverage maps and view the results of several runs, perhaps with different start-up parameters as one map. It also generates detailed summary reports in HTML format. The merging and reporting functionality are also available as standalone programs, and the Java API of the Coverage tool allows extensive automation.

JProbe hooks into Symantec's VisualCafé and IBM's VisualAge, allowing you to launch the tools from the menu and editing source files inside their environment. It also allows easy export of files from VisualAge's built-in source-control repository. The ServerSide Suite also provides templates to connect to several servlet engines (including JRun, WebLogic, WebSphere, Servlet Runner), enabling easy profiling of servlets in these different environments.

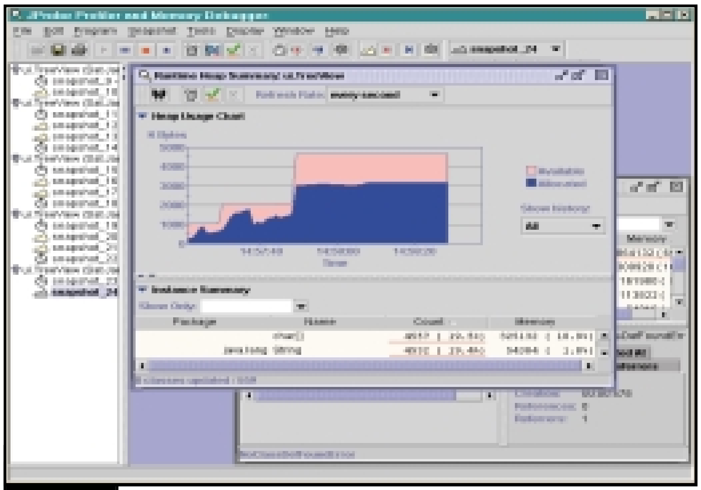


FIGURE 2 JProbe Profiler and Memory Debugger

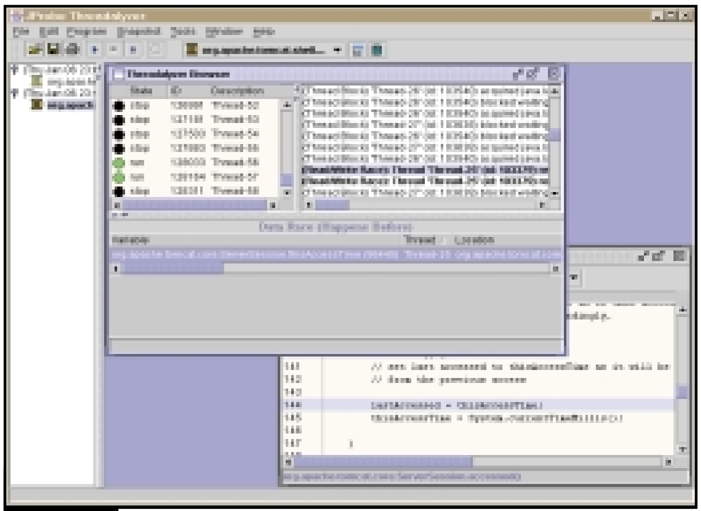


FIGURE 3 JProbe Threadalyzer

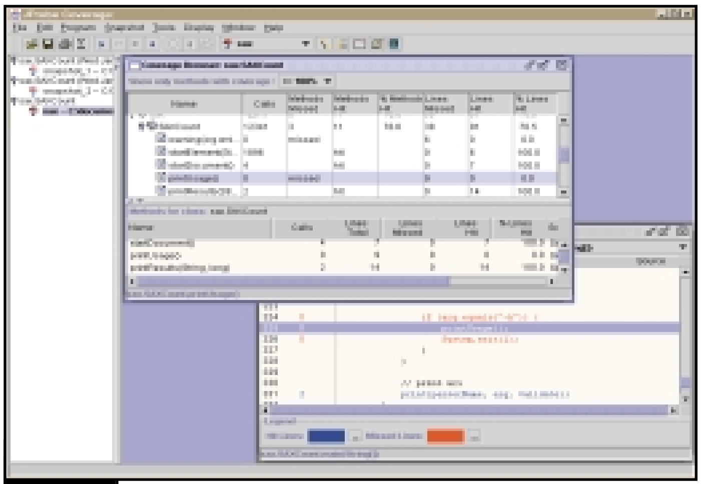


FIGURE 4 JProbe Coverage tool

I noticed several problems with the tools. I had crashes in both the JProbe tools and the JVM itself – maybe the JVMDI/JVMPI interfaces were overexercised. Another problem was nonintuitive handling of the "current directory" in complex projects – in many cases you need to repoint the tool to a source file you used a second ago even though it's relative to the current directory specified in your project and/or could be located using your CLASSPATH.

Even with these annoyances, JProbe is a recommended tool for both client and server-side code development.

MetaMata

www.metamata.com

The Application Server Turf

The one-stop shop for middle-tier applications

WRITTEN BY
AJIT SAGAR

Allow me to start this month's article with a short, albeit contrived, bedtime story. Once upon a time Jack and Jill, two childhood friends, moved into a backward village by the seashore. The residents of this village would catch fish using rudimentary tools they manufactured individually. Jack and Jill saw a business opportunity. Jill started manufacturing fishing nets, and Jack, fishing rods. Since Jack and Jill focused exclusively on these tools, their nets and rods were well designed and effective. Business thrived, the economy boomed and both friends prospered. In the meantime, two villagers, Jane and Joe, saw the opportunity to start their own businesses for manufacturing nets and rods, respectively. By this time, a rivalry had developed between Jill and Jack. Jill decided to expand her business by buying out Joe, who just had a little start-up shop, and she started offering discounted fishing nets with her rods. She would have bought out Jack, except that his business was too expensive. Jack retaliated by buying out Jane. To further cement their relationships, Jill married Joe and Jack married Jane, and they all lived competitively ever after.

This is the age of mergers, acquisitions and business marriages. In the early part of the last century there were mergers in the manufacturing and electronics industries. In the last quarter there was a wave of mergers and acquisitions in the telecom industry. That wave is still alive. One of the areas that has seen substantial growth in the past few years, especially this last year, is the field of application servers in the software industry. Tool vendors are defining and redefining their turf and areas of expertise. The norm is to establish oneself in a niche market, get a substantial cash flow going (or a perception of future profitability), seek out other companies that can complement and enhance the business, then partner, acquire, or merge businesses to gain a larger market share.

In December's **e-Java** column I wrote about how the middle tier may be split into two logical tiers: one to handle the business logic, the other to handle presentation of the data to the client. The middle tier is currently the territory of application servers. In this article we'll examine how application servers have grown from well-defined niche markets into one-stop shops for middle-tier applications.

“This is the age of mergers, acquisitions and business marriages”

Two Opposing Evolutions

There are two opposing evolutions taking place in today's application server market. One has resulted in the splitting of distributed architectures into multiple tiers. The other has vendors scrambling to own as many of these tiers as possible to expand their business. The three-tier model of distributed computing easily expands into an *n*-tier model. *Note:* The expansion of the tiers involves splitting the middle tier into functionally distinct tiers. The client and data end tiers remain more or less the same.

The middle tier is essentially an application service layer, which means the functionality of the middle tier makes it possible to serve up data to the layers in the upper tiers of a distributed application. As the technologies that enable these services became standardized, third-party vendors started offering these services. The middle tier in most three-tier applications isn't implemented as a monolithic program, but as a collection of components and services that are used in a variety of business transactions. This allows for more portable and reusable services. A side effect of this distribution of services is that different vendors have identified niche markets in which to start hawking their wares. Thus Web servers, TP monitors, groupware servers, pure database servers and the like evolved into middle-tier services.

While the vendors of existing server technologies migrated their products to the middle tier, distributed component models, such as EJB and COM, and distributed object frameworks, such as DCOM, RMI and CORBA, started solidifying. This opened up the market for new vendors to come into the arena by providing green field implementations of these completely new technologies. Then came the ORB vendors – Postmodern Computing, IONA and ObjectSpace – and the EJB vendors – WebLogic and Persistence. Once these vendors established a market niche, they started providing tighter integration with existing middleware services. This led to mergers and acquisitions. Some examples:

- Visigenic acquired Orbeline. Borland, a Java/C++/Delphi IDE vendor, acquired Visigenic to offer complete CORBA middle-tier services coupled with a development environment. The resulting company was renamed Inprise to better reflect their enterprise-level presence. The current application server offering is called Inprise Application Server.
- BEA Systems acquired WebLogic to combine TP monitor functionality with EJB offerings. Recently BEA acquired Symantec's VisualCafé Java development tool in conjunction with Warburg, an investment bank.
- Sun Microsystems acquired NetDynamics to add application server services to their existing toolset. They also acquired I-Planet to add VPN, authentication and virtual desktop services. Recently they acquired Forté to add e-

New Atlanta

www.newatlanta.com

“Not all of the application service providers have grown strictly through acquisition”

commerce functionality. The current offerings from Sun are in the form of the I-Planet Netscape Alliance Server.

- Allaire Corporation acquired BrightTiger technologies to add clustering and load balancing to their ColdFusion application server. This was followed by an acquisition of Live Software to add Java connectivity via servlets to their offerings. Recently the company acquired Valto systems, a small EJB vendor, to complete integration into Java middleware.
- SilverStream acquired GemLogic and ObjectEra to add XML integration to its application service offerings.

Not all of the application service providers have grown strictly through acquisition. IBM has developed a suite of application server products under its WebSphere family. Oracle offers an Oracle Application Server. Of course, the Microsoft platform offers competitive application services in the Windows environment.

Java and Application Servers

Web application servers have recently gained a lot of popularity since Sun completed its story on Java in the middle tier. Since J2EE was announced, it became clear what specs the application servers in the world of distributed computing will need to support. Virtually any application server vendor in the market today needs a well-defined strategy on how they're going to provide J2EE implementations in any product line that'll survive the next year.

XML and Application Servers

Application server vendors who want to stay in the market over the next few years will have to clearly define the hooks for XML integration. XML is likely to become the lingua franca of Web data formatting and presentation in the years to come. Vendors are either providing XML integration tools that are built in-house or are integrating with third-party XML integration tools. This year should see several mergers and acquisitions between existing application server vendors and XML product vendors.

Business Logic and Storefront

In December's column I identified two kinds of Web application servers that are representative of "vertical" tiers in a distributed architecture. One is the Web-front server that focuses primarily on building an online storefront via customer interaction and data presentation. The other kind of application server concentrates on business logic and integration to more complex legacy systems. As the application servers coalesce and extend their functionality, this line will blur. Two of the market leaders in these categories are Allaire's ColdFusion and BEA System's WebLogic application servers.

As mentioned above, both these product families have grown through acquisitions as well as green field implementations. It's interesting to note the different approaches taken by these two vendors. Allaire started at the Web front. Their initial offerings included Web site building products such as HomeSite and ColdFusion Studio, coupled with the ColdFusion Application Server for dynamic content generation. Allaire offered niche functionality that made ColdFusion popular by allowing HTML pages to easily access various data sources, thus eliminating the need for middle-tier integration to data stores. Hooks to security, mail and other services were also presented to complete the offering for building a storefront. To extend the other standard services that have become the flagship of application servers today, Allaire acquired BrightTiger for its clustering and load-balancing capabilities. Until then, Allaire had no presence in the Java world. The acquisition of Live Software enabled their Java connectivity. Allaire then announced Spectra, a packaged system built on ColdFusion for content management, e-commerce and personalization. They also defined WDDX, an XML-compliant serialization mechanism that allows applications to access objects written in different programming languages. With their recent acquisition of Valto Systems, Allaire's application server suite offers end-to-end services in distributed Java computing. Allaire says that it provides the only Transaction Server and Message Queue Server based purely on the Java platform.

BEA started from the other end. BEA's Tuxedo has been a very popular product in the world of transaction processing. BEA expanded into the Java middleware with its acquisition of WebLogic, one of the first vendors to adopt the EJB standard. This acquisition created one of the first companies to offer highly scalable and robust solutions in the EJB application server space. Because of their origin, the integration focus was on providing high-quality, reusable database drivers for accessing data stores in the data tier. IMHO, for highly scalable, high-transaction volume systems, the BEA product suite offers one of the best choices in the market. However, BEA didn't really have any sophisticated development tools to boast of. Their recent acquisition of Symantec's VisualCafé IDE completes their story on the Web front.

“Soon the application server vendors will start stepping on each other's toes”

Trading Places

The two examples of application server vendors I've highlighted in this article should give you a flavor of how the application server space is expanding. Note that both have their areas of strength – their turf, so to speak. Companies like BEA will, at least for the next year, continue to offer solutions in large, high-transaction volume installations. However, they'll gradually expand into the storefront and maybe target smaller companies as they gain market share. Companies like Allaire will probably look at capturing the mid-to-small installations, especially the ones that already have exposure to ColdFusion. However, soon the application server vendors will start stepping on each other's toes. For the development and e-business community this is all good news. We should see a lot more options, better prices and better support as the application server vendors are forced into competing with each other. At the same time, the drive for standards will ensure that consumers are not locked into specific vendor products.

ajit@sys-con.com

AUTHOR BIO
Ajit Sagar is a member of the technical staff of i2Technologies in Dallas, Texas, focusing on Web-based e-commerce applications and architectures. A Sun-certified Java programmer with nine years of programming experience, including three in Java, Ajit holds an MS in computer science and a BS in electrical engineering.

American Cybernetics

www.softexport.com

Domino and Java, Finally

Begin Domino development without learning a new language

WRITTEN BY
TONY PATTON

Much to their credit, Lotus recognized the momentum of Java. How could they miss it? Consequently, they moved quickly to integrate it into the Domino/Lotus Notes paradigm. Java support was added to the Domino family of products beginning with version 4.6 in 1998. The support was minimal, with documentation hard to obtain, but it was just the beginning.

During the same time, Lotus introduced the eSuite and Bean Machine products. Bean Machine is a rapid development environment for building Java applets. It was later sold to NetObjects (another IBM Company), makers of the Fusion Web site development tool. The eSuite family included a set of business productivity Java applets and a Java thin-client application for use on the once much-publicized network computers. It was recently discontinued by Lotus and will be covered later in the article. Lotus may have dropped one or two products, but Lotus Notes/Domino continues to flourish.

Domino

The latest incarnation of the Lotus Web Application Server, Domino 5, has fully embraced the plethora of Internet standards. It makes Domino a robust Web development server/platform. Java, along with the added support for numerous Internet protocols (such as IIOP, CORBA, HTML, POP, IMAP and SMTP), makes it reality.

The following features have been added and/or improved:

- The Notes/Domino IDE (Integrated Development Environment) has been expanded to support both Java and JavaScript. The IDE isn't a replacement for something like IBM's VisualAge or Symantec's VisualCafé, but it can verify syntax/grammatical errors.
- Java applets can be imported into Notes/Domino forms to take advantage of replication for distribution. Normally, Java applets need (must have) a Web server to be accessed by browser-based clients. This is still true, but the Notes client software provides full applet support.
- Domino data can be accessed remotely by applets on other Web servers/clients via CORBA.

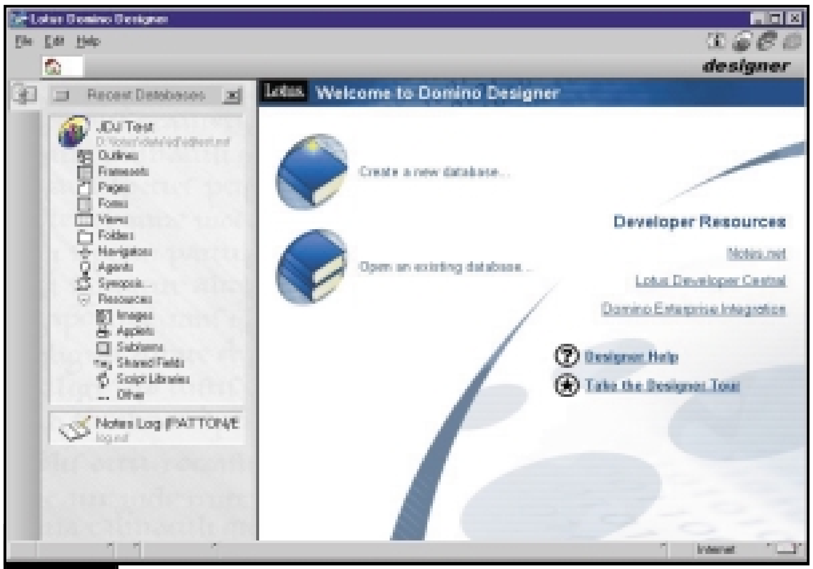


FIGURE 1 Domino Designer

- The world of Lotus Notes/Domino development has now been opened to Java developers. The Java interface can be used within the Notes development environment or by standalone Java applications or applets to access Domino data.
- The Domino server supports Java servlets that can be used to replace CGI programs. The Domino server includes a servlet engine; however, a third-party servlet engine can be used as well.
- The Domino HTTP server engine can serve up standard HTML files as well as Domino databases. The data within a Domino database is converted to HTML/JavaScript on the fly.

