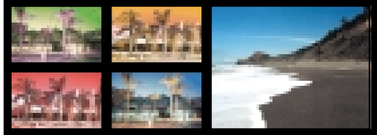
WEB SERVICES EDGE 2001 SANTA CLARA, CA OCT 24–25

JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

October 2001 Volume: 6 Issue: 10

JAVADEVELOPERSJOURNAL.COM



OCTOBER 24–25, 2001

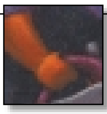
web services EDGE™ conference&expo

OCTOBER 22–25, 2001

XMLEDGE™ conference&expo 2001

SANTA CLARA CONVENTION CENTER, SANTA CLARA, CA

James Gosling in New York City delivering his JDJEdge keynote address to the international Java community



RETAILERS PLEASE DISPLAY
UNTIL DECEMBER 31, 2001
$5.99US $6.99CAN

SYS-CON MEDIA

SUBSCRIPTIONS:

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

SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM

COVER PRICE: $5.99/ISSUE
DOMESTIC: $49.99/YR. (12 ISSUES)
CANADA/MEXICO: $79.99/YR. OVERSEAS: $99.99/YR.
(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: $10/EA., INTERNATIONAL $15/EA.

EDITORIAL OFFICES:

SYS-CON MEDIA 135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9600

JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is published monthly (12 times a year) for $49.99 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals postage rates are paid at Montvale, NJ 07645 and additional mailing offices. POSTMASTER: Send address changes to: JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

ALAN WILLIAMSON EDITOR-IN-CHIEF

Java XP

Last month I wrote on the Microsoft issues with their impending release of Windows XP. I intended to go a whole editorial without mentioning *that* software company, but because of the amount of e-mail I received, I have to do something.

It appears that, as with any good news story, it comes with a fair share of disinformation. Every news source was reporting some aspect of the XP–Java saga and what it meant for the Java community. A lot of it was simply nonsense and I have to thank you for sending me posts from the "we-love-Microsoft" sites that were just downright comical. If you had the misfortune to read any of them, you'd get the idea that Java is on its way out, and that it was just a marketing gimmick from Sun. Gosh, that would be one hell of a gimmick! Even by Sun's standards.

Fortunately, nothing is ever as bad as it sounds, and we here at ***JDJ*** want to make sure our readers are kept informed of the whole debacle. You deserve the truth, and we're here to deliver. We needed someone to tell our readers what the future of Java holds and cut through all the Microsoft marketing anti-Java hype.

Who else but Scott McNealy, CEO of Sun Microsystems, Inc., could tell you how it is? Scott has never been one to shy away from a little Microsoft bashing, but when we asked him to write our Guest Editorial, he kindly agreed and put together a rational and wonderfully level-headed word of comfort for our readers. Read what Scott has to say on the next page.

On another note of Java news, I see that we might be losing one of the heralded J2EE licensees. This morning HP and Compaq announced they would be merging, creating a company that would be just slightly smaller than IBM. Early reports indicate that Carly Fiorina, present CEO of HP, will head up the consolidated company, with HP shareholders taking a 60% share of the new company and Compaq shareholders, 40%.

As with the XP stories, there's already a raft of postings concerning what's going on behind the scenes. Most of it is nonsense, I'm sure. One thing that is true, though, is a consolidation of both companies' Java offerings. It's way too early to be talking of such things, but be assured, we'll keep you up to date with any information we receive.

• • •

It's been a bad month for the embedded world, especially the Japanese market, which lost somewhere around 60,000 jobs from the major chip manufacturers. Apparently, the slowdown in demand for mobile phones, coupled with falling PC sales, is beginning to make its mark at all ends of the manufacturing process. Gateway recently announced the closing of its European operation in Ireland.

I asked our resident J2ME guru, Jason Briggs, what this means for the J2ME marketplace. His reply was that we wouldn't see a great pickup of Java-enabled devices for a while yet. He did note that it would probably sneak up on us. We'll suddenly turn around and discover that the device we're using runs on Java and we didn't even know it. Which is exactly what you'd expect.

At the end of the day we should be marketing our solutions, not the fact that they're implemented in Java. Who cares? As long as it does what we want it to, the underlying technology shouldn't matter to us as consumers. (Oh, before I forget, Jason, can you put me down on the list for one of the devices you mentioned in your editorial?)

Java is going through an interesting time in its development life. More and more people are now enjoying its benefits without ever knowing that their devices are Java powered. This is the way it should be, and we need to start moving away from simply heralding "Java-powered" whatevers in our marketing and focus more on the solutions Java is attempting to deliver.

alan@sys-con.com

AUTHOR BIO

Alan Williamson is editor-in-chief of Java Developer's Journal. In his spare time he holds the post of chief technical officer at n-ary (consulting) Ltd (www.n-ary.com), one of the first companies in the UK to specialize in Java at the server side. Rumor has it he welcomes all suggestions and comments.

WRITTEN BY SCOTT MCNEALY

The Platform of Choice

It's becoming more obvious to me every day that Java technology is the platform of choice, in more ways than one.

In the traditional sense of the phrase, it's the answer to the questions all developers ask themselves: Which platform should I develop to? Which has the biggest market and the most influence? Which provides me with a means to demonstrate my skills and creativity?

There's a second meaning as well, another question to answer: Which platform provides developers, customers, and users with more choice?

In this second meaning, "platform of choice" indicates one that runs on multiple operating systems, addresses a wide spectrum of hardware, and can spawn multiple implementations so users are free to select the one that works best (yours, of course).

By either definition Java technology is still the platform of choice, and this works to your benefit – big time. The Java platform addresses opportunities ranging from the biggest servers available to the desktop to millions of new client devices – smart cards, mobile phones, pagers, PDAs, TV set-top boxes, and automotive systems. (Clearly the definition of "client" is due for an update, too.)

Java technology is really hitting its stride now. Within six months more than six million phones enabled with Java technology have been sold in Japan, where the three largest wireless providers – NTT DoCoMo, J-Phone, and KDDI – have deployed interactive wireless services. One analyst estimates that more than 700 million JVMs will be deployed between now and 2005 on these new devices.

In the server market we're seeing choice like never before. There are 17 different applications servers available today that are compatible with the Java platform. Some are the underpinnings of IDEs, but even then IT professionals can choose the implementation that best suits their needs. And if the ISV they choose falls behind, they can replace one app server with another at a dramatically lower cost of switching.

More than 400 major corporations are working together through the Java Community Process to innovate, enhance, and expand Java technology. Offerings from Sun, IBM, HP, BEA, and Oracle, to name a few, are all built around the platform of choice.

The lonely exception, of course, is Microsoft. Unless you've been totally out of touch since mid-July, you're aware that Microsoft is not shipping a JVM with Windows XP. The reason? Microsoft has used every excuse from "the lawsuit settlement required us to do this" to my personal favorite, "We don't want Windows to have too much code." The real reason, I suspect, is that Microsoft is anti-choice.

But, at the end of the day, you'll want to know whether this move will lessen the market for your products and impact your livelihood. Rest assured that we will provide XP support. It will be the latest Java technology, not something old. We'll distribute it via PC manufacturers, Web sites, ISVs – whatever it takes to ensure that your Java apps will run anywhere.

So keep on writing to the platform of real choice, the one that's here today: the Java platform.

scott.mcnealy@sun.com

AUTHOR BIO

Scott McNealy is the chief executive of Sun Microsystems.

READERS' RESPONSE

Early online reactions to Scott McNealy's Guest Editorial. Add your feedback online at JavaDevelopersJournal.com.

Keep Up the Good Fight

Is the JVM still in the IE6 browser? If not, that would break a lot of Web sites and cause much (more) anti-Microsoftism...

I'm a multimedia developer currently learning Java and Java3D. Keep up the good fight. The relentless always win.

Wallace Jackson CEO
Mind Taffy LLC
walls@aperion.net

Microsoft Shoots Self in Foot

Java is the ideal platform for software because a program written in Java can run on multiple platforms with little or no modifications. Since Microsoft is not including a JVM in Windows XP, a void beckons to be filled. Personally, I view Microsoft's obsolete implementation of the JVM as an impediment to furthering the Java platform.

Computer builders would be wise to include an up-to-date JVM with the systems they sell.

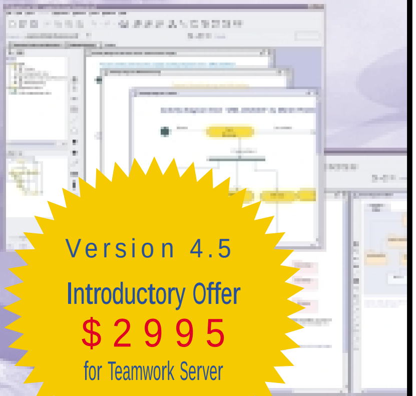
Computer buyers should ask for the JVM on the machines they buy. If the computer makers don't include a JVM, they're short-changing the customer and the customer should take his or her business to one that does.

Including the JVM is a great way for computer manufacturers to differentiate themselves from those who don't.

Peter English
PeterEnglish@yahoo.com

magicdraw™

The rational alternative



Version 4.5
Introductory Offer
$2995
for Teamwork Server

All 9 UML diagrams

Additional Features:

- Performs Java, C++ or CORBA IDL code round-trip engineering (code generation and reverse engineering); recognizes JavaDoc comments. This feature allows you to write code, reverse engineer, make changes to the model and re-generate the code without losing any implementation specific information.
- Supports UML 1.3 notation.
- Saves diagrams as bitmap PNG/JPG and scalable WMF/SVG/EPS/DXF formats.
- Provides XML interoperability – native model files are stored in XMI format.
- Integrated with Forte for Java (FFJ) IDE versions 1.0 and 2.0.

Standard Edition:	***$249***
Professional Edition:	***$499***

Floating License . . $100 additional

Upgrade for a $69 annual fee!

No Magic...
Just Technology That Works

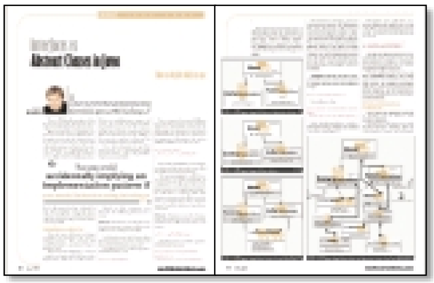
Download fully functional demo at:
www.magicdraw.com

contact us at
contacts@magicdraw.com

100% Pure Java Application

New Format

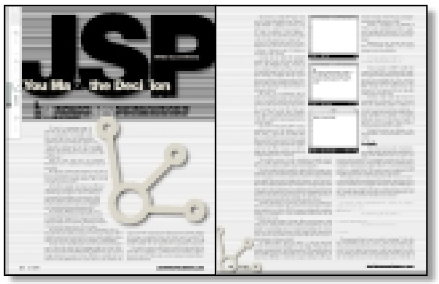
JDJ is a wonderful publication and the new format makes it even better. I'd like to see a section in each issue devoted to new and less-experienced Java programmers. "Interfaces vs Abstract Classes" by Anthony Meyer was a wonderful example of this type of article [Vol. 6, issue 4]. I handed it out to students in my last Java class because it really explained many things.



Some proposals are:

1. How and where to do validity checking on data that's entered through Swing components.
2. Which Layout managers do programmers "really" use? Do they use the Swing classes or a bought-in Layout Manager?
3. Do experienced programmers really use threads?

Douglas Maughan
dm@wdmaughan.fsnet.co.uk



Comments on Jon Stevens' Article

I just read Jon Stevens article about JSP, Struts, Velocity, etc., ["JSP: You Make the Decision," Vol. 6, issue 7]. I was really pleased that I'm not the only one who doesn't like JSPs. Great article!

Couldn't have said it better... :-)

Andreas Prohaska
ap@apeiron.de

I just finished reading Jon Stevens' article and I wanted to write you a quick thanks. You did a great job of explaining the differences between JSP and Velocity and it was a great read. I've been using JSP from the beginning and have been frustrated when working with designers because we couldn't expect them to learn JSP; they would give us HTML instead and we would have to turn that into JSP "and" do the back-end logic. Velocity does seem to provide a very nice alternative to this so I'm going to look into it now. Thanks for taking the time to write a straightforward comparison.

Kabriel Robichaux
kabriel@netobjects.com

Chat with James Gosling

Thanks for the good interview with James Gosling ["Real-World Java with James Gosling," Vol. 6, issue 8]. He references a book, *Effective Programming*, but I can't find it anywhere. Are you sure you have the correct title?

J. David Beutel
jdb@vgkk.co.jp]

Alan writes:

The book is Effective Java Programming Language Guide *(The Java Series) by Joshua Bloch, Addison-Wesley Professional; ISBN: 0201310058.*

J2EE Security Model

I read Sanjay Mahapatra's article, "J2EE Application Security Model," [Vol. 6, issue 8]. While implementing some projects using some of the more popular app servers, we've run into some limitations of the security model. While the container security descriptors are declarative, we find we need finer-grained support. It's one thing to protect certain apps based on the user's generic role, but our apps tend to have user-specific permissions, and the declaration isn't flexible enough.

In your example in Listing 3, you restrict the use of the CustomerServicesEJB to entities in the role of "customer." We want to go the next step. For instance, we would want to restrict the customer "Bob Hoozit" to viewing only his orders and his invoices.

Are we expecting too much from the J2EE security model or should we be looking at this in a different way?

Steve Suehs
ssuehs@lgc.com

Sanjay writes:

The scenario you mention: "restrict the customer 'Bob Hoozit' to viewing only his orders...," is relatively easy to implement using the current J2EE security model.

Let's say the customerID/loginID (also the name of the security Principal) for customer Bob Hoozit is: bhoozit.

1. Order creation

When creating a new order, we'll need to persist to the database, the loginID field, along with the other fields that comprise the order. To accomplish this with container-managed persistence, code similar to the snippet below should be used within the ejbCreate() method of OrderEJB.

```
this.loginId=theContext.getCallerPrinci
pal().getName() ; //
```

The above saves bhoozit in the loginID field, if bhoozit was logged in and created this new order.

2. Order retrieval

Now let's say the CustomerServicesEJB has a method retrieveOrders() and the OrderEJB (entity) has a findByCustomer() method. The CustomerServicesEJB::retrieveOrders() method will in turn need to make a method call similar to:

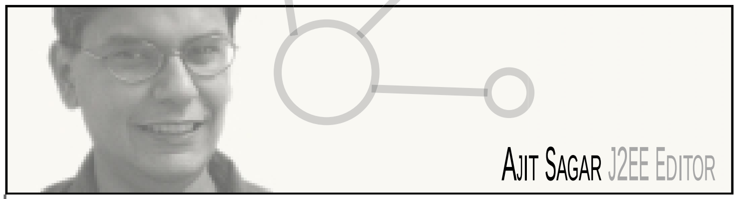
```
findByCustomer(theContext.getCallerPrin
cipal().getName() )
```

on the remote reference for OrderEJB. The above will cause the invocation of findOrdersByCustomer with the argument (bhoozit), if bhoozit was logged in and attempted this order retrieval.

This will result in each customer being able to see only his or her own orders, with just a couple of lines of code.

Hope this helps.

Sanjay Mahapatra
sanzem@excite.com



AJIT SAGAR J2EE EDITOR

Plug and Plug and Plug and Play

I recently got the security card for entering our Pennsylvania offices. The badge attaches to a hook that's linked to the string that goes around my neck so I can carry the badge at all times. I'm sure you know the type of setup I'm talking about. The JavaOne badges handed out this year were of the same type. As I examined this "gadget" more closely, I noticed that the hook is attached to a clasp, and the clasp snaps into a socket at the other end. Another piece allows me to tighten or loosen the string around my neck.

For a simple security badge holder, this gadget is really quite sophisticated. All the pieces fit together so well. At first I was quite impressed by the design. You can detach the badge, the clasp, the string – all the pieces that connect so well to each other. Then I started wondering why the manufacturer had designed such a complex system for such a simple application.

In the software world plug-and-play is a great concept. Good architectures should always be designed with component reuse in mind. However, overzealous ones often have a tendency to overdesign for reuse. It's very important to rein in the design into the context of the real business problem that applications built on the architecture try to solve. J2EE is a great platform for component-based development and promotes the design of reusable components. Along with the APIs, it offers guidelines and best practices for plugging together components to build complete applications. Design patterns should be carefully examined before applying them to a particular application. In the real world, performance, timing constraints, high availability, and other factors become additional criteria for designing a feasible solution.

For example, recall last month's discussion on the connectivity to EIS sources from the J2EE platform. The J2EE Connector Architecture (J2EE CA) provides connectivity via resource adapters to legacy, ERP, and CRM systems. However, this only supports synchronous communications. Support for asynchronous communications is expected in the next release of the spec. An alternative for asynchronous interaction is to use message-driven beans, that is, JMS. This is the advantage of open, pluggable APIs. However, mixing the two paradigms will have repercussions on the application. Issues such as performance and transaction management will have to be addressed by the application designer. The alternative in the market is to use a direct ERP adapter, such as one from WebMethods or Attunity. While WebLogic and WebMethods are working on a relationship that enables WebLogic's implementation of J2EE CA to utilize WebMethods' platform, in addition to the issues of performance, transaction integrity, and security, application designers will also have to consider the licensing and total cost of ownership (TCO) of using two application environments.

Another example is the EJB model. Until EJB 2.0, whether a component was local or remote, it was addressed via remote interfaces, which meant there was always a network hop (or RMI call) to get to an EJB component. Although application server vendors offered optimization for local component access, the spec didn't enforce this. In other words, you could plug in any component and it would act as a remote component. Luckily the local interfaces in EJB 2.0 address this problem.

Speaking of local interfaces, we have an excellent article in this issue by Alex Pestrikov. In addition to our bimonthly column on Core J2EE Patterns, the feature this month is an article on the MVC pattern in a Web application. Add to this our regular J2EE FAQs and some top-notch articles and you have another great issue.

ajit@sys-con.com

AUTHOR BIO

Ajit Sagar is the J2EE editor of JDJ and the founding editor and editor-in-chief of XML-Journal. A senior solutions architect with VerticalNet Solutions, based in San Francisco, he's well versed in Java, Web, and XML technologies.

J2EE INDEX

Local Entity Beans and Relationships

Exploring the many-to-many relationship

WRITTEN BY
ALEX PESTRIKOV

As the EJB 2.0 specification has entered its final stage, many companies are in the process of building server-side J2EE applications. The final draft of the specification has made container-managed persistence (CMP) of entity beans complete and more powerful.

Significant changes that were made to the specification in the area of entity beans include:

- ***Local entity beans*** are lightweight entity beans that exist only locally, inside the EJB container, and therefore don't need to be marshaled.
- ***EJB QL*** helps create specialized queries declaratively.
- ***Declarative entity bean*** relationships have been added.
- ***Dependent object classes*** were removed from the latest draft in favor of local entity beans.

These features are essential for real-life applications. Another exciting development is that the latest version of BEA WebLogic application server supports the draft specification in all of the preceding features. In this article I will go through an example application to demonstrate local entity beans and entity bean relationships; namely, a many-to-many relationship. I'll also create deployment descriptors for BEA WebLogic Application Server 6.1 and write a build-script to generate a deployable archive.

Remote entity beans encapsulate coarse-grained persistent objects, whereas local entity beans target fine-grained persistent objects. Local entity beans differ from remote entity beans in this manner:

- The bean interface extends javax.ejb.EJBLocalObject; the home interface extends javax.ejb.EJBLocalHome.
- The bean is never transmitted over the network. Thus its methods will not throw a java.rmi.RemoteException.
- The bean is collocated in the same JVM and its method parameters are passed by reference.

Although the last item exposes the implementation details about the object, this optimization is well deserved. Indeed, it's only logical that some objects are never accessed remotely and therefore shouldn't incur the overhead associated with the remote call. Often the local entity bean may be accessed through a facade session bean. In this case the remote clients will pass value objects to the facade session bean. This bean will then perform business logic that may involve other session beans and entity beans. The facade bean will call the local entity beans when the data needs to be persisted. (More about this later.)

Another important feature of CMP is declarative entity bean relationships. The specification allows a declaration of the following things about a relationship:

- ***Cardinality:*** One-to-one, one-to-many, and many-to-many relationships are supported.
- ***Direction:*** Unidirectional and bidirectional relationships are supported.
- ***Cascade deletes:*** If a relationship is marked for cascade delete, the dependent entity beans are removed when the main entity bean is removed.

You still have to provide abstract get and set methods in the bean class to reflect cardinality and direction. For

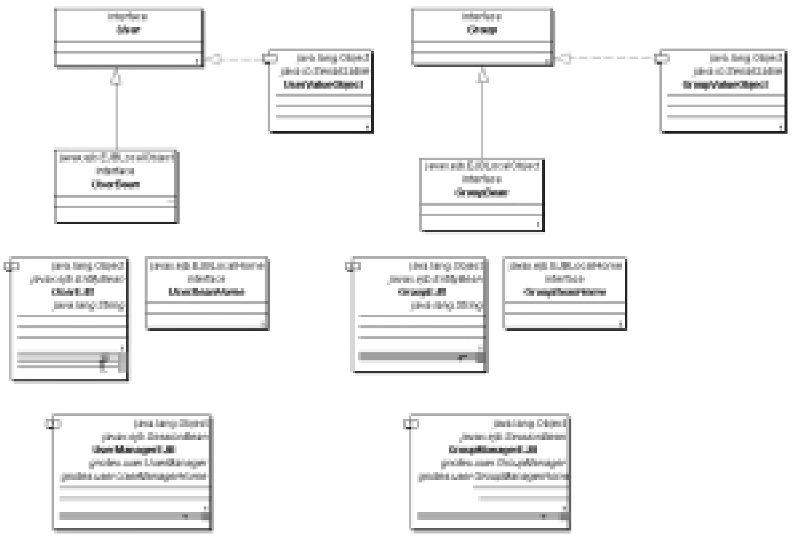


FIGURE 1 Class diagram for the example application classes (remote and home interfaces for facade beans not shown)

example, if an Order bean has many Order Line Items, then you must provide abstract getLineItems and setLineItems methods in the Order class to specify a unidirectional, one-to-many relationship.

How It's Done

Let's write a small application that utilizes local entity beans and relationships. My example will run on BEA WebLogic 6.1 Server and an Oracle database. I'll have two entity beans – User and Group. User can belong to multiple groups and a Group can have multiple users. Both beans are local entity beans and have two session facade beans: UserManager and GroupManager. I also want to navigate in both directions. Thus our entity beans have a many-to-many, bidirectional relationship. Additionally, I'll have two value objects that are sent to clients by the facade beans: UserValueObject and GroupValueObjects. The value objects allow clients to cache and use objects multiple times without remote calls. Figure 1 illustrates the classes.

In Figure 1 the entity beans' interfaces extend javax.ejb.EJBLocalObject and the entity beans' home interfaces extend the javax.ejb.EJBLocalHome interface. Another important aspect is that the value object must implement the java.io.Serializable interface for it to be sent over the network.

The entity beans have abstract get and set methods for all fields that need to be persisted. The business logic to create, update, and delete the entity beans resides in the facade session beans. Here's how the User Manager bean creates a User in the createUser method:

```
InitialContext ctx = new
InitialContext();
UserBeanHome home =
(UserBeanHome)ctx.lookup("User");
UserBean bean =
home.create(user.getUserName());
```

The Group Manager bean is responsible for adding and removing Users from Groups. Adding a User to a Group is as simple as adding a user bean to a collection. This is done in the addUserToGroup method. I had to declare this method to require a transaction in the ejb-jar.file because the collection must be retrieved and altered in the same transaction.

```
UserBean userBean =
userHome.findByPrimaryKey(userName)
;
GroupBean groupBean =
home.findByPrimaryKey(groupName);
Collection users =
groupBean.getUsers();
users.add(userBean);
```

Once I have the classes, I can write the deployment descriptors. First I write ejb-jar.xml the same as for any EJB container. Listing 1 describes a Group–User relationship.

Ejb-jar.xml descriptor declares the entity beans and relationships, whereas the actual mapping-to-storage mechanism is done in the EJB container–specific deployment descriptors. BEA WebLogic 6.1 uses two files for that purpose. The first file, weblogic-ejb-jar.xml, specifies a persistence descriptor as a part of the Weblogic-enterprise-bean element. It also specifies the name of the second descriptor file where the field mapping is done. Listing 2 shows the complete Weblogic enterprise bean element.

I specify how the fields are mapped to the table columns in the other WebLogic descriptor, weblogic-cmp-rdbms-jar.xml. I also specify how the Group–User relationship is mapped to the database through a USER_GROUP table. Listing 3 shows the relationship portion.

Once I've written the deployment descriptors and implemented the beans, I write a build script to build my application. Since EJB applications have a lot of different pieces to them, it's a good idea to create a build script as soon as the skeleton of the project is in place. I use a utility Ant to do all my builds. Ant can create directories, move files, compile, JAR, and deploy your J2EE application. It can even create database schema and run unit tests for you.

Another good thing about Ant is that the scripts are written in XML, which seems quite natural since we spent all this time with XML writing the deployment descriptors. WebLogic Server 6.1 comes with a number of examples that have Ant build scripts and can serve as templates. Listing 4 shows the fragment of the Ant script that calls the WebLogic EJB compiler to create a final deployable .jar file for our beans.

Finally, I copy the output .jar file to the WebLogic applications directory and run the Java client. The client creates the UserManager session bean and calls UserManager to create a User, passing a User value object (see Listing 5). Groups are created in a similar way. After a Group is created, I can add a User to it with just one call.

```
groupManager.addUserToGroup("alex",
"testgroup");
```

Under the hood, the container inserts a row in the table USER_GROUP that points to tables SYSUSER and SYSGROUP connecting them.

When testing my applications on WebLogic 6.1 I used a utility P6Spy available from Provision6 (www.provision6.com/) to analyze the SQL generated by the container. P6Spy driver is configured to log all SQL statements before they get to the real JDBC driver.

One thing I'd see from the SQL log is that for every get method called on an entity bean, the container generates a SELECT statement. To improve that you can use different concurrent strategy flags in the WebLogic descriptor. For example, if you set concurrency strategy to be Exclusive, then the SELECT statement is issued only once, the first time the bean is loaded. The assumption here is that all updates are going through the bean, so the bean doesn't have to be synchronized with the database for every get method. (See WebLogic documentation for more on other concurrent strategy flags and entity bean optimization.)

Similarly, for every set method on the entity bean, the container issues an UPDATE statement. The reason is that every set method is performed in a new transaction, and at the end of each transaction the container commits the data to

"A framework is of the greatest use when an application is able to pick and choose those features that are most important"

Author Bio
Alex Pestrikov, a programmer for more than five years, has spent the last three years working on Java projects. He currently writes J2EE applications for the Middleware Lab, government of Ontario, Canada. Alex holds a degree in finance and computer science from the University of Bridgeport (Connecticut), and is a master's candidate in computer science.

storage. Again, we can improve that if all set methods are called in the same transaction. Since they're called inside the facade session bean, we can mark the bean to require a transaction in the ejb-jar.xml deployment descriptor. Further optimizations are possible as required by a specific application.

Entity beans separate business logic from persistence code. Local entity beans cover the area of fine-grained objects making necessary syntactical changes to optimize performance. Declarative relationships take another step to abstract object developers from the database schema.

These features of CMP have the potential to increase the productivity of the enterprise application developers. The situation will further improve once EJB container vendors create or improve graphical tools to write deployment descriptors for CMP and relationships. For now the example deployment descriptors given in this article can help you get started.

For More Information

1. EJB 2 Proposed Final Draft 2: http://java.sun.com/products/ejb/docs.html
2. BEA Weblogic Server Release 6.1 Documentation: http://e-docs.bea.com/wls/docs61/index.html
3. Jakarta Ant Project: http://jakarta.apache.org/ant/
4. P6Spy Driver: www.provision6.com/index.htm

pestrikov@yahoo.com

Listing 1: ejb-jar.xml

```
<relationships>
  <ejb-relation>
    <ejb-relation-name>Group-User</ejb-relation-name>
      <ejb-relationship-role>
        <ejb-relationship-role-name>
          group-has-users
        </ejb-relationship-role-name>
        <multiplicity>many</multiplicity>
        <relationship-role-source>
          <ejb-name>Group</ejb-name>
        </relationship-role-source>
        <cmr-field>

   <!-- method to retrieve users
   is java.util.Collection getUsers() -->

        <cmr-field-name>users</cmr-field-name>

   <!-- you can specify java.util.Set or java.util.
   Collection if multiplicity is many.
   You do not have to provide this element if multiplicity is one
-->

        <cmr-field-type>java.util.Collection</cmr-field-type>
        </cmr-field>
        </ejb-relationship-role>

      <ejb-relationship-role>
        <ejb-relationship-role-name>
          user-belongs-to-group
        </ejb-relationship-role-name>
        <multiplicity>many</multiplicity>
        <relationship-role-source>
        <ejb-name>User</ejb-name>
        </relationship-role-source>
          <cmr-field>
            <cmr-field-name>groups</cmr-field-name>
            <cmr-field-type>
              java.util.Collection
            </cmr-field-type>
          </cmr-field>
      </ejb-relationship-role>
  </ejb-relation>
</relationships>
```

Listing 2: weblogic-ejb-jar.xml

```
<weblogic-enterprise-bean>
  <ejb-name>User</ejb-name>
    <entity-descriptor>
      <persistence>
         <persistence-type>
           <type-identifier>
                WebLogic_CMP_RDBMS
           </type-identifier>
           <type-version>6.0</type-version>
           <type-storage>
               META-INF/weblogic-cmp-rdbms-jar.xml
           </type-storage>
         </persistence-type>
         <persistence-use>
           <type-identifier>
               WebLogic_CMP_RDBMS
             </type-identifier>
             <type-version>6.0</type-version>
         </persistence-use>
         </persistence>
      </entity-descriptor>
      <local-jndi-name>User</local-jndi-name>
</weblogic-enterprise-bean>
```

Listing 3: weblogic-cmp-rdbms-jar.xml

```
<weblogic-rdbms-relation>
   <relation-name>Group-User</relation-name>
    <table-name>USER_GROUP</table-name>
    <weblogic-relationship-role>
       <relationship-role-name>
          group-has-users
       </relationship-role-name>
       <column-map>

         <!-- column from USER_GROUP table>
         <foreign-key-column>
            GROUPNAME
         </foreign-key-column>

         <!-- column from SYSUSER table>
         <key-column>GROUPNAME</key-column>
       </column-map>
    </weblogic-relationship-role>
    <weblogic-relationship-role>
       <relationship-role-name>
           user-belongs-to-group
       </relationship-role-name>
       <column-map>

         <!-- column from USER_GROUP table -->
         <foreign-key-column>
             USERNAME
         </foreign-key-column>

         <!-- column from SYSGROUP table -->
         <key-column>USERNAME</key-column>
       </column-map>
    </weblogic-relationship-role>
</weblogic-rdbms-relation>
```

Listing 4: Ant build script

```
<!-- Run ejbc to create the deployable jar file -->
  <target name="ejbc" depends="jar_ejb">
    <java classname="weblogic.ejbc" fork="yes">
      <sysproperty key="weblogic.home" value="${WL_HOME}"/>
      <arg line="-compiler javac ${dist}/user_in.jar
./user_out.jar"/>
      <classpath>
        <pathelement path="${WL_HOME}/lib/weblogic.jar"/>
      </classpath>
    </java>
  </target>
```

Listing 5: Client

```
InitialContext ic = new InitialContext();
Object objRef = ic.lookup( "UserManager" );
UserManagerHome home = (UserManagerHome)
PortableRemoteObject.narrow( objRef, UserManagerHome.class );
UserManager userManager = home.create();

User user = new UserValueObject();
user.setUserName("alex");
user.setUserType("regular");
user.setUserEmail("user@yahoo.com");
userManager.createUser(user);
```

Download the Code!
www.JavaDevelopersJournal.com

MVC²

Integrating and Mapping a Web Application MVC Pattern

Written by Steven Sweeting, Aaron Rustad, & Clive Jones

As Java technology has matured over the last few years so have we. We've learned that building complex enterprise applications that respond to change requires more than standardized APIs and virtual machines. Fortunately, we're now starting to see the widespread adoption of best practices, patterns, and even frameworks with templates and prebuilt components.

