JAVA™ DEVELOPER'S JOURNAL

The World's Leading Java Resource

July 2000 Volume: 5 Issue: 7

JAVADEVELOPERSJOURNAL.COM

Announcing... JavaCON 2000 Coming September 24–27, 2000

XML DevCon FALL 2000 Coming November 12–15, 2000

see page 24

EXCLUSIVE JavaOne Interview... James Gosling of Sun Microsystems, Inc.

FREE CD! FROM ALLAIRE

SYS-CON MEDIA

Progress Software

www.sonicmq.com/ad4.htm

Protoview

www.protoview.com

Allaire Developer Conference

www.allaire.com/conference

JAVA DEVELOPER'S JOURNAL

JDJ CONTENTS

VOLUME: 5 ISSUE: 7 JULY 2000

No Magic

www.magic-draw.com

TogetherSoft Corporation

www.togethersoft.com

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

WRITERS IN THIS ISSUE

DAVOR BISKO, MATTHIEU DEVIN, RICK HIGHTOWER, JUSTIN HILL, CHRIS KANG, COLLEEN MCCLINTOCK, JIM MILBERY, SEAN RHODY, JEFF RICHEY, CHRIS ROBERTS, AJIT SAGAR, BRUCE SCOTT, DARREN SHIELDS, JON SIEGEL, BARRY TAIT, JASON WESTRA, DEAN WILLIAMS

SUBSCRIPTIONS

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

SUBSCRIPTION HOTLINE: 800 513-7111

COVER PRICE: $4.99/ISSUE

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

PUBLISHER, PRESIDENT AND CEO:	FUAT A. KIRCAALI
VICE PRESIDENT, PRODUCTION:	JIM MORGAN
VICE PRESIDENT, MARKETING:	CARMEN GONZALEZ
GROUP PUBLISHER:	JEREMY GEELAN
CHIEF FINANCIAL OFFICER:	ELI HOROWITZ
ADVERTISING ACCOUNT MANAGERS:	ROBYN FORMA
	MEGAN RING
JDJSTORE.COM:	AMANDA MOSKOWITZ
ADVERTISING ASSISTANT:	CHRISTINE RUSSELL
ADVERTISING INTERN:	MATT KREMKAU
GRAPHIC DESIGNERS:	JASON KREMKAU
	ABRAHAM ADDO
GRAPHIC DESIGN INTERN:	AARATHI VENKATARAMAN
WEBMASTER:	ROBERT DIAMOND
WEB SERVICES CONSULTANT:	BRUNO Y. DECAUDIN
WEB DESIGNER:	STEPHEN KILMURRAY
WEB SERVICES INTERNS:	DIGANT B. DAVE
	BRYAN KREMKAU
CUSTOMER SERVICE:	ELLEN MOSKOWITZ
	STEVEN MILILLO

EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9600
SUBSCRIBE@SYS-CON.COM

JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is published monthly (12 times a year) for $49.00 by SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645. Periodicals Postage rates are paid at Montvale, NJ 07645 and additional mailing offices. POSTMASTER: Send address changes to: JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

SEAN RHODY, EDITOR-IN-CHIEF

Net Market Madness

I spent a couple of weeks in Florida recently – ignoring the Internet and hoping the market dip would go away. It felt good not to pull e-mail (all right, I did, but not every day) and it gave me some time to think about the whirlwind pace that's been the routine of the past year.

About a year ago I got involved in the development of what I call "Net Markets." You're probably familiar with the concept but you probably haven't logged onto one. In its simplest form a Net Market is eBay – a place to buy or sell stuff – and in a more complex form it's the stock exchange. But eBay isn't a true Net Market – they're for businesses, not consumers.

When things first started up in the Net Market space last year, there were few vendors and even fewer initial sites. At the beginning most sites had the eBay model – auctions, auctions and more auctions. But auctions were the model most Net Markets wanted least.

As I mentioned in an earlier column, the true model for most Net Markets is the *negotiation*. While eBay works fine for commodities (I even broke down and bought an old Sun box from eBay so I could run Solaris), most industries have more complex requirements. Some markets rely on strict anonymity between negotiators, at least until the deal is consummated. And most deals are negotiated on more than simply price or quantity, because in most industries things like delivery times, freight costs, grades of materials and even documentation are as important as price. This became obvious to most of the early vendors and some quickly adjusted their product offerings to begin to offer negotiations.

What other fundamental changes do I see occurring in the next year or so? Well, it didn't take long for software vendors to realize that the software they were providing (at a very lucrative rate, in my opinion) was making other people rich. And as they watched Net Markets spring up around their products, they realized they had a limited audience. After all, with multiple vendors and a limited number of sites per market (who competes with eBay, for example?), the software vendors were faced with the realization that they would probably host only one Net Market per industry. Couple this with a finite number of industries and things didn't look all that great. At least until someone realized that they were the best ones to create a Net Market in the first place.

So a new breed of vendor is springing up – one that wants a piece of the action in exchange for the expertise needed to create a Net Market. Alternatively, some vendors want to do it all themselves. I've seen this from infrastructure providers; now we're seeing software vendors beginning to do the same thing. Couple this with the fact that most of the original Net Market companies have now realized they can replicate their business processes in other industries to leverage their investment in intellectual capital and software expertise and we get another trend: the Net Market Hub. The Hub takes over where the Net Market ends, leveraging common infrastructure and processes to provide synergy across multiple industries.

One last trend that we'll see a great deal more of is integration. The first wave of services has been created: we can buy or partner to obtain auctions, exchanges or negotiations. The next wave will be integration of these services with several other vital areas. Logistics, CRM, ERP and accounting package integration will be the next wave for the Net Market Hub. Like every other business process, the Net Market will be grown to achieve economies of scale. We'll be able to negotiate shipping along with other aspects of the deal. We'll be able to get integrated account statements and track physical products after closure of the deal. We'll also be able to arrange credit and even book complicated transactions between more than two partners.

Where does the madness end? This year it will end with vendors becoming partners, partners becoming vendors; with logistics, credit and CRM becoming integral services rather than à la carte offerings. The winners will be the ones that can put it all together the quickest and leverage their current investments. No matter what, it promises to be an interesting year. I'm already booking my trip to Florida next year to recover.

sean@sys-con.com

AUTHOR BIO

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

J D J F E A T U R E

IMPLEMENTING

Concluding our two-part series on integrating rule engines into Java applications

INTEGRATING BUSINESS RULES IN J2EE

WRITTEN BY COLLEEN MCCLINTOCK & CHRIS ROBERTS

Part 1 of this series on business rule engines (see "Implementing Business Rules in Java," **JDJ**, *Vol. 5, issue 5) addressed the question of how to integrate the rule engine into a Java application. To review…business rules are the policies and procedures that describe or constrain the way an organization conducts business.*

They're everywhere: in corporate charters, marketing strategies, pricing polices, product and service offerings, and customer relationship management practices. Business rules are also in the legal documents that regulate your business and industry. Rule engines interpret and implement rules within software systems. Soon they may become better integrated with the Java platform.

RULE ENGINES MAKE THEIR COMEBACK

Twenty years after it first made waves, rule-based technology is making a comeback. Java developers with an eye on the e-commerce market are becoming aware of how integrating business rules and objects in Java can help expand Java into new niches within Web-based applications.

BUSINESS RULES IN JAVA PART 2

Recent activities within the Java community have brought about a reexamination of business rule engines and their relationship with Java – specifically the idea of having a rule engine as a reusable commodity within the J2EE environment. Ultimately this would require a standard API and rule language that would allow Java developers to create applications that can execute business rules using any rule engine implementation. They wouldn't be tied to a particular vendor solution or technology. The rule engine API and rule language would be analogous to what the JDBC API and SQL provide for database integration. In the meantime, vendors are providing their own solutions for J2EE integration.

This article provides an overview of the proposed integration architecture for business rules in the J2EE (Java 2 Platform, Enterprise Edition) environment. Within this context we'll discuss the requirements for integration and the steps of business rule development and deployment.

The proposed architecture would support the following requirements:

- ***Separation of business and application logic:*** Business rules are managed separately from the application.
- ***End-user access to rules:*** Business users (nonprogrammers) can add and modify business rules in an implementation-independent business rule language.
- ***Access EJB components:*** Rules access EJB objects.
- ***Standard rule engine interface:*** Rule engines are accessed via a standard API regardless of the specific rule engine implementation.

Requirements for Integration

To a large degree, the J2EE helps manage application changes by standardizing how multitier enterprise applications are developed and deployed. It accomplishes this by simplifying enterprise applications and basing them on modular components that have access to a complete set of reusable services. It also hides many details of application behavior automatically, without complex programming. Positioned as just another reusable service, the rule engine could take enterprise applications to the next level by isolating the business logic that often requires change. The use of a rule engine in the architecture permits the separation of the business rules from the application code.

This separation is one of the primary advantages of using a rule engine to implement business rules. It enables the development of flexible applications that can be easily changed to reflect new business policies. Business rules can be used to define new products and services, offer new marketing promotions, validate transactions against corporate or regulatory policy, or define new application workflows. Using a rule engine to isolate this logic enables the application to change and evolve at the speed of the business.

Rules executed by the different rule engines are written using proprietary languages specific to the third-party rule engines. In addition to a proprietary rule language, each third-party rule engine has a proprietary API used to control the assertion and retraction of Java objects to the rule engine and the loading and execution of rules. To isolate enterprise Java

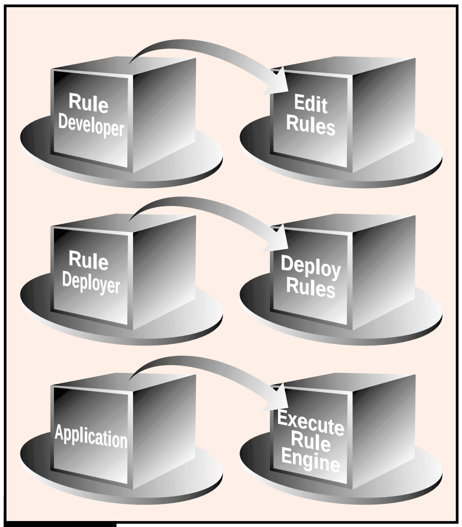


FIGURE 1 Use-case diagram for business rule implementation

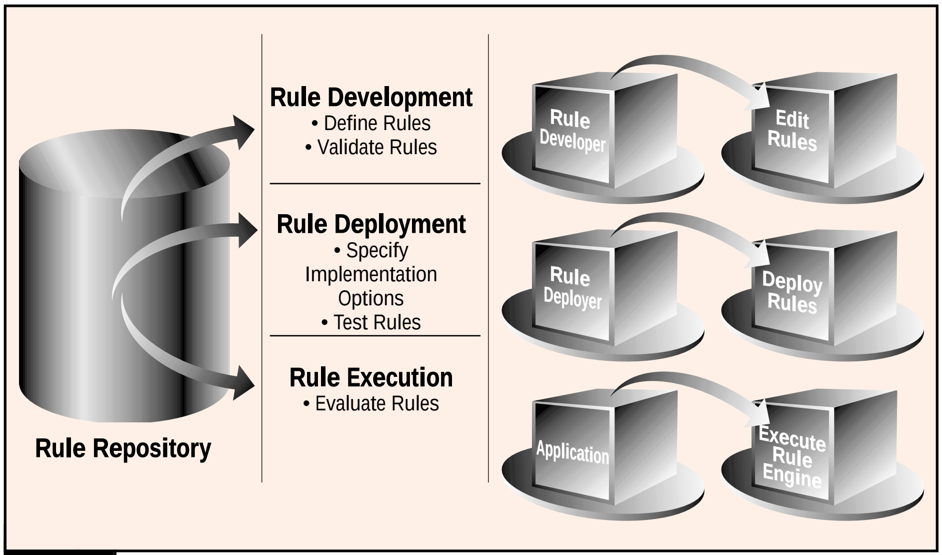


FIGURE 2 Business rule implementation process

applications from being dependent on a particular rule engine implementation, a standard API would be required.

In the absence of a rule engine integration standard, a rule engine must be compatible with the various EJB paradigms in order to execute correctly in the J2EE environment. It must be embeddable into any EJB paradigm – session (stateful or stateless) beans and entity beans. In addition, the rules executed by the rule engine must be able to reference EJB objects. In the standard Java environment Java objects are referenced in the rule engine's rule language either directly or indirectly through proxy objects, depending on the particular rule engine implementation. Since the object identification mechanism is different in the J2EE environment, the rule engine must be EJB aware. Specifically, in the EJB environment the "==" equality operator and "equals()" method don't yield predictable results. Instead, the EJBObject.isIdentical() method must be used by the rule engine when evaluating rule conditions. Today, rule engine vendors provide varying degrees of support for J2EE integration.

Overview of the Integration Architecture

In the Use Case diagram shown in Figure 1 there are three actors:

1. ***The Rule Developer*** is the business domain expert responsible for creating and maintaining the business rules. A stock trader defining the rules for the management of a client's portfolio, for example, is a Rule Developer. The Rule Developer defines business rules using the business rule language. Once defined, they're stored in a Business Rule Repository that supports business rule management functionality.
2. ***The Rule Deployer*** is the developer or other member of the IT department responsible for defining the implementation-specific characteristics associated with business rules. For example, the Rule Deployer is responsible for determining the specific applications in which the business rules are deployed, and the specific rule engine used for deployment. To do this, the Rule Deployer selects an Engine Deployer that processes the business rules in the rule configuration set translating the logical rules into the appropriate Rule Engine's native rule representation.
3. ***The Application*** obtains one or more Rule Engine instances at execution time, asserts the Java components referenced by rules to the Rule Engine and retrieves the results of business rule evaluation.

Corresponding to these three actors, the business rule implementation architecture can be described by dividing the business rule implementation process into three phases: business rule development, business rule deployment and business rule execution (see Figure 2).

Business Rule Development

During the business rule development phase shown in the class diagram in Figure 3, the business user, or Rule Developer, defines business rules using a Rule Editor. The Rule Editor, a software component that provides an interface to the Rule Repository and rule management capabilities, supports the creation and modification of business rules through either a text or a graphical user interface (GUI). If the rules are defined using a text interface, they're validated for correct syntax using the Script Validator; alternatively, a GUI can be used to step the Rule Developer through the process of creating a syntactically correct rule. Once a business rule is syntactically correct, it's persisted in the Business Rule Repository. The Repository Manager implements security and can support multiple Business Rule Repositories.

The business rule language syntax used for creating and editing rules is a high-level business-oriented representation that's implementation-independent. Rules specified in the business rule language represent logical rules. This logical rule representation may actually map to one or more physical rules in the native rule representation dictated by the specific rule engine implementation. The Business Rule Syntax supports the specification of the business rules conditions and actions. The conditions of a business rule are patterns or predicates that match on EJB properties. These conditions can be simple comparisons between properties or they can invoke math, string or application-specific operations.

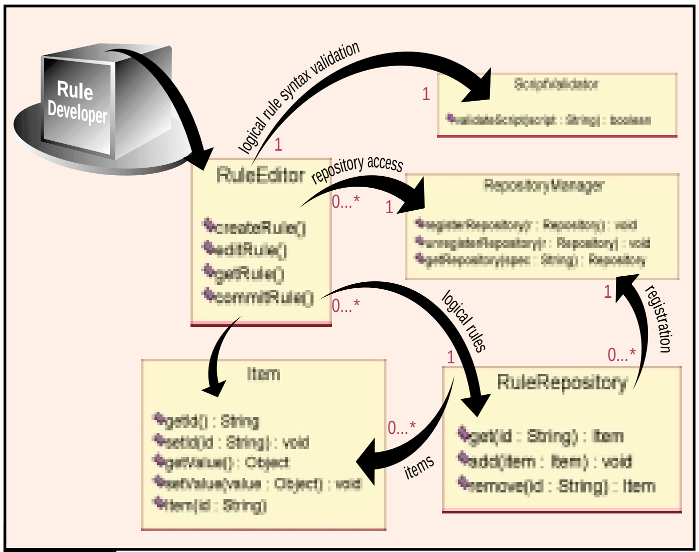


FIGURE 3 Class diagram of business rule development

Segue Software

www.segue.com

Actions can assert or retract EJBs or modify existing component properties.

The business rule can be stored in the repository as a string, a serialized object or an XML representation. A business rule can have additional properties stored in the repository such as rule name and ID, rule author, rule status, rule version and rule history.

RULE STATUS

The rule status is a property that can be used to track a business rule throughout its life cycle from definition to implementation. When business rules are first defined, they're assigned a status of *new*. Once the rule has been completely specified, along with its associated attributes, it has a status of *complete*. Next, the business rule may go through a business validation process in which other business users validate the rule. (For example, a new promotional marketing rule defined by a Rule Developer who is a business user in the marketing department may offer special billing rates for calls made during nonpeak hours for new mobile phone users. This rule may need to be validated by the product manager, the marketing manager and a representative from the billing department.)

After validation is complete, the rule status will either be *accepted* or *rejected*. Once a rule is accepted, it moves on to the deployment phase and may require unit and system integration testing. A successfully tested rule has a status of *tested*. When it has been implemented and is running in the production environment, the status is *implemented*. Finally, at the end of the promotion, the business rule may be retired and assigned a status of *inactive*.

RULE VERSION

The rule version property can be used to track the various versions of an existing rule. If the business rule in the previous example were changed to specify a different effective and expiration date, or if the promotional discount amount were changed, the user could define a new business rule to replace the outdated rule. However, rather than defining a new rule, the business user could modify the existing implemented business rule, creating a new version of the rule with a new version number and a status of *new*. This creates an audit trail of policy changes over time, a formal representation of corporate memory. Once a business rule is successfully defined and stored in the repository, it's ready for deployment.

"The rule version property can be used to track the various versions of an existing rule"

Business Rule Deployment

As you can see from the foregoing section of this article, the Rule Developer isn't concerned with business rule implementation but is responsible for managing business rules as a representation of the organization's business policy. Typically, the Rule Developer will be a nontechnical person – a marketing or HR professional, for example.

The Rule Deployer, on the other hand, is a technical person whose responsibility includes preparing business rules for implementation and specifying their deployment specifics. The business rule deployment phase is shown in Figure 4.

After using the DeploymentEditor to retrieve newly created or edited business rules from the RuleRepository, the Rule Deployer assigns them to a RuleConfigSet. This contains a package of business rules to be deployed as a group for which common implementation-specific properties are specified. Properties such as the preferred method of inferencing (forward or backward chaining) and the conflict resolution strategy may be specified.

The Rule Deployer obtains a list of registered EngineDeployers to deploy the set of business rules. There's at least one EngineDeployer for every type of rule engine. The EngineDeployer translates the logical business rules, specified in the business rule language, into the rule syntax of the specific rule engine. This creates an executable representation of the rules. The deployment process allows a business rule to be included in multiple RuleConfigSets. It also allows a RuleConfigSet to be deployed to multiple third-party rule engines.

For example, assume the Rule Deployer retrieves the mobile phone promotional rule along with a set of related business rules that define the characteristics of a specific mobile phone service package. The Rule Deployer groups these rules into a RuleConfigSet and specifies the required implementation characteristics. To deploy the business rules using ILOG's JRules rule engine, the Rule Deployer specifies "JRules" as the EngineDeployer. A corresponding JRulesEngineDeployer is registered with the DeploymentManager so that the JrulesEngineDeployer then parses the set of rules from the RuleConfigSet and generates the rules in the JRules rule language syntax. The rules are now ready for runtime invocation by any application.

There are two possible scenarios for the business rule deployment phase. In the first, the Rule Developer is able to define and edit business rules dynamically at execution time. In this scenario the rule deployment process may be completely automated so that when rules are

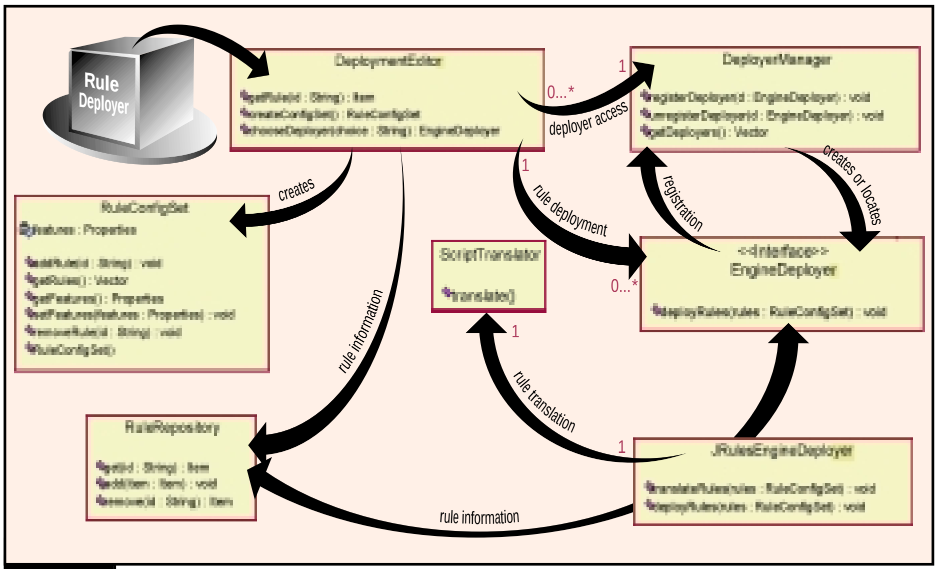


FIGURE 4 Class diagram of business rule deployment

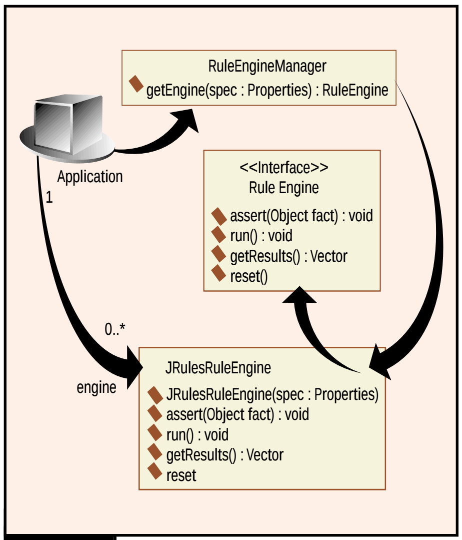


FIGURE 5 Class diagram of business rule implementation

KL Group Inc.

www.klgroup.com

deployed at execution time, implementation assumptions are built into the application to invoke the appropriate EngineDeployer and translate the business rules into executable rules.

In the second scenario the business rules aren't changed dynamically in the executing systems, but rather undergo the same migration procedures as other software. In this case the deployment process involves a human Rule Deployer as previously described. The rule deployment process often includes some form of rule correctness and consistency checking as well as unit and system integration testing. Since they can't be tested prior to implementation, rules defined and modified dynamically are usually restricted in scope to avoid unexpected application behavior.

Business Rule Execution

To invoke the business rules for processing, the Application requests a RuleEngine instance from the RuleEngineManager specifying the type of RuleEngine and the configuration set ID. The execution phase is shown in Figure 5.

The RuleEngineManager returns a valid instance of the RuleEngine (e.g., "JRulesRuleEngine") and the Application then asserts the appropriate EJBs to the RuleEngine and invokes the "run()" method to evaluate the rules. The Application obtains the results of rule evaluation using the "getResults()" method. To reuse this instance of the RuleEngine, the Application first calls the "reset()" method to clear the RuleEngine's working memory and follows the same process with a new set of asserted EJBs.

The RuleEngineManager isolates the application from the implementation details of the specific third-party RuleEngine so that the application isn't tied to a particular vendor's rule engine API – creating a simple, effective interface. The RuleEngineManager hides the details of the location and the number and types of RuleEngines available to the application. The RuleEngine API can be extended to start up and pool a predetermined number of RuleEngine instances and return instances based on application demand. The API can also be extended to act as a queueing "agent" that delivers application requests to an event-based messaging system and forwards results back from listening RuleEngine instances.

Next Steps . . .

The ultimate vision is the inclusion of a JRC (Java Rule Connectivity) API and a standard vendor-neutral rule language that would permit integrating rule engines into J2EE applications. Meanwhile, the Connector architecture may facilitate the evolution and adoption of a rule engine integration standard for J2EE. The J2EE Connector addresses the problem of providing a standard architecture for integrating heterogeneous EISs. Most EIS vendors and application server vendors use non-standard vendor-specific architectures to provide connectivity between application servers and enterprise information systems. The J2EE Connector architecture provides a Java solution to the problem of connectivity between the many application servers and EISs already in existence.

The J2EE platform enables the development of highly available, secure, reliable and scalable enterprise applications while reducing the cost and complexity of developing them. As a reusable service in the J2EE environment, rule engines would take Java applications to the next level, enabling the development of adaptable, maintainable enterprise applications.

Author Bios

Colleen McClintock, product manager for JRules at ILOG, Inc., has more than 15 years of industry experience, specializing in planning, design and implementation of rule-based systems and the integration of rules and objects.

Chris Roberts is a senior Java architect at Sun Microsystems.

cmcclintock@ilog.com

chris.roberts@east.sun.com

IBM

www.ibm.com/software/soul/devcode

Macro

www.macromed

omedia

lia.com/ultradev

Separating Presentation from Business Logic

Achieve faster server-side changes to Web page content using Java Servlets and JSP

WRITTEN BY
BARRY TAIT

Separating presentation and logic when building server-side Web-based applications allows you to generate Web pages with dynamic content easier and faster. It also enables Web designers who aren't especially experienced in application development to easily change the appearance of a Web page. For Web sites with information content that needs to change frequently, this advantage means that the update cycle can occur much more rapidly…thus bringing new information to Web site visitors faster.

Early Web-based applications were simple and usually contained both presentation and business logic, which created maintenance problems when changes occurred to either. Separating the two simplifies maintenance and allows faster and easier updates. This article will discuss two underlying technologies that can be used for presentation and logic separation: Java Servlets and JavaServer Pages (JSP). A simple application architecture in the context of a small demonstration will show you how to achieve this separation and deploy and change Web applications faster.

As the first step let me give a brief overview of Java Servlets and JSP – and of how they work together in a Web application architecture.

What's a Servlet?

Servlets are platform-independent 100% Pure Java server-side modules that fit seamlessly into the framework of an application server. They can be used to extend the capabilities of the server in a variety of ways with minimal overhead, maintenance and support. Because servlets are Java bytecode that can be downloaded or shipped across the network, they are truly "Write Once, Run Anywhere." Unlike CGI scripts, servlets involve no platform-specific consideration or modifications: they're Java application components that are downloaded, on demand, to the part of the system that needs them.

What's a JavaServer Page?

JSP is the Java platform technology for building applications containing dynamic Web content such as HTML, DHTML or XML. A JSP page is a text-based document containing static HTML and dynamic actions that describe how to process a response to the client. At development time, JSPs are very different from servlets. However, they are precompiled into servlets at runtime and executed by a JSP engine, which is installed on a Web-enabled application server such as WebSphere V3.0.

Servlets and JSP Working Together

It's possible for a servlet to take an HTTP request from a Web browser, generate the request dynamically (possibly querying back-end systems to fulfill the request) and then send a response containing an HTML or XML document to the browser.

The drawback of this approach is that the creation of the page must be handled in the Java Servlet, which means that if Web page designers wanted to change the appearance of the page, they'd have to edit and recompile the servlet. With this approach, generating pages with dynamic content still requires some application development experience. Clearly the servlet request-handling logic needs to be separated from the page presentation.

The solution is to adopt the Model-View-Controller (MVC) paradigm for building user interfaces. With MVC, the back-end system is your Model, the

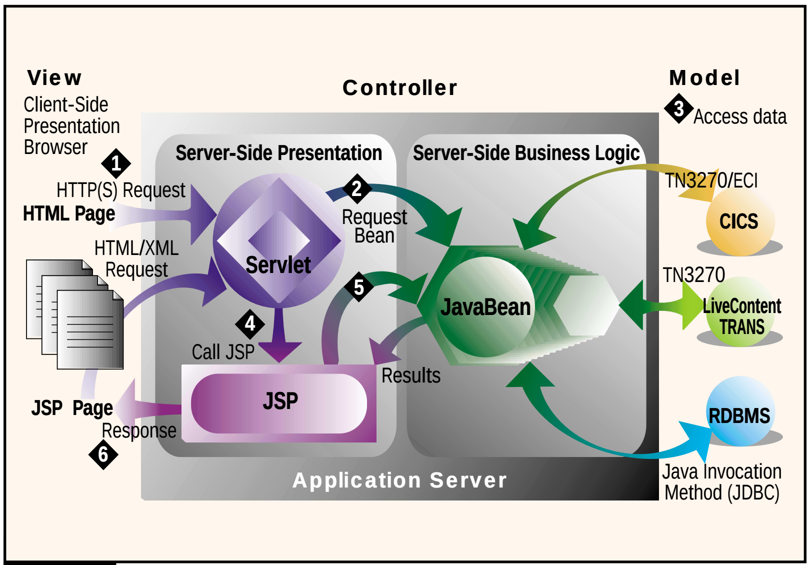


FIGURE 1 Architecture of demo application model

Sybase Inc.

www.sybase.com/products/easerver

templates for creating the look and feel of the response is the View and the code that glues it all together is the Controller. JSPs fit perfectly into this solution as a way of creating a dynamic response or View. Servlets containing logic for managing requests act as the Controller, while your existing business rules are the Model.

Let's look at a small demo application that utilizes the proposed MVC paradigm for presentation and logic separation. The demo provides browser-based access to a 3270 back-end system.

The architecture of the demo application follows the MVC design (see Figure 1). A browser using HTML and JSP pages provides the View. A set of Java Servlets and JSPs in an application server provide the Control, while the back-end CICS or PeerLogic applications provide the business rules or the Model. The flow between these servlets, HTML and JSP pages is shown in Figure 2 and described in the text that follows. (The source code for the Login Servlet and Login JSP files can be found on the ***JDJ*** Web site at www.javadevelopersjournal.com.)

Demo Application Flow Explained

THE REQUEST

The user will typically start with a page of HTML running within a browser. This page will be served from a Web-enabled application server. The application server serves files the same way a standard Web server such as Apache does. This page is an entry point into the Web application. It uses the HTML FORM tag to access the servlet – in our case, LoginServlet. Additional parameters can also be sent to the servlet within the bounds of the FORM action. An example of this would be as follows:

```
<FORM
action="http://localhost:8080/servlet
/LoginServlet" method="GET">
<input type="text" size="30"
name="firstname">
<input type="text" size="30"
name="surname">
.
.
<input type=hidden name=host
value="localhost">
<input type=hidden name=port
value="9876">
<input type=submit value=" Submit ">
</FORM>
```

The request will most likely be made using the HTTP or securely using HTTPS. This incoming request is then handled by the servlet, which uses the HttpServletRequest, getParameter() method to gain access to the FORM variables (see Listing 1).

The servlet then tests to see if the user has a session. A session is used to associate a set of unique requests from a remote client; this is necessary due to the stateless nature of the HTTP protocol.

A session is created as follows:

```
HttpSession session = req.getSes-
sion(true);
```

Objects can then be added to the session:

```
session.putValue("host", hostToCon-
nectTo);
session.putValue("port", tmpPort);
}
}
```

THE CONTROLLER

Servlets are responsible for invoking JavaBeans, which process the user's request. They are also responsible for creating the response to the user. In this proposed architecture it is done by passing the response to a JavaServer Page. For this reason (in object-oriented terms) the servlet is referred to as the *controller.*

Back to our example. The following code instantiates a new class of the type CICSEmulator called newEmulator. This is a vendor-supplied class from PeerLogic, Inc. CICSEmulator is a 3270 terminal emulator that exposes a set of methods for direct emulator manipulation. The newEmulator can be passed variables from the session, and in this case connected to the host and port of the remote system.

```
CICSEmulator newEmulator = new CICS-
Emulator();
newEmulator.setTN3270Port(portToCon-
nectTo);
newEmulator.setTN3270Host(hostToCon-
nectTo);
```

The second class instantiated is a JavaBean called AcctDetails. This bean was created using the LiveContent PATH 3270 tool from PeerLogic, Inc. AcctDetails is a data access bean; it performs the actual navigation and data retrieval from the back-end 3270 system. AcctDetails assigns the newEmulator class to be its 3270 terminal, then passes the input variables, surname and firstname to its *set* methods.

```
AcctDetails getAccountDetails = new
AcctDetails();
getAccountDetails.set3270Emulator(new
Emulator);

getAccountDetails.setSurname(input-
surname);
getAccountDetails.setFirstName(input-
firstname);
```

EXECUTING THE REQUEST USING THE DATA ACCESS BEANS

The performWork() method of the instantiated bean, getAccountDetails,

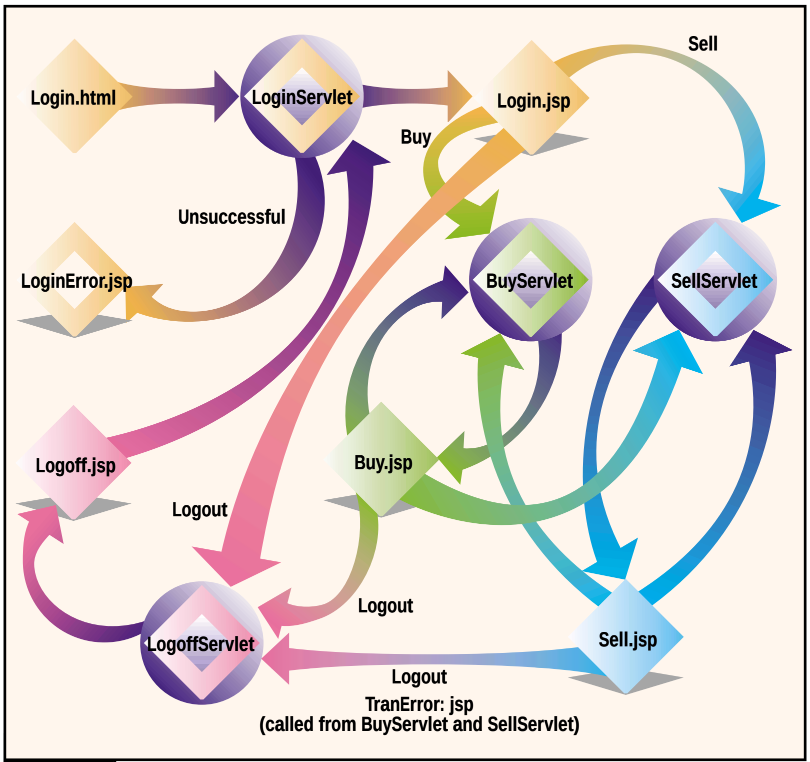


FIGURE 2 The flow of servlets, HTML and JSPs for the demo application

Zucotto

www.zucotto.com

connects to the back-end 3270 system and retrieves the account details based on the surname and first name that the user supplied:

```
 try {
getAccountDetails.performWork();
session.putValue("resultsBean",
getAccountDetails);
 }

 catch ( IllegalStateException e ) {
   // handle the error
 }
```

If successful, the servlet places the getAccountDetails bean into the user's session. This bean is given an identifier of resultsBean, as it now contains the results of the user's query. These results are then accessible through the bean's *get* methods.