The Domino Designer application is used to develop Domino-based applications. Figure 1 shows the Designer client when first opened.

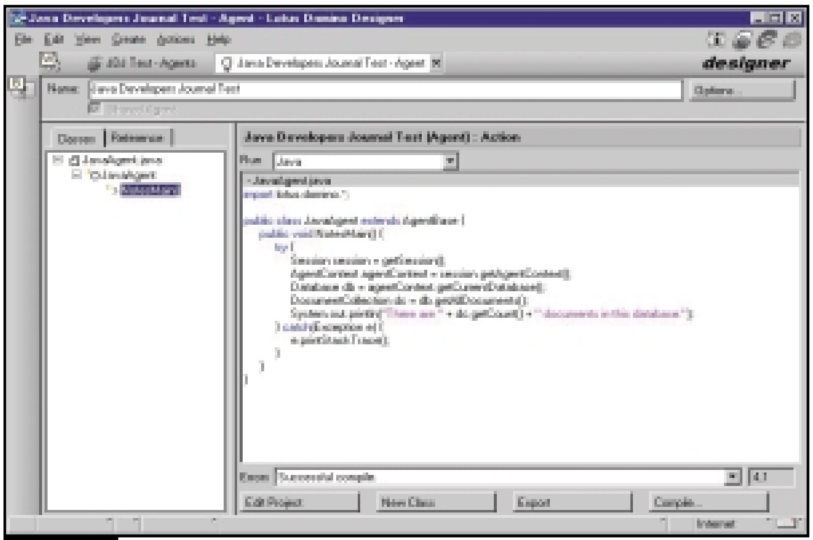
Application Structure

A brief introduction to the Domino environment is necessary at this point. A Domino application can be thought of as a group of object containers. The outermost container is the database, but the term *database* isn't used in the sense of the relational world of Informix or Oracle. A Domino database is a database of unstructured data stored as objects; it isn't a relational database. A database can contain document objects that in turn contain field objects. Documents contain data, and forms are used to enter and view the contents of a document. Document objects can be viewed using the Domino view or folder object.

At the most basic level, data is entered and presented in a Domino database via a form. The data entered/presented is stored as a document and is saved separately from the presentation/entry. Users browse a view or folder to select documents for viewing.

4th Pass

www.4thpass.com


Java Developers Journal Test - Agent - Lotus Domino Designer
File Edit View Create Actions Help
JDJ Test - Agents
Java Developers Journal Test - Agent
designer
Options...
Shared Agent
Classes
Reference
JavaAgent.java
JavaAgent
NotesMain()
Java Developers Journal Test (Agent) : Action
Edit Project
New Class
Export
Compile...
Internet

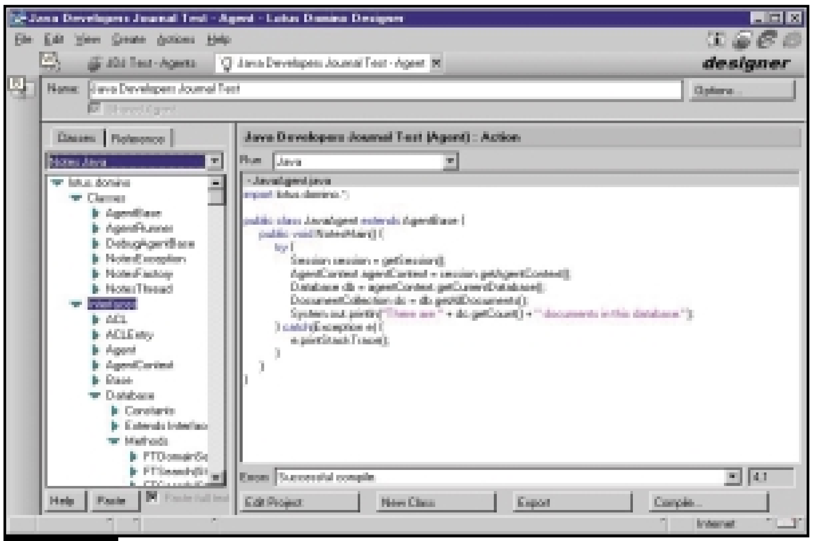
Name: Java Developers Journal Test

Run: Java

```
- JavaAgent.java
import lotus.domino.*;

public class JavaAgent extends AgentBase {
    public void NotesMain() {
        try {
            Session session = getSession();
            AgentContext agentContext = session.getAgentContext();
            Database db = agentContext.getCurrentDatabase();
            DocumentCollection dc = db.getAllDocuments();
            System.out.println("There are " + dc.getCount() + " documents in this database.");
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Errors: Successful compile

FIGURE 2 Domino Java Agent



Name: Java Developers Journal Test

Run: Java

```
- JavaAgent.java
import lotus.domino.*;

public class JavaAgent extends AgentBase {
    public void NotesMain() {
        try {
            Session session = getSession();
            AgentContext agentContext = session.getAgentContext();
            Database db = agentContext.getCurrentDatabase();
            DocumentCollection dc = db.getAllDocuments();
            System.out.println("There are " + dc.getCount() + " documents in this database.");
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Errors: Successful compile

FIGURE 3 Domino Java Help

Domino Java

Now that we have an elementary understanding of the Domino environment, let's take a look at where Java fits into the Domino development landscape. All Domino objects (database, forms, documents, views, etc.) are fully exposed and accessible in Java. Applets, standalone applications, servlets and native Domino agents can be developed.

Agents are standalone programs that perform a specific task in one or more Domino databases. They allow tasks to be scheduled, or triggered by an event or a user request. The functionality of an agent and a servlet overlap in many areas, but Lotus developed the agent structure before servlets existed.

Figure 2 shows a Domino database opened in the Designer client with a new Java agent being entered. The IDE allows you to name the agent, and the code pane is where the actual Java code is entered. The Help provided on the left-hand side of the IDE is a nice feature. It contains two tabs: Figure 2 shows the classes tab that shows the hierarchy of the current agent, and Figure 3 shows the online help available for traversing the Domino and Sun Java Classes' hierarchies.

The code we developed in Figures 2 and 3 instantiates a Domino database object and gets the number of documents in the database. It's a simple example that demonstrates the ease with which you can develop your own Java code.

Third-Party IDEs

A third-party IDE can be used by those with the need/desire for a more robust development environment. The Domino Designer IDE is functional and adequate, but the limits are quickly realized on large-scale development efforts. An IDE such as IBM's VisualAge for Java, Symantec's VisualCafé or Borland's JBuilder can be used with no problems. The environments of each must be properly set up before proceeding. This entails making the Domino classes available (JAR files located in the Domino installation directory) to the IDE, which may involve setting a path/classpath variable (Café) or actually importing the classes into the environment (VisualAge).

My favorite Java IDE, VisualAge for Java, includes instructions and samples for Domino development. This is no surprise given the fact that IBM produces both. It's nice to see two arms of the IBM behemoth working together.

JDK

You should be aware of the JDK version if you do choose to develop Domino Java code in a third-party IDE. The current version of Domino is 5, and it supports JDK 1.1.x. The JDK is part of the Domino application, and it can't be upgraded independently. That is, you have to wait for Lotus to issue an update that includes a JDK update before taking advantage of a newer JDK. This can be a bit frustrating. Lotus hasn't announced plans for adding JDK 1.2 support.

Conclusion

If you've made it this far, you may be asking, Why should I care? Well, you're a Java developer, and another piece of the development market has opened itself up to you. You have the Java skillset that enables you to quickly begin Domino development without having to learn a new language. The curve is steeper if you have to learn a new environment as well as a new language, so half the work is done.

This has to be one of the main reasons that Lotus has embraced Java with so much zest. A huge talent pool has just been made accessible to those needing Domino development help. IBM's massive Java push had to be another factor as well.

In my opinion, Domino is the most powerful Web development server/platform on the market today. It's a complete solution out of the box. It provides a robust security model, e-mail capability, full Internet standards support, workflow, enterprise integration and replication out of the box. The Domino server runs on all major operating systems, such as Solaris, AIX, AS/400, NT and Linux. Yes, it does have a version for Linux. Combine the power of Domino with Java and the sky's the limit.

tpatton@tisny.com

AUTHOR BIO
Tony Patton works with Transaction Information Systems, Inc. (www.tisny.com). Tony is certified in Sun Java, IBM VisualAge and Lotus Domino.

SIC Corporation

www.access21co.kr

Object

www.object

Design

tdesign.com

E-Business with EJBs

An e-business solution could mean the difference between life and death

WRITTEN BY
JASON WESTRA

expect great things from Enterprise JavaBeans this year, one of which is dominating the e-business front as the component model of choice for server-side application development.

Giving House Calls a New Meaning

E-business is multifaceted, encompassing e-commerce (monetary transactions over the Internet), business-to-business solutions and internal Web-based applications that provide flexibility and innovation in the services companies offer. E-business innovation has improved customer care and fostered repeat business for companies like Amazon.com, Dell and numerous online investment sites, to name a few.

However, early last year I was fascinated by the story a friend told me about her father, a radiologist from Boston, who used e-business technology to provide an extremely valuable solution to his customer, a patient with a fractured leg. During a visit to her home, my friend's father received a request on his cell phone for his opinion on a patient. He logged onto the hospital's Web site with his daughter's laptop and diagnosed the fracture by viewing X-rays over the Web!

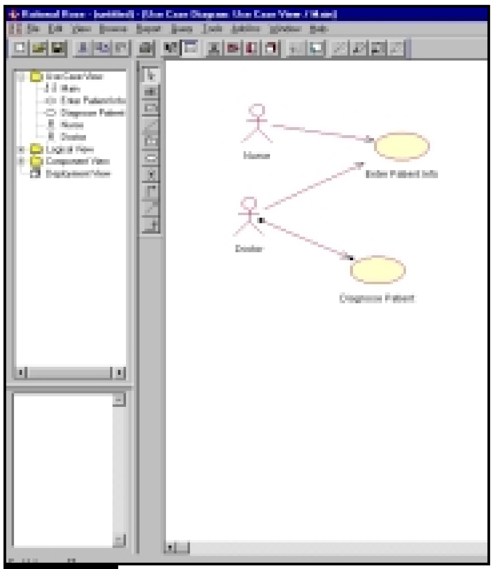


FIGURE 1 e-Patient use case diagram

This article provides a similar demonstration of the value of e-business above and beyond e-commerce. And if you had any doubts, the e-business solution I provide in this month's **EJB Home**, Acme HealthCare's e-Patient, is based on the following technologies: Enterprise JavaBeans and Java servlets. E-Patient is by no means industrial strength, but it's been developed with techniques not yet covered in **EJB Home**, such as EJB Handles and stateful session beans. I'll also provide ideas that I'll address appropriately in a future article on how to improve the application in the areas of robustness and performance.

Acme HealthCare's E-Patient Requirements Overview

The doctors at Acme HealthCare wanted the ability to enter and review patient information – even diagnose patients – by way of the Internet. In particular, doctors who had practices spanning multiple hospitals needed to be able to perform an initial "distributed" diagnosis from one location, then finish the patient assessment once they arrived at the hospital where the patient was located. Two use cases were modeled to cover usage of the e-Patient application (see Figure 1):

1. *Enter Patient Info*
2. *Diagnose Patient*

In Enter Patient Info the actor is either a nurse or a doctor. He or she enters the patient's history into the Patient Information Form and submits it. The next use case, Diagnose Patient, involves the doctor's entering a diagnosis and submitting it.

If a nurse was the actor in the Enter Patient Info use case, he or she must call the doctor to say that a patient record needs to be reviewed. The doctor who receives the call pulls up the patient record in a browser to make the initial diagnosis, which provides an avenue for others to treat the patient until the doctor arrives.

Acme HealthCare's E-Patient Technical Overview

The Java platform has all the necessary elements to solve the business requirements. The technical solution presented below specifically involves Enterprise JavaBeans with view and controller responsibilities handled in Java servlets. Included in the design are a simple static HTML form (see Figure 2) and a servlet that processes the submitted data. The current design doesn't include a database, but I'll discuss this below, along with other ideas for design improvements.

PatientSessionBean

At the heart of the system is the PatientSessionBean, a session bean that contains patient information and business logic. The bean is deployed as a stateful session bean to allow a client to access it multiple times without losing state between requests. Code for the enterprise bean component is given in Listings 1, 2 and 3 (remote interface, home interface and session bean, respectively).

An interesting feature of a stateful session bean is its ability to be passivated and activated by its container for memory and performance reasons. After a period of time has passed between client calls, the EJB container will store all non-transient, serializable attributes of the PatientSessionBean in some form of temporary persistent storage. This is usually just a file, but could be more robust depending on the container implementation. The next time the client calls the

Software Ag

www.softwareag.com

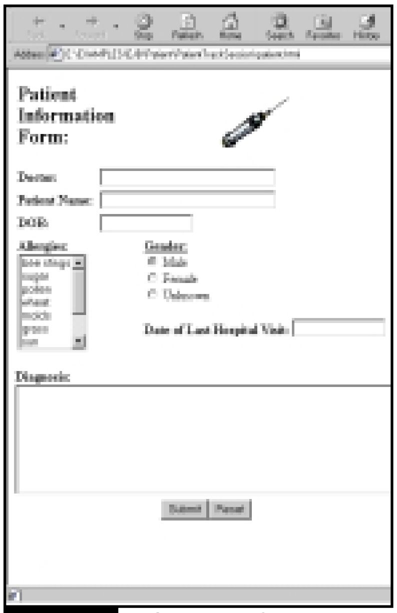


FIGURE 2 Patient HTML form

bean, the EJB container will allocate memory for the enterprise bean and service the request, a process called *activation*. Since the bean instance has been stored, a handle is used to return the bean to the active instance pool. EJB Handle objects are found through the EJBObject interface method getHandle().

Storing Handle objects can be done either persistently through serialization or you can just hold onto them in memory, as seen in Listing 4 of the PatientTrackSessionServlet.

While a session bean isn't meant to live beyond a hardware or software failure, passivation does provide a pseudo-persistent environment that allows EJB servers to handle higher volumes of users in a stateful server architecture.

User Interface

The entry point into e-Patient for the nurse actor in Enter Patient Info is the Patient Information Form, a static HTML form for entering information about the patient. When submitted, however, a Java servlet, as indicated in the tag, processes it:

```
<form method=POST action="http://ver-
getg1:7001/PatientTrackSession-
Servlet">
```

The tag is hard-coded to access my machine, probably not the best idea for a production application, but feel free to download the source code from www.JavaDevelopersJournal.com and modify it to suit your needs.

Once our Web server receives the HTTP request for submission, the PatientTrackSessionServlet processes it by parsing the form parameters and potentially modifying or creating a new PatientSessionBean. PatientTrackSession.java is not listed in full here.

The PatientTrackSessionServlet (see Listing 4) is multithreaded (e.g., a singleton servlet) and uses a hashtable to handle synchronization issues when multiple clients access it concurrently. It uses the hashtable patientCache to hold instances of javax.ejb.Handle, representing currently tracked PatientSessionBeans (active or passivated).

Implementing the Enter Patient Info Use Case

When a nurse or doctor submits the patient information to the system, the PatientTrackSessionServlet takes the patient information from the HttpRequest object, then checks to see if the patient is already entered into the system by checking the patientCache for a valid Handle object. If the patient isn't in the system, a new patient is created with the submitted information, and else use case, Diagnose Patient, is executed (see Listing 4).

Listing 1: Patient.java - remote interface

```
package PatientTrackSession;

import java.rmi.RemoteException;
import java.util.Vector;
import javax.ejb.EJBObject;

/**
 *  Patient is the remote interface representing a stateful
 *  session bean, PatientSessionBean.
 */
public interface Patient extends EJBObject
{
    public String getDoctor() throws RemoteException;
    public boolean isDoctor(String aDoctor) throws RemoteException;

    public void setName(String aName) throws RemoteException;
    public String getName() throws RemoteException;

    public void setDOBirth(String aDOB) throws RemoteException;
    public String getDOBirth() throws RemoteException;

    public void setGender(String aGender) throws RemoteException;
    public String getGender() throws RemoteException;

    public void setAllergies(Vector aAllergies) throws RemoteException;
    public Vector getAllergies() throws RemoteException;

    public void setLastVisitDate(String aLVD) throws RemoteException;
    public String getLastVisitDate() throws RemoteException;

    public void setDiagnosis(String aDiagnosis) throws RemoteException;
    public String getDiagnosis() throws RemoteException;
}
```

Listing 2: PatientHome - home interface

```
package PatientTrackSession;

import javax.ejb.EJBHome;
import javax.ejb.CreateException;
import java.rmi.RemoteException;

public interface PatientHome extends EJBHome
{
    public Patient create(String aDoctorName) throws Create
    Exception, RemoteException;
}
```

Listing 3: PatientSessionBean - session bean

```
package PatientTrackSession;

import javax.ejb.SessionBean;
import java.util.Vector;
import java.rmi.RemoteException;
import javax.ejb.SessionContext;
import javax.ejb.CreateException;

public class PatientSessionBean implements SessionBean
{
    // Business Attributes
    private String doctor;
    private String name;
    private String DOBirth;
    private String gender;
    private Vector allergies;
    private String lastVisitDate;
    private String diagnosis;

    // Business Methods
    public String getDoctor() {
        return doctor;
    }

    public boolean isDoctor(String aDoctor) {
        if (aDoctor == null)
            return false;

        String curDoctor = this.getDoctor();
        return curDoctor.equalsIgnoreCase(aDoctor);
    }

    public void setName(String aName) {
        name = aName;
    }
    public String getName(){
        return name;
    }
```

ComputerJobs.com

www.computerjobs.com

Implementing the Diagnose Patient Use Case

In this use case the patient's doctor is validated and the diagnosis is updated. Currently, only the patient's doctor has access to the patient's records. When the patient information is submitted and the patient currently exists, the Handle object referring to the PatientSessionBean is used to get a proxy to the session bean.

```
patient = (Patient)handle.getEJBOb-
ject(); // get EJBObject from handle
```

If the session bean had been passivated during this time, the container would activate it and return a proxy (EJBObject) of the component. Considering that a valid EJBObject is returned, the diagnosis is updated accordingly when the Web page is submitted.