This article looks at the MVC design pattern and reviews its implementation in Struts, a presentation-tier application framework, as well as recognizing analogies of MVC to a well-formed, EJB-tier framework that Struts can be integrated with.

MVC for J2EE

Struts is an implementation of the presentation tier using MVC. The Struts controller encapsulates the presentation logic, handling the interaction of the JSP pages view with the JavaBeans model, which might access a database through JDBC.

When we integrate with a business tier, the recommended pattern is to access a Session Façade or Business Service – analogous to an MVC view. The model could be thought of as our data access objects or entity beans. The model can be queried by the view, but updates occur through the controller, which would then represent our business logic.

When we consider the applicability of the MVC pattern to the business tier, it enables us to recognize the separation of responsibilities between the interface, model, and business logic. It demarcates responsibilities across our infrastructure components. We see, for example, how to use different interfaces or views for the same model or controller.

Struts Implementation of MVC

Struts delivers on its promise of a framework that follows the MVC design pattern by effectively segregating the various components into a small set of extendable classes. Also note the declarative nature of its XML-based action mappings and JSP tags.

Figure 1 illustrates the interaction between the key Struts components as they relate to editing user information.

Figure 2 illustrates how each major component of Struts maps into the MVC.

The Model

The Struts implementation of the Model comes by way of the abstract class, ActionForm. It manages the data required by the View (JSP) and changes its state according to the Controller (Action). In its simplest terms, the ActionForm is a standard JavaBean with getters and setters used to

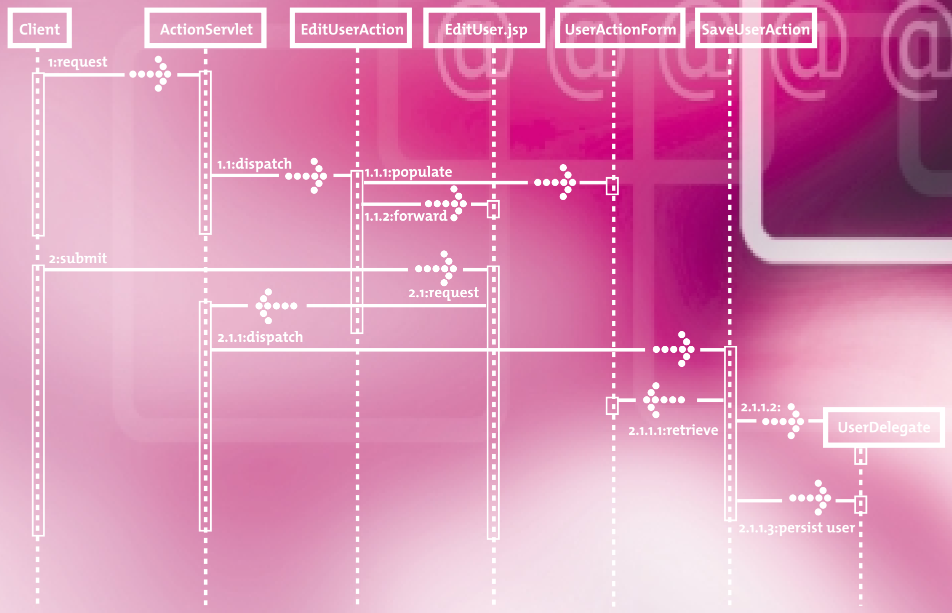


Figure 1:Struts sequence diagram

access its state. It defines no abstract methods, but you have the option of overriding the validate method to perform server-side validation of the application data.

An ActionForm used to maintain data about a user may look like Listing 1.

The View

The extensive Struts tag libraries combine to create a mechanism of presentation and interaction that allows developers to develop, and designers to design. The JSP will use the ActionForm (Model) as a means of allowing the client to interact with the application.

Listing 2 provides an example of a JSP page with Struts tags.

The Controller

Two Struts classes make up the Controller, ActionServlet and Action. Both work in concert to manage the flow of the application, but have distinct differences. The ActionServlet is responsible for mapping URI requests to specific Actions, while the Actions are responsible for interacting with the Model (ActionForms).

The developer extends the Action class to implement logical units of work that will be invoked by the ActionServlet upon receiving a request. Saving user information may be part of the SaveUserAction (see Listing 3).

UserDelegate is a class based on the J2EE Business Delegate design pattern. It hides the implementation of the business-tier's view of the model and will be explained in detail later in this article. The important thing to note is how the Struts model interacts with the business-tier's view – the action is the bridge that joins the two.

The ActionServlet is made aware of the URI to Action mappings via an XML configuration file. This file describes request URIs and how the ActionServlet should dispatch them. A typical mapping is provided in Listing 4.

When the ActionServlet receives the request such as http://myapp/SaveUser.do from the EditUser.jsp, it understands that its mapping corresponds to the path attribute defined in the XML configuration file, and that it should invoke the perform method of the SaveUserAction class. The attribute "name" defines the ActionForm (model) that the action will receive and the scope attribute defines where the form will be found (request, session, context). It also defines URI mappings that the Action will use to determine where to forward. By keeping the application mappings external, the Struts framework can be easily modified without recompiling classes.

Online Business Systems has used the Struts application framework on many successful projects including Viewtrak.com, an integrated food source management tool that tracks production of cattle from producer to consumer. The architect, Curtis Linton of Online Business Systems, used Struts as the Web application framework and the Versata Logic Suite for business logic development and management. Example code from this project has been used in this article, and updated to current versions.

Business-Tier Pattern Framework

Naturally there are many approaches to implementing business-tier design patterns such as developing infrastructure components and business components.

First, to develop the business components we should consider a declarative approach, a business rules approach. Barbara von Halle in her book, *Building Business Rule Systems*, states, "It is no longer desirable to bury rules in specifications and program code where they are locked away, requiring costly intervention and overhead to effect change."

Second, we need to pay careful attention to J2EE best practices to avoid those performance pitfalls we've read about: too many fine-grained entity beans, caching, use of CMP, and so on.

Business-Tier Design Pattern Framework

Figure 3 illustrates a well-formed design pattern framework for the business tier.

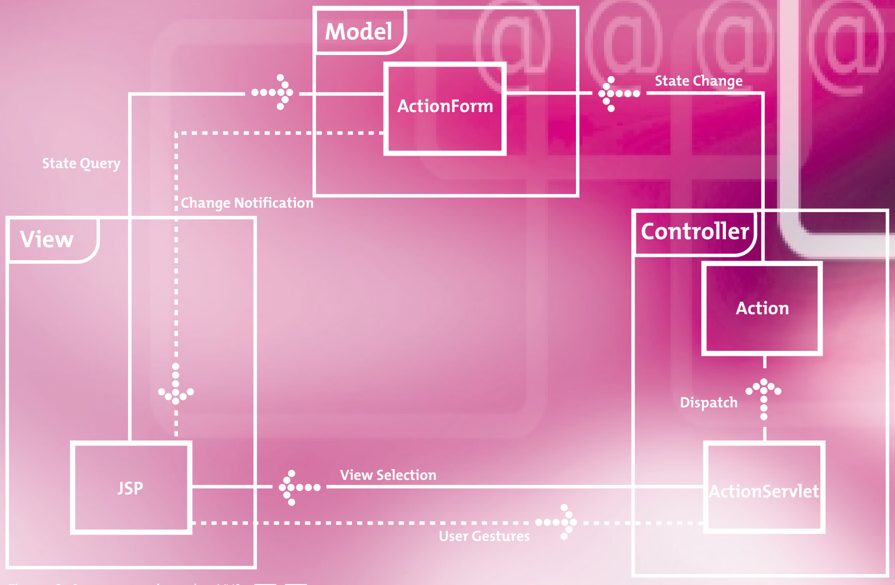


Figure 2: Struts maps into the MVC

What follows is a discussion of this pattern framework in terms of the core J2EE design patterns. Using this framework, we'll also look at how products within the Versata Logic Suite leverage and implement these patterns and how they accelerate and simplify business logic development and maintenance.

Within the Versata Logic Suite, business components are built with the Versata Logic Studio from business-logic specifications, especially business rules. Java artifacts, such as value objects, value object assemblers, business objects (entity and session beans), and data access objects, are produced as well as deployment artifacts, such as database schemas, deployment descriptors, and test data scripts.

The business components may be accessed as regular Enterprise JavaBeans through the Versata Transaction Logic Engine or through the Versata Logic Server that includes the framework that follows the patterns we describe below. The Versata Logic Server also includes the optional Versata Presentation Engine, which supports the model-based development of HTML servlet (presentation tier) and Java (client tier) Web applications.

Business Delegate

Business Delegates within the presentation tier hide the remote method invocation and interaction with the Business Service pattern on the business tier.

The Business Delegate pattern represents the Model of our presentation-tier MVC pattern.

Business Service

The Business Service is the component that presents the actual business functionality to the outside world. This could be through RMI to an EJB, through messaging with JMS, or perhaps through SOAP to a Web service.

As an interface into our business logic we would consider Business Service one of the View components of our MVC-like pattern.

Value Objects

While a client may access EJBs directly via RMI calls to attribute getter and setter methods, in practice this conversational approach is expensive. A "value object" is an encapsulation of business data that can be serialized, allowing the client to make one RMI call to retrieve or update a set of related data.

The Versata Suite enables value objects to be created declaratively through the Versata Logic Studio, allowing attributes to be selected from objects in the entity object model, and creating composite objects by joining across object relationships. Value objects are also created for each entity object by default.

These value objects are also updatable, encapsulating behavior that allows them to interact directly with the Business Service, such as update, create, delete, and refresh. This is also true for composite objects – updates may be distributed across different business objects. They're also metadata value objects accessed somewhat like javax.sql.RowSet objects.

Considering the MVC pattern, value objects represent the view. This suggests to us that they should be model-independent, not encapsulate business logic. We should look to create value objects that represent how a system presents itself to the outside world as opposed to necessarily re-presenting internal models.

Value Object Assembler

Value objects are supported at runtime by the Value Object Assembler that builds and aggregates them from underlying business objects.

Would this component be a view or a controller? Value Object Assemblers typically are involved only in query operations, so they would be equivalent to a view component.

If we were more faithful to the MVC pattern, the view objects would access the model directly for querying state, only accessing the controller for updates. In fact, we sometimes see this pattern as a performance optimization, especially for accessing large sets of data. The value object will pull

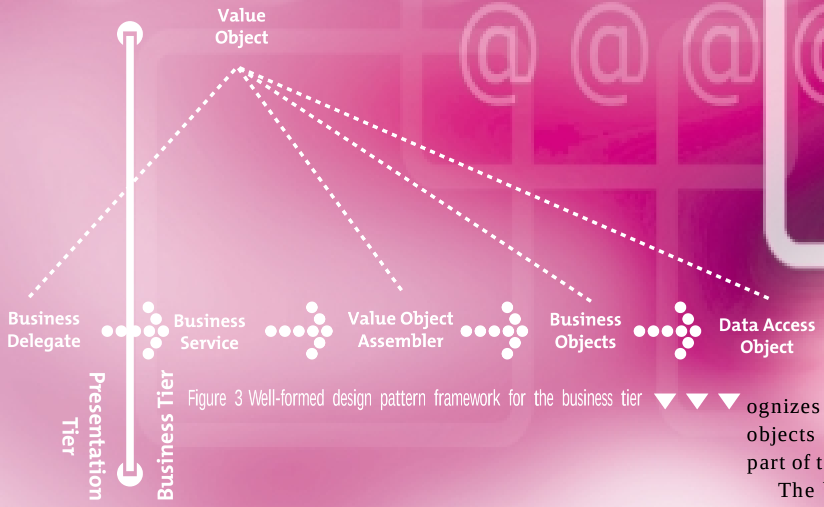


Figure 3 Well-formed design pattern framework for the business tier

data straight from the data access object or JDBC, alleviating the need to instantiate an entity bean for each object instance.

Our implementation uses this optimization transparently. The Versata Value Object Assembler instantiates only business objects for query operations when it has to. Since it knows attributes are persisted or calculated, it infers whether it can retrieve data straight back from the data access object or whether business logic needs to be executed.

Data Access Object

Enterprise applications need to access data from multiple sources, SQL databases, LDAP servers, and legacy systems, for example. Data access objects form an abstraction layer and an encapsulation of the data between the data source and the business objects. This allows the business objects to use the same API no matter where the data is coming from.

You can write data access objects or use tools. A good example is IBM's VisualAge for Java (VAJ). VAJ's Enterprise Access Builder allows you to create data access objects for sources ranging from DB2 to CICS.

The Versata Logic Server has a powerful implementation of the data access layer. Access objects are created at runtime based on object metadata, runtime configuration properties, and transaction requirements. The administrator can change the data source at runtime and tune options, such as whether to use prepared statements, the prepared statement cache-size, connection pooling, and lock isolation levels, giving a great deal of flexibility and high performance without requiring hand-coded BMP.

Data access objects clearly represent the Model of our MVC-like business-tier pattern.

Business Objects

The components of the Business Object pattern encapsulate the business logic of our system. For J2EE enterprise applications, these components will usually be session beans, entity beans, and dependent object JavaBeans. We use the term *business objects* to describe these. The J2EE pattern recognizes that business objects could also be data access objects or JDBC helpers. For our framework these would be part of the Data Access Object pattern.

The business objects operate within the EJB container of the J2EE application server and use the services for coordinated transaction management and persistence, for example.

The Versata Logic Suite supports the automated creation of business objects from specifications. The system sequences, optimizes, and translates these specifications or what we call business rules into the business objects – the Java components. Typically some custom Java coding is required, which might be entered directly into business rules or used to subclass or add code to Java components. We typically see about 98–99% of the Java component built directly from rules.

Looking at the MVC patterns, would business objects be akin to the model or the controller? You could argue that they're part of the model, the intrinsic representation of what the business is. In a way though that's what the whole business tier represents. We prefer thinking about business objects as the controller. Recall that for the presentation tier, the controller provides the presentation logic. For the business tier, the business objects handle business logic, providing a clean separation from the data (the model) and from business services (the view).

Declarative Business Logic

What Not How – Declarative Not Procedural

As elaborated by C.J. Date in his book, *What Not How*, there's a move away from procedural programming – coding the steps of *how* we accomplish something – to the declarative – just saying *what* we want to accomplish in the first place and letting the system work out the best way of doing it.

An example of declarative programming is the EJB deployment descriptor, specifying what fields should have their persistence managed by the container. EJB 2.0 goes further by adding declarative support for container-managed relationships; something you have to manage programmatically today. We've also seen the Struts action mappings and view definitions as declarative approaches.

Automation of Business Rules

Consider the business requirement that the account balance is total billed invoices less payments received. Taking a procedural approach, we could then write code to update the account balance every time a payment is received or an invoice is sent out. What about when an invoice is cancelled? What about a cancelled invoice that's reinstated? What about deleting a payment? What about when a payment bounces? We need to be conscious of a number of side effects with a procedural approach. What about when the invoice's account changes? You need to subtract from one account and add to another, but only if the invoice has been billed.

Declaratively we might express this logic as:

1. Account Balance=Total Billed–Total Payments
2. Total Billed=sum (Invoice.TotalFee where InvoiceStatus ="Billed")
3. Total Payments=sum (Payment.Total where PaymentStatus ="Active")

These are called *business rules*. Taking into account all side effects, these rules require over 300 lines of Java code across the Account, Invoice, and Payment objects. Listing 5 provides a small portion of the code built using the Versata Logic Suite as it appears in the Invoice object. The code implements logic to deal with a change in the InvoiceStatus of TotalFinalFee of an invoice and the impact on TotalBilled of an account.

Constraints are another type of business rule; they represent the policies under which a business can operate. Figure 4 shows constraints for the Customer object.

In business rules terminology we call these calculations *derivation rules* – the attribute definitions are derived from other attributes. Figure 5 shows several derivation rules for the Account business object, providing detail from the TotalBilled sum rule. Additional derivation types can be seen in the grid: a replicate, a calculation, a default, two sums, and a count.

Business Rules Paradigm Shift

To develop with business rules a paradigm shift from the procedural to the declarative is required. Fortunately, it's an easier shift than we made to the object-oriented world, and one that business analysts quickly identify with. After all, they're business rules. Some of the advantages of a rules approach are:

- ***Business logic is not hidden in code:*** Business rules can be quickly found and understood and thus speedily and reliably modified.
- ***Reduces errors:*** Business rules can be understood and signed-off by business users with no implementation errors as the system translates specifications into procedural code automatically. The testing cycle is also significantly reduced.
- ***Naturally separates business logic from presentation logic:*** J2EE architects can concentrate on infrastructure and integration, not on the supervision of business logic development and maintenance.
- ***Enables new Java developers who know the business to be quickly productive with J2EE technology.***

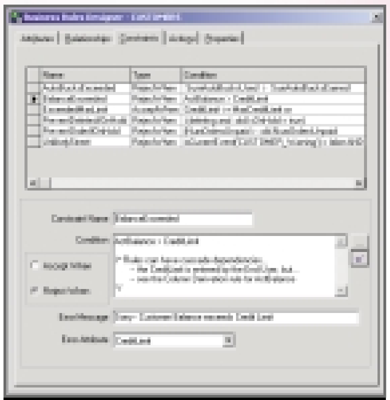


FIGURE 4 Business Rules Designer Constraints

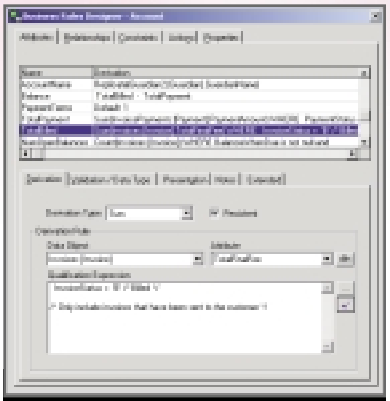


FIGURE 5 Business Rules Designer – Derivations

Summary

The Model-View-Controller pattern continues to offer a powerful abstraction to the J2EE architect, creating well defined but loosely coupled components, enabling change. Although typically applied at the presentation tier, the MVC pattern can also guide us within the business tier, providing a clear delineation of work to developers and infrastructure components. Through the discussion of the Struts framework and the Versata Logic Suite for business logic development we also saw how change is enabled through systems driven from specifications, not code – moving toward what not how.

References

1. Date, C.J. (2000). *What Not How: The Business Rules Approach to Application Development*. Addison-Wesley.
2. *The original GUIDE paper and continuing work:* www.businessrulesgroup.org
3. von Halle, B. (2001). *Building Business Rule Systems*. John Wiley & Sons, Inc. www.buildingbusinessrulesystems.com
4. Alur, D., Crupi, J., and Malks, D. (2001). *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall.
5. Sucharitakul, A. (2001). "Seven Rules for Optimizing Entity Beans." http://developer.java.sun.com/developer/technicalArticles/ebeans/sevenrules/. May.
6. Geary, D.M. (2001). *Advanced JavaServer Pages*. Prentice Hall.
7. *Struts:* http://jakarta.apache.org/struts
8. *Versata, Inc.:* www.versata.com

Author Bios

Steven Sweeting is director of product management with Versata, Inc. Ever since he developed a C++ program to write most of his college COBOL assignment, he's worked hard to avoid coding repetitive business logic.

Clive Jones is a senior enterprise Java developer with Online Business Systems, a leading consulting company employing over 300 people throughout Canada and the U.S. He is based in Winnipeg, Manitoba.

Aaron Rustad is a senior Java developer for Online Business Systems. He specializes in architecting and developing enterprise applications with a primary focus on Java and open-source technologies.

steven_sweeting@versata.com, arustad@online-can.com & cjones@online-can.com

Listing 1

```
public UserForm extends ActionForm {
 private String user = null;
 private String address = null;

 public getUser() { .. }
 public setuser() { .. }
 ..
 ..

 public ActionErrors validate (ActionMapping mapping,
      HttpServletRequest req)
{
   ActionErrors errors = new ActionErrors();
   If (user == null || user.length() < 1)
  {
  Errors.add("user", new ActionError("error.user.required"));
   }
   ..
   ..
   return errors;
 }
}
```

Listing 2

```
 <!-- Importing the tag strut tag libraries -->
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

..
..

<html:form action="/SaveUserAction" focus="user">
<table border="0" width="100%">

  <tr>
    <th align="right">
      <bean:message key="prompt.username"/>
    </th>
    <td align="left">
      <html:text property="user" size="16" maxlength="16"/>
    </td>
  </tr>

  <tr>
    <th align="right">
      <bean:message key="prompt.address"/>
    </th>
    <td align="left">
      <html:text property="address" size="16" maxlength="16"/>
    </td>
  </tr>

  <tr>
    <td align="right">
      <html:submit property="submit" value="Submit"/>
    </td>
    <td align="left">
      <html:reset/>
    </td>
  </tr>

</table>

</html:form>
..
..
```

Listing 3

```
  public final class SaveUserAction extends Action {
 Public ActionForward perform(ActionMapping mapping,
      ActionForm form,
      HttpServletRequest req,
      HttpServletResponse res ) throws
IOException,
                                    ServletException {
 ActionForward forward = null;

UserForm userForm = (UserForm) form;

UserDelegate userDelegate = new UserDelegate();

 try {
  userDelegate( userForm );
 } catch ( DelegateException e) {
  forward = mapping.findForward("failure");
 } finally {
  if (forward == null) {
   forward = mapping.findForward("success");
  }
 }
 return forward;
}
```

Listing 4

```
  <action-mapping>
 <action
   path="/SaveUser"
   type="com.demo.SaveUserAction"
   name="userForm"
   scope="request"/>
 <forward name="success" path="/saveSuccessful.jsp"/>
 <forward name="failure" path="/saveFailed.jsp"/>
</action-mapping>
```

Listing 5

```
 if ( childCascadeUpdate )
{
   BigDecimal addValue = new BigDecimal("0");
   BigDecimal subValue = new BigDecimal("0");
   BigDecimal delta    = new BigDecimal("0");

   if ((this.getInvoiceStatus()).equals("B"))
   {
    addValue = getTotalFinalFee();
   }
   if ((this.getOldInvoiceStatus()).equals("B"))
   {
    subValue = getOldTotalFinalFee();
   }
   delta = addValue.subtract(subValue);
   if ( ! delta.equals(new BigDecimal("0")) )
   {
    if ( !newParent.isInitialized() )
       newParent.setDataObject(this.getAccount());

    if ( ! newParent.isObjNull() )
    newParent.getDataObject().setAdjust("TotalBilled",delta, true);
       }
   }
   return;
}
```

Download the Code! www.JavaDevelopersJournal.com

Core J2EE Patterns

Presentation tier patterns and refactoring

WRITTEN BY
DAN MALKS

Patterns are expert solutions – recurring designs that have proven effective over time. This month's article will provide you with a bit more detail on the subject.

When applying patterns to the presentation tier, we address the following:

- *Pre- and postprocessing of a request and/or response*
- *Request handling, navigation, and dispatch*
- *View preparation and creation*
- *Application partitioning*

In other words, how do we handle a Web request, managing any necessary data access and generating presentation content for the user? How might we transform the incoming or outgoing stream? How do we determine what processing to perform to fulfill this request? What view should be used to service the request and how do we generate it? What are the logical abstractions and how do we design and decompose our software to take advantage of these abstractions?

Six presentation tier patterns are documented in the Core J2EE Patterns Catalog. The patterns and their tier categorizations are shown in Table 1. Because of the focus of this column, we'll describe just one of the presentation patterns in detail. First I want to provide some context on how the presentation tier patterns fit within a common J2EE architecture, starting with a basic description of a system.

When a client makes a Web request for a particular resource, the request is processed, a view is generated, and a result is returned to the client. To provide more detail, we can define several logical components and subcomponents of a typical Web-based architecture:

- ***Request handler:***
  - Pre- and postprocessing
  - Request processing
  - Command processing
- ***Navigation and dispatch:***
  - Navigation resolution
  - Request dispatching
- ***View processor:***
  - View preparation
  - View creation

These components and subcomponents are shown visually in Figure 1. Let's have a look at the Intercepting Filter pattern, which addresses issues in the request-handling portion of the architecture, as outlined above and in the figure.

Intercepting Filter

The Intercepting Filter pattern documents issues relating to preprocessing and postprocessing a Web request. Here are some common examples of preprocessing:

- ***Decrypting an input stream:*** The incoming data may have been encrypted for security purposes.
- ***Decompressing a stream:*** The incoming stream may have been compressed for more efficient transfer over the network.
- ***Translating various encoding schemes to a common format:*** Multiple encoding schemes require different handling. If each encoding is translated to a common format, the core request-handling mechanism can treat every request simi-

Every other month in this column we (Deepak Alur, John Crupi, Dan Malks, and other architects from the Sun Java Center (www.sun.com/service/sunps/jdc) will discuss various topics from our book, *Core J2EE Patterns: Best Practices and Strategies* (Alur, Crupi, Malks, Prentice Hall/Sun Press 2001). These topics include the 15 J2EE patterns in our catalog, design strategies, bad practices, refactorings, and pattern-driven design in J2EE technology.

TIER	PATTERN NAME
Presentation Tier	Intercepting Filter Front Controller View Helper Composite View Service to Worker Dispatcher View
Business Tier	Business Delegate Value Object Session Facade Composite Entity Value Object Assembler Value List Handler Service Locator
Integration Tier	Data Access Object Service Activator

TABLE 1 J2EE Patterns by tier

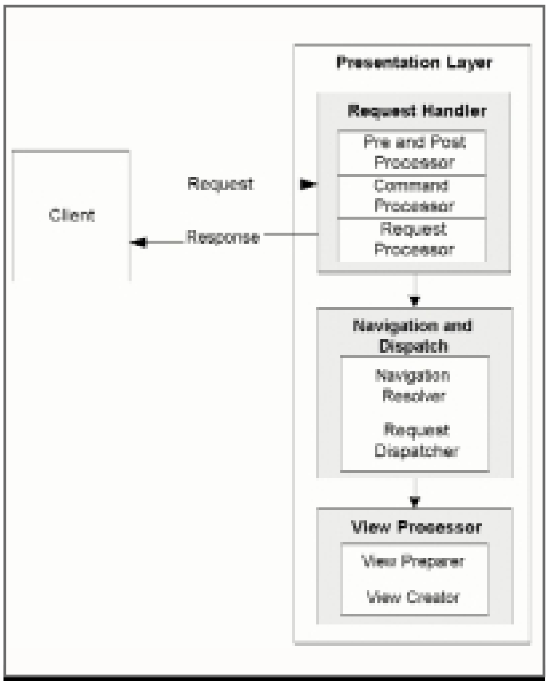


FIGURE 1 Request-handling architecture

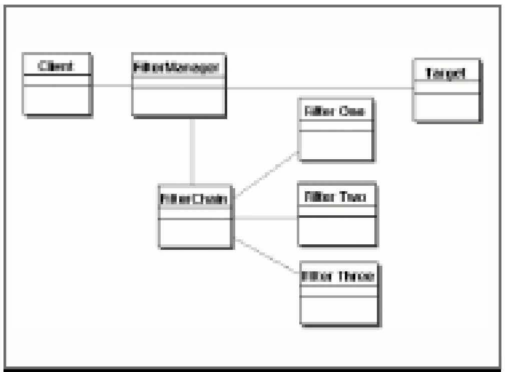


FIGURE 2 Intercepting Filter class diagram

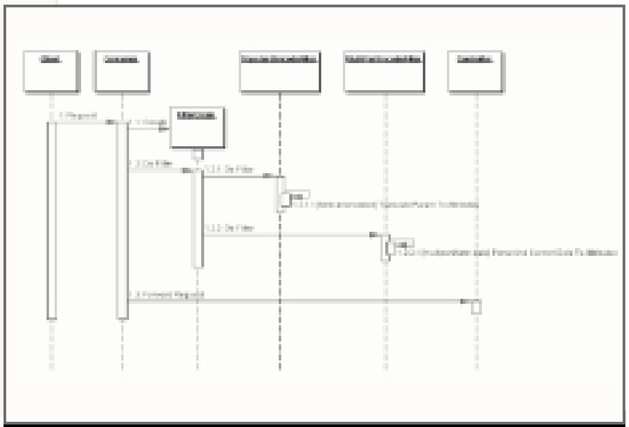


FIGURE 3 Filter sequence diagram

larly. Some common examples of encoding schemes are:

– application/x-www-form-urlencoded
– multipart/form-data

- ***Performing authentication or authorization:*** Authentication and authorization may be performed either in a filter or as part of the core processing flow, typically as part of the controller.