PASSING CONTROL TO THE JSP

As mentioned, an important part of the Model-View-Controller approach is that presentation be kept separate from business logic. As you've seen, the servlet is responsible for processing the request using the data access beans. The data access bean, getAccountDetails, contains server-side logic for accessing and retrieving data from the back-end system. The back-end system itself (or the Model), contains all of the business rules. The response (or View) the user sees is created using JSP technology. JSPs are dynamic HTML pages containing small pieces of embedded Java code.

ABOUT THE DEMO APPLICATION

The demo application uses the JavaWebserver V2.0 application server. This application server was used due to its support for the JSP V1.1 specification. Development for the demo was done using the Java Servlet Development Kit Version 2.1, the LiveContent PATH 3270/Connect and LiveContent PATH 3270 products from PeerLogic, Inc., and Allaire Homesite for JSP authoring.

The servlet passes both the request and response objects to the JSP using the following syntax:

```
getServletContext().getRequestDis-
patcher("/jspDemo/login.jsp").for-
ward(req, res);
```

The *forward* method allows the servlet to pass the processing of the response to another resource. Its request and response parameters must be the same objects that were passed to the calling servlet's *service* method.

It uses the RequestDispatcher object obtained via getRequestDispatcher() to determine the path to the JSP target resource. The pathname to the JSP must begin with a "/" and is interpreted as relative to the current context root.

ACCESSING THE RESULTS BEAN

The JSP is responsible for creating the response for the user, which it does by gaining access to the getAccountDetails bean data (referred to by the resultsBean ID). This is done using the following syntax within the HTML page:

```
<HTML>
<BODY>
.
.
<jsp:useBean id="resultsBean"
scope="session" class="acct.AcctDe-
tails " />
<jsp:setProperty name="resultsBean"
property="*"/>
```

The <jsp:useBean> action tries to find an existing object (in this case the resultsBean) using the ID and scope. In our example, we placed the getAccountDetails bean into the session and named it resultsBean. The action will therefore get the resultsBean from the user's session. Once this is done, the JSP can access any of the resultBean's *get* methods that gain access to its data. This is done using the following JSP syntax:

```
<%= resultsBean.getDetails_Title() %>
<%= resultsBean.getDetails_Initial()
%>
```

This syntax is an example of a JSP expression. Anything between the <%= and %> markers is evaluated by the JSP engine, with the results sent as output to the JSP file. The two expressions above will execute the ResultsBean *get* methods for displaying the user's title and middle initial – exciting stuff! JSP code can be embedded throughout an HTML page the same way a scripting language such as JavaScript would. JSP syntax is relatively simple to understand. This means that the JSPs can be maintained by Web page designers rather than by the application programmers, who are responsible for the servlets and data access beans. Changes can be made to a JSP without affecting the servlet and vice versa.

THE RESPONSE

The JSP is dynamically compiled into a Java Servlet at request time and cached by the application server. The advantage of this is that any subsequent request will be extremely quick. The response the user receives is basically an HTML page (with a .jsp extension) that contains dynamically generated content.

Summary

In the demo architecture I've described in this article, the client makes a request from a Web browser directly to a servlet, which then processes the request by using data access JavaBeans to retrieve data from the back-end system. The servlet wraps the results into a results bean, places it into the session, then invokes a JSP to handle the response. The servlet is in complete control of the initiated request up to the time the response is generated.

It's up to the called JSP to generate the content needed for the response to the user. The JSP should include only logic that's needed for formatting the presentation. The advantage of this type of separation is that it creates more reusable, portable platform-independent components that can be shared between applications and implemented as part of a larger application.

Separating the actual development of the servlet logic and JSP presentation means that Web page designers and application developers can work independently of each other. This development approach conforms to the MVC paradigm that was outlined in the beginning of this document.

marketing@peerlogic.com

AUTHOR BIO
Barry Tait, an experienced Java developer, is the Web architect for PeerLogic, Inc., a company offering software that facilitates Java access to legacy mainframe applications and simplifies mainframe data access. He specializes in Web architecture. Before working with Java, Barry worked on CICS development for IBM.

Listing 1

```
public class LoginServlet extends HttpServlet {

 public void doGet (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException

{
String           inputsurname = req.getParameter("surname");
String           inputfirstname = req.getParameter("firstname");
String           hostToConnectTo = req.getParameter("host");
String           tmpPort = req.getParameter("port");
int portToConnectTo = Integer.valueOf(tmpPort).intValue();
.
.
```

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

Quest Software

www.quest.com

JavaOne

EXCLUSIVE!

Interview with James Gosling

Creator of Java and Sun Fellow, Sun Microsystems, Inc.

BY JASON WESTRA & DAVE JOHNSON

James Gosling, the lead engineer and key architect behind the Java programming language and platform, has been involved in distributed computing since his arrival at Sun in 1984. His first project was the NeWSTM window system. Before joining Sun, he built a multiprocessor version of UNIX, the original Andrew window system and toolkit, and several compilers and mail systems. He also built the original UNIX "Emacs" and helped build a satellite data acquisition system.

Photos Copyright 2000: SYS-CON Media

Java Developer's Journal: James, what do you think about this JavaOne conference compared to the last five that you've attended?
James Gosling: I guess I have a hard time believing the exponent and the growth. It's been something like 35% year over year. I didn't think that there were this many cool people in the world, having so much fun!

I think the thing that's really dominated the show for me this year is that there are really people out there doing it, this stuff is out there, people are really building it. A few years ago, you know, we were saying, "Wouldn't it be cool to do telephones?" And last year we were saying, "Hmm, what's happening?" and this year it's like people have got trunkloads of telephones, trunkloads of pagers.

It's unfortunate being in the United States, where the infrastructure doesn't really support the cool stuff; maybe we should all move to Europe or Japan or someplace! But the stuff has been amazing, the fact that the SmartCard folks have been just going nuts. There's like a dozen manufacturers of the SmartCard and they're selling like a hundred million of them this year – all able to run these little cardlets, and with different levels of flexibility and power just doing some really entertaining things.

JDJ: *We've actually been interviewing people now for three days straight and some of the most interesting folks have been in the wireless and the mobile area. Do you think that's definitely been the theme for this Java conference?*
JG: It's certainly been our theme. I guess that one of the problems at these conferences is that there's no such thing as the theme, because there's just so much going on. Certainly all the enterprise stuff is a big thing, but the wireless stuff has been just going nuts, I mean there's all these people doing Java code inside the end point – for example, there were those demos this morning that the folks from Motorola did, and there are all these guys from companies like Samsung. It's just going nuts. But one of the things about the wireless world is that there's usually the other side of it, right? You can talk from one wireless thing to another, but often the most interesting thing is when the wireless gizmo is talking into the infrastructure. So even for what people think of as the "dumb" smart phones like the WAP phones and that, a lot of the infrastructure pieces of the WAP world is people using JavaServer Pages. JavaServer Pages, you know, they're not just for HTML anymore. People have been doing XML and WML, all kinds of stuff, straight out of JavaServer Pages.

JDJ: *Where are you focusing your efforts right now, what are you working on or what's the team around you working on at Sun?*
JG: What am I actually working on? Well, the project I am desperately trying to finish right now is this sort of document management thing for Sun Labs. I made the mistake a few months ago of saying, "So where do I find a copy of this?" and we have this 10-year-old filing system that's like bits of paper and there's an online file you grab and it's completely useless, so I've been dealing with all this stuff with JavaServer Pages, so you can actually find the freakin' tech reports and know where they are! Most of these tech reports are online somewhere, but who knows where? So that's what I've been spending the last month or so of hacking time doing. But actually I've been starting a project and doing stuff for developer tools. One of the things I've been interest-

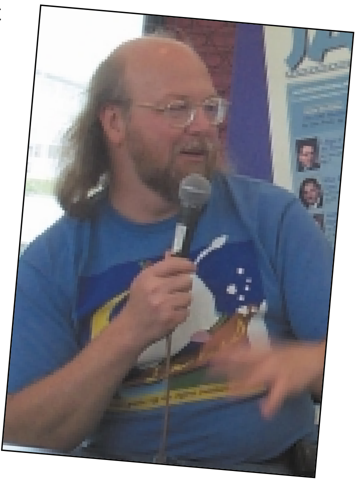
Gemstone

www.gemstone.com/welcome

ed in lately is, if you go around and look at the desks of most of the top-line developers, do you see them using any of these fancy IDEs? No. Mostly it's like Emacs, VR, Shell script...and it just goes, right? And you ask them why and it's

like, "Well, I did an IDE once and found that the layout was too simple when I actually wanted to write a program." Most of these IDEs actually try to prevent you from writing a program; they try to make it easy for people who don't know how to write programs. Near as I can tell, no one is building tools for people who actually write programs. And I'm sorry, making my keywords boldface and red doesn't count. Having a certain responsibility for Emacs, and Emacs is this thing that's like 25 years old, I find it kind of amazing that 25 years later the state of the art for folks like me hasn't gotten any better. So now I'm spending time building stuff for me, not for anybody else. I figure that if I can build a tool that I find actually useful, then I'll count myself lucky...

JDJ: ***...and the rest of the generation will probably use it as well.***
JG: Well, maybe. I mean one of the things that got me thinking about this is that I did a tool about 10 years ago. We had this little crisis in the graphics world where we had to write a whole new set of graphics libraries for an odd sort of reason. I don't know how many of you have actually written graphics algorithms, but you pull open your average graphics textbook and they have a bunch of standard algorithms. Probably the first one that you see is Bresenham's [line drawing] algorithm. It's really simple. You write it and it's like five lines of code, real easy and straightforward, and you can look at it and understand why it works and it looks like it's correct. But then you go to work for somebody that actually does graphics and you find out, where is the drawing routine? It's not five lines, it's not even 5,000 lines. It's like 50,000 lines, and it's doing this special case, that special case, horizontal lines, vertical lines...alphablending, trying to deal with one-bit frame buffers, 8-bit frame buffers, 15-bit, 16-bit, 32-bit...all of these different bit depths and this interleaves to all these issues like crash lines, and you find that you've got a gazillion different line drawing routines and you have to figure out which one to call. And a lot of the complexity is figuring out which are the optimized ones. And yet, at the core, they're all Bresenham's algorithm, and so I built this tool that was sort of a combination macropreprocessor and theorem prover, one where you could define rules to do tree replacements. Where you could say if you see this, then replace it with that – it was based on the



semantic graph, not on text strings, and if you were clever about the way the rules were built, you could actually turn it into a theorem prover. And it had basic primitives, like my favorite one was one called trade-off. Trade-off of A,B takes two code fragments presumed to compute the same thing, and it did a space estimate and a time estimate for each of them. All the code was annotated with branch probabilities and they would pick the one that was most appropriate right there. The whole goal was to be able to write the algorithm really simply and then write what were transformations on algorithms. And so if you believe the transformations, and you believe the algorithm, then you could believe that the end result was correct. Because a lot of the problem in writing these things is that you've got a thousand clones of exactly the same piece of code – kind of the same but doing it slightly differently. So this particular tool was trying to deal with that, and I thought it was really successful work. But the problem was that I managed to explain it to no more than about five people! It was a little bit too weird.

JDJ: ***Well, you've managed to explain it to a lot more than five people now. Are you going to be releasing this for open source? Or, if not, what's your view on open source right now?***
JG: Well, I don't know that I'd really want to release *that* one. Somebody did ask me for the source code a while ago and I actually did manage to find a version that would compile and run, but documentation was not real obvious and it's not exactly an obvious thing but I'm trying to build a new one. What do I think about open source? I basically like open source a lot. I mean the Java source code has been out there from the very beginning; it's good for a lot of things. One of the unfortunate things in the open source movement, though, is that they get into these really weird religious wars about, you know, "You're not open source.... I am." "No, no, he's not open source..." – there's like half a dozen different open source licenses that call themselves the one true open source and my view is like, get over it, guys. We [at Sun] get criticized because our open soiurce license doesn't work exactly like the [BSE] open source license or the Mazillo open source license or whichever, take your pick. But ours doesn't look like those because

there's something that we actually care about and the thing that we care about is that if somebody says, "This is a Java runtime," then people who plug programs into that, we think, have a right to know that they're going to run. In a community of people who are all sort of well-meaning and it's all sort of peace, love and happiness, you don't really need licenses because you can trust everybody; but in a world where there exist untrustworthy people – and certainly you have the example of one really *seriously* untrustworthy...and we ended up in court about it...[they were trying] to turn it into something that was incompatible with everything else and really frightening the community and it was and continues to be really important for us to make sure that, if somebody calls something a Java compiler or a Java runtime, that it actually runs Java programs from other people's compilers and that run on other people's runtimes, and our license merely says that we care about that. Beyond that, we're pretty close to being as free and open as you can be.

JDJ: ***I think the Java Community Process is testament to that. Have you had a chance to see any of the devices here, I'm talking about wireless and so on, there's a little PDA for example that's got a built-in wireless connection, have you see any cool gadgets like that?***
JG: Well, this place is like wall-to-wall cool gadgets! I don't know which is my favorite, but the one that turned my head sideways the most is this little cell phone that Samsung in Korea have, a tiny little thing in between the

—continued on page 142

Compuware

www.compuware.com/numega

Programming in the Small

Java can help make applications embedded in small devices much more efficient

WRITTEN BY
Bruce Scott &
Jeff Richey

Today's mobile Internet economy has opened the door to a range of new technologies that challenge traditional views of programming.

In particular, new devices are cropping up every day to meet the needs of both business and home users who regularly conduct business via laptops, PDAs and diverse Internet appliances. Visionaries and savvy product developers are also hitting the market constantly with new ways to package functionality into highly focused devices such as Web/cable set-top boxes or even car navigation systems – and telecommunications hubs and switches.

One of the most conspicuous characteristics of these small devices is how restricted in size they are compared to conventional servers and desktop PCs. Yet despite their size limitations, they must still fit a huge amount of functionality into a very small space, running the same types of applications and products that currently run on more conventional platforms. In addition, the applications and embedded databases at the heart of small-space devices need to perform at an acceptable speed. To make matters even more complicated, each device presents its own particular set of requirements in terms of memory constraints, UI capabilities and breadth of functionality. Whatever their purpose, to appeal to consumers and business purchasers in target markets, small devices of this type need to be self-managing and must require little administration.

Java Ideal for Small-Space Applications

The Java programming environment is a natural for applications confined to small devices. For one thing, Java's "write once, run everywhere" capabilities allow programmers to create code that can be used on a range of small-space platforms. Programmers using Java also have the opportunity to leverage classes and the Java factory concept and class loader capabilities to address the specific problems of small environments. Other useful Java features include interface definitions and recursion, which can improve productivity at programming time and enhance the efficiency of resulting applications.

Modular Design Key to Success

The key to creating applications that won't strain the resources of small devices is intelligent software design. The best approach is to start by viewing the software as a modular set of programs rather than a single vast piece of code. Optimal efficiency requires that functionality be segmented according to the manner in which it will be executed at runtime. In this way mechanisms such as classes and Java factories can do their jobs effectively. The Java class loader is a case in point. This feature allows mobile devices to load functionality into memory selectively, pulling only those classes required for specific operations and resulting in effective processing that meets all specific functionality requirements by using onboard memory – "heap space" – efficiently. The effectiveness of the class loader, however, is entirely dependent on whether the base architecture and the Java classes have been designed intelligently. In fact, Java programmers can still choose to write monolithic programs in the "C" tradition.

Java does present a few issues with regard to small-space programming, primarily as a result of the dearth of third-party tools. As with all new platforms, it takes some time for tools such as these to mature. Current issues include how to measure memory usage and how to cope with the high overhead associated with Java objects. Fortunately, a couple of useful workarounds can fill the gap until reliable tools become available.

Leveraging Java Classes

Java classes embody application functionality in logical chunks, potentially in very small sizes. The smaller the discrete piece of functionality, the easier it is to isolate functionality at runtime, thus ensuring efficient memory usage in a small environment. Keep in mind that it's possible to have too much of a good thing: if there are huge numbers of tiny classes, then the resulting overhead could negate the benefits of such granular modularization. Experience helps in balancing the size of classes against the associated overhead.

The use of subclasses also affects the application footprint. From a simplified perspective, if there is a Class A, then a superset called Class B can be defined as including all of the functionality of Class A plus additional functionality required for a particular situation or platform. Class C could be created as a further superset of Class B. In addition, there can be any number of branches within each subclass. By packaging application functionality in this intelligent manner, programmers can go a long way toward maximizing code reuse and eliminating redundancy, creating modularized software that can execute efficiently at runtime.

Buzzeo

www.buzzeo.com

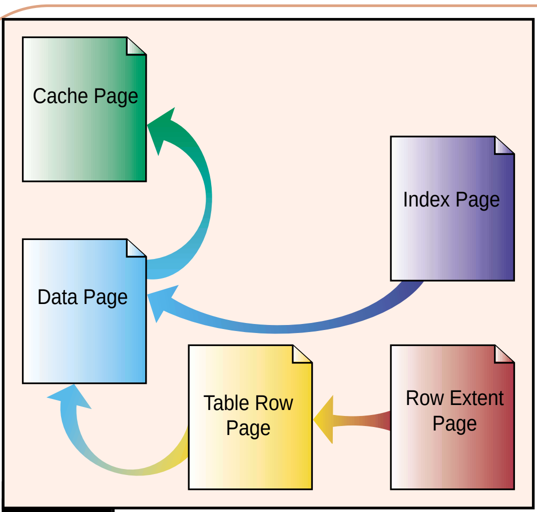


FIGURE 1 Cache page class hierarchy

A Small-Footprint Example

When basic information also needs to be included in index pages, table pages and control pages (all associated with a particular database), what we do at PointBase is to use cache pages (see Figure 1).

The class hierarchy we adopt can be summarized as follows:

1. The *cache page class* is the base class for all pages in a database.
2. The *data page* extends the class cache page to add functionality specific to data pages, such as forward and backward pointers.
3. The *table row page* extends the data page to add functionality specific to pages used to store rows. This includes information such as the number of rows on the page, the location of each field on the page, and the amount and location of free space on the page.
4. The *row extent page* extends the table row page as a special form of the table row page used for overflow information.
5. The *index page* extends the data page. Note that both the index page and table row page extend the data page. These pages have dramatically different uses, but by extending the data page they can share all the code that maintains the forward and backward pointers.

This is a simplified example – our full cache page hierarchy is much more complex than the one described here. Keep in mind that the choice of which classes to extend – and which of those subclasses to further extend – is made based on the way functionality was defined and segmented at the design stage. For maximum efficiency the designer should make sure that the class hierarchy accurately models the problem being solved and should try to reuse as much code as possible.

Class Loader: Traffic Cop

Java class loaders are the answer to "C" programs that need to load the entire executable (i.e., the .exe file) into memory at runtime. The beauty of the Java class loader is that it supplies only the classes required for the functionality requested by the application, and does so automatically. If the software has been designed properly, Java class loaders make it possible for users to run multiple applications and use a broad range of functionality within diverse applications without running into a memory wall, a critical issue for small devices that don't have a lot of room for extra baggage.

Optimizing the Factory Principle

Although the Java class loader helps preserve memory by loading functionality exactly as it's needed – providing great benefits from the point of view of memory usage – this holds true only for memory use during runtime. Persistent storage space isn't preserved. So even if only 100K of a 10MB Java program is loaded into memory, it will still take up 10MB of persistent storage despite the use of the Java class loader principle. The reason for this is that, when creating a JAR file for a set of Java classes, any class referenced by an "import" statement will be included automatically. In our example this means that all 10MB of data will be included in the relevant JAR file and downloaded, even though only 100K of functionality is needed. Fortunately, if the Java program has been written in a modular fashion, this problem can be solved through a technique many Java programmers call *factories.*

Factories aren't really a Java language feature or construct, but rather are a particular way of using a combination of Java classes to create other Java objects indirectly. To put it another way, a factory is a Java object whose sole purpose is to create other Java objects (it is, literally, a "factory" for producing Java objects). A factory may create and return objects from several classes, but each of those classes must implement the same Java interface.

Using factories with a modular design allows a Java program to be broken up into optional pieces. JAR files can then be built with certain pieces removed, thus producing a smaller JAR file.

FACTORIES AND SQL FUNCTIONALITY

Factories can be used to define SQL functionality – such as Data Definition Language (DDL), Data Manipulation Language (DML: Delete, Insert and Update), Query (Select), Security, Business Logic (triggers and routines) and other statements. All of these can be made into factories, and a good reason for using factories for this purpose is that most applications don't require all of this DBMS functionality. Dividing the functionality into factories allows the application to use only what it actually needs.

FACTORIES AND SQL SELECT STATEMENT

As a matter of fact, the factory principle can even be used within individual SQL statements. The SQL Select statement is made up of several objects: parsing, dictionary, optimization, plan generation and execution. With the use of factories, an application needs only the execution factory object at runtime (i.e., the application's JAR file would only use the SQL Select statement's execution factory object).

FACTORIES AND SQL UPDATE STATEMENT

Another view of factories emerges from considering what can be done with the SQL Update statement. There's a set of classes that know how to compile and execute the SQL Update statement. If the main UPDATE class is seen in an import statement, then all of the classes used to compile and execute the SQL Update statement will usually be included in the JAR file. However, an UPDATE factory that has the specific job of creating the UPDATE object can be created. The UPDATE factory class can be written to determine if the UPDATE class is available in the JAR file. If it is, the UPDATE factory can return the UPDATE object so that SQL Update statements can be compiled and executed as expected. On the other hand, if the UPDATE class isn't present, the factory can generate an exception or return a special version of the UPDATE object – namely, an object whose only job is to generate an error. Another possibility is that the UPDATE factory could return an UPDATE object that's smaller and has less functionality.

Through this technique JAR files can be configured with less functionality and less Java code. The factories can be used to detect this so no strange errors about a class not being found will be generated.

Big Results in a Small Space

Used in conjunction with the Java class loader, after application functionality has been carefully packaged into appropriate classes, factories extend the benefits of modularity by providing tighter control over which sets of functionality will be included in the JAR file.

HotDispatch.com

www.hotdispatch.com

The result is a highly dynamic system that "breathes" with the end user, remaining slim whenever possible but stretching as needed to accommodate more comprehensive activities.

By organizing functionality in such a versatile and efficient manner, factories also contribute to the development of "ubiquitous" software that can be deployed in various permutations depending on the platforms involved. This means that the same application could be run on everything from a cell phone to an application server, with the full range of functionality available on the server end and appropriately focused pieces of functionality available at the handheld level.

Built-in Java Benefits for Small Footprint

Java also offers a range of smaller but significant features that contribute to the efficiency of applications embedded in small devices.

INTERFACES

One of these features is Java interfaces. These allow common streams of code to treat a variety of classes as identical. In the case of cache pages as described above, an interface definition could permit the cache manager to treat all pages in the same manner, even though other parts of the application might "see" and handle them differently. This keeps the cache manager code simple and consistent for all pages, while also allowing for specialized code in other instances (such as the b-tree index manager).

OPERATOR

The Java instance of operator can be used to deviate from the general code to handle specific classes. This would occur when the code handles a set of classes that all support a particular Java interface but there's a need for some class-specific code as well.

RECURSION

Recursion is another helpful Java player; it allows a particular method to call itself over and over to any reasonable number of nestings, eliminating the need for complex loops and special test conditions. This technique is especially useful for data structures and concepts that are naturally hierarchical – for example, the computer file directory. A directory is a list of files. Some of these files are also directories, meaning that directories can contain other directories, which can contain yet other directories. This kind of structure fits very well into a recursive method. A recursive algorithm might do things such as list the fully exploded contents of a directory or search for a specific file in a directory along with all of its subdirectories.

The SQL language has many examples of recursion. For example, a WHERE clause in a SQL Select statement can contain other SQL Select statements, which in turn can contain a WHERE clause, and so on. It's because the SQL language is recursive that we at PointBase use a recursive descent parser to parse SQL statements. The same is also true of SQL expressions.

Measuring Memory Usage and Performance

In my experience there are two areas in which Java presents special challenges for the development of applications with a small footprint: measuring memory usage and performance, and coping with the overhead of Java objects. (At PointBase, luckily, we've come up with a few workarounds that can keep these problems under control until third-party tools become available to cope with them in a more exact way.)

Memory usage is a central issue for developers of small-space applications, since every byte of memory is precious in these environments. The best-case scenario would be one in which programmers could associate memory usage with specific pieces of code. Systems vendors such as Sun Microsystems, Microsoft and IBM help out to some extent by supplying profilers that contain "hooks" into their respective JVMs. Measurements generated by these profilers reflect memory usage fairly accurately for applications as a whole, which is at least a first step toward understanding how memory will be affected by a particular set of programs. However, it's still difficult to pinpoint which specific pieces of code are at fault when memory usage is unacceptably high. Some third-party tools purport to drill down to the code level, but we've found that these tools give conflicting reports, making them appear somewhat unreliable.

What can be done about the memory measurement problem? At PointBase we've become rather creative in approaching it by creating, in some test cases, thousands of objects and then measuring the amount of memory at the macro level to determine the approximate overhead per object. The object creation-time overhead can also be determined in this way.

In the area of performance the goal in the Java environment is to determine how many times each class is called and how long it takes to execute. This allows programmers to identify and improve classes that are performing more slowly. Currently, neither systems vendors nor third-party developers provide reliable tools for accurately measuring performance in this way. We expect the situation to improve over time, but in the meantime fine-tuning based on the findings of OS profilers as described above offers the best workaround.

Solutions for High Object Overhead

Testing has shown that some JVMs associate 40 bytes with every object. Thus, if an object itself requires 8–10 bytes, its overall size would be 48–50 bytes, so memory usage can grow unexpectedly large very quickly. There are situations in which a collection of diverse objects is expected to be relatively permanent (say, in a complex table). In cases like these it's possible to save a substantial amount of memory by either consolidating objects or adopting a C-style approach to coding, such as using arrays. The C-style section of code can be "wrapped" in an object, making it accessible to the rest of the application and therefore maintaining the object-oriented paradigm. In general, relatively permanent objects don't work well for small environments; it's better to use objects and then discard them.

Summary

Java is perfectly suited to the demands of the small application environment, providing exceptionally efficient and productive ways to accommodate handheld devices, new Internet appliances or any other function-specific hardware. Modular software design and the intelligent use of Java classes, factories and the Java class loader – with an extra boost from many of Java's basic programming features – help ensure that the runtime footprint remains small without interfering with the functionality and performance sophisticated end users require. Java has taken on the challenges of "ubiquitous" programming and all of the promise of yet-to-be-conceived devices that require pared-down but nevertheless powerful applications and embedded databases. This segment of the industry is still fairly new, but it's growing rapidly; current limitations will no doubt be swept away as more and more tools become available and increasing numbers of people gain appropriate skills and knowledge, and as markets for products expand and multiply.

bruce.scott@pointbase.com jeff.richey@pointbase.com

AUTHOR BIOS

Bruce Scott is president, CEO and founder of PointBase, a leader in the area of enterprise and embedded database architecture and product development. A cofounder of Oracle in 1972, Bruce cofounded Gupta Technologies in 1984, pioneering the notion of the small-footprint database server for Intel-based platforms.

Jeff Richey, vice president of engineering and cofounder of PointBase, is a recognized leader in database product development. Jeff has over 15 years of database experience, working as a core architect and development manager for IBM/DB2, HP, Oracle and Sybase. He is a patent holder of two key innovations in SQL performance.

PointBase

www.pointbase.com/edj

Customizing JFileChooser

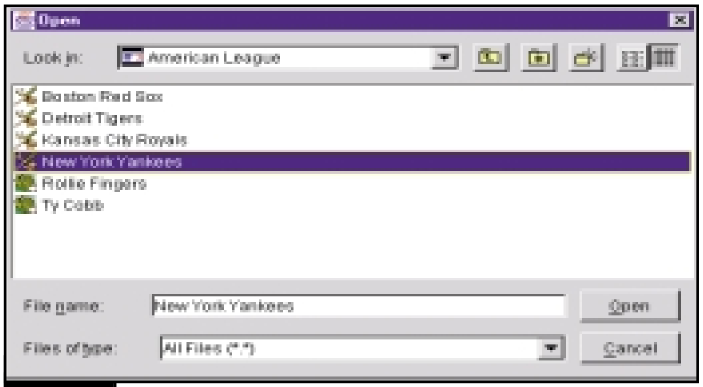
An adaptive approach for connecting to alternate data sources

WRITTEN BY JUSTIN HILL

Front-end architecture and the art of developing GUIs that are functional and intuitive have been a challenge in this industry for quite some time. The advent of Java has made things a little easier; Java's extensive Swing package has assisted in the rapid creation of GUIs for application development. While the package provides a vast array of robust graphical widgets, there comes a point when a developer will stretch the limits of the component by attempting to provide functionality that is either not supported or nonexistent. JFileChooser is an example of one such component. The purpose of this article is to provide the reader with the knowledge and understanding to extend JFileChooser's functionality in order to display information from a generic directory service.

InetSoft
Technology Group

www.inetsoftcorp.com



Look in: American League

Boston Red Sox
Detroit Tigers
Kansas City Royals
New York Yankees
Rollie Fingers
Ty Cobb

File name: New York Yankees

Files of type: All Files (*.*)

FIGURE 1 Extension to JFileChooser, connecting to a generic directory service

JFileChooser and FileSystemView

The default implementation of JFileChooser provides the user with a view of the local file system's directories and allows a user to select a file or multiple files. To provide this functionality, JFileChooser, along with its associated UI delegate, uses three helper classes located in the javax.swing.filechooser package: FileSystemView, FileView and File-Filter. To create a robust JFileChooser extension, an implementation should include modifications to all three classes, but as FileSystemView is the most crucial class, it deserves special mention.

FileSystemView is an abstract class that's the gateway to the file system. To create a file chooser that accesses an alternate file system, you'll need to provide an implementation of this class and pass it into the constructor of JFileChooser. When JFileChooser constructs itself, it checks to see if a FileSystemView has been provided; if so, it simply uses it. Otherwise, FileSystemView's getFileSystemView() method is invoked. This special method queries the local file system and determines the file separator ("/" in UNIX or "\" in Windows) in order to instantiate one of three default implementations provided in FileSystemView as inner classes.

This class doesn't actually access the local file system – or any file system, for that matter. Instead, FileSystemView directly interacts with File objects, which provide more direct access to the local file system. Note that some directory services may not represent themselves as a listing of File objects and therefore will need to be converted to representative File objects so as to enable interaction with FileSystemView and JFileChooser. We'll revisit this issue of converting generic directory service objects to File objects in a later section.

A Look Behind the Scenes

As previously mentioned, JFileChooser doesn't actually correspond with the hard disk of the computer, but instead delegates this responsibility to the File object. One of the first things that must happen to display the file chooser is the acquisition of the roots (in the case of the local file system, this would refer to A: through Z: on a Windows machine). FileSystemView's getRoots() and getHomeDirectory() are invoked to accomplish this task, and the results are placed in the file chooser's combo box. Once the roots have been obtained, the combo box's selected value is set to the home directory. Next, FileSystemView's getFiles() is invoked, and all the directories and files that are contained in the home directory are displayed in the list of the JFileChooser. This whole setup is performed for free, meaning that by simply instantiating JFileChooser this functionality is provided. As a result, it's imperative that we understand how and where this functionality is implemented to allow us to override it and add our directory service-specific features. Once the initial setup work is complete, the file chooser is ready to be displayed to the user.

As the user interacts with the file chooser, the bulk of the processing work is delegated to FileSystemView's getFiles() method. This method simply takes a File argument as its parameter (the file should be a directory), obtains the files that are in the directory and returns the results as an array of Files. Although a firmer understanding of FileSystemView is necessary, this is enough information to get us started on creating our own extension of JFileChooser.

Customizing JFileChooser

The extension to JFileChooser (see Figure 1) that I've created doesn't access the local file system. Instead, it connects to a generic directory service.

The concept of a directory service is an extension of a naming service in that it provides a mechanism for associating objects with a logical name within a searchable structure. A naming service simply maps names to objects. For example, the Domain Name System (DNS) maps people-friendly names (www.yahoo.com) to IP addresses (216.32.74.52). While a naming service allows the lookup of objects based on names, a directory service allows these objects to have elements (or attributes) associated with them. For example, in a directory service of employees, looking up the name Fred Jones would return an object that referred to Fred but also contained attributes like age, salary and job title. Directory services can exist in numerous environments:

- *Lightweight Directory Access Protocol* (LDAP)
- *CORBA services* (COS) – Naming services
- *Java Remote Method Invocation* (RMI) – Registry
- *Domain Name System* (DNS)
- *Enterprise JavaBean* (EJB) – Entity beans

The test directory service created for the purpose of this article emulates a real directory service containing a sampling of baseball teams and players. The contents of the directory service are actually a series of Java objects that are associated with each other via parent–children relationships (see Table 1).

