

JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

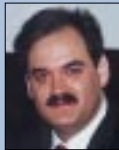
Volume: 4 Issue: 6, June 1999

JAVADEVELOPERSJOURNAL.COM

From the Editor
The XML Mambo
 by Sean Rhody pg.5



A Sign of the Times
 by George Paolini
 pg.7



What Will Come
 by Richard Soley
 pg.64



Anonymous Deployment vs Portability
 by Bruce Scott
 pg.32

E-Java

Java & XML & E-Commerce
 by Ajit Sagar pg.26

Straight Talking Happy Birthday!

by Alan Williamson pg.24

Widget Factory

Text Controls by Swing
 by Jim Crafton pg.36

Product Review

Sybase PowerJ 3.0
 by Sean Rhody pg.56



Clustering Enterprise JavaBeans

with **BEA WebLogic Server**
 pg.66



JDJ Special Feature: CORBA Object Browser

Versatile DII lets you discover and use a CORBA object at runtime without compiling stubs

P. G. Sarang
 Mohan Rajapopalan
8



JDJ Feature: JAVA & COLDFUSION

Two competing, yet complementary e-business technologies

Ajit Sagar
18

Multipatform Application Design with Java

An architecture for multipatform development

Jim Wright
46



A New Java Startup: PointBase, Inc.

Oracle cofounder Bruce Scott's new Java company

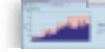
Scott Davison
50



Case Study: Corporation Benefits from JProbe During App Development

KL Group tools eliminate memory leak and improve performance

Ethan Henry
 Josephine Coombe
60



BEA WebLogic Server & EJB

Java and EJB no longer pose limitations

Dean Jacobs
66



Case Study: TRIP.com's Online Solution

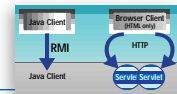
WebLogic-based online travel tool saves travelers time and money

Scott Davison
72

EJB Home: Enterprise JavaBeans

Does EJB really fit into enterprise computing?

Jason Westra
74



A Windows-Specific Java Utility Class

A little help from JNI help to answer many questions

Pat Paternostro
82

**The World's Leading Java Resource
 100,000 copies in print**

BEA WebLogic

You will receive
BEA WebLogic's
Collector's CD with
this issue in the mail!

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

ProtoView

See JDJ Special Offer at:

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Sun Microsystems

To get the hottest equipment,
you have to pay the price.
Lucky for you, it's a small one.
Ultra 5 workstations,
now only \$2,495

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

EDITORIAL ADVISORY BOARD

Ted Coombs, Bill Dunlap, David Gee, Michel Gerin,
Arthur van Hoff, Brian Maso, John Olson,
George Paolini, Kim Polese, Sean Rhody, Rick Ross,
Ajit Sagar, Richard Soley, Alan Williamson

Editor-in-Chief: Sean Rhody
Art Director: Jim Morgan
Executive Editor: M'lou Pinkham
Managing Editor: Brian Christensen
Assistant Editor: Nancy Valentine
Proofreader: Anne-Marie Babcock
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

Josephine Coombe, Jim Crafton, Scott Davison,
Ethan Henry, Pat Paternostrro, Mohan Rajagopalan,
Sean Rhody, Rick Ross, Ajit Sagar,
Poornachandra Sarang, Bruce Scott, Richard Soley,
Jason Westra, Alan Williamson, Jim Wright

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 Manager: Robyn Forma
Advertising Assistant: Megan Ring
Graphic Designers: Robin Groves
Alex Botero

SYS-CON Radio Editor: Chad Sittler
Webmaster: Robert Diamond
Customer Service: Sian O'Gorman
Paula Horowitz
Ann Marie Millillo
Mitchell Low

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.
Application to mail at Periodicals Postage rates is pending at
Pearl River, NY 10965 and additional mailing offices.

POSTMASTER: Send address changes to:
JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc.,
39 E. Central Ave., Pearl River, NY 10965-2306.

© 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.

**FROM THE EDITOR**

Sean Rhody, Editor-in-Chief



The XML Mambo

Every so often I read something that makes me scratch my head and wonder. Most recently this phenomenon occurred when I read an editorial concerning Java and XML in a Web development magazine. The author wrote that he thought the concept of XML was easy to understand in terms of its usefulness, while he was puzzled over the reason for Enterprise JavaBeans. I sat back and said to myself, Is it me, or do people who get into programming by designing Web sites always get it backwards? Not to malign the author, who did a good job of explaining both technologies, but I've been working with EJB for a year, and it seems like a pretty clear concept to me. What I can't seem to get is the importance of XML.

As I understand it, XML is a language used to provide meta data as well as data - information about the structure and content of data in a format that any XML-aware application can interpret meaningfully, in addition to the data itself. For example, someone might write a definition of a recipe (the meta data, known as Data Type Definition or DTD), and then other cooks could publish their recipes using that definition in such a way that all pages or programs written to work with one person's recipes could work with another person's recipes.

I get this. I can see where this can make the building of Web applications easier, although how much easier I'm not sure. With XML, we'll probably be able to add data type checking to pages, and do some other operations that we couldn't do on the pure content of HTML.

What I don't see is how much this buys the industry. First of all, in order for content to be useful to more than just a single company, a standard DTD needs to be created. There are bodies out there in just about every major industry defining these data types. But what if your needs require something different from the standard? How do you reconcile your DTD with theirs? It can be done, but why?

Second, I don't see how much use this will be in programming; about all I can see is field binding and some additional database features. Nice, but why is XML the next killer app? It's just a language like HTML, which was pretty useless in and of itself until the graphic browsers were invented. The browser, the interpreter or user of HTML, was really the killer app.

Probably the big use for XML will be in e-commerce. This is a likely spot, because e-commerce is all about the exchange of structured information, e.g., invoices, transactions, registrations and things like that. And it doesn't seem like a bad way to do things. By standardizing the language for describing the data, it makes it easier for disparate applications, or even disparate companies, to communicate.

I see XML as an enabler for future technology in the same way that the mouse was an enabler. The mouse became a standard input device for just about every computer. XML stands positioned to do the same for data on every computer. But the real killer apps are the programs that will use XML. I don't see XML as changing the way I program, or the way you type in data on the screen.

Now as for EJB, I can see the purpose for that. But then, I got into Web development from the programming side. So maybe I've got it backwards. Let me know. ☛

About the Author

Sean Rhody is the editor-in-chief of *Java Developer's Journal*. He is also a senior consultant with Computer Sciences Corporation, where he specializes in application architecture - particularly distributed systems. He can be reached by e-mail at sean@sys-con.com.

Computer Associates

www.cai.com/ads/jasmine/dev

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

George Paolini, Director of Corporate Marketing, JavaSoft



A Sign of the Times

If you had to conjure an image that best serves as a "sign of the times," what might it be?

Perhaps a screen shot of a rare Partridge Family album being auctioned off for an incredible sum on e-Bay. Or how about a staged photo op of some of those starched-white-shirt telco and cable guys shaking hands in the latest billion-dollar megamerger. This might be more likely: hip-looking X-gens tipping their plastic champagne glasses to celebrate as their IPO turns them into instant multimillionaires. All good images, to be sure.

But a sign in a little toyshop in San Francisco says it all for me. "See us at our new address: www.toydomain.com." That toystore is just one of millions of businesses that has figured it out. And with your help, they've made the Web the definitive platform for business infrastructure today.

It's not that every business is going to do something as radical as abandon its bricks and mortar environment and move to cyberspace. But they're going to build their businesses with the Web as their architecture. And they're going to make it go with the Java platform.

The reasons are very clear. In a global, hyper-competitive market where agility and the ability to react is key, the last thing any business executive wants is to remain locked into an IT architecture. Businesses are demanding the Web and the Java platform as the way to extend their current infrastructure, to unlock all that business logic that has been hidden from public view for years and years by arcane protocols and formats. And businesses are demanding the Web and the Java platform as assurance that they won't be locked in again.

This, of course, is all good news to you. As part of the Java community, you've been lauding the advantages of writing in the Java language for years now. It lowers development costs; it lowers deployment and maintenance costs on heterogeneous networks. But even more than that, you, as part of the Java community, have sought a technical platform based on the free exchange of ideas and fair competition.

It's hard for me to envision another technology that delivers such a level playing field. Open interfaces result in choice. Choice spurs competition. Competition is the mother of innovation.

Indeed, we are on the cusp of a new era - where Java technology innovation is delivering real business value and impacting the bottom line.

The Java platform was invented by Sun Microsystems, but it truly is a product of open industry collaboration and cooperation. There is major intellectual property invested in the Java platform not only by Sun, but also by countless other companies and independent developers. The more than 50 application programming interfaces that define the Java platform were built with the help of this community. JavaBeans, Java Media Framework, Java Database Connectivity, Enterprise JavaBeans - the list of technologies built in cooperation with the developer community goes on and on.

The collaborative way in which the Java platform has evolved has afforded us the honor of being able to use the World Wide Web to harness the intellect and creativity of the smartest people in the world - people like you. That is how the Java platform has become a technology of the highest quality in Internet time.

The growth of the Java industry and your success with the Java platform has been staggering. You now number more than 1 million, and estimates say that your numbers will triple by the year 2002. Collectively you have shipped thousands of Java applications. Your work is innovative and groundbreaking. And we at Sun Microsystems are privileged - and, quite frankly, proud - to have played a role in this success.

Our mantra has been to listen to you and deliver your requirements. You required a platform that was complete, stable, secure and fast. We delivered Java 2 - the first release of the Java platform that delivers functional completeness, uncompromising stability, bulletproof security and speed - beating the pants off any platform previously released or available in the marketplace.

You required easier access to source code and fewer restrictions for using the Java platform...and we announced a new licensing model - the Java Community Source Licensing program. This program facilitates increased and more rapid innovation and faster commercialization of products based on source technology. Licensing the Java platform is as easy as point-and-click. We have been pleased to welcome tens of thousands of new licensees into the community of Java partners through this program.

We also announced the Java Community Process initiative, which opens up the community of interested parties working to extend and develop the Java platform to a larger circle. This initiative formalizes the collaborative, industry-participative methodology for developing the Java platform. And it engages an independent auditing firm to ensure that the collaborative process for new Java platform developments is followed faithfully.

And we heard you loud and clear when you said performance, performance, performance. This spring, the Java HotSpot performance engine was released. This engine delivers unprecedented performance, breaking new ground in software design and raising the bar by providing 100% faster performance than the previous version of the Java platform.

Like you, we've done a lot of work on the Java platform. But our work isn't done yet.

We're preparing for JavaOne, Sun's worldwide Java Developer conference - the largest developer conference on the planet. There we'll unveil Java 2 Enterprise Edition, which enables component-based distributed applications to take full advantage of the power of the Java platform.

These Java platform and Java technology achievements - and those milestones we have yet to achieve - all work toward a single objective: to equip you with the tools you need to capitalize on opportunities in the new networked era. The Java platform has changed the rules of the game. But your innovation is what makes the Java platform a winner. The success of the platform ultimately rests in your hands. ☘

About the Author

George Paolini is director of corporate marketing at JavaSoft, a Sun Microsystems, Inc., business unit. A member of the Java Developer's Journal Editorial Board, he is responsible for managing all public relations, advertising, field marketing communications and developer support activities within JavaSoft. George can be reached at george.paolini@SUN.com.

One
Realm

One Realm is
offering JDJ
readers a FREE
trial version of
"I18n Expeditor"
to internationalize
your software

See JDJ
Special Offer at:
[http://www.
sys-con.com/
java/
specialsofthe-
week.html](http://www.sys-con.com/java/specialsoftheweek.html)

CORBA Object Browser

Versatile DII lets you discover and use a CORBA object at runtime without compiling stubs

by Poornachandra G. Sarang & Mohan Rajagopalan

With the rapid growth of the Internet, distributed Web-enabled applications are becoming popular. One of the most commonly used architectures for development of such applications is CORBA, which provides a platform, location and an implementation-language-neutral architecture for the development of distributed applications. In addition, the phenomenal interest in component technology has led to the development of CORBA Beans. Such CORBA components and objects will soon be available for use on the Web. When you encounter such a CORBA object, you may wish to use its services by dynamically discovering its properties (something similar to Introspect and Reflection in Java). CORBA supports this.

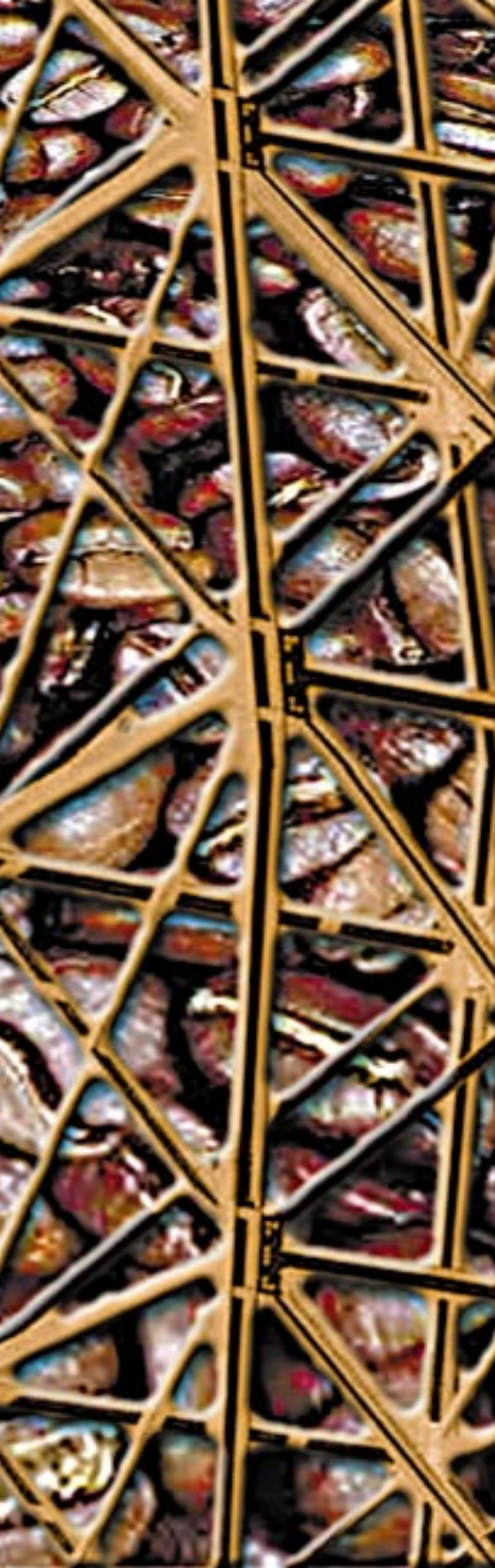
What Is Dynamic Invocation?

CORBA allows you to discover an object's properties dynamically and use its services by invoking the public methods on the object at runtime. This is known as *Dynamic Invocation Interface* (DII). It's essential for a CORBA object that allows the client to discover its properties at runtime to publish its interface definition language (IDL) for the client to look up. CORBA architecture provides for an Interface Repository (IR) in which the publisher of an object publishes its IDL. Once the client obtains a reference to an object, it can retrieve the IDL from this IR. The IDL helps the client understand the public attributes and methods of the discovered object. The client then invokes the methods on the object using the DII mechanism.

The use of IR and DII not only benefits the client in discovering and using the object, but also helps its publisher add new features to the object anytime after its deployment. The client can then dynamically discover these new features and use them. The new interface definition is published in the IR by the developer. The client now has access to all the latest features provided by the application using the DII mechanism. DII is the "dream come true" for developers since it allows them to modify previously published CORBA objects without the need to recompile and redistribute client-side stubs.

A practical example: you're developing a CORBA application for set-

ting up an online store where customers visit, make some purchases and pay for purchases with a check "snail-mailed" to your account. You'll supply a client-side application that interfaces with your server application. The client application contains the compiled stubs required to use your server application. At a later time you may wish to provide the facility of payment by credit card. Adding a new purchase method in the IDL definition does this job. However, you'll need to supply the new stubs to all your clients. If you'd used the DII mechanism in the client application,



requests that the user enter a reference to an existing CORBA object. Usually a CORBA object registers itself with the CORBA Naming Service using a unique name. We pass this name to the object browser, which looks up the Naming Service to resolve the reference to the object. Once the browser obtains a reference, it displays the IDL to the user. The user can now select an operation from the displayed list and invoke it. (A *method* is called an *operation* in CORBA terminology and we use the terms interchangeably in this article.) A method invocation may require one or more parameters and may return a result. The browser determines the types for the required parameters, constructs the parameters of appropriate data types, obtains values for each parameter from the user and, on successful completion of the method, returns the appropriate values to the user. In CORBA each parameter may be of IN, OUT or INOUT type. Since Java supports only pass by value, it supports only IN type parameters, implying that the parameters to a Java method are read-only. In CORBA the method may return the value through OUT or INOUT types of parameters. Our browser takes care of all three types of parameters.

To help understand the complex DII interface of CORBA, we'll discuss the design of the CORBA object browser. But first we'll touch on some aspects of locating the object on the Web using the Naming and URL Naming Services of Visigenic, the IOR (Interoperable Object Reference) and Interface Repository.

Locating Object: CORBA Naming Service and Web Naming Service

Our object browser needs to locate the object before trying to use its services. A CORBA object publishes a reference to itself by registering its name with the Naming Service. This is done with the help of the `bind` or `rebind` method, in which the object reference is tied to its symbolic name. Before you use this method you must start the Naming Service. On Visigenic ORB (we use this for our development) you start the Naming Service using the following command line, with the root name specified as ABCOM:

```
vbj -DORBservices=CosNaming -  
DSVCnameroot=ABCOM -DJDKrenameBug  
com.visigenic.vbroker.services.CosNaming.  
ExtFactory ABCOM namingLog
```

If you use the Java 2 platform, you start the Naming Service using the following command line:

```
tnameserv [-ORBInitialPort <Port_Number> ]
```

The default `Port_number` is 900.

The principal task of the Naming Service is to keep track of the namespace, which is the collection of object names bound to a Naming Service. The namespace may contain a hierarchy of bindings extending over several domains just like a file system on your disk. If

the namespace extends over several domains, it's called the Federated NameSpace. The binding is the logical association of the object reference to its symbolic name.

A client obtains a reference to the Naming Service by calling the `resolve_initial_references()` method of the ORB. Once the initial context of the Naming Service is resolved, it calls the `resolve` method on the `NamingContext` to obtain a reference to the desired object. The whole task of name resolution is transparent to the client.

Another way to publish your object is to convert the object reference to a string and store it in a file using ORB's `object_to_string` method. The method returns a stringified version of the IOR, which contains information such as IOP version, Host, Port, etc., which helps to uniquely identify the object on the Web. The client reads the IOR from the specified file (usually with the extension `.ior`) at the given URL, and obtains a reference to the object using the information in the IOR. Visigenic requires that the extension given to the IOR file must be `.ior` if you wish to use Visigenic GateKeeper.

Visigenic provides the URL Naming Service to locate an object using a URL rather than its generic name. This provides for interoperability among objects bound to ORBs on different machines. The following code segment shows you how to use Visigenic's URL Naming Service to locate an object running on a different ORB:

```
// create the resolver object  
com.visigenic.vbroker.URLNaming.Resolver  
URLresolver =  
com.visigenic.vbroker.URLNaming.Resolver-  
Helper.narrow(orb.resolve_initial_references  
("URLNamingResolver"));
```

```
// locate object using resolver  
obj = URLresolver.locate(http_location);
```

The URL Naming Service uses a `Resolver` Object to locate the `.ior` file and resolve the reference to the remote object. In the above code snippet we create a `Resolver` Object using the `ResolverHelper` provided with the URL Naming Service. The `Resolver` uses the `locate` method to search the specified IOR at a given URL. If the IOR file is found, it's read and the reference to the object running on the remote server is returned to the caller.

The publisher must publish the IOR file on a Web server so the client can locate it using the URL Naming Service. If you're using Microsoft's IIS (Internet Information Server), you'll put this file in the `InetPub/www.root` folder or the folder defined for public access.

Interface Repository

The Interface Repository is used for storing IDL definitions for various CORBA objects. The IDL definition of a CORBA object forms its skeleton, and the object implementation is developed from this definition. The Interface Repository is the online database that maintains a record of interfaces of all registered

the client would be able to discover the newly added method without the necessity of obtaining the new stub.

Object Browser

To emphasize the practical usability of the DII, we've developed a CORBA object browser using Java and JFC classes. The browser



Figure 1: User interface of object browser

objects. On a Visigenic ORB you start the Interface Repository using the following command line:

```
Prompt> irep I Rname fileName
```

The IRname is the name assigned to the Interface Repository, and fileName is the physical file used for storage. Once the IR is created, you can add your object definitions using the idl2ir utility:

```
Prompt> idl2ir file.idl
```

The IDL definition contained in file.idl is now added to the IR. Each entry in the IR contains a header describing the file name, time of creation, user name and location of the file on the machine. The IR stores information in simple text format. You can use a standard text editor to view this. A typical IR entry is shown below.

```
/*
File : abcom.ir
Date : Sat Jan 30 06:58:40 GMT+00:00 1999
User : Administrator
Dir : C:\OBJECTBROWSER
*/
module bank {
interface TermDeposit;
interface TermDeposit {
attribute float Interest;
float Compute(
in float Principal,
in short Period
);
};
};
```

A client obtains the IDL definition from this IR by using the `_get_interface()` method of the CORBA Object class. This returns the `InterfaceDef` object. The full description of the interface definition can now be obtained by using the `describe_interface()` method on this `InterfaceDef` object.

The following code snippet illustrates this process:

```
org.omg.CORBA.InterfaceDef objIntfce =
obj._get_i_interface();
org.omg.CORBA.FullInterfaceDescription
fullObjctInterface =
objIntfce.describe_interface();
```

You may now use the attributes member of the `FullInterfaceDescription` class to obtain information on the various attributes published by the object. Similarly, you'll use the operations member of the `FullInterfaceDescription` class to obtain information on the various operations permitted on the object.

To recapitulate, so far you've learned how to:

- Publish an object by either binding an object to a Naming Service or publishing its IOR on a Web server.
- Locate an object using either a Naming Service or Visigenic's URL Naming Service.
- Publish the Interface Definition of the object in the repository (IR).

We've also seen how a client program retrieves this Interface Definition from the IR and discovers all the attributes and operations published by the object. Most important, you've discovered the object and would now like to use its services by invoking one or more of its published methods.

Dynamic Invocation Interface

To invoke a method, you'll first need to construct a list of arguments (parameters) required by the method. CORBA provides an `NVList` object for constructing such a list. You use the `create_list()` method of ORB to construct it.

```
NVList parameterList = orb.create_list(0);
```

The `NVList` (Named Value List) contains the names and values for the various parameters

required by the method. Initially the list is created with zero elements. You then add elements by using its `add_value()` method:

```
parameterList.add_value(name, currentParameter, mode);
```

where name is the string representing the name of the parameter being added, currentParameter is the object that represents the parameter and mode indicates whether the parameter is of IN, OUT or INOUT type.

You'll need to add the required number of elements to the `NVList` depending on the number of parameters used by the desired method. Once an `NVList` is constructed, you create a request object and pass the parameters to it using the constructed `NVList` object. To create a request object, you use the `_create_request()` method of the CORBA Object class.

```
org.omg.CORBA.Object obj = new
org.omg.CORBA.Object();
org.omg.CORBA.Request request =
obj._create_request(
null,
nameMethod,
parameterList,
resultField);
```

where obj is a CORBA Object, the first parameter specifies the CORBA Context, which is set to null; nameMethod is the name of the method being invoked; parameterList is the `NVList` described above; and resultField is a `NamedValue Object` through which the result will be returned to the caller.

Once the request object is constructed, you can invoke the method on the server using the `invoke()` method:

```
request.invoke();
```

This invokes the method on the server object and returns the result, if any, in the resultField `NamedValue` object. If the method uses OUT and INOUT types of parameters, you may examine their contents for the return values.

User Interface

Having seen the various requirements for DII, we'll now discuss the user interface of our object browser, which is used for dynamically invoking the methods on CORBA servers for which the stubs aren't available at runtime. The user interface of the browser is shown in Figure 1.

The user enters the name of the server object in the input panel of the browser and clicks on the introspect button. The name specified must be the name the object is registered under with the Naming Service. Alternatively, the user can specify the name of the IOR file along with the URL at which the file is located. A typical URL string may be specified as follows:

```
http://www.abcom.com/abcom.ior
```

EnterpriseSoft

EnterpriseSoft is offering
"Report Writer 2.0"

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Once an object is located, the browser displays its IDL in the tabbed pane as seen in Figure 1. You can now click on the “Operation and Attribute Listing” tab to see the various operations and attributes published by the server. If you want to examine an attribute, click on the desired one and its various properties will be displayed in a pop-up window. If you decide to invoke a method, click on the method name, which opens a window showing the various parameters required by the method (see Figure 1). If the method doesn’t require parameters, it’s directly executed. If it does, the type and mode are displayed for each one. Next to each parameter an edit box is provided in which the user can enter the desired parameter value. The browser, however, doesn’t provide any validation on the parameter values. Once all the required parameter values are entered, the user clicks on the invoke button to invoke the server method. The program now builds the request object and invokes the method on the server. If the method completes successfully, the results will be displayed in a pop-up window (see Figure 1). The interface is fairly easy to use. We’ll now look into the design of the browser.

Browser Design

We used JFC for developing the user interface of the browser. The browser consists of seven public classes:

- The ObjectBrowser class, which provides the main user interface for the application
- The InputPanel, OutputPanel and StatusBar classes, which provide various other components of the user interface
- The BackEnd class, which is responsible for all CORBA-related activities
- The two helper classes, AttributeDescription and OperationDescriptionTable, which provide the description of attributes and methods, respectively

The various classes and their relationships to each other are shown in Figure 2.

The ObjectBrowser class (see Listing 1) is the main class of the application and is derived from the JFrame class. It creates instances of InputPanel, OutputPanel, StatusBar and BackEnd classes. The main method creates an instance of ObjectBrowser class and displays the frame to the user. The init() method of the class adds the three user-interface elements – InputPanel, OutputPanel and StatusBar – to the main display window. The InputPanel is used for accepting the object reference from the user. The OutputPanel shows the IDL listing and the operations/attribute names to the user. The StatusBar provides a status display to the user. The ObjectBrowser class provides a utility method called resolveAndIntrospect(), which is called by the InputPanel class. The method receives the object reference that’s to be resolved. The method calls the resolve() method of BackEnd class, and if the object is successfully resolved it calls the introspect() method of BackEnd class to introspect the object and to display

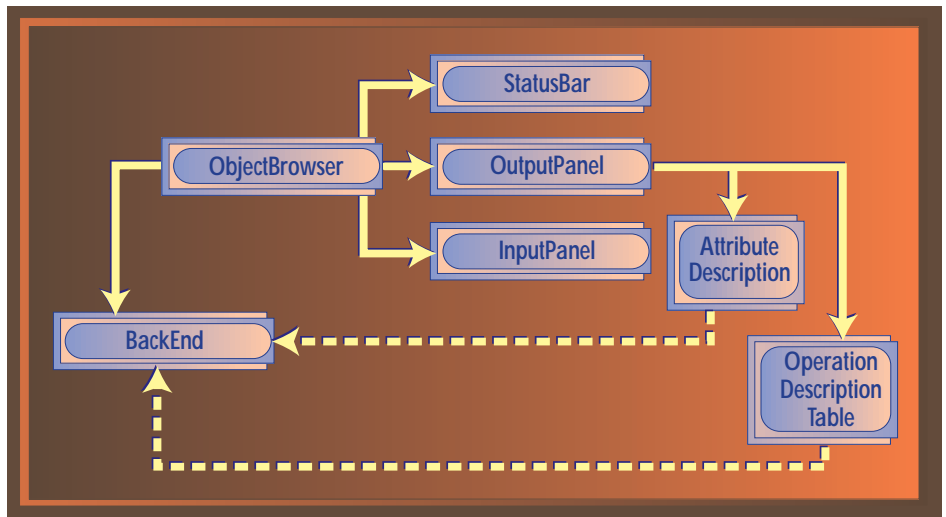


Figure 2: Class diagram of object browser

the IDL to the user on the OutputPanel.

The InputPanel class (see Listing 2) is derived from the JPanel and provides the user interface for accepting the object name. The user interface consists of an edit box and two buttons – Introspect and Exit. The event handler for the Introspect button calls the resolveAndIntrospect() method on the ObjectBrowser class by passing the received object reference. As mentioned earlier, the object reference is either the object’s registered name with the Naming Service or the complete URL containing the location of the IOR file.

The OutputPanel class (see Listing 3), derived from JTabbedPane, contains two tab panes: one for displaying the IDL to the user, the other for displaying the lists of operations and attributes. The init() method sets up the two panes. The Interface Definition pane uses a TextArea control to display the IDL to the user. The Operation and Attribute pane creates two Box objects and adds JList control to each for displaying the operations and attributes. Whenever the user changes the selection in the attributes list control, the showAttributeDescription() method is called. This in turn creates an AttributeDescription object and displays the information about the selected attribute to the user. Similarly, whenever the user changes the selection in the operations list control, the showMethodDescription() method is called. This creates the OperationDescriptionTable object and displays it to the user. The updateList() method updates the contents of the list control by first clearing it and then filling it with the new data.

The AttributeDescription class (see Listing 4) is derived from JFrame and displays the information on the selected attribute to the user. The class constructor receives the name of the attribute as a parameter and a reference to our ObjectBrowser class. The init() method calls its own getDescription() method to get the information on the desired attribute and then displays the information to the user by calling its displayFrame() method. The getDescription() method iterates through all the attributes defined in the

fullObjectInterface variable and retrieves the mode and type information for the desired attribute. The displayFrame() method then displays the attribute name, mode and type to the user.

The OperationDescriptionTable class (see Listing 5) derived from JFrame displays in tabular format the names of parameters required by the desired operation. The class also accepts the values for each IN and INOUT type of parameter from the user. In button event handler, if the user has clicked the invoke button, we copy the input parameters from the TextField objects in the table to a String array, which is then sent to the BackEnd object by calling its setParameters() method. The BackEnd will use these parameters while invoking the server method. Once the parameters are set, the program calls the local performDii() method, which passes the request to the processRequest() method of the BackEnd class. The BackEnd processes the request and displays the results.

The StatusBar class (see Listing 6) is derived from the JLabel and is simply a utility class for displaying status messages in the browser window.