What constitutes postprocessing of a request? Here are some common examples:

- ***Encrypting an output stream:*** The outgoing data may be encrypted for security purposes.
- ***Compressing a stream:*** The outgoing stream may be compressed for more efficient transfer over the network.
- ***Transformation of data for different clients:*** Transformation into HTML, XML, or WML.

Additionally, one of the forces that motivates us to consider this pattern is the desire to add and remove these processing components independently – independent of other filtering components and of the underlying core processing that fulfills the client's request.

Let's look at the class diagram (see Figure 2) that describes this pattern's structure. In the figure the actual resource that is the target of the client request, such as a servlet or JavaServer Page technology, is represented by the Target class. The individual pre- or postprocessing components that perform filtering functionality, as described above, are shown as Filter One, Filter Two, and Filter Three. One important thing to notice in this diagram is that there's no direct association between any of the filters and the target resource. Additionally, there's no direct association from any filter to one of the other filters.

This is an important point, since it clarifies that the filters are loosely coupled both to the target resource and to other filters. This allows the filters to be easily added and removed unobtrusively, as mentioned.

Filters are an excellent way to layer functionality onto your system, providing pluggable behavior that can be used to decorate core request processing. Another benefit of using this pattern is that it promotes the reuse of these various filtering components across different requests, in different combinations, and even in different applications.

There are several implementation strategies for this pattern, the most powerful of which leverages the standard filtering supported in the Servlet specification 2.3. Vendor support for this revision of the specification will be widespread in the not too distant future.

Listings 1 and 2 are excerpts from the Standard Filter Strategy code example in the book. The example describes using filters to preprocess requests, checking their encoding schemes, and translating these different schemes to a common format. The common format is to store all request states within request attributes. Subsequently, any control code that checks for incoming values will get these values from request attributes, regardless of the original encoding.

Figure 3 is the sequence diagram that shows the basic collaboration of the objects in the example. Note that in this implementation the role of the FilterManager from the class diagram is fulfilled by the Container in the sequence diagram. We hope this provides you with a basic understanding of the benefits and some implementing options for the Intercepting Filter pattern.

“The fact is that wherever there is coding, there may be refactoring”

Refactoring

There's more than one way to approach any task. This is as true with software development as with anything else. So when I tell you that people approach the task of developing software in different ways, you certainly won't be surprised. Some folks feel that most design work should precede implementation, while others like to jump in, write some code, and

PRESENTATION TIER REFACTORINGS
Introduce a controller
Introduce synchronizer token
Localize disparate logic
Hide presentation tier-specific details from the business tier
Remove conversions from view
Hide resource from a client
BUSINESS AND INTEGRATION TIER REFACTORINGS
Wrap entities with session
Introduce business delegate
Merge session beans
Eliminate inter-entity bean communication
Move business logic to session
GENERAL REFACTORINGS
Separate data access code
Refactor architecture by tiers
Use a connection pool

TABLE 2 Refactorings

start to think about how these bits of implementation fit together. The difference is basically that of top-down design versus bottom-up design.

Refactoring applies to either approach, though it's typically applied in an environment where there is an understanding that design is spread across the life of the project. That said, the fact is that wherever there is coding, there may be refactoring. Martin Fowler, in his great book *Refactoring: Improving the Design of Existing Code* (Addison-Wesley), describes refactoring as "improving the design of the code after it has been written." His book identifies many common design flaws and describes the incremental coding changes that result in improved design. The issues are typically general and not specific to any particular area of Java or software development.

The lion's share of Fowler's book is devoted to what he calls "small refactorings," meaning the design changes are at a very low level, each involving several discrete coding modifications, such as adding a parameter to a method. A small portion of the book, coauthored by Kent Beck, is devoted to "big refactorings," which exist at a higher level of abstraction and have steps that aren't as well defined or as concrete.

In our book we include some J2EE technology–specific refactorings, describing opportunities to improve the design of a J2EE technology-based system and the relevant steps involved. The format and style is based on that in Fowler's book, which we find extremely valuable. Based on Fowler and Beck's definition, the J2EE refactorings included in our book might be called "medium refactorings" based on their level of abstraction. The refactorings are listed in Table 2, categorized by tier.

We find these refactorings to be excellent companions to the patterns and bad practices described in the rest of our book. In fact, you can think about the refactorings as often providing the steps that help guide the developer from a less optimal solution, or bad practice, to a more optimal one, suggested by a pattern.

In a future article we'll provide more information on these refactorings and their relationship to the patterns in the catalog. We'll also go into greater detail on the presentation, business, and integration tiers, as well as communication across these tiers.

Thank you for reading, and please e-mail us at CoreJ2EEPatterns@sun.com to provide feedback on this article and to suggest other topics of interest.

Portions of this article contain excerpts with permission from *Core J2EE Patterns* by Deepak Alur, John Crupi, and Dan Malks (ISBN 0-13-064884-1) Copyright 2001. Sun Microsystems, Inc.

Author Bio

Dan Malks, a senior Java architect with Sun Microsystems, is currently focusing on distributed, service-based designs, patterns, and implementations. He has developed in a variety of environments, including Smalltalk and Java, while focusing on OO technologies. Dan has published articles on Java in leading industry periodicals, and holds bachelor's and master's degrees in computer science.

dan.malks@sun.com

SYS-CON MEDIA

PUBLISHER, PRESIDENT, AND CEO
FUAT A. KIRCAALI fuat@sys-con.com

ADVERTISING

SENIOR VICE PRESIDENT, SALES AND MARKETING
CARMEN GONZALEZ carmen@sys-con.com

VICE PRESIDENT, SALES AND MARKETING
MILES SILVERMAN miles@sys-con.com

ADVERTISING SALES DIRECTOR
ROBYN FORMA roybn@sys-con.com

ADVERTISING ACCOUNT MANAGER
MEGAN RING megan@sys-con.com

ASSOCIATE SALES MANAGER
CARRIE GEBERT carrieg@sys-con.com

SALES ASSISTANT
ALISA CATALANO alisa@sys-con.com

EDITORIAL

EXECUTIVE EDITOR
M'LOU PINKHAM mpinkham@sys-con.com

EDITOR
NANCY VALENTINE nancy@sys-con.com

MANAGING EDITOR
CHERYL VAN SISE cheryl@sys-con.com

ASSOCIATE EDITOR
JAMIE MATUSOW jamie@sys-con.com

ASSOCIATE EDITOR
GAIL SCHULTZ gail@sys-con.com

ASSOCIATE EDITOR
BRENDA GREENE brenda@sys-con.com

ASSISTANT EDITOR
LIN GOETZ lin@sys-con.com

PRODUCTION

VICE PRESIDENT, PRODUCTION AND DESIGN
JIM MORGAN jim@sys-con.com

ART DIRECTOR
ALEX BOTERO alex@sys-con.com

ASSOCIATE ART DIRECTOR
LOUIS F. CUFFARI louis@sys-con.com

ASSISTANT ART DIRECTOR
CATHRYN BURAK cathyb@sys-con.com

GRAPHIC DESIGNER
ABRAHAM ADDO abraham@sys-con.com

GRAPHIC DESIGNER
RICHARD SILVERBERG richards@sys-con.com

GRAPHIC DESIGNER
AARATHI VENKATARAMAN aarathi@sys-con.com

WEB SERVICES

WEBMASTER
ROBERT DIAMOND robert@sys-con.com

WEB DESIGNER
STEPHEN KILMURRAY stephen@sys-con.com

WEB DESIGNER
PURVA DAVE purva@sys-con.com

WEB DESIGNER INTERN
CAROL AUSLANDER carol@sys-con.com

ACCOUNTING

CHIEF FINANCIAL OFFICER
BRUCE KANNER bruce@sys-con.com

ASSISTANT CONTROLLER
JUDITH CALNAN judith@sys-con.com

ACCOUNTS RECEIVABLE
JAN BRAIDECH jan@sys-con.com

SYS-CON EVENTS

VICE PRESIDENT, SYS-CON EVENTS
CATHY WALTERS cathyw@sys-con.com

CONFERENCE MANAGER
MICHAEL LYNCH mike@sys-con.com

SALES EXECUTIVE, EXHIBITS
MICHAEL PESICK michael@sys-con.com

SALES EXECUTIVE, EXHIBITS
RICHARD ANDERSON richard@sys-con.com

SHOW ASSISTANT
NIKI PANAGOPOULOS niki@sys-con.com

JDJSTORE.COM
ANTHONY D. SPITZER tony@sys-con.com

Listing 1: Intercpting Filter Implementation Example: Standard Encode Filter

```
public class StandardEncodeFilter
extends BaseEncodeFilter {
    // Creates new StandardEncodeFilter
    public StandardEncodeFilter() {}

    public void doFilter(javax.servlet.ServletRequest

servletRequest,javax.servlet.ServletResponse

servletResponse,javax.servlet.FilterChain
                              filterChain)
    throws java.io.IOException, javax.servlet.ServletException {
        String contentType =
        servletRequest.getContentType();
        if ((contentType == null) ||
            contentType.equalsIgnoreCase("application/x-www-form-
urlencoded")) {
            translateParamsToAttributes(servletRequest,
servletResponse);
        }
        filterChain.doFilter(servletRequest, servletResponse);
    }
    private void translateParamsToAttributes(ServletRequest
request, ServletResponse response)
    {
        Enumeration paramNames =
        request.getParameterNames();
        while (paramNames.hasMoreElements()) {
            String paramName = (String)
                               paramNames.nextElement();
            String [] values;
            values = request.getParameterValues(paramName);
            System.err.println("paramName = " + paramName);
            if (values.length == 1)
                request.setAttribute(paramName, values[0]);
            else
                request.setAttribute(paramName, values);
        }
    }

}
```

Listing 2: Intercepting Filter Implementaion Example: MultipartEncoderFilter

```
public class MultipartEncodeFilter extends
BaseEncodeFilter {
    public MultipartEncodeFilter() {}
    public void doFilter(javax.servlet.ServletRequest
                              servletRequest,
javax.servlet.ServletResponse

servletResponse,javax.servlet.FilterChain
                              filterChain)
    throws java.io.IOException,
    javax.servlet.ServletException {
        String contentType =
        servletRequest.getContentType();
        // Only filter this request if it is multipart
        // encoding
        if (contentType.startsWith("multipart/form-data")){
            try {
                String uploadFolder =

getFilterConfig().getInitParameter("UploadFolder");

                if (uploadFolder == null) uploadFolder = ".";
                /** The MultipartRequest class is:
                * Copyright (C) 2001 by Jason Hunter
                * <jhunter@servlets.com>. All rights reserved.
                **/
                MultipartRequest multi = new

MultipartRequest(servletRequest,

uploadFolder,
                                                                            1 *

1024 * 1024 );
                Enumeration params =
                multi.getParameterNames();
                while (params.hasMoreElements()) {
                    String name = (String)params.nextElement();
                    String value = multi.getParameter(name);
                    servletRequest.setAttribute(name, value);
                }

                Enumeration files = multi.getFileNames();
                while (files.hasMoreElements()) {
                    String name = (String)files.nextElement();
                    String filename =
                    multi.getFilesystemName(name);
                    String type = multi.getContentType(name);
                    File f = multi.getFile(name);
                    // At this point, do something with the
                    // file, as necessary
                }
            }
            catch (IOException e)
            {
                LogManager.logMessage("error reading or saving
file"+ e);
            }
        } // end if
        filterChain.doFilter(servletRequest,
                             servletResponse);
    } // end method doFilter()
}
```

Download the Code!
www.JavaDevelopersJournal.com

Source Integrity Enterprise Edition 8.1 by MKS

REVIEWED BY JIM MILBERY
jmilbery@kuromaku.com

Despite the "slowdown" in technology, the developers and project managers who I speak with continue to be under enormous pressure to deliver new applications and technology at a frenetic pace. It can be incredibly difficult to juggle all the various development tasks when team members are working on multiple projects.

Developers often find themselves working with different teams for each autonomous project – and each project is often at a different stage of the project development life cycle. This combination is a breeding ground for all sorts of disasters including lost code, recurring bugs, missed deadlines, and questionable product quality. An important line of defense for project managers, developers, and quality-assurance engineers alike is a source-code control system. For large, geographically dispersed project teams, MKS Source Integrity Enterprise Edition 8.1 might just be the perfect solution.

MKS has been in the source code integrity business since 1984. SIE 8.1 represents the very latest version of their source code management product line, and it integrates with MKS Integrity Manager 4.2 to tie in source code control with defect tracking and workflow management. SIE 8.1 uses a completely new architecture that is based on an application server layer. Users can connect to the server using a Java-based client/server application and to IM through a thin-client browser-based interface.

Installing and Configuring SIE 8.1

MKS ships the software on two separate CDs – one for the Integrity Server and one for the client installation. I tested SIE on a Windows 2000 server, but SIE supports a variety of different server platforms including Linux, Solaris, and HP-UX. SIE uses a database to store security and change package data. But the real repository data (projects and archives) is stored on the filesystem. The current release supports Oracle, Microsoft SQL Server, and PointBase as the database.

I used the built-in PointBase database as the repository and followed the simple Install-Anywhere instructions to install Integrity Server. The installation is simple, but you may run into a problem with the FlexLM license manager. MKS ships out a license file with the software, and you'll need to get the FlexLM license manager configured properly before you can connect to the server. MKS offers both "named user" and "floating" licenses for SIE 8.1, which is a plus, but I'm not a big fan of FlexLM as a license manager.

Once you get the server installed the real fun begins. SIE 8.1 is loaded with features and functions for source code control and security. I can't imagine any configuration that SIE would not be able to support with its vast array of functions. For example, the server provides for five different security-realm configurations (Unix, NT, LDAP, flat file, and WebLogic Server). However, configuring the security realm requires you to edit a series of property files and manually restart the server using the Windows NT Services Manager. (There's no GUI server manager.) This is typical of IM and SIE in general. The products are loaded with functionality, but it can be difficult to figure out just how to implement a certain feature or function. In all fairness, the client installation goes much smoother, since there's very little configuration that you need to worry about once you have the server up and running.

Working with IM and SIE

MKS provides a Java client application that you can use to connect to the Integrity Server. When you install the client interface it auto-

mks
Join MKS at the Embedded Systems Conference!
[illegible]

mks
SOURCE INTEGRITY
ENTERPRISE EDITION
[illegible]

specs

MKS Inc.
410 Albert St.
Waterloo, ON
Canada, N2L 3V3
Phone: 519 884-2251
Fax: 519 884-8861
Phone: 800 265-2797
e-mail: info@mks.com
Web: www.mks.com

Test Platform
Toshiba Satellite Pro 4600, 866MHz Intel Pentium III processor, 20GB Disk, 256MB of Memory, Windows 2000 w/ Service Pack 2

Specifications
Platforms: Windows NT/2000, 98/95, Linux, Solaris, HP-UX, and AIX
Pricing: SIE Server is $4,000, SIE Client License is $750 per named user, IM Client License is $750 per named user

Source Integrity Enterprise Edition 8.1 by MKS

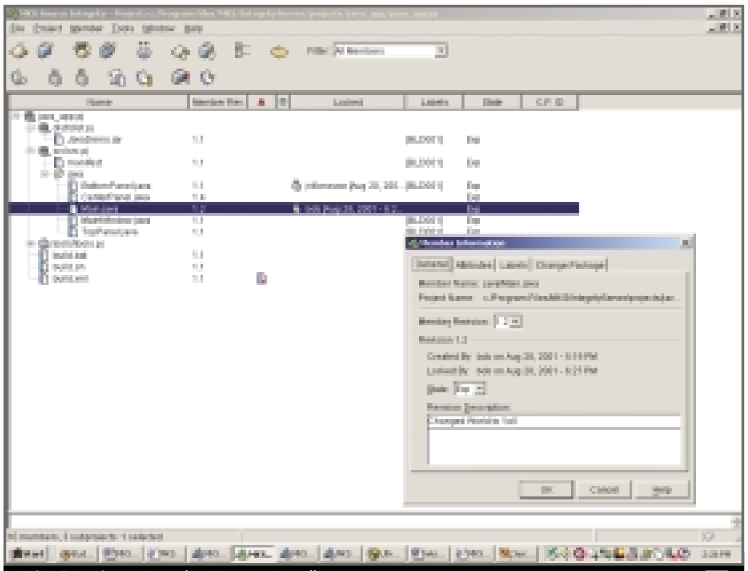
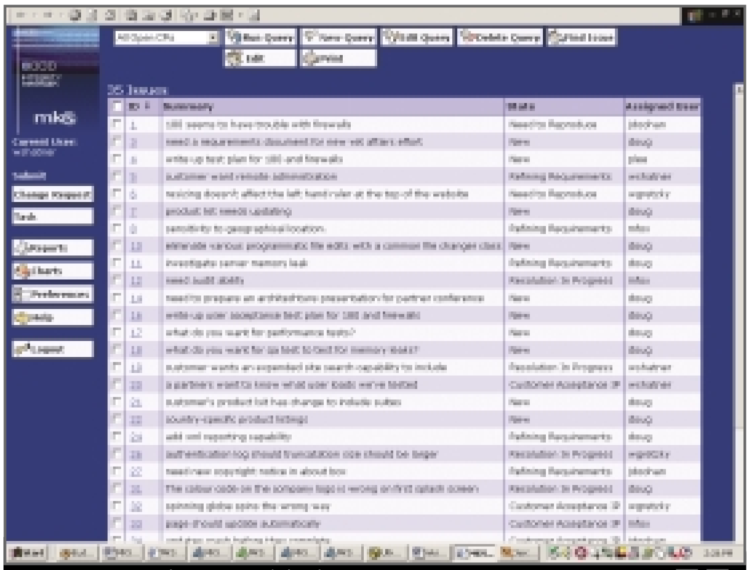


FIGURE 1 Source Integrity Manager Java client



All Open CRs | Run Query | New Query | Edit Query | Delete Query | Find Issue | Edit | Print

26 Issues

ID	Summary	State	Assigned User
1	100 seems to have trouble with firewalls	Need to Reproduce	pbohan
2	need a requirements document for new vet affairs effort	New	doug
4	write up test plan for 100 and firewalls	New	plee
5	customer want remote administration	Refining Requirements	wchafner
6	resizing doesn't affect the left hand ruler at the top of the website	Need to Reproduce	wgratzky
7	product list needs updating	New	doug
9	sensitivity to geographical location	Refining Requirements	mfox
10	eliminate various programmatic file edits with a common file changes class	New	doug
11	investigate server memory leak	Refining Requirements	doug
12	need audit ability	Resolution In Progress	mfox
13	need to prepare an architecture presentation for partner conference	New	doug
16	write up user acceptance test plan for 100 and firewalls	New	doug
17	what do you want for performance tests?	New	doug
18	what do you want for qa test to test for memory leaks?	New	doug
19	customer wants an expanded site search capability to include	Resolution In Progress	wchafner
20	a partners want to know what user loads we've tested	Customer Acceptance	wchafner
21	customer's product list has change to include [illegible]	New	doug
22	country-specific product listings	New	doug
24	add xml reporting capability	Refining Requirements	doug
26	authentication log should truncation size should be larger	Resolution In Progress	wgratzky
27	need new copyright notice in about box	Refining Requirements	pbohan
31	The colour code on the company logo is wrong on first splash screen	Resolution In Progress	doug
32	spinning globe spins the wrong way	Customer Acceptance	wgratzky
33	page should update automatically	Customer Acceptance	mfox

FIGURE 2 MKS Integrity Manager Web interface

matically detects the various IDEs you may have installed. SIE supports many of the popular IDEs, but I noticed that it didn't always support the most recent versions. For example, SIE supports Borland's JBuilder 3.0 Java IDE, but it failed to detect the JBuilder 4.0 or 5.0 releases I had installed on my desktop. However, it will support JBuilder 5.0 by the end of the year.

One of the advantages of SIE is that it does support a command-line interface as well as the more familiar GUI. Thus you can easily create command scripts to work with SIE from your own development environment as necessary. The GUI interface is useful to get the gist of the SIE environment (see Figure 1).

The SIE client connects to the server using WebLogic's proprietary protocol. I was able to connect from my laptop to the newly installed Integrity Server that was running on a separate machine. However, MKS also allows you to connect to a remote server with SIE. Using the list of sample users and the demonstration password (from the MKS Web site) I connected through my NetScreen firewall to an SIE server that was running on MKS's remote site. This feature makes SIE ideal for highly distributed teams (including teams that might include contractors who are working off-site).

Once you connect to a server, you'll be working with three basic source code objects: projects, sandboxes, and members. A *project* is a group of related files (source code, design specifications, graphic images, Web pages, etc.). As a developer, you'll work with project files through an interface known as *sandbox*. Through the sandbox you can add new *members*, which are the individual program files that make up your project. I was able to add Java source files and Jtest project test files to my local sandbox, and then synchronize the whole lot back into my project. SIE provides a wealth of security features and management functions for managing projects including lock management, check-in/out procedures, and revision tools.

One of the more powerful features of the MKS product line is its ability to integrate source code control with defect tracking through Integrity Manager. Integrity Manager provides tools for tracking defects, builds, and enhancements through a powerful workflow layer. IM works with the "change packages" that are created within SIE. For example, a change package can be associated with a group of Java source files, and IM can associate this same change package with a defect report. Thus project managers and customer service teams can track problem resolution all the way back to individual source modules.

Team members can access the Integrity Manager database through a Java client interface, or they can use a standard Web browser as shown in Figure 2. The combination of SIE 8.1 and the Integrity Manager provides a complete environment for managing source code and tracking defect resolution.

Summary

I found MKS Source Integrity Enterprise Edition 8.1 to be a very powerful and comprehensive source-code control solution. SIE 8.1 is clearly the "Cadillac" in its product category and I believe that teams of enterprise developers will find it to be a compelling solution. Geographically dispersed teams will find the remote-access features to be particularly enticing, though smaller development teams and workgroup developers may find SEI 8.1 to be overkill for their needs. However, MKS also offers Source Integrity Standard Edition for smaller, localized development teams.

JDJ Product Snapshot

- ***Target Audience:*** Project managers, Q/A team, programmers
- ***Level:*** Advanced developer
- ***Pros:*** Feature-rich product, support for client/server and Web interfaces, workflow integration with Integrity Manager 4.2, integration with popular development IDEs, flexible licensing policy, and command line interface
- ***Cons:*** Complex server configuration, overwhelming feature set, can be expensive: IM and SIE together are $1,400 per named user

Build to Spec!

Part 2

Avoiding pitfalls in the Application Portability Series

WRITTEN BY
LIZ BLAIR

While J2EE technology is designed to support portability, it's always possible to write nonportable code. In Part 2 of this series, we (the BluePrints development team) offer recommendations on how to avoid specific design, implementation, and deployment pitfalls that can compromise portability.

A comprehensive collection of the J2EE BluePrints application programming model can be found at http://java.sun.com/j2ee/blueprints. This month, our recommendations are:

- Deploy each Web application with a unique context root to avoid colliding with other applications.
- Use handles or remote object references to store session state in HttpSession.
- Be aware of changes in taglib DTD from J2EE 1.2 (JSP 1.1) to J2EE 1.3 (JSP 1.2).
- Always import all classes used in a JSP.
- Never throw resource-specific exceptions from BMP code.
- Wrap authentication code in helper classes.
- Always completely specify security roles.
- Uses of EJBContext.getCallerPrincipal().getName().

Unique Context Roots

Always specify a unique context root for every Web application you deploy to avoid naming collisions with applications already deployed.

The context root is the base path of a Web application, relative to the server's base URL. For example, if your server's base URL is http://j2eeserver.com:8000 and the context root is apps/myapp, then all components of the Web application will be accessed relative to http://j2eeserver.com:8000/apps/myapp. The <context-root> element of the Web application's deployment descriptor (web.xml) specifies the application context root.

Server behavior can be unpredictable if two applications are deployed to the same context root since it isn't clear which application should receive an incoming request. Servers and deployment tools will differ in how they handle context root collisions. A good deployment tool will detect context root collisions at deployment time and allow the deployer to choose a different context root. Selecting a unique context root for every application will prevent this problem from ever occurring. For example:

```
App1.ear:  war_runtime.xml
<web>
     <display-name>WebTier</dis
         play-name>
     <context-root>coolpetse-
         store</context-root>
</web>

App2.ear:  war_runtime.xml
<web>
          <display-
name>WebTier</display-name>
          <context-root>exoticpet-
sestore</context-root>
</web>
```

The code snippets in the preceding example show how App1 and App2 specify unique <context-root> entries so that requests can be clearly channeled to the appropriate application Web page.

> "Servers and deployment tools will differ in how they handle context root collisions"

Store Enterprise Bean Handles or Remote References in HttpSession

It's perfectly legal to store either handles or remote references to enterprise beans in HttpSession.

Every enterprise bean remote reference extends EJBObject. A remote reference's method EJBObject.getHandle() returns a handle, which is a persistable reference to the object. A handle can be serialized and persisted, and then at some later time deserialized and converted back into a remote reference to the original bean (if the bean still exists).

Some application developers believe that a handle is the only way to store a reference to an enterprise bean in HttpSession, but this isn't the case. In fact, an enterprise bean reference (acquired from a home interface create or finder method) can be stored directly in HttpSession without first converting it to a handle. A compatible application server that supports distributable servlets is required by the specification to correctly manage enterprise bean references stored in HttpSession. Therefore, remote references (that extend EJBObject) can be stored directly in HttpSession with no loss of portability.

For more information refer to the J2EE Platform Specification 1.2, section 6.6.

New TagLibrary Format: J2EE 1.2

If you're migrating applications from J2EE 1.2 to J2EE 1.3, you may need to update any TagLibrary descriptor (TLD) files in your application to the new format. The TLD file DTD has changed from J2EE 1.2 (using JSP 1.1) to J2EE 1.3 (JSP 1.2).