The stateful session bean solution solves our business requirement of allowing a doctor to enter an initial diagnosis, then return to the patient record to update it once more information is known. The technical implementation of e-Patient is adequate to show some nice features of EJB such as Handles and stateful session beans. There is room for improvement, however. Let's look at some enhancements that could be made to this e-business solution.

Feature Enhancements

Our e-business solution for patient care has a number of areas in its business requirements and technical solution that could be improved. Patient confidentiality and data security are important in health care, so business requirements restricting access to patient records should be better defined. A login process should be implemented to prevent unauthorized users from accessing the system. Likewise, EJB's support of Access Control Lists (ACLs) could be used to restrict access in a role-based manner. For instance, nurses may have the ability to enter patient information but not make a diagnosis.

Also, stateful session beans are perhaps not the best solution for keeping patient records. They should be stored instead in a database to prevent loss of data in the event of a system failure.

Last, servlets are important in a Web-enabled Java application, but JavaServer Pages (JSPs) are increasingly taking precedence over servlet development. JSPs improve productivity by allowing HTML developers to script the user interface of the application while Java developers plug in Java code where needed. Migrating the servlets in our solution to JSPs would be an evolutionary move toward future productivity gains.

I'll expand this solution in a future article to include some of the new features discussed and cover more interesting areas of EJB in the process.

Conclusion

This month we explored the topic of e-business with Enterprise JavaBeans. We covered EJB Handles, accessing enterprise beans from a servlet and the possibility of designing stateful server architectures based on stateful session beans.

The Java platform provides a foundation to build complex Web solutions. When core business processes become e-business enabled, employees perform their jobs more efficiently, bringing value to other employees, suppliers and customers alike. For example, the added value that an e-business solution brings could be the difference between life and death!

Author Bio
Jason Westra is a managing partner with Verge Technologies Group, Inc., a Java consulting firm specializing in Enterprise JavaBeans solutions.

jwestra@verge-tg.com

```
public void setDOBirth(String aDOB) {
     DOBirth = aDOB;
}
public String getDOBirth() {
     return DOBirth;
}

public void setGender(String aGender) {
     gender = aGender;
}

public String getGender(){
     return gender;
}

public void setAllergies(Vector
aAllergies) {
     allergies = aAllergies;
}
public Vector getAllergies() {
     return allergies;
}

public void setLastVisitDate(String
aLVD) {
     lastVisitDate = aLVD;
}
public String getLastVisitDate() {
     return lastVisitDate;
}

public void setDiagnosis(String
aDiagnosis) {
     diagnosis = aDiagnosis;
}
public String getDiagnosis() {
     return diagnosis;
}

// These method(s) satisfy the Ses-
// sionBean interface contract
private transient SessionContext ctx;

public void ejbCreate(String aDoctor)
     throws CreateException, Remote-
     Exception
{
     doctor = aDoctor;
}

public void setSessionContext(Ses-
sionContext aCtx) {
     ctx = aCtx;
}

public void ejbActivate() {
     System.out.println("ejbActivat-
     ing..."+doctor);
}

public void ejbPassivate() {
     System.out.println("ejbPassi-
     vate..."+doctor);
}

public void ejbRemove() {
     System.out.println("ejbRe-
     move..."+doctor);
}
}
```

Listing 4: PatientTrackSessionServlet.java Detail

```
// get parameters from HttpRequest above…

// check to see if servlet has handle
// to the patient yet
        handle = (Handle)patient
        Cache.get(name);
        if (handle == null) {
            patient = newPatient(name,
            doctor, gender, dateOfBirth,
            dateOfLastVisit, allergies,
            diagnosis);

            // add patient's handle to cache
            patientCache.put(name,
            patient.getHandle());
        }
        else {
            // if patient already
            // exists in servlet
            // cache, get his/her
            // latest diagnosis from
            // the doctor.

System.out.println("Patient: "+name+"
handle found on cache.");
            patient = (Patient)handle-
            .getEJBObject(); // get
            EJBObject from handle

            if (patient != null) {
                // only same doctor
                // can make diagnosis
                if
(patient.isDoctor(doctor)) {
                    patient.setDiagno-
                    sis(diagnosis);
                }
                else {
                    out.println("You
                    are not authorized
                    to modify Patient
                    "+name);
                    out.flush();
                    out.close();
                    return;
                }

            }
            else {
                System.out.println("no
EJBObject for handle");
                out.println("ERROR:
Could not find patient!");
                out.flush();
                out.close();
                return;
            }
        }
```

Download the Code!
The code listing for this article can also be located at
www.JavaDevelopersJournal.com

Fiorono

www.fiorano.com

JDJ FEATURE

Unlocking the Secrets of the Java Media Framework

Part 1 of 3

JMF provides the multimedia building blocks for use in your applets and applications

WRITTEN BY LINDEN DECARMO

The multimedia objects in Sun's Java Development Kits are so primitive that they're worthless for serious development. Fortunately, Sun has overhauled Java's multimedia capabilities with the release of the Java Media Framework. In this article I'll explain why the JMF architecture is a significant improvement and show you how to use these objects in your applets or applications.

Struggling with AudioClip

If you've ever used the applet's AudioClip class, you've probably grieved over its limited functionality. The most irritating restrictions are its inability to determine the length of an audio file or how much of a file has already played. To be fair, AudioClip was designed for no-frills playback of AU digital audio content. Consequently, it can't play most digital audio files, or any digital video content or Musical Instrument Digital Interface (MIDI) files (see Listing 1).

JMF to the Rescue

Unlike AudioClip, JMF is a strategic API that's part of Sun's Media and Communications APIs. It supports most popular digital audio and video file formats along with MIDI files. Furthermore, you can use JMF 2.x to record (or capture) both digital audio and video files.

There are two types of JMF releases: pure and performance packs. The pure version can run on any Java platform (i.e., Linux and other forms of UNIX), while the performance packs contain code to maximize performance for either Win32 or Solaris platforms.

Pramati

www.pramati.om

Controller Method	Controller Event	New State
realize()	RealizeCompleteEvent	Realized
prefetch()	PrefetchCompleteEvent	Prefetched
start()	StartEvent	Started
stop()	StopEvent	Varies

TABLE 1 Controllers communicate state transitions with events

Understanding JMF Requires Time

Besides supporting a greater number of media formats, JMF also offers a richer selection of multimedia objects. Foremost among these improvements is the ability to manipulate time. In fact, virtually everything in JMF revolves around the clock interface and its ability to monitor time.

JMF clocks measure two types of time: TimeBases and Media Time. The former represents the constant flow of time from a known starting point (Greenwich Mean Time is a TimeBase). By contrast, Media Time describes the amount of time that has been consumed in a media stream. Unlike a TimeBase, Media Time can be stopped, can flow backward (rewind) or can advance at irregular intervals (fast forward).

Clocks use state to determine when they should be active. By default, clocks begin in stopped state and you start them by invoking their syncstart() method. Once started, a clock attempts to synchronize (or correlate) its TimeBase with its Media Time. Clocks remain started until you issue a stop(), the media stream ends or an exception occurs (see Figure 1).

Getting Everything Under Control

Although time is an important aspect of multimedia, a robust multimedia platform provides additional features such as resource management, error handling and support for threads. JMF supports these features in the controller interface, an extension of the clock interface.

Controllers enhance the clock interface by dividing the stopped state into five stages: unrealized, realizing, realized, prefetching and prefetched (see Figure 2).

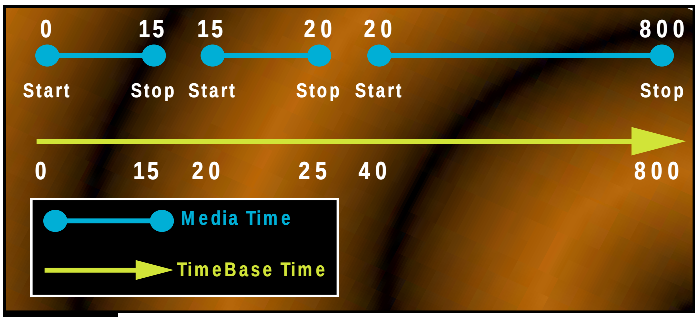


FIGURE 1 This TimeBase flows continually at 0.800 ms and isn't interruptible. The Media Time stops and restarts as the user pauses the stream.

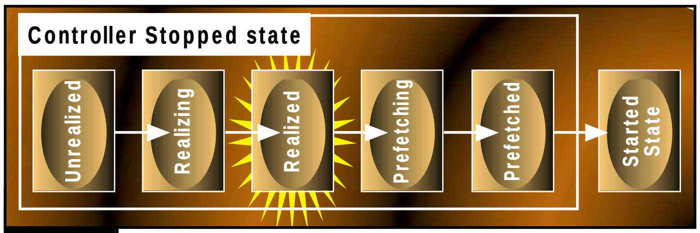


FIGURE 2 The controller divides the Clock's stopped state into five substates.

Controllers leave the unrealized state and enter the realizing state when they attempt to access the resources necessary to manipulate multimedia content. For instance, an audio controller will attempt to reserve resources on a sound card during realization. Once these resources are obtained, the controller becomes realized.

If you're using a nonrealtime operating system such as Windows 98, NT or Solaris, your programs may run irregularly. By contrast, multimedia devices consume large quantities of data at specific intervals. If your application can't obtain enough time slices to fulfill the demands of these devices, you'll hear the grating sound of audio breakups or see video frames being dropped.

The most common technique to prevent such problems is to pool buffers before they're needed. If your application can't satisfy the device's buffer demands, you can withdraw data from the previously filled buffer pool and stream it to the device, thereby preventing audiovisual hiccups.

The process of filling a controller's buffer pool is called *prefetching*. A controller leaves realized state and enters prefetching state when you request that it prefetch buffers (the numbers of buffers required will vary by controller). After these buffers have been obtained, the controller reaches the prefetch state. Once prefetched, a controller may be started.

The controller's subdivision of stopped state gives you increased granularity of control over multimedia resources. For instance, if a controller is able to realize, you know that all required hardware resources are functioning and available for use.

Another advantage of the multistate approach is the ability to detect errors. For instance, if the controller implodes while it's realizing, you know that it failed trying to acquire or initialize a multimedia resource. By contrast, if it failed to prefetch, the problem involved filling the prefetch buffers. Similarly, if it fails to start, it's probably due to an invalid media time or clock operation.

The final benefit of multiple states is the ability to maximize threads. For example, state transitions that occur rapidly (i.e., stopped to realizing and realized to prefetching) are done synchronously, while the transitions that can take extended periods of time (i.e., realizing to realized and prefetching to prefetched) are performed asynchronously. Realizing must be threaded since some hardware devices have long initialization times. Likewise, prefetching is threaded because input/output operations are often lengthy.

Never Assume

Controllers provide four primary methods to manipulate states: realize(), prefetch(), start() and stop(). The first method causes the controller to transition from unrealized to realizing and then to realized state. Similarly, prefetch() switches it from realized to prefetching and then to prefetched. The last two methods, start() and stop(), cause the controller to move into started and stopped state, respectively (see Table 1).

Because state transitions may occur asynchronously, you should never assume that the transition has completed when the controller's prefetch() or realize() methods return to your application. Rather, you should listen for the appropriate state transition event to be sent from the controller. Failure to listen for these events could result in an exception if you call an inappropriate method for a given state.

To receive controller events, you should register as a listener of the controller. Since controller events utilize the JDK 1.1 event model, if you're familiar with JavaBeans or AWT programming, you already know how to handle them.

```
// Add ourselves as a listener to the controller
player.addControllerListener(this);
```

Events permit you to monitor the state of a controller. Although most of the code in your listener method will be handling state transition events (see Listing 2), you should also listen for errors, media status events or error conditions. For instance, a well-written JMF application listens for the EndOfMediaEvent so that it's aware when playback stops:

IAM
p/u

```
// the EndOfMediaEvent indicates we've run out of data
     else if (event instanceof EndOfMediaEvent)
     {
          // rewind (i.e. set media time to 0
          player.setMediaTime(new Time(0));
     }
```

The Good Stuff

So far, we've concentrated on primitive objects and interfaces. Now we'll delve into the three components that let you play content: DataSource, Player and Manager.

DataSources are objects that retrieve data, place it in buffers and stream these buffers to client applications. Their main responsibility is to convert dedicated audio and video file formats into industry standard formats such as Pulse Code Modulation (PCM). This conversion process ensures that client applications can work with virtually any file format.

There are two types of DataSources: pull and push. *PullDataSources* supply buffers when your application requests them. Examples include those that read .AVI, .WAV and MIDI files. By contrast, *PushDataSources* stream data to your application when the data is available. TV and radio broadcasts are examples of PushDataSources since the content is continually being pushed toward you. They are more complex to use since you must handle situations in which a buffer arrives and you're busy doing something else.

MediaHandlers retrieve buffers from a DataSource, massage the data and transport the resultant buffers to a multimedia device. The most common type of MediaHandler is the *Player*, a MediaHandler that also implements the controller interface. When you request that a Player start(), it obtains data from its associated DataSource, updates its state information and sends the data to its destination device.

Managing the Missing Piece

If you simply want to play an audio file, the multitude of methods surfaced by DataSource and Player objects is intimidating. Furthermore, it can be a daunting task to find a MediaHandler to process a DataSource's output. Therefore, JMF provides the *Manager* object to shield you from these implementation details. The Manager not only finds the appropriate DataSource for your content, but it also constructs a Player, connects the Player to the DataSource and returns a Player reference to your application.

The only prerequisite for using the Manager is that you must pass it a *MediaLocator* (see Listing 3). MediaLocators inform the Manager of the data's location and the protocol that should be used to retrieve the data.

Once you've created a MediaLocator, you can call the Manager's createPlayer() method to construct a Player.

```
// create a Player with a Media Locator
player = Manager.createPlayer(mrl);
```

The first thing you should do upon successful creation of a Player is to add yourself as a listener. If you delay this call, you may miss vital warning events emanating from the Player.

Another technique you should practice is enclosing the initial interaction with the Player in try/catch blocks.

```
try
{
     // create a Player based on the medialocator
     player = Manager.createPlayer(mrl);
     // be sure to immediately listen for events...
     player.addControllerListener(this);
}
```

Listing 1

```
import java.applet.AudioClip;

public class LessPrimitive extends
java.applet.Applet
{
   AudioClip audioClip1;
   AudioClip audioClip2;

   // Applet role
   public void init()
   {

      // create two audio objects

      audioClip1 =  getAudioClip( get-
      CodeBase(), "clock1.au" );
      audioClip2 =  getAudioClip( get-
      CodeBase(), "clock2.au" );

      // play two audio files in loop mode

      audioClip1.loop();
      audioClip2.loop();

      try
      {
        // we still have NO clue how
        // long these files will take if
        // we want to play them one time
        // however, we put them in loop
        // mode, so they will play forever
        Thread.sleep( (long) 2000 );
      }
      catch (InterruptedException e)
      {
        System.out.println("Someone
        woke us up unexpectedly...");

      }

        // stop the first clip after 2
        // seconds

        audioClip1.stop();

        try
      {
        // we still have NO clue how
        // long these files will take if
        // we want to play them one time
        // however, we put them in loop
        // mode, so they will play forever
        Thread.sleep( (long) 7000 );
      }
      catch (InterruptedException e)
      {
        System.out.println("Someone
        woke us up unexpectedly...");

      }

      // stop the 2nd after another 7
      // seconds

      audioClip2.stop();

   }
}
```

Listing 2

```
public synchronized void controllerUp-
date(ControllerEvent event)
{
 // this event is received when the
 // Controller enters Realized state
 if (event instanceof RealizeCom-
 pleteEvent)
 {

        // event handling code goes
        // here.....

  }
 // this event is received when the
 // Controller finishes prefetching....
     else if (event instanceof Prefetch-
     CompleteEvent)
     {
        // event handling code goes
        // here.....

     }
}
```

Listing 3

```
MediaLocator mrl = null;
URL url = null;

// The applet tag should contain the
// path to the source media file, rela-
// tive to the html page.

if ((mediaFile = getParameter("FILE"))
== null)
{
     System.out.println("Invalid
     media.....");
}
try
{
     // create a MediaLocator based on
     // the filename
     url = new URL(getDocumentBase(),
     mediaFile);
     mediaFile = url.toExternalForm();
}
catch (MalformedURLException mue)
{
     System.out.println("Bad URL......");
}
try
{
   // Create a media locator from URL
   if ((mrl = new MediaLocator(media-
   File)) == null)
   {
    System.out.println("Can't build URL
    for " + mediaFile);
   }
```

InetSoft

www.inetsoftcorp.com

```
catch (NoPlayerException e)
{
    System.out.println("Can't find a player for " + mrl);
}
```

This is crucial because JMF Players can throw a multitude of exceptions during object creation.