Adapter

During the initial stages of creating this JFileChooser extension there were two hurdles that needed to be overcome:

1. The need to provide a way to allow for seamless communication between the directory service objects and JFileChooser's File objects
2. The need to provide an interface that would allow the JFileChooser extension to connect to additional directory services

The interface DirectoryService (see Listing 1) and its concrete implementation (DirectoryServiceSportsAdapter) help to overcome these hurdles. Not only do these classes provide the abstraction layer that allows JFileChooser to connect to alternate directory services without a major code rewrite, they also act as the necessary middle layer communication connection. JFileChooser deals directly with File objects, so the adapter has the ability to convert the directory service objects (whatever they might be) to File objects needed by the file chooser. Furthermore, the adapter allows for asynchronous communication that allows JFileChoos-

TYPE	NODE	PARENT	CHILDREN
ROOT	BASEBALL	NONE	AMERICAN LEAGUE, NATIONAL LEAGUE
LEAGUES	AMERICAN LEAGUE	BASEBALL	NEW YORK YANKEES
	NATIONAL LEAGUE	BASEBALL	CHICAGO CUBS
TEAMS	CHICAGO CUBS	NATIONAL LEAGUE	SAMMY SOSA, JOE GIRARDI
	NEW YORK YANKEES	AMERICAN LEAGUE	DEREK JETER, TINO MARTINEZ
PLAYERS	SAMMY SOSA	CHICAGO CUBS	NONE
	DEREK JETER	NEW YORK YANKEES	NONE
	JOE GIRARDI	CHICAGO CUBS	NONE
	TINO MARTINEZ	NEW YORK YANKEES	NONE

TABLE 1 A sampling of the test directory service hierarchy

Amazon.com

www.amazon.com

er to talk back to the directory service. This mode of communication is important because it allows JFileChooser to access the directory service so as to create new folders, find the root and so on.

DirectoryServiceSportsAdapter

DirectoryServiceSportsAdapter, a concrete implementation of DirectoryServiceAdapter, contains two important member variables: directorySession and HashMap. The former is a reference to the actual test directory service. One of the roles of the adapter is to provide a connection to the directory service and the directorySession variable fills this role. The latter variable is a java.util.HashMap that contains File objects as its keys and directory service-specific objects as its values. The storing of File objects and the directory service objects in the HashMap allows the relationship between the directory service and the file chooser to exist along with providing a local cache.

The concept of a local cache is important because it prevents having to constantly go back across the wire to get the object corresponding to the File. To guarantee a correct representation of the directory service, when the user selects a different directory, the directory's children are always obtained from the service. However, it's at this point that the directory service objects are cached, and from then on the cached copy is used to perform operations such as checking to see if the current filter allows the File, or obtaining the icon.

As mentioned earlier, the bulk of the work of populating JFileChooser is handled via FileSystemView's getFiles(). DirectoryServiceFileSystemView (see Listing 2) provides a specific implementation of FileSystemView by overriding the getFiles() method and delegating the functionality to DirectoryServiceAdapter's getChildren() method. The getChildren() performs the following:

```
Folder folder = (Folder)this.hashmap.get(aDir);
Node[] arrChildren = folder.getChildren();
for (int i=0; i<arrChildren.length; i++) {
 File f = new File(aDir, arrChildren[i].getName());
 this.hashmap.put(f, arrChildren[i]); }
```

If you want to use the extended file chooser widget and its adapter design to connect to an alternate directory service, you'll begin to see the benefits of the abstraction. The advantage of the adapter design pattern is now evident in that any customized directory service adapter implementation is pluggable into this framework.

DirectoryServiceFileChooser and the Helper Classes

DirectoryServiceFileChooser, a subclass of JFileChooser, acts as a factory for the file chooser creation. To provide a full implementation of JFileChooser and showcase all its features, I've created subclasses of the three helper classes (FileSystemView, FileView and FileFilter) named DirectoryServiceFileSystemView, DirectoryServiceFileView and DirectoryServiceFileFilter, respectively. The object-oriented concepts of abstraction and delegation dealing with the DirectoryServiceFileSystemView also pertain to DirectoryServiceFileView and DirectoryServiceFileFilter.

JFileChooser Is a JComponent

While customizing JFileChooser to provide interaction with directory services is the thrust of this article, the concept of embedding JFileChooser inside existing components is an interesting extension that should be covered. In most cases JFileChooser is displayed via one of its showDialog() methods and acts like a standard dialog. The following code snippet illustrates how JFileChooser is normally employed:

```
JFileChooser fc = new JFileChooser();
 int returnVal = fc.showDialog(null, "Open");
 if(returnVal == JFileChooser.APPROVE_OPTION)
   File file = fc.getSelectedFile();
```

Although JFileChooser seems to act like a JDialog, it's actually a JComponent. As a result, when the showDialog() method is invoked, the JFileChooser is created and added to a JDialog, which in turn is handled internally by the JFileChooser.

Since JFileChooser is a JComponent, it's possible to do some interesting things with it. For example, you can embed JFileChooser in your own widget simply by creating it and adding it to a specific container. The following code illustrates how to place a JFileChooser in a JFrame:

```
JFrame frame = new JFrame("File Chooser in a Frame");
frame.getContentPane().add(new JFileChooser(), "Center");
frame.pack();
frame.setVisible(true);
```

The Selector Dialog

The Selector Dialog (see Figure 2) is a dialog with an embedded JFileChooser component. The purpose of this dialog is to allow the user to populate the list on the right side of the component with entities from the left side (the file chooser). One use of this dialog – applying the baseball directory service example – could allow users to specify a list of teams they'd like to search. For example, suppose a person wanted to find all the players in baseball that make more than $5 million a year and play for certain teams. The user(s) could utilize the Selector Dialog to traverse the sports directory service (via the file chooser) and add only the teams they wanted to search.

While the left side of the Selector Dialog is simply created using the factory method of DirectoryServiceFileChooser, the remainder of the dialog was constructed to interact with the file chooser (see DirectoryServiceSelectorDialog, available on the ***JDJ*** Web site).

The three buttons in the middle of the dialog (>>, << and <<<) correspond to Add, Remove and Remove All, respectively, and perform the following functions:

- ***Add:*** Invokes DirectoryServiceFileChooser's getSelectedFiles() method and adds the results to the list. If the file already exists in the list, it is not re-added.
- ***Remove:*** Removes the selected file(s) from the list.
- ***Remove All:*** Clears the list of all files.

As you can see, the idea of embedding JFileChooser in another component can be both powerful and functional.

Future Enhancements

While DirectoryServiceFileChooser and its related classes provide a sturdy framework for creating file chooser graphical widgets that connect to generic directory services, there are three features that would provide extended functionality:

1. ***Ability to Connect to Multiple Directory Services***: In the current framework the DirectoryServiceFileChooser can connect to only one directory service at a time. However, a future enhancement could include the ability to have the file chooser widget display the contents of multiple directory services. This feature could be implemented using the Observer (Publish-Subscribe) design pattern whereby multiple directory services adapters could register themselves with DirectoryServiceFileChooser.

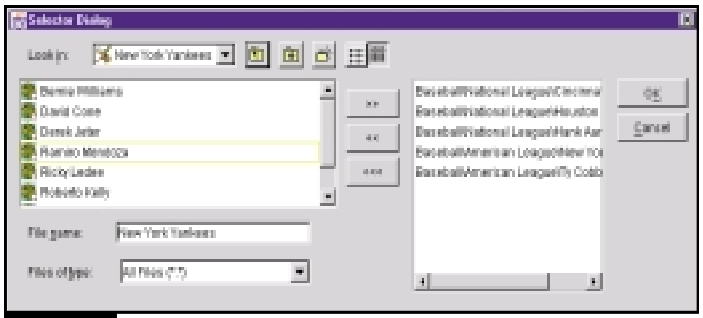

Selector Dialog
Bernie Williams
David Cone
Derek Jeter
Ramiro Mendoza
Ricky Ledee
Roberto Kelly
>>
<<
<<<
Baseball\National League\Cincinna
Baseball\National League\Houston
Baseball\National League\Hank Aa
Baseball\American League\New Yo
Baseball\American League\Ty Cobb
OK
Cancel
Look in: New York Yankees
File name: New York Yankees
Files of type: All Files (*.*)

FIGURE 2 Selector Dialog with embedded JFileChooser component

SlangSoft

www.slangsoft.com

2. ***Allowing for Runtime Connection to Directory Service:*** The ability to connect to a given directory service at runtime is nonexistent in the current framework. This task could nonetheless be accomplished through the use of Java Properties files. These files, which contain the parameters necessary for connecting to the directory service, could be read at runtime, thus allowing for dynamic connection to various directory services.

3. ***Multithreading the List of Files:*** The current implementation of DirectoryServiceFileChooser doesn't use the concept of threading to obtain the files of a directory for display in the file chooser. Unfortunately, when dealing with large directories of hundreds of files, the user experience can suffer if a user must wait for the file chooser to finish loading the files. However, if multiple threads are used to retrieve and display the files in a cursorlike fashion (i.e., 50 at a time), the user could continue to interact with the chooser as the files loaded in separate threads.

Conclusion

Customization of components is one of Swing's greatest advantages. The ability to customize JFileChooser in order to connect to alternate data sources makes it a more robust graphical widget. The concept of connecting to a generic directory service using an adapter provides DirectoryServiceFileChooser with a pluggable functionality. The end result is a framework that averts the need for refactoring the file chooser widget by decoupling it from the specific nature of the directory services.

Resources and Acknowledgments

1. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
2. Eckstein, R., Loy, M., and Wood, D. (1998). *Java Swing*. O'Reilly & Associates.
3. Swing: www.spindoczine.com/sbe/
4. The ASCltd team for their careful review and thoughtful suggestions.

AUTHOR BIO

Justin Hill, a Sun-certified Java programmer, works for Advanced Systems Consulting, a Java-centric consulting firm in Chicago. He is currently working on the Java front-end architecture for a distributed system at a major financial institution.

jhill@chicagojava.com

Listing 1

```
package filechooser;
import java.io.File;
import java.util.List;

public interface DirectoryService {
  public final static int TYPE_LEAGUE = 0;
  public final static int TYPE_TEAM = 1;
  public final static int TYPE_PLAYER = 2;
/**
 * Returns the children of the parent directory.
 */
  public File[] getChildren(File aDir);
/**
 * Returns the root (in this case, Baseball).
 */
  public File getRoot();
/**
 * Returns the type (for example, Player).
 */
  public int getType(File aFile);
/**
 * Returns whether the file is traversable.
 */
  public boolean isTraversable(File aFile);
/**
 * Creates a new folder in the containing directory.
 */
  public File createNewFolder(File aContainingDir);
/**
 * Determines if the file should be displayed
 * based on the current filter.
 */
  public boolean acceptFilter(File aFile,
                             String aCurrentFilter);
}
```

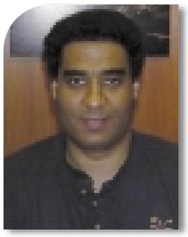
Listing 2

```
package filechooser;
import java.io.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.filechooser.*;
/**
 * This class provides a view into a generic
 * directory service.
 */
public class DirectoryServiceFileSystemView
                             extends FileSystemView {
  private DirectoryService directoryService;
  public DirectoryServiceFileSystemView(
                             DirectoryService
                             aDirectoryService){

    this.directoryService = aDirectoryService;
  }
/**
 * Creates a new folder with a default folder name.
 * This method calls into the currently opened
 * directory service to obtain the directory service
 * system specific way to create a new folder.
 */
  public File createNewFolder(File aContainingDir)
                               throws IOException {
    File f = directoryService.
                    createNewFolder(aContainingDir);
    return f;
  }
/**
 * Returns all root partitions on this system.
 */
  public File[] getRoots() {
    File[] arrFiles = new File[1];
    arrFiles[0] = directoryService.getRoot();
    return arrFiles;
  }
/**
 * Returns whether a file is hidden or not.
 * In the current system, there is no concept of
 * a hidden file.
 */
  public boolean isHiddenFile(File f) {
    return false;
  }
/**
 * Determines if the file is a root partition.
 */
  public boolean isRoot(File f) {
    File root = directoryService.getRoot();
    String rootName = root.getName();
    String fName = f.getName();
    return (rootName.equals(fName));
  }
/**
 * Returns the user's home directory.
 * The concept of a user's home directory
 * does not directly map to the Directory Service,
 * so, as a result, the root is always returned.
 */
  public File getHomeDirectory() {
    return directoryService.getRoot();
  }
/**
 * Gets the list of files. Invoked internally
 * by the file chooser when the directory
 * is changed.
 */
  public File[] getFiles(File dir,
                           boolean useFileHiding){
    File[] arrFiles =
        directoryService.getChildren(dir);
    return arrFiles;
  }
}
```

A Servlet-Based Customer Requirements Database

Harness customer feedback to help map your future



WRITTEN BY
DEAN WILLIAMS

The phrase *customer-driven* has become a mantra for many companies – asserting and embracing the central role played by customers in the daily life of today's corporations.

Notwithstanding the recent growth (and ebb) in IPOs and dot-coms, long-term business success still most often relies on the old basics, namely:

1. *Understanding what your customers really want and need*
2. *Delivering a solution to them at a reasonable price*

These two basics apply equally to software products, to tools and to services. In order to understand what your customers really want, it's often useful to actually solicit their input; accordingly, this article is about the fundamental importance of using your customers' feedback and suggestions to help prioritize your future development plans.

For many software companies, the simplest approach – that of having an e-mail "feedback" link on the company homepage – may be a perfectly adequate solution. Other companies may go a stage further and provide one or more newsgroups for soliciting and tracking customer suggestions. The third and most sophisticated approach is to use an application specifically designed to collect, track and prioritize customer feedback. Though a few commercial solutions exist, the main objective of this article is to show that, with a little effort, a fairly useful Web-based servlet solution can be implemented. More specifically (and to make this discussion a little more concrete), I'll be describing the "Feature Request Database" servlet used by IBMs VisualAge for Java development team to collect, track and prioritize customer input regarding future enhancements (see http://service5.boulder.ibm.com:8080/servlet/reqServlet3).

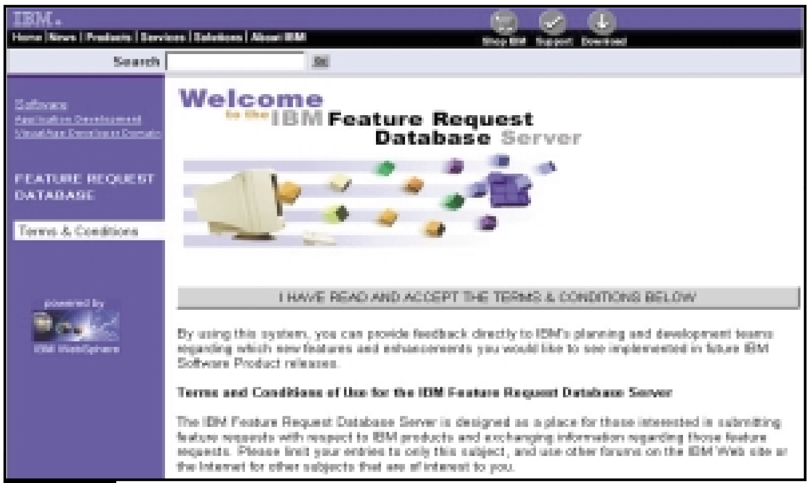
User Interface

The first screen that the customer will see (see Figure 1) contains the terms and conditions for the site and cannot be bypassed. (Should you decide to implement your own customer feedback servlet, your legal department will need to provide you with a precise wording for these terms and conditions.) The key message contained within these T's and C's is that by clicking on the "I Accept" button your customers are granting you the right to use their suggestions and incorporate them in a future release.

From a servlet point of view this is the start of the user's "session." That is, a unique session ID is generated and used as the index to user-state information that's accessed and updated by the screens that follow. If the user is entering a new feature request (see Figure 2), they are required to enter a one-line description and to select the relevant "component" (so that the correct VisualAge for Java development manager can also be notified).

Based on the one-line description, the servlet lists any similar feature requests. If a match is found, the user is encouraged to select an existing feature and add some additional comments; otherwise he or she can continue by selecting Open a New Feature.

This automatic and implicit search for any matching features helps to focus user input into fewer new feature requests, but with correspondingly more comments per request. This helps to simplify the subsequent analysis and prioritization. A more detailed description of the feature that can include links to other external Web pages or to other features must then be entered (see Figure 3). Finally (and after a confirmation screen) the feature is added to the feature database, and an e-mail is sent to the relevant component development manager.


IBM
Home | News | Products | Services | Solutions | About IBM
Shop IBM
Support
Download
Search
Software
Application Development
VisualAge Developer Domain
FEATURE REQUEST DATABASE
Terms & Conditions
powered by
IBM WebSphere
Welcome to the IBM Feature Request Database Server
I HAVE READ AND ACCEPT THE TERMS & CONDITIONS BELOW

By using this system, you can provide feedback directly to IBM's planning and development teams regarding which new features and enhancements you would like to see implemented in future IBM Software Product releases.

Terms and Conditions of Use for the IBM Feature Request Database Server

The IBM Feature Request Database Server is designed as a place for those interested in submitting feature requests with respect to IBM products and exchanging information regarding those feature requests. Please limit your entries to only this subject, and use other forums on the IBM Web site or the Internet for other subjects that are of interest to you.

FIGURE 1 Terms and conditions

Tidestone Technologies

www.tidestone.com

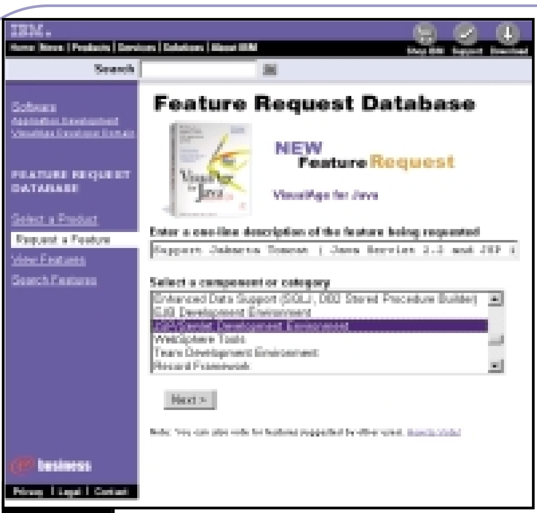

Feature Request Database
NEW Feature Request
VisualAge for Java
Enter a one-line description of the feature being requested
Select a component or category
Next >

FIGURE 2 New feature request

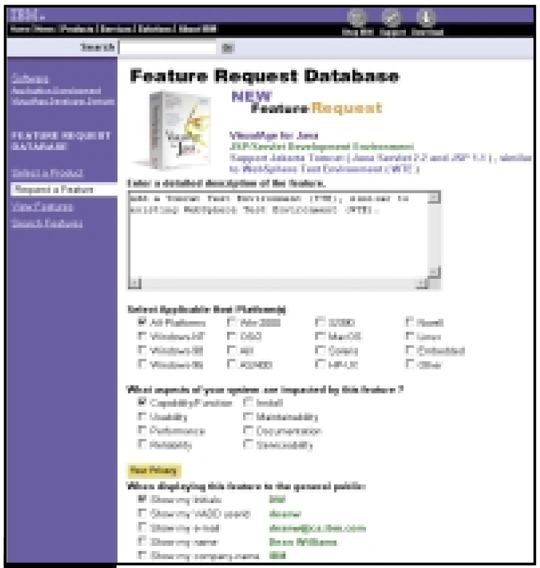

Feature Request Database
NEW Feature Request
Enter a detailed description of the feature.
Select Applicable Host Platform(s)
What aspects of your system are impacted by this feature ?
When displaying this feature to the general public:

FIGURE 3 Enter full details for this feature.

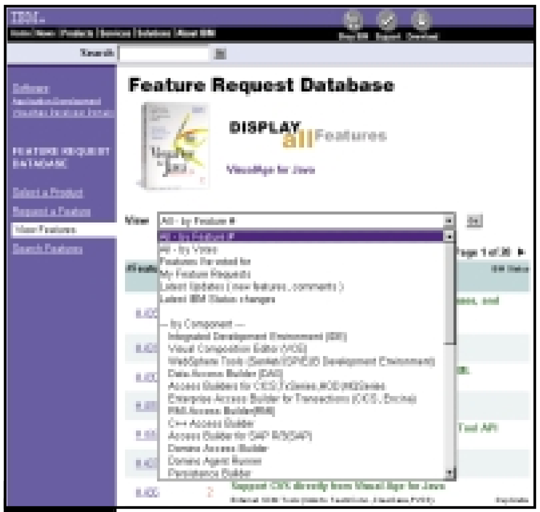


FIGURE 4 Other views

The screens required to enter a new feature have by design been kept very simple (and hopefully intuitive). As befits this keep-it-simple design approach, browsing the feature database involves selecting one of several standard report options in a list box (see Figure 4). Each report returns a list of matching features, and clicking on the feature number will display the feature's full details.

From this detail screen (see Figure 5) users can either add some additional comments or "vote" for this feature as one of their list of top-10 required features. Each user is allowed a maximum of 10 votes, one per feature.

Two-Tiered Authentication

As with many servlet-based systems, authentication is an important issue. Many of the functions described above (adding features, comments or votes) require that the user be uniquely identified and authenticated. The simplest approach is to place the servlet in a subdirectory that requires a valid userID/password to be entered before it can be accessed. Within the servlet the http ServletRequest.getRemoteUser() method will return the current userID. The downside of this approach is that it's usually all or nothing. Without a valid userID/password the user can't even browse the database. For many applications it's therefore desirable to have a two-tier approach to authentication. That is, a userID/password isn't required for simple browsing, but any function that results in information being either entered or updated would prompt for authentication.

Fortunately this two-tier (candy-store) approach can be achieved with only a small degree of extra effort. The key lies in creating an alternative path to your servlet that doesn't require authentication.

The preferred way for users to initially launch the servlet is as follows:

http://service5.boulder.ibm.com:8080/servlet/reqServlet3

In this mode the user is able to browse all information freely. It's only when users want to make a change (or view information and preferences that are specific to their userID) that they would be redirected to the original path to the servlet:

http://service5.boulder.ibm.com:8080/servlet/reqServlet2

and hence be required to log in (i.e., authenticate). Most Web application servers support the aliasing of servlets, including the specification of any required userID/password lookup file. To make this transition (from unidentified user to authenticated user) as seamless as possible, the servlet should carry over as much session state information as possible (i.e., the small degree of extra effort).

Performance

Performance and scalability are two other key factors that had to be addressed early in the application design phase. Many of the performance issues are directly related to the need to select and sort from a large list of features as quickly (performance) and efficiently (scalability) as possible. For example:

- Displaying all features sorted according to their total vote count
- Displaying all features that contain one or more search terms, ranked by their relevance

A number of file and data storage solutions are possible, ranging from the use of a traditional database engine (for example, DB2) to using linked lists, hashtables and arrays. Rather than incur the overhead of administering a full-function database, we decided to store all feature information in a simple hashtable, with additional arrays to cache both frequently used data and feature lists, sorted in various sequences.

One such sequence represents the list of all feature numbers (implemented as a static array of integers sorted according to their total votes). At first glance, re-sorting this sequence as votes are added or removed would appear to require a significant amount of CPU time (and hence adversely affect servlet response time). For a list of *n* items the usual rough estimate of the time taken to sort is of the order *n* times log*n* (using the quick sort algorithm).

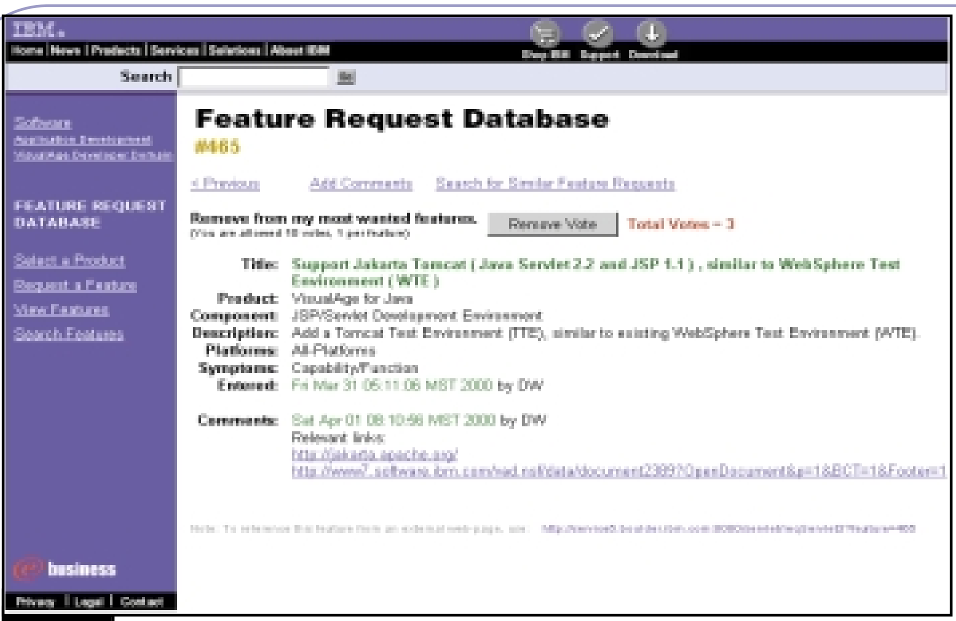
We found, however, that the sort time can be drastically reduced to only order *n* by using a simple trick, namely, the use of a bidirectional bubble sort that uses the results of the previous sort as input. Because relatively few features would have different vote counts since the last time a user had requested this sorted list, the bubble sort needs only a couple of passes over the list to resequence it based on the latest vote counts.

Persistence

Persistent data storage is managed through a set of APIs that wrapper any additions or changes made to the hashtable. Effectively, any updates to the hashtable also result in a corresponding record being appended to the end of a sequential disk logfile (where each record consists of a feature number, field name, field value

WebVision

www.webvision.com



IBM.

Home | News | Products | Services | Solutions | About IBM

Search

Software
Application Development
VisualAge Developer Domain

FEATURE REQUEST DATABASE

Select a Product
Request a Feature
View Features
Search Features

Feature Request Database

#465

< Previous   Add Comments   Search for Similar Feature Requests

Remove from my most wanted features. Remove Vote Total Votes = 3
(You are allowed 10 votes, 1 per feature)

Title:	Support Jakarta Tomcat (Java Servlet 2.2 and JSP 1.1) , similar to WebSphere Test Environment (WTE)
Product:	VisualAge for Java
Component:	JSP/Servlet Development Environment
Description:	Add a Tomcat Test Environment (TTE), similar to existing WebSphere Test Environment (WTE).
Platforms:	All-Platforms
Symptoms:	Capability/Function
Entered:	Fri Mar 31 05:11:06 MST 2000 by DW
Comments:	Sat Apr 01 08:10:56 MST 2000 by DW Relevant links: http://jakarta.apache.org/ http://www7.software.ibm.com/vad.nsf/data/document2389?OpenDocument&p=1&BCT=1&Footer=1

[illegible]

Privacy | Legal | Contact

FIGURE 5 Display feature details.

Author Bio

Dean Williams is a Java development manager at the IBM Toronto Laboratory in Canada and is currently focused on developing high-performance Java technology, including JIT compilers.

and some control/delimiter information). During servlet init(), this sequential data logfile is parsed into records and used to rebuild the hashtable (and all secondary caches). Periodically this sequential file is "compacted" – again during the servlet init() phase – so that any "old" records (i.e., records for which a newer version exists in the file) are deleted.

This ostensibly low-tech approach to persistence (versus, say, serializing objects to disk) was chosen in part for its simplicity, reliability and error-recovery characteristics. Even in the unlikely event of some system crash or hardware error corrupting one or more records in the logfile, the overall effect on the integrity of the data loaded into the hashtable would be very localized and minimal. Future plans for this data include a move to an XML-based storage format better suited for reuse by other applications.

Summary

This article has given a brief description of some of the design and implementation issues regarding building a simple servlet-based tool to capture and prioritize customer feedback. The feedback tool was developed using the WebSphere Test Environment in VisualAge for Java and was then moved to a production runtime environment provided by the WebSphere application server, Advanced Edition.

Given an opportunity, most customers are more than willing to provide you with their constructive suggestions and feedback regarding ways to improve your software tools and products. My hope is that this article may stimulate you to develop your own servlet-based e-business solution to further leverage your customers' input and incorporate it into your development plans for future releases.

I'd like to close by thanking both the Media Design and VisualAge Developer Domain teams at IBM for their many helpful suggestions and comments.

deanw@ca.ibm.com

QuickStream Software

www.quickstream.com

Visicomp, Inc.

www.visicomp.com/jdj7

PRODUCT REVIEW

Gen-it for Java 1.1

by Codagen

REVIEWED BY JIM MILBERY

AUTHOR BIO

Jim Milbery is a software consultant with Kuromaku Partners LLC (www.kuromaku.com), based in Easton, Pennsylvania. He has over 15 years of experience in application development and relational databases.

jmilbery@kuromaku.com

JAVA DEVELOPER'S JOURNAL JDJ ★★★★★★★★ WORLD CLASS AWARD

Test Environment
Client/Server:
Client: Dell Precision 410, 128MB RAM, 18GB disk drive, Windows NT Workstation 4.0 SP 5

Codagen Technologies Corporation
2075 University St., Suite 1020
Montreal, Quebec, Canada, H3A 2L1
Phone: 514 288-4802
Fax: 514 288-2446
www.codagen.com

Pricing:
Gen-it for Java $4,900/ Developer
Batch-it for Java $295/ Developer

In last month's issue of ***JDJ*** (Vol. 5 issue 6) I talked about the concept of using frameworks to automate the development of J2EE applications. Armed with this concept I took a look at Codagen Technologies' Gen-it for Java 1.1.

Codagen's product is designed around the basic premise that much of the lower-level structural code within an application is highly repetitive. Routine tasks such as persisting objects into the database, managing locks and performing integrity checks can be easily generated from a template rather than built from scratch each time.

Codagen's product is slightly different from a typical application generator in that it works in conjunction with frameworks that you may already have built. (In fact, Codagen has a partnership with IBM to leverage their San Francisco object framework with Gen-it for Java.) In the following sections I'll take a look at installing and using Gen-it for Java using one of their sample projects.

Installing and Configuring Gen-it for Java

I downloaded the 1.1 release of Gen-it for Java from Codagen's Web site. (Since this article was written, version 1.2 has been released.) The InstallShield setup kit is a little under 9MB in size and installs quickly and easily. Codagen uses Globetrotter's FlexLM software to control licensing for Gen-it for Java, but the downloaded installation kit is preconfigured with a demo license key already installed. When all is said and done, the installation process was a quick and painless affair – I had Gen-it up and running in about 15 minutes. One factor worth mentioning is that Gen-it for Java requires either Rational Rose or Visual Modeler (a version integrating with Telelogic's Tau, formerly Sterling's Cool:Jex, is scheduled for beta this month) to be installed on your local machine before the Gen-it installation can proceed. If you want to test Gen-it for yourself, you'll need to acquire one of the modeling tools in advance. (Rational allows you to download Rational Rose Enterprise Edition 2000 for a 20-day trial period as well.) Once the product is installed, it's a simple process to start working with Gen-it.

Working with Gen-it for Java

Codagen provides a tutorial that describes the basic value proposition for Gen-it for Java, and I would recommend going through it as a starting point. The Codagen folks believe strongly that frameworks are just as important for small and medium projects as they are for large projects – and the introduction section of the tutorial is dedicated to explaining this concept. As IT professionals and application developers we're certainly living in a time of tight deadlines and complex projects. I'd argue that very few projects in today's world are actually "small" projects. Sure, they might start out small; but they often end up being much larger efforts over the life of the application. Does this mean that we're underestimating the initial development effort in the first place? Not necessarily. We've all learned the lesson that large, multiyear projects are less likely to be successful than smaller, tactical projects. (We may be short on empirical evidence for this hypothesis, but it appears to be so from anecdotal evi-

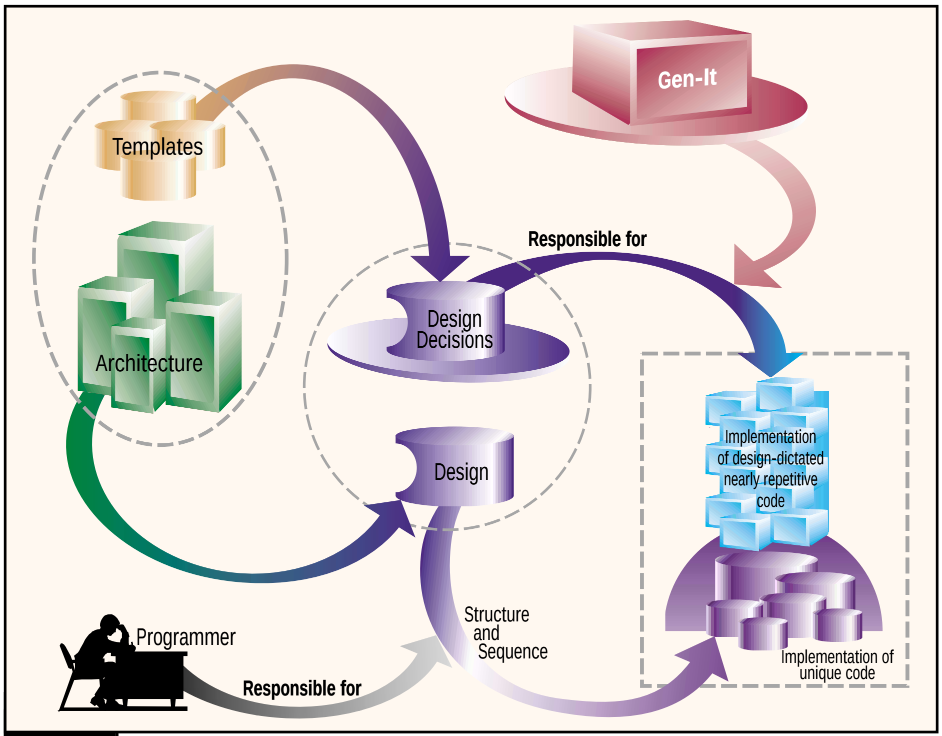


FIGURE 1 Modern software development with Gen-it

Allaire
Corporation

www.allaire.com/download

dence.) The result of this collective experience has been a trend toward projects that are much shorter in duration. This doesn't mean that we simply ignore larger, more complex, long-term projects. Rather, we have a better chance of success in managing large projects if we can break them down into a series of smaller projects with highly focused goals. Even projects that appear to be smaller in scope can grow as the needs of the business change. For many companies the first step into the Internet is a simple Web site soon to be followed by a dynamically generated Web site, online payment, interfaces to partners – the list is endless. Of course, one result of taking a large, complex project and breaking it down into smaller, more manageable chunks is that without a proper game plan it can lead to chaos. This is exactly where the folks at Codagen have designed the Gen-it product to play. Tackling larger projects (or small projects that grow into large projects over the life of the application) becomes possible when you use a well-planned architecture.