Here's an excerpt from a JSP 1.1 taglib.tld file:

```
<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>j2ee</shortname>
```

Here is the same snippet for JSP 1.2:

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>1.2</jsp-version>
  <short-name>j2ee</short-name>
```

Your application server will hopefully provide good migration tools that will update these files for you; otherwise, you may have to check and edit them manually.

The samples used in this example are just a few of the JSP DTD tag name changes. See the relevant sections of the JSP 1.1 (section 5.3.5) and JSP 1.2 (Appendix C) specifications for details.

Always Import Classes Used in JSP Pages

With the exception of classes that are guaranteed to be imported, always import any class you use in a JSP page. This may seem like an obvious point, but this item addresses a subtlety of Tomcat – and possibly other JSP containers. Tomcat (version 3.0) found in the J2EE 1.2 Reference Implementation imported more packages than those required by the specification (this has been fixed in the latest release). In particular, it was possible to use java.io.PrintWriter without importing it. Pages that did so would work in Tomcat but not in other JSP page environments. Importing all classes used by a JSP page will prevent this problem from occurring.

The only packages that can portably be used without importing are java.lang.*, javax.servlet.*, javax.http.*, and javax.servlet.jsp.*. All other packages should be imported in the JSP page.

JSP containers other than Tomcat may import classes beyond those required by the specification, but relying on the presence of those classes may make your JSP pages nonportable. See the section on the import keyword in the JSP 1.2 pfd 2 specification, section 2.10.1.1.

Never Throw Resource-Specific Exceptions from BMP Code

Entity beans that manage their own persistence should never throw exceptions (such as SQLException) from BMP code; instead, they should define and use application exceptions to report persistence errors.

Bean providers should be especially careful to avoid including back-end resource-specific details in their components' interfaces since doing so may limit where the components might be used. One easily overlooked form of resource dependence is the set of exceptions a method may throw. BMP methods don't necessarily use a SQL database to manage their persistence, so SQLException doesn't belong in the BMP method signatures.

Instead of throwing SQLException, define system- and application-level exceptions for the class and throw those exceptions in response to error conditions.

“An application developer shouldn't make any assumptions about the returned value when designing an application”

The BluePrints program recommends the Data Access Object (DAO) design pattern (http://java.sun.com/j2ee/blueprints/design_catalog/dao) to encapsulate a back-end resource, such as a SQL database. A DAO provides access to a system resource without reference to how that resource is implemented. For example, a DAO might offer methods to load, update, and delete a persistent object, and methods to set and get that object's properties, while hiding the implementation details of how those operations occur.

The DAO can catch resource-specific exceptions and translate those exceptions to the custom system- or application-level exceptions described above.

See Listing 1 for an example from the Java Pet Store. In method deleteAccount(), a SQLException is translated to an AccountDAOSysException, which extends java.lang.RuntimeException to indicate a system-level exception.

It's worth noting that the Account bean indicates application-level errors with subclasses of AccountDAOAppException (such as AccountDAODBUpdateException). An application-level exception is typically thrown when a recoverable error has occurred, such as an incorrect user password. A system-level exception (like AccountDAOSysException above) is typically thrown when a nonrecoverable error has occurred, such as a database crash. Learn more about the differences between system- and application-level exceptions in Chapter 12 of the EJB specification.

Security: User Authentication

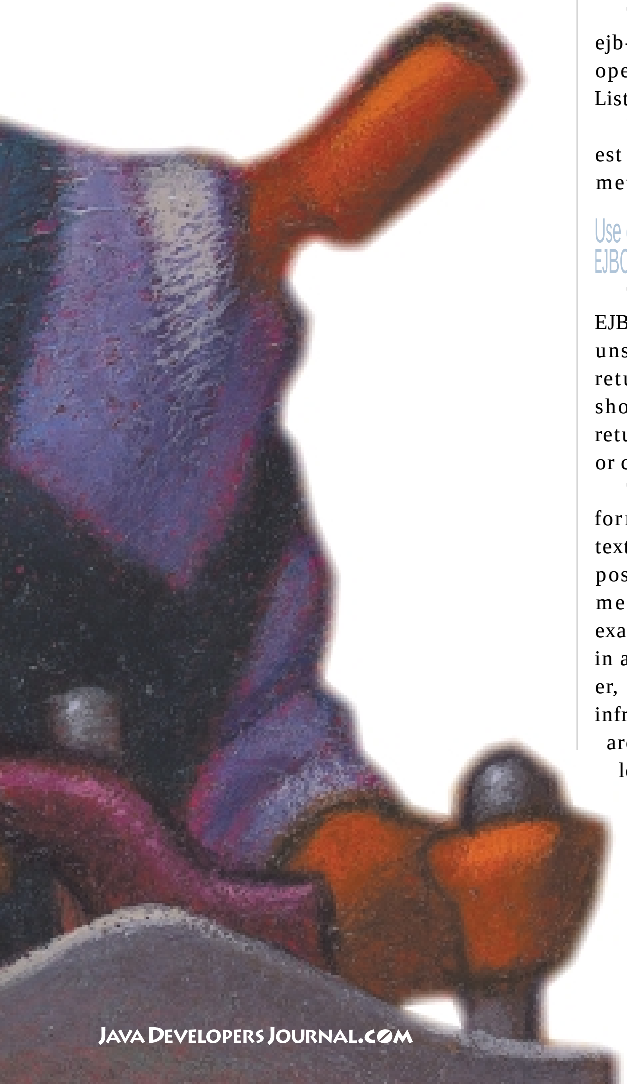
Keep all user login code in classes separate from your application classes so you can reimplement them if you

port the application or change your user authentication mechanism. The J2EE platform's long-term expectations are that developers won't be writing authentication functionality directly into their applications; however, developers will do so as long as the container-provided mechanisms aren't adequate to suit the needs of an application. If this is done, it would be wise for the developer to isolate the code so that it can be easily removed as containers become more capable.

Isolating the code specific to an authentication mechanism makes porting easier since just that part of the code will need to be rewritten. See Scenario #3 on packaging from the ***JDJ*** "Build to Spec!" article (Vol 6. issue 7) for more information.

For managing user authentication, consider using Java Authentication and Authorization Service. JAAS allows services to authenticate and enforce access controls on users. To learn more about JAAS, see the resources section at the end of this article.

See also Table 6.2 of the J2EE 1.2 platform specification, where the J2EE security permission set for each type of component (application clients, applet clients, servlets/JSPs, EJBs, and so on) are defined.



Define Method Permissions When Security Roles Are Used in EJBs

If your application uses security roles, always specify method permissions with the <method-permission> element in the EJB deployment descriptor (ejb-jar.xml) for all of the EJB client-view methods.

There is a gray area in the J2EE 1.2 specification when it comes to defining the behavior of an unspecified method permission if security roles are used. The gray area has, legitimately, been interpreted inconsistently by container vendors. This means that EJB containers may not behave consistently for EJB methods with an unspecified method permission when security roles are used.

For example, some container vendors may interpret unspecified method permissions to mean *none,* essentially an uncallable method for no user access; others may interpret unspecified method permissions to mean *everybody* for wide-open user access. In any case, if you use security roles and want the security behavior to be portable across container vendors, specify your intentions clearly with well-defined method permissions.

The application developer may choose to create a role named *everybody* and then map this role to wide-open user access. This assumes the container has the ability to map the *everybody* role to a set of users.

The *everybody* role could then be used in the ejb-jar.xml files for those methods where wide-open user access is allowed. For an example, see Listing 2.

Refer to the J2EE 1.3 specification for the latest changes for using security roles and EJB method permissions.

Use of Security APIs: EJBContext.getCallerPrincipal().getName()

The J2EE 1.2 specification leaves the result of EJBContext.getCallerPrincipal().getName() as unspecified in terms of the requirements of the returned value. An application developer shouldn't make any assumptions about the returned value when designing an application or component. Why?

The main reason is that there's no standard format for the return value of EJBContext.getCallerPrincipal().getName(). As of proposed final draft2, the EJB 2.0 specification mentions that calling methods "might, for example, use the name as a key to information in a database." In general this will work; however, there's a caveat that in complex security infrastructures this value might change. Here are some scenarios where this might be a problem:

- When running a server from one vendor and then switching to a server from another vendor while keeping the same database (e.g., using BMP)
- When running two servers from different vendors where a bean on each server needs access to that row

oak grove systems

Need a
JAVA Workflow
Solution?

reactor 5

The Process Engine for Enterprise JAVA Applications

Platform & Database Independent

J2EE Application Server Neutral
- BEA WebLogic, IBM WebSphere, JBOSS...

XML/HTML In/Out for Interoperability
- With e-business applications
- With ERP, CRM & legacy systems
- With other workflow systems

Flexibility & Speed
- Faster workflow development
- Quicker process modifications & changes
- Easier reuse of processes
- Simpler integration

DOWNLOAD @
www.oakgrovesystems.com

Download A Fully Functional Reactor 5.0 Evaluation Toolkit at www.oakgrovesystems.com

Contact Information
info@oakgrovesystems.com
626-296-6390 *(phone)*
877-777-6310 *(toll free)*
www.oakgrovesystems.com

In short, the call itself is portable but the return value isn't.

One portable example might be simply as a display value in the Web tier, for example:

```
"Hello <%=
request.getUserPrincipal().getName(
) %>"
```

Another possibility is as a log message in the EJB tier for auditing, for example:

```
"8:15 AM: User " +
ejbContext.getCallerPrincipal().get
Name() + " withdrew money."
```

Until the return values are standardized from this API, its use to produce values that must be compatible with values that may be produced by other container vendors isn't a recommended portability practice. Look for this item to be addressed in future versions of the J2EE platform specification.

We Want to Hear from You

The J2EE BluePrints team is interested in hearing about your experiences developing applications for the J2EE platform and your experiences with BluePrints. Send feedback and comments to j2eeblueprintsinterest@sun.com. Special thanks goes to the J2EE development community for their continued commitment to the J2EE platform.

Resources

1. Check out the J2EE BluePrints Web site at http://java.sun.com/j2ee/blueprints, where you can download the Java Pet Store sample application, the BluePrints book *Designing Enterprise Applications with J2EE*, and view the complete BluePrints design pattern catalog – all for free.
2. Download the J2EE Platform Specification, as well as other J2EE developer support documentation and software (including the free J2EE Reference Implementation), from http://java.sun.com/j2ee.
3. View the current list of J2EE licensees who have passed the Compatibility Test Suite at http://java.sun.com/j2ee/compatibility.
4. Learn more about JAAS, the Java Authentication and Authorization Service, at http://java.sun.com/products/jaas.

AUTHOR BIO

Elizabeth Blair is a staff engineer with Sun Microsystems, where she leads the Java 2 Platform, Enterprise Edition BluePrints development team. Liz contributed to the recent Java Pet Store sample application, with an emphasis on the tests for EIS and EJB architecture. In the past she has worked on the Compatibility Test suite (CTS) for the J2EE platform, the J2SE platform, and the WABI (Windows Applications Binary Interface) projects.

liz.blair@sun.com

Listing 1

```
public class AccountDAOImpl implements AccountDAO {

 public void remove(String id) throws AccountDAODBUpdateException,
                                      AccountDAOSysException {
          deleteAccount(id);
   }

 private void deleteAccount (String userId) throws
                                   AccountDAODBUpdateException,
                                   AccountDAOSysException {
        String queryStr = "DELETE FROM Æ";
        PreparedStatement stmt = null;
        Debug.println("queryString is: "+ queryStr);

        try {
            getDBConnection();
            //stmt = dbConnection.createStatement();
            //int resultCount = stmt.executeUpdate(queryStr);
            stmt = createPreparedStatement(dbConnection, queryStr);
            int resultCount = stmt.executeUpdate();

            if (resultCount != 1)
                throw new AccountDAODBUpdateException
                ("ERROR deleteing account from ACCOUNT_TABLE!! resultCount =
"+
                 resultCount);
        } catch(SQLException se) {
            throw new AccountDAOSysException("SQLException while removing "
+
                            "account; id = " + userId + " :\n" + se);
        } finally {
            closeStatement(stmt);
            closeConnection();
        }
    }
}
```

Listing 2

```
<security-role>
      <role-name>manager</role-name>
    </security-role>

    <security-role>
      <role-name>everybody</role-name>
    </security-role>

    <method-permission>
       <role-name>manager</role-name>
       <method>
         <ejb-name>EmployeeService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>setPerformanceRating</method-name>
         <method-params>
            <method-param>java.lang.String</method-param>
         </method-params>
      </method>
    </method-permission>

    <method-permission>
       <role-name>everybody</role-name>
       <method>
         <ejb-name>EmployeeService</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>getPerformanceRatingForm</method-name>
         <method-params/>
      </method>
    </method-permission>
```

Download the Code!
www.JavaDevelopersJournal.com

WRITTEN BY GEORGE GREY

J2ME Calls and J2SE Swings

Java has come a long way in its short life. Linux just celebrated its 10-year anniversary; for Java that milestone is still more than three years away. Who really understood, seven years ago, that Java would become the standard platform for delivering content over the Internet and for running applications on mobile phones – even the de facto platform for future smart-card technology? And what will the next few years bring for Java?

In the information appliance space we see a proliferation of different devices: PDAs, smart phones, Web tablets, in-car Internet access, games consoles, set-top boxes, home servers, wireless enterprise devices for vertical markets, and countless embedded devices with Internet access. Emerging wireless and display technologies are driving the capabilities of these devices forward.

As a result, application demands on the underlying operating system and platform are increasing dramatically. Multitasking is no longer optional; future devices must be capable of simultaneously handling 1–100 megabit wireless data streams, providing immediate response to a user action while smoothly playing a background multimedia stream. And all this using a featherlight battery pack – no 2GHz 70 watt Pentium 4 in these devices.

With this proliferation of devices, and a bewildering number of operating system choices, Java is emerging as the common platform standard that developers can use to ensure portability across different devices. Historically, however, Java hasn't been a major player in the information appliance space. What's changed?

In theory, solving the "too big, too slow" reputation that Java gained in its early days is easy: make it smaller and make it faster. For small devices this is exactly what J2ME has done. J2ME has emerged as the preferred solution to the problem of running Java applications on devices with restricted functionality. With a limited API, well suited for devices such as mobile phones, J2ME CLDC offers a viable platform for devices with processor speeds around 50MHz, and memory footprints well under 1MB.

For more powerful wireless appliances, there are many options: accelerated PersonalJava and J2ME CDC JVM technologies for Windows CE, Palm OS, Wind River, and others; various flavors of Java on various flavors of Linux; and now a full J2SE-based OS from SavaJe. The Java bandwagon in the IA space is sturdy. And it's rolling with recent announcements by major players such as DoCoMo, Motorola, Nokia, and Sharp, which have started shipping or have announced wireless devices with J2ME and PersonalJava capabilities.

What of the future, and should you write to J2ME CLDC, J2ME CDC, or J2SE? Moore's Law suggests that wireless handheld devices will get more and more powerful. Today's iPAQ contains the computing power of a three-year-old PC. Will everything eventually run J2SE (or will it be Java 3 by then)?

Unlikely. Only you know whether your application needs J2ME or J2SE. It's as much about what the target device is capable of as it is about your application's functionality. J2ME is the right choice for many applications running on phones, embedded devices, or toys, where you don't want or need AWT (let alone Swing), J2SE security, and programming APIs as diverse as Java Collections and IEEE floating point libraries.

J2ME is a great platform for small devices, but give me a powerful PDA, an enterprise wireless terminal, or a Web tablet, and I want more – a lot more. In a wireless PDA I want a phone, pager, PDA, e-mail device, an Internet-access device with high-resolution color display, all rolled into a single package the size and weight of a Palm m505.

I want to see applications written with the rich and flexible UI of Swing and Java 2D. I expect to rely on full security for wireless networking, end to end. I want seamless Internet and voice communications over wide-area networks, personal wireless networks, and corporate LANs. Certainly a number of J2SE subset/J2ME superset profiles are moving along the JSR process, but I want a full version of Java and complete applications portability, not a subset.

We shouldn't proliferate multiple flavors of J2ME. It hurts application portability and it dilutes the powerful message of the Java platform. Looking to the future I see a huge population of small devices running "fit for purpose" J2ME, and more powerful wireless devices with QVGA+ displays, 200MHz+ CPUs, and 32MB+ memory running J2SE applications – the same application in your car, on your desktop, or on your wireless PDA.

The exciting thing about today's information appliances is that, for the first time, we have the opportunity to develop and deploy powerful Java applications on a wide range of devices – from smart card to mobile phone, from PDA to PC. Both J2ME and J2SE can be used to build applications for these devices: match the device capabilities and your application API requirements to determine which to use.

So I, for one, am looking forward to Java's tenth anniversary – and Java 3 Standard Edition Plus on my mobile phone/PDA/Web tablet voice-activated gizmo.

george.grey@savaje.com

AUTHOR BIO

George Grey is CEO of SavaJe Technologies Inc., producer of SavaJe XE, an operating system for information appliances with J2SE support and bundled applications. Previously he was CEO of Psion USA; before that he founded and built Tadpole Technology, developer of the notebook-sized SPARC computer.

J 2 S E   I N D E X

Using the Façade Pattern for the Java Internationalization API

Produce globally enabled software

WRITTEN BY DAVID GALLARDO

One of the outstanding things about Java is that it facilitates building internationalized programs, programs that work in various languages and regions. Java has used Unicode as its internal character set since version 1.0. In Java 1.1 another important feature was introduced – an internationalization API.

Although the Java Internationalization API generally provides well-designed, object-oriented ways to internationalize an application, it may be a misnomer to call it an API because it's not a single class or package. Rather it's a set of loosely related classes scattered throughout several different packages. The many features and the relationships between them can be overwhelming, especially when you're trying to do something relatively straightforward and standard.

This complexity can be tamed by using the Façade pattern. This pattern suggests that you create a simple interface for a complex set of functionality by creating one object that mediates access to another set of objects that are related in some complex way. This is a convenient way to implement the Java Internationalization API.

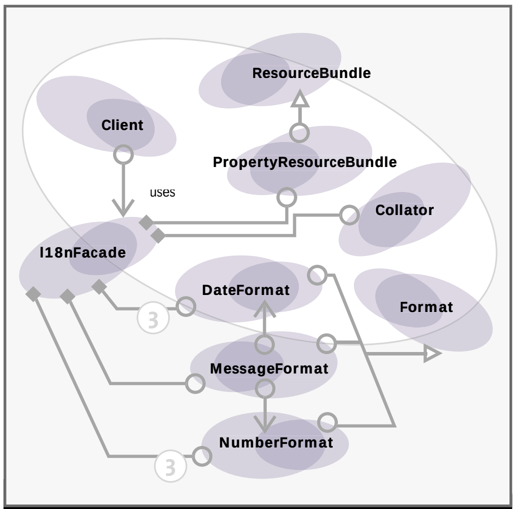


FIGURE 1 I18nFaçade class overview

Internationalization and Localization

There are two aspects to producing globally enabled software. The first is internationalization (sometimes abbreviated i18n), which means producing software that doesn't have built-in assumptions about language and cultural conventions. This should first be considered in the design phase of a project.

The second is localization (l10n), which means adapting software to work in a specific language. This is usually done in parallel with software development for at least one language and later for additional languages.

Properly internationalized software can be localized easily, without reengineering, recoding, and recompiling, by someone who specializes in translation, not programming. Each localized version must still be tested, of course, but the scope of testing can be significantly more limited than a complete test of the entire application.

Locales

An internationalized Java application uses the concept of locale to select the language and set of cultural conventions to use at runtime. A locale typically consists of two parts: a language designation and a country designation. Java uses the two-letter codes from ISO-639 to designate language and the two-letter codes from ISO-3166 to designate country. Java optionally allows a third part, a variant, that allows support for dialects within a country, for example. Java notably uses a variant to provide support for the Euro.

The Façade Pattern

The classes that we'll be using for Java internationalization generally have three things in common:

1. They have an abstract class with one or more factory methods to obtain concrete classes. The concrete classes are usually cast up to the abstract method, and we're unaware of what the actual concrete class is.
2. They usually depend on a locale for their behavior, normally specified when the factory method is called to

obtain the concrete class.
3. They provide one or more methods with various options that provide the functionality we need, such as formatting a date according to the current locale's conventions or comparing two strings.

As suggested by the Façade pattern, we create a single object, the I18nFaçade class (see Listing 1), and use it to organize and manage the various internationalization classes. (Listings 1 and 2 can be found on the ***JDJ*** Web site, www.sys-con.com/java/sourcec.cfm). We set the locale once when the Façade class is instantiated, then instantiate each of the various internationalization API classes and delegate to them as needed.

Various constructors are provided in the I18nFaçade class to allow users to specify the two options we don't allow them to change later, locale and ResourceBundle.

Users can supply a locale directly, or country and language codes from which we create a locale, or nothing, in which case we obtain the default locale from the system. Here's an example using language and country codes:

```
// Use US English, default
ResourceBundle
  I18nFacade intl = new
I18nFacade("en","US");
```

The Façade allows the use of only a single ResourceBundle – a repository of localized strings (contained in a set of related files, as we'll see later). A default ResourceBundle is specified in the Façade, which presumably is the one used for general user interface text. It's good practice though to group messages into different ResourceBundles according to type, so we've also made it possible to specify the ResourceBundle when the Façade class is instantiated. Here's an example:

```
// Use Mexican Spanish and
AppErrorMessages
 // ResourceBundle
   I18nFacade intlSpErrors =
 new I18nFacade(
 "es", "MX",
"AppErrorMessages);
```

Now we look at the ways we manage and delegate to the internationalization API classes. We briefly note that we often know which class we need to call based on the type of the parameter or parameters we're passed. We can create overloaded methods, especially in the case of formatting classes, which can help reduce complexity.

The Façade pattern is not intended to limit access to the underlying component classes. We've provided get() methods that return the underlying classes so that anything that can't be done through our simple methods, such as setting custom formatting patterns, can be done by accessing the classes directly. Listing 2 provides code that exercises the functionality in the I18nFaçade.

Text Translation

Perhaps the most obvious issue for applications that are intended to work globally is that we need to be able to translate the user-visible text. In the past people often wrote applications with print statements containing hard-coded text. Translating an application like that means going through all the source code, identifying user interface text, changing, recompiling it, and hoping we didn't break something else in the process. Separating user text by putting it in resource files maintained separately from the source code is a good way to solve this problem. (We can do this for graphics and sound files too, by using a convention involving directory paths and filenames, and putting the path and filename in the resource file.) To add support for another language we add another resource file.

The Java API provides an abstract class, ResourceBundle, with two important subclasses, PropertyResourceBundle and ListResourceBundle, to support resource files. These classes allow us to use a tag as a key to looking up an object in a set of files. Which specific file it uses depends on the current locale.

The most convenient of these to use is the PropertyResourceBundle, which simply uses text files, .properties files, that don't need to be compiled. The other options require us to create classes that contain the key and value pairs and compile them.

Let's look at an example. Suppose we have a program "Hello.java". Instead of using a string such as "Hello!" directly in our print statement, we look up the localized string using a key, such as "HI":

```
// Get a localized String using a
Tag
  String hello = intl.format("HI");
 System.out.println(hello);
```

In ResourceBundle, we have specified a base filename for the ResourceBundle. This means we must have a default properties file HelloMessages.properties:

```
# AppMessages.properties
#
# Default - English

HI = Hello!
BYE = Goodbye!
HI_NAME = Hello, {0}
```

This is the default version because the filename consists of a basename "HelloMessages" plus ".properties" and no locale information.

A French version might be called "HelloMessages_fr.properties", where "fr" specifies the language and might contain:

```
# AppMessages_fr.properties
#
# French

HI = Bon jour!
BYE = Au revoir!
HI_NAME = Bon jour, {0}!
```

There may also be a file called "HelloMessages_fr_FR", which would be specifically French French. Finally, there may also be a file "HelloMessages-_fr_FR_PARIS", which would be Parisian French.

When we search for the value according to the key we provided, the ResourceBundle will first try to find it according to the locale we specified when loading the ResourceBundle using the basename, language code, country code, and variant, each separated by underscores followed by ".properties". If this fails, it ignores the country code and looks for basename and language code separated by underscores and followed by ".properties". If this fails, it will look for just the basename followed by ".properties". This provides a nice system of defaults.

Message Formatting

Another issue to deal with is that programs often construct strings dynamically, such as by concatenation or substitution. Generally speaking, this should be avoided. Different languages put words and phrases together in ways that are hard to anticipate. For these parts to be correctly translated in isolation, each part, including its context, must be carefully and consistently documented by the programmer.

Sometimes it's unavoidable. We need to construct sentences with information that can only be known at runtime, such as when we inform users

English – U.S.	12,345.678
Spanish – Mexico	12,345.678
French – France	12_345,678
German – Germany	12.345,678

TABLE 1 Number formats

English – US:	$12,345.68
Spanish – Mexico:	$12,345.68
French – France:	12_345,68 F
German – Germany:	12.345,68 DM

TABLE 2 Currency formats

English – U.S.:	8/27/01	August 27, 2001
Spanish – Mexico:	27/08/01	27 de agosto de 2001
French – France:	27/08/01	27 ao_t 2001
German – Germany:	27.08.01	27. August 2001

TABLE 3 Data formats

English – U.S.:	5:36 p.m.	5:46:39 p.m. EDT
Spanish – Mexico:	05:37 p.m.	05:46:40 p.m.EDT
French – France:	17:38	17:46:40 EDT
German – Germany:	17:38	17:46:40 EDT

TABLE 4 Time formats

of their last login time or remaining balance. If other user-interface solutions won't do, the Java Internationalization API provides a class for message formatting, the MessageFormat class.

The MessageFormat class is one of five formatting classes, all subclasses of the abstract Format class, that we'll include in our Façade class. It shares many characteristics with the rest of its family. The MessageFormat class is an abstract class but has a static factory method, getInstance(), that returns an appropriate concrete class cast to MessageFormat.

The method we're interested in, format(), takes a pattern string, followed by an array of objects. It parses the pattern string, identifying placeholders within the pattern, which it replaces with other strings. The simplest nontrivial pattern is text with numbers in braces, such as:

```
"You have a message from {0} wait-
ing."
```

where the first object in the array of objects would be a string to be inserted in place of {0}.

Additional options are available that specify that the object in the array is to be formatted as a date or time, for example:

```
"A message from {0} arrived at
{1,date,short}."
```

When deciding whether to use these features, consider how it will affect the task of translation; keeping things simple reduces cost and errors when translating.

Number Formatting

The format for numbers, date, time, and currency vary greatly from place to place. Numbers in U.S. English, for example, use the period as the decimal separator and a comma as the thousands separator, whereas some locales use the reverse, as shown in Table 1.

(*Note:* The French number used punctuation, replaced here with an underscore, that could not be displayed correctly when I ran the code in a DOS window on Windows 98.)

These results were obtained by using code similar to the following for English:

```
  I18nFacade intl = new
I18nFacade("en","US");
  System.out.println(intl.for-
mat(12345.678));
```

Currency

Currency is a special case of formatting numbers, but there are two additional issues regarding currency. The first is more or less obvious: formatting currency is not the same as converting currency. Displaying yen as dollars is a serious error. If the application is intended to work only with one currency, even if it works with multiple languages, this is not a problem, except that a separate locale needs to be used for currency.

By default the currency locale is the same locale specified when the class was instantiated. The Façade class has setCurrencyLocale() methods to allow specifying a separate currency locale.

The second issue regarding currency is that we need to carefully consider the numeric type we use to represent it. When we're talking about most quantities, a little rounding here and there is not a problem, and float and double are fine. This isn't usually the case with currency; a penny here and a penny there often matter. This suggests that we might use an integer type, such as long, with a decimal offset. In some countries the currency for large transactions can require support for large denominations that exceed any of the native Java types. In that case it may be desirable to use a class like BigDecimal to represent numbers, perhaps wrapping in a Currency class.

For the purpose of simplifying our example, we assume that the numbers being formatted are known to represent the correct currency, that their magnitude will not exceed the capacity of the native types, and that rounding errors will be insignificant. Table 2 shows some of the currency formats.

(*Note:* The French number used punctuation, replaced here with an underscore, that could not be displayed correctly when I ran the code in a DOS window on Windows 98.)

These results were obtained by using code similar to the following for English:

```
  I18nFacade intl = new
I18nFacade("en","US");

System.out.println(intl.currencyFor
mat(12345.678));
```

Date Formatting

Date and time formats vary greatly. For dates, the order and punctuation between day, month, and year differ. In addition, verbose formats include the names of the days and months (or abbreviations) that vary according to the language. Table 3 shows short and medium formats for dates for a few countries.

(*Note:* The French name for August includes a character, displayed here as an underscore, that couldn't be displayed correctly when I ran the code in a DOS window on Windows 98.)

These results were obtained by using code similar to the following for English:

```
  I18nFacade intl = new
I18nFacade("en","US");
  Calendar cal =
Calendar.getInstance();
  Date date = cal.getTime();

System.out.println(intl.dateFormat(
date));
```

Another issue that the standard Java Internationalization API doesn't resolve is that different calendars are used in some countries or for certain purposes. Most of us are familiar with the Gregorian calendar and this is the only one for which a class, GregorianCalendar, is provided in the Java API. (This is what the Calendar.getInstance() factory method in the example above returns.)

You shouldn't use the methods in java.util.Date to construct date Strings. These have been deprecated as of Java 1.1 because they only allow you to use a Gregorian calendar.

IBM has open-source Java classes in its ICU4J project that support other calendars, such as BuddhistCalendar, HebrewCalendar, and JapaneseCalendar. See http://oss.software.ibm.com/icu4j/doc/index.html for more information about these classes and the calendars they support.

Time Formatting

Table 4 shows the short and long time formats for the same countries as above:

The code is similar to what we used to obtain dates in the preceding example, except we use the timeFormat() method:

```
  I18nFacade intl = new
I18nFacade("en","US");
  Calendar cal =
Calendar.getInstance();
  Date date = cal.getTime();

System.out.println(intl.timeFormat(
date));
```

Parsing Numbers, Dates, and Time

The same issues we considered for formatting apply to parsing user input. In addition, processing user input requires that the format we expect is clear to the user. Finally, we must validate and properly handle bad input. Specifically, the application must be able to catch and appropriately handle ParseException.

It's also the responsibility of the application to properly interpret the returned value. The dateParse() and timeParse() methods both return a Date, but in the first case the time portion is zero, and in the second the date is zero. Here's a sample call to parse a date:

```
  Date userDate;
  try
 {
     userDate =
intl.parseDate("12/8/01");
  }
  catch (ParseException e)
 {
     // Handle bad input
  }
```

The parseNumber() method returns a Number object and it's up to the application to convert it to the proper primitive using the intValue(), longValue(), floatValue(), and doubleValue() methods, for example:

```
  Number userNumber = null;
// Called parse …
  long userLong =
userNumber.longValue();
```

Sorting, Searching, and String Comparisons

In English, a once common algorithm for comparing strings – the essential operation in performing a sort – is pretty easy: convert the strings to uppercase (assuming we want a case-insensitive sort), then compare them character code by character code. This doesn't work as a general algorithm, however, because many languages use characters outside of A–Z and a–z, and some languages, such as Chinese, don't even have the concept of upper- and lowercase. Other complications include expansion, where a language sorts certain letters as though they were two, such as the German eszett (\u00DF), which is treated as two ss for sorting purposes, and contractions where two letters such as “ch” in Czech are treated as a single letter that sorts between h and i.

Besides optionally taking case into account, we also have the option of taking diacritics into account, depending on the situation. For sorting, we may want the letter “a” to sort before “á”, but for searching we may wish to consider them equal.

For these reasons (among others too numerous to list here), we shouldn't use the String.equals() and String.compareTo() to compare two strings. The Java Internationalization API has a class Collator that we should use instead. This class, like the format classes, is an abstract class with a factory method that returns a concrete class, RuleBasedCollator, cast up to Collator. Collator has a method compare() that will compare two strings using the default rules for the locale. Collator also has a property, Strength, that allows us to select between locale-specific options such as case and accent sensitivity.

Our Façade has a method compare() that will obtain a collator for us (if it doesn't already have one) and return the results of comparing two strings:

Author Bio

David Gallardo is an independent software consultant specializing in internationalization, Java and database development. Previously he led database and internationalization development at a B2B e-commerce company.