New JMF programmers remember to create the Player inside a try/catch block, but frequently use methods that depend on player creation outside the block. For instance, should the Player creation in Listing 4 fail, the failure will be caught. However, the code then attempts to add itself as a listener to a nonexistent Player, which will result in an unhandled exception.

You Choose the Level of Complexity

The beauty of JMF is that casual programmers can use the Manager to create a Player, then issue the start() method to immediately begin playback of multimedia content. The following code illustrates the quickest technique to commence playback. It isn't necessary to understand all of the state gyrations a controller goes through, but you should listen for error events and catch exceptions.

```
public void start()
{
    // when the Web page is loaded, immediately start playback
    if (player != null)
    {
        player.start();
    }
}
```

JMF also satisfies the cravings of advanced programmers for low-level access to multimedia objects. For instance, an advanced application may test for the existence of multimedia resources. If resources are available, it will then prefetch buffers to improve responsiveness. Finally, it will automatically restart playback when the media stream finishes. All of these actions are possible if you listen and react to controller events.

To implement such a system, the applet in Listing 5 calls the Player's realize() method when the applet's init() method is invoked. If it receives a RealizeCompleteEvent, it knows that the audio hardware is functional. Consequently, it issues prefetch() to fill up the Player's buffers.

When the PrefetchCompleteEvent is sent by the Player, the applet displays a set of buttons to start and stop playback. Since prefetch() has filled the Player's buffers, this applet will start playback faster than one written by a casual programmer who calls start() while the Player is in the unrealized state.

Finally, when the Player runs out of media to process, it will fire the EndOfMediaEvent to the applet. The applet in turn rewinds the file to the beginning and restarts playback by calling start(). This will cause the media to repeat forever until the user hits the stop or pause buttons on the applet.

Is There More?

As we've discovered, JMF provides the multimedia building blocks for use in your applets and applications. The clock interface permits you to control the direction and speed of playback. The controller extends the clock to provide resource management, error handling and event tracking. DataSources and Players retrieve and process, respectively, multimedia data. Finally, the Manager object simplifies the usage of DataSources and Players.

In the next part of this series we'll explore how you can energize your JMF applications with emerging Internet multimedia standards.

Author Bio

Linden deCarmo is the author of Prentice Hall's Core Java Media Framework. He is a senior software engineer at NetSpeak Corporation, where he develops advanced telephony software for IP networks.

lindend@ibm.net

```
catch (MalformedURLException mue)
{
    System.out.println("Invalid media
    file URL!");
    System.exit(0);
}
```

Listing 4

```
try
{
    // create a Player based on the
    // medialocator
    player = Manager.createPlayer(mrl);
}
catch (NoPlayerException e)
{
    // exceptions will be caught
    // here...
    System.out.println("Can't find a
    player for " + mrl);

}

// if an exception occurred in Manag-
// er.createPlayer(mrl)
// then the player reference will be
// NULL and the line below will gener-
// ate another exception
player.addControllerListener(this);
```

Listing 5

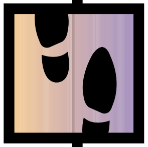
```
 // this applet will load the audio
 // file specified by the HTML page and
 // play it forever until stop or pause
 // is pressed

import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.URL;
import java.io.IOException;
import java.util.Properties;
import javax.media.*;
import java.lang.String;
import java.net.MalformedURLException;

public class AlohaJMF extends Applet
implements ControllerListener
{

    // this is the object that will
    // play the media files..
    Player player = null;

    // displays progress during download
    Component progressBar = null;


    // video window
    Component visualComponent = null;


    // GUI controls for position,
    // start, stop etc.
    Component controlComponent = null;


    int controlPanelHeight = 0;
    Panel panel = null;

    // attributes of the video window....
    int videoHeight = 0;
    int videoWidth = 0;

    // init reads the HTML params and
    // creates a Player based on those
    // parameters......
    public void init()
    {
     setLayout(null);

     // create the panel where we'll
     // show our Player
        panel = new Panel();
        panel.setLayout( null );
        add(panel);
        panel.setBounds(0, 0, 320, 240);

        // HTML page will tell us what to
        // play...
        String mediaFile = null;

        MediaLocator mrl = null;
        URL url = null;

        // Get the file to play from HTML...

        if ((mediaFile =
        getParameter("FILE")) == null)
            System.out.println("HTML page
            contains bogus file.");

        try
        {
            // create a MediaLocator based
            // on the filename
            url = new URL(getDocument-
            Base(), mediaFile);
            mediaFile = url.toExternal-
            Form();
        }
        catch (MalformedURLException mue)
        {
            System.out.println("Invalid URL");
        }

        try
        {
            // Create a media locator from URL
            if ((mrl = new MediaLocator-
            (mediaFile)) == null)
            {
              System.out.println("URL er-
              ror for:" + mediaFile);
        }
```

Building the NEW ENTERPRISE

JAVA DEVELOPER'S JOURNAL ANNOUNCES...

JavaCON 2000

CONFERENCE
September 24-27, 2000



EXHIBITION
September 25-26, 2000

SANTA CLARA
CONVENTION CENTER
Santa Clara, California

Join 100% Java Developers

Over 2,500 of your peers will be at JavaCon 2000, along with the industry's most respected technical experts, sought-after gurus and advanced users who will show you how to maximize Java for the enterprise. Make 2000 your year! Dedicate yourself to 4 days of Java intellect and master new skills from those who are defining Java's future.

W W W . J A V A C O N 2 0 0 0 . C O M

Java Developer's Journal

Puts a Twist on Today's Hottest Topics

JavaCon 2000 is your only opportunity to learn from the experts at *Java Developer's Journal* – who best combine technical expertise and practical vendor know-how.

Benefit by Attending

- *The tips and techniques you'll learn will help you do your job better.*
- *Discover new applications being developed today that you will need tomorrow.*
- *Sessions are designed for users at all levels with special sessions just for gurus.*
- *Network with fellow software developers as well as recognized Java experts.*
- *You'll learn how Java is being used for large-scale enterprise applications.*

Two-Day Exhibit

A full-scale exhibit hall packed with leading vendors will be on hand to demonstrate the latest products and answer your questions. All are welcome to fun-filled networking opportunities.

EXHIBIT HOURS
Monday, September 25, noon – 7:30 p.m.
Tuesday, September 26, noon – 5:30 p.m.

Are You a Java Guru?

Why Not Join Our Faculty?

If you're anxious to share your unique Java experience, please send an e-mail to **stewart@camelot-com.com** for details on how you may join the faculty. Topics of interest include but are not restricted to:

- Java in the Enterprise
- Embedded Java
- Java Success Stories/Real World Java
- Advanced Java Application Development
- XML & Java
- Java in the Industry/Java Business Applications
- Server-Side Java
- Java Testing and Debugging
- Object-Oriented Concepts and Design with Java
- Java and Open Source
- Component Software and Frameworks

CALL FOR PAPERS
Deadline: April 17, 2000
Acceptance: May 15, 2000

Take A Look at What You'll Learn:

- Building Mission-Critical Applications with Java
- Advanced JFC/SWING Component Integration
- Real-Time Java
- Designing High Performance e-Commerce Systems with EJB
- Combining XML and Java
- Java 2 Enterprise Edition
- Java Advanced Programming
- Java Security
- Java Runtime Internals
- Building a Multithreaded Server in Java
- Methods for Effective Java Unit Testing
- Developing COM and MTS Components in Java
- Developing Multitier Applications Using the Servlet API
- Object-Oriented Analysis and Design with Java
- Dynamic Bytecode Generation with Java
- JDBC Technology
- Developing Large-Scale Applications with Java and CORBA
- 2D and 3D Graphics in Java
- Designing Java Business Applications
- Java and Legacy Systems
- Using the Java Naming and Directory Interface API
- Java Gaming
- Programming for Devices (J2EE)
- Programming for the Desktop (J2SE)
- Java Commerce
- Using Java Agents
- Jini and JavaSpaces
- Java Testing and Debugging
- Java Exception Handling
- Garbage Collection Techniques
- Using Java with UML
- Writing Consumer Applications Using Personal Java API
- Smart Card Application Development
- Database Integration with Java

SPONSORED BY: JAVA DEVELOPER'S JOURNAL

PRODUCED BY: CAMELOT COMMUNICATIONS

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., Java Developer's Journal and Camelot Communications are independent of Sun Microsystems.

JavaCon2000

www.javacon2000.com

```
      else
      {
        try
        {
          // setup Player for media
             player = Manager.cre-
             atePlayer(mrl);
             // listen for events imme-
             // diately....
             player.addController-
             Listener(this);
             }
             catch (NoPlayerException e)
             {
                    System.out.println-
                    ("Player creation
                    failed for " + mrl);
                    System.exit(0);
             }
             }
      }
      catch (MalformedURLException mue)
      {
          System.out.println("Invalid
          media file URL!");
          System.exit(0);
      }
      catch (IOException ioe)
      {
          System.out.println("IO excep-
          tion creating player for " + mrl);
          System.exit(0);
      }


     }

     // this method is called on applet
     // creation and also when the page
     // is reloaded

     public void start()
     {
         // automatically begin playback
         // when the page is loaded....
      if (player != null)
      {
          player.start();
      }
     }

     // make sure to stop AND close all
     // resources when stop is called..
     // otherwise, we'll have resource
     // leaks and frustrated users...
     public void stop()
     {
         if (player != null)
         {
             player.stop();
             player.deallocate();
         }
     }

     public void destroy()
     {
      player.close();
     }


     // controllerUpdate is required to
     // listen for JMF events

     public synchronized void con-
     trollerUpdate(ControllerEvent event)
     {
      // Has our Player dead?
      // If so, nothing to do..event is
      // bogus.
      if (player == null)
          return;

      // When the player is Realized,
      // get the visual and control com-
      // ponents and add them to the
      // Applet
      if (event instanceof RealizeCom-
```

```
      pleteEvent)
      {
          if (progressBar != null)
          {
             panel.remove(progressBar);
             progressBar = null;
          }

          int width = 320;
          int height = 0;
          if (controlComponent == null)
               if (( controlComponent =
                     player.getControl-
                     PanelComponent())
                     != null)
                  {

                   controlPanelHeight =
                   controlComponent.get-
                   PreferredSize().height;
                   panel.add(controlCom-
                   ponent);
                   height += control-
                   PanelHeight;
                  }

          if (visualComponent == null)
                  if (( visualComponent
                     = player.getVisual-
                     Component())!= null)
                  {
                     panel.add(visual-
                     Component);
                     Dimension video
                     Size = visual
                     Component.getPre-
                     ferredSize();
                     videoWidth =
                     videoSize.width;
                     videoHeight =
                     videoSize.height;
                     width = video-
                     Width;
                     height += video
                     Height;

visualComponent.setBounds(0, 0, vide-
oWidth, videoHeight);
                  }

          panel.setBounds(0, 0, width,
          height);
          if (controlComponent != null)
          {
                  controlComponent.set-
                  Bounds(0, videoHeight,
                  width, controlPanel
                  Height);
                  controlCompo-
                  nent.invalidate();
          }

         }

         // the EndOfMediaEvent == no
         // more data
      else if (event instanceof EndOfMe-
      diaEvent)
      {
          // jump to beginning of content
          player.setMediaTime(new
          Time(0));
          // immediately restart playback...
          player.start();
      }
      else if (event instanceof Con-
      trollerClosedEvent)
          {
              // player is done, kill
              // GUI elements
          panel.removeAll();
      }
     }

}
```

Download the Code!

The code listing for this article can also be located at www.JavaDevelopersJournal.com

Meet JDJ EDITORS AND COLUMNISTS

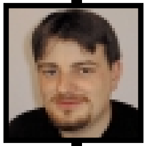
Attend the biggest Java developer event of the year and also get a chance to meet *JDJ*'s editors and columnists

JavaCON 2000

Sean Rhody *JDJ* Editor-in-Chief

Sean is the editor-in-chief of *Java Developer's Journal*. He is also a principal consultant with Computer Sciences Corporation where he specializes in application architecture – particularly distributed systems.

Alan Williamson Straight Talking Columnist



Alan is the CEO of n-ary (consulting) Ltd, the first pure Java company in the United Kingdom. The firm, which specializes solely in Java at the server side, has offices in Scotland, England and Australia. Alan is the author of two Java servlet books and contributed to the Servlet API. He has a Web site at www.n-ary.com.

Ajit Sagar e-Java Columnist

Ajit is a member of the technical staff at i2Technologies in Dallas, Texas, where he focuses on Web-based e-commerce applications and architectures. A Sun-certified Java programmer with nine years of programming experience, including three in Java, Ajit holds an MS in computer science and a BS in electrical engineering.

Jason Wesra EJB Home Columnist

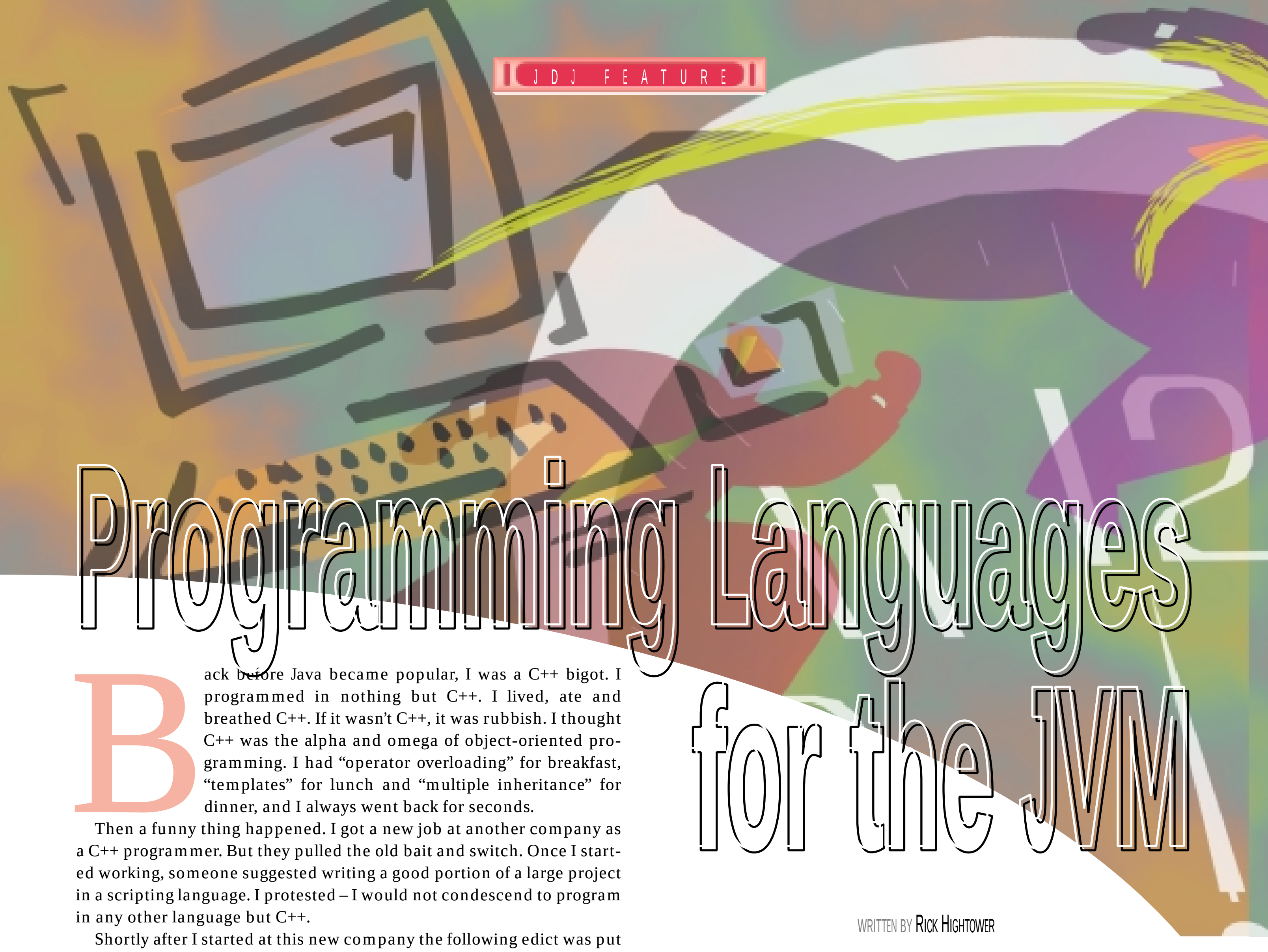
Managing partner with Verge Technologies Group, Inc., a Java consulting firm specializing in Enterprise JavaBeans solutions.