Gen-it is designed with the idea of removing the burden of implementing repetitive code from the hands of the programmer. It differs somewhat from typical frameworks in that it has been designed to work in conjunction with a framework. Thus, while most framework products are harnessed into the programmer's interactive development environment (IDE), Gen-it is plugged into the modeling environment. Figure 1 shows this relationship graphically.

The implied development methodology that Gen-it follows starts with a UML model. Having defined the model, you then choose a prefabricated framework for deployment (such as EJB or San Francisco). Once you have the framework, you then use the Gen-it tools to create a set of templates that would be used to harness the framework to your UML model. A thorough understanding of UML isn't required as Gen-it currently uses only the UML class diagram. Furthermore, Gen-it doesn't affect the visual representation of a UML model. It simply uses the items found in the static view – the class diagram – as convenient holders for extra structural information.

Codagen recommends you make use of their consulting resources in order to evaluate the Gen-it for Java product, and based on my experiences with it I would have to agree. While the tutorial attempts to walk you through the development process using the simple "Memento" framework model, inexperienced developers aren't likely to pick up on the benefits through the tutorial. Version 1.2 has reworked the tutorial to address these shortcomings.

The main interface for Gen-it is the organizer (see Figure 2), used by application architects in creating and managing templates. Templates define the specific actions implemented for the various classes in your application. In essence, the templates map the framework to your UML model.

Individual developers can be given access to the templates for use in generating code using the Batch-it interface. While Gen-it allows the application architect to modify templates, Batch-it developers are restricted to code generation based on the templates. In this way you can prevent individual developers from straying away from the framework by consolidating template management to a smaller subset of critical developers. Gen-it's real power is based on this template feature – but it's difficult to get the true flavor of templates from the downloadable kit. Even if you were able to work with templates extensively in the downloadable version, I would still recommend that you take advantage of Codagen's consulting services in evaluating Gen-it for Java. The real value in this product is in the creation and deployment of a framework throughout your application. By working closely with the Codagen experts, you're much more likely to get the most out of the templates interface. (The tutorial and user guides aren't enough to get you rolling with Gen-it on a real project.) If you're familiar with Rational Rose, at least, then part of the overall learning curve will be much easier. Gen-it for Java is linked directly into the Rational Rose IDE (see Figure 3).

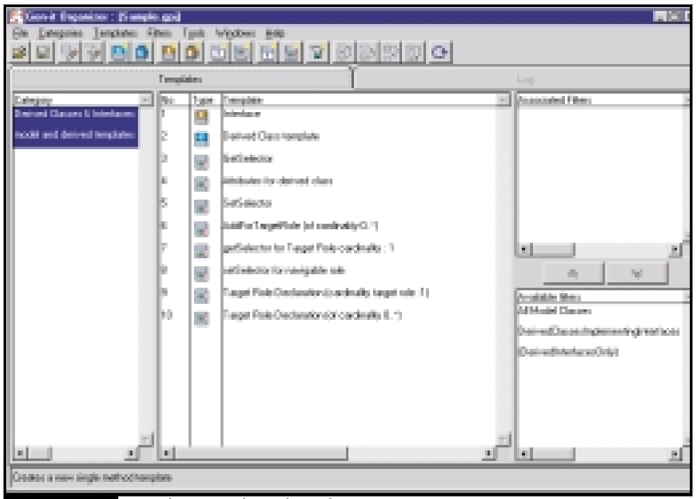
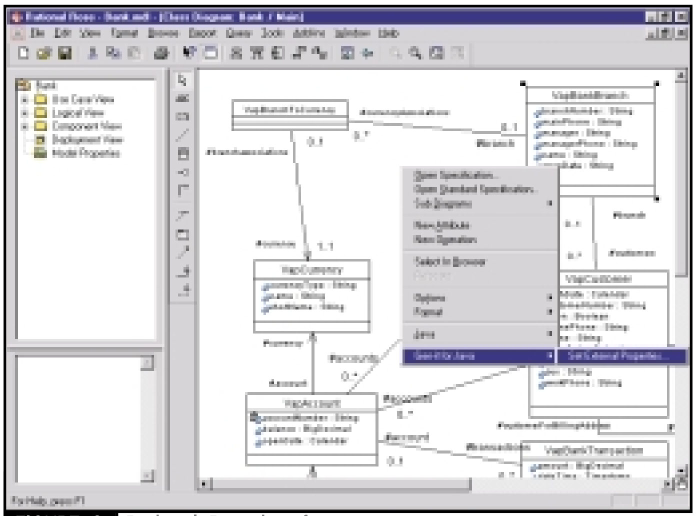


FIGURE 2 Gen-it organizer interface


File Edit View Format Browse Report Query Tools Add-Ins Window Help
Open Specification...
Select In Browser
Options
Format
Gen-it for Java
FIGURE 3 Rational Rose interface

The evaluation kit includes several Rational Rose models, including the Banking Model shown in Figure 3. Using the Gen-it tools within Rational Rose allows you to record design decisions directly in your UML model. Gen-it properties are attached to the UML model by means of External Properties, which are specific to a class, attribute or role. Gen-it uses the external properties that you set in the model to apply template definitions to your generated code. Once you select a set of classes (or packages) in your model and choose the "Gen-it" option from the menu, the Organizer interface will appear as shown in Figure 2. (Codagen provides an option that will allow Gen-it to write the generated code directly into IBM's Visual Age for Java as well.) The result is a fairly seamless transition from UML to code – provided that you have the proper templates.

Summary

The twin problems of limited resources and short deadlines can be mitigated by leveraging application frameworks. Codagen's Gen-it for Java product is a useful tool for connecting frameworks to models. Though Gen-it does require more than a casual understanding of Java and architecture, Codagen encourages you to use their technical support for evaluation and product issues. It's free and it helps them better understand customer issues. The consulting services can be used for more in-depth issues related to deeper architecture issues. Products such as Gen-it that have the potential to significantly shorten your development cycles are not simply "plugged-in and turned on." However, if you're short on Java programmers and want to make your development resources go farther, then a little upfront effort can pay big dividends down the road.

Cerebellum Software

www.cerebellumsoft.com

How to Use Java SoftReferences for Caching

You can prevent your garbage collector from throwing a dreaded OutOfMemoryError

WRITTEN BY
DARREN SHIELDS

They usually happen during the early hours of the morning, shortly before the code needs to ship: exception errors. . . .

Take the following , for example. Ever seen this before?

```
 Exception in thread "main"
java.lang.OutOfMemoryError
         at OutMem.main(OutMem.java,
Co
```

In the past, developers didn't have much control over garbage collection or memory management. Java2 has changed that by introducing the java.lang.ref package. The abstract base class Reference is inherited by classes SoftReference, WeakReference and PhantomReference. SoftReferences are well suited for use in applications needing to perform caching. WeakReferences are generally used to canonicalize several references to a single object. PhantomReferences are the "weakest" type of reference and can be used for referents that require additional cleanup after finalization.

In this article I'm going to focus on the SoftReference class and demonstrate how to use it to implement caching.

Creating a SoftReference

SoftReferences are created by passing an object to SoftReference's constructor:

```
 Object obj = new Object();
 SoftReference softRef = new SoftRef-
erence(obj);
 obj = null;
```

Please note the setting of obj to null. This is critical to proper performance of the garbage collector. A problem arises if no other stack frame places a value at that location – the garbage collector will still believe that an active (strong) reference to the object exists.

Retrieving a Reference

The following code can be used to retrieve a reference to an object:

```
 Object obj2;
 obj2 = sr.get();
 if (obj2 == null)  // GC freed this
     sr = new SoftReference(obj2 =
new Object());
```

Two items from the foregoing code are important to note. First, notice that SoftReference is immutable – you must create a new SoftReference that refers to the new referent. Second, the following code may appear to accomplish the same goal:

```
Object obj2;
 obj2 = sr.get();
 if (obj2 == null) {
  sr = new SoftReference(new
Object());
  obj2 = sr.get();
}
```

Hopefully the problem with this code is obvious: the garbage collector could run between the creation of the new Object and the call to get(). obj2 would still be null.

Where would you use a SoftReference? SoftReferences are excellent for use in memory-intensive applications that can benefit from caching. Take, for instance, a user interface such as a button with a picture on it. Typically these are implemented by subclassing Button. The paint method is then usually overridden (see Listing 1).

Reference Queues

Sometimes it may be important to know which references the garbage collector has cleared. Good-natured daemon that it is, the collector will happily place these references into a queue. All you have to do is supply the ReferenceQueue when you create the reference.

```
 ReferenceQueue rq = new Reference-
Queue();
 Object o = new Object();
 SoftReference sr = new SoftRefer-
ence(o,rq);
 o = null;
```

Later your program can query the queue by calling the nonblocking method poll. This will return a reclaimed reference or null if the queue is empty. Many cache implementations will loop through the queue during a call placing or retrieving items in the cache. Alternatively, ReferenceQueue provides two versions of the blocking method remove. One blocks indefinitely and the other will return after a timeout. Remove can be used instead of poll by a thread that blocks until items enter the queue, as in our next example.

Using SoftReferences to Implement a Cache

Programs can usually increase efficiency by reusing objects that are expensive to create. Objects such as database

Thinweb
Technologies

www.thinweb.com

connections and distributed objects – like those in RMI and CORBA – incur significant overhead in object creation, as well as create and destroy network connections. While caching is nothing new, SoftReferences allow us to grow the cache almost without bound. The guarantee that the garbage collector gives us is that it will always clear SoftReferences before throwing the dreaded OutOfMemoryError.

Listing 2 provides a class that can store instances of an object until they're needed. The cache's get method will return an object if one exists in the queue. If not, null will be returned and the program will need to create a new instance. When the program has finished using an instance, it simply passes the object to the cache's put method, where it will be available for future use.

The inner class Remover subclasses Thread and waits on the ReferenceQueue. To test the program, I wrote a cruel driver that attempts to flood the cache to ensure that items will be dropped rather than exhaust memory.

```
SoftCache sc = new SoftCache();
for (int i=0; i < 1000000; i++) {
    sc.put(new Object());
    if (i%10 == 0) {
        System.gc();
 Thread.yield();
 System.out.println("i=" + i);
    }
}
```

This is a bit like offering the cache a sip of water from a fire hose but it does the trick. To simulate the out-of-memory condition, an explicit call to gc() was added as well as a yield() to give the cache's Remover thread a chance to delete items from its vector. To coerce the garbage collector into removing some of the reference, I set the maximum heap size to a low value. Under Linux (kernel 2.2.12, JDK 1.2.2-RC3) the maximum heap size was set to 360K. The initial heap size defaults to over one megabyte, so it must be set as well. These options are set via Java's mysterious new -X parameters:

```
java -Xmx360k -Xms263k AbuseCache
```

Conclusion

Hopefully this article has demonstrated some techniques that can be useful to satiate an application's voracious appetite for memory. SoftReferences provide a way for the virtual machine to release memory held by large or infrequently used objects. Consider subclassing SoftCache to create new instances of an object you use. It could then also ensure that only objects of the proper type can be added.

darren@etrango.com.

AUTHOR BIO

Darren Shields, a team leader with The Technical Resource Connection, Inc., at the time this article was written, has extensive experience with CORBA, Java, C++, Linux and Open Source Software. Darren earned his bachelor's degree in computer science from the University of West Florida.

Listing 1

```
import java.awt.*;
import java.lang.ref.*;

public class ImageButton extends Button {
    private SoftReference imageReference=null;

    public ImageButton() {
 super();
    }

    public void paint(Graphics g) {
 Image image=null;
 if (imageReference != null)  // null first time we paint
     image = (Image)imageReference.get();
 if (image == null) {
     image = loadImage("Image name");
     imageReference = new SoftReference(image);
 }
 .
 .
 .
 image = null;
    }

    public Image loadImage(String name) {
 .
 .
 .
   }
}


}
```

Listing 2

```
  }
import java.lang.ref.*;
import java.util.Vector;

public class SoftCache {
    Vector vector=null;
    Thread remover;
    ReferenceQueue clearedRefs;

    public SoftCache() {
 vector = new Vector();
 clearedRefs = new ReferenceQueue();
 // start thread to delete cleared references from the cache
 remover = new Remover(clearedRefs,vector);
 remover.start();
    }

    public void put(Object o) {
 synchronized (vector) {
     vector.addElement(new SoftReference(o,clearedRefs));
 }
    }

    public Object get() {
 synchronized (vector) {
     if (vector.size() > 0) {
  SoftReference sr = (SoftReference)vector.elementAt(0);
  vector.remove(0);
  return sr.get();
     }
 }
 return null;
    }

    private class Remover extends Thread {
 ReferenceQueue refQ;
 Vector cache;

 public Remover (ReferenceQueue rq, Vector v) {
     super();
     refQ = rq;
     cache = v;
     setDaemon(true);
 }

 public void run() {
     try {
  while (true) {
      Object o = refQ.remove();
      synchronized (cache) {
   cache.removeElement(o);
   System.out.println("Removing " + o);
      }
  }
     } catch (InterruptedException e) { ; }
 }
    }
}
```

Download the Code!

The code listing for this article can also be located at

www.JavaDevelopersJournal.com

Flashline.com, Inc.

www.flashline.com

PowerTier 6 by Persistence

REVIEWED BY JIM MILBERY

jmilbery@kuromaku.com

Many Internet sites and applications began life as simple static Web pages. Once developers gained some initial experience working with the Web, these same applications went from being static pages to dynamic applications. In response, a host of dynamic application servers emerged in the marketplace, many of them featuring their own proprietary languages that were derivatives of HTML tags.

These application servers, a natural progression from classic three-tier computing models, could be plugged directly into your favorite Web server – an added attraction. Recently, this same market has begun to coalesce around JavaSoft's J2EE SDK as the standard backbone for application server engines. The J2EE framework evens the playing field between the large vendors and the smaller ones since they're both working from the same basic model. I recently got the chance to take a look at Persistence Software's PowerTier 6 Transactional Application Server – a hot entry in the application server race.

Installing and Configuring PowerTier

You can download PowerTier 6 from Persistence Software's Web site after you fill out a short qualification form. The installation procedure itself is relatively straightforward, but you'll need to do some postinstallation work. After the server is installed, you must run the PowerTier Profile wizard that verifies the overall configuration environment for you. I found the wizard helpful for pointing out problems, but it wasn't very helpful in terms of actually diagnosing the problem. Persistence runs a public news forum on its Web site and it seems like quite a few developers have had trouble with the configuration. There's quite a lot of functionality bundled into the package so I'm not surprised. Once you get the configuration issues out of the way, it's a relatively simple matter to get rolling with PowerTier 6. Persistence provides a thorough tutorial on the product and this document includes a simple diagram that highlights the development model with PowerTier 6 (see Figure 1).

Building a Simple Project

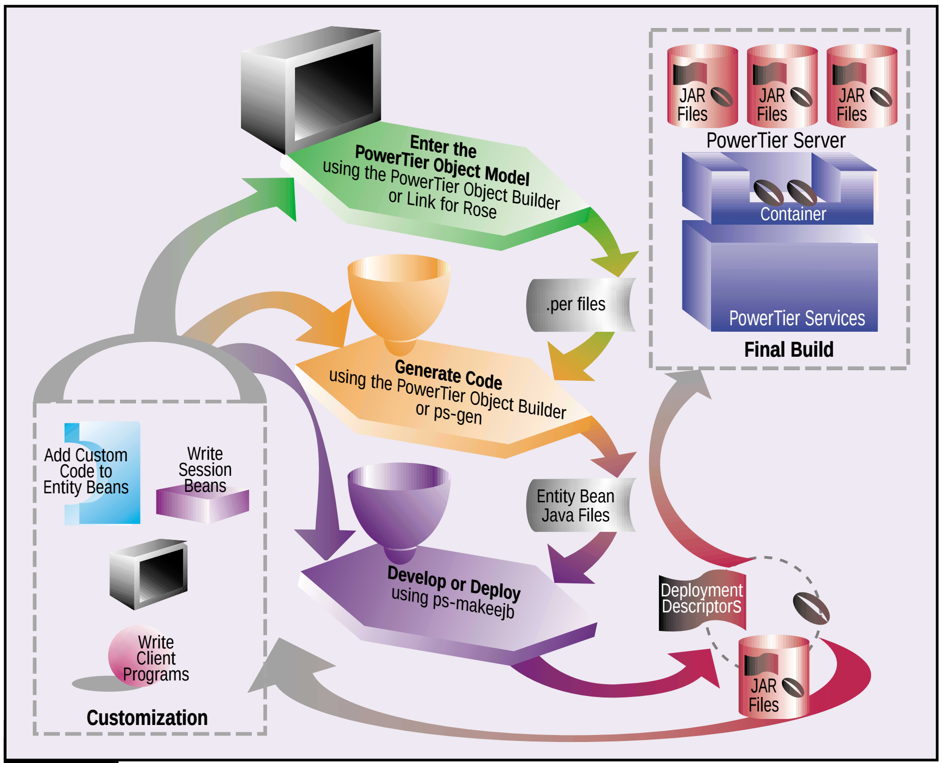
You can start designing your application with the PowerTier Object Builder or you can use PowerTier with Rational Rose. Persistence provides a link program for working with Rose version 4.0 and Rose 98. (The version on the Web site doesn't support Rational Rose Enterprise Edition 2000 per se, but I was able to use the link program with 2000 anyway.) To get the two products working together you have to make some manual changes (such as copying Rose menu files), but it's relatively easy to do. The PowerTier export procedure builds a PowerTier project file from the model within Rose. Using the default Rational Rose model of a banking application that PowerTier provides, I made some changes to the attribute definitions, then generated the project file.

Once you've generated your project file you can look at the classes using the PowerTier Object Builder (see Figure 2). The Builder isn't a full-fledged Java IDE (this is one of the drawbacks of PowerTier 6), but it's relatively easy to work with. Ideally, I would like to see the

Test Environment
Client: Dell Precision 410, 128MB RAM, 18GB disk drive, Windows NT Workstation 4.0 SP 5

Persistence Software, Inc.
1720 S. Amphlett Blvd. Third Floor
San Mateo, CA 94402-2701
Phone: 650 372-3600
Fax: 650 341-8432
www.persistence.com

Pricing:
Persistence licenses PowerTier on a concurrent developer basis at $7,500/ developer. The server is licensed on a runtime basis at $25,000/CPU for both NT and UNIX platforms, and site licenses are available.

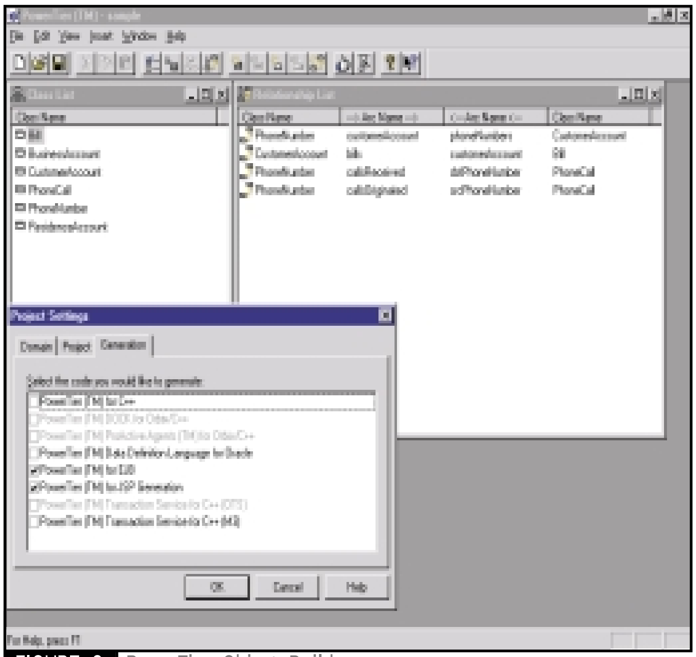

JAR Files
JAR Files
JAR Files
PowerTier Server
Container
PowerTier Services
Final Build
Enter the PowerTier Object Model using the PowerTier Object Builder or Link for Rose
.per files
Generate Code using the PowerTier Object Builder or ps-gen
Entity Bean Java Files
Develop or Deploy using ps-makeejb
Deployment Descriptors
JAR Files
Add Custom Code to Entity Beans
Write Session Beans
Write Client Programs
Customization
FIGURE 1 PowerTier development model

Appeal Virtual Machines

www.jrockit.com

Builder interface tied directly into several of the leading Java IDEs. Despite this limitation, the Builder packs a lot of power when it comes to generating the actual code. PowerTier supports EJBs, Servlets and JSPs, and comes packaged with the Apache Web server.

The Builder can actually generate the entire application (including the JavaServer Pages), as you can see from the project panel in Figure 2. I even generated simple data definition files for my Oracle database from within it. The result of the generation process included a set of source files for the EJBs and the JSPs to serve as the user interface for a simple banking application. The amount of application code for both the server and the user interface generated from this project file was quite impressive. The files themselves were organized into directories by project as well as by the client and server interfaces. The Java source files were well laid out and well documented and experienced developers should have no trouble working with the generated code. The project generation process was a snap, but the EJB-build process and server startup was a little more complex. Persistence provides a graphical interface for managing servers and applications, but the compilation process and server startup process relies on a set of script files. If you're going to have problems, this is where they'll occur. In the longer term I'd expect Persistence to provide some graphical tools for managing and building the generated source files, or they could merge the compilation interface into other leading IDEs.



Class Name
Bill
BusinessAccount
CustomerAccount
PhoneCall
PhoneNumber
ResidenceAccount

Class Name	--> Arc Name -->	<-- Arc Name <--	Class Name
PhoneNumber	customerAccount	phoneNumbers	CustomerAccount
CustomerAccount	bills	customerAccount	Bill
PhoneNumber	callsReceived	dstPhoneNumber	PhoneCall
PhoneNumber	callsOriginated	srcPhoneNumber	PhoneCall

Select the code you would like to generate:

- [ ] PowerTier (TM) for C++
- [ ] PowerTier (TM) BOOX for Odbc/C++
- [ ] PowerTier (TM) ProActive Agents (TM) for Odbc/C++
- [ ] PowerTier (TM) Data Definition Language for Oracle
- [x] PowerTier (TM) for EJB
- [x] PowerTier (TM) for JSP Generation
- [ ] PowerTier (TM) Transaction Service for C++ (OTS)
- [ ] PowerTier (TM) Transaction Service for C++ (MS)

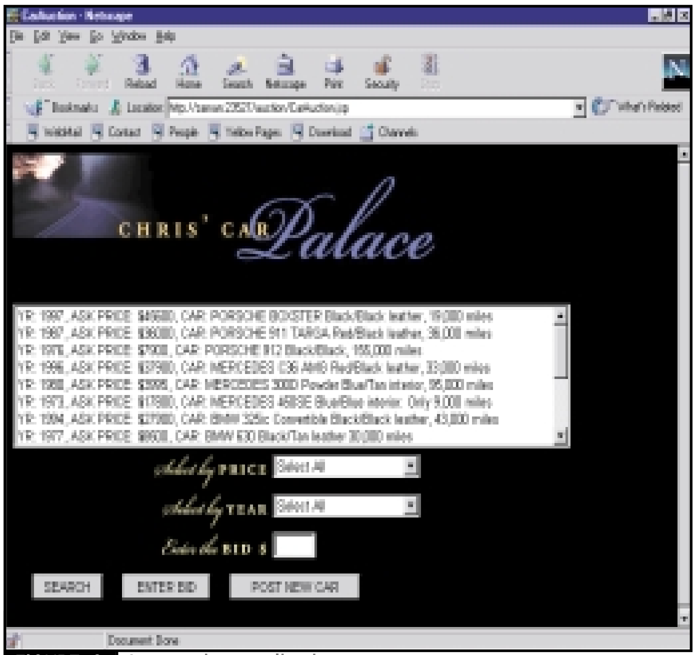
FIGURE 2 PowerTier Object Builder

PowerTier Server

The real power behind PowerTier 6 is the server engine itself. The server can be installed as a "regular" application server and can also be configured to work as an embedded application server (running inside your own code). Persistence claims to have some patented caching technology that allows PowerTier to handle heavy volumes of data with ease; Steve Eastin, director, product management at Persistence, considers the speedy performance of PowerTier to be one of its biggest assets.

PowerTier also provides connection pooling for the various databases it supports (including all the major players). Persistence supplies the JDBC drivers for the various databases and while it didn't appear to me that you could substitute your own drivers, Eastin assures me that you can use third-party JDBC drivers. There has been some discussion about this issue, however – and others relating to the database interfaces – on the developer forum on the Persistence Web site (www.persistence.com/developers/index.html).

Although I didn't perform any hands-on scalability tests with PowerTier 6, Persistence has some large customers (such as Instinet, InterShop and Shopnow.com) using the product for high-volume applications. According to Eastin, while the company has customers in many different markets, it has a strong following in the financial services, telco and dot-com verticals, and is so confident in the productivity of the PowerTier Object Builder and the runtime performance of the PowerTier Server that it's willing to help customers test out the technology firsthand. Eastin calls this strategy "PowerPilots" – essentially, prototypes of the customer's real application built with PowerTier.



```
YR: 1997, ASK PRICE: $49500, CAR: PORSCHE BOXSTER Black/Black leather, 19,000 miles
YR: 1987, ASK PRICE: $36000, CAR: PORSCHE 911 TARGA Red/Black leather, 36,000 miles
YR: 1976, ASK PRICE: $7900, CAR: PORSCHE 912 Black/Black, 155,000 miles
YR: 1996, ASK PRICE: $37900, CAR: MERCEDES C36 AMG Red/Black leather, 33,000 miles
YR: 1980, ASK PRICE: $3995, CAR: MERCEDES 300D Powder Blue/Tan interior, 95,000 miles
YR: 1973, ASK PRICE: $17900, CAR: MERCEDES 450SE Blue/Blue interior. Only 9,000 miles
YR: 1994, ASK PRICE: $27900, CAR: BMW 325ic Convertible Black/Black leather, 43,000 miles
YR: 1977, ASK PRICE: $9500, CAR: BMW 630 Black/Tan leather 30,000 miles
```

FIGURE 3 Car auction application

Platforms and Pricing

Persistence delivers the new releases of PowerTier on Windows NT and Solaris first, then ports to HP-UX, AIX and Linux shortly after. The ability to run on multiple hardware platforms is a critical capability for any application server. As a customer you'll want the flexibility to swap out hardware if your performance needs exceed any particular platform. It's also a good idea to have some choices from a purchasing perspective as well. You're more likely to get a better deal on a high-end hardware server platform if you can choose from among several competing vendors. PowerTier is written primarily in Java (there's some C++ code under the hood), so it's portable, but Persistence still takes the product through a porting process to ensure product quality. (A licensee of the J2EE, they're in the process of completing the lengthy certification process.)

Summary

If you want to take a look at PowerTier without working with it directly, you can take the interactive tour, which includes a sample car auction application (see Figure 3). The guided tour is a little bit too "sales-focused" for a developer, but the tutorial and user guide documentation are thorough enough to give you a good idea of the ins and outs of PowerTier 6. The next release of PowerTier will have a more seamless integration with Rose.

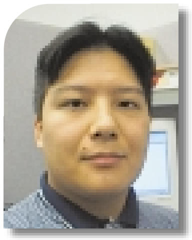
I was impressed with the overall quality of the product and the completeness of the server as regards scalability, reliability and performance. Based on my hands-on experience, I'd recommend that you take a look at PowerTier 6 for your next EJB project.

New Atlanta

www.servletexec.com

Transition to a Java-Based *n*-Tier Architecture in a B2B E-Commerce Solution: A Case Study

How a B2B service provider moved from a client/server solution to a Web-based application



WRITTEN BY
CHRIS KANG

Let me first intoduce PurchasePro.com, Inc. – a provider of Internet B2B electronic commerce services. The company's e-commerce solution consists of public and private "e-marketplaces" where businesses can buy and sell a wide range of products and services in an efficient, competitive and cost-effective manner. It was built using Microsoft technologies such as COM and SQL Server.

Until recently, our procurement solution was in the form of a Windows "fat application" that used a custom communications medium called the *Datapipe* to connect to our Microsoft SQL Server database. There were advantages as well as some inherent disadvantages to this approach. One of the advantages was that businesses rely on Microsoft to provide complete solutions for their technological needs and some businesses won't let non-Microsoft technology be a part of their solution. Another was that the learning curve for these technologies is fairly low, thus there are more qualified candidates to do the job.

The foremost disadvantage, however, was difficulty of maintenance: in an environment in which new bug fixes and hot fixes for major security holes are constantly being applied and released, keeping everyone up to date became a nightmarish task. Admittedly, the problem wasn't necessarily attributable to Microsoft technologies alone, but the frequency with which we needed to apply service packs and hot fixes was astronomical.

A second drawback was that, in releasing our solution as a Windows-executable only, we limited our market. Anyone not meeting the minimum requirements for our client application either had to buy a new computer or not use our solution. And third having the business logic in our fat application hindered us from rapidly implementing the new and improved features that keep us competitive.

Disadvantages like these prompted us to move from a client/server solution to a Web-based application and *n*-tier architecture. Moving to a Web-based solution not only opened new markets to us but allowed us to move toward implementing a true middle tier and to switch our back end from Microsoft SQL Server to Oracle. Currently our middle tier is built using Microsoft COM technology, which leverages object-oriented features such as reusability and encapsulation. Using COM allowed us to use the existing hardware and software to build a better, more robust solution. Our system integrates and interfaces with external systems using technologies including XML, EDI, OBI (Open Buying on the Internet) and various text file formats (see Figure 1). The current middle-tier architecture, though robust and solid, limits us to the Windows platform and, of course, to PC servers. With our exponential growth, outgrowing PC servers was just a matter of time.

Moving Beyond Microsoft COM

When examining possible technologies to move our middle tier into, Java – especially J2EE – presented itself as the clear winner.

The current J2EE specification provides many features ideal for solving problems. In particular, the EJB and XML technologies are perfectly suited to our needs. We needed a solution that would run on our existing hardware as

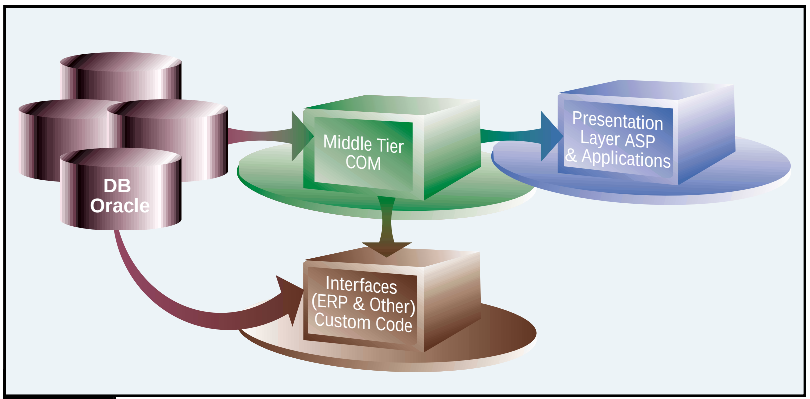

DB Oracle
Middle Tier COM
Presentation Layer ASP & Applications
Interfaces (ERP & Other) Custom Code

FIGURE 1 Current architecture

Software AG

www.softwareag.com/bolero

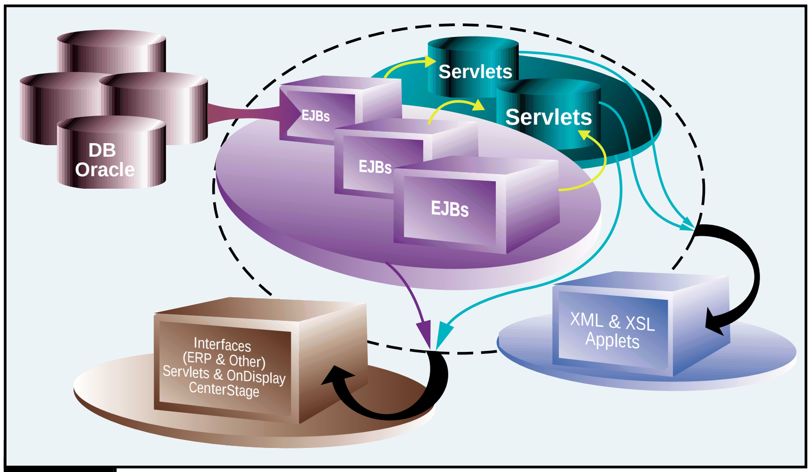


FIGURE 2 New architecture

well as on any hardware that we introduce into the middle tier in the future. One of Java's key selling points is the concept of "write once, run anywhere." Although the slogan doesn't hold completely true, making little or no modifications to the code to make it run on a completely new platform is definitely better than completely rewriting the code.

We also didn't want to be dependent on one vendor for key elements of our solution because at present we're at the mercy of Microsoft for bug fixes, enhancements and support. With Java we're able to switch JVMs, application server vendors and so on at will, if necessary. Scalability of our solution was another critical factor in our decision-making process. COM allows us to grow horizontally, but the lack of support for the Alpha platform in Windows 2000 means that there's no vertical growth. Java's ability to run on various platforms once again played to its advantage. In order to achieve vertical growth, all we have to do is buy a more powerful machine, even if it means that we have to switch the hardware platform and operating systems.