```
  I18nFacade intl = new
I18nFacade("en","US");

if(intl.compare("Apples",
"Oranges")
  {
// do something based on
this result
  }
```

The concrete class is a rule-based collator. Using it is expensive since it must parse the string according to many rules and expand or contract characters based on diacritics and other language rules. If more than a few strings are going to be compared more than a few times, such as when sorting a large set of data, it may be desirable to capture the result of this parsing in advance of sorting by calling the getCollationKey() method, perhaps as data is entered. The CollationKeys can be saved and later used to perform the comparisons instead of the strings themselves.

Additional Considerations

The intention of the Façade was to make it easier to use the Java Internationalization API. Some features are not included because it wouldn't really make those features easier to use and would only make the Façade more complicated. Other features are not included because they require a different solution or approach.

Text Boundaries

The BreakIterator class provides methods for locating character, word, and sentence boundaries. While it's relatively easy to detect these boundaries in English, many languages present difficulties. Chinese and Thai, for example, have no spaces between words. Spanish can have punctuation at the beginning of a sentence. Thai has no punctuation whatsoever. It's important to be aware of these issues, but locating text boundaries isn't something that's usually useful.

While it's indispensable for doing such things as word wrapping, dictionary lookup, and indexing, using a Façade isn't going to make the job significantly easier – it's probably best to use the BreakIterator class directly in those few specific cases where it's required.

Display, Input, and Output

I've also omitted support in the Façade for graphical display, and input and output functionality (including character set conversions), because these are entirely unlike the other functionality in the Façade. There are other design patterns better suited for this type of problem.

Conclusion

Design patterns help software developers by allowing them to consider what the problems have in common. The insight gained from solving one can be carried over when solving the next one. The problem with using the Java Internationalization API – many classes, loosely related, with many features – describes the type of problem the Façade pattern is intended to solve. Using this pattern does indeed make the Java Internationalization API easier to use.

Reference

Grand, M. (1998). *Patterns in Java: A Catalog of Reusable Design Patterns*. Wiley. pp. 205–211.

dgallardo@mediaone.net

JBuilder 5

by Borland Software Corporation

REVIEWED BY TOM INGLIS
thomas@n-ary.com

specs

Borland Software Corporation
100 Enterprise Way
Scotts Valley, CA 95066-3249
Phone: 831 431-1000
Web: www.borland.com
E-mail: customer-service@borland.com

Test Environment
Windows 2000 Professional, Intel PIII-450MHz, 128MB RAM
Windows 98SE, AMD Duron-800MHz, 512MB RAM

Supported Platforms
Solaris: ULTRASPARC2, Solaris 7,8
Windows: 98, 2000, NT 4.0
Linux: Red Hat 6.2, Mandrake 7.2, Caldera Systems 2.3, SuSE 6.3, TurboLinux 6.0
Mac: OS X (4th quarter of this year)

In June 2001, Borland released the latest version of the JBuilder series. Borland is an ever-popular presence in the Java IDE market, but with most IDEs now offering the same functionality, is there anything to set JBuilder apart?

It must be said from the outset, JBuilder 5 boasts some very nice features "on the box." IDEs have been screaming out for the packages and integration of technologies that JBuilder 5 promises are "under the hood."

- Rapid J2EE application development
- XML tools
- Deployment to multiple application servers
- Team development file management integration with MS SourceSafe, Rational Clearcase, and CVS
- JSP
- JavaBeans

Features

Borland seems to have packed everything possible into JBuilder 5. Initially, there were some reservations regarding how all these features, such as team development tools, would be presented and managed. The chaps at Borland would have to get it right for it to be both practicable and useful.

After using the IDE for a couple of weeks it is my happy duty to report that this is indeed the case.

The main interface is well laid out and highly configurable, as you would expect, with the normal array of debugging and project control options present by default. Almost everything is menu-driven and those friendly wizards we've all become accustomed to are available for most of the more commonly used features.

These wizards can sometimes be more of a nuisance than they are worth. They seem to take away some of the learning process regarding what the IDE is doing when it sets all our classpaths and creates neat packages. But, as wizards go, the ones I tried were easy to use and performed the job well.

We've all used applications that fill the screen with almost every available option/feature. Information overload if you like. Given that most of us don't have screens the size of a small cinema, it usually takes a bit of pruning to get a suitable screen.

This is not the case with JB5 (see Figure 1) since the information is hidden behind tabs. This works extremely well as it enables information to be displayed when you want it as opposed to the IDE "guessing" what you would like.

You may have noticed the Cocoon application and, no, it isn't a misprint. One of the more pleasing elements is the ability to build straight to the familiar Cocoon Web ARchive format. As Java finds its way into more and more of today's (and tomorrow's!) Web sites, the ability to deploy solutions straight to a .war package ready for distribution is both an excellent idea and well implemented.

JBuilder 5 includes a Tomcat 3.2 plugin (the Enterprise Edition also has Borland and WebLogic) so add a third-party Web server, such as Apache, and it's easy to see how you can compile, run, debug, test, and finally deploy from the comfort of one interface. The distribution also includes Borland's own AppServer 4.5.

The supplied wizard does it all for you by providing a complete Tomcat-ready solution with context mapping. What more could you ask for?

This doesn't mean it's painless. You still have to ensure dependencies/library files and that everything is set up "just so" from within JBuilder, but you would have to do this anyway! Both Enterprise and Professional editions have this built-in.

Another tasty feature is the support for version control packages such as CVS, SourceSafe, and Clearcase. I haven't used all

Borland
JBuilder 5
The leading Java development solution
7th Annual Borland Conference
Borland
JBuilder
The leading Java development solution

these packages but, of the three, CVS is familiar and bundled with JBuilder 5. Hurrah for open source.

This process would normally entail having at least two applications the IDE and the control package, ave it all integrated in one place is an excellent aspect of this impressive IDE.

By using the menu options and filling in the blanks, I created a repository and incorporated it into CVS within minutes, giving full version control.

The Enterprise edition allows you to control and create projects deployable to work stations (see Figure 2).

Never has it been so easy to maintain version control over even extremely large projects.

Deployment

As the features of JB5 were explored it quickly became apparent how much thought had been given not only to the possible deployment of a given project, but also how all the necessary steps could be combined to make this a painless operation.

As my development career has advanced I've developed a manner of "getting the job done." This would involve compiling to the standard class directory structure for initial esting/use with a final switch to .jar ckages for deployment. Most IDEs w this to happen painlessly. JBuilder es this to another level.

You can create both a class directory structure and a .jar package when you build, and that's not all. If your current project is also a Web app you can build this too – all at the click of a button. This applies across the board so you won't find yourself constantly updating the project properties to reflect what's required from any given task.

Somebody at Borland has thought about this. We, as a community of Java developers, must applaud and commend some of the tools Borland offers with this latest release since more time can be devoted getting on with the juicy bits. Several times I caught myself going ooooh, it does that? And ahhhh, that's cool.

All this has overhead so the constant building and deploying is not recommended for general use; however, it's easy to see that a lot of the deployment burden has been eliminated.

Note the specification of the two machines this IDE was tested under. Never being a shy one, this was adapted and several configurations tried (even to the extent of being slightly naughty and trying to cram it all into a 64MB system).

The results – it appeared processor speed was not anywhere near as important as masses of memory. Nearly all applications will run in direct relation to the amount of memory and JB5 is no exception. It performed admirably on both processors with 512MB installed. It says 128MB on the box, but if the intended use is to run alongside Web servers and other apps, then seriously consider the total RAM in the target machine. 256MB will provide comfortable levels.

The overall footprint isn't that much and doesn't appear to overstep similar products. JB5 sits at 2–4MB with the JVM starting at 40MB and rising to over 80MB with a medium-sized project. It also isn't a resource hog. There's nothing worse than an app that eats up 10% of your processor just by being in existence, something that has been avoided.

Help System

The help system is a bit haphazard. My personal preference is to use the Find option, gain a list of general items, and drill it down that way. This wasn't an easy task, however. Borland ran a link to almost every word in the help system – useless words with irrelevant meanings (aardvark, abbot) have links to equally useless information.

It's not impossible to use, but I found it a bit like a search result on the Internet, having to wade through the dross, keeping an eye open for that key phrase that gives you what you want.

The Manuals

Not content with simply informing us how to use their product, those clever chaps at Borland have included general programming/Java-related principles from bitwise operation to serialization. In addition, you'll also find introductions to Web applications and XML. Superb.

Steve Jobs' endorsement of Java (at last) at JavaOne 2000 has prompted a release on the Mac OS X platform. It's a delight to see this. Borland had one or two forays into the Mac software market a few years ago, but not since. Hopefully this will help bring total Mac-derived applications to the marketplace, which can only be a good thing.

JB5 has made it easier than ever to control and deploy large-scale solutions by providing many of the tools with which to do so. It has an intuitive front end with all the main features catered for and available with the minimum configuration. Borland has managed to get the mix between simplicity and functionality that others seem to find evasive.

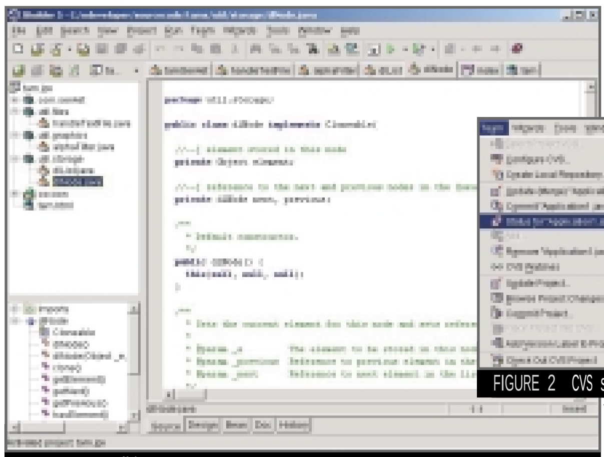


FIGURE 1 JBuilder 5

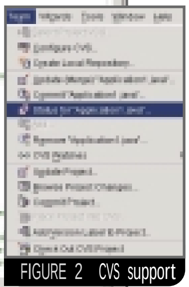


FIGURE 2 CVS support

Unlike the doctor that works for your HMO, Doctor Java is here to help you with all of your Java problems. The good Doc will not require a copayment for each visit nor ask you to fill out long arduous forms. The Doctor is here to help the readers of ***Java Developer's Journal*** find a cure for their Java system ills.

HOW CAN I SYNCHRONIZE METHOD ACCESS ACROSS TWO DIFFERENT CLASSES?

The synchronized modifier is used to control access to critical code in multithreaded programs. When a method is declared synchronized, you're instructing the program to obtain a lock (monitor) on the object that has the method. If you call another synchronized method on the same object, you don't lose the lock and other threads don't have access until the lock is freed.

This, however, does not work across objects. The best way to solve this problem is not to declare your methods as synchronized but to use a synchronized block. First, we'll look at typical code that does not lock across methods:

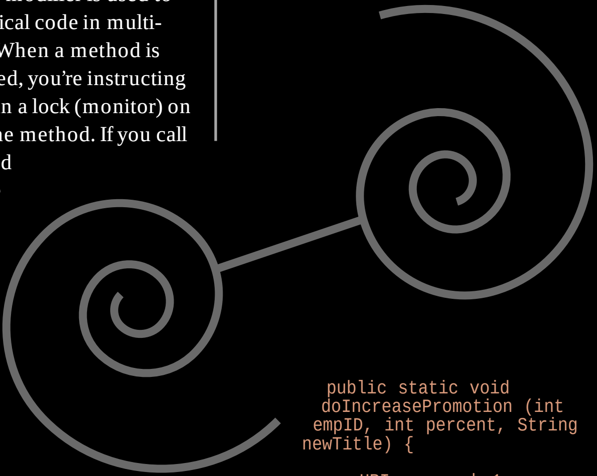
```
public synchronized void
increasePaycheck(int empId, int per-
cent) {
      ...
}

public synchronized void
givePromotion(int empId, String
newTitle) {
      ...
}
```

Let's consider doing something like this:

```
public void increasePaycheck(int per-
cent) {
      synchronized(this);
...
}
```

This would be considered a noble effort but unfortunately it does not do the trick. The main problem with this is that you are still trying to acquire a lock on this. What we really need is an alternative object to lock. Let's restructure our code and see how this can be realized.

```
public HRObject {
      private static Object lock = new
Object();
```



```
            public static void
           doIncreasePromotion (int
          empID, int percent, String
        newTitle) {

                HRIncrease hr1 = new
  HRIncrease();
          HRPromotion hr2 = new
HRPromotion();

            synchronized(lock) {
                  hr1.increasePaycheck(int
empId, int percent);
                  hr2.givePromotion(int
empId, String newTitle);
            }
      }
}
```

The preceding code example is the prescription for your woes. We basically used an "impartial" object as the basis for acquiring a lock. This approach requires that you always use the same lock object and methods similar in their approach to synchronization. We could have alternatively created a static method in a class that is synchronized and have this method make all the other method calls on your behalf.

I AM BUILDING A JSP/JDBC SITE WITH AN ISP THAT NEVER ALLOWS A DSN TYPE CONNECTION. THE DATABASE I WANT TO USE IS ACCESS. HOW DO I DO A DSN-LESS CONNECTION?

The doctor recommends seriously changing your diet and eliminating Microsoft Access as a database for use with Java and migrating to something that will scale a little better and is supportable. With that being said, let's dive into the solution to your problem. Let's say that you have an ODBC connection with the following configuration:

```
[ODBC]
DRIVER=Microsoft Access Driver (*.mdb)
UID=admin
UserCommitSync=Yes
Threads=4
SafeTransactions=0
PageTimeout=5
MaxScanRows=8
MaxBufferSize=2048
FIL=MS Access
DriverId=25
DefaultDir=c:\
DBQ=c:\doctorjava.mdb
```

To talk to an ODBC data source you will need to utilize a JDBC to ODBC bridge. Sun recommends using the bridge in situations where no native JDBC driver exists. Merant is a company that provides JDBC

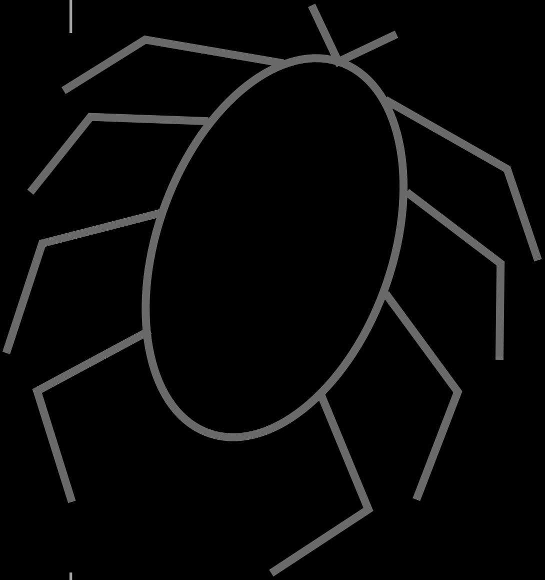
drivers that can connect to many disparate databases. To utilize the bridge, your code would look similar to:

```
try {

Class.forName("sun.jdbc.odbc.JdbcOdbcD
river");

    Connection conn =
DriverManager.getConnection("jdbc:odbc
:DRIVER=Microsoft Access Driver
(*.mdb);" +
        "DBQ=c:\\doctorjava.mdb");
    ...
}
```

The ODBC connection strings may vary in syntax depending upon the actual database and language you are connecting to.

I WOULD LIKE TO SEE ALL OF THE ELEMENTS IN THE REQUEST HEADER IN MY JSP FOR DEBUGGING PURPOSES. COULD YOU POINT ME IN THE RIGHT DIRECTION?

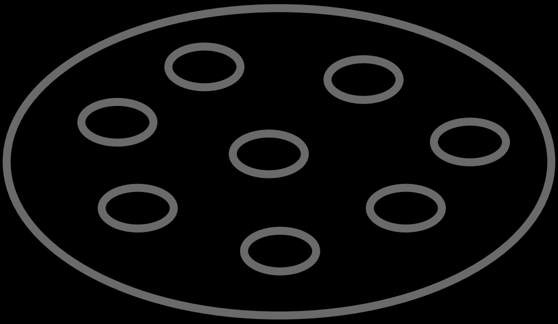
The doctor loves it when readers ask questions that can be answered quickly. To solve your problem we can take advantage of the java.util.Enumeration object. We can loop through the requests and display them to the screen. Let's take a look at some sample code.



```
<html>
<head><title>Show
        Headers</title></head>
<body>
<%
    java.util.Enumeration e =
request.getHeaderNames();
    for (e; e.hasMoreElements();) {
        String header =
(String)e.nextElement();
        out.print(header + " = " +
request.getHeader(header) + "<br>");
    }
%>
</body>
</html>
```

The doctor assumes the reason you have the need for such code is because you are having a problem related to types not being set properly on your Web server.

HOW DO I DELETE A COOKIE WITHIN A JSP PAGE?



Let's take a look at your code and see why it doesn't perform as expected.

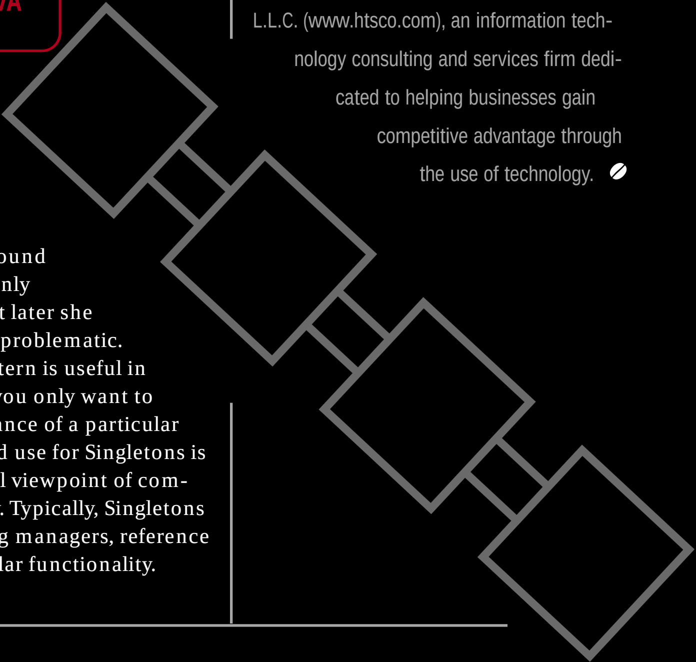
```
Cookie killCookie = new
Cookie("uid", null);
killCookie.setMaxAge(0);
```

The code removes a cookie from the browser once the browser is closed. This approach does not work in all instances. Many browsers keep it in memory and will not actually destroy the cookie until the browser instance is shut down. Let's take it one step further. In some scenarios, it may be useful to have the cookie destroyed without shutting down the browser. To realize this goal, you need to change your code slightly as follows:

```
Cookie killCookie = new Cookie("uid",
null);
killCookie.setMaxAge(0);
killCookie.setPath("/");
response.addCookie(killCookie);
```

This should do the trick nicely.

WHEN SHOULD I USE THE SINGLETON PATTERN IN MY JAVA APPLICATION?



The Singleton pattern reminds me of one of my girlfriends in high school. At first, I found that she was the only thing I wanted but later she proved to be very problematic. The Singleton pattern is useful in situations where you only want to have a single instance of a particular object. The second use for Singletons is to provide a global viewpoint of common functionality. Typically, Singletons will be used for log managers, reference services, and similar functionality.

Singletons can be problematic, like the girlfriend in high school They're typically implemented with synchronized blocks. Many threads will fight to obtain a lock on your Singleton causing a big performance bottleneck. The Singleton pattern can be problematic in a cluster as you will want more than one instance. The other problem with this pattern is that you need to make sure that its access is completely hidden from the outside world, or else it is essentially a big fat global variable.

The other thought I have is that it's always easier to make something a Singleton than to unmake something a Singleton. In my mind, this is a red flag that needs serious architectural thinking.

Letters to Doctor Java:

Send your questions, comments, and praise to doctorjava@sys-con.com. Published letters will be edited for length and clarity.

Doctor Java moonlights as an enterprise architect with Hartford Technology Services Company L.L.C. (www.htsco.com), an information technology consulting and services firm dedicated to helping businesses gain competitive advantage through the use of technology.

JASON BRIGGS J2ME EDITOR

Wishful Thinking

I've come to the conclusion that Japan is the place to live. Not that I really want to move to a country that by all reports is an extremely crowded and busy place, but the Japanese always seem to get the best gadgets. Zev Blut's article on iAppli development last month ("DoJa in NTT DoCoMo Phones," ***JDJ,*** Volume 6, issue 9) got me thinking, and I revisited a couple of Web sites that show the different mobile phones available there.

A good example is Mobile Media Japan where, if you take a look at the list of J-Sky phones (www.mobilemediajapan.com/hardware/jsky-handsets/), most have color screens and are exceptionally easy on the eyes (to use the vernacular). Style isn't limited to J-Sky phones; i-mode and EZweb also have some seriously worthy examples.

Western phones, by contrast, have predominantly gray-scale screens, and have been (until recently) generally clunky-looking when compared with their Asian cousins. Manufacturers in the West have a lot to learn from the Japanese – but then again, perhaps consumers in the West have a lot to learn as well.

Things are looking up, however: Nokia has released the 9210, a Java-enabled mobile phone/PDA, in Europe (with the 9290 scheduled for release in the U.S.). It may have a gray-scale screen on the outside, but it's full color within (you can find a review of the 9210 in this issue of ***JDJ).*** Still, it doesn't quite have the teenager-attracting good looks of some of those Japanese mobiles – but, of course, it isn't aimed at the same market.

In an attempt to help out mobile phone manufacturers in their goal to produce the "must have" mobile device du jour, let me present the specification for the phone I want in my pocket this Christmas.

This device will have the "chic factor" of one of those Matrix-style mobiles (Nokia's 7110 or Mitsubishi's J-D05 are examples of what I'm thinking of), a slot for an IBM 1GB Microdrive in the bottom, and a slot for one of those Sony memory sticks (shaped like a piece of chewing gum) in the top. In fact, why not throw another slot in the back and we'll put the operating system on a memory stick as well!

The screen should be a high-definition organic LED – and the CPU should be as fast as possible without reducing the operating time (personally, I'm hoping for a 1GHz processor that will run forever on a few milliwatts of power...but then again, I'm still waiting for those personal nuclear reactors that I saw in the movie *Ghostbusters* in the '80s to appear – so I don't hold out much hope for that).

It will, of course, be PersonalJava-enabled – or the Personal Profile, if Sun ever gets around to releasing it – with (Java) development access to 3D and MP3/audio libraries. We'll throw in various interfaces – like infrared and serial access – for good luck.

I want one. Now. Not in a couple of years' time. I've been infected by the bug of the Internet generation: I want immediate gratification. I have the attention span of a gnat and about as much patience as a turbocharged snail with nitroboosters and a penchant for drag racing.

Oh, and P.S., Mr. or Ms. Mobile Phone Manufacturer, I don't want to pay the Earth for my new toy.

Is that too much to ask for?

Probably...but I live in hope.

Anyway, wishful thinking aside, in this month's issue of ***JDJ*** – as well as the aforementioned Nokia review – you'll discover the wonders of robotic programming in Java, where David Hardin and Michael Frerking from aJile Systems describe how to develop a line-following robot using Lego Mindstorm and aJile components.

To round off last month's article on making Java for embedded computing a reality, Steven Schwarz and Vincent Perrier from Wind River write about determining if Java is right for your embedded device. They include a case study in the form of a hands-on porting exercise. Just the ticket if you want to get your hands really "dirty."

jasonbriggs@sys-con.com

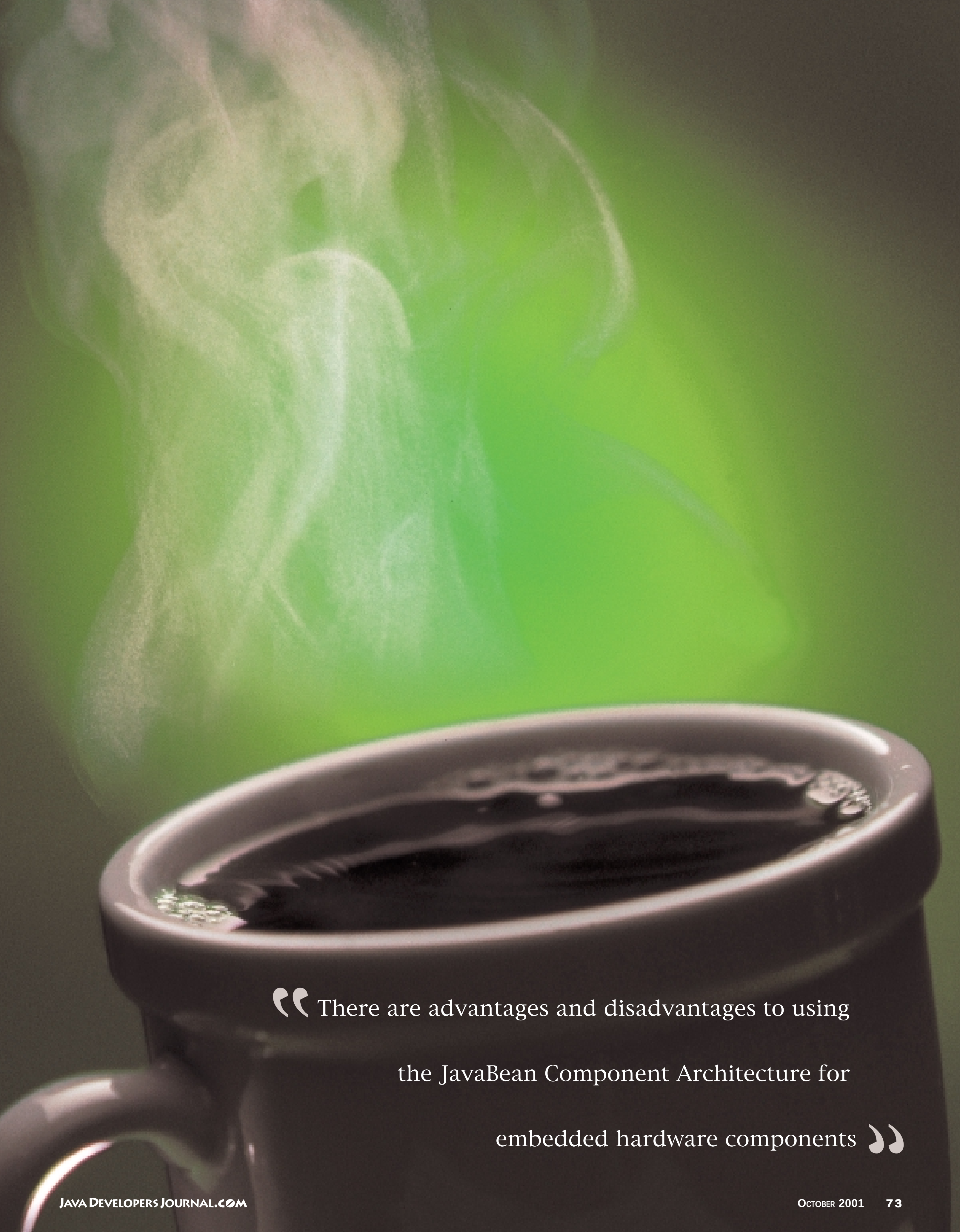
AUTHOR BIO
Jason Briggs works as a Java analyst programmer in London. He's been officially developing in Java for three years – unofficially for just over four.

A Storm in a Coffee Cup

Embedded real-time Java components for hardware interfacing

Written by David Hardin & Mike Frerking

The JavaBean Component Architecture provides a means to reuse software and, when combined with tool support, can dramatically increase developer productivity. This model has been realized primarily in graphical display applications with AWT and Java Swing components. Recent advances in J2ME and the release of the Real-Time Specification for Java (RTSJ) allow similar productivity gains to be achieved with real-time and embedded-hardware driver components.



“There are advantages and disadvantages to using the JavaBean Component Architecture for embedded hardware components”

In this article we examine the advantages and disadvantages of using the JavaBean Component Architecture for embedded hardware components. We describe a slight variation of the JavaBean model that's used in the aJile Java processor runtime libraries that make Java components more suited to time-critical and embedded applications. We follow with an example application that uses this component model to program a 100% Java-based controller based on the aJile aJ-80 Java microprocessor for Lego Mindstorms sensors. We conclude with suggestions on how to apply this component model to different applications.

The JavaBean Component Architecture

The JavaBean Component Architecture has been used successfully for graphical applications using AWT and Swing components and by third-party software vendors to distribute reusable software components using standard packaging. The JavaBean model specifies properties that can configure the characteristics of the JavaBean from inside a development environment at design time. It also specifies events that are sent from the JavaBean to the application when an event of interest occurs.

The JavaBean component specification also includes support for "bean info" files, and many development tools provide property editors to configure beans appropriately for the application under development. The JavaBean component model is well suited for graphical and nongraphical components for general-purpose computing systems, but, as we shall see, the JavaBean event model is less appropriate for time-critical or embedded systems.

Properties

JavaBean properties are get/set methods used to configure the behavior of the component at design time by a development tool, and at runtime by an application. Some extra overhead is associated with calling these get/set methods; however, it's minimal compared to the benefits gained. This method call overhead is also predictable, making JavaBean properties acceptable for embedded and real-time applications without modification.

Events

JavaBean events include an event listener interface that has one or more methods that will be called when an event is sent. An application implements the interface and registers it with the JavaBean, which calls the event methods at the appropriate time. Each of these event methods must also include a parameter of type EventObject. When a component sends an event, it creates a descendent of EventObject, initializes it, and passes it as a parameter to each registered event listener.

Normally, a JavaBean will create a new EventObject each time an event is sent since an event handler could keep a reference to the EventObject for the future. If the JavaBean reused the same EventObject, the internal state could be changed without the event listener expecting it. This event-transmission technique is less desirable for embedded or time-critical applications, because creating and discarding EventObjects is expensive. The new() operation for the EventObject is generally time-consuming and unpredictable as it could also initiate a garbage-collection cycle at an inappropriate time.

One alternative to creating a new EventObject each time an event is sent is to create one EventObject and reuse it. This would prevent the overhead associated with the new() operation and possible garbage collection at the expense of not strictly adhering to the JavaBean specification. This would also allow development tools to recognize the events and provide full tool support. This alternative still causes some overhead because the fields in the EventObject must be initialized with custom state information and the event handler has to access these fields to get the state information.

Cracked Events

A second alternative is to pass data directly to event handlers as primitive data types, bypassing the EventObject. This has the maximum speed advantage, at the expense of some development tools not recognizing these events as JavaBean events. This type of event is sometimes called a *cracked event* or *primitive event*. Development tool support for cracked events that still conform to the JavaBeans naming convention varies from no support to almost full support; Borland JBuilder, for instance, recognizes cracked events (see Figure 1). The Real-Time Specification for Java calls for a special real-time event called *AsyncEvent* that has no parameters. Perhaps, as the RTSJ gains in popularity, more tools support will be provided for this type of event.

Cracked events and their respective event listeners can be unique for each component and each type of event that's sent, or generic and reused for more than one component and for more than one event in the same component. Unique events have the advantage of being more appropriately named: each event is descriptive and specific to one type of event. Generic events, on the other hand, require fewer class definitions and are independent of any one component. This reduces the number of inner classes required to interface the components

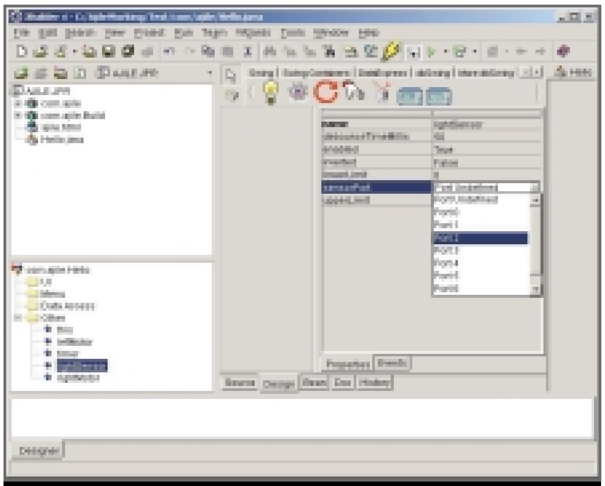


FIGURE 1 JBuilder session

and also reduces call overhead. Components utilizing generic events can be wired together many times in a pipeline fashion, with little or no "glue code" required.

The aJile Real-Time Embedded Component Model

The aJile Real-Time Embedded Component Model (aRTEC), a variation on the JavaBean Component Architecture, uses standard JavaBean properties but eschews EventObjects in favor of cracked events. These choices enable development tool support, while avoiding many of the negative runtime consequences of the JavaBean event model.

Three cracked events are used for the majority of the aRTEC components: a TriggerEvent with no parameters, an AssertEvent with one Boolean parameter, and a DataEvent with one int parameter. The trigger event is sent as an indicator that some physical condition has occurred. It could be a timeout or a sensor threshold being crossed. It can also be received to control physical conditions, such as resetting a timer.

An AssertEvent is sent when a state change occurs. A true value in the parameter indicates some state is active and a false value indicates the state is inactive. For example, a lamp component can receive an AssertEvent that will trigger the lamp to turn on when a true value is passed, and turn off when a false value is passed. A DataEvent is used when a value is associated with the event. For example, a rotation sensor can send a DataEvent when the angle of rotation changes; the new angle data is passed as the parameter. There are additional types of cracked events used in the aJile embedded component libraries, but these three have been found to be adequate for most of our real-time needs.

Many JavaBeans can send the same event to more than one registered listener, called *multicast events*. To implement multicast events, some JavaBeans keep a vector of the registered listeners. When an event occurs, the vector is traversed and the event is sent to each registered listener. Other JavaBeans implement a multicaster, which is an event listener that relays the event to two additional listeners. If one of the additional listeners is also a multicaster, the event is relayed to two more listeners. In this manner any number of listeners can be registered without the use of a vector. The use of the multicasters is hidden behind the JavaBean event registration methods.

The advantages of using multicasters in a real-time environment are clear. When only one listener is registered and an event is sent, the event is sent by a single interface call with no looping and no overhead associated with the vector. However, as more listeners are registered for the same event, events will be routed through multiple multicasters, causing more overhead. At some point, if enough listeners are registered, multicasters will have more overhead than traversing a vector; however, it's rare to have more than one or two listeners for the same event.

TriggerEvents, AssertEvents, and DataEvents all have their own multicaster classes that are used internally to implement multicast events.

Real-Time Components

Often a timer is required in a real-time or an embedded application. The aRTEC OneShotTimer component implements a timer that can be started, stopped, or restarted. When its time expires, it sends a TriggerEvent and waits to be started again. If the timer has been started but has not yet expired, it's considered to be running. If the timer is restarted while it's running, the timer starts over, effectively extending the time delay. When the OneShotTimer is started, it sends an AssertEvent with true as the value to indicate that it's running. When the time expires, it sends an AssertEvent with false as the parameter to indicate the timer is no longer running. At that time a TriggerEvent is also sent.

The OneShotTimer can receive one TriggerEvent to start the timer and one to stop it. A trigger event sent directly to this object will start the timer. Because this class can implement the TriggerEventListener interface only once, an inner class is used to provide a triggerEventListener() method that will stop the timer.

The aRTEC libraries also include a PeriodicTimer and a Counter component with similar interfaces. Generally, these components provide a friendly front end to real-time execution primitives provided by the Real-Time Specification for Java.

Lego Mindstorms Overview

To demonstrate the ability of the aRTEC components to monitor and control embedded hardware, components were written for the Lego Mindstorms robot sensors. This platform was chosen because it's well known and exhibits many of the same characteristics as other embedded and real-time applications. Lego Mindstorms hardware components include motors, touch sensors, light sensors, rotation sensors, temperature sensors, and lamps.

The Lego Mindstorms controller unit is called the RCX. For our example program we won't be using the RCX, but rather an advance prototype of the Systronix JCX controller that uses the aJile aJ-80 direct execution Java processor. The JCX provides extremely efficient and time-deterministic execution of

"The advantages of using multicasters in a real-time environment are clear"

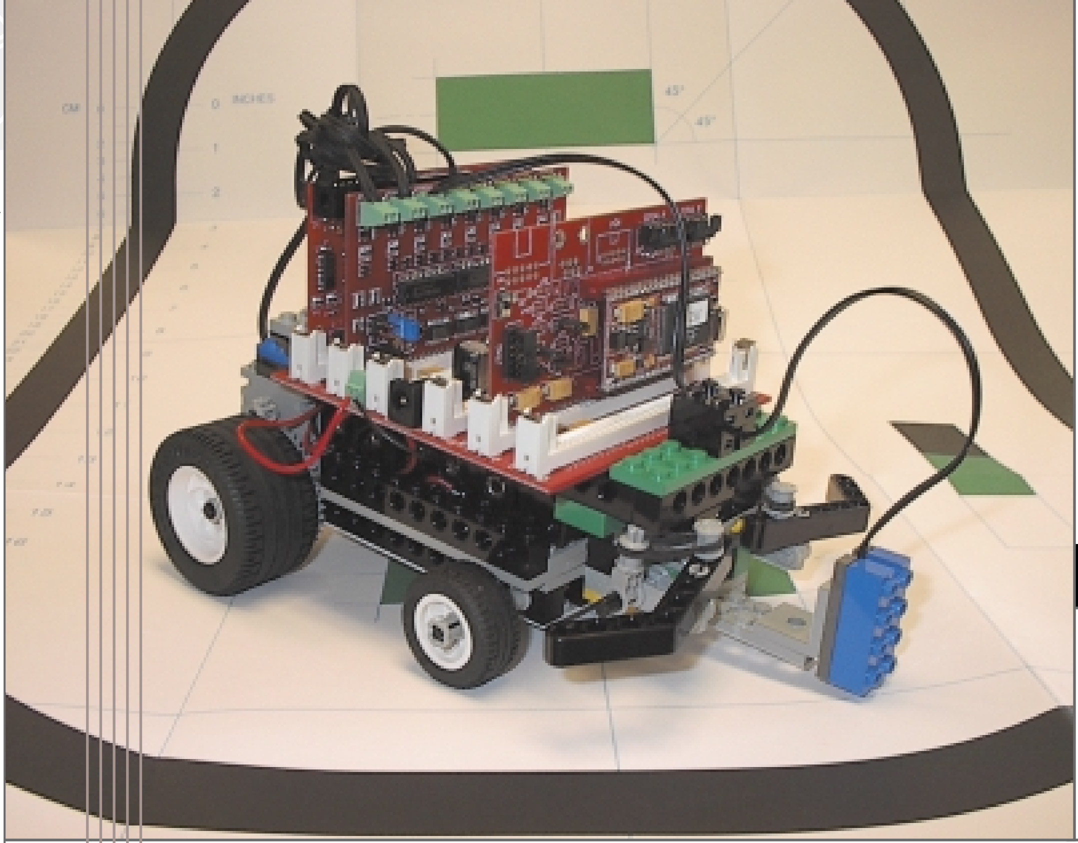
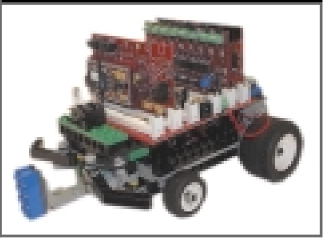


FIGURE 2
Autonomous Line Follower Mobile Robot



100% Java–control applications. The JCX is a direct replacement for the RCX unit and in its base configuration supports eight input sensors and four output ports compatible with the Lego RCX sensor and motor ports.

Lego Components

Each of the Lego Mindstorms sensors and motors has a corresponding Java component. To use the component the physical Lego hardware needs to be attached to a JCX sensor or motor port, and a corresponding software component needs to be created and initialized to the correct port number. The port number is a JavaBeans property.

Motor

The motor component has a forward, reverse, off, and brake state. This state can be set at any time through the State property. The motor component can also receive four unique TriggerEvents to set the motor state corresponding to forward, reverse, off, and brake. Because a TriggerEventListener is an interface, the motor component can directly receive only one TriggerEvent. This is because a Java object can implement an interface only once. This default trigger event will set the motor state to forward. Three inner classes that implement the TriggerEventListener interface are used to set the state to reverse, off, or brake. Applications can obtain a reference to one of these inner classes to register it as an event listener to another component by calling the getReverseListener(), getOffListener(), or getBrakeListener() methods.

The power level of the motor can be set from zero to full power. Two properties set the motor power level: MaximumPowerLevel, which determines the power range, and PowerLevel, which determines the duty cycle for the application of power to the motor when it's in the forward or reverse state. Because of the real-time responsiveness of the aJ-80 direct execution Java processor used in the JCX, the duty cycle for the motor can be set in the high kHz range.

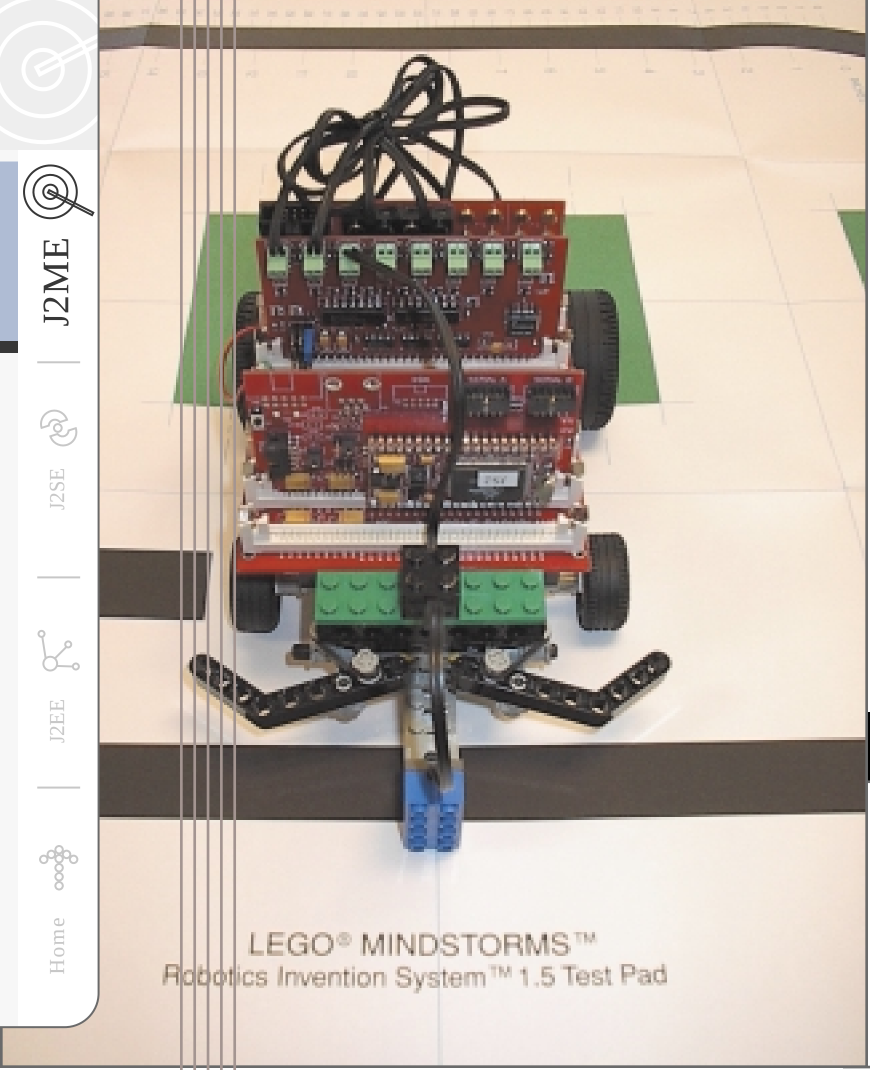
Touch Sensor

The touch sensor component monitors the state of one Lego touch sensor switch. The state of the switch can be checked by calling the isPushed() method. An AssertEvent is sent with true for the parameter when the button is pushed and false when the button is released. A TriggerEvent is also sent when the button is pushed and a second trigger event is sent when the button is released. This component also can receive an AssertEvent that will disable the touch sensor when false is passed and enable it when true is passed.

Light Sensor

The light sensor component monitors the light intensity from one Lego Mindstorms light sensor. The light intensity can be checked by calling the getLightIntensity() method, which returns a value between 0 and 4095, with 0 being the darkest. The light sensor also has a specific light range. The lower and upper limits of the range can be set using the UpperLimit and LowerLimit properties. When the light intensity transitions from inside the specified range to outside the

"Java and the JavaBean Component Architecture are an excellent platform for building reusable apps"



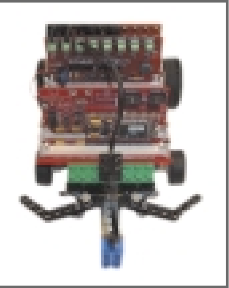
range, a TriggerEvent is sent, as is an AssertEvent with false for the parameter. When the light intensity transitions to inside the range, a second TriggerEvent is sent as well as an AssertEvent with true for the parameter. If a situation were to occur where more than one light intensity range were required, it's possible to create more than one light sensor component and assign them to the same sensor port, each component configured for a unique intensity range. The light sensor also receives an AssertEvent that will disable the sensor when false is passed and enable it when true is passed.

Example: Autonomous Mobile Robot Line Follower

To show the ease of use of the components, we implemented an Autonomus mobile robot line follower (see Figure 2). The LineFollower application (see Listing 1) uses a light sensor, two motors, and a one-shot timer to drive a wheeled robot down a black line drawn on white paper. Because most of the functionality is in the components, the line follower code is extremely small and easy to maintain. The LineFollower application creates the components, configures them, and defines any event handlers that require special processing. It's also possible to use a JavaBean-friendly developer tool such as Borland JBuilder to autogenerate the code that creates and initializes the components (see Figure 1).

The LineFollower application uses a light sensor to follow the line. When the robot is over a line, the motors will cause the robot to move forward. When the robot leaves a line, the motors will cause the robot to turn back to the line. This is done by remembering which direction the robot had to turn last time to find the line, and turn the other direction this time. This algorithm assumes that the robot crosses the line and exits on the opposite side. There's a possibility that the robot will leave the line on the same side that it entered it. This can happen if the line curves or if the robot finds the line before it completely turns toward it. In this case, the robot must again change directions to refind the line (see Figure 2). This condition is detected by monitoring the time since the robot left the line. If a time limit expires, the robot again changes directions, this time toward the line (see Figure 3).

FIGURE 3
Java-powered Line Follower Robot finding the line.



The LineFollower application is arranged into three distinct sections: creating the components, configuring their properties, and defining any special event handlers. This same pattern is used for many applications that use JavaBeans.

The left and right motor components control the left- and right-drive wheels of the robot, respectively. When both motors are in the forward state, the robot will move forward. When one motor is in the forward state and the other is in the reverse state, the robot will turn. Initially both motors are set to the forward state so the robot will move forward until it first finds the line.

A light sensor component is created and initialized to a lower threshold of 0 and an upper threshold of 1,000. Test cases showed a reading of about 800 when the light sensor was over a black line and about 1,200 when it was not. These limits will configure the sensor to be in range when it is over a black line and out of range when it is not.

When the light sensor detects a line, it sends a TriggerEvent. An inner class is registered to receive this event and in turn calls the enteringLine() method. When a line is entered, the robot should stop turning and move forward so both motors are forced into the forward state. When the light sensor detects white space, it will send another trigger event indicating that it is now out of range. A second inner class is registered to receive this event and in turn calls the leavingLine() method.

When the robot is no longer tracking a line, it needs to start turning in the opposite direction. This is done in the triggerEvent() method that checks the direction variable and sets the motors to turn the other way. The enteringLine() method also has to start the timer so that the direction will again be switched if the line is not found.

A OneShotTimer component is used to reverse the direction of the robot if a line is not found. It's initialized to a peri-

"A real-time embedded component model makes real-time embedded Java environments all the more compelling for developers" od of 500ms and will send a TriggerEvent when a timeout occurs. Instead of using an inner class to receive the trigger event, our main class implements the triggerEvent() method directly and registers itself as the listener for OneShotTimer timeouts. When a timeout occurs, the triggerEvent() method will be called and will reverse the robot's turn direction.

The design and implementation of the Mindstorms components required a few weeks of effort, but after that the LineFollower application required only about an hour to write, and most of that time was spent writing detailed comments. This is a very small example, but a large example that uses similar components could achieve similar productivity.

The LineFollower implementation on the aJ-80 processor is also very memory-efficient. Passing the code through the aJile ROMizing tool, JEMBuilder, produced a standalone executable that required less than 80KB of ROM, and only 35KB of RAM when using the standard J2ME Connected Limited Device Configuration (CLDC) runtime. Finally, because of the Java bytecode execution efficiency of the JCX platform, the robot line follower was amply powered by 6AA batteries (the same power supply used by the Lego RCX); much of the power is consumed by the motors.

Conclusion

The LineFollower example program described above demonstrates 100% Java-based components that are used to control a Lego Mindstorms robot. The components were built specifically for the Lego hardware, but the concepts are applicable to many other types of embedded and real-time applications. Both a serial port or a debounced button component would be appropriate for an embedded controller, a joystick component for a gaming box, and an error logger component for an industrial-control application. In all these cases, design and implementation time can be drastically reduced and maintainability enhanced by using a standard, embedded real-time component model.

Java and the JavaBean Component Architecture are an excellent platform for building reusable applications. However, until recently the Java platform was not practical for embedded and real-time computing. The release of J2ME, the Real-Time Specification for Java (RTSJ), and the availability of high-speed low-power Java hardware have overcome these barriers, allowing embedded developers to take advantage of the productivity and portability of the Java platform. The addition of a real-time embedded component model makes real-time embedded Java environments all the more compelling for developers.

Author Bios

David Hardin is chief technical officer and cofounder of aJile Systems, Inc. He is a coauthor of The Real-Time Specification for Java, and a member of several Java expert groups convened under the Java Community Process. Dr. Hardin holds a PhD in electrical and computer engineering from Kansas State University.

Mike Frerking is software lead at aJile Systems, Inc., where he develops tools and components for embedded and real-time Java development. Mike received his BSEE from Iowa State University.

david.hardin@ajile.com & mike.frerking@ajile.com

Listing 1

```
/**
 * Controls a JCX with a light sensor and two motors to follow a
line.
 * The line follower has two guiding rules.  First if it crosses
the line,
 * it will change the direction to find the line again.  Second,
 * if it doesn't cross the line after a specified amount of time,
 * it is probably in error and will again flip to find the line.
 */

package linefollower;

import com.ajile.jcx.*;
import com.ajile.components.OneShotTimer;
import com.ajile.events.*;

public class LineFollower implements TriggerEventListener {
  // Create left hand side motor
  Motor leftMotor = new Motor(Motor.MOTOR_PORT_0);
  // Create right hand side motor
  Motor rightMotor = new Motor(Motor.MOTOR_PORT_1);
  // Create the light sensor
  LightSensor lightSensor = new
LightSensor(LightSensor.SENSOR_PORT_2);
  // Create one shot timer
  OneShotTimer timer = new OneShotTimer();

  // Indicates last turn direction.
  public static final int LEFT = 0;
  public static final int RIGHT = 1;
  int direction = RIGHT;

  /**
   * Main entry point.  Creates one instance of this class
   */
  public static void main(String[] args) {
    LineFollower instance = new LineFollower();
  }

  /**
   * Initializes all JCX components and registers event handlers
   */
  public LineFollower() {

    // turn motors on
    leftMotor.setState(Motor.MOTOR_FORWARD);
    rightMotor.setState(Motor.MOTOR_FORWARD);

    // initialize light sensor so on the line is in range (0-2000)
    lightSensor.setLowerLimit(0);
    lightSensor.setUpperLimit(1000);

    // initialize the timer for a .5 second delay
    timer.setIntervalMillis(500);

    // register the flip triggerEvent method when a timeout occurs
    timer.addTimerListener(this);

    // use inner class to register the leavingLine method when the
    // light sensor moves out of range
    lightSensor.addOutOfRangeListener( new TriggerEventListener()
{
      public void triggerEvent() {
        leavingLine();
      }
    });

    // use inner class to register the enteringLine method when
the
    // light sensor moves in range
    lightSensor.addInRangeListener( new TriggerEventListener() {
      public void triggerEvent() {
        enteringLine();
      }
    });

  }

  /**
   * Called by the light sensor when light intensity
   * increases past the upper threshold indicating that we
   * are moving from dark to light. We lost a line.
   */
  void leavingLine() {
    // the line has been lost, so reverse turn
    triggerEvent();
    // reset the timer in case line is not refound
    timer.start();
  }

  /**
   * Called by the light sensor when light intensity
   * decreases past the upper threshold indicating that we
   * are moving from light to dark.  We found a line
   */
  void enteringLine() {
    // stop the timer
    timer.stop();
    // Line found, move forward
    leftMotor.setState(Motor.MOTOR_FORWARD);
    rightMotor.setState(Motor.MOTOR_FORWARD);
  }

  /**
   * Causes the JCX to turn in the opposite direction.
   * If the JCX is presently going straght, this routine
   * will still cause it to turn in the opposite direction
   * as last time this routine was called.
   */
  synchronized public void triggerEvent() {
    if (direction == LEFT) {
      direction = RIGHT;
      leftMotor.setState(Motor.MOTOR_REVERSE);
      rightMotor.setState(Motor.MOTOR_FORWARD);
    } else {
      direction = LEFT;
      leftMotor.setState(Motor.MOTOR_FORWARD);
      rightMotor.setState(Motor.MOTOR_REVERSE);
    }
  }
}
```

Download the Code!
www.JavaDevelopersJournal.com

Nokia 9210 Communicator

by Nokia

REVIEWED BY JASON BRIGGS

jasonbriggs@sys-con.com

info

Nokia
Keilalahdentie 4
FIN-02150 ESPOO
Finland
Phone: +358 7180 08000
Web: www.nokia.com

Specifications

Processor:	52MHz 32-bit ARM9-based RISC CPU
Memory:	14MB application, 2MB User (+ 16MB application card), 8MB Execution memory
Display:	640x200 pixels, 4096 colors
Size:	158x56x27 mm
Weight:	244g
Power:	High power Li-Ion battery
Talk time:	4–10 hours
Standby time:	Up to 230 hours
Input:	Keyboard
Ports:	Infrared, DLR-2L Cable, MultiMediaCard memory slot
Operating system:	Symbian platform
Java support:	PersonalJava preinstalled; download beta of MIDP
Availability:	First half 2002 as 9290
Price:	Estimated $700–$1000

Convergence. A word loved by PR companies and feared by nontechnical consumers. If you believe industry pundits, we'll all be carrying combination mobile phone-PDA-TV-toasters in the next few years. You'll be able to make a phone call, write a memo, watch the morning news, and cook your breakfast all at the same time while on your way to work. How close reality comes to the dream (or nightmare, depending upon your point of view) is anyone's guess, but the first devices that might fit a definition of "convergence" are emerging on the market. Nokia's 9210 Communicator is one such combination – a mobile phone and PDA.

The Communicator

The first thing you'll notice about the 9210 when you get it out of the box is its size. It's a fairly large phone, not quite retro/'80s size but bigger than the average. For example, a common (cheap) phone over here in Europe is the Nokia 5110, which measures 132x47.5x31mm and weighs about 170g. By comparison the 9210 measures 158x56x27mm and weighs in at 244g. Of course, we're not just talking about a phone here since it's also a PDA. So if you're currently lugging around a PDA and a phone – a Palm IIIc (for example) weighs around 190g – add the weight of the two together and you may already be carrying significantly more than Nokia's Communicator.

The 9210 runs Symbian's EPOC operating system, which will be familiar to loyal Psion users (alas, poor Psion, I knew him well...), and comes with a number of useful applications, such as a WWW and WAP browser, e-mail, word processor, spreadsheet, and presentation viewer. The screen is not touch sensitive – all control is done by a very small keyboard – but there are "quick" buttons on the right-hand side of the screen, for easy access to essential functions. For example, if an icon is highlighted, the word "Open" may appear beside the first button.

The mail system is one of the cooler applications on the 9210 as it integrates SMS as well as e-mail into the one program, rather than different apps for different mail types. An SMS message that's longer than 160 characters will be automatically sent as multiple SMS messages. *Mental note:* How to annoy your friends – send a 50-page text as an SMS message...

Less useful is the video player. Personally I'm not sold on the whole "watching videos on your mobile" thing. Don't get me wrong. I think it will be brilliant once organic displays become the norm and resolutions are so sharp you could slice through the door of a car quicker than you could say "Ginsu." But on a 640x200 screen with over 4,000 colors, we're not exactly talking cinematic quality.

Java Support

The important issue is the 9210's extremely strong Java support. The 9210 comes with a PersonalJava VM built in, and there's also a beta of a MIDP VM available for download. Java applications can be downloaded to the phone as straight class files and executed if they're in the default package (which is a hack and shouldn't be used as a "production" deployment mechanism). The alternative and recommended method is that they be packaged as Symbian install files (.SIS) and installed as with any other application onto the device. You can run applets, but unfortunately not through the browser.

I can already think of a couple of real-world situations where the Communicator,

using Java applications, might prove rather handy. For a company with a large, roving sales force, outfitting their entire staff with mobile phone, laptop, and mobile Internet connections would prove costly (to put it politely). A laptop solution seems overkill when you consider exactly how much use the staff of this fictional company is likely to get out of a full-blown PC. However, the 9210 is already connected and contains most of the applications the salespeople might use, and company-specific apps (a product catalog, for example) can be written in Java and quite easily installed on the device.

How well does PJava work on the 9210? It should be noted in advance that the processor, while more powerful than most mobile phones, is still only 52MHz. So performance on certain graphics operations won't be console quality, to put it mildly. However, when you consider the intended market and the above example, this doesn't really seem like that much of a negative point.

Test Results

AWT Component Test

Here we place a number of AWT components (buttons, text fields, etc.) on the screen using a couple of Layout Managers. While the 9210 displays all the components as expected, one slight problem is that with the default (white) background color of the panels, it can be quite difficult to see which component has the focus. This may be an issue when porting PersonalJava apps without modification to the phone. Of course, it's not a problem if your user-interface color scheme is configurable.

Triangle Drawing Test

Here we draw four randomly sized, filled triangles using multiple calls to the drawLine graphics method. The PersonalJava VM on the 9210 takes 200–600ms to draw the four triangles onto the 640x200 screen. Note that these are random triangles, so the slower time is indicative of a larger triangle to render.

Pixel Blit Test

Here we blit random pixels to the screen as a single image for 500 frames. On the 9210 the test runs at about 1.1 frames per second (fps) on a 320x200 window (which is half the 9210's screen size).

Image Test

This test draws 20 50x50 pixel images per frame for 500 frames, and then works out the frame rate from the time taken. This test runs at 4.4 fps.

Conclusion

If you've seen Nokia's TV commercial for the 9210, you'll probably agree that the company appears to be targeting businesspeople, the kind of people who are constantly on the go, perhaps have a need for a PDA, but are less likely to be interested in lugging a laptop around.

In my opinion, if you're intending to buy a PDA and are already carrying a mobile, you should look at the 9210 as a convenient alternative. Corporations that are tooling up their management/executive/mobile staff with PDAs and/or mobile phones should definitely consider the Communicator option.

It's a fairly large phone, as it has to support a keyboard that could be used by people bigger than a 5-year-old child; so if you're looking for a small device, the 9210 may not be right for you. However, it weighs less than an average mobile phone and PDA together, so I believe it rates highly on convenience factor alone.

With a free SDK available, and built-in and downloadable VMs, Nokia's support for Java on the 9210 is very good. Graphics performance is not mind-blowing because of the CPU speed, but it doesn't need to be when you take into account the people who are likely to be using the device.

It's a solid little unit, and the only negative point I noticed during testing was Symbian's typically lackluster PC connectivity (at least on Windows ME). Not that it didn't work; rather, it occasionally decided not to connect without a reboot. Having owned a Psion in the past, my opinion is that this is somewhat par for the course, and it happened infrequently enough that it wasn't a huge bother.

After seeing the 9210, I'm looking forward to the new Java-enabled products Nokia has up its corporate sleeve.

Java for Embedded Computing

A REALITY

Part 2 of 2

Written by Vincent Perrier & Steven Schwarz

It's more productive as a language, and as a programming environment, than C

The Java development and runtime environment, with its "write once, run anywhere" paradigm, brings enormous advantages to the embedded industry. Java code is highly reliable, easily ported, and includes features such as Internet readiness, security, and the ability to download code at runtime to upgrade or extend applications.

As such, it's ideal for Internet appliances (such as set-top boxes, Internet screen phones, or handheld devices) as well as traditional embedded devices (such as printers, medical devices, measurement and control instruments, telecommunication and datacommunication equipment now connecting to the Internet).

In Part 1 of this article (*JDJ*, Vol. 6, issue 9), we discussed some of the challenges of using Java for developing embedded systems and some of the solutions that are available to developers. In Part 2, we describe the benefits Java brings to the world of embedded software development. We also illustrate, based on a practical case study, how using Java for writing and running embedded software can benefit application developers.

Is Java Right for Your Embedded Device?

By asking the questions at the right, you should be able to determine if Java is the right development environment for your embedded device.

1. Will you want to download code to the device?
2. Will the device run in a network with a similar class of devices from other manufacturers or any other kind of devices?
3. Will there be multiple third-party content providers for the device?
4. Will user's experience (look and feel, stability, security) matter much?
5. Are you working within a short-development time frame?

If you answered yes to any of these questions, Java may be the right choice for you. Here's why.

Java Benefits

The intrinsic advantages of Java are portability, reusability, and development life-cycle enhancements. They're invaluable for any software engineering project, including embedded software development. The benefits include:

- ***Dynamic extensibility:*** A Java program can download code at runtime by fetching new class files and either replacing the same classes already loaded into memory or adding them to the application. Dynamic extensibility enables developers to extend the functionality of an application, upgrade classes

UNIX TALK	JAVA JTALK
Over 2,500 lines of commented code with comments in ad hoc format.	Under 1,600 lines of javadoc'ed code with comments in standard format.
"Typical" C program structure, with many global variables shared and modified all over the place.	Clean structure, using an interface to abstract the interaction of the jtalk session implementation and the client of the session.
A truly confusing use of alarm signals, and even setjmp/longjmp calls to create complicated artificial flows of control and get around the lack of threads, making the code difficult to understand and completely not portable.	Natural use of ordinary control structures and Java threads.
Estimated time to understand (not to write, since we didn't): two and a half days.	Estimated time to write: half day.

TABLE 1 Comparison of C code and Java code

with new versions of code, or simply fix software problems. It's also the base for implementing an application-management framework and a service-delivery platform.

- ***Security:*** Strict code verification prior to execution ensures that code doesn't try to disregard the protections imposed by the language, build direct memory access pointers, or use the wrong object. Java applets also obey limitation rules (known as the *sandbox* concept) that Web browsers enforce to forbid access to system resources, such as files and network hosts. It's impossible in theory to introduce malicious code into a Java application. Java threads can't corrupt the execution space of another thread.
- ***Portability:*** Java provides standard programming interfaces to OS-level services such as multithreading, networking, and graphics. This enables embedded system developers to write platform-independant code more quickly and easliy.
- ***Reliability:*** Java provides the reliability that embedded systems require, such as the obligation of source structuring, strong compile and runtime verifications, and an efficient exception-handling mechanism. Java's memory-management mechanism removes sources of many problems associated with dynamic memory allocation, such as memory leaks or corrupted pointers. Java also relieves the embedded programmer of the hazardous object de-allocation task.
- ***Reusable code:*** Java provides ease of code development, code reuse, and better code quality due to its true object orientation, simplicity, and development environment. A wide variety of affordable and efficient Java development tools are available as are many trained engineers.

Hands-on Porting Exercise

To illustrate how using Java for writing and running embedded software can benefit application developers, we selected a single healthy-sized example of a Java application – suitable and also useful for embedded applications. The example includes the following characteristics: it uses a simple graphical user interface, it's network-oriented, it's based on an existing version of C for easy comparison, and it's simple enough for a quick port, but large enough to be a significant case study.

Several aspects of this hands-on porting exercise are described below, basing comments on code snippets. The complete application source code is available for download on the ***JDJ*** Web site at www.sys-con.com/java/sourcec.cfm and can be freely used, modified, and distributed. The download package consists of application source code, documentation in javadoc format, Makefile, and precompiled class files and a .jar file.

The application we chose is called jtalk, a port of the Unix talk command. Here's an excerpt from the manual pages on the talk command on a FreeBSD system (obtained by typing man talk on the FreeBSD 4.3 system):

```
 Talk is a visual communication program that copies
lines from your terminal to that of another user.
 When first called, talk sends the message:

      Message from TalkDaemon@his_machine...
      talk: connection requested by
      your_name@your_machine.
      talk: respond with: talk your_name@your_machine
      to the user you wish to talk to.