MEETING

September 24-27, 2000

Santa Clara Convention Center
Santa Clara, CA

JAVA DEVELOPER'S JOURNAL CAMELOT



Programming Languages for the JVM

WRITTEN BY RICK HIGHTOWER

Back before Java became popular, I was a C++ bigot. I programmed in nothing but C++. I lived, ate and breathed C++. If it wasn't C++, it was rubbish. I thought C++ was the alpha and omega of object-oriented programming. I had "operator overloading" for breakfast, "templates" for lunch and "multiple inheritance" for dinner, and I always went back for seconds.

Then a funny thing happened. I got a new job at another company as a C++ programmer. But they pulled the old bait and switch. Once I started working, someone suggested writing a good portion of a large project in a scripting language. I protested – I would not condescend to program in any other language but C++.

Shortly after I started at this new company the following edict was put forth: "Thou shall use a scripting language." Thus I was forced by management to write a good portion of the project in a high-level scripting language. They told us to glue components written in C++ together with this scripting language (in addition to writing components in C++). At first I hated it, as any self-respecting C++ bigot would. Then, gradually, the productivity of my team – and me – skyrocketed.

I became a true believer in scripting languages. The more I saw productivity climb, the less I coded in C++ and the more I coded in the scripting language. Granted, the scripting language had some limitations, but for many tasks it was just what the doctor ordered. Have you had a similar experience with a scripting language? If not, perhaps you should.

Scripting Languages

Many scripting languages are either object-oriented or object-based. Almost all of them are interpreted and use late-bound polymorphism. This makes scripting languages extremely dynamic and easy to program, which is essential for rapid application development (RAD), gluing components together and prototyping projects.

There's a fine line between a scripting language and a programming language. For example, Smalltalk is an extremely dynamic interpreted language, yet I dare you to call it a scripting language to a Smalltalk evangelist – you'll probably get punched in the nose. When I refer to scripting languages, I'm referring essentially to languages that are mostly interpreted and extremely dynamic, that is, they employ late-bound polymorphism and dynamic typing.

Java for the most part is a glorified scripting language – granted, a scripting language on steroids. Unlike most such languages, however, Java uses statically typed polymorphism; thus it has been called a hybrid language. At first this may seem like a disadvantage, but as it turns out, statically typed polymorphism is great for systems programming, framework definition and component development. Java's design by interface is truly great for large systems and frameworks.

Calling Java a system programming language to a system programmer is likely to evoke a nasty response (similar to calling Smalltalk a scripting language to a Smalltalk evangelist). However, don't view Java as a system programming language in the classic sense. Instead, view it as a virtual system programming language for a virtual system, that is, a virtual machine: the Java Virtual Machine (JVM). And Java, like its scripting language cousins, can be very dynamic – not as dynamic as Python, Smalltalk, and their ilk, but more dynamic than C++.

The funny thing is that the parts of Java that make it a great system language are the same parts that make it a mediocre scripting language. Don't get me wrong. Java is a great language. I jumped on the Java bandwagon as soon as I could hop a ride, and it's been good to me. However, Java is not as easy to use as scripting languages are. You really need a glue language in your toolkit. Build components and frameworks with Java, but glue the frameworks with a scripting language.

Don't fret! There's a history of using high-level scripting languages with system languages. Thus it's no surprise that there are scripting languages for Java. (*Note:* Don't get hung up on the term *scripting*; you can replace it with *programming*.)

Embar
cadero

www.

embarcadero.

com

Scripting Languages and System Languages: A Marriage Made in Heaven

For example, on UNIX systems many programmers program in C and C++, then glue modules together using higher-level shell programming (KornShell, C Shell, Bourne Shell, etc. Thus C is the system language and the shell scripts are the glue.

Another example: C and C++ programmers on UNIX often use Tcl (a scripting language) to do their GUI programming and glue together classes and libraries written in C++. C++ is the systems language and Tcl is the glue.

At other times C and C++ programmers will use Python as a control language – Python often ships with the UNIX system preinstalled. Python is easy to extend with a C. Again, C is the system programming language, Python is the glue.

The most prevalent example of an object-based scripting language is Visual Basic, which is often used to glue together COM components written in a variety of languages – C++, Delphi, and so on. Thus C++ is the systems language and Visual Basic is the glue.

Scripting Equals Increased Productivity

For more information on scripting languages and increased productivity check out the paper "Scripting: Higher Level Programming for the 21st Century" by John K. Ousterhout (www.scriptics.com/people/john.ousterhout/scripting.html). The paper basically states what I have experienced: namely, a sharp increase in productivity by using a higher-level language. The paper states that a scripting language is five to 10 times more productive than a strongly typed language like Java.

Scripting languages, however, don't replace a system language; they augment it. And five to 10 times more productive seems a little high to me. My personal experience has been two to three times as fast, depending on the application. Your results may vary.

Many developers are accustomed to higher-level languages like JavaScript and Visual Basic. Thus they may be more comfortable with them when migrating to the Java (i.e., the JVM). Often, especially for prototyping, a scripting language can be a more productive environment. In addition, Java programmers are hard to find; *good* Java programmers are nearly impossible to find. There's a lot more demand than supply.

You may want to use a scripting language for the following reasons:

- To extend an application, that is, an extension language
- To debug an application
- To learn and experiment with the Java API
- To prototype a system rapidly
- To glue together subsystems and components
- To automate testing and regression testing

You can embed scripting languages into your application so that users can extend it. You can use the dynamic interactive nature of scripting languages to inspect objects at runtime to debug an application.

The best way to learn the Java API is by experimenting. Working with dynamically typed, interactive scripting languages allows you to experiment quickly. Even though the Java API is documented well, it helps to try things out interactively. I do this all the time.

The properties listed above make scripting languages ideal for prototyping. Once you start using an interactive, dynamic scripting language, you won't stop. It's addictive and productive.

Java Was Built to Be Scripted

Java has wonderful features that make creating scripting languages easy. The class reflection and bean introspection APIs are a great basis for integrating these languages. Essentially, the scripting language can get metadata about a Java's class properties, events and methods. The scripting language can then use this metadata to change properties, handle events and invoke methods.

To learn more about introspection and reflection see the API documentation under java.lang.reflect.* and java.beans.Introspector. I've had the pleasure of doing metaprogramming with COM, CORBA and Java. Out of the three, I much prefer Java's reflection and introspection mechanism and APIs. It's a lot easier to use.

Scripting Languages for the Java Virtual Machine

Some may feel that the only language for the JVM is Java. They're wrong. Like many platforms (and Java is very much a platform), the JVM has many languages. And the list of those that work in the JVM seems to keep growing. Mixing Java with a scripting language for RAD is a powerful one-two punch that could make your next project fly.

Introduction to a Regular Column

What's been said so far is to welcome you to a new column in ***Java Developer's Journal***. This column will focus on topics like other languages for the JVM, integrating Java with mainstream scripting languages like Perl and Python, special-purpose languages (rules, etc.), creating JavaServer Pages (JSP) in JavaScript and Python, SWIG, integrating with C libraries, and others. This will be the resting place for non-Java–language JVM-related topics in ***JDJ***.

The Column's First Series

To kick off the column's birth, I'll start with a multipart series on programming and scripting languages that run in the JVM. These languages are Java-friendly, and often run in the JVM in both interpreted mode and as compiled Java classes, that is, they can be compiled to Java bytecode. These languages integrate well with Java classes and beans via the introspection and reflection APIs. They're great for prototyping, gluing together Java components and rapid application development. All of the languages are 100% Pure Java.

A lot of languages work in the JVM. Rather than just pick the ones I think are best, I want to solicit your feedback on which ones are most important to you. You can take part by going to the ***JDJ*** forum and voting. However, I've made a short list of languages that I think you should consider.

1. *JPython (Python)*
2. *Rhino (JavaScript-like)*
3. *Instant Basic (Visual Basic clone)*
4. *JACL (Tcl)*
5. *BeanShell (Java-like)*
6. *Bistro (Smalltalk-like)*
7. *Skij (Lisp/Scheme-like)*

Everyone has a favorite language, and this list represents the most mature and common languages for the JVM. Most are easy to obtain, and quite a few are open source (GPL or GPL-like licenses). If you feel I'm missing a major language, let me know. We love feedback here at ***JDJ***. The following sections give some background on each language.

Short Drill-Down of Different Languages

- ***Python (JPython)***: Java implementation of Python, a high-level, extremely dynamic, object-oriented language. JPython is very close to Python, and has been certified 100% Pure Java. Recently, NetBeans (the Java IDE maker that was bought by Sun) had a poll on their Web site regarding integrating a scripting language with their Java IDE. JPython won by a landslide. If you've used it, you know why.

  You can develop JSP in JPython – called PSP for Python Server Pages. Python and JPython are open source. To learn more about JPython visit www.jpython.org and www.python.org. To learn more about Python Server Pages, which run in a Java Servlet server, see www.ciobriefings.com/psp/. (PSP is freely available.)

- ***Tcl (JACL)***: Java Command Language (JACL, i.e., Java Tcl) is a Java implementation of Tcl 8.x. With JACL you can write scripts for Java components and APIs. In addition, there is Tcl blend, which allows manipulation of Java objects directly from Tcl. Tcl is open source. To learn more about JACL see the Scriptics Web site at www.scriptics.com/products/java/. Many Tcl users have claimed significant reduction in development costs. Tcl is the definition of RAD.

JavaScript (Rhino): Java implementation of JavaScript 1.5. JavaScript is a very powerful, object-based scripting language. The name *Rhino* is from the rhino on the cover of *JavaScript: The Definitive Guide* by David Flanagan (O'Reilly & Associates). The freely available Rhino is open source and its code is covered by the NPL (Netscape Public License). Rhino is based on JavaScript 1.5, which is based on ECMAScript (standard ECMA-262 ECMAScript, a general-purpose, cross-platform programming language). Since many developers have written JavaScript, Rhino is a natural fit for doing rapid application development and prototyping in the JVM.

- ***JavaScript-like languages/services that deserve further investigation***: PolyJSP, JSP for JavaScript and WebL; Resin, another JSP for JavaScript; and FESI, a free EcmaScript.

- ***Visual Basic clone (Instant Basic)***: Halcyon's Instant Basic for Java is a Java implementation of a Visual Basic clone. They've cloned the IDE, database components, and so on. This would be great if you have a lot of Visual Basic programmers. This also allows you to quickly port existing Visual Basic applications to the Java platform. In addition, Halcyon has an iASP (Active Server Pages) version of JSP in which you can develop cross-platform Active Server Pages. JSP was Sun's answer to Microsoft's ASP. iASP is an ASP clone that works with Java; thus you can use VBScript and JavaScripts to access all types of components (JavaBeans, CORBA, EJB, etc.). To learn more about Halcyon's Visual Basic products go to www.halcyonsoft.com.

- ***Java (BeanShell)***: BeanShell is interpreted Java. The syntax is very much like Java, i.e., BeanShell executes Java statements and expressions. Like other scripting languages, BeanShell is dynamically typed; thus much of Java syntax for type declaration and casting is optional. This is a wonderful language for writing prototypes and learning new APIs (new to you) through experimentation. In addition, BeanShell, as the name implies, adds extra support for dealing with beans. I find it very easy to use, and it's freely available and open source under the GPL license. To learn more visit www.beanshell.org. (You may also want to check out Dynamic Java, which seems similar to BeanShell, at www.inria.fr/koala/djava/.)

- ***Smalltalk (Bistro)***: Bistro is a Smalltalk variant with extensions for Java features and integration. It offers software developers the ability to code in a syntax that is very readable and expressive. Like BeanShell, Bistro is dynamically typed with the option of being statically typed for closer integration with Java. Thus you can mix and match statically typed systems with those that are dynamically typed. For prototyping, the dynamically typed languages make a lot more sense. Smalltalk is purely dynamically typed. This variant has a good mix of the Java type safeness with the Smalltalk fast development. For more information on Bistro see www.jps.net/nikboyd/bistro/.

- ***Scheme (Skij)***: Skij is a small Scheme interpreter implemented in Java. Scheme is a variant of Lisp. Skij enables rapid prototyping in the Java environment. It has many advanced features like macros and first-class continuations. You can download a copy of Skij at www.alphaworks.ibm.com/formula/skij. There are at least 15 ports of Scheme to the JVM. If Skij isn't your favorite Java Scheme variant, let us know what is and why.

The above-mentioned languages are just suggestions. If your favorite isn't listed or none of them tickled your fancy, then go to Robert Tolksdorf's comprehensive list of programming languages for the JVM at http://grunge.cs.tu-berlin.de/vmlanguages.html. Over 100 languages are listed! Find one you like and tell us convincingly why it's the best thing since sliced bread. We don't want to miss the next great thing.

Convergence Dynamic to Static, and Static to Dynamic Typing

It's interesting to compare BeanShell and Bistro. The former is essentially a Java variant that has added dynamic typing with optional static typing. The latter is a Smalltalk variant that has added static typing with optional dynamic typing. Java and Smalltalk are on both ends of the typing spectrum; these variants have crossed the chasm to provide a mix of dynamic and static typing (see Figure 1).

BeanShell and Bistro are on the right track. The most popular scripting language of all time, Visual Basic, also has support for both static and dynamic typing. Note that JPython has yet another way to fill in this typing gap, as we'll see when we cover JPython in the first article in this series.

Rosetta Stone Examples for Programming Languages

This article series will give specific examples that demonstrate how scripting languages can be more productive. Some are great for certain problem domains; the series will highlight the languages working in those domains. Other languages are general purpose; the series will show all of the languages working in a general-purpose manner.

The first part in the series will cover scripting languages in general, and will introduce JPython. Every article (including the first) will demonstrate several sample applications as follows:

Embarcadero

www.

embarcadero.

com

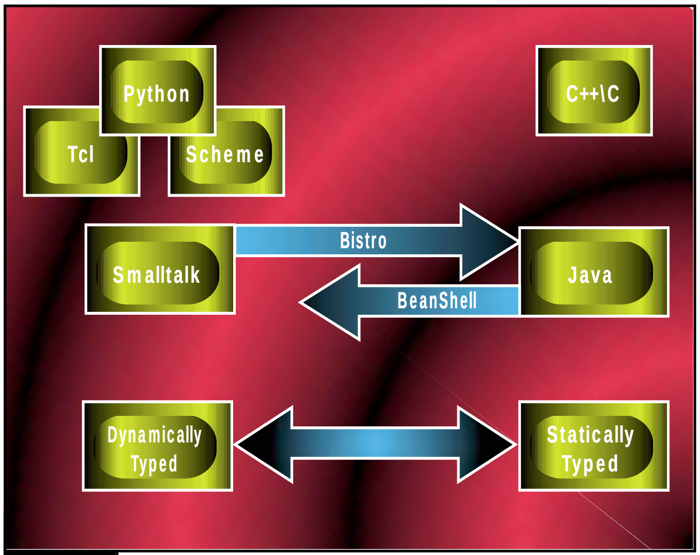


FIGURE 1 Convergence

- A simple GUI application
- A simple statistics application
- Embedding the script into an application (if applicable)
- A threaded animation application
- A simple example parsing text

For comparison, each sample application will have a corresponding Java implementation. Each part in the series will reimplement each sample application in the language covered in that issue. Since people come from many different programming language backgrounds, these applications will be like a Rosetta stone for programming languages that work in the JVM. The series will be very code-centric and hands-on.

Visualize

www.visualizeinc.com

For example, in the first article the sample applications could be implemented in JPython and Java; in the second article, in Rhino (JavaScript) and Java; in the third, in Instant Basic (Visual Basic clone) and Java. And so on. The real order and number of articles will depend on the feedback we get. You, the reader, will decide the order and importance of the language. Also, we're asking you to evaluate the languages in the article to pick the best ones.

Criteria for Judging the Best Languages

1. Ease of use
2. Embedability
3. Resemblance to parent language
4. Unique features
5. String parsing
6. Productivity
7. Working well with Java classes
8. Development environment/debugging

Let's drill down on the above criteria a bit.

- ***Ease of use***: This will cover how easy the language is to learn and use. This could be a combination of market factors. For example, the language is a lot like Visual Basic, and a lot of developers know Visual Basic. Therefore it's easy for many developers to learn. Ease of use could also be based on the syntax of the language. For example, Python is a really easy syntax to learn.