Now we come to the most important class, BackEnd, which is responsible for all CORBA-related back-end processing (see Listing 7). The class constructor receives a reference to our ObjectBrowser and copies it into a local variable. The init() method first initializes the ORB by calling its init() method:

```
orb = org.omg.CORBA.ORB.init (param, null);
```

If the ORB is successfully initialized, we call resolve_initial_references() method on the ORB to resolve a reference to the Naming Service. We then narrow (type cast) the returned object to the NamingContext object type. The initialization of the BackEnd object is complete and the object now waits for its other methods (resolve, introspect, processRequest) to be invoked. The resolve() method receives the object as a string and tries to locate the object on either the current

Inprise

www.inprise.com/appserver

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

ORB or the ORB running at the specified URL:

```
if (!resolved = resolveUsingName(object))
    resolved = resolveUsingURL(object);
```

The `resolveUsingName()` method constructs the `NameComponent` array and tries to resolve the object reference by calling the `resolve` method of the Naming Service:

```
NameComponent[] name = {new
NameComponent(object, "")};
obj = nameService.resolve(name);
```

If this resolution fails, we try to resolve using the URL Naming Service. The method `resolveUsingURL()` obtains a reference to the `URLNamingResolver` and narrows it down to the proper data type as follows:

```
org.omg.CORBA.Object resolverObj =
orb.resolve_initial_references("URLNaming
Resolver");
com.vlsi.genic.vbroker.URLNaming.Resolver
URLresolver =
com.vlsi.genic.vbroker.URLNaming.
ResolverHelper.narrow(resolverObj);
```

Next, we use the `locate` method to resolve the object reference:

```
obj = URLresolver.locate(object);
```

Once the object is successfully resolved, we obtain its interface definition by calling its `_get_interface()` method:

```
InterfaceDef objIntf =
obj._get_interface();
```

The `describe_interface()` method then obtains the full description of this interface:

```
fullObjectInterface =
objIntf.describe_interface();
```

Now we look at the method called `introspect()`, in which we declare two `Vector` type variables for storing the methods attribute and operation lists. We obtain the number of attributes and operations from the full interface description as follows:

```
final int noAttributes =
fullObjectInterface.attributes.length;
final int noOperations =
fullObjectInterface.operations.length;
```

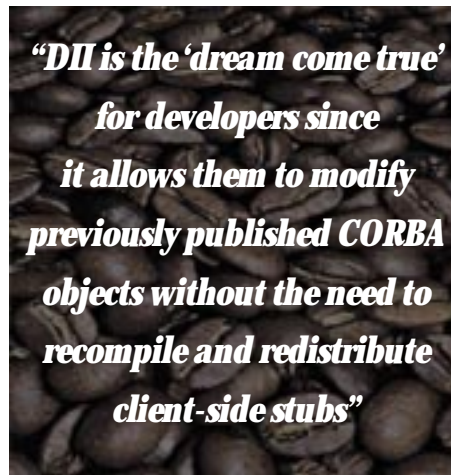
The program then obtains mode, type and name for each attribute:

```
for (int i = 0; i < noAttributes; i++)
{
...
mode =
fullObjectInterface.attributes[i].mode.value();
type =
```

```
fullObjectInterface.attributes[i].type;
name =
fullObjectInterface.attributes[i].name;
...
}
```

The attribute listing is then added to the `TextArea` control of the output panel where the IDL listing will be displayed to the user. The names of attributes are also added to the `attr` vector. This vector is used later to search for a user-selected attribute name. The program updates the list on the output panel by calling its `updateList()` method. We then construct the method signatures from the `fullObjectInterface`. This time we use an operations member variable to obtain detailed information on each operation.

Finally, we look at the most important method, `processRequest()`. This method obtains the name of the operation to be invoked as the parameter and invokes the method on the server object. First we obtain the index of



the desired method from the list of operations described in the `fullObjectInterface` object:

```
for (; i < noOperations; i++)
{
if (operationName.equals
(fullObjectInterface.operations[i].name))
break;
}
```

At this point variable `i` contains the index for the desired operation in the operations list. Next, we find the number of parameters required by this operation using the statement:

```
int noParameters =
fullObjectInterface.operations[i].
parameters.length;
```

For each parameter we now need to construct a proper object and an `NVList` containing all such parameters for dynamic invocation of the method. The ORB provides a method called `create_list()` for constructing an `NVList` object:

```
NVList parameterList = orb.create_list(0);
```

The parameter to `create_list()` method specifies the number of elements in the list. We pass zero as the parameter to construct an empty list. For each parameter we now construct an object of CORBA "Any" type using ORB's `create_any()` method:

```
Any currentParameter = orb.create_any();
```

The special type "Any" defined in CORBA is used to represent the element with any data type. We initialize this object with the proper data type by using the "type" member of `parameters[i]` variable. The program then examines the mode for the parameter, which can be `IN`, `OUT` or `INOUT` type. The method receives an input value through this parameter and returns the result to the caller through the same parameter. Depending on the value of the mode, we set our mode variable to the proper CORBA data type. For the `IN` type of parameter the mode variable is set to `org.omg.CORBA.ARG_IN.value`; for the `OUT` type it's set to `org.omg.CORBA.ARG_OUT.value`; and for the `INOUT` type it's set to `org.omg.CORBA.ARG_INOUT.value`. We also set an "accept" flag for each parameter. If the accept flag is set, it indicates we'll be assigning a value obtained from the user to this object (applicable to the `IN` and `INOUT` types of parameters).

The program then retrieves the value entered by the user for the current parameter by using `parameterValue` array. Note that all parameter values are stored as `String` data type.

We now set up a switch statement to convert this parameter value to the proper data type and assign it to our parameter object in an `NVList`. We check the parameter data type by using `kind()` method on the current parameter:

```
switch
( currentParameter.type().kind().value())
```

The CORBA "Any" class provides several `insert_xxx` types of methods to assign the objects of various data types to the "Any" object. For example, `insert_short()` method assigns an object of type "short," `insert_long()` assigns an object of type "long," etc. Thus, in the switch statement, we check for the kind of parameter required and use the corresponding `insert_xxx` method to assign the appropriate object type to our parameter object. We've programmed most of the cases in the switch statement. Some of the complicated cases are left out due to the complexity involved. For example, if you want to insert a structure data type, you'll need a Java class encapsulating the structure definition. You can then insert a Java object of this class for the desired parameter value.

Once the parameter is initialized with the proper data type and its value, we add the parameter to an `NVList` using its `add_value()` method:

```
parameterList.add_value(
fullObjectInterface.operations[i].
```

Cloudscape

Cloudscape is offering a FREE
30-day Evaluation License for
"Cloudscape Developer"

Cloudscape invites all systems
integrators to rock with us at a
special reception at JavaOne

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

```
parameters[j].name,  
currentParameter, mode);
```

The `add_value()` method requires the name of the operation, the value and the mode type of the parameter.

Once the `NVList` of operation parameters is constructed, we need to create a `Named Value` object for the return value of the method. We construct and initialize this `NamedValue` object using the following lines of code:

```
Any resultParameter = orb.create_any();  
resultParameter.type  
(fullObjectInterface.operations[i].result);
```

```
org.omg.CORBA.NamedValue resultField =  
orb.create_named_value("resultField",  
resultParameter,  
ParameterMode._PARAM_OUT);
```

We set the proper data type for the result by using the "result" data member of the desired operation. The `NamedValue` object is constructed using the `create_named_value()` method of `CORBA` Object class. At this stage we're ready to build our request object, which we do by calling the `_create_request()` method on the `CORBA` object:

```
Request request =  
obj._create_request(null,  
fullObjectInterface.operations[i].name,  
parameterList, resultField);
```

The `_create_request()` method requires four parameters. The first specifies the `CORBA` context and is set to null in this case; the second specifies the operation name; the third specifies the `NVList` containing the list of parameters required by the operation; and the fourth is another `NVList` in which the return value, if any, is returned.

You're now ready to invoke the method on the server, a simple process. You use `invoke()` method on the request object to invoke the method on the remote server:

```
request.invoke();
```

The method is executed on the remote server and the result, if any, will be returned in the result `NamedValue` object, `OUT` parameters and `INOUT` parameters. We now display the results to the user by constructing a `JFrame` object and displaying the information in label controls. The return value of the method is obtained using the expression `request.result().value()`. Similarly, the values of `OUT` and `INOUT` parameters are obtained using the expression `resultList.item(j).value()`. If the method invocation fails for any reason, we construct the `JFrame` object to display the appropriate message to the user.

Having seen the design of the object browser, we now examine how to compile and run the entire application.

Running the Browser

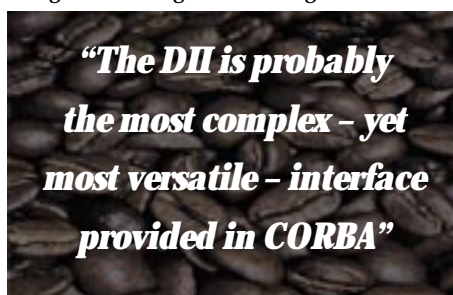
We used `JDK 1.1.5` and `Swing` classes for the development of the object browser. We've created several batch files for compiling and running the code. The `swingcompile.bat` file (see Listing 8) compiles the entire source. This file contains the following statement:

```
javac -cl asspath %SWINGPATH% %1
```

where `SWINGPATH` is the system variable defined on our system that sets up the class path for Java, Visigenic CORBA and Swing classes.

The parameter to the batch file is our main Java class - `ObjectBrowser.java`. If your classpath is set properly, running `swingcompile` batch file should compile all the relevant files.

Before you run the browser, you need to do certain startup operations, such as starting `OSAgent`, starting the `Naming Service`, etc.,



with the help of `startup.bat` file (see Listing 9). This batch file starts the `OSAgent` using the following command line:

```
start OSagent -c
```

Next, it starts the interface repository using the following command line:

```
start irep InterfaceRepository ABCOM.ir
```

The command `irep` starts the repository service. The name of our repository is `InterfaceRepository`, and it's stored in the file called `ABCOM.ir` in the current folder. Next, the `CORBA` Naming Service is started using the following command line:

```
start vbj -DORBservices=CosNaming  
-DSVCnameroot=ABCOM -DJDKrenameBug  
com.visigenic.vbroker.services.CosNaming.  
ExtFactory ABCOM namingLog
```

Our `Naming Service` starts with the root object set to "ABCOM".

This completes the operation of our startup batch file. When you run this batch file, you'll notice that three windows pop up on your terminal; one is used by `OSAgent`, the second by `Interface Repository` and the third by the `Naming Service`.

Now we're ready to run our object browser, which is started using `show.bat` file (see Listing 10):

```
vbj -DORBservices=CosNaming -
```

```
DSVCnameroot=ABCOM -VBjcl asspath  
"%SWINGPATH%" ObjectBrowser
```

Running this batch file starts our object browser. You're now ready to discover any object either in the current `ORB` or the `ORB` running at some known URL. We've provided a test server application for your convenience. It's called `SampleServer` (see `IDL` Listing in `Sample.idl` file) (Listing 11). The server provides three sample methods:

- `DoubleValue()`, which takes an `IN` parameter of type float and returns a float value to the caller as a method return value
- `CountCharacters()`, which takes one string parameter of `IN` type and returns the character count in an `OUT` type parameter
- `ReverseString()`, which takes an `INOUT` type parameter of type string, reverses the string and sends it to the caller through the same parameter

The implementation files, along with the compiled code, can be downloaded from the *JDJ* Web site. You'll need to run `idl2ir` to load the `IDL` into your `IR` before running the `SampleServer` application. The application registers itself with the `Naming Service` using the name "SampleObject". Type this name in the browser input panel and click on `introspect` to display the `IDL`. You can then select any of the three listed methods, invoke them and test the result.

Conclusion

The `DII` is probably the most complex - yet most versatile - interface provided in `CORBA`. It allows you to discover and use a `CORBA` object at runtime without the need for compiled stubs. We've described the entire `DII` mechanism and the design of a `CORBA` object browser that's based on the discussed techniques. 🍌

Acknowledgments

We'd like to thank *Kamal Shah* and *Bhakti Mehta* for their valuable help in the development and testing of the `CORBA` object browser.

▶▶▶▶▶ CODE LISTING ▶▶▶▶▶

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

About the Authors

Dr. Poornachandra G. Sarang is president and CEO of *ABCOM Information Systems Pvt. Ltd.*, a consulting firm specializing in Internet, Java, CORBA, Visual C++ and VB programming and training. He can be reached at sarang@abcom.com.

Mohan Rajagopalan is currently studying for a degree in computers, and plans to pursue a graduate degree specializing in distributed systems and heterogeneous networks. The `CORBA` object browser was developed as part of his practical training at *ABCOM*. He can be reached at mohan@abcom.com.

 sarang@abcom.com mohan@abcom.com

Oracle

Oracle is offering
FREE internet seminars

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



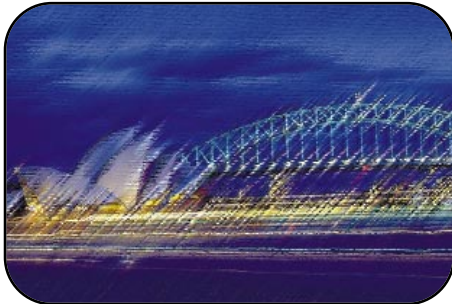
AN ONLINE AIRLINE TICKET STORE USING JAVA AND COLDFUSION

**Working
Together:**
*Competing, yet
complementary,
technologies*

Part 1

by Ajit Sagar

Online stores are the new, next-generation, “revolutionize the world as we see it today” way of doing business. In the context of business transactions, online stores use the global Internet to facilitate purchase and sale of goods and services. The ability to support online sales is an essential component of the new e-commerce paradigm for Internet-based businesses today. Putting together an enterprise-level application for an Internet store involves design and integration of various technologies that play specific roles in a distributed computing environment. A distributed topology is a prerequisite for building such Internet applications since the Internet is inherently distributed in nature.



Due to the plethora of alternative technologies available in the computing arena today, designing an enterprise application involves choosing between technologies based on feasibility, applicability, cost, availability and several other factors. No solution is "the right one." The hard part is to figure out which technology to use to provide a particular functionality. The challenge is to take competing and complementary technologies and make them play nicely together.

This is the first in a series of articles on some of the prominent technologies available to build a simple Internet-based Ticket Store application. This application offers online purchase of airline tickets as well as goods sold in airports. Our discussion is focused on two popular technologies: Java platform components (Java Applets and Servlets, RMI and JDBC) and Allaire Corporation's ColdFusion application server. The modules implemented using Java technologies will be presented here and in three subsequent issues of *Java Developer's Journal*. The modules implemented using ColdFusion will be discussed in Volume 1, issues 4-6, of *ColdFusion Developer's Journal*.

One objective of this design is to illustrate how Java Servlets can be used as access mechanisms in server modules that serve up data to front-end storefront modules implemented in ColdFusion. A simple protocol for exchanging name-value parameters between a Java Servlet running in the middle tier and a ColdFusion engine running in a front-end tier will be used for this purpose.

I'm assuming that readers are familiar with the Java technologies mentioned earlier. Knowledge of ColdFusion is not assumed for the *JDJ* articles. This article will concentrate on the modules that constitute the application, the technologies used to build the store and the "protocol" for sending data back and forth between the Java components and the ColdFusion templates.

Please note that this is not a real-world application, but one I put together to demonstrate how components and services implemented in an application server (like ColdFusion) and the Java platform may be used to build a distributed e-commerce application. This application will evolve over the next few months. Readers are encouraged to provide feedback if they'd like to take part in defining the scope and design of this application.

The Online Ticket Store

The Online Ticket Store is an Internet-based application that allows an Internet user to log in and purchase tickets via a browser. It accepts credit card payments for purchases and also maintains corporate accounts for its customers. The store is a front end for the services offered via airline agency back offices. These back-office locations provide ticket price/availability information and accept ticket reservations. The store:

- Acts as a ticket agent and may be used to

purchase tickets, comparison-shop between different airlines and determine flight itineraries.

- Provides an interface to a virtual airline store that allows catalog sales of merchandise such as clothes, appliances, computer equipment and other goods usually offered in airline magazine catalogs.
- Allows Internet users to lease equipment for in-flight use, including laptops and printer, portable CD players and music CDs. The idea is that a person who uses the Internet to reserve a flight can instruct the airline to have the leased equipment available on the flight he or she will be taking. The equipment will be available to the passenger on boarding and will be surrendered when he or she leaves the plane.

The software components that make up the Online Ticket Store, their functions and the technologies used for their implementation are illustrated in Table 1. Only the modules used for ticket transactions are listed. The modules used for building the virtual store will be discussed in a future article. We're not too concerned about the airline's back offices. Our focus on the back office will be limited to the access mechanisms used for the services offered by the back office. For example, how an airline prices and reserves tickets is beyond the scope of this article. However, the format in which a back office accepts a reservation request and the format of the response will be defined here and in subsequent articles. The transport mechanism (RMI/CORBA/other) will also be discussed and implemented, although security issues involved in transporting the data will not.

The application modules are illustrated in Figure 1.

Application Framework

The framework for this application is distributed among the following tiers:

- **Client UI:** The end-user interface into the Ticket Store. The client UI is responsible for the front-end interaction with the customer and connection to the data tier that maintains the Internet user's data and conducts transactions with the Internet user. In our application this is typically a Web browser.
- **Merchant Server tier:** Where the application server that serves data to the client UI resides. It has a data store for the customer's profile and maintains the shopping cart, catalog for merchandise, etc. It interacts with the Services Access tier to get information from the various data sources (airlines).
- **Services Access tier:** Middleware tier that accepts service requests from the Merchant Server tier, routes them to the Application Services tier and serves back the response to the Merchant Server tier.
- **Application Services tier:** Offers the ticket price/availability quote services.

Component	Function	Main Implementation Technologies
Personal Profile Manager	Maintains a personalized profile for end customer, including data about flight preferences, frequency of transactions, history of goods purchased, etc. May be updated by customer.	ColdFusion, Microsoft Access database
Travel Profile Manager	Keeps track of customer's travel history, preferred airlines, etc. The data here, captured and maintained by the Ticket Store, will be of interest to the airlines that collaborate with the store.	Java Platform (Servlets, JDBC, RMI), Microsoft Access database
Payment Manager	Manages payments made by customer while interacting with the system. Consists primarily of credit card-based payment management.	ColdFusion
Login Manager	Manages user login, authentication, passwords.	ColdFusion and Java Servlets
Shopping Cart	Keeps track of customer's current purchase as well as pending orders for purchased goods.	ColdFusion
Ticket Reservation and Sales Broker	The main operational module for conducting transactions related to online ticket sales.	ColdFusion and Java Servlets
Ticket Pricer	Interfaces with back-office modules to get ticket price and availability information.	Java Servlets, Java networking API, RMI, CORBA, JDBC

Table 1: Online Ticket Store functions and technologies used

Figure 2 illustrates the application framework tiers.

Java Servlets and ColdFusion

The Merchant Server interacts primarily with the Service Access tier via a URL connection and is implemented using the ColdFusion 4.0 application server. For the purposes of this application I developed a custom tag called CF_Servlet (see *CFDJ* Vol. 1, issue 4). While details of the custom tag aren't relevant

to the *JDJ* articles, they will be discussed in *CFDJ*.

Before describing the access mechanism, I'll elucidate why specific modules are implemented in ColdFusion while others are implemented using different Java technologies. ColdFusion is an application server whose most important feature is its ability to connect to data created and maintained in other applications. It allows the building of dynamic queries on the fly to retrieve data from such

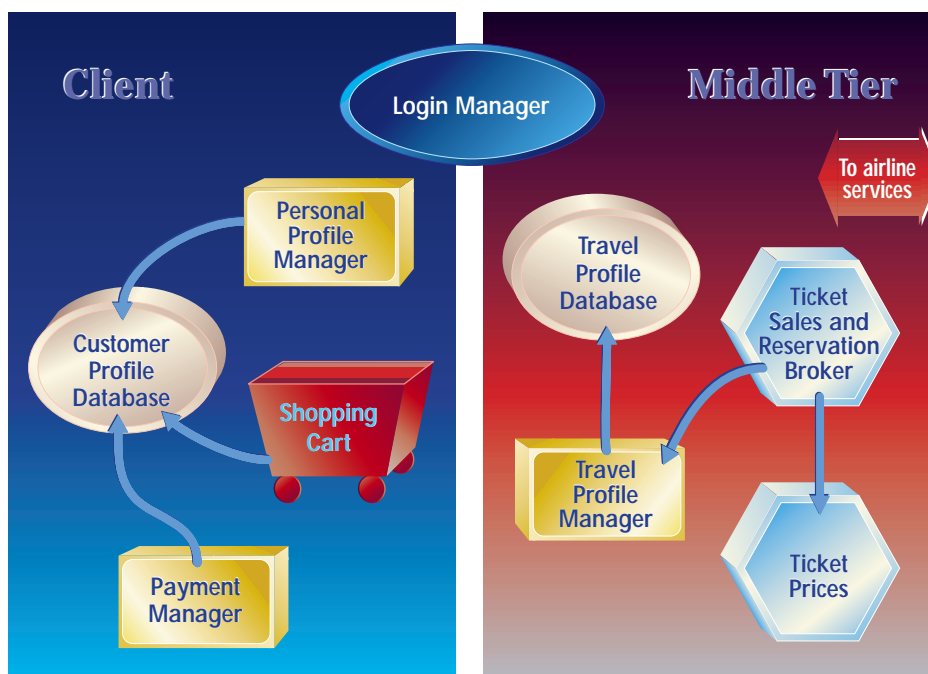


Figure 1: Application modules

applications. It achieves this through the use of a very flexible tag-based markup language. This makes it an ideal tool for creating dynamic Web application components such as shopping carts, account management modules, purchasing modules, customer profiles, etc.

Servlets, on the other hand, excel at making server-side services available to the client in a dynamic and interactive fashion. Servlets can be used to efficiently access a variety of services offered across different tiers of a distributed architecture. Servlets basically serve HTML to the client. They also bring access control and enhanced security into the equation. They are closely tied to the Web server they run in and thus help extend the server's capabilities to the client. One thing they're NOT designed for is building sophisticated GUIs. They are primarily "HTML servers."

Typically, ColdFusion-based applications and Java Servlets can be used in conjunction if the division of functionality leverages these strengths. In our application, all data pertaining to the customer should reside in the ColdFusion components and be presented to the customer in a snazzy and sophisticated user interface. However, when back-office services like ticket price/availability quotes need to be dynamically accessed (e.g., for submitting a request for a ticket quote), Java Servlets provide an ideal access mechanism to these services. The Merchant Server tier uses the Service Access tier:

1. To access server-side Java services. Non-Java (C++) services could be accessed using JNI.
2. To gain access to RMI/CORBA services.
3. To perform sophisticated computation-intensive tasks on the server as opposed to burdening the client with this responsibility.

This is based on the premise that the airline back-office services are publicly published as RMI/CORBA services. The Service Access tier dynamically queries the various airlines for the ticket price/availability and returns a quote based on some predetermined selection rules. The Service Access tier also maintains a database that captures each interaction with the Application Service tier. This database can be used for statistical analysis in the future, after the data has been accumulated over a period of time. The data would be valuable to the airline agencies. For example, if most customers have been rejecting an airline's quotes because the price is too high, the carrier would probably want to be aware of this fact.

Implementation

The rest of this article discusses the implementation of a test scenario for the application. Basically, it shows the interaction between a Java Servlet and an RMI server for getting a quote for a ticket. The servlet should be accessible from the ColdFusion template described in the corresponding *CFDJ* issue.

InterBase

Interbase is offering a
FREE 5-user evaluation
version of their embedded
database to JDJ readers

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

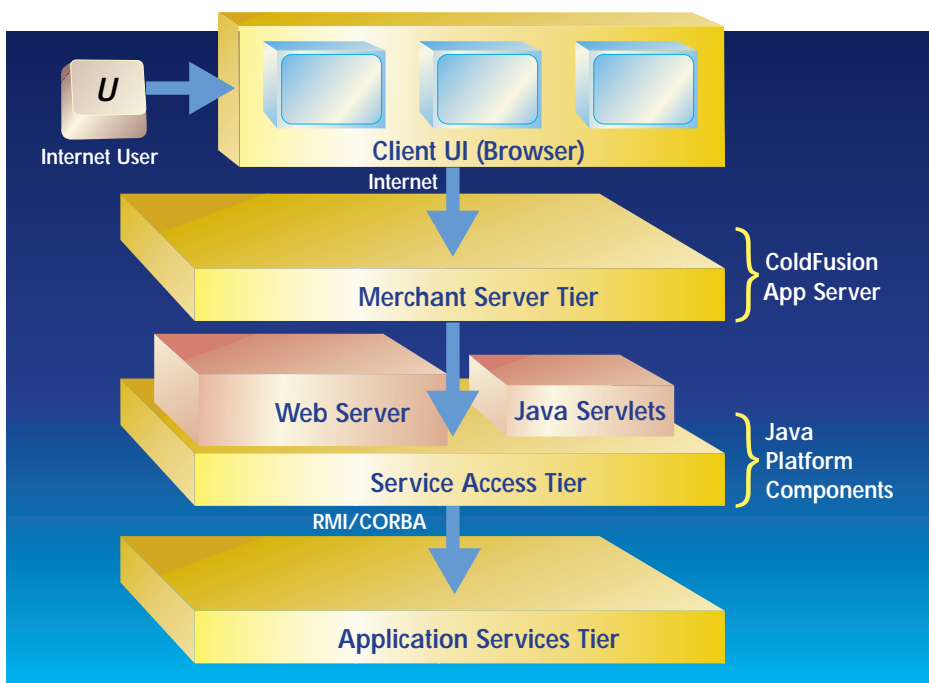


Figure 2: Application framework tiers

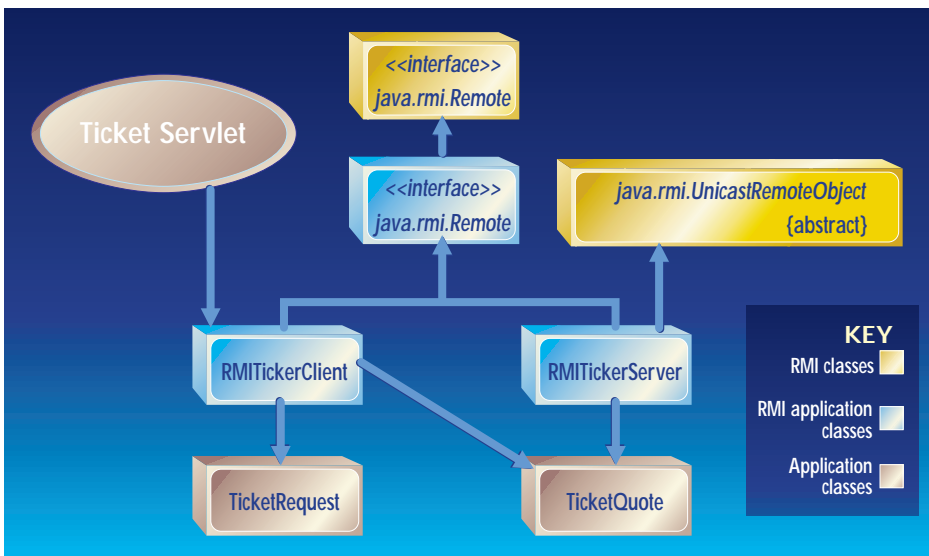


Figure 3: How the classes interact.

However, you can also run it from a regular browser by invoking the corresponding URL.

Before we define the protocol for communication between the Merchant Server and the Service Access tiers, let's examine the information we'd like to pass between the two. A typical end-to-end transaction in this system would have the following steps. Please note that for this transaction we're not making use of the Customer Profile Manager, Travel Profile Manager, Shopping Cart and other components that are a part of the application implemented in ColdFusion:

1. The user feeds in his or her input for a flight quote request via the Client UI (Web browser). This actually invokes the ColdFusion template on the Merchant Server.
2. The request goes to the Merchant Server. A ColdFusion template passes the data to the Service Access layer by invoking a servlet. Let us call this the TicketServlet.

3. The TicketServlet packages the data into a TicketQuery object.
4. The TicketServlet sends the TicketQuery object to the different airline carriers as a request for a price and availability quote.
5. The TicketServlet receives the quote from the different carriers in the form of TicketQuote objects.
6. The TicketServlet chooses the price quote/quotes based on some predefined selection rules.
7. The TicketServlet packages the response into HTML and passes it back to the Merchant Server.
8. The corresponding ColdFusion template at the Merchant Server converts the response into a Web page that it serves back to the Client UI. For this article the UI is in the form of a browser at the client site.

Here, we'll just look at the classes that will

implement this scenario. The logic in these example classes is hard-coded. The actual implementation will be developed in subsequent articles. The classes used for the basic scenario are:

- **TicketServlet** (Listing 1): Receives a ticket request from the Merchant Server, forwards it to an RMITicketServer and passes back the response to the Merchant Server.
- **TicketQuery** (Listing 2): Encapsulates a request for a ticket quote.
- **TicketQuote** (Listing 3): Encapsulates a ticket quote.
- **TicketServerList** (Listing 4): This is the interface for the services offered by the application services tier.
- **RMITicketServer** (Listing 5): Processes the TicketRequest object and sends back a TicketResponse object.
- **RMITicketClient** (Listing 6): This is accessed by the TicketServlet for submitting the request to a Ticket Server.

The TicketServlet is the access mechanism for the RMI services offered by the RMITicketServer. The access protocol between the ColdFusion template and the TicketServlet is very simple. Since Servlets are accessible via a URL, the most convenient way to pass data between the two is in the form of Name-Value parameters that can be passed in the query string of the URL. The Servlet creates a TicketQuery object from these parameters. The TicketQuery object is propagated directly to an RMITicketServer via a RMITicketClient object. The RMITicketServer creates a TicketQuote and passes it back to the TicketServlet. The TicketServlet creates an output string in the form of Name-Value pairs, which are passed back on its output stream to the ColdFusion template.

The flight quote requested from the client is simple and doesn't contain all the fields necessary to process the actual flight request that will be used in our application. In fact, the TicketQuote only contains a single field, which is the price for the flight. The refined classes for the application will be discussed in the next part of this article series.

Environment

The code for this article was developed using JDK 1.1.7B on an NT 4.0 workstation. The TicketServlet is accessed using Jrun Pro 2.2. The code listings may be obtained from the **JDJ** Web site. 🍌

About the Author

Ajit Sagar, a member of the technical staff at i2 Technologies in Dallas, Texas, holds an MS in computer science and a BS in electrical engineering. He focuses on Web-based e-commerce applications and architectures. Ajit is a Sun-certified Java programmer with nine years of programming experience, including two and a half in Java. You can e-mail him at Ajit_Sagar@i2.com.



Ajit_Sagar@i2.com

Blue Sky

www.blueskysoftware.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



Happy Birthday!

A look at what's happened over the past year and what's going on now

by Alan Williamson

Can you believe it? I know I certainly can't. This column is officially celebrating its twelfth issue, and being the mathematical genius that I am – and since this joyous magazine is printed on a monthly schedule – I can safely deduce that our first year anniversary is upon us. Fantastic. Experts reckon that most marriages break up in the first year, so I guess we've successfully made it past the hardest part.

On that note I think we'll take a whistlestop tour of what I've been ranting about over the last year and look at any changes that make the arguments posed in the last 12 months seem silly now.

Updates

I've looked at a variety of issues that touch most of us in the Java world. As regulars know, this column isn't afraid to walk on the ice of controversy. It's been known for us to name names, and to make some of the bigger names accountable for their actions. A regular figure in this has been Oracle.

Oracle produces its own set of JDBC drivers for us developers to use to interact with their excellent back-end database. On the database front, we can't really complain. They do handle data very well. If you remember, however, back last summer we had a real problem using this functionality due to the poor implementation of their driver. Back then, Oracle didn't really embrace the developer community.