 At this point, the recipient of the message should
reply by typing:

              talk  your_name@your_machine
```

The talk utility is a two-way, screen-oriented communication program. Once communication is established, the two parties can type simultaneously, with their output displayed in separate regions of the screen.

The talk application sets up a peer-to-peer communication session with a remote user on another host. A talkd daemon has to be running on both the caller's and the callee's machines and act as a repository for active talk invitations from one party to another. The caller has to be able to leave an announcement on the callee's machine that an invitation exists and has to leave the invitation itself on the caller's own machine. Once the session is connected, the talk daemon isn't used; it's just peer-to-peer between the two talk programs.

Since talkd daemons exist for Unix machines only, a port of talkd in Java was also created as part of this case study so talk can be used on Windows machines as well as in a non-Unix embedded Java application environment, such as Wind River's VxWorks and Personal JWorks.

Note: There are two incompatible talk protocols of different ages. The older one uses UDP port 517 and is in use on Solaris. The newer one (which we chose to implement) uses UDP port 518 and is in use on BSD systems, such as FreeBSD or Wind River's BSD/OS.

Technically, the port 517 protocol is called *talk* and the port 518 protocol is called *ntalk*. The port number 518 is currently hard-coded into jtalk and jtalkd. But it wouldn't help to change them to 517 because the older talk protocol on port 517 is incompatible.

To run the jtalk program on your JVM, download the case study package, put jtalk.jar somewhere on your file system, add the location of jtalk.jar to your classpath, and invoke as follows:

```
java com.windriver.jtalk.jtalk {person} [{terminal}]
```

If the machine on which you're running jtalk doesn't run a talk daemon already, you'll have to first launch jtalkd with the following command:

```
java com.windriver.jtalk.jtalkd
```

The syntax for {person} can be any of the following:

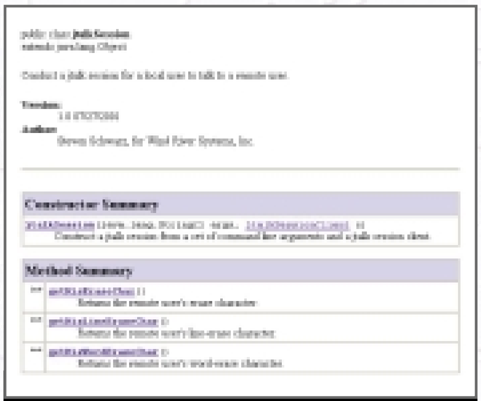
```
username
username@hostname
hostname!username
hostname:username
```

{terminal} is optional. It's required when responding to an invitation from someone with more than one login to be sure you get to the right one.

If you can't run multiple JVMs on your system (which is the case on most embedded RTOSes), the download package includes a basic "shell" with a simple self-explanatory GUI to help you launch the jtalkd and jtalk applications:

```
java
com.windriver.jtalk.jtalkShell
```

If you want to recompile the source code, you can use the Makefile for GNU make included in the download package.



public class **jtalkSession**
extends java.lang.Object

Construct a jtalk session for a local user to talk to a remote user.

Version:
1.0 [illegible]
Author:
Steven Schwarz, for Wind River Systems, Inc.

Constructor Summary
jtalkSession(java.lang.String[] args, jtalkSessionClient c)   Construct a jtalk session from a set of command line arguments and a jtalk session client.

Method Summary	
[illegible]	getRemoteEraseChar() Returns the remote user's erase character.
[illegible]	getRemoteLineEraseChar() Returns the remote user's line-erase character.
[illegible]	getRemoteWordEraseChar() Returns the remote user's word-erase character.

FIGURE 1 HTML documentation page for jtalkSession class

The original C files from which jtalk was created are also included in the download package for your convenience. These files come from the FreeBSD 4.3 distribution.

Choosing to port an existing Unix application, coded as a C program, was deliberate so a head-to-head comparison could be easily done between the C code and the Java code. Table 1 summarizes the major characteristics of the two versions.

The experience with jtalk showed that Java can be used more productively as a language, and as a programming environment, than C/C++ for certain types of applications, especially in the embedded area.

Some of the productivity advantages of Java are at the language level. For example:

- The language's insistence on strong type checking promotes the careful arrangement of code into well-designed classes, saving effort when supporting the code later.
- The *jtalk program* is cleanly organized in a directory structure com/windriver/jtalk following the package organization, leading to clean, easily understandable, and maintainable code. Each Java source file starts with:

```
package com.windriver.jtalk;
```

- The *C code for the talk program* has a strange organization, with global variables shared by many modules. Most of the time it took to port the code was actually figuring out how that code worked. In addition, the use of global variables makes the application non-reentrant, which can be a major issue for many embedded systems. As a consequence, it's impossible to have multiple talk sessions on an embedded system.
- The *Java language's class inheritance* (where a subclass extends a superclass, adding new features or overriding other features) and interface concepts (making implementations pluggable) both lead to more flexible reuse of existing Java code. In C, you usually have to modify to reuse code.
- The file *jtalkSessionClient.java* (see Listing 1) defines an interface for a client session implemented among other interfaces in the AWT area, and makes the creation of GUIs much easier by the main class jtalk in jtalk.java (see Listing 2).
- *Exception handling* is another fantastic built-in language feature that Java programs can efficiently take advantage of to manage abnormal situations or a rupture in the usual flow of execution. For example, jtalk defines a new RuntimeException called jtalkSessionException (see Listing 3) that is thrown by jtalk when a problem occurs in jtalkSession.java (see Listing 4) when trying to get the hostname information.
- Another example of productivity gains from language-level features of Java is *automatic garbage collection*, freeing developers from tedious management of dynamic memory allocations, often prone to a generation of nasty bugs.

Productivity advantages of Java from the overall programming environment are, for example:

- The existence of the *javadoc tool* means that you can generate helpful documentation in HTML format for Java code right from the program source, leading to better documented source code and more available documentation during all engineering phases.
- Figure 1 shows a snapshot of the HTML documentation page for the jtalkSession class,
- The existence of *standard API libraries* for almost every imaginable application programming need leads to quick coding of applications. This also leads to smaller application code size, as seen in this case study.
- If you're an application developer, you can *write your application faster* using Java and the preexisting standard APIs. If you're a system developer, however, you may find Java too abstract or high level, lacking features like direct access to memory or interrupt handling.
- The most important preexisting API sets relevant to embedded applications running in PersonalJava types of platforms, such as Wind River's Personal JWorks, are *networking and user interface APIs.* Thus, if your application performs network communication and has a user interface, you should at least consider using Java to more productively code those portions of your application.

To understand Java's suitability to write networking code, look at the original C source for the talk program. Note all the occurrences of signal() calls to set or clear a handler for the SIGALRM used to manage connection timeouts. Instead of setting a timeout on a socket the way it's done in the Java program, the C version sets a timeout for the process; if the timeout occurs, something is done. In one case, in invite.c, there's even a setjmp/longjmp pair, with the longjmp coming from the tail end of the SIGALRM handler used there. The effect is to create a crazy kind of "loop"(see "flowchart" in Listing 5).

This loops every 30 seconds, remaking the invitation repeatedly until at some point accept() manages to complete before the SIGALRM is delivered.

In the Java sources, we simply set a 30-second timeout on the ServerSocket. The equivalent accept() call then returns if there's no incoming connection before the 30 seconds elapse. We just have a simple loop in the Java sources (see Listing 6).

A Language and Platform

In this case study, we've demonstrated how Java can benefit embedded software application developers, especially those focusing on graphics and multithreaded networking. Java can benefit these users when used both as as a programming language and a development platform.

Java's standard, complete, and easy-to-use APIs allow developers to write portable, more compact, more efficient, simpler, and more easily maintainable code than in C, for which network, threading, and graphics APIs are usually proprietary. In addition, developers can also benefit from the language's advanced concepts such as object orientation, packages, exceptions, and strong typing, as well as from the advantages offered by Java's development platform, including documentation generation, cross-platform portability, and security features (which enable them to write more efficiently for better code).

Although it wasn't covered in this article, Java is ideal as an execution platform. Java's capability for dynamic and secure code updates provides the essential building block for creating an application and content-management framework that enables a service-delivery platform. Java's standardized API and portable code allows application developers to write platform-independent software content or services independently from embedded-system manufacturers. As long as applications are written to a Java specification that's compatible with the system's Java application environment, end users of such systems will be able to download, install, and run those applications on their systems.

Author Bios

Vincent Perrier is Wind River's product manager for Java platforms. He has a computer science and engineering degree from the University of Nantes in France.

Steven Schwarz is a senior member of the technical staff at Wind River. Recently he has been making processor-independent ports of CLDC and MIDP-reference implementations to VxWorks.

vincent.perrier@windriver.com & steven.schwarz@windriver.com

Listing 1: jtalkSessionClient.java

```
public interface jtalkSessionClient
{

    /**
     * This method is called by the session to deliver a status
       message
     * to the client.
     */
    public void        jtalkSessionStatus(String s);

    /**
     * This method is called by the session to get a character
       typed
     * by the local user of the session; each such character is
     * transmitted by the session to the remote user of the ses
       sion.
     */
    public int         jtalkSessionITyped();

    /**
     * This method is called by the session to inform the client
       of a
     * character typed by the remote user of the session.
     */
    public void        jtalkSessionHeTyped(int b);

    /**
     * This method is called by the session when it is about to
       shut
     * down.
     */
    public void        jtalkSessionEnding();
}
```

Listing 2: jtalk.java

```
public class                   jtalk
    extends                    Frame
    implements     ActionListener,
   KeyListener,
   WindowListener,
   jtalkSessionClient
```

Listing 3: jtalkSessionException.java

```
public class       jtalkSessionException
     extends       RuntimeException
{
     /**
      * Construct a jtalk session exception without a detail mes
        sage.
      */
     public jtalkSessionException() {
 super();
     }

     /**
      * Construct a jtalk session exception with a detail message.
      */
     public jtalkSessionException(String s) {
 super(s);
     }
}
```

Listing 4: jtalkSession.java

```
try {
     myInetAddr = java.net.InetAddress.getLocalHost();
     if (args[arg].equals("-host")) {
          myInetAddr = java.net.InetAddress.getByName(args[arg +
1]);
          arg += 2;
          nargs -= 2;
     }
     myHostName = myInetAddr.getHostName();
}
catch (Throwable t) {
     t.printStackTrace();
     throw new jtalkSessionException("can t determine local host
name");
}
client.jtalkSessionStatus("talk@" + myHostName + " starting.");
```

Listing 5: In the process context

```
1. <announce invitation to callee's talkd daemon>
2. <leave actual invitation on local talkd daemon>
3. <use setjmp() to establish this as a jump-to point>
4. <set SIGALRM timer for 30 seconds>
5. <call accept() to accept an incoming connection>
6. <cancel SIGALRM timer>

Code snippet from invite.c:
void
invite_remote()
{

 setitimer(ITIMER_REAL, &itimer, (struct itimerval *)0);
 message("Waiting for your party to respond");
 signal(SIGALRM, re_invite);
 (void) setjmp(invitebuf);
 while ((new_sockt = accept(sockt, 0, 0)) < 0) {
  if (errno == EINTR)
   continue;
  p_error("Unable to connect with your party");
 }

In the SIGALRM handler context:
1. <announce the invitation again>
2. <leave the invitation again>
3. <use longjmp to jump to the point 3 in the process context>

Code snippet from invite.c:
void
re_invite(signo)
 int signo;
{
 message("Ringing your party again");
 waddch(my_win.x_win, '\n');
 if (current_line < my_win.x_nlines - 1)
  current_line++;
 /* force a re-announce */
 msg.id_num = htonl(remote_id + 1);
 announce_invite();
 longjmp(invitebuf, 1);
}
```

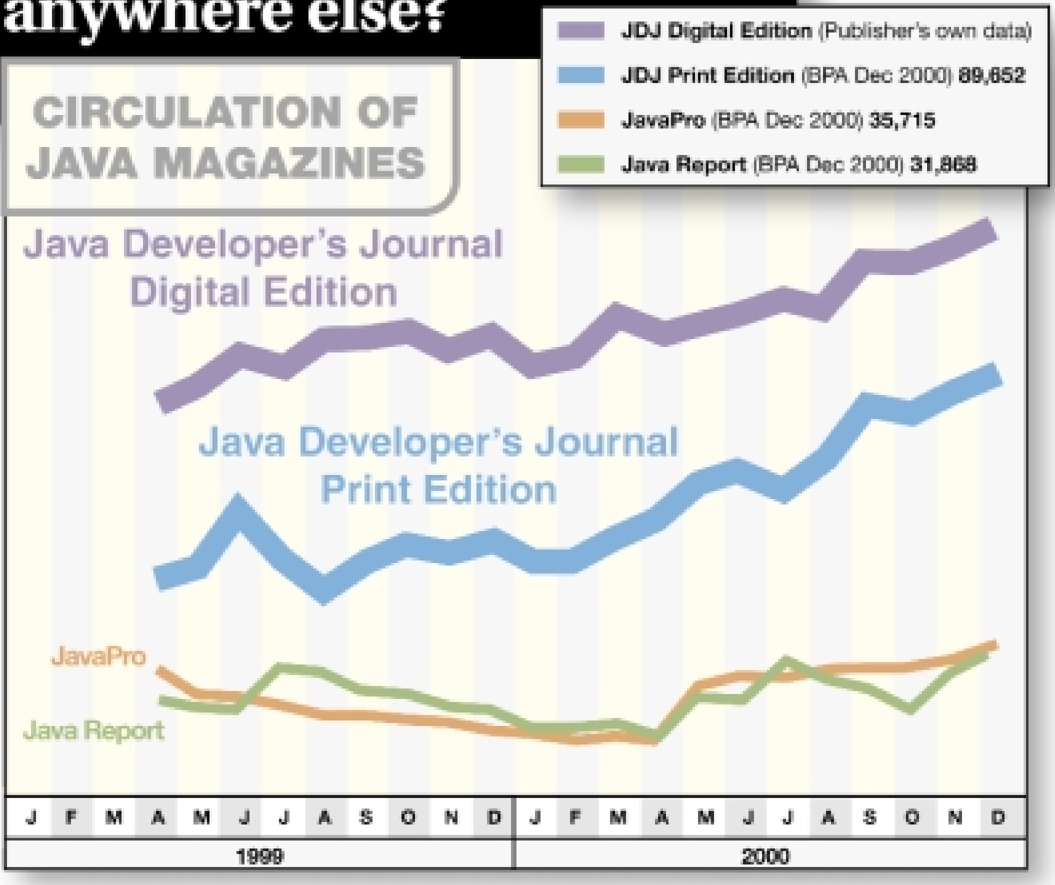
Listing 6: jatlkSession.java

```
void issueInvitation()
    {

 for (;;) {
     if (current_idnum >= 0) {
  client.jtalkSessionStatus(RINGING);
  current_idnum++;
     }

     client.jtalkSessionStatus(ANNOUNCING);

     msg.setMessage((byte) talkdMessage.ANNOUNCE,
      current_idnum,
      myUserName,
      hisUserName,
      hisTerminal,
      server,
      control,
      myInetAddr);
     controlTransaction(hisInetAddr);
     remote_idnum = rs.getIdNum();

     if (rs.getAnswer() != talkdResponse.SUCCESS) {
  throw new jtalkSessionException("invitation rejected: " +
         rs.decodeAnswer());
     }

     client.jtalkSessionStatus(INVITING);

     msg.setMessage((byte) talkdMessage.LEAVE_INVITE,
      current_idnum,
      myUserName,
      hisUserName,
      hisTerminal,
      server,
      control,
      myInetAddr);
     controlTransaction(myInetAddr);
     local_idnum = rs.getIdNum();

     client.jtalkSessionStatus(WAITING2);

     try {
  data = server.accept();
  data.setSoTimeout(1 * 1000);
  dataInput = data.getInputStream();
  dataOutput = data.getOutputStream();
     }
     catch (InterruptedIOException x) {
  current_idnum = remote_idnum;
  continue;
     }
     catch (Throwable t) {
  t.printStackTrace();
  new jtalkSessionException("can't get connection");
     }
     finally {
  msg.setMessage((byte) talkdMessage.DELETE,
          local_idnum,
          myUserName,
          hisUserName,
          hisTerminal,
          null,
          control,
          myInetAddr);
  controlTransaction(myInetAddr);

  msg.setMessage((byte) talkdMessage.DELETE,
          remote_idnum,
          myUserName,
          hisUserName,
          hisTerminal,
          null,
          control,
          myInetAddr);
  controlTransaction(hisInetAddr);
     }
     break;
 }
    }
```

Download the Code!
www.JavaDevelopersJournal.com

It's a no-brainer

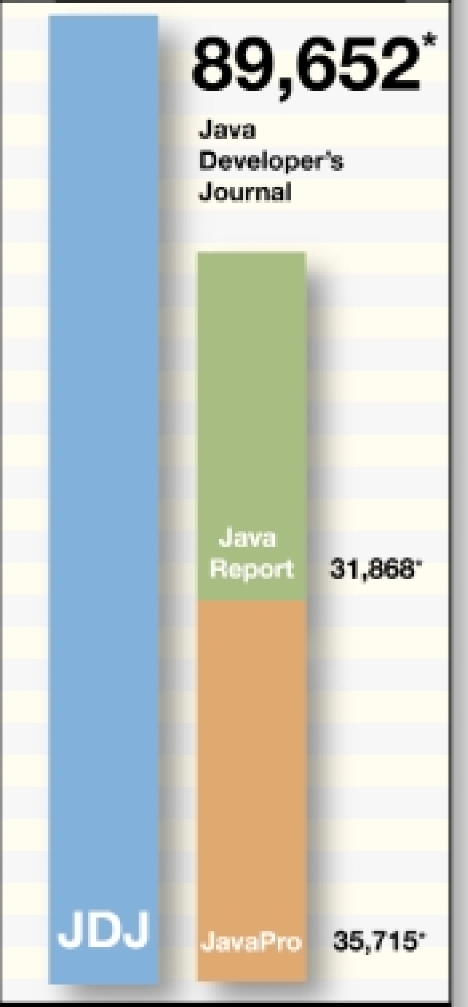
Why would you advertise anywhere else?



Your ad in *JDJ* reaches twice as many recipients as all other Java pubs combined for 1/2 the cost...*

*independent survey

They Don't Quite Stack Up



JDJ reaches more Java developers than all other Java magazines COMBINED!

* Most recent six month average monthly circulation of three magazines as per BPA International

carmen@sys-con.com

miles@sys-con.com

SYS-CON MEDIA

www.JavaDevelopersJournal.com

(201) 802-3020

EXTREME PERFORMANCE TUNING

PART 2 of 3

Uncommonly good solutions to common application problems

Written by James McGovern

The performance of J2EE-based applications sometimes doesn't live up to users' expectations. Usually it's impossible to quantify exactly where the bottlenecks are. Many developers spend time searching for articles on the Internet only to find the same old tips about using the synchronized keyword and string concatenation without ever finding information that's useful. This article will help you find the holy grail of Java performance.

In my previous article (***JDJ***, Vol. 6, issue 9) we focused on tips that are common to most applications. The tips presented here focus on common problems found within applications that utilize JSP, EJB, JNDI, and JDBC. Future articles will cover some powerful techniques to help your applications perform better.

Session Invalidation

Many application servers typically have a default timeout value of 30 minutes for cleaning up dead sessions. When an application server can't hold any more sessions in memory, it may cause the operating system to page out portions of memory. The application server may also swap unused sessions to disk based on the most recently used algorithm or even throw OutofMemoryExceptions. In a high-volume system, serializing sessions can be expensive.

Calling HttpSession.invalidate() is the recommended way to clean up a session when you no longer need to use it. This method is usually called as part of an application's logout page.

Sessions and JSP

The J2EE specification requires all JSP objects that can be referenced in JSP source and tags be usable without explicit declaration. For Web pages that don't require session tracking, you can save resources by turning off automatic session creation using the following page directive:

```
<%@ page session="false"%>
```

Servlet and Memory Usage

Many application developers are guilty of storing way too much information in the users session. Sometimes these objects don't get garbage collected in a timely manner. The typical performance symptom may be periodic slowdowns that are reflected in the user experience but aren't traceable to any particular component. If you're monitoring the JVM's heap size, this may be reflected as significant peaks and drops in memory usage instead of a normal pattern.

There are a couple of ways to even out memory usage in this scenario. The first recommendation is to have all beans that are scoped as session implement the HttpSessionBindingListener interface. This allows you to explicitly release resources that are used within the bean by implementing the method valueUnbound().

The other approach is to simply expire sessions more quickly. Most application servers have settings that usually specify the interval. You can also do it yourself programmatically by calling session.setMaxInactiveInterval(), which specifies the time in seconds between client requests before the servlet container will invalidate the session.

HTTP Keep-Alive

The majority of Web servers on the market, including iPlanet, IIS, and Apache, support HTTP Keep-Alive. This function keeps an open connection from client to server so that subsequent requests to the server don't have to be established or reestablished. This is usually a good technique for sites serving static content. The problem with sites with heavier loads is that the benefit of keeping a connection open for a single client also has a performance penalty as it ties up resources when the process is idle. This resource usage is even more important if your Web and application server are one and the same.

Choose the Right Include Mechanism

A typical JSP architecture may break out headers, footers, and navigation into their own resources, included in each JSP page as appropriate. Currently there are two methods to include a resource: the include directive and the include action.

- ***Include directive <%@ include file="copyleft.html" %> :*** Includes the content of the resource at compile time. The page with the directive and the resource are merged into one file before final compilation. When resources are resolved at compile time, they'll always be faster than resources resolved at runtime.
- ***Include action <jsp:include page="copyleft.jsp" /> :*** Includes the response generated by executing the specified page. Since this is done at runtime, we can vary the output produced. Use this action only for content that changes often and for scenarios in which pages to include can't be decided until the main page has been requested.

Use Cache Tagging Features

Several vendors have added cache tagging features to their application servers for use with JSP. BEA's WebLogic Server introduced this feature in the 6.0 versions of its product. This feature is also supported by the Open Symphony project. JSP cache tagging allows both fragments and page-level information to be cached. When a JSP page executes, if a tagged fragment is found in cache, the code that creates the fragment is skipped.

Page-level caching catches requests for specific URLs and caches the resulting output. This feature is extremely useful for shopping cart/catalog and/or portal home pages. In this scenario a page-level cache can store the resulting content to satisfy future requests. Here's an example URL that would be a good candidate: http://quote.yahoo.com/q?s=hig&d=t.

Use of cache tagging features provides performance increases for applications where there's significant logic; and has less of an effect for truly architected sites that utilize an MVC (Model View Controller)-based framework.

"This feature is extremely useful for shopping cart/catalog and/or portal home pages"

Always Access Entity Beans from Session Beans

Accessing entity beans directly is bad for performance. When a client application accesses an entity bean, each get method is a remote call. A session bean accessing the entity bean is local and can collect all data in a structure and return it by value. You can read more about the value pattern in *Design Patterns* by the Gang of Four.

Using a session bean to wrap access of an entity bean allows for better transaction management as the session bean will commit only when it reaches a transaction boundary. Each direct call to a get method results in a transaction. The container will execute a store-and-load after each transaction on an entity bean.

At times using an entity bean will result in bad performance. If the only purpose for an entity bean is to retrieve and update values, you'll gain better performance using JDBC within session beans.

Use Read-Only in the Deployment Descriptor

The deployment descriptor for an entity bean allows all get methods to be set as Read-Only. This increases performance when the unit of work in a transaction contains no methods other than read-only, as the container won't invoke the store.

Cache Access to EJB Homes

EJB Home interfaces are obtained through a JNDI naming lookup. This operation requires significant resources. A good place to put lookup code is within a servlet's init() method. If your application requires EJB access by multiple servlets, it would be wise to create an EJBHomeCache class. This class typically would be implemented as a singleton.

Consider Local Interfaces for EJBs

Local interfaces is an addition to the EJB 2.0 specification, allowing a bean to avoid the overhead of a remote invocation call. Consider the following code:

```
PayrollBeanHome home = (PayrollBeanHome)
javax.rmi.PortableRemoteObject.narrow (ctx.lookup ("PayrollBeanHome"),
PayrollBeanHome.class);