Our new middle tier will be database-independent and will consist of EJBs and servlets (see Figure 2). PurchasePro.com's own blend of XML will be used extensively in this new design. EJBs allow us to separate the database layer from the business logic. Most, if not all, of the business logic will reside at the EJB level in the new architecture, which will allow us to change databases or add additional layers of abstraction as necessary. By implementing EJBs, we'll be able to leverage such features as application-server-based caching, transactions, resource pooling and security. Servlets, combined with XML and XSL, separate the presentation layer from the core business logic. Servlets will match the XML with the XSL and process it through the XSLT engine to generate the required file whether it's HTML, WML or whatever.

One of our biggest problems is that our presentation layer programmers not only need to know the ins and outs of designing a user interface but are also required to be intimately familiar with PurchasePro.com's complex business logic. Currently, a tremendous amount of time, about a month on average, is spent bringing the presentation layer programmer up to speed on our business logic. By completely separating the business logic from the presentation layer, the presentation layer programmers can concentrate on designing more functional and efficient user interfaces instead of worrying about implementing the business logic.

One of the most requested and popular features in our current system is the use of a custom "skin." A skin allows for a change in the look and feel of the Web application. Using skins, we can change the logo, color schema and so on. Even with skins, the basic look and feel of the Web application remains unchanged. By leveraging XML and XSL, we can achieve a look that is truly customized. For example, a customer can display the purchase order in the same layout as their paper-based purchase order and even include their logo. We can format the same purchase order to be viewable on a PDA. As far as the form of presentation goes, the possibilities are endless – and it'll be as easy as creating an XSL sheet.

> "By completely separating the business logic from the presentation layer, programmers can concentrate on designing more functional and efficient user interfaces instead of worrying about implementing the business logic"

Redesigning Our Enterprise Resource Planning

With the redesign of the middle tier, our ERP interface and integration architecture needed to be redesigned as well. Currently all new projects involving ERP interfacing need to be written from scratch, and integration projects need major reworking in order to deploy them. Needless to say, this process is very time-consuming and tedious. Even though we have standardized methods of interfacing to – or integrating with – our system, there aren't any abstract classes that expedite such common chores as logging and error notification. We also create a separate executable for each interface that we create.

To alleviate the problem, the new architecture will have classes that execute tasks such as logging, error notification and setup. All interfaces will be dynamically loaded plug-ins to the main servlets. Since common tasks needn't be reprogrammed for each interface, turnaround time will be greatly reduced. The new architecture will not only have existing features such as extensive logging, pager and e-mail notification of errors, it will also implement new features such as message queuing and tighter integration with ERP systems through the use of OnDisplay's (www.ondisplay.com) CenterStage suite of tools.

Instead of building a custom interface to each customer's ERP system, we'll be using OnDisplay's certified interfaces to systems such as SAP, Peoplesoft and Oracle. We'll use OnDisplay's document transformation tool to handle most of our document transformation to formats such as EDI, delimited text file and fixed-width file.

FTP has been one of the most popular methods of transportation in exchanging documents for our clients, mainly because it's readily available. But the

Intuitive Systems, Inc.

www.optimizeit.com

problem with FTP is that it doesn't guarantee the delivery of data, and queuing needed to be done by a separate application – an added cost to our clients. Nor can FTP be used in a real-time environment, which is fine for most clients but for those who require real-time transactions we needed to create custom programming to satisfy their needs.

With the recent release of XMLConnect, a free product available at www.xml-connect.net (provided by OnDisplay), guaranteeing the delivery of data has become a reality for those of our customers who couldn't afford a solution. Despite the name, XMLConnect can transport many types of files and can be automated using its available (though rather limited) APIs. XMLConnect's ability to communicate with the CenterStage suite allows us to let clients who might not be able to afford OnDisplay's CenterStage suite exchange documents with us. It can be used in a real-time and a batch-mode system.

Seamless Login

XML integration is becoming popular in the enterprise. Technologies such as the cXML (www.cxml.org) punch-out model allow us to integrate with customers that already have an electronic catalog available on the Web. Seamless login using XML is another popular method of integrating with our system. A DTD has been created specifically for use on seamless login and we'll continue to support cXML punch-out with clients who currently use it. We're developing DTDs for all of our documents and functionality so that the clients who wish to create their own application can do so and still communicate with us as if they are using our Web application. The problem with XML, as far as using it for B2B document exchange is concerned, is that there's no standard such as EDI's X12. Since we can't force our flavor of XML on all our clients, we need a method of transforming the various XML documents to a format usable by our middle tier. We'll create an XML gateway to receive all sorts of XML documents and transform them into our proprietary format. The transformation engine will free our middle tier from worrying about translating different types of documents and concentrate instead on processing them. The transformation engine will not only receive different XML documents but will send them to the appropriate client as well.

Remaining Problems

One of the toughest challenges we've faced and continue to face during this migration concerns the speed of our solution. Or lack of it. Although our Java architects and programmers have yet to fully optimize the new architecture, preliminary results are less than promising. In certain tasks the new solution runs almost twice as slow as the current system. We were able to track down the main cause of our problems: one of our biggest bottlenecks is the Oracle JDBC driver. Under a simulated load of 50 concurrent users, almost half the time is spent in the JDBC driver. We've yet to find a suitable replacement for the driver.

Coming from a Microsoft background, I've found the IDEs for Java to be less than adequate. We have tried a variety of IDEs and reluctantly settled on Sun's Forté for Java. I know that the IDE doesn't matter in the end product, but it sure is nice to work with an IDE that increases productivity rather than decreases it.

The lack of a frozen code base has always been a challenge for us. Even though we'd like to freeze the code, the speed of growth of our company and our customer base makes it impossible to freeze the code long enough to carry out a major task such as converting to a Java-based middle tier.

flyinelvi@earthlink.net

AUTHOR BIO

Chris Kang is a lead ERP interfacing programmer at PurchasePro.com (www.purchasepro.com). A Microsoft certified professional, Chris is also involved in the Java migration of the whole middle tier.

Ensemble Systems

www.ensemble-systems.com

Modis Solutions

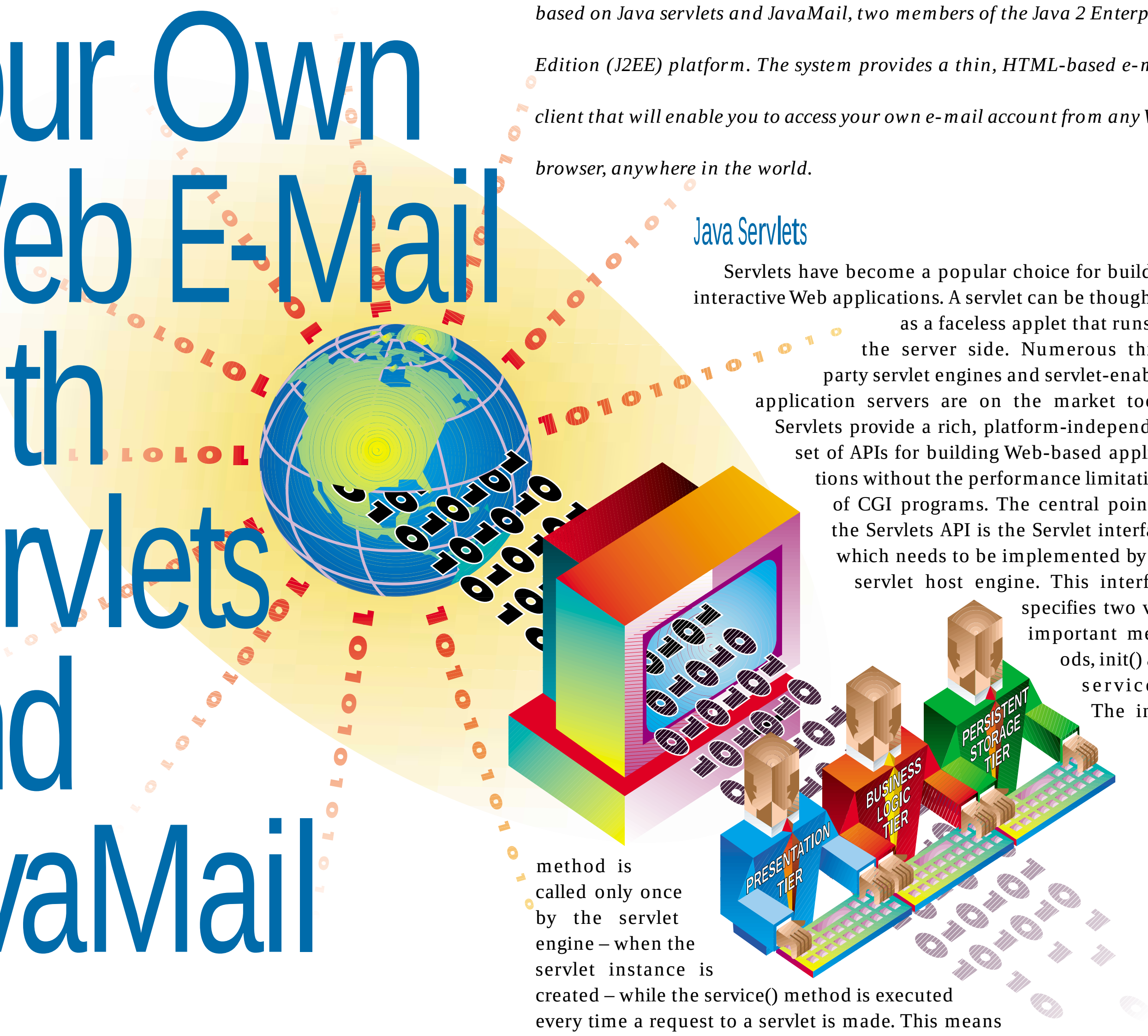
www.idea.com

Create Your Own Web E-Mail with Servlets and JavaMail

A THREE-TIER APPLICATION

WRITTEN BY DAVOR BISKO

Have you ever wanted to have your own Web e-mail system, rather than relying on free Web e-mail services? In this article I'll show you how to build a scalable Web e-mail system based on Java servlets and JavaMail, two members of the Java 2 Enterprise Edition (J2EE) platform. The system provides a thin, HTML-based e-mail client that will enable you to access your own e-mail account from any Web browser, anywhere in the world.



Java Servlets

Servlets have become a popular choice for building interactive Web applications. A servlet can be thought of as a faceless applet that runs on the server side. Numerous third-party servlet engines and servlet-enabled application servers are on the market today. Servlets provide a rich, platform-independent set of APIs for building Web-based applications without the performance limitations of CGI programs. The central point of the Servlets API is the Servlet interface, which needs to be implemented by the servlet host engine. This interface specifies two very important methods, init() and service(). The init() method is called only once by the servlet engine – when the servlet instance is created – while the service() method is executed every time a request to a servlet is made. This means that, depending on the servlet engine, multiple threads can execute the service() method at the same time, which is important if you have servlet member variables that need to be thread-safe. The Servlet API provides simple implementations of the Servlet interface in the form of the GenericServlet and HttpServlet classes. The e-mail servlet described here extends the HttpServlet class and overrides its init(), doPost() and doGet() methods.

JavaMail

The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API is a part of the J2EE. The e-mail servlet described here has been developed and tested with JavaMail version 1.1.3. The JavaMail API includes implementations of the IMAP and SMTP service providers. The POP3 service provider is available for download as a separate set of APIs. Almost 100 interfaces and classes are provided with the JavaMail API. The main ones to note are the Session, Store, Folder, Address, Message and BodyPart classes.

Cruel World

www.cruelworld.com

Three-Tier Thin Client Web Architecture

Multitier Web architectures have been widely adopted by today's enterprises. The Web e-mail solution described here is a three-tier application with a presentation tier, a business logic tier and a persistent storage tier (see Figure 1). The presentation tier is a Java servlet that processes user requests and generates HTML pages. The business logic tier uses JavaMail to validate users; track sessions; retrieve, compose and delete messages; and handle attachments. The persistency tier can be any e-mail server that supports SMTP and POP3 or IMAP.

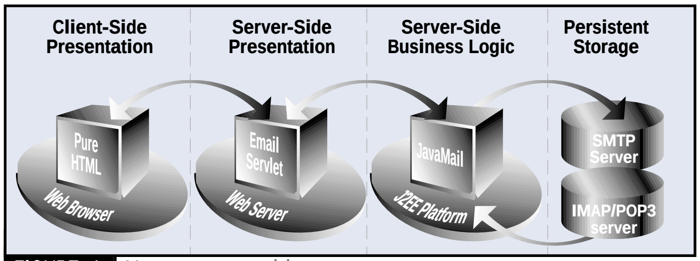


FIGURE 1 Message composition page

Initialization

The Servlet interface specifies that each implementation must have the init() method. The host servlet engine calls the init() method just once and must complete successfully before the servlet can receive any requests. In this case the init() method retrieves configuration parameters and initializes the default mail session. The parameters I used to develop and test the servlet are in Listing 1. The Java source for the init() method is in Listing 2. To successfully send and receive e-mail, the servlet needs the IP addresses of the incoming and outgoing mail servers. In most cases these addresses are the same, but there are cases when the servers might be separated. Next, the incoming mail protocol parameter needs to be assigned to the class variable, thus avoiding the parameter lookup every time a user logs on. JavaMail supports IMAP by default. The POP3 protocol is also supported, but has to be downloaded from JavaSoft as a separate product.

The next parameter to be retrieved is the Internet domain served by the e-mail servlet. This value is used to compose the reply e-mail address in the outgoing e-mail messages.

Next, you obtain the default e-mail session that will be shared among all servlet users. This e-mail session is used in the login process to retrieve the e-mail store for each user. The last parameter in the initialization process is optional. It's useful to turn the debug feature on and off. There are two different debug levels – one at the Mail Servlet level, the other at the JavaMail level.

Login

The first page presented to the user is the login page, generated by the e-mail servlet when called without parameters. This page is very simple, with only two edit fields (username and password) and two buttons (submit and reset). It also contains a hidden field called *command* that is the key to navigating the e-mail servlet. It tells the servlet about the next operation to be performed. The value of this field in the login page – "login" – signals the servlet to call its doLogin() method.

The first step is to extract the values of the submitted username and password fields. If any of these two fields are missing, the login page with the error will be sent back to the Web browser. The next step is to construct the URLName to be used for the user's authentication and e-mail store retrieval. The URLName needs the protocol, e-mail server address, port number, folder name, username and password parameters. In this case -1 is specified for the port number, which means the default port for the requested protocol will be used. If there's no provider for the specified protocol, the exception will be thrown and the error with the login page will be sent back to the Web browser.

After the user's store has been obtained, the user's inbox folder is retrieved from the store. This is actually the place where the user will be authenticated. If the authentication fails or something else goes wrong, the exception will be thrown and the error with the login page will be sent to the Web browser.

Client Sessions Tracking

The Java Servlets API provides an interface for user sessions handling. Which way it will implement this interface is up to the servlet host engine. Some support only simple session tracking by using cookies, which is a problem if a user disables cookies in his or her browser. Most of the commercial servlet engines implement more sophisticated user session tracking mechanisms without the need for cookies to be enabled in the Web browser.

The servlet engine should also give you a mechanism to configure user session parameters such as inactivity and timeout. You'll need to take all these things into consideration for your servlet engine selection. A user session can be obtained from the request object by calling its getSession() method.

```
HttpSession httpSession = req.getSession(true);
if (httpSession == null)
{
  // The servlet engine doesn't support user sessions or cookies
have been disabled by the user.
 … display error page here
}
```

The getSession() method accepts a boolean that indicates if a session needs to be created or the current user session is to be used. The doLogin() method is the only place in the servlet where "true" is passed, so a new session is created. All other operations assume that the user has already logged in and his or her session has already been created, so "false" is passed to the getSession() call.

After the user session has been created, the username, the password and the folder need to be stored into the session. These values are stored for later use by other servlet operations. The HttpSession interface provides putValue() and removeValue() methods to store and remove parameters.

Message List

After the user has been authenticated, the message list page is sent to the browser. This page is also generated when the Mail Server receives the "msglist" command.

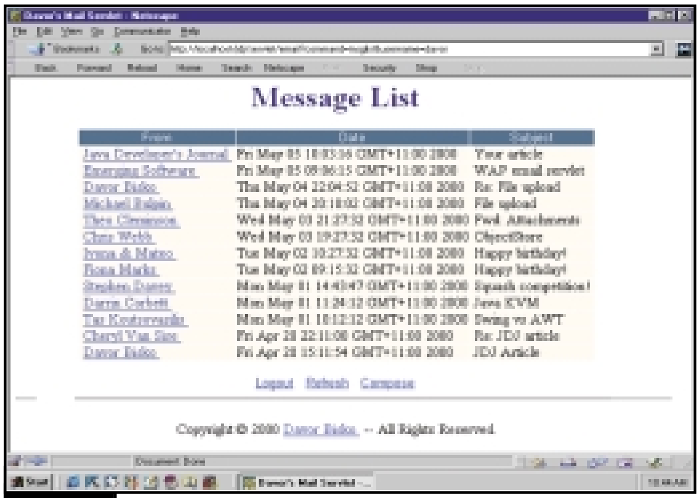
doMessageList() is the method that does this (see Listing 3). This method's first step is to open the inbox mail folder stored in the user's session and fetch a list of messages from the folder. To do this you need to create a fetch profile to tell the folder how to fetch messages. In this case only a list of user messages is displayed so the fetch profile is set to fetch message envelopes only, i.e., only message headers without their bodies will be fetched.

```
Folder folder = null;
Message[] msgs = null;
try
{
folder = (Folder)httpSession.getValue("folder");
if (!folder.isOpen())
folder.open(Folder.READ_WRITE);

msgs = folder.getMessages();
FetchProfile fp = new FetchProfile();
fp.add(FetchProfile.Item.ENVELOPE);
folder.fetch(msgs, fp);
 }
catch ….
```

Elixir Technology

www.elixirtech.com/elixircase.com



Davor's Mail Servlet - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Security Shop
Document: Done

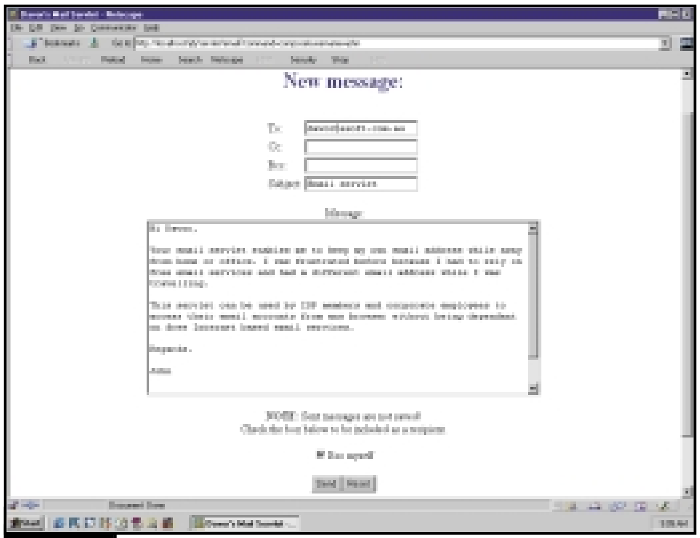
Message List

From	Date	Subject
Java Developer's Journal	Fri May 05 10:03:16 GMT+11:00 2000	Your article
Emerging Software	Fri May 05 09:06:15 GMT+11:00 2000	WAP email servlet
Davor Bisko	Thu May 04 22:04:52 GMT+11:00 2000	Re: File upload
Michael Rubin	Thu May 04 20:18:02 GMT+11:00 2000	File upload
Theo Clemenson	Wed May 03 21:27:32 GMT+11:00 2000	Fwd: Attachments
Chris Webb	Wed May 03 19:27:32 GMT+11:00 2000	ObjectStore
Ivana & Mateo	Tue May 02 10:27:32 GMT+11:00 2000	Happy birthday!
Fiona Marks	Tue May 02 09:15:32 GMT+11:00 2000	Happy birthday!
Stephen Dunne	Mon May 01 14:43:47 GMT+11:00 2000	Squash competition!
Darren Corbett	Mon May 01 11:24:12 GMT+11:00 2000	Java KVM
Tai Kostowenski	Mon May 01 10:12:12 GMT+11:00 2000	Swing vs AWT
Cheryl Van Sise	Fri Apr 28 22:11:08 GMT+11:00 2000	Re: JDJ article
Davor Bisko	Fri Apr 28 15:11:54 GMT+11:00 2000	JDJ Article

Logout Refresh Compose

Copyright © 2000 Davor Bisko -- All Rights Reserved.

FIGURE 2 Multitier e-mail servlet architecture



Davor's Mail Servlet - Netscape
Back Reload Home Search Netscape Security Shop
Document: Done

New message:

To: davor@[illegible].com.au
Cc:
Bcc:
Subject: Email servlet

Message

```
Hi Davor,

Your email servlet enables me to keep my own email address while away
from home or office. I was frustrated before because I had to rely on
free email services and had a different email address while I was
travelling.

This servlet can be used by ISP members and corporate employees to
access their email accounts from any browser without being dependent
on free Internet based email services.

Regards,

John
```

NOTE: Sent messages are not saved!
Check the box below to be included as a recipient

☑ Bcc myself

Send Reset

FIGURE 3 Message list page generated by the servlet

The next step is to loop through the array of fetched messages and generate the HTML table for the message details. Each row in the table shows the message "from," "date" and "subject" fields (see Figure 2). If you wish, you can add more fields to be displayed, such as the "cc" or "reply to."

The "from" field in the table is created as a hyperlink, where the username and the message number are set as the parameters to be sent by the browser when the hyperlink is selected. At the bottom of the page the standard operation hyperlinks are displayed.

Message Retrieval

As mentioned above, each message in the displayed message link has a hyperlink that, when selected, sends the retrieve command and the message number to the e-mail servlet. This triggers the doRetrieve() method call, which opens the user's folder and retrieves the requested message number from the folder. If the message exists, the displayMessage() method is called to generate the HTML page for the message header and body. First, the message header elements subject, date sent, sender and recipients will be retrieved to be embedded in the HTML page. The next task is to determine the message content type. If the content type is "text/plain" the content is sent as is to the browser. If the content type is "multipart/*" the message content will be cast to the Multipart object type, which will contain message parts. Each message part is retrieved as the Java BodyPart object type, which is done by going through a loop and retrieving body part objects from the multipart object. For each body part the displayPartDetails() method is called to generate the HTML code if the part is a plain or HTML text. If the part is of any other type, such as a gif or a zip file, only the file name will be displayed as a hyperlink. The user can then click on this hyperlink to retrieve the part. In this case the body part is sent as a raw array of bytes to the browser, which will either be able to handle it or prompt the user for the part to be saved.

Compose, Reply, Forward

The e-mail composition page is generated by the doCompose() method called by the servlet when a new, reply or forward message is needed. The page is simple, with input fields for the recipient addresses (to, cc and bcc), the subject and the message body. While the message subject and body are empty when a new message is being composed, for the reply and forward commands the original message fields must be assigned. This e-mail servlet doesn't handle user folders, so e-mails sent aren't saved – but users can be given the option to bcc themselves, as in the example shown in Figure 3.

When a message is composed and the user clicks on the send button, the servlet will call its doSend() method. This method is quite straightforward and its Java source is in Listing 4. After creating a MimeMessage object, it extracts the values for the recipients, message subject and the body from the message composition page. These values are placed into the created message object along with the sent date and the mailer values. The last step is to call a static method send() of the JavaMail Transport class with the message as the parameter. From this point onward the JavaMail and the SMTP server will handle message delivery. If one of them can't deliver the message, the MessagingException will be thrown and the error page will be sent to the browser.

Attachments

Well, this is where you'll have to roll your own! Java Servlet API doesn't handle multipart/form-data content types such as when the browser uploads a file to the Web server. First, you'll need to set the form enco-type to multipart/form-data in the doCompose() method, as follows:

```
toClient.println( "<form encotype=multipart/form-data action=" +
HttpUtils.getRequestURL(req) + " method=post>" );
```

You'll also include a field in the compose form to select a file to be attached:

```
<input type=file size=20 name=Attach>
```

Since there is another content type to handle now, you'll have to check it in the doProcess() before doing any processing.

```
String contentType = req.getHeader("Content-Type");
if (contentType != null &&
     contentType.toLowerCase().startsWith("multipart/form-data"))
{
  MultipartServletRequest msr = new MultipartServletRequest(req);

  doSend(msr, res);
}
else // do normal processing
```

The MultipartServletRequest is the class you need to create to extract the request parameters and the uploaded file. As the file upload is a frequently requested functionality for Web sites and intranets, I don't know why a similar class isn't included in the Servlets API. It's an important class, but unfortunately I can't include it here due to limits of space.

Security

When you implement this solution security will be a major issue, so you might consider an encryption technique such as the Secure Sockets

Fiorano

www.fiorano.com

Cloud

www.cloud

scape

lscape.com

Layer (SSL) to protect the content of the data exchanged between the servlet and the browser. This should be used for the login page – or maybe for all servlet operations if the content of your e-mail is sensitive.

The other security concern is denial-of-service attacks in which huge amounts of data are sent to the Web server with the aim of bringing it down. You can prevent this by checking the length of content received in the servlet before processing the request. Currently, the maximum content length in the e-mail servlet is 1MB, but you'll probably need to increase this limit if you're intending to send big attachments via the servlet.

Enhancements

The e-mail servlet described is based on the Java servlets technology, used when the presentation layer doesn't change often. If you frequently change the look of your Web pages, you should consider converting this e-mail servlet into a JavaServer Page (JSP). You could characterize servlets, briefly, as "HTML code in Java code," while JSPs are "Java code contained in HTML code."

Another architectural modification to this solution may be considered. If you need to distribute its processing to achieve better scalability, you would extract the business logic from the servlet and wrap it into Enterprise JavaBean(s). The bean(s) would run on a separate box and would be created and used by the servlet.

Summary

The solution described in this article is just a starting point for creating a Web e-mail system. With a little imagination you can extend the functionality of the system I have described and integrate it into your Web site. The purpose of this article is to show you how powerful Java servlets are when combined with other J2EE technologies such as JavaMail. Moreover, Java's support for open standards enables this solution to run on any platform and to support any e-mail server.

Author Bio

Davor Bisko, an architect specialist at Unisys in Melbourne, Australia, is focused on multitier Web applications. He holds a master's in IT and is a Sun-certified Java programmer, developer and architect with nine years of object-oriented application design and development experience in C++ and Java.

davor@esoft.com.au

Listing 1

```
# email servlet
email.code=au.com.esoft.servlet.MailServlet
email.initparams=\
  sendserver=smtp.esoft.com.au,\
  recvserver=pop3.esoft.com.au,\
  protocol=pop3,\
  domain=esoft.com.au,\
  debug=true
```

Listing 2

```
public void init(ServletConfig config)
                   throws ServletException
{
  super.init(config);

  sendServer = getInitParameter("sendserver");
  if (sendServer == null)
    throw new ServletException("Send mail server parameter
    missing!");

  recvServer = getInitParameter("recvserver");
  if (recvServer == null)
    throw new ServletException("Incoming mail server parame-
    ter missing!");

  protocol = getInitParameter("protocol");
  if (protocol == null)
    protocol = "pop3";

  domain = getInitParameter("domain");
  if (domain == null)
    throw new ServletException("Domain parameter missing!");

  Properties props = System.getProperties();
  props.put("mail.smtp.host", sendServer);
  mailSession = Session.getDefaultInstance(props, null);

  String debugParam = getInitParameter("debug");
  if ("true".equals(debugParam))
  {
    debug = true;
    mailSession.setDebug(true);
  }
  else
  {
    debug = false;
    mailSession.setDebug(false);
  }
}
```

Listing 3

```
public void doMessageList(HttpServletRequest req,
                       HttpServletResponse res)
                       throws ServletException, IOException
{
  PrintWriter toClient = res.getWriter();

  // Get the session for this user.
  HttpSession httpSession = req.getSession(false);
  sendPageStart(toClient);
  Folder folder = null;
  Message[] msgs = null;
  try
  {
    folder = (Folder)httpSession.getValue("folder");
    if (!folder.isOpen())
      folder.open(Folder.READ_WRITE);

    msgs = folder.getMessages();
    FetchProfile fp = new FetchProfile();
    fp.add(FetchProfile.Item.ENVELOPE);
    folder.fetch(msgs, fp);
  }
  catch (MessagingException me)
    toClient.println("<tr><td>" + me.toString() +
                                 "</td></tr>");

  if (msgs.length > 0 )
  {
    sendReplyHeader(toClient, "Message List");
    sendTableHeader(toClient);
  }
  else
    sendReplyHeader(toClient,
          "There are no messages in the mailbox!");

  // For each message, show its header
  for (int i = msgs.length; i > 0; i--)
  {
    try
    {
      Message m = msgs[i-1];
      if (m.isSet(Flags.Flag.DELETED))
        continue;

      // Generate HTML code for the message header details.
      toClient.println("<tr bgcolor=#eeeecc><td><a href=" +
                             HttpUtils.getRequestURL(req) +
                             ?command=retrieve&username=" +
                               username + "&msg=" + i +
                              ">" +
                              m.getFrom()[0].toString() +
                              "</a></td><td>" +
                              m.getSentDate() +
                              "</td><td>" +
                             m.getSubject() +
                             "&nbsp</td></tr>");
```

Iam Consulting

www.iamx.com

```
    }
    catch (MessagingException me)
      toClient.println("<tr><td>" +
                                   me.toString() +
                                   "</td></tr>");
}

  if (msgs.length > 0 )
    sendTableFooter(toClient);

  // Send the mail options.
  toClient.println( "<center>" );
  toClient.println( "<nobr>&#160;" +
                            "<font face=Arial, Helvetica " +
                            "size=2 color=#FFFFFF " +
                            "style=color:#FFFFFF; " +
                            "text-decoration:none><a href=" +
                             HttpUtils.getRequestURL(req) +
                            "?command=logout&username=" +
                             username +

">Logout</a></font>&#160;</nobr> ");
  toClient.println( "<nobr>&#160;" +
                            "<font face=Arial, Helvetica " +
                            "size=2 color=#FFFFFF " +
                            "style=color:#FFFFFF; " +
                            "text-decoration:none><a href=" +
                               HttpUtils.getRequestURL(req) +
                               "?command=msglist&username=" +
                               username +

">Refresh</a></font>&#160;</nobr> ");
  toClient.println( "<nobr>&#160;" +
                            "<font face=Arial, Helvetica " +
                            "size=2 color=#FFFFFF " +
                            "style=color:#FFFFFF; " +
                            "text-decoration:none><a href=" +
                               HttpUtils.getRequestURL(req) +
                               "?command=compose&username=" +
                               username +

">Compose</a></font>&#160;</nobr> ");
  toClient.println( "</center>" );

  try
    folder.close(true);
  catch (MessagingException me)
    me.printStackTrace();

sendPageEnd(toClient);
}
```

Listing 4

```
public void doSend(HttpServletRequest req,
                                   HttpServletResponse res)
                                   throws ServletException,
                                   IOException
{
  PrintWriter toClient = res.getWriter();

  // Get the session for this user.
  HttpSession httpSession = req.getSession(false);

  sendPageStart(toClient);
  try
  {
    MimeMessage msg = new MimeMessage(mailSession);
    InternetAddress sender =
                new InternetAddress(username + '@' + domain);
    msg.setFrom(sender);
    String recipients = req.getParameter("to");
    // JavaMail follows RFC822 standard which allows
    // only comma separated addresses, so we have to
    // replace semicolons with commas, if any
    recipients = recipients.replace(';', ',');
    Address[] address =
              (Address[])InternetAddress.parse(recipients, false);
    msg.addRecipients(Message.RecipientType.TO, address);
    // Parse the 'cc' recipients and add them to the message.
    recipients = req.getParameter("cc");
    recipients = recipients.replace(';', ',');
    address = (Address[])InternetAddress.parse(recipients, false);
    msg.addRecipients(Message.RecipientType.CC, address);
    // Parse the 'bcc' recipients and add them to the message.
    recipients = req.getParameter("bcc");
    recipients = recipients.replace(';', ',');
    address = (Address[])InternetAddress.parse(recipients, false);
    msg.addRecipients(Message.RecipientType.BCC, address);
    // Check if we have to send the message to the sender too.
    String bccMyself = req.getParameter("bccmyself");
    debugMsg("Bcc myself = " + bccMyself);
    if (bccMyself != null)
    msg.addRecipient(Message.RecipientType.BCC, sender);
    // Set the message subject.
    String subject = req.getParameter("subject");
    msg.setSubject(subject);
    // Put the message body in the message.
    String body = req.getParameter("body");
    msg.setText(body);
    // Set other message parameters.
    msg.setHeader("X-Mailer", productName);
    msg.setSentDate(new Date());
    // Finally send the message.
    Transport.send(msg);
    sendReplyHeader(toClient, "Message sent.");
  }
  catch (AddressException ae)
    sendReplyHeader(toClient, ae.getMessage());
  catch (MessagingException me)
    sendReplyHeader(toClient, me.getMessage());

  toClient.println( "<br><br>" );
  toClient.println( "<center>" );
  toClient.println( "<nobr>&#160;" +
                         "<font face=Arial, Helvetica " +
                         "size=2 color=#FFFFFF " +
                         "style=color:#FFFFFF; " +
                         "text-decoration:none>" +
                         "<a href=" +
                          HttpUtils.getRequestURL(req) +
                         "?command=logout&username=" +
                          username +

                         ">Logout</a></font>&#160;</nobr> ");
                          toClient.println( "<nobr>&#160;" +
                         "<font face=Arial, Helvetica " +
                         "size=2 color=#FFFFFF " +
                         "style=color:#FFFFFF; " +
                         "text-decoration:none>" +
                         "<a href=" +
                          HttpUtils.getRequestURL(req) +
                         "?command=msglist&username=" +
                          username +
                         ">Message
                          List</a></font>&#160;</nobr> ");
                          toClient.println( "<nobr>&#160;" +
                         "<font face=Arial, Helvetica " +
                        "size=2 color=#FFFFFF " +
                        "style=color:#FFFFFF; " +
                        "text-decoration:none>" +
                        "<a href=" +
                         HttpUtils.getRequestURL(req) +
                        "?command=compose&username=" +
                         username +

                        ">Compose</a></font>&#160;</nobr> ");
                         toClient.println( "</center>" );

  sendPageEnd(toClient);
}
```

Softwired

www.softwired-inc.com/ibus

CORBAsecurity — The State of the Art and of the Market

Examining the state of distributed object security

WRITTEN BY
JON SIEGEL

The Internet originally interconnected a small number of computers at universities and research labs. It was used to share resources and to send e-mail – an incidental application that over time grew into one of the major uses of the network. Everyone knew everyone else, and security was far from the priority that it is today. All this has now changed....