- ***Embedability***: One reason many people use scripting languages is to embed the language into a large application to make it more extensible. Examples of this in the industry are Visual Basic for Applications (VBA) for Microsoft Excel and LotusScript for Lotus Notes. The advantage of using many of these languages is that you can embed them into your own application. In addition, you can use them to control a large system of components and frameworks written in Java. Thus this criterion is how easily you can embed the scripting language in your application and how well the language integrates with Java.

- ***Resemblance to parent language***: This section covers how closely the language resembles the language of origin. For example, how closely does JPython resemble Python? Rhino resemble JavaScript? Bistro resemble Smalltalk? This is an important criterion because it can affect how portable the code is from a legacy system (by legacy I mean non-Java – I *am* a Java bigot!), and document the pitfalls for people familiar with those languages. Also, the resemblance can affect how fast it takes you to get up to speed in the Java variation of the language.

- ***Unique features***: What makes this language cool? What features set this language apart from Java or the other languages? This could be a set of language features that gear the language to a particular problem domain. For example, the language could have a library that's good for generating XML and HTML documents, and the language is easy to integrate with JSP; thus the language is great for Web programming.

- ***String parsing***: Many scripting languages are really keen at doing common tasks like string parsing. Some really excel at it. This covers the string-parsing capabilities of the languages being covered.

- ***Productivity***: If this language is more productive than Java, then this section will highlight that fact. Productivity can be made up of several factors. For example, Smalltalk and Python have an extensive class library that can make it easy to perform common tasks. In addition, Python has built-in language support for collection objects including collection literals that let you define a collection. These language constructs and class libraries make programming strikingly productive.

- ***Working well with Java classes and APIs***: Some languages can compile to Java bytecode, a Java class you can use from other Java classes. Some can extend a Java class into the equivalent of a class for that lan-

"The mere assertion that scripting languages improve productivity five to 10 times merits your interest"

guage. This section will measure how well a developer can integrate this language into existing Java projects.

For example, a language that allows you to define a class that subclasses a Java class may be considered to have very good Java integration.

As another example, Bistro extends Smalltalk syntax so that methods can have type signature. This makes integration with Java easier. Smalltalk ordinarily is only dynamically typed; however, Bistro can be dynamically typed or statically typed. This is a cool feature that will make integrating with Java easier.

- ***Development environment/debugging***: Some languages have facilities to develop and debug code. Some of the development environments are a lot nicer than others. This section will compare the development environments of these scripting languages.

- ***Other items to be covered***: What are the origins of the languages? Why were they developed? What problem domain were they developed for? This often helps in understanding why a language is adept in certain areas.

Rosetta Stone Hello, World

You've probably seen some reference to the famous (read: infamous) "Hello, World" sample program that a lot of programming language books put as their first example program (thanks to the creators of the C language).

Let's continue the tradition in this column, and give you a taste of some of these scripting languages. Instead of covering one language, however, we'll do two from the language series. Also, instead of a simple "Hello, World," we'll have a "say hello" button. When the user selects this button, another window will pop up and display "Hello, World" in an 18-point bold font. To start this off, we'll show this program in Java as in Listing 1.

Since you likely know Java well, I won't explain the listing in detail. Now use it to compare some of the other languages. The code snippets in Listings 2 and 3 will be as close to Listing 1 as possible (as close as the language permits while still showing the advantages of the featured language).

Listing 2 is the JavaScript listing for the "say hello world sample" (Rhino at JavaScript 1.4).

Several JSP implementations support JavaScript, and a lot of people have used JavaScript and JScript, so this listing might be familiar to you. We'll devote a whole article to JavaScript running in the JVM.

Now let's show one of my personal favorites – JPython. JPython integrates nicely with the Java bean properties and event model. Also, JPython is very expressive (it weighs in at about two thirds the size of the shorter of the other two listings – size measured in bytes, not lines, since Java can all be on one line).

JPython has a lot of momentum, and its syntax is easy to learn and real tight (not verbose). A version of JSP, PSP (Python Server Pages), works in a servlet engine. Keep an eye out for this one. (Note the double underbar, i.e., __sayHello denotes that sayHello is a private method.) The first article in the language series will be devoted to JPython, which is to JavaBeans what Visual Basic is to ActiveX. We'll cover it in more detail in the next article.

Parting Shots

Interest is growing in some of these languages. The JDK 1.3 has added more features for hooking scripting languages to events. IDE developers are starting to include scripting language support in their tools. One or two of these languages are likely to be the next Java or XML, that is, insanely popular Internet Phenomena.

Components (JavaBeans) and distributed components (CORBA, EJB, RMI) have a symbiotic relationship with high-level languages. For example, Visual Basic did well because of VBX, OCX and ActiveX components. And COM/ActiveX/DCOM did well because of tools like Visual Basic, PowerBuilder, Delphi, and so on. On the Java platform we have the component models; we need the glue, that is, tools for the high-level languages – such as debuggers and IDEs.

Scripting languages are dynamic, interactive environments that help you develop Java code rapidly. The mere assertion that scripting languages improve productivity five to 10 times merits your interest. Go find a scripting language that runs in the JVM and conquer your next Java project in record time. Then let us know how it went.

AUTHOR BIO

Rick Hightower currently works at Buzzeo Corporation (www.buzzeo.com), the maker of ZEOLogix, an EJB Application Server, Rules Engine and Workflow. He is a principal software engineer working on an EJB container implementation and distributed event management. In addition, he is author of a book, Programming the Java APIs with JPython, *to be published by Addison Wesley Longman.*

rick_m_hightower@hotmail.com

Embarcadero

www.

embarcadero.

com

Listing 1: Java listing for "say hello world"

```
import javax.swing.*;
import java.awt.Font;
import java.awt.event.*;

class MyFrame extends JFrame{
 public MyFrame(){
  JButton sayHello;
  sayHello = new JButton("say hello");
  sayHello.setMnemonic('h');
  this.getContentPane().add(sayHello);
  this.setVisible(true);
  this.pack();

  sayHello.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent ae){
    JButton b = (JButton)ae.getSource();
    b.setEnabled(false);
    sayHello();
   }
  });
 }
 private void sayHello(){
  JFrame helloFrame;
  JLabel helloLabel;
  Font font;
  helloFrame = new JFrame("Hello Frame");
  helloLabel= new JLabel("Hello World");
  font = new Font("Arial", Font.BOLD, 20);
  helloLabel.setFont(font);
  helloFrame.getContentPane().add(helloLabel);
  helloFrame.pack();
  helloFrame.setVisible(true);
 }
 public static void main(String [] args){
  MyFrame frame = new MyFrame();
  frame.setTitle("My Frame");
 }
}
```

Listing 2: JavaScript lisitng for "say hello world"

```
function MyFrame(){
 sayHello = new Packages.javax.swing.JButton("say hello");

 sayHello.setMnemonic('h');
 this.frame = new Packages.javax.swing.JFrame();

 this.frame.getContentPane().add(sayHello);
 this.frame.setVisible(true);
 this.frame.pack();

 sayHello.addActionListener( new
Packages.java.awt.event.ActionListener() {
  __parent__ : this,
  actionPerformed : function(ae) {
ae.getSource().setEnabled(false);
   this.__parent__.sayHello();
  }
 });

 function sayHello(){
  helloFrame = new Packages.javax.swing.JFrame("Hello
Frame");
  helloLabel= new Packages.javax.swing.JLabel("Hello World");
  font = new java.awt.Font("Arial", java.awt.Font.BOLD, 20);
  helloLabel.setFont(font);
  helloFrame.getContentPane().add(helloLabel);
  helloFrame.pack();
  helloFrame.setVisible(true);
 }

 this.sayHello=sayHello;
}

function main(){
 frame = new MyFrame();
 frame.frame.setTitle("My Hello");
}
main();
```

Listing 3: JPython/Python Lisitng for "say hello world"

```
from javax.swing import JFrame, JButton, JLabel
from java.awt import Font

class MyFrame (JFrame):
 def __init__(self):
  sayHello = JButton("say hello", mnemonic=ord('h'))
  self.contentPane.add(sayHello)
  self.visible=1
  self.pack()
  sayHello.actionPerformed = self.__sayHello

 def __sayHello(self, ae):
  ae.source.enabled=0
  helloFrame = JFrame("Hello Frame")
  helloLabel= JLabel("Hello World", font=Font("Arial",
                     Font.BOLD, 20))
  helloFrame.contentPane.add(helloLabel)
  helloFrame.pack()
  helloFrame.visible=1

if __name__=="__main__":
  frame = MyFrame()
  frame.title="My Frame"
```

Download the Code!

The code listing for this article can also be located at

www.JavaDevelopersJournal.com

IMHO, Paolini

—continued from page 7

Over the past four years we have continually refined the process. Early on, for instance, we had very restrictive licensing terms and an informal, undocumented process for collaborating. Over the past four years we have relaxed the licensing terms considerably, most recently making the source to Java 2, Standard Edition runtime, available for free. At the same time we have been working assiduously to formalize the process for how others can get involved in evolving the standard. Dozens of companies throughout the world participate in this documented, audited process. This process, in fact, allows anyone in the industry to participate in expert groups for defining new APIs. Right now there are proposals ranging from real-time OS to an interface for Braille.

I'd find it difficult to come up with any technology in the industry that has undergone such rapid evolution while maintaining a source base that the industry can rely on to do what it's supposed to do.

Okay, you say, but why not just let a standards body take it over? This is their area of expertise.

We've looked at going that route. And quite frankly, we're hard pressed to find an organization that moves at Internet speed and can promise to deliver a model that fosters innovation while preserving compatibility.

I recently received an e-mail from Bruce Scott, CEO of PointBase, who noted that "the reality is that Sun's Java Community Process is more open to the Java community than a public standards process would be."

Other key Internet de facto standards – such as HTTP, TCP/IP and XML – aren't "blessed" by traditional standards bodies like ISO and ECMA. Instead, they're managed by groups similar to the Java Community Process.

So what's next? Right now we're working overtime to refine the Java Community Process still further. We're plowing new ground, as we have been for the past four years. We've tried lots of things that haven't worked out. And when that happens we hear from the industry – immediately.

We're constantly looking for input into what we can do to improve our process as well as our communication to the world for what we are doing with the Java platform. Examples of new technology initiatives can be found at Sun's Java Community Process site at http://java.sun.com/aboutJava/communityprocess.

In the meantime, if you have a moment or two, I'll dig out the home videos.

Elixir

www.elixir.com

JAVA SERVLETS ADVANCED FEATURES

PART 1

Enterprise-level Web application development

WRITTEN BY A.V.B. SUBRAHMANYAM

There's a lot of action going on with Java servlets. The recent public release of Java Servlet Specification v2.2 by Sun Microsystems enhances the functionality of the programming model and the deployment and runtime infrastructure of servlets, which provides for better packaging, security, distribution and management of servlet-based Web applications. The servlet technology is now a part of the Java 2 Enterprise Edition (J2EE) architecture and is expected to play an important role in the Web/enterprise application server market that has hitherto been dominated by proprietary programming models.

When Java servlets were introduced by Sun in early 1997, the primary goal was to provide a Java language alternative to the CGI (Common Gateway Interface) model. Accordingly, the initial model was designed to serve dynamic content for incoming HTTP requests. Those who followed servlets from their inception will remember their early description as applets on the server side. Such descriptions were perhaps appropriate with the initial servlet model. Over the past two years, however, Sun has revised the specification significantly to let servlets cater to developing production-quality Web applications. Most of the development, deployment and runtime features of the servlet model have changed considerably over several revisions. Servlets have now passed their infancy, and are beginning to be used for developing mission-critical Web applications with commercial application servers such as WebSphere, Netscape Application Server, NetDynamics, WebLogic and Orion.

So where do servlets stand today?

Servlet Programming Model — Overview

Java servlets are small, server-side programs that can be composed into dynamic Web applications. Servlets aren't user-invokable applications, but are hosted on servlet containers (more about them later). Servlet containers operate in tandem with Web servers, and invoke servlets based on requests from these servers. When deploying servlets onto servlet containers, you can also specify canonical names to them. The servlet container maintains the mapping between these canonical names and servlet classes.

The servlet programming model is lightweight. Its core classes are HttpServlet, HttpServletRequest, HttpServletResponse, HttpSession, ServletContext and ServletConfig. Of these, the HttpServlet is the class that your application servlets extend from; the rest are interfaces. Container vendors provide implementations for all these classes/interfaces.

KL Group

www.klgroup.com

Servlet programming consists of overriding the init(), service() (or one of its derivatives to process GET, POST, HEAD, DELETE, OPTIONS, PUT and TRACE requests) and destroy() methods of the HttpServlet class. Refer to Table 1 for a brief overview of the programming model. In this table the core classes/interfaces of the servlet API are categorized based on their responsibilities.

At deployment time the site administrator configures the Web server to delegate all servlet requests to the servlet container. When a client user agent (typically a Web browser) sends an HTTP request whose request path maps to a servlet, the Web server delegates the request to the servlet container. In response to this, the servlet container creates HttpServletRequest and HttpServletResponse objects, sets up the environment, creates/locates a servlet instance that corresponds to the path specified in the incoming request, and invokes its service method with the HttpServletRequest and HttpServletResponse objects. The HttpServletRequest and HttpServletResponse objects are Java language objects that encapsulate the request and response streams from the client. The servlet's service method can process the request and write content dynamically to the response stream. It can also forward the request and response objects to a JSP to create dynamic content. The service method can make use of additional utility classes and back-end layers composed of databases, Enterprise JavaBeans, CORBA servers and more to process the request.

This model is similar to the well-known CGI model for building Web sites with dynamic content. In addition to this basic facility, the servlet specification has facilities for session tracking and state management. (For an introductory discussion of these aspects, see references 3 and 4 in the Resources section at the end of this article.) However, the model and its level of abstraction aren't adequate for building enterprise-scale Web applications. These require higher-level abstractions for security, scalability, management and more. In the remaining sections of this article we'll look at some of the advanced facilities that the current servlet specification provides for building and managing large Web applications.

Web Applications, Servlets and Virtual Sandboxes

The J2EE architecture specifies Web applications as the Web-interface part of the J2EE's enterprise computing model. Since the J2EE model is emerging as a standard application server model, let's look at servlets from the point of view of Web applications.

A Web application is a collection of Java servlets, JSP pages, HTML/XML documents and other resources organized in a structured hierarchy of directories. You can also package constituents of Web applications into Web archive (WAR) files. A WAR file is primarily an application deployment unit.

Servlet containers provide the runtime environment for servlet-based Web applications. They also provide a host of other services, some of which we'll discuss here.

At the time of deployment, you can associate a Web application with a specific path of the Web server. This path serves as a document root for serving servlets and other resources that are part of the Web application and therefore define a name tree under this path. For example, if you have a Web application called MyWebApp, you can associate it with the path /MyWebApp of your Web server. All the constituents of the public directory of your Web applications as well as servlets can be accessed relative to this path.

As shown in Figure 1, the directory structure of a Web application consists of two parts. The first part is the public directory structure containing HTML/XML documents, JSP pages, images, applets, and so on. The container appropriately serves the directory's contents against incoming requests. The second part is a special WEB-INF directory that contains the following:

- A /classes subdirectory with all the class files (servlets and other helper classes) of your application
- A /lib subdirectory containing all the JAR files of your application
- A web.xml file that is the deployment descriptor of the Web application

The contents of this directory are for use only by the containers, and the containers won't serve these directly to clients.

What are the implications of the above organization?

First, a Web application (whether archived into a WAR file or not) allows you to package all constituents of the application into one physical unit. Prior to the introduction of the notion of Web applications, you'd usually do the following to deploy an application:

- Copy all your servlets under a directory (which is generally configurable) specified by the container vendor.
- Configure the servlets following the procedure/user interfaces specified by the container vendor.
- Keep all the publicly accessible resources (such as HTML files and images) under the Web server's public directory tree.

When compared to this, the Web application directory structure helps to organize Web applications in a clutterless manner. The publicly accessible resources, as well as your servlet and other classes, can now remain together. Thus you can deploy multiple Web applications independently on the same container. Container vendors provide facilities to deploy or remove Web applications as independent units.

The second important implication is the deployment descriptor. This is an attempt toward standardizing the deployment configuration of Web applications. The deployment descriptor, an XML document with a DTD (Document Type Definition – the current standard for specifying XML documents) specified in the Java servlet specification, allows you to postpone certain decisions from the build time to the deployment time. Examples of such decisions include initialization data for your servlets, canonical names for servlets, MIME type mappings, security (more about this later), database parameters and log file names. It's good practice to defer most of the hard-coding from your servlets to the deployment descriptor.