PayrollBean bean = (PayrollBean) javax.rmi.PortableRemoteObject.narrow
(home.create(), PayrollBean.class);
```

The first statement indicates that we want to find the bean's home interface. This lookup is via JNDI, which is an RMI call. We then locate the remote object and return the proxy reference. This too is an RMI call. The second statement demonstrates creating an instance. This code points to a stub that creates an IIOP request and transmits it over the network. This too is an RMI call.

To implement this functionality, all you have to do is extend from EJBLocalObject instead of EJBObject. My Pentium 700MHz machine, a change from remote to local interfaces, sped up method calls overall by 20%.

> **"There's only one appropriate answer:** consider using RMI and JNDI together**"**

To implement local interfaces, you may have to make the following changes:

1. ***Methods can no longer throw java.rmi.RemoteException.*** The rule also applies to exceptions extended from RemoteException, such as TransactionRequiredException, TransactionRolledBackException, and NoSuchObjectException. EJBs provide equivalent local exceptions (TransactionRequiredLocalException, TransactionRolledBackLocalException, and NoSuchObjectLocalException).
2. ***All data and return values are passed by reference, not by value.***
3. ***The local interface must be used on the machine where the EJB is deployed.*** In simpler terms, everything must be within the same JVM. This limits you to deploying on nonclusterable applications.
4. ***References for beans that implement local interfaces aren't serializable.***

Consider Writing Your Own Stubs

Stubs are responsible for forwarding method invocations to remote beans, and are typically generated by most deployment tools. If you haven't already implemented the value pattern, you can implement this functionality before anything is transmitted over the wire by modifying the stub. In modifying the stub, you can also implement your own caching routine or even compress data before it's transmitted.

Normally this technique isn't recommended and would be against previous recommendations about being clever. You'll have to worry about the deployment tool overwriting your changes as well as having your stub figure out when to reload data if another client changes information.

Essential Technologies for Today's B2B Integration Challenges

Web Services EDGE

PRE-REGISTER ONLINE TODAY SAVE $300

SEPTEMBER

New York, NY

web services EDGE conference & expo EAST COAST 24-25

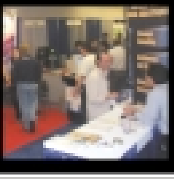
JDJ EDGE conference & expo 23-26

OCTOBER

Santa Clara, CA

web services EDGE conference & expo WEST COAST 24-25

XML EDGE conference & expo 2001 22-25



each venue features:

- Over 100 Information-Packed Sessions
- Two Distinct Conferences for Just One Registration Fee
- Pioneers and Visionaries
- Leading Vendors Presenting New and Problem-Solving Products

ColdFusion fast track
SEPT. 24-25, 2001
TWO
CONCENTRATED DAYS OF LEARNING TO JUMPSTART YOUR PROFESSIONAL CREDENTIALS!

Participating Companies, Sponsors and Exhibitors as of June 1, 2001

TogetherSoft, SilverStream, sitraka software, sonic software, aligo, KENAMEA, shinka technologies, SAVAJE, INFERDATA CORPORATION, bea, InCert, SEAGULL, DataQuad, ParaSoft, idoox, CAPE CLEAR, MERANT, PointBase, hp invent, pingtel, TopCoder, NOVA LABS, Rogue Wave Software, Bestning, Rational, DataMirror, CodeHound.com, OASIS, ORACLE, DATA JUNCTION, ENTEGRITY Solutions, TOWER, xml spy, TALKING BLOCKS, WHITEHILL, AdvanTel, XML.ORG, XMLCities, XML-JOURNAL, Programmer's Paradise, COLDFUSION Developer's Journal, WebSphere, WebServices Journal, WebLogic, webtechniques, JAVA Developer's Journal, SD Times, wireless

Secure Your Seat Now!

Two Easy Ways to Register

1 On the Web: www.SYS-CON.com

2 By Telephone: 201 802-3069

Be sure to visit www.SYS-CON.com for travel information. Reserve Your Hotel Room Early!

SAVE THE DATES! WIRELESS EDGE MAY 7-9
BUSINESS & TECHN
wirele
DRIVEN to Distraction
SMS IN THE U.S. MARKET: Why Doesn't It Work Here?
MACHINE to Machine Communications

Generating Primary Keys

There are many clever ways of generating primary keys within an EJB. I'll list several common techniques and then explain why they're all bad.

You can use the database's built-in identity (SQL Server IDENTITY or Oracle's SEQUENCE). This makes the implementation of your EJB nonportable (bad).

You could have an entity bean increment its own value, but this is bad as it requires serializable transactions, which are also slow.

You could use a time service such as NTP, but this requires native code and locks your bean to the particular OS. This approach also allows for the potential of generating two primary keys in the same millisecond on multiple CPU servers.

You could architect your bean and steal some ideas from Microsoft by creating a GUID, but you'll run into the fact that Java can't determine the MAC address of your network card without resorting to JNI, which will make your bean OS-dependent.

You'll also run into an issue if you try to utilize System.currentTimeMillis(), which will have the same issues previously mentioned. It'll be hard to identify a comparable formula. You could try to implement statics, but you'll fail as they aren't supported in the EJB specification.

There are several other approaches but they all have their limitations. There's only one appropriate answer: consider using RMI and JNDI together. You'll start with binding the RMI remote object to the JNDI tree via the RMI registry using the JNDI service provider interface. Your clients will look up the singleton via JNDI. Here's an example:

```
public class keyGen extends UnicastRemoteObject
    implements Remote {

    private static long keyVal = System.currentTimeMillis();

    public static synchronized long getKey()
        throws RemoteException {
            return keyVal++; }
```

“Many applications should consider placing a constraint on cache growth”

JDBC and Unicode

Hopefully you've read the typical industry recommendations about using JDBC such as using connection pools, preferring stored procedures or direct SQL, using type 4 drivers, removing extra columns from the result set, using prepared statements when practical, having your DBA tune the query, and choosing the appropriate transaction levels.

Besides the more obvious choices, the best action you can take to increase performance is to consider storing all character data in Unicode (Code page 13488). Java processes all data in Unicode and therefore the database driver doesn't have to perform a conversion. But beware: taking this step will cause your database size to grow, as Unicode requires 2 bytes per character. You'll have to worry about performance when you implement this tip if you have non-Unicode applications accessing the data, as a conversion will also occur.

JDBC and I/O

When your application requires access to a large result set, consider implementing block fetches. By default, JDBC fetches 32 rows at a time. As an example, web services EDGE conference&expo™

October 24–25, 2001

XMLEDGE conference&expo 2001

October 22–25, 2001™

Santa Clara Convention Center Santa Clara, CA

FREE Registration

Register for one...attend both

See them First at XMLEdge2001

Fundamentally Improving the Speed, Cost & Flexibility of Business Applications

Many New XML Products Will Hit the Market in Q3 of 2001

Opening Keynoter

Charles Goldfarb

The Father of XML

SYS-CON MEDIA

Owned & produced by SYS-CON EVENTS

WEB SERVICES CEO PANEL

Litwack
CEO
SilverStream

O'Connor
President
Sonic Software

Kutay
CEO
AltoWeb

O'Toole
Chairman
Cape Clear

Slama
CEO
Shinka Technologies

Morris
CEO
IONA Technologies

w w w . s y s - c o n . c o m

Platinum Sponsor: ca Gold Sponsor: GADOS Silver Sponsor: SilverStream Corporate Sponsors: xml spy Association Sponsors: OASIS XML

Media Sponsors: XML Journal, Java Developer's Journal, WebServices, WebLogic, WebSphere, ColdFusion Developer's Journal, wireless, SD Times, Programmer's Paradise

WEB SERVICES: THE NEW KILLER
WebServ
PREMIER ISSUE
WSJ2.COM
BUILDING WEB SERVICES USING
How to Harness the Con
Web Services CEOs Speak Out

if you needed to iterate through a result set of 5,000 rows, this would cause JDBC to make 157 calls to the database to fetch data. If you changed the block size to 512, this would require only 10 round-trips.

This tip may not work in several scenarios. If you use scrollable result sets or specify FOR UPDATE as part of the query, blocking isn't used. Another technique to use is the Page-by-Page Iterator pattern.

Consider Using an In-Memory Database

Many applications have the need to store a significant amount of data on a per-user basis in the session. Typically, you may see this implemented as a shopping cart and catalog. Since this type of data is represented as row/column data, many applications may create either large vectors or hashmaps. Keeping this type of data in session severely limits scalability as you must have at least the amount of memory per session times the maximum number of concurrent users, which can either be a very expensive server and/or could stretch garbage collection times to unbearably long periods.

To achieve slightly better scalability, some people have offloaded the shopping cart/catalog functionality to the database tier. The fundamental problem with the database tier is based on the architecture of most relational databases. The main principle at work is that they try to make all writes durable; hence all performance is tied to the ability to physically write the data to disk. Relational databases try to reduce I/O, especially for read operations, but accomplish this goal only by execution of complex algorithms that implement caching and are the main reason that the database tier's number-one bottleneck is usually CPU.

There is an alternative. Consider using an in-memory database. Several vendors have addressed this market. I'm a fan of TimesTen, but you're welcome to choose your own. They start by allowing data to be temporarily written but not necessarily persisted to disk, and keeping all operations in memory. As a result, they also don't need complex algorithms to reduce I/O, and are faster because they can employ simpler locking mechanisms.

Develop a Smarter Caching Mechanism

Many application developers have implemented caching mechanisms for frequently requested data. These caches are typically implemented as hashmaps. The main problem with this approach is that there's no constraint on how large the hashmap can grow. Many applications should consider placing a constraint on cache growth by using algorithms utilizing a strategy of keeping only the most recently used objects. This is best accomplished by combining a hashmap with a LinkedList.

The code should implement the following logic:

1. If the cache is full, remove the last object from the tail of the list and insert the new object at the head of the list.
2. If the cache isn't full, and you want to insert a new object, put it at the head of the list.
3. If the object is already in cache, move it to the head of the list.

Conclusion

Optimizing code is one of the last things developers should consider. Slow applications that work are preferred to fast applications that don't. Keep in mind that performance is sometimes in perception. It's possible to optimize a user's perception of performance without necessarily optimizing an application. Consider providing immediate feedback, as users will be happier to view a screen that paints immediately and takes 10 seconds to process than to have a screen that paints itself in seven seconds.

AUTHOR BIO

James McGovern is an enterprise architect with Hartford Technology Services Company LLC, an information technology services firm dedicated to helping businesses gain competitive advantage through the use of technology. His focus is on developing high-availability Internet applications.

james.mcgovern@htsco.com

wireless EDGE conference&expo

Plan to Exhibit

Provide the Resources to Implement Wireless Strategy

The conference will motivate and educate. The expo is where attendees will want to turn ideas into reality. Be ready to offer solutions.

INTERNATIONAL WIRELESS BUSINESS & TECHNOLOGY CONFERENCE & EXPO

CONFERENCE & EXPO

CONFERENCE & EXPO

Shaping Wireless Strategy for the Enterprise

Santa Clara, CA

May 7-9, 2002

Wireless Edge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, *i-technology* professionals and IT/IS management will eagerly attend.

Plan to Attend the 3-Day Conference

WHO SHOULD ATTEND

Mobile & Wireless Application Professionals who are driving their enterprise's wireless initiatives:

- Program Developers
- Development Managers
- Project Managers
- Project Leaders
- Network Managers
- Senior IT and Business Executives

SHAPE YOUR WIRELESS STRATEGY! SAVE THE DATES!

Conference Tracks

Track One: Development	Track Two: Connectivity	Track Three: Wireless Apps	Track Four: Hardware	Track Five: Business Futures
WAP	Smart Cards	Education	Cell Phones/ WorldPhones	Wireless in Vertical Industries
i-Mode	Wireless LANs incl. Bluetooth	Health Care	PDAs	The WWWW
Bluetooth / 802.11	UMTS/3G Networks	Entertainment	Headphones/ Keyboards / Peripherals	Unwired Management
Short Messaging	Satellite Broadband	Transport	Transmitters Base Stations	From 3W to 4W: Issues and Trends
Interactive Gaming		Financial Services	Tablets	"Always-On" Management
GPS / Location-Based		Supply Chain Management		Exploiting the Bandwidth Edge
Wireless Java				Unplugged Valueware
XML & Wireless Technologies				Wireless Sales & Marketing

FOR INFORMATION CALL

201 802-3084

EXCLUSIVE SPONSORSHIPS AVAILABLE

Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of

SYS-CON MEDIA

SYS-CON EVENTS

SPEAKER PROPOSALS INVITED

WWW.SYS-CON.COM

JDJ SPECIALS
Collect All Three!

JDJStore.com
$175.99
JDJ, XML-J, CFDJ
ALL THREE!!!
The Complete Works

JDJ BESTSELLERS
1. JProbe Profiler Serverside with 1 Year Gold Support Edition v2.8.1 from Sitraka Windows 2000/98/95
$969.99
2. ChantⓇ SpeechKit™ 3 Professional Edition Speech Recognition Developer's Software
$799.00
3. JClass Standard Suite with One Year Gold Support With Source Code from Sitraka
$3999.00
4. JFCSuite Enterprise Edition with Source Code v2.0 from Infragistics
$2494.99
5. Visual Café Standard Edition v4.0 from WebGain
$97.99

What's Online...

October 2001

JDJ *Online*

Check in every day for up-to-the-minute news, events, and developments in the Java industry. Can't get to the newsstand on time? Visit www.javadevelopersjournal.com and be the first to know what's happening in the industry.

Search Java Jobs

Java Developer's Journal is proud to offer an employment portal for IT professionals. Get direct access to the best companies in the nation. Learn about the "hidden job market" and how you can find it. If you're an IT professional curious about the job market, this is the site to visit.

Simply type in the keyword, job title, and location and get instant results. You can search by salary, company, or industry.

Need more help? Our experts can assist you with retirement planning, putting together a résumé, immigration issues, and more.

Radio Interviews

Tune in to SYS-CON radio (via the Shoutcast Network) and hear the industry's top movers and shakers, interviewed by ***JDJ*** editors. Listen to James Gosling, the father of Java, ***JDJ*** Advisory Panel members, CEOs, product managers, and other top executives as they share their insight and opinions on today's hottest issues.

Product Reviews

Considering a product upgrade? Want to know the ins-and-outs of a new product before you purchase it? Then make sure you read our in-depth product reviews.

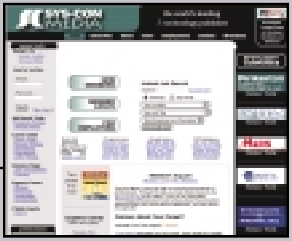
Our writers test and evaluate a host of Java products to aid your decision-making process. We get behind the hype and give you the facts. All reviews are written by experts and leaders in the information technology industry.

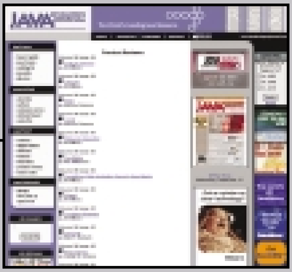
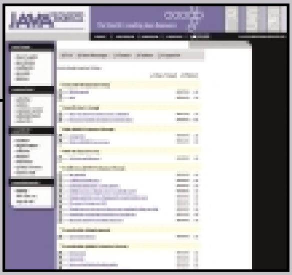
JavaDevelopersJournal.com Developer Forums

Join our new Java mailing list community. You and other IT professionals, industry gurus, and ***Java Developer's Journal*** writers can engage in Java discussions, ask technical questions, talk to vendors, find Java jobs, and more. Voice your opinions and assessments on topical issues – or hear what others have to say. Monitor the pulse of the Java industry!

JavaBuyersGuide.com

JavaBuyersGuide.com is your best source anywhere, anytime on the Web for Java-related software and products in more than 20 mission-critical categories, including application servers, books, code, IDEs, modeling tools, and profilers. Check the Buyer's Guide for the latest and best Java products available today.



Click for a FREE 30-day trial
AltoWeb

M-1 Mobile App Server
aligo
Download Now!

KL Group

Get SonicMQ™
PROGRESS SOFTWARE

DataDirect
No Excuses
J2EE CERTIFIED DRIVERS

IONA
Get 1 of 1000
FREE copies of

mobilizing eBusiness

spiritsoft
open java messaging

From Zero G
www.ZeroG.com

the world's #1 JavaBeans

TEOCO and Kada Team to Develop Mobile Enterprise Apps

(*Fairfax, VA / Burlington, MA*) – TEOCO has joined the Kada Mobile Developer Network (MDN). Developers at TELCO will utilize the Kada Mobile Platform (Kada Mobile) to tune, optimize, and deploy next-generation mobile applications.
www.kadasystems.com
www.teoco.com

ProSyst Launches mBedded Server 5.0

(*Cologne, Germany*) – ProSyst Software AG announced mBedded Server 5.0. The release offers complete support for OSGi standard 2.0, more flexibility in tests and implementations, and new capabilities for both the development of net-ready devices and the design of applications using Open Service Gateways scenarios.
www.prosyst.com

realMethods Announces New 1.5 Beta of Framework

(*Bridgewater, MA*) – realMethods, provider of the realMethods Framework, announced lower pricing and a complimentary program for their flagship J2EE Application Framework. realMethods also revealed forthcoming product enhancements in the release of the realMethods Framework 1.5 Beta.

The Complimentary Framework program gives realMethods' clients the opportunity to use the Framework for one year at no cost.
www.realmethods.com

iPlanet Web Services Integration Platform

(*Santa Clara, CA*) – iPlanet launched the EAI and B2B versions of iPlanet Integration Server, its latest release to the Sun ONE Web services vision. Director of product marketing, Sanjay Sarathy, spoke to ***JDJ*** regarding the significance of this release: "This release enables iPlanet to be one of the only platforms that enable the full Web services integration service."

This release includes a complete implementation of SOAP and XSLT modules, the latter based on the NetBeans open-source architecture, to allow developers to easily create rules and schemas.

Full details, along with a free download, can be found at www.iplanet.com.

Vertel Announces Java Edition of OM Product Line

(*Woodland Hills, CA*) – Vertel Corp. announced the availability of the Java Edition of its Object Management (OM) product line.

The OM product line provides network management functionality for embedded equipment applications, telecom management applications, and integration with legacy as well as next-generation operations support systems (OSS) in both C++ and Java.
www.vertel.com

IONA Announces Orbix/E 2.0

(*Boston*) – IONA announced a C/C++ and Java version of Orbix/E 2.0 (previously marketed as Orbacus/E), a lightweight distributed computing platform optimized for embedded applications.

Orbix/E 2.0 for Java offers a pure Java implementation with functional equivalence to the C/C++ version of the product. With support for Sun Microsystems' J2ME, J2SE, and PersonalJava specifications, Orbix/E 2.0 is suitable for a variety of wireless and handheld devices.
www.iona.com

Espial Delivers First Scalable Java Browser

(*Ottawa, Canada*) – Espial announced the release of Espial Escape 4.8, a 100% Java browser offering a scalable configuration that matches the memory requirements of a wide range of devices. Escape 4.8 allows the developer to selectively disable support for certain Internet standards so the browser can be tailored to run in resource-constrained designs or offer full functionality for other, more powerful devices.
www.espial.com

LogicChain Acquired to Form Crisp Wireless

(*New York, NY*) – LogicChain, Inc., announced that Crisp Partners has acquired an equity stake in the company to form a new mobile content–services company, Crisp Wireless. Crisp Wireless, focused on mobile content, entertainment, and multimedia application and service development, launches operations backed by a combination of LogicChain's mLogic Engine and Crisp Partners, a group with executive-level experience from premier wireless and media companies.
www.crispwireless.com

CyberTeams Upgrades Content Management Products

(*Frederick, MD*) – CyberTeams, Inc., released version 2.1 of WebSite Director Pro, its Web content–management system. In addition to expanding its content-management capabilities, the release includes a new Java-based user interface, enhanced workflow management, robust template-based content deployment, and database publishing capabilities.
www.cyberteams.com

Sims Computing Releases Flux 3.1

(*Billings, MT*) – Sims Computing introduced Flux 3.1, a software component that provides job scheduling functionality to Java and J2EE applications.

Flux 3.1 contains 30 improvements over Flux 3.0, including a Web-based management console for administering Flux from a Web browser and support for interacting with Flux using XML.
www.simscomputing.com

Insignia Names Potts as Senior VP of Sales

(*Fremont, CA*) – Insignia Solutions has appointed Lamar Potts, former Be, Inc., executive, as senior vice president of worldwide sales. Potts is responsible for the day-to-day management of sales including direct and OEM sales, sales strategy, and business development. He will report to Mark McMillan, Insignia's president and chief operating officer.
www.insignia.com

i-net Announces New Release of Crystal-Clear

(*Berlin, Germany*) – i-net Software has introduced Crystal-Clear version 2.0, a Java report engine that enables the cross-platform usage of Crystal Reports. This new release comes in two flavors: an EJB version and a standard version, and both support runtime design components and crosstabs. A new configuration tool provides an easy interface for the deployment of the report engine.
www.inetsoftware.de

Next Month

JAVA DEVELOPER'S JOURNAL

Who's Hiring Now?

Five tips for finding a good fit

WRITTEN BY
BILL BALOGLU &
BILLY PALMIERI

Since most of our past articles have dealt with Java jobs from the engineer's perspective, we decided to write this month's column from the hiring manager's point of view.

A year ago you never would have been reading this article. At that time the battle for technical talent was fierce. Companies large and small were fighting tooth and nail for engineers. Recent college grads with no real-world experience were landing fat-salaried jobs. Companies couldn't get foreign-trained engineers into the United States fast enough. Money was no object.

Diving into this gaping breech of opportunity were hundreds of recruiting firms, providing engineers to the talent-starved tech industry.

Then suddenly it stopped.

Wall Street devoured its dot-com young and the Age of the Start-up quickly became the Age of the Layoff. Industry titans from Intel to HP are now shedding workers by the thousands.

Yet companies still need strong talent to fill key technical positions. And despite the fact that many technical people are now available, both hiring managers and skilled engineers report a new set of frustrations.

As companies try to save money by slashing their recruiting budgets and by using outside or in-house recruiters, the time-consuming task of recruiting has now fallen on the shoulders of individual hiring managers and HR staff.

Many of our client managers tell us that this method isn't working. Managers in mid-sized to large companies tell us they're getting lots of résumés, which gives the illusion that top engineers are like cherries to be picked off a tree.

But the managers report that most of those innumerable résumés are from B- and C-level candidates, not the A-list engineers.

A-list engineers (who are still in short supply) tell us they don't like submitting their résumés directly to companies for fear of getting lost in the shuffle. Their experience is often not recognized by HR staff (whose expertise is in benefits, policies, and paperwork, not in identifying technical talent).

Many seasoned engineers are also reluctant to negotiate their own rate or salary package with a hiring manager. When employees must negotiate their own deals, they often carry that tension or conflict into the workplace. Managers at smaller companies have an even tougher time because they're now expected to do all of the recruiting themselves. Sifting through hundreds of résumés, phone-screening candidates, and checking references can be extremely time-consuming.

Glut of Professionals

Anyone in a position to hire wants to find the best person for the job. The temptation of many hiring managers is to take someone who's overqualified simply because market conditions have caused those people to become available.

At first it looks like a great deal to hire, say, an architect for an individual contributor role. But what that hiring manager may not realize is that overqualified employees will quickly become bored and will be gone as soon as they find more challenging jobs that are a true fit for their skills.

A key part of making a successful placement is matching the candidate's professional and personal goals with the position.

Since many hiring managers today have no choice but to take on the responsibility of recruiting for their own hires, we'd like to share a few key recruiting tips that should make the process easier:

1. *Be specific about what you're looking for.* The job description should clearly state all of the responsibilities of the position, including what tools will be used. Required skills should be listed completely (including the minimum number of years' experience in key skills). Desired skills should be listed separately but completely (including experience in a certain industry, such as wireless or software development). Intangible skills should be included (such as working well in a team environment).

2. *Start by asking everyone you know for referrals, both inside and outside of your company.* Statistics prove that the best placements come through personal and professional referrals.

3. *Look for both technical abilities and personal qualities that make the candidate a fit for your position.* This is someone you'll be working with every day. Make sure that his or her personality meshes with yours and that the person will get along well with the others in your group.

4. *Be specific in your phone screen, covering technical skills as well as personal motivations and goals.* Before you bring anyone in for a face-to-face interview with members of your team, be sure that he or she is a strong candidate. Asking your engineers to meet with someone who's not right for the job can cast doubt on your credibility.

5. *Make sure that members of your team focus on different topics in the face-to-face interview.* Make sure you cover all your bases.

• • •

If as a hiring manager you're confident that doing your own recruiting is working, then great. If not, a better investment of your time and efforts could be working with an outside resource that specializes in the types of positions you're offering.

Many managers have learned that just as it takes a thief to catch a thief, it takes a professional recruiter to catch a true professional.

AUTHOR BIOS
Bill Baloglu is a principal at ObjectFocus (www. ObjectFocus.com), a Java staffing firm in Silicon Valley. Previously he was a software engineer for 16 years. Bill has extensive OO experience and has held software development and senior technical management positions at several Silicon Valley firms.

Billy Palmieri is a seasoned staffing industry executive and a principal of ObjectFocus. His prior position was at Renaissance Worldwide, a multimillion-dollar global IT consulting firm, where he held several senior management positions in the firm's Silicon Valley operations.

billb@objectfocus.com

billy@objectfocus.com

Hacking...

WRITTEN BY
BLAIR WYMAN

Life happens at a dizzying pace. It seems like yesterday that I was writing my first Cubist Thread, in which my abundance of personal failings was first publicly perused. One that didn't make the list at the time, but for which I should be roundly criticized, is vanity.

Oh, I guess my account of frustrated aspirations to rock stardom could be construed as vanity, though it's nowhere near as egregious as this most recent incident.

The other day at the mall, while hunting for something or other, I was struck by a desire to go into the bookstore and buy a hard copy of ***Java Developer's Journal***. I knew that I'd find my musings amongst its pages, and I never had the experience of walking up to a newsstand and buying something containing my own words.

I plunked down my hard-earned cash, tore off the plastic cover right there at the register, and turned to the back page. "There I am." I said (as matter-of-factly as I could muster) as I showed my picture to the clerk.

She smiled and said, "Hey, that's cool." I probably looked like an overheated basset hound as that big grin spread across my face.

People in Minnesota are so nice. What I probably deserved was something more along the lines of, "So what? Do you want a cookie?" …or maybe, "You should be properly ashamed of your conspicuous vanity, you pompous noodle!"

But that's just like Minnesota – people are nice around these parts. Hot dish, anyone?

The fact is, I think people are generally nice most everywhere. I've met a lot of people in the many and various jobs I've had, but most of my "people skills" were probably obtained as I was "hacking" in a major American city.

Oh, now... Wait a minute. I don't mean the pejorative, computer-related verb "to hack," as in "to gain unauthorized access to a computer." (Of course, I pretentiously decry that definition – see the book *Hackers* by Steven Levy.)

In fact, I'm not talking about any computer-related form of hacking at all – I'm talking about driving a hack, aka taxicab. I drove a cab in one of the larger cities in the U.S. for a couple of years, and believe me when I tell you there may be no better way to meet a broad cross section of humanity.

Oh, I won't bore you with any of the wild stories I have from those years – yet – but meeting someone for the first time, taking their money in exchange for transportation, and then sharing something as personal as a ride with them is a fast lesson in human nature.

Well, I guess I could "tease" you with highlights from the stories that will undoubtedly influence this column, at least to the extent they influenced me. I've been conned, overtipped, stiffed, robbed, cheated, belittled, and downright threatened by some of the best.

One of my "brushes with fame" includes some serious haranguing by the comedian Eddie Murphy. Late one evening I remember picking up two gentlemen: a smaller, wiry fellow who sat in the seat directly behind me, and another man (best described as large enough to deserve his own zip code) who sat diagonally behind me.

All the way to the club the smaller fellow was joking with me, declaring his amazement that anyone like me would be dumb enough to drive a cab in this city at night. "You're crazy!" he said, "...and you're probably going to get robbed!" The other fellow just sat there, attracting the planet Jupiter toward him to a significantly greater degree than anyone else for miles around, and silently glaring at me whenever I turned around.

Well, I'd already been robbed by this time, and I told him as much, but he kept ribbing me. I was actually a little scared, so I just did the usual – tried to "cut wise" and make my passengers laugh. On this occasion, though, it just plain wasn't working. Every "witty" thing I said was fielded with incredible aplomb, and returned back to me doubled in every dimension.

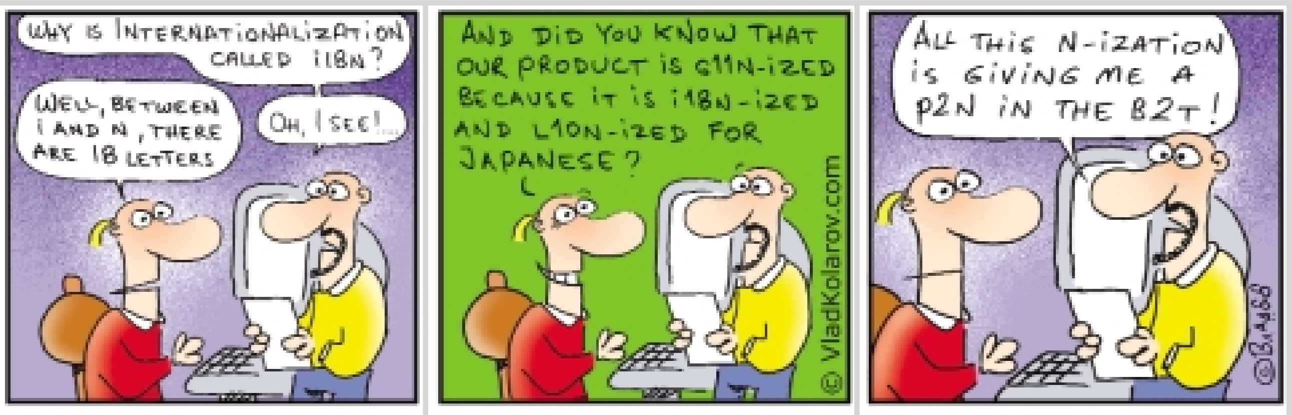
When the ride was over, and it was apparent that I'd be okay, the passenger identified himself to me through the window. "Do you know who I am?" he asked. "I'm Eddie Murphy!"

It was 1981 or so, and I don't think I had seen him on *Saturday Night Live*, yet – Saturday night was a big work night for cabbies, after all. One thing was certain, though – his incredible talent was unmaskable. To this day I've never felt so utterly "bested" as I did that night.

Eddie, if you're out there, I'm a huge fan.... Oh, and thanks for not robbing me.

blair@blairwyman.com

AUTHOR BIO
Blair Wyman is a software engineer working for IBM in Rochester, Minnesota, home of the IBM iSeries.


WHY IS INTERNATIONALIZATION CALLED I18N?
WELL, BETWEEN I AND N, THERE ARE 18 LETTERS
OH, I SEE!...
AND DID YOU KNOW THAT OUR PRODUCT IS G11N-IZED BECAUSE IT IS I18N-IZED AND L10N-IZED FOR JAPANESE?
© VladKolarov.com
ALL THIS N-IZATION IS GIVING ME A P2N IN THE B2T!