These days, security is a top priority for ISPs and consumers alike. Hundreds of millions of people are now connected to the Internet and items of tremendous value are at risk daily – from stocks and bonds to military and government secrets.

Security comes to mind as the topic for this month's **CORBA Corner** because every spring the Object Management Group hosts a workshop on Distributed Object Computing Security, or DOCSEC. This year's conference was also cosponsored by the National Security Agency (NSA) – which has been involved in every workshop since their inception – and by Concept 5 Technologies. Because so many vendors and serious security users get together to describe their experiences at this workshop, it provides an excellent opportunity to sum up the state of distributed object security. I'm starting this column with descriptions of a few standard distributed-security architectures to establish a base; then I'll summarize key points from the conference.

Security in Java and EJB

In Java, network security is commonly provided by the Secure Sockets Layer, which adds a secure network connection to the Java "sandbox." The SSL implementation – depending on the cipher suite – provides identification and authentication (primarily authenticating the server to the client), and may protect messages against interception (by encryption), modification (with a checksum) and replay (with a timestamp).

Chapter 15 of the Enterprise JavaBeans 1.1 specification defines a somewhat richer security architecture, albeit with a looser set of rules. Fundamentally, it places responsibility for security on the container (and therefore on the container provider), removing the burden from the application developer and allowing security policies to be set and changed at assembly or deployment time rather than development time. Security functionality includes individual privileges, group privileges and delegation, although the details are left to the container, which in addition defines the principle and its format. Because the EJB security model is defined in terms of a single container environment, it doesn't define a network protocol nor does it tell how to spread a security umbrella over containers from multiple vendors.

CORBAsecurity

CORBAsecurity defines a much richer security architecture. As with SSL, it starts with identification and authentication of the client, the user (without restricting the authentication mechanism, which might be a password, code card or a biocharacteristic such as thumbprint or retina scan) and the server to ensure that the client isn't talking to a "Trojan horse." It also places a checksum on the message to protect against modification, encrypts it to protect against interception and includes a timestamp to protect against replay.

Going further, CORBA refines the concept of delegation. Earlier we mentioned delegation without giving a definition. Delegation occurs when our client calls an object to perform some operation for us and that object, instead of performing all the work itself, delegates part of it to some other object that it invokes on our behalf. Nothing restricts delegation to a single level – multiple layers of delegation are common in object-oriented systems, whether local or distributed. When we program a client to invoke an object, we may not assume that the object will perform the entire operation itself. In fact, we may not even assume that the operation will execute on the same hardware that we dispatch our invocation to.

In its simplest form, delegation takes the form of impersonation – the intermediate object passes the caller's credentials down the chain. This object – and all other objects that receive them – is empowered (at least as far as security is concerned) to do anything that the client could do. Imagine passing this power to a server that makes transactions on your bank account or your retirement account. If you didn't have absolute trust in the first server object to do the right thing every time, you'd be very reluctant to make the first invocation. While this simple form of delegation is all that many environments provide, CORBAsecurity defines a number of refinements restricting, for example, which objects are authorized to use the credentials for delegation, how many levels deep delegation may proceed, which privileges are delegated or how long the credentials remain valid.

But wait, there's more. In a system with an established set of resources – including objects in a CORBA system or CPUs, printers, disks and files in other systems – you may want to control access to individual resources or to classes of resources, either by individual users or by groups of users. This is done by defining permissions in access control lists (ACLs), which are then enforced by the security infrastructure. CORBAsecurity includes ACL-based restrictions.

You may think that a perfect security system is one that can't possibly be breached, but it's not true. Security experts know that any security system

Evergreen

www.evergreen.com/jdj.html

can be breached – in an otherwise perfect system, by insider access – so the best security architectures and installations audit security-relevant actions. When there's an actual or suspected breach, security administrators can examine the audit records, hopefully identify what happened, fix the damage and possibly identify the perpetrator.

And finally, there's one service that falls under the security umbrella: nonrepudiation. This function relies on a trusted third party to permanently archive evidence of something important to a transaction on behalf of both parties – for example, that a source really did originate a message (which might be an order or something electronic of intrinsic value, such as a program or musical recording) or that a site really did receive a message or electronic payment. Nonrepudiation, therefore, plays a vital role in some B2B transactions.

State of the Market

The state of the market for CORBAsecurity implementations is very good. Four of the companies attending this year's DOCSEC had solutions. Let's look briefly at two of them now; the other two, from Adiron and Gradient, will come up in our presentation of the Workshop proceedings.

CONCEPT 5

Workshop cosponsor Concept 5 furnishes the security implementation marketed as Orbix Security 3.0 by Iona and as TPBroker Security by Hitachi. This engine will appear on the market with other ORBs in the future. Implemented in an ORB-independent manner, it provides CORBAsecurity Level 2 functionality, including authorization and identification; access control, including both individual and group privileges and support for multiple domains; delegation in a number of forms; auditing; administration and (going beyond the CORBA specification here) support for single sign-on – a welcome feature, especially in large enterprises. Even though the Orbix product entered general availability just last March, six large enterprises are already progressing toward rollout, with implementation architecture studies well underway.

TIVOLI SYSTEMS

IBM subsidiary Tivoli Systems, Inc., markets a CORBAsecurity implementation named Policy Director for CORBA. Also ORB-independent, it's available for both C++ and Java versions of Orbix and Visibroker (allowing them to interoperate securely), as well as for IBM's ORB. IBM's WebSphere, Enterprise Edition, incorporates security based on this implementation. With Release 3 in general availability since the first quarter of this year, Policy Director for CORBA has already been put into production at T. Rowe Price, a European telecommunications company and a large multinational bank.

The Workshop

Two days of tutorials (the first on CORBA, the next on CORBAsecurity) were followed by two days of papers and panels interspersed with lots of good discussion. Each of the final two days was divided into three topical sessions and a panel discussion. On the first day the three sessions covered Commercial Systems, Legacy System Integration and Government/Defense Systems, and the panel moderated a Security Users' Roundtable. On the second day sessions covered Standards, Policy Management and Performance Issues, and the panel moderated a Security Implementers' Roundtable.

OMG has posted electronic copies of the workshop slide presentations on the Web at www.omg.org/meetings/docsec/-presentations.html. While I'll describe many of the presentations here, space restricts me to just a few sentences for each, so if this piques your interest, head for this Web page. You'll miss the speakers' comments, however, which are important since most slides aren't the entire story for the well-presented material at this workshop. And you'll miss the audience interaction too; since this was very much a workshop and not a conference, much of the value of this event came from the audience give-and-take. If you find yourself wishing you knew more, watch for notices for next year's event and attend in person!

COMMERCIAL SYSTEMS SESSION

Viktor Ohnjec, of Genesis Development Corporation, opened the Commercial Systems session with a presentation on gathering security requirements. He explained why security-sensitive applications should be designed with security in mind from the get-go – not grafted on at the last minute. This helps prevent blunders that may compromise security or cost more than necessary (or both).

Dr. Stuart Katzke, chief scientist for the Information Assurance Solutions Group of the U.S. National Security Agency (NSA), spoke about the NIST/NSA National Information Assurance Partnership (NIAP) program for information assurance. The Common Criteria (CC), an ISO standard, defines security functionality requirements. Companies may spend a lot of time and money not only building systems that comply with the CC, but also testing these systems and having them certified compliant. Until now, certification by a laboratory in one country hasn't been recognized by other countries, forcing the repetition of duplicate procedures in country after country. To save companies time, money and resources better devoted to developing new security functionality, and to make more certified products available, a number of countries have banded together to accredit each others' testing laboratories and to mutually recognize their evaluation results. Current participants are the United States, Canada, France, Germany, the United Kingdom and, just recently, Australia and New Zealand. For more information, surf to http://niap.nist.gov/ccscheme/index.html.

INTEGRATION OF LEGACY SYSTEMS SESSION

Speakers in this session were Sam Lumpkin of 2AB, Inc., Heather Hinton of Tivoli and Ulrich Lang of Technosec. Sam started with security requirements, but didn't repeat any of the points from the first talk. He pointed out, for example, that at a large enterprise you might be required to trust every employee at one level, contractors at a lesser level and outsiders not at all. That might sound okay, but when you ask for lists of employees and contractors, you may discover that the lists don't exist because Human Resources records may be out of date, or because lists of contractors may be so incomplete that they're nearly useless. When this happens, which it does often, even the best security installations can't give the protection that management counts on.

Heather discussed the effects of components and composition on security, showing that security isn't compositional and trust isn't transitive. Ulrich pointed out the complexity of the telecomms environment – multiple companies each providing multiple services with multiple components. Not only can an individual service consist of multiple technical parts, but in order to be successful it must also interact with billing and other business services to provide income for the company. One essential part of this multicompany technical and business environment was nonrepudiation, described earlier.

GOVERNMENT/DEFENSE SESSION

GCHQ, the UK's analog to the USA's NSA, prototyped their security requirements using commercial CORBAsec implementations: ORBacus 3.1.3 from Object Oriented Concepts, Adiron's implementation of CORBAsecurity and Java 1.2.2. They discussed where their prototype succeeded, where it encountered difficulties yet still managed to operate and where it just couldn't manage. For details – and there were many – check the slides on OMG's Web site.

Unify Corporation

www.ewavecommerce.com

USERS' ROUNDTABLE

Six security users participated in the roundtable at the end of the first day: Viktor Ohnjec of Genesis Development Corporation, a consulting company that designs object-oriented systems for many large enterprises; Sam Lumpkin of 2AB, Inc., another consulting company well known for their CORBA work; Ulrich Lang of Technosec, a small consultancy concentrating on CORBA security, particularly for telecomms applications; Jia-Ling Lin from Eidea Labs, a company with an object-based infrastructure and framework product; Alberto Avritzer from AT&T, who brought in his company's experience with large-scale secure developments; and someone from GCHQ who added CORBA knowledge, security knowledge and experience to the panel. This large and diverse group of security users provided a detailed and entertaining session. Moderators were Dr. Richard Soley of OMG and Jishnu Mukerji of HP.

Viktor Ohnjec, in his opening remarks, said that his company was frequently called in after a system went online and the security pieces didn't work. Typically, they found that security was designed piece by piece, probably after everything else was frozen. Designs and architectures that took security into account from the beginning were much more likely to succeed. Alberto Avritzer sketched the scale of AT&T's software environment and their security requirements. They run an interoperability laboratory and require vendors to demonstrate interoperability and compliance to standards when they purchase software. A representative from GCHQ said they were moving to EJB and application servers, but intended to keep using C++ (both legacy code and newly written apps) and therefore needed a standards-based environment.

Panelists and members of the audience discussed security modularity in terms of the OMG specifications and what's available in the industry. For modularity the standards use GSS-API interfaces to the components that provide basic security functionality. Unfortunately, the most popular over-the-wire security mechanism is SSL, which doesn't conform to either GSS-API's interfaces or its functionality profiles. All agreed that making an SSL-based security system interoperate with a GSS-API system is a challenge!

Richard Soley challenged the panelists and audience with the following question: "Is security hard because the tools make it hard, or because the problem is intrinsically hard?" He proposed that security was just plain difficult. David Chizmadia pointed out that CORBAsecurity interfaces for users weren't complex and that most of the complexity was for the implementers. The user interface set is small, and for people who are willing to use defaults, it's easy to use. The complexity for the implementer comes in making the application programmer interface small.

STATE OF THE STANDARDS SESSION

Starting day two, Petra Hoepner of GMD Fokus described a telecomms middleware platform that takes a customer from online service subscription through provisioning and delivery of the service to online billing. It's based on OMG and TINA-C standards, and requires security throughout. Diagrams in the slide set show the various forms that security must take as you work through the stages of the interaction, so be sure to check the presentations Web site for details.

In a presentation that could have been a bit dry but instead kept everyone's interest, Polar Humenn involved the audience in working out the formal calculus of delegation, credential usage and other security functions. Polar, who is very active in security at OMG and is also chair of OMG's Security Revision Task Force, will use the results to guide the evolution of CORBAsecurity.

SECURITY POLICY MANAGEMENT SESSION

In the next session Polar presented a new model for access control in Level 2 CORBAsecurity, and Alex Kotopoulis of Segue Software discussed enforcement of business rules using a tool based on OMG's licensing CORBAservice.

PERFORMANCE ISSUES SESSION

In the final session before the Implementers' Roundtable, three investigators presented measurements of security's impact on performance. David Shambroom of GTE Laboratories, who began his project before implementations of CORBAsecurity were widely available, implemented his own CORBAsecurity using SSL and interceptors, and found the expected linear dependence on the number of operations and data size. The tables in his slide set showed the dependence on machine architecture and encryption algorithm. Patrick Dunne, a consultant working for GCHQ, presented measurements on the commercially available Adiron security implementation. Adding credentials to IIOP didn't slow it down, but moving to SLIIOP did. Finally, Tammy Blaser of NASA presented the most detailed set of measurements. NASA is building a jet engine simulation package for use by suppliers. It's written in C++ and distributed securely using CORBA. Data size dependence on security shows a peak at small payloads from initialization overhead – linear dependence beyond this peak reveals the encryption load. The slides on OMG's Web site show Tammy's data in detail.

IMPLEMENTERS' ROUNDTABLE

Not all of the Implementers' Roundtable participants came from companies that produce and market CORBAsecurity products. Ron Monzillo, lead on J2EE security, contributed insights on interworking between Java and CORBA security, especially important now with the close ties between the CORBA Component Model and Enterprise JavaBeans. Gregg Tally of Network Associates represented their firewall product, which supports IIOP, although it doesn't use OMG's firewall specification. Fred Dushin spoke for Hitachi America, which resells Concept 5's product on their platforms. David Chang of IBM represented their CORBA product as a total solution, which includes security – as Level 2 CORBAsecurity – where it needs to. Three vendors spoke about their own implementations: Polar Humenn of Adiron, Ryan Eberhardt of Gradient and Don Flinn of Concept 5.

The audience asked questions regarding access control, nonrepudiation, auditing, delegation and other capabilities besides the wire protocol and transport aspects that seemed to dominate the two days of presentations and discussion. Speakers from various companies, including Concept 5, Adiron and Sun, pointed out that they had rich security functionality and hadn't neglected these aspects.

Don Flinn pointed out one very important reason to include auditing capability: certain large computer users, especially hospitals, can't use strict access control because if it causes a delay, or if lack of a password prevents access, patients could die. Instead, they audit every action on the system. In response to a question on security in Microsoft's products, Polar pointed out that their security primarily used Kerberos and that interoperability with their certificates and authentication structure should be possible. As the roundtable closed, Richard Soley followed up a question about security interoperability by asking who would participate in a security interoperability showcase at next year's DOCSEC workshop. Everyone on the dais said they would. Have we mentioned that we expect next year's workshop to be really interesting?

siegel@omg.org

AUTHOR BIO

Jon Siegel, the Object Management Group's director of technology transfer, was an early practitioner of distributed computing and OO software development. Jon writes articles and presents tutorials and seminars about CORBA, and his new book, CORBA 3 Fundamentals and Programming, *has just been published by John Wiley and Sons.*

4th Pass

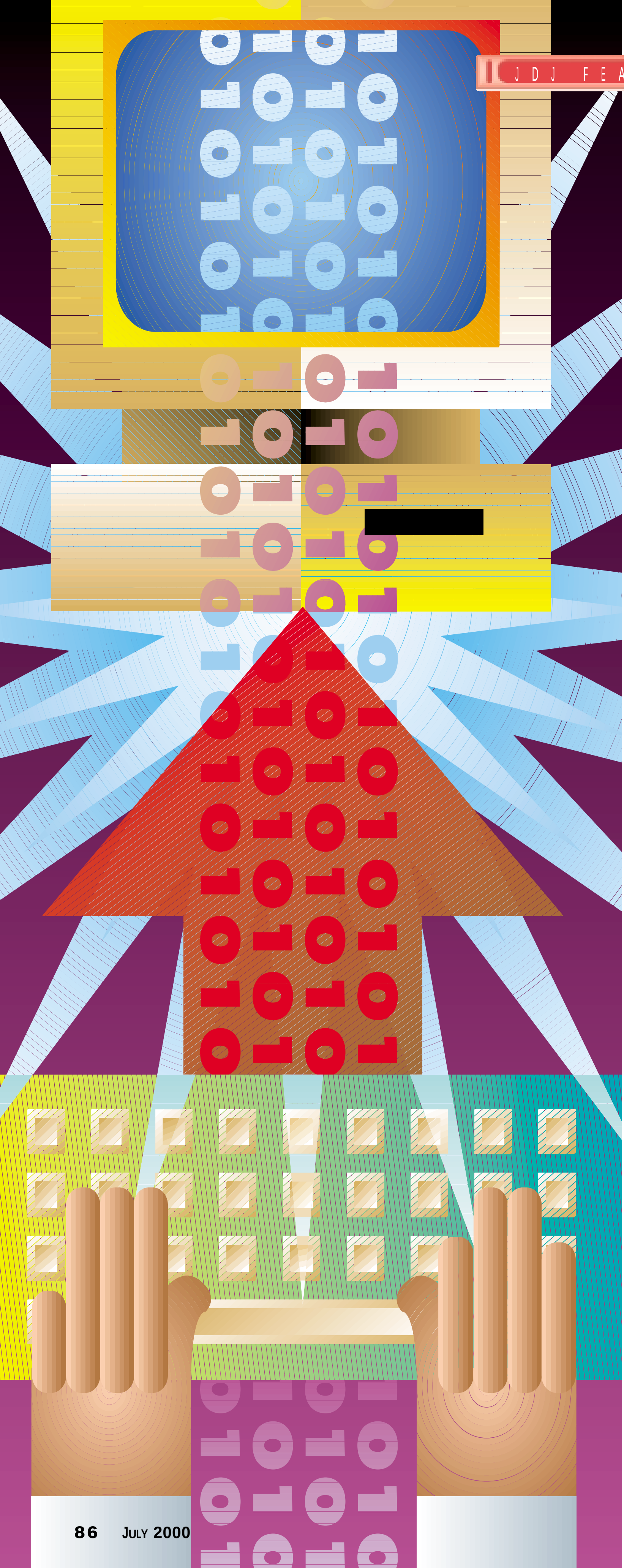
www.4thpass.com

JavaCo

www.javaco

on2000

on2000.com



JDJ FEATURE

BeanShell & DynamicJava: Java Scripting with Java

If you want the best of two worlds, here's how to get it

WRITTEN BY RICK HIGHTOWER

Part 5 of a series discussing the many languages that compile and/or run on the Java platform

The past three articles in this series have highlighted the strengths of scripting languages. They're interactive and dynamic, and allow you to experiment, debug and prototype solutions quickly. However, the most common response when I speak to die-hard Java fanatics is, "Yeah, but I'll have to learn another language and I already know Java" (I consider myself a die-hard Java fanatic to a degree).

To be honest, this is a barrier that most won't cross. But what if you could have the best of both worlds? What if you could have your cake and eat it too?

Well, in a sense you can. You can use a Java-like scripting language. Now when I say that, most of you were probably thinking JavaScript. (Admit it.) But I'm not talking about JavaScript.

Java and JavaScript have a similar syntax, both deriving from C/C++. However, the way that JavaScript defines methods, event handlers and classes is quite different from Java. In fact, other than the similar names and common C heritage, JavaScript and Java have little in common. (Consider the inheritance by prototype model by JavaScript versus the more common inheritance model found in Java. However, Rhino is a good marriage of Java and JavaScript.)

If you know Java, learning JavaScript syntax will take a while, but not too long. However, you don't have to learn another syntax to get the advantages of a dynamic scripting language. Several scripting languages are heavily based on the Java syntax.

Two of the most popular, BeanShell and DynamicJava, are interpreted versions of Java. Both have interactive interpreters that allow experimenting with Java APIs. Both closely follow the Java syntax. (DynamicJava is much closer.)

BeanShell Background

BeanShell is a Java source interpreter with object scripting language features. In addition to some scripting commands, BeanShell executes standard Java statements. As stated earlier, this gives BeanShell an advantage over scripting languages like JPython and NetRexx because Java programmers can learn it quickly.

One advantage of BeanShell over plain Java is you can use BeanShell interactively for Java experimentation and debugging. BeanShell, which is easy to embed, can be used as a scripting engine for your applications. Not surprisingly, the author of BeanShell compares the relationship of Java and BeanShell to the relationship of C and Tcl/TK.

The name *BeanShell* has always kind of thrown me for a loop. When I first looked into it (almost two years ago), the name made me think of C Shell or KornShell, command interpreters (shells) for the UNIX world. Somehow the name seems to be related to shell scripting, although BeanShell has little to do with KornShell and more to do with Python, Perl and Tcl. That said, BeanShell does have some UNIX-like shell commands as follows: cd(), cat(), dir(), pwd(), rm(), exec(), and so on.

Unlike DynamicJava, BeanShell supports scripted objects as simple method closures like those in JavaScript (see Part 1 of this series [Vol. 5, issue 2] for examples of JavaScript method closure). DynamicJava uses the Java syntax for adding methods to a class.

BeanShell closely mimics the Java language. For example, it uses the full Java statements and expressions syntax. It emulates strongly typed variables and methods. It mimics Java operators (arithmetic, logical and bitwise). BeanShell could aptly be renamed *little Java*. It closely follows the syntax of the language in many respects. It also adds features that are essential to a scripting language. For example, it allows dynamically typed (untyped) variables. It has the following scripting features: methods with optionally loose typing of arguments and return values, scripted objects with JavaScript-like method closures, and scripted AWT/Swing event handlers. Like JPython, BeanShell has convenience syntax for working with JavaBean properties. It also has many utilities for working with beans, e.g., loading a bean (load) and saving a bean (save(bean)). In addition, BeanShell has simplified ways for working with primitive wrapper types and hashtables.

One BeanShell claim to fame is that it has such a small footprint – less than 150K.

DynamicJava Background

DynamicJava supports all features and syntax defined by the Java language, including class and inner class definition and multithreading. And it adds features that make scripting easier:

- Statements can be written outside classes and methods in the top-level namespace.
- Typing is dynamic.
- There's more than one package.
- Methods are defined outside classes.
- Comments can begin with #.

Where BeanShell is a subset of Java with scripting features, DyamicJava is a superset of Java with scripting features. Dynamic Java can be found at www.inria.fr/koala/djava/. The focus on DynamicJava seems to be compatibility with Java. It boasts the full implementation of the Java language specification in interpreted mode.

Rosetta Stone Examples

For comparison, each BeanShell sample application will have a corresponding Java

Meet JDJ

EDITORS AND COLUMNISTS

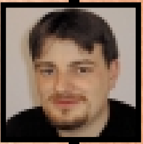
Attend the biggest Java developer event of the year and also get a chance to meet *JDJ*'s editors and columnists!

JavaCON 2000

Sean Rhody
Editor-in-Chief, JDJ

Sean is the editor-in-chief of ***Java Developer's Journal.*** He is also a principal consultant with Computer Sciences Corporation.

Alan Williamson
JDJ Straight Talking Columnist

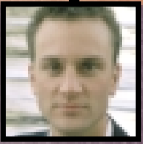


Alan is the "Straight Talking" columnist of ***JDJ,*** a well-known Java expert, author of two Java books and contributed to the Servlet API. Alan is the CEO of n-ary Consulting Ltd., with offices in Scotland, England and Australia.

Ajit Sagar
Editor-in-Chief, XML-Journal

Ajit is the founding editor of ***XML-Journal*** and a well respected expert in Internet technologies. He is a member of a leading e-commerce firm in Dallas, Texas, where he focuses on Web-based e-commerce applications and architectures.

Jason Westra
EJB Home Columnist



Jason is the "Enterprise JavaBeans" columnist of ***JDJ*** and CTO with Verge Technologies Group, Inc., a Java consulting firm specializing in Enterprise JavaBeans solutions.

MEETING
September 24-27, 2000
Santa Clara Convention Center
Santa Clara, CA

JAVA DEVELOPER'S JOURNAL

CAMELOT COMMUNICATIONS

implementation. This article covers the following sample applications.

- A simple class definition (and object creation)
- A simple GUI application
- A simple statistics application
- Embedding of script in an application (if applicable)

Before we go into the first example let's cover the basics of BeanShell. If you haven't installed it, please follow the directions in the next section.

Installing BeanShell

To install BeanShell go to www.beanshell.org/download.html. On this page are two options: BeanShell with the Utilities and Shell commands (the deluxe) and the core interpreter version (regular). When I go to a carwash I always order the deluxe version. I may not need an undercarriage wash, but why take chances? For BeanShell you should download the deluxe version because you can use the utilities to learn BeanShell. The deluxe version is bsh-1.0.jar.

Thus, to set up the classpath, you can do the following:

UNIX:
```
export CLASSPATH=$CLASSPATH: usr/rickh/BeanShell/bsh-1.0.jar
```

Windows:
```
set classpath=%classpath%;c:\BeanShell\bsh-1_0.jar
```

If you're using a version of Java that doesn't include Swing, you need to add the swingall JAR file to the classpath as well. Some of the BeanShell utilities use Swing. Once you set up the classpath, you can start BeanShell in a graphical interactive interpreter or a console interactive interpreter.

To start BeanShell in a GUI interactive interpreter, do this:

```
java bsh.Console
```

or run as text-only on the command line:

```
java bsh.Interpreter
```

The graphical interpreter isn't that graphical.

BeanShell doesn't have an install package. However, installing and getting it up and running was brain-dead easy (just the way I like it). The directions are easy, and if you know Java well it won't be a problem to set up the environment to run it.

BEANSHELL 101: A SIMPLE "CLASS"

BeanShell doesn't have classes like Java, JPython, NetRexx, C++ and so on. Instead, it allows object construction via nested methods similar to the way JavaScript and Perl define "classes." Although not classes per se, they have a similar effect, that is, they enable you to create objects or instances of the "class."

The example below defines an Employee "class" in BeanShell:

```
Employee (){
 String firstName, lastName;
…
 manager = null;

 __init(){
  super.firstName = "John";
…
 }
 return this;
}
```

For comparison, the equivalent Java Employee class would be defined as follows:

```
public class Employee{
 private String firstName, lastName;
…

 private Employee manager;

 public Employee(){
  firstName = "John";
…
 }

 …
}
```

Notice that the use of *super* in the BeanShell class isn't the same as it is in Java. In BeanShell it refers to the method that contains the current method. Thus super in the _init method refers to the Employee instance namespace – weird. Also notice that the manager variable in the BeanShell class is untyped, which means we can assign it later to any type. Compare the BeanShell Employee "class" in Listing 1 to roughly the equivalent Java class in Listing 2.

As you compare the two listings, notice that they both define toString methods. Also notice that, with the BeanShell "class," methods, arguments and return types don't have to be typed. Be sure to compare the getManager method in both the BeanShell version and the Java version.

To create an instance of an Employee and print it to the screen, you'd do the following:

```
print(Employee().toString());
```

The equivalent Java statement would be:

```
System.out.println(new Employee());
```

You can use Java objects in BeanShell, and if you use the Java Employee class in the BeanShell script, you can use the same syntax that you'd use in Java in BeanShell. Thus BeanShell allows you to do object composition with its Object scripting and to work with Java classes as you would in Java. The toString isn't recognized as a special method as it is in Java (for Object scripting).

Next we create two instances of employee called joe and ron and print those employees to the console. We print joe, who is ron's manager, by invoking the getManager method of ron – first in BeanShell, then in Java.

BeanShell:
```
joe = Employee();
joe.init("Joe", "Batista", 100, null, 1);
ron = Employee(); ron.init("Ron", "Furgeson", 101, joe, 1);
print(ron.toString());
print(ron.getManager().toString());
```

Java:
```
 Employee joe = new Employee("Joe", "Batista", 100, null, 1);
 Employee ron = new Employee("Ron", "Furgeson", 101, joe, 1);
 System.out.println(ron);
 System.out.println(ron.getManager());
```

As I said, the syntax is similar but not identical. Again, if you used a true Java Employee class, the code for Java and BeanShell would be identical. But I wanted to compare the way that BeanShell defines "classes" with the way that Java does.

A SIMPLE GUI

Now that we've created a simple class, we'll create a simple GUI. I admit that the class and the GUI are nonsensical – the idea is to demonstrate the interactive nature of BeanShell.

If you have BeanShell installed, let's pretend we're prototyping this GUI. Fire up the interactive interpreter by typing java bsh.Console as the system prompt. (You can opt to follow along in the DynamicJava interactive interpreter; it should work out just as well.)

Import the JFrame from the javax.swing package.

```
bsh % import javax.swing.JFrame;
```

(Note that the bsh % is the BeanShell prompt, i.e., you don't need to type bsh % each time.)

Create an instance of the frame, set its size to 200 by 200 and make it visible. First create an instance.

```
bsh % frame = new JFrame("My Prototype");
```

There's a convenient way to set the frame's visible property. (Note that this can be used to set properties dynamically from a configuration file.)

```
bsh % frame{"visible"}=true;
```

You can also use the Java way to set the visible property to true.

```
bsh % frame.setVisible(true);
```

Now set the size of the frame to 200 by 200.

```
bsh % frame.setSize(200,200);
```

Note: If you remember the JPython article (***JDJ***, Vol. 5, issue 3), the foregoing three steps were done in one line of code.

In BeanShell any bean property of a class can be set wth the bean{"property_name"}=value syntax. By bean property I mean a property as defined by a getter and a setter method, that is, the bean "design pattern" for properties.

At this point all we have is a stupid-looking gray box. Let's add some components to it – some labels, text fields and an okay button. As we develop this GUI application, I'll point out some of the features of BeanShell. (Since this is a demo, we're not going to meet any GUI style guidelines.) First we need to import a few classes from javax.swing.

```
bsh % import javax.swing.JButton;
bsh % import javax.swing.JLabel;
bsh % import javax.swing.JTextField;
bsh % import javax.swing.JPanel;
```

We could have used the * syntax; for example, we could have said import javax.swing *; similar to the way you'd do it in Java.

Next, create a pane (JPanel instance) and add it to the frame. Before we do this let's invoke the show command, which will show us what is returned from expressions.

```
bsh % pane = new JPanel();
bsh % frame.getContentPane().add(pane);
```

The foregoing prints:

```
<javax.swing.JPanel[,0,0,0x0,invalid
layout=java.awt.FlowLayout,alignmentX=null
alignmentY=null,border=,flags=34
maximumSize=,minimumSize=,preferredSize=
defaultLayout=java.awt.FlowLayout[hgap=5,
vgap=5,align=center]]>
```

Now we want to add components to this pane using the GridBag layout. If this isn't familiar to you, it'll be good practice. First import the GridBag layout classes.

```
bsh % import java.awt.GridBagLayout;
bsh % import java.awt.GridBagConstraints;
```

Next, change the layout of the pane to an instance of GridBagLayout.

```
bsh % pane.setLayout(new GridBagLayout());
```

Now add the first component to the pane using a GridBagConstraint – a JLabel. This will use all of the default values of the GridBagConstraints (see Figure 1).

```
bsh % GridBagConstraints constraint = new
GridBagConstraints();
bsh % pane.add(new JLabel("Name"), con-
straint);
bsh % frame.validate();
```

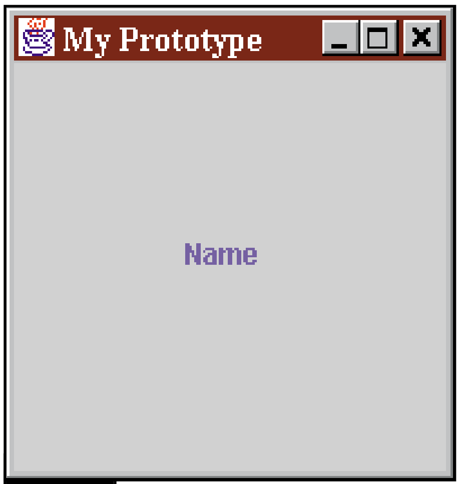


FIGURE 1 Simple form created in the interactive interpreter

Now add another JLabel, the label on the second row of the grid.

```
bsh % constraint.gridy=1;
bsh % pane.add(new JLabel("ID"), con-
straint);
bsh % frame.validate();
```

Add a text field on the first row in the second column. Then pack the frame (see Figure 2).

```
bsh % name = new JTextField(25);
bsh % constraint.gridy=0;
bsh % constraint.gridx=1;
bsh % constraint.weightx=80.00;
bsh % pane.add(name, constraint);
bsh % frame.pack();
```

Now add a second text field for the employee ID, this time to the right on the second row. Then pack the frame.

```
bsh % id = new JTextField(10);
bsh % constraint.gridy=1;
bsh % pane.add(id, constraint);
bsh % frame.pack();
```

As you can see, this isn't what we want. The text field components are centered and look silly (see Figure 3). I forgot to align the text field to the left in their cells (not really – I forgot on purpose).

Let's remove the components, then add them back with the proper alignment (see Figure 4). See how useful it is to be able to experiment with the layout in the interactive interpreter?