Another issue we've touched on is that of recruitment and finding that special developer. Well, our search is still continuing. And boy, are we learning a lot about the process along the way! Last time, I commented that the overall standard of Java is becoming lower and lower as everyone and their dog decide they want to jump on the Java bandwagon. I've seen no evidence in the time between that article and this one to alter my opinion.

It's the time of year when the universities open up their doors and a raft of graduates pour out. Many of them have degrees, which, I have to say, probably aren't worth the paper they're written on. I'm sure the American system is no different, but here in the UK there is a big drive to keep bums on

seats in the universities as long as possible as this affects the universities' funding from the government. This has the knock-on effect of trying not to fail people. When I was at university the pass rate for our finals was 40%. Even back then I thought this ludicrous. You only had to know 40% of your subject to get a degree! Doesn't seem right, does it?

Well, I think the industry is going to suffer badly from it. We're experiencing this now: people coming out with real generic degrees, such as "I.T.," thinking they have a skill set, only to realize that they aren't a great deal of use to employers. The reason I pick on the IT brigade is that some of them have taken a small Java course and suddenly think they're software engineers. I don't think so.

Everywhere we read, there's a real serious skill shortage. I can see this now; before, I couldn't. I couldn't equate the numbers originating from final year degrees to the number of positions actively sought by employers. It was an equation that on paper should have added up. In reality it didn't.

My concern with all this, as regulars will know, is keeping the quality of development up. Let's not pretend to people that if they take a six-month course in Java they're suddenly software engineers. This isn't doing them any good, and it sure as hell isn't doing the industry as a whole any favors. I'd love to hear your thoughts on this, so please join our mailing list and let me know what you think.

Mailing List

Whoa! Let me say "thank you" to all the subscribers to our *Straight Talking* mailing list. This is providing some very interesting and stimulating conversation. The quality of posts coming through on the list is quite astounding, and the mix of people we have contributing ensures a balanced debate from all walks. Please feel free to join us. You don't have to contribute immediately – just listen to the rants and raves of everyone. It isn't a technical list. We don't debate Java problems. We look at the issues and concerns facing the upcoming Java developer. To join send an e-mail to listserv@listserv.n-ary.com with

"subscribe straight_talking-1" in the body of the e-mail. From there you'll get instructions on how to participate.

Salute of the Month

Let me introduce you to a new section: *Salute of the Month*. Over the last few months I have been openly thanking people for their various contributions to the Java world and I thought I might as well formalize it. So here goes, the first of a new series.

As this column rolls into a new year of life, I can say I have met many new friends through this magazine you're holding in your hand. Over the 12 months I've been slowly building a picture of who you are, and trying to write material that is of interest to you. One of my biggest surprises was the fact that many of you aren't developers. Many of you have never coded a single line of Java in your life and probably never will. I received an e-mail from an atypical reader from this group a couple of months ago. Sallee Gambino, a recruitment consultant in New Jersey, e-mailed me to say how much she valued my column as it helped her keep abreast of the latest developments in the Java universe. Over a period of e-mails, I probed this lady for more information, trying to figure out how reading this column and this magazine helped Sallee recruit better.

As you know, I've had my share of things to say about agencies that can't even tell the difference between JavaScript and Java. My experience has shown me that many of them fall into this category. So you can imagine my surprise when I bump into someone that is actively taking an interest in the area they are recruiting for as opposed to blind CV-keyword matching. Sallee wants to catch out a lot of the buzzword kings and therefore offer a higher caliber of candidate to her end client. Also, knowing the industry allows Sallee to understand more of her clients' needs. ☺

About the Author

Alan Williamson is CEO of n-ary Ltd, a Java consultancy company with offices in Scotland, England and Australia, specializing solely in Java at the server side. Alan is the author of two Java Servlet books and contributed to the 2.1 Servlet API. He can be reached at alan@n-ary.com (www.n-ary.com) and welcomes all suggestions and comments.



alan@n-ary.com

Zero G Software

www.zerog.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



Java and XML in the World of E-Commerce

How these two technologies provide such a powerful combination for enabling e-commerce applications

by Ajit Sagar

In last month's e-Java we discussed the technologies and APIs offered by the Java platform that play specific roles in e-commerce solutions for the enterprise. We also took a high-level glance at how they fit in an *n*-tier commerce application. Java provides substantial support for e-commerce-based applications; however, Java alone doesn't provide an end-to-end solution for e-commerce-based businesses. Components of the Java platform integrate with complementary technologies to offer complete Internet commerce solutions.

Nowadays the acronym "XML" is found lurking in almost any literature or site that mentions e-commerce. Some consider XML (Extensible Markup Language) to be the enabling technology that will allow the World Wide Web to become a breeding ground for open commerce-based applications. XML helps expand existing technologies into new dimensions of document publishing and content management, addressing the needs of a distributed Web-based environment.

The Java platform and XML are complementary technologies that together can provide a solid foundation for commerce-based transactions across the Internet. This month in e-Java we'll look at how Java and XML can work in the world of e-commerce. Please note that this article doesn't focus on XML itself (although a brief discussion is provided in the next section) or on Java, but rather on how the two technologies provide such a powerful combination for enabling e-commerce applications.

Data Content, EDI and XML

The premise behind Internet commerce is that business transactions can be conducted efficiently and safely over different tiers of the Internet. The evolution of client/server architectures, which are an integral concept behind the Internet, has seen a migration from two-tier architectures to multitier distributed architectures. This

places greater demand on the safe and correct passage of electronic data from an Internet source to the corresponding destination. The whole purpose of such commerce transactions is to transfer data content between the two parties participating in the trade.

EDI (Electronic Data Interchange) has been the enabler for electronic transactions for several years now. EDI is a process for exchanging data in an electronic format between heterogeneous applications and/or platforms in an automated fashion, i.e., without manual intervention. However, when applied to an Internet environment, it falls short on several accounts, some of which are:

- Scalability across multiple tiers of the Internet
- Lack of support for dynamic data content definition
- Slow adoption to standards
- Lack of flexibility in defining business rules

XML leverages EDI technology and adds the necessary pieces to migrate it to an Internet environment. It does so by adding the following:

- Ability to separate the data and structure (electronic content) from the processes
- Ubiquitous interconnectivity between trading entities (this isn't directly provided by XML, but rather by the Internet)
- Support for dynamic data format definitions
- Middle-tier storage for caching data
- Widely accepted standards governed by the W3C (World Wide Web Consortium)

HTML and XML

HTML is a presentation markup language that offers a standard format for displaying data in a Web browser. While it provides a standard format for displaying data, the focus is on the visual presentation of data and not on sophisticated data types, as required by e-commerce applications. XML, on the other hand, focuses on data content and provides a clear separation between content and visual presentation. It does this by allowing the applications to define tags that describe the data they're exchanging across the Internet. XML allows the tag information to be embedded in the document itself, so that the document is "self-describing." XML standards allow business partners

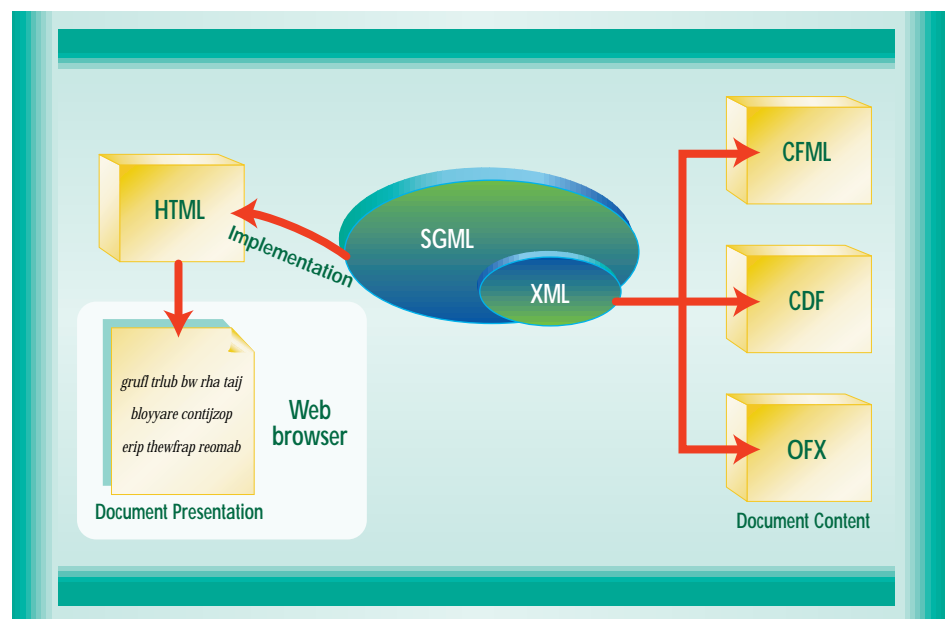


Figure 1: Relationship between SGML, HTML and XML

Riverton Software

www.riverton.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

to define their own self-describing markup languages for supporting electronic commerce.

Both HTML and XML are derivatives of SGML (Standard Generalized Markup Language). However, while HTML is an implementation of SGML for visual presentation, XML is a subset of SGML used for writing markup languages that specify the format for expressing data content. Figure 1 shows the relationship between HTML, SGML and XML.

XML + Java = Portable Objects

So how does all this relate to Java? Well, there are two main aspects of an Internet commerce application. We've taken a look at the first - data format and content. XML enables electronic data to be represented in a platform and language-independent manner. This data can be available across several different enterprise domains. However, data by itself doesn't do anything. It needs to be processed by applications. For XML data to be used in a platform-independent manner across the Internet, it needs a hardware-independent environment. The Java platform provides such an environment by offering a distributed, homogeneous computing architecture that resides on the different tiers of the Internet. Java "Internet-enables" XML documents by adding networking and cross-tier transportation capabilities.

Java and XML together address the issue of portability in e-commerce, an issue that's crucial for successful implementation of Internet-enabled commerce applications. XML provides data that's portable across different application domains. It does so by representing data in a common, well-known format. Java provides code that's portable across different hardware and operating systems. The code adds the functionality for processing and manipulating the data. A natural consequence of this alliance is highly robust, componentized and maintainable applications.

The synergy between XML documents and Java code is due to the fact that the data encapsulated by XML tags can be easily expressed as Java objects and vice versa. The entities that take part in an e-commerce application are defined in an object-oriented manner as "data objects." This gives rise to "objects" that are portable across the different tiers of a distributed application. You can almost look at these as "portable objects" that morph into XML elements (defined by tags) when they need to be transported between hosts. They morph back into Java objects when they need to be processed or manipulated.

In other words, working between XML and Java involves the following steps:

- Constructing a Java object model from an XML document by creating Java objects from XML tags.

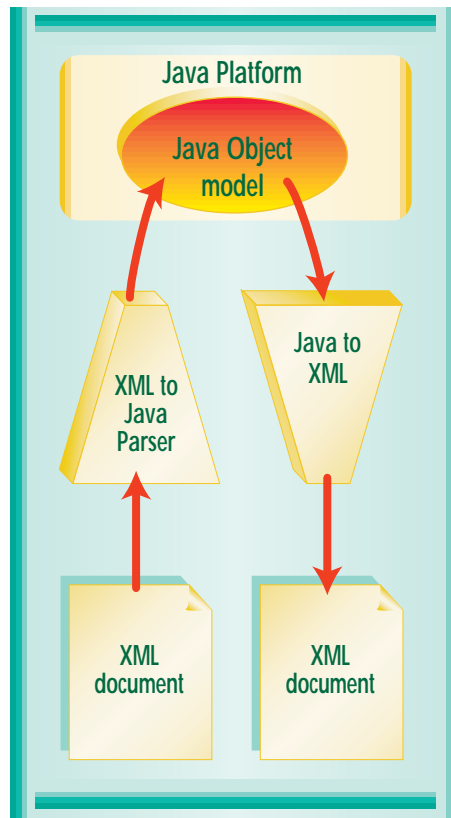


Figure 2: XML and Java document representation

- Using the Java object model to process the data. This should focus on computation-intensive tasks that need to be performed on the data to support the overall business transaction.
- Generating an XML document from the Java object model by converting the Java objects to the corresponding XML data tags.

The transformations between XML and Java are illustrated in Figure 2.

DOM and SAX

An XML document has to be processed in order to be used. This requires an XML processor that can:

- Parse XML documents - an XML parser
- Create XML documents - an XML editor
- Provide an API to access the parts of an XML document

The parsers and editors may be written in Java, which makes them portable across various platforms. The real key to the XML-Java interoperability, however, is the APIs used to access the elements of an XML document. An additional advantage Java brings to XML parsers and interpreters is allowing the XML interface with the Java data processing application to be accessible via different Java components - server-side Java Servlets, client-side Applets, middle-tier EJBs or JavaBeans in an IDE. Java also adds its robust security services to the mix. As mentioned earlier, Java components reside in different tiers of a distributed application.

The XML APIs are of two types:

1. **Document Object Model (DOM):** DOM is a platform-independent, language-neutral, tree-based API. It compiles an XML document into an internal tree structure and allows an application to navigate that tree based on the API it exposes. One of the specifications of the DOM interface is in Java (the others are in OMG IDL and ECMAScript). The Java API for DOM is known as *Java Language Binding* and describes the Java DOM interface.
2. **Simple API for XML (SAX):** SAX is a simple, event-based API for XML parsers. SAX doesn't require the creation of an internal tree structure to represent the complete parsed XML document. Instead, when parsing an XML document, it provides

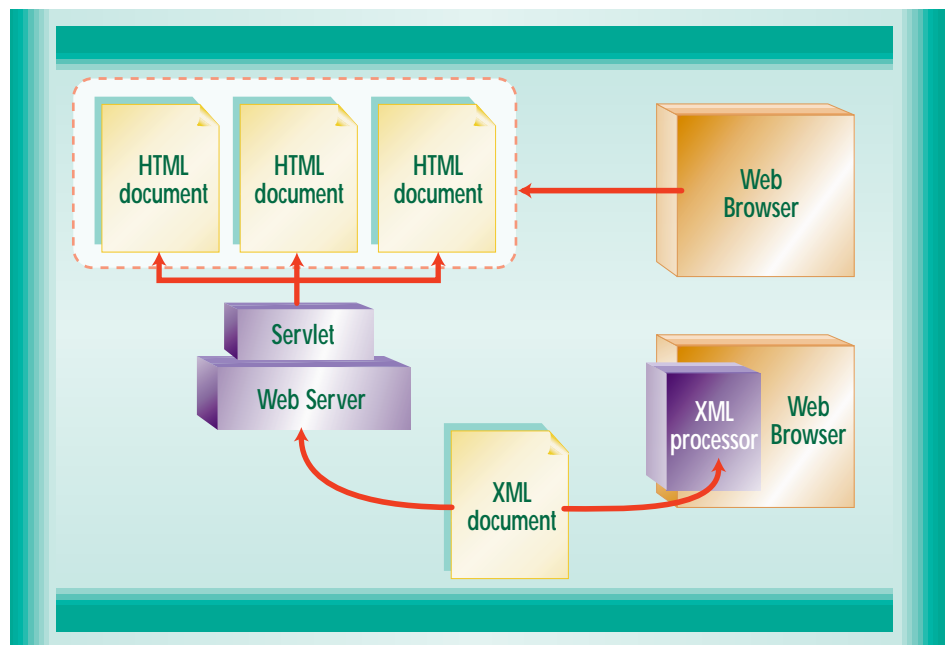


Figure 3: XML presentation in a Web browser

Progress Software

www.apptivity.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

events that indicate the detection of the following: start/end of the document, Document Type Declaration (DTD), start/end of elements, element attributes, character data, unparsed entities, processing instructions and more. SAX provides a Java interface that allows any Java application to access any XML parser as long as the parser has a SAX driver.

Publishing XML

In a transaction, XML may be used to transport data between the two partners. The data that's obtained at one end of the transaction may be processed but need not be displayed in a UI (indeed, that's the advantage of separating content from pre-

sentation). On the other hand, one of the desirable features of Web-enabled data is the ability to view the data in a Web browser. Such data may be in the form of a catalog, a quote request/response, etc. In a simple client/server scenario, there are several ways of rendering XML data in a browser:

- Process XML into HTML on the server and send to the client browser.
- Process XML into HTML directly on the client and display on the browser.
- Display the XML document directly.

Of these three, the third mechanism makes use of XSL (Extensible Style Language). The other two can be achieved by Java platform components. Java Servlets

can be used to process XML on the server and send the output to the client as plain HTML. XML-aware Java Applets can be used to process XML on the client. The applet can bind the required parts of the XML document into particular HTML components for display. Browsers like Microsoft's IE 4.0 support this functionality.

These two mechanisms for rendering XML in a Web browser are illustrated in Figure 3.

Role of XML and Java in Enterprise Application Integration

This is the year in which the Java platform APIs are really making their mark in integrating enterprise applications. Although the definition of Java's Enterprise APIs has been there for a while, they're only now achieving a level of maturity that makes them suitable in real-world enterprises. EAI is the mechanism for integrating several disjointed applications to provide a common application interface. XML allows Java objects to be represented as data that moves across middleware tiers that may not be Java-based.

XML/EDI DataBots

Remember that XML transports data content, not behavior. The XML/EDI initiative is based on the concept of active objects that have inherent processes associated with them. These are defined by rule templates that can be supplemented by XML/EDI data manipulation agents (DataBots) to ensure that users can express their requirements in high-level, natural language. Currently, the ECMAScript subset of the Java programming language provides the vehicle that permits the DataBots to be deployed and received along with XML/EDI messages.

Trading Places

Java and XML complement each other in facilitating Internet-based e-commerce applications. However, the compatibility between such technologies, their common environments and their complementary role in designing e-commerce applications is largely because they've found a common home in the Internet. In other words, the Internet is the environment that has created the need for, and promoted the wide acceptance of, enabling technologies like Java and XML in the world of e-commerce. ☛

About the Author

Ajit Sagar, a member of the technical staff at i2 Technologies in Dallas, Texas, holds an MS in computer science and a BS in electrical engineering. He focuses on Web-based e-commerce applications and architectures. Ajit is a Sun certified Java programmer with nine years of programming experience, including two and a half in Java. You can e-mail him at Ajit_Sagar@i2.com.



Ajit_Sagar@i2.com

Specialized
Software
www.specialized-software.com/jdj

Cerebellum Software

www.cerebellumsoft.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Anonymous Deployment vs Portability

With Java's technical/business architecture, the dynamics of an embedded database can change

by Bruce Scott



A lot has been said and written about Java's "write once, run anywhere" (WORA) capability. There have been both supporters and detractors (e.g., "write once, debug anywhere"). Java's statement of WORA raised expectations about Java to the highest possible level and Java is often measured against these expectations.

The inventors of Java made the WORA claim based largely on Java's architecture, which includes both a technical and business architecture. The former includes such things as the Java Virtual Machine, Java bytecodes and standardized Java APIs. With the JVM, model applications can be developed based on a standardized language and API, and be protected from the need to incorporate special code for each supported platform.

The Java business architecture is supported by Sun's JVM licensing program that essentially distributes the development of platform-specific JVMs throughout the computer industry with such companies as Microsoft, Apple, IBM and Oracle. It's interesting to note that Java supports not only distributed computing from a technical viewpoint, but distributed industrywide software development from a business viewpoint as well. Java is successful and will continue to be successful because of the synergy between its technical architecture and business architecture.

It's true that Java initially didn't live up to the WORA expectations it set for itself. Even now there isn't a perfect WORA world for Java developers. This is an ongoing process and it appears that there is continuing progress.

The real value of Java can be determined not by how it measures up against its lofty expectations, but by how it compares to the old world of portability.

My experience with portability comes from the early days of Oracle. The first version was developed in assembly language. We soon discovered C, and set about to create the world's most portable database. History has shown that we did indeed accomplish this.

As with Java, a technical and business architecture supports Oracle portability. The technical architecture was simple. We segregated code into generic (kernel) modules and operating system-dependent (OSD) modules. Initially there were 30 or so OSD modules. Today there are hundreds. OSD modules are rewritten and maintained for each platform Oracle supports. Essentially Oracle has created its own virtual machine. If invented today it might be called the OVM.

The Oracle portability business architecture is quite simple as well. With Oracle's multibillion-dollar revenue base, it can finance the continual cost of porting to the 70+ platforms Oracle supports today. It's estimated that for every kernel database engineer there are about three porting engineers at Oracle. Even Oracle, however, is feeling the pinch of the high cost of porting, and

has started to reduce the number of supported platforms.

This technical/business portability architecture has served Oracle well over the past two decades. But what about companies in the embedded database market? The model breaks down on the business side. Embedded database companies have revenue bases that are an order of magnitude smaller than Oracle's. The result is that these companies are forced to select a small number of platforms to support. They all support Wintel and possibly two to three others.

On the other side of the fence, customers are faced with a large installation matrix. They're required to purchase, install and maintain particular versions of Oracle with particular hardware and operating system versions. This must all be done ahead of time, before deployment. Should computers be added to a distributed system, more installations and configurations must be done.

Enter Java and "anonymous deployment."

With Java's technical/business architecture, the dynamics of an embedded database company can change - and has changed in the case of PointBase. At PointBase we don't employ a single porting engineer, yet we have more porting engineers than Oracle. Our porting engineers are on the payrolls of such companies as Sun, Microsoft, IBM and Novell. With our small history (a year plus) and small engineering staff our product has been used by our customers on more platforms than Oracle supported in its first five years.

From our customers' point of view, the predeployment step of buying and installing a platform-specific product is completely eliminated. Our customers can deploy their data

and data management through the Internet to an anonymous platform. This, after all, is what many have chosen Java for.

The old model of portability creates barriers to an emerging multipatform world in terms of both cost and time to market. As new platforms are created, data management solutions for those platforms need to be made available - rapidly. Those developing e-commerce servers know they're faced with many viable server platforms including Windows/NT, Solaris, Netware, OS/400 and Linux. People developing mobile applications are developing for Wintel laptops today, but expect to be able to move these applications to future mobile platforms. The emerging world of Internet devices and Web appliances will be inherently multipatform. As Java applications are created for these platforms, the data management solution needs to be as anonymously deployable as the Java applications themselves. ☛

About the Author

Bruce Scott, president, CEO and founder of PointBase, is a leader in the area of enterprise and embedded database architecture and product development. A cofounder of Oracle in 1977, Bruce cofounded Gupta Technology in 1984, pioneering the notion of the small footprint database server for Intel-based platforms.

"People developing mobile applications today expect to be able to move these applications to future mobile platforms"

Object Domain Systems

www.objectdomain.com/odtrial

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

KL G

www.klgroup.com

See JDJ Spe
<http://www.sys-con.com/ja>

roup

om/jclass/look

cial Offer at:
va/specialsoftheweek.html

Text Controls by Swing

Using Model View Controller architecture

by Jim Crafton

When Java first came out, one of its acknowledged weaknesses was the lack of an advanced set of GUI components. This was especially evident in the text controls, which lacked many of the advanced features found in the native text controls of operating systems such as Windows. With the release of the Java Foundation Classes (also known as Swing), Java finally had a robust and sophisticated collection of controls, especially text controls.

With this in mind, this series of articles will show how to build a simple syntax, with emphasis on text control using Swing's Model View Controller (MVC) architecture and JTextPane. The control will handle keyword highlighting by changing the font style to a bold format. It will handle both single-line ("//...") and multiline ("/*...*/") comments by changing the style to an italic format and the font color to a user-defined one. Strings (any text between quotes) and numbers will also be formatted by having their respective font colors changed to user-defined ones. The control will support all of the Java language keywords as a default, but will allow the user to change or completely replace the keywords with their own.

Explanation of MVC in Swing Document Model

The Swing set of GUI components provides a complete package of components, classes and interfaces that rely heavily on the Model View Controller architecture. With this in mind, I've provided the following brief overview of what MVC means to Swing, especially the Swing text components.

Model View Controller in Simple Terms

The Model View Controller architecture is a commonly used OO technique to distinguish the data, the rendering of the data and user interaction with the rendering and the data. In MVC the Model holds all the data necessary for the View, the View is responsible for deciding how the data in the Model is presented and the Controller is used to handle events and processing between the View and Model (see Figure 1).

How does the Model View Controller apply to Swing's text class, specifically the Document classes?

Swing uses a condensed version called *UI delegate* in which the Controller and View are

combined into one, with the Model remaining separate, as shown in Figure 2. In this article we're concerned primarily with implementation of a Model, specifically the Document model used throughout the Swing text controls.

Pros and Cons of Using Swing's Model View Controller Architecture

Using MVC throughout the Swing class hierarchy has allowed Sun Microsystems to create a very sophisticated architecture that allows all sorts of advanced customizations. The only downside is that the complexity of performing even simple tasks with the controls tends to intimidate newcomers to the Swing style of doing things. Hopefully, these articles will clear up at least part of the confusion in working with the text controls.

The Document

In Swing, as I mentioned before, the Document serves as the Model in the MVC architecture. So what is the Document? It's an interface describing the core functions of the Model and has several implementations already present in the Swing text package: the AbstractDocument class, the PlainDocument class and the DefaultStyledDocument class.

Every Document is composed of a series of elements arranged in hierarchical order. These elements, which are represented by the Element interface (part of the com.sun.java.swing.text package in JDK 1.1.x), serve many purposes. They break the Document into logical sections and can tell you where you are within the Document. In our case the primary use of the elements is to determine our place quickly so we only have to deal with the related text without having to sort through all of the text all of the time.

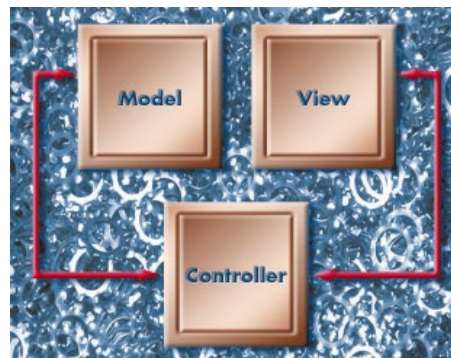


Figure 1: The Model View Controller

While the Element interface is great for navigation, it doesn't tell you much about how it looks. This is done by the AttributeSet interface. The AttributeSet also has a more complex descendant called the MutableAttributeSet interface, which allows us to modify the attribute rather than merely access information from it. The default implementation of these interfaces is provided in the SimpleAttributeSet class. Using this class you can modify font size and color, italic and bold styles and many other characteristics.

These two classes now enable us to determine our position, get a representative chunk of text and make modifications to the text attributes accordingly.

Let's take a look at how we could get our position using the Element interface. Assume that we've been provided with a variable (int offs) that represents the current offset in all the text, not just the chunk in the Element. The "this" variable points to a Document implementation.

```
Element element =
    this.getParagraphElement(offs);
int elementStartOffs =
    element.getStartOffset();
int elementTextLen =
    element.getEndOffset() -
    element.getStartOffset();
String elementText = this.getText
    (elementStartOffs, elementTextLen);
```

The elementText variable now represents only the text within the bounds of our particular element. This eliminates the need to have to parse through all the text. To better explain this, let's say you're chugging along, typing merrily away. As long as your cursor is always at the last character of text, backing up and determining where the next space is (which would signify a word break) is pretty easy. But what happens if you suddenly decide to go back and place the cursor in the middle of all the text you just typed? How do you determine where you are without processing through all the text? This is where using the Elements, as we're doing above, saves us.

Modifying the attributes of a select portion of text is also easy, as in our next example:

```
package MyTest;
import java.util.*;
public class TestClass {
    int i = 0;
```


OpenLink Software

www.openlinksw.com/virtuoso

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Now let's say the `elementText` variable we mentioned above contains the text "import java.util.*;" (as in preceding code). The "this" variable references an instance of some `DefaultStyledDocument` object.

```
SimpleAttributeSet bold =
    new SimpleAttributeSet();
StyleConstants.setBold(bold, true);
String word = elementText.substring(0, 6);
this.remove(off - word.length(),
    word.length());
this.insertString(off - word.length(),
    word, bold);
```

First we create the attribute by using a new instance of the `SimpleAttributeSet`. Using the `StyleConstants.setBold` method will modify our new attribute variable. Notice that we have to remove the text we're about to change with the Document's `remove` method. Then a call is sent to `insertString`, passing in the position, the string and the current attribute settings. By doing this we can change the formatting of the text "import" to a bold font:

```
package MyTest;
import java.util.*;
public class TestClass {
    int i = 0;
```

Now that we know how to work with and modify the Document's text, let's look at how we're actually going to go about setting up our syntax highlighting. For simplicity's sake I chose to create a new class called `CodeDocument` that extends the `DefaultStyledDocument`. As I mentioned earlier, three classes are already coded for us in Swing: the `AbstractDocument`, which is the base class for all the Swing Document implementations; the `PlainDocument` for straight text handling; and the `DefaultStyledDocument`, which handles all the fancy text formatting commonly found in RTF or HTML files. Since we'll need the text formatting attributes, it was a no-brainer to decide to extend `DefaultStyledDocument` (see code above).

Capturing text input turned out to be quite easy: we just override the `insertString(int offs, String str, AttributeSet a)` method and place our own functionality here. The `insertString` method is called whenever any kind of

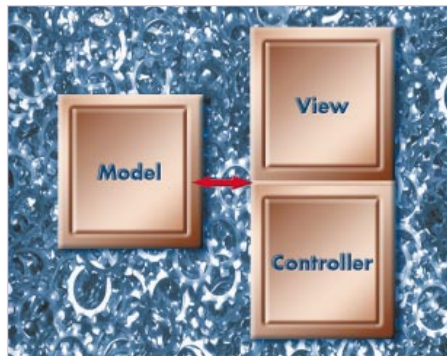


Figure 2: The Swing UI delegate relationship

text is entered, either by keystroke (making the `str` value just one character) or by programmatically adding text to the Document. To support this added functionality we'll add some attributes to our `CodeDocument` class. We'll need to keep track of the current word that we're on, so we'll have an attribute called "word" of type `String`. We'll also keep track of the current position, modifying it each time we enter our overridden `insertString()` method. We'll call this attribute `currentPos` of type `int`. We could use an attribute to hold the bold settings so we don't have to keep creating one. We could also use an attribute of type `Vector` to hold all the keywords. We'll call this variable "keywords."

Now the class looks something like what is contained in Listing 1.

We have two functions in this new class: the overridden `insertString` and a new function called `checkForKeyword`. The former is called every keystroke, or any other time text is inserted. The first thing we do is call the superclass's `insertString` function to ensure that the string is properly put into the Document's text. We then see whether the string is one character (most likely from a keystroke) or multiple characters (which we won't handle at the moment). If it's one character, we get that character and compare it: any character that signifies a word break (defined as a space, "{", "}", "(", ")", carriage return or ";") will cause us to call the `checkForKeyword` function.

This function basically pulls off the current word from our current position (stored in the `currentPos` attribute) and compares it with the list of keywords that the class has. If a match is found, the word is removed and replaced with the same word, but with bold-

style formatting. Walking through the function, the first thing it does is set a local variable called "offs" equal to the current position. From this position, as discussed above, we can get the current element text, based on the element's offsets (using the `element.getStartOffset()` and `element.getEndOffset()` functions). If the text retrieved has a zero length, we can safely return from the function.