The third implication is that each Web application is associated with a different context, so you can bundle your servlets in more than one application. Instances of such servlets remain independent in the same Java Virtual Machine of the container and don't share the same context (more about context in the next section). Thus this notion of a Web

MODEL	CONSTITUENTS	RESPONSIBILITIES
Servlet Environment	ServletConfig	To access configuration information (loaded from deployment descriptor)
	ServletContext	To access environment common to all servlet instances in a Web application in a JVM
	HttpSession	To maintain association between Web application and client; also allows conversational state management
Servlet Invocation	HttpServletRequest	To encapsulate incoming HTTP request information
	HttpServletResponse	To encapsulate HTTP response
Application Logic	The init() method of your servlet	Any specific initialization of your servlet instance before it's ready to accept requests
	The service() method of your servlet	To implement parsing HTTP request, setting or getting information from servlet environment, your application logic and response generation by writing to HttpServletResponse object or redirecting to JSP page
	The destroy() method of your servlet	To implement any specific cleanup of your servlet instance before it's finally withdrawn from service

TABLE 1 Constituents of the servlet programming model

Concentric

www.concentrichost.net

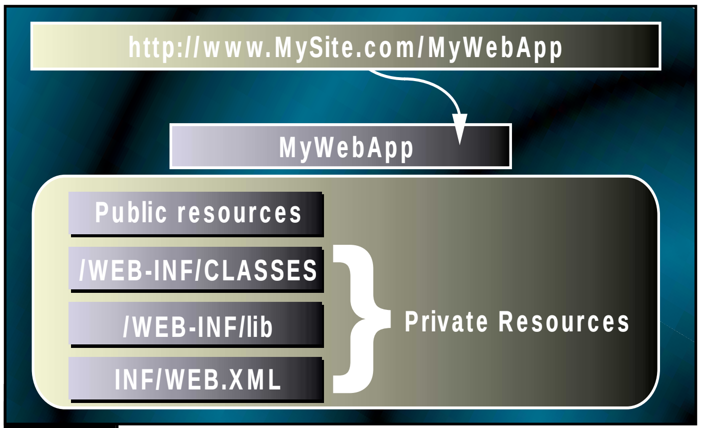


FIGURE 1 A Web application and its association to a Web server path

application introduces a virtual sandbox within a JVM. *Note:* Previously, you could deploy the same servlet more than once with different canonical names and initialization parameters. However, all such servlets share the same context since there was no notion of a Web application.

Finally, different Web applications hosted on the same container can't share clients' session information. That is, if you have two Web applications deployed on the same container, servlets in both applications see two different sessions for the same client. This is a marked difference from the old servlet model in which a container (aka servlet engine) can establish only one client session. You now need to implement more ingenious ways to share data between your Web applications.

Containers – Not Just Hosts

The container, the runtime component of the servlet architecture, provides runtime and network services for hosting servlet-based Web applications. *Note:* While the servlet specification uses the term *servlet container*, the J2EE specification uses a more generic term, *Web container*, to denote that Web containers can manage Web applications containing JSP pages in addition to servlets.

Also, prior to version 2.2 of the specification, *servlet engine* was the term used to denote what is now called a *container*. While these terms represent the same in basic functionality, the change in language indicates a shift of emphasis from processing – that's what an engine does – to providing a runtime for hosting.

Although a container essentially provides runtime and network services for hosting Web applications, it's worthwhile to interpret a container as a request interceptor between network services and the servlet instances. This allows us to discuss what goes on between the container receiving a request and a servlet instance being invoked to handle that request. It also lets us look at certain innovations that containers can implement by virtue of their request-interception capabilities.

First, what goes on within a typical servlet container after it receives a request and before it invokes a servlet instance?

1. ***Request and response marshaling and unmarshaling:*** The container maps the requests and responses (typically over HTTP) into Java language objects (the HttpServletRequest and HttpServletResponse objects) that servlet instances can access. This eliminates certain HTTP protocol-level semantics from servlet development. Examples include extracting GET/POST parameters, cookies and other headers from requests, setting cookies, and so on.
2. ***Request mapping:*** How does the container resolve an incoming request to a servlet class? This is based on canonical names assigned to servlets (in the deployment descriptor) and the root of the Web application. For example, consider a Web application with its root at /MyWebApp. If it consists of a servlet SnoopServlet with name Snoop (specified in the deployment descriptor), the container maps all requests to http://my.site.com/MyWebApp/Snoop to an instance of SnoopServlet. Refer to the Servlet API specification for more details on other mapping rules. The servlet container maintains a naming context for all servlet classes. Within this context the container can identify a servlet class to handle the request.
3. ***Environment setup:*** Containers also create and manage the environment in which servlet instances exist and operate. As discussed earlier, the environment includes the ServletContext, ServletConfig and HttpSession objects.
4. ***Life cycle:*** Containers are responsible for managing the life cycle of servlet instances, which involves locating or creating, initializing and destroying servlet instances. Containers manage the life cycle of HttpSession and ServletContext objects. Based on your servlet's threading model, they can also manage a pool of servlet instances and allocate instances to incoming requests.

From a client's point of view, the container does the above-mentioned tasks while intercepting the request (see Figure 2) and delegating to a servlet instance, although from the container's point of view these are essential tasks in handling a request (see Figure 3).

In addition, containers can perform certain advanced tasks while intercepting a request.

REQUEST SERIALIZATION

Request serialization is one of the techniques used to create client/server applications with a limited number of threads/objects to handle requests. In the case of servlets, servlet containers may choose to implement request serialization for SingleThreadModel servlets. That is, the container may maintain a fixed number of instances of such servlets, and can keep the requests waiting in a queue to be processed when the number of concurrent requests to the servlet is more than the available number of instances. When an instance becomes free, the container can delegate a waiting request to it. In general, request serialization requires the underlying deployment framework (in this case, the container) to be able to represent requests in serializable structures, then deserialize them later for processing. Such a capability is inherent with servlet containers since containers receive HTTP requests and map them to method invocations on service methods of servlet instances.

DECLARATIVE SECURITY

Declaritive security is a recent addition to the servlet API. A traditional approach to implementing authentication and access control is called *programmatic security* and involves the following steps:

1. Program the application to have one more entry point with login forms.
2. Authenticate the user after he or she submits the login form, and save some kind of flag in the associated session to indicate that the user has been authenticated.

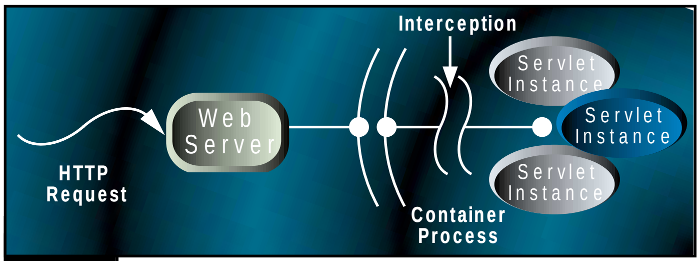


FIGURE 2 Container as an interceptor and provider of services

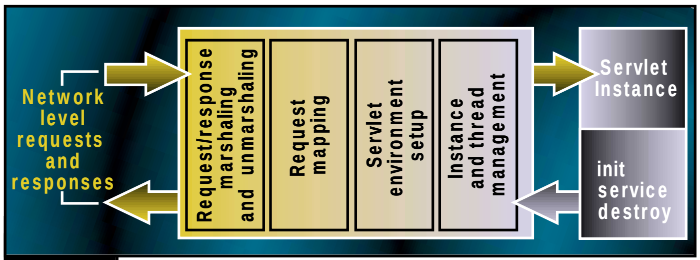

Network level requests and responses
Request/response marshaling and unmarshaling
Request mapping
Servlet environment setup
Instance and thread management
Servlet Instance
init service destroy

FIGURE 3 Basic container services

Riverton

www.riverton.com

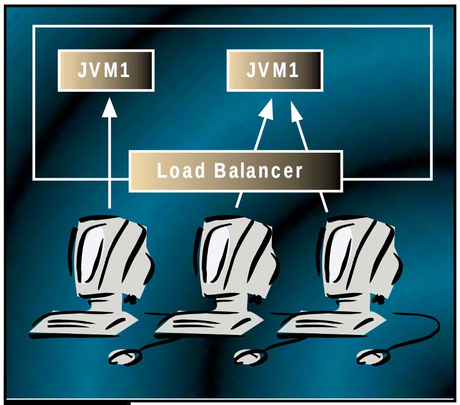


FIGURE 4A Instance-independent load balancing (sticky)

3. To prevent users from directly accessing your servlets, each servlet should check for this flag.

With this approach each servlet would be programmed "to protect itself" from unauthorized access. To protect static resources (HTML pages, images, etc.), you'll be required to use the Web server's authentication mechanisms. That is, static and dynamic constituents of your application require different approaches to implementing authentication.

The servlet specification has now introduced another approach, *declarative security*, in which the container takes the responsibility of protecting Web applications based on the container's interception ability. Here's a scenario:

- Use the container-provided tools to create, for example, users, passwords and groups/roles.
- Add a security-constraint entry in the deployment descriptor of your Web application (see Listing 1 for an example). All GET and POST requests to resources under /MyWebApp require the user to have a role called *manager*. Based on this information, the container can enforce a login before delegating the request to servlets or before serving static resources. In the example below, the container uses form-based authentication and invokes the login.html page for any request under /MyWebApp.
- When the user requests a resource for the first time, say, /MyWebApp/buy, the container automatically serves the /MyWebApp/login.html page, which requires the user to fill in the login name and password. Once the user submits this form, and if the supplied credentials belong to a user with the role "manager," the container authenticates the user, then redirects the user to /MyWebApp/buy. If the credentials don't match, the container serves the specified /MyWebApp/error.html. The authentication information will be valid for the life of the session.

Following are some of the implications of this approach:

- Since it's based on the URL patterns associated with different constituents of Web applications, it applies uniformly to servlets as well as static resources.
- The servlets needn't be programmed to check credentials every time they're invoked.
- The security requirements of the application can be changed at deployment time without having to change/recompile servlets.

In addition to the form-based approach, the servlet specification also supports basic, digest and HTTP client-certificate–based authentication.

SESSION/CONTEXT PASSIVATION

Passivation is the process of swapping contents of session and context data to a persistent storage (for example, serialized to files). Containers can implement passivation to free some of the session and context objects and reuse them for different users/applications. The passivated sessions/contexts can be activated on demand when there's a client request requiring the session/context.

Distribution and Scalability

The new servlet specification addresses the issue of scalability by providing for distributable Web applications and distributable containers. A distributable container consists of a number of JVMs running on one or more host machines. In this setup you can deploy your

Career Central

www.careercentral.com

SD2000

servlets on a number of JVMs. Depending on the load-balancing strategy (vendor-specific), containers route incoming requests to one of these JVMs. In such a setup all requests from a client may or may not be handled by the same JVM. For seamless processing of requests in these cases, the container can employ one of the following strategies:

- A single JVM may handle all requests that are part of a single client session. That is, the container maintains an association between a client/session and a JVM. This JVM affinity preserves the integrity of state and sessions, since within a session the client requests are always served by the same JVM (see Figure 4A).
- Requests that are part of a single session may not be handled within a single JVM (see Figure 4B). In this case the container doesn't guarantee JVM affinity and will be required to provide distributed state- and session-management facilities. This involves distributing the state (request-specific as well as application state-specific) among all the nodes in the cluster after processing each request, making one of the nodes responsible for storing the state, or a combination of both.

The first approach is sticky in nature and doesn't require elaborate distribution of state information across multiple JVMs. The second strategy is more performance intensive and may defeat the purpose of load balancing. Hence, the servlet specification doesn't require the second approach. Instead, it expects the containers to make sure that a single node in a cluster will handle all requests pertaining to a user session. However, we can expect some application server vendors to provide instance-independent load balancing.

For your Web application to avail itself of this facility, you should mark your Web applications as distributable in the deployment descriptor. Once marked, the container can activate the servlets contained in the Web application on multiple JVMs. The container can then employ any load-balancing strategy to distribute the incoming requests to servlets in one of the JVMs.

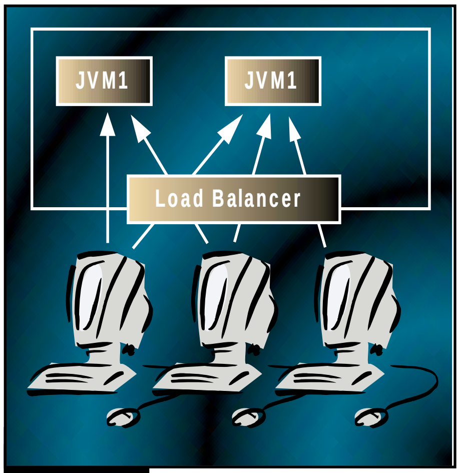


FIGURE 4B Instance-dependent load balancing

"Java servlets are one of the major components of the J2EE architecture"

Although thus far the servlet specification is silent about the failover capability of servlet containers, it's one of the features we can expect from some of the Web application server vendors. It requires that the container swap the HttpSession (and perhaps ServletContext too) objects of each application (in each JVM) to a persistent media whenever there's a change in the state of these objects. This way, active client sessions can be reactivated upon restarting the container (or the host nodes thereof).

In addition, containers may implement session/context swapping between JVMs in the cluster to shift load from one JVM to another or to add/remove hosts in the cluster.

Summary

Java servlets are one of the major components of the J2EE architecture. As most major application server vendors are moving in the direction of the J2EE application programming model, Java servlets have gained added importance. Servlets are now replacing most of the older, proprietary programming models for building Web applications.

To summarize this discussion, the servlet programming model now addresses enterprise-level Web application development. The notion of Web applications and containers has profound implications on the way dynamic Web sites can be built and managed – a major leap from the earlier single-JVM, single-application model.

In Part 2 of this article we'll look at certain precautions that need to be taken while building servlet-based Web applications in the changed scenario.

Resources

1. Java Servlet Specification v2.2: http://java.sun.com/products/servlet/2.2/index.html
2. Java 2 Enterprise Edition: http://java.sun.com/products/j2ee
3. Hunter, J., and Crawford, W. (1998). *Java Servlet Programming*. O'Reilly & Associates.
4. Servlet Essentials: www.novocode.com/doc/servlet-essentials

AUTHOR BIO

Dr. Subrahmanyam is a technical consultant with the electronic commerce division of Wipro Technologies, based in Bangalore, India. You can visit him at his Web site, www.Subrahmanyam.com.

subrahmanyam_avb@technologist.com

Listing 1: Deployment descriptor for form-based authentication

```
<security-constraint>
   <!-- Define a collection of resources -->
   <web-resource-collection>
   <web-resource-name>My Web Application</web-resource-name>
   <url-pattern>/</url-pattern>
   <http-method>POST</http-method>
   <http-method>GET</http-method>
</web-resource-collection>

   <!-- Define authentication requirements -->
<auth-constraint>
   <role-name>manager</role-name>
</auth-constraint>

   <!-- Specify how user-container communication should be
handled -->
<user-data-constraint>
   <transport-guarantee>NONE</transport-guarantee>
</user-data-constraint>
</security-constraint>

<!-- Specify how authentication should be done -->
<login-config>
   <auth-method>FORM</auth-method>
   <form-login-config>
      <form-login-page>/login.html</form-login-page>
      <form-error-page>/error.html</form-error-page>
   </form-login-config>
</login-config>
```

Download the Code!
The code listing for this article can also be located at www.JavaDevelopersJournal.com