Remove the ID and name.

```
bsh % pane.remove(id);
bsh % pane.remove(name);
```

Now add back the ID and name with the proper alignment.

```
bsh % constraint.anchor=GridBagConstraints.WEST; //anchor West
bsh % pane.add(id, constraint);
//add the id
bsh % constraint.gridy=0;
//set gridy for name
bsh % pane.add(name, constraint);
//add name
bsh % frame.pack();
```

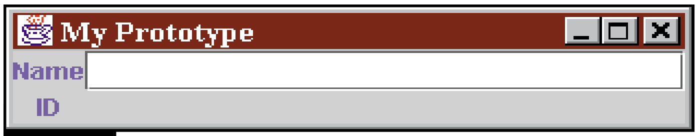


FIGURE 2 Added more components, packed the frame, arranged components

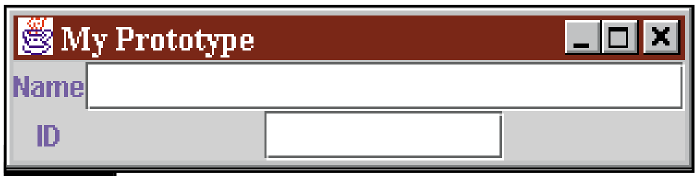


FIGURE 3 Added another component...oops – made a mistake... better fix it

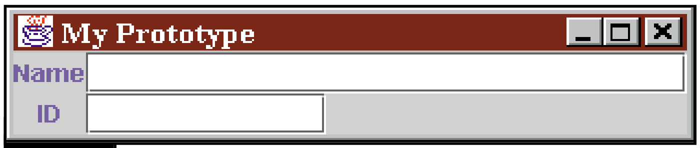


FIGURE 4 Fixed layout – looks good!

The above demonstrates the interactive, experimental environment in BeanShell. You can explore the cause and effect without the normal recompile, retest environment. Also, you can do this in a very Java-like syntax so when you're ready to write the real version there won't be much conversion.

Bean events are easily handled in BeanShell. Just as with bean properties, BeanShell adds features to make event handling easier. To handle an event simply define the appropriately named method from the java.awt.event listener interface and register the corresponding "this" type reference with the component.

Ajile Systems

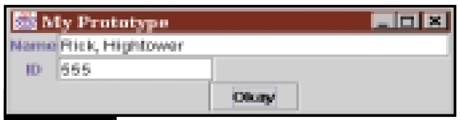
www.ajile.com

Computer Work.com

www.computerwork.com

Unlike Java, the "this" keyword means current namespace and the "super" keyword means the encasing namespace. To demonstrate this, let's set up an okay button. When the okay button gets clicked, this prototype application will print out the employee's name and ID.

First create and add a button to the GUI (see Figure 5).



Name: Rick, Hightower
ID: 555

FIGURE 5 Added okay button and event handler

```
bsh % okay = new JButton("Okay");
bsh % constraint.gridx=1;
bsh % constraint.gridy=2;
bsh %constraint.anchor=GridBagConstraints.-
CENTER;
bsh % pane.add(okay, constraint);
bsh % frame.pack();
```

Next, create a function. The function prints out the value of the name and ID text. Then register the current namespace with the okay button's actionListener.

```
bsh % actionPerformed( event ) {
          print ("Name " + name.get-
Text());
          print ("ID " + id.getText());
}
bsh % okay.addActionListener(this);
```

Enter some text in the Name and ID field and hit the okay button. This is a simple session, creating a simple GUI. For those of you who followed along with BeanShell, let me know what you think of it. I'm new to BeanShell, but I like it.

ROSETTA STONE GUI

The above was to show the interactive interpreter that comes with BeanShell. Now let's create a GUI based on the one we created above in both BeanShell and Java. In earlier articles we wrote the same GUI in NetRexx, JPython, Java and so on. You can compare this BeanShell to JPython and NetRexx by looking at past articles in this series (Vol. 5, issues 3 and 5, respectively).

Listing 3 shows the employee form that we prototyped in the interactive interpreter. Listing 4 shows the Java version of that prototype. If you compare the two, you'll notice they're very similar. Some notable differences are that the BeanShell version doesn't have a true class definition. Instead, it has some nested methods. Also, the BeanShell version of the "class" doesn't have a constructor or a main method. It took me about two minutes to convert the Java version to BeanShell. Since BeanShell is a subset of Java, the rest of the Rosetta Stone examples would be quite meaningless, that is, we already highlighted the major differences between BeanShell and Java.

DYNAMICJAVA AND THE ROSETTA STONE EXAMPLES

Since DynamicJava is a fully functional interpreted version of Java, Rosetta Stone examples written in Java work in DynamicJava with few if any changes. In DynamicJava you define your classes just as you do in Java. Just like BeanShell, DynamicJava has an interactive interpreter. In addition, you can use loosely typed variables. DynamicJava is quite cool in its own right.

ROSETTA STONE ADDING BEANSHELL IN JAVA

A good example of embedding BeanShell into a Java program is shown on the BeanShell Web site. The example is listed with some additional comments below:

```
import bsh.Interpreter;  //Import the Bean-
Shell Interpreter
    ...
Interpreter i = new Interpreter();   //Cre-
ate a new instance of the interpreter.
i.setVariable("foo", 5);
//Define a variable
i.eval("bar = foo*10");
//Evaluate an expression.
System.out.println("bar =
"+i.getVariable("bar") ); //Get the value
of a variable.
```

Scorecard

How does BeanShell score? Here's my opinion.

AUTHOR'S SCORECARD FOR BEANSHELL

Ease of use	10
Embeddability	10
Resemblance to parent language	7.5
Unique features	7.5
String parsing	0
Productivity	6
Working well with Java classes	9
Development environment/debugging	2

And here's my opinion of DynamicJava.

AUTHOR'S SCORECARD FOR DYNAMICJAVA

Ease of use	10
Embeddability	10
Resemblance to parent language	10
Unique features	5
String parsing	0
Productivity	5
Working well with Java classes	10
Development environment/debugging	2

Thus I give both scripting languages a score of 52 out of 80 based on the above criteria. Now remember, those criteria are based on a scripting-language philosophy of keeping things easy for the developer (see ***JDJ***, Vol. 5, issue 2, for more information on the power of scripting languages). Drop by the ***JDJ*** Forum to grade BeanShell and DynamicJava.

Emar-
cadero

www.embar-

cadero.com/-

develop

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

- ***Ease of use:*** BeanShell was easy to install and easy to start using.
- ***Embeddablity:*** BeanShell is easy to embed.
- ***Resemblance to parent language:*** BeanShell strives to be a subset of Java with features added for scripting. However, the object scripting in place of Java classes is difficult at best, and quite different from Java. BeanShell seems to favor JavaScript's way of handling classes (although by no means is it identical to JavaScript). On the wish list for BeanShell is the ability to define classes using the Java style. If BeanShell did that, it would easily get a 10 in this category. I like the way DynamicJava defines classes.
- ***Unique features:*** Neither BeanShell nor DynamicJava do well in this category. For example, the convenient way of handling bean properties saves one keystroke over the Java way of handling events.
- ***String parsing:*** DynamicJava and BeanShell don't have any special string parsing abilities. However, in a future release BeanShell plans on adding regular expression capabilities into the language à la Perl. If BeanShell did that it would easily get a 10 in this category.
- ***Productivity:*** BeanShell and DynamicJava can access the complete Java APIs, which is an extensive class library. However, unlike Python, which has built-in language support for collection objects including collection literals that let you define a collection, iterate easily over Java collection and work with Bean properties as you do in TCL when creating an instance of a class, BeanShell and DynamicJava don't have any of the really convenient language features. Python and NetRexx language constructs make programming strikingly productive.
- ***Working well with Java classes and APIs:*** You can instantiate Java classes, invoke Java methods and easily set up bean events in both, but you can't subclass Java classes and interfaces in BeanShell the way you can in DynamicJava, NetRexx and JPython.
- ***Development environment/debugging:*** It's good to have an interactive interpreter, and BeanShell and DynamicJava have a good one. However, if you're used to having GUI builders, debugging in an IDE, setting watches and so on, forget about it. The development environment for these languages isn't up to par.

Parting Shots

Scripting languages are good for prototyping and rapid application development (refer to first article in series). Most Java developers don't want to learn new language syntax to get the advantage of scripting languages. DynamicJava and BeanShell fill a gap.

I think BeanShell has the advantage over DynamicJava in that its creator wants to see it evolve into a better scripting language. For example, he wants to add support for regular expression. On the other hand, the creator of DynamicJava just wants to be syntax compatible with Java as much as possible. I think that BeanShell and DynamicJava have a long way to go to get to the levels of mature scripting languages. DynamicJava won't add more features to make it easier on the programmer, that is, more scripting-like (see interview with creator of DynamicJava).

I think if BeanShell added the features that make JPython work so well with Java it would be much better. BeanShell syntax is more in tune with what people are used to (more so than NetRexx and JPython); it's Java-like. The JPython feature set is more in tune with what a scripting language should be. Combine them and throw in a few cool NetRexx features for good measure and you'd have one lean, mean, Java-like scripting language. NetRexx and JPython are mature scripting languages – more mature than BeanShell – but BeanShell has quite a niche (being more like Java).

Components need scripting languages. ActiveX has Visual Basic. JavaBeans has BeanShell.

AUTHOR BIO

Rick Hightower currently works at Buzzeo Corporation (www.buzzeo.com), maker of ZEOlogic, an EJB application server rules engine and workflow. He is a principal software engineer working on an EJB container implementation and distributed event management. In addition, he is author of a book, Programming the Java APIs with JPython, *to be published by Addison Wesley.*

rick_m_hightower@hotmail.com

Syntion AG

www.syntion.com

Interviews...

Pat Niemeyer Creator of BeanShell

***R. Hightower:* What features do you plan for BeanShell?**

P. Niemeyer: Well, there's a brief "wishlist" on the Web site at www.beanshell.org/wishlist.html. Some features that will almost certainly be in the next release are more dynamic class loading and class reloading, taking advantage of the API that allows introspection of private and field methods, better error reporting and a more complete test suite.

I also have submissions from people right now that add regular expression support on a par with Perl. I've concocted a way to add this without deviating too much from standard Java syntax. That's always been a goal, to maintain as much real Java syntax as possible and simply extend/loosen it.

I think there's a need for a language that scales semantically between scripting and rigorous, static typing. I think that loose Java is "the natural scripting language" for Java. It requires almost no learning curve.

One feature that bsh got in the latest version takes advantage of a new reflection capability in JDK1.3 – dynamic proxy creation. This allows bsh to emulate any type of interface. You've always been able to script event listeners, etc., in bsh; however, the types were limited to the known types in AWT and Swing. Now, with 1.3, you can script any kind of interface.

In the future I'd like to extend the capability to be able to generate arbitrary types, e.g., extend existing types as subclasses or script abstract base classes. This may sound more arcane, but it's important for one reason. It would allow the bsh syntax to be extended to cover class definitions as well. Right now bsh only understands statements and expressions as well as its own extended method closures (as in Perl or JavaScript, methods that return "this" and act like objects). It would be nice if bsh could interpret 100% of the Java syntax including classes.

I'd also like to be able to have bsh spew out compilable code someday (not that bsh isn't reasonably fast. It's really just limited by the speed of the reflection API right now). But it would make a lot of people happy if they could flip a switch and turn their bsh script into a compilable Java file.

I also think bsh should provide better support for serving as a JavaBean inside various IDEs, to wire together sophisticated connections.

***RH:* I see that you've made BeanShell opensource. How many developers are working to extend BeanShell features?**

PN: There are a couple of hundred people on the developer's list. I'm currently in the process of moving development over from the beanshell.org ISP to sourceforge.net (an opensource facility with lots of neat features like public CVS, etc.).

I think I'll do a maintenance release for bug fixes and then start working on a 2.0 with the new features.

By the way, there'll be a brief section on BeanShell in the third edition of my book *Exploring Java* (O'Reilly & Associates). The book is to be retitled *Learning Java*. I mention this because it will probably prompt me to get back to work on the next release...we've been stuck at 1.0 for a long time.

***RH:* Do you know of any commercial products that use BeanShell as their scripting language?**

PN: I started maintaining a list but it's incomplete. First, everyone should know that BeanShell is distributed with Emacs as part of Paul Kinnucan's JDE to serve as a Java interpreter. I believe he also uses it internally. Various other IDEs are providing plug-ins for it: Sun's NetBeans has been interested. They did a poll recently and JPython and BeanShell were the top choices. I believe the ElixirIDE (http://elixir-tech.com/ElixirIDE/) has a plug-in now. And someone just wrote one for JBuilder (www.multimania.com/bjb/index.en.html). CERN, the physics people, use it in a sophisticated visualization app. Two years ago Corel (the WordPerfect people) contributed a lot of bug fixes and help but then their Java efforts sort of dried up. I am aware of some Jigsaw integration and the JOS (www.jos.org) people are considering it for their shell language. I am probably forgetting some important ones. Tens of thousands of people have downloaded the code and I have perhaps a thousand license forms (it's just a request – the code is LGPL, of course). I hope to put a better list together when I update the Web site.

***RH:* What special features set BeanShell apart from other scripting languages?**

PN: Other interesting things to mention are that BeanShell can be easily embedded in Java or, vice versa, bsh scripts can call any Java app or API seamlessly. By *seamlessly* I mean that live Java objects can be passed into and returned from scripts. Bsh also has a "server mode" which allows you to run bsh inside a running app and then open a console from your Web browser (or telnet) into the application and poke around like a debugger. (This feature is shaky in some releases.)

Stephan Hillion Creator of DynamicJava

***RH:* What features do you plan for DynamicJava?**

Stephan Hillion: There is no additional feature to add to DynamicJava. The objective is to have a scripting language as close as possible to the Java Language Specification. In the current version almost all the Java features are supported: there's no need to extend the core interpreter for the moment – perhaps when things like genericity or assertions [are] included in the Java language. I opened a Pandora's box by adding scripting features to the language (C-style functions, stand-alone statements), but I don't want other extensions to be added. The actual scripting features are minimal and allow both easy scripting and easy transition between scripts and real Java programs. And nobody asks for other features.

***RH:* How would you compare DynamicJava to BeanShell?**

SH: The main difference with BeanShell is the object model: BeanShell has a JavaScript-like object model while DynamicJava recognizes and interprets all the Java constructs including classes and inner classes definitions. The consequences are that you can go farther in prototyping (with DynamicJava).

***RH:* Any guess as to the number of people using DynamicJava?**

SH: Since the last release in March, about 500 downloads. It's the only indication I have.

***RH:* What special features set DynamicJava apart from other scripting languages that run in the JVM?**

SH: DynamicJava is Java: no wrappers to Java objects – DynamicJava objects are Java objects (usable by Java objects), no new language to learn for Java developers. You can take a script, and with very few modifications it becomes a true Java class.

Emar-
cadero

www.embar-
cadero.com/-
develop

Listing 1: BeanShell Employee “class”

```
Employee (){
 String firstName, lastName;
 int id, dept;
 manager = null;

 __init(){
  print("This far");
  super.firstName = "John";
  super.lastName = "Doe";
  super.id = 1;
  super.manager=null;
  super.dept=1;
 }

 init(String fname, String lname, int id, manager, int dept){
  super.firstName =        fname;
  super.lastName  =        lname;
  super.id        =        id;
  super.manager   =        manager;
  super.dept      =        dept;
 }

 getManager(){
  return this.manager;
 }

 String toString(){
  StringBuffer buf = new StringBuffer();
  buf.append(super.lastName+',');
  buf.append(super.firstName+',');
  buf.append(""+super.id);
  return buf.toString();
 }

 __init();
 return this;
}

print(Employee().toString());
joe = Employee(); joe.init("Joe", "Batista", 100, null, 1);
ron = Employee(); ron.init("Ron", "Furgeson", 101, joe, 1);
print(ron.toString());
print(ron.getManager().toString());
```

Listing 2: Java Employee class

```
public class Employee{
 private String firstName, lastName;
 private int id, dept;
 private Employee manager;

 public Employee(){
  firstName = "John";
  lastName = "Doe";
  id = 1;
  manager=null;
  dept=1;
 }

 public Employee(String fname, String lname, int id, Employee
 manager, int dept){
  firstName      =        fname;
  lastName       =        lname;
  this.id                 =        id;
  this.manager   =        manager;
  this.dept      =        dept;
 }

 public Employee getManager(){
  return manager;
 }

 public String toString(){
  StringBuffer buf = new StringBuffer();
  buf.append(lastName+',');
  buf.append(firstName+',');
  buf.append(""+id);
  return buf.toString();
 }
 …
 …
}
```

Listing 3: BeanShell EmployeeForm

```
import javax.swing.*;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import employee.Employee;

EmployeeForm(){
  JFrame frame;
 JTextField name;
 JTextField id;

 void init(){
  super.frame = new JFrame("Employee Form");
  pane = new JPanel();
  super.frame.getContentPane().add(pane);

  pane.setLayout(new GridBagLayout());

   // Create a name, and id text field.
  super.name = new JTextField(25);
  super.id = new JTextField(10);

   // Create and add a "Name" and "ID" label.
  JLabel nameLabel = new JLabel("Name");
  nameLabel.setLabelFor(name);
  nameLabel.setDisplayedMnemonic('N');
  GridBagConstraints constraint = new GridBagConstraints();
  pane.add(nameLabel, constraint);

  JLabel idLabel = new JLabel("ID");
  idLabel.setLabelFor(id);
  idLabel.setDisplayedMnemonic('I');
  constraint.gridy=1;
  pane.add(idLabel, constraint);

   // Add the name and ID text field to the form.
  constraint.gridy=0; constraint.gridx=1;
  constraint.weightx=80.00;
  constraint.anchor=GridBagConstraints.WEST;
  pane.add(name, constraint);
  constraint.gridy=1;
  pane.add(id, constraint);

   // Create an okay button, add it, and set up its event handler.
  JButton okay = new JButton("Okay");
  okay.setMnemonic('O');
  constraint.gridx=1; constraint.gridy=2;
  constraint.anchor=GridBagConstraints.EAST;
  pane.add(okay, constraint);
  okay.addActionListener(this);

  frame.setVisible(true);
  frame.pack();

 }

  void actionPerformed(ActionEvent event){
  handleOkay();
 }

 void handleOkay(){

  String name, fname, lname;
  int index=0;
  int id =0;

  name = super.name.getText();
  index = name.indexOf(" ");
  fname = name.substring(0, index);
  lname = name.substring(index+1, name.length());

  id = Integer.parseInt(super.id.getText());

  Employee employee = new Employee(fname, lname, id, null, 100);
  System.out.println(""+employee);
 }

 init();
 return this;
}

ef = EmployeeForm();
```