The next chunk of code handles the problem of typing some text for a while and then moves the cursor back to someplace in the middle of the newly entered text. If this is the case, the `offs` variable needs to be translated to match up properly with the array of characters in the element text. With this taken care of, we're ready to start walking backwards through the text. Backwards, you say? Yes, because the `offs` variable represents the last typed position, which would be at the end of a word. So we set up a loop and continue to move back from our position until we hit a delimiter (defined as "{", "}", "(", ")", or a space). Now we have a word, and we can compare it against our list of keywords using the `Vector` class's `contains()` function.

Well, we're nowhere near being done, but we have the main outline of a class ready, we know the methods we need to override, and we know the key concepts we need to get the job done. In the next article we'll see how this all comes together in a working Document class that we can then plug into a `JTextPane`. ☛

References

1. Topley, K. (1998). *Core Java Foundation Classes*. Prentice Hall PTR. Englewood Cliffs, NJ: Prentice-Hall.
2. Eckstein, R., Loy, R., and Wood, D. (1998). *Java Swing*. O'Reilly and Associates.

▼▼ FULL CODE LISTING BELOW ▼▼

About the Author

Jim Crafton is a staff consultant with Computer Sciences Corporation, where he specializes in object-oriented development. He also develops advanced graphics software for Windows and the BeOS. He can be reached at ddiego@one.net and has a Web site at <http://w3.one.net/~ddiego/>.



ddiego@one.net

Listing 1:

```
import com.sun.java.swing.*;
import com.sun.java.swing.text.*;
import java.util.*;
import java.awt.*;

public class CodeDocument extends DefaultStyledDocument{
    private String word = "";
    private SimpleAttributeSet bold = new SimpleAttributeSet();
    private SimpleAttributeSet normal = new SimpleAttributeSet();
    private int currentPos = 0;
    private Vector keywords = new Vector();

    public CodeDocument() {
        //set the bold attribute
        StyleConstants.setBold(bold, true);
    }
}
```

```
private void checkForKeyword(){
    int offs = this.currentPos;
    Element element = this.getParagraphElement(offs);
    String elementText = "";
    try{
        //this gets our chunk of current text for the element we're on
        elementText = this.getText(element.getStartOffset(),
            element.getEndOffset() - element.getStartOffset());
    }
    catch(Exception ex){
        //whoops!
        System.out.println("no text");
    }
    int strLen = elementText.length();
    if (strLen == 0) {return;}
}
```

cyrus intersoft, inc.

www.cyrusintersoft.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

```

int i = 0;
if (element.getStartOffset() > 0){
// translates backward if necessary
offs = offs - element.getStartOffset();
}
if ((offs >= 0) && (offs <= strLen-1)){
i = offs;
while (i >0){
// the while loop walks back until we hit a delimit er
i--;
char charAt = elementText.charAt(i);
if ((charAt == ' ') | (i == 0) | (charAt == '(') |
(charAt == ')') |
(charAt == '{') | (charAt == '}')){
// if i == 0 then we're at the begini nng
if(i != 0){
i++;
}
word = elementText.substring(i, offs); //skip the period
String s = word.trim().toLowerCase();
// this is what actually checks for a matching keyword
if (keywords.contains(s)){
try{
// remove the old word and formatting
this.remove(currentPos - word.length(), word.length());

// replace it with the same word, but new formatting
// we MUST call the super class insertString method here,
// otherwise we *would end up in an infinite loop !!!!
super.insertString(currentPos - word.length(), word, bold);
}
catch (Exception ex){
ex.printStackTrace();
}
}
break;
}
}
}

```

```

}
}
public void insertString(int offs,
String str,
AttributeSet a) throws BadLocati onExcepti on{
if (offs < 0){
return;
}
currentPos = offs;
char strChar;
super.insertString(offs, str, normal);
int strLen = str.length();
if (strLen > 1){
}
else{
strChar = str.charAt(0);
swi tch (strChar){
case ('{'):case ('}'):case (' ': case('\n'):
case ('('):case (')'):case (';'):case ('.'){
checkForKeyword();
wordStart = offs;
}
break;
} //end swi tch
}
}
public Vector getKeywords(){
return thi s.keywords;
}
public void setKeywords(Vector aKeywordLi st){
if (akeywordLi st != null){
thi s.keywords = akeywordLi st;
}
}
}

```


CODE LISTING


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

FINDaHOST.com

www.findahost.com

See JDJ Special Offer at:

<http://www.sys-con.com/java/specialsoftheweek.html>

SL Corporation

www.sl.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



Steve Henning
Vice President, Marketing

with Steve Henning of Blue Lobster Software

JDJ: Can you give us a little background on Blue Lobster and then jump into Stingray and Mako?

Henning: Blue Lobster was started to help people access and leverage data that is resident on their mainframes in the Web world. Where did we

get that name? Blue Lobster's founders were from IBM, Big Blue...the Lobster is a visual metaphor for the mainframe...both the mainframe and the lobster have a forbidding outer shell yet there is tasty data in the middle. Blue Lobster is different from traditional Web and host solutions in that we are really concentrated on helping people build e-commerce and online customer support applications.

Stingray handles terminal-based applications on the mainframe that you can get only through the 32/70 presentation layer. Stingray produces Java objects that encapsulate screen navigation, and getting and setting of data with the mainframe applications, and so you get reusable objects that can be hosted on the server side. And they effectively separate the business logic from the presentation logic for use in applications in e-commerce.

Mako handles different types of applications on the mainframe. Some applications actually allow you to go directly to

the transaction rather than through the presentation layer of the application. If you look at a typical CICS application on the mainframe, it has a 32/70 presentation layer, but it's integrated with a number of underlying CICS transactions that can be accessed directly through an API like EXCI or ECI. These applications are really hard to Web-enable because traditionally the applications themselves, the transactions, and the data input and output format are written in COBOL. What Mako does is give developers a COBOL expert system that automatically maps transactions and the associated input and output data, which is typically stored in COBOL paper books, directly to a Java representation. We also have a server component that can be used at runtime that will allow the mapping to take place between COBOL and Java. All our products are built as 100% Pure Java and produce JavaBeans that make it really quick to get your applications out on the Web.

JDJ: Do Mako and Stingray work together?

Henning: You can use the objects that come from Mako and Stingray together. Typically, if you look at an e-commerce application, you're going to have people who want to cull from many mainframe data sources so you might have a couple of Stingray objects that are encapsulating transactions with a terminal base application and then you might also use the Mako product to encapsulate some direct access to CICS transactions. You write some new business logic on the serv-

er to build the commerce application that uses this data. You might even incorporate some middle-tier data sources like an SQL server database. So they work very well together.

JDJ: Can you give us a real-time example of how it is being used and maybe tell us who is using it?

Henning: Davis Vision built an online e-commerce application for their subscribers and member doctors to purchase glasses, contact lenses, etc. They wanted to streamline that process and put it on the Web so they could give better service. They incorporated Stingray to record transactions with the CICS applications on their mainframe. When they did that they got Java Legacy business objects that would represent these transactions and put them through a Java registration process to create COM objects. When their members access this application, they are given an active server page through a Microsoft Web server that allows them to get to the main business logic of the application. Requests for services and orders for glasses are stored and forwarded to the Stingray-generated Legacy business objects and then they interact with the mainframe application directly, storing the data onto the mainframe databases and the results are passed back through Stingray and served up as straight HTML to the consumers of the services. It has been a really successful architecture for them, and it could only have been built with a product like Stingray that supports server-side applications.

Instantiations

www.instantiations.com

See JDJ Special Offer at:

<http://www.sys-con.com/java/specialsoftheweek.html>

Tidestone Technology

Tidestone is offering a FULL trial version of "Formula One" their award-winning Pure Java spreadsheet software

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Micro

msdn.microsoft.com

See JDJ Special
<http://www.sys-con.com/java>

rosoft

ft.com/visualc

cial Offer at:
va/specialsoftheweek.html



Multiplatform Application Design with Java

An architecture for multiplatform development

by Jim Wright

Multiplatform code isn't a new occurrence or concept in software development. C and even C++ are cross-platform languages if you only use the standard libraries and refrain from using the platform-specific options offered by your compiler vendor. A recompile is required, but the source code can be made to work without modification. Other languages are also available across platforms, including scripting languages like REXX, PERL and Python; they allow us to skip the recompiled, generalized languages like BASIC and COBOL, and other small-following or specialized languages like Forth. So what does Java offer that we don't get from these other tools? Furthermore, what problems do we, as multiplatform developers, still need to solve? This article addresses these questions and presents a simple framework for developing multiplatform applications in Java.

To understand what Java gives us, we must first ask the question: "What are the traditional problems with writing cross-platform code using C or other languages?" I'll divide these issues into a few broad categories. The first and most obvious (at least for people with desktop PC backgrounds) is the user interface. Along with the workspace GUI issues, I'll also lump in other less visible "interfaces," often those between the application and the system that, while not presented directly to the end user, often differ across platforms. Some possible examples of this might be text formats (ASCII versus EBCDIC), permissions models (file or process permissions, etc.) or security models. Sometimes it's difficult to spot the assumptions you've made about all your application's interfaces to the user and the system.

The next problem is the availability of compiler implementations and third-party libraries used to supplement the functionality of the standard libraries, with features like database support and additional ADT implementations.

Finally, there is testing, the bane of multiplatform development. The testing effort for standalone applications on different operating systems is fairly linear. However, when you add distributed application models, the size of the test matrix grows geometrically, as do the number of possible issues related to interoperability, latency and other factors.

Today: Multiplatform Development in Java vs Other Platforms

So how does Java assist with these obstacles? It certainly helps that the UI offers a uniform event-based GUI model with which we can develop applications having a cross-platform consistency. While there have been some complaints about the performance and/or look and feel of Java apps relative to those with platform-native interfaces, we've begun to see applications using the newest Java UI technology, with outstanding results.

Regarding compiler availability, Java delivers a standard compiler implementation that's available across most common development platforms. Plus, Java has recently introduced access to the source for creating modifications and implementations on new platforms. Differing C++ implementations have haunted developers for years. Even differing implementations on the same platform have caused problems. (Witness the VC++/Borland C++ incompatible implementations that persisted on Windows.) The consistency of Java compilers also helps significantly with the use of third-party libraries, ensuring that the chosen library will at least build successfully on your platform(s).

That's the good news.

What about the areas where Java doesn't help - and may even hurt us? The foremost problem with multiplatform application development in Java is testing. It adds another complexity multiplier to the test matrix on virtual machine releases and, worse yet, multiple implementations of each release on some platforms. Further complexity is layered on by browser VMs (especially if parts of the application can run inside the browser and others outside) and browsers with plug-in VMs. With layer upon layer of multiplicative complexity from VM variations, the time required for testing can more than double on Java projects compared to the time using more traditional development tools. Some development efforts have seen testing resources jump from 20 to 50% and higher in the transition to Java. The only real solution is more compatible VM implementations. We can only hope that Sun, IBM and Microsoft hear our pleas: "Make the virtual machines more compatible!" Fortunately, progress is being made on

this front, albeit more slowly than we'd like.

Putting aside the testing/VM compatibility issue, one primary obstacle remains: the system interfaces for which Java doesn't directly provide an abstraction layer. While it provides strong assistance with application GUIs via AWT and/or JFC, Java doesn't provide packaged solutions to many system interface problems. More are being tackled every day in new APIs and through Sun's new Java Extensions model, e.g., the Java Communication API for serial communication or the Java Cryptography Extension. This is a slow process, though, and there will always be interfaces that aren't covered. The solution here is simple (and certainly not new), although seldom well executed - provide the abstraction layer for all nontrivial interfaces to the system that supports pluggable platform-specific implementations.

Platform Pack Solution

At InstallShield our products have to interface to the system using virtually every platform-specific interface in the spectrum, solving many of the problems that aren't directly addressed by Java. The installer has to support almost everything, because it must be able to set up a wide range of applications, from a simple text editor to an application server. In aggregate, the set of applications using our installer utilizes virtually every platform-specific service - system-event logging, security and permissions, the file system and on and on ad infinitum.

Some may ask why we should be using platform-specific implementations rather than simply providing cross-platform Java implementations. First of all, it isn't always possible. For example, the creation of icons in desktop services: though it's often essential for rendering an application usable, Java doesn't directly support it, and there's no cross-platform solution.

Second, the use of native services is usually necessary for applications to interface successfully to the existing IT environment. Many new applications must work alongside existing native code, and to do so they often use the same system services as the older packages. Additionally, it's a great benefit for the system administrator to be able to administer applications using the standard tools for the platform. To accomplish this, the application must interface directly to those native tools. Of course, while it's desirable to leverage platform-specific features where possible, we also need to support platforms for which we haven't created platform-specific code, or for which any partic-

Object International

www.oi.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

ular feature is simply not implemented.

When attempting to solve platform-specific problems, our natural tendency is to simply make it work with the platform we're targeting using JNI (this frequently happens when targeting Windows on the client side). While this handily solves the problem for each platform on a "one-at-a-time" basis, it gives up the "write once, run anywhere" promise of Java.

To fully realize this vision, what's needed is a robust and flexible architecture for implementing a set of platform-specific services in a cross-platform application, while also maintaining gracefully degraded performance on other systems. We call our solution "Platform Packs." For the most part, it follows the same pattern as Java's existing interfaces to platform services - the separation of the definition of functionality in cross-platform Java classes from the implementation of that functionality in platform-specific VMs.

The basic idea of the model is the creation and use of a set of named system services for the implementation of platform-specific items. At application startup, a "Service Manager" examines the system properties to determine the current running platform; then it finds an implementer for each service for that platform and performs any necessary startup for the service implementers, usually launching a service-implementer application, either one for each service or one for all of them.

Why a separate application? First, these platform-specific services are usually in a native executable generated with C/C++ or another native development tool and native libraries. Second, even if the service implementer is a set of Java classes, you may want to run the implementer on a separate machine or in a security context separate from that of the Java VM.

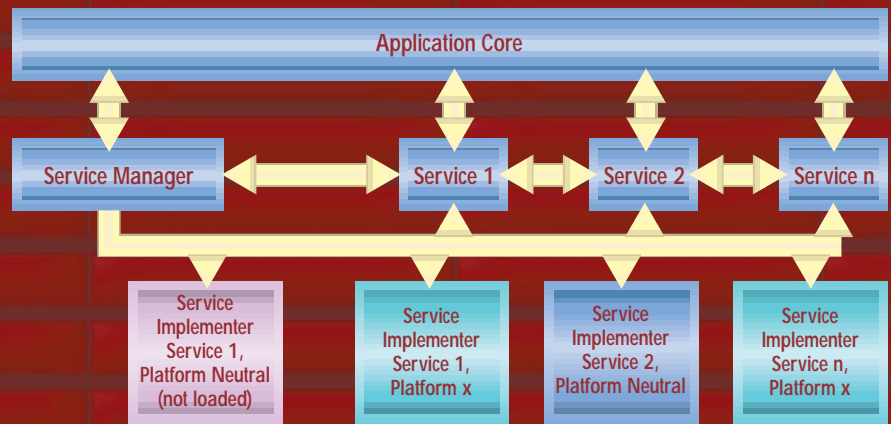
What if a service implementer for a given service doesn't exist for the current platform? The solution is to create a platform-neutral implementation for each service. When an unknown platform is encountered or an implementation of any particular service doesn't exist for the current platform, the platform-neutral service implementers are used. For some services, a platform-neutral implementation won't be available or obvious. However, in these cases the developer simply creates a non-functional shell service and lets the caller know that the service isn't implemented on any calls.

One example of a service requiring this approach is desktop services - creating icons and such. There's no platform-neutral way of doing this (yes, I can hear your screams, CDE fans!), so the platform-neutral desktop service must simply return an `E_SERVICE_NOT_IMPL` error in any call to the service.

The following is a list, by no means all-inclusive, of other services you'll need to think about in your application:

- **Security** (accommodation of the platform and/or application user model)
- **Command shell and environment services**
- **File system access** (this may be less trivial

Action Sequence for an Application Using the Platform Pack Model



1. The application initializes the Service Manager.
2. The Service Manager discovers the platform through the system properties.
3. The Service Manager initializes each service and the service implementers 1..n for the platform. For any service not implemented explicitly for the platform, it initializes the implementer from the Platform Neutral Pack.
4. The Service Manager adds services 1..n to the available services list, and returns from the initialization sequence.
5. The application requests the use of service x.
6. The Service manager hands back the service object for service x.
7. The application requests action y from the service x.
8. The service x calls the service implementer, either for platform x or the platform-neutral implementer (the application doesn't know or care) to perform the requested action. The services may also call each other in performing a given action, requesting other services from the Service Manager using the same method as in the Application Core.
9. The service implementer returns a success/error code, which is then passed back to the application that requested action.

then you think - file permission models differ, text file implementations differ, and some platforms may not even directly support files or paths in the way you're used to on Windows or UNIX)

- **Permissions** (which may or may not need to be separated from the file system or security models depending on the needs of the application)
- **Registry/VPD (Vital Product Data) services** (product install registration, etc.)
- **System event logging**

Conclusion

The core of this strategy is simply to implement as much as possible in Java, including a pure Java implementation of each service area to form a platform-neutral pack. Additional platform-specific functionality can then be implemented in platform-specific native code on a service-by-service basis and activated at application startup by the service manager. Keep in mind that you may not require all services to be implemented for each pack. It's essential that the implementer of the Platform Pack for each environment be able to pick and choose when to use the platform-neutral implementation and when to use native implementation. An example from our product is the registry, which uses the platform-neutral implementation on Solaris and the native implementation on Windows, which maps directly to the Windows registry.

This allows use of the product across vir-

tually all platforms, but with gracefully degraded functionality on those platforms for which no native functionality Platform Pack exists. Additionally, if your service manager allows the detection and creation of the set of available services at runtime, new applications built on the existing framework will be able to request and utilize new services without requiring modification to the service manager or any existing services.

I hope this discussion has provided a useful look at a successful strategy for the implementation of multiplatform applications in Java. Java clearly solves some of the biggest problems with development for heterogeneous systems, assisting us with consistent compilers, class libraries, some system interfaces and UI. While it doesn't solve all of our problems, a little forethought in the design of flexible system interfaces can greatly ease the burden of transitioning applications to new systems. 🍌

About the Author

Jim Wright lives in Santa Cruz and serves as the general manager of Java product development and director of developer relations at InstallShield Software Corporation. A consultant before joining InstallShield, Jim has worked in the areas of Windows application development, software installation, networked and widely distributed applications, and encryption systems. Jim can be reached at jamesw@installshield.com.



jamesw@installshield.com

Borland

www.borland.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



First Look at PointBase, Inc.

A database that can be packaged with data and application logic and distributed over the Net

by Scott Davison

In early 1998, Bruce Scott, one of the cofounders of both Oracle Corporation and Gupta Technology, opened the doors of his fourth start-up company. With this one, PointBase, Scott returned to his database roots with a simple but ambitious business plan: develop the next generation of database software for managing data anywhere on the Net. Specifically, the database should serve applications at three levels in the Net and coexist optimally with major enterprise-class data management systems such as Oracle, Sybase and DB2. Targeted were the *server level*, typically hosting e-commerce applications; the *client level*, with applications for mobile users; and the *Internet appliance level*, for devices such as Web-based PDAs (personal digital assistants) and set-top boxes.

A little over a year later, PointBase is well on its way to realizing that plan with the release of version 2.1 of their Server and Mobile Editions in May and their recent decision to make free demo versions available from their Web site at www.pointbase.com.

PointBase's database products are built with 100% Pure Java to take advantage of Java's open platform architecture and Internet readiness. Supporting Java and CORBA/DCOM standards, PointBase has "drop in" compatibility with all major Java application development environments, application servers and e-commerce servers. Its self-management capabilities ensure reliable operations, a requirement of software that will be distributed across the Net.

With its new products, PointBase is putting unprecedented data-processing power in a small, portable package - exactly what is needed by the company's targeted applications. PointBase offers extremely low-cost ownership through innovations in data integration, ease of use, extensibility and adherence to industry standards. Because of PointBase's compatibility with corporate databases and seamless data synchronization capabilities, PointBase can extend a central repository of data out to millions of mobile workers via dial-up, network and wireless cellular connections over the Net. And because of its exceptionally small footprint (requiring as little as 270 KB of

RAM), PointBase is particularly useful for applications such as Internet catalogs, which are structured to deliver fully integrated packages of data and data management capabilities in one download.

As noted, PointBase is led by founder, president and chief executive Bruce Scott, a pioneer in the database industry and a leader in the area of enterprise and embedded database architecture and product development. Indeed, Scott "wrote the book" on the first iterations of SQL, which has become the industry standard for powerful database applications. Along with Larry Ellison, Bob Miner and Ed Oates, Bruce cofounded Oracle in 1997 and authored more than half of Oracle's first-generation products.

After Oracle, Scott cofounded Gupta Technology (later renamed Centura Software), where he came up with the notion of the small-footprint workgroup server for the Intel-based platform, and delivered to the marketplace Gupta's SQLBase - the first database to target the needs of workgroups and mobile database applications.

During the first few months of PointBase's operation, Scott garnered industrywide recognition for its ambitious goals. Scott is joined by cofounder Jeff Richey, who has a track record of developing world-class databases, including development work on Oracle, DB/2 and Sybase. Shortly after Richey was introduced to Scott's plan, he left IBM, where he was slated to manage a major overhaul of DB/2, to join Scott at PointBase.

In November 1998, PointBase received *Red Herring* magazine's "Catch of the Season Award." The honor recognizes PointBase for its "experienced management team, strong focus on a viable long-term market and excellent Java-based technology." In the March 1999 edition, *Red Herring* featured PointBase in its regular "One to Watch" feature, again for its innovative approach in the database market.

Managing Data Anywhere on the Net

The arrival of the Internet has led to the emergence of a dramatically new networked

computing environment. Unlike the closed architectures of the past, networks today can use the Internet's open standards to link different levels of servers, workstations, devices and back-end systems.

Today's databases, however, have yet to leverage the potential of the Internet to extend the power of distributed data processing to new levels. Client/server systems have made the transition to the Internet, but they still rely on closed technology and legacy code, drawbacks that limit their effectiveness. And the databases so far available for the Internet lack the open architecture, platform independence and standards support to fully enable distributed computing.

Scott remembers when porting relational databases onto every platform was a nightmare of epic proportions - until Java surfaced. "Now those issues are receding..." PointBase takes full advantage of the distributive power of the Internet and the cross-platform independence of Java to provide *anonymous deployment*, a term PointBase has trademarked to describe the feature of being able to download their database onto any platform and have it run flawlessly as long as Java is supported on the target machine.

"It wasn't that long ago that naysayers claimed it was impossible to create a set of standards and write software that could be used on any platform," says Scott. "Well, we're here. We're now in a position to deliver data processing power to all levels of the organization over the Internet."

Applications Across the Board and Across the Net

E-commerce systems are a phenomenal growth area, but databases haven't kept pace with their functionality. To support Web-based e-commerce, a database - just like the e-commerce system it supports - must provide:

1. Seamless integration with a variety of enterprise-class databases
2. Platform independence to enable customers to leverage their existing infrastructure
3. Distributed object computing with support for JDBC
4. Adaptive management for worry-free operations
5. The ability to support local updates to avoid direct exposure of back-end systems to the Web.

Intuitive Systems, Inc

Intuitive Systems, Inc. is offering a FREE download of Optimizeit! 3.0 Professional demo version.

Note: Please type
"JDJ Coupon"
in the comments box.

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

The Object

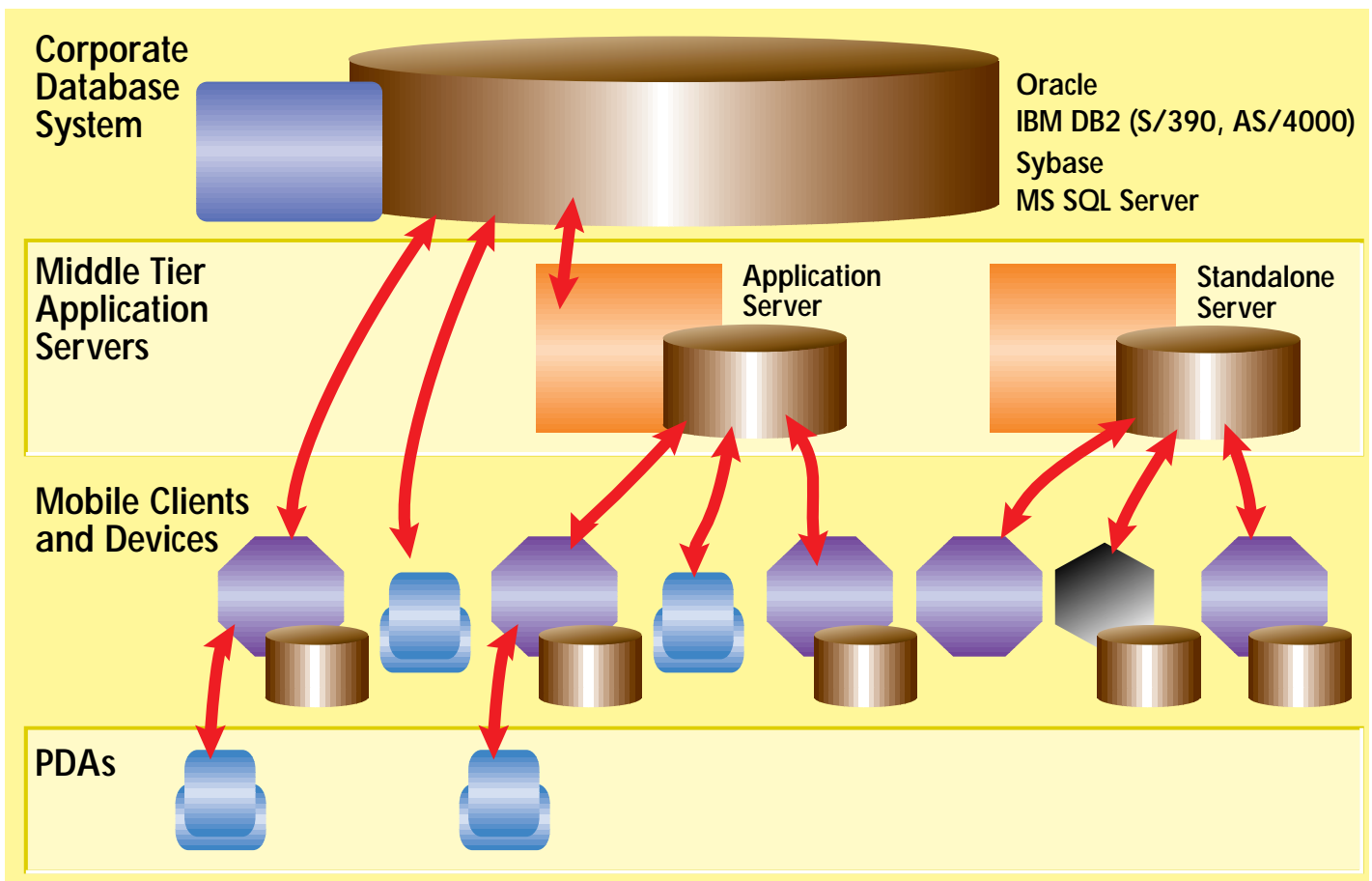
www.object.com

See JDJ Special
<http://www.sys-con.com/java>

ct People

people.com

cial Offer at:
java/specialsoftheweek.html



PointBase is the only multiuser, pure Java database that provides this set of features.

Projections of the number of mobile workers indicate that somewhere between 60 million and 108 million of them will participate in the workforce by the end of 2001. Whatever the actual number turns out to be, one thing is certain: there will be a lot of them. As this increasingly mobile workforce takes their applications on the road, there is a growing disconnect between the data and applications they leave behind in the enterprise and the versions they carry in their laptops. Databases that support mobile applications must deal with these issues, and in addition need to provide SQL compatibility and the ability to reconcile data bidirectionally with the corporate database. Moreover, to be mobile, an application must have a small footprint; however, to be trusted with mission-critical functions, a mobile application must have a level of functionality that complements that of the prime-time enterprise applications. In addition, ease of use and a high degree of automated, worry-free maintenance are must-haves; mobile applications must not require a technical database administrator to ensure continual operation. PointBase meets these requirements.

Internet appliances share many of the same requirements of e-commerce systems and mobile applications – in fact, they are mobile, or remote, computing devices themselves, with limited functionality. To begin with, software for such appliances – Internet-connected PDAs, Web/cable set-top boxes,

even devices such as car navigation systems and telecommunications hubs and switches – requires an exceptionally small footprint. Such applications must be self-managing, provide zero or near-zero administration and be easily deployed on a wide range of platforms. PointBase meets these requirements too.

Product Attributes

PointBase products are certified for Apple, HP-UX, IBM (AS/400, RS6000 AIX and OS/390), Linux, Windows 95/98/NT, Novell NetWare and Sun Solaris. They support data replication to and from Oracle, Sybase, IBM DB/2 (AS/400, OS/390, UDB), Microsoft SQL Server and Lotus Notes. DataMirror, an industry leader in data replication, data extraction and data transformation, provides the critical components for the PointBase heterogeneous replication technology. This compatibility means customers can leverage their investments in existing technology, and can painlessly scale their database solutions with the growth of their business.

PointBase products also support the CORBA/DCOM standards for distributed object computing, the JDBC standard for Java database connectivity and Internet standards including FTP and HTTP. And PointBase products are SQL compliant, with native support for DB/2 and Oracle SQL statement sets.

PointBase products can be extended easily on Java servers. In the new world of Java development, application developers expect

to extend, customize and specialize products they use through the magic of Java object-oriented programming. PointBase meets this expectation by providing a fully extensible database to allow for customer-oriented customization. Gone are the days of one-size-fits-all databases.

PointBase Server Edition is the first multiuser Java database available in the market. It is well suited to e-commerce applications and Java servers. In addition to the features above, it provides:

- **Multiuser concurrent use:** Supports unlimited users with concurrency management and row-level locking.
- **Multiuser security:** Provides server-based multiuser security and an open naming and directory framework. Encryption is fully supported.

PointBase Mobile Edition is a multithreaded single-user database designed for mobile clients and Web-based appliances. With its small footprint, the Mobile Edition doesn't take up a lot of space on the hard disk, as little as 270 KB of RAM – next to nothing when compared to the competition. ☛

About the Author

Scott Davison is one of the founding editors of SYS-CON Publications, Inc., the publisher of Java Developer's Journal. Scott can be reached at scott@sys-con.com.



scott@sys-con.com

Developmentor

www.developmentor.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



Sybase PowerJ 3.0

by Sybase

Steep learning curve and price notwithstanding, this is a product to consider



by Sean Rhody

Web-development of every kind is one focus of Sybase PowerJ 3.0 (see Figure 1). In addition to a complete Java development environment, PowerJ comes with a set of tools that differentiates it from other Java IDEs on the market. These tools are PowerDynamo, a Web site hosting tool that allows you to drive a Web site from a database; PowerSite, a Web site management tool; Adaptive Server Anywhere, a small-footprint relational database; ObjectCycle, a source code control package; and Enterprise Application Server. Sybase is also lowering the price of PowerJ, placing it in the \$600-\$800 range. Given the tools that come in the package, that's not an unreasonable price.