ADVERTISING INDEX

ADVERTISER	URL	PH	PG
4TH PASS	WWW.4THPASS.COM	877.484.7277	29
AMERICAN CYBERNETICS	WWW.SOFTEXPORT.COM	800.899.0100	27
APACHE 2000	WWW.APACHECON.COM	650.404.9944	89
APPLIED REASONING	WWW.APPLIEDREASONING.COM	800.260.2772	35
BEA WEBLOGIC	WWW.WEBLOGIC.BEASYS.COM	800.817.4BEA	2
CAREER CENTRAL	WWW.CAREERCENTRAL.COM/JAVA	888.946.3822	28
CAREER OPPORTUNITY ADVERTISERS		800.846.7591	90-101
CERTIFY ON-LINE	WWW.CERTIFYONLINE.COM	877.JAVA YES	66
COMPUTERJOBS.COM	WWW.COMPUTERJOBS.COM		39
CONCENTRIC NETWORK	WWW.CONCENTRICHOST.NET	800.476.0196	77
DEVELOPMENTOR	WWW.DEVELOP.COM	800.699.1932	87
ELIXIR TECHNOLOGY	WWW.ELIXIRTECH.COM/DOWNLOAD/	65 532.4300	63
EMBARCADERO	WWW.EMBARCADERO.COM/ADMINISTER		65
EMBARCADERO	WWW.EMBARCADERO.COM/DESIGN		67
EMBARCADERO	WWW.EMBARCADERO.COM/DEVELOP		69
FIORANO SOFTWARE, INC.	WWW.FIORANO.COM	408.354.3210	45
IAM CONSULTING	WWW.IAMX.COM	212.580.2700	61
IBM	WWW.IBM.COM/DEVELOPERWORKS		13
JAVA DEVELOPER'S JOURNAL	WWW.JAVADEVELOPERSJOURNAL.COM	914.735.0300	79
JAVA DEVELOPER'S JOURNAL	WWW.JAVADEVELOPERSJOURNAL.COM	914.735.0300	79-82
JAVACON 2000	WWW.JAVACON2000.COM	914.735.0300	83
JAVELIN	WWW.JAVELINTECH.COM	612.630.1063	47
JDJ CONSULTING SERVICES	WWW.OPENJOBS@SYS-CON.COM		41
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	88
KL GROUP INC.	WWW.KLGROUP.COM/PERFORMANCE	888.328.9597	15
KL GROUP INC.	WWW.KLGROUP.COM/INTERFACE	888.328.9596	71
KL GROUP INC.	WWW.KLGROUP.COM/DEADLINE	888.328.9596	104
METAMATA, INC.	WWW.METAMATA.COM	510.796.0915	49
NEW ATLANTA	WWW.NEWATLANTA.COM/	678.366.3211	57
NUMEGA	WWW.COMPUWARE.COM/NUMEGA	800.4-NUMEGA	19
OBJECT DESIGN	WWW.OBJECTDESIGN.COM/JAVLIN	800.962.9620	52-53
OBJECTSWITCH CORPORATION	WWW.OBJECTSWITCH.COM/IDC35/	415.925.3460	55
OFFICE.COM		212.995.7742	47
PERSISTENCE	WWW.PERSISTENCE.COM		17
POINTBASE	WWW.POINTBASE.COM/JDJ	877.238.8798	25
PRAMATI	WWW.PRAMATI.COM/J2EE.HTM	914.876.3007	51
PROTOVIEW	WWW.PROTOVIEW.COM	800.231.8588	3
QUICKSTREAM SOFTWARE	WWW.QUICKSTREAM.COM	888.769.9898	20
RESPONSE SYSTEMS SERVICES, INC.		212.295.4305	41
RIVERTON SOFTWARE CORPORATION	WWW.RIVERTON.COM	781.229.0070	85
SEGUE SOFTWARE	WWW.SEGUE.COM/ADS/CORBA	800.287.1329	11
SIC CORPORATION	WWW.ACCESS21.CO.KR	822.227.398801	43
SILVERSTREAM SOFTWARE, INC.	WWW.SILVERSTREAM.COM	888.823.9700	103
SOFTWARE AG	WWW.SOFTWAREAG.COM/BOLERO	925.472.4900	21
SYBASE INC.	WWW.SYBASE.COM	800.8.SYBASE	31
SYMANTEC	WWW.VISUALCAFE.COM		4
TIDESTONE TECHNOLOGIES	WWW.TIDESTONE.COM	800.884.8665	37
TOGETHERSOFT LLC	WWW.TOGETHERSOFT.COM	919.772.9350	6
VISICOMP, INC.	WWW.VISICOMP.COM	831.335.1820	73
VISUALIZE INC.	WWW.VISUALIZEINC.COM	602.861.0999	87
VSI	WWW.VSI.COM/BREEZE	800.556.4VSI	75
YOUCENTRIC	WWW.YOUCENTRIC.COM/NOBRAINER	888.462.6703	33

Geek Cruises

www.geekcruises.com

Develop-mentor

www.develop.com

Get Your Own Subscription to the Finest Technical Journals in the Industry!

1-800-513-7111
www.sys-con.com

JDJ NEWS

IBM and AOL Expand iPlanet Products on AIX

(*Somers, NY / Mountain View, CA*) – IBM and America Online, Inc. (AOL), announce that the Sun-Netscape Alliance will significantly expand the number of iPlanet Internet infrastructure and e-commerce application products that are ported to the IBM AIX UNIX platform.

Under the terms of the agreement, resources for porting, testing and supporting the applications will be supplied by the Sun-Netscape Alliance. IBM will provide on-site technical assistance, development resources, hardware and software to the Alliance to assist in the porting.
www.ibm.com

Unify, Evergreen Internet Form Strategic Partnership

(*New York, NY*) – Unify Corporation and Evergreen Internet announced a strategic partnership at the December eBusiness Conference and Expo in New York.

As a result of the partnership, Unify enhances its Unify eWave Commerce product by being able to deliver e-commerce components such as merchandise service, shopping cart, inventory management, transaction manager, and more, along with 100+ JavaBeans for powerful, dynamic and customized Web site automation. Evergreen Internet will integrate its products with the Unify eWave Engine Java application server. www.ever-green.com www.unify.com

Allaire Acquires Valtó Systems

(*Cambridge, MA*) – Allaire Corporation has acquired Valtó Systems, a pioneer in Enterprise JavaBeans server technology, for 225,000 shares of Allaire common stock. The acquisition further expands Allaire's position in the emerging e-business platform market.

The new Java technologies will complement, support and integrate with Allaire's entire product offering, including the JRun JSP server, the ColdFusion Web application server and the Allaire Spectra packaged system.
www.allaire.com
www.valto.com

Zucotto Offers Integrated Solutions for Wireless Applications

(*San Diego, CA*) – Zucotto Systems Inc. has a fully embedded Java and Jini solution for hand-held devices.
www.zucotto.com

Versant Releases VAR 2.1

(*Fremont, CA*) – Versant Corporation introduces Versant Asynchronous Replication (VAR) 2.1, an extensible, Java-based replication solution for enterprises.

VAR 2.1 offers several new features, including performance improvements for high-bandwidth links and a flexible manual replication option. VAR 2.1 also supports a rules-based policy for batching of replication transactions. Supported platforms include Windows NT and UNIX.
www.versant.com

Tidestone's Formula One Supports Oracle8*i* Lite

(*Overland Park, KS*) – Tidestone Technologies, Inc.'s Java spreadsheet technology, Formula One, supports Oracle8*i* Lite, the Internet platform for mobile computing.

A demo illustrating how Formula One can be used to add data intelligence to Oracle8*i* Lite applications can be accessed on the Web at http://oracle.tidestone.com/demos_f1j/demo_oracle_wtg.jsp .

Harder to Build a Platform for E-Commerce Apps, Says Ovum

(*Boston, MA*) – Continuing uncertainty over Java standards will make it harder for companies to formalize their Web application development strategies, claims Ovum, an independent research and consulting company, in a recent report.

Ovum predicts that it will take at least six months before there are enough standard implementations of the Java platform for developers to make a sensible choice. Until there is real, broad support for the Java platform, development tools and application servers will continue to be closely bound to each other, making it difficult to select tools and runtime services independently.
www.ovum.com

Whirlpool Announces Key Steps in Extension of Global Business Strategy

(*Benton Harbor, MI*) – Whirlpool Corporation has announced agreements to partner with both Cisco Systems and Sun Microsystems in the development of its Networked Home Solutions Initiative. The new initiative will bring the power of broadband Internet technology to a new generation of Whirlpool networked home products and enhanced services.
www.whirlpoolcorp.com

JDJ NEWS

HiT Software Releases E-Generation DB2 Access

(*San Jose, CA*) – HiT Software has introduced HiT ODBC/DB2 v3.1, high-performance direct connectivity middleware for DB2 server access. The new software provides latest generation ODBC level 3 compliance, SSL v3.0 encryption support, higher performance and increased flexibility.

HiT ODBC/DB2 client and server versions and their associated Developer Edition are available immediately. Trial downloads of the runtime middleware can be downloaded at www.hit.com.

Cimmetry Launches Java-Based Version of AutoVue

(*Montreal, Quebec*) – Cimmetry Systems Inc. has announced its upcoming release of JVue, a Java-based version of its flagship viewing and markup product, AutoVue. The new release, code-named JVue, provides the features of AutoVue in a true thin-client, zero administration solution.

JVue is planned for release in the first quarter of this year. www.cimmetry.com/jvue

IBM RS/6000 S80 Sets Java Performance Records

(*Somers, NY*) – IBM has announced that a six-way RS/6000 S80 server running the AIX operating system has set new records for Java performance and scalability, surpassing the previous record holder, a Sun E6500 server containing three times more than the number of processors. The achievement establishes the S80 as the leading computing platform for Java server applications.

A download of IBM AIX Developer Kit, Java 2 Technology Edition, Version 1.2.2 for AIX Version 4.3.3 is available from the IBM Java Developer Kit download site at www.ibm.com/java/jdk/download/index.html.

PurchaseSoft Adds Forward Auctioning

(*Minneapolis, MN*) – PurchaseSoft, Inc., has added forward auctioning capabilities to its Java- and HTML-based products, PurchaseSmart and SourceSmart. www.purchasesoft.com

ShopNow.com Connects with Persistence Software

(*San Mateo, CA*) – ShopNow.com Inc., one of the Internet's largest shopping portals, has chosen Persistence PowerTier as the application server behind its online transaction processing system. The solution will enable ShopNow.com to create a system that will sustain up to 100 transactions per second. www.shopnow.com www.persistence.com

Sun Introduces .com Home of the Future

(*Las Vegas, NV*) – Giving consumers the opportunity to experience the home of the future, Sun Microsystems, Inc., unveiled the .com Home at the January Consumer Electronics Show in Las Vegas.

The .com Home exhibit demonstrates Internet-enabled solutions using Sun's technologies from leading industry partners and manufacturers in the wireless, interactive television and home gateway markets. Sun's .com Home demonstrates how dozens of different consumer appliances and services can interact seamlessly into a smart home network, offering a high level of consumer convenience. www.sun.com.

Vignette Acquires Engine 5

(*Austin, TX*) – Vignette Corporation has acquired Engine 5, Ltd., a pioneer in enterprise-wide Java server technology. The acquisition deepens Vignette's commitment to Java and the J2EE standard. The Engine 5 EJB server technology will be leveraged throughout the Vignette product family and will complement, support and integrate with J2EE-based offerings such as IBM's newly announced WebSphere Commerce Suite, Oracle's iStore and BEA's WebLogic. www.vignette.com.

JDJ Announces JavaCon 2000 Conference

(*Pearl River, NY*) – ***Java Developer's Journal*** is launching JavaCon 2000: Building the New Enterprise. This key Java-focused conference will be presented on September 24–27, 2000, at the Santa Clara Convention Center in California.

"JavaCon 2000 is expected to be the largest Java developer conference of the year after JavaOne. It will provide valuable insight for Java developers charged with building the next generation of Java enterprise applications," according to Fuat Kircaali, founder of SYS-CON Publications and publisher of ***Java Developer's Journal***.

Attendees will be able to select from a variety of sessions, including JavaSpaces and Jini; Building Mission-Critical Applications with Java; Advanced JFC/SWING Component Integration; Real-Time Java; Designing High Performance e-Commerce Systems with EJB; Java, XML and DCOM; Using Java, CORBA, and XML for a Distributed Object Architecture; Java in Embedded Systems; Advanced JavaBeans and more.

For online registration go to www.javacon2000.com.

POET Launches Object Server Suite v6.0

(*San Mateo, CA*) – POET Software has released an ODMG 3.0-compliant version of its object database management system, Object Server Suite (OSS) v6.0.

Built around POET's FastObject technology, the POET Object Server Suite provides developers with a cost-effective object database solution for creating complex packaged Java and C++ applications. www.poet.com.

Voting Time for *JDJ* Readers' Choice Awards

(*Pearl River, NY*) – ***Java Developer's Journal*** Readers' Choice Awards – the "Oscars" of the software industry – are given to the best Java products of the year. Since January 13, ***JDJ*** readers have been logging on to www.sys-con.com/java/readers-choice2000 and voting for their favorite software products in 17 categories. Voting continues through May 31. Those receiving the most votes will receive awards during a special ceremony at the JavaOne Expo in June.

Last year more than 15,000 ***JDJ*** readers cast their votes for the best products of the year. Winners and finalists in 14 award categories were acknowledged for their contributions involving development of Java-based solutions that respond to and meet the increasing demands of the industry.

Track Data Launches a Java Version of MyTrack for Macs

(*New York, NY*) – Track Data Corporation released a Java-based beta version of its successful myTrack online trading and market data system that's compatible with the Mac operating system. MyTrack delivers free-streaming delayed quotes and unlimited free real-time extended quotes, as well as breaking company news, a trade-by-trade log, charting for technical analysis and a proprietary library of intraday market statistics. The company also offers online trading through its myTrack Internet-based personal investment service. The new version is available to download at www.mytrack.com.

JDJ S
Spr

www.jdjs

Store
ead

store.com

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Employment
Ad

Java Record Circulation

www.javadevelopersjournal.com

Java Record Circulation

www.javadevelopersjournal.com

Internet Standards: Opening the Door for the Next Generation of E-Business

WRITTEN BY ROD SMITH

Why Open Standards?

When you consider the dynamic connections and just-in-time integration resulting from today's networked business relationships, it's easy to appreciate how keeping standards proprietary is clearly the strategy of the last computing generation. As e-business enters the 21st century, open standards will fuel the Internet, further speeding the pace of the new economy. The importance of open standards extends across all industries because they enable markets to grow and evolve faster. By creating a framework in which all players can participate, open standards are integral to the success of the Internet as a business channel.

The Internet's brief history has already shown that the free exchange of information is good business for everyone. It's no coincidence that underlying technologies like TCP/IP, the communication protocol for the Internet, and HyperText Transfer Protocol (HTTP), the rules for exchanging files on the Web, have enabled the economy to reach new highs recently. As industry standards, their value lies in providing an established and open foundation for effective use of the Internet for e-business and e-commerce.

If software may be thought of as providing the "electrical functionality" of e-business, let's make some real-life analogies in support of open standards. Picture being unable to purchase a new appliance without first checking to see whether it adhered to your electric company's unique specifications for generating power. Imagine if you couldn't call a friend in a neighboring state because your local phone companies couldn't agree on a standard to allow their telephone networks to communicate. According to International Data Corporation, the number of IP telephony minutes will reach 135 billion by 2004, and revenue for this service will rise from $480 million in 1999 to $19 billion by 2004. This exponential growth wouldn't be possible without standards for telephone communication.

We believe that XML, an open standard from the World Wide Web Consortium (W3C), is a key part of e-business' feverish growth spurt. Companies that have built business models based on open standards such as XML are already seeing their substantial marketplace success enhanced by the ubiquity of these technologies.

How Do Internet Predictions Support the Need for Open Standards?

More people than ever are using and sharing information online – IDC forecasts that the U.S. Internet user population will reach 137 million, and the worldwide Internet population will exceed 274 million this year alone. The recent spate of Web-enabled mobile phones and wireless personal digital assistants (PDAs) is providing a new benchmark of Internet accessibility, particularly among today's increasingly mobile workforce.

Consider the burgeoning business-to-business market, valued at $1.3 trillion by Forrester Research by the year 2003. In the years ahead, as we've seen with online retailers, markets will become more discriminating in the B2B segment. Leading companies will have a strong Internet strategy integrated with other non-Internet channels. These companies will alter the concept of an Internet endeavor as a standalone entity, disconnected from the core business. Leaders have already recognized this and are well on their way to tighter integration of the two.

Therefore, even more so in the 21st century and beyond, the entire success of a business in the online realm will hinge on its ability to exchange information seamlessly, regardless of where it's generated or headed. E-business technologies such as Java and XML can enable these transactions to flow smoothly throughout the enterprise only when they are based on true open standards.

When you consider how the majority of Fortune 500 companies use multiple, disparate operating systems, the implications for internal data integration alone – let alone conducting e-business with outside partners – are staggering without the help of a common, easily accessible base code. Savvy customers recognize and demand the investment protection afforded by open standards to ensure an application's extensibility into current and future infrastructures.

While portions of these technologies have matured sufficiently, we remain vigilant to ensure that vendor-neutral standards organizations serve as the gatekeepers to their further development. With the understanding that interoperability of various processes is secured, small and large companies alike can successfully conduct e-business and compete more effectively in the marketplace.

As further proof, consider that IDC predicted a need for 750,000 Java programmers in 2000. Clearly, nurturing creativity among the development community is important for everyone. Developers need to be able to write to a clear, pervasively distributed platform of APIs and functions. This demand underscores the importance of ensuring that open standards truly remain open for the benefit of all.

How Do We Get There?

IBM is deeply committed to industry standards that best support the multiplatform, multivendor and dynamic environment that the Internet has enabled. Simply put, we believe that e-business is all about "cooperating on standards and competing on implementation," and we act on these beliefs. This can only occur in the context of true, cross-industry commitment to these standards.

To bring ubiquity to computing, the industry requires a foundation of open standards managed by vendor-neutral standards organizations. As organizations like the W3C, ISO, OASIS and ECMA have proved: "If you define it, they will use it." The reliance of these consortiums on cross-industry involvement speeds the time-to-market of technology by drawing on the perspective, expertise and resources that only a group of leading companies can provide. Standards processes managed by a lone vendor will not work to bring the ubiquity we as an industry require. Only vendor-neutral standards bodies can provide the stability developers need to deliver the next generation of e-business solutions.

There is an important issue at stake here: the freedom to develop in an environment that will remain open, available, and consistent. IBM will remain vocal on issues relating to standardization of key technologies for the evolution of e-business. It's simply the right thing to do.

AUTHOR BIO

Rod Smith is vice president of Java Software for IBM Corporation. Rod has served in many capacities at IBM, most recently as chief technologist and vice president of Internet technology for the Network Computing Software Division. Rod, a member of the Institute of Electrical and Electronics Engineers, Inc., holds a BA and an MS in economics from Western Michigan.

Silverstream

www.silverstream.com

KL Group

www.klgroup.com