Listing 4: Java EmployeeForm

```
import javax.swing.*;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import employee.Employee;

public class EmployeeForm extends JFrame{
 private JTextField name;
 private JTextField id;

 public EmployeeForm(){
  super("Employee Form");
  JPanel pane = new JPanel();
  getContentPane().add(pane);

  pane.setLayout(new GridBagLayout());

   // Create a name, and id text field.
  name = new JTextField(25);
  id = new JTextField(10);

   // Create and add a "Name" and "ID" label.
  JLabel nameLabel = new JLabel("Name");
  nameLabel.setLabelFor(name);
  nameLabel.setDisplayedMnemonic('N');
  GridBagConstraints constraint = new GridBagConstraints();
  pane.add(nameLabel, constraint);

  JLabel idLabel = new JLabel("ID");
  idLabel.setLabelFor(id);
  idLabel.setDisplayedMnemonic('I');
  constraint.gridy=1;
  pane.add(idLabel, constraint);

   // Add the name and ID text field to the form.
  constraint.gridy=0; constraint.gridx=1;
  constraint.weightx=80.00;
  constraint.anchor=GridBagConstraints.WEST;
  pane.add(name, constraint);
  constraint.gridy=1;
  pane.add(id, constraint);

   // Create an okay button, add it, and set up its event han-
dler.
  JButton okay = new JButton("Okay");
  okay.setMnemonic('O');
  constraint.gridx=1; constraint.gridy=2;
  constraint.anchor=GridBagConstraints.EAST;
  pane.add(okay, constraint);
  okay.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent event){
    handleOkay();
   }
  });

  this.setVisible(true);
  this.pack();
 }

 public void handleOkay(){

  String name, fname, lname;
  int index=0;
  int id =0;

  name = this.name.getText();
  index = name.indexOf(" ");
  fname = name.substring(0, index);
  lname = name.substring(index+1, name.length());

  id = Integer.parseInt(this.id.getText());

  Employee employee = new Employee(fname, lname, id, null, 100);
  System.out.println(""+employee);
 }

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

Download the Code!

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

Visualize Inc.

www.visualizeinc.com

E-Commerce Market for EJB Solutions

Thoughts on EJB in the present e-commerce space... but first, an anniversary review of all EJB Home columns

WRITTEN BY
JASON WESTRA

I'd like this month to offer some editorial thoughts on the e-commerce market for EJB solutions – but first let me just say "Happy Birthday" to the EJB Home column and briefly recap the articles that have appeared here over the past year....

Part 1: "Happy Birthday, EJB Home!"

Has anybody else been counting? This month marks the twelfth **EJB Home** column in ***Java Developer's Journal.*** Over the past year, **EJB Home** has striven to educate both corporate developers and business leaders on Enterprise JavaBeans topics. Following suit with the rest of *JDJ*, **EJB Home** has been dedicated to keeping you abreast of technical EJB content, reviewing new EJB products and providing awareness of political and market news on the EJB front.

If you're a newcomer to **EJB Home**, you may be interested in which topics have been previously covered in this column. If you're a regular who has mislaid your copy of the issue containing a particularly interesting **EJB Home** column, the summaries below will refresh your memory and point you in the direction of which issue you'll want to revisit digitally at www.sys-con.com/java/archives.

EJB Home Recap

Vol. 4, Issue 6: *"Enterprise JavaBeans: Where does EJB fit into the Java Platform for the Enterprise?"*

The inaugural issue of **EJB Home,** in the JavaOne99 Special Edition of ***JDJ***, provides an architectural overview of the center-stage role played by EJB in the Java Platform for the Enterprise, now called J2EE. It details the importance of APIs such as JDBC, JTA/JTS and JNDI to the component execution environment that makes EJB a successful architecture for building distributed, scalable, server-side applications in Java.

Vol. 4, Issue 8: *"Enterprise JavaBeans Persistence"*

Provides a comparison of entity bean persistence models: CMP (container-managed persistence) and BMP (bean-managed persistence). The August article discusses the ramifications of choosing a particular persistence model and includes examples of how to develop each type of entity bean.

Vol. 4, Issue 9: *"XML DTD for EJB Deployment Descriptors"*

With the release of EJB 1.1 specification, deployment descriptors for EJBs became XML-based. The September column covers the basics behind the specification's new guidelines and the impact it has on the portability of your beans across EJB servers, and provides an example of a deployment descriptor converted from a proprietary format to the XML format described in section 16.6 of the EJB 1.1 specification.

Vol. 4, Issue 10: *"The Business Advantage of EJB: Developing Portable EJB Applications" (Part 1)*

Part 1 in a two-part series, the October **EJB Home** presents various types of portability including language, component model, resource and platform portability. The article gives an in-depth overview of the portability goals the EJB specification 1.0 attempts to address and which types of portability are out of scope.

Vol. 4, Issue 11: *"The Business Advantage of EJB: Developing Portable EJB Applications" (Part 2)*

Is EJB portability a myth or reality? This November article presents a comprehensive list of EJB 1.0 portability traps to avoid and how to ensure an easy port to another EJB server. While J2EE is an architectural framework for portable, server-side Java applications, not all EJB vendors have embraced the J2EE standard. Without full compliance among EJB vendors, these guidelines of portability still apply.

Vol. 4, Issue 12 : *"The Oracle at Boulder"*

This December column ends 1999 with predictions regarding the future of EJB in the year 2000. Mergers and buyouts in the EJB vendor market are discussed as well as the server-side component war between

MetaMata Inc.

www.metamata.com

Microsoft's COM/DCOM model and Sun's Enterprise JavaBeans component model. Last, this article discusses success factors for EJB projects in Y2K and why some EJBs may be in danger of failing.

Vol. 5, Issue 1: *"Securing Your Company Data with EJBs"*

EJB's role-based security model is a widely misunderstood yet powerful tool in the EJB arsenal. This January 2000 article lays the groundwork for a high-level understanding of the security model and tops the discussion off with a classic bank account example to demonstrate the interactions of security as it pertains to EJB context-passing and container-managed authorization to access EJB resources (methods).

Vol. 5, Issue 2: *"E-Business with EJBs"*

This February column presents a value proposition of EJB in an e-business scenario. Specifically, the column details how Java Servlets and EJB features such as handles and stateful session beans can play a role in providing stateful e-business solutions. It also presents numerous technical topics not covered in previous **EJB Home** articles. The concepts are exercised with full source code in a small prototype. Developers can use this sample application as a project, expanding it with more scalable techniques for a real-life application.

Vol. 5, Issue 3: *"VisualCafé Enterprise Edition for WebLogic"*

IDEs supporting integrated development, debugging and deployment of EJBs are few and far between. VisualCafé, a leader in Java development tools, has integrated with BEA's WebLogic Server to provide a best-of-breed approach toward EJB development. This March article presents key factors to consider when looking for your corporate standard EJB IDE. It describes what areas VisualCafé excels in and which factors it needs to improve to become a top-notch EJB IDE.

Vol. 5, Issue 4:*"What Do MTS and EJB Have in Common?"*

This April article sheds light on the stateless component architectures offered by Microsoft in the form of MTS and Sun's Enterprise JavaBeans. Stateless session beans are the main topic of this column, including a discussion about how clustering of stateless components offers easy load-balancing and failover for your middle tier.

Vol. 5, Issue 5: *"Developing Coarse-Grained Business Components"*

This May article clarifies some less-than-obvious details in the EJB 1.1 specification regarding developing coarse-grained entity beans. It reviews commonly encountered technical problems such as how to cache dependent objects and ensure they're refreshed correctly, how to map a BMP (bean-managed Persistence) entity bean to multiple tables with JDBC and how to declaratively change the behavior of your component through environment entries.

Vol. 5, Issue 6 *(JavaOne2000 Special Edition): "PowerTier 6's PowerPage: Instantly Web-Enabling Your EJBs!"*

This product review covers a new feature of the PowerTier 6 for EJB server, PowerPage. PowerPage offers JavaServer Page code generation to offer servlet/JSP developers a head start or a working model on which to base their code.

I hope that **EJB Home** has been as enlightening for you to read as it has been delightful for me to write. I thank those of you who have e-mailed me your feedback on the column or presented ideas for topics you're dying to read about. Many of your ideas, questions and concerns have been catalysts for monthly topics. I always try to be punctual in my responses to your individual questions and observations and I warmly encourage you to provide even more feedback in the coming months.

Now, with the birthday celebration over, let's switch gears and discuss new markets for Enterprise JavaBeans.

Part 2: New Markets for EJB

I've seen the market demand for EJB rise greatly over the past year – but not in those areas many thought it would proliferate most. However, as markets are won, so are they lost. Thus I want to raise awareness vis-à-vis the rise and death of the ERP market and maybe the rise and death of EJ…well, let's not go there this month.

THE DEATH OF ERP

In days bygone (1998, to be exact), when EJB was in its infancy, I recall numerous folk – clients and colleagues alike – researching, plotting and planning on using EJB as a replacement for their monolithic, Web-disabled ERP (enterprise resource planning) applications. It is *Enterprise* JavaBeans, isn't it? And Java has become synonymous with *Web-enabled*, right?

The ERP market has plummeted, but not because these systems were replaced with EJB applications. ERP companies have been wavering for the past year and a half because the industry, generally speaking, is done with replacing legacy HR, billing and manufacturing systems. Likewise, ERP companies, war-torn from Y2K battles, weren't able to muster the armies necessary to ensure their products were Web-enabled and attacking the vast new market of e-commerce.

Don't blame (or applaud) EJB for the death of ERP; rather, applaud EJB for allowing quick-moving, Java-savvy companies to present e-business applications over the Web. Applaud early EJB product vendors like The Theory Center, Evergreen Internet, Inc. (www.evergreen.com), and Chematch.com (www.chematch.com) for taking advantage of EJB to deliver exciting product offerings. I'm not talking about an HR system to track employee information here; I'm talking about innovative products like components that jump-start the development of e-commerce sites and auction sites that provide real-time, global trading online.

Although I'm not The GartnerGroup, my industry contacts in the EJB world have led me to the conclusion that there are countless more e-commerce applications than mission-critical enterprise applications built or being built on the EJB platform today. EJB has successfully redefined itself from an enterprise application platform (à la 1998) to an e-platform for building scalable, Web-enabled solutions.

THE RISE OF EJB IN NEW MARKETS

The transformation of EJB from savior of costly, time-consuming, monolithic ERP applications to the preferred platform for developing e-business solutions hinges on five factors: hype, price, time-to-market, standards and scalability/ reliability.

1. ***Hype:*** I learned my ABCs from an English textbook. It never equated the letters "EJB" with "VC" – i.e., venture capital(ists). However, these acronyms are becoming more and more synonymous with one another as EJB has become more than just a platform aimed at the enterprise. I think the phenomenon of start-ups gaining venture capital based on hot technologies like XML, Jini and EJB has increased the adoption of EJB over the past year. What VC is going to fund an effort to develop a manufacturing system that competes with SAP in PowerBuilder? Market hype is in the e-commerce space and EJB is an enabling technology for building the e-commerce sites VC's love.

2. ***Price:*** ERP applications are hideously expensive. Well, production quality EJB

Sic Corporation

www.sic21.com

servers aren't cheap either, but at least respectable, open-source options exist with jBoss (www.ejboss.org) and enhydra (www.enhydra.org). In any event, newly formed start-ups with innovative e-commerce business plans are probably not going to look toward ERP companies, which are trying to reinvent themselves as e-business vendors. These companies' products are historically overpriced and have a reputation for over-budget implementations.

Instead, speaking from experience, start-ups are partnering with companies like Verge that know how to implement e-commerce applications in EJB under budget and on time. Which leads to my next point quite well…

3. ***Time-to-market:*** The e-commerce wave moves fast. Start-ups with an idea need to strike quickly for fear of losing the claim to fame of being the "first-to-market" and the market share that comes with being first. Since traditional ERP vendors have been notorious for long, drawn-out implementations, start-ups are afraid to implement solutions that need to be up and running yesterday. How can an ERP product firm compete with EJB, especially when it has third-party component suites specifically tailored around e-commerce and touting the ability to have your site up in 30–60 days?

4. ***Standards:*** EJB is becoming widely accepted as a standard for server-side development. Standards are attractive to firms for many reasons, including portability and plug-in-play capabilities. Portability eases the minds of dot-coms betting the farm on a technology that might otherwise be obsolete in six months. The ability for standards-based components to plug-and-play with other components enhances time-to-market. For instance, third-party components can be bought prebuilt with 50–70% of an application's logic already done.

5. ***Scalability/reliability:*** Scalability is a necessity in the e-commerce market, in which Internet use is volatile and a start-up's site can't afford downtime. Online markets are competitive. Users can easily click to another site offering a similar service or product without the faintest hint of loyalty. When combined with the rest of the J2EE platform, such as JavaServer Pages (JSPs) and servlets, EJB is a scalable solution for Web access.

Since it's specifically built on top of a transactional model, EJB also provides a reliability factor that permits CEOs of e-commerce companies to sleep better at night. The Java Transaction Service and Java Transaction API are integral parts of the Enterprise JavaBeans specification. If a dot-com doesn't want to lose track of an online customer's order, transaction support is a necessity. Once again, EJB is an excellent fit for e-commerce.

WILL THE BLEEDING EVER STOP?

Enterprise JavaBeans has made headway in the dot-com arena with positive results…and VCs love it. The VC industry poured a record amount of money – some $22 billion – into start-ups the first quarter of 2000. At this writing there's been no determination of whether or not the market crash in April will stop the hemorrhage of capital into e-commerce start-ups.

If the well dries up, how will it affect the demand for Enterprise JavaBeans e-commerce solutions? What will be the next big market, and will EJB be flexible enough to secure a favorable portion of it? I intend to provide my answers to this in an upcoming article. Please hold tight until then, and let's continue this EJB ride for another 12 issues!

jwestra@vergecorp.com

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

InetSoft Technology Group

www.inetsoftcorp.com

Youcentric

www.youcentric.com/nobrainer

Oracle Internet Application Server 8*i*— A Sneak Preview

A rich programming environment for developing and deploying Web applications

WRITTEN BY
MATTHIEU DEVIN

Over the past few years, maturity of the Internet, as well as the introduction of a host of other technologies, has led to the development of highly complex Web solutions that test the functionality and scalability of even the most versatile application servers.

Oracle Corporation was among the first enterprise software vendors to offer an application server for developing Web-based solutions. The servers of today, however, must support the caching of data for better availability and scalability. They also have to support integration with disparate applications at both the logic and the data layers, and support mobile and intermittently connected devices. And they must have up-to-date support for the Java development model.

With its new Internet application server (see Figure 1) Oracle responds to the increasingly complex nature of the new breed of solutions by delivering everything the original Oracle application server (OAS) product offered, *plus*. The pluses include essential features from Java 2 APIs, a wide range of integration technologies such as JMS and XML, and efficient database caching for improved scalability. This article describes the benefits of the new server for Java programmers and how they can fully use the Oracle Internet Platform for e-business solutions.

Oracle Internet Platform

This platform is based on two server products with a common Java runtime environment: the Oracle Internet Application Server for middle-tier solutions and the Oracle8*i* database for data-tier requirements.

When developing Web applications backed by an Oracle8*i* database, you can use a variety of languages, application frameworks and tools. These range from 3GL offerings such as Java, C and PL/SQL to highly declarative frameworks such as Oracle Business Components for Java, Oracle Forms and Oracle Portal-to-Go. You'll typically use these languages and frameworks with the Oracle Internet Application Server to deploy multitier Web applications served by one or several Oracle8*i* instances holding your data. For applications with well-defined scalability requirements, such as a predictable user base or a development environment, or simply when a two-tier solution fits your application architecture, you can deploy the same components, without the new server, on a single system running only the Apache Web server and Oracle8*i*. Many developers find these configurations adequate for deployment of intranet Web applications. Because both the Internet Application Server and Oracle8*i* use a common infrastructure, application components can be moved seamlessly between the tiers without any changes to your code. This flexibility allows you to test various configurations for optimum performance. It also gives you the scalability needed to accommodate rapid growth as user demand escalates.

Web Applications

The Oracle Internet Platform focuses on the development of Web applications – applications that are accessed primarily through HTTP – including applications for live Web users accessed from Web browsers or mobile devices, as well as applications accessed by other applications across the Internet, as is the case for many business-to-business applications. The packaging of an XML development kit with the Oracle Internet Application Server and the advanced capabilities of the Oracle AQ asynchronous communication mechanisms using JMS APIs are necessary building blocks for such applications.

Oracle HTTP Web Server, Powered by Apache

The Oracle Internet Application Server has many components. One of them, the Apache Web server, is the HTTP entry point into Oracle Internet Application Server applications. Apache benefits from the contributions of the best programmers of the Internet and is a proven, scalable and robust Web server. The other components are integrated with Apache either through Apache "mods" supplied by Oracle or through servlets or cgi-bin gateways.

Some of the Apache extensions offered by Oracle – the mods – are standard Apache features; others have been developed specifically for the Oracle Internet Application Server. Oracle supports all these extensions and ports them to every operating system on which Oracle8*i* is available. For Java programmers the server leverages mod_jserv, which enables you to run servlets and JSPs. It can use any Java Virtual Machine (JVM) running on your system. On Solaris, Oracle bundles Sun's JDK. Oracle has also developed a stand-alone JSP compiler (Oracle JSP) that you can download from http://otn.oracle.com. Oracle JSP 1.1 supports tag libraries without requiring a servlet 2.2 environment, enabling you to use tag libraries on an Apache Web server

KL Group Inc.

www.klgroup.com

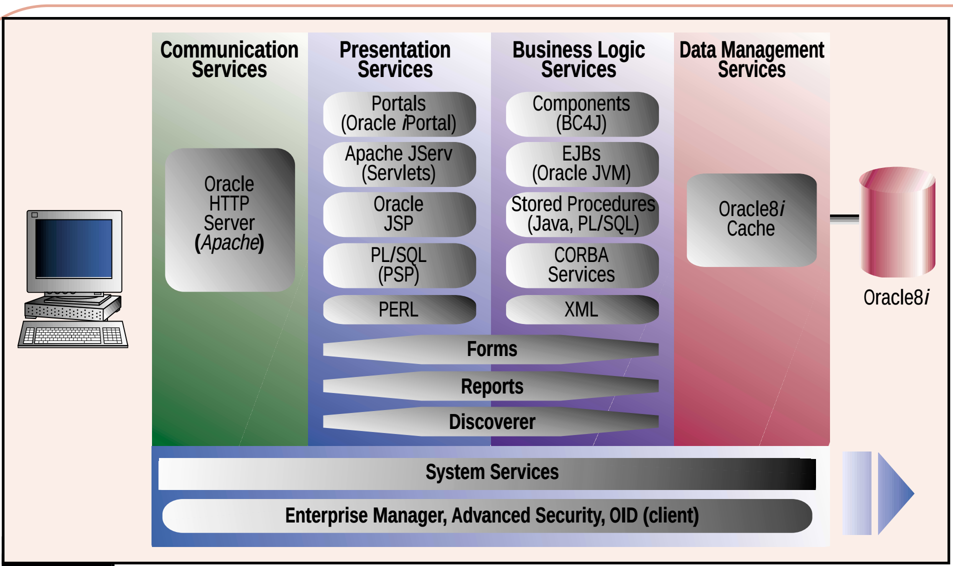


FIGURE 1 Major components of the Oracle Internet Application Server 8*i*

with mod_jserv. The Oracle Internet Application Server comes with mod_plsql, which enables you to delegate the handling of HTTP requests to PL/SQL and Java stored procedures running in an Oracle database. Running Web applications in the database has proved very scalable for many OAS-powered Web sites. This ability, which was carried over from OAS to the new server, provides a good migration path for OAS customers.

Starting with Release 3, Oracle8*i* comes with an embedded servlet engine and the same Apache bundle as the Internet Application Server. The engine enables you to run servlets 2.2 and JSPs with the Oracle8*i* JVM (formerly known as Oracle JServer) directly in the database address space. A new Apache mod, mod_8i, delegates the handling of URLs to the database, thereby providing a more scalable, robust and secure environment for running servlets than does mod_jserv. In addition, SQL-intensive servlets running in the database benefit from a shorter-path JDBC driver and can run faster than in a stand-alone JVM.

Oracle8*i* Cache

Many real-world Web applications have been designed with homegrown, middle-tier caches to improve scalability. Developers will typically write shell or Perl programs that extract data from a central database and populate file system caches on the Web server systems. The Oracle Internet Application Server provides Oracle8*i* Cache to install SQL caches in the middle tier.

This full-fledged SQL engine brings read-only SQL data closer to the middle-tier logic. The cache management tool enables you to point to a database and create caches on multiple nodes, handpicking tables that should be instantiated in the middle tier. By running the SQL caches on the same tier as the application logic, you avoid network round-trips when accessing cached data. This is likely to make the application run faster, but its primary advantage is that it reduces the load on the database, which improves scalability. In effect, you end up with more CPUs to handle the SQL load. The cache is read-only and can be refreshed on demand or on a timer basis.

Release 2 will also let you run the Oracle8*i* JVM and PL/SQL in the cache. By running servlets, JSPs, EJBs or even CORBA objects in the cache, you obtain a highly scalable, reliable and available execution environment for your J2EE applications. The server runs the same J2EE container in the middle tier as the database does in the back end. This gives you the choice of deploying your J2EE components in either the middle tier or the back-end database without having to modify them.

Other Components

In addition to support for Java and XML, the Oracle Internet Application Server comes with several other modules. Among them is a set of application frameworks for portal construction and long-running forms-based applications, business intelligence and ad hoc reporting. Most of the frameworks are extensible in PL/SQL. Some of them are now being extended to support Java programming, so expect more Java-based application frameworks from Oracle in the future.

For the Java Programmer

The Oracle Internet Application Server is a full-fledged Web application environment that enables you to develop using the Oracle Internet Platform and deploy your application on multiple tiers. It caters well to the needs of the Java programmer; all J2EE APIs are supported, and you can run various J2EE-based components. You also get a collection of Java libraries and frameworks to help you write applications.

An application server requires a set of development tools. For the Java programmer Oracle supplies the JDeveloper programming environment that enables you to develop and deploy J2EE-based components including JSPs, servlets and EJBs. One of the newest features of JDeveloper is support for attaching a debugger to Java code running in the Oracle8*i* JVM in the cache or data tiers.

J2EE and Other Java Libraries

The Internet Application Server and Oracle8*i* support all the J2EE APIs. The details depend on the versions, but the key is that the J2EE implementations are identical in both servers, which allows you to move your code freely between the middle (cache) tiers and the Oracle8*i* data tier. Release 1 of the Oracle Internet Application Server supports servlets and JSPs in the Apache mod_jserv tier. In the next release of both products (Oracle8*i* Release 3 and Oracle Internet Application Server Release 2) you can also run servlets and JSPs in the middle and data tiers. In all versions EJBs always run in the middle and data tiers. The other J2EE APIs, such as JDBC, JTA, JMS and JavaMail, are available in both tiers.

In addition to J2EE, the Internet Application Server supplies several Java libraries, or frameworks, such as the XML development toolkit and the Portal-to-Go

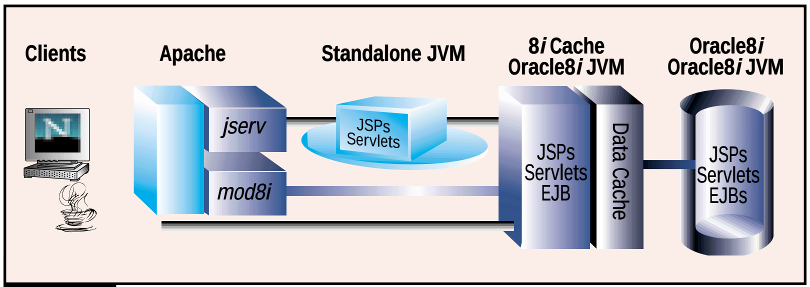


FIGURE 2 Oracle Internet Application Server Java tiers

Starbase

www.starbase.com

framework, to support WAP devices. Of particular interest is the Oracle Business Components for Java, an object-relational mapping tool and application framework targeted to the SQL-savvy Java programmer. BC4J enables you to define Java objects representing the results of various SQL queries and can help share data between Java objects coming from the same rows, even if they're fetched by unrelated SQL queries. This model, for Java programmers with a SQL background, complements the EJB entity bean implementation of the Internet Application Server. BC4J allows the expert programmer to optimize SQL queries and data access patterns.

Flexible Java Deployment

Two different JVMs are offered with the Internet Application Server to run your Java components: a stand-alone JVM such as Sun's JDK and the Oracle8*i* JVM embedded in the cache and data tiers (see Figure 2). The stand-alone JVM is intended for use with Apache mod_jserv. The Oracle8*i* JVM is preferred with the Internet Application Server. You can run servlets, JSPs and EJB clients in the stand-alone JVM. The Oracle8*i* JVM runs directly in the address space of the Oracle8*i* caches and database. You can use it to run EJBs, CORBA objects and Java stored procedures. In Release 2 of the Oracle Internet Application Server and in Oracle8*i* Release 3, you can also use the Oracle8*i* JVM to run servlets and JSPs directly in the cache and data tiers.

EJBs always run in the Oracle8*i* JVM, either in the caches or in the database. In either case the EJBs will run close to the persistent SQL storage, which is highly desirable because EJBs are indeed data-centric components. Many EJB developers are using EJBs as Java-friendly front ends to SQL data. All the JDBC or SQLJ code stays hidden behind the EJB remote interfaces, and servlet programmers calling EJBs don't have to know anything about the SQL schema of the application. This is a good way to design applications if you want to be able to change the database schema without having to reimplement your servlets. Running EJBs in a cache or in the data tier is even more interesting for entity beans, which are even more data-centric than session beans.

Depending on your application, EJBs may benefit from running in a cache tier. Obviously, EJBs that don't update SQL data will benefit greatly from the cache, which is read-only. By replicating the tables used by the EJBs in the middle tier, you'll immediately improve the scalability of your application. EJBs that update data may also benefit from the cache, depending on their data access pattern. Updatable entity beans, especially the ones that rely on the container to manage their persistence (also known as CMP entity beans), will require careful design at both the Java and SQL levels to be able to benefit from the cache at all. In some cases you may get better performance by running EJBs that update data in the data tier, and calling them from read-only EJBs running in the cache tiers. One advantage of having the same EJB container in the cache and data tiers is that you can easily try out different deployment scenarios until you find the best architecture for your application.

Servlets can run either in stand-alone JVMs or in the Oracle8*i* JVM. Stand-alones are controlled by Apache mod_jserv, and this setup works well for stateless applications. Many Web applications running today have been designed to be stateless to improve scalability, failover and availability. In these applications most of the conversational state is stored in a database across calls. If a JVM running servlets crashes, you can quickly bring up another JVM to take over, and the conversational state can be restored from the database. To end users it just looks like an HTTP failure – once the page is reloaded, work easily resumes. One problem with the stand-alone JVMs is their tendency to hang or crash under heavy memory usage. They can easily get lost in endless garbage collection runs and stop responding. To work around this problem, Apache mod_jserv allows you to distribute the HTTP load across a pool of JVMs. This divides the memory usage across the different JVMs and provides live backups in case one of the JVMs crashes. It also helps to improve the throughput of your application because more threads can run servlets at the same time. Managing the JVM pool isn't easy, but management tools will be available to help you in the future.

The Oracle8*i* JVM, running directly inside the cache or database, is more robust than stand-alone JVMs. It was designed from the start to be a server VM, and supports heavy memory usage without the problems of stand-alone JVMs. It lets each client see a dedicated JVM when there is, in fact, a pool of server processes taking turns at executing Java on behalf of the different clients. In addition, the Oracle8*i* JVM provides faster access to SQL than the stand-alone JVMs. It uses the shorter-path JDBC KPRB driver that accesses the SQL engine without any interprocess communication. Because the SQL engine and the Oracle8*i* JVM are linked in a single executable, the JDBC driver only needs a C function call to execute queries. Finally, Oracle8*i* Release 3 comes with the JServer accelerator compiler that speeds up Java code running in the Oracle8*i* JVM to JIT speed, making the Oracle8*i* JVM a good contender in the high-performance Java world.

Consider running servlets in the Oracle8*i* JVM if they use a lot of memory (stateful servlets are a good example) or if they need fast access to SQL and EJBs. Servlets running in the Oracle8*i* JVM have direct in-memory access to EJBs. The EJBs are activated in the same JVM as the servlets, removing the network overhead incurred by the RMI/IIOP calls that are necessary when the servlets run in stand-alone JVMs. Running servlets in the Oracle8*i* JVM produces a more stable and robust system. Stateless servlets or servlets using a small amount of per-client memory will work fine in the stand-alone JVMs. However, be aware that because of security and integrity reasons the Oracle8*i* JVM doesn't allow you to use JNI and doesn't implement some of the AWT interfaces; therefore you must run servlets that depend on these features in the stand-alone JVMs.

A New Generation of Application Servers

Combining Apache with a middle-tier SQL cache executing the state-of-the-art Oracle8*i* JVM and the proven PL/SQL engine provides an extremely powerful application server. Adding a family of application frameworks for portals, forms, business intelligence, mobile device access and asynchronous communications makes the Oracle Internet Application Server one of the richest programming environments available for developing and deploying Web applications. The tight integration of the server with the Oracle8*i* database, together with the shared J2EE container, broadens the deployment options from a single system for lower scalability needs to multiple forms of systems for utmost Internet scalability. Providing Java programming and remote debugging tools, the server can be used in conjunction with the Oracle8*i* database to fulfill the promise of the Oracle Internet Platform to furnish an end-to-end solution for the development of Web applications.

• • •

For more information on the Oracle Internet application server, visit www.oracle.com/ip/deploy/ias/. Developers should go to http://otn.oracle.com/products/ias/.

For more information on the scalability of the Oracle JServer VM (Oracle8*i* JVM), see the November 1999 issue of ***Java Developer's Journal*** (Vol. 4, issue 11).

mdevin@us.oracle.com

AUTHOR BIO
Matthieu Devin is a director of development in Oracle's Server Technologies division and the chief architect for the Oracle Internet Application Server product.

Protoview

www.protoview.com

JDJ Store

www.jdjstore.com

Pervasive and Red Hat Promote Tango 2000 and Pervasive SQL 2000 to Linux community

(*Austin, TX*) – Pervasive Software Inc. is partnering with Red Hat to spark deployment of sophisticated e-business applications for the Linux operating system.

As a member of Red Hat's ISV Partner Program, Pervasive has provided 30-day trial versions of Tango 2000 Application Server (small business edition) and Pervasive. SQL 2000 Server (20-user version), which are now shipping with an application CD included with Red Hat Linux 6.2.

www.pervasive.com

MKS and Pretzel Logic Software Partner to Deliver Integrated Management Solution

(*Chicago, IL*) – Mortice Kern Systems Inc., a leader in e-business management solutions, has announced a strategic alliance with Pretzel Logic Software, Inc., a leading provider of dynamic application management infrastructure solutions.

The two companies will partner to deliver vital integration with such technologies as Vignette StoryServer and Media Surface's MediaSurface, extending the code to content change management capabilities of the MKS Integrity Framework to the Web personalization and relationship management marketplace.

www.mks.com
www.pretzel.com

Microware Announces PersonalJava 3.0.1

(*Des Moines, IA*) – Microware Systems Corporation, a leading provider of embedded software solutions, has released the next generation of its PersonalJava Solutions for OS-9 for cellular and smart handheld devices, digital TV and Internet appliances.

New enhancements include faster execution, smaller footprint, multiwindows support and Java Profiler Interface.

www.microware.com

Sun and Metrowerks Announce Strategic Relationship

(*Miami, FL*) – Sun Microsystems and Metrowerks have announced a relationship to provide a comprehensive tool set that integrates Java Card development components with Metrowerks' professional software development environment. Metrowerks will incorporate Sun's Java Card components into its CodeWarrior IDE.

It is planned that future versions of the Java Card technology-based tool set from Metrowerks will offer support for major smart card manufacturers' products.

www.metrowerks.com
www.sun.com

Advanced Software Technologies Unveils GDPro 4.1

(*Littleton, CO*) – Advanced Software Technologies, Inc., introduces an upgrade to the company's flagship UML visual modeling tool. GDPro 4.1 provides a new Web System Browser Report, with functionality that allows developers to easily produce and share comprehensive diagrams of their software designs over the Internet.

Other new features include Java Reverse Engineering performance enhancements, context-sensitive pop-up windows to speed diagram population and a Rational Rose diagram import feature.

www.advancedsw.com

Jinfonet Software Launches New Version of JReport

(*Washington, DC*) – Jinfonet Software, Inc. announced the general availability of JReport Professional and JReport Enterprise Server Version 2.2. The product is available for immediate download rom the Jinfonet Web site.

www.jinfonet.com.

Imperial Sugar Chooses SilverStream

(*Burlington, MA*) – Imperial Sugar Company, a leading supplier of bulk sugar and sweetener in the U.S, has selected the SilverStream Application Server as the power behind its recently deployed e-business portal applications.

Imperial Sugar developed and deployed its company-wide online community with the SilverStream Application Server to enable their 1,425 employees and management to stay up to date on corporate information and human resources information.

www.imperialsugar.com
www.silverstream.com

Gemstone Teams With Mercury Interactive

(*Beaverton, OR*) – GemStone Systems, Inc., a software infrastructure technology leader for the new B2B economy, and Mercury Interactive Corp., one of the leaders in Web performance management solutions, have announced a strategic alliance that provides a reliable, automated load-testing solution for e-businesses seeking to optimize time-to-market goals. As a result of the partnership, GemStone customers will be able to take advantage of Mercury Interactive's LoadRunner to load-test and benchmark enterprise-class applications built on the GemStone/J application server using J2EE technology.

www.gemstone.com

TowerJ Now Supports Java 2

(*Austin, TX*) – Tower Technology Corporation is now offering TowerJ 3.5, an upgrade that supports Java 2. This new release is available on Sun Solaris, HP/UX, Linux, Tru64 UNIX and Windows NT platforms with others forthcoming. TowerJ 3.5 is the first high-performance Java 2 JVM for Intel-based Windows NT and Linux.

www.towerj.com

IBM Brings Wireless and Realtime Technology to E-Markets

(*Boston, MA*) – IBM has announced the first software that enables businesses to create online marketplaces that interact with handheld devices such as mobile phones, PDAs and pagers.

The WebSphere Commerce Suite (Marketplace Edition) software offers all the functions and tools for building a successful and scalable marketplace, including various dynamic trading models such as exchange, RFP/RFQ and auctions. In addition, the solution provides powerful aggregated catalog, secure membership registration and access control management, as well as easy-to-use business intelligence and reporting functions. Combined with Lotus Sametime, buyers and sellers can communicate and receive online customer assistance to easily exchange goods and services.

www.software.ibm.com

RSW Introduces First Testing Solution for EJBs

(*Waltham, MA*) – RSW Software Inc., the e-business application testing technology leader and a business unit of Teradyne, Inc., has introduced EJB-test, the industry's first solution to test the scalability and functionality of EJB middle-tier applications.

Optimized for BEA WebLogic and IBM WebSphere application servers, EJB-test provides an easy-to-use and highly accurate means for testing EJBs early in the development life-cycle – providing companies with a significant saving in application time-to-market and an increase in the overall quality of resulting e-business applications. With this announcement, RSW Software continues to expand its full lifecycle e-business testing solutions.

A fully functional version of EJB-test is available for evaluation at the RSW Software Web site.

www.rswsoftware.com.

XML DevCon 2000

www.xmldevcon2000.com

JDJ NEWS

Unify Launches eWave ServletExec 3.0

(*San Jose, CA*) – Unify Corporation has announced the immediate availability of Unify eWave ServletExec 3.0, which enables server-side Java functionality and seamless integration with backend systems, allowing companies to provide dynamic, real-time, secure access to corporate data by customers, employees, partners and suppliers.

Unify eWave ServletExec 3.0 incorporates full support for key components of the J2EE standards, including Java Servlet API 2.2 and JSP 1.1 technologies. A development version of Unify eWave ServletExec 3.0 is now available for free download from the Unify eWave Web site. www.UnifyeWave.com

Inprise/Borland and Corel Terminate Proposed Mergers

(*Scotts Valley, CA*) – Inprise/Borland Corporation has announced that its merger agreement with Corel Corporation has been terminated by mutual agreement of the two companies.

Dale Fuller, Inprise/Borland interim president and CEO said, "Much has changed since the merger was agreed to more than three months ago, and our board concluded that it would be best to cancel the merger on an amicable basis."

Commenting on Inprise/Borland's future, Fuller said: "Inprise/Borland is well positioned today with improving operating results and substantial liquid assets. Future operations will continue to follow our increasingly successful strategy of creating solutions that enable companies to move their businesses to the Internet." www.borland.com

Arbortext Launches Epic 4.0

(*Ann Arbor, MI*) – Arbortext, Inc., announces the launch of Epic 4.0, featuring Epic E-Content Engine (E3), a server-side system that enables e-businesses to attract and retain more customers by providing more personalized, dynamic and easily searchable content for improved presale and post-sale interactions. In addition, Epic 4.0 adds compatibility with Oracle 8*i*FS to the extensive list of repositories that Arbortext supports. www.arbortext.com

XML DevCon 2000 Delegate Registration, Exhibit Floor Sold Out – Fall, Winter Programs Announced

(*Montvale, NJ*) – XML DevCon 2000 delegate registrations closed on June 4 with over 3,500 registered attendees, making this the largest gathering of XML users ever assembled.

With record-breaking attendance and a sold-out exhibit floor, XML DevCon 2000 has established itself as the premier event for XML professionals. Announcements regarding upcoming XML initiatives were delivered by sponsors throughout the event.

SYS-CON Media and Camelot Communications also announced four new XML DevCon events. For more information on upcoming **SYS-CON Media** and Camelot Communications events, see www.XMLDevCon2000.com or www.sys-con.com.

JDJ June 2000 Issue Breaks Another New Record

(*Montvale, NJ*) – ***JDJ***'s special June JavaOne Focus issue became the largest issue in **SYS-CON**'s history. ***JDJ*** last month reached over 300,000 Java professionals with more than 200 pages worth of content.

As a media cosponsor of JavaOne 2000, ***Java Developer's Journal*** and **SYS-CON Radio** brought live coverage, news, interviews and commentary from San Francisco. Full JavaOne coverage can be found at www.sys-con.com.

ADVERTISER INDEX

ADVERTISER	URL	PH	PG
4TH PASS	WWW.4THPASS.COM	877.484.7277	83
ALLAIRE CORPORATION	WWW.ALLAIRE.COM/DOWNLOAD	888.939.2545	49
ALLAIRE DEVELOPER CONFERENCE	WWW.ALLAIRE.COM/CONFERENCE		4
AJILE SYSTEMS	WWW.AJILE.COM		90
AMAZON.COM	WWW.AMAZON.COM/JAVA		37
APPEAL VIRTUAL MACHINES	WWW.JROCKIT.COM	46.8.402.28.73	57
BUZZEO	WWW.BUZZEO.COM	800.804.4724	29
CAREER OPPORTUNITY ADVERTISERS		800.846.7591	126-141
CEREBELLUM SOFTWARE	WWW.CEREBELLUMSOFT.COM	888.862.9898	51
CLOUDSCAPE	WWW.CLOUDSCAPE.COM	510.239.1900	72-73
CODEMARKET	WWW.CODEMARKET.NET	914.638.2159	41
COMPUTERWORK.COM	WWW.COMPUTERWORK.COM	800.691.8413	90
COMPUWARE	WWW.COMPUWARE.COM/NUMEGA	800.4.NUMEGA	27
CRUEL WORLD	WWW.CRUELWORLD.COM	650.847.3581	67
ELIXIR TECHNOLOGY	WWW.ELIXIRTECH.COM/ELIXIRCASE.COM	65 532.4300	69
EMBARCADERO	WWW.EMBARCADERO.COM/DEVELOP	415.834.3131	89,91,93
ENSEMBLE SYSTEMS	WWW.ENSEMBLE-SYSTEMS.COM		64
EVERGREEN	WWW.EVERGREEN.COM/JDJ.HTML	408.926.4500	79
FIORANO	WWW.FIORANO.COM	408.354.3210	71
FLASHLINE.COM, INC.	WWW.FLASHLINE.COM	800.259.1961	55
GEMSTONE	WWW.GEMSTONE.COM/WELCOME	503.533.3000	25
HOTDISPATCH.COM	WWW.HOTDISPATCH.COM	650.234.9752	31
IAM CONSULTING	WWW.IAMX.COM	212.580.2700	75
IBM	WWW.IBM.COM/SOFTWARE/SOUL/DEVCODE		15
INETSOFT TECHNOLOGY CORP	WWW.INETSOFTCORP.COM	732.235.0137	35,100
INTUITIVE SYSTEMS, INC	WWW.OPTIMIZEIT.COM	408.245.8540	63
JAVACON2000	WWW.JAVACON2000.COM		84-85
JAVACON2000	WWW.JAVACON2000.COM		87
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	107
KL GROUP INC.	WWW.KLGROUP.COM/FASTER	888.361.3264	13
KL GROUP INC.	WWW.KLGROUP.COM/COURSE	888.361.3264	103
KL GROUP INC.	WWW.KLGROUP.COM/GREAT	888.361.3264	144
MACROMEDIA	WWW.MACROMEDIA.COM/ULTRADEV		16-17
METAMATA, INC.	WWW.METAMATA.COM	510.796.0915	97
MODIS SOLUTIONS	WWW.IDEA.COM	703.821.8809	65
NEW ATLANTA	WWW.SERVLETEXEC.COM		59
NO MAGIC	WWW.MAGICDRAW.COM	303.914.8074	5
POINTBASE	WWW.POINTBASE.COM/EDJ	877.238.8798	33
PROGRESS SOFTWARE	WWW.SONICMQ.COM/AD4.HTM	800.989.3773	2
PROTOVIEW	WWW.PROTOVIEW.COM	800.231.8588	3,107
QUEST SOFTWARE	WWW.QUEST.COM	949.754.8000	23
QUICKSTREAM SOFTWARE	WWW.QUICKSTREAM.COM	888.769.9898	46
SEGUE SOFTWARE	WWW.SEGUE.COM	800.287.1329	11
SIC CORPORATION	WWW.SIC21.COM	822.227.398801	99
SLANGSOFT	WWW.SLANGSOFT.COM	972.2.648.2424	39
SOFTWARE AG	WWW.SOFTWAREAG.COM/BOLERO	925.472.4900	61
SOFTWIRED	WWW.SOFTWIRED-INC.COM/IBUS		77
STARBASE	WWW.STARBASE.COM	888.STAR700	105
SYBASE INC.	WWW.SYBASE.COM/PRODUCTS/EASERVER	800.8.SYBASE	19
SYNTION AG	WWW.SYNTION.COM		92
SYS-CON MEDIA, INC.	WWW.SYS-CON.COM	800.513.7111	14,95
THINWEB TECHNOLOGIES	WWW.THINWEB.COM	877.THINWEB	53
TIDESTONE TECHNOLOGIES	WWW.TIDESTONE.COM	800.884.8665	43
TOGETHERSOFT CORPORATION	WWW.TOGETHERSOFT.COM	919.833.5550	6
UNIFY CORPORATION	WWW.EWAVECOMMERCE.COM	800.GO.UNIFY	81
VISICOMP, INC.	WWW.VISICOMP.COM/JDJ7	831.335.1820	47
VISUALIZE INC.	WWW.VISUALIZEINC.COM	602.861.0999	95
WEBVISION	WWW.WEBVISION.COM	800.531.5057	45
XML DEVCON 2000	WWW.XMLDEVCON2000.COM		109
YOUCENTRIC	WWW.YOUCENTRIC.COM/NOBRAINER	888.462.6703	101
ZUCOTTO	WWW.ZUCOTTO.COM	613.789.0090	21

15 pgs

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

Career Opportunities

GOSLING INTERVIEW

—continued from page 26

‘You guys [*JDJ*] already do a lot of what I think is the really important thing, namely, going through **tutorials,** if you want to call them that. **Some of the things** that I’ve seen in ***JDJ*** are in the space between **tutorials** and **case studies...’**

size of a postage stamp and a credit card, there was an Internet browser and it ran Java code and all the other usual stuff besides being really teeny. But it was done in this opalescent mother-of-pearl thing, it was clearly designed to be like a piece of women’s jewelry, it was really strange.

JDJ: ***What type of things would you like to see in* Java Developer’s Journal*? We definitely support the education of our readers and, with your vision, what type of things would you like to see us include in the journal?***

JG: You guys already do a lot of what I think is the really important thing, namely, going through tutorials, if you want to call them that. Some of the things that I’ve seen in *JDJ* are in the space between tutorials and case studies, you know, like, “This is how you do this, or this is how I did that,” and when I look at it that’s often the most valuable thing, for me.

JDJ: ***Some of the topics we cover are Enterprise JavaBeans, servlets as well as GUI widgets. We have dedicated columns, etc. Are there any other facets of Java, since it’s such a broad topic, that you’d like to see covered on a monthly basis. I understand that you’re working, for instance, on the realtime spec – is that something that’s important enough for us to get it in there? Is there enough support in the industry right now for us to start educating readers on it?***

JG: Well, I’ve certainly been involved in the realtime stuff. The realtime community tends to be relatively small and pretty fanatical. I don’t know what percentage of your readers are realtime types...[but] I would be tickled if there were actually realtime articles in there. Actually, if there were enough realtime Java programmers to justify a regular column...that would actually make me really nervous. I mean think about it: it’s like enough people writing flight control systems for airplanes and factory floors and whatever to actually [get the attention of] a large mainline magazine for programming that stuff...that would be a little odd. One of the things I noticed that was a little odd was that we did this thing on Monday night, a sort of extended Q&A session, and at one point one of the people on stage asked the people in the audience, “Stand up if you’re writing server software,” then, “Stand up if you’re writing embedded cell phone software” and a few people rose; then “Stand up if you’re writing software for smart cards,” and like one person stood up. It was interesting that there was this sort of inverse relationship between volume and the number of developers working there. I mean the volume of smart cards just dwarfs everything else, and the number of people actually developing smart card stuff is pretty small. Embedded stuff is a lot bigger, but it’s not nearly the kind of diversity of stuff that’s going on in the infrastructure that all these pieces are talking to.

JDJ: ***Could you give a real layman’s explanation of what the realtime specification for Java is going to be utilized for? For example, you can write A hardware driver with it, I believe. Is that correct?***

JG: Yes, you could write hardware drivers with it – actually, people have written hardware drivers with Java straight before, you know. It’s not too hard to get there. What’s really at the core of any system that calls itself realtime is that you’re dealing with deterministic timing, so there are sets of classes for dealing with timing of events that recur. There’s all sorts of weird scheduling stuff, and these things sort of interrelate. In a typical computer system schedulers are oriented toward maximizing throughput. Maximizing throughput and maximizing determinism are often at cross-purposes, and so one of the things in the scheduling of realtime systems is this notion of feasibility analysis, where you’ve got a set of threads and you’re asking the question, Can you actually meet the demands of all of these threads; and the threads get to say, “I need this,” and one of the answers from the feasibility analysis is “No” – at which point, you know, it’s like you press the On button on the airplane, and the system goes through a feasibility analysis and it says, “Did he install a big enough CPU? No? [Then] I’m not going to start the engine.” Whereas in most systems you just try as hard as you can. Another piece of the determinism has to do with the garbage collection: there’s a mechanism in there for running threads concurrently with the garbage collector, and that turns out to be just plain ugly. One of the things that there was a certain amount of hope for early on was that the state of the art in realtime garbage collection would be good enough that we could just say, you know, “Use a realtime garbage collector.” But after studying bunches of PhD theses on realtime garbage collection, they were close but no cigar. I mean a lot of them had problems, like, “Well, it’s realtime mostly but every now and again we just have to stop.” If you have to stop once every three or four hours, your airplane isn’t going to make it from San Francisco to New York! The other common problem is that they would say, “We’re realtime, we can do [GCs] in under 3–4 milliseconds,” and for some people that actually works – [but] one of the problems in the realtime world is that it means different things to different people. So for some people realtime means getting it done within the second; for others, realtime means well under a millisecond. The demo that was done this morning, [by] the guys from Agile – the specialized Java chip – their definition of realtime responsiveness, which really shows up in their thread-to-thread context switch time, is like 500 nanoseconds. They can do a thread switch faster than some systems can add!

JDJ: ***Can you tell us what you think about the JavaOne 2000 conference, and what you look forward to seeing at JavaOne next year?***

JG: The whole thing about JavaOne for me is just the amount of really cool people. People ask me, “Don’t you get upset with people coming up and asking you really stupid questions?” and the answer is No – actually, almost nobody asks me stupid questions, they usually ask me really good questions, questions that have me going, “I don’t know the answer to that” – and I find that really cool. And just the sort of connections that get made, you know, I don’t know how many of these funny contacts that have goine on are going to turn into anything – these contests are not “Throw a nickel into the cup and get a dollar.” They are “Write a Java application in six hours and start a new company.” I don’t know if the fellow who won the Motorola contest is going to do with his baseball application, but that’s pretty amazing, I don’t think anywhere else would you see people attending who could participate in contests where that kind of stuff went on. It’s just wonderful.

JDJ: ***Have you been to any of the many parties going on after JavaOne is complete each day here?***

JG: Well, my day at JavaOne starts at about 6 a.m. – so I’ve been to a couple of things, but typically by about 8 p.m. all I can do is just...I’m just flat and I end up doing things...unfortunately for me this is a job, and I end up having to go and do things. I haven’t had a chance to go to as many parties as I would like to, but they’ve generally been pretty good.

Silverstream

www.silverstream.com

KL Group Inc.

www.klgroup.com