I'll begin my review by focusing on PowerJ itself. This is the third version of PowerJ, and it's had several years to be polished. The IDE features color-syntax highlighting, background compilation and drag-and-drop coding. It also provides a large number of helpful features including the ability to remember common settings (called *profiles*) so they can be reused. Figure 2 shows the transaction properties. By selecting a previously stored profile, you can load all your common JDBC settings. What's also nice about PowerJ is that the designers have realized that the connections and data can come from one source at design time and another at runtime. For example, when you create a component for Enterprise Application Server (EAS) you'll typically use a connection cache from the server, but when you're testing you'll use a local JDBC connection. Not only is this provided for, but PowerJ can even look up the connection caches inside EAS for you so you don't have to worry about typing errors.

Select a component from the palette and drop it on the screen to place a new instance. Then drag that component into the code window and the properties window opens, allowing you to select the method or attribute you wish to work with. This feature isn't limited to items on your screen – you can drag objects from a variety of views onto the code window. Experienced coders will probably ignore this for the most part, but it's good for new developers, or for finding

that odd property or method you just can't remember.

PowerJ 3.0 also provides the Java version of one of Sybase's crown jewels – the DataWindow. Programmers familiar with PowerBuilder know that this control kept PowerBuilder in the market and competitive with Visual Basic even though VB costs thousands less.

It's hard to describe the DataWindow – and its nonvisual brother, the DataStore – to people who haven't used it. That's because it can be so many different things. It can be used to create complex reports for display, to create a grid control, to design the entire input portion of a screen and to easily build and work with drop-down data. Plus it can enforce business rules.

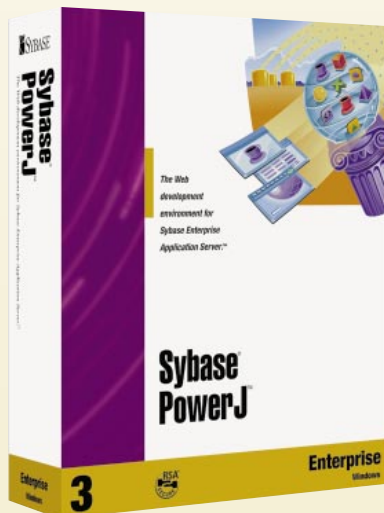
The DataWindow also offers an intriguing option for application partitioning. One of the difficulties in designing any partitioned application is figuring out how to transfer data from the screen to the server. Consider an order-entry screen for some direct marketing company: the typical user may create multiple line items for a single order, then send them back to the server to be processed. There may be rules associated with the items, such as a discount for purchases exceeding a certain amount.

The DataWindow provides a solution for packaging this data by allowing you to get the state of the data from the control. This state can be passed into the server, or obtained from the server, and synchronized. Additionally, after the data has been initially obtained from the server (revising the order, for example), only the

changes need to be sent back. The DataWindow supports the ability to obtain only the changed rows and return them to the server.

Another interesting feature of the DataWindow is the ability to transmit not just the data but the presentation itself to the client. This allows the server to swap views as needed, perhaps in response to security requirements.

The DataWindow is also available in an HTML version that can be used in PowerDynamo. This version converts the DataWindow within the PowerDynamo environment to a JSP that allows the same or nearly the same presentation variety and enforcement as the Java version. And you can



Sybase PowerJ 3.0

Sybase, Inc.

6475 Christie Ave.

Emeryville, CA 94608-1050

Phone: 800 8-SYBASE

Fax: 978 369-5071

www.sybase.com

Sybase PowerJ 3.0 is available for \$595.

reuse the DataWindow definitions that you create between PowerBuilder, PowerJ, PowerDynamo and even Sybase's Power++, which is a C++ development environment.

Developing a DataWindow definition is easy. PowerJ includes a DataWindow Builder utility that provides wizards to walk you through the steps. Developers familiar with PowerBuilder will realize that the DataWindow Builder is in reality a featured limited version of PowerBuilder.

I started with the Employee table from the sample database, then selected a free-form presentation style and was on my way. To make it more useful, I'd edit some of the fields, turning things like the Department ID into a drop-down list box. I'd also probably hide the employee ID field and make it a sequence in the database, turn the sex choice into radio buttons for male and female, and put min and max limits on the salary. After that, this definition would be ready to use. I'd create a screen to use this component, and a server-side object that would process the input using a DataStore.

A DataStore is the same as a DataWindow except that it doesn't display at runtime. Obviously this is useful in a server component, but you can have a DataStore on the client side and then drive multiple views from a single set of data.

PowerJ provides the ability to create what Sybase calls *Visual Classes*, which perversely enough are not visual at runtime. This ability allows you to do visual drag-and-drop programming on a class that you would normally edit completely in a code window. Sometimes this is useful, but many developers will still prefer to work completely in the code window.

The development environment of PowerJ is also geared toward team development and deployment of code, not just coding itself. ObjectCycle is included as a source code control solution. One advantage of this product is that every Sybase development tool works with ObjectCycle, and it's free. It's also not a bad control product. If you wish to use some other tool, such as PVCS, that's also supported. This in itself isn't really novel – JBuilder provides PVCS, for example. But the deployment options in PowerJ, and in its accompanying tools, are unique. By default there are a number of options, and you can configure more yourself. These options include deploying Java code to a Sybase database to run

KL Group

www.klgroup.com/truth

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

inside the database, to PowerDynamo for use in a Web site, to an EAS server, to a file system or to a Web site. Because of the way PowerJ organizes its source code into a hierarchy of Targets, you can choose to deploy all dependencies automatically as well, making it easy to ensure that the entire distribution gets where it's supposed to go.

As stated earlier, PowerJ ships with a wealth of accompanying tools. Adaptive Server Anywhere (ASA) is a single-user version of Sybase's departmental server. This is no toy database. Besides supporting two SQL syntaxes, the original Watcom SQL and Sybase's Transact SQL, ASA supports database replication and the use of Java. ASA is also extremely easy to administer, making it an ideal development tool.

PowerDynamo is a Web server that allows you to dynamically drive the content from your database in a simple, easy-to-develop fashion. I didn't dive into PowerDynamo deeply, but the demonstration I got from Sybase was impressive and fast.

PowerSite is a Web site management tool and HTML authoring program. While not as polished as FrontPage, PowerSite is more flexible when used in a team environment. PowerSite's HTML editor is easy to use, straightforward and supports all server-side models. Once again Sybase has loaded this tool with drag-and-drop ability. PowerSite also features its own dialect for creating pages, one that allows the pages to be translated to ASP, JSP or PowerDynamo so they can be deployed in IIS, Netscape or PowerDynamo servers. If you're unsure where you'll be served up from, this is a good way to keep your options open, but it'll probably be easier to write to the model you'll be working with. PowerSite also makes this easy by placing all of the models on the drag-and-drop palette.

The most impressive add-on to PowerJ is Enterprise Application Server 3.0 (EAS). Under its old name of Jaguar, EAS was one of the first offerings in the application server arena, and offers the broadest support for development, including COM, CORBA, EJB and Native PowerBuilder. PowerJ comes with a development license – to deploy an application you'll need to purchase additional licenses. EAS was one of our Editor's Choice winners this year, based largely on the solid Java support and the ability to integrate into just about any distributed coding environment. EAS runs on NT or Solaris, and will also be available on AIX and HP-UX at a later date. Unfortunately, the EJB support is only at the 0.4 spec level, but Sybase plans to bring this up to 1.0 compliance in a point release before the end of the year.

I tested PowerJ with EAS and was impressed by the integration provided. When you design screens you can drag EAS connections right onto them, and the setup wizards make it simple to establish the necessary connections. When you create a component for use within EAS, PowerJ creates all the necessary methods for you and then provides a fairly easy way to add attributes and methods. Unfortunately, this is driven from a menu rather than a toolbar, so there are annoying extra steps needed to perform this function. You also have to add all methods this way, or your interface class will be out of sync with your implementation class. (You can always edit the interface class directly, so this isn't a large problem.)

One of the most valuable integration features of PowerJ is its multithreaded remote debugging. When debugging a multithreaded application, you can select the thread you wish to monitor and then control the state of all the other threads.

Possibly the most exciting feature, though, is debugging server code remotely. You can pick a component within EAS (you must be running the debug version of EAS, which is provided), set breakpoints and watch points, and wait for it

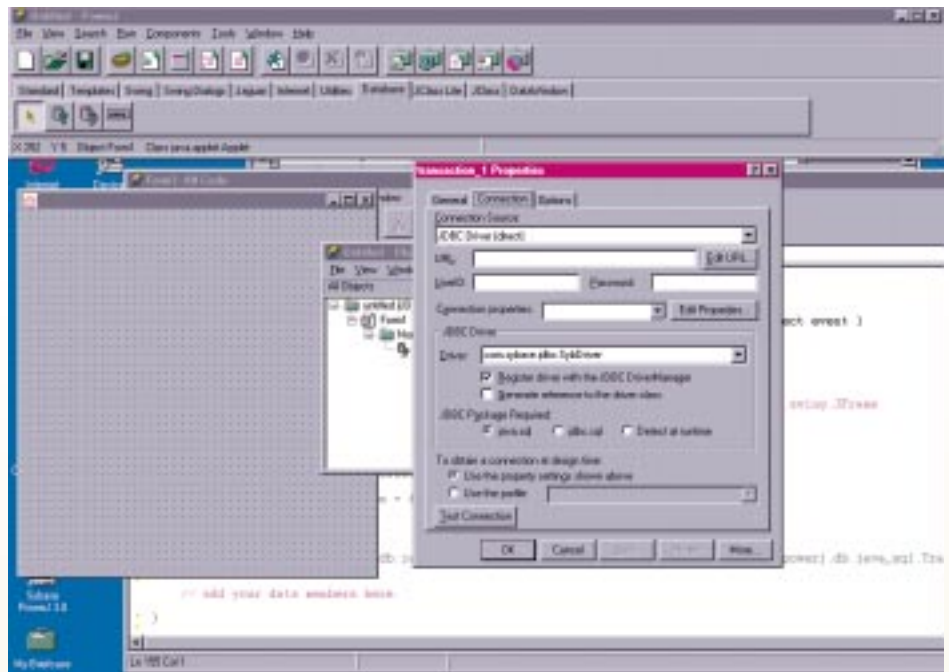


Figure 1: Pure Java implementation of the Web DataWindow

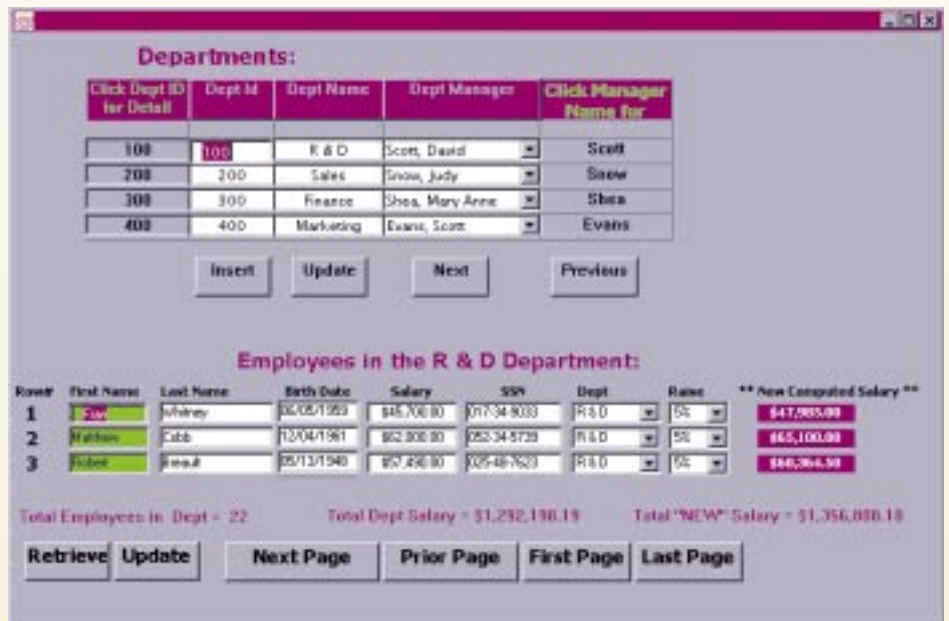


Figure 2: Transaction properties

to be executed. I can't tell you how valuable this is in getting server code debugged. And you can do it from anywhere as long as you can make a TCP/IP connection to the server machine.

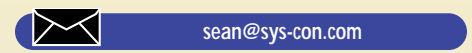
There are some areas for improvement in PowerJ. This tool was originally aimed at corporate developers, and it shows that focus. It's harder to learn than tools like JBuilder. Part of that is due to the increased functionality it provides, but also there's just too much of a learning curve. To really use the tool you need to learn about targets, projects, objects, classes and deployments. It'll take the average developer longer to come up to speed on PowerJ. In general, I think it'll be worth it if you have a need for anything besides basic development.

Pricing is still somewhat higher than other solutions, although with the tools included the pricing is more than reasonable. Sybase has also provided a Java-only learning

edition that's downloadable from their Web site. It's not known whether they will continue this when version 3.0 is released. Still, with the price now well under \$1,000, if you have any need for more than just a straight Java IDE, this is definitely a product to consider. And if you have a shop that needs to leverage previous investments in PowerBuilder expertise, this is the tool for you. ☛

About the Author

Sean Rhody is the editor-in-chief of **Java Developer's Journal**. He is also a senior consultant with Computer Sciences Corporation, where he specializes in application architecture – particularly distributed systems. He can be reached by e-mail at sean@sys-con.com.



Flashline.com

Flashline.com is offering 30% off JavaBeans to JDJ readers who use the coupon code "jdj"

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

CASE STUDY

by Ethan Henry & Josephine Coombe

Information Architects Corporation Benefits from JProbe During Development of Robust, Scalable Application

During the development of Metaphoria DTS, Information Architects Corporation uses JProbe Profiler with JProbe Memory Debugger from KL Group to eliminate a memory leak and dramatically improve the application's performance and scalability

Managing information in the enterprise environment is no small task. These days companies are looking to robust and scalable content-publishing technologies to manage and distribute information as quickly and effectively as possible. A number of challenges have to be overcome first, however. Data is often scattered across and outside the enterprise itself – in multiple repositories and multiple formats. Additionally, companies today quite simply expect more from their data: many want information to be customized to the user's requirements and delivered on a dynamic, as-needed basis.

Developing the Enterprise Information Portal

Enter Information Architects Corporation, a worldwide developer of the latest breed of data management solutions: Enterprise Information Portals – online information centers that feed dynamic, customized data to individual users.

Information Architects (IA), headquartered in Charlotte, North Carolina, provides a full complement of products and services that answer such next-generation information requirements. Catering to large enterprises around the globe, the company offers management consulting, design, development and deployment of virtual information portal solutions. Customers benefit from IA's expertise in transforming existing information systems architecture to support a heterogeneous, scalable, flexible and ubiquitous Internet/intranet architecture. To help accomplish this goal, IA develops Internet software – specifically, middleware applications.

One of IA's flagship products is Metaphoria DTS, a robust and highly scalable Web application framework that provides access to multiple back-end data sources, and presents the data in a unified, Web-friendly format. This ability is critical to financial companies, for example, which have large quantities of data issuing from diverse sources. Data may originate from any number of locations – including external Web sites, databases, FTP files or local files – and can then be put online. Metaphoria DTS contains components that understand these various protocols and can plug in and immediately use any new protocols that arrive on the enterprise scene without forcing developers to rewrite their exist-

ing applications. Powerful data analysis and display capabilities make moving and transforming enterprise data fast and seamless.

When sensitive data must be pulled together, IA's product is an ideal content-publishing mechanism, and consequently, often used for intranets. The new breed of intranet, the information portal, can deliver all manner of content to individual members of an organization. For example, a user could have ready access to customized personnel information such as remaining vacation days.

To ensure that applications are robust and reliable, IA makes memory debugging and per-



formance optimization a standard part of its software development cycle. IA believes that every developer on the team should test regularly for memory leaks and performance bottlenecks. Finding these problems early keeps them from piling up at the end of a project when time is limited. While developing Metaphoria DTS, IA used JProbe Profiler and JProbe Memory Debugger from KL Group. JProbe helped them clear up memory leaks quickly and improve performance.

IA's Metaphoria DTS is delivering significant benefits to its customers. Not only is their data being transformed efficiently and quickly, companies using Metaphoria DTS are avoiding the often staggering financial and morale-related costs associated with major retraining initiatives. Because data format conversion is automated by Metaphoria DTS, employees within IA customer sites can continue using their traditional methods of managing information. In large enterprise environments with massive and varied data, as well as longstanding methods for handling it,



egh@klgroup.com jac@klgroup.com

The Theory Center

Theory Center is offering a 10% off license on Theory Center's JumpStart components for eBusiness with coupon code "jdj"/subscriber

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Metaphoria DTS delivers the best of both worlds: companies can keep existing data management processes intact while simultaneously evolving relatively painlessly to a more up-to-date information management system.

Java Fulfills Its Potential

Metaphoria DTS was prototyped using Jigsaw (the reference Web server from the World Wide Web Consortium), and the first shipping version was built exclusively in Java using servlets. Evan Coyne Maloney, Advance Development Group Leader at IA and a key member of the Metaphoria DTS development team, assigns much of the credit for the application's success to the Java platform itself.

Maloney points out that the platform's portability allowed them to write code that could be used with any Web server, allowing them to avoid building in a custom Web server. The application needed to be compatible with any given client's existing server, and with Java this posed no problem because of the portability provided by the standard Java Servlet API. Any other language would have limited IA's target market for the application significantly, as many companies aren't interested in changing their Web server to accommodate other new services. Because these servlets work with virtually any type of Web server, it was the ideal choice.

"There are still some people out there who are surprised when I tell them that we have a site up that's serving 2 million requests per day," Maloney says. "I just let them know that these days Java is, in fact, the perfect server environment for writing applications."

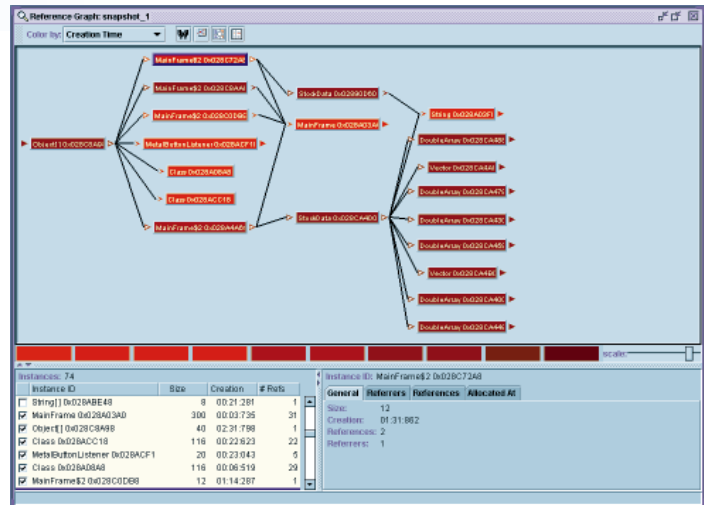
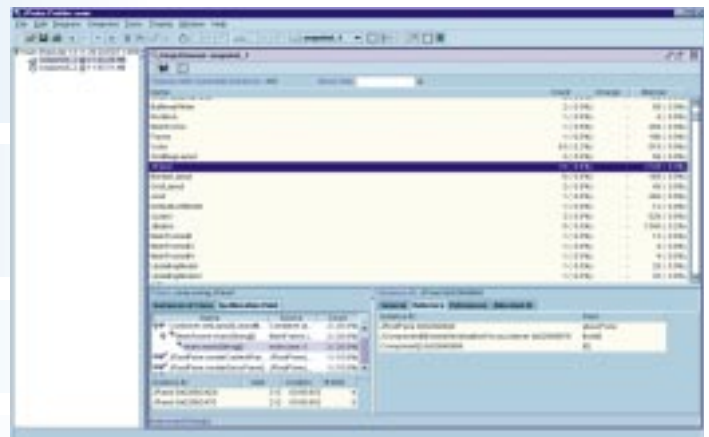
The ability of Java to interact with the native platform environment enhances the features that Metaphoria DTS provides. Although the core product is written entirely in Java, a DTS add-on component allows the product to use COM to extract data directly from Windows applications. This allows DTS to utilize COM interfaces (which don't change) to extract data from files rather than requiring DTS to "know" the file formats (which change frequently). Bypassing the file formats allowed IA to avoid the problem of accommodating changing versions of file types.

JProbe Helps Ensure a Scalable Application

While Java may have been the ideal solution, building and optimizing the application was still challenging. Scalable performance was a key goal. The application went through rigorous stress-testing during the development. Part of this process involved profiling the application with KL Group's JProbe Profiler. IA's developers knew intuitively that they had a performance problem, but hadn't isolated the cause.

IA realized they needed an accurate and highly effective performance tool to help assess the problem, but it took some time to identify the right solution and integrate it into their development. They tried, but subsequently rejected, another competitive profiler because of slow performance, says Maloney. IA finally turned to KL Group's JProbe Profiler, an advanced, highly accurate profiling tool with an intuitive graphical interface. JProbe Memory Debugger, fully integrated with JProbe Profiler, gave Maloney and his team the critical data IA needed on memory allocation.

1/8 Ad



"Literally within five minutes of running JProbe 2.0 with a development version of our product," Maloney recalls, "we found a problem – a memory leak."

JProbe's runtime Memory Usage Graph identified the problem immediately. "As soon as we looked at that graph, we knew we had a memory leak; the plateaus were getting higher and higher right in front of our eyes." By tracking down the cause of the problem using JProbe Memory Debugger's Heap Browser, IA quickly eliminated the offending leak, which was caused by the program's holding object references too long. Specifically, while generating the structure of a Web site, the application was creating many large objects and stuffing them in a container. But object references weren't being released properly – the objects were not getting garbage collected and the memory footprint kept growing. Soon the disk started thrashing and performance degraded noticeably.

Using JProbe, they were able to reduce the memory footprint substantially. The application then ran much faster and was more scalable – an important factor for a server-based application that needs to service millions of requests a day.

For IA, JProbe offered an indisputable return on investment. "Considering how quickly we found a problem, JProbe was certainly well worth the purchase price! We only wished we had tried it sooner," Maloney says. "In fact, we immediately ordered three copies with GSS so we could implement performance profiling and memory debugging regularly and on a team-wide basis. More important, JProbe allowed us to increase the scalability of our application, helping us to achieve one of our primary goals in developing Metaphoria DTS." 🍌

About the Authors

Ethan Henry is KL Group's Java Evangelist and can be reached at egh@klgroup.com.

Josephine Coombe is the head of strategic communications at KL Group. She can be reached at jac@klgroup.com.

Insignia

www.insignia.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

What Will Come

Implementation technology—software development in heterogeneous platforms

by Richard Soley



In the swirl of events around the announcement last year of the upcoming CORBA 3.0, the attention centered on the upcoming CORBA Component Model. While CORBA has existed in some form or another as an adopted technology of the OMG's open, neutral, standards-setting process since October 1991, this year's major revision point for the first time will address a wealth of issues that point at deployment rather than interface. Let me explain.

A decade ago the Object Management Group started down the road to infrastructure and end-user standards for interoperability in heterogeneous enterprise-wide systems. Ever since, the focus of our work has been on declaration of interfaces. Interoperability achieved through declared interfaces, especially through declaration in an interface definition language (such as OMG IDL), is a powerful concept: it supports team-oriented software construction (through clearly spelled-out interface contracts that can be built on many platforms in many languages); it is the basis for real architectural design of systems; and it is critical for long-term system maintainability. On the technical side, it also takes away from the programmer the pain of dealing with interprocess communication and network communication protocols, allowing programmers to concentrate instead on system definition (and therefore the "business logic" that needs to be implemented). All from a simple, unassuming IDL.

In 1995 OMG took a huge step forward when we not only fleshed out better support for portability, but, more important, defined a basic interoperability protocol, the Generic Inter-ORB Protocol (GIOP), for connecting CORBA systems (along with the better-known TCP/IP mapping of GIOP, the Internet Inter-ORB Protocol, or IIOP). It's important to note that most programmers never see GIOP or IIOP; rather, CORBA implementations layer automatically generated stubs, skeletons—in effect, generated protocols—over IIOP to ensure end-to-end interoperability between applications relying on CORBA to perform their interapplication communications.

Since those days the OMG itself has changed character significantly. Today it comprises 800 member companies, nearly half of them end users of information technology (like Citigroup, Boeing, Bellsouth and 3M), or vertical-market independent software vendors (ISVs) rather than vendors (though big names like Microsoft, Sun Microsystems, Hewlett-Packard and Fujitsu are longtime members). The OMG's open, neutral, consensus-making process today has underway about a hundred standards processes. They range from the traditional OMG infrastructure definition and maintenance (CORBA itself, the Unified Modeling Language, services for security and authentication, transactional integrity, event notification, application internationalization and even print service management) to vertical-market (in OMG parlance, "domain") specifications, from manufacturing part definitions and telecommunications network management to life sciences genomic maps and air traffic control systems.

Obviously, the end-user-oriented specification processes have a more immediate and obvious impact on the average end user and vertical-market ISV, who can see more immediately the value of a standardized patient identification interface that will make a hospital merger work more smoothly.

Nevertheless, the CORBA technology space marches on, and for the first time addresses what some might consider an implementation technology—specifically, software deployment in heterogeneous platforms.

In fact, the CORBA 3.0 processes to update CORBA focus on three major areas:

- For the first time, CORBA specifications now address real-time provision of service and minimum-footprint (or embedded) implementation. Both areas are about maintaining interoperability, with some small sacrifice of code portability, in the exceptional circumstances of embedded, real-time, special-purpose processors. In addition, this category of new additions to CORBA features standardized interfaces for specifying required transport quality of service as well as the marshaling engine (the "IIO" engine, in effect), which allows applications to layer on special-purpose communications transports such as messaging platforms.
- A large and important part of CORBA 3.0—and, more important, a group of interfaces that have already been completed—we term *Internet/Java specifications*. These include a standardized firewall specification (for passing IIO communications through TCP/IP firewalls in a standard way). Interestingly, one of these specifications also allows for mapping, in a standardized way, from Java to IDL (rather than just the more common IDL to Java). This allows Java programmers to write only Java (and have their IDL interface definitions automatically generated from their Java code, leveraging the closeness of the CORBA IDL and Java object models). To my mind, the most exciting specifications in this category are the extensions to the CORBA Core that allow for binary portability of CORBA applications written in Java (more exactly, compiled for the Java Virtual Machine). Since the JVM represents a shared executable environment that runs on every kind of computer, CORBA-based applications can be deployed portably for the first time in binary form, a major step forward that can't be accomplished in the usual heterogeneous setting.
- As I mentioned above, CORBA 3.0 also addresses deployment of distributed component software. This includes a large number of pieces and the ability of application programmers to (1) pass objects over the network by value rather than by reference; (2) support multiple interfaces on a single deployed object; and (3) integrate scripting languages ranging from Tcl to REXX to ECMAScript into rapidly deployed combinations of CORBA applications and, especially, a CORBA Component Model that allows declaration, in an extension of the usual CORBA IDL style, of the deployment characteristics (such as transactional integrity, secure access and connection details) of distributed, enterprise-wide clients and servers.

"CORBA specifications now address real-time provision of service and minimum footprint implementation"

The greatest impact of these three components will likely be on the server side. Over the years, a large number of excellent tools have appeared to simplify the construction of software for the desktop, software for the client. Application builders typically start from the viewpoint of the design of a data display or update window and work toward the server with visual components that support front-end application duties. In contrast, the wide availability of deployment technologies such as CORBA Components (as well as Sun's Enterprise JavaBeans, closely aligned with the CORBA Component Model Java mapping) will make graphical server software design possible. Finally, the application service designer and builder—for both the middle tier and the back end—will be able to drag server components from a pallet, connect them graphically, select deployment options such as transactional contexts and security domains, and quickly deploy new business logic.

Which is the point, right? And it will come. ☺

About the Author

Dr. Richard Mark Soley, president and technical director of the Object Management Group, Inc., leads the OMG technology committees. These committees are responsible for producing standards documents, adopting OMG-standard technology and proposing new technologies. He is a member of the Editorial Board of *Java Developer's Journal*.

InetSoft

InetSoft is offering
FREE "StyleReport"

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Clustering the BEA WebLogic Application

Java and EJB no longer pose limitations

by Dean Jacobs

Mission-critical Web-based applications – customer self-service, distribution channel and supply chain management, online trading and banking – must be deployed on a cluster of servers in order to provide scalability and high availability. Scalability means that servers can be dynamically added or removed as needed to meet user demand, and that the overall load of requests is distributed among the servers so that resources remain fully utilized. High availability means that there is no “single point of failure” in either the system or the application, and that requests automatically failover from non-working components to working components. Ideally, clustering should be transparent to applications: externally, the cluster should present a “single-system image.” In addition to simplifying the task of application development, this allows off-the-shelf components to be deployed without modification.

The Java Enterprise APIs are rapidly becoming the primary programming model for Web-based applications. These APIs present two particularly challenging aspects for a clustering solution. First, they require integration with front-end Web servers, a fixed technology that is external to the cluster. Second, they require back-end management of objects, which by their nature have internal state. In contrast, conventional middleware such as TP monitors generally support only stateless RPC-based services. The hard part about managing state is that excessive communication between servers – to replicate objects for availability, for example – can interfere with scalability.

The BEA WebLogic Server provides an integrated implementation of the Java Enterprise APIs. A BEA WebLogic Cluster is a group of WebLogic servers that coordinate their actions to provide scalable, highly available services in a transparent manner. Since the WebLogic Server is written entirely in Java, WebLogic clusters are independent from the underlying hardware and operating system. Thus a WebLogic cluster can be composed of, say, uniprocessor Intel machines running Microsoft NT, large-scale Sun multiprocessors running Solaris, and IBM AS/400s. In contrast,

platform-specific clustering solutions require that every node run the same operating system. Of course, this allows them to use proprietary hardware, such as shared disks, multitailed disks and high-speed interconnects, for communication between servers. As an alternative, WebLogic uses highly optimized protocols based on new commodity technologies such as IP multicast.

This **JDJ** feature article presents an overview of BEA WebLogic Clusters.

Architecture of a BEA WebLogic Cluster

Figure 1 shows a high-level view of the architecture of a WebLogic cluster. BEA WebLogic Server provides software-based clustering to ensure scalability and high availability for Web and Java deployments. WebLogic clustering uniquely supports



transparent replication, load balancing and failover for Web page generation (presentation logic) and Enterprise JavaBeans components (business logic).

The Web server front end supports dynamic construction of HTML pages using Java Servlets, Java HTML and Java Server Pages (JSP). The application-logic back end hosts objects and components using Java Remote Method Invocation (RMI), Enterprise JavaBeans (EJB) and the Java Naming and Directory Interface (JNDI). Other back-end Java Enterprise APIs, such as JDBC and JMS, are clustered using RMI, EJB and JNDI in much the same way as applications. The front and back

ends are made up of rather different components that are clustered independently.

The Web Server Front End

A WebLogic cluster may be positioned behind standard Web servers such as Netscape Enterprise Server or Microsoft Internet Information Server (IIS). HTTP requests from Web clients, such as browsers, may be handled by these Web servers or the WebLogic front end. Requests for dynamically generated pages are proxied from the Web servers to WebLogic Servlet/JHTML/JSP engines in the front end. This is accomplished using Web server proxy plug-ins, e.g., defined according to the Netscape API (NSAPI) or the Microsoft Internet Server API (ISAPI).

The first line of clustering uses “DNS Round Robin” between the Web clients and the Web servers. DNS, the Internet’s Domain Name Service, resolves a Web site’s name to a list of IP addresses for the site’s Web servers. Each time it gets a lookup request, DNS shuffles the list of addresses it returns. A Web client generally contacts the first server on the list provided by DNS. After some timeout period, or if this server fails, the client makes another DNS request and continues with a new server. This provides a simple form of load balancing and failover. It is possible to install more sophisticated IP-level load balancing and failover schemes that, for example, take into account Web server load, remove failed servers from the list returned by DNS and/or ensure that a client session is always handled by the same Web server (modulo failures).

The second line of clustering is for dynamically generated pages and goes between the Web servers and Servlet/JHTML/JSP engines in the front end of the cluster. The Web server proxy plug-ins perform load balancing and failover between the Servlet/JHTML/JSP engines. They use a session-level round-robin algorithm that is weighted by information about server load, which is piggybacked onto HTTP responses. If the WebLogic front end is configured to handle all HTTP requests, so that the standard Web servers are missing, then the situation looks even brighter. Since the load balancing and failover algorithm is part of the server, it uses information about server load that is shared across the cluster as a matter of course. More important, this algorithm

interland

www.interland.net

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

prefers the local Servlet/JHTML/JSP engine, unless the load is very unevenly distributed, so the request never has to leave the address space of the JVM.

When a Web client first contacts a cluster of Web servers, a session is created that lasts until some idle timeout expires. The Java standards include the notion of Servlet Session State, which is automatically retained on the servers during the session. As an example, Servlet Session State might be used to retain the contents of a shopping cart in a retail application. WebLogic clusters provide for highly available Servlet Session State using disk-based or in-memory replication (as described in more detail later).

The Application-Logic Back End

In the back end, a user- or system-level service is clustered by making an instance of an object (RMI) or component (EJB) that provides the service available on several different servers. An unclustered service is invoked by calling methods of a stub, which marshals the arguments and passes them to a particular remote object. A clustered service is invoked by calling methods of a smart stub, which can find the possible instances of the service and switch between them as needed for load balancing and failover. A variety of load balancing and failover algorithms are provided. It is possible to specify the particular algorithm to use with a given service at the time that service is deployed.

The default is a transaction-level round-robin algorithm that attempts to colocate all services invoked within the same transaction. This algorithm takes server load into account only if the stub appears on a server, since load information is expensive to obtain on a client. When the Servlet/JHTML/JSP engine invokes a clustered back-end service, server-side load balancing occurs. A programmed client may invoke a clustered service directly, resulting in client-side load balancing, or it may have the service invoked on its behalf within the cluster.

There are two forms of clustered back-end services: stateless, which are instance-neutral; and stateful, which are instance-specific. These forms are treated quite differently within the cluster.

Stateless Services

A stateless service may not maintain state on behalf of an application, rather like a conventional RPC. It may of course access application state, but only by loading it temporarily into memory from a database, file system or other external medium. The EJB component model provides a natural way of implementing stateless services, namely, stateless session beans. Stateless services can also be implemented as RMI objects, but then it is up to the programmer to abide by this restriction.

The stateless service model has been

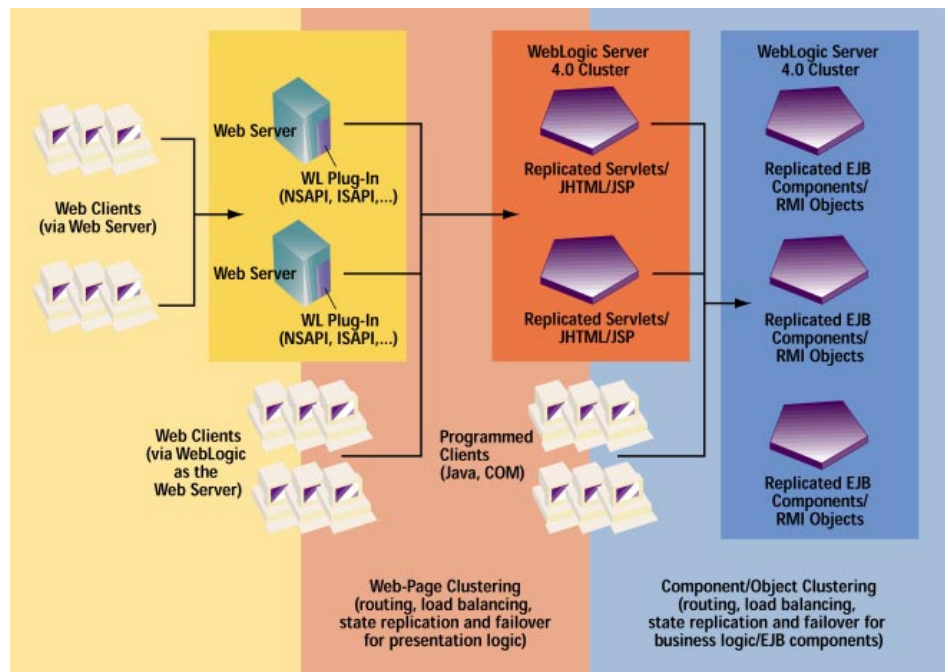


Figure 1: BEA WebLogic cluster architecture

widely advocated because it promotes scalability. There are two reasons for this. First, it obviates the need to back up state in the interests of availability, e.g., by replicating it within the cluster. Second, it allows load balancing to occur on every invocation of the service. This is because the service is “instance-neutral,” that is, it doesn’t matter which instance of the service is invoked.

When a stateless service is deployed in a WebLogic cluster, an instance of the service is created on each server that hosts it. A smart stub obtains references to these instances from the clusterwide naming service and switches between them as needed for load balancing and failover. Retries occur only if it can be guaranteed that a failed operation did not have side effects, e.g., because it never got started, it was transactional and an abort clearly occurred, or it was declared to be idempotent. If such cases do not apply, application code may contain explicit retries, perhaps after undoing side effects. Other than this, clustering is completely transparent to the application.

WebLogic clusters support an important special case of stateless services: service factories that create unclustered stateful service objects. The factory itself is stateless, so its stub can do load balancing and failover in the usual way. The service objects created by the factory are not clustered, however, and may therefore maintain state on behalf of an application. Since this state is not backed up, it will be lost if the object fails. Application code must therefore contain an explicit retry loop that creates a new instance of the object. EJB stateful session beans fit naturally into this model, since they are not persistent. This model may also be used with RMI objects.

Stateful Services

A stateful service may maintain state on behalf of an application. Such a service is “instance-specific” in the sense that each request is intended for a particular instance of the service. In a cluster, the state must be backed up in the interests of availability and can migrate in the interests of load balancing or availability. The cluster must therefore provide some kind of internal activation service that finds or creates service instances. If an instance can be concurrently used by several clients, as is the case for persistent components such as EJB entity beans that are accessed by a global key, then this service must ensure that conflicts do not arise.

One approach to state maintenance is to keep the state in a database or other persistent store. This is particularly suitable for persistent components, but may also be applied to transient objects. This approach scales like stateless services, and in fact differs only in that the latter requires explicit disk reads/writes. The activation service can avoid concurrency conflicts here simply by relying on underlying database locking. In a WebLogic cluster, EJB entity beans always use this approach (see Figure 2).

A related approach is to maintain a write-through cache, which keeps a current copy of the state in memory to avoid subsequent reads. This makes it considerably harder to avoid concurrency conflicts, and doing so can interfere with scalability. Databases are very good at caching objects in memory and doing the minimal disk I/O necessary to provide transactional protection. Application servers may not do much better for persistent components, and so such caching may be best applied to transient objects that are used by a single client.

Force 5

Force 5 is offering a **FREE**
30-day evaluation of "JCloak"

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

GET YOUR OWN!

Subscribe today and receive "JDJ Digital Edition" FREE!

two years
24 issues

\$69⁹⁹

save \$30!

one year
12 issues

\$39⁹⁹

save \$10!

\$69 one year Canada/Mexico
\$99 one year all other countries

1 800-513-7111

or subscribe online for faster service
subscribe@sys-con.com

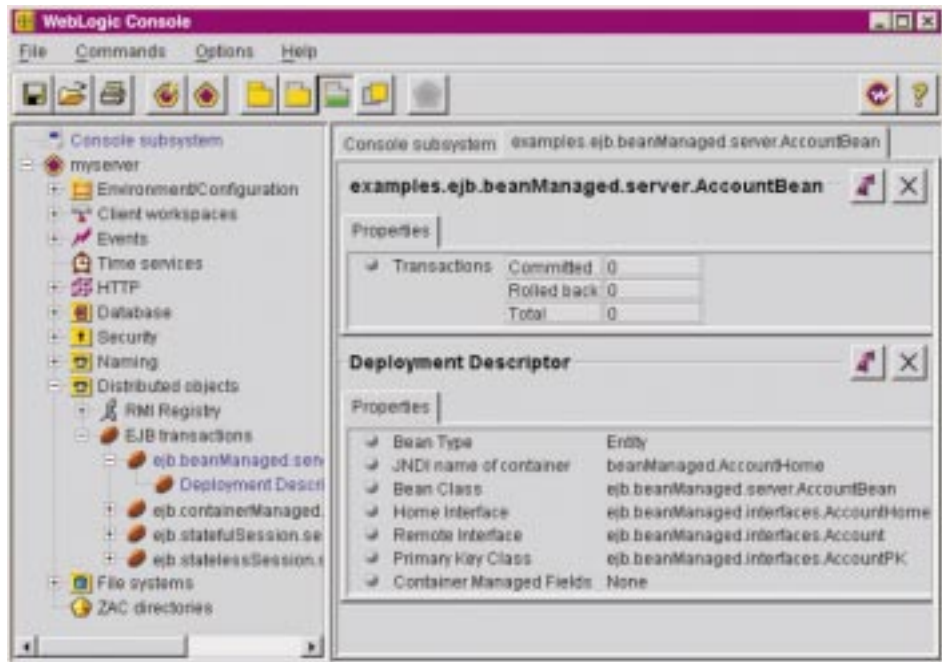


Figure 2: BEA WebLogic Server management console displaying information about a deployed entity bean

A third approach is to keep a secondary copy in memory on another machine. This is of course more susceptible to failures and isn't suitable for persistent components. The hard part here is determining when and how the state of an object has changed. (Persistent components are generally just written out on transaction boundaries.) If the application programmer is made responsible, presumably through some proprietary API, then the feature becomes harder to use. If the system is made responsible, then the feature may be less efficient since unnecessarily large updates may be performed more often than necessary.

In a WebLogic cluster, stateful session beans and RMI objects can be configured to use in-memory replication. The replication system relies on the programmer to determine when and how the state of an object has changed. It then takes care of transporting an update delta from the primary copy to the secondary copy. Scalability comes from distributing the primaries and secondaries across the cluster. This is in contrast to replication systems that keep all of the objects on (1) a fixed-size subset of the servers or (2) all of the servers.

The Naming Service

Access to clustered services is obtained through a JNDI-compliant naming service, which is itself replicated across the cluster so there is no single point of failure. To offer an instance of a clustered service, a server advertises a provider at a particular node in the replicated naming tree. Each server in the cluster adds a stub for this provider to a service pool stored at the node in its copy of the tree. When a client looks up the service, it obtains a smart stub that knows about the pool at this node. When the stub needs to

find a provider for load balancing or failover, it chooses from a list obtained from this pool.

Conclusion

The BEA WebLogic Server had evolved to meet the demands for scalability and high availability for mission-critical Web-based applications. BEA WebLogic Clusters provide scalable, highly available services in a transparent manner. The challenges of software-only clustering have been met by a combination of careful state management and highly optimized protocols based on new commodity technologies such as IP multicast. Initial measurements show that WebLogic clusters are both high performance and highly scalable. As an example, RMI benchmarks have shown that the throughput of a WebLogic cluster servicing 10,000 active clients scales linearly up to 10 servers, providing a maximum of 7,942 round-trip method invocations per second. In this benchmark, each server was on single-processor running Microsoft NT 4.0 with the JavaSoft JVM and the Symantec JIT. Similarly, tests at IBM have shown linear scaling up to 12 AS/400 processors. In general, experience with the BEA WebLogic Server has shown that Java and EJB do not pose limitations on performance as previously believed, and in fact can deliver the levels of performance, scalability and high availability required for mission-critical Web-based applications. ♦

About the Author

Dean Jacobs is an architect at WebXpress, a BEA company, where he is responsible for the WebLogic Server core and WebLogic clusters. He received his Ph.D. in computer science from Cornell University in 1985. He can be reached at dean@weblogic.com.



dean@weblogic.com

Sales Vision

www.salesvision.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

CASE STUDY

by Scott Davison

Demanding business

travelers with tight

budgets and time

constraints save all

around with

WebLogic-based

online travel

tool, intelliTRIP

TRIP.com's Online Travel Solution

There are nearly 6 million business travelers using the Internet, responsible for more than \$30 billion in travel expenses each year. For the Internet-based travel services industry, this is a tremendous revenue opportunity and the competition for this market is very intense.

At the same time, business travelers are perhaps the most demanding of travelers. Often, many of their trips are scheduled at the last moment, and under budget constraints. Additionally, these travelers may be booking frequent or regular trips, and don't want to spend more time than necessary searching for lowest fares or convenient flight times.

TRIP.com's intelliTRIP is an online search tool for travel reservations along with a host of additional information services, all accessible through any standard Web browser.

Saving time for the traveler is a major benefit of intelliTRIP; it allows the user to use a single request to access information from several different sources in just 90 seconds, rather than searching one by one through multiple airlines to gather flight information. IntelliTRIP relies on the BEA WebLogic Server and its Enterprise JavaBeans (EJB) technology as the foundation of its component-based architecture to deliver this fast and easy-to-use service.

One-stop Shopping at www.intelliTRIP.com

Users access their account via www.intelliTRIP.com, and specify their desired itinerary. IntelliTRIP immediately searches the entire airline inventory and provides the user with a list of as many as 25 flight options. This list will also include Internet-based specials that are not currently available outside the individual Web sites.

IntelliTRIP is a tool that searches multiple travel and airline Web sites, enabling users to compare fares, trip routes and airlines to find the most desirable flight plan. The BEA WebLogic Server manages each transaction. The direct airline connection is an advantage of IntelliTRIP, ensuring that users get the lowest possible fares guaranteed by the airline.

Making the BEA WebLogic Connection

"Helping business travelers find the quickest, easiest way to comparison shop airline fares has always been our objective," says Steve Graese, software development manager at TRIP.com. "So we are always looking for ways not only to stay ahead of our competition, but also to make sure we are continually using the latest technologies to bring the most useful information to our users and bring it to them faster."

TRIP.com currently handles hundreds of simultaneous users, and estimates usage rates as high as 14 million hits per month, with market growth as high as 50% annually. With these high usage numbers, scalability of the underlying system was a prime requirement for TRIP.com when selecting a Web application server. BEA WebLogic was selected as a crucial part of the IntelliTRIP

service since it met the requirements of scalability, reliability and security. In addition, BEA WebLogic's comprehensive implementation of EJB technology and support of Java and other industry standards provide a foundation for growth as the Internet-based business market continues to expand.

Before the implementation of BEA WebLogic, Web interfaces were a Netscape plug-in that could not support many of the browsers used by travelers. This limited the ability for IntelliTRIP to reach a large portion of its intended audience, and made it difficult for users to take advantage of its search capabilities.

Another concern of TRIP.com was to ensure that the IntelliTRIP infrastructure would be able to support the expected growth of its market. "We're looking at growth rates as high as 50% a year," Graese estimates. "So we know we're going to be adding servers to handle that kind of volume. We don't want IntelliTRIP users to quickly lose patience if there are any delays in providing the information they need. BEA WebLogic is meeting today's demands and it's what we'll be relying on as we grow."

That's why TRIP.com selected the BEA WebLogic Server for IntelliTRIP. Based on Java standards, BEA WebLogic is browser-independent and provides additional features that are compliant with any Web browser. "Our decision to go with BEA WebLogic was driven by a need to support all of IntelliTRIP's users," says Graese. "We were happy to see just how much BEA WebLogic added to our capabilities. [It] had so many features built in Enterprise JavaBeans, servlets and security that we could concentrate our efforts on the IntelliTRIP application itself, and not worry so much about building the infrastructure. BEA WebLogic is serving a critical function for us and for the IntelliTRIP user."

TRIP.com began developing the BEA WebLogic-based solution in December 1998, and launched in April 1999. Graese adds, "WebLogic saved us countless development dollars and labor hours, allowing us to provide the best possible product to our users in the shortest possible amount of time." IntelliTRIP utilizes the following BEA WebLogic features:

- Enterprise Java Beans EJBs for the secure sharing of transactional business components
- Servlets for supporting non-Java clients in using Web browsers
- Connection pooling for databases and query caching
- Authorization control lists for reliable security of access and transactions
- Built-in secure sockets layer for transaction security over public networks

Technical Specifications

- Three-tier, 100% Java-based architecture
- Sun Solaris 450 Enterprise Server
- Sun Java Virtual Machines 1.1.7.05
- Netscape Enterprise Server
- BEA WebLogic Server

About the Author

Scott Davison is one of the founding editors of SYS-CON Publications, Inc., the publisher of Java Developer's Journal. Scott can be reached at scott@sys-con.com.



scott@sys-con.com

Compuware

NuMega is offering a chance to win a FREE copy of "NuMega DevPartner 1.22 for Java!" a \$689 value.

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



Enterprise JavaBeans

Where does EJB fit into the Java platform for the Enterprise?

by Jason Westra

Mountaineering in the Office?

Developers and managers often think of enterprise applications as insurmountable mountain peaks that only experts can climb. Much like mountain climbing, building a large-scale, enterprise-wide system is a daunting task, not for the faint of heart!

As a consultant, I have spent much time as an application architect, leading clients to the summit of an enterprise-wide system. In my experience, guiding an unskilled climbing party is dangerous. Similarly, building an enterprise-wide application without first educating your developers in distributed component technologies is a perilous undertaking.

JavaSoft has presented us with APIs such as the Java Naming and Directory Interface (JNDI) API, Java Servlet API and Java Message Service (JMS) API to help in our efforts. However, their new specification, Enterprise JavaBeans (EJB), holds the most promise to ease development of enterprise applications.

EJB's component-based development approach provides an exciting step toward enabling the average developer to make significant impacts for their organization quickly. EJB "empowers" the developer by hiding many of the complexities relating to persistence, transaction management, security, resource pooling, distributed object location, the integration of disparate technologies, and more. However, while any technology may tout its ease of use, it usually takes time to understand exactly how to best apply the technology to solve business problems.

The first issue on everyone's agenda is: "Where does this new technology fit into our existing architecture?" or even "Where does this new technology fit into an enterprise Java architecture?" While choosing a route in climbing isn't easy, neither is finding your way through the numerous APIs of the Java platform for the Enterprise.

This article seeks to educate you on EJB's role in the Java platform for the Enterprise. Since describing EJB's ability to integrate with disparate technologies would constitute a lengthy article by itself, I'll focus on explaining the role of EJBs in a purely Java-based application solution today. Figure 1

puts the Java Enterprise APIs into perspective. Welcome to the Westra Party. Let's begin our trek!

Trailhead: Enterprise JavaBeans

When guiding an inexperienced party, a mountain guide generally will take the path of least resistance to the top. Enterprise JavaBeans represents the path of least resistance to the corporate developer who wishes to build scalable, portable, enterprise applications quickly. As JavaBeans has led the component revolution for portable, client-side development, EJB looks to do the same for portable server-side component development. As stated before, EJB eases development for corporate developers by hiding the complexities of building applications with distributed architectures. Let's review some key concepts in the EJB specification that make it easy to use, then dig into EJB's role within the Java Enterprise APIs.

The Enterprise JavaBeans specification, supported by industry leaders such as IBM, AOL/Netscape, BEA Systems and Oracle, defines a component model for building *n*-tiered Java applications in a distributed architecture. The specification defines two types of components or EnterpriseBeans:

- **SessionBeans:** Components that contain business logic and maintain session with a particular client
- **EntityBeans:** Components that represent business entities and are inherently persistent

Thus a ShoppingCartBean that holds products a user wishes to purchase from a Web site would be a prime candidate for a SessionBean. Likewise, the OrderBean in this Web application would be an example of an EntityBean that may contain price information, order date and a shipping address while stored in the seller's database.

Apart from defining EnterpriseBeans, the central ingredient of the EJB specification's component model with respect to simplifying distributed systems development is the concept of a component execution environment. Such an environment typically consists of an EJB Server and EJB containers.

JavaBean client components typically run within a visual container. Similarly, EJB containers allocate a process within their EJB Server for your server-side components (i.e., SessionBeans and EntityBeans) to execute. The container shelters your component from its runtime platform by managing all interactions with the operating system for the component. Thus, together, an EJB Server and its containers provide your components with access to runtime services such as persistence, distributed transaction management, threading and resource pooling.

Any vendor following the Enterprise JavaBeans specification can build EJB Server functionality into their products. For instance, database management systems, application servers, component transaction servers, transaction processing monitors and object transaction managers are all products that could include EJB Server functionality. Oracle's RDBMS, Novera JBusiness Application Server from Novera Software, Sybase's Jaguar Component Transaction Server and BEA System's M3 Object Transaction Manager, respectively, are products to watch now and in the future for EJB compliance.

The exciting thing about the EJB specification is that it sets a standard for server-side components to obtain distributed runtime services, much like the JavaBeans specification does for client-side beans. Also, application components that utilize their container only for services are guaranteed interoperability when migrated to other EJB Servers provided by third-party vendors. As we'll see, EJB Server functionality relies heavily on the Java Enterprise APIs. With that quick overview, we've reached our first stop, Base Camp.

Base Camp: Java Enterprise Persistence APIs

At Base Camp we discover how database interface standards integrate with Enterprise JavaBeans. The three leading Java DB standards – the JDBC 2.0, SQLJ and ODMG 2.0 Binding APIs – cover three related types of persistence models. The JDBC 2.0 API provides an object interface to SQL database calls. The SQLJ specification defines the use of embedded SQL running on the JDBC API to provide portable stored procedures and data types. Last, the ODMG 2.0 Binding API provides for the transparent storage of Java

IMI Systems

www.imisys.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

objects to an object database. While the EJB specification doesn't require an EJB Server to implement a specific storage API, most EJB vendors to date have provided support for JDBC. They'll likely add support for the remaining two enterprise persistence APIs as demand increases.

The EJB specification provides two mechanisms for storing EntityBeans: bean-managed persistence and container-managed persistence. Bean-managed persistence requires you to perform your own database connections. You must also understand at least one of the three complex storage APIs discussed above or object serialization. This approach offers you flexibility, but neither utilizes an EJB Server's power nor provides truly portable business components. For instance, what if your bean-managed EntityBean, originally built to access an RDBMS, now needs to be reused in an application that stores the component in a serialized format?

To solve this problem and more, EJB provides container-managed persistence. Container-managed persistence relies on the EJB server to generate the appropriate code to store the EntityBean. This component-based approach reduces development time by shielding you from writing complex storage code. It also promotes bean portability by not locking your EntityBeans into a particular storage scheme. The Java Enterprise Persistence APIs offer powerful options in data storage, while EJB's container-managed persistence simplifies development of database applications. What else does EJB offer to ease our trek?

Rest Stop: Java Transaction Service API

At this rest stop we'll review the Java Transaction Service to understand its relationship with Enterprise JavaBeans. The JTS is based on CORBA's OTS and provides an interface to manage distributed transactions to multiple resources with a two-phase commit protocol. Thus JTS provides EJB applications with the ability to perform distributed transactions across multiple databases and even promises to allow transactions spanning different EJB Servers! JTS provides important functionality, but EJB adds critical packaging or encapsulation that conceals the complexities of transaction management from application developers through container-managed transactions. With the latter, all transaction boundaries are formed implicitly by the container and the EJB Server. You can wire numerous EJBs together to create complex business logic without having to code a single explicit transaction demarcation.

Component-based development techniques allow transaction rules to be declared at EJB deployment without code modification. Transaction attributes can be declared at the bean level or they can be very granular,

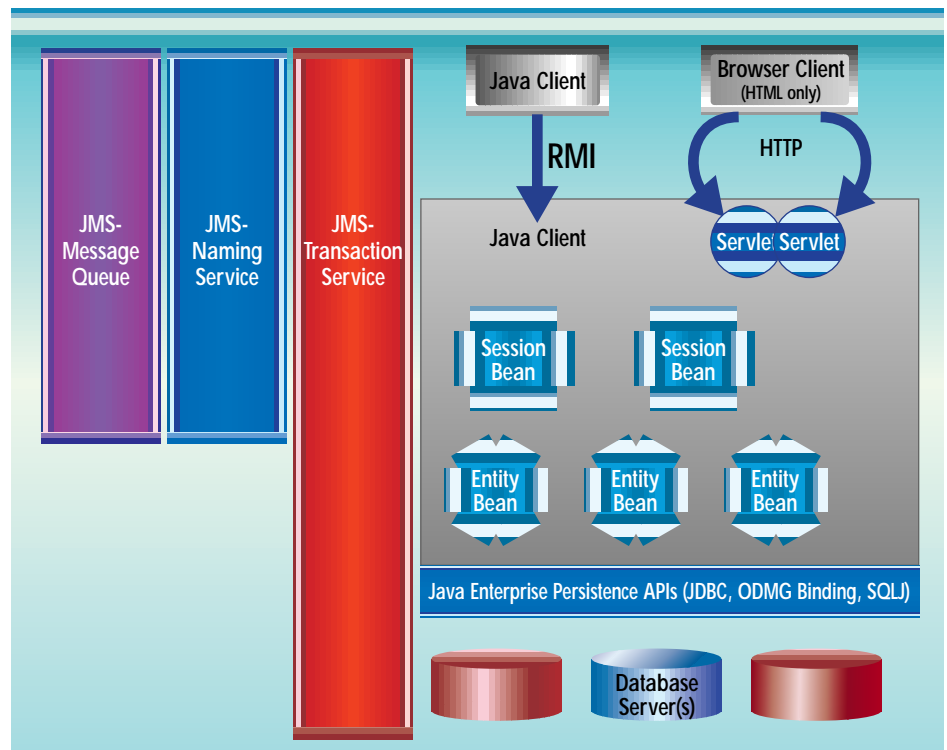


Figure 1: Standard Java Enterprise architecture

declared on a per-method basis. For example, an EJB deployed with TX_SUPPORTS in your initial application may be redeployed using TX_REQUIRED by utilizing your EJB Server's deployment wizards to modify the component's properties! While EJB also offers bean-managed transactions, only the most experienced developers should attempt to use this capability. Custom bean-managed transactions carry serious liabilities such as limited portability and reusability, and complexity of development. Figure 2 lists valid transaction attributes.

Java Naming and Directory Interface API

"This mountain is enormous. Where are we anyway?"

Mountaineers often use a receiver device to calculate their locations automatically from the global positioning system (GPS). Just as a satellite can transparently locate your position for you, EJB Servers use the JNDI to easily locate distributed Enterprise JavaBeans. JNDI presents a single interface to multiple naming and directory services in your enterprise. With it you can connect to heterogeneous services such as LDAP, NIS and CORBA (COS) Naming, and to an RMI Registry.

Because the EJB specification requires the use of JNDI, you don't need to know, for instance, what naming service to use or that your ShoppingCartBean is running on your NT server while your OrderBean is running on your UNIX box. You simply code to a single interface and access all distributed components transparently. Also, redeploying your components to another location has no

effect on your lookup code. However, keep in mind that redeploying them to another vendor's EJB Server may require lookup code changes if you haven't insulated your calls with sound, object-oriented techniques.

Java Remote Method Invocation ***"Base Camp to Westra Party....Do you copy?"***

Distributed communications is essential in mountaineering. At Base Camp we left behind a two-way radio and Hank, the radio operator, to inform us on important matters like dangerous weather conditions. Likewise, the ability to communicate with distributed objects is essential to scalable enterprise applications.

JavaSoft developed RMI to allow distributed communications between Java Virtual Machines in an object-oriented fashion. Objects implementing the java.io.Serializable interface can be transported via RMI without manual byte streams management, which is a leap beyond socket communications!

However, taking advantage of RMI can be tedious work. It begins with defining your remote implementation class. With RMI, a distributed object doesn't move and all access is through a remote interface that you must define as well. This contains a subset of its distributed object's methods, presenting only methods you want to call remotely on your distributed object. Next, a tool is used to generate skeletons and stubs from your remote implementation class. These classes utilize a marshaling scheme to inflate and deflate objects passed to and from your remote object. Now that your RMI classes (remote implementation class, remote inter-

ZTI

www.zti.ca

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

face, client stub and remote skeleton) are ready for deployment, you must perform routine chores, such as starting a bootstrap registry service and binding your distributed references with a unique name to the registry. To access your distributed objects you must code lookup calls to the `java.rmi.Naming` class, passing URLs as keys. To totally confuse the situation, you now wonder which lookup service to really use, the JNDI API or the `java.rmi.Naming` class!

EJB takes much of the tedium and confusion out of developing distributed objects. Your EJB objects must adhere to Java RMI rules, such as arguments and return values must be legal types, and you must define a remote interface for each `SessionBean` or `EntityBean`. However, EJB does conceal the need to perform bootstrap registry services, register your EJBs and access numerous naming services to get a distributed reference. EJB Servers provide skeleton and stub generation for your EJBs, as well as life-cycle management (activation and deactivation) of your components. Some products may even offer automatic generation of remote interfaces for your EJBs. Finally, as stated earlier, the EJB specification prevents confusion over multiple naming services by requiring the use of JNDI to look up all components. The RMI Registry can be accessed via JNDI's API.

Why would you use regular RMI, if EJB hides much of RMI's complexities? The most compelling reason is that the current specification for EJB doesn't support shared, stateful services. The closest EJB has is a stateful `SessionBean`, but that exists only on a per-client basis. It can't be shared between multiple users. This is an area where coding an RMI server would be a suitable solution. For instance, you might use an RMI server as a shared service to load reference table data into memory only once.

From this example we can see that EJB is not a "be-all, end-all" solution to our needs. Other Java Enterprise APIs will directly impact our distributed architecture as well. In any event, the ability to communicate remotely is a necessity in our enterprise-wide solution, and working with EJB makes life even easier.

Java Message Service API

"Westra Party to Base Camp....I just received your message. Sorry, my com-link was turned off."

When a message is important, it must have guaranteed delivery, especially when climbers' lives are at stake. The need for guaranteed message delivery is common in distributed enterprise applications as well. In the Java platform for the Enterprise, the JMS guarantees message delivery through asynchronous communications based on messaging queuing, and publish and subscribe mechanisms. The JTS provides a degree of message guarantee in that it raises an excep-

TX_NOT_SUPPORTED	The container will always invoke the method without a transaction context, suspending a transaction if currently within a transaction scope.
TX_BEAN_MANAGED	The enterprise bean can use the <code>java.jts.Usertransaction</code> interface to explicitly create transaction boundaries.
TX_REQUIRED	The enterprise bean must be within the transaction scope. If the client invokes it within an existing transaction, its transaction context is used. If a client invokes the bean with no transaction, a new transaction is automatically created by the container.
TX_SUPPORTS	The enterprise bean is invoked within a transaction if one currently exists. If none exist, the bean is invoked without a transaction.
TX_REQUIRES_NEW	The container starts a new transaction on each call to the enterprise bean.

Figure 2: Transaction attributes

tion if not all components in a transaction are accessed successfully. However, the current specification for Enterprise JavaBeans doesn't support asynchronous messaging. So if your enterprise-wide solution needs guaranteed messaging, choose an application server that supports not only EJB, but provides JMS support as well.

Java Servlet API

"Look what I found, an old ice ax!"

Climbing equipment is outdated every year; technically advanced products make old gear obsolete. The relation of Java servlets to Enterprise JavaBeans is similar. In mid-1997, JavaSoft released the Java Servlet API to define how Java servlets and Web servers communicate with one another and provide the first real use of server-side Java. To utilize servlets, your Web server must be Java-enabled and support the Java Servlet API. Most *n*-tiered Java applications currently in production have been built around servlet functionality. For these applications, customized servlets provided load balancing, fail-over, session management, database access, servlet-oriented security and business rules processing on a middle tier.

Much of the functionality just described has been replaced by EJB with the help of application servers. Application servers supporting the EJB specification allow you to define and execute business logic, session management and database access logic from within your `SessionBeans` and `EntityBeans` instead.

Is there a need to use servlets in an EJB implementation? Aren't they in opposition to each other? Yes...and Maybe. Just as a thrifty climber makes use of old equipment when he finds it, servlets remain an integral part of a distributed enterprise Java solution. Servlets are great for accessing EJBs when a limitation prevents the remote invocation of the `EnterpriseBean` from a client. For instance, if you build an Internet application that is pure HTML, you'll have a problem calling your EJBs from your browser.

Likewise, if corporate policy prevents using IIOP to cross its firewalls, servlets are a perfect choice to communicate directly with your EJBs once the firewall is breached with an HTTP request. In your Java enterprise solution, servlets will move to more of a router and HTML builder/parser role, routing requests to the appropriate EJB as well as dynamically building/parsing HTML pages from EJB data. Move your database access and business rule processing into EJBs, relieving servlets of these responsibilities.

Summary/Conclusion

On our journey up the mountain of distributed systems development, we looked at Enterprise JavaBeans' role in the Java platform for the Enterprise. With EJB, JavaSoft has provided the final piece of gear you need to build scalable, enterprise-wide systems! Previously released APIs of the Java platform for the Enterprise either supplement EJB's weaknesses or lay the foundation for EJB's strengths. EJB is the glue that provides a uniform bundling of the Java Enterprise APIs, increasing application reliability and scalability as well as guaranteeing portability, ease of use and interoperability across EJB Servers. The Java Enterprise APIs are a powerful set of interfaces, and with the help of tools from vendors such as Novera Software, BEA and IBM you'll be well equipped to build your enterprise-wide system quickly.

We came far this month, but we haven't reached the summit yet, nor have we covered this topic in detail. Look to www.javasoft.com/products/OV_enterpriseProduct.html for more information about the Java Enterprise APIs. Happy trekking! 🏔️

About the Author:

Jason Westra resides in Boulder, Colorado, where he is a senior consultant with the component-based development practice of CSC Consulting. He can be reached at jwestra@uswestmail.net.



jwestra@uswestmail.net

Slangsoft

www.slangsoft.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

eBusiness Journal

Introducing eBusiness Journal

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

JDJ Advertising

www.sys-con.com/java/adnews.htm

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

NSICOM

www.nsicom.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



A Windows-Specific Java Utility Class

Written with the help of the JNI, this class can answer many questions

by Pat Paternostro

How do I disable a Frame's Minimize and Maximize buttons? How do I obtain the amount of a drive's free disk space? How do I retrieve the window handle of a Java Frame? How can I read the label of a disk drive? These and other similar questions are quite prevalent on the Java Usenet newsgroups. The answer to these questions is always the same: you need to use the Java Native Interface (JNI). These types of questions are so commonplace (not only in the newsgroups) that I decided to provide the answers in the form of a Windows-specific Java utility class written with the help of the JNI.

Background

One of Java's many strengths is its platform-independent nature. This feature allows a pure Java program written under a Win32 operating system (i.e., Win95/98/NT) to be easily ported to a UNIX operating system without any change to the program's compiled bytecode. However, there are times when a call to an operating system function from within a Java program is necessary and desirable. The JNI is the mechanism by which a Java program can call an OS specific function. The JNI comes complete with its own set of data types and native methods that can create Java objects (arrays, Strings, etc.), call Java methods, and catch and throw exceptions. A detailed explanation of the JNI is beyond the scope of this article. For more information look at the JNI section of Sun's Java Tutorial located at www.javasoft.com/docs/books/tutorial/native1.1/index.html. Alternatively, read Sun's JNI Specification at <http://java.sun.com/products/jdk/1.1/docs/guide/jni/spec/jniTOC.doc.html>.

Implementation

The Java utility class, JUtil, contains 21 static (class) methods: three pure Java

methods whose implementations reside in a Java source file (see Listing 1) and 18 native methods whose implementations reside in a C source file (see Listing 2). Table 1 lists the method names, where the methods are implemented, and a brief method description.

The implementations for the 18 native JUtil methods make use of native JNI methods and Win32 API functions to accomplish their tasks. The C function prototypes are generated from the Java class file using the JDK javah utility that creates a C header file (see Listing 3). This C header file must be

a JUtil method is called from within your Java program. The JUtil class is made final, so it can't be subclassed (as it's strictly a utility class), and the constructor is made private so the class can't be instantiated.

I'm not going to detail the use of all the JUtil methods as I've provided an HTML help file generated from the Java source file's javadoc comments. Nevertheless, I will cover the most utilized methods.

Quite a few of the JUtil native methods deal with manipulating a Java window. These methods require a Win32 window handle to perform their native operation. You retrieve a Java window's Win32 window handle by using the JUtil class's getHwnd() method and passing it the title of the window. Although it's not an ideal solution, as you can have multiple windows with the same name, it's the only viable way of retrieving the required information. This method allows you to retrieve the Win32 handle of any application window currently running on the system. You must take care to specify the exact window title, including case and punctuation.

When a Java Frame is set unresizable, via the Frame class's setResizable() method, the user is prevented from resizing the Frame with the mouse. Unfortunately, this method doesn't disable the Frame's Minimize and Maximize buttons or its equivalent system menu items. Using the JUtil class's setMinimizeEnabled() and setMaximizeEnabled() methods you can disable those buttons and menu items for the specified window. Except for the Close menu item, there is a JUtil native method to manipulate all of a window's system menu items (i.e., Restore, Move, Size, Minimize and Maximize). (Note: Starting with JDK 1.1.7B and Java 2 [formerly JDK 1.2] setting a Frame unresizable will visually remove the Frame's Minimize and Maximize buttons and disable those system menu items.)

At times you may want to start your main Java program's window in a maximized state. You can initiate this action with Pure Java methods by determining the screen size, via the Toolkit class's getScreenSize() method, and then sizing the window to the retrieved screen dimen-



Figure 1: The JUtilTest program

included in the C source file, which is compiled into a Dynamic Link Library (DLL) file.

Since all the methods are static, there's no need to instantiate an object of the JUtil class to use them. Naturally, you'll have to qualify the method names with the class name when invoking the methods. I call the System class's loadLibrary() method inside a static block of the Java source file (see Listing 1). This method loads the DLL file at class load time, which occurs the first time

Visualize Inc.

Visualize is offering a
FREE 30-day evaluation
of VantagePoint or
DataVista Pro

See JDJ Special Offer at:
[http://www.syscon.com/java/
specialsoftheweek.html](http://www.syscon.com/java/specialsoftheweek.html)

The Theory Center

www.theorycenter.com

See JDJ Special Offer at:
[http://www.syscon.com/java/
specialsoftheweek.html](http://www.syscon.com/java/specialsoftheweek.html)

reservoir labs

info@reservoir.com

See JDJ Special Offer at:
[http://www.syscon.com/java/
specialsoftheweek.html](http://www.syscon.com/java/specialsoftheweek.html)

Wall Street Wise

[www.wallstreetwise.
com/jspell.html](http://www.wallstreetwise.com/jspell.html)

See JDJ Special Offer at:
[http://www.syscon.com/java/
specialsoftheweek.html](http://www.syscon.com/java/specialsoftheweek.html)

sions. However, the presence of the Windows taskbar or any other dockable desktop toolbar – such as Microsoft’s Office toolbar – is not taken into account, resulting in the taskbar becoming obscured by the window. Using the JUtil class’s `setWindowMaximized()` method maximizes the window without obscuring any dockable desktop toolbars. To programmatically minimize (iconify) a window use the JUtil class’s `setWindowMinimized()` method.

Another useful JUtil method is the `setWindowRestored()` method. This method restores the specified window to its original state before it is minimized or maximized. If the window is minimized (iconified), it not only restores the window’s original state, it makes the window active (selected) as well. (Note: Starting with Java 2, the `Frame` class provides the `setState()` method to control the `Frame`’s state. Passing `Frame.NORMAL` to the `setState()` method will restore the `Frame` to its original state whereas passing `Frame.ICONIFIED` will minimize the `Frame`.)

In my opinion, the most useful of the window-specific JUtil native methods is `setWindowAlwaysOnTop()`. This method allows you to set the specified window as the topmost window in the z-order. Again, you can achieve this behavior via pure Java methods but the implementation always seems to fall short of the desired behavior. For instance, most programmers will implement the `WindowListener` interface on a `Frame` and provide an implementation for the `WindowListener`’s `windowDeactivated()` method from which the `Window` class’s `ToFront()` method is called to keep the `Frame` activated at all times. Unfortunately, this results in not allowing the user of the Java program to select any other application currently visible on the desktop. The true behavior of setting a window as the topmost window in the z-order allows the user to activate (select) any other application window while keeping the topmost window visible but deactivated.

I’ve noted where the newer versions of the JDK (i.e., 1.1.7B and Java 2) obviate the need for a couple of the JUtil methods. However, the window-specific JUtil native methods can be used to manipulate any window from a Java program, not just the Java program’s own windows.

I’ve included a Java test program (see Figure 1) that exercises all of the JUtil methods. This test program along with the javadoc generated HTML help file should be sufficient to get you started using the JUtil class.

Anomaly

I mentioned earlier that the JUtil class contains native methods that manipulate all of a window’s system menu items except the `Close` menu item. These methods will either enable or disable the system menu item and associated button (if any)

Method Name	Where Implemented	Brief Method Description
<code>getConsoleChar()</code>	C source file	Retrieves the character typed at the command console
<code>getLogicalDrives()</code>	C source file	Retrieves a system’s logical drives
<code>getFreeDiskSpace()</code>	C source file	Retrieves the specified drive’s free disk space
<code>getDriveType()</code>	C source file	Retrieves the specified drive’s type
<code>getVolumeLabel()</code>	C source file	Retrieves the specified volume’s label
<code>setVolumeLabel()</code>	C source file	Sets the specified volume’s label
<code>getCurrentDirectory()</code>	C source file	Retrieves the current directory
<code>setCurrentDirectory()</code>	C source file	Sets the current directory
<code>getHwnd()</code>	C source file	Retrieves the Win32 window handle for the specified window
<code>setWindowMinimized()</code>	C source file	Minimizes the specified window
<code>setWindowMaximized()</code>	C source file	Maximizes the specified window
<code>setWindowRestored()</code>	C source file	Restores the specified window
<code>setWindowRestoreEnabled()</code>	C source file	Enables/disables the specified window’s Restore button
<code>setWindowMoveEnabled()</code>	C source file	Enables/disables the specified window’s Move system menu item
<code>setWindowSizeEnabled()</code>	C source file	Enables/disables the specified window’s Size system menu item
<code>setWindowMinimizeEnabled()</code>	C source file	Enables/disables the specified window’s Minimize button
<code>setWindowMaximizeEnabled()</code>	C source file	Enables/disables the specified window’s Maximize button
<code>setWindowAlwaysOnTop()</code>	C source file	Sets the specified window as the topmost window in the z-order
<code>setContainerDefaultFont()</code>	Java source file	Sets the specified container’s components to the specified font
<code>setMenuBarDefaultFont()</code>	Java source file	Sets the specified menu bar’s menus and menu items to the specified font
<code>copyFile()</code>	Java source file	Copies the specified source file to the specified destination file

Table 1: JUtil Class methods

on the specified window. While disabling these system menu items works flawlessly, the opposite isn’t true. Using these methods to enable the window’s system menu items will visually set the menu item as selectable but won’t initiate the menu action when the item is actually selected. I’m at a complete loss to explain this behavior as the Win32 API function used to enable the menu items works without any problem when called directly from a non-Java Windows program.

Summary

The JUtil class and its associated DLL file provides a Windows-specific solution for many of the most frequently asked Java Usenet ques-

tions. Although the class is not comprehensive, it is quite useful and can be used as a building block for adding your own methods. ☛

▼▼▼▼▼ CODE LISTING ▼▼▼▼▼

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

About the Author

Pat Paternostro is the director of education for Tri-Com Consulting Group in Rocky Hill, Connecticut. Tri-Com provides programming services for a variety of development tasks. You can reach Pat at ppaternostro@tricomgroup.com.

ppaternostro@tricomgroup.com

9NetAvenue

9NetAvenue is offering 3 months of FREE Java Hosting to JDJ readers.

*Offer expires 6/30/99
Use code ""javadev"

See JDJ Special Offer at:

<http://www.sys-con.com/java/specialsoftheweek.html>

IBM

www.as400.ibm.com

See JDJ Special
<http://www.sys-con.com/java>

M

www.java.com/hotapps4

Special Offer at:

www.java.com/specialsoftheweek.html

JDJ Readers' Choice Awards Winners to Be Announced at JavaOne!

(Pearl River, NY) – Over the past few months 15,470 JDJ readers cast their votes for the best Java tools in 14 product categories. JDJ Readers' Choice



Awards, also known as the "Oscars of the software industry," will be distributed at JavaOne '99 on June 15. This year, one winner and two finalists will be awarded in each category. For details visit www.javadevelopersjournal.com.

JDJ and SYS-CON Radio Named Media Cosponsors of JavaOne '99

(Pearl River, NY) – *Java Developer's Journal* and SYS-CON Radio have been named media cosponsors of JavaOne '99. SYS-CON Radio will provide live coverage of the conference, which will be held from June 15 to June 18 this year. "Tune in" to www.javadevelopersjournal.com for more information.



object-oriented OLAP solution that supports interactive decision making. Anyone within a company can graphically and securely search, view, manipulate and report mission-critical information anytime, anywhere.

For details visit their Web site at www.zti.com.

Elixir and Barr Partner to Integrate Management and Datastream Conversion

(Ventura, CA) – Elixir Technologies Corporation and Barr Systems, Inc., have signed a cooperative marketing agreement that will enable both companies to enhance their enterprise-wide document and data solution offerings. Under the agreement, both companies will have the ability to resell each others' products and integrate them into comprehensive solutions that address document composition, printing management and datastream conversion.

For details visit the Elixir Web site at www.elixir.com.

ObjectSpace Brings Its Vision to EIS Council

(Dallas, TX) – ObjectSpace, Inc., a supplier of distributed computing solutions for the enterprise, will join EAI market leaders in the creation of the Enterprise Integration Standards Council.



According to International

Data Corporation, the EAI market is expected to approach \$1 billion in products and services by the year 2000. The council's purpose is to identify the scope of business forces that are shaping the enterprise integration market, identify business problems that enterprise integration is intended to solve, establish a common definition of enterprise integration and define enterprise integration terminology.

For more information visit www.objectspace.com or www.iswatch.com.

Cerebellum Releases Version 1.2 of Development Platform

(Pittsburgh, PA) – Cerebellum Software Inc. has developed Cerebellum 1.2, the first of their next-generation application development platforms based on total data independence. This



makes it possible to quickly and easily access, manage and combine data in disparate, incompatible sources.

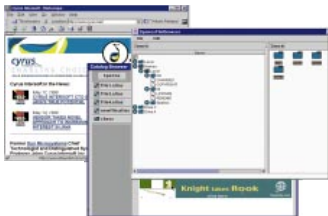
Similar to the way Java offers application developers OS independence, Cerebellum provides database developers with a platform-independent development environment and a single graphical user interface for accessing data and visually creating queries.

Additional details are available from www.cerebellumsoft.com.

Cyrus Intersoft Delivers Anytime, Anywhere Computing

(Los Angeles, CA) – Cyrus Intersoft, Inc., has released Speiros, a pervasive network computing platform that enables anytime, anywhere computing.

Speiros delivers an environment in which a user can access data, programs and services from any Java-enabled device anywhere in the world. With it, a user can securely use any computing resource – applications, file systems, printers, e-mail or



other networks – from a desktop, palmtop, set-top, kiosk or any other network-capable device.

For more information go to www.cyrusintersoft.com.

Secant Extreme Enterprise Server for EJB

(Cleveland, OH) – The Secant Extreme Enterprise Server for EJB provides a complete environment for assembling, deploying and maintaining scalable multitier business systems. The product is based on Secant time-tested and

proven Object Transaction Monitor and object data management technology. This technology includes implementations of CORBA ORBs and services such as transactions, security, persistence, events, concurrency and locking.

Using the standard EJB to CORBA Mapping, Secant Extreme Enterprise Server for EJB provides a robust and scalable EJB 1.0-compliant container and server that provides the complete deployment environment for reusable enterprise beans. Visit their Web site at www.secant.com for additional details.

ZTI to Launch Java 2-Compliant Version of O3

(Nepean, ON) – Zim Technologists International Inc. has announced plans to release O3 version 2.0, a completely Java 2-compliant business intelligence software. This tool allows users to search, view, manipulate, analyze and report data across the entire enterprise.



ZTI's O3 version 2 is targeted to medium and large businesses looking to consolidate and analyze massive amounts of data. O3 is a client/server, intranet-ready,

Oracle Cofounder Launches New Company

(San Mateo, CA) – Bruce Scott, cofounder of both Oracle Corporation and Gupta Technology (now Centyra Software) formally launched his new startup to facilitate growth of Internet-based embedded database applications. Scott founded PointBase, Inc., formerly known as DataBahn, to bring to market a family of portable 100% Pure Java embedded database products, including the industry's

first corporate data "hotsync" capability, to support e-business, Web-based mobile workforce applications and a range of Internet appliances, such as



Bruce Scott of PointBase, Inc. recently visited the JDJ offices in Pearl River, New York

Web-based PDAs and set-top boxes. For more information visit www.pointbase.com.

Macromedia

Macromedia is offering a
FREE 30-day trial of
"Dreamweaver 2"

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

ADVERTISER INDEX

Advertiser	Page
9NetAvenue, Inc. www.9netave.com	85 888-9NETAVE
Activated Intelligence www.activated.com	91 212-896-8220
Bateman, Inc. www.batemaninc.com	91 805-383-3338
BEA WebLogic www.weblogic.beasys.com	2 800-817-4BEA
Blue Sky Software www.blue-sky.com	23 800-559-4423
borland.com www.interbase.com/products/demoj dj.html	800-451-7788 x7183 21
borland.com www.borland.com/jbuilder/ads/javadev	800-336-6464 49
Computer Associates International, Inc. www.cai.com/ads/jasmine/dev	6 888-7JASMINE
Career Opportunity Advertisers 800-846-7591	92-99
Cerebellum Software www.cerebellumsoft.com	31 888-862-9898
Cloudscape, Inc. www.cloudscape.com	15 888-59JAVA1
Compuware Corporation www.compuware.com/numega	73 888-686-3434
Cyrus Intersoft, Inc. www.cyrusintersoft.com	39 612-331-6600
DevelopMentor www.develop.com	55 800-699-1932
e-Business Journal www.sys-con.com	80 800-513-7111
Elixir Technology www.elixir.com.sg	101/117 65 532-4300
EnterpriseSoft www.EnterpriseSoft.com	11 510-742-6700
FINDaHOST.com www.findahost.com/host/jdj	40&62 440-257-6690
Flashline.com, Inc. www.flashline.com	59 216-861-4000
Force 5 Software, Inc. www.force5.com	69 408-735-0665

Advertiser	Page
IBM Corporation www.as400.ibm.com/hotapps4	86-87 800-772-2227
IMI Systems Inc. www.imisys.com	75 800-828-0180
InetSoft Technology Corp www.inetsoftcorp.com	65 732-235-0137
Inprise Corporation www.inprise.com/appserver	13 800-336-6464
Insignia Solutions, Inc. www.insignia.com	63 800-848-7677
Instantiations Inc. www.instantiations.com	42 800-808-3737
InterLand, Inc. www.interland.net	67 800-217-0985
Intuitive Systems, Inc. www.optimizeit.com	51 408-245-8540
Java Developer's Journal subscribe@sys-con.com	70 800-513-7111
Java Developer's Journal.com www.sys-con.com	81 800-513-7111
KL Group Inc. www.klgroup.com/jclass/look	34-35 888-328-9597
KL Group Inc. www.klgroup.com/culprits	BC 888-328-9597
KL Group Inc. www.klgroup.com/truth	57 888-328-9597
Macromedia www.macromedia.com	89 800-457-1774
Microsoft Corporation www.msdn.microsoft.com/visualc	44-45 800-509-8344
NSI COM, Ltd. www.NSI.com	81 877 480-3311
Object Domain Systems, Inc. www.objectdomain.com	33 919-461-4904
Object International Software www.togetherj.com	47 919-772-9350
ObjectSpace, Inc. www.objectspace.com/go/universal	IBC 800-OBJECT 1
OneRealm, Inc. www.OneRealm.com/JDJ	7 303-247-1284

Advertiser	Page
OpenLink Software, Inc. www.openlinksw.com/virtuoso	37 781-273-0900
Oracle Corporation www.oracle.com/info/32	17 800-633-1072 x23839
Progress Software Corporation www.apptivity.com	29 800-477-6473 x4700
ProtoView www.protoview.com	3 800-231-8588
Reservoir Labs, Inc. www.reservoir.com	83 212-780-0527
Riverton Software Corporation www.riverton.com	27 781-229-0070
Sales Vision www.salesvision.com	71 800-275-4314
Sun Microsystems, Inc. www.sun.com/service/suned	4 800-422-8020
SL Corporation www.sl.com	41 415-927-1724
Slangsoft www.slangsoft.com	79 972-375-18127
Specialized Software www.SpecializedSoftware.com/jdj/	800-328-2825 x6576 30
The Object People, Inc. www.objectpeople.com	52-53 613-225-8812
The Theory Center www.theorycenter.com	61 888-843-6791
Tidestone Technologies, Inc. www.tidestone.com	43 888-880-0665
/training/etc, Inc. www.trainingetc.com	91 410-531-9953
Visualize Inc. www.visualizeinc.com	83 602-861-0999
Wall Street Wise Software www.wallstreetwise.com/jspell.html	212-348-5031 83
WEB99 www.mfweb.com	86 800-441-8826
Zero G Software www.ZeroG.com	25 415-512-7771
Zim Technologies International www.zti.ca	77 613-727-1397

Activated Intelligence

www.activated.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Bateman Inc.

www.batemaninc.com

Training Etc.

www.trainingetc.com

Employment Ad

new

Employment Ad

new

Employment Ad

new

Employment Ad

new

Employment Ad

new

Employment Ad new

Employment Ad

new

Employment Ad

new



The Great JavaLobby Developer Survey of 1999

Your help is needed

THE GRIND

by Rick Ross

“Great market
research
information
about Java
should be
available to
all of us”



Rick Ross is president and founder of the JavaLobby (www.javalobby.org), which currently has more than 36,000 members. He is also president of Activated Intelligence (www.activated.com) and can be reached at rick@activated.com.



rick@activated.com

I'm sure you've often seen articles citing market research reports from high-powered industry analyst groups. Market research is a multibillion-dollar industry in its own right, and major economic and political decisions are often based on the information gleaned from these reports. I'm also sure there have been numerous high-quality reports produced to examine every aspect of Java – I've been fortunate enough to have seen a few. The problem is that these reports are usually distributed only to the few decision makers who have the budget to afford the hefty price tags they invariably carry. Sure, you can sometimes read the free one-page summaries released to help promote these studies, but the full reports often cost many thousands of dollars to purchase.

Not Just for Those Who Can Afford to Pay

So many of us in the Java developer community work independently or in small companies that we may never be able to afford this type of industry intelligence, even though it could potentially be invaluable to us. Too much guesswork dominates our decision making as a result, but it doesn't need to be this way. Great market research information and insights about Java technology, the market and trends in the Java developer community should be available to all of us, not just to those who can afford to pay. The JavaLobby is launching a new project to help make sure this is the case – The Great JavaLobby Developer Survey of 1999. This new project should be of interest to all of us, and particularly to the vendors who advertise here in *Java Developer's Journal!*

The idea is simple: we can use our own skills to develop and deploy a worldwide survey on the Internet, and we can collectively benefit from the knowledge we gain. It couldn't be easier. And who better to teach us about the Java developer market if not ourselves? The Internet has made it possible to gather this information economically and distribute the results – without the services of those high-powered industry analysts who would gladly separate us from our money, if only we had it. JavaLobby will endeavor to use quality scientific methods to gather useful and statistically valid data – and we welcome the help and sponsorship of major Java players to obtain the services of research experts. Most, if not all, of the results obtained from the survey will be both public and free.

Three Steps to Success

Your help is needed to make The Great JavaLobby Developer Survey of 1999 a success, and I hope you'll be ready and willing to get involved. There are several ways you can help, and all of them are easy. First, we'll definitely need you to participate by responding to the survey. It will be online, and you'll probably be hearing about it from a lot of people. Details will certainly be available at the JavaLobby Web site (www.javalobby.org). Second, we'll also need you to tell your Java friends and colleagues about the survey and ask them to take it as well. Finally, we hope you'll participate in the discussion of the results, which will undoubtedly supply some food for thought and possibly some surprises. Overall, this should be a fantastic opportunity for us, as a worldwide community of developers, to help ourselves!

Great Benefits

There will be great benefits in it for you, too. If we can work together to make this survey a success, you should be able to enjoy a much more focused and detailed “big picture” of the Java market and its shifting trends as a result. You should also be better able to understand how your personal views compare and contrast with those of your Java developer peers around the world. The survey should help you by providing you with better intelligence of your own to help guide your personal career and product planning. It just may give us a glimpse into Java's future. Who knows?

It should be really exciting to follow this survey project as it unfolds, and I hope you'll be there to take part. It will undoubtedly be a learning experience for all of us. Thanks for reading, and I'll see you here next month! ☺

JavaOne

<http://java.sun.com/javaone/>

See JDJ Special Offer at:

<http://www.sys-con.com/java/specialsoftheweek.html>

JBDJ News

Borland and InLine Software Sign Online Distribution Agreement (Leesburg, VA) - InLine Software Corporation has announced a distribution agreement with Borland, which has agreed to offer Assembly Line Standard, InLine's Enterprise JavaBeans engine, for sale from its Web site. Borland will offer it as a standalone software product and in conjunction with other products, such as JBuilder.



Assembly Line Standard Edition, part of InLine's Assembly Line product family, is designed to automate the process of building EJBs, making it easier, safer and faster to build, test, deploy, maintain and upgrade enterprise-scale applications based on Sun Microsystems' Enterprise JavaBeans technology. A software plug-in, Assembly Line Standard extends the functionality of IDEs, such as Borland's JBuilder.

Visit www.borland.com or www.inline-software.com.



Innoview Announces Advanced Support for Localization Team

(Helsinki, Finland) - Innoview Data Technologies has announced that all recent Multilizer product upgrades add enhanced support for the whole localization team. The technical improvements benefit individual users and simplify the work between team members, increasing productivity.



The power of the architecture lies in the fact that everyone on the localization team works with the same resource, called *dictionary*. In practice this means that developers use Multilizer components for making the software access the translations. Linguists work with dictionaries, using Language Manager, which automates many routine tasks and ensures linguistic consistency.

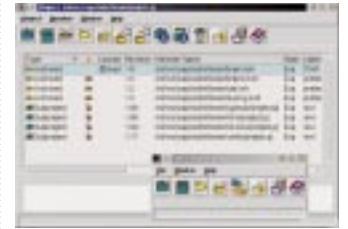
More information is available on Innoview's Web site at www.multilizer.com.

Ensemble Delivers Stronger Integration

(Richmond, BC) - Systems Inc. has announced the availability of its latest integration between Rational Rose 98i and Borland Delphi 4. Ensemble has added new capabilities that enhance and simplify generation and reverse engineering of Object Pascal code between

Rational Rose 98i object models and Borland Delphi projects. Ensemble developed RDL 3 in an inclusive effort with Rose Delphi Link customers, Rational Software Corporation and Borland.

For more information visit www.ensemble-systems.com.



Linux Support for MKS Source Integrity

(Waterloo, ON) - Mortice Kern Systems Inc. has announced its support for the Linux operating system, making MKS Source Integrity, a component of MKS's enterprise software management solution (the MKS Integrity Framework), immediately available for the Linux platform. The company also pledged long-term support for Linux, indicating the popular OS will continue as a key supported platform for MKS in future releases of MKS Source Integrity.

For more information visit www.mks.com/solution/si/.



USA.NET Names MCI WorldCom Exec as President and CEO

(Colorado Springs, CO) - USA.NET, a provider of e-mail and advanced electronic messaging solutions, has appointed John Gerdelman as its new president and CEO. Gerdelman, a former president of MCI WorldCom, has more than 20 years of sales, marketing, network and information technology experience.

The addition of Gerdelman, who will oversee strategic operations of USA.NET, diversifies USA.NET's leadership and creates a reorganization of the company's management team.

Visit www.usa.net.



Referentia Systems Provides Multimedia Training for JBuilder 3

(Honolulu, HI) - Referentia Systems Incorporated and Borland, the software development tools division of Inprise Corporation, have released an updated version of the popular *Building Database Applications* interactive multimedia training volume for JBuilder 3.

The training volume will help JBuilder 3 users build database applications quickly and efficiently. It features over 45 in-depth lessons and concept

animations covering key techniques for developing Java database applications. Users will gain a strong understanding of the JBuilder data model, master a solid foundation of essential and advanced database techniques, and learn various development strategies. The updated content covers the new database features and functionality of JBuilder 3, with upgraded examples using dbSwing components.

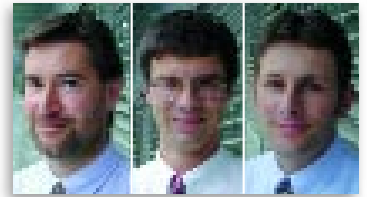
For more information visit their Web site at www.referentia.com.

Referentia

DataReturn

www.datareturn.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>



Electronic Business Systems

A Case for Messaging Middleware

SoftWired iBus provides a messaging solution for the Java platform

by Marcel Altherr, Martin Erzberger & Silvano Maffei

On April 28 of this year we attended an insightful Crossfire Webcast event entitled "Messaging vs Distributed Object Technologies" hosted by Bill Ruh from Concept Five Technologies. On the program were Richard Soley, CEO of Object Management Group, and Thomas Laffey, CTO of Talarian Corporation. Marcus Shipley, Director of Enterprise IT Architecture, USAA, was also a participant. Marcus's job was to stipulate requirements for middleware products used in enterprise application integration and advanced e-business systems development. Richard was advocating middleware solutions based on the OMG CORBA model, and Thomas was speaking in favor of messaging based on the publish/subscribe and the message queuing model.

For those of you who didn't tune into this one-hour conversation, we'll briefly summarize what we think was the essence of the debate. Electronic business systems are becoming more distributed, heterogeneous and complex. Furthermore, the demands on availability, responsiveness, extensibility and integration with existing applications are increasing.

CORBA middleware is well accepted as it provides a standard way of building distributed, heterogeneous systems using typed interfaces and a request/reply-style interaction. Messaging systems, on the other hand, provide an easy-to-understand programming model and have been successfully used in some of the most mission-critical financial information systems worldwide since the '70s.

All the participants readily agreed that both kinds of middleware are required for next-generation enterprise application integration and e-business systems, and that none of them can do the job alone. In this short article we'll talk about messaging middleware for the Java platform as folks from the Java community are probably more familiar with the ORB type of middleware than with messaging middleware. We'll talk about Sun's Java Message Service and about SoftWired iBus, a novel messaging solution for the Java platform.

What Is Messaging, Anyway?

Messaging is a model of distributed computing in which applications communicate by

exchanging self-describing message objects. A sending application can create a message object, put data such as a stock quote into the message, label it with information about its destination and submit it. The receiver of the message extracts the information and processes it.

Now there are at least two fundamentally different ways of implementing messaging middleware: message queuing and publish/subscribe. The models differ mainly in quality of service and how messages are addressed.

Both models are important because they have many application areas and are easy to understand. This allows developers to become



productive more quickly. Both models have been used successfully for many years in the areas of financial transaction processing, manufacturing automatization, real-time information delivery and systems integration.

A problem with messaging middleware is that only now are API standards starting to emerge, and many of the solutions in this area are proprietary and often somewhat outdated as they were designed years ago. Recently there has been a surge of innovation with the emergence of the Java Enterprise platform. Two such innovations are the Java Message Service (JMS) and the SoftWired iBus messag-

ing middleware. The latter is one of the few implementations of JMS done from the ground up and purely in Java.

JMS - The Java Message Service

JMS provides a standard way for Java applications to create, send and receive an enterprise messaging system's messages. JMS was developed by Sun Microsystems' JavaSoft division and is an integral part of the enterprise edition of the Java platform.

JMS covers both message queuing (a.k.a. the JMS point-to-point model) and publish/subscribe. The standard demands that at least one of the two models be implemented to claim JMS compliance.

The point-to-point model is about working with persistent queues of messages. It is point to point in that a client sends a message to a specific queue, and a receiver can pick it up from the queue. This model can be thought of as "a reliable e-mailing system for applications." The advantages are that sender and receiver need not be running at the same time and that high reliability is guaranteed as messages won't be lost in case of a failure caused by the sender, receiver or queue handler. The disadvantages are that performance suffers from the fact that all messages are passed via a queue handler and that this type of system is not well suited for fast one-to-many delivery of messages, or to transmit business events in near-realtime.

The JMS publish/subscribe model defines how JMS applications publish messages into channels and subscribe to those channels to receive the messages. This model works much like radio transmission, with senders and receivers communicating in a decoupled way using a "transmission channel" metaphor. It's a group communication model that allows a message to be sent efficiently to a dynamically growing and shrinking set of receivers. The advantage is that senders and receivers can easily "tap" into channels to exchange messages in a one-to-one or one-to-many fashion. Although the JMS publish/subscribe model provides an option for making messages persistent, it doesn't allow control over persistent messages provided by the point-to-point model.

JMS will become important by offering sophisticated messaging services to applications running as standalones or inside an EJB server.

iBus - Messaging Purely in Java

JMS is provided in the form of a specification document and a set of Java interfaces. The idea is that vendors of messaging products will provide a compliant implementation. This is what SoftWired Inc., a company specializing in Java messaging solutions for e-Business, is doing.

Our flagship product, the SoftWired iBus, is a messaging middleware written completely in Java. iBus has been available since late 1996 and is actively used in many projects worldwide. The core of iBus consists of a light-weight (a 170 kb JAR file) publish/subscribe message bus with a JavaBeans API. The JMS-compliant version of iBus provides the publish/subscribe functionality required by JMS.

A discriminating feature of iBus is versatility. The system can run atop virtually any communication protocol. iBus version 2.0 can transmit messages via reliable IP multicast, TCP and UDP. Extending iBus to use APIs for infrared or satellite communication is fairly straightforward. Although e-business systems are our main application area, iBus is well suited also for embedded systems applications.

Of course, iBus and JMS alone can't provide a complete messaging solution for e-business systems. The iBus product line thus consists of JMS, interfaces to applications written in C and C++, HTTP tunneling gateways, VPN-like bridges for forwarding messages from one intranet to another, and security based on the SSL standard.

Outlook

We envision messaging solutions such as iBus to be used as a "backplane" in building scalable, robust e-business systems that interconnect business objects running in application servers, naming services, database servers and legacy information systems, and clients running in Web browsers. Those subsystems will be implemented using a variety of techniques and standards, notably CORBA, RMI, EJB and proprietary tools. Messaging systems allow us to let those subsystems talk to each other using a natural and proven mechanism - namely, the exchange of user-defined message objects.

Resources

- iBus messaging solution (download): www.softwired-inc.com/ibus
- JMS standard: <http://java.sun.com/products/jms>
- EAI Crossfire Webcast events: www.concept5.com/crossfire
- SoftWired Inc., info@softwired-inc.com, www.softwired-inc.com

About the Authors

Marcel Altherr, Martin Erzberger and Silvano Maffei together form the management of SoftWired Inc., which specializes in middleware solutions for electronic business applications. They can be reached at info@softwired-inc.com.



info@softwired-inc.com

GET YOUR OWN!

Subscribe Today and receive the "CFDJ Digital Edition" FREE

at COLDFUSIONJOURNAL.com

1 800-513-7111
or subscribe online for faster service
subscribe@sys-con.com

GET YOUR OWN!

Subscribe Today and receive the "JBDJ Digital Edition" FREE

at JBUILDERJOURNAL.com

1 800-513-7111
or subscribe online for faster service
subscribe@sys-con.com



Fetch Parameters

Providing a reliable method to fetch and reuse your parameters

by Mark Northrup

During the development of our applets we discovered two problems. First, they all had different parameters for essentially the same action; second, the conversion of the parameters was done to varying levels of reliability. A reliable collection of methods to fetch parameters proved to be an invaluable way to ensure that all our applets used the same parameter names for the same actions and that the conversion of the parameters was done consistently and reliably.

Design Decisions

The primary objective of this class is to provide reliable reusable methods to fetch parameters. To accomplish this objective we required that the methods run and return meaningful data despite any errors in the parameters.

An assumption was made that this object would run only with applets, and that error messages, if generated, would not generally be seen. Therefore, error messages were deemed of little or no value. If a parameter isn't provided or is invalid, that parameter's method will return a default value. In addition, where it made sense, we decided that all parameter methods should return the type of the parameter, that is, a color parameter will return a color. In some cases conventions were established to reflect ease of use in the invoking code.

In the initial design we examined building a message object and associate methods with such attributes such as color, font style and direction. This proved limiting, as only a relatively small subset of our applets actually had messages associated with them. This approach could have also yielded a larger number of classes to download with a concomitant degradation in load and runtime.

The decision not to build a message object had an unforeseen ramification. If the message was not an object, then the font definition for the message would have to be done by the applet not in this object. While initially irritating, we came to view this as a feature. We discovered that while we frequently changed the font size and style, we almost never changed the typeface. We actually decided to leave it out of

the final object as it didn't seem necessary!

The color determination function proved so ubiquitous in both applets and applications that we decided to break it out as a separate object. The fundamental function of this object was to receive either a hexadecimal color code or a string for a Java-supported color and return the color equivalent of that color string.

In the initial version of this class we ran most of the type conversions "commando," without the try and catch. The assumption was that the user was a technical user and would see the error and correct it. This proved to be a poor coding technique and resulted in unpredictable behavior that was difficult to debug.

"... the START() method is called on every refresh, so a change in parameters can be updated with a refresh"

In many of the applets there was a tendency to get all the parameters in the INIT() method and store them in variables. This caused problems for two reasons. First, the INIT() method is called only once after the initial applet load, while the START() method is called on every refresh, so a change in a parameter can be updated with a refresh. This is a very useful feature when fine-tuning a Web page. Second, the parameter is stored already. There is no reason to have a second variable for the same data. When you need it, use its associated "get" method. The advantage of this approach is that the values can change "dynamically" and are only fetched if and when they are required. This approach lowers the initialization time by distributing

the parameter load time across all methods.

Determining Color

On the Web, color is everywhere. Many of our applications need and use color information as well. At minimum, an applet will need a foreground and background color; perhaps frames, panels or canvases will need additional color data as well.

A single method class to provide color determination seemed the optimal solution. The data can have only three formats and there was no easy way to separate them without analyzing the string data first. We also wanted to ensure maximum flexibility for the Web page developers by giving them as many choices as possible. This yielded a compact and efficient object that met all the criteria outlined above.

Because of the diversity of applications in which this class could be used, we elected to derive it from object rather than java.awt.Color, which seemed the most obvious point for it to extend from.

The function of this object is to translate a string that represents a color into that color. It fundamentally creates an expanded decode method that is available in java.awt.Color (Java version 1.1.x), by providing the ability to pass "#" initialized strings (as in HTML) or actual color names. Decode is not available in Java version 1.0.2, and this object must provide backwards compatibility for the most popular Web browsers. This class must implement its own decode-like logic. The original intent of the class was to work with applets. The decode method in version 1.1 supported decimal and octal options, but they won't be supported in this class.

This object contains one function, DetermineColor(String). The string may be one of three formats: a Java-supported color name, a six-digit hexadecimal number or a six-digit hexadecimal number preceded by a "#". The Java-supported color names are black, blue, cyan, dgray (for dark gray), gray, green, lgray (light gray), magenta, orange, pink, red, yellow and white. The reason behind the support of the "#" preceding the hexadecimal digits is that this is a supported format in HTML, and it would facilitate copying the colors into the parameter fields of an applet.

If the string is not a Java-supported color name, then it is tested to determine if it is a valid hexadecimal number. If so, it is converted

NetBeans

www.netbeans.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

into a color and returned. If it is not a valid number, the color is set to black. If the number is not the "proper" length, the color is set to white.

The Parameters

When we started this project we thought that the message would be the most common parameter. A survey of our applets showed that the colors and the dimension parameters were actually the most common. Following those were speed and direction.

The color parameters use the DetermineColor class to return the color passed in the parameters. The parameters are "color_t" and "color_b", and the related methods are getForegroundColor and getBackgroundColor. If the value is missing, the default foreground color is white and the default background color is black. For either parameter, if the parameter is not acceptable (as outlined above in Determining Color), the returned color will depend on the error in the parameter.

The dimension parameters are used most frequently in the applet resize method. The parameters are height and width. The related methods are getHeight(Applet) and getWidth(Applet). Each method returns an integer. If the value is missing, the default width becomes 100 and the default height is 20. However, if it's invalid, the default width becomes 150 and the default height becomes 25.

The "speed" parameter is most frequently

used in the Thread.sleep(long), which is the number of milliseconds to pause before continuing. Thus this parameter may also be used as a pause delay setting for a continuous operation. The default speed is 100 if the value is missing and 150 if it is invalid.

The direction parameters are probably the most unusual of this collection. Instead of returning a direction, it returns a signed integer. There are only three values it can have, +1, 0 or -1. These values were chosen to assist the computations for animation. We encountered scenarios in which we supported both horizontal and vertical motion; the parameters allowed us to define the type of motion for a particular implementation. Graphics areas start with the top left being 0,0. While the *x*-axis follows the algebraic notation, the *y*-axis is the opposite. It is best to think of this as an absolute value rather than signed. Horizontal scrolling is accomplished by incrementing to go to the right and decrementing to go left. Also, vertical motion is achieved by incrementing to go down and decrementing to go up.

The "message" parameter uses the getMessage(Applet) method to obtain a String message. There is no editing of the string. The default is "17 Web Place".

To date, only two font attributes were found to be of any interest; their parameters are "font_size" and "font_format". These parameters will be fetched using getFontSize(Applet) and getFontFormat(Applet) methods. Both

methods return an integer value – either the font point size or the integer equivalent to format (bold, regular, italic). The default font size is 12 if the value is missing and 10 if it is invalid. The default font format is "PLAIN".

Conclusion

This class provides a convenient, standard and reliable collection of methods to fetch parameters and use them in a wide collection of applets. The advantages of this encapsulation over a simple getParameter() method invocation are:

- All or most parameter names are consistent across all of the corporate applets.
- Standardized methods ensure reliable conversion, standard defaults and return values.
- The added flexibility leads to a more robust use of parameters than is common in most applets. 🍌

About the Author

Mark A. Northrop has 18 years of experience on MVS, TPF, VM, RSX-11M-Plus, Windows 95, Windows NT, Unix and VMS operating systems, supporting systems and application software. He is a cofounder of 17 Web Place and is currently a technical engineer at Trans-World Airlines. Mark can be reached for questions or comments at MAN_Fam@TFS.Net.



MAN_Fam@TFS.Net

JDJ Advertising

www.sys-con.com/java/adnews.htm

See JDJ Special Offer at:

<http://www.sys-con.com/java/specialsoftheweek.html>

Object Management Group

www.omg.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

SYS-CON

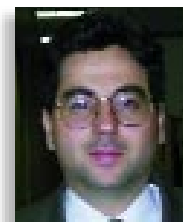
Tune into JDJ's Exclusive

See JDJ Spe
<http://www.sys-con.com/ja>

N Radio

ive JavaOne '99 Coverage

Special Offer at:
java/specialsoftheweek.html



Reflecting on XML

Use the Java reflection classes to dynamically create XML

by Daniel Rosengarten

Any discussion of the Java reflection classes first has to explain their importance and the types of tasks that can be solved with them. The Java reflection classes can help solve many generic tasks by abstracting out the task from the specific use of the task. The reflection classes provide a dynamic (runtime) way to find out information about a class: its methods, fields, inheritance, interfaces and so on. By understanding the reflection classes, many repetitive tasks can be made generic and quickly applied to a large set of classes. Some examples include automatic creation of classes to create objects from a class name, SQL statements and XML (Extensible Markup Language) generation. This article offers an overview of the reflection classes, a simple analysis and design for generating XML, an explanation of its implementation and, hopefully, some insights into Java programming.

The AutoXML class and the AutoXMLable and XMLable interfaces were written for this article, which also uses the classes Person, FullName, Address and SampleXML to demonstrate the use of the AutoXML class. My ultimate goal is to demonstrate the creation of XML for any class with the following code:

```
String myXML=AutoXML.toXML(myObject);
```

Overview of the Reflection Classes

In general, the Reflection classes help describe and manipulate a Class, a Field and a Method, and have supporting classes to describe a class Constructor, an Array and a field or method Modifier. (The class for reflection permissions and an InvocationTargetException are not covered here.)

To navigate through the many classes and methods, just remember the basics of Java: an object is an instance of a class and has a getClass() method. The object returned is of type Class. The name of the class can be obtained by using the getName() method. This is important because the AutoXML tagname in general is the same as the class name (without the package name). A class has attributes (obtained by getFields), which can then be manipulated. A class has methods that are obtained by getMethods(). A method can take parameters, obtained by getParameterTypes().

A method has a return value, which is obtained using getReturnType(). Finally, class attributes, methods and the method's return object can have modifiers (public, static, primitive, etc.) and can be obtained using getModifiers().

Unlike the other methods, getModifiers() returns an int, and you should call the appropriate static method to determine the specific property of the modifier. For example:

```
int currentModifier =
    currentMethod.getModifiers();
if (Modifier.isPublic(currentModifier))
{System.out.println("Method is public");}
```

There are many methods in java.lang.reflect for getting an int, a double, etc. The key to reducing the complexity of programs is realizing that the method get() returns an object and that String.valueOf() can handle all of the nine primitive types.

Task Description: XML

XML is a format that describes data and is becoming popular on the Internet. XSL (Extensible Style Language) is used to change the presentation of the XML data. For more information on the specifications and benefits of XML, check out www.w3c.org/.

The XML data is between a "tag" that describes the type of Data. The code in Listing 1 might be used to describe a class called FullName having attributes of FirstName, MiddleName and LastName. Contained objects have their respective XML within the tag of the container. Expanding the above example to describe a Person might look like the code in Listing 2.

This fits into an object-oriented framework quite easily. Each class only needs to know how to generate its own XML piece according to the following procedure (assuming each class has a method called toXML() in a method called toXML()):

1. Put classname between beginning tag delimiters.
2. If the attribute is an array, iterate over the entire array for steps 3-7.
3. For each attribute, put the attribute name and optional indices within a tag.
4. If the attribute is a primitive type, append the value of the attribute.
5. If the attribute is a complex object that is

XMLable (implements XMLable), then call the object's toXML method.

6. After each attribute, append an end tag for the attribute.

7. After all attributes are XML'd, append the end tag for the object.

The drawback of this direct implementation approach is that someone needs to look at each class that might be affected and at the possible attributes and contained objects, code the routine and test it.

Another way to approach the task is to look at a specific class, determine whether it should be included in the XML output and go through each public accessor method to create the XML output. This approach requires coding and testing one class and an interface. The drawback is that each object is inspected at runtime to determine the attributes and objects to include. The usefulness of this approach is that adding automatic XMLability to a new class is trivial.

Another approach attacks the problem from the source side. You can parse through each class (either by file or by class), find the class definitions, build the toXML() and append the code to the class definition.

These two approaches highlight the classic problem of whether to build it now (at compile time) or build it later (at runtime).

This article takes the dynamically generated approach, using the methods to get the XML information. The methods could also have been written to use the class attributes, but this doesn't demonstrate enough of the Reflection classes. The procedure to dynamically create XML code is similar to the procedure to hard-code XMLability, except when dealing with embedded arrays and collections. The JDK 1.2 has the interface Collection, which unified accessing Vectors, Lists, and so on. This interface has the method toArray(), which returns an array representation of the Collection. To write a routine for arrays, you need to be aware that Java arrays don't have to be symmetrical. In Java you can have an array that can look like the array in Listing 3. This Array doesn't have memory allocated for myArray[1][2], myArray[1][3], and myArray[2][3] because those elements don't exist.

To create the XML for an array of unknown dimensions, a recursive method called parseArray was added. It was the trickiest part of the AutoXML class to get right. This recur-

PointBase

www.pointbase.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

sive type routine is necessary to iterate over all of the objects.

Another point to mention about the design of AutoXML is that someone else might want to implement his or her own toXML() in certain classes. Automatic generation might be priceless at development time, but if performance needs to be improved, it's important to provide a way. By adding an interface XMLable, the AutoXML class now understands two interfaces: AutoXMLable and XMLable. If the XML of class is to be automatically generated, the class should implement AutoXMLable. If the class has its own implementation, XMLable should be used. Finally, if the object's class doesn't implement either interface or is null, it should be skipped.

When trying to build XML, it's important to have the ability to select the classes. Security classes are a good example of classes that shouldn't be able to be used automatically by the AutoXML class. This is accomplished by using two interfaces: AutoXMLable and XMLable. If the class implements XMLable, the AutoXML class will call toString(). If the class implements AutoXMLable, the AutoXML class will build the XML for the class. If the class doesn't implement either interface, the class will be skipped and an empty string will be returned.

Another area that deserves attention is the invoke method in class Method. This allows a function to be dynamically called on an instantiated object. This is used to get the objects contained in the current object and could cause a potential problem if the class has a function similar to the one below:

```
public getNextNumber() {return
    ++currentNumber; }
```

The AutoXML class won't be aware the object is being changed and will go merrily on its way, so be careful about method names. A better name might have been nextNumber().

Example

The example is a set of very simple classes for Person, FullName and Address. The Sam-

pleXML class demonstrates the ease of use of the AutoXML class. The XML is given in Listing 4. Person and Address implement the AutoXMLable interface, allowing AutoXML to create the XML. FullName implements XMLable to provide its own way of creating the XML tag. The classes have been kept simple for demonstration purposes, but feel free to change the example classes. Change getAddress() into an array... make two-dimensional arrays of FullName. Experiment with the class definitions and see how the XML is generated.

A Walk in the Code...

The program begins in SampleXML::main(). This creates a Person named John Doe having two addresses.

- Next, AutoXML::toXML() is called (passing the object to be AutoXML'd) and outputs the results.
- toXML() first checks to ensure that the object passed is not null, gets the name of the object's class name, strips the package name, calls parseClass and then wraps the result in a tag.
- parseClass() iterates over the methods in the class to determine if the method will return a useful piece of data. Following bean-naming guidelines, only the prefixes get and is (if the return type is Boolean) are understood. Also, the method must be public nonstatic. The method name is stripped of its prefix. If the return parameter is a primitive, a string or date, the tag is created for the attribute; otherwise the returned object is broken down into its subcomponents (in expandClass()) or array elements (parseArray()).
- expandClass() if the return parameter is AutoXMLable, XMLable or Collectable. The method is invoked and the result is sent to toXML()
- parseArray() takes an n-dimensional array and recursively reduces the dimension of the array until it is a one-dimensional array that can be iterated over. If the result isn't primitive, the array object is sent to toXML().
- wrapTag() puts a tagname into parameters,

and information into an XML tag.

What Isn't Covered in the Example

The reflection classes have a few very useful methods that weren't needed for the AutoXML class. The most powerful, Class.forName(String className), is a simple method that returns the class object represented by the String passed. In combination with the newInstance() method, an object can be created (calling the constructor without parameters) by knowing only the name of the class. This opens the door for dynamically adding functionality to a program without recompiling. But that's another article.

I also didn't cover security, determination of inheritance structure, class loading or hashing capabilities. Nor did I cover XSL, which describes the formatting of the XML data. Microsoft has a tool to combine XML and XSL to create HTML output. An equivalent AutoXSL class that is similar to the AutoXML class can be created. The difference is that XSL would create tags without the data being embedded. Listing 4 shows an example of XSL. Try to write the class yourself, using the AutoXML for the algorithms.

Conclusion

As the example shows, the reflection classes are extremely useful for adding functionality to a program. With these classes you can easily create generic routines, scripting engines and more dynamic programs. 📧

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

About the Author

Daniel Rosengarten, a senior systems analyst at Sanford C. Bernstein, is involved in the development of several OO financial systems using C++, PowerBuilder, Visual Basic and Java. A Sun-certified Java developer, he has an MBA from the University of Arizona and a BS in electrical engineering. Daniel can be reached at 70451.2212@compuserve.com.

 70451.2212@compuserve.com

Listing 1: Sample XML code for object

```
<FULLNAME> <FI RSTNAME>Dani el </FI RSTNAME> <MI DDLENAME>Joel <MI DDLENAME><LASTNAME>Rosengarten</LASTNAME></FULLNAME>
```

Listing 2: Sample XML code for objects with arrays

```
<Person><Addresses
index1="0"><Address><Ci ty>AZ</Ci ty><Street>123 Mai n
St. </Street><Zi p>85715</Zi p></Address></Addresses><Ful l Name>Jo
hn Doe</Ful l Name></Person>
```

Listing 3: A Java Array

```
myArray[1][1] = 1
myArray[2][1]=201
myArray[2][2]=202
myArray[3][1]=301
myArray[3][2]=302
myArray[3][3]=303
```

Listing 4: SampleXML class output

```
<Person><AddressArray
index1="0"><Address><Ci ty>AZ</Ci ty><State>Tombstone</State
><Street>1313 Mocki ngbi rd
Ln</Street><Zi p>85500</Zi p></Address></AddressArray><A
ddresses
index1="0"><Address><Ci ty>AZ</Ci ty><State>Tucson</State><Stree
t>123 Mai
n St. </Street><Zi p>85715</Zi p></Address></Addresses><Addresses
index1="1"><Addre
ss><Ci ty>AZ</Ci ty><State>Tombstone</State><Street>1313 Mock-
i ngbi rd Ln</Street><Z
i p>85500</Zi p></Address></Addresses><Ful l Name>John Doe</Ful l -
Name></Person>
```


Allaire

www.allaire.com

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Building Enterprise Portals

Finally, a way to think about how complex technology and business tie together cohesively



by Jeremy Allaire

Every year I find myself contemplating the dramatic changes in the Internet industry over the previous year. And every year the changes seem more dramatic, more exciting – and, most important, clearer. Everyone involved in the Internet industry does the same thing, I'm sure, and as part of this ongoing reflection we try and find meaning in a few major concepts to help us grapple with all the change and opportunity. These concepts typically end up in buzzwords that we internalize and then attempt to indoctrinate our peers (and customers) with this new understanding.

For example, in 1993, "the Internet" meant "Online Services," which meant AOL, CompuServe and Prodigy. This was about as complex an idea as people could come up with, and given how vague and overwhelming it all seemed, that was probably okay.

So it was for the next six years. Each year we in the industry tried to figure out what it was all about, building products and services to support it, and, most important, giving it a name, or a concept. We need (and needed) these keywords and concepts, which we used to map complex technology to business or consumer reality.

In 1994 and 1995 it was the "Internet," which meant "wild-wild-west," "experimentation" and "new opportunities," but that's about it. After people (including many of you) began to really work with the technology, they figured out there were opportunities beyond building brochure-ware Web sites. They saw there were real opportunities in automating a business and leveraging the incredible cost economies of the Web computing model, and in reaching new customers. Thus they were dubbed "e-commerce" and "intranets." People finally understood this computing model wasn't about brochures, but about changing how information and products were used, created and sold. In fact, for quite some time, most companies focused exclusively on intranets and the unbounded opportunities that lay therein.

Sometime in late 1996, and into 1997, the concept of the "extranet" emerged. This was a site that was NOT an Internet site, but also NOT an intranet site; still deployed on the Internet, but secured like an intranet. Whoa! But it did have meaning, and that meaning was that you could use the Internet to tie together companies and organizations in a way that wasn't possible before.

Since 1997 and through 1998, we've taken these concepts and run with them. We've built products and services that enable these terms and concepts and we use them fluidly in selling what we do to our customers.

During this time some pretty significant things have been happening in our industry, all of which bode well for our future. First, corporations are finally "figuring out" the value of the Web. After years of experimentation, working with early adopter technology and through the inevitable pressures of customer feedback and competitive threats, many corporations are now at a point to fully embrace the Web business and computing platform throughout their business, even reinventing their business around this new economy. Second, we've seen the spectacular success of the most visible Internet companies, the e-commerce portals and "dot com" players. We're seeing this, aspiring and learning, and realizing that the earth is moving beneath us. And finally, all of this hard work and learning is establishing best-practices in both the technology infrastructure and the business and organizational models. Indeed, these three trends, happening over the past year or so, indicate that we're on the verge of a massive mainstream explosion in adoption of the Internet.

Out of all this, of course, must come a new organizing concept, one that ties together everything we've learned and becomes the basis for how the mainstream thinks about and creates the next wave of Internet business. I believe the organizing concept we'll come to know, speak and understand is the "Portal." Everywhere I go I hear customers talk about building a "Family Portal" or a "Kids Portal," and, most recently, and perhaps most significantly, an "Enterprise Portal." Increasingly, it seems, "Portal" has come to mean "successful Internet site" or "successful Internet business," and no longer carries its original meaning, "search engine." I strongly believe that the Enterprise Portal is the right organizing concept for furthering our work.

Understanding Enterprise Portals

An Enterprise Portal is the combination of software and technology infrastructure, new business models and new organizational structures that combine to create an Internet-centric business. In short, an Enterprise Portal is what companies need to build in order to become Internet-centric companies. In my estimation, Enterprise Portals reflect three distinct observations about the Internet business landscape.

The first is the realization that truly Internet-centric companies don't view their Internet, intranet and extranet systems as separate. Instead, they see the pervasiveness of the Web simply as a part of their business. An Enterprise Portal represents this idea. Your Web systems become a portal to your entire business – internally, for your employees managing their work, then extending out through private and secure interfaces to customers, suppliers and partners. The Enterprise Portal allows us, for the first time, to think of the Web as the fabric of our business, and recognizes the reality that Web systems aren't about isolated corporate


applications, but about an overall approach to doing business in the Internet economy. Indeed, with Enterprise Portals the Web becomes your business, and your business becomes the Web.

A second observation is that Enterprise Portals, by being modeled on the best-practices of "dot com" companies, pave the way to understanding the four broad solution components in running an Internet-centric business: rich content, e-commerce, customer interaction management and collaboration. All successful portals center around a rich-content application infrastructure, including models for dynamic publishing, workflow, roles-based security models and content asset management. Likewise, they tie in commerce systems, including Web transaction management infrastructure, as well as common systems for merchandise management and order processing. Finally, they bridge these components with applications that enhance end-user and customer experience, such as personalization, user forums and collaboration tools for document management, threaded discussions and managing projects over the Web.

The third key observation is that Enterprise Portals represent the technology and business best-practices in the Web environment. They demonstrate that there are well-observed user interface models, known systems and development architectures, and – most important – Web-native business and organization models. This shift toward best-practices-based approaches is critical as mainstream companies look to scale their Web efforts to enterprise-wide levels.

Conclusion

Enterprise Portals give us a model for our businesses. At Allaire it's driving us to build a comprehensive platform that spans visual tools, application servers and packaged systems for building and managing an Enterprise Portal. For corporate customers it provides a model and call to arms to think about their business and technology strategy in the Internet age. For solution companies it should drive new solutions based on best-practices in Enterprise Portals.

In any case, we have a new mantra, a new buzzword and, finally, a way to think about how these complex technology and business issues tie together in a cohesive manner, driving forward the next generation of the Internet economy. 

About the Author

Jeremy Allaire is a cofounder and vice president of technology strategy at Allaire. He helps determine the company's future product direction and is responsible for establishing key strategic partnerships within the Internet industry. Jeremy has been a regular author and analyst on Internet technologies for the past seven years, and he holds degrees in both political science and philosophy from Macalester College.

jeremy@allaire.com

Elixir

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>

Object

www.objectspace.com

See JDJ Special
<http://www.sys-con.com/java>

Space

[.com/go/universal](http://java.com/go/universal)

Special Offer at:
java/specialsoftheweek.html

JProbe

www.klggroup.com/culprits

See JDJ Special Offer at:
<http://www.sys-con.com/java/specialsoftheweek.html>